POLITECNICO DI TORINO

Dipartimento di Ingegneria Strutturale, Edile e Geotecnica

Master of Science in Civil Engineering

Master's Degree Thesis

Optimization of shape and arch cross-section for arch bridges

Supervisor:

prof. MARANO, Giuseppe Carlo

Co-supervisor:

prof. MANUELLO BERTTETO, Amedeo

Candidate

SUAREZ, Lisandro

July 2020

Acknowledgments

Firstly, I would like to thank to the Universidad Nacional de Córdoba for forming me both academically and personally, and for giving me the opportunity to finish my degree abroad where I expanded my own horizons. I hope to return some of all that it gave me. Also, to every person I met during the studies in Córdoba.

I am very grateful to the Politecnico di Torino and all their professors for receiving me and making me feel like home. Especially, I would like to thank to my supervisor Professor Giuseppe Carlo Marano for guiding me during the development of this work and to Professor Amedeo Manuello who also help us. And, also, to every wonderful person I met here in Italy, they made this experience unforgettable.

I would like to express my gratitude to my family for supporting me during this long trip. To my parents, Tato and Alicia, who gave me the possibility to study away from home, none of this would have been possible without them. To my brothers, Eze and Nacho, who supported me during 4 years in Córdoba. And, to my grandmother Lola and my aunt Elena who also helped to make this possible.

Finally, to all my friends who, despite the distance, they are still there supporting me. I cannot name them, for sure I forget someone, but they know who they are. I hope to see them soon.

Lisandro Suarez

Abstract

Arches are structural members which have been developed for a very long time in building field. At the beginning, they were designed with limited materials and a conservative criterion, on the safety side, that led to material wasting and to a limited maximum span length range. Over time, the amount of material used to build them has been considerably decreasing because of new structural materials and technologies have emerged such that, nowadays, there is a better and proper use of materials related to their strength allowing a greater maximum span length range. Currently, engineers deal with a vast range of solutions and materials, therefore, they must look for an optimal solution in terms of cost and resistance.

The aim of this work is to find the optimal shape and cross-section of an arch, subjected to self-weight and different load conditions, by minimizing the total volume of the arch and satisfying the design requirements. To achieve it, a code written in MATLAB, which numerically solves the differential equation system of the arch, is proposed in which displacements and internal forces can be obtained and thus analyse how stresses change and minimize the arch volume by varying the geometrical parameters.

Contents

Table of figures

Chapter 1

1 Introduction

1.1 General overview

Finding the optimal solution between many others is a typical work for engineers nowadays. The reason falls in a correct use of the material to avoid elevated costs and resistance waste. Currently, the technologies and new materials allow us to have a vast range of solutions which is great, but, at the same time, it forces us, as engineers, to design a proper structure minimizing cost and highlighting the main attributes of the materials.

The purpose of this work will not be the final solution for any project that has to be built, but it may be a great tool for pre-dimensioning in such a way the designer, knowing the constant geometrical constraints of the project, can obtain an optimal design for that values in a quicker and efficient way, and start the design with an optimal solution.

1.2 Main objectives

The main objective is to find the optimal shape and arch cross-section for a given span L, cross-section external radius r and material. The cross-section to be considered is a steel circular full section. Therefore, the parameters to vary are the initial angle θ_i (with a subsequent variation of the radius of curvature R, assumed as constant for sake of simplicity), and the cross-section radius r. By varying these parameters, different displacement and internal actions will be obtained throughout a MATLAB's code. Thus, the idea is to minimize the total volume of the arch respecting certain design requirements for maximum normal stresses. To achieve the aim, a genetic algorithm was applied which finds the solution in a population of points, and not in a single point like standard algorithms.

A general result can be pursued by plotting dimensionless parameters, therefore an independence from the span length is attained. In this way, by fixing all the parameters and by varying the load conditions, optimal values for initial angle and radius will be obtained and will compose a curve. The dimensionless chart will be plotted with curvature radius-to-span length ratio (*R/L*) in x axis versus rise-to-span length ratio (*f/L*) in y axis. Then, several curves can be obtained by varying the span length.

1.3 Thesis structure

The thesis is composed of five chapters, without considering this. Firstly, arches and curved beam theory will be introduced in Chapter 2 in which will be established the static, kinematic and constitutive equations to obtain the differential equations which give the three possible displacements of an arch.

Afterwards, in Chapter 3, the Boundary-Value Problem (BVP) is introduced and the algorithm used to solve the differential equations in MATLAB is explained.

In Chapter 4, an isolated arch is analysed defining the different parameters and transforming the differential equations into a first-order differential equation system in order to be able to use the solver proposed by MATLAB.

Then, the optimization process is described in Chapter 5. The idea is to get the maximum efficiency between the loads which the arch can carry and its strength by minimizing the total volume of the arch as long as the material strength is fulfilled. To this point, a design vector is defined containing the initial angle and section radius which will be the values to be varied, and the several configurations are evaluated to get the optimum solution in an algorithm.

Finally, in Chapter 6 the results are discussed, and conclusions are made.

Moreover, the codes will be attached in the annex part.

Chapter 2

2 Arches and curved beam theory

In this chapter historical and introductory aspects about arches and curved beam theory are presented. The idea is to highlight the most important aspects of arches and to obtain the differential equations that govern its behaviour. Therefore, static, kinematic, and constitutive equations are presented and substituted between them in such a way as to a system of three differential equations is obtained in function of the possible displacements (normal, tangential and rotation) that the arch can undergo. After that, some simplifications can be performed in such a way the differential equations are reduced to a unique equation in function of the tangential displacement.

2.1 Introduction

Arch structures have been used in construction field, for its structural capability and its aesthetical purposes, for a long time. The arches are curved structural elements whose transversal section is negligible against their length and can cover large spans. Generally, they are made of different materials as masonry, concrete, structural steel, among others.

Today, between different uses one of the most important is the bridge construction. Arch bridges are one of the older sorts of bridges and can be found everywhere due to their functionality, aesthetic, and their natural strength. At the beginning, they were erected by stones and their maximum span was limited. From the past to today, the introduction of new structural materials, like concrete and steel, allows arch bridges to cover larger spans.

In the following figures, some famous steel arch bridges are presented. In figure 2-1, the *Sydney Harbour Bridge* (1932, Australia) is displayed. It is one of the most famous steel truss arch bridge. The arch has a maximum span of 502 m. and a rise from the water to the top of 134 m.

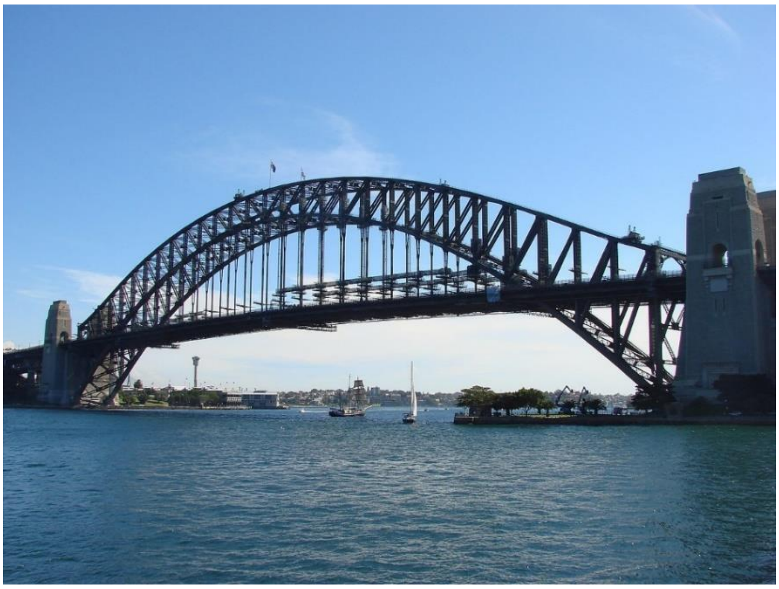

Figure 2-1. Sydney Harbour Bridge (Australia) (taken from https://it.wikipedia.org/wiki/Sydney_Harbour_Bridge)

In Figure 2-2, it is presented the *Chaotianmen Bridge* (2009, China) which is, currently, the world's longest through arch bridge with a maximum span of 552 m. It is composed by steel truss and tie girders. The arch has a rise of 140 m from middle supports to arch top.

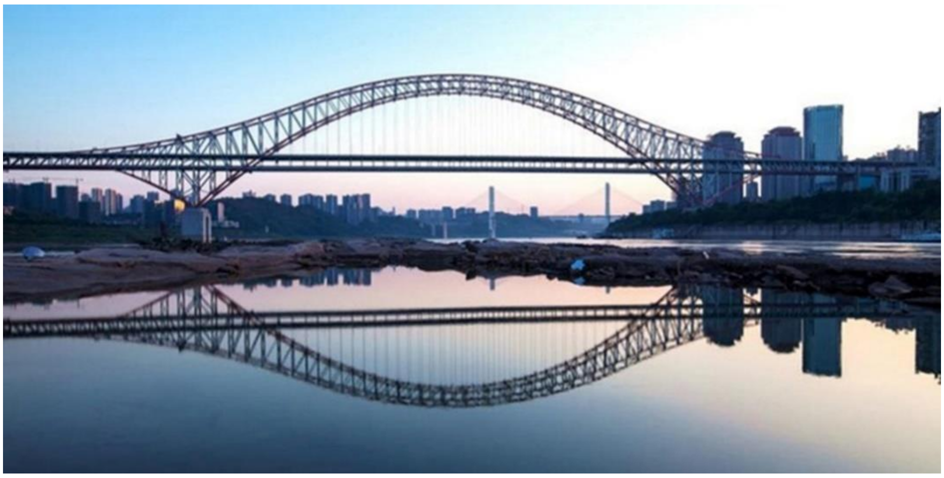

Figure 2-2. Chaotianmen Bridge. (2009) (taken from https://www.ichongqing.info/)

In Figure 2-3 is shown the world's longest arch concrete bridge also located in China. It is the *Qinglong Railway Bridge* (2016) which has a maximum span of 445 m and it has a clearance of 295 m.

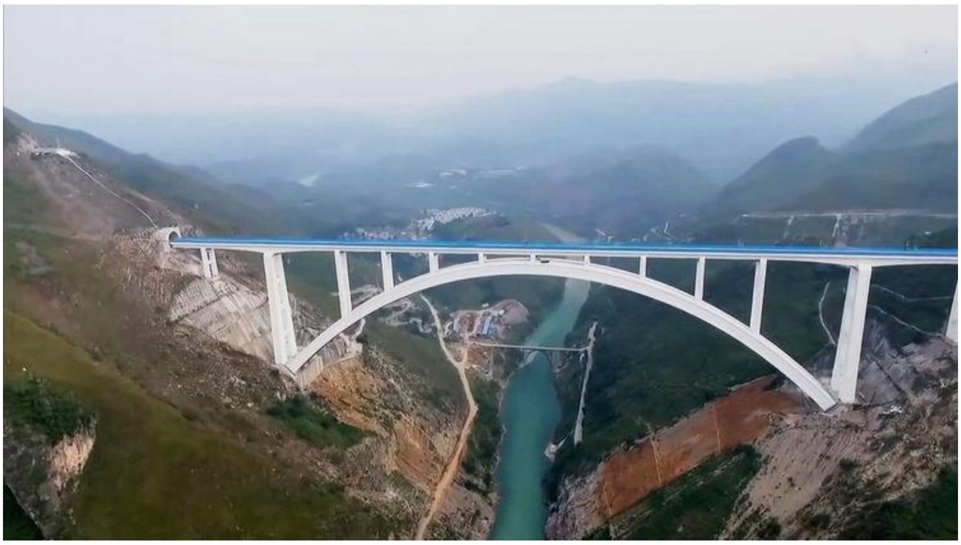

Figure 2-3. Qinglong Railway Bridge (taken from http://www.highestbridges.com/)

The basic principle of this kind of bridge is its curved shape able to carry the load along the curve to the constraints disposed at each end, called bridge abutments. The inclined force coming from the arch is supported by the abutments and it is called thrust and it increases as the rise of the arch decreases. The thrust must be restrained in order to avoid a collapse of the arch and to allow arch to be able to self-supporting. Arches can sustain large vertical loads and they mainly work in compression; therefore, the tensile stresses are normally minimized.

The main feature of an arch from the structural point of view is that even if the structure is preferentially subjected to vertical loads, horizontal reactions may take place. Therefore, the presence of the horizontal thrust is the main difference in mechanical behaviour between arches and beams.

Depending on the position of the deck, the bridge can be categorized as:

- Deck arch bridge: The deck is located above the arch and it is supported by columns rising from the arch.
- Through arch bridge: The deck is below the arch and it is suspended by means of hangers which are steel members subjected to tension.
- Half-through arch bridge: The deck is located at an intermediate position, between the above solutions, and it is partially connected by columns and partially by hangers.

The three kind of bridges are shown below in Figure 2-1.

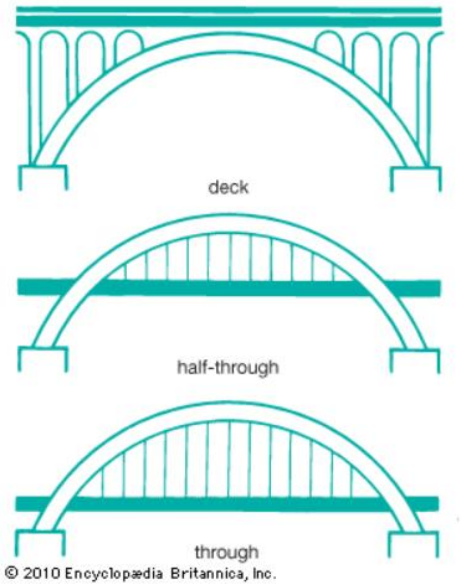

Figure 2-4. Types of arch bridge (taken from https://kids.britannica.com/students/)

There is another solution to absorb the external forces instead of using abutments and it is including a tie between the arch ends. In this way, the force is sustained by the tie in tension, and it is called tied arch bridge. For this case, the deck by through arch or half-through arch can work as a tie by tensile strength. This is a good solution when the soil presents poor sustaining characteristics.

The structural analysis, as in all the cases, is performed to an idealized and simplified shape of the structure so that the axial line of the structural components is representing the actual geometries.

2.2 Curved beam theory

Generally, arch can be studied as a curved beam that is a structural element whose axis is not straight, but it has a certain curvature. So, it is defined as a polar unidimensional continuum.

The different portions composing the beam do not necessarily present the same curvature, it will depend on the curvilinear axis, called "s", defined as the coordinate going along the curved beam axis. This coordinate will easily describe the element. As a sake of simplicity, a constant radius R will be considered, therefore the increment ds can be expressed as follows:

$$ds = Rd\theta \tag{2.1}$$

Where R is the radius of curvature of the arch middle line and $d\theta$ is the angle between the normal vectors of belonging to two sections.

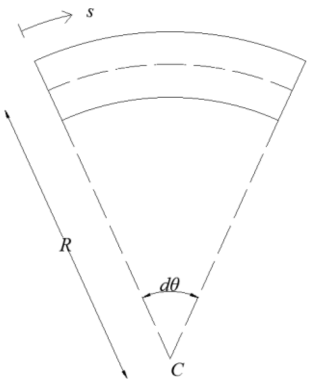

Figure 2-5. Definition of the curvilinear axis s.

In order to study the behaviour of the curved beam subjected to different load conditions, it is necessary to get the kinematic equations and static ones. Moreover, constitutive relationships will be used to relate them.

2.3 Static equations

The following development to get the differential equations is based on Scienza delle Costruzioni (Carpinteri) [2].

An infinitesimal element of a curved beam defined with a certain width well described by the angle *dθ* and a constant curvature radius *R* is considered.

To make a static analysis is necessary to identify the generalised static quantities *N, T and M*, which makes the static problem satisfying the dual kinematic condition. These quantities are shown in Figure 2-64. With *C* is represented the centre of curvature of the curvilinear axis.

