GOCE DELCEV UNIVERSITY - STIP
FACULTY OF COMPUTER SCIENCE

ISSN 2545-479X print
ISSN 2545-4803 on line

BALKAN JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS (BJAMI)

BALKAN JOURNAL
OF APPLIED MATHEMATICS AND INFORMATICS

YEAR 2018

VOLUME I, Number 1

GOCE DELCEV UNIVERSITY - STIP, REPUBLIC OF MACEDONIA
FACULTY OF COMPUTER SCIENCE

ISSN 2545-479X print
ISSN 2545-4803 on line

BALKAN JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS

(BJAMI)

YEAR 2018 VOLUME I, Number 1

AIMS AND SCOPE:
BJAMI publishes original research articles in the areas of applied mathematics and informatics.

Topics:
1. Computer science;
2. Computer and software engineering;
3. Information technology;
4. Computer security;
5. Electrical engineering;
6. Telecommunication;
7. Mathematics and its applications;
8. Articles of interdisciplinary of computer and information sciences with education, economics, environmental, health, and engineering.

Address of the editorial office
Goce Delcev University – Stip
Faculty of philology
Krste Misirkov 10-A
PO box 201, 2000 Štip,
R. of Macedonia

BALKAN JOURNAL
OF APPLIED MATHEMATICS AND INFORMATICS (BJAMI), Vol 1

ISSN 2545-479X print
ISSN 2545-4803 on line
Vol. 1, No. 1, Year 2018

EDITORIAL BOARD

CONTENT

Aleksandar, Velinov, Vlado, Gicev
PRACTICAL APPLICATION OF SIMPLEX METHOD FOR SOLVING
LINEAR PROGRAMMING PROBLEMS 7

Biserka Petrovska , Igor Stojanovic , Tatjana Atanasova Pachemska
CLASSIFICATION OF SMALL DATA SETS OF IMAGES WITH
TRANSFER LEARNING IN CONVOLUTIONAL NEURAL NETWORKS 17

Done Stojanov
WEB SERVICE BASED GENOMIC DATA RETRIEVAL 25

Aleksandra Mileva, Vesna Dimitrova
SOME GENERALIZATIONS OF RECURSIVE DERIVATES
OF k-ary OPERATIONS 31

Diana Kirilova Nedelcheva
SOME FIXED POINT RESULTS FOR CONTRACTION
SET - VALUED MAPPINGS IN CONE METRIC SPACES 39

Aleksandar Krstev, Dejan Krstev, Boris Krstev, Sladzana Velinovska
DATA ANALYSIS AND STRUCTURAL EQUATION MODELLING
FOR DIRECT FOREIGN INVESTMENT FROM LOCAL POPULATION 49

Maja Srebrenova Miteva, Limonka Koceva Lazarova
NOTION FOR CONNECTEDNESS AND PATH CONNECTEDNESS IN
SOME TYPE OF TOPOLOGICAL SPACES 55

The Appendix

**Aleksandra Stojanova , Mirjana Kocaleva , Natasha Stojkovikj , Dusan Bikov ,
Marija Ljubenovska , Savetka Zdravevska , Biljana Zlatanovska , Marija Miteva ,
Limonka Koceva Lazarova**
OPTIMIZATION MODELS FOR SHEDULING IN KINDERGARTEN
AND HEALTHCARE CENTES 65

**Maja Kukuseva Paneva, Biljana Citkuseva Dimitrovska, Jasmina Veta Buralieva,
Elena Karamazova, Tatjana Atanasova Pacemska**
PROPOSED QUEUING MODEL M/M/3 WITH INFINITE WAITING
LINE IN A SUPERMARKET 73

Maja Mijajlovikj1, Sara Srebrenkoska, Marija Chekerovska, Svetlana Risteska,
Vineta Srebrenkoska
APPLICATION OF TAGUCHI METHOD IN PRODUCTION OF SAMPLES
PREDICTING PROPERTIES OF POLYMER COMPOSITES 79

**Sara Srebrenkoska, Silvana Zhezhova, Sanja Risteski, Marija Chekerovska
Vineta Srebrenkoska Svetlana Risteska**
APPLICATION OF FACTORIAL EXPERIMENTAL DESIGN IN
PREDICTING PROPERTIES OF POLYMER COMPOSITES 85

**Igor Dimovski, Ice Gjumandeloski, Filip Kochoski, Mahendra Paipuri,
Milena Veneva , Aleksandra Risteska**
COMPUTER AIDED (FILAMENT WINDING) TAPE PLACEMENT
FOR ELBOWS. PRACTICALLY ORIENTATED ALGORITHM 89

The Appendix of the first number of Balkan Journal of Applied Mathematics and Informatics, is devoted to the reports of the First Modelling Week in Macedonia, which was held in Stip, 12-16 February 2018.

The First Modelling Week in Macedonia was organized by Faculty of Computer Science - Department of Mathematics and Statistics, Faculty of Electrical Engineering and Faculty of Technology with the support of the TD 1409 MI-NET Cost Action. The aims of the Modelling Week were: widening, broadening and sharing knowledge relevant to the Action's objectives through working on modern and actual problems which can be solved with mathematics and mathematical modelling.

The Modelling Week was organized under auspices of Prof. Blazo Boev, Rector of the Goce Delcev University, Stip, Macedonia.

The Program Committee of the First Modelling Week were:

1. Vineta Srebrenkoska, PhD – Macedonia
2. Tatjana Atanasova – Pachemska, PhD – Macedonia
3. Poul G. Hjorth, PhD – Denmark
4. Wojciech Okrasinski, PhD – Poland
5. Joerg Elzenbach, PhD – Germany
6. Gregoris Makrides, PhD – Cyprus
7. Biljana Jolevska – Tuneska, PhD – Macedonia
8. Limonka Koceva Lazarova, PhD - Macedonia

In the First Modelling Week in Macedonia participated 34 participants from Macedonia, Bulgaria, Portugal and Denmark. The Modelling Week was aimed towards Masters, PhD students, Early Career Investigators (up to 8 years after their PhD). All the participants were split in three groups in order to solve the three problems which were set:

Problem 1 - Scheduling in kindergarten, proposed by Limonka Koceva Lazarova

Problem 2 - Determining the optimal number of cash boxes to increase the efficiency of the customer service and determining the way of storage of products in the warehouse. How to manage stocks in the warehouse, proposed by Tatjana Atanasova – Pachemska.

