U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Filament Winding Simulation

Rodrigo de Lacerda Aroso Furtado

Supervisors:

Prof. Dr. António Torres Marques
Dr. Hugo Faria

A Thesis submitted for the degree of
Master of Science in Mechanical Engineering
to the Faculty of Engineering, University of Porto

Porto, March 2016

Acknowledgements

I take this opportunity to express my gratitude to Prof. Dr. António Torres Marques and Dr. Hugo Faria for the opportunity to build this project. It is thanks to their help and support that I was able to learn and evolve in a field which was at first unfathomable to me. On this note, I would also like to thank INEGI for hosting me and giving me the chance to learn in a safe and rewarding environment.

To José Cerqueira and Rui Marques, for all their help and availability during this work, for taking the time to teach me the tricky aspects of filament winding (both in theory and in practice) and for providing me the knowledge and tools needed to complete this work.

I must acknowledge the work that has been done before me and so I would like to give a special thanks to all who worked in the field as their research was both an inspiration and a foundation to my own.

A special thanks to all my friends, in particular, Carolina Furtado, João Sottomayor, Manel Sousa, Pedro Cavaleiro and Rodrigo Paiva Tavares for all the support and also all the fun that we had together not only during the elaboration of the thesis but also during the previous 5 years. I would be remiss if I did not thank all my friends in BEST (Board of European Students of Technology) because it is in part due to the time spent with them in the association that I was able to grow into a better person and hopefully as an engineer.

Finally, I would like to thank my family for their unconditional support and help and above all having provided me with shining examples of work ethic and perseverance.

Abstract

Filament winding is one of the most used automated manufacturing processes for composite parts. It yields high strength and lightweight products and has been widely implemented in many industries due to the quality of the produced parts and the cost-effectiveness of the process itself.

The study of fibre trajectories on the mandrel surface is of vital importance in order to produce parts with the desired mechanical properties, reduce manufacturing times and save costs.

With the study of the fibre trajectories in mind, firstly geodesic trajectories are presented as these have no lateral forces and guarantee fibre stability. Secondly, non-geodesic trajectories are proposed in a way that maintains fibre stability and avoids fibre slippage.

With a way to describe these trajectories already in place, a method of creating patterns of fibre trajectories which cover the whole mandrel and allow the production of complete pieces was proposed. Therefore, it became necessary to determine a way of controlling a machine in order to produce these pieces. A method of controlling machine axes movement in relation to the fibre trajectories on the mandrel is presented and tested.

The study of the fibre trajectories lead to the realization that in order to truly describe the tow path, it would be necessary to create a model that accounted for different types of cross-sections and fibre overlap as these create stresses and should be studied in finite element analysis.

In order to automatically create the fibre trajectories and machine control data with only some specific design inputs, a Matlab^{TM} code named Pathwind was created, presented, tested and experimentally verified to varying degrees of success.

Resumo

Enrolamento filamentar é um dos processos de fabrico automatizados mais utilizados para a criação de peças em materiais compósitos. Produz peças leves de elevada resistência e já foi adotada por várias indústrias devido à qualidade das peças produzidas e ao baixo custo de produção associado ao processo.

O estudo das trajetórias das fibras na superfície do mandril é de vital importância de modo a produzir peças com as desejadas propriedades mecânicas, reduzir tempos de produção e poupar nos custos.

Com o estudo das trajetórias das fibras em mente, inicialmente são apresentadas trajetórias geodésicas pois estas não têm forças laterais e garantem a estabilidade da fibra. Posteriormente são propostas trajetórias não geodésicas que mantenham a estabilidade da fibra e não provoqquem escorregamento de fibras.

Tendo já definido um processo de descrever estas trajetórias, um método de criar padrões de trajetórias de fibras que cubram todo o mandril e permitem a produção de peças completas é proposto. Foi portanto necessário determinar uma maneira de controlar uma máquina de modo a produzir estas peças. Um método de controlar os eixos da máquina em relação às trajetórias da fibra no mandril é apresentado e testado.

O estudo das trajetórias das fibras levou à realização que de modo a descrever as verdadeiras trajetórias das fibras, seria necessário criar um modelo que tivesse em conta diferentes tipos de secções e sobreposições de fibras pois estes criam tensões que deviam ser estudados com recurso a métodos de elementos finitos.

De forma a criar trajetórias de fibras e dados de controlo de máquina automaticamente só com algumas especificações iniciais, um programa em Matlab^{TM} foi criado, apresentado, testado e experimentalmente verificado.

Contents

Contents i

List of figures v

List of Symbols and Abbreviations vii

1 Introdution 1
- 1.1 Motivation and Objective . . . 1
- 1.2 Thesis outline . . . 3

2 Literature Review 5
- 2.1 Introduction to Filament Winding . . . 5
- 2.2 Trajectories and differential geometry . . . 6
  - 2.2.1 Fundamental forms of surfaces . . . 6
  - 2.2.2 Geodesics . . . 8
  - 2.2.3 Curvatures . . . 10
  - 2.2.4 Geodesics revisited . . . 11

3 Path Creation 13
- 3.1 Geodesic paths . . . 13
  - 3.1.1 Cylinders . . . 13
  - 3.1.2 Cones . . . 14
  - 3.1.3 Other geometries . . . 15
- 3.2 Non-geodesic paths . . . 16
  - 3.2.1 Cylinders . . . 16
  - 3.2.2 Cones . . . 19
- 3.3 Turnaround zones . . . 21
- 3.4 Pattern generation . . . 21

4 Machine Programming 23
- 4.1 Machine configuration . . . 23
- 4.2 Axes movements . . . 24
- 4.3 Machine parameters . . . 27
  - 4.3.1 Dimensions . . . 27
  - 4.3.2 Resolutions . . . 27
  - 4.3.3 Maximum Velocities . . . 28
  - 4.3.4 Maximum Accelerations . . . 28
- 4.4 Automatic generation of CNC code . . . 28

5 Modelling of Real Tow Path **31**
5.1 Tow cross-section 31
5.1.1 Ellipsoidal Cross-section 32
5.1.2 Lenticular Cross-section 33
5.1.3 Round Cross-section 34
5.1.4 Racetrack Cross-section 34
5.2 Fibre overlap 34

6 Results and Discussion **37**
6.1 Virtual Testing 37
6.1.1 Test 1 - Cylindrical mandrel 37
6.1.2 Test 2 - Conical mandrel 41
6.1.3 Test 3 - Cylindrical mandrel 45
6.1.4 Test 4 - Conical mandrel 49
6.2 Experimental validation 53

7 Conclusion and Future Work **57**
7.1 Conclusion 57
7.2 Future Work 58

A Matlab Code **61**
A.1 Pathwind 61
A.2 Christoffel coefficients function 132
A.3 Pattern determination 134

B User Manual **139**
B.1 Structure 139
B.2 Data input 140
B.3 Machine parameters input 141

List of Figures

1.1 Pultrex MODwind 1S 6NC - 6 axis Filament Winding Machine present at the INEGI facilities . . . 2

2.1 Simple lathe winder [1] . . . 5

3.1 Geodesic on a cylinder [2] . . . 13
3.2 α with respect to the meridional and parallel coordinates [3] . . . 14
3.3 Geodesic on a cone [2] . . . 15
3.4 Conical shell [3] . . . 15
3.5 Forces acting along a fibre path [4] . . . 16
3.6 Cylindrical mandrel . . . 17
3.7 Increments on the mandrel surface . . . 19
3.8 Conical mandrel . . . 19
3.9 Turnaround of a fibre on a mandrel surface . . . 21
3.10 Pattern Number and Skip Index explained [5] . . . 22

4.1 Axes movements in the Pultrex MODwind 1S 6NC . . . 23
4.2 Mandrel and pay-out eye 1 . . . 24
4.3 Mandrel and pay-out eye 2 . . . 25
4.4 Mandrel and pay-out eye 3 . . . 25
4.5 Mandrel and pay-out eye 4 . . . 26

5.1 Tow cross-sectional shapes; (a) Ellipsoidal, (b) Lenticular, (c) Rectangular, (d) Circular, (e) Racetrack, and (f) Tow element. [6] . . . 31
5.2 Tow cross-sectional shapes; (a) Semi-ellipsoidal, (b) Semi-lenticular, (c) Rectangular, (d) Semi-circular, (e) Semi-racetrack, and (f) Tow element. [6] . . . 31
5.3 Pathwind tow cross section . . . 32
5.4 Ellipsoidal tow cross-section. . . . 32
5.5 Lenticular tow cross-section. [7] . . . 33
5.6 Round tow cross-section. . . . 34
5.7 Racetrack tow cross-section. . . . 34
5.8 Tow cross-section as a collection of paths . . . 35
5.9 Representation of tow overlap . . . 35

6.1 Parameter variation along one cycle . . . 38
6.2 Tow trajectory along one cycle . . . 39
6.3 Tow paths along the mandrel for all the cycles in one layer . . . 39
6.4 Machine axes' movements along one cycle . . . 40

6.5 Machine axes' velocities along one cycle 40
6.6 Completely wound piece with tow overlap 41
6.7 Completely wound piece with tow overlap in detail 41
6.8 Parameter variation along one cycle 42
6.9 Tow trajectory along one cycle 43
6.10 Tow paths along the mandrel for all the cycles in one layer 43
6.11 Machine axes' movements along one cycle 44
6.12 Machine axes' velocities along one cycle 44
6.13 Completely wound piece with tow overlap 45
6.14 Parameter variation along one cycle 46
6.15 Tow trajectory along one cycle 46
6.16 Tow paths along the mandrel for all the cycles in one layer 47
6.17 Machine axes' movements along one cycle 47
6.18 Machine axes' velocities along one cycle 48
6.19 Completely wound piece with tow overlap 48
6.20 Completely wound piece with tow overlap - detail 49
6.21 Parameter variation along one cycle 50
6.22 Tow trajectory along one cycle 50
6.23 Tow paths along the mandrel for all the cycles in one layer 51
6.24 Machine axes' movements along one cycle 51
6.25 Machine axes' velocities along one cycle 52
6.26 Tow overlap with one cycle wound 52
6.27 Completely wound piece with tow overlap 53
6.28 Winding process in the geodesic area 6.28a and in the turn around area 6.28b 54
6.29 Completely wound piece 55
6.30 Size of the produced part 6.30a and of the geodesic section 6.30b . . 55
6.31 Tow width 6.30a and angle measurement in the geodesic winding area 6.30b 56

List of Symbols and Abbreviations

A	Mandrel rotation
AFP	Advanced fibre placement
α	Winding angle
b	Tow width
B	Feed eye rotation
β	Angle in tow cross-sections
C	Yaw rotation
CAD	Computer aided design
CAM	Computer aided manufacturing
COPV	Composite over-wrapped pressure vessel
$\Delta\phi$	Angle of mandrel axis rotation
$\Delta\phi_n$	Angle of mandrel axis rotation in one cycle
$\Delta\phi_{dwell}$	Angle of mandrel axis rotation while dwelling in the same z coordinate
E	Coefficient of the first fundamental form
f_b	Lateral force on the fibre (tangential to the mandrel surface)
f_f	Friction force between the fibre and the surface
f_n	Normal force on the fibre (normal to the mandrel surface)
F	Coefficient of the first fundamental form
ϕ	Angle in the parallel direction of a surface of revolution
FEA	Finite Element Analysis
G	Coefficient of the first fundamental form
Γ	Christoffel coefficients
H	Mean curvature
INEGI	Institute of Science and Innovation in Mechanical and Industrial Engineering
k	Curvature
k_1	Principal curvature
k_2	Principal curvature
k_g	Geodesic curvature
k_m	Meridional curvature
k_n	Normal curvature
k_p	Parallel curvature
K	Gaussian curvature
L	Coefficient of the second fundamental form

L_f	Arclength
$L_{f-cycle}$	Length of fibre used in one cycle
$L_{f-total}$	Total length of fibre used
M	Coefficient of the second fundamental form
μ	Static coefficient of friction
N	Coefficient of the second fundamental form
N_c	Number of cycles
Ω	Angle between $\Delta\phi_n$ and the closest full rotation ($2k\pi$)
PN	Pattern number
r	Surface radius
R	Cylinder radius
ρ	Surface radius and meridian coordinate
s	Arclength
S	Vector describing a surface
SI	Skip index
t_y	Tow thickness
θ	Spherical coordinate
τ	Top angle of a cone
u	Independent parameter
v	independent parameter
x	x coordinate
X	Carriage translation
y	y coordinate
Y	Cross carriage translation
z	z coordinate
z_{min}	Lowest z coordinate of the fibre on the mandrel
z_{max}	Greatest z coordinate of the fibre on the mandrel
Z	Vertical feed eye translation

Chapter 1

Introdution

1.1 Motivation and Objective

Filament Winding is a manufacturing process for composite parts and components which yields high strength and lightweight products. It is mainly used for the production of axis-symmetric parts - such as COPV (Composite Over-wrapped Pressure Vessels), storage tanks, different types of high pressure pipes, transmission drive shafts, pole vaulting poles and many others - due to the kinematics of the process itself which makes these types of geometries easier to manufacture. However, it can also be used to produce non axis-symmetric parts with great success and wide implementation as can be observed in the manufacturing of wind turbine blades[8].

Due to the relative ease with which the process can be automated, it has proved to be not only a high performance manufacturing method but also very cost effective when producing large numbers. Due to these two factors, this process has been widely adopted in industries from all around the world and for a wide variety of applications [9].

The process uses a continuous reinforcement (glass fibre, carbon fibre, wire, etc.) either previously impregnated (prepreg: thermoset or thermoplastic) or locally impregnated in an open bath or enclosed device before reaching the mandrel (wet winding) with a thermosetting matrix that is placed on a revolving mandrel and then cured. Depending on how the continuous reinforcement is applied on the mandrel, different trajectories can be made and different winding patterns can be obtained which in turn produce pieces with different appearance and mechanical properties.[10][9]

In INEGI (Institute of Science and Innovation in Mechanical and Industrial Engineering) - Institute where the thesis was developed - there has been a long history of work in filament winding. Through this work, a lot of information regarding different aspects of the process have been thoroughly studied and investigated and have transformed the institute into a reference of excellence in filament winding development. The Filament winding machine that INEGI has at its disposal is a Pultrex MODwind 1S 6NC (Figure 1.1) which has 6 controllable axes and is able to produce most of the conceivable parts. However, during all of this work, the actual trajectories of the continuous reinforcement had never been closely studied, making

the institute rely on external software to design its produced parts.

Figure 1.1: Pultrex MODwind 1S 6NC - 6 axis Filament Winding Machine present at the INEGI facilities

The study of trajectories of the continuous reinforcement on a mandrel surface is of vital importance because it allows the evaluation and determination of the trajectories which provide the best mechanical properties within the designer's specified geometrical and physical limits.

Therefore, the aim of this thesis is to better understand, model and evaluate the different types of trajectories that can be made along the surfaces of axis-symmetric mandrels, interpret how these can be used to produce complete parts with different winding patterns and finally implement them in a filament winding machine which will produce the desired parts.

To achieve this goal, a numerical code was proposed which would be able to automatically create the trajectories for the continuous reinforcement according to the geometric and material specifications provided by the designer. This code was named Pathwind and was developed according to what was determined to be INEGI's specific needs.

To develop a functioning and robust code, the establishment and implementation of a model which could perform the following tasks was needed:

- describe different geodesic paths along the surfaces of cylinders and truncated cones;
- describe different non-geodesic paths while still guaranteeing fibre stability (no slippage between fibre and mandrel or fibre and fibre) along the surfaces of cylinders and truncated cones;
- determine different winding patterns for specific mandrels, tow widths and winding angles;
- determine axes movements for a filament winding machine and generate CNC code for its controller in order to produce parts with a previously selected pattern;

- Attempt to describe the real tow path with different types of cross sections and tow overlap.

For the development of the code and in order to first test the different concepts and ideas, it was determined that a first approach of only allowing mandrels which were cylinders or truncated cones was accepted. This is also in line with the recent work being developed at INEGI in filament winding which had been specifically for different sizes and layups of cylindrical tubes. Any other axis-symmetric mandrel geometry can be discretized into a multitude of cylinders and truncated cones and, therefore, this approach can be used in the future for a more generic solution.

This code was developed in Matlab^{TM}.

1.2 Thesis outline

The thesis is divided into seven chapters and two appendixes:

- Chapter 1 introduces the problem and defines the objectives for the thesis, as well as its structure;
- Chapter 2 provides some background information about filament winding and introduces the mathematical concepts needed to understand the development of the code;
- Chapter 3 describes and explains the models used to characterize the tow paths along a mandrel surface and how they were implemented;
- Chapter 4 describes the Filament Winding machine which was used in INEGI and provides a solution to how the different axes' should move in order to produce the previously defined paths along an actual mandrel surface;
- Chapter 5 specifies in which way the tow path can be described in order to completely and truly characterize its behaviour;
- Chapter 6 presents the results of several virtual tests on the code developed and shows an actual filament wound part which was made solely by the use of the Pathwind code, thus validating it experimentally;
- Chapter 7 has the conclusions to the different parts of the project and presents the future work which still needs to be addressed;
- Appendix A displays the Matlab^{TM} code which was produced for this thesis. A.1 is the code of the main Pathwind program, A.2 is a function used by the main program and produces the fundamental forms and Christoffel coefficients of the mandrel surface and A.3 is a pattern determination algorithm [11];
- Appendix B is a simple user manual which explains how to use the Pathwind program.

Chapter 2

Literature Review

2.1 Introduction to Filament Winding

Filament winding first started in the 1950s and is one of the oldest methods for processing composites. Due to its simplicity, it was one of the first to be automated and therefore one of the first to be able to guarantee consistent results to its parts. As a result of this simplicity and ease to automate, it remains one of the most competitive processes in making quality lightweight parts as well as being highly cost-effective [3][12].

The first filament wound parts were the rocket engine cases in the 1950s and in order to automate this process, the first filament winding machines were developed. These were simple 2 axes machines (the spindle rotation and the carriage translation) which were adapted from lathes (producing the lathe type filament winder or lathe winder). From the lathe winder, improvements were made in order to open the possibility of creating different, more complex parts. Consequently, new axes were introduced beginning with the cross carriage while still producing only axis-symmetric parts. From then on, different configurations for filament winding machines were attempted and after the lathe winder came the polar winder, the racetrack winder and the tumble winder for example [9][10].

Figure 2.1: Simple lathe winder [1]

During the 1960s the aerospace industry transformed the composite industry

and started heavily investing in its research. With this research came the dawn of the Composite Overwrapped Pressure Vessels (COPV) which allowed much greater internal pressures while simultaneously weighing less than their metallic predecessors. This was obviously highly appreciated in rocket design and COPV have been used in their designs ever since [1].

The sports and automotive industries have also been major backers of the filament winding manufacturing process and have adopted it for a variety of products. As has been mentioned before, transmission drive shafts have been made in this way, especially for racing vehicles, reducing weight and increasing performance. Pole vaulting poles are now made exclusively by filament winding as it is possible to control the elasticity and rigidity by altering the winding patterns. Golf clubs made in this way have also been widely adopted in recent years, providing yet another industry where this technology can grow. All of these products are cylindrical or conical filament wound parts.

The advent of the computer and CNC technology provided a major leap in filament winding. It granted the ability to better control fibre placement and positional accuracy, greater speed control and the possibility of interaction between the newly invented simulation software and the machine allowed for more complex shapes and forms to be made. During the years many CAD/CAM software and environments were developed which would enable a user of a CNC-controlled filament winding machine the possibility of designing parts for different mandrel shapes and produce their own parts [3].

Some of the existing CAD/CAM software for filament winding applications are: CadFil^{TM}, CadWind^{TM}, FiberGraphix^{TM}, CadPath^{TM} and ComposicaD^{TM}. These software provide varying degrees of control over the design process and different applications, such as: axis-symmetric and non axis-symmetric mandrels (including elbows and T-junctions)[13][14], FEA (Finite Element Analysis) mesh output, built-in material data bases, graphic display of the process and final product, graphical user interfaces, machine output with time and speed optimization. Besides these available software, throughout the years many institutions and companies have developed their own software either for permanent personal use or merely for research purposes. The constant improvement of CAD software for filament winding has helped advance the industry and increase the possible applications of the manufacturing process itself.

2.2 Trajectories and differential geometry

2.2.1 Fundamental forms of surfaces

The fundamental forms of a surface permit the calculation of curvature and other metric properties.

It was considered that a three dimensional surface S can be parametrized as a vector function of two independent real parameters u and v in the following way:

$$\mathrm{S}(u,v) = \{x(u,v), y(u,v), z(u,v)\} \quad \text{with} \quad u,v \in \Re \tag{2.1}$$

The arc length of a curve belonging to this surface is given by [15][16][17][18][19]:

$$s = \int \sqrt{(dx)^2 + (dy)^2 + (dz)^2} \tag{2.2}$$

The differentials encountered in equation 2.2 can be expressed in the following way [18]:

$$d\#^2 = \left(\frac{\partial \#}{\partial u}\right)^2 (du)^2 + 2\frac{\partial \#}{\partial u}\frac{\partial \#}{\partial v} dudv + \left(\frac{\partial \#}{\partial v}\right)^2 (dv)^2 \tag{2.3}$$

where # is a general symbol which represent x, y or z.

From equations 2.2 and 2.3 we obtain the following:

$$s = \int \left(\sqrt{G + 2F\frac{du}{dv} + E\left(\frac{du}{dv}\right)^2}\right) dv = \int \left(\sqrt{G\left(\frac{dv}{du}\right)^2 + 2F\frac{dv}{du} + E}\right) du \tag{2.4}$$

where:

$$E = \left(\frac{\partial x}{\partial u}\right)^2 + \left(\frac{\partial y}{\partial u}\right)^2 + \left(\frac{\partial z}{\partial u}\right)^2 = \mathrm{S}_u.\mathrm{S}_u \tag{2.5}$$

$$F = \frac{\partial x}{\partial u}\frac{\partial x}{\partial v} + \frac{\partial y}{\partial u}\frac{\partial y}{\partial v} + \frac{\partial z}{\partial u}\frac{\partial z}{\partial v} = \mathrm{S}_u.\mathrm{S}_v \tag{2.6}$$

$$G = \left(\frac{\partial x}{\partial v}\right)^2 + \left(\frac{\partial y}{\partial v}\right)^2 + \left(\frac{\partial z}{\partial v}\right)^2 = \mathrm{S}_v.\mathrm{S}_v \tag{2.7}$$

The three coefficients shown in equations 2.5, 2.6 and 2.7 (E, F and G) are the coefficients of the first fundamental form. Where E represents the metric in the meridional direction and G represents the metric in the parallel direction. S_u and S_v are the derivatives of the surface vector S in respect to u and v respectively

[18][16][3][19].

The coefficients of the second fundamental form will also be important in the derivation of curvatures and are shown below:

$$L = \frac{\det \begin{bmatrix} \mathrm{S}_{uu} \\ \mathrm{S}_u \\ \mathrm{S}_v \end{bmatrix}}{\sqrt{EG - F^2}} = \mathbf{n}(u,v).\mathrm{S}_{uu} \tag{2.8}$$

$$M = \frac{\det \begin{bmatrix} \mathrm{S}_{uv} \\ \mathrm{S}_u \\ \mathrm{S}_v \end{bmatrix}}{\sqrt{EG - F^2}} = \mathbf{n}(u,v).\mathrm{S}_{uv} \tag{2.9}$$

$$N = \frac{\det \begin{bmatrix} \mathrm{S}_{vv} \\ \mathrm{S}_u \\ \mathrm{S}_v \end{bmatrix}}{\sqrt{EG - F^2}} = \mathbf{n}(u,v).\mathrm{S}_{vv} \tag{2.10}$$

Since axis-symmetric surfaces have a direction u which is perpendicular to the direction v, by definition F will be zero. If we consider our case of an axis-symmetric surface (shell of revolution), we can redefine our surface and determine the values of the fundamental forms [12][3]:

$$\mathrm{S}(u,v) = \{a(u)\cos(v), a(u)\sin(v), b(u)\} \quad \text{with} \quad u, v \in \Re \tag{2.11}$$

$$E = a'(u)^2 + b'(u)^2, \quad F = 0, \quad G = a(u)^2 \tag{2.12}$$

$$L = -\frac{a''b' - a'b''}{\sqrt{(a')^2 + (b')^2}}, \quad M = 0, \quad N = \frac{ab'}{\sqrt{(a')^2 + (b')^2}} \tag{2.13}$$

2.2.2 Geodesics

Geodesics are a class of curves which are generally described as being the shortest path along a surface between 2 points. It is often described by imagining you are driving a car on the surface of the earth and going in a straight line. Even though you are not actually going in a straight line, you are still taking the shortest path. The importance of geodesic paths for filament winding is quite simple: in geodesic paths there are no lateral forces of any kind applied, the only forces applied will be normal to the surface and so, fibre slippage will be avoided.

As an example, the geodesics along a sphere are its great circles (equatorial circles in which the centre is coincident with the centre of the sphere) and in a cylinder they form a helix.

In order to find the minimum path, a function is proposed that relates u and v in a way that minimizes the arclength [3][12]:

$$s = \int f(u, v, \frac{dv}{du})du \tag{2.14}$$

If we express $\frac{dv}{du}$ as v' and fix the begin and endpoints:

$$\delta s = \delta \int_{u_1}^{u_2} f(u, v, v')du = \int_{u_1}^{u_2} (\frac{\partial f}{\partial v}\partial v + \frac{\partial f}{\partial v'}\partial v')du = \int_{u_1}^{u_2} (\frac{\partial f}{\partial v}\partial v + \frac{\partial f}{\partial v'}\frac{\partial}{\partial u}\partial v)du = 0 \tag{2.15}$$

$$\delta s = \int_{u_1}^{u_2} \frac{\partial f}{\partial v} - \partial v \partial u + \frac{\partial f}{\partial v'}\partial v\Big|_{u_1}^{u_2} - \int_{u_1}^{u_2} \partial v \frac{d}{du}(\frac{\partial f}{\partial v'})du = \int_{u_1}^{u_2} (\frac{\partial f}{\partial v} - \frac{d}{du}(\frac{\partial f}{\partial v'}))\delta v du = 0 \tag{2.16}$$

And as the integral will be zero independent of the change to δv, this leads us to the Euler-Lagrange equation [3][12]:

$$\frac{\partial f}{\partial v} - \frac{d}{du}(\frac{\partial f}{\partial v'}) = 0 \tag{2.17}$$

By replacing the function f with the expression for curve length (expression 2.4) and after rearrangement, we obtain [3]:

$$\frac{v'^2 E_v + 2v' F_v + G_v}{2\sqrt{Ev'^2 + 2Fv' + G}} - \frac{d}{du}(\frac{Ev' + F}{2\sqrt{Ev'^2 + 2Fv' + G}}) = 0 \tag{2.18}$$

Where E_v, F_v and G_v are the derivatives of E, F and G with respect to v respectively. In order to solve the above equation and since the fundamental forms are usually in function of both u and v, numerical methods are usually employed. In the case of geodesics, this problem is usually solved using Christoffel coefficients which are displayed below [18][12]:

$$\Gamma^u_{vv} = \frac{2EF_v + EE_u - FE_u}{2(EG - F^2)}, \quad \Gamma^u_{uv} = \frac{EG_v - FE_u}{2(EG - F^2)}, \quad \Gamma^u_{uu} = \frac{EG_u - 2FF_u + FG_v}{2(EG - F^2)} \tag{2.19}$$

$$\Gamma^v_{vv} = \frac{GE_v - 2FF_v + FE_u}{2(EG - F^2)}, \quad \Gamma^v_{uv} = \frac{GE_u - FG_v}{2(EG - F^2)}, \quad \Gamma^v_{uu} = \frac{2GF_u - GG_v - FG_u}{2(EG - F^2)} \tag{2.20}$$

$$\Gamma^u_{uv} = \Gamma^u_{vu}, \quad \Gamma^v_{uv} = \Gamma^v_{vu} \tag{2.21}$$

2.2.3 Curvatures

Curvature can be explained as the tendency of a line to change direction. If the curvature of a curve has higher value, it is less straight.

If we consider an angle α between the fibre direction and the meridional, depending on this angle, the curvature k_n which is normal to the surface will have a maximum and a minimum value which are known as the principal curvatures and in the case of axis-symmetric surfaces they are the meridian and parallel curvatures.

