RELIABILITY-BASED DESIGN OPTIMIZATION OF A CARBON FIBER-REINFORCED POLYMER BRIDGE

by

JULIO RODRÍGUEZ SÁNCHEZ

A thesis submitted to the Department of Structural Mechanics and Hydraulic Engineering,

in partial fulfillment of the requirements for the degree of

MÁSTER DE ESTRUCTURAS

Supervisor: Dr. Guillermo Rus Carlborg

Department of Structural Mechanics and Hydraulic Engineering

University of Granada, Campus de Fuentenueva,

18001 Granada, Spain

September 2013

ABSTRACT

RELIABILITY-BASED DESIGN OPTIMIZATION OF A CARBON FIBER-REINFORCED POLYMER BRIDGE

Composite materials are gaining importance in civil engineering applications, such as bridges, due to their high stiffness and strength in relation to their low weight. The advantage of using FRP composites for civil engineering structures relies not only on their mechanical efficiency, but also on their ability to adapt to hostile environment conditions.

However, the long-term behavior of composites under fatigue and damage conditions is still partially understood. Nevertheless, the existing body of knowledge and a know-how may allow to conceptualize new designs of carbon fiber-reinforced polymer (CFRP) bridges, considering the uncertainties and optimizing the structure to be safe and cost-efficient through the reliability-based design optimization (RBDO) method.

In this paper, a new all-composite FRP bridge typology is introduced. The structural system is composed by 4 families of 5 CFRP symmetrically disposed straps, connected to 4 corner-supports and combined with a central CFRP strap. A glass fiber-reinforced polymer (GFRP) deck rests on top of several GFRP variable-section transversal beams, which transfer the loads from the deck to the main net of CFRP laminates. The new-concept of bridge was studied by a finite element (FE) model of the structure, checking that it fulfills the structural requirements stated in the design code EC-2.

This FE model was further used to adjust a surrogate model of the bridge that was subsequently used for the optimization algorithm. A damage evolution model was implemented in the optimization algorithm to consider the stiffness reduction due to fatigue damage during the lifetime. Several wind-tunnel experiments together with accelerated mechanical aging test will be used to validate the design.

The results of this work is not only a new optimized bridge concept, but also a scientific design approach that allows us to conceive rational structural designs made of new materials.

RESUMEN

OPTIMIZACIÓN BASADA EN FIABILIDAD DEL DISEÑO DE UN PUENTE DE FIBRA DE CARBONO

Los materiales compuestos continúan incrementando su importancia en aplicaciones de ingeniería civil, como por ejemplo en puentes, debido a su gran rigidez y resistencia en relación con su bajo peso. La ventaja del uso de materiales compuestos en estructuras de ingeniería civil no radica sólo en su eficiencia mecánica, sino también en su capacidad de adaptarse a condiciones ambientales agresivas.

Aunque todavía no se comprende en su totalidad el comportamiento a largo plazo de los materiales compuestos sometidos a fatiga y daño, el cuerpo de conocimiento existente sobre la materia permite conceptualizar nuevos diseños de puentes de fibra de carbono, considerando diversas incertidumbres y optimizando la estructura para que sea eficiente en cuanto a coste, a través del método de optimización basado en fiabilidad (Reliability-Based Design Optimization, RBDO).

En este trabajo se estudia una nueva tipología de puente constituido exclusivamente por materiales compuestos. El sistema estructural está compuesto por 4 familias de 5 tirantes de fibra de carbono cada una, simétricamente dispuestas, que convergen en un apoyo y se enlazan a un tirante central. Un tablero de fibra de vidrio descansa sobre varias vigas transversales de fibra de vidrio y de sección variable, que transmiten las cargas sufridas por el tablero a los tirantes de fibra de carbono, sobre los que descansan.

La nueva tipología de puente fue diseñada utilizando un modelo de elementos finitos, comprobando que el puente cumple los requisitos establecidos en el EC-2. Después, el modelo de elementos finitos se empleó para ajustar un modelo de barras del puente, menos preciso pero más rápido.

Un modelo de evolución de daño fue aplicado al algoritmo de análisis estructural, de manera que la reducción de rigidez debida a la apertura de grietas puede ser determinada a través de la vida útil del puente. Este algoritmo permite reproducir de manera más precisa las condiciones de carga y las características del material, permitiendo así un diseño más riguroso del puente. Finalmente, el modelo de la estructura se incluyó en un proceso de RBDO, del que resultó el puente óptimo en términos de fiabilidad y coste.

En definitiva, el resultado de este no es sólo un concepto innovador de puente, sino también un enfoque de diseño científico que permite a los ingenieros concebir nuevos diseños racionales de estructuras hechas con materiales avanzados.

ACKNOWLEDGMENTS

I would like to thank the responsible for the direction of my research, Dr. Guillermo Rus Carlborg of the Department of Structural Mechanics. He brought me back to University to work in the exciting area of composites materials. His philosophical thinking has had a great influence on my work throughout this time. I couldn't forget my friends and colleges of the Non Destructive Evaluation Laboratory. There have been a lot of enjoyable moments in the collaborations and I have learned much from them, especially in the areas outside my research topic. Also I would like to thank my colleges of the Department of Structural Mechanics for their friendly help and advise.

And finally, I need to express my sincere gratitude to my family. I'm in debt with them for the comprehension I receive in my Phd work.

AUTORIZACIÓN

D. .....

Profesor del departamento de

.....

de la Universidad de Granada, como director del Proyecto Fin de Máster de D.

.....

Informa:

que el presente trabajo, titulado:

.....

Ha sido realizado y redactado por el mencionado alumno bajo nuetra dirección, y con esta fecha autorizo a su presentación.

Granada, a ..... de ..... de 20.....

Fdo. .....

Los abajo firmantes autorizan a que la presente copia de Proyecto Fin de Máster se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a ..... de ..... de 20.....

(Firmas y números de DNI / NIE del alumno y de los tutores)

Informe de valoración del proyecto

El tribunal constituido para la evaluación del Proyecto Fin de Máster titulado:

.....

Realizado por el alumno:

.....

Y dirigido por el tutor:

.....

Ha resuelto la calificación de:

- ☐ SOBRESALIENTE (9-10 puntos)
- ☐ NOTABLE (7-8.9 puntos)
- ☐ APROBADO (5-6.9 puntos)
- ☐ SUSPENSO

Con la nota[1]: ☐ puntos.

El Presidente: .....

El Secretario: .....

El Vocal: .....

Granada, a ..... de ..... de 20.....

[1]Solamente con un decimal.

Contents

Resumen **iv**

Abstract **1**

1 Introduction **1**
1.1 Motivation and Objectives 1
1.2 Thesis Organization 8

2 Methodology **9**
2.1 Geometry parameterization and FE model 10
2.2 Surrogate Model 14
2.3 Damage evolution 17
2.4 Optimization function 18

3 Results **23**
3.1 Mesh Convergence 23
3.2 Design checks 23
3.3 Surrogate model adjustment 25
3.4 Optimum bridge 28
3.5 Optimum bridge design checks 35

4 Discussion and Conclusions **38**
4.1 Discussion 38
4.2 Conclusions 42

A Parametric FE Model of the Bridge Codes **44**

B Surrogate Model Adjustment Codes **186**

C Parametric Lifetime MA Model of the Bridge Codes **194**

List of Figures **251**

List of Tables **253**

Bibliography **255**

Chapter 1

Introduction

1.1 Motivation and Objectives

In addition to its inherent complexities, the problem of optimal design with composites materials cannot be isolated from sources of uncertainty. Composite structures are subjected to variable loads that cannot be taken into account in an accurate way following the current design specifications. Design codes overcome this deficiency by imposing high safety factors that lead to a more expensive design. Also, civil engineering infrastructures lifetime is usually prescribed by codes, which might not be the optimum lifespan for the structural system. Uncertainty turns even more essential for civil engineering design of composite structures like bridges, which bear numerous random load cycles that affect composite laminates in terms of material degradation. A new design approach for composite structures might be developed, that permits engineers to optimize costs of structural designs including sources of uncertainty and material properties evolution through time, surpassing this way the overconservative safety factors commonly used in today's engineering.

The goal of structural engineering design is to conceive a structure that maximizes utility of the customers while minimizes the construction, main-

tenance, operation and dismantling costs that society will suffer. This objective turns more complicated when uncertainty in loads and material properties is considered, and different failure mechanisms are taken into account. If a fail occurs, the utility of the structure is reduced, so a low probability of failure of the system has to be satisfied while trying to reduce the cost of the infrastructure. Therefore, a compromise should be sought between the risk of utility and the cost over the lifespan of the structure. [1].

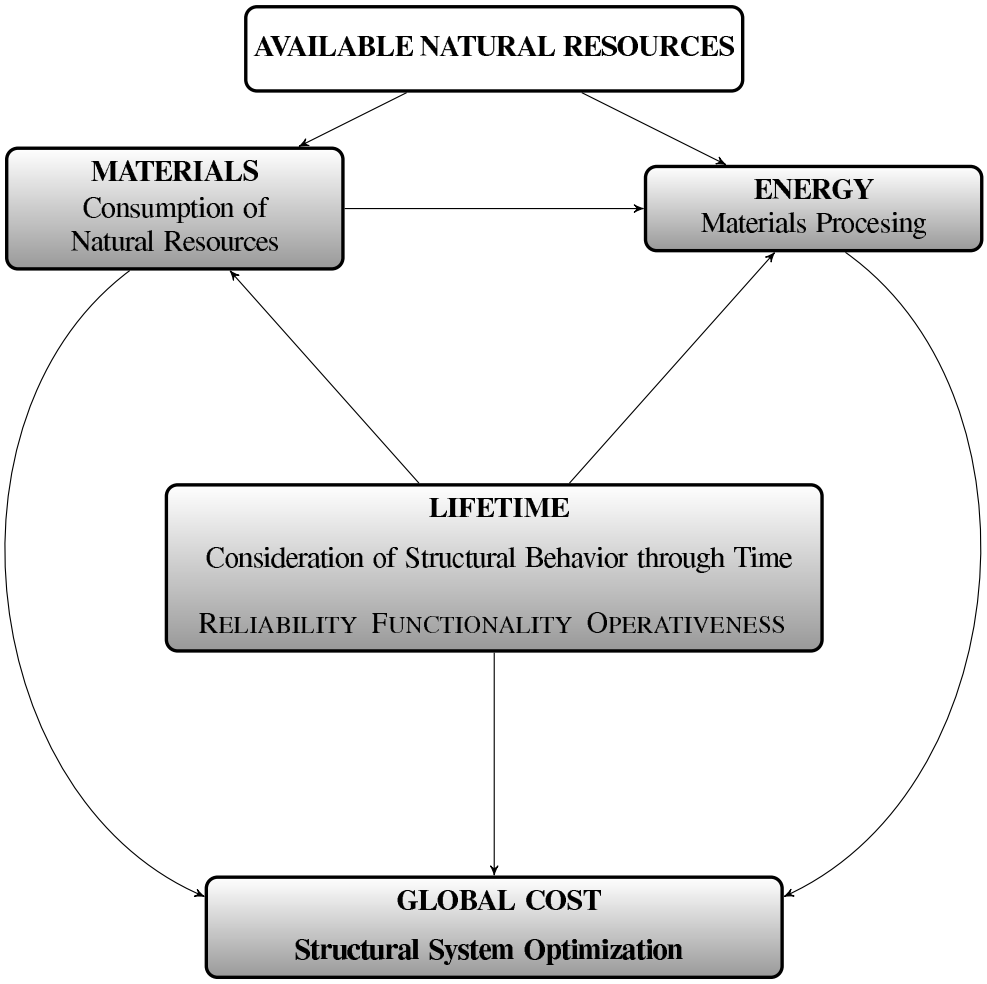


Figure 1.1: Global structural system optimization flow chart.

Reliability is defined as the inverse of the failure probability. The reliability-based design optimization method (RBDO) permits to design for a specific risk or failure mechanism, with a certain reliability level, including sources of uncertainty. The RBDO method has a promising potential, but it is still

unexplored. Its main difficulty comes from the computational complexity of the high-cost stochastic algorithms utilized in the analysis, which results in a drawback compared to the conventional deterministic design approaches. [2].

Carbon fiber-reinforced polymers (CFRP) have been used for decades in several fields, such as aerospace or bridge decks, and are increasing their importance in civil engineering constructions [3–5]. An intense research has been made in CFRP towards reliability design methods for CFRP laminates design [6].

The all-composite bridge design presented in this paper was registered as patent [7]. The bridge structure corresponds to a flipped arch bridge, which will be subjected almost exclusively to tension stresses due to vertical loads. Since CFRPs present high tensile stress resistance, the shape of the structure permits to take advantage of the material properties, allowing the designer to reduce the amount of material needed [8]. The bridge is composed by 4 families of 5 CFRP straps symmetrically allocated, starting each of them from a common support, and that are connected to a set of central CFRP straps at different points. 17 variable cross section-glass fiber reinforced polymer (GFRP) beams transfer the load applied to a GFRP plate that rests on top of them to the CFRP straps bed. Every structural component of the bridge is bonded by an adhesive resin to the others in contact with it. The construction process is as follows: the bridge is first subjected to its own weight and a small deflection is permitted, then the extremes of the deck are anchored and finally the traffic is allowed to pass over it. This way, the structure will improve its mechanical behavior by selfstressing itself.

Damage characterization of CFRP is crucial to address a study on RBDO of CFRP structures. The complex behavior of damaged CFRP laminates and the large number of variables needed to describe it makes

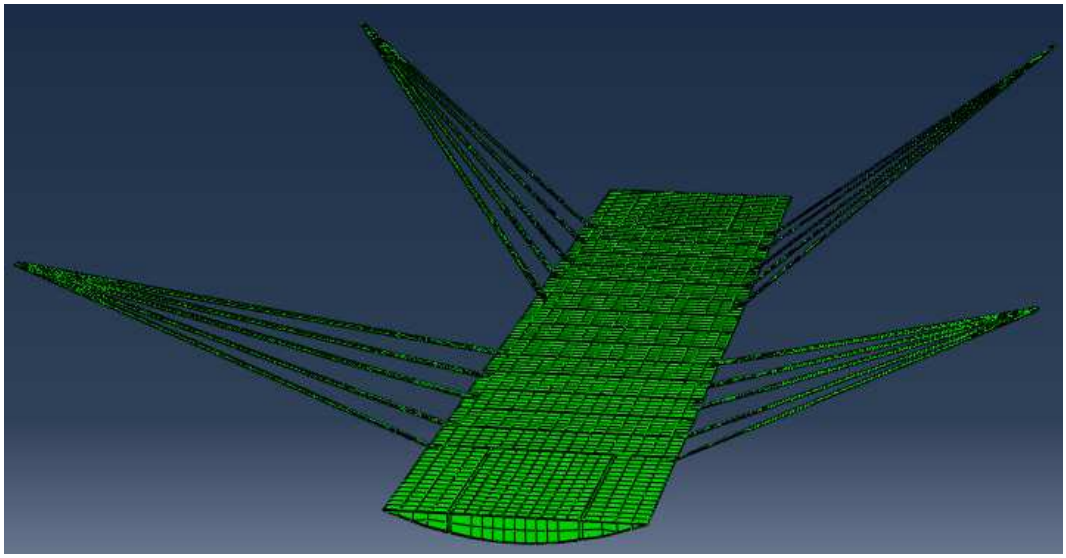
difficult to estimate the progression of failure and its incidence in the mechanical properties of the material. Several methods have been proposed to model damage initiation and evolution and stiffness degradation in CFRP members that are capable of reproducing experimental results [9–11]. Thus, there is enough knowledge about CFRP to perform a RBDO approach for a CFRP structure.

Finite Elements (FE) models are widely used in structural analysis because of its accuracy in determining displacement and stresses, but are inadequate for RBDO due to its high computational cost [12]. This way, simpler models should be developed that permit to dramatically reduce the computation time of the structural analysis without loss of accuracy. Thus, the surrogate-base models are mathematical models that use data drawn from FE models to provide fast approximations of structural analysis, useful for optimization purposes [13].

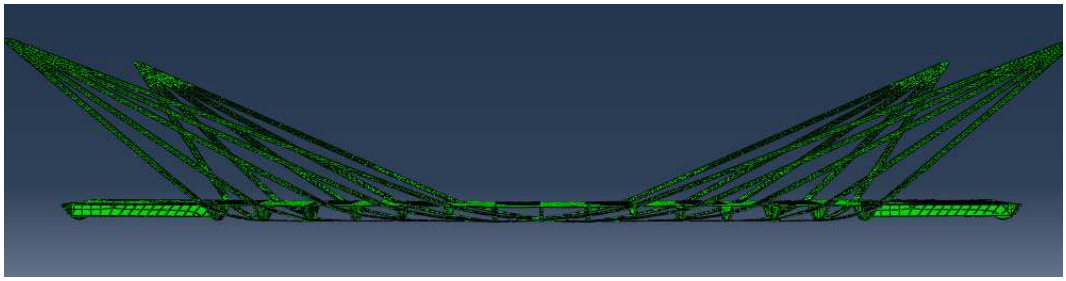
Although RBDO has been already applied in structural system design, it has never been utilized in optimization of civil engineering composite structures, and also the new mode shape-based surrogate model is an incoming physics-meaningful approximation method capable of modeling behavior of structures through its lifetime and useful for RBDO purposes. Additionally, a cost function of the bridge that accounts for design, material, construction, maintenance and monitoring cost, and lifespan of the structure will be optimized.

The goal of this paper is to optimize a new-shaped selfstressed CFRP bridge through the RBDO. In order to achieve the optimal design of the new-concept all-composite bridge, a finite element model of the structure will be developed. Then, a new surrogate model approach, which utilizes mode shapes of the structure to adjust displacements and stresses, will be generated, and finally applied to the optimization of the cost of the bridge

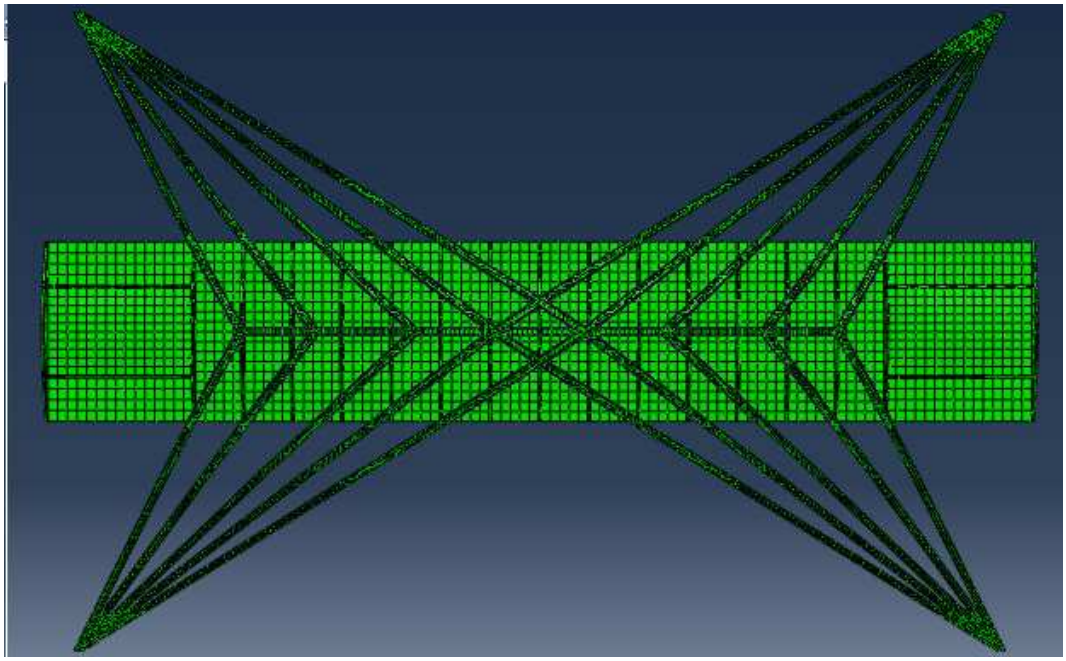
through its whole life, including geometrical and sectional properties, and also monitoring equipment and the lifespan itself as optimization variables. This way, an innovative, highly-precise design method capable of modeling the performance of the structure through its lifetime will be stated, allowing engineers to dramatically reduce safety factors and, consequently, the cost of the structure-monitoring system.



(a) General view of the bridge.



(b) Lateral view of the bridge.

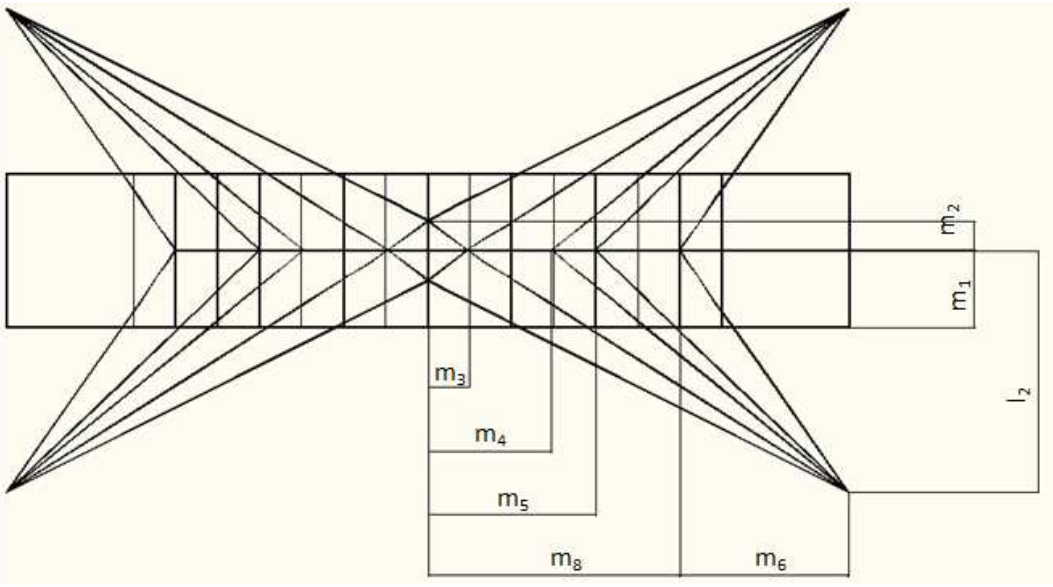


(c) Bottom view of the bridge.

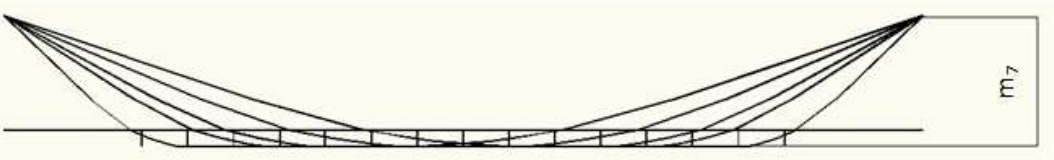
Figure 1.2: 3D Model of the geometry of the CFRP Bridge

Parameter	Minimum Value [m]	Maximum Value [m]
m_1	0.5	0.5
m_2	0.2	0.4
m_3	0.3	0.6
m_4	1.0	1.5
m_5	2.0	2.5
m_6	1.2	1.2
m_7	0.6	1.75
m_8	3.0	3.0
$l_1 = (5 - m_6)$	3.8	3.8
l_2	1.5	1.5

Table 1.1: Value ranges for the geometric parameters of the bridge.



(a) Bottom view.



(b) Lateral view.

Figure 1.3: Sketch of the geometry of the CFRP Bridge

1.2 Thesis Organization

This thesis is organized as follows: The present chapter deals with the motivation and organization of the research work presented herein. Chapter 2 to 4 are dedicated to the design and reliability-based optimization of the bridge type, and it is presented in the format of a scientist paper prepared to be submitted to *Journal of Composite Structures*. Finally, this document is closed with three appendices: Appendix A shows the MatLab and Python code scripts for the parametric finite element model of the bridge, Appendix B comprises the scripts utilized to adjust the surrogate model, and finally Appendix C presents the codes programmed in MatLab to simulate the lifetime of the bridge.

Chapter 2

Methodology

In order to achieve the optimal design of the new-concept all-composite bridge, a finite element model of the structure will be developed. To overcome the computational cost, a new surrogate model approach, which utilizes mode shapes of the structure to adjust displacements and stresses, will be generated, and finally applied to the optimization of the cost of the bridge through its whole lifespan, including geometrical and sectional properties, and also monitoring equipment as optimization variables. The optimization process is shown in the following flow chart.

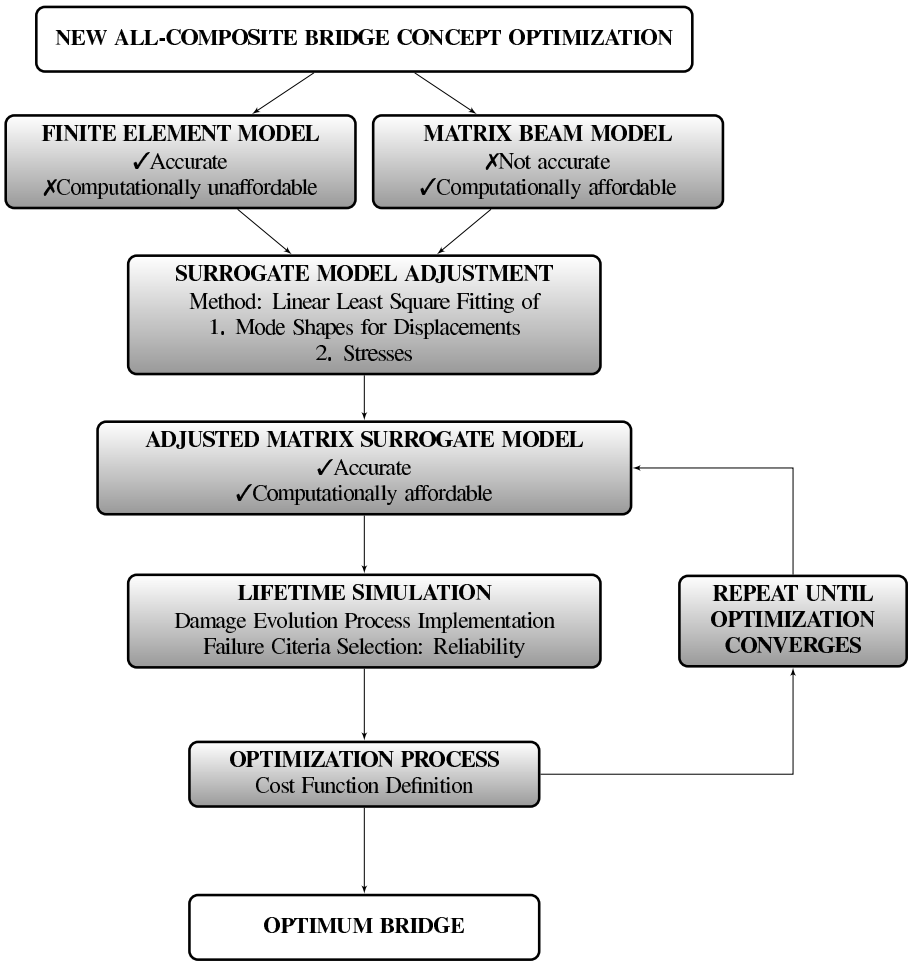


Figure 2.1: Bridge optimization process flow chart.

2.1 Geometry parameterization and FE model

The optimization process requires a mathematical model of the behavior of the bridge. This model is used to compute all those variables involved in the optimization function, as maximum displacements, maximum stresses, cost and damage of structural members during the lifespan of the bridge, among others.

As described earlier, the optimization includes geometric and sectional variables of the members of the bridge, such as, for example, pylons height, straps width or central square opening. Therefore, a parametric model of the behavior of the bridge is needed. The parametric model of the bridge

permits to define the geometry and sectional properties of the structure from the set of master variables previously specified, and obtain through a structural analysis the values of the variables included in the optimization function for different geometric and sectional configurations of the structure. This way, the parametric model of the bridge allows calculation of different individuals in order to select the best specimen, according to the optimization function. The design variables of the bridge are: the position of straps-central axis joints, the amplitude of the central square, the pylons height and the straps width.

Three different types of CFRP straps are used: one corresponding to the central straps, another to the extreme straps of the families and the last for the inner straps of every family. The stacking sequence of all the CFRP straps is $[0_2/90_4]_s$. Also, two different GFRP laminates are utilized: $[0_2/90_4]_s$ for the variable cross-section beams and $[0_4/90_8]_s$ for the deck plate. The bridge is simply-supported at the four CFRP strap family supports and at the extreme GFRP beams that support the deck plate. This way, the span of the bridge is equal to its total length.

CFRP Family	Parameter Name	Minimum Value m	Maximum Value m
Central Axis	b_1	0.02	0.10
Exterior Bridles	b_2	0.02	0.10
Interior Bridles	b_3	0.02	0.10

Table 2.1: Value ranges for the sectional properties of the bridge.

Two different models were developed: a FE model implemented in ABAQUS® and a matrix analysis (MA) model programmed in MATLAB®. The FE model is more accurate than the MA model because of its more comprehensive characterization of the mechanical behavior of structures, but it requires higher computational resources. For optimization purposes it is

non-viable to use the FE model because a very long time is needed to converge to the optimum solution. For this reason, a more accurate FE model was developed to adjust a less accurate but more computationally-efficient MA model, so the latter might be used in the optimization procedure stated previously in this work. The FE model is a parametric FE shell model of the bridge implemented in the software ABAQUS. The FE analysis provides the stresses and displacements that are needed to find the optimum bridge given a FE model of the structure. Also, the FE model allows to compute the structural behavior including the geometric non-linearity that characterizes the type of bridge described in this work. A code in Matlab creates the geometry of the bridge, and then ABAQUS is called to solve the structural analysis programmed in a Python script.

All the boundary conditions, contacts and material properties described earlier in this paper were applied, and an adaptative mesh was developed. The mesh was created to be denser near the structural elements that will develop higher stresses, as supports and contact points. Two different contacts were defined: one between the top surfaces of the straps and the bottom surfaces of the bottom flanges of the GFRP beams, and another between the top surfaces of the top flanges of those beams and the bottom surface of the deck plate. The straps and the top flanges surfaces were assumed to be the master surfaces for each contact. The surfaces in contact are supposed to be tied, which means that they will present the same displacements, being bonded together. Later, a mesh study was conducted, achieving convergence of the solution. The sizes of the elements in the selected mesh are of 1 cm for the CFRP straps and of 10 cm for the GFRP beams and deck.

Then, since the FE model was assumed to accurately describe the behavior of the bridge, the design checks of the structure stated by the norm IAP-11 [14] and the Eurocode 2 [15] were performed using this model. Three

load cases were studied: the serviceability limit state, the ultimate limit state and the vibrations limit state.

For every design check two load steps were considered. First, the bridge is selfstressed by its own weight, and then the traffic load is applied. The value of the traffic load depends on the design check considered. Also, the middle points of every geometric and sectional variables interval were taken to perform the design checks. For the serviceability limit state, the bridge model was subjected to the frequent serviceability load combination, that is composed by the dead load of the structure and a uniform traffic load of 2 kPa applied on the deck plate. The maximum displacement of the bridge is computed as half the difference between the maximum displacement obtained after the traffic load is applied and the maximum displacement of the bridge subjected to its own weight only. This definition responds to construction considerations, because once the bridge is installed, it deflects subjected to dead load only and then the deck is fixed, and after that the traffic is allowed over the deck. In addition, a countershaft is permitted for design, as it is a common practice for optimizing pedestrian bridges designs. The countershaft is assumed to be half the maximum displacement of the bridge previously defined, up to a maximum of 2.5 cm. For the ultimate limit state, the bridge model was subjected to the extreme ultimate load combination, that is composed by the dead load of the structure multiplied by 1.35 and a uniform traffic load of 6.75 kPa applied on the deck plate. The maximum stress considered is the maximum von Mises stress in the bridge for the extreme load case. For the vibration limit state, the bridge model is subjected to its own weight only, with the traffic load reduced to zero. For this load case, the first global natural vibration frequency is obtained through a frequency analysis performed in ABAQUS. A global natural vibration frequency is the frequency of a mode shape that involves the major

part of the structure [16].

Later on, a set of 1000 sample bridges were computed. The sample set was design using the Latin Hypercube Algorithm [17], with the geometric and sectional properties inside the intervals previously defined. The applied load was the frequent serviceability load defined by the IAP-11. The stresses and displacements obtained with ABAQUS were used afterwards to adjust the less accurate MA model.

The MA model is a parametric 12 DOF MA bar model of the bridge programmed in MATLAB. The code creates the geometry of the bar model of the bridge, and then an algorithm that implements the matrix analysis of bars is run to solve the structural analysis. The boundary conditions and material properties were also implemented in this model, but only the frequent serviceability load was applied. The geometric non-linearity of the bridge was introduced in the model following the procedure described in [18]. The MA model provides the displacements of the nodes of the structure, and the stresses are computed as the strains multiplied by the Young's Modulus of the members. The strains are defined as the length increment of a bar divided by its initial length.

2.2 Surrogate Model

A fast and accurate model of the structure is needed to run the optimization algorithm, because of computational cost limitations. For that reason the MA model was adjusted using data acquired through the FE model of the bridge in ABAQUS [19]. The same set of 1000 bridges was computed using the MA model, and the results were compared to those obtained with the FE model implemented in ABAQUS. A Linear Least Square Fitting (LLSF) was proposed to adjust the differences in displacements and stresses between the two models [20].

A surrogate model approach based on the mode shapes of the structure was implemented to perform the adjustment between the two different bridge models. This type of surrogate model is a physics-based adjustment which allows to decrease the number of adjustment parameters needed, since the mode shapes comprise the displacements of all the degrees of freedom in the most common deflections of the bridge. In this approach, adjustment coefficients equal to the selected number of mode shapes were obtained for each optimization parameter. The first 11 mode shapes of the bridge were acquired through the FE model in ABAQUS, and a fitting analysis was performed in order to determine the number of mode shapes to take into account for adjustment considerations, resulting that 8 mode shapes minimizes the error in estimating the maximum vertical displacement of the bridge with a reduced global error for all other degrees of freedom.

Six degrees of freedom for the 115 nodes that model to the CFRP straps in the MA model were adjusted using a LLSF with the first 8 global mode shapes of the bridge, each of them multiplied by a coefficient and the value of one of the 8 geometric and sectional optimization parameters. The mode shapes approach permits to reduce the number of adjustment parameters from $6\,\text{DOF} \times 115\,\text{nodes} \times 8\,\text{variables} = 5520\,\text{parameters}$ to $8\,\text{variables} \times 8\,\text{modeshapes} = 64\,\text{parameters}$, dramatically diminishing the sample size needed to calculate them. The first 8 mode shapes were utilized to provide a better adjustment, as will be shown later.

The maximum stress of each of the three families of straps was adjusted by a LLSF using a coefficient that multiplies each optimization parameter for each maximum stresses of the families of CFRP members. The optimization parameters were previously defined as Jeffreys' parameters [21], using the upper and bottom limit of the intervals defined for each of them. This way, a total of 80 coefficients for displacements and 24 coefficients for

stresses were computed. The system of equations of the LLSF for displacements is:

$$\Delta u_i = \sum_{j=1}^{N_s} \sum_{k=1}^{N_v} \alpha_{jk} v_k \{m_j\} \tag{2.1}$$

Where N_s and N_v are the numbers of mode shapes and optimization variables, Δu_i are the difference in displacements obtained through the FE analysis and the ones given by the MA model for each DOF i, α_{jk} are the adjustment coefficients, v_k the design variables and $\{m_j\}$ the mode shapes. The system of equations of the LLSF for stresses is:

$$\Delta s_f = \sum_{k=1}^{N_v} \alpha_k v_k \tag{2.2}$$

Where Δs_f are the difference in stresses obtained through the FE analysis and the MA model for each strap family f, $\{\alpha_k\}$ are the adjustment coefficients and v_k the design variables.

In order to validate the accuracy of the adjustments the vector norm of the difference between the approximation and the values computed by the FE model for displacements and stresses was computed for all samples, and then its mean was plotted versus the number of samples utilized. The norm of the difference is defined as:

$$\{d\} = \{u\} - \{u^*\} \tag{2.3}$$

$$\|e\| = \sqrt{\sum \{d_i\}^2} \tag{2.4}$$

Where $\{u\}$ are the displacements obtained through the FE analysis and $\{u^*\}$ are the displacements given by the MA model. Then, the error in estimating the maximum displacement of the bridge is determined for all the samples taken into account, and the mean values of this error are determined. The same procedure was followed to estimate the adequacy of the adjustment of stresses.