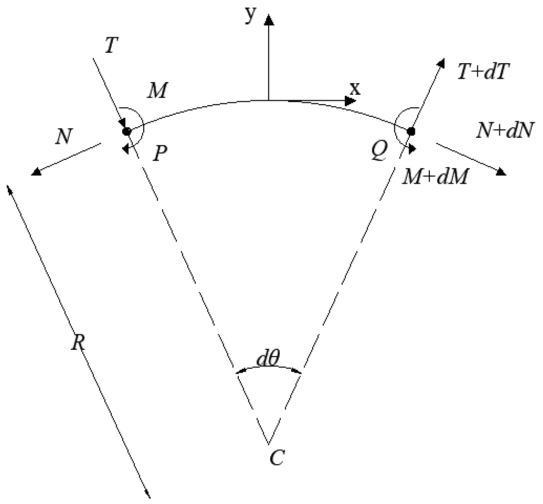

Figure 2-6. Generalized static quantities on the curved geometry described by the axis.

An equilibrium analysis is made to the infinitesimal element regarding the three degrees of freedom. The reference system is fixed with the origin at the barycentre centre of the element, x-direction tangent to the axis of the beam and radial y-direction.

The element is subjected to generic forces in x-direction, *Px,* and y-direction, *Py.* The equilibrium in x-direction, y-direction and to the rotation regarding the barycentre are reported in the following. As the angle $d\theta$ is considered very small, the cosine and sine can be assumed as:

$$\cos(d\theta) \cong 1$$
$$\sin(d\theta) \cong d\theta$$

- Equilibrium in x-direction:

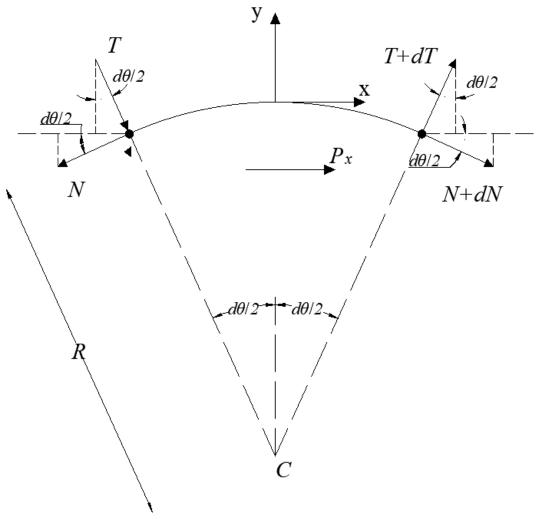

Figure 2-7. Acting forces in x-direction.

$$\sum F_x = 0 \Rightarrow -N + N + dN + T\frac{d\theta}{2} + (T + dT)\frac{d\theta}{2} + P_x ds = 0$$

$$\frac{dN}{ds} + \frac{T}{r} + P_x = 0 \qquad (2.2)$$

- Equilibrium in y-direction:

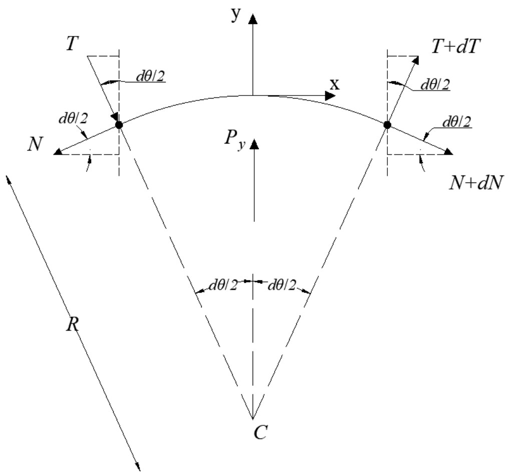

Figure 2-8. Acting forces in y-direction.

$$\sum F_y = 0 \Rightarrow -T + T + dT - N\frac{d\theta}{2} - (\mathrm{N}+ dN)\frac{d\theta}{2} + P_y ds = 0$$

$$\frac{dT}{ds} - \frac{N}{r} + P_y = 0 \qquad (2.3)$$

- Rotational equilibrium respect to the barycenter

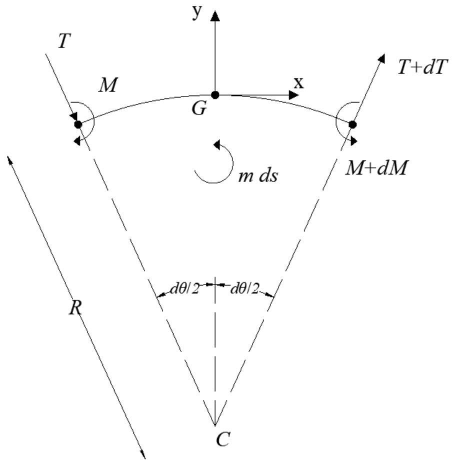

Figure 2-9. Moment and forces generating moment around G.

$$\sum M_G = 0 \Rightarrow -M + M + dM + T\frac{ds}{2} + (\mathrm{T}+ dT)\frac{ds}{2} + m \cdot ds = 0$$

$$\frac{dM}{ds} + T + m = 0 \qquad (2.4)$$

At this point, the indefinite equations of equilibrium were obtained and can be compacted in matrix form as:

$$\begin{bmatrix} -\frac{d}{ds} & -\frac{1}{r} & 0 \\ \frac{1}{r} & -\frac{d}{ds} & 0 \\ 0 & -1 & -\frac{d}{ds} \end{bmatrix} \begin{Bmatrix} N \\ T \\ M \end{Bmatrix} = \begin{Bmatrix} P_x \\ P_y \\ m \end{Bmatrix} \tag{2.5}$$

2.4 Kinematic equations

Firstly, the possible displacements that can take place respect to the different points constituting the curved beam axis will be defined as follows:

u: displacement along the tangent at the point under examination.

v: displacement along the normal direction to the point under examination.

φ: rotation of section passing through the point under examination.

The element before deformation is described by three points P, Q and C. While after deformation, the same point in the changed position will be identified by P’, Q’ and C’.

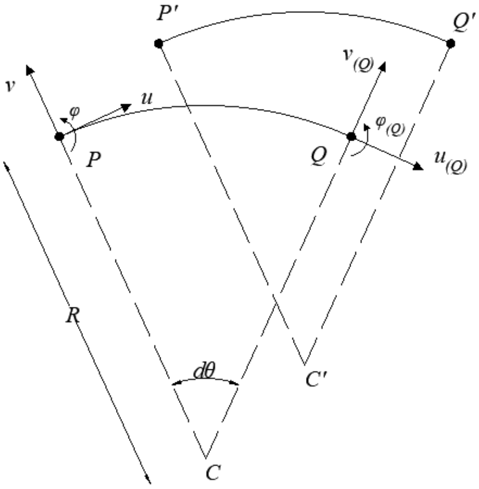

Figure 2-10. Infinitesimal element before and after deformation.

It is necessary to find the generalised deformations related to the dual kinematic quantities respect the static ones by subtracting the rigid ones from the total displacements of Q, and thus obtaining the pure deformations.

The total displacements of Q are written as:

$$\begin{Bmatrix} u_{(\mathrm{Q})} \\ v_{(\mathrm{Q})} \\ \varphi_{(\mathrm{Q})} \end{Bmatrix} = \begin{Bmatrix} u \\ v \\ \varphi \end{Bmatrix} + \begin{Bmatrix} \dfrac{du}{ds} ds \\ \dfrac{dv}{ds} ds \\ \dfrac{d\varphi}{ds} ds \end{Bmatrix} \quad (2.6)$$

Ergo, they are equal to the sum of the displacements of P plus the infinitesimal displacements increasing in PQ.

By applying the three kind of displacement that a curved beam can undergo, the rigid displacements of point Q can be computed.

Applying a rigid displacement *u* to point P.

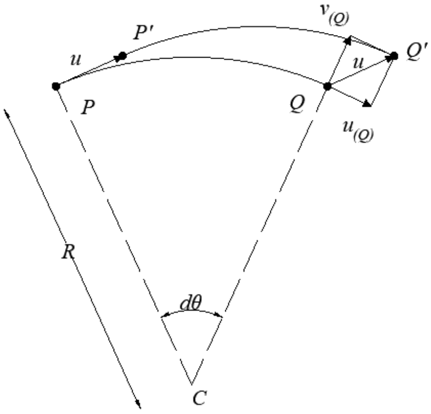

Figure 2-11. Rigid displacement u applied to point P.

At Q, there will be two contributes of rigid displacement. One of them parallel to $u_{(Q)}$ and the another one to $v_{(Q)}$, defined as:

$$u_{(Q)} = u \cdot \cos(d\theta)$$

$$v_{(Q)} = u \cdot \sin(d\theta)$$

Then, a rigid displacement *v* is applied to point P.

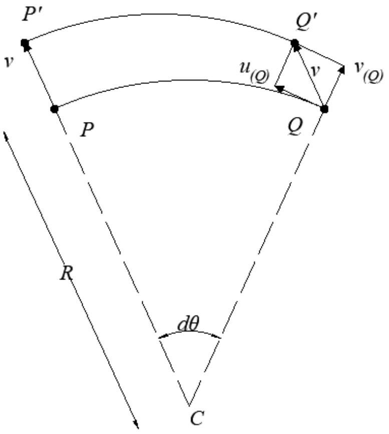

Figure 2-12. Rigid displacement v applied to point P.

This time, the contributes at Q will be:

$$u_{(Q)} = -v \cdot \sin(d\theta)$$

$$v_{(Q)} = v \cdot \cos(d\theta)$$

Finally, a rigid rotation is applied to P equal to φ.

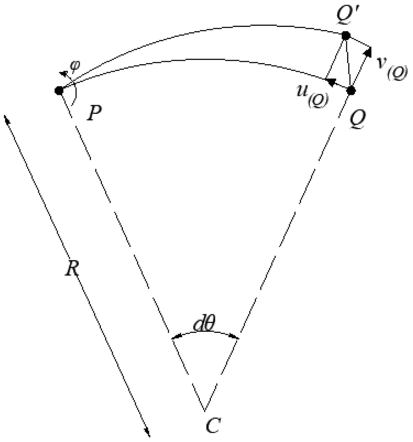

Figure 2-13. Rigid rotation φ applied to point P.

As before, there will be two contributes of rigid displacement equal to:

$$u_{(Q)} = -\varphi \cdot ds \cdot \sin(d\theta)$$

$$v_{(Q)} = \varphi \cdot ds \cdot \cos(d\theta)$$

by approximating $QQ' \cong \varphi \cdot ds$.

The localisation of point Q, after the application of a generic rigid displacement to point P, is easily computed by summing all the contributes above written:

$$\begin{Bmatrix} u_{(Q)R} \\ v_{(Q)R} \\ \varphi_{(Q)R} \end{Bmatrix} = \begin{Bmatrix} u \cdot \cos(d\theta) - v \cdot \sin(d\theta) - \varphi \cdot ds \cdot \sin(d\theta) \\ v \cdot \cos(d\theta) + u \cdot \sin(d\theta) + \varphi \cdot ds \cdot \cos(d\theta) \\ \varphi \end{Bmatrix}$$

As $d\theta$ is considered very small, the above equations become:

$$\cos(d\theta) \cong 1$$
$$\sin(d\theta) \cong d\theta$$

And by neglecting infinitesimal of higher order, the total rigid displacements of point Q are obtained:

$$\begin{Bmatrix} u_{(Q)R} \\ v_{(Q)R} \\ \varphi_{(Q)R} \end{Bmatrix} = \begin{Bmatrix} u \\ v \\ \varphi \end{Bmatrix} + \begin{Bmatrix} -v \cdot d\theta \\ u \cdot d\theta + \varphi \cdot ds \\ 0 \end{Bmatrix} \qquad (2.7)$$

Finally, the deformations are obtained as the difference between the total displacement of the generic point Q and the rigid ones:

$$\begin{Bmatrix} du_{(Q)} \\ dv_{(Q)} \\ d\varphi_{(Q)} \end{Bmatrix} = \begin{Bmatrix} \dfrac{du}{ds} ds - (-v \cdot d\theta) \\ \dfrac{dv}{ds} ds - (u \cdot d\theta + \varphi \cdot ds) \\ \dfrac{d\varphi}{ds} ds \end{Bmatrix} \qquad (2.8)$$

Expressing $d\theta = \dfrac{ds}{r}$, the generalized deformations are obtained:

$$\begin{Bmatrix} du_{(Q)} \\ dv_{(Q)} \\ d\varphi_{(Q)} \end{Bmatrix} = \begin{Bmatrix} \dfrac{du}{ds} + \dfrac{v}{r} \\ \dfrac{dv}{ds} - \dfrac{u}{r} - \varphi \\ \dfrac{d\varphi}{ds} \end{Bmatrix} ds$$

$$\Rightarrow \begin{Bmatrix} \varepsilon \\ \gamma \\ \kappa \end{Bmatrix} = \begin{Bmatrix} \dfrac{du}{ds} + \dfrac{v}{r} \\ \dfrac{dv}{ds} - \dfrac{u}{r} - \varphi \\ \dfrac{d\varphi}{ds} \end{Bmatrix} \qquad (2.9)$$

It is possible to make a physical analysis to understand the added terms that do not appear in a straight beam.

The term v/r in ε represents the extensional deformation of the arch. In fact, if a displacement *v* (along the normal direction) is applied at P and Q, as in Figure 2-14, an increasing in the length of the arch is appreciated. And it is equal to the difference between the length of the deformed arch and to the non-deformed one divided by the initial length:

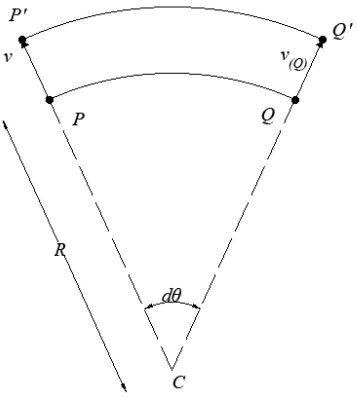

Figure 2-14. Axial deformation ε

$$\varepsilon = \frac{(r+v)\cdot d\vartheta - r\cdot d\vartheta}{r\cdot d\vartheta} = \frac{v}{r}$$

So, as it is observed, ε does not depend on the displacement tangent to the axis. Anyways, it depends on *v* and represents the elongation of the arch.

While the term u/r in γ represents the shear deformation of the arch. In fact, if a displacement *u* (along the axis) is applied at P and Q, as in Figure 2-15, it is observable the arch does not change his length, but it suffers a rotation equal to $\varphi = u/r$. Concluding, the arch undergoes a shear deformation.

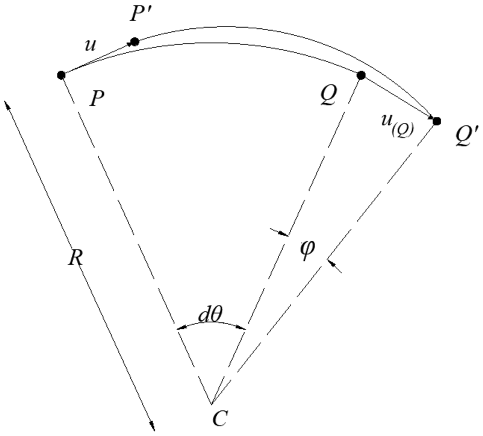

Figure 2-15. Shear deformation γ.