Problem 3 - Optimization of the industrial processes for production of advanced polymer composites by implementation of the full factorial experimental design, proposed by Vineta Srebrenkoska.

The third problem was split in three subproblems.

All of the solutions are presented in form of reports in this appendix.

Thanks for the editors of the Balkan Journal of Applied Mathematics and Informatics, about their support for publishing of the results from The First Modelling Week in Macedonia.

UDK: 004.942:[620.22-021.385:621.77

COMPUTER AIDED (FILAMENT WINDING) TAPE PLACEMENT FOR ELBOWS. PRACTICALLY ORIENTATED ALGORITHM

Igor Dimovski[1], Ice Gjumandeloski[2], Filip Kochoski[2], Mahendra Paipuri[3], Milena Veneva[4] , Aleksandra Risteska[5]

[1]Department of Mathematics, Institute for Advanced Composites and Robotics, Prilep, Macedonia
igord@iacr.edu.mk
[2]Institute for Advanced Composites and Robotics, Prilep, Macedonia
ice_gjumandeloski@yahoo.com, filip.kocho@gmail.com
[3]Departamento de Engenharia Civil, Arquitectura e Georrecursos, Instituto Superior Técnico, Lisboa
mahendra.paipuri@gmail.com
[4]Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
milena.p.veneva@gmail.com
[5]Faculty of Computer science, Goce Delcev University, Stip, Macedonia
aleksandra.risteska@ugd.edu.mk

Abstract: Filament winding is one of the most used automated techniques for manufacturing of composite objects with different open-end or closed-end structures. Mathematical model for covering an elbow mandrel with composite material is considered. The nature of the comprising equations is elaborated in detail. A practically orientated algorithm for filament winding for elbows is formulated and its open-source implementations in Python and MATLAB are presented. The results from the constructed algorithm are presented and discussed.
Keywords: filament winding for elbows, tape placement for elbows, practical algorithm, open-source, CAD.

1. Introduction and problem statement

Filament winding is one of the most used automated techniques for manufacturing of composite objects with different open-end or closed-end structures. The process consists of winding glass fiber or carbon fiber strands, impregnated in a bath with a resin, kept under tension over a rotating spindle mandrel. **Figure 1** shows the filament winding of an elbow. Typically, the tension is in the range from 1.1 to 4.4 N [1]. The pattern and the angle of the fiber winding are controlled from a delivery (or feed) eye on a carriage. Once the coverage has achieved a desired thickness, the resin is cured and then the mandrel can be or can be not extracted, depending on the shape and the application of the object. A video which shows how filament winding works can be seen here: [ArcoIndustries]. (2013, April 24). *Fiberglass Elbow Filament Winding Demo 1* [Video File]. Retrieved from https://www.youtube.com/watch?v=0f63pfHNK6U. There are four types of filament winding according to the winding angle – hoop, helical, circumferential, and polar.

Objects with cross sections with very small acute angles cannot be filament wound [1]. Strategies for filament winding for different shapes exist in the literature, e.g.:

- elbow – [2], [3];
- toroidal – [4];
- T-shaped – [5];
- generalization – [6].

Other than filament winding can be applied for various set of shapes, among the advantages of the technique are also its economics, since the fiber and the resin are used in their lowest cost form and the fiber is

continuous over the entire path. Another upside comes from the fact that the process can be automated which results in cost savings in the case of large production volumes.

There exist different CAD/CAM software solutions for filament winding for different shapes, e.g. **Mikrosam Winding Expert** [7] or **Cadfil® - Filament Winding Software & Technology** [8] (some other software solutions for filament winding, as well as their upsides and downsides are listed in [9]), but they are not distributed free of charge.

There is a need of an user-friendly formulated algorithm (it is not the case in most of the ones available in the literature) and an open-source implementation of the algorithm for filament winding for elbows and so this was the problem we needed to solve during the First Modelling Week in Macedonia. As a first step an algorithm is proposed to cover the entire elbow with tapes without considering the dwell[1].

Figure 1. *(Example of filament winding of an elbow part.)*[2]

If we have done a solution of the problem that includes dwelling, we would need to be aware of the geodesic curvature k_g [2] and the normal curvature k_n [6] of the path that make the slippage coefficient [2]:

$$\lambda = \frac{k_g}{k_n}, \tag{1}$$

and we must keep $|\lambda| < \mu$, where μ is the coefficient of friction between the fibers and the mandrel. Our algorithm can be extended to add dwelling to the problem.

In our solution, we have chosen to have a geodesic path, *i.e.* $k_g = 0$ of the fibers (tape), so we can use a certain number of tapes to make a layer, with all tapes from the layer going in the same direction.

[1] Dwell is the amount of mandrel rotation with no carriage movement at the start and at the end of the filament winding.
[2] The picture was taken as a screenshot from [ArcoIndustries]. (2013, April 24). Fiberglass Elbow Filament Winding Demo 1 [Video File]. Retrieved from https://www.youtube.com/watch?v=0f63pfHNK6U

At the end of the First Modelling Week in Macedonia our team gave a talk on the problem which was assigned to us, as well as our solution. The presentation can be found on ResearchGate: https://www.researchgate.net/project/Computer-Aided-Filament-Winding-Tape-Placement-for-Elbows-Practically-Orientated-Algorithm/update/5abbf6624cde269658661549

The aim and the main contribution of this note is to give some clarification on the origin of the formulae needed for mathematical modelling of tape placement for elbow with the concepts of differential geometry and following [2], as well as to formulate a practically orientated algorithm. Finally, MATLAB and Python implementations of the algorithm are provided.

The layout of the paper is as follows: in **Section 2**, we introduce the mathematical model and the practically orientated algorithm. **Section 3** presents the source code of the algorithm implemented in MATLAB and Python. In **Section 4** we state and discuss the results. In the final **Section 5** some concluding remarks and notes for future work are listed. Afterwards, the acknowledgments and the references follow.