The principal curvatures can be related to the Gaussian curvature (K) and to the mean curvature (H) by means of the first and second fundamental forms [18][3][16][19]:

$$K = k_1 k_2 = \frac{LN - M^2}{EG - F^2}, \quad H = \frac{1}{2}(k_1 + k_2) = \frac{LG - 2MF + NE}{2(EG - F^2)} \tag{2.22}$$

$$k_1 = H + \sqrt{H^2 - K}, \quad k_2 = H - \sqrt{H^2 - K} \tag{2.23}$$

We can now express the curvature of the fibre placed on top of and normal to our axis-symmetric surface in the following way:

$$k_n = (k_1 cos^2\alpha + k_2 sin^2\alpha) \tag{2.24}$$

Where k_1 is the meridional curvature k_m and can be expressed in terms of the first and second fundamental forms [4]:

$$k_1 = k_m = N/G \tag{2.25}$$

And k_2 is the parallel curvature k_p and can also be expressed in terms of the first and second fundamental forms [4]:

$$k_2 = k_p = L/E \tag{2.26}$$

I will now introduce the concept of geodesic curvature, k_g. It is a curvature which is tangential to the surface. For a curve to be a geodesic, its geodesic curvature must be equal to 0. Geodesic curvature can be expressed as [18]:

$$k_g = \sqrt{EG - F^2}(\Gamma^u_{vv}u'^3 + (2\Gamma^u_{vu} - \Gamma^v_{vv})u'^2v' + (\Gamma^u_{uu} - 2\Gamma^v_{vu})u'v'^2 - \Gamma^v_{uu}v'^3 + u'v'' - u''v') \tag{2.27}$$

Or:

$$k_g = \sqrt{EG - F^2}\det\begin{bmatrix} u' & u'' + \Gamma^u_{vv}v'^2 + 2\Gamma^u_{uv}u'v' + \Gamma^u_{uu}u'^2 \\ v' & v'' + \Gamma^v_{vv}v'^2 + 2\Gamma^v_{uv}u'v' + \Gamma^v_{uu}u'^2 \end{bmatrix} \tag{2.28}$$

And as the Liouville formula [12]:

$$k_g = \frac{d\alpha}{ds} - \frac{E_v}{2\sqrt{G}}\frac{\partial \ln E}{\partial v}cos\alpha + \frac{1}{2\sqrt{E}}\frac{\partial \ln G}{\partial u}sin\alpha \tag{2.29}$$

And we can express the total curvature by means of the normal curvature and the geodesic curvature:

$$k^2 = k_n^2 + k_g^2 \tag{2.30}$$

2.2.4 Geodesics revisited

A geodesic can now be determined by first expressing the geodesic equations in 2 second order differential equations which we obtained in equation 2.28 by equalling the geodesic curvature k_g to 0 [18][12]:

$$\begin{cases} v'' + \Gamma^v_{vv}v'^2 + 2\Gamma^v_{uv}u'v' + \Gamma^v_{uu}u'^2 = 0 \\ u'' + \Gamma^u_{vv}v'^2 + 2\Gamma^u_{uv}u'v' + \Gamma^u_{uu}u'^2 = 0 \end{cases} \tag{2.31}$$

Where u', u'', v' and v'' are the first and second order derivatives of u and v. The above system of two second order differential equations can be expressed as a system of four first order differential equations like so in order to help solve them:

$$\begin{cases} v' = p \\ u' = q \\ p' = -\Gamma^v_{vv}p^2 - 2\Gamma^v_{uv}pq - \Gamma^v_{uu}q^2 = 0 \\ q' = -\Gamma^u_{vv}p^2 - 2\Gamma^u_{uv}pq - \Gamma^u_{uu}q^2 = 0 \end{cases} \tag{2.32}$$

Chapter 3

Path Creation

3.1 Geodesic paths

3.1.1 Cylinders

As was said before, the geodesic paths along a cylinder describe the form of a helix. If a cylinder surface is spread open, the helix can be seen to describe a straight line and it is easy to view the angle α that the curve does with the meridional direction [3]:

Figure 3.1: Geodesic on a cylinder [2]

Figure 3.2: α with respect to the meridional and parallel coordinates [3]

A helix can easily be parametrized in the following way:

$$x = R\cos(\phi), \quad y = R\sin(\phi), \quad z = C\phi \tag{3.1}$$

Where R is the cylinder radius, ϕ is the angular parameter and C is a constant.
As we can see from figures 3.1 and 3.2 and from the parametrization in equation 3.1:

$$\alpha = \arctan(\frac{R\phi}{C\phi}) = \arctan(\frac{R}{C}) \tag{3.2}$$

Or:

$$C = \frac{R}{\tan\alpha} \tag{3.3}$$

3.1.2 Cones

The description of a geodesic along a conical surface is well defined by the so called Clairaut relation [8][3][12][11]:

$$C_1 = \rho\sin(\alpha) \tag{3.4}$$

Where ρ is the radius at a specific section of the cone, α is the angle that the curve makes with the meridional direction of the cone (its generatrix) at that specific section and C_1 is a constant.

In the case of the cone it is also easy to see that if a conical surface is spread open, the geodesic will describe a straight line:

Figure 3.3: Geodesic on a cone [2]

The geodesic path on a cone can be described with only the angle ϕ and the radius ρ. The angle can be described as a function of the radius [3]:

$$\phi(\rho) = \frac{1}{\sin\tau}(\arcsin(\frac{C_1}{\rho_{max}}) - \arcsin(\frac{C_1}{\rho})) \tag{3.5}$$

Where τ is the angle that a generatrix makes with the axis (or centerline) of the cone and ρ_{max} is the maximum radius of the cone.

Figure 3.4: Conical shell [3]

We can now parametrize the curve:

$$x = \cos\phi, \quad y = \sin\phi, \quad z = \frac{(\rho - \rho_1)(z_2 - z_1)}{\rho_2 - \rho_1} + z_1 \tag{3.6}$$

3.1.3 Other geometries

As has been discussed before, the focus of this thesis is on cylinders and cones. However, for most engineering applications, an axis-symmetric surface can be approximated, to any degree of precision, by a series of segments composed of cones and cylinders and so, it is possible to create a geodesic path onto that surface. The convergence of the geodesic path on the real surface and on the approximated surface has been studied and proven by Gold [20]. It is therefore safe to calculate geodesic paths for an approximated surface and maintain its accuracy and properties.

3.2 Non-geodesic paths

In order to correctly define a non-geodesic path over the surface of a mandrel, we must always guarantee fibre stability. There must be no slippage between the fibre and the mandrel. Therefore, we must study the forces acting on the fibre and determine the possible paths it can take.

Figure 3.5: Forces acting along a fibre path [4]

The stability of the fibre depends on the lateral force f_b (due to the change of the fibre angle) being less than the frictional force f_f. For a static coefficient of friction μ, we consider a Coulomb friction force where [4][15][21][22]:

$$f_f = \mu f_n \tag{3.7}$$

We can therefore conclude that:

$$|f_b/f_n| \leq \mu \tag{3.8}$$

Considering λ to be a parameter number whose value is between $-\mu$ and μ, $|\lambda| \leq \mu$ in order to guarantee fibre stability. The same expression as equation 3.8 can be expressed in terms of the geodesic and normal curvatures[4][15][3][21]:

$$|k_g/k_n| = |\lambda| \tag{3.9}$$

3.2.1 Cylinders

To describe the path in a cylinder we won't be able to use polar coordinates since z cannot be defined as a function of ρ. Therefore, spherical coordinates will be used to describe its surface and calculate the trajectory[4].

Figure 3.6: Cylindrical mandrel

The independent parameters used to describe the surface of the cylinder and the fibre trajectories will be ϕ which represents the angle along the z axis and is the parallel parameter and θ which represents the meridional parameter [4][15]:

$$\rho(\theta) = \frac{R}{\sin\theta} = g(\theta) \tag{3.10}$$

$$S(\phi,\theta) = \{g(\theta)\sin(\theta)\cos(\phi), g(\theta)\sin(\phi), g(\theta)\cos(\theta)\} \tag{3.11}$$

We can now calculate the fundamental forms and the meridional and parallel curvatures can be expressed in terms of the surface parameters[4]:

$$k_m(\theta) = \frac{N}{G} = \frac{2 \times \left(\frac{dg(\theta)}{d\theta}\right)^2 - g(\theta) \times \frac{d^2g(\theta)}{d\theta^2} + g^2(\theta)}{\left(g^2(\theta) + \left(\frac{dg(\theta)}{d\theta}\right)^2\right)^{3/2}} \tag{3.12}$$

$$k_p(\theta) = \frac{L}{E} = \frac{g(\theta) \times \sin(\theta) - \frac{dg(\theta)}{d\theta} \times \cos(\theta)}{g(\theta) \times \sin(\theta) \times \left(g^2(\theta) + \left(\frac{dg(\theta)}{d\theta}\right)^2\right)^{1/2}} \tag{3.13}$$

The terms of the geodesic curvature simplify greatly since $F = 0$ and $M = 0$. By solving equation 2.27 with the newly calculated fundamental forms we are reduced to [4]:

$$k_g = -\frac{d\alpha(\theta)}{d\theta}\frac{\cos(\alpha(\theta))}{\sqrt{G(\theta)}} - \frac{1}{2}\frac{E_\theta}{E}\frac{\sin(\alpha(\theta))}{\sqrt{G(\theta)}} \tag{3.14}$$

By substituting the freshly calculated k_g and k_n into equation 3.9 we arrive at an ordinary differential equation that describes α as a function of θ [4][15][3][23]:

$$\frac{d\alpha(\theta)}{d\theta} = \frac{1}{2}\frac{E_\theta}{E}\tan(\alpha(\theta)) + \lambda\left(\frac{\sqrt{G}}{\cos(\alpha(\theta))}(k_m\cos^2(\alpha(\theta)) + k_p\sin^2(\alpha(\theta)))\right) \tag{3.15}$$

We can completely describe the curve by expressing ϕ as a function of α and θ. In a similar fashion, we can also determine the arclength between 2 points as a function of α and θ as seen below [4][15][3]:

$$\phi(\theta) = \int_{\theta_a}^{\theta_b}\sqrt{\frac{G}{E}}\tan(\alpha(\theta))d\theta \tag{3.16}$$

$$L_f(\theta) = \int_{\theta_a}^{\theta_b}\frac{\sqrt{G}}{\cos(\alpha(\theta))}d\theta \tag{3.17}$$

It is now possible to apply a numerical resolution to the above expressions (3.15, 3.16 and 3.17). To solve equation 3.15, the modified Euler method was applied [4]:

$$\alpha_{i+1} = \alpha_i + h(\frac{d\alpha}{d\theta}(\alpha_{i+1},\theta_{i+1}) + \frac{d\alpha}{d\theta}(\alpha_i,\theta_i))/2 \tag{3.18}$$

Where h is the step and is equal to $(\theta(i+1)-\theta(i))$. Equations (3.16 and 3.17) were solved in a similar manner as can be seen below [4]:

$$\phi_i = \phi_{i-1} + \frac{1}{2}(\theta_i - \theta_{i-1})\left(\sqrt{G_i/E_i}\tan(\alpha_i) + \sqrt{G_{i-1}/E_{i-1}}\tan(\alpha_{i-1})\right) \tag{3.19}$$

$$L_{f_i} = L_{f_{i-1}} + \frac{1}{2}(\theta_i - \theta_{i-1})\left(\sqrt{G_i}/\cos(\alpha_i) + \sqrt{G_{i-1}}/\cos(\alpha_{i-1})\right) \tag{3.20}$$

In order to maintain a constant length increment ds along the fibre path, θ_{i+1} was calculated in the following way:

$$dz = ds\cos(\alpha) = rd\phi \tag{3.21}$$

$$d\phi = \frac{ds\cos(\alpha)}{R} \tag{3.22}$$

Figure 3.7: Increments on the mandrel surface

$$\theta_{i+1} = \arctan\left(\frac{1}{\frac{1}{\tan(\theta_i)} - \frac{ds\cos(\alpha)}{R}}\right) \tag{3.23}$$

3.2.2 Cones

Since conical shells can be described in polar coordinates, these will be used due to their simplicity. The method used to solve the trajectories for conical mandrels is identical to the one used for cylindrical ones but with a different coordinate system and, therefore, different parameters:

Figure 3.8: Conical mandrel

The independent parameters used to describe the surface of the cone and the fibre trajectories will be ϕ which represents the angle along the z axis and is the parallel parameter and ρ which represents the radius and is the meridional parameter:

$$z(\rho) = \frac{(\rho - \rho_1)(z_2 - z_1)}{\rho_2 - \rho_1} + z_1 \quad \text{as was seen in equation 3.6} \tag{3.24}$$

$$S(\phi, \rho) = \{\rho\cos(\phi), \rho\sin(\phi), z(\rho)\} \tag{3.25}$$

By following the same methodology as before, we arrive at the ordinary differential equation that describes α as a function of ρ [3][23]:

$$\frac{d\alpha(\theta)}{d\rho} = \frac{1}{2}\frac{E_\rho}{E}\tan(\alpha(\rho)) + \lambda\left(\frac{\sqrt{G}}{\cos(\alpha(\rho))}(k_m\cos^2(\alpha(\rho)) + k_p\sin^2(\alpha(\rho)))\right) \tag{3.26}$$

We can also describe the curve by expressing ϕ as a function of α and ρ and also determine the arclength between two points as a function of α and ρ as seen below:

$$\phi(\rho) = \int_{\rho_a}^{\rho_b}\sqrt{\frac{G}{E}}\tan(\alpha(\rho))d\rho \tag{3.27}$$

$$L_f(\rho) = \int_{\rho_a}^{\rho_b}\frac{\sqrt{G}}{\cos(\alpha(\rho))}d\rho \tag{3.28}$$

And the numerical resolution also follows what was previously shown as can be seen below:

$$\alpha_{i+1} = \alpha_i + h(\frac{d\alpha}{d\rho}(\alpha_{i+1},\rho_{i+1}) + \frac{d\alpha}{d\rho}(\alpha_i,\rho_i))/2 \tag{3.29}$$

Where h is now equal to $(\rho(i+1) - \rho(i))$.

$$\phi_i = \phi_{i-1} + \frac{1}{2}(\rho_i - \rho_{i-1})\left(\sqrt{G_i/E_i}\tan(\alpha_i) + \sqrt{G_{i-1}/E_{i-1}}\tan(\alpha_{i-1})\right) \tag{3.30}$$

$$L_{f_i} = L_{f_{i-1}} + \frac{1}{2}(\rho_i - \rho_{i-1})\left(\sqrt{G_i}/\cos(\alpha_i) + \sqrt{G_{i-1}}/\cos(\alpha_{i-1})\right) \tag{3.31}$$

In a similar manner to what was done above, z_{i+1} and consequently ρ_{i+1} were calculated in a way that keeps the increment in the arclenght ds approximately constant:

$$z_{i+1} = z_i + ds\cos(\alpha_i)\cos(\tau) \tag{3.32}$$

3.3 Turnaround zones

The turnaround area in a Filament Winding tube or cone is the area where the fibre starts changing the winding angle in order to come back around to where it had passed before at an angle α but now passes at an angle $\pi - \alpha$.

Figure 3.9: Turnaround of a fibre on a mandrel surface

This is achieved by fixing a value to the parameter λ and defining the path of the fibre until it reaches a winding angle α of $\pi/2$ (boundary condition where the fibre changes direction of progression along its axis of revolution). From there, the winding can dwell on the same parallel on the mandrel and keep winding at an α of $\pi/2$ if this is desired. To continue turning, the path does the symmetric of the change that was done before until the angle reaches $\pi - \alpha$.

3.4 Pattern generation

Winding patterns are the patterns formed by the fibre going back and forth along a mandrel and crossing and overlapping itself in certain points.

An important concept to have in mind is the amount of cycles for a given tow bandwidth b and angle α needed to completely cover the mandrel:

$$N_c = ceilling(\frac{2\pi R_{max}}{b} \cos(\alpha|_{r=R_{max}})) \tag{3.33}$$

For two consecutive cycles to be placed continuously in the correct place, there must be a $\Delta\phi$ angle between the beginning and the end of a cycle. In order to achieve the correct $\Delta\phi$ angle while maintaining the overall shape of the tow on the mandrel, it is possible to simply let the fibre dwell in the middle of the turn around when $\alpha = \pi/2$ for a certain angle of $\Delta\phi_{dwell}$. For cylinders, this $\Delta\phi_{dwell}$ is divided equally between both turn arounds while for cones it is divided proportionally to the perimeters of where the dwells are taking place. For the sake of the $\Delta\phi$ angle between

cycles being correct, it must follow a rule. If $\Delta\phi_n$ is the angular change that would normally happen in a cycle without outside interference and $\Omega = \Delta\phi_n mod(2\pi)$, then:

$$\Delta\phi_{dwell} = \Omega \pm \frac{SI}{PN}\frac{2\pi}{N_c} \quad \text{as long as } \frac{N_c}{PN} \in \mathbb{N} \tag{3.34}$$

Where PN is the Pattern Number and represents the number of cycles that need to pass before the tow is placed next to the first one, SI is the Skip Index and represents the number of cycles that the pattern skips. This is explained in the following diagrams (Figure 3.10) where $PN = 5$ and $SI = 1$ and $SI = 2$ respectively:

Figure 3.10: Pattern Number and Skip Index explained [5]

These types of patterns can only be made if $\frac{N_c}{PN} \in \mathbb{N}$ which is not always true. Also, they only work if the turn around is made by a non-geodesic path while this is not always necessary or wanted because the turn around can be made geodesically if there is a small polar opening as is the case of most COPV.

A numerical method to determine possible winding patterns for specific tow widths, winding angles and mandrel geometries (which always comply with the condition $\frac{N_c}{PN} \in \mathbb{N}$) was presented by Johansen et al. [11] and was implemented in A.3.

Chapter 4

Machine Programming

4.1 Machine configuration

The Filament Winding machine present at INEGI's facilities and the one on which this study will focus is, as has been seen before, the Pultrex MODwind 1S 6NC. This is a machine with 6 numerically controlled axis as the name suggests. The axes that the machine has available are (as can be viewed in Figure 4.1):

- Spindle or Mandrel (Axis A);
- Carriage (Axis X)
- Cross carriage (Axis Y)
- Pay-out or Feed-eye rotation (Axis B)
- Vertical axis (Axis Z)
- Yaw axis (Axis C)

Figure 4.1: Axes movements in the Pultrex MODwind 1S 6NC

Due to its high number of movable axes, this machine is capable of producing almost any kind of part. However, the study of this thesis focuses in axis-symmetric parts and, in particular, in cylinders and cones and therefore, not all axes will be analysed and used.

The four axes which will be used are the spindle/mandrel rotation, the carriage movement, the cross carriage movement and the pay-out/feed-eye rotation. These are sufficient for the intended purposes and would even be enough for much more complex parts [24].

4.2 Axes movements

In order to study the movement required by each axis, it was required to study the position that each axis has to be in when the tow is being placed on the mandrel at a specific point. By knowing the relationship between the point on the mandrel and the axes' position, it becomes easy to transform the tow path coordinates into axis movements [25][26][24][13].

Figure 4.2: Mandrel and pay-out eye 1

Figure 4.3: Mandrel and pay-out eye 2

Figure 4.4: Mandrel and pay-out eye 3

Figure 4.5: Mandrel and pay-out eye 4

As can be seen in figure 4.4, we can now express α as a function of X and Y:

$$\frac{X}{Y - YRef} = \tan\alpha \tag{4.1}$$

Where $YRef$ is the yaw radius and Y and X represent the distance between the mandrel centreline where the tow is being placed and the pay-out eye.

From figure 4.3 we can determine the distance e as a function of known distances and angles:

$$e = \sqrt{(Y - YRef)^2 - \rho^2}\sin(\beta) = \sqrt{(Y - YRef)^2 - \rho^2}\frac{\rho}{Y - YRef} \tag{4.2}$$

Where ρ is the mandrel radius at the deposition location and β is the angle that the tow makes with the horizontal when leaving the pay-out eye.

$$X' = \frac{\sqrt{(Y - YRef)^2 - \rho^2}}{\tan\alpha}\cos\beta \tag{4.3}$$

It is now possible to determine the angle $B1$:

$$B_1 = \arctan(\frac{e}{X'}) = \arctan(\frac{\rho \tan \alpha}{(Y - YRef) \cos \beta}) \tag{4.4}$$

And it is possible to determine B in relation to $B1$:

$$B = \frac{\pi}{2} - B_1 \tag{4.5}$$

By fixing the distance Y, it is now possible to have a geometric description of how three different axes have to move in relation to the path on the mandrel (which is determined by the movement of the fourth axis - the spindle rotation): the carriage movement X, the cross carriage movement Y and the pay-out rotation B. This is a simple approach of how the axes should move. In this manner, the movement of the cross carriage exists only to maintain a certain distance to the mandrel surface. If more complex mandrels are used, more axes should be used and in different ways.

4.3 Machine parameters

In this section the data referring to the physical constraints of the Pultrex MODwind 1S 6NC Filament Winding Machine will be displayed. These constraints were used to generate the CNC code - which was later read by the machine - in order to avoid any form of malfunction.

This data was provided by INEGI and was obtained through careful measurements and known machine data. These are the values used while employing external software to program the machine.

4.3.1 Dimensions

Carriage range (along the X axis as measured from the spindle housing): 165.15mm to 2644.7mm.
Cross carriage range (along the Y axis as measured from the forward face of the carriage): -120mm to 311mm.
Vertical axis range (Z axis as measured from the spindle axis): 0mm to 364mm.
Pay-out eye width: 50mm.
Yaw radius (as measured from the Yaw axis C to the tip of the feed eye): 260mm.

4.3.2 Resolutions

Mandrel (axis A): 360 increments/revolution.
Carriage (axis X): 1 increment/mm.

Cross carriage (axis Y): 1 increment/mm.
Pay out rotation (axis B): 360 increments/revolution.
Vertical axis (axis Z): 1 increment/mm.
Yaw axis (axis C): 360 increments/revolution.

4.3.3 Maximum Velocities

Mandrel (axis A): 150 revolutions/minute.
Carriage (axis X): 1 m/s.
Cross carriage (axis Y): 0.5 m/s.
Pay out rotation (axis B): 115 revolutions/minute.
Vertical axis (axis Z): 0.5 m/s.
Yaw axis (axis C): 100 revolutions/minute.

4.3.4 Maximum Accelerations

The accelerations in this context are defined as the maximum permissible velocity change of an axis between two control data records.

Mandrel (axis A): 8 revolutions/minute.
Carriage (axis X): 4 m/s.
Cross carriage (axis Y): 30 m/s.
Pay out rotation (axis B): 60 revolutions/minute.
Vertical axis (axis Z): 1 m/s.
Yaw axis (axis C): 45 revolutions/minute.

4.4 Automatic generation of CNC code

The Sinumerik controller which is used with the machine requires two different files. One Main Program File (*.mpf) and one Sub Program File (.spf). The Sub Program File is a sub-routine within the Main Program File which has the command lines to order the machine to move in a way to make the tow complete one full cycle around the mandrel. The Main Program File has to start the machine, place every machine part in the start position (beginning of the cycle) and call the SPF the correct number of times (N_c) in order to produce a part.

The feed rate of each command line was calculated as a function of total axes movement and time[5]:

$$Feed = \frac{\frac{\sqrt{A_1^2+A_2^2+A_3^2+A_4^2}}{t}}{60} \tag{4.6}$$

Where A_n is the movement (amount of increments) in the nth axis and t is the time in seconds which the command line takes to perform.

The movement on each axis is calculated as the difference of increments in this axis' direction between two consecutive points on the path.

The time which is used to calculate the feed rate in each command line is the maximum of the times that each axis takes to move its allotted increments at its maximum velocity. This way we ensure that the machine is moving at its maximum speed while maintaining the correct path and not surpassing its speed limitations.

An example of a line of CNC code (part of the sub program file) which was developed by the Pathwind code and was tested and validated as seen in Chapter 6.2 is shown below:

```
N04640 A1.675 X-0.164 Y0.000 B-0.301 F36746.837
```

N04640 is the command line, A1.675 is the incremental movement in axis A (rotation in degrees for axis A), X-0.164 is the incremental movement in axis X (translation in mm for axis X), Y0.000 is the incremental movement in axis Y (translation in mm for axis Y), B-0.301 is the incremental movement in axis B (rotation in degrees for axis B) and F36746.837 is the feed which was calculated as shown in expression 4.6.

Chapter 5

Modelling of Real Tow Path

The path that a fibre tow makes after being wound around a mandrel can not be simply described by its paths along the surface. There are ups, downs, tightenings and releases in the sections where there is fibre overlap. This fibre crimp can create residual stresses and should be studied. That is why one of the objectives of the presented code was to try and model the reality of the tow path.

5.1 Tow cross-section

Throughout the years many different tow cross-sections have been presented as a representation of how the tow actually should be before or after tension is applied. These studies have mostly been done for Representative Volume Elements (RVE) of fabrics or for Advanced Fibre Placement (AFP) techniques but I will apply their studies to our case of filament winding [6][7][27][28].

The proposed cross-sections that should be applied to filament winding for studying are the following ones:

Figure 5.1: Tow cross-sectional shapes; (a) Ellipsoidal, (b) Lenticular, (c) Rectangular, (d) Circular, (e) Racetrack, and (f) Tow element. [6]

Figure 5.2: Tow cross-sectional shapes; (a) Semi-ellipsoidal, (b) Semi-lenticular, (c) Rectangular, (d) Semi-circular, (e) Semi-racetrack, and (f) Tow element. [6]

These cross-sections can either represent the tow along its entire path or a variation of cross sections depending on the position, tension and other outside influences can happen along the tow path. When tension is applied to a tow which is on the surface of the mandrel, a semi-lenticular or semi-ellipsoidal cross-section might be the most appropriate for example. Future studies need to be made in order to determine the best cross-section to be applied depending on the situation. For now, the presented Pathwind Matlab^{TM} code is only able to make rectangular or semi-ellipsoidal cross sections and maintain the cross-section through its entire path.

Figure 5.3: Pathwind tow cross section

5.1.1 Ellipsoidal Cross-section

Figure 5.4: Ellipsoidal tow cross-section.

It is possible to define the edge of the tow (coordinates x and y) in function of the tow width and thickness:

$$t_y = 2b1 \tag{5.1}$$

$$b = 2a1 \tag{5.2}$$

$$x = a1\cos(\beta), \quad y = b1\sin(\beta) \tag{5.3}$$

Where t_y is the tow thickness, b is the tow width, $a1$ and $b1$ are the major and minor axes of the ellipse respectively. β is the angle between the positive x semi-axis and the direction of a point P on the ellipse edge.

5.1.2 Lenticular Cross-section

Figure 5.5: Lenticular tow cross-section. [7]

It is possible to define the edge of the tow in function of the tow width and thickness:

$$R1 = \frac{b^2 + t_y^2}{4t_y} \tag{5.4}$$

$$\gamma = \arccos(1 - \frac{t_y}{2R1}) \tag{5.5}$$

$$A = 2\gamma R1^2 - R1^2 \sin(2\gamma) \tag{5.6}$$

Where t_y is the tow thickness, b is the tow width, $R1$ is the radius of curvature, A is the cross-sectional area and γ is the represented angle.

5.1.3 Round Cross-section

Figure 5.6: Round tow cross-section.

It is possible to define the edge of the tow (coordinates x and y) in function of the tow radius:

$$t_y = 2R1 \tag{5.7}$$

$$b = 2R1 \tag{5.8}$$

$$x = R1\cos(\beta), \quad y = R1\sin(\beta) \tag{5.9}$$

Where t_y is the tow thickness, b is the tow width and $R1$ is the radius. β is the angle between the positive x semi-axis and the direction of a point P on the circle edge.

5.1.4 Racetrack Cross-section

Figure 5.7: Racetrack tow cross-section.

The racetrack cross-section can be represented by 2 semi circles on both ends of the rectangle. Since the properties of the circle have already been shown and the form of a rectangle is well known, it is believed that this shape is already sufficiently defined.

5.2 Fibre overlap

In order to provide an accurate framework in which to calculate and represent the tow path, it is represented as a collection of paths. This collection of paths is united

and forms a cohesive cross-section and follows the path as it was defined before.

Figure 5.8: Tow cross-section as a collection of paths

The overlap can then be performed whenever the tow encounters another part of itself. The tow is therefore always in contact with either the mandrel or a previous section of itself. Whenever a path on the tow encounters a previous section of the tow, the next point on its path will have $\rho = \rho_{mandrel} + t_y/\cos\tau$ and $z = z_{mandrel} + t_y \sin\tau$. Then, if while on top of the tow it encounters yet another part of the tow, then $\rho = \rho_{mandrel} + 2t_y/\cos\tau$ and $z = z_{mandrel} + 2t_y \sin\tau$.

(a) *(b)*

Figure 5.9: Representation of tow overlap

This is the basic principle of how the code deals with fibre overlap and while it is still a simplistic approach, it is a start in a process which still needs to be refined.

In appendix A.1, it can be viewed that the last section of the Pathwind code is solely dedicated to this method of performing fibre overlap.

Chapter 6

Results and Discussion

In this chapter four virtual tests are presented (with different characteristics and input parameters) and the experimental validation of one of these tests (Test 1). The virtual tests are simulations made using the Pathwind code and the presented results are all the relevant outputs of this code.

The input parameters are all design specifications by the process engineer or by the machine operator and the outputs are the basis for the CNC code that will be generated to control the machine axes.

The tests shown in this chapter were deliberately chosen as they all have reasonable input parameters that can be applied in INEGI's filament winding machine. Similar pieces are regularly produced by INEGI and so, these tests were deemed interesting for future reference. Of the four tests, Test 1 and Test 3 have cylindrical mandrels while Test 2 and Test 4 have conical mandrels.

6.1 Virtual Testing

In all the tests presented here, a λ parameter of 0.14 was chosen as this is a generally accepted value for the winding of wet and dry fibre [5] on a metallic mandrel. The process used in INEGI is wet winding and therefore this permits the validation of the results.

6.1.1 Test 1 - Cylindrical mandrel

In this test, the physical limitations of filament winding with typical winding parameters are tested on an existing cylindrical mandrel.

Input parameters:
Tow width: b=6mm
Mandrel radius: r=75mm
Coordinate of left end of mandrel: zim=-500mm
Coordinate of the left end of the geodesic winding area: zi=0mm
Coordinate of the right end of the geodesic winding area: zfg=300mm

Coordinate of right end of mandrel: zfm=1000mm
Winding angle: α=45°
λ parameter: λ=0.14

For these input parameters, the following results were obtained:
z_{min}=-222.99mm
z_{max}=522,99mm
Part length=745.98mm
N_c=56
$L_{f-cycle}$=3322.7mm
$L_{f-total}$=186.17m

Figure 6.1: Parameter variation along one cycle

The variations of the parameters indicated in figure 6.1 are according to what is expected from the winding of a cylindrical piece. α varies between its initial value α_i and $\pi - \alpha_i$, turning around when it is equal to $\pi/2$ as is to be expected. Both ϕ and L_f keep increasing in a manner that is to be expected along one cycle.

Figure 6.2: Tow trajectory along one cycle

Figure 6.2 shows the first cycle ending right next to where it began, making it possible to completely cover the mandrel in this way.

Figure 6.3: Tow paths along the mandrel for all the cycles in one layer

It can be observed in figure 6.3 that the 56 cycles cover the mandrel surface in a uniform pattern.

Figure 6.4: Machine axes' movements along one cycle

The machine axes' movements shown in figure 6.4 are according to what is to be expected and was presented in chapter 4 for the winding of a cylindrical piece.

Figure 6.5: Machine axes' velocities along one cycle

It appears that there are some punctual inconsistencies in the velocities of axes A and X. Whether these inconsistencies are only from the velocity determination calculated for the graphs or if they actually translated into the CNC code and machine control remains to be ascertained. There were some aggressive accelerations in the production of the piece as is discussed in Chapter 6.2 and they might have been due to these inconsistencies.

Figure 6.6: Completely wound piece with tow overlap

Figure 6.7: Completely wound piece with tow overlap in detail

The fibre overlap performed by the Pathwind code appears to show some irregularities in the dwell area of the turn arounds. It is likely due to the fact that the dwell rotation, the tow thickness and the number of cycles were all high.

6.1.2 Test 2 - Conical mandrel

In this test, the physical limitations of filament winding with typical winding parameters are tested on a conical mandrel.

Input parameters:
Tow width: b=5mm
Mandrel radius on the left end: rim=30mm
Mandrel radius on the right end: rfm=50mm
Coordinate of left end of mandrel: zim=0mm
Coordinate of the left end of the geodesic winding area: zi=100mm

Coordinate of the right end of the geodesic winding area: zfg=300mm
Coordinate of right end of mandrel: zfm=600mm
Winding angle: α=75°
λ parameter: λ=0.14

For these input parameters, the following results were obtained:
z_{min}=88.7mm
z_{max}=399.1mm
Part length=310.4mm
N_c=30
$L_{f-cycle}$=1766.3mm
$L_{f-total}$=52.99m

Figure 6.8: Parameter variation along one cycle

The variation of parameters indicated in figure 6.8 is according to what is expected from the winding of a conical piece. α varies between its initial value α_i and decreases as the radius increases until it reaches the turn around area and it starts to increase until it reaches $\pi/2$ as is to be expected. Both ϕ and L_f keep increasing in a manner that is to be expected along one cycle.