The dual condition between both analyses, static and kinematic, is verified. Since the static matrix is the transpose of the kinematic one, excepting for the signs of the terms that appear with an odd derivative order.

$$\varepsilon = D\cdot u \Rightarrow \begin{Bmatrix} \varepsilon \\ \gamma \\ \kappa \end{Bmatrix} = \begin{bmatrix} \frac{d}{ds} & \frac{1}{r} & 0 \\ -\frac{1}{r} & \frac{d}{ds} & -1 \\ 0 & 0 & \frac{d}{ds} \end{bmatrix} \begin{Bmatrix} u \\ v \\ \varphi \end{Bmatrix} \tag{2.10}$$

$$D^* \cdot \sigma = b \Rightarrow \begin{bmatrix} -\frac{d}{ds} & -\frac{1}{r} & 0 \\ \frac{1}{r} & -\frac{d}{ds} & 0 \\ 0 & -1 & -\frac{d}{ds} \end{bmatrix} \begin{Bmatrix} N \\ T \\ M \end{Bmatrix} = \begin{Bmatrix} P_x \\ P_y \\ m \end{Bmatrix} \tag{2.11}$$

2.5 Constitutive relationships and boundary conditions

The constitutive relationships between the strains of the curved beam and the static quantities are expressed by the Hooke's law as follows:

$$\sigma = E\cdot\varepsilon \Rightarrow \begin{Bmatrix} N \\ T \\ M \end{Bmatrix} = \begin{bmatrix} EA & 0 & 0 \\ 0 & GA_T & 0 \\ 0 & 0 & EI \end{bmatrix} \begin{Bmatrix} \varepsilon \\ \gamma \\ \kappa \end{Bmatrix} \tag{2.12}$$

The boundary conditions must be defined and are the same imposed to the polar unidimensional continuum. The conditions are imposed to the constrained boundary (kinematic) and to the free boundary (mechanical).

$$\begin{aligned} u &= \bar{u} & \pm N &= f_x \\ v &= \bar{v} & \pm T &= f_y \\ \varphi &= \bar{\varphi} & \pm M &= \mu \end{aligned} \tag{2.13}$$

2.6 Elastic equilibrium problem

The elastic equilibrium problem is defined throughout the above obtained equations that are the equilibrium equations (2.5), the congruence equations (2.9) and the constitutive relationships (2.12).

The system of differential equations is obtained by substituting the Eq. (2.9) into Eq. (2.12).

$$\begin{aligned} N &= EA\left(\frac{du}{ds}+\frac{v}{R}\right) \\ T &= GA_T\left(\frac{-u}{R}+\frac{dv}{ds}-\varphi\right) \\ M &= EI\frac{d\varphi}{ds} \end{aligned} \tag{2.14}$$

Finally, the Eq. (2.14) is substituted into (2.5). So, the system of differential equation that describes the behavior of a curved beam is the following:

$$\begin{aligned} &-EA\left(\frac{d^2u}{ds^2}+\frac{1}{R}\frac{dv}{ds}\right)-GA_T\left(-\frac{u}{R^2}+\frac{1}{R}\frac{dv}{ds}-\frac{\varphi}{R}\right)=P_x \\ &EA\left(\frac{1}{R}\frac{du}{ds}+\frac{v}{R^2}\right)-GA_T\left(-\frac{1}{R}\frac{du}{ds}+\frac{d^2v}{ds^2}-\frac{d\varphi}{ds}\right)=P_y \\ &-GA_T\left(-\frac{u}{R}+\frac{dv}{ds}-\varphi\right)-EI\frac{d^2\varphi}{ds^2}=m \end{aligned} \tag{2.15}$$

It is a 6th order differential equation system as the three variables u, v and φ are derived twice with respect to *s*. Therefore, six boundary conditions are required to obtain the desired solution.

2.7 Approximations: Negligible shear effect and inextensibility condition

It can be possible to reduce the problem to one variable-dependence by means of some simplifications.

The displacement method gives a differential equations system which cannot be analytically solved. Therefore, a few simplifications will be applied. The first one is to consider the radius *R* constant and equal to *R* (arch). This means a condition of non-shear strain.

Imposing the non-shear strain condition means:

$$\gamma = 0$$

So, the second equation of Eq. (2.9) becomes:

$$\gamma = \frac{dv}{ds} - \frac{u}{R} - \varphi \Rightarrow \varphi = \frac{dv}{ds} - \frac{u}{R} \tag{2.16}$$

For simplicity issues, a curved beam with a constant radius *R* is considered. This simplification does not carry to important losses in the model.

The kinematic problem becomes only function of translational variables. Thus, it is reduced into a two equations system:

$$\begin{aligned} \varepsilon &= \frac{du}{ds} + \frac{v}{R} \\ \kappa &= \frac{d\varphi}{ds} = \frac{d^2v}{ds^2} - \frac{1}{R}\frac{du}{ds} \end{aligned} \tag{2.17}$$

where the second and third equation of Eq. (2.9) have been reduced into a one by derivation of the Eq. (2.16). The equation system can be written in operational form:

$$\begin{Bmatrix} \varepsilon \\ \kappa \end{Bmatrix} = \begin{bmatrix} \dfrac{d}{ds} & \dfrac{1}{R} \\ -\dfrac{1}{R}\dfrac{d}{ds} & \dfrac{d^2}{ds^2} \end{bmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix} \tag{2.18}$$

Regarding the static part, the condition of non-shear strain does not mean that the shear contribution will be equal to zero, since in order to get a non-deformable beam, there must be an effort such to undo the eventual deformations. Analytically, there is the constitutive relationship:

$$T = GA_T \cdot \gamma \tag{2.19}$$

where the deformation is $\gamma = 0$ but the stiffness $GA_T = \infty$. So, now it is possible to reduce, with respect to the variable *T*, the undefined equilibrium equations. From the third equation of Eq. (2.5):

$$T = -\frac{dM}{ds} \Rightarrow \frac{dT}{ds} = -\frac{d^2M}{ds^2} \tag{2.20}$$

substituting into the firsts two equations of Eq. (2.5):

$$\begin{aligned} &-\frac{dN}{ds} - \frac{T}{R} = P_x \Rightarrow -\frac{dN}{ds} + \frac{1}{R}\frac{dM}{ds} = P_x \\ &\frac{N}{R} - \frac{dT}{ds} = P_y \Rightarrow \frac{N}{R} + \frac{d^2M}{ds^2} = P_y \end{aligned} \tag{2.21}$$

And in operational form:

$$\begin{bmatrix} -\frac{d}{ds} & \frac{1}{R}\frac{d}{ds} \\ \frac{1}{R} & \frac{d^2}{ds^2} \end{bmatrix} \begin{Bmatrix} N \\ M \end{Bmatrix} = \begin{Bmatrix} P_x \\ P_y \end{Bmatrix} \tag{2.22}$$

In the constitutive relationship, the equation relating the shear to its deformation is eliminated.

$$\begin{Bmatrix} N \\ M \end{Bmatrix} = \begin{bmatrix} EA & 0 \\ 0 & EI \end{bmatrix} \begin{Bmatrix} \varepsilon \\ \kappa \end{Bmatrix} \tag{2.23}$$

There is another simplification to apply in order to reach an analytical solution and that is consider the axial deformation equal to zero. This is called inextensibility.

Inextensibility can be written, formally, as:

$$\varepsilon = 0$$

It implies

$$\varepsilon = \frac{du}{ds} + \frac{v}{R} = 0 \Rightarrow v = -\frac{du}{ds} \cdot R \tag{2.24}$$

Operating and substituting in the second equation of Eq. (2.17):

$$\kappa = \frac{d\varphi}{ds} = -R \cdot \frac{d^3u}{ds^3} - \frac{1}{R}\frac{du}{ds} \tag{2.25}$$

In the matrix form

$$\{\kappa\} = \begin{bmatrix} -R\dfrac{d^3}{ds^3} & -\dfrac{1}{R}\dfrac{d}{ds} \end{bmatrix} \{u\} \tag{2.26}$$

Regarding the static part, as in the implicit equations of congruence, there are some implications in the undefined equations of equilibrium caused by inextensibility. To get it, it is only necessary to reduce the Eq. (2.21) with respect to *N*:

$$\frac{dN}{ds} = \frac{1}{R}\frac{dM}{ds} - P_x \tag{2.27}$$

By deriving the second equation of Eq. (2.21) and substituting it into Eq. (2.27):

$$R\frac{dM^3}{ds^3} + \frac{1}{R}\frac{dM}{ds} - P_x - R\frac{P_y}{ds} = 0 \tag{2.28}$$

In matrixial form

$$\begin{bmatrix} R\dfrac{d^3}{ds^3} & \dfrac{1}{R}\dfrac{d}{ds} \end{bmatrix} \{M\} = \left\{ P_x + R\frac{dP_y}{ds} \right\} \tag{2.29}$$

Finally, the constitutive relationship will be

$$\{M\} = [EI]\{\kappa\} \tag{2.30}$$

The problem is solved through the displacement method. Substituting into Eq. (2.30), the Eq. (2.26):

$$M = -EI\left(R\frac{d^3u}{ds^3} + \frac{1}{R}\frac{du}{ds} \right) \tag{2.31}$$

Then, substituting the latter equation into Eq. (2.28) and operating some mathematical simplifications:

$$-EI\left(R^2\frac{d^6u}{ds^6} + 2\frac{d^4u}{ds^4} + \frac{1}{R^2}\frac{d^2u}{ds^2} \right) = P_x + R\frac{dP_y}{ds} \tag{2.32}$$

Six boundary conditions are necessary to solve the problem since the differential equation has six orders of derivation. Taking the boundary conditions previously defined and replacing the relations resulting by the imposition of the simplifications, the conditions on the constrained boundary are obtained:

$$
\begin{aligned}
&u=\overline{u}\\
&v=\overline{v}=-R\frac{du}{ds}\\
&\varphi=\overline{\varphi}=-R\cdot\frac{d^2u}{ds^2}-\frac{u}{R}
\end{aligned}
\qquad (2.33)
$$

While regarding the free boundary:

$$
\begin{aligned}
&N=EI\left(R^2\frac{d^5u}{ds^5}+\frac{d^3u}{ds^3}\right)+R\cdot P_y=\pm f_x\\
&T=EI\left(R\frac{d^4u}{ds^4}+\frac{1}{R}\frac{d^2u}{ds^2}\right)=\pm f_y\\
&M=-EI\left(R\frac{d^3u}{ds^3}+\frac{1}{R}\frac{du}{ds}\right)=\pm\mu
\end{aligned}
\qquad (2.34)
$$

Chapter 3

3 Boundary-value problem: Theory and the proposed algorithm

This chapter will introduce some theorical aspects regarding to the Boundary Value problem. Firstly, some mathematical theorems are presented in order to introduce the issue. Afterwards, the algorithm utilized by MATLAB is described and explained and the syntax used in the code for solving our problem is shown.

3.1 Introduction

According to Shampine [3], ordinary differential equations (ODEs) appeared in many applications of different engineering fields. They are so useful to explain and reproduce a mathematical description of complex phenomena occurring in real life. A system of ODEs admits many solutions; hence it is necessary to specify the values of all its components at a single point x=a in order to get an expected result. This is the so-called Initial Value Problem (IVP). Nevertheless, a solution is determined at more than one single point. A Boundary-value problem (BVP) is a system of ordinary differential equations subjected to specified boundary condition at more than one point. Therefore, the main goal is to find a solution to the ODEs which also satisfies the boundary conditions. This kind of problem may have no solution, a finite number or infinite solutions. There are programs for solving BVPs, and due to the last fact, an initial guess must be provided by the user to get a solution and the solver performance will depend on the quality of the guess.

In this research paper, solutions to the 6th order differential equation, above developed, are exposed by using *bvp4c Method* (MATLAB). An explanation about BVP mathematical theory and the solver's algorithm will be given in this section.

3.2 Boundary-value problem

Before talking about the algorithm used by MATLAB program, some theorems will be introduced regarding the two-point boundary-value problem, such as the existence and uniqueness of solution, and after that, also a numerical method will be briefly introduced.

3.2.1 Mathematical theories

Consider, for instance, the following 2nd order differential equation with boundary conditions in the form:

$$\begin{cases} y'' = f(x,y,y') \\ y(a)=\alpha, \quad y(b)=\beta \end{cases} \tag{3.1}$$

- Theorem 1

Suppose that f in (3.1) is continuous on the set

$$D = \{(x,y,y') \mid a \le x \le b,\ -\infty < \mathrm{y} < \infty,\ -\infty < \mathrm{y}' < \infty\},$$

and that $\frac{\partial f}{\partial y}$ *and* $\frac{\partial f}{\partial y'}$ *are also continuous on D. If*

1. $\frac{\partial f}{\partial y}(x,y,y') > 0$ *for all* $(x,y,y') \in D$, *and*
2. *A constant* M *exists, with* $\left|\frac{\partial f}{\partial y'}(x,y,y')\right| \le M, \forall (x,y,y') \in \mathrm{D}$,

then (3.1) has a unique solution.

When the function $f(x,y,y')$ has the special form

$$f(x,y,y') = p(x)y' + q(x)y + r(x),$$

the differential equation become a so-called linear problem. The previous theorem can be simplified for this case.

- *Corollary 1*

If the linear two-point boundary-value problem

$$\begin{cases} y'' = p(x)y' + q(x)y + r(x) \\ y(a) = \alpha, \qquad y(b) = \beta \end{cases}$$

satisfies

$p(x), q(x),$ *and* $r(x)$ *are continuous on* $[a,b]$, *and*

$q(x) > 0$ *on* $[a,b]$,

then, the problem has a unique solution.

- Theorem 2

The boundary-value problem

$$\begin{cases} y'' = f(x,y) \\ y(0) = 0, \qquad y(1) = 0 \end{cases}$$

has a unique solution if $\frac{\partial f}{\partial y}$ is continuous, non-negative and bounded in the strip $0 \le x \le 1$ and $-\infty < \mathrm{y} < \infty$.

3.2.2 Numerical methods: Shooting method

The solutions of BVPs are not easy to obtain analytically. This implies the use of numerical methods to give numerical estimations to the solutions. There are many methods to approximate the solution. In the present thesis the work is focused on the Shooting method.

Consider the Eq. (3.1). The idea of shooting method is to solve a related initial-value problem with a guess for $y'(a)$, say z. The corresponding IVP

$$\begin{cases} y'' = f(x,y,y') \\ y(a) = \alpha, \qquad y'(a) = z \end{cases} \tag{3.2}$$

can then be solved by, for instance, Runge-Kutta method. This approximate solution can be denoted as y_z and hope $y_z(b) = \beta$. If not, another guess must be used for $y'(a)$, and try to solve an altered IVP (3.2) again. This process is repeated and can be done systematically.

3.3 The proposed algorithm (MATLAB's code)

MATLAB possesses many differential equation solvers that cover a wide range of application in science and engineering fields. There are solvers for EDOs posed as initial-value problem as boundary-value problem.

For BVPs, there are two solvers *bvp4c and bvp5c* working on problems that have two-boundary conditions, multipoint conditions, singularities in the conditions and unknown parameters.

After the explanation of the method, a syntax utilized in the code is provided.

3.3.1 *Bvp4c* method

The following explanation of *bvp4c* methodology is based on the development of Shampine [3].

Consider, again, the 2nd order differential equation (3.1). In order to run the solver, it is necessary to reduce the equation to a system of first order equations, since the solvers implement a collocation method to approximate the solution of BVPs subjected to general non-linear, two-point boundary conditions of the form:

$$\begin{aligned} &y' = f(x,y), \qquad a \le x \le b, \\ &g(y(a), y(b)) = 0 \end{aligned} \tag{3.3}$$

The approximate solution **S**(x) is a continuous function that is a cubic polynomial on each subinterval $I_n = [x_n, x_{n+1}]$ of a mesh $a = x_0 < x_1 < ... < x_N = b$. It satisfies the boundary conditions applied to **y**:

$$g(\mathrm{S}(a), \mathrm{S}(b)) = 0$$

and, likewise, the solution **S**(x) satisfies the differential equations (collocates) at both ends and the midpoint of each subinterval I_n

$$\begin{aligned} S'(x_n) &= f(x_n, S(x_n)), \\ S'\left(\frac{x_n + x_{n+1}}{2}\right) &= f\left(\frac{x_n + x_{n+1}}{2}, S\left(\frac{x_n + x_{n+1}}{2}\right)\right), \\ S'(x_{n+1}) &= f(x_{n+1}, S(x_{n+1})) \end{aligned}$$

These conditions imply a system of non-linear algebraic equations for the coefficients defining **S**(x). Conversely to shooting method, the solution **y**(x) is approximated over the whole interval $[a,b]$ and the boundary conditions are always considered. The non-linear algebraic equations are solved iteratively by linearization. The approximated solution **S**(x) will be a fourth order approximation or fifth order one (this will depend on the choice between *bvp4c* or *bvp5c*) to an isolated solution **y**(x), i.e., $\| y(x)-S(x)\| \leq Ch^4$. h is the maximum of the step sizes $h_n = x_{n+1} - x_n$ and C is a constant.