2.3 Damage evolution

The damage evolution algorithm in the bridge structural components was developed using the damage mechanisms described by Talreja y Singh [22]. First, the Young's Modulus of each member of the bridge is computed, and the initial micro crack density is stated. Second, the analysis of the structure subjected to the frequent serviceability load is performed, obtaining the maximum stresses of each of the 3 different types of members. Then, with those maximum stresses 3 energy release rates are calculated, and lately inserted in a previously empirically adjusted Paris Law to determine the new micro crack densities [23]. Finally, with those new micro crack densities the reduction in Young's Modulus caused by damage due to micro crack propagation for the 3 straps families are obtained through a variational model [24]. The new Young's Modulus are utilized for a new structural analysis. Therefore, this algorithm permits to model the progression of damage in structural members of CFRP.

The continuous and discrete Paris Law are presented:

$$\frac{\mathrm{d}\rho}{\mathrm{d}n} = A(\Delta G)^{\alpha} \tag{2.5}$$

$$\rho_n = \rho_{n-1} + A\left(\Delta G(\rho_{n-1})\right)^{\alpha} \tag{2.6}$$

The energy release rate is computed as:

$$\Delta G = \frac{\sigma h}{2\rho t_{90}}\left(E^*\left(\frac{\rho}{2}\right) - E^*(\rho)\right) \tag{2.7}$$

Where ρ is the micro crack density, ΔG the energy release rate, A, α are empyricall constants, σ is the axial tension acting in the laminate, h is the laminate thickness, t_{90} is the thickness of all [90°] plies and $E^*(\rho)$ is the Young's modulus of the laminate with micro crack density ρ.

It is important to highlight that with this algorithm the damage of all members in the strap family is supposed to be equal to the highest

local damage encountered in that family. The need of data to adjust the Paris Law was the main reason to select an invariant $[0_2/90_4]_s$ stacking sequence. That laminate was tested by [25], and sufficient data was acquired to properly adjust the variables needed for the damage evolution algorithm.

The scheme of the damage evolution algorithm is shown in Figure 2.2.

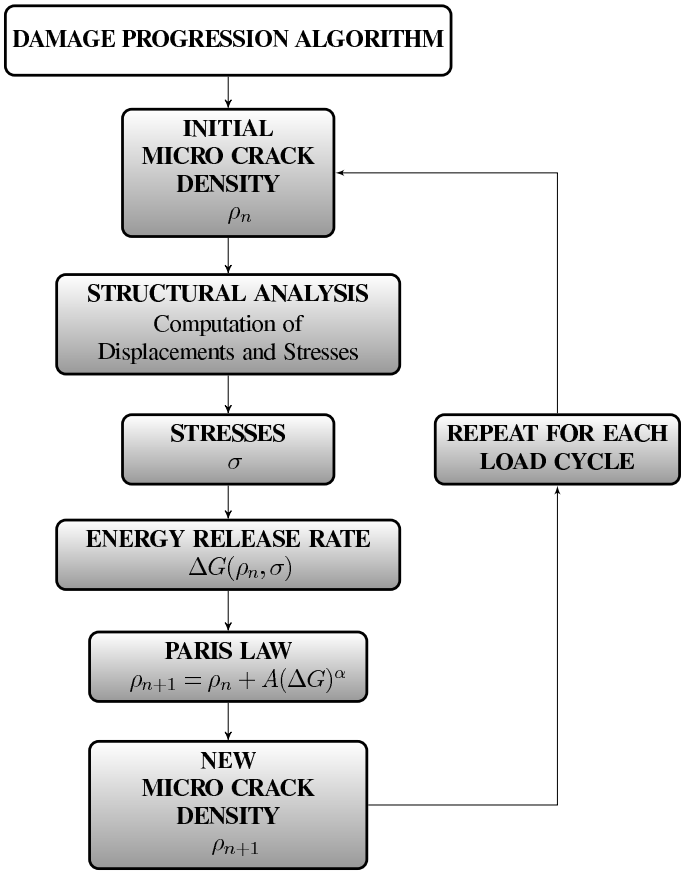


Figure 2.2: Damage evolution algorithm flow chart.

2.4 Optimization function

With the adjusted MA model, the multiobjective optimization algorithm NSGA2 [26] was run to obtain the bridge with the minimum yearly cost. The NSGA2 algorithm provides a Pareto front, which is a space of minimum solutions from where the optimum bridge will be selected, regarding the given constraints. An optimization of 5 generations of 20 individuals each

was selected. The cost of the bridge is the sum of the initial cost, the monitoring cost and the maintenance and repair cost that eventually will be needed during the lifetime of the bridge.

Parameter	Minimum Value	Maximum Value
m_1	0.5 m	0.5 m
m_2	0.2 m	0.4 m
m_3	0.3 m	0.6 m
m_4	1.0 m	1.5 m
m_5	2.0 m	2.5 m
m_6	1.2 m	1.2 m
m_7	0.6 m	1.75 m
m_8	3.0 m	3.0 m
$l_1 = (5 - m_6)$	3.8 m	3.8 m
l_2	1.5 m	1.5 m
Accelerometers, N_{ac}	4 units	12 units
US Sensors, N_{us}	12 units	12 units
Quality of US Sensors, Q_{us}	Low(= 1)	High(= 2)
Cycles, N_k	50 cycles	250 cycles

Table 2.2: Value ranges for the optimization variables.

The initial investment is composed by material cost, design cost, construction cost and monitoring equipment cost. The monitoring cost comprises the money that will be spent in processing the data acquired and the maintenance of the sensors. Finally, the maintenance and repair costs are the expenditures that will take place if the bridge fails, either in serviceability or in ultimate limit state. The bridge fails in serviceability limit state if the maximum displacement surpasses by twice the maximum displacement permitted by the IAP-11. The bridge fails in ultimate limit state if the

maximum stress of a member of the structure reaches a value higher than the yielding stress of CFRP.

The maintenance and repair costs depend on the bridge management decisions that will be based on the information registered by the sensors. This way, this part of the total cost of the structure will be tied to the probability of failure of the bridge and the probability of detection of failure by the sensors. Then, these costs will be computed as the product of the probability of taking a certain management decision and the cost of performing that action.

The maintenance and repair costs are computed for every load cycle. A load cycle is defined as one application of the frequent serviceability load that was described before. The lifespan of the bridge is the number of cycles that the bridge can take until the micro crack density reaches a value called micro crack saturation density, when delamination of the CFRP structural members initiates. This micro crack saturation density is equal to 450 micro cracks per meter for the chosen laminate stacking sequence [25].

The cost is finally divided by the lifespan of the bridge to determine the yearly cost of the structure. Also, the monitoring equipment was included in the optimization. Two different monitoring systems were selected: accelerometers to detect displacements of the bridge, and ultrasonic monitoring equipment to assess damage in laminates. The probability of detection (POD) curves of both types of sensors are according to [27–29].A minimum and a maximum number of accelerometers with a certain probability of detection were established, setting the amount of sensors as an optimization variable. The amount of ultrasonic devices to be installed was fixed at 12, and two different equipment qualities were introduced into the optimization process. Then, the number of accelerometers and the quality of the ultrasonic sensors are to be optimized.

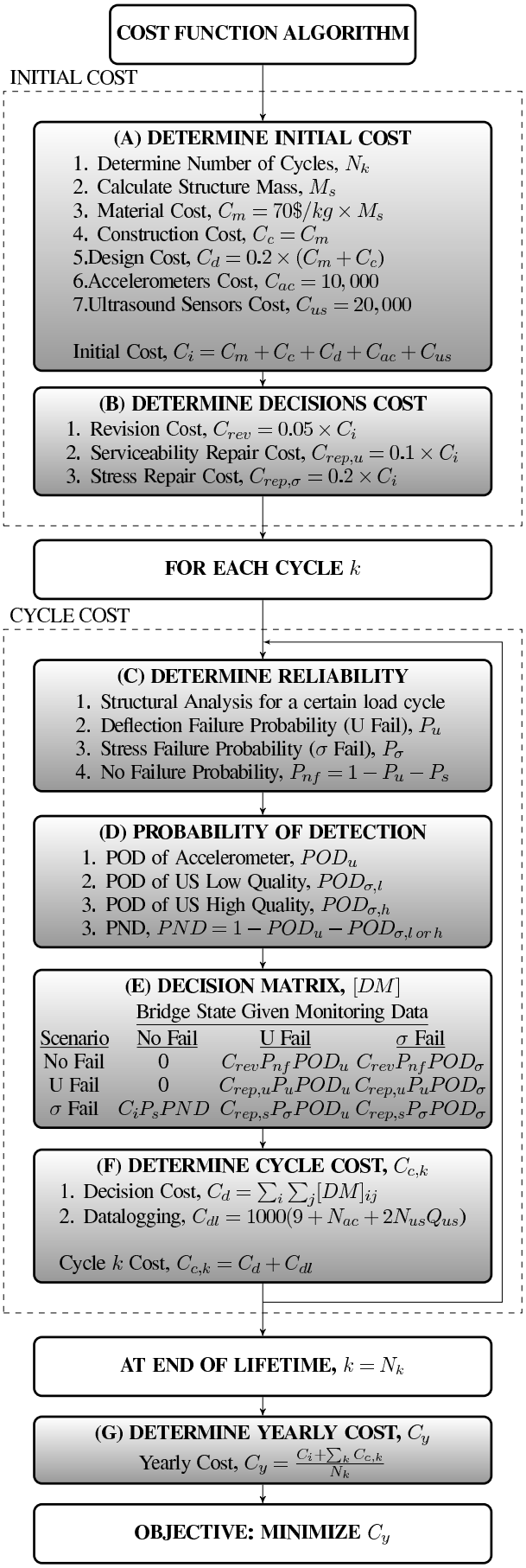


Figure 2.3: Cost function computation algorithm flow chart.

The Young's modulus of fibers in CFRP laminates was taken as a Gaussian variable, with standard deviation equal to the tenth of its mean. For optimization purposes a set of 20 different Young's moduli were utilized for each bridge A chart stating all the assumptions and hypothesis of the cost function is presented.

Since the most restrictive design check is the serviceability limit state, the optimum bridge is suppose to be such that it meets the deflection limit stated by the IAP-11 at the end of its lifetime, which should be as long as possible in order to minimize the yearly expenditures of the structure, since the initial cost is the major past of the total cost of the bridge. The expected optimum monitoring system will be composed of both accelerometers and ultrasonic devices, since the two are needed to select the best bridge management decision at each moment and to avoid expenditures due to catastrophic failure of the structure.

Nevertheless, since the cost associated with an undetected serviceability failure was set to 0, it might happen that the optimum bridge configuration is such that the structure meets the ultimate limit state requirements during its whole life, but ignores serviceability limit state by ommiting accelerometers, which are utilized to measure deflections, as monitoring equipment. This will be a mathematicall solution for the optimization of the bridge that, despite of the fact that it resists the applied loads during its lifetime, is not an acceptable solution following the IAP-11 specifications, so it is not considered as a possible structural solution.

Chapter 3

Results

3.1 Mesh Convergence

The mesh convergence study gives the following results.

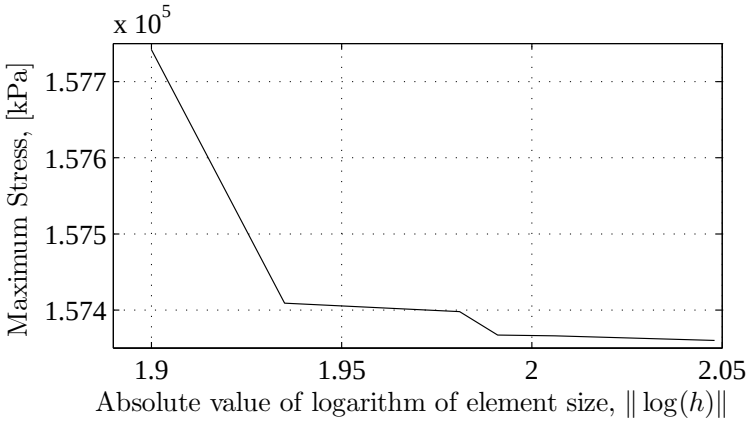


Figure 3.1: Mesh convergence study on maximum stress in kPa. Convergence of the maximum stress computed by the FE model is achieved when the number of elements increments

3.2 Design checks

The design checks stated by the IAP-11 code for the bridge were performed, and the results are shown in the following figures.

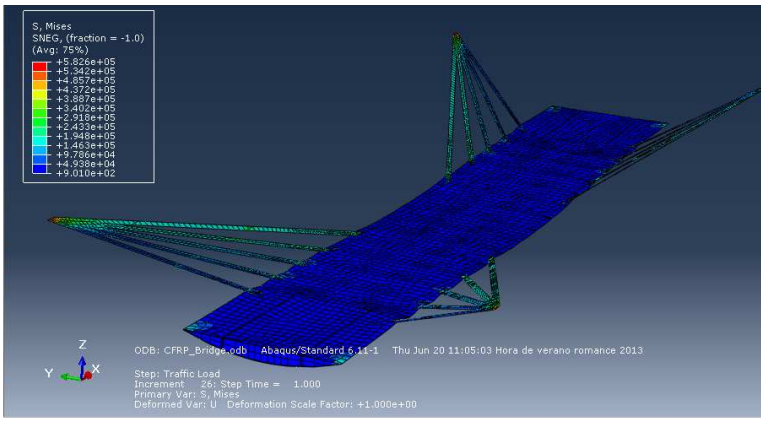


Figure 3.2: The maximum stress is reached at the straps supports, and that it is lower than the yielding stress of CFRP.

Maximum Stress [kPa]	Limit Stress by IAP-11 [kPa]
5.83×10^5	12×10^5

Table 3.1: Value of maximum stress in the bridge and yielding stress of CFRP.

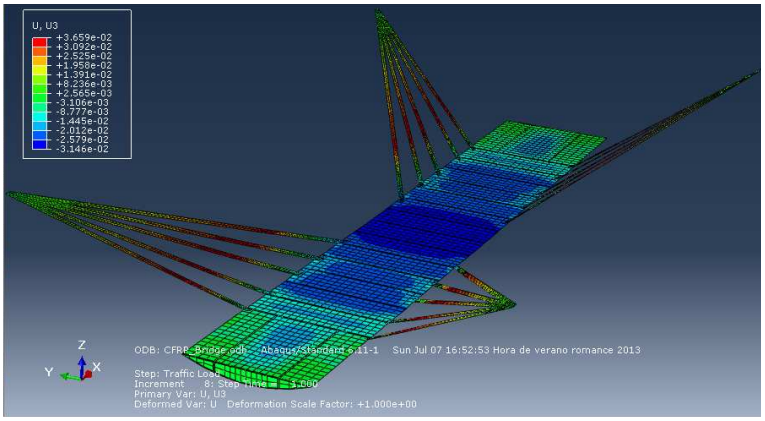


Figure 3.3: Note that the maximum vertical displacement develops at midspan, and that its value is about twice the maximum value permitted by the IAP-11. Nevertheless, a tolerable displacement can be achieved by optimizaing the geometry of the bridge and the sections of the CFRP members.

Maximum Displacement [m]	Limit Displacement by IAP-11 [m]
3.2×10^-2	1.66×10^-2

Table 3.2: Value of maximum vertical displacement of the bridge and maximum vertical displacement prescribed by the IAP-11.

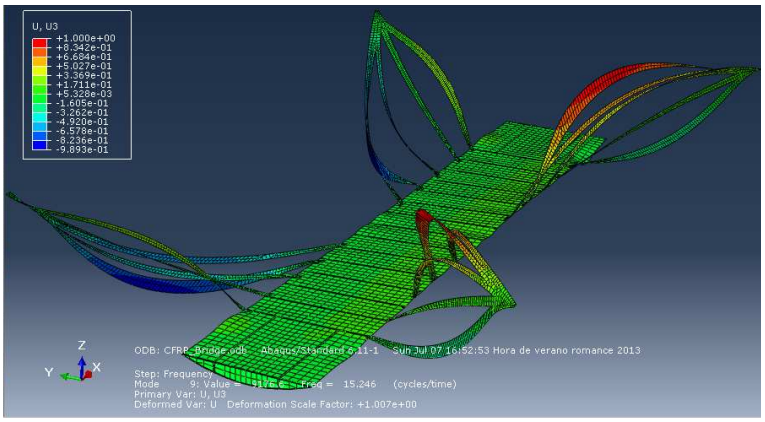


Figure 3.4: The frequency of the first mode shape shown in this figure is 15.2 Hz, which is above the minimum frequency allowed by the IAP-11.

Frequency [Hz]	Limit Frequency by IAP-11 [Hz]
15.25	5

Table 3.3: Value of first global mode shape frequency and minimum frequency allowed by the IAP-11.

3.3 Surrogate model adjustment

For the LLSF adjustment, several results were acquired, like the mean error norm or the plot of number of samples versus value of some adjustment coefficients. These results are shown in the following tables and figures.

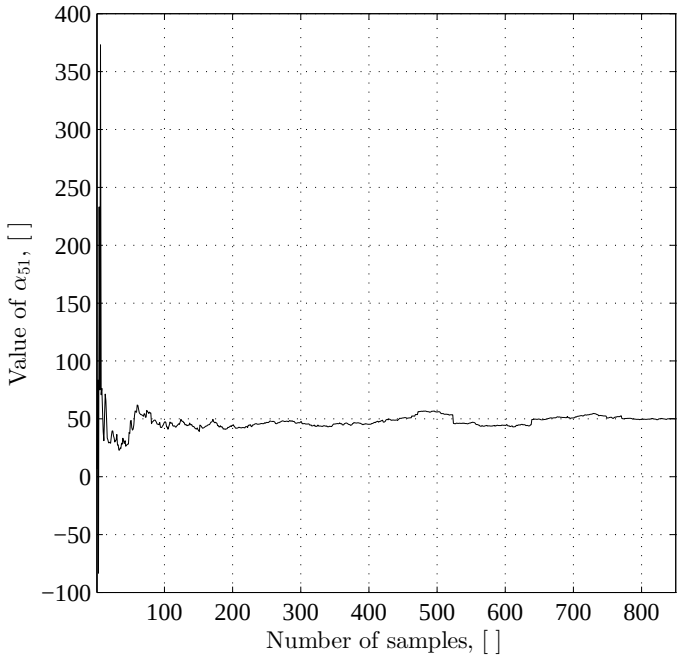


Figure 3.5: The value of the coefficient utilized to adjust displacements converges to an unique value for a set of more than 800 samples.

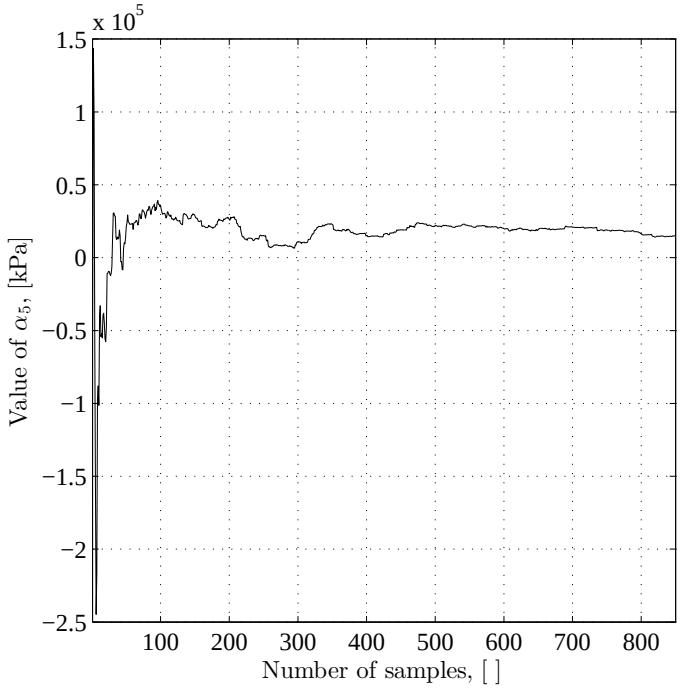


Figure 3.6: The value of the coefficient utilized to adjust stresses converges to an unique value for a set of about 800 samples.

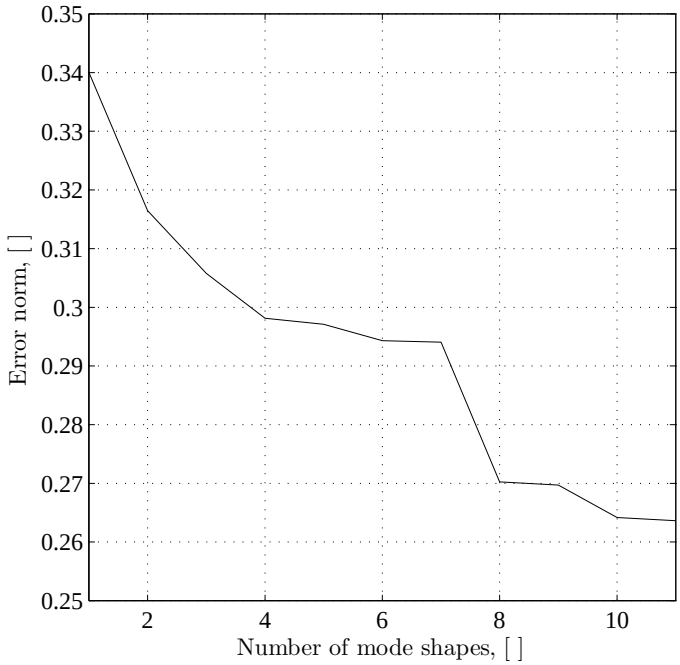


Figure 3.7: The graph shows the gain of overall accuracy in all displacements by adding mode shapes to the adjustment process.

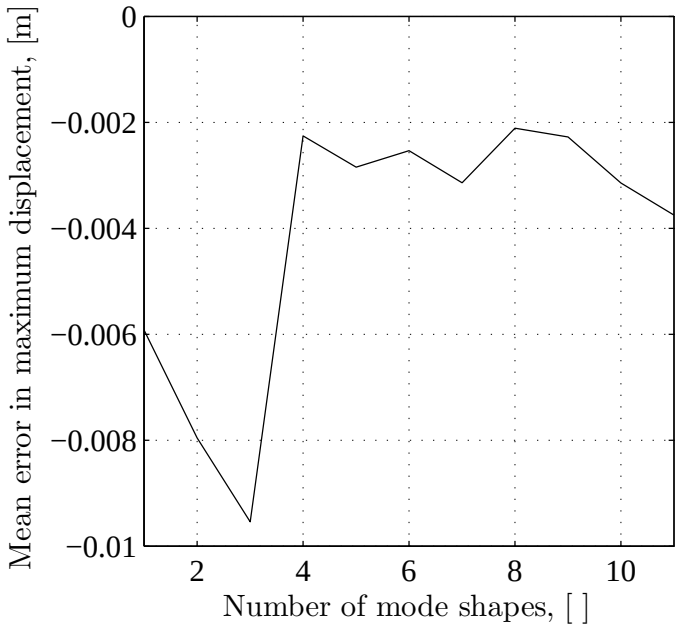


Figure 3.8: Although taking more mode shapes into account improves the overall estimation of displacements, it is shown that using the first 8 mode shapes provides a better estimate for the maximum displacement developed at midspan of the bridge.

Also, the computational effort needed to perform the structural analysis

is reduced from 3 to 5 minutes with ABAQUS to 1 second using the adjusted MA model in MATLAB.

3.4 Optimum bridge

Two optimal bridge configurations were obtained through the genetic algorithm, which are described by the optimization variables values exposed in the following tables.

This bridge configuration corresponds to the one that meets the ultimate limit state requirements but does not check the serviceability limit state specifications, since monitoring equipment to perform this check is ommited.

Parameter	Optimum Value [m]
m_1	0.5
m_2	0.23
m_3	0.37
m_4	1.4
m_5	2.4
m_6	1.2
m_7	0.87
m_8	3.0
l_1	3.8
l_2	1.5
b_1	0.05
b_2	0.05
b_3	0.052

Table 3.4: First set of optimum values for the geometric parameters and sectional properties of the bridge.

Parameter	Optimum Value
Accelerometers	0 Devices
US Equipment	12 Devices, Low Quality
Lifetime	250 cycles

Table 3.5: First set of optimum values for monitoring equipment and lifetime of the bridge.

This bridge configuration corresponds to the one that meets the ultimate limit state requirements and also the serviceability limit state specifications, monitoring the maximum deflection of the structure with accelerometers. It is assumed to be the optimum bridge.

Parameter	Optimum Value [m]
m_1	0.5
m_2	0.25
m_3	0.4
m_4	1.2
m_5	2.0
m_6	1.2
m_7	0.75
m_8	3.0
l_1	3.8
l_2	1.5
b_1	0.08
b_2	0.08
b_3	0.08

Table 3.6: Second set of optimum values for the geometric parameters and sectional properties of the bridge.

Parameter	Optimum Value
Accelerometers	8 Devices
US Equipment	12 Devices, Low Quality
Lifetime	67 cycles

Table 3.7: Second set of optimum values for monitoring equipment and lifetime of the bridge.

The total cost of the optimum bridge is:

Total Cost = $\$117,560.00$

This cost is distributed in the way presented in Figure 3.9.

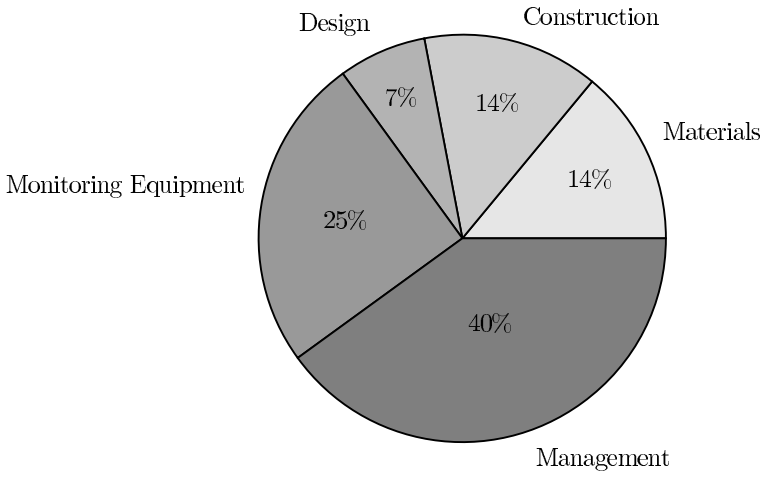
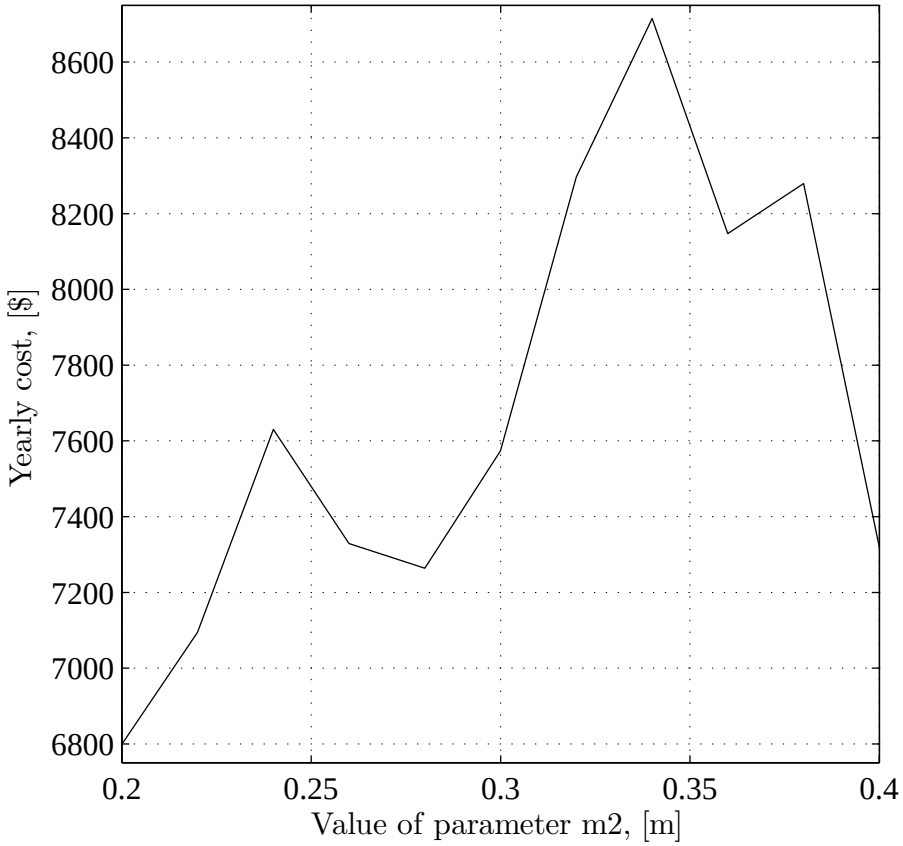
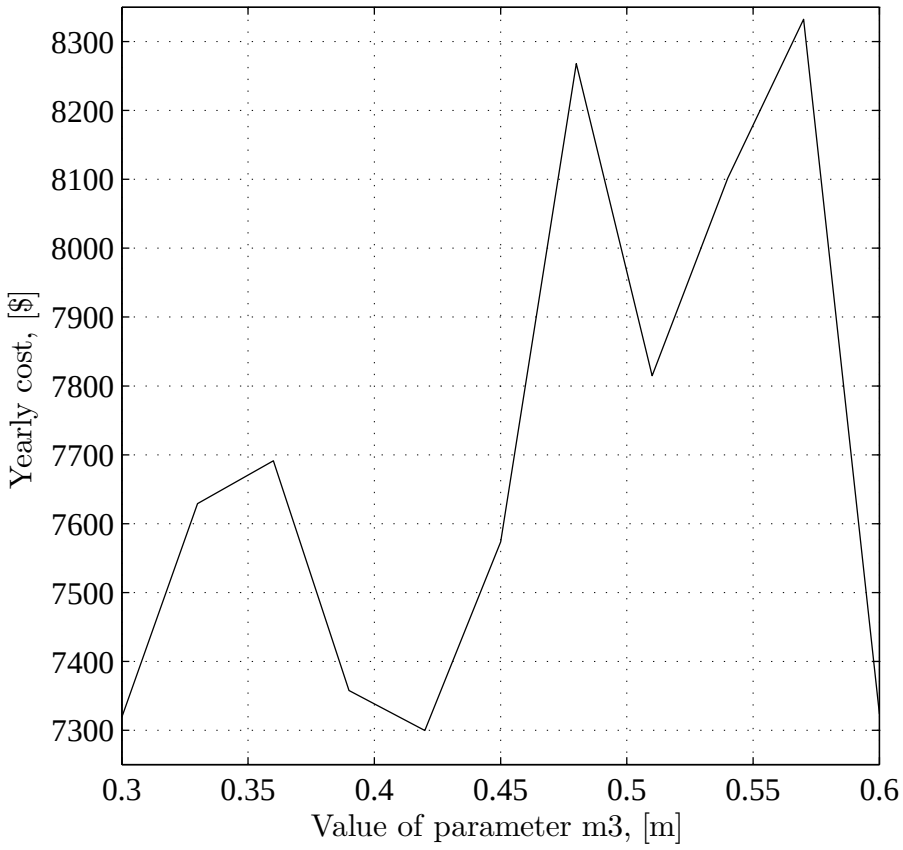


Figure 3.9: It can be noticed that the initial cost of the bridge, which comprises material, construction, design and monitoring equipment cost sums up to 60% of the total cost, while bridge management cost, which is composed by datalogging and management decisions cost, corresponds to the 40% of total cost.

Sensitivity of yearly cost with respect to design variables can be observed in Figure 3.10 (a)-(i).

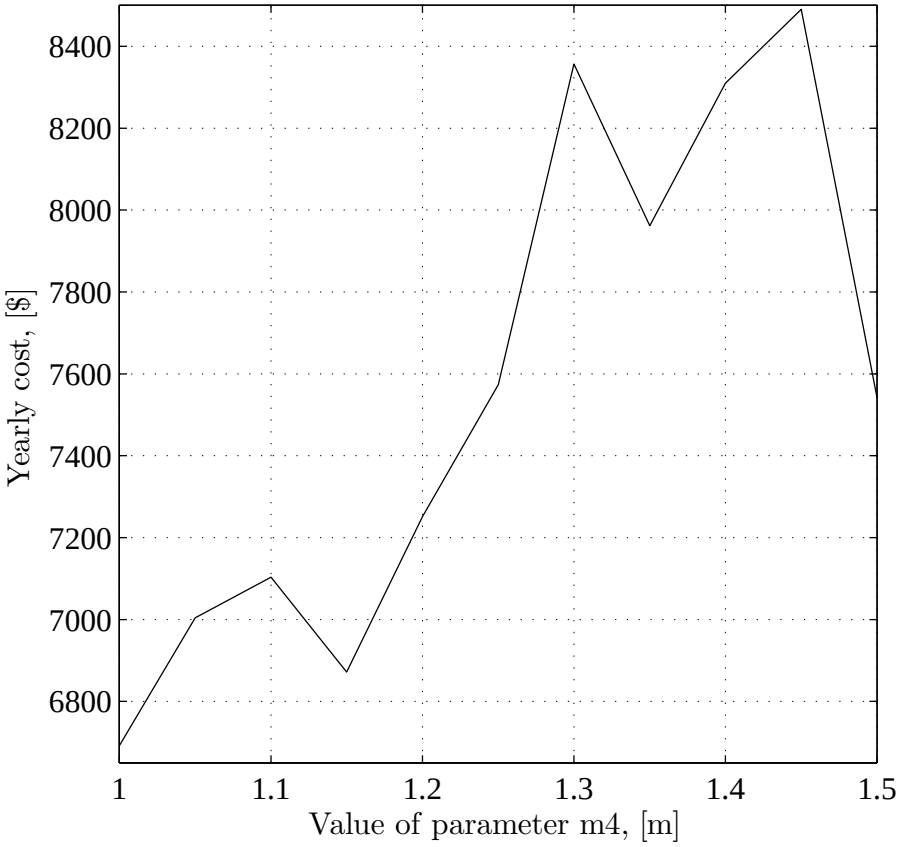


(a) Sensitivity with respect to m_2.

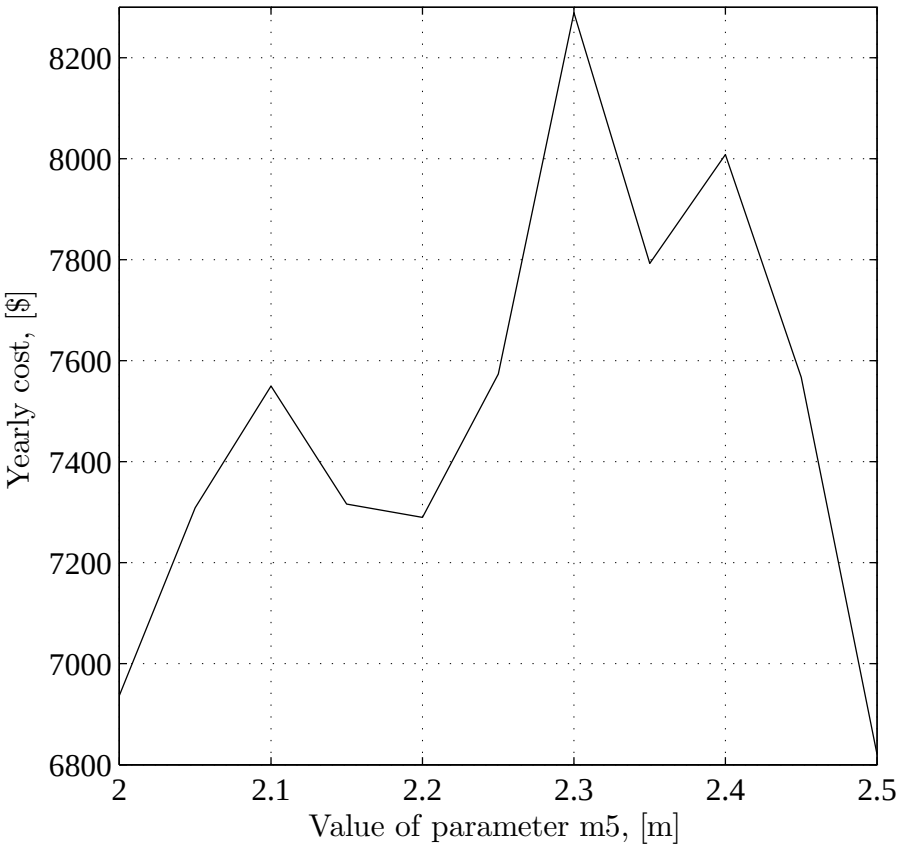


(b) Sensitivity with respect to m_3.