Figure 6.9: Tow trajectory along one cycle

Figure 6.9 shows the first cycle ending right next to where it began, making it possible to completely cover the mandrel in this way.

Figure 6.10: Tow paths along the mandrel for all the cycles in one layer

It can be observed in figure 6.10 that the 30 cycles cover the mandrel surface in a uniform pattern.

Figure 6.11: Machine axes' movements along one cycle

The machine axes' movements for axes A, B and X shown in figure 6.11 are according to what is to be expected for the winding of a conical piece. However, the movement of axis Y has some irregularities throughout the cycle and even though the irregularities are small in value, they might alter the desired fibre path on the mandrel. I believe that the reason for these increases and decreases in axis Y are related to the numerical way that the distance Y is calculated (incrementally until the fixed distance to the mandrel is achieved).

Figure 6.12: Machine axes' velocities along one cycle

It now appears that there are many inconsistencies in the velocities of all axes. Whether these inconsistencies are only from the velocity determination calculated for the graphs or if they actually translated into the CNC code and machine control remains to be ascertained.

Figure 6.13: Completely wound piece with tow overlap

The irregularities in the fibre overlap seen in Test 1 are also present in this test and are likely due to the same causes. There are also some thickness increases in all the sections where there are fibre crossings and they are also probably due to the same causes.

6.1.3 Test 3 - Cylindrical mandrel

In this test, the physical limitations of filament winding with typical winding parameters are tested on a cylindrical mandrel.

Input parameters:
Tow width: b=10mm
Mandrel radius: r=15mm
Coordinate of left end of mandrel: zim=-500mm
Coordinate of the left end of the geodesic winding area: zi=0mm
Coordinate of the right end of the geodesic winding area: zfg=300mm
Coordinate of right end of mandrel: zfm=1000mm
Winding angle: α=20°
λ parameter: λ=0.14

For these input parameters, the following results were obtained:
z_{min}=-208.77mm
z_{max}=508.77mm
Part length=717.54mm
N_c=9
$L_{f-cycle}$=1884.5mm
$L_{f-total}$=16.96m

Figure 6.14: Parameter variation along one cycle

As was seen in Test 1, the variations of the parameters indicated in figure 6.14 are according to what is expected from the winding of a cylindrical piece and present the same characteristics as Test 1.

Figure 6.15: Tow trajectory along one cycle

As was previously seen in Test 1 and Test 2, figure 6.15 shows the first cycle ending right next to where it began, making it possible to completely cover the mandrel in this way.

Figure 6.16: Tow paths along the mandrel for all the cycles in one layer

It can be observed in figure 6.16 that the 9 cycles cover the mandrel surface in a uniform pattern.

Figure 6.17: Machine axes' movements along one cycle

The machine axes' movements shown in figure 6.17 are according to what is to be expected from the winding of a cylindrical piece and have the same general movements as those in Test 1 (the axes movement for all cylinders are similar).

Figure 6.18: Machine axes' velocities along one cycle

It appears that there are some punctual inconsistencies in the velocities of axes A, B and X. The inconsistencies in this test are more pronounced than previously seen and show some jumps in the velocities of those three axes. Whether these inconsistencies are only from the velocity determination calculated for the graphs or if they actually translated into the CNC code and machine control remains to be determined.

Figure 6.19: Completely wound piece with tow overlap

Figure 6.20: Completely wound piece with tow overlap - detail

The fibre overlap seen in figures 6.19 and 6.20 seem to represent reality better than the previous tests. The thickness seems to be correct throughout the tube and the overlaps and crossings can all be clearly observed. This appears to be due to the lower number of cycles wound which produce less thick and more consistent overlaps. However, there still appear some wavy areas in the turn around areas.

6.1.4 Test 4 - Conical mandrel

In this test, the physical limitations of filament winding with typical winding parameters are tested on a conical mandrel.

Input parameters:
Tow width: b=15mm
Mandrel radius on the left end: rim=15mm
Mandrel radius on the right end: rfm=60mm
Coordinate of left end of mandrel: zim=-500mm
Coordinate of the left end of the geodesic winding area: zi=0mm
Coordinate of the right end of the geodesic winding area: zfg=300mm
Coordinate of right end of mandrel: zfm=1500mm
Winding angle: α=25°
λ parameter: λ=0.14

For these input parameters, the following results were obtained:
z_{min}=-307.3113mm
z_{max}=1195.3mm
Part length=1502.6226mm
N_c=14
$L_{f-cycle}$=4167.4mm
$L_{f-total}$=58343.6mm=58.3436m

Figure 6.21: Parameter variation along one cycle

The variation of parameters indicated in figure 6.21 is according to what is expected from the winding of a conical piece and what is seen in Test 2. α varies between its initial value α_i and decreases as the radius increases until it reaches the turn around area and it starts its long increase until it reaches $\pi/2$ as is to be expected of this winding due to its low winding angle. Both ϕ and L_f keep increasing in a manner that is to be expected along one cycle.

Figure 6.22: Tow trajectory along one cycle

Figure 6.22 shows the first cycle ending right next to where it began, making it possible to completely cover the mandrel in this way.

Figure 6.23: Tow paths along the mandrel for all the cycles in one layer

It can be observed in figure 6.23 that the 14 cycles cover the mandrel surface in a uniform pattern.

Figure 6.24: Machine axes' movements along one cycle

The machine axes' movements for axes A, B and X shown in figure 6.24 are according to what is to be expected for the winding of a conical piece and was already seen in Test 2. However, like Test 2, the movement of axis Y has some irregularities throughout the cycle probably due to the same cause.

Figure 6.25: Machine axes' velocities along one cycle

The velocities of all the axes once again have many spikes and inconsistencies. It seems that the axes movement for conical mandrels have many inconsistencies.

Figure 6.26: Tow overlap with one cycle wound

Figure 6.27: Completely wound piece with tow overlap

The fibre overlap in this test seems to have worked better than in other cases but there are still some thickness irregularities in the turn around areas.

6.2 Experimental validation

As was said before, the experimental validation was made to the same specifications as Virtual Test 1. The part was produced using the Main Program File and the Sub Program File (CNC code) that were an output of the Pathwind Matlab^{TM} code.

Input parameters:
Tow width: b=6mm
Mandrel radius: r=75mm
Coordinate of left end of mandrel: zim=-500mm
Coordinate of the left end of the geodesic winding area: zi=0mm
Coordinate of the right end of the geodesic winding area: zfg=300mm
Coordinate of right end of mandrel: zfm=1000mm
Winding angle: α=45°
λ parameter: λ=0.14

For these input parameters, the following results were obtained:
z_{min}=-222.99mm
z_{max}=522,99mm
Part length=745.98mm
N_c=56
$L_{f-cycle}$=3322.7mm
$L_{f-total}$=186.07m

To produce the whole part, the software determined that for a tow width of 6mm, 56 cycles were necessary to completely cover the mandrel. After the Main Program File was executed, the machine positioned itself in the starting position and called the Sub Program File 56 times in succession.

Dry fibre winding was used in the production of the part. This means that no thermosetting plastic matrix was used and, therefore, the part was not permanent (it had to be unwound after completion).

During the process itself, the machine axes moved according to what was expected and normally happens when other software is employed to create the CNC code. There were aggressive accelerations in some areas but nothing that does not normally happen. A careful study should be made into the generation of the feed in order to limit these bursts of acceleration/deceleration.

(a)

(b)

Figure 6.28: Winding process in the geodesic area 6.28a and in the turn around area 6.28b

During the winding, a slight gap can be observed between the fibres of two consecutive paths in figure 6.28a.

Figure 6.29: Completely wound piece

(a)

(b)

Figure 6.30: Size of the produced part 6.30a and of the geodesic section 6.30b

From figure 6.30a we can observe that the total length of the produced part was 740mm which is very close to the predicted 745.98mm. The 5.98mm deviation is probably due to a slight fibre slippage as the tension of the tow after the turn around exerts a force in the direction of the middle of the part.

From figure 6.30b we can observe that the 300mm for the geodesic area seem to have produced a completely geodesic winding.

(a)

(b)

Figure 6.31: Tow width 6.30a and angle measurement in the geodesic winding area 6.30b

It is possible to observe in figure 6.31a that the tow width was not the 6mm which was input in the Pathwind code. The actual tow width was 5.15mm and this is the reason that we can observe a slight gap between the tows of consecutive cycles. Despite this slight gap, total coverage of the mandrel was almost obtained. By observing figure 6.29, we can observe that the mandrel can not be easily viewed - the fibre covers it.

From figure 6.31b it is possible to observe that the winding angle in the geodesic area of the part is exactly 45º throughout the geodesic zone as was requested. This means that there was likely no fibre slippage in the geodesic area of the part and if there was actually fibre slippage in the process, it was likely contained in the turn around areas.

Overall, the produced part was very similar to the design specifications and the process itself and axes movements behaved as expected. Further experimental validation is required but this first validation is enough to prove that the concepts and models presented throughout the thesis and implemented in the Pathwind code are credible.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The main objective of this work was to study, understand and evaluate the different types of fibre trajectories that can be made along the surface of an axis-symmetric mandrel and implement them in a real filament winding machine.

Models were presented to describe geodesic paths along cylindrical and conical surfaces. These models were successfully implemented and validated with virtual tests and for the case of cylinders, they were validated experimentally.

The description of non-geodesic paths that still guarantee fibre stability was also presented and implemented in the Pathwind code. The successful production of a filament wound part with non-geodesic trajectories and no significant observation of fibre slippage indicates that the presented models work. Nevertheless, more validation is still required.

A method for creating different winding patterns in cylinders and cones is proposed and implemented and the fact that the virtual tests and the experimental validation show complete mandrel coverage seem to suggest that the method works. A method presented by Johansen et al. [11] which serves to determine which patterns are possible for different winding parameters is also discussed and implemented and can be viewed in A.3.

A study of filament winding machine axes movements is made and a method of creating CNC code that can be used by the machine present at INEGI's facilities is presented. A part is then produced using this code. The fact that the produced part had the solicited geometry indicates that the axes movements were accurate. Some aggressive accelerations and decelerations were observed and although this is also observed when using machine control programs form other CAD/CAM software, further study needs to be made in order to improve axis movement and manufacturing speed. It was also observed in chapter 6 that there appear to be many inconsistencies in the velocity determination of the different axes. This is especially evident in the production in conical mandrels.

Different ways to describe the real tow path along a filament wound part are

made. Different cross-sections for the tow and a way to describe fibre overlap are presented and implemented. The fibre overlap shown as an output of the Pathwind code demonstrated inconsistencies such as excessive thickness in some areas and a few ups and downs that should not be present. Further study needs to be done into the real trajectories of fibres as it is important to understand and model the stresses caused by fibre overlap.

Overall, the automatic generation of tow trajectories on a mandrel surface for specific design inputs by the Pathwind code works well. It was shown that Pathwind is a tool that implements the presented models and helped validate them to some degree. Further work still needs to be done to improve and validate this tool but it was shown that it is already possible to produce different cylindrical and conical parts using the Pathwind code.

7.2 Future Work

The work presented throughout this document serves as the basis for a filament winding simulation tool. However, there is still much work to be done.

First of all, experimental validation of a multitude of different parts with varying geometrical and physical parameters needs to be conducted. The experimental validation shown in Chapter 6.2 serves as a proof of concept but does not completely validate the presented models nor their full potential and physical limits.

The Pathwind code serves as a tool to produce cylindrical and conical parts but it still falls short of the capabilities and applications of the available CAD/CAM software. It is therefore suggested that the following improvements and additions be made to the code:

- enable the input of different mandrel geometries including non axis-symmetric ones;
- produce a FEA mesh output compatible and able to be transferred into FEA platforms;
- make a graphic user interface for a user friendly interface;
- produce a graphic display of the process kinematics;
- improve the machine output, the manufacturing speed and optimize the movement of all the axes;
- new models of determining the real tow trajectories and cross sections need to be studied and implemented.

If the above improvements are made to the Pathwind code and validated experimentally, it is my opinion that the code would be equal in quality to the filament winding simulation software presently available. Therefore, a deep control of the process kinematics would be granted and the filament winding group's capabilities relevantly improved.

Bibliography

[1] Alting L. Todd RH, Allen DK. Fundamental Principles of Manufacturing Processes. New York: Industrial Press Inc. 1994.

[2] http://www.jp-petit.org/science/ingeph_f/Dessins/a011.gif.

[3] Sotiris Koussios. Filament Winding: A Unified Approach. 2004.

[4] E. Vargas Rojas, D. Chapelle, D. Perreux, B. Delobelle, and F. Thiebaud. Unified approach of filament winding applied to complex shape mandrels. Composite Structures, 116:805–813, 2014.

[5] CADWIND User manual - Process simulation for filament winding, Material S. A. 2007.

[6] Mahmood Ansar, Wang Xinwei, and Zhou Chouwei. Modeling strategies of 3D woven composites: A review. Composite Structures, 93(8):1947–1963, 2011.

[7] P. Potluri and a. Manan. Mechanics of non-orthogonally interlaced textile composites. Composites Part A: Applied Science and Manufacturing, 38(4):1216–1226, 2007.

[8] Hugo Faria. Analytical and numerical modelling of the filament winding process. 2007.

[9] L Ainsworth. The state of fillament winding. (March), 1971.

[10] Donald V. Rosato and Dominick V. Rosatot. Reinforced Plastics Handbook Third edition. page 1113, 2004.

[11] B S Johansen, Aa Lystrup, and M T Jensen. CADPATH: A complete program for the CAD-, CAE- and CAM-winding of advanced fibre composites. Journal of Materials Processing Technology, 300(3-4):194–200, 1998.

[12] Lei Zu. Design and Optimization of Filament Wound Composite Pressure Vessels. 2012.

[13] Johan Scholliers and Hendrik Van Brussel. Computer-integrated filament winding: computer-integrated design, robotic filament winding and robotic quality control. Composites Manufacturing, 5(1):15–23, 1994.

[14] Haisheng Li, Youdong Liang, and Hujun Bao. CAM system for filament winding on elbows. Journal of Materials Processing Technology, 161:491–496, 2005.

[15] S. Koussios, O. K. Bergsma, and A. Beukers. Filament winding. Part 1: Determination of the wound body related parameters. Composites Part A: Applied Science and Manufacturing, 35:181–195, 2004.

[16] Andrew Russell Forsyth. Lectures on the differential geometry of curves and surfaces. pages xxiii, 525 p., 1912.

[17] Toponogov Rovenski. Differential geometry of curves and surfaces. 1976.

[18] Paul A. Blaga. Lectures on Classical Differential Geometry. Editura Risoprint. 2005.

[19] John Oprea. Differential Geometry and its Applications. 2007.

[20] S.P. Gold. Filament Winding Surface of Revolution, Technical Report SCL-DR-65-105, Sandia Corporation, Livermore Laboratory, Livermore, California. 1966.

[21] Cheol U. Kim, Ji H. Kang, Chang S. Hong, and Chun G. Kim. Optimal design of filament wound structures under internal pressure based on the semi-geodesic path algorithm. Composite Structures, 67(4):443–452, 2005.

[22] J. De Carvalho, M. Lossie, D. Vandepitte, and H. Van Brussel. Optimization of filament-wound parts based on non-geodesic winding. Composites Manufacturing, 6(2):79–84, 1995.

[23] Jae Sung Park, Chang Sun Hong, Chun Gon Kim, and Cheol Ung Kim. Analysis of filament wound composite structures considering the change of winding angles through the thickness direction. Composite Structures, 55(1):63–71, 2002.

[24] W Polini and L Sorrentino. Influence of winding speed and winding trajectory on tension in robotized filament winding of full section parts. Composites Science and Technology, 65(10):1574–1581, 2005.

[25] S. Koussios, O. K. Bergsma, and a. Beukers. Filament winding. Part 2: Generic kinematic model and its solutions. Composites Part A: Applied Science and Manufacturing, 35:197–212, 2004.

[26] S. Koussios, O.K. Bergsma, and a. Beukers. Filament winding: kinematics, collision control and process optimisation through application of dynamic programming. Composites Part A: Applied Science and Manufacturing, 37:2088–2104, 2006.

[27] N. Vernet and F. Trochu. Analysis and modeling of 3D Interlock fabric compaction behavior. Composites Part A: Applied Science and Manufacturing, 80:182–193, 2016.

[28] Byung Chul Kim, Paul M. Weaver, and Kevin Potter. Computer Aided Modelling of Variable Angle Tow Composites Manufactured by Continuous Tow Shearing. Composite Structures, 129:256–267, 2015.