As previously said, BVPs can have more than one solution. For this reason, in order to reach the solution desired is necessary that the user proposes a good guess. The guess considers a deduction for an initial mesh that expose the behavior of the desired solution. This method adapts the mesh to obtain an accurate numerical solution with a modest number of mesh points. The quality of the solution will depend on the accuracy of the supplied guess, and this is often the hardest part of solving a BVP. The continuity of **S**(x) on [a,b] and collocation at the ends of each subinterval imply that **S**(x) also has a continuous derivative on [a,b].

Bvp4c method controls the error made when approximating **y**(x) using **S**(x) that helps it deal with poor guesses. For the approximation **S**(x), the residual **r**(x) in the ODEs can be defined as

$$r(x) = S'(x) - f(x, S(x))$$

$$S'(x) = f(x, S(x)) + r(x)$$

The residual in the boundary conditions is $g(\mathrm{S}(a), \mathrm{S}(b))$. *Bvp4c* controls the size of these residuals. If these ones are uniformly small, **S**(x) is a good solution (this means that **S**(x) is close to **y**(x)).

Shooting codes also control the size of these residuals. At each step an IVP code controls the local error, which is equivalent to controlling the size of the residual of an appropriate continuous extension of the formula used, and the nonlinear equation solver is used to find initial values for which the residual in the boundary conditions is small.

There are some benefits for residual controlling. Residuals are well-defined no matter how bad the approximate solution, and residuals can be evaluated anywhere simply by evaluating $f(x, S(x))$ or $g(S(a), S(b))$.

Bvp4c is based on algorithms that are plausible even when the initial mesh is very poor yet furnish the correct results as $h \to 0$.

The MATLAB User Guide provides the following definition for the command: *"bvp4c is a finite difference code that implements the 3-stage Lobatto IIIa formula. This is a collocation formula, and the collocation polynomial provides a C1-continuous solution that is fourth-order* *accurate uniformly in the interval of integration. Mesh selection and error control are based on the residual of the continuous solution.*

The collocation technique uses a mesh of points to divide the interval of integration into subintervals. The solver determines a numerical solution by solving a global system of algebraic equations resulting from the boundary conditions and the collocation conditions imposed on all the subintervals. The solver then estimates the error of the numerical solution on each subinterval. If the solution does not satisfy the tolerance criteria, then the solver adapts the mesh and repeats the process. You must provide the points of the initial mesh, as well as an initial approximation of the solution at the mesh points." [4]

3.3.2 Syntax

As said previously, *bvp4c* solves boundary value problems for ODEs by collocation. The basic syntax used in MATLAB is written as follows

sol = bvp4c (@odefun, @bcfun, solinit, options)

where,

- *@odefun* is a function handle that evaluates the differential equation expressed as a system of first-order equations of the form $y' = f(x, y)$. It generally is written as

  dydx = odefun(x,y,p1,p2…)

  where p1, p2, ... are known parameters. The output is a column vector representing $f(x, y)$.

- *@bcfun* is, also, a function handle that compute the residual in the boundary conditions. It has the following form

  res = bcfun (ya, yb, p1, p2, …)

  The output is a column vector representing $bc(y(a), y(b))$.

- *solinit* forms the initial guess for the solver by using *bvpinit*(x,v) where x is a vector that specifies an initial mesh whereas v is a guess for the solution and can be either a vector or a function. The output is a structure with fields

  x: ordered nodes of the initial mesh with SOLINIT.x(1) = a and SOLINIT.x(end) = b.

  y: initial guess for the solution with SOLINIT.y(:,i) a guess for y(x(i)) at the node SOLINIT.x(i)

- *options* is an optional integration argument. It changes the default integration properties by naming the property and establishing the value. If it is not called in SOL, the solution is computed by the default parameters.

Finally, the *bvp4c* solver integrates a system of first-order ordinary differential equation on the interval$[a, b]$, subject to general two-point boundary conditions. The solution is continuous on $[a, b]$ and has a continuous first derivative there. The output SOL will be a structure with the following fields

- SOL.solver : It will be the chosen solver that can be *bvp4c* or *bvp5c*
- SOL.x : Mesh selected by bvp4c and formed by ***bvpinit.***
- SOL.y : Approximation to y (x) at the mesh points of SOL.x
- SOL.yp : Approximation to y' (x) at the mesh points of SOL.x
- SOL.stats : Computational cost statistics

Chapter 4

4 Arch structural analysis

This chapter will present a structural analysis of an arbitrary arch subjected to its self-weight and to a load condition q_0. The idea is to try the bvp-solver and to analyze how internal actions and stresses change along the arch by varying the initial angle which will be the one to optimize , together to the section radius, in the next chapter to obtain a proper solution for a given fixed parameters like span length, material, among others.

A briefly introduction about how to insert the differential equation system of the arch in the code is given. Then, some charts and results are shown to differentiate between the different cases. Finally, the coupled problem between arch and deck connected by means of hangers is intended to be evaluated, however due to the limitations given by the bvp-solver, this aim was not reached.

4.1 Isolated arch

Firstly, an isolated arch was analyzed by means of a first simple code written in MATLAB which is based on the method previously developed. Some solutions were performed to a fixed span L in order to compare the behavior of the arch by varying the initial angle, and to verify if the solver works fine.

The arch was subjected to its dead load γA and to an arbitrary load condition q_0. The load condition can represent the deck weight and any considered variable load.

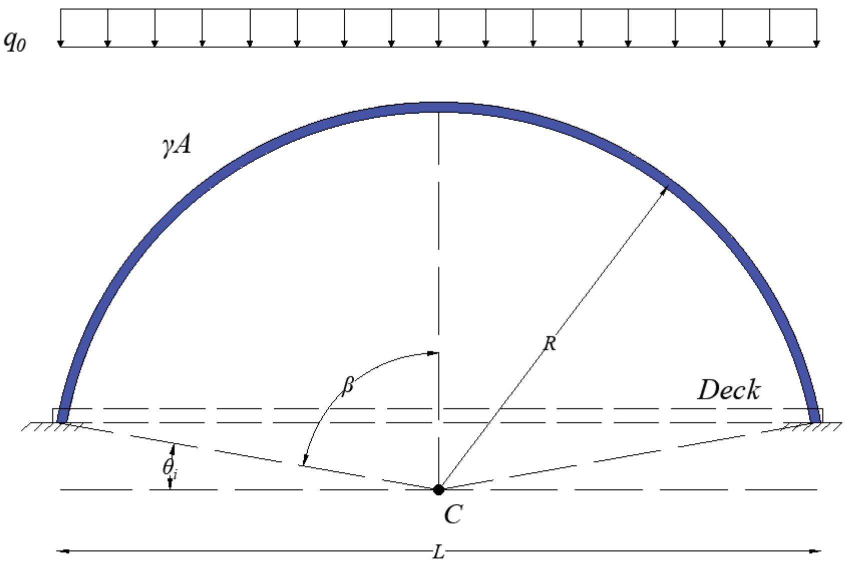

Figure 4-1. Representation of arch.

In Figure 4-1 is shown a representation of the arch whose span length is *L*. The considered configuration will have a certain initial angle θ_i (angle measured from the horizontal axis to the section where the arch starts), and a certain radius R, taking as constant, whose curvature center is *C*. These parameters are related as follows

$$R = \frac{L}{2\sin\beta}$$
$$\beta = \frac{\pi}{2} - \theta_i \qquad (4.1)$$

β is the angle measured from the vertical axis to the section where the arch starts.

These values are considered as *Input data*, but they are not enough, so it is also necessary to insert data about the cross-section like the radius *r* and kind of material, therefore Young Modulus *E*, shear modulus *G* and specific weight γ.

Once defined these parameters, the section properties must be computed. As a full circular section is taken under consideration, the section properties are computed as

$$A = \pi r^2 \qquad (4.2)$$

$$J = \frac{\pi r^4}{4} \qquad (4.3)$$

$$A_s = 0.9A \qquad (4.4)$$

$$W = \frac{\pi r^3}{4} \tag{4.5}$$

Where *A* is the section area, *J* is the inertia, *A_s* is the shear area and *W* is the section modulus.

As previously introduced, this problem is a boundary value problem in which boundary values are imposed on each end of the arch by means of restraints.

The *bvp4c* method solves equations of the form $y' = f(x, y)$. Therefore, the obtained system of differential equations must be reduced to a system of first-order equations.

This transformation is easy to do, and it is based on [5]. Suppose an n-*th* order equation as follows:

$$y^{(n)} = f\left(x, y, y', ..., y^{(n-1)}\right) \tag{4.6}$$

The dependent variables $y_1, y_2, ..., y_n$ can be defined as

$$y_1 = y \text{ , } y_2 = y' \text{ ,..., } y_n = y^{(n-1)} \tag{4.7}$$

Notice that $y_1' = y' = \mathrm{y}_2$, $y_2' = y'' = y_3$, and so on. So, the substitution of Eq. (4.7) into Eq. (4.6) gives the following system of *n* first-order equations:

$$\begin{aligned} y_1' &= \mathrm{y}_2, \\ y_2' &= \mathrm{y}_3, \\ &\vdots \\ y_{n-1}' &= \mathrm{y}_n, \\ y_n' &= f(x, y_1, y_2, ..., y_n) \end{aligned} \tag{4.8}$$

Thus, remembering the complete set of equations

$$\begin{aligned} &-EA\left(\frac{d^2u}{ds^2} + \frac{1}{R}\frac{dv}{ds}\right) - GA_T\left(-\frac{u}{R^2} + \frac{1}{R}\frac{dv}{ds} - \frac{\varphi}{R}\right) = P_t \\ &EA\left(\frac{1}{R}\frac{du}{ds} + \frac{v}{R^2}\right) - GA_T\left(-\frac{1}{R}\frac{du}{ds} + \frac{d^2v}{ds^2} - \frac{d\varphi}{ds}\right) = P_n \\ &-GA_T\left(-\frac{u}{R} + \frac{dv}{ds} - \varphi\right) - EI\frac{d^2\varphi}{ds^2} = m \end{aligned} \tag{4.9}$$

and defining the dependent variables as

$$
\begin{aligned}
&u_1 = u \\
&v_1 = v \\
&\varphi_1 = \varphi \\
&u_2 = u^I = u_1{}^I \\
&v_2 = v^I = v_1{}^I \\
&\varphi_2 = \varphi^I = \varphi_1{}^I
\end{aligned}
\qquad (4.10)
$$

the following system of first-order equations is produced:

$$
\begin{Bmatrix} u_1{}' \\ v_1{}' \\ \varphi_1{}' \\ u_2{}' \\ v_2{}' \\ \varphi_2{}' \end{Bmatrix}
=
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\frac{D}{R^2} & 0 & \frac{D}{R} & 0 & -\frac{D}{R}-\frac{1}{R} & 0 \\
0 & \frac{1}{DR^2} & 0 & \frac{1}{DR}+\frac{1}{R} & 0 & 1 \\
\frac{D'}{R} & 0 & D' & 0 & -D' & 0
\end{bmatrix}
\begin{Bmatrix} u_1 \\ v_1 \\ \varphi_1 \\ u_2 \\ v_2 \\ \varphi_2 \end{Bmatrix}
+
\begin{Bmatrix} 0 \\ 0 \\ 0 \\ -\frac{P_t}{EA} \\ -\frac{P_n}{EI} \\ 0 \end{Bmatrix}
\qquad (4.11)
$$

where $D = \frac{GA_s}{EA}$, $D' = \frac{GA_s}{EI}$ (only to simplify the computation).

The forces are evaluated in the local system, so the self-weight and the considered load will be projected to the tangential and normal direction of the section under consideration.

$$
P_n = (\gamma A + q_0)\sin\left(\vartheta_i + \frac{s}{R}\right) \qquad (4.12)
$$

$$
P_t = (\gamma A + q_0)\cos\left(\vartheta_i + \frac{s}{R}\right) \qquad (4.13)
$$

A more compact form can be written for the above system as:

$$
\{Z'\} = [A]\{Z\} + \{f\} \qquad (4.14)
$$

where $[A]$ is a matrix that taking into account the mechanic properties of the arch together with the geometrical ones, $\{Z\}$ is a vector that contains the dependent variables previously defined, $\{Z'\}$ is a first-order vector of the dependent variables and $\{f\}$ is an independent vector that contains the forces.

Moreover, the boundary conditions must be given. For all the cases, the arch was considered fully fixed at each end, so the boundary conditions are:

$$u = 0$$
$$v = 0 \qquad \textit{for } s=0 \textit{ and } s=Smax.$$
$$\varphi = 0$$

where *Smax* is the final coordinate of the arch.

Finally, the number of nodes of the initial mesh must be defined. For all the cases, the number was *200*. Additionally, the relative tolerance for the residual *RelTol* was set by default and it applies to all components of the residual vector with 0.1% accuracy.

Once the code has run, the output of *bvp4c* is a structure element (MATLAB's notation) in which the three first rows of the solution are the displacements u, v, φ and the following three are their first derivatives for each point of the mesh. With them, it is possible to compute the internal forces acting at each section of the arch as previously said:

$$\begin{aligned} N &= EA\left(\frac{du}{ds}+\frac{v}{R}\right) \\ T &= GA_T\left(\frac{-u}{R}+\frac{1}{R}\frac{dv}{ds}-\varphi\right) \\ M &= EI\frac{d\varphi}{ds} \end{aligned} \tag{4.15}$$

In this way, the maximum stress at each section can be evaluated according to Navier.

$$\sigma_{\max} = \frac{N}{A}+\frac{M}{W} \tag{4.16}$$

4.1.1 Case study

Considering the following values for each parameter, the following results were obtained. The results have been compared to the ones obtained by FEM in *SAP2000*, and they were similar with a minimum difference.

For this case, a hollow section made of steel was taken under consideration. In the following, the fixed parameters are presented:

Cross-section: Full circular section

- r = 0.5 m

Material: Steel

- E = 200000 MPa
- γ = 78.5 kN/m^3

- G = 76903069 MPa

Geometrical constraint

- Span length = 80 m

The last parameter to be defined is the initial angle θ_i which different values were considered. Consequently, the curvature radius of the arch also changes.

- $\theta_i = 0°$, R = 40 m. (a)
- $\theta_i = 30°$, R = 46.19 m. (b)
- $\theta_i = 60°$, R = 80 m. (c)
- $\theta_i = 90°$, R = ∞ (d)

Moreover, a design load equal to 150 kN/m was considered.

Once defined the values, the following results were obtained by running the code.