Figure 3.10: Sensitivity analysis of the optimum CFRP Bridge

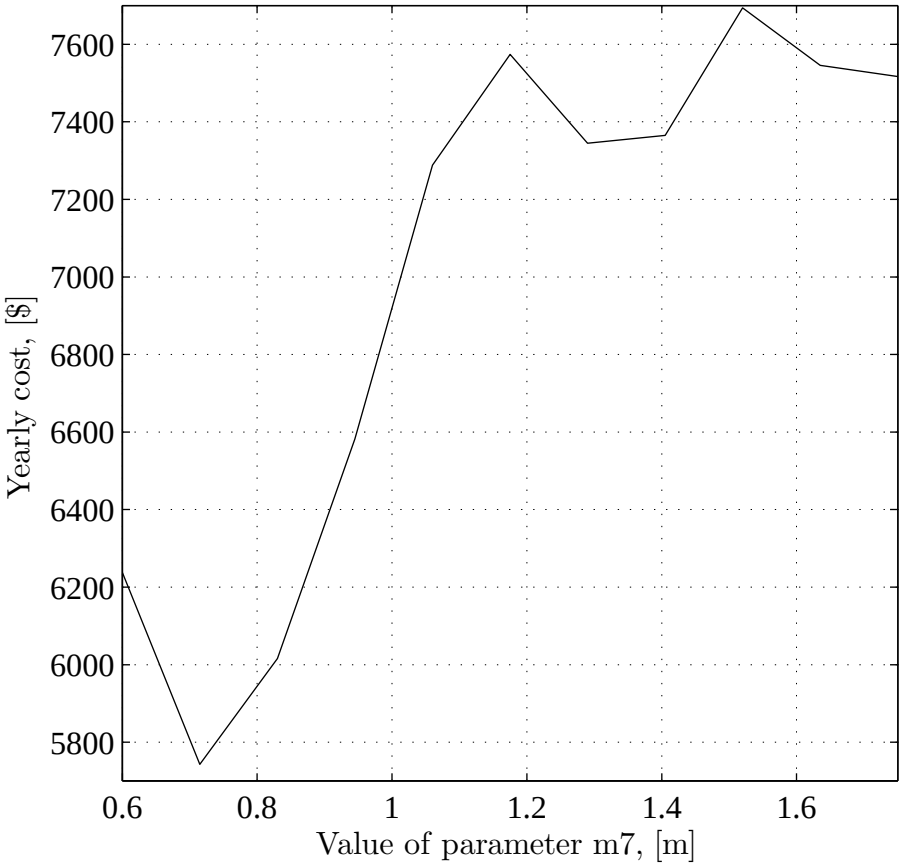


(c) Sensitivity with respect to m_4.

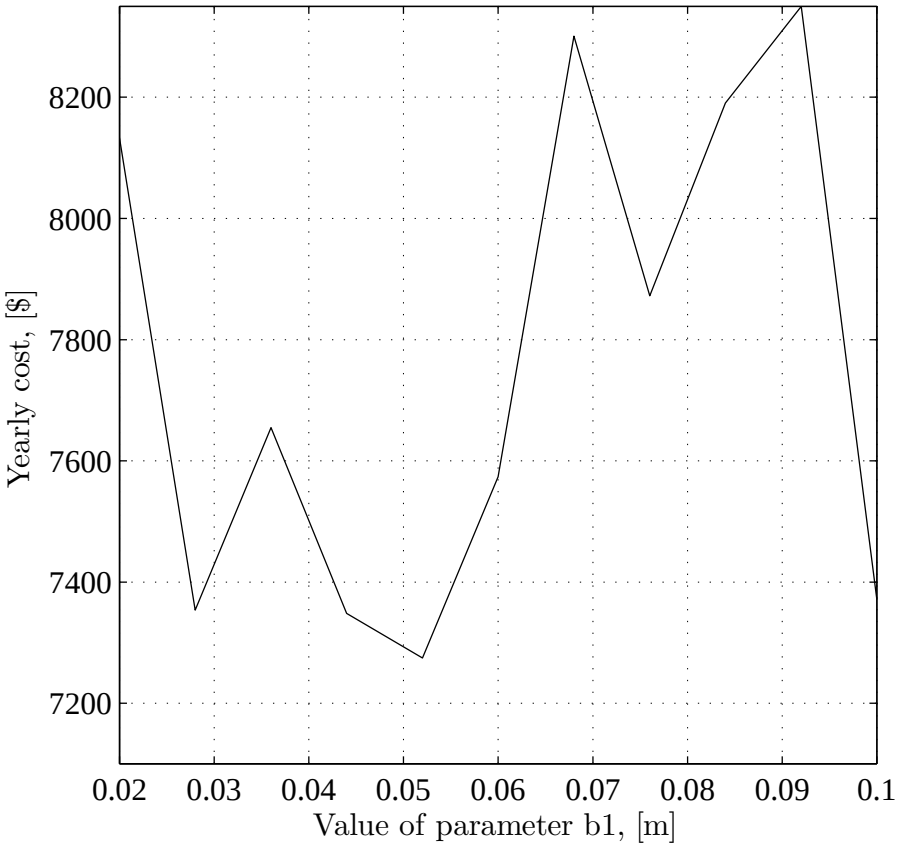


(d) Sensitivity with respect to m_5.

Figure 3.10: Sensitivity analysis of the optimum CFRP Bridge

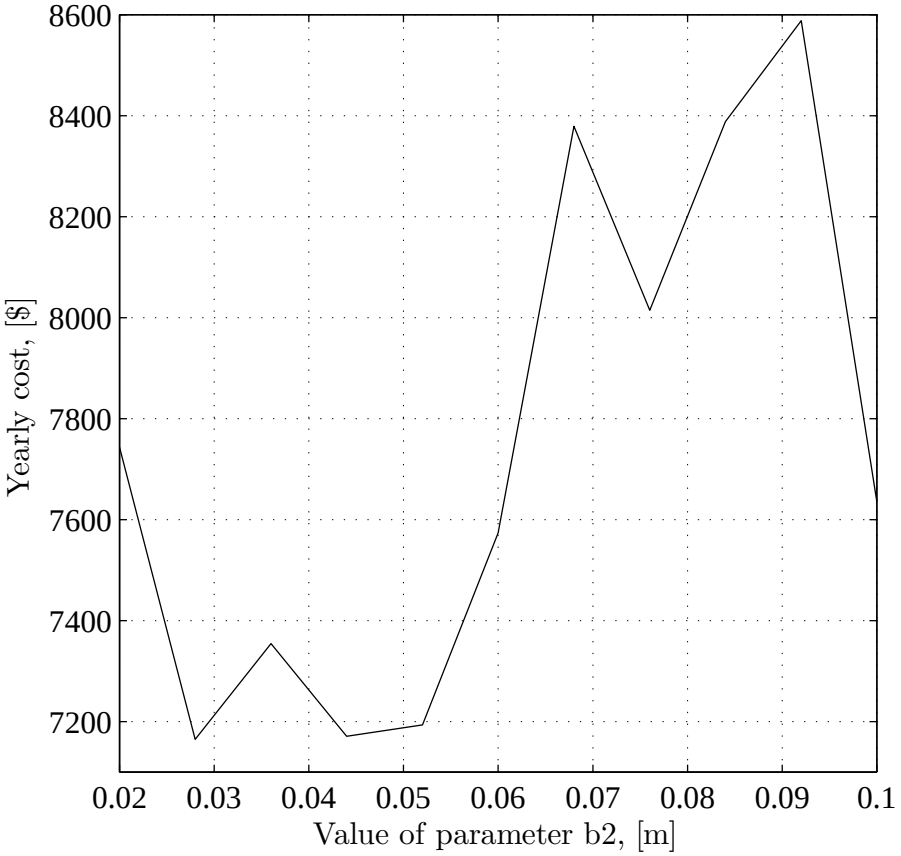


(e) Sensitivity with respect to m_7.

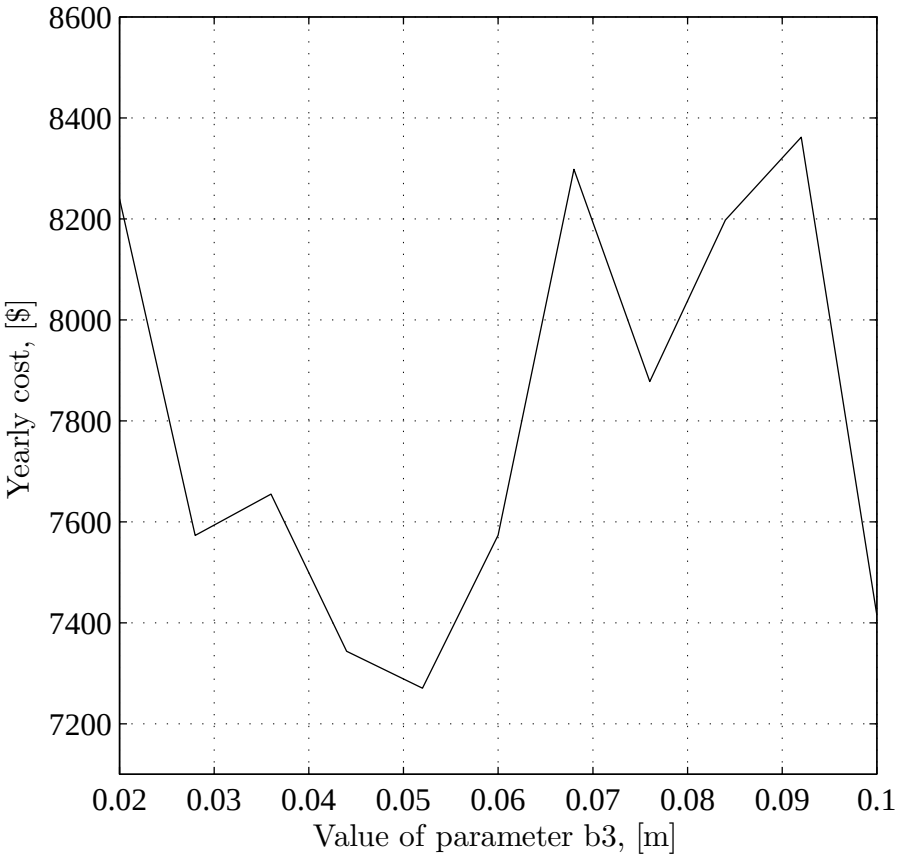


(f) Sensitivity with respect to b_1.

Figure 3.10: Sensitivity analysis of the optimum CFRP Bridge

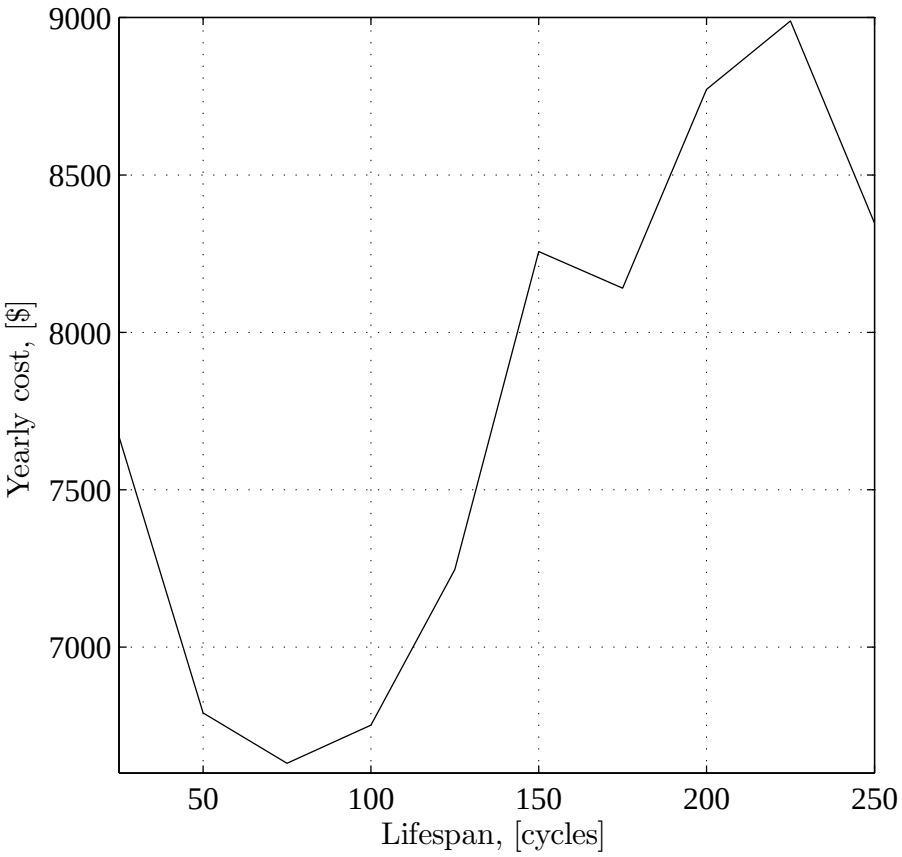


(g) Sensitivity with respect to b_2.



(h) Sensitivity with respect to b_3.

Figure 3.10: Sensitivity analysis of the optimum CFRP Bridge



(i) Sensitivity with respect to lifetime.

Figure 3.10: Sensitivity analysis of the optimum CFRP Bridge

3.5 Optimum bridge design checks

The optimum bridge was checked using the FE model. The geometry of the optimum bridge can be seen in the following figure.

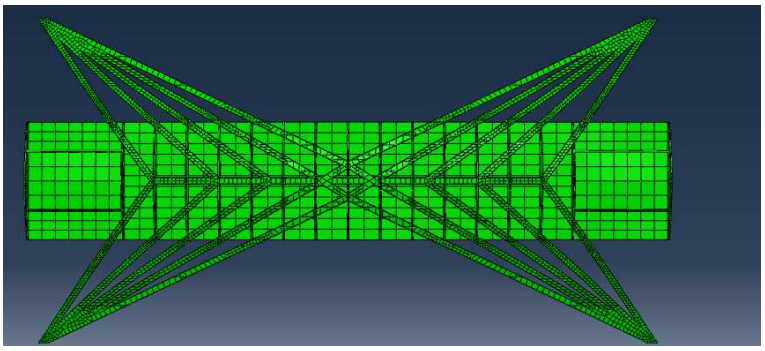


Figure 3.11: Note that the straps that converge to the central square tend to be continuous, making the central square as small as it was permitted in optimization.

The design checks stated by the IAP-11 code for the optimum bridge were also performed given the results that are shown in the following figures.

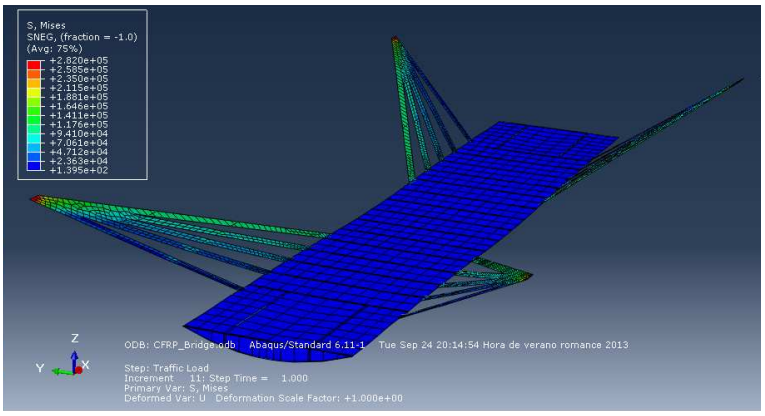


Figure 3.12: The maximum stress is reached at the straps supports, and that it is lower than the yielding stress of CFRP.

Maximum Stress [kPa]	Limit Stress by IAP-11 [kPa]
2.82×10^5	12×10^5

Table 3.8: Value of maximum stress in the bridge and yielding stress of CFRP.

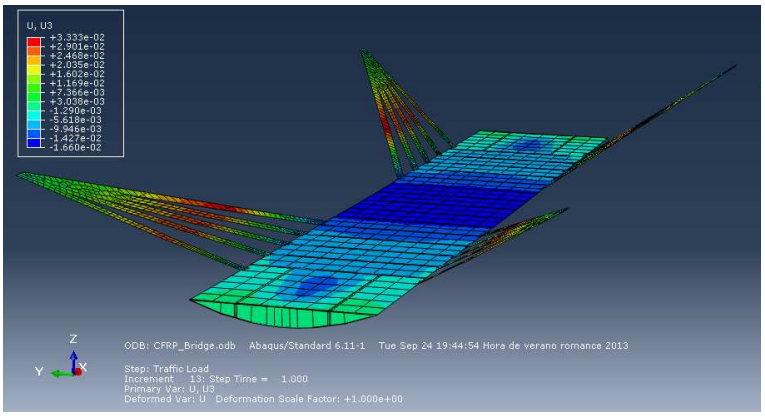


Figure 3.13: Note that the maximum vertical displacement develops at midspan, and that its value is about twice the maximum value permitted by the IAP-11. Nevertheless, a tolerable displacement can be achieved by optimizaing the geometry of the bridge and the sections of the CFRP members.

Maximum Displacement [m]	Limit Displacement by IAP-11 [m]
$1.66 \times 10^{-}2$	$1.67 \times 10^{-}2$

Table 3.9: Value of maximum vertical displacement of the bridge and maximum vertical displacement prescribed by the IAP-11.

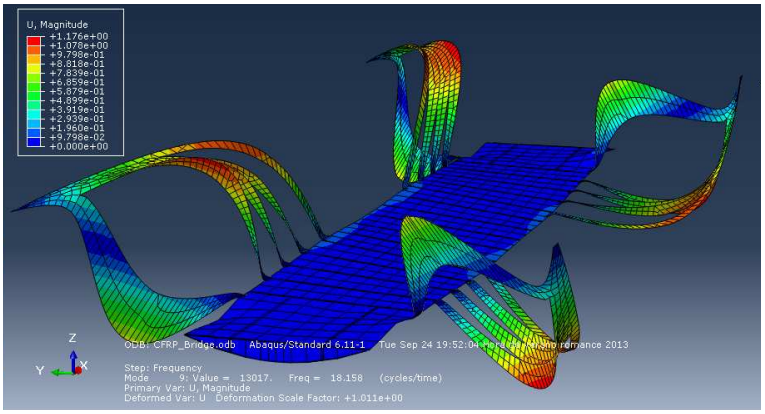


Figure 3.14: The frequency of the first mode shape shown in this figure is 15.2 Hz, which is above the minimum frequency allowed by the IAP-11.

Frequency [Hz]	Limit Frequency by IAP-11 [Hz]
18.16	5

Table 3.10: Value of first global mode shape frequency and minimum frequency allowed by the IAP-11.

Chapter 4

Discussion and Conclusions

4.1 Discussion

For a bridge to pass the serviceability state design check, its maximum displacement should be less than or equal its length in meters divided by 1200, following the IAP-11. After the structural analysis it follows that it is possible for a bridge to meet the requirements stated by the norm. For a bridge to pass the ultimate state design check, its maximum stress should be less than the yielding stress of the material where the stress occurs, following the IAP-11. After the structural analysis it turns out that the maximum stress is roughly 600 MPa, which is well below 1200 MPa, the yielding stress of the CFRP laminate employed in the bridge design. For a bridge to pass the vibration state design, the first global frequency should be greater than 5 Hz. Since the lowest frequency of vibration of the bridge is greater than 15 Hz, the bridge meets the vibrational requirements stated by the IAP-11.

The study of the design checks stated by the IAP-11 bridge norm, using the middle point of the interval for each geometric and sectional variable, leads to think that the type of bridge presented can meet the structural requirements proposed by the norm. The most restrictive scenario for the

CFRP Bridge is the serviceability limit state, because the structure is so flexible that develops large displacements. This is a minor problem, since the maximum deflection is small enough that a countershaft might be designed to overcome the serviceability check. Also, it is to be noted that the bridge is subjected to half the yielding stress of the CFRP laminates used, so a safety factor of 2 is implicit in design. This low tension level implies a slow micro crack damage progression, which results in a longer lifespan and more structural safety for the bridge. Therefore, the bridge proposed in this paper is a safe and cheap structural solution for pedestrian bridges that span 10 m.

The LLSF adjustment via mode shapes for displacement results a very accurate way of reproducing in a MA model the displacements given by a FE model analysis of the bridge. The mode shapes were utilized to reduce the amount of adjustment parameters, and the mean error committed for the maximum displacement of the bridge is around 2 mm, which is small enough to be acceptable for optimization procedures. The maximum error and the standard deviation for the cited DOF are also sufficiently reduced to assume the displacements adjustment as a good approximation. The LLSF adjustment for stresses also permits to approximate very accurately the maximum stress produced in each of the three families of CFRP straps. The maximum mean error is in the order of 10^4 kPa, which is small compared to the order of 10^5 kPa of maximum stress produced in the bridge. Also the standard deviation and the maximum error are in the range of the mean error, so the adjustment is thought to accurately reproduce the stresses given by the FE model through the analysis of the MA model of the bridge.

Convergence of adjustment coefficients is obtained for a number of samples less than the 1000 FE computations. Then, it turns out that the number of samples is large enough for adjustment considerations. Also,

the error vector norm of displacements might be reduced by adding mode shapes to the adjustment algorithm. This reduction tends to be lower as more mode shapes are taking into account, and it goes up to almost 0 from the 8^{th} to the 11^{th} mode shapes, so a number of 8 mode shapes is considered to provide a good adjustment for all displacements through a LLSF. Also, the mean error of the maximum displacement of the bridge is minimized adjusting with 8 mode shapes, and since this displacement will be used as a failure criteria, it can be inferred that the better possible adjustment for displacements includes the first 8 mode shapes of the bridge.

The time reduction of complete optimization of the structure is of approximately 7 days for the adjusted surrogate model compared to about 1050 days for the FE model, which justifies the effort in developing a surrogate model to determine the optimum bridge.

The damage evolution algorithm is capable of reproducing the deterioration of stiffness of a CFRP laminate subjected to cyclic loads. However, a more precise experimental investigation would be needed because the Paris Law parameters used in the algorithm were adjusted with data from CFRP coupons subjected to a higher stress level than that suffered by the straps of the bridge. Therefore, a Paris Law adjusted with new experimental data applying stresses similar to those produced in the structural members of the bridge will improve the accuracy of the damage evolution algorithm.

The genetic optimization algorithm is an accurate way to find the optimum bridge because the lifetime cost of the structure that is to be minimized has different local minima. Genetic optimization algorithms are capable of finding the global minimum regardless of the number of local minima in the optimization interval.

The optimization results in the two optimum bridges that were supposed beforehand. One of them is the solution without monitoring equipment to

measure deflections, which will not be taken as an acceptable solution since it does not meet the serviceability limit state requirements stated by the IAP-11. The other bridge meets both ultimate and servicability checks, and monitoring equipment to detect both types of failure are considered. This bridge configuration will be taken as appropriate for design purposes, and will be assumed as the optimum bridge.

The geometrical and sectional paramenters that configures the optimum bridge take values that are close to those expected before the optimization. Since the critical failure mode is the serviceability limit state, the straps width of the optimum bridge are such that the maximum displacement is always below the limit stated by the IAP-11, for the whole life of the bridge. Then, the result of the optimization process matches the supposed geometry and straps sections of the optimum bridge, which fails in serviceability at the end of its lifetime. Both accelerometers and ultrasonic devices will be utilized to monitor the optimum bridge, since they permit to determine the most appropriate bridge management decision at every time by measure displacements and stresses respectively, reducing this way the operational cost of the structure. However, it is surprising that the optimum lifetime of the bridge is less than the upper limit set for the optimization procedure. This fact responds to the mechanical behavior of CFRP, which cracks with each load cycle up to a maximum crack density, when fail occurs at delamination process initiates. Thick CFRP laminates are needed to avoid delamination at large numbers of load cycles, which dramatically rises the maintenance cost of the bridge, and also the initial cost. Then, if lifetime is long it turns out that the maintenance cost strongly increases, making more convinient to dismantle the bridge than to keep wasting money to repair it. Finally, it can be inferred that the optimization has been performed correctly, and that the optimization algorithm proposed in this work is capable

of determining the optimum configuration of a structural system, including monitoring system and lifespan, in terms of cost.

4.2 Conclusions

The following concluding remarks can be extracted from the present work:

- CFRP is a material with a great potential in civil engineering due to its outstanding mechanical properties. The usage of this material will result in innovative structural design conceptions.

- The proposed all-composite selfstressed bridge provides a safe, cheap and aesthetic design for future short-span infrastructures such as pedestrian or highway bridges.

- The mode shapes-based surrogate model is a physics-based surrogate model that allows to analize structures using a computationally efficient matrix bar model. Since the model is based on structural analysis theory, it provides a meaningful approximation to structural behavior, unlike other mathematics-based surrogate model methods.

- The RBDO method procedure stated in the article is capable of determining the optimal configuration of a structure, including monitoring system, and even taking into account damage progression and failure detection. More complex cost functions can be suggested to approximate the optimization problem to different real-life conditions.

- Safety factors can be reduced in design if structural performance is known at every time, and monitoring provides real-time data about structural performance, which can be used to always adopt the better management decision. This way, Monitoring is the key for future structural optimization.

- Lifetimes imposed by structural design specification need to be changed, since it turns out that every structural system has its own optimum lifespan in terms of cost. Then, a revision of building codes is mandatory to improve design of structures.

- Application of currents developments in RBDO suppose an excellent opportunity for civil engineering to adopt more accurate design methodologies that will reduce conservative safety factors which result in more expensive structures. Future research in RBDO will provide the knowledge needed to a revolution in structural design in civil engineering.