Appendix A

Matlab Code

A.1 Pathwind

```
%Programa base

%% Inicialização
clear all; close all; clc;
addpath([pwd,'\functions']);
addpath([pwd,'\Resultados']);

%% Dados

%Mandril - caso conico, rfm>rim
rim=15; %raio inicial mandril - caso cilindro, assume rc=rim
rfm=15; %raio final mandril

zim=-500; %z inicio mandril
zfm=1000; %z final mandril

%Zona util do mandril
zig=0; %z inicio da zona de enrolamento geodésico
zi=0; %z inicio do enrolamento tem de ser igual a zig
zfg=300; %z final da zona de enrolamento geodésico

%Alfa inicial zona geodésica (em zi)
alfainicio=20; %alfa em graus. domínio: ]-90,90]

%Coeficiente de escorregamento
niu=0.14;
lambda1=niu;

%Incremento na curva
ds=2;
dsta=2; %ds no turnaround

%Propriedades do material
%largura de banda
b=10;

%number of rovings
NR=10;
%roving width (mm)
RW=b/NR;
%TEX value (g/km)
TEX=800;
%fibre volume content/ fibre mass content (%) - o que não se usa=0
fvc=0;
fmc=60;
%fibre density (g/cm3)
fibdens=1.77;
%resin density (g/cm3)
resdens=1.2;

%Número de paths
NP=6;

%Velocidade da máquina
vel=1.5; %multiplica pelo tempo necessário para cada passo
```

```
%Outputs
    nomeoutput='04.03.2016_rc75_b10_-500,0,300,1000,alfa20';    %nome da pasta e↙
ficheiros (sem espaços)
    filepath='C:\Users\Rodrigo\Dropbox\Tese\Programa';
    %Trajectórias
    gravartrajectorias='s'; %Se for para gravar, escrever 's' se nao, escrever outra↙
coisa

    %CNC
    cnc='s';
    gravarcnc='s';  %Se for para gravar, escrever 's' se nao, escrever outra coisa

%Distância da cabeça a ao mandril - distância fixa
D=50;

%Sections - sobreposições só funcionam com secções rectangulares
seccoes='rectangular'; %rectangular ou eliptical
 belip=1; %espessura medida da fibra no caso eliptico

%Sobreposições
sobre='sobreposiçao2'; %'sobreposição1' - 1 ciclo.    'sobreposição2' - todos os↙
ciclos   'sobreposição3' - matriz reorganizada ainda não funcional


%% Machine Parameters
%Reference
RefA=0;
RefX=0;
RefY=260; %distancia do eixo Y ao fim da cabeça
RefB=0;
RefZ=0;
RefC=0;

%Dimensions
Xmin=165.15; %mm
Xmax=2644.7; %mm
Ymin=-120; %mm
Ymax=311; %mm
Zmin=0; %mm
Zmax=364; %mm
Cmin=-50; %°
Cmax=50; %°
POew=50; %mm
POe=0; %mm
YawR=260; %mm

%Velocity
VA=150; %360°/min
VX=1; %m/s
VY=0.5; %m/s
VB=115; %360°/min
VZ=0.5; %m/s
VC=100; %360°/min
```

```
%Acceleration
AA=8; %360°/min
AX=4; %m/s
AY=30; %m/s
AB=60; %360°/min
AZ=1; %m/s
AC=45; %360°/min

%Resolution
RA=360; %inc/360°
RX=1; %inc/mm
RY=-1; %inc/mm
RB=360; %inc/360°
RZ=1; %inc/mm
RC=360; %inc/360°

%Processing time
minproc=0.03; %minimium processing time (s)

%Velocidades Máximas - inc/s
VAmax=abs(VA*RA/60);
VXmax=abs(VX*1000*RX);
VYmax=abs(VY*1000*RY);
VBmax=abs(VB*RB/60);
VZmax=abs(VZ*1000*RZ);
VCmax=abs(VC*RC/60);

%Acelerações máximas
AAmax=abs(AA*RA);
AXmax=abs(AX*60*1000*RX);
AYmax=abs(AY*60*1000*RY);
ABmax=abs(AB*RB);
AZmax=abs(AZ*60*1000*RZ);
ACmax=abs(AC*RC);

%% Cálculos iniciais
alfai=alfainicio*pi/180; %alfa em rad
lm=zfm-zim; %comprimento do mandril
lg=zfg-zig; %comprimento zona geodésica

dm=rfm-rim; %diferença de raio do mandril

rig=((zig*dm/lm)+rim-(dm*zim/lm)); %raio no inicio da zona geodésica
ri=((zi*dm/lm)+rim-(dm*zim/lm)); %raio na secção do inicio do enrolamento
rfg=((zfg*dm/lm)+rim-(dm*zim/lm)); %raio no final da zona geodésica

tau=atan(dm/lm);

zeroteta=(zfg+zig)/2; %em coordenadas esféricas, o zero fica a meio

rc=rim; %se for cilindro fica com o raio menor do mandril


%Padrão cilindro
if rig==rfg
```

```
    beff=b/cos(alfai);
    nreal=2*pi*rc/beff;
    n1=ceil(nreal);
end

% Espessura
fd=fmc*fibdens+(1-fmc)*resdens;
esp=(fmc*(TEX/100000)/((RW/10)*(fd-(1-fmc)*resdens)))*10;




%↙
----------------------------------------------------------------------------------------------↙
-----------------------
%↙
----------------------------------------------------------------------------------------------↙
-----------------------
%↙
----------------------------------------------------------------------------------------------↙
-----------------------
%-----------------------------------------------↙
Mandril-------------------------------------------------------


%% Mandril

% Gerar Mandril
if rim==rfm
    tipomandril='cilindro'     %cilindro ou cone
else
    tipomandril='cone'
end

switch lower(tipomandril)

    case {'cilindro'}
        tc = 0:pi/20:2*pi;
        k=1;
        X=zeros(2,size(tc,2));
        Y=zeros(2,size(tc,2));
        Z=zeros(2,size(tc,2));
        for tc = 0:pi/20:2*pi
            X(1,k)=rc*cos(tc);
            X(2,k)=rc*cos(tc);
            Y(1,k)=rc*sin(tc);
            Y(2,k)=rc*sin(tc);
            Z(1,k)=zim;
            Z(2,k)=zfm;
            k=k+1;
        end

    case {'cone'}
        %Mandril
```

```
        tc = 0:pi/20:2*pi;
        k=1;
        X=zeros(2,size(tc,2));
        Y=zeros(2,size(tc,2));
        Z=zeros(2,size(tc,2));
        for tc = 0:pi/20:2*pi
            X(1,k)=rim*cos(tc);
            X(2,k)=rfm*cos(tc);
            Y(1,k)=rim*sin(tc);
            Y(2,k)=rfm*sin(tc);
            Z(1,k)=zim;
            Z(2,k)=zfm;
            k=k+1;
        end

    otherwise
          %Mandril
        tc = 0:pi/20:2*pi;
        k=1;
        X=zeros(2,size(tc,2));
        Y=zeros(2,size(tc,2));
        Z=zeros(2,size(tc,2));
        for tc = 0:pi/20:2*pi
            X(1,k)=rim*cos(tc);
            X(2,k)=rim*cos(tc);
            Y(1,k)=rim*sin(tc);
            Y(2,k)=rim*sin(tc);
            Z(1,k)=zig;
            Z(2,k)=zig;
            k=k+1;
        end

end



%Superfície parametrizada para coordenadas esféicas ou polares
if rim==rfm
    coordenadas='esfericas'
else
    coordenadas='polares'
end

switch lower(coordenadas)
    case {'polares'}
        %% Superfície parametrizada do mandril em coordenadas polares

        %syms u v a(u) b(u) t
        %S=[a*cos(v),a*sin(v),b]; %S(u,v)=general shell of revolution

        % S=[cos(v)*cos(u), cos(v)*sin(u), sin(v)]

        %Cone
        syms fi ro
        %ro=((z*dm/lm)+rim-(dm*zim/lm));
```

```
        z=ro*lm/dm-rim*lm/dm+zim;
        S=[ro*cos(fi), ro*sin(fi), ro*lm/dm-rim*lm/dm+zim];

        %% Propriedades da superfície - coordenadas polares

        %[chr,gg1,gg2,e,g,ev,eu,gv,gu]=Christoffel_funcao_1(S,u,v,ro)
        [chr,gg1,gg2,e,g,ero,efi,gro,gfi]=Christoffel_funcao_1(S,fi,ro,z)
        zlinha=diff(z,ro)
        z2linha=diff(zlinha,ro)


    case {'esfericas'}
        %% Superfície parametrizada do mandril em coordenadas esfericas

        %Cilindro
        syms teta fi ro
        ro=rc/sin(teta);
        Se=[ro*sin(teta)*cos(fi), ro*sin(teta)*sin(fi),ro*cos(teta)];

        %% Propriedades da superfície - coordenadas esfericas
        %[chr,gg1,gg2,e,g,ev,eu,gv,gu]=Christoffel_funcao_1(S,u,v,ro)
        [chr,gg1,gg2,e,g,eteta,efi,gteta,gfi]=Christoffel_funcao_1(Se,fi,teta,ro);


end



%↙
----------------------------------------------------------------------------------------↙
-----------------------
%↙
----------------------------------------------------------------------------------------↙
-----------------------
%↙
----------------------------------------------------------------------------------------↙
-----------------------
%-----------------------------------------------------↙
Trajectória1-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
        c1=sin(alfai)*ri; %começando a trajetoria em ri

        z=[];
        fi=[];
        fi1=[];
        ro=[];
        alfa1=[];
        lf=[];

        alfa1(1,1)=alfai;
        z(1,1)=zi;
        ro(1,1)=((z(1,1)*dm/lm)+rim-(dm*zim/lm));
        fi1(1,1)=(1/sin(tau))*(asin(c1/rfg)-asin(c1/ro(1,1)));
        lf(1,1)=0;

        i=1;

        while z(1,i)<zfg
            i=i+1;
            z(1,i)=z(1,i-1)+ds*cos(alfa1(1,i-1))*cos(tau);
            ro(1,i)=((z(1,i)*dm/lm)+rim-(dm*zim/lm));

            fi1(1,i)=(1/sin(tau))*(asin(c1/rfg)-asin(c1/ro(1,i)));

            %alfa1(1,i)=atan((ro(1,i)*(fi1(1,i)-fi1(1,i-1)))/((z(1,i)-z(1,i-1))/cos↙
(tau)));
            alfa1(1,i)=asin(c1/ro(1,i));
            dalfadro(1,i)=(alfa1(1,i)-alfa1(1,i-1))/(ro(1,i)-ro(1,i-1));
            lf(1,i)=lf(1,i-1)+ds;

        end

        dalfadro(1,1)=dalfadro(1,2);

        if z(1,i)~=zfg
            z(1,i)=zfg;
            ro(1,i)=((z(1,i)*dm/lm)+rim-(dm*zim/lm));
            fi1(1,i)=(1/sin(tau))*(asin(c1/rfg)-asin(c1/ro(1,i)));
            %alfa1(1,i)=atan((ro(1,i)*(fi1(1,i)-fi1(1,i-1)))/((z(1,i)-z(1,i-1))/cos↙
(tau)));
            alfa1(1,i)=asin(c1/ro(1,i));
            lf(1,i)=lf(1,i-1)+(z(1,i)-z(1,i-1))/(cos(tau)*cos(alfa1(1,i-1)));
            dalfadro(1,i)=(alfa1(1,i)-alfa1(1,i-1))/(ro(1,i)-ro(1,i-1));
        end

        %alfa=zeros(1,size(alfa1,2));
        alfa=alfa1;

        %passo extra
        %i=i+1;
        %zpe=z(1,i-1)+0.00001*cos(alfa1(1,i-1))*cos(tau);
        %rope=((zpe*dm/lm)+rim-(dm*zim/lm));
        %fi1pe=(1/sin(tau))*(asin(c1/rfg)-asin(c1/rope));
        %alfa1(1,i)=atan((rope*(fi1pe-fi1(1,i-1)))/((zpe-z(1,i-1))/cos(tau)));
```

```
        %for j=1:size(alfa,2)
        %    alfa(1,j)=alfa1(1,j+1);
        %end

        fi(1,:)=fi1(1,:)-fi1(1,1);

        %Cartesianas
        x=ro.*cos(fi);
        y=ro.*sin(fi);


    case {'cones integral'}    %coordenadas polares
        %% Cálculos iniciais

        fi=[];
        lf=[];
        alfa=[];t
        dalfadro=[];
        ro=[];
        z=[];

        alfa(1,1)=alfai;
        fi(1,1)=0;
        lf(1,1)=0;

        z(1,1)=zi;
        ro(1,1)=((z(1,1)*dm/lm)+rim-(dm*zim/lm));

        n=1;
        j=1;

        %% Cálculo de fi, alfa e Lf
        while z(1,n)<zfg

           ro(1,n+1)=ro(1,n)+ds*cos(alfa(1,n))*sin(tau);
           z(1,n+1)=ro(1,n+1)*lm/dm-rim*lm/dm+zim;

           dalfadro0=-0.5*ero*tan(alfa(1,n))/gg1(1,1);
           dalfadro1=-0.5*ero*tan(alfa(1,n))/gg1(1,1);

           dalfadro0=double(subs(dalfadro0,'ro',ro(1,n)));
           dalfadro0=double(subs(dalfadro0,'fi',fi(1,n)));

           dalfadro1=double(subs(dalfadro1,'ro',ro(1,n+1)));
           dalfadro1=double(subs(dalfadro1,'fi',fi(1,n)));

           dalfadro(1,n)=dalfadro0;

           alfa(1,n+1)=alfa(1,n)+(ro(1,n+1)-ro(1,n))*(dalfadro0+dalfadro1)/2;

           E1=vpa(subs(gg1(1,1),'ro',ro(1,n)));
           E1=vpa(subs(E1,'fi',fi(1,n)));
           G1=vpa(subs(gg1(2,2),'ro',ro(1,n)));
           G1=vpa(subs(G1,'fi',fi(1,n)));
```

```
            E2=vpa(subs(gg1(1,1),'ro',ro(1,n+1)));
            E2=vpa(subs(E1,'fi',fi(1,n)));
            G2=vpa(subs(gg1(2,2),'ro',ro(1,n+1)));
            G2=vpa(subs(G1,'fi',fi(1,n)));

            n=n+1;

            fi(1,n)=fi(1,n-1)+0.5*(ro(1,n)-ro(1,n-1))*(tan(alfa(1,n-1))*sqrt(G1/E1)+tan↙
(alfa(1,n))*sqrt(G2/E2)); %verificar contas
            lf(1,n)=lf(1,n-1)+0.5*(ro(1,n)-ro(1,n-1))*(sqrt(G1)/cos(alfa(1,n-1))+sqrt↙
(G2)/cos(alfa(1,n))); %verificar contas

        end

        if z(1,n)~=zfg
            z(1,n)=zfg;
            ro(1,n)=((z(1,n)*dm/lm)+rim-(dm*zim/lm));

            dalfadro0=-0.5*ero*tan(alfa(1,n))/gg1(1,1);
            dalfadro1=-0.5*ero*tan(alfa(1,n))/gg1(1,1);
            dalfadro0=double(subs(dalfadro0,'ro',ro(1,n-1)));
            dalfadro1=double(subs(dalfadro1,'ro',ro(1,n)));
            dalfadro(1,n)=dalfadro0;
            alfa(1,n)=alfa(1,n-1)+(ro(1,n)-ro(1,n-1))*(dalfadro0+dalfadro1)/2;

            E1=vpa(subs(gg1(1,1),'ro',ro(1,n-1)));
            E1=vpa(subs(E1,'fi',fi(1,n-1)));
            G1=vpa(subs(gg1(2,2),'ro',ro(1,n-1)));
            G1=vpa(subs(G1,'fi',fi(1,n-1)));

            E2=vpa(subs(gg1(1,1),'ro',ro(1,n)));
            E2=vpa(subs(E1,'fi',fi(1,n-1)));
            G2=vpa(subs(gg1(2,2),'ro',ro(1,n)));
            G2=vpa(subs(G1,'fi',fi(1,n-1)));

            fi(1,n)=fi(1,n-1)+0.5*(ro(1,n)-ro(1,n-1))*(tan(alfa(1,n-1))*sqrt(G1/E1)+tan↙
(alfa(1,n))*sqrt(G2/E2)); %verificar contas
            lf(1,n)=lf(1,n-1)+0.5*(ro(1,n)-ro(1,n-1))*(sqrt(G1)/cos(alfa(1,n-1))+sqrt↙
(G2)/cos(alfa(1,n))); %verificar contas

        end

        %% Cartesianas
        x=ro.*cos(fi);
        y=ro.*sin(fi);


    case {'cilindros analitica'}
        %% Trajectória geodésica para cilindros - hélices
        % Helice e superficie
        dz=ds*cos(alfai);
        z=[];
        z(1,:)=zi:dz:zfg;
```

```
        if z(1,size(z,2))~=zfg
            z(1,size(z,2)+1)=zfg;
        end

        c=rc/tan(alfai);

        fi=z/c;

        x=rc*cos(fi);
        y=rc*sin(fi);


        %para dar continuidade estes dados são necessários no TA
        alfa=ones(1,size(x,2)).*alfai;

        lf=sqrt((fi.*rc).^2+(z-zig).^2); %fazer contas outra vez

        tetai=atan(rc/(lg/2));
        tetaf=pi-tetai;
        dalfadteta=zeros(1,size(x,2));

        teta=atan(rc./(zeroteta-z));
        tetap=[];
        tetap(1,1)=teta(1,1);

        for n=1:size(teta,2)

            if teta(1,n)<0
                tetap(1,n)=teta(1,n)+pi;
            else
                tetap(1,n)=teta(1,n);
            end
        end

        if z(1,size(z,2))~=zfg
            z(1,size(z,2)+1)=zfg;
        end



    case {'cilindros integral'}    %coordenadas esfericas
        %% Cálculos iniciais

        tetai=atan(rc/(lg/2-zi));
        tetaf=pi-atan(rc/(lg/2));

        fi=[];
        lf=[];

        alfa=[];
        dalfadteta=[];
        alfa(1,1)=alfai;

        teta=[];
        teta1=tetai;
```

```
        teta(1,1)=teta1;

        tetap=[];
        tetap(1,1)=tetai;

        fi(1,1)=0;
        lf(1,1)=0;

        n=1;
        j=1;

        %% Cálculo de fi, alfa e Lf
        while tetap(1,n)<tetaf

            %Gerar teta para ds constante
            j=n+1;
            teta(1,j)=teta1;
            teta2=atan(1/(-(ds*cos(alfa(1,n)))/rc+1/tan(teta1)));
            teta1=teta2;

            if teta(1,j)<0
                tetap(1,j)=teta(1,j)+pi;
            else
                tetap(1,j)=teta(1,j);
            end

           dalfadteta0=0.5*eteta*tan(alfa(1,n))/gg1(1,1);
           dalfadteta1=0.5*eteta*tan(alfa(1,n))/gg1(1,1);

           dalfadteta0=double(subs(dalfadteta0,'teta',tetap(1,n)));

           dalfadteta1=double(subs(dalfadteta1,'teta',tetap(1,n+1)));

           dalfadteta(1,n)=dalfadteta0;

           alfa(1,n+1)=alfa(1,n)+(tetap(1,n+1)-tetap(1,n))*(dalfadteta0+dalfadteta1)↙
/2;

           E1=vpa(subs(gg1(1,1),'teta',tetap(1,n)));
           E1=vpa(subs(E1,'fi',fi(1,n)));
           G1=vpa(subs(gg1(2,2),'teta',tetap(1,n)));
           G1=vpa(subs(G1,'fi',fi(1,n)));

           E2=vpa(subs(gg1(1,1),'teta',tetap(1,n+1)));
           E2=vpa(subs(E1,'fi',fi(1,n)));
           G2=vpa(subs(gg1(2,2),'teta',tetap(1,n+1)));
           G2=vpa(subs(G1,'fi',fi(1,n)));

           n=n+1;

           fi(1,n)=fi(1,n-1)+0.5*(tetap(1,n)-tetap(1,n-1))*(tan(alfa(1,n-1))*sqrt↙
(G1/E1)+tan(alfa(1,n))*sqrt(G2/E2));
           lf(1,n)=lf(1,n-1)+0.5*(tetap(1,n)-tetap(1,n-1))*(sqrt(G1)/cos(alfa(1,n-1))↙
+sqrt(G2)/cos(alfa(1,n)));
```

```
        end

        %ultimo passo
        if tetap(1,n-1)~=tetaf
           tetap(1,n)=tetaf;
           dalfadteta0=0.5*eteta*tan(alfa(1,n))/gg1(1,1);
           dalfadteta1=0.5*eteta*tan(alfa(1,n))/gg1(1,1);

           dalfadteta0=double(subs(dalfadteta0,'teta',tetap(1,n)));

           dalfadteta1=double(subs(dalfadteta1,'teta',tetap(1,n+1)));

           dalfadteta(1,n)=dalfadteta0;

           alfa(1,n+1)=alfa(1,n)+(tetap(1,n+1)-tetap(1,n))*(dalfadteta0+dalfadteta1)↙
/2;

           E1=vpa(subs(gg1(1,1),'teta',tetap(1,n)));
           E1=vpa(subs(E1,'fi',fi(1,n)));
           G1=vpa(subs(gg1(2,2),'teta',tetap(1,n)));
           G1=vpa(subs(G1,'fi',fi(1,n)));

           E2=vpa(subs(gg1(1,1),'teta',tetap(1,n+1)));
           E2=vpa(subs(E1,'fi',fi(1,n)));
           G2=vpa(subs(gg1(2,2),'teta',tetap(1,n+1)));
           G2=vpa(subs(G1,'fi',fi(1,n)));

           n=n+1;

           fi(1,n)=fi(1,n-1)+0.5*(tetap(1,n)-tetap(1,n-1))*(tan(alfa(1,n-1))*sqrt↙
(G1/E1)+tan(alfa(1,n))*sqrt(G2/E2));
           lf(1,n)=lf(1,n-1)+0.5*(tetap(1,n)-tetap(1,n-1))*(sqrt(G1)/cos(alfa(1,n-1))↙
+sqrt(G2)/cos(alfa(1,n)));

        end

        %% Cartesianas
        ro=rc./sin(tetap);
        x=ro.*sin(tetap).*cos(fi);
        y=ro.*sin(tetap).*sin(fi);
        z=-ro.*cos(tetap)+zeroteta;


end
%%



%↙
-----------------------------------------------------------------------------------------↙
------------------------
%↙
-----------------------------------------------------------------------------------------↙
------------------------
```

```
%↙
-----------------------------------------------------------------------------------------↙
------------------------
%--------------------------------------------------↙
Turnaround1------------------------------------------------------
```

```
%% Turnaround 1

if rim==rfm
    turnaround='cilindro'
else
    turnaround='cone'
end

switch lower(turnaround)

    case {'cilindro'}

        %% Cálculos iniciais
        alfaita=alfa(1,size(alfa,2));

        teta1=tetaf;
        tetata=[];
        tetata(1,1)=teta1;
        tetatap=[];
        tetatap(1,1)=teta1;

        alfata=[];
        alfata(1,1)=alfaita;

        fita=[];
        fita(1,1)=fi(1,size(fi,2));

        lfta=[];
        lfta(1,1)=lf(1,size(lf,2));

        dalfadtetata=[];
        dalfadtetata(1,1)=dalfadteta(1,size(dalfadteta,2));

        n=1;
        j=1;

        %% Primeira metade
        while alfata(1,n)<pi/2

            %Gerar teta para ds constante
            j=j+1;
            tetata(1,j)=teta1;
            teta2=atan(1/(-(dsta*cos(alfata(1,n)))/rc+1/tan(teta1)));
            teta1=teta2;
```

```
            if tetata(1,j)<0
                tetatap(1,j)=tetata(1,j)+pi;
            else
                tetatap(1,j)=tetata(1,j);
            end

            %Cálculo das formas fundamentais em tetatap(1,n)
            E1=vpa(subs(gg1(1,1),'teta',tetatap(1,n)));
            E1=vpa(subs(E1,'fi',fita(1,n)));
            G1=vpa(subs(gg1(2,2),'teta',tetatap(1,n)));
            G1=vpa(subs(G1,'fi',fita(1,n)));

            L1=vpa(subs(gg2(1,1),'teta',tetatap(1,n)));
            L1=vpa(subs(L1,'fi',fita(1,n)));
            N1=vpa(subs(gg2(2,2),'teta',tetatap(1,n)));
            N1=vpa(subs(N1,'fi',fita(1,n)));

            %Cálculo de dalfadteta em tetatap(1,n)
            dalfadtetata0=0.5*eteta*tan(alfata(1,n))/gg1(1,1)+lambda1*((sqrt(G1)/cos↙
(alfata(1,n)))*(((cos(alfata(1,n))^2)*N1/G1)+((sin(alfata(1,n))^2)*L1/E1)));

            %Cálculo das formas fundamentais em tetatap(1,n+1)
            E2=vpa(subs(gg1(1,1),'teta',tetatap(1,n+1)));
            E2=vpa(subs(E2,'fi',fita(1,n)));
            G2=vpa(subs(gg1(2,2),'teta',tetatap(1,n+1)));
            G2=vpa(subs(G2,'fi',fita(1,n)));

            L2=vpa(subs(gg2(1,1),'teta',tetatap(1,n+1)));
            L2=vpa(subs(L2,'fi',fita(1,n)));
            N2=vpa(subs(gg2(2,2),'teta',tetatap(1,n+1)));
            N2=vpa(subs(N2,'fi',fita(1,n)));

            %Cálculo de dalfadteta em tetatap(1,n)
            dalfadtetata1=0.5*eteta*tan(alfata(1,n))/gg1(1,1)+lambda1*((sqrt(G2)/cos↙
(alfata(1,n)))*(((cos(alfata(1,n))^2)*N2/G2)+((sin(alfata(1,n))^2)*L2/E2)));

            dalfadtetata(1,n)=dalfadtetata0;

            %Cálculo de alfa
            alfata(1,n+1)=alfata(1,n)+(tetatap(1,n+1)-tetatap(1,n))*↙
(dalfadtetata0+dalfadtetata1)/2;
            %Cálculo de fi e Lf
            fita(1,n+1)=fita(1,n)+0.5*(tetatap(1,n+1)-tetatap(1,n))*(tan(alfata(1,n))↙
*sqrt(G1/E1)+tan(alfata(1,n+1))*sqrt(G2/E2));
            lfta(1,n+1)=lfta(1,n)+0.5*(tetatap(1,n+1)-tetatap(1,n))*(sqrt(G1)/cos↙
(alfata(1,n))+sqrt(G2)/cos(alfata(1,n+1)));

            n=n+1;

        end


        dalfadtetata(1,n)=dalfadtetata(1,n-1);
        alfata(1,n)=pi/2;
```

```
%% Dwell

nmeio=n;
alfatameio=alfata(1,n);
fitameio=fita(1,n);
dfita=dsta/rc;

deltafita=fita(1,n)-fita(1,1);

fitafin1=fita(1,n)+deltafita;
fitafin2=2*fitafin1;

p=1;
while fitafin2>p*2*pi
    p=p+1;
end

dif=p*2*pi-fitafin2;

if dif/2-pi/n1>=0
    fidwell=dif/2-pi/n1;
else
    fidwell=dif/2+pi/n1;
end

while fita(1,n)<fitameio+fidwell
    n=n+1;
    fita(1,n)=fita(1,n-1)+dfita;
    lfta(1,n)=lfta(1,n-1)+rc*dfita;
    alfata(1,n)=pi/2;
    dalfadtetata(1,n)=dalfadtetata(1,n-1);
    tetata(1,n)=tetata(1,n-1);
    tetatap(1,n)=tetatap(1,n-1);
end

fita(1,n)=fita(1,nmeio)+fidwell;
lfta(1,n)=lfta(1,n-1)+rc*(fita(1,n)-fita(1,n-1));

nfimrest=n;
teta1=tetatap(1,n);
j=n;


%% Segunda metade

parte2turnaround='simetrico'

switch lower(parte2turnaround)
    case {'integral'}

        while alfata(1,n)<(pi-alfaita)

            %Gerar teta para ds constante
            j=j+1;
            tetata(1,j)=teta1;
```

```
                    teta2=atan(1/(-(dsta*cos(alfata(1,n)))/rc+1/tan(teta1)));
                    teta1=teta2;

                    if tetata(1,j)<0
                        tetatap(1,j)=tetata(1,j)+pi;
                    else
                        tetatap(1,j)=tetata(1,j);
                    end

                    E1=vpa(subs(gg1(1,1),'teta',tetatap(1,n)));
                    E1=vpa(subs(E1,'fi',fita(1,n)));
                    G1=vpa(subs(gg1(2,2),'teta',tetatap(1,n)));
                    G1=vpa(subs(G1,'fi',fita(1,n)));

                    L1=vpa(subs(gg2(1,1),'teta',tetatap(1,n)));
                    L1=vpa(subs(L1,'fi',fita(1,n)));
                    N1=vpa(subs(gg2(2,2),'teta',tetatap(1,n)));
                    N1=vpa(subs(N1,'fi',fita(1,n)));

                    dalfadtetata0=0.5*eteta*tan(alfata(1,n))/gg1(1,1)+lambda1*((sqrt↙
(G1)/cos(alfata(1,n)))*(((cos(alfata(1,n))^2)*N1/G1)+((sin(alfata(1,n))^2)*L1/E1)));


                    E2=vpa(subs(gg1(1,1),'teta',tetatap(1,n+1)));
                    E2=vpa(subs(E2,'fi',fita(1,n)));
                    G2=vpa(subs(gg1(2,2),'teta',tetatap(1,n+1)));
                    G2=vpa(subs(G2,'fi',fita(1,n)));

                    L2=vpa(subs(gg2(1,1),'teta',tetatap(1,n+1)));
                    L2=vpa(subs(L2,'fi',fita(1,n)));
                    N2=vpa(subs(gg2(2,2),'teta',tetatap(1,n+1)));
                    N2=vpa(subs(N2,'fi',fita(1,n)));

                    dalfadtetata1=0.5*eteta*tan(alfata(1,n))/gg1(1,1)+lambda1*((sqrt↙
(G2)/cos(alfata(1,n)))*(((cos(alfata(1,n))^2)*N2/G2)+((sin(alfata(1,n))^2)*L2/E2)));

                    dalfadtetata(1,n)=dalfadtetata0;

                    alfata(1,n+1)=alfata(1,n)+(tetatap(1,n+1)-tetatap(1,n))*↙
(dalfadtetata0+dalfadtetata1)/2;

                    fita(1,n+1)=fita(1,n)+0.5*(tetatap(1,n+1)-tetatap(1,n))*(tan↙
(alfata(1,n))*sqrt(G1/E1)+tan(alfata(1,n+1))*sqrt(G2/E2));
                    lfta(1,n+1)=lfta(1,n)+0.5*(tetatap(1,n+1)-tetatap(1,n))*(sqrt(G1)↙
/cos(alfata(1,n))+sqrt(G2)/cos(alfata(1,n+1)));

                    n=n+1;

                end

            case {'simetrico'}

                for i=0:nmeio-1
                    n=n+1;
                    fita(1,nfimrest+i)=fita(1,nfimrest)+fita(1,nmeio)-fita(1,nmeio-i);
```

```
                    lfta(1,nfimrest+i)=lfta(1,nfimrest)+lfta(1,nmeio)-lfta(1,nmeio-i);

                    alfata(1,nfimrest+i)=alfata(1,nmeio)+(alfata(1,nmeio)-alfata(1,↙
nmeio-i));

                    tetatap(1,nfimrest+i)=tetatap(1,nmeio-i);
                    tetata(1,nfimrest+i)=tetatap(1,nmeio-i);
                    dalfadtetata(1,nfimrest+i)=-dalfadtetata(1,nmeio-i);

                end
                n=n-1;
            end

            nfinal=n;
            alfafinal=alfata(1,n);
            fitafinal=fita(1,n);

            % Cartesianas
            rota=rc./sin(tetatap);
            xta=rota.*sin(tetatap).*cos(fita);
            yta=rota.*sin(tetatap).*sin(fita);
            zta=-rota.*cos(tetatap)+zeroteta;


        case {'cone'}
            %% Cálculos iniciais
            alfaita=alfa(1,size(alfa,2));


            %Padrão - sempre a assumir rfg>rig
            beff=b/cos(alfaita);
            nreal=2*pi*ro(1,size(ro,2)-1)/beff;
            n1=ceil(nreal);


            alfata=[];
            alfata(1,1)=alfaita;

            alfata21=[];
            alfata21(1,1)=pi-alfai;

            fita=[];
            fita(1,1)=fi(1,size(fi,2));

            fita21=[];
            fita21(1,1)=0;

            lfta=[];
            lfta(1,1)=lf(1,size(lf,2));

            lfta21=[];
            lfta21(1,1)=0;

            dalfadrota=[];
            dalfadrota(1,1)=dalfadro(1,size(dalfadro,2));
```

```
        dalfadrota21=[];
        dalfadrota21(1,1)=-dalfadro(1,1);

        rota=[];
        rota(1,1)=ro(1,size(ro,2));

        rota21=[];
        rota21(1,1)=rig;

        zta=[];
        zta(1,1)=z(1,size(z,2));

        zta21=[];
        zta21(1,1)=zig;

        n=1;
        nta=1;

        if alfaita<0

        %% Primeira metade do TA1
        while alfata(1,n)>-pi/2

            rota(1,n+1)=rota(1,n)+ds*cos(alfata(1,n))*sin(tau);
            zta(1,n+1)=rota(1,n+1)*lm/dm-rim*lm/dm+zim;

            %Cálculo das formas fundamentais em rota(1,n)
            E1=vpa(subs(gg1(1,1),'ro',rota(1,n)));
            E1=vpa(subs(E1,'fi',fita(1,n)));
            G1=vpa(subs(gg1(2,2),'ro',rota(1,n)));
            G1=vpa(subs(G1,'fi',fita(1,n)));

            L1=vpa(subs(gg2(1,1),'ro',rota(1,n)));
            L1=vpa(subs(L1,'fi',fita(1,n)));
            N1=vpa(subs(gg2(2,2),'ro',rota(1,n)));
            N1=vpa(subs(N1,'fi',fita(1,n)));

            %Cálculo de dalfadro em rota(1,n)
            dalfadrota0=-0.5*ero*tan(alfata(1,n))/gg1(1,1)+lambda1*((sqrt(G1)/cos↙
(alfata(1,n)))*(((cos(alfata(1,n))^2)*N1/G1)+((sin(alfata(1,n))^2)*L1/E1)));
            lambdacenas2(1,n)=lambda1*((sqrt(G1)/cos(alfata(1,n)))*(((cos(alfata(1,n))↙
^2)*N1/G1)+((sin(alfata(1,n))^2)*L1/E1)));
            lambdacenas21(1,n)=double(subs(lambdacenas2(1,n),'ro',rota(1,n)));


            %Cálculo das formas fundamentais em rota(1,n+1)
            E2=vpa(subs(gg1(1,1),'ro',rota(1,n+1)));
            E2=vpa(subs(E2,'fi',fita(1,n)));
            G2=vpa(subs(gg1(2,2),'ro',rota(1,n+1)));
            G2=vpa(subs(G2,'fi',fita(1,n)));

            L2=vpa(subs(gg2(1,1),'ro',rota(1,n+1)));
            L2=vpa(subs(L2,'fi',fita(1,n)));
            N2=vpa(subs(gg2(2,2),'ro',rota(1,n+1)));
```

```
            N2=vpa(subs(N2,'fi',fita(1,n)));

            %Cálculo de dalfadro em rota(1,n)
            dalfadrota1=-0.5*ero*tan(alfata(1,n))/gg1(1,1)+lambda1*((sqrt(G2)/cos↙
(alfata(1,n)))*(((cos(alfata(1,n))^2)*N2/G2)+((sin(alfata(1,n))^2)*L2/E2)));

            dalfadrota0=double(subs(dalfadrota0,'ro',rota(1,n)));
            dalfadrota1=double(subs(dalfadrota1,'ro',rota(1,n+1)));

            dalfadrota(1,n)=dalfadrota0;

            %Cálculo de alfa
            alfata(1,n+1)=alfata(1,n)+(rota(1,n+1)-rota(1,n))*↙
(dalfadrota0+dalfadrota1)/2;

            %Cálculo de fi e Lf
            fita(1,n+1)=fita(1,n)+0.5*(rota(1,n+1)-rota(1,n))*(tan(alfata(1,n))*sqrt↙
(G1/E1)+tan(alfata(1,n+1))*sqrt(G2/E2));
            lfta(1,n+1)=lfta(1,n)+0.5*(rota(1,n+1)-rota(1,n))*(sqrt(G1)/cos(alfata(1,↙
n))+sqrt(G2)/cos(alfata(1,n+1)));

            n=n+1;

        end

        dalfadrota(1,n)=dalfadrota(1,n-1);

        dfi=fi(1,size(fi,2))-fi(1,1);
        dfita=fita(1,size(fita,2))-fita(1,1);
        fitameio=fita(1,n);
        nmeio=n;

        disp('fim TA1')




        %% Primeira metade do TA2
        alfata21(1,1)=-pi-alfai;

        while alfata21(1,nta)<-pi/2

            rota21(1,nta+1)=rota21(1,nta)-ds*cos(alfata21(1,nta))*sin(tau);
            zta21(1,nta+1)=rota21(1,nta+1)*lm/dm-rim*lm/dm+zim;

            %Cálculo das formas fundamentais em tetatap(1,n)
            E1=vpa(subs(gg1(1,1),'ro',rota21(1,nta)));
            E1=vpa(subs(E1,'fi',fita21(1,nta)));
            G1=vpa(subs(gg1(2,2),'ro',rota21(1,nta)));
            G1=vpa(subs(G1,'fi',fita21(1,nta)));

            L1=vpa(subs(gg2(1,1),'ro',rota21(1,nta)));
            L1=vpa(subs(L1,'fi',fita21(1,nta)));
            N1=vpa(subs(gg2(2,2),'ro',rota21(1,nta)));
            N1=vpa(subs(N1,'fi',fita21(1,nta)));
```

```
            %Cálculo de dalfadteta em tetatap(1,n)
            dalfadrota0=-0.5*ero*tan(alfata21(1,nta))/gg1(1,1)-lambda1*((sqrt(G1)/cos↙
(alfata21(1,nta)))*(((cos(alfata21(1,nta))^2)*N1/G1)+((sin(alfata21(1,nta))^2)↙
*L1/E1)));

            %dalfadrota0=-0.5*ero*tan(alfata21(1,nta))/gg1(1,1)+lambda1*((1/cos↙
(alfata21(1,nta)))*(((zlinha+zlinha^3-rota21(1,nta)*z2linha)/(rota21(1,nta)*↙
(1+zlinha^2)))*sin(alfata21(1,nta))^2+z2linha/(1+zlinha^2)));
            lambdacenas(1,nta)=lambda1*((sqrt(G1)/cos(alfata21(1,nta)))*(((cos↙
(alfata21(1,nta))^2)*N1/G1)+((sin(alfata21(1,nta))^2)*L1/E1)));
            lambdacenasz(1,nta)=lambda1*((1/cos(alfata21(1,nta)))*(((zlinha+zlinha^3-↙
rota21(1,nta)*z2linha)/(rota21(1,nta)*(1+zlinha^2)))*sin(alfata21(1,nta))^2+z2linha/↙
(1+zlinha^2)));
            geocenas(1,nta)=0.5*ero*tan(alfata21(1,nta))/gg1(1,1);

            %Cálculo das formas fundamentais em tetatap(1,n+1)
            E2=vpa(subs(gg1(1,1),'ro',rota21(1,nta+1)));
            E2=vpa(subs(E2,'fi',fita21(1,nta)));
            G2=vpa(subs(gg1(2,2),'ro',rota21(1,nta+1)));
            G2=vpa(subs(G2,'fi',fita21(1,nta)));

            L2=vpa(subs(gg2(1,1),'ro',rota21(1,nta+1)));
            L2=vpa(subs(L2,'fi',fita21(1,nta)));
            N2=vpa(subs(gg2(2,2),'ro',rota21(1,nta+1)));
            N2=vpa(subs(N2,'fi',fita21(1,nta)));

            %Cálculo de dalfadteta em tetatap(1,n)
            %dalfadrota1=-0.5*ero*tan(alfata21(1,nta))/gg1(1,1)+lambda1*((1/cos↙
(alfata21(1,nta)))*(((zlinha+zlinha^3-rota21(1,nta+1)*z2linha)/(rota21(1,nta+1)*↙
(1+zlinha^2)))*sin(alfata21(1,nta))^2+z2linha/(1+zlinha^2)));
            dalfadrota1=-0.5*ero*tan(alfata21(1,nta))/gg1(1,1)-lambda1*((sqrt(G2)/cos↙
(alfata21(1,nta)))*(((cos(alfata21(1,nta))^2)*N2/G2)+((sin(alfata21(1,nta))^2)↙
*L2/E2)));

            dalfadrota0=double(subs(dalfadrota0,'ro',rota21(1,nta)));
            dalfadrota1=double(subs(dalfadrota1,'ro',rota21(1,nta+1)));
            lambdacenas1(1,nta)=double(subs(lambdacenas(1,nta),'ro',rota21(1,nta)));
            lambdacenasz1(1,nta)=double(subs(lambdacenasz(1,nta),'ro',rota21(1,nta)));
            geocenas1(1,nta)=double(subs(geocenas(1,nta),'ro',rota21(1,nta)));
            dalfadrota21(1,nta)=dalfadrota0;

            %Cálculo de alfa
            alfata21(1,nta+1)=alfata21(1,nta)+abs(rota21(1,nta+1)-rota21(1,nta))*abs↙
(dalfadrota0+dalfadrota1)/2;
            %Cálculo de fi e Lf
            fita21(1,nta+1)=fita21(1,nta)+0.5*(rota21(1,nta+1)-rota21(1,nta))*(tan↙
(alfata21(1,nta))*sqrt(G1/E1)+tan(alfata21(1,nta+1))*sqrt(G2/E2));
            lfta21(1,nta+1)=lfta21(1,nta)+0.5*(rota21(1,nta+1)-rota21(1,nta))*(sqrt↙
(G1)/cos(alfata21(1,nta))+sqrt(G2)/cos(alfata21(1,nta+1)));

            nta=nta+1;

        end
```

```
        nmeio2=nta;

        disp('fim TA2')

        dalfadrota21(1,nta)=dalfadrota21(1,nta-1);

        dfita21=fita21(1,size(fita21,2))-fita21(1,1);


        else

        %% Primeira metade do TA1
        while alfata(1,n)<pi/2

            rota(1,n+1)=rota(1,n)+ds*cos(alfata(1,n))*sin(tau);
            zta(1,n+1)=rota(1,n+1)*lm/dm-rim*lm/dm+zim;

            %Cálculo das formas fundamentais em rota(1,n)
            E1=vpa(subs(gg1(1,1),'ro',rota(1,n)));
            E1=vpa(subs(E1,'fi',fita(1,n)));
            G1=vpa(subs(gg1(2,2),'ro',rota(1,n)));
            G1=vpa(subs(G1,'fi',fita(1,n)));

            L1=vpa(subs(gg2(1,1),'ro',rota(1,n)));
            L1=vpa(subs(L1,'fi',fita(1,n)));
            N1=vpa(subs(gg2(2,2),'ro',rota(1,n)));
            N1=vpa(subs(N1,'fi',fita(1,n)));

            %Cálculo de dalfadro em rota(1,n)
            dalfadrota0=-0.5*ero*tan(alfata(1,n))/gg1(1,1)-lambda1*((sqrt(G1)/cos↙
(alfata(1,n)))*(((cos(alfata(1,n))^2)*N1/G1)+((sin(alfata(1,n))^2)*L1/E1)));
            lambdacenas2(1,n)=lambda1*((sqrt(G1)/cos(alfata(1,n)))*(((cos(alfata(1,n))↙
^2)*N1/G1)+((sin(alfata(1,n))^2)*L1/E1)));
            lambdacenas21(1,n)=double(subs(lambdacenas2(1,n),'ro',rota(1,n)));


            %Cálculo das formas fundamentais em rota(1,n+1)
            E2=vpa(subs(gg1(1,1),'ro',rota(1,n+1)));
            E2=vpa(subs(E2,'fi',fita(1,n)));
            G2=vpa(subs(gg1(2,2),'ro',rota(1,n+1)));
            G2=vpa(subs(G2,'fi',fita(1,n)));

            L2=vpa(subs(gg2(1,1),'ro',rota(1,n+1)));
            L2=vpa(subs(L2,'fi',fita(1,n)));
            N2=vpa(subs(gg2(2,2),'ro',rota(1,n+1)));
            N2=vpa(subs(N2,'fi',fita(1,n)));

            %Cálculo de dalfadro em rota(1,n)
            dalfadrota1=-0.5*ero*tan(alfata(1,n))/gg1(1,1)-lambda1*((sqrt(G2)/cos↙
(alfata(1,n)))*(((cos(alfata(1,n))^2)*N2/G2)+((sin(alfata(1,n))^2)*L2/E2)));

            dalfadrota0=double(subs(dalfadrota0,'ro',rota(1,n)));
            dalfadrota1=double(subs(dalfadrota1,'ro',rota(1,n+1)));

            dalfadrota(1,n)=dalfadrota0;
```

```
            %Cálculo de alfa
            alfata(1,n+1)=alfata(1,n)+(rota(1,n+1)-rota(1,n))*↙
(dalfadrota0+dalfadrota1)/2;

            %Cálculo de fi e Lf
            fita(1,n+1)=fita(1,n)+0.5*(rota(1,n+1)-rota(1,n))*(tan(alfata(1,n))*sqrt↙
(G1/E1)+tan(alfata(1,n+1))*sqrt(G2/E2));
            lfta(1,n+1)=lfta(1,n)+0.5*(rota(1,n+1)-rota(1,n))*(sqrt(G1)/cos(alfata(1,↙
n))+sqrt(G2)/cos(alfata(1,n+1)));

            n=n+1;

        end

        dalfadrota(1,n)=dalfadrota(1,n-1);

        dfi=fi(1,size(fi,2))-fi(1,1);
        dfita=fita(1,size(fita,2))-fita(1,1);
        fitameio=fita(1,n);
        nmeio=n;

        disp('fim TA1')



        %% Primeira metade do TA2
        alfata21(1,1)=pi-alfai;

        while alfata21(1,nta)>pi/2

            rota21(1,nta+1)=rota21(1,nta)+ds*cos(alfata21(1,nta))*sin(tau);
            zta21(1,nta+1)=rota21(1,nta+1)*lm/dm-rim*lm/dm+zim;

            %Cálculo das formas fundamentais em tetatap(1,n)
            E1=vpa(subs(gg1(1,1),'ro',rota21(1,nta)));
            E1=vpa(subs(E1,'fi',fita21(1,nta)));
            G1=vpa(subs(gg1(2,2),'ro',rota21(1,nta)));
            G1=vpa(subs(G1,'fi',fita21(1,nta)));

            L1=vpa(subs(gg2(1,1),'ro',rota21(1,nta)));
            L1=vpa(subs(L1,'fi',fita21(1,nta)));
            N1=vpa(subs(gg2(2,2),'ro',rota21(1,nta)));
            N1=vpa(subs(N1,'fi',fita21(1,nta)));

            %Cálculo de dalfadteta em tetatap(1,n)
            dalfadrota0=-0.5*ero*tan(alfata21(1,nta))/gg1(1,1)-lambda1*((sqrt(G1)/cos↙
(alfata21(1,nta)))*(((cos(alfata21(1,nta))^2)*N1/G1)+((sin(alfata21(1,nta))^2)↙
*L1/E1)));

            %dalfadrota0=-0.5*ero*tan(alfata21(1,nta))/gg1(1,1)+lambda1*((1/cos↙
(alfata21(1,nta)))*(((zlinha+zlinha^3-rota21(1,nta)*z2linha)/(rota21(1,nta)*↙
(1+zlinha^2)))*sin(alfata21(1,nta))^2+z2linha/(1+zlinha^2)));
            lambdacenas(1,nta)=lambda1*((sqrt(G1)/cos(alfata21(1,nta)))*(((cos↙
(alfata21(1,nta))^2)*N1/G1)+((sin(alfata21(1,nta))^2)*L1/E1)));
```

```
            lambdacenasz(1,nta)=lambda1*((1/cos(alfata21(1,nta)))*(((zlinha+zlinha^3-↙
rota21(1,nta)*z2linha)/(rota21(1,nta)*(1+zlinha^2)))*sin(alfata21(1,nta))^2+z2linha/↙
(1+zlinha^2)));
            geocenas(1,nta)=0.5*ero*tan(alfata21(1,nta))/gg1(1,1);

            %Cálculo das formas fundamentais em tetatap(1,n+1)
            E2=vpa(subs(gg1(1,1),'ro',rota21(1,nta+1)));
            E2=vpa(subs(E2,'fi',fita21(1,nta)));
            G2=vpa(subs(gg1(2,2),'ro',rota21(1,nta+1)));
            G2=vpa(subs(G2,'fi',fita21(1,nta)));

            L2=vpa(subs(gg2(1,1),'ro',rota21(1,nta+1)));
            L2=vpa(subs(L2,'fi',fita21(1,nta)));
            N2=vpa(subs(gg2(2,2),'ro',rota21(1,nta+1)));
            N2=vpa(subs(N2,'fi',fita21(1,nta)));

            %Cálculo de dalfadteta em tetatap(1,n)
            %dalfadrota1=-0.5*ero*tan(alfata21(1,nta))/gg1(1,1)+lambda1*((1/cos↙
(alfata21(1,nta)))*(((zlinha+zlinha^3-rota21(1,nta+1)*z2linha)/(rota21(1,nta+1)*↙
(1+zlinha^2)))*sin(alfata21(1,nta))^2+z2linha/(1+zlinha^2)));
            dalfadrota1=-0.5*ero*tan(alfata21(1,nta))/gg1(1,1)-lambda1*((sqrt(G2)/cos↙
(alfata21(1,nta)))*(((cos(alfata21(1,nta))^2)*N2/G2)+((sin(alfata21(1,nta))^2)↙
*L2/E2)));

            dalfadrota0=double(subs(dalfadrota0,'ro',rota21(1,nta)));
            dalfadrota1=double(subs(dalfadrota1,'ro',rota21(1,nta+1)));
            lambdacenas1(1,nta)=double(subs(lambdacenas(1,nta),'ro',rota21(1,nta)));
            lambdacenasz1(1,nta)=double(subs(lambdacenasz(1,nta),'ro',rota21(1,nta)));
            geocenas1(1,nta)=double(subs(geocenas(1,nta),'ro',rota21(1,nta)));
            dalfadrota21(1,nta)=dalfadrota0;

            %Cálculo de alfa
            alfata21(1,nta+1)=alfata21(1,nta)+(rota21(1,nta+1)-rota21(1,nta))*abs↙
(dalfadrota0+dalfadrota1)/2;
            %Cálculo de fi e Lf
            fita21(1,nta+1)=fita21(1,nta)+0.5*(rota21(1,nta+1)-rota21(1,nta))*(tan↙
(alfata21(1,nta))*sqrt(G1/E1)+tan(alfata21(1,nta+1))*sqrt(G2/E2));
            lfta21(1,nta+1)=lfta21(1,nta)+0.5*abs(rota21(1,nta+1)-rota21(1,nta))*(sqrt↙
(G1)/cos(alfata21(1,nta))+sqrt(G2)/cos(alfata21(1,nta+1)));

            nta=nta+1;

        end

        nmeio2=nta;

        disp('fim TA2')

        dalfadrota21(1,nta)=dalfadrota21(1,nta-1);

        dfita21=fita21(1,size(fita21,2))-fita21(1,1);
        end
```

```
        %% Dwell

        dfita1=dsta/rota(1,size(rota,2));
        dfita2=dsta/rota21(1,size(rota21,2));


        fifim=dfi*2+2*dfita+2*dfita21;

        p=1;
        while fifim>p*2*pi
            p=p+1;
        end

        dif=p*2*pi-fifim;
        difpadrao=2*pi/n1;


        if dif-difpadrao>=0
            fidwell=dif-difpadrao;
        else
            fidwell=dif+difpadrao;
        end

        fidwell1=fidwell*rota(1,size(rota,2))/(rota(1,size(rota,2))+rota21(1,size↙
(rota21,2)));
        fidwell2=fidwell*rota21(1,size(rota21,2))/(rota(1,size(rota,2))+rota21(1,size↙
(rota21,2)));

        alfata(1,n)=pi/2;
        while fita(1,n)<fitameio+fidwell1
            n=n+1;
            fita(1,n)=fita(1,n-1)+dfita1;
            lfta(1,n)=lfta(1,n-1)+rc*dfita1;
            alfata(1,n)=alfata(1,n-1);
            dalfadrota(1,n)=dalfadrota(1,n-1);
            rota(1,n)=rota(1,n-1);
            zta(1,n)=zta(1,n-1);
        end

        fita(1,n)=fita(1,nmeio)+fidwell1;
        lfta(1,n)=lfta(1,n-1)+rc*(fita(1,n)-fita(1,n-1));

        nfimrest=n;

        %% Segunda metade

        parte2turnaround='simetrico'

        switch lower(parte2turnaround)
            case {'integral'}


            case {'simetrico'}

                for i=0:nmeio-1
```

```
                    n=n+1;

                    fita(1,nfimrest+i)=fita(1,nfimrest)+fita(1,nmeio)-fita(1,nmeio-i);
                    lfta(1,nfimrest+i)=lfta(1,nfimrest)+lfta(1,nmeio)-lfta(1,nmeio-i);

                    alfata(1,nfimrest+i)=alfata(1,nmeio)+(alfata(1,nmeio)-alfata(1,↙
nmeio-i));

                    rota(1,nfimrest+i)=rota(1,nmeio-i);
                    zta(1,nfimrest+i)=zta(1,nmeio-i);
                    dalfadrota(1,nfimrest+i)=-dalfadrota(1,nmeio-i);

               end
               n=n-1;
        end

        nfinal=n;
        alfafinal=alfata(1,n);
        fitafinal=fita(1,n);

        % Cartesianas
        xta=rota.*cos(fita);
        yta=rota.*sin(fita);

end




%↙
----------------------------------------------------------------------------------------↙
------------------------
%↙
----------------------------------------------------------------------------------------↙
------------------------
%↙
----------------------------------------------------------------------------------------↙
------------------------
%-------------------------------------------------↙
Trajectória2-------------------------------------------------



%% Trajectória da zona útil 2

if rim==rfm
    traj2='cilindro simetrico'
else
    traj2='cone simetrico'
end

switch lower(traj2)
```

```
    case {'cone analitica'} %descontinuidade no inicio
        %% Trajectoria geodésica para cones - Koussios
        c1=sin(pi-alfai)*ri; %ver qual o angulo que deve estar aqui

        z2=[];
        fi2=[];
        fi21=[];
        ro2=[];
        alfa2=[];
        alfa21=[];
        dalfadro2=[];
        lf2=[];

        alfa2(1,1)=alfata(1,size(alfata,2));
        alfa21(1,1)=alfa2(1,1);