2. Mathematical model and algorithm

2.1 Choice of coordinate system and building a representation of the elbow

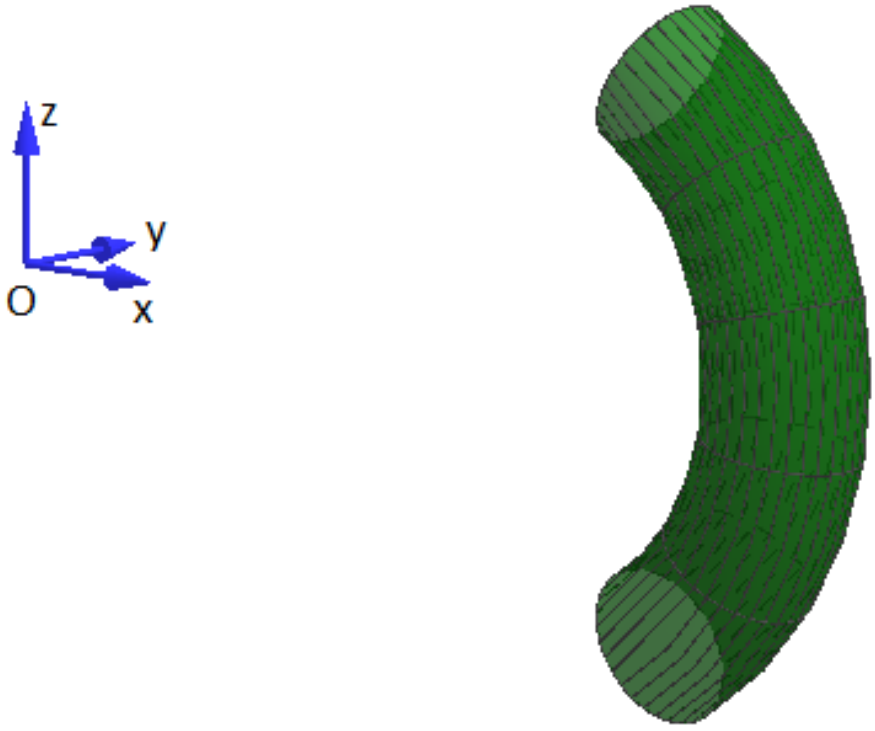

Figure 2. (*The elbow in a coordinate system.*)

An orthogonal coordinate system $Oxyz$ is chosen (see **Figure 2**). The surface of the elbow in a (θ, φ) coordinate system (planes with constant θ passing through the torus's axis) [10] can be parameterized as a surface of revolution. Then, the formulae that represent the surface $T(\theta, \varphi)$ of the elbow with respect to $Oxyz$, are:

$$T(\theta,\varphi) = \begin{pmatrix} (R + r\cos\varphi)\cos\theta \\ r\sin\varphi \\ (R + r\cos\varphi)\sin\theta \end{pmatrix}, \theta \in \left[-\frac{\theta_{span}}{2}, \frac{\theta_{span}}{2}\right], \varphi \in [0, 2\pi), \tag{2}$$

where R and r are constants of the elbow – the distance from the center of the tube (the origin of the coordinate system) to the center of the torus (major radius) and the radius of the tube (minor radius), respectively; θ and φ are surface-defining parameters of the elbow – toroidal and poloidal directions, respectively (see **Figure 3** and **Figure 4**). Without loss of generality, we consider the case when $R > r$.

The coefficients of the first fundamental form of $T(\theta, \varphi)$ are (see [11], [12]):

$$\sqrt{E} = \sqrt{T_\theta T_\theta} = R + r\cos\varphi, \tag{3}$$

$$F = T_\theta T_\varphi = 0, \tag{4}$$

$$\sqrt{G} = \sqrt{T_\varphi T_\varphi} = r. \tag{5}$$

From $F = 0$ it follows that the coordinate system (θ, φ) is orthogonal and hence Liouville's formula for the geodesic curvature [13] can be applied:

$$k_g = \frac{d\alpha}{ds} + \frac{(\sqrt{E})_\varphi}{\sqrt{EG}}\cos\alpha + \frac{(\sqrt{G})_\theta}{\sqrt{EG}}\sin\alpha. \tag{6}$$

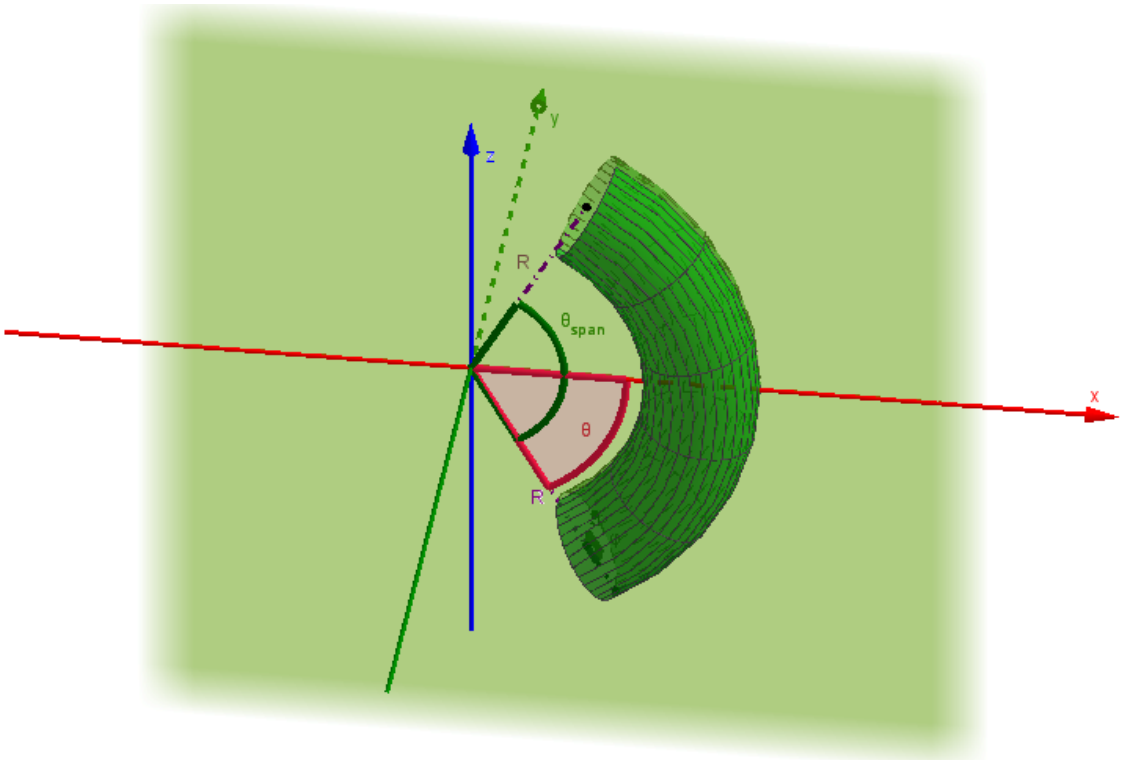

Figure 3. (*Parameters of the elbow.*)