Appendix A

Parametric FE Model of the Bridge Codes

```
%Created by Julio Rodríguez Sáchez

tic
clc, clear
time=tic;

%% Number of Samples
n=1;

%rnd_range=[ m1  m2  m3  m4   m5   m6   m7   m8  l1  l2    b1
       b2   b3 ]
 min_range=[ 0.5 0.2 0.3  1   2    1.2  0.6  3  3.8  1.5  0.03
     0.03 0.03];
 max_range=[ 0.5 0.4 0.6  1.5 2.5  1.2  1.75 3  3.8  1.5  0.10
     0.10 0.10];

min_range(1,1:10)=min_range(1,1:10)*66/10;
max_range(1,1:10)=max_range(1,1:10)*66/10;
```

```
16
17 for o=1:n
18
19 timel=tic;
20 %% Avoiding Computation Problems
21  displacements=zeros(32,3);
22  rotations=zeros(32,3);
23  stresses=zeros(32,1);
24  save -ascii 'displacements.txt' displacements;
25  save -ascii 'rotations.txt' rotations;
26  save -ascii 'stressesc.txt' stresses;
27  save -ascii 'stressese.txt' stresses;
28  save -ascii 'stressesi.txt' stresses;
29  save -ascii 'displacements0.txt' displacements;
30  save -ascii 'rotations0.txt' rotations;
31
32 %% Variable Range Definition
33  lamcal_cfrp;
34  lamcal_gfrp;
35  geomcal;
36  parameters(:,o)=[m1 m2 m3 m4 m5 m6 m7 m8 l1 l2 b1 b2 b3 b4 b5
       b6 ...
37       h1 h2 h3 h4 h5 YM1 YM2 YM3 YM4 YM5]';
38  parametersm(:,o)=parameters(:,o);
39
40 %% Section Variables
41 % [Central Axis; Exterior Bridles; Interior Bridles]
42 section=[b1; b2; b3; b4; b5; b6];
43 depth=[h1; h2; h3; h4; h5];
44
45 %% Material Properties
46 mprop=[YM1 SM1; YM2 SM2; YM3 SM3; YM4 SM4; YM5 SM5];
47
48 %% Mesh Parameters
```

```
ms=[0 0.25];
mn=[1 2 2];

%% FE Model
timefe=tic;
Abaqus;
fprintf(' FEM Analysis Completed  ======> ');
toc(timefe)
%% Matrix Analysis Model
% timema=tic;
% ma;
% fprintf(' MBM Analysis Completed  ======> ');
% toc(timema)

%% Computation of Surrogate Element Matrix
d1m(:,o)=d1(:,1);
% d2m(:,o)=d2(:,1);
% dmatrix(:,o)=d1./d2;
s1m(:,o)=s1(:,1);
% s2m(:,o)=s2(:,1);
% smatrix(:,o)=s1./s2;
% sfv(:,o)=k*d1;

%% End
 delete_files;
fprintf(' Sample Completed  ======> %d\n', o);
% toc(timel)
save -ascii 'd1.txt' d1m;
% save -ascii 'd2.txt' d2m;
% save -ascii 'dmatrix.txt' dmatrix;
save -ascii 's1.txt' s1m;
% save -ascii 's2.txt' s2m;
% save -ascii 'smatrix.txt' smatrix;
save -ascii 'parameters.txt' parametersm;
```

```
% save -ascii 'sfv.txt' sfv;
% fprintf(' Maximum Displacement  ======> %d\n', d1(687,1));
% fprintf(' Maximum Stress  ======> %d\n', max(s));
fprintf(' Time  ======> %d\n', toc(time));

end
toc
```

```
%Created by Julio Rodríguez Sáchez

%% Laminate Calculator

%%
 % First it is needed to determine the plies that makes the
    laminate.
 % Here we will consider just 4 families of plies (plies with
    different
 % fiber orientations)
 % Set the characteristics of every ply in the command lines
    below
 % Units are in m, GPa, degrees
 % Call the calculator this way: "sol=lamcalsym([m n p q r+ r-
    h b e d])
 % where m,n,p and q are the fiber orientations of plies 1,2,3
    and 4,
 % in the workspace directions (m,n p and q are written in
    degrees),r+ and
 % r- are the stacking sequence index above and below the
    midplane,
 % respectively (r+ and r- must be a positive number or zero),
    h is the total
 % thickness of the laminate, b is the with of the section and
    e is the
```

```
17  % thickness of the flanges the section might have
18  % Variable d is stated for decision purposes between 3-ply
        laminates or
19  % other else. If a 3-ply laminate has to be calculated d has
        to be equal
20  % to 1. For other cases (homogeneous, 2-ply or 4-ply laminates
        ) d has to
21  % be not equal to one
22  % For calculation purposes this code works with normalized
        axial forces
23  % and bending moments, wich makes everything easier to compute
         (stacking
24  % sequence-flexural stress interaction can be messy, so we
        will just work
25  % with normalized stresses and thus we can keep an easier
        calculation
26  % going on)
27
28
29  %%%PLY DEFINITION (STANFORD)
30
31  Ex=41.7e9;                      %[Pa] On-axis (in-plane) (Young's
        modulus)
32  Ey=13e9;                    %[Pa] On-axis (in-plane) (Transverse
        Young's modulus
33  Es=3.4e9;                    %[Pa] Shear modulus (in-plane)
34  nux=0.300;                    %[no units] in plane Poisson's
        ratio
35  nuy=nux*(Ey/Ex);              %[no units] in- plane plane Poisson
        's ratio
36  nuyz=0.42;                    %[no units] out of plane Poisson's
        ratio
37  Gyz=Ey/(2*(1+nuyz));          %[Pa] out of plane shear modulus
38  plythick=0.203e-3;           %[m] Ply Thickness
```

```
39 G_d0=9.07e13;                %[Pa/m] this is a purely
       experimental parameter. See Ogihara1995, Table 3.
40 G_d0=2.07e11;                %[Pa/m] this is one trial by Manuel
41 G_c=150.609;                 %[J/m^-2] intralaminar critical
       energy release rate  150.609
42
43
44 %%%LAMINATE DEFINITION (GUDMUNDSON)
45 lam_type='x-ply';
46 StackSeq=[0 45 -45 90 90 -45 45 0];          %total laminate.
       From top to bottom
47 Sub_s_StackSeq=[0];                     %sublaminate 1 (Remember:
        only one of the 0° Ply stack)
48 Sub_90_StackSeq=[45 -45 90 90 -45 45];          %sublaminate 2
49 Laminate.StackSeq=StackSeq;
50 h=numel(Laminate.StackSeq)*plythick/2;        %[m] Laminate half
       -thickness
51 B=0.001;
52
53 path(path,'lib')
54 stiffness;
55
56 b4=0.05;h4=size(StackSeq,2)*plythick;YM4=E_0*10^(-3);SM4=Gyz
       *10^(-6);
57
58 %%%LAMINATE DEFINITION (GUDMUNDSON)
59 lam_type='x-ply';
60 StackSeq=[0 0 45 45 -45 -45 90 90 90 90 -45 -45 45 45 0 0];
              %total laminate. From top to bottom
61 Sub_s_StackSeq=[0];                     %sublaminate 1 (Remember:
        only one of the 0° Ply stack)
62 Sub_90_StackSeq=[45 -45 90 90 -45 45];          %sublaminate 2
63 Laminate.StackSeq=StackSeq;
```

```
64 h=numel(Laminate.StackSeq)*plythick/2;        %[m] Laminate half
       -thickness
65 B=0.001;
66
67 path(path,'lib')
68 stiffness;
69
70 b5=2;h5=size(StackSeq,2)*plythick;YM5=E_0*10^(-3);SM5=Gyz
       *10^(-6);
71 b6=2;
72 b7=0.05;
```

```
1  %Created by Julio Rodríguez Sáchez
2
3  %% Laminate Calculator
4
5  %%
6   % First it is needed to determine the plies that makes the
        laminate.
7   % Here we will consider just 4 families of plies (plies with
        different
8   % fiber orientations)
9   % Set the characteristics of every ply in the command lines
        below
10  % Units are in m, GPa, degrees
11  % Call the calculator this way: "sol=lamcalsym([m n p q r+ r-
        h b e d])
12  % where m,n,p and q are the fiber orientations of plies 1,2,3
        and 4,
13  % in the workspace directions (m,n p and q are written in
        degrees),r+ and
14  % r- are the stacking sequence index above and below the
        midplane,
```

```
15 % respectively (r+ and r- must be a positive number or zero), h is the total
16 % thickness of the laminate, b is the with of the section and e is the
17 % thickness of the flanges the section might have
18 % Variable d is stated for decision purposes between 3-ply laminates or
19 % other else. If a 3-ply laminate has to be calculated d has to be equal
20 % to 1. For other cases (homogeneous, 2-ply or 4-ply laminates) d has to
21 % be not equal to one
22 % For calculation purposes this code works with normalized axial forces
23 % and bending moments, wich makes everything easier to compute (stacking
24 % sequence-flexural stress interaction can be messy, so we will just work
25 % with normalized stresses and thus we can keep an easier calculation
26 % going on)
27
28
29 for k=1:esamples
30
31 %%%PLY DEFINITION (STANFORD)
32
33    Ex=normrnd(127.553e9,127.553e8);              %[Pa] On-axis (in-plane) (Young's modulus)
34    Ey=normrnd(8.411e9,8.411e8);                  %[Pa] On-axis (in-plane) (Transverse Young's modulus
35    Es=normrnd(6.205e9,6.205e8);                  %[Pa] Shear modulus (in-plane
```

```
36      nux=normrnd(0.309,0.0309);                  %[no units] in
           plane Poisson's ratio
37      nuy=nux*(Ey/Ex);            %[no units] in- plane plane
           Poisson's ratio
38      nuyz=0.49;                  %[no units] out of plane
           Poisson's ratio
39      Gyz=Ey/(2*(1+nuyz));        %[Pa] out of plane shear
           modulus
40      plythick=0.1524e-3;         %[m] Ply Thickness
41      G_d0=9.07e13;               %[Pa/m] this is a purely
           experimental parameter. See Ogihara1995, Table 3.
42      G_d0=2.07e11;               %[Pa/m] this is one trial by
           Manuel
43      G_c=150.609;                %[J/m^-2] intralaminar critical
            energy release rate  150.609
44
45      %%%LAMINATE DEFINITION (GUDMUNDSON)
46      lam_type='x-ply';
47      StackSeq=[0 0 90 90 90 90 90 90 90 90 0 0];  %total
           laminate. From top to bottom
48      Sub_s_StackSeq=[0 0];            %sublaminate 1 (Remember:
           only one of the 0° Ply stack)
49      Sub_90_StackSeq=[90 90 90 90 90 90 90 90];   %sublaminate 2
50      Laminate.StackSeq=StackSeq;
51      h=numel(Laminate.StackSeq)*plythick/2;       %[m] Laminate
           half-thickness
52      B=0.07;                                 %[m] Laminate half-
           width (if variable section, choose the critical one)
53
54      cd('lib')
55      stiffness;
56
57      cd('../')
58
```

```
59      YM1(k)=E_0*10^(-3);YM2(k)=E_0*10^(-3);YM3(k)=E_0*10^(-3);
60      SM1(k)=Gyz*10^(-3);SM2(k)=Gyz*10^(-3);SM3(k)=Gyz*10^(-3);
61
62  end
63
64      b1=min_range(1,11)+(max_range(1,11)-min_range(1,11))*rand;
65      b2=min_range(1,12)+(max_range(1,12)-min_range(1,12))*rand;
66      b3=min_range(1,13)+(max_range(1,13)-min_range(1,13))*rand;
67      h1=plythick*size(StackSeq,2);h2=plythick*size(StackSeq,2);
            h3=plythick*size(StackSeq,2);
```

```
1  function [a_art,eng_cnst]=LamTheory(PlyData,StackSeq)
2
3  %This function calculates the normalized A and a stiffness
       matrices of a
4  %symmetric or unsymmetric laminate. Note that if unsymmetric
       laminate, the
5  %stiffness and engineering constants are valid for in-plane
       loads only,
6  %wich is the typical case in fatigue.
7
8
9  % %prueba
10 % clear all
11 %  StackSeq=[90 90 90 90]; %total laminate. From top to bottom
12 %  nplies=numel(StackSeq);  %Total number of plies
13 % families=[0,90]; %N° of fiber angle orientation
14 % norientfamil=numel(families); %N° of orientation angles
15 % %
16 % Ex=127.553e9;                %[Pa] On-axis (in-plane) (Young's
       modulus)
17 % Ey=8.411e9;                  %[Pa] On-axis (in-plane) (
       Transverse Young's modulus
```

```
% Es=6.205e9;                    %[Pa] Shear modulus (in-plane)
% nux=0.309;                     %[no units] in plane Poisson's ratio
% nuy=nux*(Ey/Ex);               %[no units] in- plane plane Poisson's ratio
% nuyz=0.49;                     %[no units] out of plane Poisson's ratio
% Gyz=Ey/(2*(1+nuyz));           %[Pa] out of plane shear modulus
% plythick=0.135e-3;             %[m] Ply Thickness
% G_d0=9.07e13;                  %[Pa/m] this a purely experimental parameter. See Ogihara1995, Table 3.
%
% PlyData=[Ex,Ey,Es,nux,nuy,nuyz,Gyz,plythick];   % characterizing vector


Ex=PlyData(1);
Ey=PlyData(2);
Es=PlyData(3);
nux=PlyData(4);
nuy=PlyData(5);
plythick=PlyData(8);

nplies=numel(StackSeq);

Qxx=Ex/(1-nux*nuy);
Qyy=Ey/(1-nux*nuy);
Qxy=nuy*Qxx;
Qyx=nux*Qyy;
Qss=Es;

Qonx=[Qxx, Qxy, 0;...
   Qyx Qyy 0;...
```

```
47      0 0 Qss];
48
49
50
51
52  %Acc_Soffx={};          %Initialize a storage matrix with Soffx
        for each ply
53  %aux_effstiff={};        %Initialize an auxiliary storage
        matrix
54  %A={};                  %Initialize  "A" storage matrix
55  %acc_effstiff=zeros(3,3);       %Initializa a auxiliary storage
        matrix
56  %betamatrix={};
57  Acc_Qoffx=zeros(3,3);
58
59  for i=1:nplies
60      theta=StackSeq(i)*(pi/180);     %fiber angle: degree*(degree
            ->rad)= [rad]
61      m=cos(theta);
62      n=sin(theta);
63
64
65              tranfmatrix=[m^4 n^4 2*(m^2)*(n^2) 4*(m^2)*(n^2);
                    ...  %Transformation matrix
66                  n^4 m^4 2*(m^2)*(n^2) 4*(m^2)*(n^2);...
67                  (m^2)*(n^2) (m^2)*(n^2) (m^4)+(n^4) -4*(m^2)*(
                        n^2);...
68                  (m^2)*(n^2) (m^2)*(n^2) -2*(m^2)*(n^2) ((m^2)
                        -(n^2))^2;...
69                  (m^3)*n -m*(n^3) m*(n^3)-(m^3)*n 2*(m*(n^3)-(m
                        ^3)*n);...
70                  m*(n^3) -(m^3)*n (m^3)*n-m*(n^3) 2*((m^3)*n-m
                        *(n^3))];
71
```

```
72              Qoffx=tranfmatrix*[Qonx(1,1);Qonx(2,2);Qonx(2,1);
                    Qonx(3,3)];
73
74
75
76      %Soffx=tranfmatrix*[Sonx(1,1) Sonx(2,2) Sonx(2,1) Sonx(3,3)
            ]';
77      %Soffx=[Soffx(1) Soffx(3) Soffx(5);...
78          %Soffx(3) Soffx(2) Soffx(6);...
79          %Soffx(5) Soffx(6) Soffx(4)];
80
81          Qoffx=[Qoffx(1) Qoffx(3) Qoffx(5);...
82          Qoffx(3) Qoffx(2) Qoffx(6);...
83          Qoffx(5) Qoffx(6) Qoffx(4)];
84
85      Acc_Qoffx=Acc_Qoffx+Qoffx;
86      %Acc_Qoffx{i,1}=Qoffx;
87  end
88
89  %normalized laminate stiffness matrix
90  A_norm=Acc_Qoffx./nplies;
91
92  %a_norm=inv(A_norm);
93
94
95  dtA=det(A_norm);
96  %normalized laminate compliance matrix
97  a_art.a_11=(A_norm(2,2)*A_norm(3,3)-(A_norm(2,3))^2)/dtA;
98  a_art.a_22=(A_norm(1,1)*A_norm(3,3)-(A_norm(1,3))^2)/dtA;
99  a_art.a_66=(A_norm(1,1)*A_norm(2,2)-(A_norm(1,2))^2)/dtA;
100 a_art.a_12=(-A_norm(1,2)*A_norm(3,3)+A_norm(1,3)*A_norm(2,3))/
        dtA;
101 a_art.a_16=(A_norm(1,2)*A_norm(2,3)-A_norm(2,2)*A_norm(1,3))/
        dtA;
```

```
a_art.a_26=(A_norm(1,2)*A_norm(1,3)-A_norm(1,1)*A_norm(2,3))/
    dtA;

%a_art=[a_11,a_12,a_16;a_12,a_22,a_26;a_16,a_26,a_66];

%Laminate engineering constants

eng_cnst.E_1=1/a_art.a_11;
eng_cnst.E_2=1/a_art.a_22;
eng_cnst.E_6=1/a_art.a_66;
eng_cnst.G=eng_cnst.E_6;
eng_cnst.nu_12=-a_art.a_12/a_art.a_22;
eng_cnst.nu_21=-a_art.a_12/a_art.a_11;
eng_cnst.nu_61=a_art.a_16/a_art.a_11;
eng_cnst.nu_16=a_art.a_16/a_art.a_66;
eng_cnst.nu_62=a_art.a_26/a_art.a_22;
eng_cnst.nu_26=a_art.a_26/a_art.a_66;

%eng_cnst=[E_1,E_2,E_6,G,nu_12,nu_21,nu_61,nu_16,nu_62,nu_26];


%end
```

```
%Created by Julio Rodríguez Sáchez

m1=min_range(1,1)+(max_range(1,1)-min_range(1,1));          %
    deck thickness
m2=min_range(1,2)+(max_range(1,2)-min_range(1,2))*rand;     %
    distance between directrix line and outline joints (type 2.4
    )
m3=min_range(1,3)+(max_range(1,3)-min_range(1,3))*rand;     %
    distance between center and first nearest joint (type 1.6)
```

```
6  m4=min_range(1,4)+(max_range(1,4)-min_range(1,4))*rand;      %
       distance between center and second joint (type 1.4)
7  m5=min_range(1,5)+(max_range(1,5)-min_range(1,5))*rand;      %
       distance between center and third joint (type 1.3)
8  m6=min_range(1,6)+(max_range(1,6)-min_range(1,6));           %
       distance between extreme line of deck and pylons
9  m7=min_range(1,7)+(max_range(1,7)-min_range(1,7))*rand;      %
       pylons height
10 m8=min_range(1,8)+(max_range(1,8)-min_range(1,8));           %
       distance between center and fourth joint (type 1.4)
11 l1=min_range(1,9)+(max_range(1,9)-min_range(1,9));           %
       middle of total lenght of deck
12 l2=min_range(1,10)+(max_range(1,10)-min_range(1,10));        %
       middle of total widht of deck
```

```
1  %Created by Julio Rodríguez Sáchez
2
3  shell;
4  l_coor=10/66*l_coor;
5  % This is the main body of bridge frame model
6  % Every variable regarding space form of the bridge will be
       defined and
7  % declared here
8
9  %
```

```
10
11 xcoord=l_coor(:,1);
12 ycoord=l_coor(:,2);
13 zcoord=l_coor(:,3);
14 inipoint=l_conc(:,1);
15 endpoint=l_conc(:,2);
```

```
16 h=section(:,1);
17 b=depth(:,1);
18 e=mprop(:,1);
19 g=mprop(:,2);
20
21 save -ascii 'xcoord.txt' xcoord; save -ascii 'ycoord.txt'
      ycoord;
22 save -ascii 'zcoord.txt' zcoord;
23 save -ascii 'inipoint.txt' inipoint; save -ascii 'endpoint.txt'
       endpoint;
24 save -ascii 'h.txt' h; save -ascii 'b.txt' b; save -ascii '
      e.txt' e;
25 save -ascii 'g.txt' g; save -ascii 'ms.txt' ms; save -ascii '
      mn.txt' mn
26
27 % PATH = getenv('PATH');
28 % setenv('PATH', [PATH ':/home/end/abaqus/Commands']);
29
30 % mo='noGUI';
31 mo='script';
32
33 % Make part(run Abaqus)
34 % unix(['abaqus cae ',mo,'=CFRP_Bridge.py']);   %Unix system
35 system(['abaqus cae ',mo,'=CFRP_Bridge.py']); %Windows system?
36
37 d=dlmread('displacements.txt');
38 r=dlmread('rotations.txt');
39 d0=dlmread('displacements0.txt');
40 r0=dlmread('rotations0.txt');
41 sc=dlmread('stressesc.txt');
42 se=dlmread('stressese.txt');
43 si=dlmread('stressesi.txt');
44
45  % Rearranging displacements
```

```
v=[32 22 26 25 24 23 17 21 20 19 18 12 16 15 14 13 7 11 10 9 8
    2 6 5 4 ...
   3 31 30 29 28 27 1];
for i=1:32
    disp1(v(1,i),1:3)=d(i,1:3);
    disp0(v(1,i),1:3)=d0(i,1:3);
    disp1(v(1,i),4:6)=r(i,1:3);
    disp0(v(1,i),4:6)=r0(i,1:3);
end
disp0(27:52,1)=disp0(1:26,1);disp0(27:52,2)=-disp0(1:26,2);
disp0(27:52,3)=disp0(1:26,3);disp0(27:52,4)=-disp0(1:26,4);
disp0(27:52,5)=disp0(1:26,5);disp0(27:52,6)=-disp0(1:26,6);
disp0(53:78,1)=-disp0(1:26,1);disp0(53:78,2)=disp0(1:26,2);
disp0(53:78,3)=disp0(1:26,3);disp0(53:78,4)=disp0(1:26,4);
disp0(53:78,5)=-disp0(1:26,5);disp0(53:78,6)=-disp0(1:26,6);
disp0(79:104,1)=-disp0(1:26,1);disp0(79:104,2)=-disp0(1:26,2);
disp0(79:104,3)=disp0(1:26,3);disp0(79:104,4)=-disp0(1:26,4);
disp0(79:104,5)=-disp0(1:26,5);disp0(79:104,6)=disp0(1:26,6);
disp1(27:52,1)=disp1(1:26,1);disp1(27:52,2)=-disp1(1:26,2);
disp1(27:52,3)=disp1(1:26,3);disp1(27:52,4)=-disp1(1:26,4);
disp1(27:52,5)=disp1(1:26,5);disp1(27:52,6)=-disp1(1:26,6);
disp1(53:78,1)=-disp1(1:26,1);disp1(53:78,2)=disp1(1:26,2);
disp1(53:78,3)=disp1(1:26,3);disp1(53:78,4)=disp1(1:26,4);
disp1(53:78,5)=-disp1(1:26,5);disp1(53:78,6)=-disp1(1:26,6);
disp1(79:104,1)=-disp1(1:26,1);disp1(79:104,2)=-disp1(1:26,2);
disp1(79:104,3)=disp1(1:26,3);disp1(79:104,4)=-disp1(1:26,4);
disp1(79:104,5)=-disp1(1:26,5);disp1(79:104,6)=disp1(1:26,6);
for i=1:4
    disp1(104+i,1:3)=d(v(1,i+26),1:3);
    disp0(104+i,1:3)=d0(v(1,i+26),1:3);
    disp1(104+i,4:6)=r(v(1,i+26),1:3);
    disp0(104+i,4:6)=r0(v(1,i+26),1:3);
end
disp0(109,1:3)=d0(v(1,32),1:3);disp0(109,4:6)=r0(v(1,32),1:3);
```

```
disp1(109,1:3)=d(v(1,32),1:3);disp1(109,4:6)=r(v(1,32),1:3);
for i=1:4
    disp1(109+i,1:3)=d(v(1,-i+31),1:3);
    disp0(109+i,1:3)=d0(v(1,-i+31),1:3);
    disp1(109+i,4:6)=r(v(1,-i+31),1:3);
    disp0(109+i,4:6)=r0(v(1,-i+31),1:3);
end
disp0(114,1:3)=d0(v(1,31),1:3);disp0(114,4:6)=r0(v(1,31),1:3);
disp0(115,1:3)=d0(v(1,31),1:3);disp0(115,4:6)=r0(v(1,31),1:3);
disp1(114,1:3)=d(v(1,31),1:3);disp1(114,4:6)=r(v(1,31),1:3);
disp1(115,1:3)=d(v(1,31),1:3);disp1(115,4:6)=r(v(1,31),1:3);
for i=1:115
    for j=1:6
        d1(6*(i-1)+j,1)=disp1(i,j)-disp0(i,j);
    end
end
 % Rearranging stresses
s1(1,1)=max(sc); s1(2,1)=max(se);s1(3,1)=max(si);

toc
```

```
%Created by Julio Rodríguez Sáchez

% Laminae definition
% This is the main body of bridge frame model
% Every variable regarding space form of the bridge will be
   defined and
% declared here

%
```

```
% These are the variables that control the parametric form of
    the bridge

hc=section(1,1)*66/10;
heb=section(2,1)*66/10;
hib=section(3,1)*66/10;
htf=section(5,1)*66/10;
hbf=section(6,1)*66/10;


%
```

```
% Model True Construction I (Right Nodes Position)
m3_probe=0; m4_probe=1; m5_probe=2; m8_probe=3;

while (m4<m3 || m5<m4 || m8<m5)
if m4<m3
    m3_probe=m3; m3=m4; m4=m3_probe;
end
if m5<m4
    m4_probe=m4; m4=m5; m5=m4_probe;
end
if m8<m5
    m5_probe=m5; m5=m8; m8=m5_probe;
end
end

% Model True Construction II (Tune Node Position)
if m3==m4
    m4=m4+0.01;
end
if m4==m5
```

```
40      m5=m5+0.01;
41  end
42  if m5==m8
43      m8=m8+0.01;
44  end
45
46  % Definition of nodes
47  % Planar position of nodes (x & y axis) are defined in the
        following matrix
48  % "l_coor"
49
50  a1=l1+m6-m8;
51  a2=l1+m6-m5;
52  a3=l1+m6-m4;
53  a4=l1+m6-m3;
54  a5=l1+m6;
55  b1=l2+l2*m7/m1/2;
56  b2=l2+l2*m7/m1/2-m2;
57  t1=atan(b1/a1);
58  t2=atan(b1/a2);
59  t3=atan(b1/a3);
60  t4=atan(b1/a4);
61  t5=atan(b2/a5);
62  t6=atan(m2/m3);
63  t7=atan(b1/a5);
64  st=tan(t5)+tan(t6);
65  ts=tan(t4)+tan(t6);
66  yiv(1,1)=(a2/b1-a1/b1)^(-1)*(m8-m5-1/2*(heb/sin(t1)+hib/sin(t2)
        ));yiv(1,2)=(a3/b1-a2/b1)^(-1)*(m5-m4-1/2*(hib/sin(t2)+hib/
        sin(t3)));
67  yiv(1,3)=(a4/b1-a3/b1)^(-1)*(m4-m3-1/2*(hib/sin(t3)+hib/sin(t4)
        ));yiv(1,4)=(a5/b2-a4/b1)^(-1)*(m3-1/2*(hib/sin(t4)+heb/sin(
        t5))+m2*a5/b2);
68  yi=min(yiv);
```

```
% xcoordinate ycoordinate zcoordinate %#node

p_coor=[ a5+1/2*heb/sin(t1)    b1                0 0 0 0 ... %
      1   % Support (RIGHT)
        ;(m8+a1/b1*(yi-0.5))+1/2*heb/sin(t1)  yi-0.5  0 0 0 0
           ... %  2 % Bridle 1 (RIGHT)
        ;(m8+a1/b1*(yi-0.5))-1/2*heb/sin(t1)  yi-0.5  0 0 0 0
           ... %  3 % Bridle 1 (LEFT)
        ; a2/b1*(a2/b1-a1/b1)^(-1)*(m8-m5-1/2*(heb/sin(t1)+hib/
           sin(t2)))+m5+1/2*hib/sin(t2)              (a2/b1-a1/
           b1)^(-1)*(m8-m5-1/2*(heb/sin(t1)+hib/sin(t2)))
                      0 0 0 0 ... %  4 % Bridle 1-2
        ;(m5+a2/b1*(yi-0.5))+1/2*hib/sin(t2)  yi-0.5  0 0 0 0
           ... %  5 % Bridle 2 (RIGHT)
        ;(m5+a2/b1*(yi-0.5))-1/2*hib/sin(t2)  yi-0.5  0 0 0 0
           ... %  6 % Bridle 2 (LEFT)
        ; a3/b1*(a3/b1-a2/b1)^(-1)*(m5-m4-1/2*(hib/sin(t2)+hib/
           sin(t3)))+m4+1/2*hib/sin(t3)              (a3/b1-a2/
           b1)^(-1)*(m5-m4-1/2*(hib/sin(t2)+hib/sin(t3)))
                      0 0 0 0 ... %  7 % Bridle 2-3
        ;(m4+a3/b1*(yi-0.5))+1/2*hib/sin(t3)  yi-0.5  0 0 0 0
           ... %  8 % Bridle 3 (RIGHT)
        ;(m4+a3/b1*(yi-0.5))-1/2*hib/sin(t3)  yi-0.5  0 0 0 0
           ... %  9 % Bridle 3 (LEFT)
        ; a4/b1*(a4/b1-a3/b1)^(-1)*(m4-m3-1/2*(hib/sin(t3)+hib/
           sin(t4)))+m3+1/2*hib/sin(t4)              (a4/b1-a3/
           b1)^(-1)*(m4-m3-1/2*(hib/sin(t3)+hib/sin(t4)))
                      0 0 0 0 ... % 10 % Bridle 3-4
        ;(m3+a4/b1*(yi-0.5))+1/2*hib/sin(t4)  yi-0.5  0 0 0 0
           ... % 11 % Bridle 4 (RIGHT)
        ;(m3+a4/b1*(yi-0.5))-1/2*hib/sin(t4)  yi-0.5  0 0 0 0
           ... % 12 % Bridle 4 (LEFT)
        ; a5/b2*((a5/b2-a4/b1)^(-1)*(m3-00-1/2*(hib/sin(t4)+heb
           /sin(t5))+m2*a5/b2)-m2)+1/2*hib/sin(t4)    (a5/b2-a4/
```

```
            b1)^(-1)*(m3-1/2*(hib/sin(t4)+heb/sin(t5))+m2*a5/b2)
                  0 0 0 0 ... % 13 % Bridle 4-5
84          ;(a5/b2*(yi-0.5-m2))+1/2*heb/sin(t5)  yi-0.5  0 0 0 0
               ... % 14 % Bridle 5 (RIGHT)
85          ;(a5/b2*(yi-0.5-m2))-1/2*heb/sin(t5)  yi-0.5  0 0 0 0
               ... % 15 % Bridle 5 (LEFT)
86          ; a5-1/2*heb/sin(t5)  b1               0 0 0 0 ... %
               16   % Support (LEFT)
87          ];
88
89  j_coor=[ (m8+a1/b1*0.5)+1/2*heb/sin(t1)          0.5      0 0
       0 0 ... %   1  % Joint #1
90          ; m8+1/2*(heb/sin(t1))                   0        0 0
               0 0 ... %   2  % Joint #1
91          ; (m8+a1/b1*0.5)+1/2*heb/sin(t1)        -0.5      0 0
               0 0 ... %   3  % Joint #1
92          ; (m8+a1/b1*0.5)-1/2*heb/sin(t1)        -0.5      0 0
               0 0 ... %   4  % Joint #1
93          ; m8-1/2*(heb/sin(t1)-hc/tan(t1))       -hc/2     0 0
               0 0 ... %   5  % Joint #1
94          ; m8-1/2*(heb/sin(t1)-hc/tan(t1))-0.5   -hc/2     0 0
               0 0 ... %   6  % Joint #1
95          ; m8-1/2*(heb/sin(t1)-hc/tan(t1))-0.5    hc/2     0 0
               0 0 ... %   7  % Joint #1
96          ; m8-1/2*(heb/sin(t1)-hc/tan(t1))        hc/2     0 0
               0 0 ... %   8  % Joint #1
97          ; (m8+a1/b1*0.5)-1/2*heb/sin(t1)         0.5      0 0
               0 0 ... %   9  % Joint #1
98          ; (m5+a2/b1*0.5)+1/2*hib/sin(t2)         0.5      0 0
               0 0 ... %  10  % Joint #2
99          ; m5+1/2*(hib/sin(t2)+hc/tan(t2))        hc/2     0 0
               0 0 ... %  11  % Joint #2
100         ; m5+1/2*(hib/sin(t2)+hc/tan(t2))+0.5    hc/2     0 0
               0 0 ... %  12  % Joint #2
```

```
101         ; m5+1/2*(hib/sin(t2)+hc/tan(t2))+0.5    -hc/2     0 0
               0 0 ... % 13 % Joint #2
102         ; m5+1/2*(hib/sin(t2)+hc/tan(t2))        -hc/2     0 0
               0 0 ... % 14 % Joint #2
103         ; (m5+a2/b1*0.5)+1/2*hib/sin(t2)         -0.5      0 0
               0 0 ... % 15 % Joint #2
104         ; (m5+a2/b1*0.5)-1/2*hib/sin(t2)         -0.5      0 0
               0 0 ... % 16 % Joint #2
105         ; m5-1/2*(hib/sin(t2)-hc/tan(t2))        -hc/2     0 0
               0 0 ... % 17 % Joint #2
106         ; m5-1/2*(hib/sin(t2)-hc/tan(t2))-0.5    -hc/2     0 0
               0 0 ... % 18 % Joint #2
107         ; m5-1/2*(hib/sin(t2)-hc/tan(t2))-0.5     hc/2     0 0
               0 0 ... % 19 % Joint #2
108         ; m5-1/2*(hib/sin(t2)-hc/tan(t2))         hc/2     0 0
               0 0 ... % 20 % Joint #2
109         ; (m5+a2/b1*0.5)-1/2*hib/sin(t2)          0.5      0 0
               0 0 ... % 21 % Joint #2
110         ; (m4+a3/b1*0.5)+1/2*hib/sin(t2)          0.5      0 0
               0 0 ... % 22 % Joint #3
111         ; m4+1/2*(hib/sin(t3)+hc/tan(t3))         hc/2     0 0
               0 0 ... % 23 % Joint #3
112         ; m4+1/2*(hib/sin(t3)+hc/tan(t3))+0.5     hc/2     0 0
               0 0 ... % 24 % Joint #3
113         ; m4+1/2*(hib/sin(t3)+hc/tan(t3))+0.5    -hc/2     0 0
               0 0 ... % 25 % Joint #3
114         ; m4+1/2*(hib/sin(t3)+hc/tan(t3))        -hc/2     0 0
               0 0 ... % 26 % Joint #3
115         ; (m4+a3/b1*0.5)+1/2*hib/sin(t3)         -0.5      0 0
               0 0 ... % 27 % Joint #3
116         ; (m4+a3/b1*0.5)-1/2*hib/sin(t3)         -0.5      0 0
               0 0 ... % 28 % Joint #3
117         ; m4-1/2*(hib/sin(t3)-hc/tan(t3))        -hc/2     0 0
               0 0 ... % 29 % Joint #3
```

```
118         ; m4-1/2*(hib/sin(t3)-hc/tan(t3))-0.5    -hc/2     0 0
               0 0 ... % 30 % Joint #3
119         ; m4-1/2*(hib/sin(t3)-hc/tan(t3))-0.5     hc/2     0 0
               0 0 ... % 31 % Joint #3
120         ; m4-1/2*(hib/sin(t3)-hc/tan(t3))         hc/2     0 0
               0 0 ... % 32 % Joint #3
121         ; (m4+a3/b1*0.5)-1/2*hib/sin(t3)          0.5      0 0
               0 0 ... % 33 % Joint #3
122         ; (m3+a4/b1*0.5)+1/2*hib/sin(t4)          0.5      0 0
               0 0 ... % 34 % Joint #4
123         ; m3+1/2*(hib/sin(t4)+hc/tan(t4))         hc/2     0 0
               0 0 ... % 35 % Joint #4
124         ; m3+1/2*(hib/sin(t4)+hc/tan(t4))+0.5     hc/2     0 0
               0 0 ... % 36 % Joint #4
125         ; m3+1/2*(hib/sin(t4)+hc/tan(t4))+0.5    -hc/2     0 0
               0 0 ... % 37 % Joint #4
126         ; m3+1/2*(hib/sin(t4)+hc/tan(t4))        -hc/2     0 0
               0 0 ... % 38 % Joint #4
127         ; (m3+a4/b1*0.5)+1/2*hib/sin(t4)         -0.5      0 0
               0 0 ... % 39 % Joint #4
128         ; (m3+a4/b1*0.5)-1/2*hib/sin(t4)         -0.5      0 0
               0 0 ... % 40 % Joint #4
129         ; (m2+m3*tan(t4)+1/2*hc/cos(t6)-1/2*hib/cos(t4))/ts -m2
               -1/2*hc/cos(t6)+(m2+m3*tan(t4)+1/2*hc/cos(t6)-1/2*
               hib/cos(t4))/ts*tan(t6)  0 0 0 0 ... % 41 % Joint
               #4
130         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5    -hc/2-0.5*tan(
               t6)-hc/cos(t6)                    0 0 0 0 ... % 42
                % Joint #4
131         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5    -hc/2-0.5*tan(
               t6)                               0 0 0 0 ... % 43
                % Joint #4
132         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)        -hc/2
                        0 0 0 0 ... % 44 % Joint #4
```

```
133         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5    -hc/2     0 0
               0 0 ... %  45  % Joint #4
134         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5     hc/2     0 0
               0 0 ... %  46  % Joint #4
135         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)         hc/2
                          0 0 0 0 ... %  47  % Joint #4
136         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5     hc/2+0.5*tan(
               t6)                      0 0 0 0 ... %  48  % Joint #4
137         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5     hc/2+0.5*tan(
               t6)+hc/cos(t6)                     0 0 0 0 ... %  49
                % Joint #4
138         ; (m2+m3*tan(t4)+1/2*hc/cos(t6)-1/2*hib/cos(t4))/ts  m2
               +1/2*hc/cos(t6)-(m2+m3*tan(t4)+1/2*hc/cos(t6)-1/2*
               hib/cos(t4))/ts*tan(t6)  0 0 0 0 ... % 50  % Joint
               #4
139         ; (m3+a4/b1*0.5)-1/2*hib/sin(t4)          0.5
                        0 0 0 0 ... % 51  % Joint #4
140         ];
141
142 s_coor=[ 0                                        m2+1/2*heb
                  0 0 0 0 ... % 1  % Joint #5
143       ; (a5/b2*0.5)-1/2*heb/sin(t5)              m2+0.5
                     0 0 0 0 ... % 2 % Joint #5
144       ; (a5/b2*0.5)+1/2*heb/sin(t5)              m2+0.5
                     0 0 0 0 ... % 3 % Joint #5
145       ;  1/2*(heb/cos(t5)+hc/cos(t6))/st         m2-0.5*heb/cos
              (t5)+1/2*((heb/cos(t5)+hc/cos(t6))/st)*tan(t5)
              0 0 0 0 ... % 4 % Joint #5
146       ;  1/2*heb/cos(t5)+0.5                      (m3-1/2*heb/
              cos(t5)-0.5)*tan(t6)+1/2*hc/cos(t6)   0 0 0 0 ... % 5
                % Joint #5
147       ;  1/2*heb/cos(t5)+0.5                      (m3-1/2*heb/
              cos(t5)-0.5)*tan(t6)-1/2*hc/cos(t6)   0 0 0 0 ... % 6
                % Joint #5
```

```
148         ; 0                                                 m2-1/2*hc/cos(
                t6)  0 0 0 0 ... % 7  % Joint #5
149         ; -1/2*heb/cos(t5)-0.5                              (m3-1/2*heb/
                cos(t5)-0.5)*tan(t6)-1/2*hc/cos(t6)   0 0 0 0 ... % 8
                 % Joint #5
150         ; -1/2*heb/cos(t5)-0.5                              (m3-1/2*heb/
                cos(t5)-0.5)*tan(t6)+1/2*hc/cos(t6)   0 0 0 0 ... % 9
                 % Joint #5
151         ; -1/2*(heb/cos(t5)+hc/cos(t6))/st                  m2-0.5*heb/cos
                (t5)+1/2*((heb/cos(t5)+hc/cos(t6))/st)*tan(t5)
                            0 0 0 0 ... % 10  % Joint #5
152         ;-(a5/b2*0.5)-1/2*heb/sin(t5)                       m2+0.5
                        0 0 0 0 ... % 11  % Joint #5
153         ;-(a5/b2*0.5)+1/2*heb/sin(t5)                       m2+0.5
                        0 0 0 0 ... % 12  % Joint #5
154         ; 0                                                 -m2-1/2*heb
                         0 0 0 0 ... % 13  % Joint #6
155         ; (a5/b2*0.5)-1/2*heb/sin(t5)                       -m2-0.5
                        0 0 0 0 ... % 14  % Joint #6
156         ; (a5/b2*0.5)+1/2*heb/sin(t5)                       -m2-0.5
                        0 0 0 0 ... % 15 % Joint #6
157         ;  1/2*(heb/cos(t5)+hc/cos(t6))/st                  -m2+0.5*heb/cos
                (t5)-1/2*((heb/cos(t5)+hc/cos(t6))/st)*tan(t5)
                0 0 0 0 ... % 16  % Joint #6
158         ;  1/2*heb/cos(t5)+0.5                              -(m3-1/2*heb/
                cos(t5)-0.5)*tan(t6)-1/2*hc/cos(t6)   0 0 0 0 ... %
                17  % Joint #6
159         ;  1/2*heb/cos(t5)+0.5                              -(m3-1/2*heb/
                cos(t5)-0.5)*tan(t6)+1/2*hc/cos(t6)   0 0 0 0 ... %
                18 % Joint #6
160         ; 0                                                 -m2+1/2*hc/cos(
                t6)  0 0 0 0 ... % 19  % Joint #5
161         ; -1/2*heb/cos(t5)-0.5                              -(m3-1/2*heb/
                cos(t5)-0.5)*tan(t6)+1/2*hc/cos(t6)   0 0 0 0 ... %
```

```
            20 % Joint #6
162         ; -1/2*heb/cos(t5)-0.5                          -(m3-1/2*heb/
              cos(t5)-0.5)*tan(t6)-1/2*hc/cos(t6)    0 0 0 0 ... %
              21 % Joint #6
163         ; -1/2*(heb/cos(t5)+hc/cos(t6))/st        -m2+0.5*heb/cos
              (t5)-1/2*((heb/cos(t5)+hc/cos(t6))/st)*tan(t5)
                          0 0 0 0 ... % 22  % Joint #6
164         ;-(a5/b2*0.5)-1/2*heb/sin(t5)             -m2-0.5
                        0 0 0 0 ... % 23  % Joint #6
165         ;-(a5/b2*0.5)+1/2*heb/sin(t5)             -m2-0.5
                        0 0 0 0 ... % 24  % Joint #6
166         ];
167
168 b_coor=[ (m8+a1/b1*10)+1/2*heb/sin(t1)    10                0 0
        0 0 ... %   1   % Bridle #1R (RIGHT)
169         ; (m8+a1/b1*6)+1/2*heb/sin(t1)      6               0 0
              0 0 ... %   2   % Bridle #1R
170         ; (m8+a1/b1*4)+1/2*heb/sin(t1)      4               0 0
              0 0 ... %   3   % Bridle #1R
171         ; (m8+a1/b1*2)+1/2*heb/sin(t1)      2               0 0
              0 0 ... %   4   % Bridle #1R
172         ; (m8+a1/b1*1)+1/2*heb/sin(t1)      1               0 0
              0 0 ... %   5   % Bridle #1R
173         ; (m8+a1/b1*1)-1/2*heb/sin(t1)      1               0 0
              0 0 ... %   6   % Bridle #1L (LEFT)
174         ; (m8+a1/b1*2)-1/2*heb/sin(t1)      2               0 0
              0 0 ... %   7   % Bridle #1L