        dalfadro2(1,1)=dalfadrota(1,size(dalfadrota,2));
        dalfadro2(1,2)=dalfadro2(1,1);
        z2(1,1)=zfg;
        ro2(1,1)=((z2(1,1)*dm/lm)+rim-(dm*zim/lm));
        fi21(1,1)=0;
        lf2(1,1)=lfta(1,size(lfta,2));

        i=1;

        while z2(1,i)>zig
            i=i+1;
            z2(1,i)=z2(1,i-1)-ds*cos(alfa21(1,i-1))*cos(tau);
            ro2(1,i)=((z2(1,i)*dm/lm)+rim-(dm*zim/lm));

            fi21(1,i)=(1/sin(tau))*(asin(c1/rfg)-asin(c1/ro2(1,i)));  %fazer contas↙
para fazer fi ir aumentando

            alfa21(1,i)=atan((ro2(1,i)*(fi21(1,i)-fi21(1,i-1)))/((z2(1,i)-z2(1,i-1))↙
/cos(tau)));
            alfa21(1,i)=alfa21(1,i)+pi;
            dalfadro2(1,i)=(alfa21(1,i)-alfa21(1,i-1))/(ro2(1,i)-ro2(1,i-1));
            lf2(1,i)=lf2(1,i-1)+ds;

        end


        if z2(1,i)~=zig
            z2(1,i)=zig;
            ro2(1,i)=((z2(1,i)*dm/lm)+rim-(dm*zim/lm));
            fi21(1,i)=(1/sin(tau))*(asin(c1/rfg)-asin(c1/ro2(1,i)));
            alfa21(1,i)=atan((ro2(1,i)*(fi21(1,i)-fi21(1,i-1)))/((z2(1,i)-z2(1,i-1))↙
/cos(tau)));
            alfa21(1,i)=alfa21(1,i)+pi;
            dalfadro2(1,i)=(alfa21(1,i)-alfa21(1,i-1))/(ro2(1,i)-ro2(1,i-1));
            lf2(1,i)=lf2(1,i-1)+(z2(1,i)-z2(1,i-1))/(cos(tau)*cos(alfa21(1,i-1)));
        end

        alfa2=zeros(1,size(alfa21,2));
        i=i+1;
```

```
        %passo extra
            z2pe=z2(1,i-1)-0.00001*cos(alfa21(1,i-1))*cos(tau);
            ro2pe=((z2pe*dm/lm)+rim-(dm*zim/lm));
            fi21pe=(1/sin(tau))*(asin(c1/rfg)-asin(c1/ro2pe));
            alfa21(1,i)=atan((ro2pe*(fi21pe-fi21(1,i-1)))/((z2pe-z2(1,i-1))/cos↙
(tau)));
            alfa21(1,i)=alfa21(1,i)+pi;



        for j=1:size(alfa2,2)
           alfa2(1,i)=alfa21(1,i+1);
        end

        fi2=-fi21+fita(1,size(fita,2));

        %Cartesianas
        x2=ro2.*cos(fi2);
        y2=ro2.*sin(fi2);

    case {'cone simetrico'}

        fi2=[];
        lf2=[];
        alfa2=[];
        dalfadro2=[];
        ro2=[];
        z2=[];

        n=size(z,2);

        for i=1:size(z,2)
           z2(1,i)=z(1,n);
           ro2(1,i)=ro(1,n);
           dalfadro2(1,i)=-dalfadro(1,n);
           if alfai>0
               alfa2(1,i)=pi-alfa(1,n);
           else
               alfa2(1,i)=-pi-alfa(1,n);
           end
           fi2(1,i)=fita(1,size(fita,2))+fi(1,size(fi,2))-fi(1,n);
           lf2(1,i)=lfta(1,size(lfta,2))+lf(1,size(lf,2))-lf(1,n);

           n=n-1;
        end
        alfa2(1,1)=alfata(1,size(alfata,2));
        %% Cartesianas
        x2=ro2.*cos(fi2);
        y2=ro2.*sin(fi2);


    case {'cones integral'}    %coordenadas polares
        %% Cálculos iniciais

        fi2=[];
```

```
        lf2=[];
        alfa2=[];
        dalfadro2=[];
        ro2=[];
        z2=[];

        z2(1,1)=zfg;
        ro2(1,1)=rfg;
        if alfai>0
            alfa2(1,1)=pi-alfa(1,size(alfa,2));
        else
            alfa2(1,1)=-pi-alfa(1,size(alfa,2));
        end

        fi2(1,1)=fita(1,size(fita,2));
        lf2(1,1)=lfta(1,size(lfta,2));

        n=1;

        %% Cálculo de fi, alfa e Lf
        while z2(1,n)<zfg

           ro2(1,n+1)=ro2(1,n)+ds*cos(alfa2(1,n))*sin(tau);
           z2(1,n+1)=ro2(1,n+1).*lm/dm-rim*lm/dm+zim;

           dalfadro0=-0.5*ero*tan(alfa2(1,n))/gg1(1,1);
           dalfadro1=-0.5*ero*tan(alfa2(1,n))/gg1(1,1);

           dalfadro0=double(subs(dalfadro0,'ro',ro2(1,n)));

           dalfadro1=double(subs(dalfadro1,'ro',ro2(1,n+1)));

           dalfadro2(1,n)=dalfadro0;

           alfa2(1,n+1)=alfa2(1,n)+(ro2(1,n+1)-ro2(1,n))*(dalfadro0+dalfadro1)/2;

           E1=vpa(subs(gg1(1,1),'ro',ro2(1,n)));
           E1=vpa(subs(E1,'fi',fi2(1,n)));
           G1=vpa(subs(gg1(2,2),'ro',ro2(1,n)));
           G1=vpa(subs(G1,'fi',fi2(1,n)));

           E2=vpa(subs(gg1(1,1),'ro',ro2(1,n+1)));
           E2=vpa(subs(E1,'fi',fi2(1,n)));
           G2=vpa(subs(gg1(2,2),'ro',ro2(1,n+1)));
           G2=vpa(subs(G1,'fi',fi2(1,n)));

           n=n+1;

           fi2(1,n)=fi2(1,n-1)+0.5*(ro2(1,n)-ro2(1,n-1))*(tan(alfa2(1,n-1))*sqrt↙
(G1/E1)+tan(alfa2(1,n))*sqrt(G2/E2)); %verificar contas
           lf2(1,n)=lf2(1,n-1)+0.5*(ro2(1,n)-ro2(1,n-1))*(sqrt(G1)/cos(alfa2(1,n-1))↙
+sqrt(G2)/cos(alfa2(1,n))); %verificar contas

        end
```

```
        if z2(1,n)~=zig
           z2(1,n+1)=zig;
           ro2(1,n+1)=((z2(1,n+1)*dm/lm)+rim-(dm*zim/lm));
           dalfadro0=0.5*ero*tan(alfa2(1,n))/gg1(1,1);
           dalfadro1=0.5*ero*tan(alfa2(1,n))/gg1(1,1);
           dalfadro0=double(subs(dalfadro0,'ro',ro2(1,n)));
           dalfadro1=double(subs(dalfadro1,'ro',ro2(1,n+1)));
           dalfadro2(1,n)=dalfadro0;
           alfa2(1,n+1)=alfa2(1,n)+(ro2(1,n+1)-ro2(1,n))*(dalfadro0+dalfadro1)/2;

           E1=vpa(subs(gg1(1,1),'ro',ro2(1,n)));
           E1=vpa(subs(E1,'fi',fi2(1,n)));
           G1=vpa(subs(gg1(2,2),'ro',ro2(1,n)));
           G1=vpa(subs(G1,'fi',fi2(1,n)));

           E2=vpa(subs(gg1(1,1),'ro',ro2(1,n+1)));
           E2=vpa(subs(E1,'fi',fi2(1,n)));
           G2=vpa(subs(gg1(2,2),'ro',ro2(1,n+1)));
           G2=vpa(subs(G1,'fi',fi2(1,n)));

           n=n+1;

           fi2(1,n)=fi2(1,n-1)+0.5*(ro2(1,n)-ro2(1,n-1))*(tan(alfa2(1,n-1))*sqrt↙
(G1/E1)+tan(alfa2(1,n))*sqrt(G2/E2)); %verificar contas
           lf2(1,n)=lf2(1,n-1)+0.5*(ro2(1,n)-ro2(1,n-1))*(sqrt(G1)/cos(alfa2(1,n-1))↙
+sqrt(G2)/cos(alfa2(1,n))); %verificar contas

        end

        %% Cartesianas
        x2=ro2.*cos(fi2);
        y2=ro2.*sin(fi2);


    case {'cilindro analitica'}
        %% Trajectória geodésica para cilindros - hélices
        % Helice

        dz2=ds*cos(pi-alfai);

        z2=[];
        z2(1,:)=zfg:dz2:zig;

        if z2(1,size(z2,2))~=zig
            z2(1,size(z2,2)+1)=zig;
        end

        c2=rc/tan(pi-alfai);

        fi21=z2/c2;
        fi2=fita(1,size(fita,2))+fi21-fi21(1,1);

        x2=rc*cos(fi2);
        y2=rc*sin(fi2);
```

```
        %para dar continuidade estes dados são necessários no TA
        alfa2=ones(1,size(x2,2)).*(pi-alfai);

        lf2=sqrt(((fi2-fita(1,size(fita,2))).*rc).^2+(zta(1,size(zta,2))-z2).^2)+lfta↙
(1,size(lfta,2));

        tetai2=pi-tetai;
        tetaf2=tetai;

        dalfadteta2=zeros(1,size(x2,2));

        teta2=atan(rc./(zeroteta-z2));
        tetap2=[];
        tetap2(1,1)=teta2(1,1);

        for n=1:size(teta2,2)

            if teta2(1,n)<0
                tetap2(1,n)=teta2(1,n)+pi;
            else
                tetap2(1,n)=teta2(1,n);
            end
        end

    case {'cilindro simetrico'} %

        fi2=[];
        lf2=[];
        alfa2=[];
        dalfadteta2=[];
        z2=[];
        tetap2=[];
        teta2=[];

        n=size(z,2);

        for i=1:size(z,2)
           z2(1,i)=z(1,n);
           tetap2(1,i)=tetap(1,n);
           teta2(1,i)=teta(1,n);
           dalfadteta2(1,i)=-dalfadteta(1,n);
           if alfai>0
               alfa2(1,i)=pi-alfa(1,n);
           else
               alfa2(1,i)=-pi-alfa(1,n);
           end
           fi2(1,i)=fita(1,size(fita,2))+fi(1,size(fi,2))-fi(1,n);
           lf2(1,i)=lfta(1,size(lfta,2))+lf(1,size(lf,2))-lf(1,n);

           n=n-1;
        end
        alfa2(1,1)=alfata(1,size(alfata,2));
        %% Cartesianas
```

```
        x2=rc*cos(fi2);
        y2=rc*sin(fi2);
        tetaf2=tetai;

    case {'cilindro esfericas'}    %integrar coordenadas esfericas
        %% Cálculos iniciais

        tetai2=pi-tetai;
        tetaf2=tetai;

        fi2=[];
        lf2=[];

        alfa2=[];
        dalfadteta2=[];
        alfa2(1,1)=alfata(1,size(alfata,2));

        teta2=[];
        teta1=tetai2;
        teta2(1,1)=teta1;

        tetap2=[];
        tetap2(1,1)=tetai2;

        fi2(1,1)=fita(1,size(fita,2));
        lf2(1,1)=lfta(1,size(lfta,2));

        n=1;
        j=1;

        %% Cálculo de fi, alfa e Lf
        while tetap2(1,n)>tetaf2

            %Gerar teta para ds constante
            j=n+1;
            teta2(1,j)=teta1;
            teta3=atan(1/(-(ds*cos(alfa2(1,n)))/rc+1/tan(teta1)));
            teta1=teta3;

            if teta2(1,j)<0
                tetap2(1,j)=teta2(1,j)+pi;
            else
                tetap2(1,j)=teta2(1,j);
            end

           dalfadteta0=0.5*eteta*tan(alfa2(1,n))/gg1(1,1);
           dalfadteta1=0.5*eteta*tan(alfa2(1,n))/gg1(1,1);

           dalfadteta0=double(subs(dalfadteta0,'teta',tetap2(1,n)));

           dalfadteta1=double(subs(dalfadteta1,'teta',tetap2(1,n+1)));

           dalfadteta2(1,n)=dalfadteta0;

           alfa2(1,n+1)=alfa2(1,n)+(tetap2(1,n+1)-tetap2(1,n))*↙
```

```
(dalfadteta0+dalfadteta1)/2;

            E1=vpa(subs(gg1(1,1),'teta',tetap2(1,n)));
            E1=vpa(subs(E1,'fi',fi2(1,n)));
            G1=vpa(subs(gg1(2,2),'teta',tetap2(1,n)));
            G1=vpa(subs(G1,'fi',fi2(1,n)));

            E2=vpa(subs(gg1(1,1),'teta',tetap2(1,n+1)));
            E2=vpa(subs(E1,'fi',fi2(1,n)));
            G2=vpa(subs(gg1(2,2),'teta',tetap2(1,n+1)));
            G2=vpa(subs(G1,'fi',fi2(1,n)));

            n=n+1;

            fi2(1,n)=fi2(1,n-1)+0.5*(tetap2(1,n)-tetap2(1,n-1))*(tan(alfa2(1,n-1))*sqrt↙
(G1/E1)+tan(alfa2(1,n))*sqrt(G2/E2));
            lf2(1,n)=lf2(1,n-1)+0.5*(tetap2(1,n)-tetap2(1,n-1))*(sqrt(G1)/cos(alfa2(1,↙
n-1))+sqrt(G2)/cos(alfa2(1,n)));

         end

         if tetap2(1,n-1)~=tetaf2
            tetap2(1,n)=tetaf2;
            dalfadteta0=0.5*eteta*tan(alfa2(1,n))/gg1(1,1);
            dalfadteta1=0.5*eteta*tan(alfa2(1,n))/gg1(1,1);

            dalfadteta0=double(subs(dalfadteta0,'teta',tetap2(1,n)));

            dalfadteta1=double(subs(dalfadteta1,'teta',tetap2(1,n+1)));

            dalfadteta2(1,n)=dalfadteta0;

            alfa2(1,n+1)=alfa2(1,n)+(tetap2(1,n+1)-tetap2(1,n))*↙
(dalfadteta0+dalfadteta1)/2;

            E1=vpa(subs(gg1(1,1),'teta',tetap2(1,n)));
            E1=vpa(subs(E1,'fi',fi2(1,n)));
            G1=vpa(subs(gg1(2,2),'teta',tetap2(1,n)));
            G1=vpa(subs(G1,'fi',fi2(1,n)));

            E2=vpa(subs(gg1(1,1),'teta',tetap2(1,n+1)));
            E2=vpa(subs(E1,'fi',fi2(1,n)));
            G2=vpa(subs(gg1(2,2),'teta',tetap2(1,n+1)));
            G2=vpa(subs(G1,'fi',fi2(1,n)));

            n=n+1;

            fi2(1,n)=fi2(1,n-1)+0.5*(tetap2(1,n)-tetap2(1,n-1))*(tan(alfa2(1,n-1))*sqrt↙
(G1/E1)+tan(alfa2(1,n))*sqrt(G2/E2));
            lf2(1,n)=lf2(1,n-1)+0.5*(tetap2(1,n)-tetap2(1,n-1))*(sqrt(G1)/cos(alfa2(1,↙
n-1))+sqrt(G2)/cos(alfa2(1,n)));

         end

         %% Cartesianas
```

```
        ro2=rc./sin(tetap2);
        x2=ro2.*sin(tetap2).*cos(fi2);
        y2=ro2.*sin(tetap2).*sin(fi2);
        z2=-ro2.*cos(tetap2)+zeroteta;


end




%↙
-----------------------------------------------------------------------------------------------↙
------------------------
%↙
-----------------------------------------------------------------------------------------------↙
------------------------
%↙
-----------------------------------------------------------------------------------------------↙
------------------------
%--------------------------------------------------↙
Turnaround2-----------------------------------------------------



%% Turnaround 2
if rim==rfm
    turnaround2='cilindro'
else
    turnaround2='cone'
end

switch lower(turnaround2)
    case {'cone'}
        zta2=[];
        fita2=[];
        rota2=[];
        alfata2=[];
        lfta2=[];
        dalfadrota2=[];

        zta2(1,:)=zta21(1,:);
        fita2(1,:)=fi2(1,size(fi2,2))+fita21(1,:);
        rota2(1,:)=rota21(1,:);
        alfata2(1,:)=alfa2(1,size(alfa2,2))-abs(-alfata21(1,:)+alfata21(1,1));
        alfata2(1,size(alfata21,2))=pi/2;
        lfta2(1,:)=lf2(1,size(lf2,2))-lfta21(1,:);
        dalfadrota2(1,:)=dalfadrota21(1,:);

        fitameio2=fita21(1,size(fita21,2))+fi2(1,size(fi2,2));
        n=size(fita21,2);
```

```
        %Dwell2
        while fita2(1,n)<fitameio2+fidwell2
            n=n+1;
            fita2(1,n)=fita2(1,n-1)+dfita2;
            lfta2(1,n)=lfta2(1,n-1)+rc*dfita2;
            alfata2(1,n)=alfata2(1,n-1);
            dalfadrota2(1,n)=dalfadrota2(1,n-1);
            rota2(1,n)=rota2(1,n-1);
            zta2(1,n)=zta2(1,n-1);
        end

        fita2(1,n)=fita2(1,nmeio2)+fidwell2;
        lfta2(1,n)=lfta2(1,n-1)+rc*(fita2(1,n)-fita2(1,n-1));

        nfimrest2=n;


        for i=0:nmeio2-1
             n=n+1;

             fita2(1,nfimrest2+i)=fita2(1,nfimrest2)+fita2(1,nmeio2)-fita2(1,nmeio2-↙
i);
             lfta2(1,nfimrest2+i)=lfta2(1,nfimrest2)+lfta2(1,nmeio2)-lfta2(1,nmeio2-↙
i);
             if alfai>0
                 alfata2(1,nfimrest2+i)=pi/2-(alfata2(1,nmeio2-i)-pi/2);
             else
                 alfata2(1,nfimrest2+i)=-pi/2+(-alfata2(1,nmeio2-i)-pi/2);
             end
             rota2(1,nfimrest2+i)=rota2(1,nmeio2-i);
             zta2(1,nfimrest2+i)=zta2(1,nmeio2-i);
             dalfadrota2(1,nfimrest2+i)=-dalfadrota2(1,nmeio2-i);

        end
        n=n-1;

        %% Cartesianas
        xta2=rota2.*cos(fita2);
        yta2=rota2.*sin(fita2);


    case {'cilindro'}
        %% Cálculos iniciais
        alfaita2=alfa2(1,size(alfa2,2));

        teta1=tetaf2;

        tetata2=[];
        tetata2(1,1)=teta1;

        tetatap2=[];
        tetatap2(1,1)=teta1;

        alfata2=[];
        alfata2(1,1)=alfaita2;
```

```
fita2=[];
fita2(1,1)=fi2(1,size(fi2,2));

lfta2=[];
lfta2(1,1)=lf2(1,size(lf2,2));

dalfadtetata2=[];
dalfadtetata2(1,1)=dalfadteta2(1,size(dalfadteta2,2));
n=1;

%% Primeira metade
for i=2:nmeio
    n=n+1;
    fita2(1,i)=fita2(1,1)+fita(1,i)-fita(1,1);

    lfta2(1,i)=lfta2(1,1)+lfta(1,i)-lfta(1,1);

    alfata2(1,i)=pi-alfata(1,i);

    tetatap2(1,i)=tetatap2(1,1)-tetatap(1,i)+tetatap(1,1);
    tetata2(1,i)=tetatap2(1,i);

    dalfadtetata2(1,i)=-dalfadtetata(1,i);

end

%% Dwell
nmeio2=n;
alfatameio2=alfata2(1,n);
fitameio2=fita2(1,n);
dfita=dsta/rc;

while fita2(1,n)<fitameio2+fidwell
    n=n+1;
    fita2(1,n)=fita2(1,n-1)+dfita;
    lfta2(1,n)=lfta2(1,n-1)+rc*dfita;
    alfata2(1,n)=alfata2(1,n-1);
    dalfadtetata2(1,n)=dalfadtetata2(1,n-1);
    tetata2(1,n)=tetata2(1,n-1);
    tetatap2(1,n)=tetatap2(1,n-1);
end

fita2(1,n)=fita2(1,nmeio2)+fidwell;
lfta2(1,n)=lfta2(1,n-1)+rc*(fita2(1,n)-fita2(1,n-1));
nfimrest2=n;



%% Segunda metade

for i=0:nmeio2-1

    n=n+1;
    fita2(1,nfimrest2+i)=fita2(1,nfimrest2)+fita2(1,nmeio2)-fita2(1,nmeio2-i);
```

```
            lfta2(1,nfimrest2+i)=lfta2(1,nfimrest2)+lfta2(1,nmeio2)-lfta2(1,nmeio2-i);

            alfata2(1,nfimrest2+i)=pi-alfata2(1,nmeio2-i);

            tetatap2(1,nfimrest2+i)=tetatap2(1,nmeio2-i);
            tetata2(1,nfimrest2+i)=tetatap2(1,nmeio2-i);
            dalfadtetata2(1,nfimrest2+i)=-dalfadtetata2(1,nmeio2-i);

        end

        n=n-1;

        % Cartesianas
        rota2=rc./sin(tetatap2);
        xta2=rota2.*sin(tetatap2).*cos(fita2);
        yta2=rota2.*sin(tetatap2).*sin(fita2);
        zta2=-rota2.*cos(tetatap2)+zeroteta;
        zta2(1,1)=z2(1,size(z2,2));
        zta2(1,size(zta2,2))=zig;


end



%↙
----------------------------------------------------------------------------------------↙
-----------------------
%↙
----------------------------------------------------------------------------------------↙
-----------------------
%↙
----------------------------------------------------------------------------------------↙
-----------------------



%% Ciclo
ficiclo=[];
zciclo=[];
alfaciclo=[];
rociclo=[];
lfciclo=[];

ficiclo(1,:)=[fi fita fi2 fita2];
alfaciclo(1,:)=[alfa alfata alfa2 alfata2];
lfciclo(1,:)=[lf lfta lf2 lfta2];

if rim==rfm
    tetaciclo=[];
    dalfadtetaciclo=[];
    tetaciclo(1,:)=[tetap tetatap tetap2 tetatap2];
    dalfadtetaciclo(1,:)=[dalfadteta dalfadtetata dalfadteta2 dalfadtetata2];
    rociclo=ones(1,size(tetaciclo,2))*rc;
```

```
else

    dalfadrociclo=[];
    rociclo=[ro rota ro2 rota2];
    dalfadrociclo=[dalfadro dalfadrota dalfadro2 dalfadrota2];
end


% Cartesianas
xciclo=[x xta x2 xta2];
yciclo=[y yta y2 yta2];
zciclo=[z zta z2 zta2];



%% Espessura média por volume de secções
%Espessura média da zona útil
lfzu1c=lf(1,size(lf,2))+lf2(1,size(lf2,2))-lf2(1,1);
lfzu=lfzu1c*n1;

vfibra=b*lfzu*esp;
zutil=lg;
h1=zfg-(-rim*lm/dm+zim);

if rim==rfm
    dr=((vfibra/(pi*zutil)+rig^2)^0.5)-rig;
else
    dr=((pi*rfg^4 + pi*rig^4 + 24*rfg*vfibra*tan(tau) - 24*rig*vfibra*tan(tau) -
2*pi*rfg^2*rig^2 + 12*vfibra*zutil*tan(tau)^2 + 4*pi*h1^2*rfg^2*tan(tau)^2 +
4*pi*h1^2*rig^2*tan(tau)^2 + 4*pi*rig^2*zutil^2*tan(tau)^2 + 4*pi*h1*rfg^3*tan(tau) +
4*pi*h1*rig^3*tan(tau) - 4*pi*rig^3*zutil*tan(tau) - 4*pi*h1*rfg*rig^2*tan(tau) -
4*pi*h1*rfg^2*rig*tan(tau) + 4*pi*rfg^2*rig*zutil*tan(tau) - 8*pi*h1^2*rfg*rig*tan
(tau)^2 - 8*pi*h1*rig^2*zutil*tan(tau)^2 + 8*pi*h1*rfg*rig*zutil*tan(tau)^2)^(1/2) -
pi^(1/2)*rfg^2 + pi^(1/2)*rig^2 - 2*pi^(1/2)*h1*rfg*tan(tau) + 2*pi^(1/2)*h1*rig*tan
(tau) - 2*pi^(1/2)*rig*zutil*tan(tau))/(pi^(1/2)*(4*rfg - 4*rig + 2*zutil*tan(tau)));
end
espmedia=dr*cos(tau);

%Espessura média por secções ao longo do mandril
[zmax,imax]=max(zciclo(:));
[zmin,imin]=min(zciclo(:));

lrta1=lfta(nfimrest)-lfta(nmeio);
lrta2=lfta2(nfimrest2)-lfta2(nmeio2);

espmed='z';
switch (espmed)
    case {'i'}
        %variando i
        zl=zmin;
        e=2;
        lfesp=[];
        ze=[];
        ze1=[];
```

```
        ze2=[];
        lfesp(1,1)=lrta2;
        ze(1,1)=zmin;
        ze1(1,1)=zmin;
        ze2(1,1)=zmin;

        for i=(size(z2,2)+size(z,2)+size(zta,2)+nmeio):-5:(size(z,2)+nfimrest)
            lfesp1=lfciclo(1,i);
            lfesp2=lfciclo(1,i-5);
            lfesp(1,e)=2*abs(lfesp1-lfesp2);
            ze1(1,e)=zciclo(1,i);
            ze2(1,e)=zciclo(1,i-5);
            ze(1,e)=(ze1(1,e)+ze2(1,e))/2;
            e=e+1;
        end

        lfesp(1,e)=lrta1;
        ze1(1,e)=zmax;
        ze2(1,e)=zmax;
        ze(1,e)=zmax;

        lfespt=lfesp.*n1;
        espsec=zeros(1,size(lfesp,2));

        for e=1:size(lfesp,2)
            %para cones - rfg e rig sao ro1 e ro2
            zutil=abs(ze2(1,e)-ze1(1,e));
            h1=ze2(1,e)-(-rim*lm/dm+zim);
            vfibra=b*lfespt(1,e)*esp;
            if rim==rfm
                dr=((vfibra/(pi*zutil)+rig^2)^0.5)-rig;
            else
                dr=((pi*rfg^4 + pi*rig^4 + 24*rfg*vfibra*tan(tau) - 24*rig*vfibra*tan↙
(tau) - 2*pi*rfg^2*rig^2 + 12*vfibra*zutil*tan(tau)^2 + 4*pi*h1^2*rfg^2*tan(tau)^2 +↙
4*pi*h1^2*rig^2*tan(tau)^2 + 4*pi*rig^2*zutil^2*tan(tau)^2 + 4*pi*h1*rfg^3*tan(tau) +↙
4*pi*h1*rig^3*tan(tau) - 4*pi*rig^3*zutil*tan(tau) - 4*pi*h1*rfg*rig^2*tan(tau) -↙
4*pi*h1*rfg^2*rig*tan(tau) + 4*pi*rfg^2*rig*zutil*tan(tau) - 8*pi*h1^2*rfg*rig*tan↙
(tau)^2 - 8*pi*h1*rig^2*zutil*tan(tau)^2 + 8*pi*h1*rfg*rig*zutil*tan(tau)^2)^(1/2) -↙
pi^(1/2)*rfg^2 + pi^(1/2)*rig^2 - 2*pi^(1/2)*h1*rfg*tan(tau) + 2*pi^(1/2)*h1*rig*tan↙
(tau) - 2*pi^(1/2)*rig*zutil*tan(tau))/(pi^(1/2)*(4*rfg - 4*rig + 2*zutil*tan(tau)));
            end
            espsec(1,e)=dr*cos(tau);
        end

        figure (7)
        plot(ze,espsec)
        title('ze espsex')
        xlabel('z')
        ylabel('Espessura')

        figure (8)
        plot(ze,lfesp)
        title('ze,lfesp')
        xlabel('z')
        ylabel('Lf')
```

```
    case {'z'}
        %Variando z
        c=imin;
        zcena=zciclo(1,c);
        c2=1;

        for i=zmin:5:zmax
            p=0;
            zcena1=zcena;
            while zcena<i
                c=c-1;
                zcena=zciclo(1,c);
                p=p+1;
            end

            deltaz=abs(zcena-zcena1);
            lfespz(1,c2)=2*abs(lfciclo(1,c)-lfciclo(1,c+p));
            zespz(1,c2)=zciclo(1,c);
            c2=c2+1;
        end

        lfesptz=lfespz.*n1;
        espsecz=zeros(1,size(lfespz,2));

        for e=1:size(lfespz,2)-1
            %para cones - rfg e rig sao ro1 e ro2
            zutil=abs(zespz(1,e)-zespz(1,e+1));
            h1=zespz(1,e+1)-(-rim*lm/dm+zim);
            vfibra=b*lfesptz(1,e)*esp;
            if rim==rfm
                dr=((vfibra./(pi*zutil)+rig^2)^0.5)-rig;
            else
                dr=((pi*rfg^4 + pi*rig^4 + 24*rfg*vfibra*tan(tau) - 24*rig*vfibra*tan↙
(tau) - 2*pi*rfg^2*rig^2 + 12*vfibra*zutil*tan(tau)^2 + 4*pi*h1^2*rfg^2*tan(tau)^2 +↙
4*pi*h1^2*rig^2*tan(tau)^2 + 4*pi*rig^2*zutil^2*tan(tau)^2 + 4*pi*h1*rfg^3*tan(tau) +↙
4*pi*h1*rig^3*tan(tau) - 4*pi*rig^3*zutil*tan(tau) - 4*pi*h1*rfg*rig^2*tan(tau) -↙
4*pi*h1*rfg^2*rig*tan(tau) + 4*pi*rfg^2*rig*zutil*tan(tau) - 8*pi*h1^2*rfg*rig*tan↙
(tau)^2 - 8*pi*h1*rig^2*zutil*tan(tau)^2 + 8*pi*h1*rfg*rig*zutil*tan(tau)^2)^(1/2) -↙
pi^(1/2)*rfg^2 + pi^(1/2)*rig^2 - 2*pi^(1/2)*h1*rfg*tan(tau) + 2*pi^(1/2)*h1*rig*tan↙
(tau) - 2*pi^(1/2)*rig*zutil*tan(tau))/(pi^(1/2)*(4*rfg - 4*rig + 2*zutil*tan(tau)));
            end
            espsecz(1,e)=dr*cos(tau);
        end

        %figure (7)
        %plot(zespz,lfespz)
        %title('zespz,lfespz')
        %xlabel('z')
        %ylabel('Lf')

        %figure (8)
        %plot(zespz(1,1:size(zespz,2)),espsecz)
        %title('zespz,espsecz')
        %xlabel('z')
```

```
        %ylabel('Espessura')
end


%% Restantes Paths - NP

switch (lower(seccoes))
    case {'rectangular'}
dfilargura(1,:)=(b./(2.*rociclo(1,:))).*(cos(alfaciclo(1,:)));
%dfilargura=[dfilargura1(1,1:(size(z,2)+nfimrest-1)) -dfilargura1(1,(size(z,2)↙
+nfimrest):(size(z,2)+size(zta,2)+size(z2,2)+nfimrest2-1)) dfilargura1(1,(size(z,2)↙
+size(zta,2)+size(z2,2)+nfimrest2):size(dfilargura1,2))];
dzlargura(1,:)=(b/2).*sin(alfaciclo(1,:)).*cos(tau);
%dzlargura=[dzlargura1(1,1:(size(z,2)+nfimrest-1)) -dzlargura1(1,(size(z,2)+nfimrest):↙
(size(z,2)+size(zta,2)+size(z2,2)+nfimrest2-1)) dzlargura1(1,(size(z,2)+size(zta,2)↙
+size(z2,2)+nfimrest2):size(dzlargura1,2))];

ficiclola=zeros(2*NP+1,size(ficiclo,2));
zciclola=zeros(4*NP+2,size(ficiclola,2));
rociclola=zeros(2*NP+1,size(ficiclo,2));

for i=1:(2*NP+1)
    ficiclola(i,:)=ficiclo(1,:)-dfilargura(1,:)+(i-1)*(dfilargura(1,:)/NP);
    zciclola(i,:)=zciclo(1,:)+dzlargura(1,:)-(i-1)*(dzlargura(1,:)/NP);
    rociclola(i,:)=((zciclola(i,:)*dm/lm)+rim-(dm*zim/lm));
end

% Cartesianas multiplas linhas
xciclola=zeros(4*NP+2,size(ficiclola,2));
yciclola=zeros(4*NP+2,size(ficiclola,2));
for n=1:2*NP+1
   xciclola(n,:)=rociclola(n,:).*cos(ficiclola(n,:));
   yciclola(n,:)=rociclola(n,:).*sin(ficiclola(n,:));

end
c=1;
for n=2*NP+2:2*(2*NP+1)
    xciclola(n,:)=(rociclola((2*NP+2-c),:)+esp/cos(tau)).*cos(ficiclola((2*NP+2-↙
c),:));
    yciclola(n,:)=(rociclola((2*NP+2-c),:)+esp/cos(tau)).*sin(ficiclola((2*NP+2-↙
c),:));
    zciclola(n,:)=zciclola((2*NP+2-c),:)-esp*sin(tau);
    c=c+1;
end

    case {'lenticular'} %necessário dar espessura


    case {'eliptical'} %necessário dar espessura (belip) - meia elipse

        aelip=b/2;

        dfilargura(1,:)=(b./(2.*rociclo(1,:))).*(cos(alfaciclo(1,:)));
%dfilargura=[dfilargura1(1,1:(size(z,2)+nfimrest-1)) -dfilargura1(1,(size(z,2)↙
+nfimrest):(size(z,2)+size(zta,2)+size(z2,2)+nfimrest2-1)) dfilargura1(1,(size(z,2)↙
```

```
+size(zta,2)+size(z2,2)+nfimrest2):size(dfilargura1,2))];
dzlargura(1,:)=(b/2).*sin(alfaciclo(1,:)).*cos(tau);
%dzlargura=[dzlargura1(1,1:(size(z,2)+nfimrest-1)) -dzlargura1(1,(size(z,2)+nfimrest):↙
(size(z,2)+size(zta,2)+size(z2,2)+nfimrest2-1)) dzlargura1(1,(size(z,2)+size(zta,2)↙
+size(z2,2)+nfimrest2):size(dzlargura1,2))];

ficiclola=zeros(4*NP+2,size(ficiclo,2));
zciclola=zeros(4*NP+2,size(ficiclola,2));
rociclola=zeros(4*NP+2,size(ficiclo,2));
xelip=zeros(4*NP+2,size(ficiclo,2));
tetaelip=zeros(4*NP+2,size(ficiclo,2));
yelip=zeros(4*NP+2,size(ficiclo,2));

for i=1:(2*NP+1)
    ficiclola(i,:)=ficiclo(1,:)-dfilargura(1,:)+(i-1)*(dfilargura(1,:)/NP);
    zciclola(i,:)=zciclo(1,:)+dzlargura(1,:)-(i-1)*(dzlargura(1,:)/NP);
    rociclola(i,:)=((zciclola(i,:)*dm/lm)+rim-(dm*zim/lm));
end

for i=(2*NP+2):(4*NP+2)
    ficiclola(i,:)=ficiclo(1,:)-dfilargura(1,:)+(i-1)*(dfilargura(1,:)/NP);
    zciclola(i,:)=zciclo(1,:)+dzlargura(1,:)-(i-1)*(dzlargura(1,:)/NP);
    rociclola(i,:)=((zciclola(i,:)*dm/lm)+rim-(dm*zim/lm));
    xelip(i,:)=-aelip+(i-(2*NP+2))*aelip/NP;
    tetaelip(i,:)=acos(xelip(i,:)/aelip);
    yelip(i,:)=belip*sin(tetaelip(i,:));
end


% Cartesianas multiplas linhas
xciclola=zeros(4*NP+2,size(ficiclola,2));
yciclola=zeros(4*NP+2,size(ficiclola,2));
for n=1:2*NP+1
   xciclola(n,:)=rociclola(n,:).*cos(ficiclola(n,:));
   yciclola(n,:)=rociclola(n,:).*sin(ficiclola(n,:));

end
c=1;
for n=2*NP+2:2*(2*NP+1)
    xciclola(n,:)=(rociclola((2*NP+2-c),:)+yelip(n,:)/cos(tau)).*cos(ficiclola↙
((2*NP+2-c),:));
    yciclola(n,:)=(rociclola((2*NP+2-c),:)+yelip(n,:)/cos(tau)).*sin(ficiclola↙
((2*NP+2-c),:));
    zciclola(n,:)=zciclola((2*NP+2-c),:)-yelip(n,:)*sin(tau);
    c=c+1;
end

end
%% Plot
scrsz = get(0,'ScreenSize');
x0=0; y0=0; width=scrsz(3); height=scrsz(4);
posfig=[x0 y0 width height];


        %Mandril e trajectoria com multiplas linhas
```

```
        figure ('Name','First cycle','NumberTitle','off','Position',posfig);
        %surf(Z,X,Y,'FaceColor','c','FaceAlpha',.↙
7,'EdgeColor','none','LineStyle','none')
        %hold on;
        surf(zciclola(:,:),yciclola(:,:),xciclola↙
(:,:),'FaceColor','r','EdgeColor','none','LineStyle','none')
        hold on;
        %plot3(zciclo(1,:),xciclo(1,:),yciclo(1,:),'r','Linewidth',2);
        plot3(zciclola(1,:),yciclola(1,:),xciclola(1,:),'k','Linewidth',0.5);
        plot3(zciclola(:,1),yciclola(:,1),xciclola(:,1),'k','Linewidth',0.5);
        plot3(zciclola(:,size(zciclola,2)),yciclola(:,size(zciclola,2)),xciclola(:,↙
size(zciclola,2)),'k','Linewidth',0.5);
        plot3(zciclola((2*NP+1),:),yciclola((2*NP+1),:),xciclola↙
((2*NP+1),:),'k','Linewidth',0.5);
        plot3(zciclola((2*NP+2),:),yciclola((2*NP+2),:),xciclola↙
((2*NP+2),:),'k','Linewidth',0.5);
        plot3(zciclola(2*(2*NP+1),:),yciclola(2*(2*NP+1),:),xciclola(2*↙
(2*NP+1),:),'k','Linewidth',0.5);
        plot3(zciclola((3*NP+2),:),yciclola((3*NP+2),:),xciclola↙
((3*NP+2),:),'b','Linewidth',0.5);
        axis equal
        plot3(z(1),y(1),x(1),'-go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .↙
63],'MarkerSize',7)
        plot3(z(size(z,2)),y(size(y,2)),x(size(x,2)),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',5)
        plot3(zta(nmeio),yta(nmeio),xta(nmeio),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',5)
        plot3(zta(nfimrest),yta(nfimrest),xta(nfimrest),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',5)
        plot3(zta(size(zta,2)),yta(size(yta,2)),xta(size(xta,2)),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',5)
        plot3(z2(size(z2,2)),y2(size(y2,2)),x2(size(x2,2)),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',5)
        plot3(zta2(nmeio2),yta2(nmeio2),xta2(nmeio2),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',5)
        plot3(zta2(nfimrest2),yta2(nfimrest2),xta2(nfimrest2),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',5)
        plot3(zta2(size(zta2,2)),yta2(size(yta2,2)),xta2(size(xta2,2)),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',7)
        %title('Mandril e trajectória no mandril do primeiro ciclo')
        title('Graphic representation of the tow along the first cycle')
        xlabel('z / mm')
        ylabel('y / mm')
        zlabel('x / mm')
        set(gcf,'color','w');
        grid on;



        figure ('Name','Parameter variation along 1↙
cycle','NumberTitle','off','Position',posfig);

        %Alfa em funcao de z
        subplot(2,2,1)
        plot(zciclo(1,:),alfaciclo(1,:))
```

```
title('Variation of \alpha related to z')
xlabel('z / mm')
ylabel('\alpha')
set(gcf,'color','w');
grid on;
%set(gcf,'GridAlpha',0.12);

if rim==rfm
%dalfadteta em funcao de teta
subplot(2,2,2)
plot(tetaciclo(1,:),dalfadtetaciclo(1,:))
title('Variation of d\alpha/d\theta related to the meridian \theta')
xlabel('\theta')
ylabel('d\alpha/d\theta')
set(gcf,'color','w');
grid on;
%set(gcf,'GridAlpha',0.1);
else
%dalfadro em funcao de ro
subplot(2,2,2)
plot(rociclo(1,:),dalfadrociclo(1,:))
title('Variation of d\alpha/d\rho related to the meridian \rho')
xlabel('\rho / mm')
ylabel('d\alpha/d\rho')
set(gcf,'color','w');
grid on;
end

%Phi em funcao de z
subplot(2,2,3)
plot(zciclo(1,:),ficiclo(1,:))
title('Variation of \phi related to z')
xlabel('z / mm')
ylabel('\phi')
set(gcf,'color','w');
grid on;
%set(gcf,'GridAlpha',0.07);

%Comprimento da trajectoria (Lf) em funcao de z
subplot(2,2,4)
plot(zciclo(1,:),lfciclo(1,:))
title('Variation of Lf related to z')
xlabel('z / mm')
ylabel('Lf / mm')
set(gcf,'color','w');
grid on;
%set(gcf,'GridAlpha',0.05);


%↙
----------------------------------------------------------------------------------------↙
-----------------------
%↙
```

```
-----------------------------------------------------------------------------------------↙
-----------------------
%↙
-----------------------------------------------------------------------------------------↙
-----------------------




%% Próximos ciclos
%dficiclo=max(ficiclo(1,:))-(ceil(max(ficiclo(1,:))/(2*pi))*2*pi);


figure ('Name','Fibre trajectories in the first layer','NumberTitle','off','Position',↙
posfig);
surf(Z,X,Y,'FaceColor','c','FaceAlpha',.7,'EdgeColor','none','LineStyle','none')
hold on;
plot3(zciclola(NP+1,:),yciclola(NP+1,:),xciclola(NP+1,:),'k');
axis equal
plot3(zciclola(NP+1,1),yciclola(NP+1,1),xciclola(NP+1,1),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',7)
title('Trajectories in the first layer')
xlabel('z /mm')
ylabel('y /mm')
zlabel('x /mm')
set(gcf,'color','w');

dficiclo=ficiclo(1,size(ficiclo,2));

for i=2:n1
    ficiclo(i,:)=ficiclo(i-1,:)+dficiclo;
    rociclo(i,:)=rociclo(1,:);

    xciclo(i,:)=rociclo(1,:).*cos(ficiclo(i,:));
    yciclo(i,:)=rociclo(1,:).*sin(ficiclo(i,:));
    zciclo(i,:)=zciclo(1,:);


    %surf(zciclola,yciclola,↙
xciclola,'FaceColor','r','EdgeColor','none','LineStyle','none')
    %    plot3(zciclo2(i,:),yciclo2(i,:),xciclo2(i,:),'k','Linewidth',0.5);
    %    plot3(zciclo3(i,:),yciclo3(i,:),xciclo3(i,:),'k','Linewidth',0.5);
    %    plot3(zciclol(6,:),yciclol(6,:),xciclol(6,:),'k','Linewidth',0.5);
    %    plot3(zciclol(4,:),yciclol(4,:),xciclol(4,:),'k','Linewidth',0.5);
    plot3(zciclo(i,:),yciclo(i,:),xciclo(i,:),'r')
    plot3(zciclo(i,1),yciclo(i,1),xciclo(i,1),'-↙
go','MarkerEdgeColor','k','MarkerFaceColor',[.49 1 .63],'MarkerSize',7)
end


%% Ficheiros exportados
gravartrajectorias=lower(gravartrajectorias);
if gravartrajectorias=='s'

pathname=fullfile(filepath,'\Resultados');
```

```
mkdir(pathname,nomeoutput)
nomeoutputgeo=[nomeoutput 'geopath.txt'];
nomeoutputmaq=[nomeoutput 'maqpath.txt'];
geopathname=fullfile(pathname,'\',nomeoutput,nomeoutputgeo);
maqpathname=fullfile(pathname,'\',nomeoutput,nomeoutputmaq);

path=zeros(n1*size(xciclo,2),3);
maq=zeros(n1*size(xciclo,2),4);
for j=1:n1
    for k=1:size(xciclo,2)
        path(k+((j-1)*size(xciclo,2)),1)=xciclo(j,k);
        path(k+((j-1)*size(xciclo,2)),2)=yciclo(j,k);
        path(k+((j-1)*size(xciclo,2)),3)=zciclo(j,k);

        maq(k+((j-1)*size(xciclo,2)),1)=ficiclo(j,k);
        maq(k+((j-1)*size(xciclo,2)),2)=zciclo(j,k);
        maq(k+((j-1)*size(xciclo,2)),3)=rociclo(j,k);
        maq(k+((j-1)*size(xciclo,2)),4)=alfaciclo(1,k);

    end
end


% Trajectorias cartesianas (x    y    z)
save(geopathname,'path','-ascii','-double','-tabs')

% Trajectorias cilindricas/esfericas (fi    z    ro    alfa)
save(maqpathname,'maq','-ascii','-double','-tabs')
%dlmwrite('maqpath.txt',maq,'delimiter','\t','precision','%.6f')

end


%% CNC

if cnc=='s'

Bcnc=zeros(1,size(zciclo,2));
Bcnc2=zeros(1,size(zciclo,2));
Ycnc=zeros(1,size(zciclo,2)); %Posição do fim da cabeça (onde passa a fibra). Ycnc é
só usado para coord relativas por isso RefY é aplicado no mpf
Xcnc=zeros(1,size(zciclo,2));
Acnc=ficiclo(1,:);

for k=1:size(zciclo,2)

   %Bcnc(1,k)=alfaciclo(1,k);

   ycnc=rociclo(1,k);
   i=0;
   rocnc=rociclo(1,k);

   while ycnc<rocnc+D
       i=i+0.01;
       ycnc=ycnc+i;
```

```
        dxcnc=ycnc/tan(alfaciclo(1,k));
        if rim==rfm
            rocnc=rim;
            ycnc=rocnc+D;
        else
            rocnc=(((zciclo(1,k)+dxcnc)*dm/lm)+rim-(dm*zim/lm));
        end
    end

    Ycnc(1,k)=ycnc;
    Xcnc(1,k)=zciclo(1,k)+dxcnc;
    Bcnc2(1,k)=atan(rociclo(1,k)*tan(alfaciclo(1,k))/(Ycnc(1,k)*cos(asin(rociclo(1,k)↙
/Ycnc(1,k)))));
    if Bcnc2(1,k)<0
        Bcnc2(1,k)=Bcnc2(1,k)+pi;
    end
    Bcnc(1,k)=pi/2-Bcnc2(1,k);
end
Bcnc=Bcnc.*(180/pi);
Acnc=Acnc.*(180/pi);

%Coordenadas relativas
difalfaciclo=zeros(1,size(zciclo,2)-1);
difBcnc=zeros(1,size(zciclo,2)-1);
difYcnc=zeros(1,size(zciclo,2)-1);
difXcnc=zeros(1,size(zciclo,2)-1);
difAcnc=zeros(1,size(zciclo,2)-1);

for k=1:size(zciclo,2)-1
    difalfaciclo(1,k)=alfaciclo(1,k+1)-alfaciclo(1,k);
    difBcnc(1,k)=Bcnc(1,k+1)-Bcnc(1,k);
    difXcnc(1,k)=Xcnc(1,k+1)-Xcnc(1,k);
    difAcnc(1,k)=Acnc(1,k+1)-Acnc(1,k);
    difYcnc(1,k)=Ycnc(1,k+1)-Ycnc(1,k);
end
t1=1:size(difBcnc,2);
t2=1:size(Bcnc,2);

tempo=[];
VXmaximo=[];
VYmaximo=[];
VAmaximo=[];
VBmaximo=[];
FeedMTP=[];
AAreal=[];
ABreal=[];
AXreal=[];
AYreal=[];
aceleA=0;
aceleB=0;
aceleX=0;
aceleY=0;
VXmaximo(1,1)=0;
VYmaximo(1,1)=0;
VAmaximo(1,1)=0;
```

```
VBmaximo(1,1)=0;
i=1;
j=1;
contador=0;
breaker=0;
csb1=0;
while i<(size(difBcnc,2)+1)%for i=2:size(difBcnc,2)+1
    i=i+1;
    j=j+1;

    if breaker==0
    difAcnc1(1,j-1)=difAcnc(1,i-1);
    difBcnc1(1,j-1)=difBcnc(1,i-1);
    difXcnc1(1,j-1)=difXcnc(1,i-1);
    difYcnc1(1,j-1)=difYcnc(1,i-1);
    else
    end
    breaker=0;

    tempo(1,j-1)=difAcnc1(1,j-1)/(VAmax);
    tempo(2,j-1)=difBcnc1(1,j-1)/(VBmax);
    tempo(3,j-1)=difXcnc1(1,j-1)/(VXmax);
    tempo(4,j-1)=difYcnc1(1,j-1)/(VYmax);
    tempo(5,j-1)=vel*max(tempo(1:4,j-1));

    VXmaximo(1,j)=difXcnc1(1,j-1)/tempo(5,j-1);
    VYmaximo(1,j)=difYcnc1(1,j-1)/tempo(5,j-1);
    VAmaximo(1,j)=difAcnc1(1,j-1)/tempo(5,j-1);
    VBmaximo(1,j)=difBcnc1(1,j-1)/tempo(5,j-1);

    %Menor tempo de produção (velocidades máximas sempre)
    FeedMTP(1,j-1)=((difAcnc1(1,j-1)^2+difBcnc1(1,j-1)^2+difXcnc1(1,j-1)^2+difYcnc1(1,↙
j-1)^2)^0.5)/(tempo(5,j-1)/60);

    %Acelerações
    AAreal(1,j)=abs(VAmaximo(1,j)-VAmaximo(1,j-1));
    AA1=VAmaximo(1,j)-VAmaximo(1,j-1);
    ABreal(1,j)=abs(VBmaximo(1,j)-VBmaximo(1,j-1));
    AB1=VBmaximo(1,j)-VBmaximo(1,j-1);
    AXreal(1,j)=abs(VXmaximo(1,j)-VXmaximo(1,j-1));
    AX1=VXmaximo(1,j)-VXmaximo(1,j-1);
    AYreal(1,j)=abs(VYmaximo(1,j)-VYmaximo(1,j-1));
    AY1=VYmaximo(1,j)-VYmaximo(1,j-1);

    %AA1=1;
    %AB1=1;
    %AX1=1;
    %AY1=1;

    csa=1;
    csb=1;
    if difAcnc1(1,j-1)==0 && difXcnc1(1,j-1)==0 && difYcnc1(1,j-1)==0  && difBcnc1(1,↙
j-1)==0
        contador=contador+1;
        difAcnc1(1,j-1)=difAcnc1(1,j-1)+difAcnc(1,i);
```

```
        difBcnc1(1,j-1)=difBcnc1(1,j-1)+difBcnc(1,i);
        difXcnc1(1,j-1)=difXcnc1(1,j-1)+difXcnc(1,i);
        difYcnc1(1,j-1)=difYcnc1(1,j-1)+difYcnc(1,i);
        breaker=1;
        j=j-1;
    end
end
%Parar maquina
j=j+1;
VXmaximo(1,j)=0;
VYmaximo(1,j)=0;
VAmaximo(1,j)=0;
VBmaximo(1,j)=0;
AAreal(1,j)=abs(VAmaximo(1,j)-VAmaximo(1,j-1));
AA1=VAmaximo(1,j)-VAmaximo(1,j-1);
ABreal(1,j)=abs(VBmaximo(1,j)-VBmaximo(1,j-1));
AXreal(1,j)=abs(VXmaximo(1,j)-VXmaximo(1,j-1));
AYreal(1,j)=abs(VYmaximo(1,j)-VYmaximo(1,j-1));

%Desaceleração
pAA=ceil(AAreal(1,j)/AAmax);

            %passos novos
            difAcncpasso=difAcnc1(1,j-2)/pAA;
            difBcncpasso=difBcnc1(1,j-2)/pAA;
            difXcncpasso=difXcnc1(1,j-2)/pAA;
            difYcncpasso=difYcnc1(1,j-2)/pAA;
            AApasso=AAreal(1,j)/pAA;
            poop2=j;
for p=1:pAA

                difAcnc1(1,j-2)=difAcncpasso;
                difBcnc1(1,j-2)=difBcncpasso;
                difXcnc1(1,j-2)=difXcncpasso;
                difYcnc1(1,j-2)=difYcncpasso;

                if AA1<0
                    VAmaximo(1,j)=VAmaximo(1,j-1)-AApasso;
                    tempope=difAcncpasso/VAmaximo(1,j);
                    if p==pAA
                        break
                    end
                    tempo(5,j-2)=tempope;
                    VBmaximo(1,j)=difBcncpasso/tempope;
                    VXmaximo(1,j)=difXcncpasso/tempope;
                    VYmaximo(1,j)=difYcncpasso/tempope;

                else
                    VAmaximo(1,j)=VAmaximo(1,j-1)+AApasso;
                    tempope=difAcncpasso/VAmaximo(1,j);
                    if p==pAA
                        break
                    end
                    tempo(5,j-2)=tempope;
                    VBmaximo(1,j)=difBcncpasso/tempope;
```

```
                    VXmaximo(1,j)=difXcncpasso/tempope;
                    VYmaximo(1,j)=difYcncpasso/tempope;

                end
                AAreal(1,j)=abs(VAmaximo(1,j)-VAmaximo(1,j-1));
                ABreal(1,j)=abs(VBmaximo(1,j)-VBmaximo(1,j-1));
                AXreal(1,j)=abs(VXmaximo(1,j)-VXmaximo(1,j-1));
                AYreal(1,j)=abs(VYmaximo(1,j)-VYmaximo(1,j-1));
                FeedMTP(1,j-2)=(difAcnc1(1,j-2)^2+difBcnc1(1,j-2)^2+difXcnc1(1,j-2)↙
^2+difYcnc1(1,j-2)^2)^0.5/(tempo(5,j-2)/60);
                if p<pAA
                    j=j+1;
                end
end
            tempo(5,j-1)=0;

%Verificação
aceleA=0;
aceleAa=[];
a=0;
aceleB=0;
aceleBa=[];
btt=0;
aceleX=0;
aceleXa=[];
x=0;
aceleY=0;
aceleYa=[];
y=0;
velociA=0;
velociB=0;
velociX=0;
velociY=0;

ACNC=zeros(1,size(difAcnc1,2)+1);
BCNC=zeros(1,size(difAcnc1,2)+1);
XCNC=zeros(1,size(difAcnc1,2)+1);
YCNC=zeros(1,size(difAcnc1,2)+1);
ACNC(1,1)=Acnc(1,1);
BCNC(1,1)=Bcnc(1,1);
XCNC(1,1)=Xcnc(1,1);
YCNC(1,1)=Ycnc(1,1);

velocicheckA=zeros(1,size(difAcnc1,2)+1);
velocicheckB=zeros(1,size(difAcnc1,2)+1);
velocicheckX=zeros(1,size(difAcnc1,2)+1);
velocicheckY=zeros(1,size(difAcnc1,2)+1);
velocicheckA(1,1)=0;
velocicheckB(1,1)=0;
velocicheckX(1,1)=0;
velocicheckY(1,1)=0;

acelecheckA=zeros(1,size(difAcnc1,2));
acelecheckB=zeros(1,size(difAcnc1,2));
acelecheckX=zeros(1,size(difAcnc1,2));
```

```
acelecheckY=zeros(1,size(difAcnc1,2));

tempoplot=zeros(1,size(difAcnc1,2)+1);
tempoplot(1,1)=0;

for j=2:size(difAcnc1,2)+1
    ACNC(1,j)=ACNC(1,j-1)+difAcnc1(1,j-1);
    BCNC(1,j)=BCNC(1,j-1)+difBcnc1(1,j-1);
    XCNC(1,j)=XCNC(1,j-1)+difXcnc1(1,j-1);
    YCNC(1,j)=YCNC(1,j-1)+difYcnc1(1,j-1);
    tempoplot(1,j)=tempoplot(1,j-1)+tempo(5,j-1);
    velocicheckA(1,j)=difAcnc1(1,j-1)/tempo(5,j-1);
    velocicheckB(1,j)=difBcnc1(1,j-1)/tempo(5,j-1);
    velocicheckX(1,j)=difXcnc1(1,j-1)/tempo(5,j-1);
    velocicheckY(1,j)=difYcnc1(1,j-1)/tempo(5,j-1);
    acelecheckA(1,j-1)=abs(velocicheckA(1,j)-velocicheckA(1,j-1));
    acelecheckB(1,j-1)=abs(velocicheckB(1,j)-velocicheckB(1,j-1));
    acelecheckX(1,j-1)=abs(velocicheckX(1,j)-velocicheckX(1,j-1));
    acelecheckY(1,j-1)=abs(velocicheckY(1,j)-velocicheckY(1,j-1));
end

j=j+1;
velocicheckA(1,j)=0;
    velocicheckB(1,j)=0;
    velocicheckX(1,j)=0;
    velocicheckY(1,j)=0;
    acelecheckA(1,j-1)=abs(velocicheckA(1,j)-velocicheckA(1,j-1));
    acelecheckB(1,j-1)=abs(velocicheckB(1,j)-velocicheckB(1,j-1));
    acelecheckX(1,j-1)=abs(velocicheckX(1,j)-velocicheckX(1,j-1));
    acelecheckY(1,j-1)=abs(velocicheckY(1,j)-velocicheckY(1,j-1));
    t4=1:size(velocicheckA,2);

t3=1:size(ACNC,2);
for j=1:size(acelecheckA,2)
if acelecheckA(1,j)>AAmax
    aceleA=aceleA+1;
    a=a+1;
    aceleAa(1,a)=j;
end
if acelecheckB(1,j)>ABmax
    aceleB=aceleB+1;
    btt=btt+1;
    aceleBa(1,btt)=j;
end
if acelecheckX(1,j)>AXmax
    aceleX=aceleX+1;
    x=x+1;
    aceleXa(1,x)=j;
end
if acelecheckY(1,j)>AYmax
    aceleY=aceleY+1;
    y=y+1;
    aceleYa(1,y)=j;
end
if VAmaximo(1,j)>VAmax
```

```
    velociA=velociA+1;
end
if VBmaximo(1,j)>VBmax
    velociB=velociB+1;
end
if VXmaximo(1,j)>VXmax
    velociX=velociX+1;
end
if VYmaximo(1,j)>VYmax
    velociY=velociY+1;
end
end



%Fazer os ficheiros
gravarcnc=lower(gravarcnc);
if gravarcnc=='s'

    nomeoutputspf=[nomeoutput 'CNCciclo.spf'];
    nomeoutputspf1=[nomeoutput 'CNCciclo'];
    nomeoutputmpf=[nomeoutput 'CNCmain.mpf'];