For the case a):

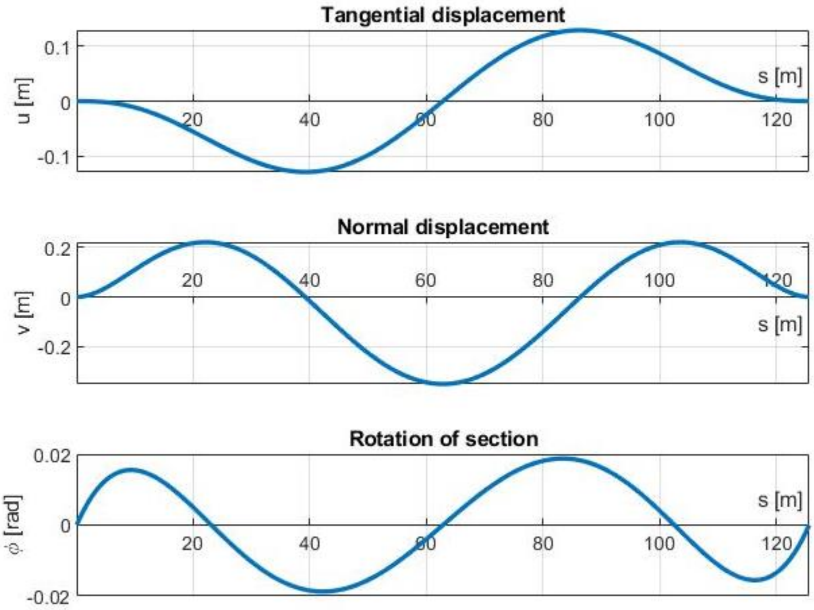

Figure 4-2. Displacement for case a)

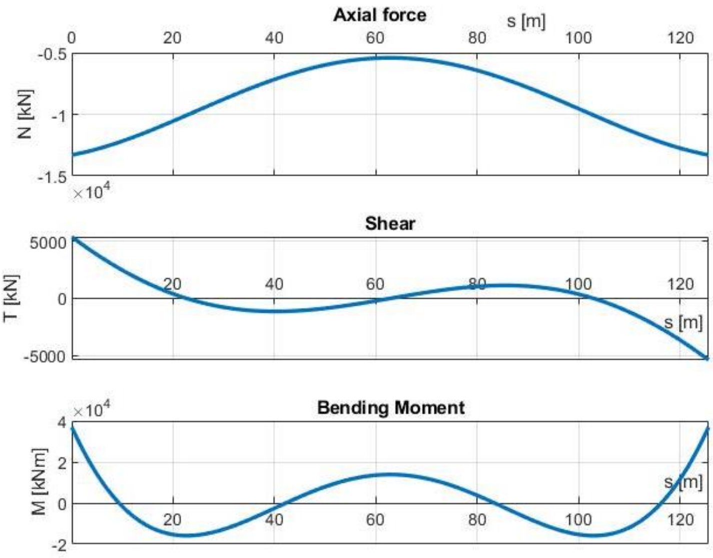

Figure 4-3. Internal actions for case a)

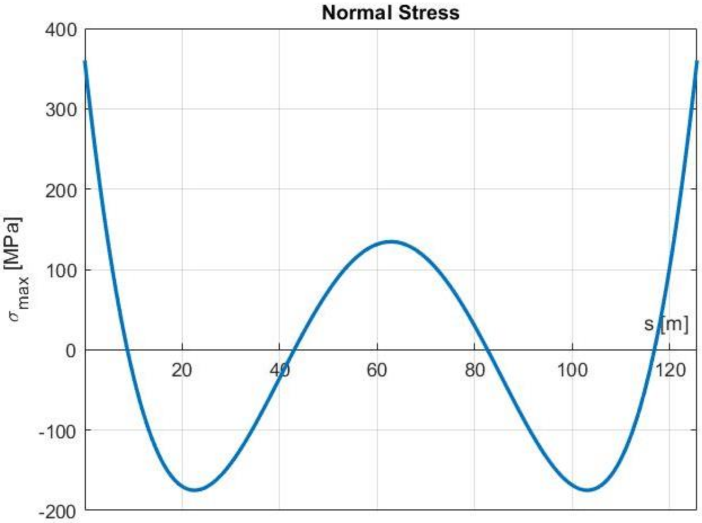

Figure 4-4. Normal stresses for case a)

For case b)

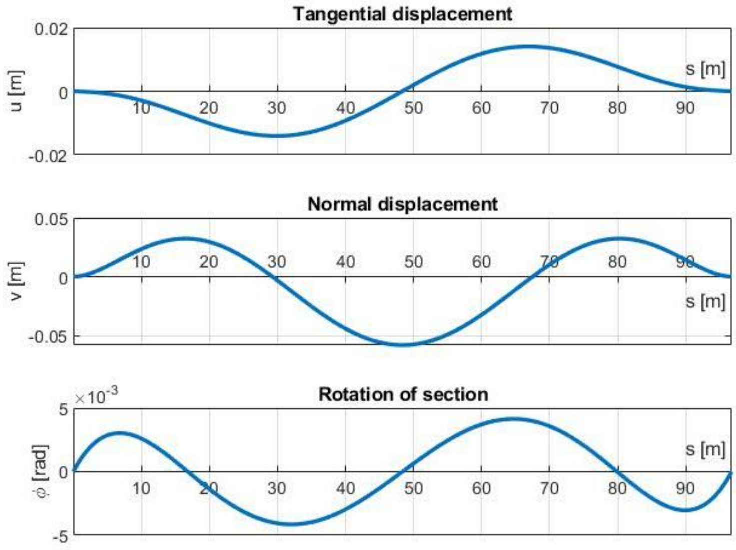

Figure 4-5. Displacement for case b)

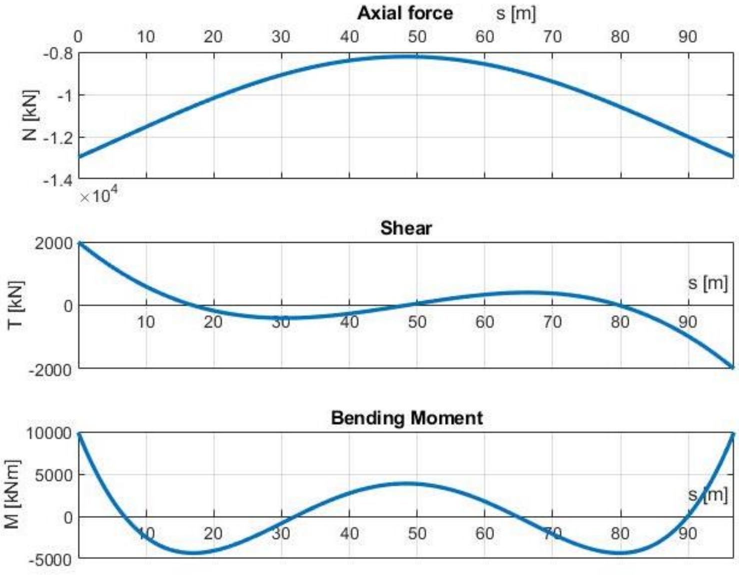

Figure 4-6. Internal actions for case b)

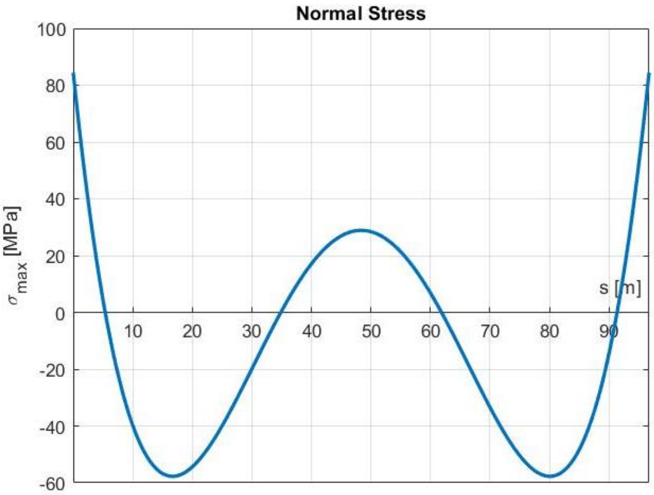

Figure 4-7. Normal stresses for case b)

For case c)

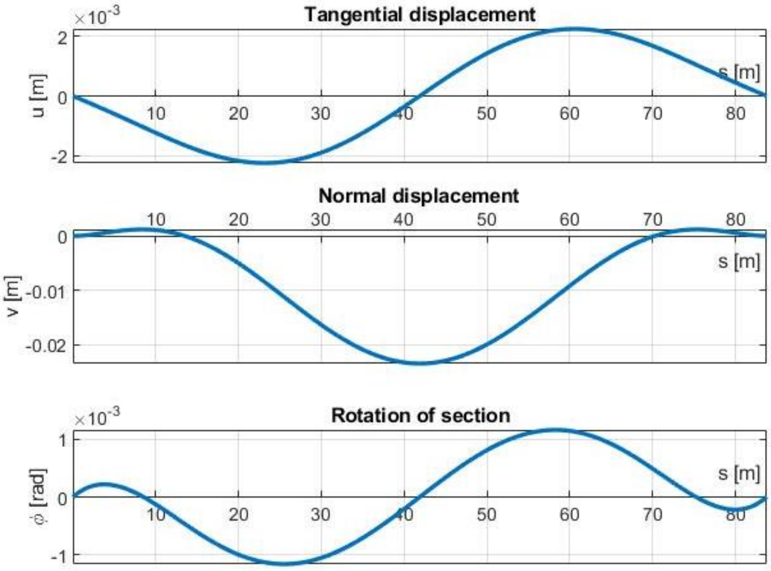

Figure 4-8. Displacements for case c)

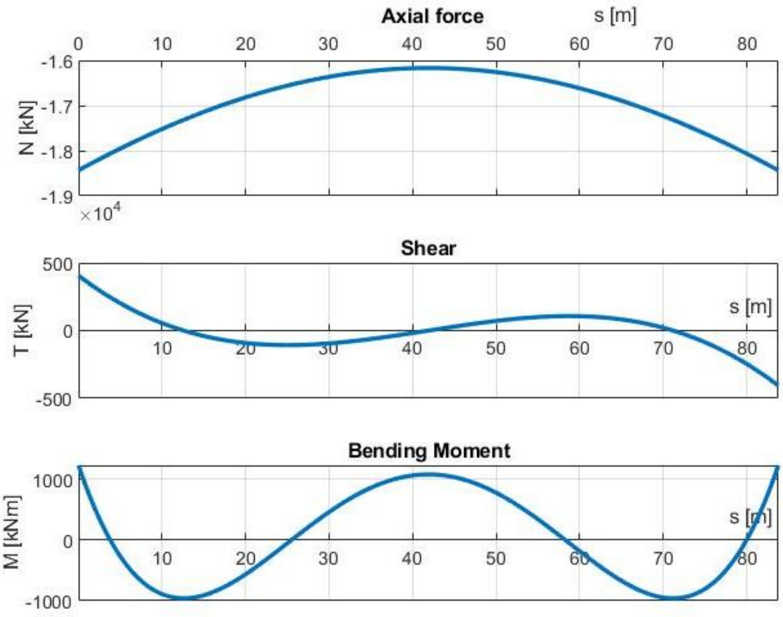

Figure 4-9. Internal actions for case c)

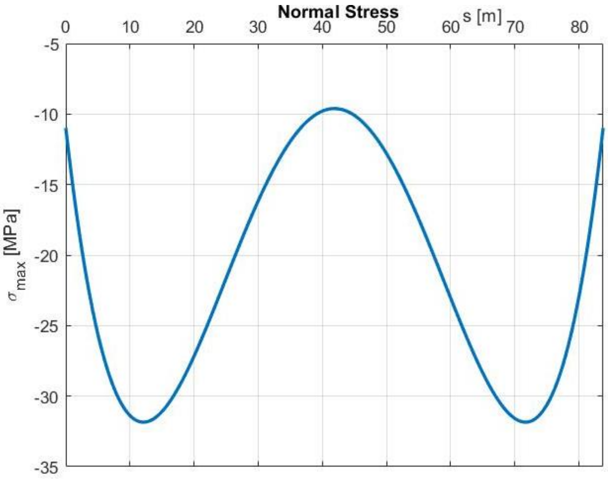

Figure 4-10. Normal stresses for case c)

For case d)

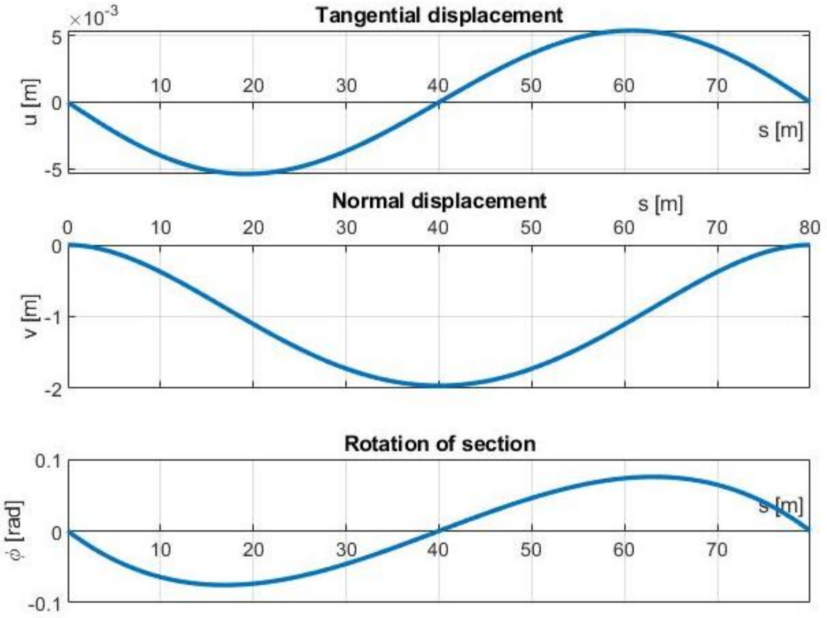

Figure 4-11. Displacements for case d)

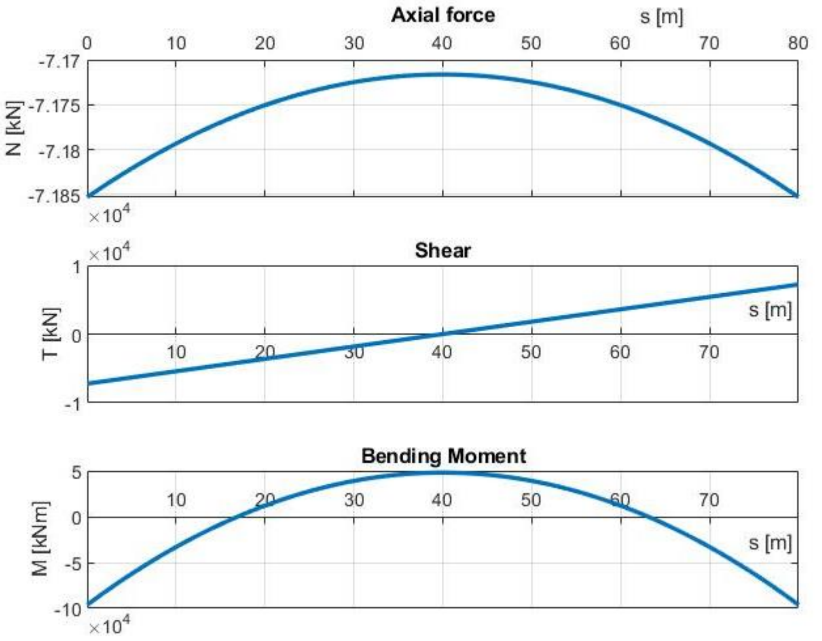

Figure 4-12. Internal actions for case d)

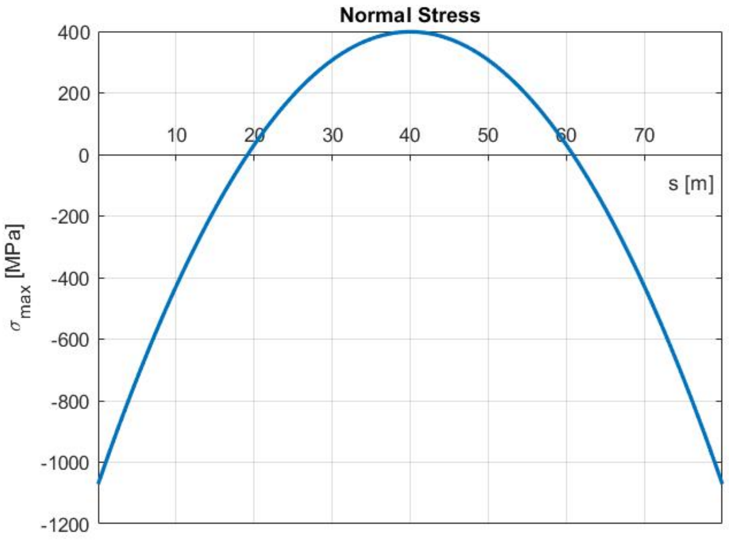

Figure 4-13. Normal stresses for case d)

It can be seen that by lowering the arch, i.e. increasing the initial angle or diminishing the rise *f*, the normal stresses start to decrease. They pass from tensile stresses to compressive ones. However, in the limit, case d), the negative values are also so high overpassing the design requirements and increasing the possibility of instability problem. As θ_i increases, the axial force increases too (in negative values) and the bending moment decreases. Consequently, the idea is to obtain a momentless configuration in order to have a purely compressed arch.

In this way, the aim of the next chapter will be to find the optimal value of the initial angle in such a way the arch volume is minimized and the design requirements are fulfilled, but, besides the initial angle, also will be modified the radius of the cross-section.