175         ; (m8+a1/b1*4)-1/2*heb/sin(t1)      4               0 0
              0 0 ... %   8   % Bridle #1L
176         ; (m8+a1/b1*6)-1/2*heb/sin(t1)      6               0 0
              0 0 ... %   9   % Bridle #1L
177         ; (m8+a1/b1*10)-1/2*heb/sin(t1)    10               0 0
              0 0 ... %  10   % Bridle #1L
```

```
178         ; (m5+a2/b1*10)+1/2*hib/sin(t2)    10                0 0
                0 0 ... %  11   % Bridle #2R (RIGHT)
179         ; (m5+a2/b1*6)+1/2*hib/sin(t2)      6                0 0
                0 0 ... %  12   % Bridle #2R
180         ; (m5+a2/b1*4)+1/2*hib/sin(t2)      4                0 0
                0 0 ... %  13   % Bridle #2R
181         ; (m5+a2/b1*2)+1/2*hib/sin(t2)      2                0 0
                0 0 ... %  14   % Bridle #2R
182         ; (m5+a2/b1*1)+1/2*hib/sin(t2)      1                0 0
                0 0 ... %  15   % Bridle #2R
183         ; (m5+a2/b1*1)-1/2*hib/sin(t2)      1                0 0
                0 0 ... %  16   % Bridle #2L (LEFT)
184         ; (m5+a2/b1*2)-1/2*hib/sin(t2)      2                0 0
                0 0 ... %  17   % Bridle #2L
185         ; (m5+a2/b1*4)-1/2*hib/sin(t2)      4                0 0
                0 0 ... %  18   % Bridle #2L
186         ; (m5+a2/b1*6)-1/2*hib/sin(t2)      6                0 0
                0 0 ... %  19   % Bridle #2L
187         ; (m5+a2/b1*10)-1/2*hib/sin(t2)    10                0 0
                0 0 ... %  20   % Bridle #2L
188         ; (m4+a3/b1*10)+1/2*hib/sin(t3)    10                0 0
                0 0 ... %  21   % Bridle #3R (RIGHT)
189         ; (m4+a3/b1*6)+1/2*hib/sin(t3)      6                0 0
                0 0 ... %  22   % Bridle #3R
190         ; (m4+a3/b1*4)+1/2*hib/sin(t3)      4                0 0
                0 0 ... %  23   % Bridle #3R
191         ; (m4+a3/b1*2)+1/2*hib/sin(t3)      2                0 0
                0 0 ... %  24   % Bridle #3R
192         ; (m4+a3/b1*1)+1/2*hib/sin(t3)      1                0 0
                0 0 ... %  25   % Bridle #3R
193         ; (m4+a3/b1*1)-1/2*hib/sin(t3)      1                0 0
                0 0 ... %  26   % Bridle #3L (LEFT)
194         ; (m4+a3/b1*2)-1/2*hib/sin(t3)      2                0 0
                0 0 ... %  27   % Bridle #3L
```

```
195        ; (m4+a3/b1*4)-1/2*hib/sin(t3)      4                0 0
              0 0 ... %  28   % Bridle #3L
196        ; (m4+a3/b1*6)-1/2*hib/sin(t3)      6                0 0
              0 0 ... %  29   % Bridle #3L
197        ; (m4+a3/b1*10)-1/2*hib/sin(t3)    10                0 0
              0 0 ... %  30   % Bridle #3L
198        ; (m3+a4/b1*10)+1/2*hib/sin(t4)    10                0 0
              0 0 ... %  31   % Bridle #4R (RIGHT)
199        ; (m3+a4/b1*6)+1/2*hib/sin(t4)      6                0 0
              0 0 ... %  32   % Bridle #4R
200        ; (m3+a4/b1*4)+1/2*hib/sin(t4)      4                0 0
              0 0 ... %  33   % Bridle #4R
201        ; (m3+a4/b1*2)+1/2*hib/sin(t4)      2                0 0
              0 0 ... %  34   % Bridle #4R
202        ; (m3+a4/b1*1)+1/2*hib/sin(t4)      1                0 0
              0 0 ... %  35   % Bridle #4R
203        ; (m3+a4/b1*1)-1/2*hib/sin(t4)      1                0 0
              0 0 ... %  36   % Bridle #4L (LEFT)
204        ; (m3+a4/b1*2)-1/2*hib/sin(t4)      2                0 0
              0 0 ... %  37   % Bridle #4L
205        ; (m3+a4/b1*4)-1/2*hib/sin(t4)      4                0 0
              0 0 ... %  38   % Bridle #4L
206        ; (m3+a4/b1*6)-1/2*hib/sin(t4)      6                0 0
              0 0 ... %  39   % Bridle #4L
207        ; (m3+a4/b1*10)-1/2*hib/sin(t4)    10                0 0
              0 0 ... %  40   % Bridle #4L
208        ; (a5/b2*(10-m2))+1/2*heb/sin(t5)  10                0 0
              0 0 ... %  41   % Bridle #5R (RIGHT)
209        ; (a5/b2*(6-m2))+1/2*heb/sin(t5)    6                0 0
              0 0 ... %  42   % Bridle #5R
210        ; (a5/b2*3)+1/2*heb/sin(t5)        m2+3              0 0
              0 0 ... %  43   % Bridle #5R
211        ; (a5/b2*2)+1/2*heb/sin(t5)        m2+2              0 0
              0 0 ... %  44   % Bridle #5R
```

```
212         ; (a5/b2*1)+1/2*heb/sin(t5)         m2+1              0 0
                0 0 ... %  45   % Bridle #5R
213         ; (a5/b2*1)-1/2*heb/sin(t5)         m2+1              0 0
                0 0 ... %  46   % Bridle #5L (LEFT)
214         ; (a5/b2*2)-1/2*heb/sin(t5)         m2+2              0 0
                0 0 ... %  47   % Bridle #5L
215         ; (a5/b2*3)-1/2*heb/sin(t5)         m2+3              0 0
                0 0 ... %  48   % Bridle #5R
216         ; (a5/b2*(6-m2))-1/2*heb/sin(t5)     6                0 0
                0 0 ... %  49   % Bridle #5L
217         ; (a5/b2*(10-m2))-1/2*heb/sin(t5)  10                 0 0
                0 0 ... %  50   % Bridle #5L
218         ];
219
220 c_coor=[  m8-1/2*(heb/sin(t1)-hc/tan(t1))-0.5    hc/2      0 0 0
        0 ... %  1   % 1st Part (UPPER)
221         ;  m5+1/2*(hib/sin(t2)+hc/tan(t2))+0.5    hc/2      0 0 0
                0 ... %  2   % 1st Part (UPPER)
222         ;  m5+1/2*(hib/sin(t2)+hc/tan(t2))+0.5   -hc/2      0 0 0
                0 ... %  3   % 1st Part (BOTTOM)
223         ;  m8-1/2*(heb/sin(t1)-hc/tan(t1))-0.5   -hc/2      0 0 0
                0 ... %  4   % 1st Part (BOTTOM)
224         ;  m5-1/2*(hib/sin(t2)-hc/tan(t2))-0.5    hc/2      0 0 0
                0 ... %  5   % 2nd Part (UPPER)
225         ;  m4+1/2*(hib/sin(t3)+hc/tan(t3))+0.5    hc/2      0 0 0
                0 ... %  6   % 2nd Part (UPPER)
226         ;  m4+1/2*(hib/sin(t3)+hc/tan(t3))+0.5   -hc/2      0 0 0
                0 ... %  7   % 2nd Part (BOTTOM)
227         ;  m5-1/2*(hib/sin(t2)-hc/tan(t2))-0.5   -hc/2      0 0 0
                0 ... %  8   % 2nd Part (BOTTOM)
228         ;  m4-1/2*(hib/sin(t3)-hc/tan(t3))-0.5    hc/2      0 0 0
                0 ... %  9   % 3rd Part (UPPER)
229         ;  m3+1/2*(hib/sin(t4)+hc/tan(t4))+0.5    hc/2      0 0 0
                0 ... % 10   % 3rd Part (UPPER)
```

```
230         ;  m3+1/2*(hib/sin(t4)+hc/tan(t4))+0.5   -hc/2      0 0 0
                0 ... % 11   % 3rd Part (BOTTOM)
231         ;  m4-1/2*(hib/sin(t3)-hc/tan(t3))-0.5   -hc/2      0 0 0
                0 ... % 12   % 3rd Part (BOTTOM)
232         ;  (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5    hc/2      0 0 0
                0 ... % 13   % 4th Part (UPPER)
233         ;  0                                       hc/2      0 0 0
                0 ... % 14   % 4th Part (UPPER)
234         ;  0                                      -hc/2      0 0 0
                0 ... % 15   % 4th Part (BOTTOM)
235         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5    -hc/2      0 0 0
                0 ... % 16   % 4th Part (BOTTOM)
236         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5     hc/2+0.5*tan(
               t6)+hc/cos(t6)          0 0 0 0 ... % 17   % 5th Part
               (UPPER)
237         ;  1/2*heb/cos(t5)+0.5                    (m3-1/2*heb/cos
               (t5)-0.5)*tan(t6)+0.5*hc/cos(t6)    0 0 0 0 ... % 18
                 % 5th Part (UPPER)
238         ;  1/2*heb/cos(t5)+0.5                    (m3-1/2*heb/cos
               (t5)-0.5)*tan(t6)-0.5*hc/cos(t6)    0 0 0 0 ... % 18
                 % 5th Part (BOTTOM)
239         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5     hc/2+0.5*tan(
               t6)                     0 0 0 0 ... % 20   % 5th Part
               (BOTTOM)
240         ; (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5    -hc/2-0.5*tan(
               t6)                     0 0 0 0 ... % 21   % 6th Part
               (UPPER)
241         ;  1/2*heb/cos(t5)+0.5                   -(m3-1/2*heb/
               cos(t5)-0.5)*tan(t6)+0.5*hc/cos(t6)    0 0 0 0 ... %
               22   % 6th Part (UPPER)
242         ;  1/2*heb/cos(t5)+0.5                   -(m3-1/2*heb/
               cos(t5)-0.5)*tan(t6)-0.5*hc/cos(t6)    0 0 0 0 ... %
               23   % 6th Part (BOTTOM)
```

```
243         ;  (m2-hc/2*(1/cos(t6)+1))/tan(t6)-0.5   -hc/2-0.5*tan(
               t6)-hc/cos(t6)                      0 0 0 0 ... % 24
               % 6th Part (BOTTOM)
244         ];
245
246 d_coor=[ a5    6              0 0 0 0 ... %   1   % Bridle #1R
        (RIGHT)
247         ; a5    4              0 0 0 0 ... %   2   % Bridle #1R
                (RIGHT)
248         ; a5    2              0 0 0 0 ... %   3   % Bridle #1R
                (RIGHT)
249         ; a5    1              0 0 0 0 ... %   4   % Bridle #1R
                (RIGHT)
250         ; a5    0.5            0 0 0 0 ... %   5   % Bridle #1R
                (RIGHT)
251         ; a5    0              0 0 0 0 ... %   6   % Bridle #1R
                (RIGHT)
252         ; a5   -0.5            0 0 0 0 ... %   7   % Bridle #1R
                (RIGHT)
253         ; a5   -1              0 0 0 0 ... %   8   % Bridle #1R
                (RIGHT)
254         ; a5   -2              0 0 0 0 ... %   9   % Bridle #1R
                (RIGHT)
255         ; a5   -4              0 0 0 0 ... %  10   % Bridle #1R
                (RIGHT)
256         ; a5   -6              0 0 0 0 ... %  11   % Bridle #1R
                (RIGHT)
257         ];
258
259 f_coor=[ a5-htf/2    6              0 0 0 0 ... %   1   %
       Bridle #1R (RIGHT)
260         ; a5-htf/2   -6             0 0 0 0 ... %   2   %
               Bridle #1R (RIGHT)
```

```
261         ; a5-htf/2    4               0 0 0 0 ... %   3   %
                Bridle #1R (RIGHT)
262         ; a5-hbf/2    2               0 0 0 0 ... %   4   %
                Bridle #1R (RIGHT)
263         ; a5-hbf/2    1               0 0 0 0 ... %   5   %
                Bridle #1R (RIGHT)
264         ; a5-hbf/2    0.5             0 0 0 0 ... %   6   %
                Bridle #1R (RIGHT)
265         ; a5-hbf/2    0               0 0 0 0 ... %   7   %
                Bridle #1R (RIGHT)
266         ; a5-hbf/2   -0.5             0 0 0 0 ... %   8   %
                Bridle #1R (RIGHT)
267         ; a5-hbf/2   -1               0 0 0 0 ... %   9   %
                Bridle #1R (RIGHT)
268         ; a5-hbf/2   -2               0 0 0 0 ... %  10   %
                Bridle #1R (RIGHT)
269         ; a5-hbf/2   -4               0 0 0 0 ... %  11   %
                Bridle #1R (RIGHT)
270         ; a5+htf/2    6               0 0 0 0 ... %  12   %
                Bridle #1R (RIGHT)
271         ; a5+htf/2   -6               0 0 0 0 ... %  13   %
                Bridle #1R (RIGHT)
272         ; a5+htf/2    4               0 0 0 0 ... %  14   %
                Bridle #1R (RIGHT)
273         ; a5+hbf/2    2               0 0 0 0 ... %  15   %
                Bridle #1R (RIGHT)
274         ; a5+hbf/2    1               0 0 0 0 ... %  16   %
                Bridle #1R (RIGHT)
275         ; a5+hbf/2    0.5             0 0 0 0 ... %  17   %
                Bridle #1R (RIGHT)
276         ; a5+hbf/2    0               0 0 0 0 ... %  18   %
                Bridle #1R (RIGHT)
277         ; a5+hbf/2   -0.5             0 0 0 0 ... %  19   %
                Bridle #1R (RIGHT)
```

```
278         ; a5+hbf/2   -1               0 0 0 0 ... %  20   %
               Bridle #1R (RIGHT)
279         ; a5+hbf/2   -2               0 0 0 0 ... %  21   %
               Bridle #1R (RIGHT)
280         ; a5+hbf/2   -4               0 0 0 0 ... %  22   %
               Bridle #1R (RIGHT)
281         ];
282
283 t_coor=[ a5    6               0 0 0 0 ... %   1   % Bridle #1R
       (RIGHT)
284         ; a5   -6               0 0 0 0 ... %  11   % Bridle #1R
              (RIGHT)
285         ;-a5    6               0 0 0 0 ... %   1   % Bridle #1R
              (RIGHT)
286         ;-a5   -6               0 0 0 0 ... %  11   % Bridle #1R
              (RIGHT)
287         ];
288
289 k_coor=[ a5    3               0 0 0 0 ... %   1   % Bridle #1R
       (RIGHT)
290         ; a5-1.5*66/10    3               0 0 0 0 ... %   2   %
               Bridle #1R (RIGHT)
291         ; a5    3+hbf/2               0 0 0 0 ... %   3   %
               Bridle #1R (RIGHT)
292         ; a5-1.5*66/10    3+hbf/2               0 0 0 0 ... %
                4   % Bridle #1R (RIGHT)
293         ; a5    3-hbf/2               0 0 0 0 ... %   5   %
               Bridle #1R (RIGHT)
294         ; a5-1.5*66/10    3-hbf/2               0 0 0 0 ... %
                6   % Bridle #1R (RIGHT)
295         ; a5   -3               0 0 0 0 ... %   7   % Bridle #1R
              (RIGHT)
296         ; a5-1.5*66/10   -3               0 0 0 0 ... %   8   %
               Bridle #1R (RIGHT)
```

```
297         ; a5   -3-hbf/2               0 0 0 0 ... %   9   %
                Bridle #1R (RIGHT)
298         ; a5-1.5*66/10   -3-hbf/2               0 0 0 0 ... %
                10   % Bridle #1R (RIGHT)
299         ; a5   -3+hbf/2               0 0 0 0 ... %  11   %
                Bridle #1R (RIGHT)
300         ; a5-1.5*66/10   -3+hbf/2               0 0 0 0 ... %
                12   % Bridle #1R (RIGHT)
301         ; a5    3               0 0 0 0 ... %  13   % Bridle #1R
                 (RIGHT)
302         ; a5-1.5*66/10    3               0 0 0 0 ... %  14   %
                Bridle #1R (RIGHT)
303         ; a5    3+htf/2               0 0 0 0 ... %  15   %
                Bridle #1R (RIGHT)
304         ; a5-1.5*66/10    3+htf/2               0 0 0 0 ... %
                 16   % Bridle #1R (RIGHT)
305         ; a5    3-htf/2               0 0 0 0 ... %   17   %
                Bridle #1R (RIGHT)
306         ; a5-1.5*66/10    3-htf/2               0 0 0 0 ... %
                 18   % Bridle #1R (RIGHT)
307         ; a5   -3               0 0 0 0 ... %  19   % Bridle #1R
                 (RIGHT)
308         ; a5-1.5*66/10   -3               0 0 0 0 ... %  20   %
                Bridle #1R (RIGHT)
309         ; a5   -3-htf/2               0 0 0 0 ... %  21   %
                Bridle #1R (RIGHT)
310         ; a5-1.5*66/10   -3-htf/2               0 0 0 0 ... %
                 22   % Bridle #1R (RIGHT)
311         ; a5   -3+htf/2               0 0 0 0 ... %   23   %
                Bridle #1R (RIGHT)
312         ; a5-1.5*66/10   -3+htf/2               0 0 0 0 ... %
                 24   % Bridle #1R (RIGHT)
313         ];
314
```

```
315     for i=1:16
316         l_coor(i,1)=p_coor(i,1);
317         l_coor(i,2)=p_coor(i,2);
318         l_coor(i+1*16,1)=-p_coor(i,1);
319         l_coor(i+1*16,2)=p_coor(i,2);
320         l_coor(i+2*16,1)=p_coor(i,1);
321         l_coor(i+2*16,2)=-p_coor(i,2);
322         l_coor(i+3*16,1)=-p_coor(i,1);
323         l_coor(i+3*16,2)=-p_coor(i,2);
324     end
325     % Points Created: 64 points
326     for i=1:51
327         l_coor(i+64,1)=j_coor(i,1);
328         l_coor(i+64+51,1)=-j_coor(i,1);
329         l_coor(i+64,2)=j_coor(i,2);
330         l_coor(i+64+51,2)=j_coor(i,2);
331     end
332     % Points Created:102 ; Total Points Created: 166
333     for i=1:24
334         l_coor(i+166,1)=s_coor(i,1);
335         l_coor(i+166,2)=s_coor(i,2);
336     end
337     % Points Created:24 ; Total Points Created: 190
338     for i=1:50
339         l_coor(i+190,1)=b_coor(i,1);
340         l_coor(i+190,2)=b_coor(i,2);
341         l_coor(i+190+1*50,1)=(-1)*b_coor(i,1);
342         l_coor(i+190+1*50,2)=b_coor(i,2);
343         l_coor(i+190++2*50,1)=b_coor(i,1);
344         l_coor(i+190+2*50,2)=(-1)*b_coor(i,2);
345         l_coor(i+190+3*50,1)=(-1)*b_coor(i,1);
346         l_coor(i+190+3*50,2)=(-1)*b_coor(i,2);
347     end
348     % Points Created:200 ; Total Points Created: 390
```

```
    for i=1:11
        l_coor(i+390,1)=d_coor(i,1);
        l_coor(i+390,2)=d_coor(i,2);
    end
% Points Created:64 ; Total Points Created: 401
    for i=1:22
        l_coor(i+401,1)=f_coor(i,1);
        l_coor(i+401,2)=f_coor(i,2);
    end
% Points Created:22 ; Total Points Created: 423
    for i=1:4
        l_coor(i+423,1)=t_coor(i,1);
        l_coor(i+423,2)=t_coor(i,2);
    end
% Points Created:4 ; Total Points Created: 427
       for i=1:24
        l_coor(i+427,1)=k_coor(i,1);
        l_coor(i+427,2)=k_coor(i,2);
       end
% Points Created:12 ; Total Points Created: 439

    p_conc=[  1   2 ... %  1
            ;  2   3 ... %  2
            ;  3   4 ... %  3
            ;  4   5 ... %  4
            ;  5   6 ... %  5
            ;  6   7 ... %  6
            ;  7   8 ... %  7
            ;  8   9 ... %  8
            ;  9  10 ... %  9
            ; 10  11 ... % 10
            ; 11  12 ... % 11
            ; 12  13 ... % 12
            ; 13  14 ... % 13
```

```
383             ; 14  15 ... % 14
384             ; 15  16 ... % 15
385             ; 16   1 ... % 16
386             ];
387     p_conc=p_conc+0;
388
389     j_conc=[  1   2 ... %  1
390             ;  2   3 ... %  2
391             ;  3   4 ... %  3
392             ;  4   5 ... %  4
393             ;  5   6 ... %  5
394             ;  6   7 ... %  6
395             ;  7   8 ... %  7
396             ;  8   9 ... %  8
397             ;  9   1 ... %  9
398             ; 10  11 ... % 10
399             ; 11  12 ... % 11
400             ; 12  13 ... % 12
401             ; 13  14 ... % 13
402             ; 14  15 ... % 14
403             ; 15  16 ... % 15
404             ; 16  17 ... % 16
405             ; 17  18 ... % 17
406             ; 18  19 ... % 18
407             ; 19  20 ... % 19
408             ; 20  21 ... % 20
409             ; 21  10 ... % 21
410             ; 22  23 ... % 22
411             ; 23  24 ... % 23
412             ; 24  25 ... % 24
413             ; 25  26 ... % 25
414             ; 26  27 ... % 26
415             ; 27  28 ... % 27
416             ; 28  29 ... % 28
```

```
417             ; 29  30 ... % 29
418             ; 30  31 ... % 30
419             ; 31  32 ... % 31
420             ; 32  33 ... % 32
421             ; 33  22 ... % 33
422             ; 34  35 ... % 34
423             ; 35  36 ... % 35
424             ; 36  37 ... % 36
425             ; 37  38 ... % 37
426             ; 38  39 ... % 38
427             ; 39  40 ... % 39
428             ; 40  41 ... % 40
429             ; 41  42 ... % 41
430             ; 42  43 ... % 42
431             ; 43  44 ... % 43
432             ; 44  45 ... % 44
433             ; 45  46 ... % 45
434             ; 46  47 ... % 46
435             ; 47  48 ... % 47
436             ; 48  49 ... % 48
437             ; 49  50 ... % 49
438             ; 50  51 ... % 50
439             ; 51  34 ... % 51
440             ];
441         j_conc=j_conc+64;
442
443     s_conc=[  1   2 ... %  1
444             ;  2   3 ... %  2
445             ;  3   4 ... %  3
446             ;  4   5 ... %  4
447             ;  5   6 ... %  5
448             ;  6   7 ... %  6
449             ;  7   8 ... %  7
450             ;  8   9 ... %  8
```

```
451             ;  9  10 ... %  9
452             ; 10  11 ... % 10
453             ; 11  12 ... % 11
454             ; 12   1 ... % 12
455             ; 13  14 ... % 13
456             ; 14  15 ... % 14
457             ; 15  16 ... % 15
458             ; 16  17 ... % 16
459             ; 17  18 ... % 17
460             ; 18  19 ... % 18
461             ; 19  20 ... % 19
462             ; 20  21 ... % 20
463             ; 21  22 ... % 21
464             ; 22  23 ... % 22
465             ; 23  24 ... % 23
466             ; 24  13 ... % 24
467             ];
468     s_conc=s_conc+166;
469
470     b_conc=[  0   1 ... %   1
471             ;  1   2 ... %   2
472             ;  2   3 ... %   3
473             ;  3   4 ... %   4
474             ;  4   5 ... %   5
475             ;  5   0 ... %   6
476             ;  0   6 ... %   7
477             ;  6   7 ... %   8
478             ;  7   8 ... %   9
479             ;  8   9 ... %  10
480             ;  9  10 ... %  11
481             ; 10   0 ... %  12
482             ;  0  11 ... %  13
483             ; 11  12 ... %  14
484             ; 12  13 ... %  15
```

```
485             ; 13  14 ... % 16
486             ; 14  15 ... % 17
487             ; 15   0 ... % 18
488             ;  0  16 ... % 19
489             ; 16  17 ... % 20
490             ; 17  18 ... % 21
491             ; 18  19 ... % 22
492             ; 19  20 ... % 23
493             ; 20   0 ... % 24
494             ;  0  21 ... % 25
495             ; 21  22 ... % 26
496             ; 22  23 ... % 27
497             ; 23  24 ... % 28
498             ; 24  25 ... % 29
499             ; 25   0 ... % 30
500             ;  0  26 ... % 31
501             ; 26  27 ... % 32
502             ; 27  28 ... % 33
503             ; 28  29 ... % 34
504             ; 29  30 ... % 35
505             ; 30   0 ... % 36
506             ;  0  31 ... % 37
507             ; 31  32 ... % 38
508             ; 32  33 ... % 39
509             ; 33  34 ... % 40
510             ; 34  35 ... % 41
511             ; 35   0 ... % 42
512             ;  0  36 ... % 43
513             ; 36  37 ... % 44
514             ; 37  38 ... % 45
515             ; 38  39 ... % 46
516             ; 39  40 ... % 47
517             ; 40   0 ... % 48
518             ;  0  41 ... % 49
```

```
519           ; 41  42 ... %  50
520           ; 42  43 ... %  51
521           ; 43  44 ... %  52
522           ; 44  45 ... %  53
523           ; 45   0 ... %  54
524           ;  0  46 ... %  55
525           ; 46  47 ... %  56
526           ; 47  48 ... %  57
527           ; 48  49 ... %  58
528           ; 49  50 ... %  59
529           ; 50   0 ... %  60
530           ];
531    b_conc=b_conc+190;
532
533    c_conc=[ 71  76 ... %  1
534           ; 70  77 ... %  2
535           ; 83  88 ... %  3
536           ; 82  89 ... %  4
537           ; 95 100 ... %  5
538           ; 94 101 ... %  6
539           ;110 161 ... %  7
540           ;109 160 ... %  8
541           ;113 171 ... %  9
542           ;112 172 ... % 10
543           ;107 184 ... % 11
544           ;106 183 ... % 12
545           ;175 164 ... % 13
546           ;174 163 ... % 14
547           ;186 158 ... % 15
548           ;187 157 ... % 16
549           ;151 146 ... % 17
550           ;152 145 ... % 18
551           ;139 134 ... % 19
552           ;140 133 ... % 20
```

```
553             ;127 122 ... % 21
554             ;128 121 ... % 22
555             ];
556
557     d_conc=[  1   2 ... %  1
558             ;  2   3 ... %  2
559             ;  3   4 ... %  3
560             ;  4   5 ... %  4
561             ;  5   6 ... %  5
562             ;  6   7 ... %  6
563             ;  7   8 ... %  7
564             ;  8   9 ... %  8
565             ;  9  10 ... %  9
566             ; 10  11 ... % 10
567             ; 11   1 ... % 11
568             ];
569     d_conc=d_conc+390;
570
571     f_conc=[  1   2 ... %  1
572             ;  1   3 ... %  2
573             ;  3   4 ... %  3
574             ;  4   5 ... %  4
575             ;  5   6 ... %  5
576             ;  6   7 ... %  6
577             ;  7   8 ... %  7
578             ;  8   9 ... %  8
579             ;  9  10 ... %  9
580             ; 10  11 ... % 10
581             ; 11   2 ... % 11
582             ; 12  13 ... % 12
583             ; 12  14 ... % 13
584             ; 14  15 ... % 14
585             ; 15  16 ... % 15
586             ; 16  17 ... % 16
```

```
587           ; 17  18 ... % 17
588           ; 18  19 ... % 18
589           ; 19  20 ... % 19
590           ; 20  21 ... % 20
591           ; 21  22 ... % 21
592           ; 22  13 ... % 22
593           ];
594     f_conc=f_conc+401;
595
596     t_conc=[  1   2 ... %  1
597           ;  3   4 ... %  2
598           ];
599     t_conc=t_conc+423;
600
601     k_conc=[  1   2 ... %  1
602           ;  3   4 ... %  2
603           ;  5   6 ... %  3
604           ;  7   8 ... %  4
605           ;  9  10 ... %  5
606           ; 11  12 ... %  6
607           ; 13  14 ... %  7
608           ; 15  16 ... %  8
609           ; 17  18 ... %  9
610           ; 19  20 ... % 10
611           ; 21  22 ... % 11
612           ; 23  24 ... % 12
613           ];
614     k_conc=k_conc+427;
615
616     for i=1:16
617         l_conc(i,1)=p_conc(i,1);
618         l_conc(i,2)=p_conc(i,2);
619         l_conc(i+1*16,1)=p_conc(i,1)+1*16;
620         l_conc(i+1*16,2)=p_conc(i,2)+1*16;
```

```
621         l_conc(i+2*16,1)=p_conc(i,1)+2*16;
622         l_conc(i+2*16,2)=p_conc(i,2)+2*16;
623         l_conc(i+3*16,1)=p_conc(i,1)+3*16;
624         l_conc(i+3*16,2)=p_conc(i,2)+3*16;
625     end
626     for i=1:51
627         l_conc(i+64,1)=j_conc(i,1);
628         l_conc(i+64+51,1)=j_conc(i,1)+51;
629         l_conc(i+64,2)=j_conc(i,2);
630         l_conc(i+64+51,2)=j_conc(i,2)+51;
631     end
632     for i=1:24
633         l_conc(i+166,1)=s_conc(i,1);
634         l_conc(i+166,2)=s_conc(i,2);
635     end
636     for i=1:60
637         l_conc(i+190,1)=b_conc(i,1);
638         l_conc(i+190,2)=b_conc(i,2);
639         l_conc(i+190+1*60,1)=b_conc(i,1)+1*50;
640         l_conc(i+190+1*60,2)=b_conc(i,2)+1*50;
641         l_conc(i+190+2*60,1)=b_conc(i,1)+2*50;
642         l_conc(i+190+2*60,2)=b_conc(i,2)+2*50;
643         l_conc(i+190+3*60,1)=b_conc(i,1)+3*50;
644         l_conc(i+190+3*60,2)=b_conc(i,2)+3*50;
645     end
646     for j=1:4
647         for i=1:5
648             l_conc(191+60*(j-1)+12*(i-1),1)=2+3*(i-1)+16*(j-1);
649             l_conc(190+60*(j-1)+12*i,2)=3+3*(i-1)+16*(j-1);
650         end
651     end
652     for i=1:2
653         l_conc(196+60*(i-1),2)=65+51*(i-1);
654         l_conc(197+60*(i-1),1)=73+51*(i-1);
```

```
655         l_conc(196+60*(i-1)+1*12,2)=74+51*(i-1);
656         l_conc(197+60*(i-1)+1*12,1)=85+51*(i-1);
657         l_conc(196+60*(i-1)+2*12,2)=86+51*(i-1);
658         l_conc(197+60*(i-1)+2*12,1)=97+51*(i-1);
659         l_conc(196+60*(i-1)+3*12,2)=98+51*(i-1);
660         l_conc(197+60*(i-1)+3*12,1)=115+51*(i-1);
661     end
662     for i=1:2
663         l_conc(196+60*(i+1),2)=67+51*(i-1);
664         l_conc(197+60*(i+1),1)=68+51*(i-1);
665         l_conc(196+60*(i+1)+1*12,2)=79+51*(i-1);
666         l_conc(197+60*(i+1)+1*12,1)=80+51*(i-1);
667         l_conc(196+60*(i+1)+2*12,2)=91+51*(i-1);
668         l_conc(197+60*(i+1)+2*12,1)=92+51*(i-1);
669         l_conc(196+60*(i+1)+3*12,2)=103+51*(i-1);
670         l_conc(197+60*(i+1)+3*12,1)=104+51*(i-1);
671     end
672 l_conc(196+4*12,2)=169;
673 l_conc(197+4*12,1)=168;
674 l_conc(196+4*12+1*60,2)=177;
675 l_conc(197+4*12+1*60,1)=178;
676 l_conc(196+4*12+2*60,2)=181;
677 l_conc(197+4*12+2*60,1)=180;
678 l_conc(196+4*12+3*60,2)=189;
679 l_conc(197+4*12+3*60,1)=190;
680     for i=1:22
681         l_conc(i+430,1)=c_conc(i,1);
682         l_conc(i+430,2)=c_conc(i,2);
683     end
684     for i=1:11
685         l_conc(i+452,1)=d_conc(i,1);
686         l_conc(i+452,2)=d_conc(i,2);
687     end
688     for i=1:22
```

```
689         l_conc(i+463,1)=f_conc(i,1);
690         l_conc(i+463,2)=f_conc(i,2);
691     end
692     for i=1:2
693         l_conc(i+485,1)=t_conc(i,1);
694         l_conc(i+485,2)=t_conc(i,2);
695     end
696    for i=1:12
697         l_conc(i+487,1)=k_conc(i,1);
698         l_conc(i+487,2)=k_conc(i,2);
699     end
700
701
702 % Z Coordinate Generation
703 % Parabolic
704 for b=1:size(l_coor,1)-24
705     if abs(l_coor(b,2))<l2
706         l_coor(b,3)=-m1*(1-(l_coor(b,2)/l2)*(l_coor(b,2)/l2));
707     else
708         l_coor(b,3)=m1*(abs(l_coor(b,2))-l2)/l2*2;
709     end
710 end
711 slp=(-m1*(1-(4/l2)*(4/l2))-(-m1*(1-(2/l2)*(2/l2))))/2;
712 for b=size(l_coor,1)-24+1:size(l_coor,1)-18
713         l_coor(b,3)=-m1*(1-(4/l2)*(4/l2))+slp*(l_coor(b,2)-4);
714 end
715 for b=size(l_coor,1)-18+1:size(l_coor,1)-12
716         l_coor(b,3)=-m1*(1-(4/l2)*(4/l2))+slp*(abs(l_coor(b,2))
              -4);
717 end
718 for b=size(l_coor,1)-12+1:size(l_coor,1)
719         l_coor(b,3)=-m1*(1-(6/l2)*(6/l2));
720 end
```

```
# -*- coding: mbcs -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
import visualization
import os
import datetime
import shutil
import time
from abaqus import *
from abaqusConstants import *
from caeModules import *

from numpy import*
xcoord = genfromtxt('xcoord.txt');
ycoord = genfromtxt('ycoord.txt');
zcoord = genfromtxt('zcoord.txt');
inipoint=genfromtxt('inipoint.txt');
endpoint=genfromtxt('endpoint.txt');
h=genfromtxt('h.txt');
b=genfromtxt('b.txt');
e=genfromtxt('e.txt');
g=genfromtxt('g.txt');
ms=genfromtxt('ms.txt');
```

```
35 mn=genfromtxt('mn.txt');
36
37 mdb.models.changeKey(fromName='Model-1', toName='CFRP Bridge')
38
39 mdb.models['CFRP Bridge'].Part(dimensionality=THREE_D, name='
       CFRP Bridge',
40     type=DEFORMABLE_BODY)
41
42 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].ReferencePoint(
       point=(0.0, 0.0,
43     0.0))
44 for i in range(0, 390):
45     mdb.models['CFRP Bridge'].parts['CFRP Bridge']
           .DatumPointByOffset(point=
46     mdb.models['CFRP Bridge'].parts['CFRP Bridge']
           .referencePoints[1], vector=(
47     xcoord[i], ycoord[i], zcoord[i]))
48 for i in range(0, 452):
49     ini=int(inipoint[i])+1
50     end=int(endpoint[i])+1
51     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].WirePolyLine
           (mergeWire=OFF,
52     meshable=ON, points=((
53     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].datums[ini],
54     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].datums[end])
           , ))
55
56 # Create Shell: Supports
57 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
       edgeList=
58     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
           (((xcoord[0]+(xcoord[1]-xcoord[0])/2, ycoord[0]+(ycoord
           [1]-ycoord[0])/2, zcoord[0]+(zcoord[1]-zcoord[0])/2), ),
            ((xcoord[1]+(xcoord[2]-xcoord[1])/2, ycoord[1]+(ycoord
```

```
        [2]-ycoord[1])/2, zcoord[1]+(zcoord[2]-zcoord[1])/2), ),
         ((xcoord[2]+(xcoord[3]-xcoord[2])/2, ycoord[2]+(ycoord
        [3]-ycoord[2])/2, zcoord[2]+(zcoord[3]-zcoord[2])/2), ),
         ((xcoord[3]+(xcoord[4]-xcoord[3])/2, ycoord[3]+(ycoord
        [4]-ycoord[3])/2, zcoord[3]+(zcoord[4]-zcoord[3])/2), ),
         ((xcoord[4]+(xcoord[5]-xcoord[4])/2, ycoord[4]+(ycoord
        [5]-ycoord[4])/2, zcoord[4]+(zcoord[5]-zcoord[4])/2), ),
         ((xcoord[5]+(xcoord[6]-xcoord[5])/2, ycoord[5]+(ycoord
        [6]-ycoord[5])/2, zcoord[5]+(zcoord[6]-zcoord[5])/2), ),
         ((xcoord[6]+(xcoord[7]-xcoord[6])/2, ycoord[6]+(ycoord
        [7]-ycoord[6])/2, zcoord[6]+(zcoord[7]-zcoord[6])/2), ),
         ((xcoord[7]+(xcoord[8]-xcoord[7])/2, ycoord[7]+(ycoord
        [8]-ycoord[7])/2, zcoord[7]+(zcoord[8]-zcoord[7])/2), ),
         ((xcoord[8]+(xcoord[9]-xcoord[8])/2, ycoord[8]+(ycoord
        [9]-ycoord[8])/2, zcoord[8]+(zcoord[9]-zcoord[8])/2), ),
         ((xcoord[9]+(xcoord[10]-xcoord[9])/2, ycoord[9]+(ycoord
        [10]-ycoord[9])/2, zcoord[9]+(zcoord[10]-zcoord[9])/2),
        ), ((xcoord[10]+(xcoord[11]-xcoord[10])/2, ycoord[10]+(
        ycoord[11]-ycoord[10])/2, zcoord[10]+(zcoord[11]-zcoord
        [10])/2), ), ((xcoord[11]+(xcoord[12]-xcoord[11])/2,
        ycoord[11]+(ycoord[12]-ycoord[11])/2, zcoord[11]+(zcoord
        [12]-zcoord[11])/2), ), ((xcoord[12]+(xcoord[13]-xcoord
        [12])/2, ycoord[12]+(ycoord[13]-ycoord[12])/2, zcoord
        [12]+(zcoord[13]-zcoord[12])/2), ), ((xcoord[13]+(xcoord
        [14]-xcoord[13])/2, ycoord[13]+(ycoord[14]-ycoord[13])
        /2, zcoord[13]+(zcoord[14]-zcoord[13])/2), ), ((xcoord
        [14]+(xcoord[15]-xcoord[14])/2, ycoord[14]+(ycoord[15]-
        ycoord[14])/2, zcoord[14]+(zcoord[15]-zcoord[14])/2), ),
         ((xcoord[15]+(xcoord[0]-xcoord[15])/2, ycoord[15]+(
        ycoord[0]-ycoord[15])/2, zcoord[15]+(zcoord[0]-zcoord
        [15])/2), ), ),
59      tryAnalytical=False)
60  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
    edgeList=
```

```
61      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            (((xcoord[16]+(xcoord[17]-xcoord[16])/2, ycoord[16]+(
            ycoord[17]-ycoord[16])/2, zcoord[16]+(zcoord[17]-zcoord
            [16])/2), ), ((xcoord[17]+(xcoord[18]-xcoord[17])/2,
            ycoord[17]+(ycoord[18]-ycoord[17])/2, zcoord[17]+(zcoord
            [18]-zcoord[17])/2), ), ((xcoord[18]+(xcoord[19]-xcoord
            [18])/2, ycoord[18]+(ycoord[19]-ycoord[18])/2, zcoord
            [18]+(zcoord[19]-zcoord[18])/2), ), ((xcoord[19]+(xcoord
            [20]-xcoord[19])/2, ycoord[19]+(ycoord[20]-ycoord[19])
            /2, zcoord[19]+(zcoord[20]-zcoord[19])/2), ), ((xcoord
            [20]+(xcoord[21]-xcoord[20])/2, ycoord[20]+(ycoord[21]-
            ycoord[20])/2, zcoord[20]+(zcoord[21]-zcoord[20])/2), ),
             ((xcoord[21]+(xcoord[22]-xcoord[21])/2, ycoord[21]+(
            ycoord[22]-ycoord[21])/2, zcoord[21]+(zcoord[22]-zcoord
            [21])/2), ), ((xcoord[22]+(xcoord[23]-xcoord[22])/2,
            ycoord[22]+(ycoord[23]-ycoord[22])/2, zcoord[22]+(zcoord
            [23]-zcoord[22])/2), ), ((xcoord[23]+(xcoord[24]-xcoord
            [23])/2, ycoord[23]+(ycoord[24]-ycoord[23])/2, zcoord
            [23]+(zcoord[24]-zcoord[23])/2), ), ((xcoord[24]+(xcoord
            [25]-xcoord[24])/2, ycoord[24]+(ycoord[25]-ycoord[24])
            /2, zcoord[24]+(zcoord[25]-zcoord[24])/2), ), ((xcoord
            [25]+(xcoord[26]-xcoord[25])/2, ycoord[25]+(ycoord[26]-
            ycoord[25])/2, zcoord[25]+(zcoord[26]-zcoord[25])/2), ),