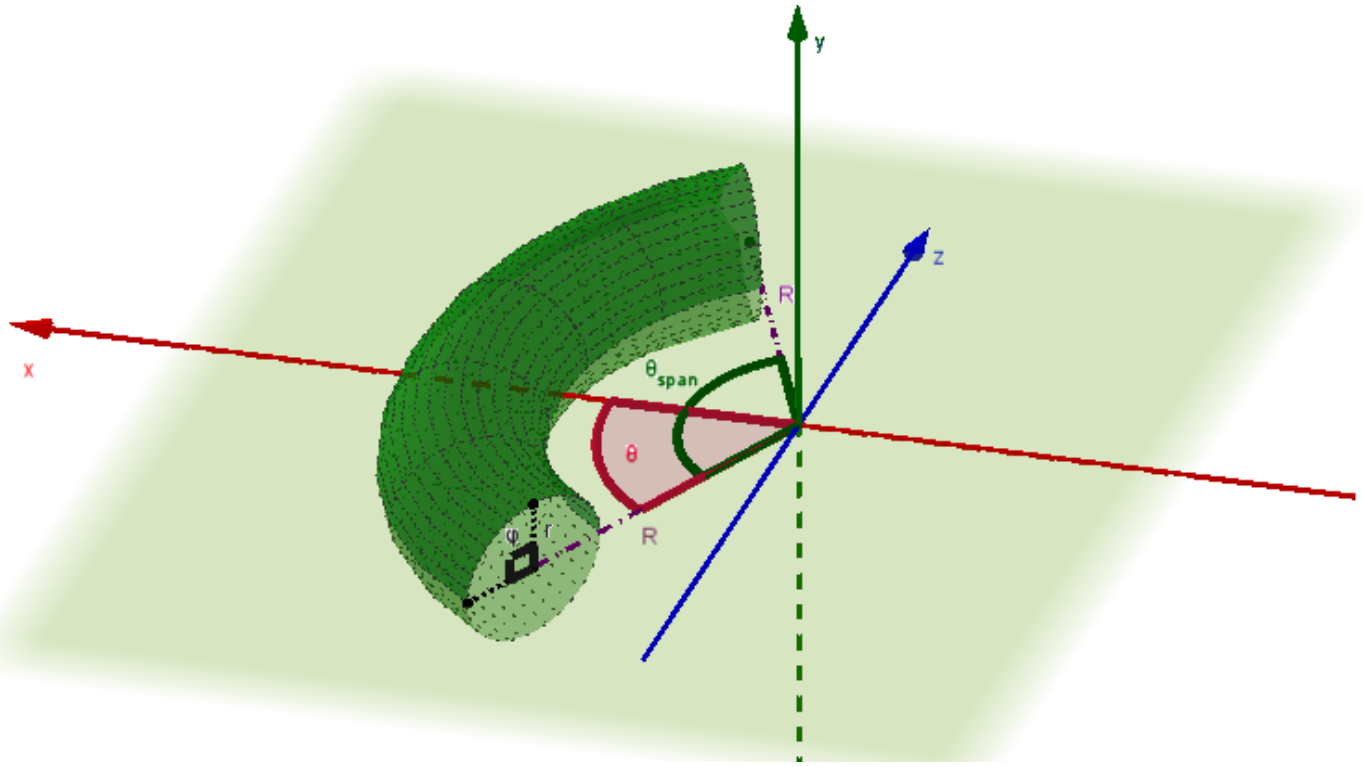

Figure 4. (*Parameters of the elbow.*)

In the case of an elbow, Eq. (7) follows:

$$k_g = \frac{d\alpha}{ds} + \frac{\sin\varphi}{R + r\cos\varphi}\cos\alpha. \tag{7}$$

2.2 Defining the winding angle α

The winding angle α is measured in the plane formed by the unit vectors e_θ and e_s, *i.e.* direction of course with path s; α_0 is a design constant that represents the initial value of the angle α when $\varphi = 0$.

2.3 Assumption for geodesic curves

Following [2], if $k_g = 0$, then from Eq. (7) it follows that:

$$\frac{d\alpha}{ds} = -\frac{\sin\varphi}{R + r\cos\varphi}\cos\alpha. \tag{8}$$

The length of a curve on the surface $T(\theta, \varphi)$ in meridional direction and parallel direction are given by (see [6]):

$$ds_{meridional} = \sqrt{E}\, d\theta, \tag{9}$$

$$ds_{parallel} = \sqrt{G}\, d\varphi. \tag{10}$$

The latter two formulae follow from the definition of the coefficients of the first fundamental form (Eq. (3), (4), and (5)). Then, if follows that (see **Figure 5**):

$$\frac{d\varphi}{ds} = \frac{\sin\alpha}{r}. \tag{11}$$

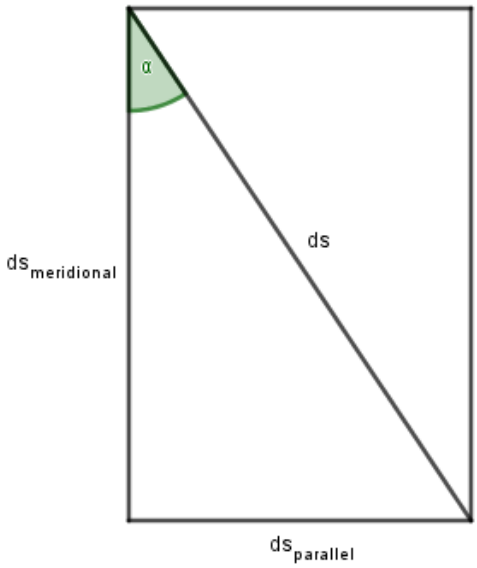

Figure 5. *(Winding angle.)*

Now, if the formula for the part length (see [11], [12]) is written and Eq. (11) is substituted in it, it follows:

$$\frac{d\theta}{ds} = \frac{\cos\alpha}{R + r\cos\varphi}. \tag{12}$$

Eq. (8), (11) and (12) form the system which determines the geodesics of the elbow. Eliminating the arc length s from this system of differential equations, we have:

$$\frac{d\alpha}{d\varphi} = -\frac{r\sin\varphi}{R + r\cos\varphi}\frac{\cos\alpha}{\sin\alpha}, \tag{13}$$

$$\frac{d\theta}{d\varphi} = \frac{r}{R + r\cos\varphi}\frac{\cos\alpha}{\sin\alpha}. \tag{14}[3]$$