    cncspf=fullfile(pathname,'\',nomeoutput,nomeoutputspf);
    cncmpf=fullfile(pathname,'\',nomeoutput,nomeoutputmpf);

    Y0=RefY+rfm+D; %rfm pois é o maior raio do mandril
    Y1=RefY+rociclo(1,1)+D;
    F0=5000;
    X0=Xmin+zi-zim;
% Ficheiro CNC .mpf
fileID = fopen(cncmpf,'w');
fprintf(fileID,'%4s %4s\r\n',';Pathwind','INEGI');
fprintf(fileID,';%s\r\n',datestr(clock));
fprintf(fileID,'M00\r\nN010 G01 G64 G90 G94 Y%.3f F%.3f\r\nN020 X%.3f B%.3f C0↙
Z0\r\nN030 Y%.3f\r\nM00\r\nN040 %s P%d\r\nN050 M30\r\n',Y0,F0,X0,alfainicio,Y1,↙
nomeoutputspf1,n1);
fclose(fileID);

%Feedmax=zeros(1,size(difAcnc,2));
%Feedmax(1,1)=((difAcnc(1,1)^2+difBcnc(1,1)^2+difYcnc(1,1)^2+difYcnc(1,1)^2)^0.5)↙
/tempo(5,1); %tempo/60
% Ficheiro CNC .spf
fileID = fopen(cncspf,'w');
fprintf(fileID,'N010 G01 G91 G64 G94 A%.3f X%.3f Y%.3f B%.3f F%.3f\r\n',difAcnc(1,1),↙
difXcnc(1,1),difYcnc(1,1),difBcnc(1,1),FeedMTP(1,1));
for j=2:size(difAcnc1,2)-1
    %Feedmax(1,j)=((difAcnc(1,j)^2+difBcnc(1,j)^2+difYcnc(1,j)^2+difYcnc(1,j)^2)^0.5)/↙
(tempo(5,j)); %tempo/60?

    if difAcnc1(1,j)==0 && difBcnc1(1,j)==0 && difXcnc1(1,j)==0 && difYcnc1(1,j)==0
    else
        fprintf(fileID,'N0%i0 A%.3f X%.3f Y%.3f B%.3f F%.3f\r\n',j,difAcnc1(1,j),↙
difXcnc1(1,j),difYcnc1(1,j),difBcnc1(1,j),FeedMTP(1,j));
    end
end
```

```
fprintf(fileID,'RET\r\n');
fclose(fileID);

end

figure ('Name','Machine axes movement related to vector↙
position','NumberTitle','off','Position',posfig);

        %B em funcao de t
        subplot(2,2,1)
        plot(t3,BCNC)
        title('Movement of axis B')
        xlabel('')
        ylabel('B / °')
        set(gcf,'color','w');
        grid on;

        %A em funcao de t
        subplot(2,2,2)
        plot(t3,ACNC)
        title('Movement of axis A')
        xlabel('')
        ylabel('A / °')
        set(gcf,'color','w');
        grid on;

        %X em funcao de t
        subplot(2,2,3)
        plot(t3,XCNC)
        title('Movement of axis X')
        xlabel('')
        ylabel('X / mm')
        set(gcf,'color','w');
        grid on;

        %Y em funcao de t
        subplot(2,2,4)
        plot(t3,YCNC);
        title('Movement of axis Y')
        xlabel('')
        ylabel('Y / mm')
        set(gcf,'color','w');
        grid on;


figure ('Name','Machine axis movement related to time','NumberTitle','off','Position',↙
posfig);

        %B em funcao de t
        subplot(2,2,1)
        plot(tempoplot,BCNC)
        title('Movement of axis B')
        xlabel('t / s')
        ylabel('B / °')
        set(gcf,'color','w');
```

```
        grid on;

        %A em funcao de t
        subplot(2,2,2)
        plot(tempoplot,ACNC)
        title('Movement of axis A')
        xlabel('t / s')
        ylabel('A / °')
        set(gcf,'color','w');
        grid on;

        %X em funcao de t
        subplot(2,2,3)
        plot(tempoplot,XCNC)
        title('Movement of axis X')
        xlabel('t / s')
        ylabel('X / mm')
        set(gcf,'color','w');
        grid on;

        %Y em funcao de t
        subplot(2,2,4)
        plot(tempoplot,YCNC);
        title('Movement of axis Y')
        xlabel('t / s')
        ylabel('Y / mm')
        set(gcf,'color','w');
        grid on;

        figure ('Name','Velocidade dos eixos da máquina em função do↙
tempo','NumberTitle','off','Position',posfig);
        tempoplot1=zeros(1,size(tempoplot,2)+1);
        tempoplot1(1,1:size(tempoplot,2))=tempoplot(1,:);
        tempoplot1(1,size(tempoplot1,2))=tempoplot1(1,size(tempoplot1,2)-1)+0.0001;

        %B em funcao de t
        subplot(2,2,1)
        plot(tempoplot1,velocicheckB)
        title('Velocity of axis B')
        xlabel('t / s')
        ylabel('VB / °/s')
        set(gcf,'color','w');
        grid on;

        %A em funcao de t
        subplot(2,2,2)
        plot(tempoplot1,velocicheckA)
        title('Velocity of axis A')
        xlabel('t / s')
        ylabel('VA / °/s')
        set(gcf,'color','w');
        grid on;

        %X em funcao de t
        subplot(2,2,3)
```

```
        plot(tempoplot1,velocicheckX)
        title('Velocity of axis X')
        xlabel('t / s')
        ylabel('VX / mm/s')
        set(gcf,'color','w');
        grid on;

        %Y em funcao de t
        subplot(2,2,4)
        plot(tempoplot1,velocicheckY);
        title('Velocity of axis Y')
        xlabel('t / s')
        ylabel('VY / mm/s')
        set(gcf,'color','w');
        grid on;

end



%% Sobreposições
switch lower(sobre)

        case {'sobreposiçao3','sobreposição3'}
            disp('Sobreposição')

%criar matriz para ordenar por z
zsobre=zeros(size(zciclo,2)*size(zciclo,1),1);
alfasobre=zeros(size(zciclo,2)*size(zciclo,1),1);
fisobre=zeros(size(zciclo,2)*size(zciclo,1),1);
rosobre=zeros(size(zciclo,2)*size(zciclo,1),1);
xsobre=zeros(size(zciclo,2)*size(zciclo,1),1);
ysobre=zeros(size(zciclo,2)*size(zciclo,1),1);
numeracaomatriz=zeros(size(zciclo,2)*size(zciclo,1),1);
contsobre=0;

for j=1:size(zciclo,1)
    for i=1:size(zciclo,2)
        contsobre=contsobre+1;
       zsobre(contsobre,1)=zciclo(j,i);
       alfasobre(contsobre,1)=alfaciclo(1,i);
       fisobre(contsobre,1)=ficiclo(j,i);
       rosobre(contsobre,1)=rociclo(j,i);
       xsobre(contsobre,1)=xciclo(j,i);
       ysobre(contsobre,1)=yciclo(j,i);
       numeracaomatriz(contsobre,1)=contsobre;
    end
end

%Novo vector fi - errado!!!!!
finovo1=zeros(size(fisobre));
for i=1:size(fisobre,1)
   finovo1(i,1)=fisobre(i,1);
   while finovo1(i,1)>2*pi
       finovo1(i,1)=finovo1(i,1)-2*pi;
```

```
    end
end

%matriz com todos os ciclos
matrizsobre=[zsobre numeracaomatriz finovo1 rosobre alfasobre xsobre ysobre];

%matriz ordenada
matrizsobreordenada=sortrows(matrizsobre);

case{'coco'}
%restantes paths para a matriz ordenada
dfilargurasobre(:,1)=(b./(2.*rosobre(:,1))).*(cos(alfasobre(:,1)));
dzlargurasobre(:,1)=(b/2).*sin(alfasobre(:,1)).*cos(tau);

fisobrela=zeros(2*NP+1,size(fisobre,1));
zsobrela=zeros(4*NP+2,size(fisobre,1));
rosobrela=zeros(2*NP+1,size(fisobre,1));

for i=1:(2*NP+1)
    fisobrela(i,:)=finovo1(1,:)-dfilargura(1,:)+(i-1)*(dfilargura(1,:)/NP);
    zsobrela(i,:)=zciclo(1,:)+dzlargura(1,:)-(i-1)*(dzlargura(1,:)/NP);
    rosobrela(i,:)=((zciclola(i,:)*dm/lm)+rim-(dm*zim/lm));
end

firealla=zeros(size(ficiclo,1),(2*NP+1));
        for l=1:(2*NP+1)
            firealla(:,l)=fireal-dfilargura(1,k)+(l-1)*(dfilargura(1,k)/NP);
        end

        %T=[cos(alfaciclo(1,i)) sin(alfaciclo(1,i));-sin(alfaciclo(1,i)) cos(alfaciclo↙
(1,i))];


for k=1:size(zsobre,1)
    i=k-1;
    if i>0 && i<=size(zsobre,1)
    while abs(matrizsobreordenada(i,1)-matrizsobreordenada(k,1))/cos(tau)<b
        dfi=abs(matrizsobreordenada(i,2)-matrizsobreordenada(k,2));
        if dfi>pi
            dfi=2*pi-dfi;
        end

        if dfi<(b/matrizsobreordenada(k,3))

        end

        i=i-1;
        if i<=0
            break
        end
        if i>size(zsobre,1)
            break
        end
    end
    end
```

```
    j=k+1;
    if j>0 && j<=size(zsobre,1)
    while abs(matrizsobreordenada(j,1)-matrizsobreordenada(k,1))/cos(tau)<b
        dfi=abs(matrizsobreordenada(j,2)-matrizsobreordenada(k,2));
        if dfi>pi
            dfi=2*pi-dfi;
        end


        j=j+1;
        if j<=0
            break
        end
        if j>size(zsobre,1)
            break
        end
    end
    end
end




    case {'sobreposiçao1','sobreposiçao2','sobreposição1','sobreposição2'}
        if lower(sobre)=='sobreposiçao1'
        disp('Sobreposição do primeiro ciclo')
        else
        disp('Sobreposição')
        end


%Novo vector fi
finovo1=zeros(n1,size(ficiclo,2));
for i=1:size(ficiclo,2)
   finovo1(1,i)=ficiclo(1,i);
   while finovo1(1,i)>2*pi
       finovo1(1,i)=finovo1(1,i)-2*pi;
   end
end

% Sobreposição
for k=1:size(zciclo,2)
    i=1;
    while ficiclo(1,i)<(ficiclo(1,k)-2*pi)
        if abs(zciclo(1,i)-zciclo(1,k))/cos(tau)<b
        if abs(finovo1(1,i)-finovo1(1,k))<(b/rociclo(1,i))

        finovo=ficiclo(1,k);
        fireal=ficiclo(1,k);
        dfireal=abs(ficiclo(1,i)-finovo);

        while finovo>0
            finovo=finovo-2*pi;
```

```
            dfi1=abs(ficiclo(1,i)-finovo);
            if dfi1<dfireal
                fireal=finovo;
                dfireal=dfi1;
            end
        end

        firealla=zeros((2*NP+1),1);
        for l=1:(2*NP+1)
            firealla(l,1)=fireal-dfilargura(1,k)+(l-1)*(dfilargura(1,k)/NP);
        end

        T=[cos(alfaciclo(1,i)) sin(alfaciclo(1,i));-sin(alfaciclo(1,i)) cos(alfaciclo↙
(1,i))];


        % Multiplas linhas
        for l=1:(2*NP+1)
            for m=1:(2*NP+1)
                if rociclola(l,k)>rociclola(m,i)
                    distla(m,l)=(((zciclola(l,k)-zciclola(m,i))/cos(tau))^2+((firealla↙
(l,1)-ficiclola(m,i))*rociclola(l,k))^2)^0.5;
                    distlav(m,l,:)=[(zciclola(l,k)-zciclola(m,i))/cos(tau) (firealla↙
(l,1)-ficiclola(m,i))*rociclola(l,k)];
                    distlav1=[(zciclola(l,k)-zciclola(m,i))/cos(tau); (firealla(l,1)-↙
ficiclola(m,i))*rociclola(l,k)];
                else
                    distla(m,l)=(((zciclola(l,k)-zciclola(m,i))/cos(tau))^2+((firealla↙
(l,1)-ficiclola(m,i))*rociclola(m,i))^2)^0.5;
                    distlav(m,l,:)=[(zciclola(l,k)-zciclola(m,i))/cos(tau) (firealla↙
(l,1)-ficiclola(m,i))*rociclola(m,i)];
                    distlav1=[(zciclola(l,k)-zciclola(m,i))/cos(tau); (firealla(l,1)-↙
ficiclola(m,i))*rociclola(m,i)];
                end
                distlava1=T*distlav1;
                distlava(m,l)=distlava1(2,1);
            end
        end


        for l=1:(2*NP+1)
         if abs(distla(1,l))<b && abs(distla(2*NP+1,l))<b

            minimo=abs(distlava(1,l));
            posi=1;
            for m=1:(2*NP+1)
                minimo1=abs(distlava(m,l));
                if minimo1<minimo
                    minimo=minimo1;
                    posi=m;
                end
            end


            if posi~=2*NP+1 && posi~=1
```

```
                if abs(distlava(posi-1,l))<=abs(distlava(posi+1,l))
                    rociclola(l,k)=((abs(distlava(posi-1,l))*rociclola(posi,i)+abs↙
(distlava(posi,l))*rociclola(posi-1,i))/(abs(distlava(posi-1,l))+abs(distlava(posi,↙
l))))+(esp/cos(tau));
                    zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                else
                    rociclola(l,k)=((abs(distlava(posi+1,l))*rociclola(posi,i)+abs↙
(distlava(posi,l))*rociclola(posi+1,i))/(abs(distlava(posi+1,l))+abs(distlava(posi,↙
l))))+(esp/cos(tau));
                    zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                end
            else
                if posi==2*NP+1
                    rociclola(l,k)=((abs(distlava(posi-1,l))*rociclola(posi,i)+abs↙
(distlava(posi,l))*rociclola(posi-1,i))/(abs(distlava(posi-1,l))+abs(distlava(posi,↙
l))))+(esp/cos(tau));
                    zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                else
                    rociclola(l,k)=((abs(distlava(posi+1,l))*rociclola(posi,i)+abs↙
(distlava(posi,l))*rociclola(posi+1,i))/(abs(distlava(posi+1,l))+abs(distlava(posi,↙
l))))+(esp/cos(tau));
                    zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                end
            end

        end
        end

    else
        if abs(finovo1(1,i)-finovo1(1,k))>(2*pi-(b/rociclo(1,i)))

            finovo=ficiclo(1,k);
        fireal=ficiclo(1,k);
        dfireal=abs(ficiclo(1,i)-finovo);

        while finovo>0
            finovo=finovo-2*pi;
            dfi1=abs(ficiclo(1,i)-finovo);
            if dfi1<dfireal
                fireal=finovo;
                dfireal=dfi1;
            end
        end

        firealla=zeros((2*NP+1),1);
        for l=1:(2*NP+1)
            firealla(l,1)=fireal-dfilargura(1,k)+(l-1)*(dfilargura(1,k)/NP);
        end

        T=[cos(alfaciclo(1,i)) sin(alfaciclo(1,i));-sin(alfaciclo(1,i)) cos(alfaciclo↙
(1,i))];


        % Multiplas linhas
        for l=1:(2*NP+1)
```

```
            for m=1:(2*NP+1)

                if rociclola(l,k)>rociclola(m,i)
                    distla(m,l)=(((zciclola(l,k)-zciclola(m,i))/cos(tau))^2+((firealla↙
(l,1)-ficiclola(m,i))*rociclola(l,k))^2)^0.5;
                    distlav(m,l,:)=[(zciclola(l,k)-zciclola(m,i))/cos(tau) (firealla↙
(l,1)-ficiclola(m,i))*rociclola(l,k)];
                    distlav1=[(zciclola(l,k)-zciclola(m,i))/cos(tau); (firealla(l,1)-↙
ficiclola(m,i))*rociclola(l,k)];
                else
                    distla(m,l)=(((zciclola(l,k)-zciclola(m,i))/cos(tau))^2+((firealla↙
(l,1)-ficiclola(m,i))*rociclola(m,i))^2)^0.5;
                    distlav(m,l,:)=[(zciclola(l,k)-zciclola(m,i))/cos(tau) (firealla↙
(l,1)-ficiclola(m,i))*rociclola(m,i)];
                    distlav1=[(zciclola(l,k)-zciclola(m,i))/cos(tau); (firealla(l,1)-↙
ficiclola(m,i))*rociclola(m,i)];
                end
                distlava1=T*distlav1;
                distlava(m,l)=distlava1(2,1);
            end
        end


        for l=1:(2*NP+1)
         if abs(distla(1,l))<b && abs(distla(2*NP+1,l))<b

            minimo=abs(distlava(1,l));
            posi=1;
            for m=1:(2*NP+1)
                minimo1=abs(distlava(m,l));
                if minimo1<minimo
                    minimo=minimo1;
                    posi=m;
                end
            end


            if posi~=2*NP+1 && posi~=1
                if abs(distlava(posi-1,l))<=abs(distlava(posi+1,l))
                    rociclola(l,k)=((abs(distlava(posi-1,l))*rociclola(posi,i)+abs↙
(distlava(posi,l))*rociclola(posi-1,i))/(abs(distlava(posi-1,l))+abs(distlava(posi,↙
l))))+(esp/cos(tau));
                    zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                else
                    rociclola(l,k)=((abs(distlava(posi+1,l))*rociclola(posi,i)+abs↙
(distlava(posi,l))*rociclola(posi+1,i))/(abs(distlava(posi+1,l))+abs(distlava(posi,↙
l))))+(esp/cos(tau));
                    zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                end
            else
                if posi==2*NP+1
                    rociclola(l,k)=((abs(distlava(posi-1,l))*rociclola(posi,i)+abs↙
(distlava(posi,l))*rociclola(posi-1,i))/(abs(distlava(posi-1,l))+abs(distlava(posi,↙
l))))+(esp/cos(tau));
                    zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
```

```
                else
                    rociclola(l,k)=((abs(distlava(posi+1,l))*rociclola(posi,i)+abs↙
(distlava(posi,l))*rociclola(posi+1,i))/(abs(distlava(posi+1,l))+abs(distlava(posi,↙
l))))+(esp/cos(tau));
                    zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                end
            end

        end
        end

        end
        end
        end
    i=i+1;
    end
end

% Cartesianas multiplas linhas
xciclola=zeros(4*NP+2,size(ficiclola,2));
yciclola=zeros(4*NP+2,size(ficiclola,2));
for n=1:2*NP+1
   xciclola(n,:)=rociclola(n,:).*cos(ficiclola(n,:));
   yciclola(n,:)=rociclola(n,:).*sin(ficiclola(n,:));

end
c=1;
for n=2*NP+2:2*(2*NP+1)
    xciclola(n,:)=(rociclola((2*NP+2-c),:)+esp/cos(tau)).*cos(ficiclola((2*NP+2-↙
c),:));
    yciclola(n,:)=(rociclola((2*NP+2-c),:)+esp/cos(tau)).*sin(ficiclola((2*NP+2-↙
c),:));
    zciclola(n,:)=zciclola((2*NP+2-c),:)-esp*sin(tau);
    c=c+1;
end

        rociclolae=[];
        ficiclolae=[];
        zciclolae=[];
        rociclolae(1,:,:)=rociclola(:,:);
        ficiclolae(1,:,:)=ficiclola(:,:);
        zciclolae(1,:,:)=zciclola(:,:);

        %Plot Mandril e trajectoria com multiplas linhas
        figure ('Name','Primeira camada','NumberTitle','off','Position',posfig);
        surf(Z,X,Y,'FaceColor','c','FaceAlpha',.↙
7,'EdgeColor','none','LineStyle','none')
        hold on;
        surf(zciclola(:,:),yciclola(:,:),xciclola↙
(:,:),'FaceColor','r','EdgeColor','none','LineStyle','none')
        %hold on;
        %plot3(zciclo(1,:),xciclo(1,:),yciclo(1,:),'r','Linewidth',2);
        plot3(zciclola(1,:),yciclola(1,:),xciclola(1,:),'k','Linewidth',0.5);
        plot3(zciclola((2*NP+1),:),yciclola((2*NP+1),:),xciclola↙
((2*NP+1),:),'k','Linewidth',0.5);
```

```
        plot3(zciclola((2*NP+2),:),yciclola((2*NP+2),:),xciclola↙
((2*NP+2),:),'k','Linewidth',0.5);
        plot3(zciclola(2*(2*NP+1),:),yciclola(2*(2*NP+1),:),xciclola(2*↙
(2*NP+1),:),'k','Linewidth',0.5);
        plot3(zciclola((3*NP+2),:),yciclola((3*NP+2),:),xciclola↙
((3*NP+2),:),'b','Linewidth',0.5);
        axis equal
        title('Primeira camada')
        xlabel('z')
        ylabel('y')
        zlabel('x')
        set(gcf,'color','w');
        grid on;

    switch  lower(sobre)
        case {'sobreposiçao2','sobreposição2'}
        disp('Sobreposição de todos os ciclos primeira camada')

        % Todos os ciclos
        for o=2:n1
            % pegar nas trajectórias já geradas e adicionar a largura/espessura
            % (isto pode ser lá em cima, só não é representado)
            %verificar se tem paths por baixo

            % Restantes Paths - NP
            ficiclola=zeros(2*NP+1,size(ficiclo,2));
            zciclola=zeros(4*NP+2,size(ficiclola,2));
            rociclola=zeros(2*NP+1,size(ficiclo,2));

            for i=1:(2*NP+1)
                ficiclola(i,:)=ficiclo(o,:)-dfilargura(1,:)+(i-1)*(dfilargura(1,:)↙
/NP);
                zciclola(i,:)=zciclo(o,:)+dzlargura(1,:)-(i-1)*(dzlargura(1,:)/NP);
                rociclola(i,:)=((zciclola(i,:)*dm/lm)+rim-(dm*zim/lm));
            end


            %Sobreposição ciclos anteriores
            %Novo vector fi
            for i=1:size(ficiclo,2)
                finovo1(o,i)=ficiclo(o,i);
                while finovo1(o,i)>2*pi
                    finovo1(o,i)=finovo1(o,i)-2*pi;
                end
            end

            for g=1:o-1

                for k=1:size(zciclo,2) %ciclo 'o'
                    %i=1;
                    for i=1:size(zciclo,2)  %while ficiclo(1,i)<(ficiclo(1,k)-2*pi) %↙
ciclo 'g'
                        if abs(zciclo(g,i)-zciclo(o,k))/cos(tau)<b
                            if abs(finovo1(g,i)-finovo1(o,k))<(b/rociclolae(g,NP+1,i))
```

```
                                finovo=ficiclo(o,k);
                                fireal=ficiclo(o,k);
                                dfireal=abs(ficiclo(g,i)-finovo);

                                while finovo>0
                                    finovo=finovo-2*pi;
                                    dfi1=abs(ficiclo(g,i)-finovo);
                                    if dfi1<dfireal
                                        fireal=finovo;
                                        dfireal=dfi1;
                                    end
                                end

                                firealla=zeros((2*NP+1),1);
                                for l=1:(2*NP+1)
                                    firealla(l,1)=fireal-dfilargura(1,k)+(l-1)*↙
(dfilargura(1,k)/NP);
                                end

                                T=[cos(alfaciclo(1,i)) sin(alfaciclo(1,i));-sin↙
(alfaciclo(1,i)) cos(alfaciclo(1,i))];


                                % Multiplas linhas
                                for l=1:(2*NP+1)
                                    for m=1:(2*NP+1)

                                        if rociclola(l,k)>rociclolae(g,m,i)
                                            distla(m,l)=(((zciclola(l,k)-zciclolae(g,↙
m,i))/cos(tau))^2+((firealla(l,1)-ficiclolae(g,m,i))*rociclola(l,k))^2)^0.5;
                                            distlav(m,l,:)=[(zciclola(l,k)-zciclolae↙
(g,m,i))/cos(tau) (firealla(l,1)-ficiclolae(g,m,i))*rociclola(l,k)];
                                            distlav1=[(zciclola(l,k)-zciclolae(g,m,i))↙
/cos(tau); (firealla(l,1)-ficiclolae(g,m,i))*rociclola(l,k)];
                                        else
                                            distla(m,l)=(((zciclola(l,k)-zciclolae(g,↙
m,i))/cos(tau))^2+((firealla(l,1)-ficiclolae(g,m,i))*rociclolae(g,m,i))^2)^0.5;
                                            distlav(m,l,:)=[(zciclola(l,k)-zciclolae↙
(g,m,i))/cos(tau) (firealla(l,1)-ficiclolae(g,m,i))*rociclolae(g,m,i)];
                                            distlav1=[(zciclola(l,k)-zciclolae(g,m,i))↙
/cos(tau); (firealla(l,1)-ficiclolae(g,m,i))*rociclolae(g,m,i)];
                                        end
                                        distlava1=T*distlav1;
                                        distlava(m,l)=distlava1(2,1);
                                    end
                                end


                                for l=1:(2*NP+1)
                                    if abs(distla(1,l))<b && abs(distla(2*NP+1,l))<b

                                        minimo=abs(distlava(1,l));
                                        posi=1;
                                        for m=1:(2*NP+1)
                                            minimo1=abs(distlava(m,l));
```

```
                                        if minimo1<minimo
                                            minimo=minimo1;
                                            posi=m;
                                        end
                                    end


                                if posi~=2*NP+1 && posi~=1
                                    if abs(distlava(posi-1,l))<=abs(distlava↙
(posi+1,l))
                                        rociclola(l,k)=((abs(distlava(posi-1,l))↙
*rociclolae(g,posi,i)+abs(distlava(posi,l))*rociclolae(g,posi-1,i))/(abs(distlava↙
(posi-1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                        zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                    else
                                        rociclola(l,k)=((abs(distlava(posi+1,l))↙
*rociclolae(g,posi,i)+abs(distlava(posi,l))*rociclolae(g,posi+1,i))/(abs(distlava↙
(posi+1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                        zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                    end
                                else
                                    if posi==2*NP+1
                                        rociclola(l,k)=((abs(distlava(posi-1,l))↙
*rociclolae(g,posi,i)+abs(distlava(posi,l))*rociclolae(g,posi-1,i))/(abs(distlava↙
(posi-1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                        zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                    else
                                        rociclola(l,k)=((abs(distlava(posi+1,l))↙
*rociclolae(g,posi,i)+abs(distlava(posi,l))*rociclolae(g,posi+1,i))/(abs(distlava↙
(posi+1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                        zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                    end
                                end

                                end
                            end

                        else
                            if abs(finovo1(g,i)-finovo1(o,k))>(2*pi-(b/rociclolae↙
(g,1,i)))

                                finovo=ficiclo(o,k);
                            fireal=ficiclo(o,k);
                            dfireal=abs(ficiclo(g,i)-finovo);

                            while finovo>0
                                finovo=finovo-2*pi;
                                dfi1=abs(ficiclo(g,i)-finovo);
                                if dfi1<dfireal
                                    fireal=finovo;
                                    dfireal=dfi1;
                                end
                            end

                            firealla=zeros((2*NP+1),1);
```

```
                                for l=1:(2*NP+1)
                                    firealla(l,1)=fireal-dfilargura(1,k)+(l-1)*↙
(dfilargura(1,k)/NP);
                                end

                                T=[cos(alfaciclo(1,i)) sin(alfaciclo(1,i));-sin↙
(alfaciclo(1,i)) cos(alfaciclo(1,i))];


                                % Multiplas linhas
                                for l=1:(2*NP+1)
                                    for m=1:(2*NP+1)

                                        if rociclola(l,k)>rociclolae(g,m,i)
                                            distla(m,l)=(((zciclola(l,k)-zciclolae(g,↙
m,i))/cos(tau))^2+((firealla(l,1)-ficiclolae(g,m,i))*rociclola(l,k))^2)^0.5;
                                            distlav(m,l,:)=[(zciclola(l,k)-zciclolae↙
(g,m,i))/cos(tau) (firealla(l,1)-ficiclolae(g,m,i))*rociclola(l,k)];
                                            distlav1=[(zciclola(l,k)-zciclolae(g,m,i))↙
/cos(tau); (firealla(l,1)-ficiclolae(g,m,i))*rociclola(l,k)];
                                        else
                                            distla(m,l)=(((zciclola(l,k)-zciclolae(g,↙
m,i))/cos(tau))^2+((firealla(l,1)-ficiclolae(g,m,i))*rociclolae(g,m,i))^2)^0.5;
                                            distlav(m,l,:)=[(zciclola(l,k)-zciclolae↙
(g,m,i))/cos(tau) (firealla(l,1)-ficiclolae(g,m,i))*rociclolae(g,m,i)];
                                            distlav1=[(zciclola(l,k)-zciclolae(g,m,i))↙
/cos(tau); (firealla(l,1)-ficiclolae(g,m,i))*rociclolae(g,m,i)];
                                        end
                                        distlava1=T*distlav1;
                                        distlava(m,l)=distlava1(2,1);
                                    end
                                end


                                for l=1:(2*NP+1)
                                    if abs(distla(1,l))<b && abs(distla(2*NP+1,l))<b

                                        minimo=abs(distlava(1,l));
                                        posi=1;
                                        for m=1:(2*NP+1)
                                            minimo1=abs(distlava(m,l));
                                            if minimo1<minimo
                                                minimo=minimo1;
                                                posi=m;
                                            end
                                        end


                                    if posi~=2*NP+1 && posi~=1
                                        if abs(distlava(posi-1,l))<=abs(distlava↙
(posi+1,l))
                                            rociclola(l,k)=((abs(distlava(posi-1,l))↙
*rociclolae(g,posi,i)+abs(distlava(posi,l))*rociclolae(g,posi-1,i))/(abs(distlava↙
(posi-1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                            zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
```

```
                                        else
                                            rociclola(l,k)=((abs(distlava(posi+1,l))↙
*rociclolae(g,posi,i)+abs(distlava(posi,l))*rociclolae(g,posi+1,i))/(abs(distlava↙
(posi+1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                            zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                        end
                                    else
                                        if posi==2*NP+1
                                            rociclola(l,k)=((abs(distlava(posi-1,l))↙
*rociclolae(g,posi,i)+abs(distlava(posi,l))*rociclolae(g,posi-1,i))/(abs(distlava↙
(posi-1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                            zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                        else
                                            rociclola(l,k)=((abs(distlava(posi+1,l))↙
*rociclolae(g,posi,i)+abs(distlava(posi,l))*rociclolae(g,posi+1,i))/(abs(distlava↙
(posi+1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                            zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                        end
                                    end

                                    end
                                end

                            end
                        end
                        %i=i+1;
                    end
                end
            end
            end



            %Sobreposição próprio ciclo - copiar de cima

            for k=1:size(zciclo,2)
                    i=1;
                    while ficiclo(o,i)<(ficiclo(o,k)-2*pi)
                        if abs(zciclo(o,i)-zciclo(o,k))/cos(tau)<b
                            if abs(finovo1(o,i)-finovo1(o,k))<(b/rociclo(o,i))

                                finovo=ficiclo(o,k);
                                fireal=ficiclo(o,k);
                                dfireal=abs(ficiclo(o,i)-finovo);

                                while finovo>0
                                    finovo=finovo-2*pi;
                                    dfi1=abs(ficiclo(o,i)-finovo);
                                    if dfi1<dfireal
                                        fireal=finovo;
                                        dfireal=dfi1;
                                    end
                                end
```

```
                                firealla=zeros((2*NP+1),1);
                                for l=1:(2*NP+1)
                                    firealla(l,1)=fireal-dfilargura(1,k)+(l-1)*↙
(dfilargura(1,k)/NP);
                                end

                                T=[cos(alfaciclo(1,i)) sin(alfaciclo(1,i));-sin↙
(alfaciclo(1,i)) cos(alfaciclo(1,i))];


                                % Multiplas linhas
                                for l=1:(2*NP+1)
                                    for m=1:(2*NP+1)

                                        if rociclola(l,k)>rociclola(m,i)
                                            distla(m,l)=(((zciclola(l,k)-zciclola(m,↙
i))/cos(tau))^2+((firealla(l,1)-ficiclola(m,i))*rociclola(l,k))^2)^0.5;
                                            distlav(m,l,:)=[(zciclola(l,k)-zciclola(m,↙
i))/cos(tau) (firealla(l,1)-ficiclola(m,i))*rociclola(l,k)];
                                            distlav1=[(zciclola(l,k)-zciclola(m,i))↙
/cos(tau); (firealla(l,1)-ficiclola(m,i))*rociclola(l,k)];
                                        else
                                            distla(m,l)=(((zciclola(l,k)-zciclola(m,↙
i))/cos(tau))^2+((firealla(l,1)-ficiclola(m,i))*rociclola(m,i))^2)^0.5;
                                            distlav(m,l,:)=[(zciclola(l,k)-zciclola(m,↙
i))/cos(tau) (firealla(l,1)-ficiclola(m,i))*rociclola(m,i)];
                                            distlav1=[(zciclola(l,k)-zciclola(m,i))↙
/cos(tau); (firealla(l,1)-ficiclola(m,i))*rociclola(m,i)];
                                        end
                                        distlava1=T*distlav1;
                                        distlava(m,l)=distlava1(2,1);
                                    end
                                end


                                for l=1:(2*NP+1)
                                    if abs(distla(1,l))<b && abs(distla(2*NP+1,l))<b

                                        minimo=abs(distlava(1,l));
                                        posi=1;
                                        for m=1:(2*NP+1)
                                            minimo1=abs(distlava(m,l));
                                            if minimo1<minimo
                                                minimo=minimo1;
                                                posi=m;
                                            end
                                        end


                                    if posi~=2*NP+1 && posi~=1
                                        if abs(distlava(posi-1,l))<=abs(distlava↙
(posi+1,l))
                                            rociclola(l,k)=((abs(distlava(posi-1,l))↙
*rociclola(posi,i)+abs(distlava(posi,l))*rociclola(posi-1,i))/(abs(distlava(posi-1,l))↙
+abs(distlava(posi,l))))+(esp/cos(tau));
```

```
                                        zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                    else
                                        rociclola(l,k)=((abs(distlava(posi+1,l))↙
*rociclola(posi,i)+abs(distlava(posi,l))*rociclola(posi+1,i))/(abs(distlava(posi+1,l))↙
+abs(distlava(posi,l))))+(esp/cos(tau));
                                        zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                    end
                                else
                                    if posi==2*NP+1
                                        rociclola(l,k)=((abs(distlava(posi-1,l))↙
*rociclola(posi,i)+abs(distlava(posi,l))*rociclola(posi-1,i))/(abs(distlava(posi-1,l))↙
+abs(distlava(posi,l))))+(esp/cos(tau));
                                        zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                    else
                                        rociclola(l,k)=((abs(distlava(posi+1,l))↙
*rociclola(posi,i)+abs(distlava(posi,l))*rociclola(posi+1,i))/(abs(distlava(posi+1,l))↙
+abs(distlava(posi,l))))+(esp/cos(tau));
                                        zciclola(l,k)=zciclola(l,k)-esp*sin(tau);
                                    end
                                end

                                end
                            end

                        else
                            if abs(finovo1(o,i)-finovo1(o,k))>(2*pi-(b/rociclo(o,↙
i)))

                                finovo=ficiclo(o,k);
                                fireal=ficiclo(o,k);
                                dfireal=abs(ficiclo(o,i)-finovo);

                                while finovo>0
                                    finovo=finovo-2*pi;
                                    dfi1=abs(ficiclo(o,i)-finovo);
                                    if dfi1<dfireal
                                        fireal=finovo;
                                        dfireal=dfi1;
                                    end
                                end

                                firealla=zeros((2*NP+1),1);
                                for l=1:(2*NP+1)
                                    firealla(l,1)=fireal-dfilargura(1,k)+(l-1)*↙
(dfilargura(1,k)/NP);
                                end

                                T=[cos(alfaciclo(1,i)) sin(alfaciclo(1,i));-sin↙
(alfaciclo(1,i)) cos(alfaciclo(1,i))];


                                % Multiplas linhas
                                for l=1:(2*NP+1)
                                    for m=1:(2*NP+1)
```

```
                                            if rociclola(l,k)>rociclola(m,i)
                                                distla(m,l)=(((zciclola(l,k)-zciclola↙
(m,i))/cos(tau))^2+((firealla(l,1)-ficiclola(m,i))*rociclola(l,k))^2)^0.5;
                                                distlav(m,l,:)=[(zciclola(l,k)-↙
zciclola(m,i))/cos(tau) (firealla(l,1)-ficiclola(m,i))*rociclola(l,k)];
                                                distlav1=[(zciclola(l,k)-zciclola(m,↙
i))/cos(tau); (firealla(l,1)-ficiclola(m,i))*rociclola(l,k)];
                                            else
                                                distla(m,l)=(((zciclola(l,k)-zciclola↙
(m,i))/cos(tau))^2+((firealla(l,1)-ficiclola(m,i))*rociclola(m,i))^2)^0.5;
                                                distlav(m,l,:)=[(zciclola(l,k)-↙
zciclola(m,i))/cos(tau) (firealla(l,1)-ficiclola(m,i))*rociclola(m,i)];
                                                distlav1=[(zciclola(l,k)-zciclola(m,↙
i))/cos(tau); (firealla(l,1)-ficiclola(m,i))*rociclola(m,i)];
                                            end
                                            distlava1=T*distlav1;
                                            distlava(m,l)=distlava1(2,1);
                                        end
                                    end


                                    for l=1:(2*NP+1)
                                        if abs(distla(1,l))<b && abs(distla(2*NP+1,l))↙
<b

                                            minimo=abs(distlava(1,l));
                                            posi=1;
                                            for m=1:(2*NP+1)
                                                minimo1=abs(distlava(m,l));
                                                if minimo1<minimo
                                                    minimo=minimo1;
                                                    posi=m;
                                                end
                                            end


                                            if posi~=2*NP+1 && posi~=1
                                                if abs(distlava(posi-1,l))<=abs↙
(distlava(posi+1,l))
                                                    rociclola(l,k)=((abs(distlava↙
(posi-1,l))*rociclola(posi,i)+abs(distlava(posi,l))*rociclola(posi-1,i))/(abs(distlava↙
(posi-1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                                    zciclola(l,k)=zciclola(l,k)-↙
esp*sin(tau);
                                                else
                                                    rociclola(l,k)=((abs(distlava↙
(posi+1,l))*rociclola(posi,i)+abs(distlava(posi,l))*rociclola(posi+1,i))/(abs(distlava↙
(posi+1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                                    zciclola(l,k)=zciclola(l,k)-↙
esp*sin(tau);
                                                end
                                            else
                                                if posi==2*NP+1
                                                    rociclola(l,k)=((abs(distlava↙
(posi-1,l))*rociclola(posi,i)+abs(distlava(posi,l))*rociclola(posi-1,i))/(abs(distlava↙
```

```
(posi-1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                                        zciclola(l,k)=zciclola(l,k)-↙
esp*sin(tau);
                                                    else
                                                        rociclola(l,k)=((abs(distlava↙
(posi+1,l))*rociclola(posi,i)+abs(distlava(posi,l))*rociclola(posi+1,i))/(abs(distlava↙
(posi+1,l))+abs(distlava(posi,l))))+(esp/cos(tau));
                                                        zciclola(l,k)=zciclola(l,k)-↙
esp*sin(tau);
                                                    end
                                                end

                                            end
                                        end

                                    end
                                end
                            end
                            i=i+1;
                        end
                    end


        % Cartesianas multiplas linhas
        xciclola=zeros(4*NP+2,size(ficiclola,2));
        yciclola=zeros(4*NP+2,size(ficiclola,2));
        for n=1:2*NP+1
           xciclola(n,:)=rociclola(n,:).*cos(ficiclola(n,:));
           yciclola(n,:)=rociclola(n,:).*sin(ficiclola(n,:));
        end
        c=1;
        for n=2*NP+2:2*(2*NP+1)
           xciclola(n,:)=(rociclola((2*NP+2-c),:)+esp/cos(tau)).*cos(ficiclola↙
((2*NP+2-c),:));
           yciclola(n,:)=(rociclola((2*NP+2-c),:)+esp/cos(tau)).*sin(ficiclola↙
((2*NP+2-c),:));
           zciclola(n,:)=zciclola((2*NP+2-c),:)-esp*sin(tau);
           c=c+1;
        end

        %Arrays com todas as coordenadas
        rociclolae(o,:,:)=rociclola(:,:);
        ficiclolae(o,:,:)=ficiclola(:,:);
        zciclolae(o,:,:)=zciclola(:,:);
        xciclolae(o,:,:)=xciclola(:,:);
        yciclolae(o,:,:)=yciclola(:,:);


        %Plot de tudo
        surf(zciclola(:,:),yciclola(:,:),xciclola↙
(:,:),'FaceColor','r','EdgeColor','none','LineStyle','none')
        plot3(zciclola(1,:),yciclola(1,:),xciclola(1,:),'k','Linewidth',0.5);
        plot3(zciclola((2*NP+1),:),yciclola((2*NP+1),:),xciclola↙
((2*NP+1),:),'k','Linewidth',0.5);
        plot3(zciclola((2*NP+2),:),yciclola((2*NP+2),:),xciclola↙
```

```
((2*NP+2),:),'k','Linewidth',0.5);
        plot3(zciclola(2*(2*NP+1),:),yciclola(2*(2*NP+1),:),xciclola(2*↙
(2*NP+1),:),'k','Linewidth',0.5);
        plot3(zciclola((3*NP+2),:),yciclola((3*NP+2),:),xciclola↙
((3*NP+2),:),'b','Linewidth',0.5);
        end
    end
end

disp('Fim');
```