             ((xcoord[26]+(xcoord[27]-xcoord[26])/2, ycoord[26]+(
            ycoord[27]-ycoord[26])/2, zcoord[26]+(zcoord[27]-zcoord
            [26])/2), ), ((xcoord[27]+(xcoord[28]-xcoord[27])/2,
            ycoord[27]+(ycoord[28]-ycoord[27])/2, zcoord[27]+(zcoord
            [28]-zcoord[27])/2), ), ((xcoord[28]+(xcoord[29]-xcoord
            [28])/2, ycoord[28]+(ycoord[29]-ycoord[28])/2, zcoord
            [28]+(zcoord[29]-zcoord[28])/2), ), ((xcoord[29]+(xcoord
            [30]-xcoord[29])/2, ycoord[29]+(ycoord[30]-ycoord[29])
            /2, zcoord[29]+(zcoord[30]-zcoord[29])/2), ), ((xcoord
            [30]+(xcoord[31]-xcoord[30])/2, ycoord[30]+(ycoord[31]-
            ycoord[30])/2, zcoord[30]+(zcoord[31]-zcoord[30])/2), ),
```

```
        ((xcoord[31]+(xcoord[16]-xcoord[31])/2, ycoord[31]+(
        ycoord[16]-ycoord[31])/2, zcoord[31]+(zcoord[16]-zcoord
        [31])/2), ), ),
62      tryAnalytical=False)
63  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
    edgeList=
64      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[32]+(xcoord[33]-xcoord[32])/2, ycoord[32]+(
        ycoord[33]-ycoord[32])/2, zcoord[32]+(zcoord[33]-zcoord
        [32])/2), ), ((xcoord[33]+(xcoord[34]-xcoord[33])/2,
        ycoord[33]+(ycoord[34]-ycoord[33])/2, zcoord[33]+(zcoord
        [34]-zcoord[33])/2), ), ((xcoord[34]+(xcoord[35]-xcoord
        [34])/2, ycoord[34]+(ycoord[35]-ycoord[34])/2, zcoord
        [34]+(zcoord[35]-zcoord[34])/2), ), ((xcoord[35]+(xcoord
        [36]-xcoord[35])/2, ycoord[35]+(ycoord[36]-ycoord[35])
        /2, zcoord[35]+(zcoord[36]-zcoord[35])/2), ), ((xcoord
        [36]+(xcoord[37]-xcoord[36])/2, ycoord[36]+(ycoord[37]-
        ycoord[36])/2, zcoord[36]+(zcoord[37]-zcoord[36])/2), ),
        ((xcoord[37]+(xcoord[38]-xcoord[37])/2, ycoord[37]+(
        ycoord[38]-ycoord[37])/2, zcoord[37]+(zcoord[38]-zcoord
        [37])/2), ), ((xcoord[38]+(xcoord[39]-xcoord[38])/2,
        ycoord[38]+(ycoord[39]-ycoord[38])/2, zcoord[38]+(zcoord
        [39]-zcoord[38])/2), ), ((xcoord[39]+(xcoord[40]-xcoord
        [39])/2, ycoord[39]+(ycoord[40]-ycoord[39])/2, zcoord
        [39]+(zcoord[40]-zcoord[39])/2), ), ((xcoord[40]+(xcoord
        [41]-xcoord[40])/2, ycoord[40]+(ycoord[41]-ycoord[40])
        /2, zcoord[40]+(zcoord[41]-zcoord[40])/2), ), ((xcoord
        [41]+(xcoord[42]-xcoord[41])/2, ycoord[41]+(ycoord[42]-
        ycoord[41])/2, zcoord[41]+(zcoord[42]-zcoord[41])/2), ),
        ((xcoord[42]+(xcoord[43]-xcoord[42])/2, ycoord[42]+(
        ycoord[43]-ycoord[42])/2, zcoord[42]+(zcoord[43]-zcoord
        [42])/2), ), ((xcoord[43]+(xcoord[44]-xcoord[43])/2,
        ycoord[43]+(ycoord[44]-ycoord[43])/2, zcoord[43]+(zcoord
        [44]-zcoord[43])/2), ), ((xcoord[44]+(xcoord[45]-xcoord
```

```
        [44])/2, ycoord[44]+(ycoord[45]-ycoord[44])/2, zcoord
        [44]+(zcoord[45]-zcoord[44])/2), ), ((xcoord[45]+(xcoord
        [46]-xcoord[45])/2, ycoord[45]+(ycoord[46]-ycoord[45])
        /2, zcoord[45]+(zcoord[46]-zcoord[45])/2), ), ((xcoord
        [46]+(xcoord[47]-xcoord[46])/2, ycoord[46]+(ycoord[47]-
        ycoord[46])/2, zcoord[46]+(zcoord[47]-zcoord[46])/2), ),
         ((xcoord[47]+(xcoord[32]-xcoord[47])/2, ycoord[47]+(
        ycoord[32]-ycoord[47])/2, zcoord[47]+(zcoord[32]-zcoord
        [47])/2), ), ),
65      tryAnalytical=False)
66  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
        edgeList=
67      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[48]+(xcoord[49]-xcoord[48])/2, ycoord[48]+(
        ycoord[49]-ycoord[48])/2, zcoord[48]+(zcoord[49]-zcoord
        [48])/2), ), ((xcoord[49]+(xcoord[50]-xcoord[49])/2,
        ycoord[49]+(ycoord[50]-ycoord[49])/2, zcoord[49]+(zcoord
        [50]-zcoord[49])/2), ), ((xcoord[50]+(xcoord[51]-xcoord
        [50])/2, ycoord[50]+(ycoord[51]-ycoord[50])/2, zcoord
        [50]+(zcoord[51]-zcoord[50])/2), ), ((xcoord[51]+(xcoord
        [52]-xcoord[51])/2, ycoord[51]+(ycoord[52]-ycoord[51])
        /2, zcoord[51]+(zcoord[52]-zcoord[51])/2), ), ((xcoord
        [52]+(xcoord[53]-xcoord[52])/2, ycoord[52]+(ycoord[53]-
        ycoord[52])/2, zcoord[52]+(zcoord[53]-zcoord[52])/2), ),
         ((xcoord[53]+(xcoord[54]-xcoord[53])/2, ycoord[53]+(
        ycoord[54]-ycoord[53])/2, zcoord[53]+(zcoord[54]-zcoord
        [53])/2), ), ((xcoord[54]+(xcoord[55]-xcoord[54])/2,
        ycoord[54]+(ycoord[55]-ycoord[54])/2, zcoord[54]+(zcoord
        [55]-zcoord[54])/2), ), ((xcoord[55]+(xcoord[56]-xcoord
        [55])/2, ycoord[55]+(ycoord[56]-ycoord[55])/2, zcoord
        [55]+(zcoord[56]-zcoord[55])/2), ), ((xcoord[56]+(xcoord
        [57]-xcoord[56])/2, ycoord[56]+(ycoord[57]-ycoord[56])
        /2, zcoord[56]+(zcoord[57]-zcoord[56])/2), ), ((xcoord
        [57]+(xcoord[58]-xcoord[57])/2, ycoord[57]+(ycoord[58]-
```

```
        ycoord[57])/2, zcoord[57]+(zcoord[58]-zcoord[57])/2), ),
         ((xcoord[58]+(xcoord[59]-xcoord[58])/2, ycoord[58]+(
        ycoord[59]-ycoord[58])/2, zcoord[58]+(zcoord[59]-zcoord
        [58])/2), ), ((xcoord[59]+(xcoord[60]-xcoord[59])/2,
        ycoord[59]+(ycoord[60]-ycoord[59])/2, zcoord[59]+(zcoord
        [60]-zcoord[59])/2), ), ((xcoord[60]+(xcoord[61]-xcoord
        [60])/2, ycoord[60]+(ycoord[61]-ycoord[60])/2, zcoord
        [60]+(zcoord[61]-zcoord[60])/2), ), ((xcoord[61]+(xcoord
        [62]-xcoord[61])/2, ycoord[61]+(ycoord[62]-ycoord[61])
        /2, zcoord[61]+(zcoord[62]-zcoord[61])/2), ), ((xcoord
        [62]+(xcoord[63]-xcoord[62])/2, ycoord[62]+(ycoord[63]-
        ycoord[62])/2, zcoord[62]+(zcoord[63]-zcoord[62])/2), ),
         ((xcoord[63]+(xcoord[48]-xcoord[63])/2, ycoord[63]+(
        ycoord[48]-ycoord[63])/2, zcoord[63]+(zcoord[48]-zcoord
        [63])/2), ), ),
68      tryAnalytical=False)
69
70  # Create Shell: Joints
71  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
        edgeList=
72      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            (((xcoord[64]+(xcoord[65]-xcoord[64])/2, ycoord[64]+(
            ycoord[65]-ycoord[64])/2, zcoord[64]+(zcoord[65]-zcoord
            [64])/2), ), ((xcoord[65]+(xcoord[66]-xcoord[65])/2,
            ycoord[65]+(ycoord[66]-ycoord[65])/2, zcoord[65]+(zcoord
            [66]-zcoord[65])/2), ), ((xcoord[66]+(xcoord[67]-xcoord
            [66])/2, ycoord[66]+(ycoord[67]-ycoord[66])/2, zcoord
            [66]+(zcoord[67]-zcoord[66])/2), ), ((xcoord[67]+(xcoord
            [68]-xcoord[67])/2, ycoord[67]+(ycoord[68]-ycoord[67])
            /2, zcoord[67]+(zcoord[68]-zcoord[67])/2), ), ((xcoord
            [68]+(xcoord[69]-xcoord[68])/2, ycoord[68]+(ycoord[69]-
            ycoord[68])/2, zcoord[68]+(zcoord[69]-zcoord[68])/2), ),
             ((xcoord[69]+(xcoord[70]-xcoord[69])/2, ycoord[69]+(
            ycoord[70]-ycoord[69])/2, zcoord[69]+(zcoord[70]-zcoord
```

```
        [69])/2), ), ((xcoord[70]+(xcoord[71]-xcoord[70])/2,
        ycoord[70]+(ycoord[71]-ycoord[70])/2, zcoord[70]+(zcoord
        [71]-zcoord[70])/2), ), ((xcoord[71]+(xcoord[72]-xcoord
        [71])/2, ycoord[71]+(ycoord[72]-ycoord[71])/2, zcoord
        [71]+(zcoord[72]-zcoord[71])/2), ), ((xcoord[72]+(xcoord
        [64]-xcoord[72])/2, ycoord[72]+(ycoord[64]-ycoord[72])
        /2, zcoord[72]+(zcoord[64]-zcoord[72])/2), ),
73      ), tryAnalytical=False)
74  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
        edgeList=
75      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[73]+(xcoord[74]-xcoord[73])/2, ycoord[73]+(
        ycoord[74]-ycoord[73])/2, zcoord[73]+(zcoord[74]-zcoord
        [73])/2), ), ((xcoord[74]+(xcoord[75]-xcoord[74])/2,
        ycoord[74]+(ycoord[75]-ycoord[74])/2, zcoord[74]+(zcoord
        [75]-zcoord[74])/2), ), ((xcoord[75]+(xcoord[76]-xcoord
        [75])/2, ycoord[75]+(ycoord[76]-ycoord[75])/2, zcoord
        [75]+(zcoord[76]-zcoord[75])/2), ), ((xcoord[76]+(xcoord
        [77]-xcoord[76])/2, ycoord[76]+(ycoord[77]-ycoord[76])
        /2, zcoord[76]+(zcoord[77]-zcoord[76])/2), ), ((xcoord
        [77]+(xcoord[78]-xcoord[77])/2, ycoord[77]+(ycoord[78]-
        ycoord[77])/2, zcoord[77]+(zcoord[78]-zcoord[77])/2), ),
76      ((xcoord[78]+(xcoord[79]-xcoord[78])/2, ycoord[78]+(ycoord
        [79]-ycoord[78])/2, zcoord[78]+(zcoord[79]-zcoord[78])
        /2), ), ((xcoord[79]+(xcoord[80]-xcoord[79])/2, ycoord
        [79]+(ycoord[80]-ycoord[79])/2, zcoord[79]+(zcoord[80]-
        zcoord[79])/2), ), ((xcoord[80]+(xcoord[81]-xcoord[80])
        /2, ycoord[80]+(ycoord[81]-ycoord[80])/2, zcoord[80]+(
        zcoord[81]-zcoord[80])/2), ), ((xcoord[81]+(xcoord[82]-
        xcoord[81])/2, ycoord[81]+(ycoord[82]-ycoord[81])/2,
        zcoord[81]+(zcoord[82]-zcoord[81])/2), ), ((xcoord[82]+(
        xcoord[83]-xcoord[82])/2, ycoord[82]+(ycoord[83]-ycoord
        [82])/2, zcoord[82]+(zcoord[83]-zcoord[82])/2), ), ((
        xcoord[83]+(xcoord[84]-xcoord[83])/2, ycoord[83]+(ycoord
```

```
        [84]-ycoord[83])/2, zcoord[83]+(zcoord[84]-zcoord[83])
        /2), ), ((xcoord[84]+(xcoord[73]-xcoord[84])/2, ycoord
        [84]+(ycoord[73]-ycoord[84])/2, zcoord[84]+(zcoord[73]-
        zcoord[84])/2),
77      ), ), tryAnalytical=False)
78  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
    edgeList=
79      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[85]+(xcoord[86]-xcoord[85])/2, ycoord[85]+(
        ycoord[86]-ycoord[85])/2, zcoord[85]+(zcoord[86]-zcoord
        [85])/2), ), ((xcoord[86]+(xcoord[87]-xcoord[86])/2,
        ycoord[86]+(ycoord[87]-ycoord[86])/2, zcoord[86]+(zcoord
        [87]-zcoord[86])/2), ), ((xcoord[87]+(xcoord[88]-xcoord
        [87])/2, ycoord[87]+(ycoord[88]-ycoord[87])/2, zcoord
        [87]+(zcoord[88]-zcoord[87])/2), ), ((xcoord[88]+(xcoord
        [89]-xcoord[88])/2, ycoord[88]+(ycoord[89]-ycoord[88])
        /2, zcoord[88]+(zcoord[89]-zcoord[88])/2), ), ((xcoord
        [89]+(xcoord[90]-xcoord[89])/2, ycoord[89]+(ycoord[90]-
        ycoord[89])/2, zcoord[89]+(zcoord[90]-zcoord[89])/2), ),
80      ((xcoord[90]+(xcoord[91]-xcoord[90])/2, ycoord[90]+(ycoord
        [91]-ycoord[90])/2, zcoord[90]+(zcoord[91]-zcoord[90])
        /2), ), ((xcoord[91]+(xcoord[92]-xcoord[91])/2, ycoord
        [91]+(ycoord[92]-ycoord[91])/2, zcoord[91]+(zcoord[92]-
        zcoord[91])/2), ), ((xcoord[92]+(xcoord[93]-xcoord[92])
        /2, ycoord[92]+(ycoord[93]-ycoord[92])/2, zcoord[92]+(
        zcoord[93]-zcoord[92])/2), ), ((xcoord[93]+(xcoord[94]-
        xcoord[93])/2, ycoord[93]+(ycoord[94]-ycoord[93])/2,
        zcoord[93]+(zcoord[94]-zcoord[93])/2), ), ((xcoord[94]+(
        xcoord[95]-xcoord[94])/2, ycoord[94]+(ycoord[95]-ycoord
        [94])/2, zcoord[94]+(zcoord[95]-zcoord[94])/2), ), ((
        xcoord[95]+(xcoord[96]-xcoord[95])/2, ycoord[95]+(ycoord
        [96]-ycoord[95])/2, zcoord[95]+(zcoord[96]-zcoord[95])
        /2), ), ((xcoord[96]+(xcoord[85]-xcoord[96])/2, ycoord
        [96]+(ycoord[85]-ycoord[96])/2, zcoord[96]+(zcoord[85]-
```

```
        zcoord[96])/2),
81      ), ), tryAnalytical=False)
82  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
        edgeList=
83      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            (((xcoord[97]+(xcoord[98]-xcoord[97])/2, ycoord[97]+(
            ycoord[98]-ycoord[97])/2, zcoord[97]+(zcoord[98]-zcoord
            [97])/2), ), ((xcoord[98]+(xcoord[99]-xcoord[98])/2,
            ycoord[98]+(ycoord[99]-ycoord[98])/2, zcoord[98]+(zcoord
            [99]-zcoord[98])/2), ), ((xcoord[99]+(xcoord[100]-xcoord
            [99])/2, ycoord[99]+(ycoord[100]-ycoord[99])/2, zcoord
            [99]+(zcoord[100]-zcoord[99])/2), ), ((xcoord[100]+(
            xcoord[101]-xcoord[100])/2, ycoord[100]+(ycoord[101]-
            ycoord[100])/2, zcoord[100]+(zcoord[101]-zcoord[100])/2)
            , ), ((xcoord[101]+(xcoord[102]-xcoord[101])/2, ycoord
            [101]+(ycoord[102]-ycoord[101])/2, zcoord[101]+(zcoord
            [102]-zcoord[101])/2), ), (
84      (xcoord[102]+(xcoord[103]-xcoord[102])/2, ycoord[102]+(
            ycoord[103]-ycoord[102])/2, zcoord[102]+(zcoord[103]-
            zcoord[102])/2), ), ((xcoord[103]+(xcoord[104]-xcoord
            [103])/2, ycoord[103]+(ycoord[104]-ycoord[103])/2,
            zcoord[103]+(zcoord[104]-zcoord[103])/2), ), ((
85      xcoord[104]+(xcoord[105]-xcoord[104])/2, ycoord[104]+(
            ycoord[105]-ycoord[104])/2, zcoord[104]+(zcoord[105]-
            zcoord[104])/2), ), ((xcoord[105]+(xcoord[106]-xcoord
            [105])/2, ycoord[105]+(ycoord[106]-ycoord[105])/2,
            zcoord[105]+(zcoord[106]-zcoord[105])/2), ), ((
86      xcoord[106]+(xcoord[107]-xcoord[106])/2, ycoord[106]+(
            ycoord[107]-ycoord[106])/2, zcoord[106]+(zcoord[107]-
            zcoord[106])/2), ), ((xcoord[107]+(xcoord[108]-xcoord
            [107])/2, ycoord[107]+(ycoord[108]-ycoord[107])/2,
            zcoord[107]+(zcoord[108]-zcoord[107])/2), ), ((xcoord
            [108]+(xcoord[109]-xcoord[108])/2, ycoord[108]+(ycoord
            [109]-ycoord[108])/2, zcoord[108]+(zcoord[109]-zcoord
```

```
        [108])/2), ), ((xcoord[109]+(xcoord[110]-xcoord[109])/2,
         ycoord[109]+(ycoord[110]-ycoord[109])/2, zcoord[109]+(
        zcoord[110]-zcoord[109])/2), ), ((xcoord[110]+(xcoord
        [111]-xcoord[110])/2, ycoord[110]+(ycoord[111]-ycoord
        [110])/2, zcoord[110]+(zcoord[111]-zcoord[110])/2), ),
        ((xcoord[111]+(xcoord[112]-xcoord[111])/2, ycoord[111]+(
        ycoord[112]-ycoord[111])/2, zcoord[111]+(zcoord[112]-
        zcoord[111])/2), ), ((xcoord[112]+(xcoord[113]-xcoord
        [112])/2, ycoord[112]+(ycoord[113]-ycoord[112])/2,
        zcoord[112]+(zcoord[113]-zcoord[112])/2), ), ((xcoord
        [113]+(xcoord[114]-xcoord[113])/2, ycoord[113]+(ycoord
        [114]-ycoord[113])/2, zcoord[113]+(zcoord[114]-zcoord
        [113])/2), ), ((xcoord[114]+(xcoord[97]-xcoord[114])/2,
        ycoord[114]+(ycoord[97]-ycoord[114])/2, zcoord[114]+(
        zcoord[97]-zcoord[114])/2), ), ), tryAnalytical=False)
87 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
    edgeList=
88  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[115]+(xcoord[116]-xcoord[115])/2, ycoord
        [115]+(ycoord[116]-ycoord[115])/2, zcoord[115]+(zcoord
        [116]-zcoord[115])/2), ), ((xcoord[116]+(xcoord[117]-
        xcoord[116])/2, ycoord[116]+(ycoord[117]-ycoord[116])/2,
         zcoord[116]+(zcoord[117]-zcoord[116])/2), ), ((xcoord
        [117]+(xcoord[118]-xcoord[117])/2, ycoord[117]+(ycoord
        [118]-ycoord[117])/2, zcoord[117]+(zcoord[118]-zcoord
        [117])/2), ), ((xcoord[118]+(xcoord[119]-xcoord[118])/2,
         ycoord[118]+(ycoord[119]-ycoord[118])/2, zcoord[118]+(
        zcoord[119]-zcoord[118])/2), ), ((xcoord[119]+(xcoord
        [120]-xcoord[119])/2, ycoord[119]+(ycoord[120]-ycoord
        [119])/2, zcoord[119]+(zcoord[120]-zcoord[119])/2), ),
        ((xcoord[120]+(xcoord[121]-xcoord[120])/2, ycoord[120]+(
        ycoord[121]-ycoord[120])/2, zcoord[120]+(zcoord[121]-
        zcoord[120])/2), ), ((xcoord[121]+(xcoord[122]-xcoord
        [121])/2, ycoord[121]+(ycoord[122]-ycoord[121])/2,
```

```
        zcoord[121]+(zcoord[122]-zcoord[121])/2), ), ((xcoord
        [122]+(xcoord[123]-xcoord[122])/2, ycoord[122]+(ycoord
        [123]-ycoord[122])/2, zcoord[122]+(zcoord[123]-zcoord
        [122])/2), ), ((xcoord[123]+(xcoord[115]-xcoord[123])/2,
         ycoord[123]+(ycoord[115]-ycoord[123])/2, zcoord[123]+(
        zcoord[115]-zcoord[123])/2), ),
89      ), tryAnalytical=False)
90  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
        edgeList=
91      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[124]+(xcoord[125]-xcoord[124])/2, ycoord
        [124]+(ycoord[125]-ycoord[124])/2, zcoord[124]+(zcoord
        [125]-zcoord[124])/2), ), ((xcoord[125]+(xcoord[126]-
        xcoord[125])/2, ycoord[125]+(ycoord[126]-ycoord[125])/2,
         zcoord[125]+(zcoord[126]-zcoord[125])/2), ), ((xcoord
        [126]+(xcoord[127]-xcoord[126])/2, ycoord[126]+(ycoord
        [127]-ycoord[126])/2, zcoord[126]+(zcoord[127]-zcoord
        [126])/2), ), ((xcoord[127]+(xcoord[128]-xcoord[127])/2,
         ycoord[127]+(ycoord[128]-ycoord[127])/2, zcoord[127]+(
        zcoord[128]-zcoord[127])/2), ), ((xcoord[128]+(xcoord
        [129]-xcoord[128])/2, ycoord[128]+(ycoord[129]-ycoord
        [128])/2, zcoord[128]+(zcoord[129]-zcoord[128])/2), ),
92      ((xcoord[129]+(xcoord[130]-xcoord[129])/2, ycoord[129]+(
        ycoord[130]-ycoord[129])/2, zcoord[129]+(zcoord[130]-
        zcoord[129])/2), ), ((xcoord[130]+(xcoord[131]-xcoord
        [130])/2, ycoord[130]+(ycoord[131]-ycoord[130])/2,
        zcoord[130]+(zcoord[131]-zcoord[130])/2), ), ((xcoord
        [131]+(xcoord[132]-xcoord[131])/2, ycoord[131]+(ycoord
        [132]-ycoord[131])/2, zcoord[131]+(zcoord[132]-zcoord
        [131])/2), ), ((xcoord[132]+(xcoord[133]-xcoord[132])/2,
         ycoord[132]+(ycoord[133]-ycoord[132])/2, zcoord[132]+(
        zcoord[133]-zcoord[132])/2), ), ((xcoord[133]+(xcoord
        [134]-xcoord[133])/2, ycoord[133]+(ycoord[134]-ycoord
        [133])/2, zcoord[133]+(zcoord[134]-zcoord[133])/2), ),
```

```
        ((xcoord[134]+(xcoord[135]-xcoord[134])/2, ycoord[134]+(
        ycoord[135]-ycoord[134])/2, zcoord[134]+(zcoord[135]-
        zcoord[134])/2), ), ((xcoord[135]+(xcoord[124]-xcoord
        [135])/2, ycoord[135]+(ycoord[124]-ycoord[135])/2,
        zcoord[135]+(zcoord[124]-zcoord[135])/2),
93      ), ), tryAnalytical=False)
94  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
        edgeList=
95      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[136]+(xcoord[137]-xcoord[136])/2, ycoord
        [136]+(ycoord[137]-ycoord[136])/2, zcoord[136]+(zcoord
        [137]-zcoord[136])/2), ), ((xcoord[137]+(xcoord[138]-
        xcoord[137])/2, ycoord[137]+(ycoord[138]-ycoord[137])/2,
         zcoord[137]+(zcoord[138]-zcoord[137])/2), ), ((xcoord
        [138]+(xcoord[139]-xcoord[138])/2, ycoord[138]+(ycoord
        [139]-ycoord[138])/2, zcoord[138]+(zcoord[139]-zcoord
        [138])/2), ), ((xcoord[139]+(xcoord[140]-xcoord[139])/2,
         ycoord[139]+(ycoord[140]-ycoord[139])/2, zcoord[139]+(
        zcoord[140]-zcoord[139])/2), ), ((xcoord[140]+(xcoord
        [141]-xcoord[140])/2, ycoord[140]+(ycoord[141]-ycoord
        [140])/2, zcoord[140]+(zcoord[141]-zcoord[140])/2), ),
96      ((xcoord[141]+(xcoord[142]-xcoord[141])/2, ycoord[141]+(
        ycoord[142]-ycoord[141])/2, zcoord[141]+(zcoord[142]-
        zcoord[141])/2), ), ((xcoord[142]+(xcoord[143]-xcoord
        [142])/2, ycoord[142]+(ycoord[143]-ycoord[142])/2,
        zcoord[142]+(zcoord[143]-zcoord[142])/2), ), ((xcoord
        [143]+(xcoord[144]-xcoord[143])/2, ycoord[143]+(ycoord
        [144]-ycoord[143])/2, zcoord[143]+(zcoord[144]-zcoord
        [143])/2), ), ((xcoord[144]+(xcoord[145]-xcoord[144])/2,
         ycoord[144]+(ycoord[145]-ycoord[144])/2, zcoord[144]+(
        zcoord[145]-zcoord[144])/2), ), ((xcoord[145]+(xcoord
        [146]-xcoord[145])/2, ycoord[145]+(ycoord[146]-ycoord
        [145])/2, zcoord[145]+(zcoord[146]-zcoord[145])/2), ),
        ((xcoord[146]+(xcoord[147]-xcoord[146])/2, ycoord[146]+(
```

```
        ycoord[147]-ycoord[146])/2, zcoord[146]+(zcoord[147]-
        zcoord[146])/2), ), ((xcoord[147]+(xcoord[136]-xcoord
        [147])/2, ycoord[147]+(ycoord[136]-ycoord[147])/2,
        zcoord[147]+(zcoord[136]-zcoord[147])/2),
97    ), ), tryAnalytical=False)
98 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
    edgeList=
99    mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[148]+(xcoord[149]-xcoord[148])/2, ycoord
        [148]+(ycoord[149]-ycoord[148])/2, zcoord[148]+(zcoord
        [149]-zcoord[148])/2), ), ((xcoord[149]+(xcoord[150]-
        xcoord[149])/2, ycoord[149]+(ycoord[150]-ycoord[149])/2,
         zcoord[149]+(zcoord[150]-zcoord[149])/2), ), ((xcoord
        [150]+(xcoord[151]-xcoord[150])/2, ycoord[150]+(ycoord
        [151]-ycoord[150])/2, zcoord[150]+(zcoord[151]-zcoord
        [150])/2), ), ((xcoord[151]+(xcoord[152]-xcoord[151])/2,
         ycoord[151]+(ycoord[152]-ycoord[151])/2, zcoord[151]+(
        zcoord[152]-zcoord[151])/2), ), ((xcoord[152]+(xcoord
        [153]-xcoord[152])/2, ycoord[152]+(ycoord[153]-ycoord
        [152])/2, zcoord[152]+(zcoord[153]-zcoord[152])/2), ), (
100   (xcoord[153]+(xcoord[154]-xcoord[153])/2, ycoord[153]+(
        ycoord[154]-ycoord[153])/2, zcoord[153]+(zcoord[154]-
        zcoord[153])/2), ), ((xcoord[154]+(xcoord[155]-xcoord
        [154])/2, ycoord[154]+(ycoord[155]-ycoord[154])/2,
        zcoord[154]+(zcoord[155]-zcoord[154])/2), ), ((
101   xcoord[155]+(xcoord[156]-xcoord[155])/2, ycoord[155]+(
        ycoord[156]-ycoord[155])/2, zcoord[155]+(zcoord[156]-
        zcoord[155])/2), ), ((xcoord[156]+(xcoord[157]-xcoord
        [156])/2, ycoord[156]+(ycoord[157]-ycoord[156])/2,
        zcoord[156]+(zcoord[157]-zcoord[156])/2), ), ((
102   xcoord[157]+(xcoord[158]-xcoord[157])/2, ycoord[157]+(
        ycoord[158]-ycoord[157])/2, zcoord[157]+(zcoord[158]-
        zcoord[157])/2), ), ((xcoord[158]+(xcoord[159]-xcoord
        [158])/2, ycoord[158]+(ycoord[159]-ycoord[158])/2,
```

```
        zcoord[158]+(zcoord[159]-zcoord[158])/2), ), ((xcoord
        [159]+(xcoord[160]-xcoord[159])/2, ycoord[159]+(ycoord
        [160]-ycoord[159])/2, zcoord[159]+(zcoord[160]-zcoord
        [159])/2), ), ((xcoord[160]+(xcoord[161]-xcoord[160])/2,
         ycoord[160]+(ycoord[161]-ycoord[160])/2, zcoord[160]+(
        zcoord[161]-zcoord[160])/2), ), ((xcoord[161]+(xcoord
        [162]-xcoord[161])/2, ycoord[161]+(ycoord[162]-ycoord
        [161])/2, zcoord[161]+(zcoord[162]-zcoord[161])/2), ),
        ((xcoord[162]+(xcoord[163]-xcoord[162])/2, ycoord[162]+(
        ycoord[163]-ycoord[162])/2, zcoord[162]+(zcoord[163]-
        zcoord[162])/2), ), ((xcoord[163]+(xcoord[164]-xcoord
        [163])/2, ycoord[163]+(ycoord[164]-ycoord[163])/2,
        zcoord[163]+(zcoord[164]-zcoord[163])/2), ), ((xcoord
        [164]+(xcoord[165]-xcoord[164])/2, ycoord[164]+(ycoord
        [165]-ycoord[164])/2, zcoord[164]+(zcoord[165]-zcoord
        [164])/2), ), ((xcoord[165]+(xcoord[148]-xcoord[165])/2,
         ycoord[165]+(ycoord[148]-ycoord[165])/2, zcoord[165]+(
        zcoord[148]-zcoord[165])/2), ), ), tryAnalytical=False)
103 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
    edgeList=
104  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((( xcoord[166]+(xcoord[167]-xcoord[166])/2, ycoord
        [166]+(ycoord[167]-ycoord[166])/2, zcoord[166]+(zcoord
        [167]-zcoord[166])/2), ), ((xcoord[167]+(xcoord[168]-
        xcoord[167])/2, ycoord[167]+(ycoord[168]-ycoord[167])/2,
         zcoord[167]+(zcoord[168]-zcoord[167])/2), ), ((xcoord
        [168]+(xcoord[169]-xcoord[168])/2, ycoord[168]+(ycoord
        [169]-ycoord[168])/2, zcoord[168]+(zcoord[169]-zcoord
        [168])/2), ), ((xcoord[169]+(xcoord[170]-xcoord[169])/2,
         ycoord[169]+(ycoord[170]-ycoord[169])/2, zcoord[169]+(
        zcoord[170]-zcoord[169])/2), ), ((xcoord[170]+(xcoord
        [171]-xcoord[170])/2, ycoord[170]+(ycoord[171]-ycoord
        [170])/2, zcoord[170]+(zcoord[171]-zcoord[170])/2),
```

```
105     ), ((xcoord[171]+(xcoord[172]-xcoord[171])/2, ycoord[171]+(
            ycoord[172]-ycoord[171])/2, zcoord[171]+(zcoord[172]-
            zcoord[171])/2), ), ((xcoord[172]+(xcoord[173]-xcoord
            [172])/2, ycoord[172]+(ycoord[173]-ycoord[172])/2,
            zcoord[172]+(zcoord[173]-zcoord[172])/2), ), ((xcoord
            [173]+(xcoord[174]-xcoord[173])/2, ycoord[173]+(ycoord
            [174]-ycoord[173])/2, zcoord[173]+(zcoord[174]-zcoord
            [173])/2), ), ((xcoord[174]+(xcoord[175]-xcoord[174])/2,
             ycoord[174]+(ycoord[175]-ycoord[174])/2, zcoord[174]+(
            zcoord[175]-zcoord[174])/2), ), ((xcoord[175]+(xcoord
            [176]-xcoord[175])/2, ycoord[175]+(ycoord[176]-ycoord
            [175])/2, zcoord[175]+(zcoord[176]-zcoord[175])/2), ),
            ((xcoord[176]+(xcoord[177]-xcoord[176])/2, ycoord[176]+(
            ycoord[177]-ycoord[176])/2, zcoord[176]+(zcoord[177]-
            zcoord[176])/2), ), ((xcoord[177]+(xcoord[166]-xcoord
            [177])/2, ycoord[177]+(ycoord[166]-ycoord[177])/2,
            zcoord[177]+(zcoord[166]-zcoord[177])/2), ), ),
            tryAnalytical=False)
106 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].CoverEdges(
        edgeList=
107     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((( xcoord[178]+(xcoord[179]-xcoord[178])/2, ycoord
            [178]+(ycoord[179]-ycoord[178])/2, zcoord[178]+(zcoord
            [179]-zcoord[178])/2), ), ((xcoord[179]+(xcoord[180]-
            xcoord[179])/2, ycoord[179]+(ycoord[180]-ycoord[179])/2,
             zcoord[179]+(zcoord[180]-zcoord[179])/2), ), ((xcoord
            [180]+(xcoord[181]-xcoord[180])/2, ycoord[180]+(ycoord
            [181]-ycoord[180])/2, zcoord[180]+(zcoord[181]-zcoord
            [180])/2), ), ((xcoord[181]+(xcoord[182]-xcoord[181])/2,
             ycoord[181]+(ycoord[182]-ycoord[181])/2, zcoord[181]+(
            zcoord[182]-zcoord[181])/2), ), ((xcoord[182]+(xcoord
            [183]-xcoord[182])/2, ycoord[182]+(ycoord[183]-ycoord
            [182])/2, zcoord[182]+(zcoord[183]-zcoord[182])/2),
```

```
108     ), ((xcoord[183]+(xcoord[184]-xcoord[183])/2, ycoord[183]+(
            ycoord[184]-ycoord[183])/2, zcoord[183]+(zcoord[184]-
            zcoord[183])/2), ), ((xcoord[184]+(xcoord[185]-xcoord
            [184])/2, ycoord[184]+(ycoord[185]-ycoord[184])/2,
            zcoord[184]+(zcoord[185]-zcoord[184])/2), ), ((xcoord
            [185]+(xcoord[186]-xcoord[185])/2, ycoord[185]+(ycoord
            [186]-ycoord[185])/2, zcoord[185]+(zcoord[186]-zcoord
            [185])/2), ), ((xcoord[186]+(xcoord[187]-xcoord[186])/2,
             ycoord[186]+(ycoord[187]-ycoord[186])/2, zcoord[186]+(
            zcoord[187]-zcoord[186])/2), ), ((xcoord[187]+(xcoord
            [188]-xcoord[187])/2, ycoord[187]+(ycoord[188]-ycoord
            [187])/2, zcoord[187]+(zcoord[188]-zcoord[187])/2), ),
            ((xcoord[188]+(xcoord[189]-xcoord[188])/2, ycoord[188]+(
            ycoord[189]-ycoord[188])/2, zcoord[188]+(zcoord[189]-
            zcoord[188])/2), ), ((xcoord[189]+(xcoord[178]-xcoord
            [189])/2, ycoord[189]+(ycoord[178]-ycoord[189])/2,
            zcoord[189]+(zcoord[178]-zcoord[189])/2), ), ),
            tryAnalytical=False)
109
110 # Create Shell: Bridle 1
111 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
112     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
113     xcoord[64]+(xcoord[194]-xcoord[64])/2, ycoord[64]+(ycoord
            [194]-ycoord[64])/2, zcoord[64]+(zcoord[194]-zcoord[64])
            /2), ), ), side2=(
114     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[72]+(xcoord[195]-xcoord[72])/2, ycoord
            [72]+(ycoord[195]-ycoord[72])/2, zcoord[72]+(zcoord
            [195]-zcoord[72])/2), ), ))
115 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
```

```
116     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
117     xcoord[194]+(xcoord[193]-xcoord[194])/2, ycoord[194]+(
            ycoord[193]-ycoord[194])/2, zcoord[194]+(zcoord[193]-
            zcoord[194])/2), ), ), side2=(
118     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[195]+(xcoord[196]-xcoord[195])/2, ycoord
            [195]+(ycoord[196]-ycoord[195])/2, zcoord[195]+(zcoord
            [196]-zcoord[195])/2), ), ))
119 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
120     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
121     xcoord[193]+(xcoord[192]-xcoord[193])/2, ycoord[193]+(
            ycoord[192]-ycoord[193])/2, zcoord[193]+(zcoord[192]-
            zcoord[193])/2), ), ), side2=(
122     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[196]+(xcoord[197]-xcoord[196])/2, ycoord
            [196]+(ycoord[197]-ycoord[196])/2, zcoord[196]+(zcoord
            [197]-zcoord[196])/2), ), ))
123 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
124     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
125     xcoord[192]+(xcoord[191]-xcoord[192])/2, ycoord[192]+(
            ycoord[191]-ycoord[192])/2, zcoord[192]+(zcoord[191]-
            zcoord[192])/2), ), ), side2=(
126     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[197]+(xcoord[198]-xcoord[197])/2, ycoord
            [197]+(ycoord[198]-ycoord[197])/2, zcoord[197]+(zcoord
            [198]-zcoord[197])/2), ), ))
127 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
```

```
128     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
129     xcoord[191]+(xcoord[190]-xcoord[191])/2, ycoord[191]+(
            ycoord[190]-ycoord[191])/2, zcoord[191]+(zcoord[190]-
            zcoord[191])/2), ), ), side2=(
130     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[198]+(xcoord[199]-xcoord[198])/2, ycoord
            [198]+(ycoord[199]-ycoord[198])/2, zcoord[198]+(zcoord
            [199]-zcoord[198])/2), ), ))
131 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
132     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
133     xcoord[190]+(xcoord[199]-xcoord[190])/2, ycoord[190]+(
            ycoord[199]-ycoord[190])/2, zcoord[190]+(zcoord[199]-
            zcoord[190])/2), ), ), side2=(
134     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[1]+(xcoord[2]-xcoord[1])/2, ycoord[1]+(
            ycoord[2]-ycoord[1])/2, zcoord[1]+(zcoord[2]-zcoord[1])
            /2), ), ))
135
136 # Create Shell: Bridle 2
137 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
138     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
139     xcoord[73]+(xcoord[204]-xcoord[73])/2, ycoord[73]+(ycoord
            [204]-ycoord[73])/2, zcoord[73]+(zcoord[204]-zcoord[73])
            /2), ), ), side2=(
140     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[84]+(xcoord[205]-xcoord[84])/2, ycoord
            [84]+(ycoord[205]-ycoord[84])/2, zcoord[84]+(zcoord
            [205]-zcoord[84])/2), ), ))
```

```
141 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
142      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
143      xcoord[204]+(xcoord[203]-xcoord[204])/2, ycoord[204]+(
             ycoord[203]-ycoord[204])/2, zcoord[204]+(zcoord[203]-
             zcoord[204])/2), ), ), side2=(
144      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[205]+(xcoord[206]-xcoord[205])/2, ycoord
             [205]+(ycoord[206]-ycoord[205])/2, zcoord[205]+(zcoord
             [206]-zcoord[205])/2), ), ))
145 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
146      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
147      xcoord[203]+(xcoord[202]-xcoord[203])/2, ycoord[203]+(
             ycoord[202]-ycoord[203])/2, zcoord[203]+(zcoord[202]-
             zcoord[203])/2), ), ), side2=(
148      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[206]+(xcoord[207]-xcoord[206])/2, ycoord
             [206]+(ycoord[207]-ycoord[206])/2, zcoord[206]+(zcoord
             [207]-zcoord[206])/2), ), ))
149 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
150      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
151      xcoord[202]+(xcoord[201]-xcoord[202])/2, ycoord[202]+(
             ycoord[201]-ycoord[202])/2, zcoord[202]+(zcoord[201]-
             zcoord[202])/2), ), ), side2=(
152      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[207]+(xcoord[208]-xcoord[207])/2, ycoord
             [207]+(ycoord[208]-ycoord[207])/2, zcoord[207]+(zcoord
             [208]-zcoord[207])/2), ), ))
```

```
153 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
154     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
155     xcoord[201]+(xcoord[200]-xcoord[201])/2, ycoord[201]+(
            ycoord[200]-ycoord[201])/2, zcoord[201]+(zcoord[200]-
            zcoord[201])/2), ), ), side2=(
156     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[208]+(xcoord[209]-xcoord[208])/2, ycoord
            [208]+(ycoord[209]-ycoord[208])/2, zcoord[208]+(zcoord
            [209]-zcoord[208])/2), ), ))
157 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
158     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
159     xcoord[200]+(xcoord[209]-xcoord[200])/2, ycoord[200]+(
            ycoord[209]-ycoord[200])/2, zcoord[200]+(zcoord[209]-
            zcoord[200])/2), ), ), side2=(
160     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[4]+(xcoord[5]-xcoord[4])/2, ycoord[4]+(
            ycoord[5]-ycoord[4])/2, zcoord[4]+(zcoord[5]-zcoord[4])
            /2), ), ))
161
162 # Create Shell: Bridle 3
163 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
164     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
165     xcoord[85]+(xcoord[214]-xcoord[85])/2, ycoord[85]+(ycoord
            [214]-ycoord[85])/2, zcoord[85]+(zcoord[214]-zcoord[85])
            /2), ), ), side2=(
166     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[96]+(xcoord[215]-xcoord[96])/2, ycoord
            [96]+(ycoord[215]-ycoord[96])/2, zcoord[96]+(zcoord
```

```
        [215]-zcoord[96])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[214]+(xcoord[213]-xcoord[214])/2, ycoord[214]+(
        ycoord[213]-ycoord[214])/2, zcoord[214]+(zcoord[213]-
        zcoord[214])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[215]+(xcoord[216]-xcoord[215])/2, ycoord
        [215]+(ycoord[216]-ycoord[215])/2, zcoord[215]+(zcoord