Integrating Eq. (13) and determining the constant of integration taking into account the initial condition (see **Section 2.2**), we have:

$$\cos\alpha = \frac{(R + r)\cos\alpha_0}{R + r\cos\varphi}. \tag{15}$$

Substituting with Eq. (15) into Eq. (14), it follows that:

$$f(\varphi) = \frac{d\theta}{d\varphi} = \frac{r(R + r)\cos\alpha_0}{(R + r\cos\varphi)\sqrt{(R + r\cos\varphi)^2 - (R + r)^2\cos^2\alpha_0}}. \tag{16}$$

[3] One should note that the minus to the right-hand side of the equality sign in [2], Eq. (1.8) is a typo.

2.4 Defining criterion

So as to be able to define a stopping criterion which will tell us when the entire surface is going to be covered, the following formulae are considered:

$$\theta_{cycle} = \int_0^{2\pi} \frac{r\,(R+r)\,\cos\alpha_0}{(R+r\cos\varphi)\sqrt{(R+r\cos\varphi)^2-(R+r)^2\cos^2\alpha_0}}\,\mathrm{d}\varphi, \tag{17}$$

$$M = \left\lfloor \frac{(R+r)\,\theta_{cycle}\,\sin\alpha_0}{w} \right\rfloor + 1, \tag{18}$$

$$\theta_i = \frac{i\theta_{cycle}}{M}, i = \overline{0, M-1}, \tag{19}$$

where w is the tape width. Eq. (18) gives the number of needed tapes so as the whole surface of the elbow to be completely covered. The tapes should be placed uniformly, starting at $\theta = 0$, moving with a step $\theta_i, i = \overline{0, M-1}$ given by Eq. (19), with a period θ_{cycle} which is given by Eq. (17).

2.5 The idea

Because we needed M number of tapes to cover the surface, our idea for their position is to have the base line, *i.e.* $i = 0$, to pass at point described by $(\theta, \varphi) = (0, 0)$, and so to have every i-th line to pass at point described by $(\theta, \varphi) = (\theta_i, 0)$. With this considerations, we need to find where the line should start from, and where it will end (see **Figure 6**). This can be easily done if we split this subproblem into two parts 1) going from the middle to the end, and 2) going from the start to the middle. To that purpose, we need to find the correct starting and ending angles. Since we know that the starting and the ending angles for θ for each line i are $\theta_{-,i} = -\frac{\theta_{span}}{2}$ and $\theta_{+,i} = \frac{\theta_{span}}{2}$, respectively, the next thing we should do is to find the appropriate angles $\varphi_{-,i}$ and $\varphi_{+,i}$. To that purpose, we use the following formulae:

$$\Delta\theta_{+,i} = \frac{\theta_{span}}{2} - \theta_i, \tag{20}$$

$$\Delta\theta_{+,i} = \int_0^{\varphi_{+,i}} \frac{r\,(R+r)\,\cos\alpha_0}{(R+r\cos\varphi)\sqrt{(R+r\cos\varphi)^2-(R+r)^2\cos^2\alpha_0}}\,\mathrm{d}\varphi, \tag{21}$$

$$\Delta\theta_{-,i} = \theta_i - \left(\frac{-\theta_{span}}{2}\right), \tag{22}$$

$$\Delta\theta_{-,i} = \int_{\varphi_{-,i}}^{0} \frac{r\,(R+r)\,\cos\alpha_0}{(R+r\cos\varphi)\sqrt{(R+r\cos\varphi)^2-(R+r)^2\cos^2\alpha_0}}\,\mathrm{d}\varphi. \tag{23}$$

In Eq. (21) and Eq. (23), we can see that the angles $\varphi_{-,i}$ and $\varphi_{+,i}$ that we need to compute can be found as limits of the integrals. To extract their values we used the bisection method (see [14], [15], [16]), also known as the dichotomy method. The bisection method is a technique for finding a real-valued root of a continuous function, defined in a closed interval for which endings it is known that the function values have opposite signs, by successively narrowing the range of values inside which the root is known to exist using interval halving. The existence of a root follows from Bolzano's theorem [17]. The bisection method is known to be slowly convergent (see [14], [15]), but here we were not looking for a performance.

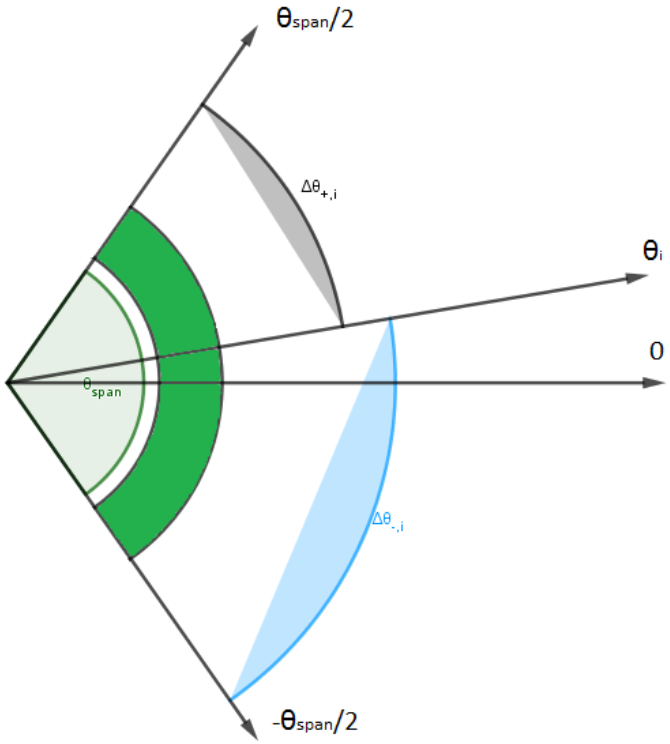

Figure 6. (*Representation of the angles and the angle intervals of* θ_i)

Then, the classical fourth-order explicit Runge-Kutta method (see [15], [16]) was applied to the system of ODEs which consists of Eq. (13) and Eq. (14). The classical fourth-order Runge-Kutta method is an explicit method for numerical solving of an initial value problem or system where the integrand is calculated at four intermediate stages within a time step. That is, the approximation of the solution in point *n+1* is calculated as a sum of the solution in point *n* plus the weighted sum of four addends, each of which is a product of the subinterval size and the slope specified by the right-hand side of the differential equation. The total accumulated error is on the order of $O(h^4)$, where *h* is the interval size.