It can be noted also that in the limit, where the curvature radius is infinite, the results are the same as a fully fixed straight beam.

A variation of the stresses respect to the initial angle θ_i is displayed in the following figure. It can be observed that for an initial angle lesser than 45°, tensile stresses will govern the behavior. Whereas for initial angles greater than 45°, the arch will be governed by compressive stress.

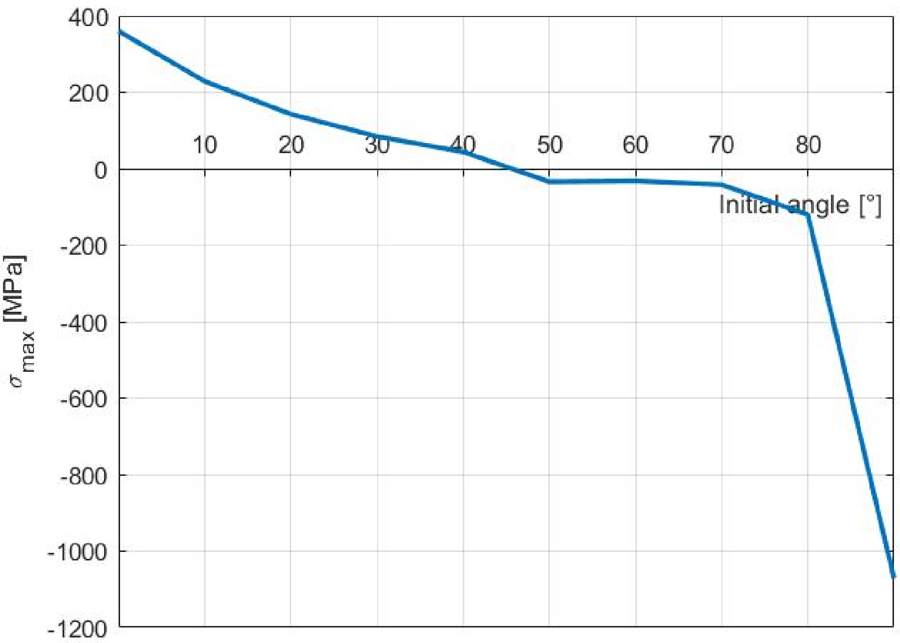

Figure 4-14. Variation of stresses respect to the initial angle.

4.2 Coupled problem between arch and deck

As previously said, in the case of tied-arch bridges, the arch can be connected to the deck by means of steel elements, known as hangers, working completely in tension. So, at this point, it is our interesting to analyze how the arch behaves when it is connected to the deck through the hangers. This carries us to a coupled problem between both elements.

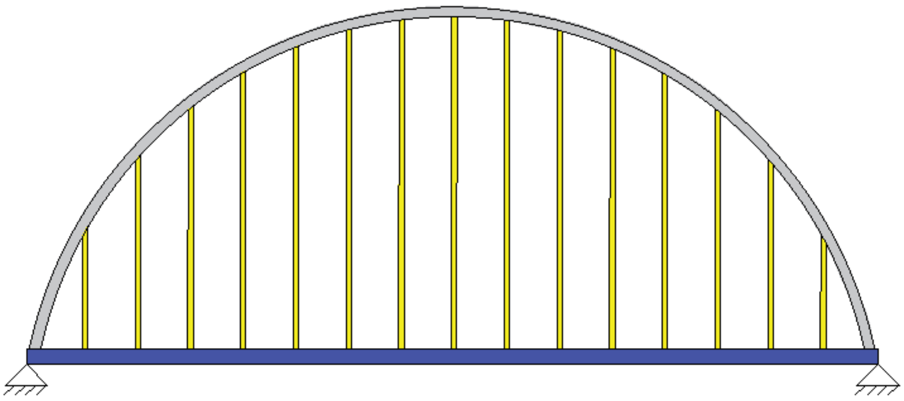

Figure 4-15. Arch and deck connected by hangers.

The main idea is to consider the hangers as an equivalent spread membrane, with unitary depth, acting between the arch and the deck. Hence, by meshing the structure, a strip will be obtained containing an arch and beam differential linked by a portion of the membrane.

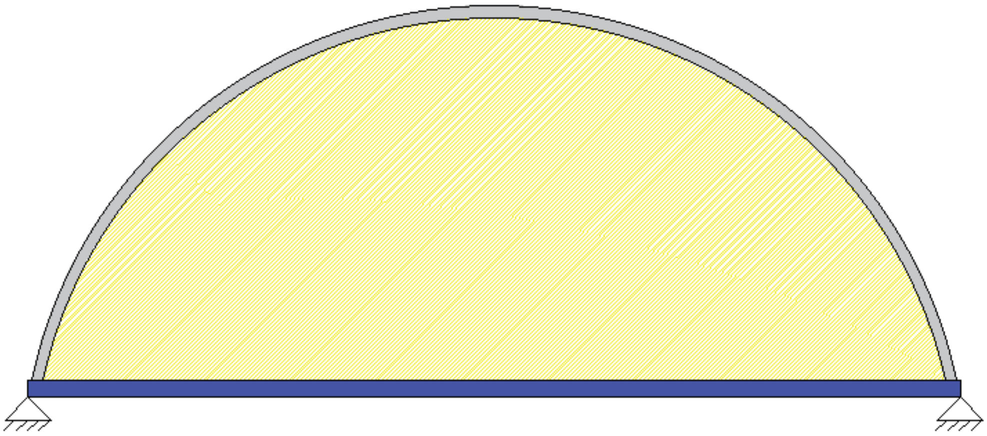

Figure 4-16. Arch and deck connected by an equivalent spread membrane.

Therefore, a new system of equations is generated by adding the, well-known, beam equation of 4th order into the latter system Eq. (4.11). The equation to be added is the following one:

$$(EI)_D \eta^{IV} = q_D + f_H \tag{4.17}$$

$$f_H = (y_A - \eta) K_H \tag{4.18}$$

where η is the beam deflection, *EI* its flexural stiffness and *q* its self-weight. The subscript *D* refers to the deck to not confuse parameters.

Besides the general load *q,* a new term is introduced. f_H is a force taking into account the coupling condition of the problem. So, the coupling can be performed considering an extra force acting on the beam equal to the difference between arch and beam displacements, in the global reference system, multiplied by the stiffness of the hanger K_H. y_A is the arch vertical displacement obtained by projecting *u* and *v* into the global reference system.

As reported before, it is necessary to convert the 4th order equation into a first-order system of equations. This is

$$\begin{aligned} \eta_1 &= \eta \\ \eta_2 &= \eta^I = \eta_1{}^I \\ \eta_3 &= \eta^{II} = \eta_2{}^I \\ \eta_4 &= \eta^{III} = \eta_3{}^I \\ \eta_5 &= \eta^{IV} = \eta_4{}^I = \frac{q_D + f_H}{(EI)_D} \end{aligned} \tag{4.19}$$

Once obtained the beam system Eq. (4.19), now it is possible to add it into the Eq. (4.11) and thus obtain the complete coupled system of 1st order equation between both problems.

$$\begin{Bmatrix} u_1' \\ v_1' \\ \varphi_1' \\ u_2' \\ v_2' \\ \varphi_2' \\ \eta_1' \\ \eta_2' \\ \eta_3' \\ \eta_4' \end{Bmatrix} = \left[\begin{array}{cccccc|cccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{D}{R^2} & 0 & \frac{D}{R} & 0 & \frac{-D}{R}-\frac{1}{R} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{DR^2} & 0 & \frac{1}{DR}+\frac{1}{R} & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{D'}{R} & 0 & D' & 0 & -D' & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{K_h \cdot \cos(\vartheta)}{(EI)_D} & \frac{K_h \cdot \sin(\vartheta)}{(EI)_D} & 0 & 0 & 0 & 0 & \frac{-K_h}{(EI)_D} & 0 & 0 & 0 \end{array}\right] \begin{Bmatrix} u_1 \\ v_1 \\ \varphi_1 \\ u_2 \\ v_2 \\ \varphi_2 \\ \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{Bmatrix} + \left\{\begin{array}{c} 0 \\ 0 \\ 0 \\ \frac{-P_t}{EA} \\ \frac{-P_n}{GA_s} \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ \frac{-q}{(EI)_D} \end{array}\right\} \tag{4.20}$$

Where $D = \dfrac{GA_s}{EA}$ and $D' = \dfrac{GA_s}{EI}$.

As before, it can be written in a more compact form:

$$\{\overline{Z}'\} = [A]\{\overline{Z}\} + \{\overline{f}\} \tag{4.21}$$

4.2.1 Limitations

The system of equation Eq. (4.20) is ready to be inserted in the code, however, the function *bvp4c* only admits the definition of mesh in 1D. It means that, in this case, the user can only define the mesh subdivision either only to the arch or only to the beam one as both problems are deducted in different coordinates reference system. Because of this, it is necessary to deduct the differential equations in the same reference system to be able to run the program with only one variable. In this thesis we have chosen to focus on the problem of the isolated arch considering the solution for the coupled solution subject to future developments.

Chapter 5

5 Shape and arch cross-section optimization

This is the main chapter of the work as the optimization aspects are introduced and an application of an algorithm in order to get the best solution for our problem is performed. Consequently, some concepts regarding the structural optimization will be given and, moreover, a briefly explanation of the utilized algorithm. Afterwards, the algorithm is applied to our case and the results will be shown. Finally, the problem will be generalized introducing some dimensionless parameters which will be plotted in such a way the problem will be independent from certain parameters like span length, material, among others.

5.1 Introduction

In the previous chapter, a structural analysis of an arch with an arbitrary configuration was performed in which its behavior under determined load was obtained. However, it is possible to obtain an improve of the response by taking the correct shape and dimensions of the structure. Here, an optimization process will be exposed. This topic has emerged over the past years, and, perhaps, it is one of the most important due to it tries to reduce costs and to exploit the material properties, two fundamental pillars of engineering. The subject can be described as:

"Structural optimization may be defined as the rational establishment of a structural design that is the best of all possible designs within a prescribed objective and a given set of geometrical and/or behavioral limitations". [8]

In structural optimization, the set of parameters is split into *preassigned parameters* and *design variables*. So, the problem consists in determining the optimal values of the design variables such that they maximize or minimize a specific function called *objective function* while satisfying a set of geometrical and/or behavioral requirements called *constraints,* specified prior to design [7].

The design variables can be classified, according to [8], in:

a) Configuration of a structure (Constructive layout and topology)
b) Material properties
c) Geometry – shape
d) Supports – loading
e) Cross-section

The *i-th* design variable is denoted by x_i and all *n* design variables are composed in a vector $\boldsymbol{x}$ called *design vector,* which lies in the design space.

$$\mathbf{x}^T = [x_1, x_2, ..., x_n] \tag{5.1}$$

Any set of design variables represents a point in the design space which is an *n-dimensional* Euclidean space.

Regarding the preassigned parameters, they can differentiate between *independent* and *dependent* variables. It means that, for instance, by defining certain cross-section parameters like radius (independent), the cross-section properties, like area (dependent), can be evaluated with it. Generally, these values are obtained in structural design.

There may be infinite possible solutions, but they are also subjected to certain design requirements by the normative. Therefore, many of them do not verify conditions. In order to omit such design, the design requirements are expressed as constraints prior to optimization. There are two kind of constraints:

a) Geometrical constraints
b) Behavioral constraints

The constraints are formulated in the form of equality and/or inequality constraints:

$$h_i(x) = 0 \qquad (i = 1, ..., q), \tag{5.2}$$

$$g_j(x) \leq 0 \qquad (i = 1, ..., p). \tag{5.3}$$

Each inequality constraint is represented by a surface in the design space which encompass all points ***x*** for which the condition is verified as an equality constraint $g_i\,(x)=0$. It differentiates from admissible, or feasible, to inadmissible, or infeasible, designs. In the end, a subdomain of the design space will be created limited by the constraint surface.

Finally, the objective function is expressed in terms of the design variables in such a way that its value can be evaluated for any point in the design space. This function is our interest, and it will be minimized or maximized by the optimal set of design variables withing the feasible design space. Generally, it is a scalar function defined as:

$$f := f(\mathbf{x}) \tag{5.4}$$

Eventually, the problem formulation consists in determining the values of the design variables x_i such that the objective function f attains an extreme value while, simultaneously, all constraints are verified.

$$\min\left\{ f(\mathbf{x}) \mid \mathbf{h}(\mathbf{x}) = 0\ ,\ g(\mathbf{x}) \le 0 \right\} \qquad \forall \quad \mathbf{x} \in \mathbb{R}^n \tag{5.5}$$

Where $\mathbb{R}^n$ is the n-dimensional set of real numbers, ***x*** is the design vector of n variables, *f(x)* is the objective function, ***g(x)*** is the vector of the p inequality constraints and ***h(x)*** is the vector of the q equality constraints.

All above described is related to the mathematical formulation, however an extra tool is required and that is the computational algorithm. In this way, the optimization problem can be divided in two phases

1) Mathematical formulation where design vector, objective function and constraints are defined
2) Application of an optimization algorithm

There are many kinds of optimization algorithm like classical (derivative-based), genetic, among others. In particular, the genetic algorithm can be more suitable for a variety of problems, regardless if the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear. In this way, for the second phase, the genetic algorithm was selected.

5.2 Genetic Algorithm

According to *MATLAB User Guide* [4] and *Goldberg* [9], the Genetic Algorithm (GA) is a method for solving both constrained and unconstrained optimization problems. It is a search algorithm based on the mechanics of natural selection and natural genetics, process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution.

There are some differences between a standard optimization algorithm and the genetic one. Genetic algorithms differ in two main ways:

1. The GA generates a population of points at each iteration in which the best point in the population approaches an optimal solution. Whereas the standard one generates a single point at each iteration and the sequence of points approaches an optimal solution.
2. The GA selects the next population by computation which uses random number generators (probabilistic transition rules). While the standard one selects the next point in the sequence by a deterministic computation.

There are other differences like GA uses the objective function information, not derivatives, and, finally, the GA works with a coding of the parameter set, not the parameters themselves.

The GA uses three main types of rules at each iteration to create the next generation from the current population:

1. Reproduction: This rule selects the individuals, called parents, that contribute to the population at the next generation.
2. Crossover: This rule combines two parents to form children for the next generation.
3. Mutation: This rule applies random changes to individual parents to form children.

Following these rules, the new population is ready to be tested, i.e. the objective function is computed.

Basically, the following outline [4] shows how the genetic algorithm works:

1. The algorithm begins by creating a random initial population.

2. The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following steps:

   a. Scores each member of the current population by computing its objective value. These values are called the raw fitness scores.

b. Scales the raw fitness scores to convert them into a more usable range of values. These scaled values are called expectation values.

c. Selects members, called parents, based on their expectation.

d. Some of the individuals in the current population that have lower fitness are chosen as *elite*. These elite individuals are passed to the next population.

e. Produces children from the parents. Children are produced either by making random changes to a single parent—*mutation*—or by combining the vector entries of a pair of parents—*crossover*.

f. Replaces the current population with the children to form the next generation.

3. The algorithm stops when one of the stopping criteria is met.

In the following, the optimization process is applied to the arch.

5.3 Arch structural optimization

After the introduction about structural optimization, for our case, finding the optimal shape and cross-section that minimizes the arch volume is the objective.

Therefore, the three components, previously mentioned, are:

- Design variables: *Initial angle* θ_i which measure how lowered, $f(\theta_i)$, is the arch (Figure 5-1), and *section radius r* for a full circular section (Figure 5-2).
- Objective function: The *volume of arch* which must be minimized
- Constraint: The normal stress $\sigma_{\max}$ at each section must be lesser than the design strength of the material.

Where *f* is the rise of the arch.

According to the used initial angle which goes [0; π/2], a certain radius of curvature will be obtained. Thus, as the initial angle increases, a more lowered arch is obtained (the rise *f* decreases), and consequently, a greater outward force. The extreme case will be an arch of infinite radius, that is to say a straight beam.