        [216]-zcoord[215])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[213]+(xcoord[212]-xcoord[213])/2, ycoord[213]+(
        ycoord[212]-ycoord[213])/2, zcoord[213]+(zcoord[212]-
        zcoord[213])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[216]+(xcoord[217]-xcoord[216])/2, ycoord
        [216]+(ycoord[217]-ycoord[216])/2, zcoord[216]+(zcoord
        [217]-zcoord[216])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[212]+(xcoord[211]-xcoord[212])/2, ycoord[212]+(
        ycoord[211]-ycoord[212])/2, zcoord[212]+(zcoord[211]-
        zcoord[212])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[217]+(xcoord[218]-xcoord[217])/2, ycoord
        [217]+(ycoord[218]-ycoord[217])/2, zcoord[217]+(zcoord
        [218]-zcoord[217])/2), ), ))
```

```
179 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
180      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
181      xcoord[211]+(xcoord[210]-xcoord[211])/2, ycoord[211]+(
             ycoord[210]-ycoord[211])/2, zcoord[211]+(zcoord[210]-
             zcoord[211])/2), ), ), side2=(
182      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[218]+(xcoord[219]-xcoord[218])/2, ycoord
             [218]+(ycoord[219]-ycoord[218])/2, zcoord[218]+(zcoord
             [219]-zcoord[218])/2), ), ))
183 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
184      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
185      xcoord[210]+(xcoord[219]-xcoord[210])/2, ycoord[210]+(
             ycoord[219]-ycoord[210])/2, zcoord[210]+(zcoord[219]-
             zcoord[210])/2), ), ), side2=(
186      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[7]+(xcoord[8]-xcoord[7])/2, ycoord[7]+(
             ycoord[8]-ycoord[7])/2, zcoord[7]+(zcoord[8]-zcoord[7])
             /2), ), ))
187
188 # Create Shell: Bridle 4
189 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
190      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
191      xcoord[97]+(xcoord[224]-xcoord[97])/2, ycoord[97]+(ycoord
             [224]-ycoord[97])/2, zcoord[97]+(zcoord[224]-zcoord[97])
             /2), ), ), side2=(
192      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[114]+(xcoord[225]-xcoord[114])/2, ycoord
             [114]+(ycoord[225]-ycoord[114])/2, zcoord[114]+(zcoord
```

```
        [225]-zcoord[114])/2), ), ))
193 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
194     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
195     xcoord[224]+(xcoord[223]-xcoord[224])/2, ycoord[224]+(
        ycoord[223]-ycoord[224])/2, zcoord[224]+(zcoord[223]-
        zcoord[224])/2), ), ), side2=(
196     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[225]+(xcoord[226]-xcoord[225])/2, ycoord
        [225]+(ycoord[226]-ycoord[225])/2, zcoord[225]+(zcoord
        [226]-zcoord[225])/2), ), ))
197 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
198     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
199     xcoord[223]+(xcoord[222]-xcoord[223])/2, ycoord[223]+(
        ycoord[222]-ycoord[223])/2, zcoord[223]+(zcoord[222]-
        zcoord[223])/2), ), ), side2=(
200     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[226]+(xcoord[227]-xcoord[226])/2, ycoord
        [226]+(ycoord[227]-ycoord[226])/2, zcoord[226]+(zcoord
        [227]-zcoord[226])/2), ), ))
201 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
202     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
203     xcoord[222]+(xcoord[221]-xcoord[222])/2, ycoord[222]+(
        ycoord[221]-ycoord[222])/2, zcoord[222]+(zcoord[221]-
        zcoord[222])/2), ), ), side2=(
204     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[227]+(xcoord[228]-xcoord[227])/2, ycoord
        [227]+(ycoord[228]-ycoord[227])/2, zcoord[227]+(zcoord
        [228]-zcoord[227])/2), ), ))
```

```
205 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
206     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
207     xcoord[221]+(xcoord[220]-xcoord[221])/2, ycoord[221]+(
            ycoord[220]-ycoord[221])/2, zcoord[221]+(zcoord[220]-
            zcoord[221])/2), ), ), side2=(
208     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[228]+(xcoord[229]-xcoord[228])/2, ycoord
            [228]+(ycoord[229]-ycoord[228])/2, zcoord[228]+(zcoord
            [229]-zcoord[228])/2), ), ))
209 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
210     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
211     xcoord[220]+(xcoord[229]-xcoord[220])/2, ycoord[220]+(
            ycoord[229]-ycoord[220])/2, zcoord[220]+(zcoord[229]-
            zcoord[220])/2), ), ), side2=(
212     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[10]+(xcoord[11]-xcoord[10])/2, ycoord[11]+(
            ycoord[10]-ycoord[11])/2, zcoord[10]+(zcoord[11]-zcoord
            [10])/2), ), ))
213
214 # Create Shell: Bridle 5
215 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
216     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
217     xcoord[168]+(xcoord[234]-xcoord[168])/2, ycoord[168]+(
            ycoord[234]-ycoord[168])/2, zcoord[168]+(zcoord[234]-
            zcoord[168])/2), ), ), side2=(
218     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[167]+(xcoord[235]-xcoord[167])/2, ycoord
            [167]+(ycoord[235]-ycoord[167])/2, zcoord[167]+(zcoord
```

```
        [235]-zcoord[167])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[234]+(xcoord[233]-xcoord[234])/2, ycoord[234]+(
        ycoord[233]-ycoord[234])/2, zcoord[234]+(zcoord[233]-
        zcoord[234])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[235]+(xcoord[236]-xcoord[235])/2, ycoord
        [235]+(ycoord[236]-ycoord[235])/2, zcoord[235]+(zcoord
        [236]-zcoord[235])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[233]+(xcoord[232]-xcoord[233])/2, ycoord[233]+(
        ycoord[232]-ycoord[233])/2, zcoord[233]+(zcoord[232]-
        zcoord[233])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[236]+(xcoord[237]-xcoord[236])/2, ycoord
        [236]+(ycoord[237]-ycoord[236])/2, zcoord[236]+(zcoord
        [237]-zcoord[236])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[232]+(xcoord[231]-xcoord[232])/2, ycoord[232]+(
        ycoord[231]-ycoord[232])/2, zcoord[232]+(zcoord[231]-
        zcoord[232])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[237]+(xcoord[238]-xcoord[237])/2, ycoord
        [237]+(ycoord[238]-ycoord[237])/2, zcoord[237]+(zcoord
        [238]-zcoord[237])/2), ), ))
```

```
231 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
232     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
233     xcoord[231]+(xcoord[230]-xcoord[231])/2, ycoord[231]+(
            ycoord[230]-ycoord[231])/2, zcoord[231]+(zcoord[230]-
            zcoord[231])/2), ), ), side2=(
234     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[238]+(xcoord[239]-xcoord[238])/2, ycoord
            [238]+(ycoord[239]-ycoord[238])/2, zcoord[238]+(zcoord
            [239]-zcoord[238])/2), ), ))
235 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
236     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
237     xcoord[230]+(xcoord[239]-xcoord[230])/2, ycoord[230]+(
            ycoord[239]-ycoord[230])/2, zcoord[230]+(zcoord[239]-
            zcoord[230])/2), ), ), side2=(
238     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[13]+(xcoord[14]-xcoord[13])/2, ycoord[13]+(
            ycoord[14]-ycoord[13])/2, zcoord[13]+(zcoord[14]-zcoord
            [13])/2), ), ))
239
240 # Create Shell: Bridle 6
241 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
242     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
243     xcoord[115]+(xcoord[244]-xcoord[115])/2, ycoord[115]+(
            ycoord[244]-ycoord[115])/2, zcoord[115]+(zcoord[244]-
            zcoord[115])/2), ), ), side2=(
244     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[123]+(xcoord[245]-xcoord[123])/2, ycoord
            [123]+(ycoord[245]-ycoord[123])/2, zcoord[123]+(zcoord
```

```
        [245]-zcoord[123])/2), ), ))
245 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
246     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
247     xcoord[244]+(xcoord[243]-xcoord[244])/2, ycoord[244]+(
            ycoord[243]-ycoord[244])/2, zcoord[244]+(zcoord[243]-
            zcoord[244])/2), ), ), side2=(
248     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[245]+(xcoord[246]-xcoord[245])/2, ycoord
            [245]+(ycoord[246]-ycoord[245])/2, zcoord[245]+(zcoord
            [246]-zcoord[245])/2), ), ))
249 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
250     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
251     xcoord[243]+(xcoord[242]-xcoord[243])/2, ycoord[243]+(
            ycoord[242]-ycoord[243])/2, zcoord[243]+(zcoord[242]-
            zcoord[243])/2), ), ), side2=(
252     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[246]+(xcoord[247]-xcoord[246])/2, ycoord
            [246]+(ycoord[247]-ycoord[246])/2, zcoord[246]+(zcoord
            [247]-zcoord[246])/2), ), ))
253 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
254     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
255     xcoord[242]+(xcoord[241]-xcoord[242])/2, ycoord[242]+(
            ycoord[241]-ycoord[242])/2, zcoord[242]+(zcoord[241]-
            zcoord[242])/2), ), ), side2=(
256     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[247]+(xcoord[248]-xcoord[247])/2, ycoord
            [247]+(ycoord[248]-ycoord[247])/2, zcoord[247]+(zcoord
            [248]-zcoord[247])/2), ), ))
```

```
257 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
258     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
259     xcoord[241]+(xcoord[240]-xcoord[241])/2, ycoord[241]+(
            ycoord[240]-ycoord[241])/2, zcoord[241]+(zcoord[240]-
            zcoord[241])/2), ), ), side2=(
260     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[248]+(xcoord[249]-xcoord[248])/2, ycoord
            [248]+(ycoord[249]-ycoord[248])/2, zcoord[248]+(zcoord
            [249]-zcoord[248])/2), ), ))
261 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
262     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
263     xcoord[240]+(xcoord[249]-xcoord[240])/2, ycoord[240]+(
            ycoord[249]-ycoord[240])/2, zcoord[240]+(zcoord[249]-
            zcoord[240])/2), ), ), side2=(
264     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[17]+(xcoord[18]-xcoord[17])/2, ycoord[17]+(
            ycoord[18]-ycoord[17])/2, zcoord[17]+(zcoord[18]-zcoord
            [17])/2), ), ))
265
266 # Create Shell: Bridle 7
267 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
268     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
269     xcoord[124]+(xcoord[254]-xcoord[124])/2, ycoord[124]+(
            ycoord[254]-ycoord[124])/2, zcoord[124]+(zcoord[254]-
            zcoord[124])/2), ), ), side2=(
270     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[135]+(xcoord[255]-xcoord[135])/2, ycoord
            [135]+(ycoord[255]-ycoord[135])/2, zcoord[135]+(zcoord
```

```
        [255]-zcoord[135])/2), ), ))
271 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
272     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
273     xcoord[254]+(xcoord[253]-xcoord[254])/2, ycoord[254]+(
        ycoord[253]-ycoord[254])/2, zcoord[254]+(zcoord[253]-
        zcoord[254])/2), ), ), side2=(
274     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[255]+(xcoord[256]-xcoord[255])/2, ycoord
        [255]+(ycoord[256]-ycoord[255])/2, zcoord[255]+(zcoord
        [256]-zcoord[255])/2), ), ))
275 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
276     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
277     xcoord[253]+(xcoord[252]-xcoord[253])/2, ycoord[253]+(
        ycoord[252]-ycoord[253])/2, zcoord[253]+(zcoord[252]-
        zcoord[253])/2), ), ), side2=(
278     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[256]+(xcoord[257]-xcoord[256])/2, ycoord
        [256]+(ycoord[257]-ycoord[256])/2, zcoord[256]+(zcoord
        [257]-zcoord[256])/2), ), ))
279 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
280     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
281     xcoord[252]+(xcoord[251]-xcoord[252])/2, ycoord[252]+(
        ycoord[251]-ycoord[252])/2, zcoord[252]+(zcoord[251]-
        zcoord[252])/2), ), ), side2=(
282     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[257]+(xcoord[258]-xcoord[257])/2, ycoord
        [257]+(ycoord[258]-ycoord[257])/2, zcoord[257]+(zcoord
        [258]-zcoord[257])/2), ), ))
```

```
283 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
284     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
285     xcoord[251]+(xcoord[250]-xcoord[251])/2, ycoord[251]+(
            ycoord[250]-ycoord[251])/2, zcoord[251]+(zcoord[250]-
            zcoord[251])/2), ), ), side2=(
286     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[258]+(xcoord[259]-xcoord[258])/2, ycoord
            [258]+(ycoord[259]-ycoord[258])/2, zcoord[258]+(zcoord
            [259]-zcoord[258])/2), ), ))
287 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
288     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
289     xcoord[250]+(xcoord[259]-xcoord[250])/2, ycoord[250]+(
            ycoord[259]-ycoord[250])/2, zcoord[250]+(zcoord[259]-
            zcoord[250])/2), ), ), side2=(
290     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[20]+(xcoord[21]-xcoord[20])/2, ycoord[20]+(
            ycoord[21]-ycoord[20])/2, zcoord[20]+(zcoord[21]-zcoord
            [20])/2), ), ))
291
292 # Create Shell: Bridle 8
293 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
294     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
295     xcoord[136]+(xcoord[264]-xcoord[136])/2, ycoord[136]+(
            ycoord[264]-ycoord[136])/2, zcoord[136]+(zcoord[264]-
            zcoord[136])/2), ), ), side2=(
296     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[147]+(xcoord[265]-xcoord[147])/2, ycoord
            [147]+(ycoord[265]-ycoord[147])/2, zcoord[147]+(zcoord
```

```
        [265]-zcoord[147])/2), ), ))
297 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
298     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
299     xcoord[264]+(xcoord[263]-xcoord[264])/2, ycoord[264]+(
        ycoord[263]-ycoord[264])/2, zcoord[264]+(zcoord[263]-
        zcoord[264])/2), ), ), side2=(
300     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[265]+(xcoord[266]-xcoord[265])/2, ycoord
        [265]+(ycoord[266]-ycoord[265])/2, zcoord[265]+(zcoord
        [266]-zcoord[265])/2), ), ))
301 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
302     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
303     xcoord[263]+(xcoord[262]-xcoord[263])/2, ycoord[263]+(
        ycoord[262]-ycoord[263])/2, zcoord[263]+(zcoord[262]-
        zcoord[263])/2), ), ), side2=(
304     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[266]+(xcoord[267]-xcoord[266])/2, ycoord
        [266]+(ycoord[267]-ycoord[266])/2, zcoord[266]+(zcoord
        [267]-zcoord[266])/2), ), ))
305 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
306     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
307     xcoord[262]+(xcoord[261]-xcoord[262])/2, ycoord[262]+(
        ycoord[261]-ycoord[262])/2, zcoord[262]+(zcoord[261]-
        zcoord[262])/2), ), ), side2=(
308     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[267]+(xcoord[268]-xcoord[267])/2, ycoord
        [267]+(ycoord[268]-ycoord[267])/2, zcoord[267]+(zcoord
        [268]-zcoord[267])/2), ), ))
```

```
309 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
310     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
311     xcoord[261]+(xcoord[260]-xcoord[261])/2, ycoord[261]+(
            ycoord[260]-ycoord[261])/2, zcoord[261]+(zcoord[260]-
            zcoord[261])/2), ), ), side2=(
312     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[268]+(xcoord[269]-xcoord[268])/2, ycoord
            [268]+(ycoord[269]-ycoord[268])/2, zcoord[268]+(zcoord
            [269]-zcoord[268])/2), ), ))
313 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
314     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
315     xcoord[260]+(xcoord[269]-xcoord[260])/2, ycoord[260]+(
            ycoord[269]-ycoord[260])/2, zcoord[260]+(zcoord[269]-
            zcoord[260])/2), ), ), side2=(
316     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[23]+(xcoord[24]-xcoord[23])/2, ycoord[23]+(
            ycoord[24]-ycoord[23])/2, zcoord[23]+(zcoord[24]-zcoord
            [23])/2), ), ))
317
318 # Create Shell: Bridle 9
319 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
320     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
321     xcoord[148]+(xcoord[274]-xcoord[148])/2, ycoord[148]+(
            ycoord[274]-ycoord[148])/2, zcoord[148]+(zcoord[274]-
            zcoord[148])/2), ), ), side2=(
322     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[165]+(xcoord[275]-xcoord[165])/2, ycoord
            [165]+(ycoord[275]-ycoord[165])/2, zcoord[165]+(zcoord
```

```
        [275]-zcoord[165])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[274]+(xcoord[273]-xcoord[274])/2, ycoord[274]+(
        ycoord[273]-ycoord[274])/2, zcoord[274]+(zcoord[273]-
        zcoord[274])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[275]+(xcoord[276]-xcoord[275])/2, ycoord
        [275]+(ycoord[276]-ycoord[275])/2, zcoord[275]+(zcoord
        [276]-zcoord[275])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[273]+(xcoord[272]-xcoord[273])/2, ycoord[273]+(
        ycoord[272]-ycoord[273])/2, zcoord[273]+(zcoord[272]-
        zcoord[273])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[276]+(xcoord[277]-xcoord[276])/2, ycoord
        [276]+(ycoord[277]-ycoord[276])/2, zcoord[276]+(zcoord
        [277]-zcoord[276])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[272]+(xcoord[271]-xcoord[272])/2, ycoord[272]+(
        ycoord[271]-ycoord[272])/2, zcoord[272]+(zcoord[271]-
        zcoord[272])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[277]+(xcoord[278]-xcoord[277])/2, ycoord
        [277]+(ycoord[278]-ycoord[277])/2, zcoord[277]+(zcoord
        [278]-zcoord[277])/2), ), ))
```

```
335 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
336     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
337     xcoord[271]+(xcoord[270]-xcoord[271])/2, ycoord[271]+(
            ycoord[270]-ycoord[271])/2, zcoord[271]+(zcoord[270]-
            zcoord[271])/2), ), ), side2=(
338     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[278]+(xcoord[279]-xcoord[278])/2, ycoord
            [278]+(ycoord[279]-ycoord[278])/2, zcoord[278]+(zcoord
            [279]-zcoord[278])/2), ), ))
339 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
340     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
341     xcoord[270]+(xcoord[279]-xcoord[270])/2, ycoord[270]+(
            ycoord[279]-ycoord[270])/2, zcoord[270]+(zcoord[279]-
            zcoord[270])/2), ), ), side2=(
342     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[26]+(xcoord[27]-xcoord[26])/2, ycoord[26]+(
            ycoord[27]-ycoord[26])/2, zcoord[26]+(zcoord[27]-zcoord
            [26])/2), ), ))
343
344 # Create Shell: Bridle 10
345 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
346     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
347     xcoord[176]+(xcoord[284]-xcoord[176])/2, ycoord[176]+(
            ycoord[284]-ycoord[176])/2, zcoord[176]+(zcoord[284]-
            zcoord[176])/2), ), ), side2=(
348     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[177]+(xcoord[285]-xcoord[177])/2, ycoord
            [177]+(ycoord[285]-ycoord[177])/2, zcoord[177]+(zcoord
```

```
        [285]-zcoord[177])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[284]+(xcoord[283]-xcoord[284])/2, ycoord[284]+(
        ycoord[283]-ycoord[284])/2, zcoord[284]+(zcoord[283]-
        zcoord[284])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[285]+(xcoord[286]-xcoord[285])/2, ycoord
        [285]+(ycoord[286]-ycoord[285])/2, zcoord[285]+(zcoord
        [286]-zcoord[285])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[283]+(xcoord[282]-xcoord[283])/2, ycoord[283]+(
        ycoord[282]-ycoord[283])/2, zcoord[283]+(zcoord[282]-
        zcoord[283])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[286]+(xcoord[287]-xcoord[286])/2, ycoord
        [286]+(ycoord[287]-ycoord[286])/2, zcoord[286]+(zcoord
        [287]-zcoord[286])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[282]+(xcoord[281]-xcoord[282])/2, ycoord[282]+(
        ycoord[281]-ycoord[282])/2, zcoord[282]+(zcoord[281]-
        zcoord[282])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[287]+(xcoord[288]-xcoord[287])/2, ycoord
        [287]+(ycoord[288]-ycoord[287])/2, zcoord[287]+(zcoord
        [288]-zcoord[287])/2), ), ))
```

```
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
    side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
    xcoord[281]+(xcoord[280]-xcoord[281])/2, ycoord[281]+(
        ycoord[280]-ycoord[281])/2, zcoord[281]+(zcoord[280]-
        zcoord[281])/2), ), ), side2=(
    mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[288]+(xcoord[289]-xcoord[288])/2, ycoord
        [288]+(ycoord[289]-ycoord[288])/2, zcoord[288]+(zcoord
        [289]-zcoord[288])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
    side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
    xcoord[280]+(xcoord[289]-xcoord[280])/2, ycoord[280]+(
        ycoord[289]-ycoord[280])/2, zcoord[280]+(zcoord[289]-
        zcoord[280])/2), ), ), side2=(
    mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[29]+(xcoord[30]-xcoord[29])/2, ycoord[30]+(
        ycoord[29]-ycoord[30])/2, zcoord[29]+(zcoord[30]-zcoord
        [29])/2), ), ))

# Create Shell: Bridle 11
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
    side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
    xcoord[66]+(xcoord[294]-xcoord[66])/2, ycoord[66]+(ycoord
        [294]-ycoord[66])/2, zcoord[66]+(zcoord[294]-zcoord[66])
        /2), ), ), side2=(
    mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[67]+(xcoord[295]-xcoord[67])/2, ycoord
        [67]+(ycoord[295]-ycoord[67])/2, zcoord[67]+(zcoord
```

```
        [295]-zcoord[67])/2), ), ))
375 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
376     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
377     xcoord[294]+(xcoord[293]-xcoord[294])/2, ycoord[294]+(
        ycoord[293]-ycoord[294])/2, zcoord[294]+(zcoord[293]-
        zcoord[294])/2), ), ), side2=(
378     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[295]+(xcoord[296]-xcoord[295])/2, ycoord
        [295]+(ycoord[296]-ycoord[295])/2, zcoord[295]+(zcoord
        [296]-zcoord[295])/2), ), ))
379 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
380     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
381     xcoord[293]+(xcoord[292]-xcoord[293])/2, ycoord[293]+(
        ycoord[292]-ycoord[293])/2, zcoord[293]+(zcoord[292]-
        zcoord[293])/2), ), ), side2=(
382     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[296]+(xcoord[297]-xcoord[296])/2, ycoord
        [296]+(ycoord[297]-ycoord[296])/2, zcoord[296]+(zcoord
        [297]-zcoord[296])/2), ), ))
383 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
384     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
385     xcoord[292]+(xcoord[291]-xcoord[292])/2, ycoord[292]+(
        ycoord[291]-ycoord[292])/2, zcoord[292]+(zcoord[291]-
        zcoord[292])/2), ), ), side2=(
386     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[297]+(xcoord[298]-xcoord[297])/2, ycoord
        [297]+(ycoord[298]-ycoord[297])/2, zcoord[297]+(zcoord
        [298]-zcoord[297])/2), ), ))
```

```
387 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
388     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
389     xcoord[291]+(xcoord[290]-xcoord[291])/2, ycoord[291]+(
            ycoord[290]-ycoord[291])/2, zcoord[291]+(zcoord[290]-
            zcoord[291])/2), ), ), side2=(
390     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[298]+(xcoord[299]-xcoord[298])/2, ycoord
            [298]+(ycoord[299]-ycoord[298])/2, zcoord[298]+(zcoord
            [299]-zcoord[298])/2), ), ))
391 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
392     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
393     xcoord[290]+(xcoord[299]-xcoord[290])/2, ycoord[290]+(
            ycoord[299]-ycoord[290])/2, zcoord[290]+(zcoord[299]-
            zcoord[290])/2), ), ), side2=(
394     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[33]+(xcoord[34]-xcoord[33])/2, ycoord[33]+(
            ycoord[34]-ycoord[33])/2, zcoord[33]+(zcoord[34]-zcoord
            [33])/2), ), ))
395
396 # Create Shell: Bridle 12
397 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
398     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
399     xcoord[78]+(xcoord[304]-xcoord[78])/2, ycoord[78]+(ycoord
            [304]-ycoord[78])/2, zcoord[78]+(zcoord[304]-zcoord[78])
            /2), ), ), side2=(
400     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[79]+(xcoord[305]-xcoord[79])/2, ycoord
            [79]+(ycoord[305]-ycoord[79])/2, zcoord[79]+(zcoord
```

```
        [305]-zcoord[79])/2), ), ))
401 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
402     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
403     xcoord[304]+(xcoord[303]-xcoord[304])/2, ycoord[304]+(
        ycoord[303]-ycoord[304])/2, zcoord[304]+(zcoord[303]-
        zcoord[304])/2), ), ), side2=(
404     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[305]+(xcoord[306]-xcoord[305])/2, ycoord
        [305]+(ycoord[306]-ycoord[305])/2, zcoord[305]+(zcoord
        [306]-zcoord[305])/2), ), ))
405 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
406     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
407     xcoord[303]+(xcoord[302]-xcoord[303])/2, ycoord[303]+(
        ycoord[302]-ycoord[303])/2, zcoord[303]+(zcoord[302]-
        zcoord[303])/2), ), ), side2=(
408     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[306]+(xcoord[307]-xcoord[306])/2, ycoord
        [306]+(ycoord[307]-ycoord[306])/2, zcoord[306]+(zcoord
        [307]-zcoord[306])/2), ), ))
409 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
410     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
411     xcoord[302]+(xcoord[301]-xcoord[302])/2, ycoord[302]+(
        ycoord[301]-ycoord[302])/2, zcoord[302]+(zcoord[301]-
        zcoord[302])/2), ), ), side2=(
412     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[307]+(xcoord[308]-xcoord[307])/2, ycoord
        [307]+(ycoord[308]-ycoord[307])/2, zcoord[307]+(zcoord
        [308]-zcoord[307])/2), ), ))
```

```
413 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
414     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
415     xcoord[301]+(xcoord[300]-xcoord[301])/2, ycoord[301]+(
            ycoord[300]-ycoord[301])/2, zcoord[301]+(zcoord[300]-
            zcoord[301])/2), ), ), side2=(
416     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[308]+(xcoord[309]-xcoord[308])/2, ycoord
            [308]+(ycoord[309]-ycoord[308])/2, zcoord[308]+(zcoord
            [309]-zcoord[308])/2), ), ))
417 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
418     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
419     xcoord[300]+(xcoord[309]-xcoord[300])/2, ycoord[300]+(
            ycoord[309]-ycoord[300])/2, zcoord[300]+(zcoord[309]-
            zcoord[300])/2), ), ), side2=(
420     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[36]+(xcoord[37]-xcoord[36])/2, ycoord[36]+(
            ycoord[37]-ycoord[36])/2, zcoord[36]+(zcoord[37]-zcoord
            [36])/2), ), ))
421
422 # Create Shell: Bridle 13
423 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
424     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
425     xcoord[90]+(xcoord[314]-xcoord[90])/2, ycoord[90]+(ycoord
            [314]-ycoord[90])/2, zcoord[90]+(zcoord[314]-zcoord[90])
            /2), ), ), side2=(
426     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[91]+(xcoord[315]-xcoord[91])/2, ycoord
            [91]+(ycoord[315]-ycoord[91])/2, zcoord[91]+(zcoord
```

```
        [315]-zcoord[91])/2), ), ))
427 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
428     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
429     xcoord[314]+(xcoord[313]-xcoord[314])/2, ycoord[314]+(
        ycoord[313]-ycoord[314])/2, zcoord[314]+(zcoord[313]-
        zcoord[314])/2), ), ), side2=(
430     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[315]+(xcoord[316]-xcoord[315])/2, ycoord
        [315]+(ycoord[316]-ycoord[315])/2, zcoord[315]+(zcoord
        [316]-zcoord[315])/2), ), ))
431 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
432     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
433     xcoord[313]+(xcoord[312]-xcoord[313])/2, ycoord[313]+(
        ycoord[312]-ycoord[313])/2, zcoord[313]+(zcoord[312]-
        zcoord[313])/2), ), ), side2=(
434     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[316]+(xcoord[317]-xcoord[316])/2, ycoord
        [316]+(ycoord[317]-ycoord[316])/2, zcoord[316]+(zcoord
        [317]-zcoord[316])/2), ), ))
435 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
436     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
437     xcoord[312]+(xcoord[311]-xcoord[312])/2, ycoord[312]+(
        ycoord[311]-ycoord[312])/2, zcoord[312]+(zcoord[311]-
        zcoord[312])/2), ), ), side2=(
438     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[317]+(xcoord[318]-xcoord[317])/2, ycoord
        [317]+(ycoord[318]-ycoord[317])/2, zcoord[317]+(zcoord
        [318]-zcoord[317])/2), ), ))
```

```
439 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
440      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
441      xcoord[311]+(xcoord[310]-xcoord[311])/2, ycoord[311]+(
             ycoord[310]-ycoord[311])/2, zcoord[311]+(zcoord[310]-
             zcoord[311])/2), ), ), side2=(
442      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[318]+(xcoord[319]-xcoord[318])/2, ycoord
             [318]+(ycoord[319]-ycoord[318])/2, zcoord[318]+(zcoord
             [319]-zcoord[318])/2), ), ))
443 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
444      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
445      xcoord[310]+(xcoord[319]-xcoord[310])/2, ycoord[310]+(
             ycoord[319]-ycoord[310])/2, zcoord[310]+(zcoord[319]-
             zcoord[310])/2), ), ), side2=(
446      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[39]+(xcoord[40]-xcoord[39])/2, ycoord[39]+(
             ycoord[40]-ycoord[39])/2, zcoord[39]+(zcoord[40]-zcoord
             [39])/2), ), ))
447
448 # Create Shell: Bridle 14
449 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
450      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
451      xcoord[102]+(xcoord[324]-xcoord[102])/2, ycoord[102]+(
             ycoord[324]-ycoord[102])/2, zcoord[102]+(zcoord[324]-
             zcoord[102])/2), ), ), side2=(
452      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[103]+(xcoord[325]-xcoord[103])/2, ycoord
             [103]+(ycoord[325]-ycoord[103])/2, zcoord[103]+(zcoord
```

```
        [325]-zcoord[103])/2), ), ))
453 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
454     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
455     xcoord[324]+(xcoord[323]-xcoord[324])/2, ycoord[324]+(
        ycoord[323]-ycoord[324])/2, zcoord[324]+(zcoord[323]-
        zcoord[324])/2), ), ), side2=(
456     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[325]+(xcoord[326]-xcoord[325])/2, ycoord
        [325]+(ycoord[326]-ycoord[325])/2, zcoord[325]+(zcoord
        [326]-zcoord[325])/2), ), ))
457 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
458     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
459     xcoord[323]+(xcoord[322]-xcoord[323])/2, ycoord[323]+(
        ycoord[322]-ycoord[323])/2, zcoord[323]+(zcoord[322]-
        zcoord[323])/2), ), ), side2=(
460     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[326]+(xcoord[327]-xcoord[326])/2, ycoord
        [326]+(ycoord[327]-ycoord[326])/2, zcoord[326]+(zcoord
        [327]-zcoord[326])/2), ), ))
461 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
462     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
463     xcoord[322]+(xcoord[321]-xcoord[322])/2, ycoord[322]+(
        ycoord[321]-ycoord[322])/2, zcoord[322]+(zcoord[321]-
        zcoord[322])/2), ), ), side2=(
464     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[327]+(xcoord[328]-xcoord[327])/2, ycoord
        [327]+(ycoord[328]-ycoord[327])/2, zcoord[327]+(zcoord
        [328]-zcoord[327])/2), ), ))
```

```
465 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
466      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
467      xcoord[321]+(xcoord[320]-xcoord[321])/2, ycoord[321]+(
             ycoord[320]-ycoord[321])/2, zcoord[321]+(zcoord[320]-
             zcoord[321])/2), ), ), side2=(
468      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[328]+(xcoord[329]-xcoord[328])/2, ycoord
             [328]+(ycoord[329]-ycoord[328])/2, zcoord[328]+(zcoord
             [329]-zcoord[328])/2), ), ))
469 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
470      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