The last thing we do so as to find the exact trajectory of the *M* number of lines is to numerically solve the Eq. (16) for each line $i = \overline{0, M-1}$ with $\theta_{start} = -\frac{\theta_{span}}{2}$, and $\varphi_i \in [\varphi_{-,i}, \varphi_{+,i}]$.

2.6 The algorithm

To sum up, the steps needed to be done so as an elbow to be tape covered are as follows:

a) compute θ_{cycle} (Eq. (17));

b) compute *M* (Eq. (18));

c) compute $\theta_i, i = \overline{0, M-1}$ (Eq. (19));

d) compute $\varphi_{-,i}$ and $\varphi_{+,i}, i = \overline{0, M-1}$, solving Eq. (21) and Eq. (23) by applying the *fzero* built-in function in the first MATLAB implementation, and the bisection method in the Python implementation and in the second MATLAB implementation;

e) solve Eq. (16) using the *ode45* built-in function in the first MATLAB implementation or the classical fourth-order explicit Runge-Kutta method in the Python implementation, and solve the system of ODEs (13)—(14) with an initial condition α_0 in the second MATLAB implementation;

f) extract the necessary points;

g) plot.

3. Implementations

The algorithm described in **Section 2.6** was implemented in MATLAB [18] and in Python [19] (Anaconda (5.1.0): Py3.6 [20]). The source codes can be found on ResearchGate:

- MATLAB implementation – variant 1:

https://www.researchgate.net/project/Computer-Aided-Filament-Winding-Tape-Placement-for-Elbows-Practically-Orientated-Algorithm/update/5abd30254cde260d15d5b55c

- MATLAB implementation – variant 2:

https://www.researchgate.net/project/Computer-Aided-Filament-Winding-Tape-Placement-for-Elbows-Practically-Orientated-Algorithm/update/5abd3150b53d2f63c3c2cbfb

- Python implementation:

https://www.researchgate.net/project/Computer-Aided-Filament-Winding-Tape-Placement-for-Elbows-Practically-Orientated-Algorithm/update/5abd31994cde260d15d5b58d

Here, we are going to present two of them.

3.1 MATLAB implementation – variant 1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Algorithm for tape placement for elbows.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This source code is licensed under a Creative Commons Attribution 4.0
% International License.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
format long
hold on
clc
clear
global R
global r
global alfa_0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Test values and initial conditions:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R=20;
r=5;
w=1;
alfa_0 = 78*pi/180;
teta_span = 2*pi/3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% f = d_theta/d_phi:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f = @(fi) (r.*(R+r).*cos(alfa_0))./((R+r.*cos(fi)).* ...
           sqrt((R+r.*cos(fi)).^2-((R+r)^2).*(cos(alfa_0)).^2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute teta_cycle (step a)):
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
teta_cycle = quadgk(f,0,2*pi);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute M (step b)):
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = floor(((R+r)*teta_cycle*sin(alfa_0))/(w)) + 1;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the torus:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fi_i = 0;
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4]);
options2 = odeset('RelTol',1e-8,'AbsTol',1e-8);

Teta=linspace(-teta_span/2,teta_span/2,100);
Fi=linspace(0,2*pi,100);
[TetaM,FiM]=meshgrid(Teta,Fi);
X=(R+r*cos(FiM)).*cos(TetaM);
Y=r*sin(FiM);
Z=(R+r*cos(FiM)).*sin(TetaM);

h=surf(X,Y,Z);
axis equal
axis([0 40 -20 20 -20 20])
set(h,'FaceColor','g','EdgeColor','g','FaceAlpha',0.5,'EdgeAlpha',0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute θ_i, (step c):
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
teta_part = teta_span*teta_cycle/2;

for i=1:M
    % Compute θ_i, (step c):
    tita(i) = (i-1)*teta_cycle/M;
    % Compute fi_i
    fun = @(fi_i) (quadgk(f,0,fi_i)-tita(i));
    FI_I(i) = fzero(fun, 0);
    % Compute fi_{-,i} and fi_{+,i} (step d):
    fun = @(fi_start) (quadgk(f,fi_start,0)-(teta_span/2 + i*teta_cycle/M));
    FI_START(i) = fzero(fun, 0);
    fun = @(fi_end) (quadgk(f,0,fi_end)-(teta_span/2 - i*teta_cycle/M));
    FI_END(i) = fzero(fun, 0);
    % (step e):
    [phi,theta] = ode45 (@(phi,theta) r*(R+r)*cos(alfa_0)/(R+r*cos(phi))/ ...
    sqrt((R+r*cos(phi))^2-((R+r)*cos(alfa_0))^2),[FI_START(i) FI_END(i)], ...
    -teta_span/2, options2);
    % (step g):
    plot3((R+r*cos(phi)).*cos(theta), r*sin(phi), ...
          (R+r*cos(phi)).*sin(theta), 'b', 'linewidth',2);
    plot3((R+r*cos(phi)).*cos(theta), -r*sin(phi), ...
          (R+r*cos(phi)).*sin(theta), 'r', 'linewidth',2);
    view([90,0,0])
    F(i) = getframe;
end

[FI_I' tita']
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Play the recorded movie frames:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
movie(F,0)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