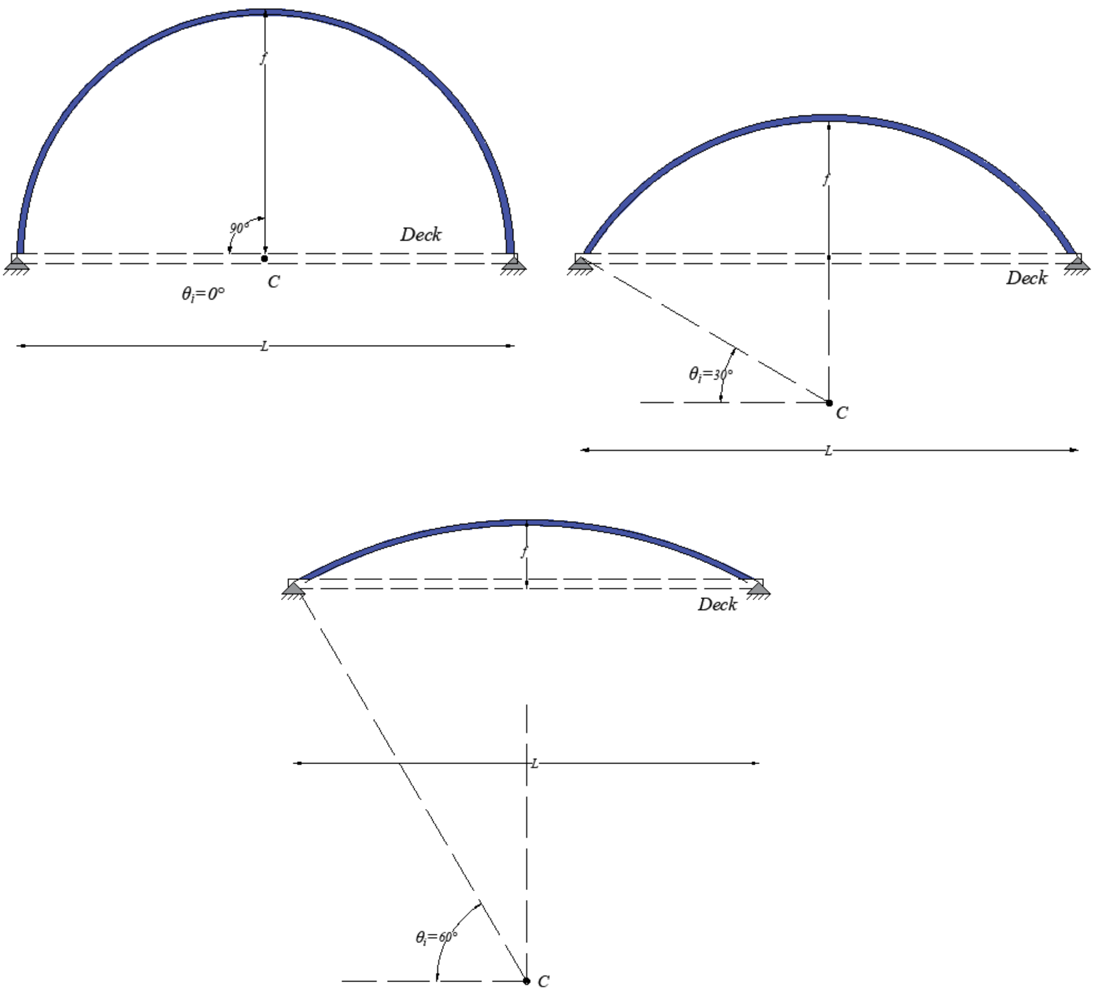

Figure 5-1. Possible initial angles for a fixed span showing how the rise f is lowered

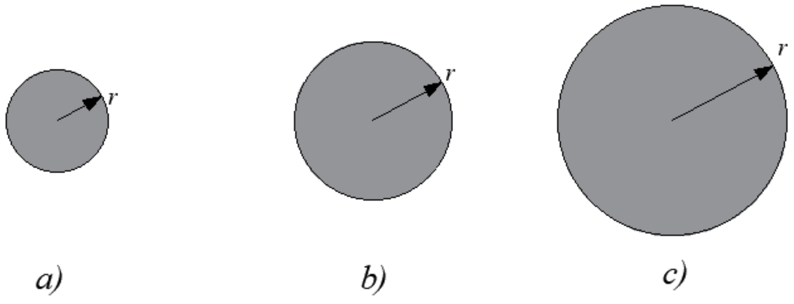

Figure 5-2. Radius variation. a) Small radius, b) Medium radius, c) Large radius

So, the problem formulation is the following:

$\min\left(Vol(\theta_i,t)=S_{\max}(\theta)\cdot A(r)\right)$ according to $\sigma_{\max}\leq f_{yd}$

Where *Vol* is the arch volume computed as the total length of the arch multiplied by the section area, f_{yd} is the design strength of the material calculated as

$$f_{yd} = \frac{f_{yk}}{\gamma_s}$$

Whereas $\sigma_{\max}$ is the maximum normal stress along the arch and it is computed according to Navier's equation as seen before

$$\sigma_{\max} = \frac{N}{A} + \frac{M}{W}$$

Where *N* and *M* are the axial force and bending moment, respectively. And, *A* and *W* are the section area and the section modulus, respectively.

A generalized code is provided in which defining the input data, and by applying the genetic algorithm, returns a series of dimensionless plots considering the span length, rise, and load conditions.

5.3.1 Code

The code is composed by four files:

- Arch_optimization.m: general file.
- Arch.m: function file.
- Arch_analysis.m: function file.
- OF.m: function file.

In the general file *Arch_optimization.m*, the constant input data is inserted which are the span length *L*, the material properties (*E, G, γ and* f_{yd}), the number of nodes *n* and the design load q_0. All these data will be saved in a structure called *Data.arch.*

The initial angle θ_i and the section radius *r* will vary for each evaluation, so the number of design variables *nvar* is equal to 2. They will compose the design vector called *x* in the code. According to the syntax used for the genetic algorithm in MATLAB, the lower and upper bounds will be defined and they will go from LB = [0, 0.2] to UB = [π/2, 1.5] where the first column corresponds to the initial angle whereas the second one to the radius.

After that, the constraint function must be called, and it corresponds to the second file *Arch.m*. It will be defined as a handle function of the design vector *x*. Moreover, the constant data must be evaluated there, so it will also be dependent of it. Then, the objective function is also called, and it corresponds to the fourth file *OF.m*. It is called in the same form as the constraint function, i.e. depending on the design vector and on the constant data. Therefore, in the code, they can be seen as

```
% Constraint function
Const_fun = @(x)Arch (x, Data);
%Objective function
OF_fun    = @(x)OF   (x, Data);
```

Finally, the function *ga* which contains the genetic algorithm is called. It will return the optimal values of the design vector as *x_opt* and *OF_opt*. The syntax is

```
% Optimization with GA
 [x_opt,OF_opt]=ga(OF_fun, nvars,[],[],[], [], LB, UB, Const_fun);
```

The second file calls the input data and receive certain values for the design vector according to the genetic algorithm. With them, the section properties are evaluated and utilized for the solver function which is the third file *Arch_analysis.m*. It returns the internal actions and displacement and with them, the normal stresses are calculated and the maximum one is taken away in order to perform the strength verification. As steel has similar behavior either compressive strength and tensile one, the absolute value of the maximum stresses is compared to the design yielding strength. The constraint is evaluated as

```
constraint      = sigma_max/fyd - 1;
```

As previously said, the third file is the solver function which is based on the algorithm explained in Chapter 3. So, it contains the *bvp4c* solver, the first-order differential equation system and the boundary conditions and will return an output of internal actions and displacements.

Finally, the last file is the objective function *OF.m.* in which the volume of the arch is evaluated, and it is tried to be minimized. It is computed as

```
vol = s(end)*A*gamma;
```

This is done for a determined load condition and span length. Nevertheless, a for-loop can be applied containing different values for the span length and for the load conditions in such a way several optimal solutions for different values of these can be obtained. Therefore, a for-loop was run with varying the load condition q_0 from 100 kN/m^3 to 1000 kN/m^3 and span length varying from 80 m to 200 m.

Moreover, a more generalized chart can be reached by taking dimensionless parameters. Thus, a radio-to-span length ratio and rise-to-span length ratio were evaluated, and several curves were plotted.

5.3.1.1 Structural steel

For steel, the following constant data was used

Cross-section: Full circular section

- r = 0.5 m

Material: Steel

- E = 200000 MPa
- $\gamma = 78.5$ kN/m^3
- G = 76903069 MPa

The section properties were evaluated as in the previous chapter.

The following dimensionless parameters are now introduced in order to generalize the problem, in this way, it can be considered any material, span length and load condition.

- Rise-to-span ratio $\frac{f}{L}$
- Curvature radius-to-span ratio $\frac{R}{L}$

In the following figures, results for Rise-to-Span ratio vs Curvature radius-to-Span ratio, and Volume vs Initial angle are presented.

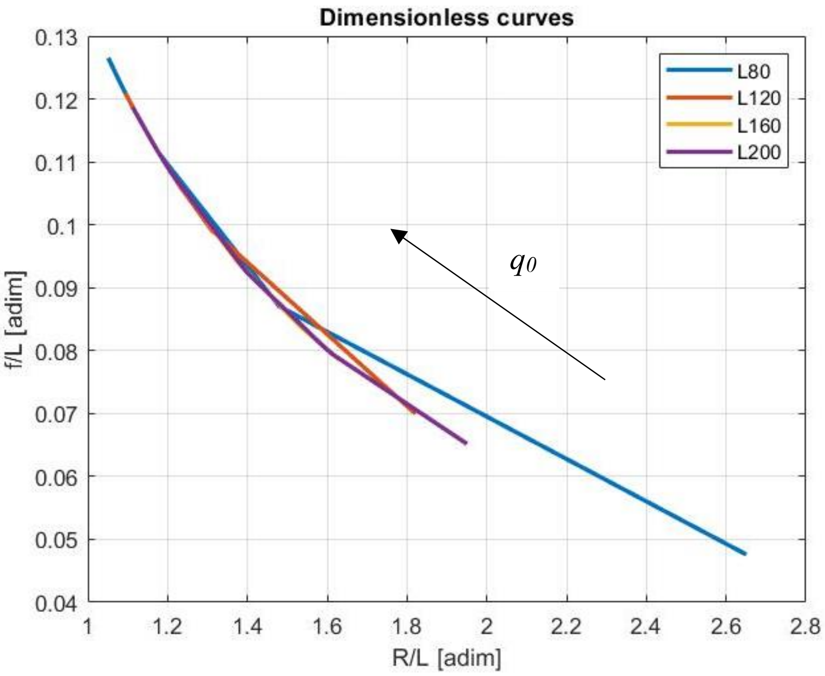

Figure 5-3. Dimensionless curves

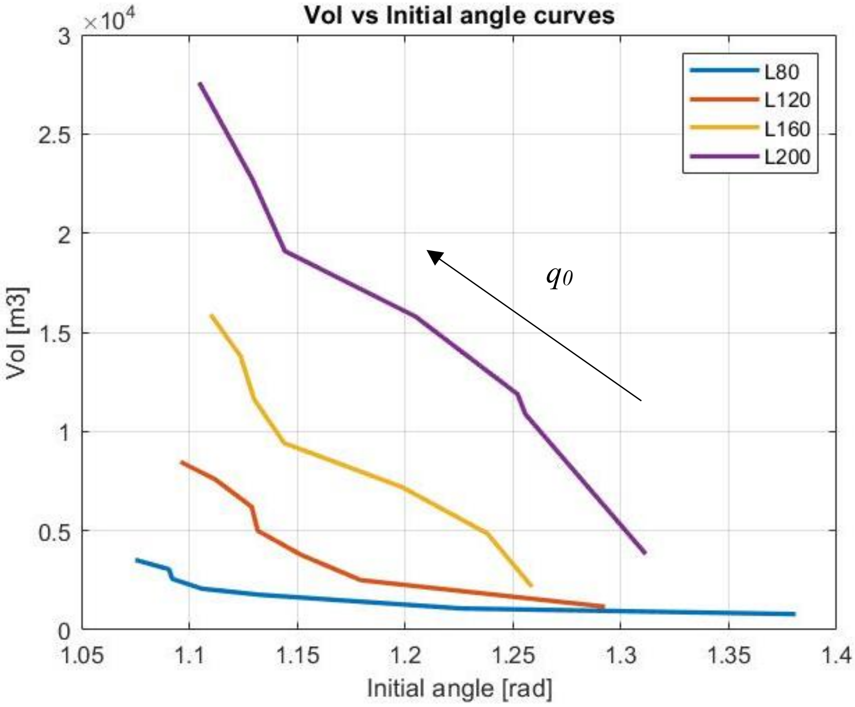

Figure 5-4. Minimum volume vs Initial angle curves

The arrow indicates the direction of load increment.

As can be seen in the last chart, there are some points breaking the curves in different slopes. This may be since the algorithm did not find the global minimum but a local one (in Figure 5-4). In order to obtain a smoother graph, the population size of the algorithm should be increased in order to the algorithm being capable to localize the global minimum. Nevertheless, this would imply a greater computational cost.

5.3.1.2 Concrete

Finally, the same can be performed for concrete. Therefore, by changing the constant data for the concrete properties, similar charts can be provided.

Besides the material properties, the constraint function must be also changed. As the concrete has different behavior for tensile stresses and compressive ones, an additional constraint regarding the tensile strength must be added. It can be simplified to no tensile strength; therefore, tensile stresses must be lesser than zero. This constraint must be added to the compressive strength verification. So, in the code will be written as a vector containing both verification

```
constraint      = [sigma_min/fcd - 1; sigma_max]
```

Chapter 6

6 Discussion of results and conclusion

Finally, conclusions about the work are performed in this chapter and the future works to be done are suggested.

6.1 Conclusions

To conclude, in the present work analytical solutions for the optimal shape and arch cross-section have been presented. They were obtained for a particular case of full circular section, constant along the arch, and for constant radius of curvature (circular arch). Anyway, the code can be re-written for other kinds of section and for parabolic arch, for instance. To this point, it is only necessary to change the calculation of the properties and to establish a law for the variable radius.

The total volume of the arch was used as objective function, i.e. the provided solutions are the minimum ones for certain fixed variables. Moreover, strength constraints were also established in order to satisfy the design requirements.

Some sensitivity analyses were carried out to obtain the optimal solutions in a dimensionless formulation. Different values for the design load and span length were considered to plot different curves and to adapt the problem to different constraints.

Beyond the particularities of the considered configuration, the results allow to underline how the design variables govern the solution.

From the first chart, the dimensionless one, it can be seen that the rise and the curvature radius are not sensitive to different span length. That is to say, whichever is the span length, the ratio between the rise and the span, for different values of span length, will have the same slope. If rise-to-curvature radius is plotted, the curves will be parallel with the same inclination. The dimensionless formulation allows us to obtain a general behaviour whichever is the span length.

Moreover, it can be noted that by decreasing the load conditions (right zone), there is a decreasing of the rise-to-span ratio with respect to the curvature radius-to-span ratio. It means that the arch will be more lowered, and consequently the curvature radius will be higher, as the load is very low.

From the second chart, Volume vs Initial angle, for lower loads (right zone), the volume needed to bear the actions is minimized, because there is an initial angle too high, i.e. a small rise, and therefore, the arch length becomes shorter. For high loads (left zone), a greater amount of volume is needed either for the bigger rise as for the needed area to bear the loads.

In this case, the needed volume will depend on the span length. Greater is the span length, the more amount of steel has to be provided.

These charts can be a useful tool for a first stage of design. Starting the pre-dimensioning with the optimal solution for certain constraints accelerates the design procedure in a proper way. Thus, by fixing material properties and span, length optimal values for rise, radius of curvature and volume can be achieved. In this way, the design variables are obtained very quick, and they can be inserted in a finite element model for a more refined analysis in order to get the corresponding optimal cross-section dimensions and an estimation of the costs.