A.2 Christoffel coefficients function

```
function [chr,gg1,gg2,e,g,ev,eu,gv,gu]=Christoffel_funcao_1(S,u,v,ro)

S

%Primeira formula fundamental
e=formula(diff(S,u));
E1=dot(e,e);
E=simplify(E1);

g=formula(diff(S,v));
G1=dot(g,g);
G=simplify(G1);

F1=dot(g,e);
F=simplify(F1);

gg1(1,1)=E;
gg1(1,2)=F;
gg1(2,1)=F;
gg1(2,2)=G;

n=cross(e,g)/norm(cross(e,g));

L=dot(n,diff(e,u));
M=dot(n,diff(e,v));
N=dot(n,diff(g,v));

gg2(1,1)=L;
gg2(1,2)=M;
gg2(2,1)=M;
gg2(2,2)=N;

%Símbolos Christoffel
chr(1,1,1)=simplify((G*(diff(E,u))-2*F*diff(F,u)+F*diff(E,v))/(2*(E*G-F^2)));
chr(1,1,2)=simplify((2*E*(diff(F,u))-E*diff(E,v)-F*diff(E,u))/(2*(E*G-F^2)));
chr(1,2,1)=simplify((G*(diff(E,v))-F*diff(G,u))/(2*(E*G-F^2)));
chr(2,1,1)=chr(1,2,1);
chr(1,2,2)=simplify((E*(diff(G,u))-F*diff(E,v))/(2*(E*G-F^2)));
chr(2,1,2)=chr(1,2,2);
chr(2,2,1)=simplify((2*G*(diff(F,v))-G*diff(G,u)-F*diff(G,v))/(2*(E*G-F^2)));
chr(2,2,2)=simplify((E*(diff(G,v))-2*F*diff(F,v)-F*diff(G,u))/(2*(E*G-F^2)));


ev=simplify(diff(gg1(1,1),v));
eu=simplify(diff(gg1(1,1),u));

gv=simplify(diff(gg1(2,2),v));
gu=simplify(diff(gg1(2,2),u));

end
```