3.2 Python implementation

```
##############################################################################
#
# Algorithm for tape placement for elbows.
#
##############################################################################
#
# This source code is licensed under a Creative Commons Attribution 4.0
# International License.
#
##############################################################################
# Importing the needed modules:
##############################################################################
from sympy import *
from mpmath import *
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
##############################################################################
# Function for extracting coordinates from nested lists of angles:
##############################################################################
def xyz(R,r,ph,th):
    m = len(ph)
    s = len(ph[0])
    x = np.empty((m,s)) # numpy matrix
    y = np.empty((m,s))
    z = np.empty((m,s))
    for i in range(m):
        for j in range(s):
            x[i,j] = (R + r * cos(ph[i][j])) * cos(th[i][j])
            y[i,j] = r * sin(ph[i][j])
            z[i,j] = (R + r * cos(ph[i][j])) * sin(th[i][j])
    return (x,y,z)
##############################################################################
# Test values and initial conditions:
##############################################################################
w = 0.5 # tape width
R = 20 # the distance from the center of the tube to the center of the torus
r = 5 # radius of the tube
alpha_0 = 78*pi/180 # initial winding angle
theta_span = 2*pi/3 # span angle of the elbow

##############################################################################
# f = d_theta/d_phi:
##############################################################################
f = lambda phi : r * (R + r) * cos(alpha_0) / (R+r*cos(phi)) \
                 /sqrt((R + r * cos(phi))**2 - (R + r)**2 * cos(alpha_0)**2)
##############################################################################
# Compute theta_cycle (step a)):
##############################################################################
theta_cycle = N(quad(f,[0,float(2 * pi)]))
##############################################################################
# Compute M (step b)):
##############################################################################
M = int(floor((R + r) * theta_cycle * sin(alpha_0) / w) + 1)
```

```
################################################################################
# Compute θ_i, i = 0,M − 1 (step c)):
################################################################################
thetai = []
for i in range(M):
    thetai.append(N(theta_cycle * i / M))
################################################################################
# Declarations of some needed variables:
################################################################################
theta_plus = theta_span/2
theta_minus = -theta_span/2

phi_minus = []
phi_minus_l = []
phi_minus_r = []
phi_plus = []
phi_plus_l = []
phi_plus_r = []
################################################################################
# Initializing phi_minus[i] and phi_plus[i] value pairs for the
# bisection method:
################################################################################
for i in range(M):
    phi_minus_l.append(2.0 * pi * (theta_minus - thetai[i])/theta_cycle - pi)
    phi_minus_r.append(phi_minus_l[i] + 2.0 * pi)
    phi_plus_l.append(2.0 * pi * (theta_plus - thetai[i])/theta_cycle - pi)
    phi_plus_r.append(phi_plus_l[i] + 2.0 * pi)


################################################################################
# Solving the integral equation by bisection method (step d))
# phi_minus[i]:
################################################################################
for i in range(M):
    t = True
    while(t):
        phi_minus_m = 0.5*(phi_minus_l[i] + phi_minus_r[i])
        f_minus_l = thetai[i] - theta_minus - N(quad(f,[phi_minus_l[i],0]))
        f_minus_r = thetai[i] - theta_minus - N(quad(f,[phi_minus_r[i],0]))
        f_minus_m = thetai[i] - theta_minus - N(quad(f,[phi_minus_m,0]))
        if sign(f_minus_l) == sign(f_minus_m):
            phi_minus_l[i] = phi_minus_m
        else:
            phi_minus_r[i] = phi_minus_m
        t = abs(f_minus_m) > 10**-10
    phi_minus.append(phi_minus_m)
################################################################################
# Solving the integral equation by bisection method (step d))
# phi_plus[i]:
################################################################################
for i in range(M):
    t = True
    while(t):
        phi_plus_m = 0.5 * (phi_plus_l[i] + phi_plus_r[i])
        f_plus_l = theta_plus-thetai[i] - N(quad(f,[0,phi_plus_l[i]]))
        f_plus_r = theta_plus-thetai[i] - N(quad(f,[0,phi_plus_r[i]]))
        f_plus_m = theta_plus-thetai[i] - N(quad(f,[0,phi_plus_m]))
```

```
        if sign(f_plus_l) == sign(f_plus_m):
            phi_plus_l[i] = phi_plus_m
        else:
            phi_plus_r[i] = phi_plus_m
        t = abs(f_plus_m) > 10**-10
    phi_plus.append(phi_plus_m)

steps = 500
phi_span = zeros(M,1)
d_phi = zeros(M,1)
for i in range(M):
    # we have different angle spans for each line:
    phi_span[i,0] = phi_plus[i] - phi_minus[i]
    # we have different step size for each line:
    d_phi[i,0] = phi_span[i,0]/steps

PHI = []
for i in range(M):
    # nesting lists inside list:
    PHI.append([])
for i in range(M):
    for j in range(steps+1):
        # filling the lists with angles:
        PHI[i].append(phi_minus[i] + j * d_phi[i,0])

THETA = []
for i in range(M):
    # nesting lists inside list, each internal list starts with theta_minus:
    THETA.append([theta_minus])
##############################################################################
# 4th-order Runge-Kutta for every line (step e)):
##############################################################################
for i in range(M):
    for j in range(steps):
        k1 = f(PHI[i][j])
        k2 = f(PHI[i][j] + d_phi[i,0]/2)
        k3 = f(PHI[i][j] + d_phi[i,0]/2)
        k4 = f(PHI[i][j] + d_phi[i,0])
        # adding elements (angles) to nested list:
        THETA[i].append(THETA[i][j] + d_phi[i,0] * (k1 + 2*k2 + 2*k3 + k4)/6)
##############################################################################
# Extracting the coordinates of the points (step f)):
##############################################################################
X,Y,Z = xyz(R,r,PHI,THETA)
##############################################################################
# Visualization of the tape placement (step g)):
##############################################################################
fig = plt.figure(figsize=[16,16])
ax  = fig.add_subplot(1,1,1,projection='3d')
for i in range(M):
    #right hand screw:
    ax.plot(X[i,:],Y[i,:],Z[i,:],color='#ff0000',linewidth=0.4)
    # left hand screw:
    ax.plot(X[i,:],Y[i,:],-Z[i,:],color='#0000ff',linewidth=0.4)
ax.set_xlim(-10,30)
ax.set_ylim(-20,20)
ax.set_zlim(-20,20)
```

```
ax.xaxis.pane.fill=False
ax.yaxis.pane.fill=False
ax.zaxis.pane.fill=False
ax.xaxis.pane.set_edgecolor('white')
ax.yaxis.pane.set_edgecolor('white')
ax.zaxis.pane.set_edgecolor('white')
ax.view_init(10,225)
ax.grid(b = None)
plt.show()
###########################################################################
# End
###########################################################################
```

4. Results and discussion

The following test values and initial conditions were used as input values for the algorithm:

$$R = 20;\ \ r = 5;\ \ w = 0.5;\ \ \alpha_0 = 78°;\ \ \theta_{span} = 120°. \tag{23}$$

In this case, the number of needed tapes so as the whole surface of the elbow to be completely covered is $M = 23$.