6.2 Future works

So far, the optimization was applied to a constant cross-section, nevertheless, in order to expand the optimization process and to perform a deeper evaluation about the shape and arch cross-section, a variable cross-section along the arch can be considered. In this way, a more proper use of material can be achieved. To this point, the differential equation must be re-defined considering the area, shear area and inertia varying along the curvilinear axis.

$$N = EA_{(s)}\left(\frac{du}{ds} + \frac{v}{R}\right)$$

$$T = GA_{T(s)}\left(\frac{-u}{R} + \frac{dv}{ds} - \varphi\right)$$

$$M = EI_{(s)}\frac{d\varphi}{ds}$$

Obtaining

$$-E\left(\frac{dA_{(s)}}{ds}\frac{du}{ds} + A_{(s)}\frac{d^2u}{ds^2} + \frac{A_{(s)}}{R}\frac{dv}{ds} + \frac{dA_{(s)}}{ds}\frac{v}{R}\right) - GA_T\left(-\frac{u}{R^2} + \frac{1}{R}\frac{dv}{ds} - \frac{\varphi}{R}\right) = P_x$$

$$EA\left(\frac{1}{R}\frac{du}{ds} + \frac{v}{R^2}\right) - GA_T\left(-\frac{dA_{T(s)}}{ds}\frac{u}{R} - \frac{A_{T(s)}}{R}\frac{du}{ds} + \frac{dA_{T(s)}}{ds}\frac{dv}{ds} + A_{T(s)}\frac{d^2v}{ds^2} - \frac{dA_{T(s)}}{ds}\varphi - A_{T(s)}\frac{d\varphi}{ds}\right) = P_y$$

$$-GA_T\left(-\frac{u}{R} + \frac{dv}{ds} - \varphi\right) - E\left(\frac{dI_{(s)}}{ds}\frac{d\varphi}{ds} + I_{(s)}\frac{d^2\varphi}{ds^2}\right) = m$$

Future works regarding these aspects can be evaluated and, thus, the material properties can be exploited in a more efficient way.

Further works can be also performed regarding the coupled problem between the deck and the arch. For this issue, other alternatives, like using other type of solver or deducting the equations in a same reference system, can be evaluated in order to generalize the problem.

Bibliography

[1] Karnovsky, I. A. (2011). *Theory of arched structures: strength, stability, vibration*.

[2] Carpinteri, A. (1992). *Scienza delle costruzioni* (Vol. 1). Pitagora

[3] Shampine, L. F., Kierzenka, J., & Reichelt, M. W. (2000). *Solving boundary value problems for ordinary differential equations in MATLAB with bvp4c*. *Tutorial notes*, *2000*, 1-27..

[4] MATLAB User Guide. https://it.mathworks.com/help/matlab/

[5] Edwards, C. H., & Penney, D. E. (2009). *Ecuaciones Diferenciales Y Problemas Con Valores De Frontera. Cómputo Y Modelado. Cuarta Edición*.

[6] Trentadue, F., Fiore, A., Greco, R., Marano, G. C., & Lagaros, N. D. (2020). *Structural optimization of elastic circular arches and design criteria*. *Procedia Manufacturing*, *44*, 425-432.

[7] Eschenauer, H., Olhoff, N., & Schnell, W. (1996). *Applied structural mechanics: fundamentals of elasticity, load-bearing structures, structural optimization*.

[8] Olhoff, N.; Taylor, J.E.: (1983). *On Structural Optimization*. J. Appl. Mechanics, 50, 1139-1151.

[9] Goldberg, D. E., & Holland, J. H. (1988). *Genetic algorithms and machine learning*.

Appendix A

In the following, the code utilized in Chapter 4 is presented.

```
%% Arch structural analysis for a full circular section
clc
clear all


%% Input data

n=200;                   % Number of nodes along the arch

% Arch configuration

L=80;                    % Span                [m]
thetai_deg=30;           % Initial angle       [degrees]
thetai=thetai_deg*pi/180;% Initial angle       [rad]
beta=pi/2-thetai;        % Beta angle          [rad]
R=L/(2*sin(beta));       % Curvature radius    [m]

theta=linspace(0,2*beta,n);% Angle running the curvilinear axis[rad]
s=R*theta;               % Curved axis         [m]


% Material properties

gamma=78.5;              % Steel weight        [kN/m3]
E=1.999*10^8;            % Young modulus       [kN/m2]
G=76903069;              % Shear modulus       [kN/m2]

% Cross-section properties

r1=0.5;                  % Section radius         [m]
J = pi*r1^4/4;           % Inertia                [m4]
A=pi*(r1^2);             % Section area           [m2]
As = 0.9*A;              % Shear section area     [m2]
W=pi*(r1^4)/(4*r1);      % Section modulus        [m3]

%Dead weight

P1=-gamma*A;             % Self-weight            [kN/m]

vol=2*beta*R*A*gamma;    % Arch volume            [m3]

%% Boundary value solver

options = bvpset('stats','on','RelTol',1e-3);
solinit=bvpinit(s,[0 0 0 0 0 0]);
sol = bvp4c(@odefun,@bcfun,solinit,options);

u0=deval(sol,s);

%Displacements
```

```
u=sol.y(1,:);             % Tangential displacement   [m]
v=sol.y(2,:);             % Normal displacement       [m]
phi=sol.y(3,:);           % Rotation of section       [rad]
u1=sol.y(4,:);            % 1st derivative of u
v1=sol.y(5,:);            % 1st derivative of v
phi1=sol.y(6,:);          % 1st derivative of phi


% Internal actions

N=E*A*(u1+v/R);           % Axial force              [kN]
T=G*As*(-u/R+v1-phi);     % Shear force              [kN]
M=E*J*phi1;               % Bending moment           [kNm]

N0=N(1);                  % Axial force at s = 0
N3=N(n/2);                % Axial force at s = Smax/2
N1=N(n);                  % Axial force at s = Smax
M0=M(1);                  % Bending moment at s = 0
M3=M(n/2);                % Bending moment at s = Smax/2
M1=M(n);                  % Bending moment at s = Smax
T0=T(1);                  % Shear force at s = 0
T3=T(n/2);                % Shear force at s = Smax/2
T1=T(n);                  % Shear force at s = Smax

%Stresses

sigma=(M/W+N/A)/1000;     % Normal stress [MPa]
maxsigma=max(abs(sigma)); % Maximum normal stress


%% Plotting

figure

subplot(3,1,1);
plot(s,u,'LineWidth',2);
title('Tangential displacement')
xlabel('s [m]');
ylabel('u [m]');
grid on
xlim([0 s(n)])
ax=gca;
ax.XAxisLocation = 'origin';

subplot(3,1,2);
plot(s,v,'LineWidth',2);
title('Normal displacement')
xlabel('s [m]');
ylabel('v [m]');
grid on
xlim([0 s(n)])
ax=gca;
ax.XAxisLocation = 'origin';

subplot(3,1,3);
plot(s,phi,'LineWidth',2);
```

```
title('Rotation of section')
xlabel('s [m]');
ylabel('\phi [rad]');
grid on
xlim([0 s(n)])
ax=gca;
ax.XAxisLocation = 'origin';

figure
subplot(3,1,1);
plot(s,N,'LineWidth',2);
title('Axial force')
xlabel('s [m]');
ylabel('N [kN]');
grid on
xlim([0 s(n)])
ax=gca;
ax.XAxisLocation = 'origin';

subplot(3,1,2);
plot(s,T,'LineWidth',2);
title('Shear')
xlabel('s [m]');
ylabel('T [kN]');
grid on
xlim([0 s(n)])
ax=gca;
ax.XAxisLocation = 'origin';

subplot(3,1,3);
plot(s,M,'LineWidth',2);
title('Bending Moment')
xlabel('s [m]');
ylabel('M [kNm]');
grid on
xlim([0 s(n)])
ax=gca;
ax.XAxisLocation = 'origin';

figure
plot(s,sigma,'LineWidth',2);
title('Normal Stress')
xlabel('s [m]');
ylabel('\sigma_m_a_x [MPa]');
grid on
xlim([0 s(n)])
ax=gca;
ax.XAxisLocation = 'origin';



%% Local functions

% Differential equation system

function dzds=odefun(s,z)
```

```
% Arch configuration

L=80;                         % Span                [m]
thetai_deg=30;                % Initial angle       [degrees]
thetai=thetai_deg*pi/180;     % Initial angle       [rad]
beta=pi/2-thetai;             % Beta angle          [rad]
R=L/(2*sin(beta));            % Curvature radius    [m]

% Material properties

gamma=78.5;                   % Steel weight        [kN/m3]
E=1.999*10^8;                 % Young modulus       [kN/m2]
J=(pi*(r1^4))/4;              % Inertia             [m4]

% Cross-section properties

r1=0.5;                       % Section radius      [m]
A=pi*(r1^2);                  % Section area        [m2]
As=0.9*A;                     % Shear section area  [m2]
G=76903069;                   % Shear modulus       [m3]
q0=-50;                        % Design load        [kN/m]


P1=-gamma*A;                 % Self-weight          [kN/m]

Pn=(P1+q0)*sin(thetai+s/R);  % Normal projection to the arch of P1
Pt=(P1+q0)*cos(thetai+s/R);  % Tangential projection to the arch of
P1

% 1st order differential equation system

dzds=[z(4)
      z(5)
      z(6)
      (-G*As*(-z(1)/R^2+z(5)/R-z(3)/R)-Pt)/(E*A)-z(5)/R
      (-Pn+E*A*(z(4)/R+z(2)/R^2))/(G*As)+z(4)/R+z(6)
      -G*As*(-z(1)/R+z(5)-z(3))/(E*J)];
end

%Boundary conditions

function res=bcfun(za,zb)

res= [za(1)         % u(0)      = 0
      za(2)         % v(0)      = 0
      za(3)         % phi(0)    = 0
      zb(1)         % u(Smax)   = 0
      zb(2)         % v(Smax)   = 0
      zb(3)];       % phi(Smax) = 0
end
```

Appendix B

The following code was utilized in Chapter 5.

1. File: *Arch_optimization.m*

```
%% Shape and arch cross-section optimization

tic

clear all
clc

L = 60:20:200;

x_opt = zeros(length(L),2);
OF_opt = zeros(length(L),1);


for ii=1:length(L)

% Elaboration Data

Data.elaboration.n = 200;                    % Mesh

% Constant values

Data.arch.L     = L(1,ii);            % Span, in meters     [m]
Data.arch.gamma = 78.5;               % Specific weight     [kN/m3]
Data.arch.E     = 1.999*10^8;         % Young modulus       [kN/m2]
Data.arch.G     = 76903069;           % Shear modulus       [kN/m2]
Data.arch.q0    = -150; %q(1,ii);     % Design load         [kN/m]
Data.arch.fyd   = 275/1.15;           % Design strength     [MPa]

% Design variables

nvars = 2;          % Number of variables (Initial angle and radius)

thetai_max = 89*pi/180;

LB    = [0 0.2];   % Lower bound (1st column thetai, 2nd one radius)
UB    = [thetai_max 1.5];  % Upper bound

% Constraint function

Const_fun = @(x)Arch(x,Data);


%Objective function

OF_fun    = @(x)OF(x,Data);

% Optimization with GA
```

```
options = 
optimoptions('ga','PlotFcn','gaplotbestf','PopulationSize',100);

[x_opt(ii,:),OF_opt(ii,:)]=ga(OF_fun, nvars,[],[],[],[], LB, UB, 
Const_fun);


end

% Dimensionless plot

beta = pi/2 - x_opt(:,1);
R    = Data.arch.L./(2*sin(beta));
rise = R.*(1 - cos(beta));

rise_to_span  = rise/Data.arch.L;
radio_to_span = R/Data.arch.L;

figure
plot(radio_to_span,rise_to_span,'LineWidth',2)
xlabel('R/L [adimm]');ylabel('f/L [adim]');
title('Steel')
grid on

timeElapsed = toc;
```

2. File: *Arch.m*

```
%% Evaluation of constraint function
function [constraint, ceq] = Arch(x, Data)

n                   = Data.elaboration.n;

% Design variables

thetai              = x(1);
r1                  = x(2);

% Constant data

L                   = Data.arch.L;
gamma               = Data.arch.gamma;
E                   = Data.arch.E;
G                   = Data.arch.G;
q0                  = Data.arch.q0;
fyd                 = Data.arch.fyd;


% Geometrical parameters

J                   = (pi*(r1^4))/4;      % Inertia             [m4]
A                   = pi*(r1^2);          % Section area        [m2]
As                  =0.9*A;               % Shear area          [m2]
W                   = pi*(r1^4)/(4*r1);   % Resistant modulus   [m3]
```

```
beta                = pi/2 - thetai;
R                   = L./(2*sin(beta));   % Curvature radius    [m]


[s_vec, u0_vec, v0_vec, phi0_vec, u1_vec, v1_vec, phi1_vec] =
arch_analysis(thetai, R, r1, gamma, E, G, n, q0);

%Results of arch analysis and evaluation of internal actions and
stresses

s_vec        = s_vec;         % Curvilinear axis        [m]

u0_vec       = u0_vec;        % Tangential displacement [m]
v0_vec       = v0_vec;        % Normal displacement     [m]
phi0_vec     = phi0_vec;      % Rotation                [rad]


M_vec        = E*J*phi1_vec;                    % Bending moment [kNm]
V_vec        = G*As*(-u0_vec/R + v1_vec - phi0_vec);% Shear      [kN]
N_vec        = E*A*(u1_vec + v0_vec/R);          % Axial force    [kN]
sigma_Max_vec= (M_vec/W + N_vec/A)/1000;        % Normal stress  [MPa]

sigma_max       = max(abs(sigma_Max_vec));       % Maximum stress

% Constraint function

constraint      = sigma_max/fyd - 1;        % Strength verification

ceq             = [];

end
```

3. File: *Arch_analysis.m*

```
function [s_vec, u0_vec, v0_vec, phi0_vec, u1_vec, v1_vec, phi1_vec]
= arch_analysis(thetai, R, r1, gamma, E, G, n, q0)

% Boundary value solver

beta1   = pi/2 - thetai;
theta   = linspace(0,2*beta1,n);  %Angle running the curvilinear
axis [rad]
s       = R*theta;                %Curvilinear axis               [m]

options = bvpset('stats','on','RelTol',1e-3);
solinit = bvpinit(s,[0 0 0 0 0 0]);
sol     = bvp4c(@odefun,@bcfun,solinit,options);


u0_vec  = sol.y(1,:);
v0_vec  = sol.y(2,:);
phi0_vec= sol.y(3,:);
u1_vec  = sol.y(4,:);
```

```
v1_vec  = sol.y(5,:);
phi1_vec= sol.y(6,:);

s_vec = s;

%% EDO to solve

function dzds=odefun(s,z)

thetai_rad = thetai*pi/180;
beta       = pi/2 - thetai_rad;

J  = (pi*(r1^4))/4;
A  = pi*(r1^2);
As = 0.9*A;

P1 = -gamma*A; %[kN/m]

% Projection of forces to normal and tangential dir

Pn = (P1+q0)*sin(thetai+s/R);
Pt = (P1+q0)*cos(thetai+s/R);

% 1st order differential equation system

dzds=[z(4)
      z(5)
      z(6)
      (-G*As*(-z(1)/R^2 + z(5)/R - z(3)/R) - Pt)/(E*A) - z(5)/R
      (-Pn + E*A*(z(4)/R + z(2)/R^2))/(G*As) + z(4)/R   + z(6)
      -G*As*(-z(1)/R + z(5) - z(3))/(E*J)];
end

%% Boundary conditions

function res=bcfun(za,zb)

res= [za(1)   % u(0)      = 0
      za(2)   % v(0)      = 0
      za(3)   % phi(0)    = 0
      zb(1)   % u(Smax)   = 0
      zb(2)   % v(Smax)   = 0
      zb(3)]; % phi(Smax) = 0
end
end
```

4. File: *OF.m*

```
function vol = OF(x,Data)

% Constant data

n     = Data.elaboration.n;
L     = Data.arch.L;
gamma = Data.arch.gamma;

% Design variables

r1    = x(2);
A     = pi*(r1^2);

beta      = pi/2 - x(1);
R         = L./(2*sin(beta));
theta     = linspace(0,2*beta,n);
s         = R*theta;

% Objective Function

vol = s(end)*A*gamma;      % Volume of arch [m3] (To be minimized)

end
```