A.3 Pattern determination

```
%Determinação do padrão
clear all;
clc;

%Dados padrão
N=89;
d=13;

%Dados requisitos
r=20;
bmin=2;
bmax=6;
alfamin=40*pi/180;
alfamax=50*pi/180;

fifinalmax=55; %para alfamax
fifinalmin=45;%para alfamin


%% Dando valores de N e d
A(1,1)=N;
A(1,2)=d;
C(1,1)=1;
C(1,2)=0;
D(1,1)=0;
D(1,2)=1;
E(1,2)=1;

i=3;
A(1,i)=mod(A(1,i-2),A(1,i-1));
B(1,i)=floor(A(1,i-2)/A(1,i-1));
C(1,i)=B(1,i)*C(1,i-1)+C(1,i-2);
D(1,i)=B(1,i)*D(1,i-1)+D(1,i-2);
E(1,i)=E(1,i-1)*(-1);

while mod(A(1,i-2),A(1,i-1))~=0
    i=i+1;
A(1,i)=mod(A(1,i-2),A(1,i-1));
B(1,i)=floor(A(1,i-2)/A(1,i-1));
C(1,i)=B(1,i)*C(1,i-1)+C(1,i-2);
D(1,i)=B(1,i)*D(1,i-1)+D(1,i-2);
E(1,i)=E(1,i-1)*(-1);
end

m=C(1,i-1);
n=D(1,i-1);
s=E(1,i-1);


%% Dando intervalos de b e alfa
Nmin=2*pi*r*cos(alfamax)/bmax;
Nmax=2*pi*r*cos(alfamin)/bmin;

Nmin1=ceil(Nmin);
Nmax1=floor(Nmax);
```

```
int=Nmax1-Nmin1;
l=Nmin1;
pad=0;

for i=1:int+1

   Dmin=fifinalmin*l;
   Dmax=fifinalmax*l;
   Dmin1=ceil(Dmin);
   Dmax1=floor(Dmax);

   p=0;
   l1(i,1)=l;
   for j=Dmin1:Dmax1
       p=p+1;
      d1(i,p)=mod(j,l);

      if d1(i,p)~=0

         A(1,1)=l;
        A(1,2)=d1(i,p);
        C(1,1)=1;
        C(1,2)=0;
        D(1,1)=0;
        D(1,2)=1;
        E(1,2)=1;

        k=3;
        A(1,k)=mod(A(1,k-2),A(1,k-1));
        B(1,k)=floor(A(1,k-2)/A(1,k-1));
        C(1,k)=B(1,k)*C(1,k-1)+C(1,k-2);
        D(1,k)=B(1,k)*D(1,k-1)+D(1,k-2);
        E(1,k)=E(1,k-1)*(-1);

        while mod(A(1,k-2),A(1,k-1))~=0
            k=k+1;
            A(1,k)=mod(A(1,k-2),A(1,k-1));
            B(1,k)=floor(A(1,k-2)/A(1,k-1));
            C(1,k)=B(1,k)*C(1,k-1)+C(1,k-2);
            D(1,k)=B(1,k)*D(1,k-1)+D(1,k-2);
            E(1,k)=E(1,k-1)*(-1);
        end

        m1=C(1,k-1);
        n1=D(1,k-1);
        s=E(1,k-1);

         if d1(i,p)*n1-l*m1==1
            pad=pad+1;
            padrao(pad,1)=l;                %N
            padrao(pad,2)=d1(i,p);          %d
            padrao(pad,3)=m1;               %m
            padrao(pad,4)=n1;               %n
            padrao(pad,5)=s;                %sinal
```

```
            padrao(pad,6)=d1(i,p)*n1-l*m1;  %resultado da dio
            padrao(pad,7)=j;                %D
            padrao(pad,8)=j/l;              %w=D/N
         else
             if d1(i,p)*n1-l*m1==-1
                 pad=pad+1;
                padrao(pad,1)=l;
                padrao(pad,2)=d1(i,p);
                padrao(pad,3)=m1;
                padrao(pad,4)=n1;
                padrao(pad,5)=s;
                padrao(pad,6)=d1(i,p)*n1-l*m1;
                padrao(pad,7)=j;
                padrao(pad,8)=j/l;
             end
         end
      end
   end
   l=l+1;
end
```

Appendix B

User Manual

Pathwind is a software that aims to simulate filament winding along axis-symmetric mandrels. It is able to generate paths and patterns for different mandrel geometries and tow properties as well as generate CNC code for different filament winding machines. The software was first developed as part of a Master thesis while working in INEGI.

B.1 Structure

The software follows this general structure:

1. Data input
   (a) Mandrel geometry
   (b) Tow properties
   (c) Friction coefficient
   (d) Machine parameters (only necessary if CNC output is required)
   (e) Desired discretization (distance between points in the path)
   (f) Type of overlap requested
2. Path creation for first cycle
   (a) Geodesic path for useful area
   (b) First turnaround
   (c) Geodesic path
   (d) Second turnaround
3. First cycle vector assembly
4. Representing the tow geometry along the first cycle
5. Representing the variation of different parameters along one cycle
6. Determining the path for all cycles

7. Geometric output of paths
8. CNC code
   (a) Generating CNC code
   (b) Verifying that position, velocity and acceleration are within boundary conditions
   (c) CNC code output
   (d) Machine axis movement
9. Tow overlap and real trajectories

B.2 Data input

rim – Mandrel radius on the left end in mm
rfm - Mandrel radius on the right end in mm
zim – Coordinate of the left end of the mandrel in mm
zfm - Coordinate of the right end of the mandrel in mm
zig - Coordinate of the left geodesic end of the mandrel in mm
zi - Coordinate of the beginning of the winding in mm
zfg - Coordinate of the right geodesic end of the mandrel in mm
alfainicio - winding angle at the initial winding coordinate (zi) in radians $\alpha \in]-\pi/2, \pi/2[$
niu - Static friction coefficient μ
lambda1 - λ parameter where $\lambda = c\mu$ and $c \in [0, 1]$
ds - increment for calculation along the geodesic paths in mm
dsta - increment for calculation along the turnaround paths in mm
b - bandwidth in mm
NR - Number of rovings
RW=b/NR - roving width in mm
TEX - TEX value in g/km
fvc - fibre volume content
fmc - fibre mass content
fibdens - fibre density in g/cm^3
resdens - resin density in g/cm^3
NP - Number of paths per tow (actual number of paths will be $4NP$)
vel - Machine velocity (multiplies the time that each command line takes to perform the axes movement)
nomeoutput - Name of the new folder where outputs will be stored
filepath - Filepath for where the new folder will be created
gravartrajectorias - Specifies if the path trajectories are to be saved as a .txt. If saving is required then the variable should be equal to 's'.
cnc - Specifies if a CNC output is required. This relates to the calculations and the outputs. If it is required then the variable should be equal to 's'.
gravarcnc - Specifies if the CNC code output files are to be saved as a .mpf and .spf. If saving is required then the variable should be equal to 's'.
D - Fixed distance of the pay-out eye to the mandrel in mm
sobre - Specifies if a overlapped output is required. This relates to the calculations

and the outputs. If it is required then the variable should be equal to 'sobreposição1' if only one cycle is desired and 'sobreposição2' if all cycles are required.

B.3 Machine parameters input

Reference positions
RefA - Reference position of axis A
RefX - Reference position of axis X
RefY - Reference position of axis Y
RefB - Reference position of axis B
RefZ - Reference position of axis Z
RefC - Reference position of axis C

Dimensions
Xmin - Minimum value of X
Xmax - Maximum value of X
Ymin - Minimum value of Y
Ymax - Maximum value of Y
Zmin - Minimum value of Z
Zmax - Maximum value of Z
Cmin - Minimum value of C
Cmax - Maximum value of C
POew - Pay out eye width
POe - Pay out eye eccentricity
YawR - Yaw radius

Velocities
VA - Maximum velocity of A in revolutions/min
VX - Maximum velocity of X in m/s
VY - Maximum velocity of Y in m/s
VB - Maximum velocity of B in revolutions/min
VZ - Maximum velocity of Z in m/s
VC - Maximum velocity of C in revolutions/min

Accelerations - defined as the maximum permissible velocity change of an axis between two control data records
AA - Maximum acceleration of A in revolutions/min
AX - Maximum acceleration of X in m/s
AY - Maximum acceleration of Y in m/s
AB - Maximum acceleration of B in revolutions/min
AZ - Maximum acceleration of Z in m/s
AC - Maximum acceleration of C in revolutions/min

Resolutions
RA - axis resolution of A in inc/revolution
RX - axis resolution of X in inc/mm
RY - axis resolution of Y in inc/mm

RB - axis resolution of B in inc/revolution
RZ - axis resolution of Z in inc/mm
RC - axis resolution of C in inc/revolution