The results achieved with our algorithm can be seen in **Figure 7** and **Figure 8**.

Figure 7. (*Tape placement with two fibers.*)

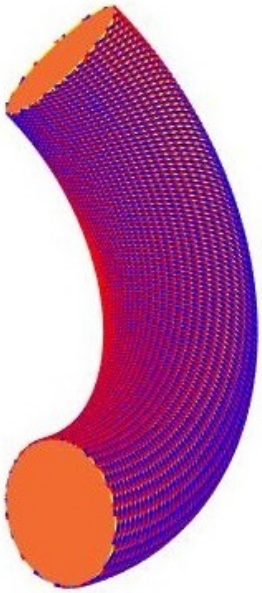

Figure 8. (*Complete tape placement.*)

The generated videos which show how the tape placement of an elbow works can be found on ResearchGate: https://www.researchgate.net/project/Computer-Aided-Filament-Winding-Tape-Placement-for-Elbows-Practically-Orientated-Algorithm/update/5abd33aab53d2f63c3c2cc6c
They were both generated using MATLAB (the second implementation).

In the previous section we presented two implementations of the algorithm – in MATLAB and in Python. Both of them have advantages and disadvantages. The former is better regarding the aspect that it relies on built-in functions exclusively. This means that the user does not have to know the details about the applied mathematical methods. In this sense, our code is encapsulated and hence more user-friendly. One huge

disadvantage is that the user needs to have a license so as to use MATLAB. On the other hand, the Python code and Python as a whole are open-source. However, the user needs to be able to implement all the needed methods themselves.

5. Concluding remarks and future work

In the process of solving this problem, we have tried different approaches for getting to the solution, so the solution and its code presented in this paper was selected by our choice as the simplest, although it was not the most efficient one. The solution as such is appropriate for tape laying, but not for filament winding, because we have not included dwelling. The main goal of our presented solution was to make it and its code a good and approachable introduction to filament winding and tape laying algorithms and the obstacles while delivering a complete filament winding solution.

Except the presented program codes in MATLAB and Python, we have made one more program code in MATLAB which differs from the presented one in the approaches to the numerical solving of the equations. The results from all three program codes are satisfactory. In future, we are going to try to implement dwelling as well.

The whole project is available on ResearchGate:
https://www.researchgate.net/project/Computer-Aided-Filament-Winding-Tape-Placement-for-Elbows-Practically-Orientated-Algorithm

Figures 2--6 in this paper have been generated using GeoGebra [21].

Acknowledgements The authors want to express their gratitude to the Cost Action TD1409 Grant.

References

[1] Mallick, P.K. (1993). *Fiber-Reinforced Composites: Materials, Manufacturing, and Design*. Taylor & Francis, 2nd edition.

[2] Hai-sheng, L., You-dong, L. (2002). *Computer Aided Filament Winding for Elbows:* Journal of Software, 13(4).

[3] Skjærholt, I. (2012). *Integration Tools for Design and Process Control Of Filament Winding,* Master thesis, Norwegian University of Science and Technology, Institutt for produktutvikling og materialer.

[4] Lei, Z., Qin-Xiang, H., Qing-Qing, N. (2007). *Pattern Design for Non-geodesic Winding Toroidal Pressure Vessels,* CD-ROM Proceedings of The Sixteenth International Conference on Composite Materials.

[5] Seereeram, S. (1991). *An All-geodesic Algorithm for Filament Winding of a T-shaped Form,* IEEE Transactions on Industrial Electronics, 38(6), pp. 484—490.

[6] Koussious, S. (2004). *Filament Winding: a Unified Approach,* PhD thesis report, Delft University Press, Delft, ISBN 90-407-2551-9.

[7] Mikrosam A.D., 2008. *MAW 20 FB5/1 and MAW 20 LS6/1 Winding Expert Module – S Manual. http://mikrosam.com/new/article/en/winding-expert/.*

[8] Cadfil® -- Filament Winding Software & Technology *CADFIL-Elbow Winding. http://www.cadfil.com/.*

[9] Furtado, R. L. A. (2016). *Filament Winding Simulation,* Master thesis, University of Porto, Porto.

[10] Irons, M. The Curvature and Geodesics of the Torus, *http://www.rdrop.com/~half/math/torus/index.xhtml.*

[11] Petkanchin, B. (1964). *Diferencialna geometria,* Nauka i izkustvo, Sofia, 2nd edition, 802 p. (in Bulgarian).
[12] Ivanova-Karatopraklieva, I. (1994). *Diferencialna geometria,* Sofia University Press, Sofia, 432 p. (in Bulgarian).
[13] Angenent, S. B., *Liouville's Formula* in Math 561 — Differential Geometry (with special relativity). Notes, *http://www.math.wisc.edu/~angenent/561/*.
[14] Demirovich, B. P., Maron, I. A., (1966). *Osnovy vychislitelnoy matematiki*, 3rd edn., Nauka, Moscow, 664 p. (in Russian).
[15] Samarskii, A., Goolin, A. (1989). *Chislennye Metody,* Nauka, Moscow, 432 p. (in Russian).
[16] Süli, E., Mayers, D. F. (2003). *An Introduction to Numerical Analysis.* Cambridge University Press, pp. 433.
[17] Apostol, T. M. (1967). *Calculus, 2nd ed., Vol. 1: One-Variable Calculus, with an Introduction to Linear Algebra.* Waltham, MA, Blaisdell, p. 143.
[18] MATLAB and Statistics Toolbox Release 2015, *The MathWorks*, Inc., Natick, Massachusetts, United States, *https://www.mathworks.com/*.
[19] Python Core Team (2015). *Python: A dynamic, open source programming language*. Python Software Foundation, *https://www.python.org/*.
[20] Anaconda Software Distribution. *Computer software. Vers. 5.1.0. Anaconda, 2018.* *https://anaconda.com*.
[21] GeoGebra Classic 5.0.433.0-d 2018, *http://www.geogebra.org*.