471      xcoord[320]+(xcoord[329]-xcoord[320])/2, ycoord[320]+(
             ycoord[329]-ycoord[320])/2, zcoord[320]+(zcoord[329]-
             zcoord[320])/2), ), ), side2=(
472      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[42]+(xcoord[43]-xcoord[42])/2, ycoord[42]+(
             ycoord[43]-ycoord[42])/2, zcoord[42]+(zcoord[43]-zcoord
             [42])/2), ), ))
473
474 # Create Shell: Bridle 15
475 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
476      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
477      xcoord[180]+(xcoord[334]-xcoord[180])/2, ycoord[180]+(
             ycoord[334]-ycoord[180])/2, zcoord[180]+(zcoord[334]-
             zcoord[180])/2), ), ), side2=(
478      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[179]+(xcoord[335]-xcoord[179])/2, ycoord
             [179]+(ycoord[335]-ycoord[179])/2, zcoord[179]+(zcoord
```

```
        [335]-zcoord[179])/2), ), ))
479 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
480     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
481     xcoord[334]+(xcoord[333]-xcoord[334])/2, ycoord[334]+(
        ycoord[333]-ycoord[334])/2, zcoord[334]+(zcoord[333]-
        zcoord[334])/2), ), ), side2=(
482     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[335]+(xcoord[336]-xcoord[335])/2, ycoord
        [335]+(ycoord[336]-ycoord[335])/2, zcoord[335]+(zcoord
        [336]-zcoord[335])/2), ), ))
483 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
484     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
485     xcoord[333]+(xcoord[332]-xcoord[333])/2, ycoord[333]+(
        ycoord[332]-ycoord[333])/2, zcoord[333]+(zcoord[332]-
        zcoord[333])/2), ), ), side2=(
486     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[336]+(xcoord[337]-xcoord[336])/2, ycoord
        [336]+(ycoord[337]-ycoord[336])/2, zcoord[336]+(zcoord
        [337]-zcoord[336])/2), ), ))
487 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
488     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
489     xcoord[332]+(xcoord[331]-xcoord[332])/2, ycoord[332]+(
        ycoord[331]-ycoord[332])/2, zcoord[332]+(zcoord[331]-
        zcoord[332])/2), ), ), side2=(
490     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[337]+(xcoord[338]-xcoord[337])/2, ycoord
        [337]+(ycoord[338]-ycoord[337])/2, zcoord[337]+(zcoord
        [338]-zcoord[337])/2), ), ))
```

```
491 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
492     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
493     xcoord[331]+(xcoord[330]-xcoord[331])/2, ycoord[331]+(
            ycoord[330]-ycoord[331])/2, zcoord[331]+(zcoord[330]-
            zcoord[331])/2), ), ), side2=(
494     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[338]+(xcoord[339]-xcoord[338])/2, ycoord
            [338]+(ycoord[339]-ycoord[338])/2, zcoord[338]+(zcoord
            [339]-zcoord[338])/2), ), ))
495 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
496     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
497     xcoord[330]+(xcoord[339]-xcoord[330])/2, ycoord[330]+(
            ycoord[339]-ycoord[330])/2, zcoord[330]+(zcoord[339]-
            zcoord[330])/2), ), ), side2=(
498     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[45]+(xcoord[46]-xcoord[45])/2, ycoord[45]+(
            ycoord[46]-ycoord[45])/2, zcoord[45]+(zcoord[46]-zcoord
            [45])/2), ), ))
499
500 # Create Shell: Bridle 16
501 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
502     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
503     xcoord[117]+(xcoord[344]-xcoord[117])/2, ycoord[117]+(
            ycoord[344]-ycoord[117])/2, zcoord[117]+(zcoord[344]-
            zcoord[117])/2), ), ), side2=(
504     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[118]+(xcoord[345]-xcoord[118])/2, ycoord
            [118]+(ycoord[345]-ycoord[118])/2, zcoord[118]+(zcoord
```

```
        [345]-zcoord[118])/2), ), ))
505 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
506     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
507     xcoord[344]+(xcoord[343]-xcoord[344])/2, ycoord[344]+(
        ycoord[343]-ycoord[344])/2, zcoord[344]+(zcoord[343]-
        zcoord[344])/2), ), ), side2=(
508     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[345]+(xcoord[346]-xcoord[345])/2, ycoord
        [345]+(ycoord[346]-ycoord[345])/2, zcoord[345]+(zcoord
        [346]-zcoord[345])/2), ), ))
509 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
510     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
511     xcoord[343]+(xcoord[342]-xcoord[343])/2, ycoord[343]+(
        ycoord[342]-ycoord[343])/2, zcoord[343]+(zcoord[342]-
        zcoord[343])/2), ), ), side2=(
512     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[346]+(xcoord[347]-xcoord[346])/2, ycoord
        [346]+(ycoord[347]-ycoord[346])/2, zcoord[346]+(zcoord
        [347]-zcoord[346])/2), ), ))
513 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
514     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
515     xcoord[342]+(xcoord[341]-xcoord[342])/2, ycoord[342]+(
        ycoord[341]-ycoord[342])/2, zcoord[342]+(zcoord[341]-
        zcoord[342])/2), ), ), side2=(
516     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[347]+(xcoord[348]-xcoord[347])/2, ycoord
        [347]+(ycoord[348]-ycoord[347])/2, zcoord[347]+(zcoord
        [348]-zcoord[347])/2), ), ))
```

```
517 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
518     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
519     xcoord[341]+(xcoord[340]-xcoord[341])/2, ycoord[341]+(
            ycoord[340]-ycoord[341])/2, zcoord[341]+(zcoord[340]-
            zcoord[341])/2), ), ), side2=(
520     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[348]+(xcoord[349]-xcoord[348])/2, ycoord
            [348]+(ycoord[349]-ycoord[348])/2, zcoord[348]+(zcoord
            [349]-zcoord[348])/2), ), ))
521 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
522     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
523     xcoord[340]+(xcoord[349]-xcoord[340])/2, ycoord[340]+(
            ycoord[349]-ycoord[340])/2, zcoord[340]+(zcoord[349]-
            zcoord[340])/2), ), ), side2=(
524     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[49]+(xcoord[50]-xcoord[49])/2, ycoord[49]+(
            ycoord[50]-ycoord[49])/2, zcoord[49]+(zcoord[50]-zcoord
            [49])/2), ), ))
525
526 # Create Shell: Bridle 17
527 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
528     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
529     xcoord[129]+(xcoord[354]-xcoord[129])/2, ycoord[129]+(
            ycoord[354]-ycoord[129])/2, zcoord[129]+(zcoord[354]-
            zcoord[129])/2), ), ), side2=(
530     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[130]+(xcoord[355]-xcoord[130])/2, ycoord
            [130]+(ycoord[355]-ycoord[130])/2, zcoord[130]+(zcoord
```

```
        [355]-zcoord[130])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[354]+(xcoord[353]-xcoord[354])/2, ycoord[354]+(
        ycoord[353]-ycoord[354])/2, zcoord[354]+(zcoord[353]-
        zcoord[354])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[355]+(xcoord[356]-xcoord[355])/2, ycoord
        [355]+(ycoord[356]-ycoord[355])/2, zcoord[355]+(zcoord
        [356]-zcoord[355])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[353]+(xcoord[352]-xcoord[353])/2, ycoord[353]+(
        ycoord[352]-ycoord[353])/2, zcoord[353]+(zcoord[352]-
        zcoord[353])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[356]+(xcoord[357]-xcoord[356])/2, ycoord
        [356]+(ycoord[357]-ycoord[356])/2, zcoord[356]+(zcoord
        [357]-zcoord[356])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[352]+(xcoord[351]-xcoord[352])/2, ycoord[352]+(
        ycoord[351]-ycoord[352])/2, zcoord[352]+(zcoord[351]-
        zcoord[352])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[357]+(xcoord[358]-xcoord[357])/2, ycoord
        [357]+(ycoord[358]-ycoord[357])/2, zcoord[357]+(zcoord
        [358]-zcoord[357])/2), ), ))
```

```
543 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
544     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
545     xcoord[351]+(xcoord[350]-xcoord[351])/2, ycoord[351]+(
            ycoord[350]-ycoord[351])/2, zcoord[351]+(zcoord[350]-
            zcoord[351])/2), ), ), side2=(
546     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[358]+(xcoord[359]-xcoord[358])/2, ycoord
            [358]+(ycoord[359]-ycoord[358])/2, zcoord[358]+(zcoord
            [359]-zcoord[358])/2), ), ))
547 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
548     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
549     xcoord[350]+(xcoord[359]-xcoord[350])/2, ycoord[350]+(
            ycoord[359]-ycoord[350])/2, zcoord[350]+(zcoord[359]-
            zcoord[350])/2), ), ), side2=(
550     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[52]+(xcoord[53]-xcoord[52])/2, ycoord[52]+(
            ycoord[53]-ycoord[52])/2, zcoord[52]+(zcoord[53]-zcoord
            [52])/2), ), ))
551
552 # Create Shell: Bridle 18
553 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
554     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
555     xcoord[141]+(xcoord[364]-xcoord[141])/2, ycoord[141]+(
            ycoord[364]-ycoord[141])/2, zcoord[141]+(zcoord[364]-
            zcoord[141])/2), ), ), side2=(
556     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[142]+(xcoord[365]-xcoord[142])/2, ycoord
            [142]+(ycoord[365]-ycoord[142])/2, zcoord[142]+(zcoord
```

```
        [365]-zcoord[142])/2), ), ))
557 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
558     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
559     xcoord[364]+(xcoord[363]-xcoord[364])/2, ycoord[364]+(
        ycoord[363]-ycoord[364])/2, zcoord[364]+(zcoord[363]-
        zcoord[364])/2), ), ), side2=(
560     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[365]+(xcoord[366]-xcoord[365])/2, ycoord
        [365]+(ycoord[366]-ycoord[365])/2, zcoord[365]+(zcoord
        [366]-zcoord[365])/2), ), ))
561 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
562     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
563     xcoord[363]+(xcoord[362]-xcoord[363])/2, ycoord[363]+(
        ycoord[362]-ycoord[363])/2, zcoord[363]+(zcoord[362]-
        zcoord[363])/2), ), ), side2=(
564     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[366]+(xcoord[367]-xcoord[366])/2, ycoord
        [366]+(ycoord[367]-ycoord[366])/2, zcoord[366]+(zcoord
        [367]-zcoord[366])/2), ), ))
565 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
566     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
567     xcoord[362]+(xcoord[361]-xcoord[362])/2, ycoord[362]+(
        ycoord[361]-ycoord[362])/2, zcoord[362]+(zcoord[361]-
        zcoord[362])/2), ), ), side2=(
568     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[367]+(xcoord[368]-xcoord[367])/2, ycoord
        [367]+(ycoord[368]-ycoord[367])/2, zcoord[367]+(zcoord
        [368]-zcoord[367])/2), ), ))
```

```
569 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
570     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
571     xcoord[361]+(xcoord[360]-xcoord[361])/2, ycoord[361]+(
            ycoord[360]-ycoord[361])/2, zcoord[361]+(zcoord[360]-
            zcoord[361])/2), ), ), side2=(
572     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[368]+(xcoord[369]-xcoord[368])/2, ycoord
            [368]+(ycoord[369]-ycoord[368])/2, zcoord[368]+(zcoord
            [369]-zcoord[368])/2), ), ))
573 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
574     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
575     xcoord[360]+(xcoord[369]-xcoord[360])/2, ycoord[360]+(
            ycoord[369]-ycoord[360])/2, zcoord[360]+(zcoord[369]-
            zcoord[360])/2), ), ), side2=(
576     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[55]+(xcoord[56]-xcoord[55])/2, ycoord[56]+(
            ycoord[55]-ycoord[56])/2, zcoord[55]+(zcoord[56]-zcoord
            [55])/2), ), ))
577
578 # Create Shell: Bridle 19
579 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
580     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
581     xcoord[153]+(xcoord[374]-xcoord[153])/2, ycoord[153]+(
            ycoord[374]-ycoord[153])/2, zcoord[153]+(zcoord[374]-
            zcoord[153])/2), ), ), side2=(
582     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[154]+(xcoord[375]-xcoord[154])/2, ycoord
            [154]+(ycoord[375]-ycoord[154])/2, zcoord[154]+(zcoord
```

```
        [375]-zcoord[154])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[374]+(xcoord[373]-xcoord[374])/2, ycoord[374]+(
        ycoord[373]-ycoord[374])/2, zcoord[374]+(zcoord[373]-
        zcoord[374])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[375]+(xcoord[376]-xcoord[375])/2, ycoord
        [375]+(ycoord[376]-ycoord[375])/2, zcoord[375]+(zcoord
        [376]-zcoord[375])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[373]+(xcoord[372]-xcoord[373])/2, ycoord[373]+(
        ycoord[372]-ycoord[373])/2, zcoord[373]+(zcoord[372]-
        zcoord[373])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[376]+(xcoord[377]-xcoord[376])/2, ycoord
        [376]+(ycoord[377]-ycoord[376])/2, zcoord[376]+(zcoord
        [377]-zcoord[376])/2), ), ))
mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
     xcoord[372]+(xcoord[371]-xcoord[372])/2, ycoord[372]+(
        ycoord[371]-ycoord[372])/2, zcoord[372]+(zcoord[371]-
        zcoord[372])/2), ), ), side2=(
     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[377]+(xcoord[378]-xcoord[377])/2, ycoord
        [377]+(ycoord[378]-ycoord[377])/2, zcoord[377]+(zcoord
        [378]-zcoord[377])/2), ), ))
```

```
595 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
596     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
597     xcoord[371]+(xcoord[370]-xcoord[371])/2, ycoord[371]+(
            ycoord[370]-ycoord[371])/2, zcoord[371]+(zcoord[370]-
            zcoord[371])/2), ), ), side2=(
598     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[378]+(xcoord[379]-xcoord[378])/2, ycoord
            [378]+(ycoord[379]-ycoord[378])/2, zcoord[378]+(zcoord
            [379]-zcoord[378])/2), ), ))
599 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
600     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
601     xcoord[370]+(xcoord[379]-xcoord[370])/2, ycoord[370]+(
            ycoord[379]-ycoord[370])/2, zcoord[370]+(zcoord[379]-
            zcoord[370])/2), ), ), side2=(
602     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[58]+(xcoord[59]-xcoord[58])/2, ycoord[58]+(
            ycoord[59]-ycoord[58])/2, zcoord[58]+(zcoord[59]-zcoord
            [58])/2), ), ))
603
604 # Create Shell: Bridle 20
605 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
606     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
607     xcoord[188]+(xcoord[384]-xcoord[188])/2, ycoord[188]+(
            ycoord[384]-ycoord[188])/2, zcoord[188]+(zcoord[384]-
            zcoord[188])/2), ), ), side2=(
608     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[189]+(xcoord[385]-xcoord[189])/2, ycoord
            [189]+(ycoord[385]-ycoord[189])/2, zcoord[189]+(zcoord
```

```
        [385]-zcoord[189])/2), ), ))
609 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
610     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
611     xcoord[384]+(xcoord[383]-xcoord[384])/2, ycoord[384]+(
        ycoord[383]-ycoord[384])/2, zcoord[384]+(zcoord[383]-
        zcoord[384])/2), ), ), side2=(
612     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[385]+(xcoord[386]-xcoord[385])/2, ycoord
        [385]+(ycoord[386]-ycoord[385])/2, zcoord[385]+(zcoord
        [386]-zcoord[385])/2), ), ))
613 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
614     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
615     xcoord[383]+(xcoord[382]-xcoord[383])/2, ycoord[383]+(
        ycoord[382]-ycoord[383])/2, zcoord[383]+(zcoord[382]-
        zcoord[383])/2), ), ), side2=(
616     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[386]+(xcoord[387]-xcoord[386])/2, ycoord
        [386]+(ycoord[387]-ycoord[386])/2, zcoord[386]+(zcoord
        [387]-zcoord[386])/2), ), ))
617 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
618     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
619     xcoord[382]+(xcoord[381]-xcoord[382])/2, ycoord[382]+(
        ycoord[381]-ycoord[382])/2, zcoord[382]+(zcoord[381]-
        zcoord[382])/2), ), ), side2=(
620     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[387]+(xcoord[388]-xcoord[387])/2, ycoord
        [387]+(ycoord[388]-ycoord[387])/2, zcoord[387]+(zcoord
        [388]-zcoord[387])/2), ), ))
```

```
621 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
622     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
623     xcoord[381]+(xcoord[380]-xcoord[381])/2, ycoord[381]+(
            ycoord[380]-ycoord[381])/2, zcoord[381]+(zcoord[380]-
            zcoord[381])/2), ), ), side2=(
624     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[388]+(xcoord[389]-xcoord[388])/2, ycoord
            [388]+(ycoord[389]-ycoord[388])/2, zcoord[388]+(zcoord
            [389]-zcoord[388])/2), ), ))
625 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
626     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
627     xcoord[380]+(xcoord[389]-xcoord[380])/2, ycoord[380]+(
            ycoord[389]-ycoord[380])/2, zcoord[380]+(zcoord[389]-
            zcoord[380])/2), ), ), side2=(
628     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[61]+(xcoord[62]-xcoord[61])/2, ycoord[61]+(
            ycoord[62]-ycoord[61])/2, zcoord[61]+(zcoord[62]-zcoord
            [61])/2), ), ))
629
630 # Create Shell: Central Axis
631 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
632     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
633     xcoord[69]+(xcoord[70]-xcoord[69])/2, ycoord[69]+(ycoord
            [70]-ycoord[69])/2, zcoord[69]+(zcoord[70]-zcoord[69])
            /2), ), ), side2=(
634     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[75]+(xcoord[76]-xcoord[75])/2, ycoord[75]+(
            ycoord[76]-ycoord[75])/2, zcoord[75]+(zcoord[76]-zcoord
```

```
        [76])/2), ), ))
635 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
636     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
637     xcoord[81]+(xcoord[82]-xcoord[81])/2, ycoord[81]+(ycoord
        [82]-ycoord[81])/2, zcoord[81]+(zcoord[82]-zcoord[81])
        /2), ), ), side2=(
638     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[87]+(xcoord[88]-xcoord[87])/2, ycoord[87]+(
        ycoord[88]-ycoord[87])/2, zcoord[87]+(zcoord[88]-zcoord
        [87])/2), ), ))
639 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
640     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
641     xcoord[93]+(xcoord[94]-xcoord[93])/2, ycoord[93]+(ycoord
        [94]-ycoord[93])/2, zcoord[93]+(zcoord[94]-zcoord[93])
        /2), ), ), side2=(
642     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[99]+(xcoord[100]-xcoord[99])/2, ycoord
        [99]+(ycoord[100]-ycoord[99])/2, zcoord[99]+(zcoord
        [100]-zcoord[99])/2), ), ))
643 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
    method=SHORTEST_PATH,
644     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .edges.findAt((
645     xcoord[111]+(xcoord[112]-xcoord[111])/2, ycoord[111]+(
        ycoord[112]-ycoord[111])/2, zcoord[111]+(zcoord[112]-
        zcoord[111])/2), ), ), side2=(
646     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        ((    xcoord[170]+(xcoord[171]-xcoord[170])/2, ycoord
        [170]+(ycoord[171]-ycoord[170])/2, zcoord[170]+(zcoord
        [171]-zcoord[170])/2), ), ))
```

```
647 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
648     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
649     xcoord[105]+(xcoord[106]-xcoord[105])/2, ycoord[105]+(
            ycoord[106]-ycoord[105])/2, zcoord[105]+(zcoord[106]-
            zcoord[105])/2), ), ), side2=(
650     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[182]+(xcoord[183]-xcoord[182])/2, ycoord
            [182]+(ycoord[183]-ycoord[182])/2, zcoord[182]+(zcoord
            [183]-zcoord[182])/2), ), ))
651 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
652     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
653     xcoord[173]+(xcoord[174]-xcoord[173])/2, ycoord[173]+(
            ycoord[174]-ycoord[173])/2, zcoord[173]+(zcoord[174]-
            zcoord[173])/2), ), ), side2=(
654     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[162]+(xcoord[163]-xcoord[162])/2, ycoord
            [162]+(ycoord[163]-ycoord[162])/2, zcoord[162]+(zcoord
            [163]-zcoord[162])/2), ), ))
655 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
656     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
657     xcoord[185]+(xcoord[186]-xcoord[185])/2, ycoord[185]+(
            ycoord[186]-ycoord[185])/2, zcoord[185]+(zcoord[186]-
            zcoord[185])/2), ), ), side2=(
658     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[156]+(xcoord[157]-xcoord[156])/2, ycoord
            [156]+(ycoord[157]-ycoord[156])/2, zcoord[156]+(zcoord
            [157]-zcoord[156])/2), ), ))
```

```
659 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
660      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
661      xcoord[150]+(xcoord[151]-xcoord[150])/2, ycoord[150]+(
             ycoord[151]-ycoord[150])/2, zcoord[150]+(zcoord[151]-
             zcoord[150])/2), ), ), side2=(
662      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[144]+(xcoord[145]-xcoord[144])/2, ycoord
             [144]+(ycoord[145]-ycoord[144])/2, zcoord[144]+(zcoord
             [145]-zcoord[144])/2), ), ))
663 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
664      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
665      xcoord[138]+(xcoord[139]-xcoord[138])/2, ycoord[138]+(
             ycoord[139]-ycoord[138])/2, zcoord[138]+(zcoord[139]-
             zcoord[138])/2), ), ), side2=(
666      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[132]+(xcoord[133]-xcoord[132])/2, ycoord
             [132]+(ycoord[133]-ycoord[132])/2, zcoord[132]+(zcoord
             [133]-zcoord[132])/2), ), ))
667 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
668      side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .edges.findAt((
669      xcoord[126]+(xcoord[127]-xcoord[126])/2, ycoord[126]+(
             ycoord[127]-ycoord[126])/2, zcoord[126]+(zcoord[127]-
             zcoord[126])/2), ), ), side2=(
670      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             ((    xcoord[120]+(xcoord[121]-xcoord[120])/2, ycoord
             [120]+(ycoord[121]-ycoord[120])/2, zcoord[120]+(zcoord
             [121]-zcoord[120])/2), ), ))
```

```
671 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].BlendFaces(
        method=SHORTEST_PATH,
672     side1=(mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .edges.findAt((
673     xcoord[108]+(xcoord[109]-xcoord[108])/2, ycoord[108]+(
            ycoord[109]-ycoord[108])/2, zcoord[108]+(zcoord[109]-
            zcoord[108])/2), ), ), side2=(
674     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            ((    xcoord[159]+(xcoord[160]-xcoord[159])/2, ycoord
            [159]+(ycoord[160]-ycoord[159])/2, zcoord[159]+(zcoord
            [160]-zcoord[159])/2), ), ))
675
676 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(faces=
677     mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .faces.getSequenceFromMask((
678     '[#7ff #0:3 #1ff8 ]', ), ), name='Central Axis')
679 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(faces=
680     mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .faces.getSequenceFromMask((
681     '[#1f800 #7ff8 #80001ffe #e00007ff #1e007 ]', ), ), name=
682     'Exterior Bridles')
683 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(faces=
684     mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .faces.getSequenceFromMask((
685     '[#fffe0000 #ffff8007 #7fffe041 #1ffff800 #4 ]', ), ), name
            =
686     'Interior Bridles')
687
688 mdb.models['CFRP Bridge'].Material(name='CFRP - Central Axis')
689 mdb.models['CFRP Bridge'].materials['CFRP - Central Axis']
        .Density(table=((1.7,
690     ), ))
691 mdb.models['CFRP Bridge'].materials['CFRP - Central Axis']
        .Elastic(table=((
```

```
692     e[0], 0.3), ))
693 mdb.models['CFRP Bridge'].Material(name='CFRP - Exterior
        Bridles')
694 mdb.models['CFRP Bridge'].materials['CFRP - Exterior Bridles']
        .Density(table=((1.7,
695     ), ))
696 mdb.models['CFRP Bridge'].materials['CFRP - Exterior Bridles']
        .Elastic(table=((
697     e[1], 0.3), ))
698 mdb.models['CFRP Bridge'].Material(name='CFRP - Interior
        Bridles')
699 mdb.models['CFRP Bridge'].materials['CFRP - Interior Bridles']
        .Density(table=((1.7,
700     ), ))
701 mdb.models['CFRP Bridge'].materials['CFRP - Interior Bridles']
        .Elastic(table=((
702     e[2], 0.3), ))
703
704 mdb.models['CFRP Bridge'].HomogeneousShellSection(idealization=
        NO_IDEALIZATION,
705     integrationRule=SIMPSON, material='CFRP - Central Axis',
            name='Section - Central Axis',
706     numIntPts=5, poissonDefinition=DEFAULT, preIntegrate=OFF,
            temperature=
707     GRADIENT, thickness=b[0], thicknessField='',
            thicknessModulus=None,
708     thicknessType=UNIFORM, useDensity=OFF)
709 mdb.models['CFRP Bridge'].HomogeneousShellSection(idealization=
        NO_IDEALIZATION,
710     integrationRule=SIMPSON, material='CFRP - Exterior Bridles'
            , name='Section - Exterior Bridles',
711     numIntPts=5, poissonDefinition=DEFAULT, preIntegrate=OFF,
            temperature=
```

```
712     GRADIENT, thickness=b[1], thicknessField='',
            thicknessModulus=None,
713     thicknessType=UNIFORM, useDensity=OFF)
714 mdb.models['CFRP Bridge'].HomogeneousShellSection(idealization=
        NO_IDEALIZATION,
715     integrationRule=SIMPSON, material='CFRP - Interior Bridles'
            , name='Section - Interior Bridles',
716     numIntPts=5, poissonDefinition=DEFAULT, preIntegrate=OFF,
            temperature=
717     GRADIENT, thickness=b[2], thicknessField='',
            thicknessModulus=None,
718     thicknessType=UNIFORM, useDensity=OFF)
719 mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .SectionAssignment(offset=0.0,
720     offsetField='', offsetType=MIDDLE_SURFACE, region=
721     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].sets['
            Central Axis'],
722     sectionName='Section - Central Axis', thicknessAssignment=
            FROM_SECTION)
723 mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .SectionAssignment(offset=0.0,
724     offsetField='', offsetType=MIDDLE_SURFACE, region=
725     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].sets['
            Exterior Bridles'],
726     sectionName='Section - Exterior Bridles',
            thicknessAssignment=
727     FROM_SECTION)
728 mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .SectionAssignment(offset=0.0,
729     offsetField='', offsetType=MIDDLE_SURFACE, region=
730     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].sets['
            Interior Bridles'],
731     sectionName='Section - Interior Bridles',
            thicknessAssignment=
```

```
732     FROM_SECTION)
733
734 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(name='Points
    ', vertices=
735     mdb.models['CFRP Bridge'].parts['CFRP Bridge']
            .vertices.findAt(((xcoord[109],
736     ycoord[109], zcoord[109]), ), ((xcoord[230], ycoord[230],
            zcoord[230]), ), ((xcoord[234], ycoord[234],
737     zcoord[234]), ), ((xcoord[233], ycoord[233], zcoord[233]),
            ), ((xcoord[232], ycoord[232], zcoord[232]), ), ((xcoord
            [231], ycoord[231], zcoord[231]), ), ((xcoord[220],
            ycoord[220], zcoord[220]), ), ((xcoord[224], ycoord
            [224], zcoord[224]), ), ((xcoord[223], ycoord[223],
            zcoord[223]),
738     ), ((xcoord[222], ycoord[222], zcoord[222]), ), ((xcoord
            [221], ycoord[221], zcoord[221]),
739     ), ((xcoord[210], ycoord[210], zcoord[210]), ), ((xcoord
            [215], ycoord[215], zcoord[215]), ), ((
740     xcoord[213], ycoord[213], zcoord[213]), ), ((xcoord[212],
            ycoord[212], zcoord[212]), ), ((
741     xcoord[211], ycoord[211], zcoord[211]), ), ((xcoord[200],
            ycoord[200], zcoord[200]), ), ((
742     xcoord[204], ycoord[204], zcoord[204]), ), ((xcoord[203],
            ycoord[203], zcoord[203]), ), ((
743     xcoord[202], ycoord[202], zcoord[202]), ), ((xcoord[201],
            ycoord[201], zcoord[201]), ), ((
744     xcoord[190], ycoord[190], zcoord[190]), ), ((xcoord[194],
            ycoord[194], zcoord[194]), ), ((xcoord[193], ycoord
            [193], zcoord[193]), ), ((xcoord[192], ycoord[192],
            zcoord[192]), ), ((xcoord[191], ycoord[191], zcoord
            [191]), ), ((xcoord[172], ycoord[172], zcoord[172]), ),
            ((xcoord[113], ycoord[113], zcoord[113]), ), ((xcoord
            [86], ycoord[86], zcoord[86]), ), ((xcoord[74], ycoord
            [74], zcoord[74]), ), ((xcoord[65], ycoord[65], zcoord
```

```
        [65]), ), ((xcoord[0], ycoord[0], zcoord[0]), ), ))
745
746 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(edges=
747     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[0]-0.02, ycoord[0], zcoord[0]), ), ((-xcoord
        [0]+0.02, -ycoord[0], zcoord[0]), ), ((-xcoord[0]+0.02,
        ycoord[0], zcoord[0]), ), ((xcoord[0]-0.02, -ycoord[0],
        zcoord[0]), ), ), name=
748     'Bridles Supports')
749
750 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(edges=
751     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[381]+0.02,
752     ycoord[381], zcoord[381]), ), ((xcoord[371]+0.02, ycoord
        [371], zcoord[371]), ), ((
753     xcoord[361]+0.02, ycoord[361], zcoord[361]), ), ((xcoord
        [351]+0.02, ycoord[351], zcoord[351]), ),
754     ((xcoord[341]+0.02, ycoord[341], zcoord[341]), ), ((xcoord
        [331]-0.02, ycoord[331], zcoord[331]),
755     ), ((xcoord[321]-0.02, ycoord[321], zcoord[321]), ), ((
        xcoord[311]-0.02, ycoord[321], zcoord[321]),
756     ), ((xcoord[301]-0.02, ycoord[301], zcoord[301]), ), ((
        xcoord[291]-0.02, ycoord[291], zcoord[291]),
757     ), ((xcoord[281]+0.02, ycoord[281], zcoord[281]), ), ((
        xcoord[271]+0.02, ycoord[271], zcoord[271]),
758     ), ((xcoord[261]+0.02, ycoord[261], zcoord[261]), ), ((
        xcoord[251]+0.02, ycoord[251], zcoord[251]),
759     ), ((xcoord[241]+0.02, ycoord[241], zcoord[241]), ), ((
        xcoord[231]-0.02, ycoord[231], zcoord[231]),
760     ), ((xcoord[221]-0.02, ycoord[221], zcoord[221]), ), ((
        xcoord[211]-0.02, ycoord[211], zcoord[211]),
761     ), ((xcoord[201]-0.02, ycoord[201], zcoord[201]), ), ((
        xcoord[191]-0.02, ycoord[191], zcoord[191]),
762     ), ), name='Fine Mesh')
```

```
763
764 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(edges=
765     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            (((xcoord[380]+0.02,
766     ycoord[380], zcoord[380]), ), ((xcoord[370]+0.02, ycoord
            [370], zcoord[370]), ), ((
767     xcoord[360]+0.02, ycoord[360], zcoord[360]), ), ((xcoord
            [350]+0.02, ycoord[350], zcoord[350]), ),
768     ((xcoord[340]+0.02, ycoord[340], zcoord[340]), ), ((xcoord
            [330]-0.02, ycoord[330], zcoord[330]),
769     ), ((xcoord[320]-0.02, ycoord[320], zcoord[320]), ), ((
            xcoord[310]-0.02, ycoord[320], zcoord[320]),
770     ), ((xcoord[300]-0.02, ycoord[300], zcoord[300]), ), ((
            xcoord[290]-0.02, ycoord[290], zcoord[290]),
771     ), ((xcoord[280]+0.02, ycoord[280], zcoord[280]), ), ((
            xcoord[270]+0.02, ycoord[270], zcoord[270]),
772     ), ((xcoord[260]+0.02, ycoord[260], zcoord[260]), ), ((
            xcoord[250]+0.02, ycoord[250], zcoord[250]),
773     ), ((xcoord[240]+0.02, ycoord[240], zcoord[240]), ), ((
            xcoord[230]-0.02, ycoord[230], zcoord[230]),
774     ), ((xcoord[220]-0.02, ycoord[220], zcoord[220]), ), ((
            xcoord[210]-0.02, ycoord[210], zcoord[210]),
775     ), ((xcoord[200]-0.02, ycoord[200], zcoord[200]), ), ((
            xcoord[190]-0.02, ycoord[190], zcoord[190]),
776     ), ), name='Coarse Mesh - Upper')
777
778 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(edges=
779     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
            (((xcoord[382]+0.02,
780     ycoord[382], zcoord[382]), ), ((xcoord[372]+0.02, ycoord
            [372], zcoord[372]), ), ((
781     xcoord[362]+0.02, ycoord[362], zcoord[362]), ), ((xcoord
            [352]+0.02, ycoord[352], zcoord[352]), ),
```

```
782     ((xcoord[342]+0.02, ycoord[342], zcoord[342]), ), ((xcoord
            [332]-0.02, ycoord[332], zcoord[332]),
783     ), ((xcoord[322]-0.02, ycoord[322], zcoord[322]), ), ((
            xcoord[312]-0.02, ycoord[312], zcoord[312]),
784     ), ((xcoord[302]-0.02, ycoord[302], zcoord[302]), ), ((
            xcoord[292]-0.02, ycoord[292], zcoord[292]),
785     ), ((xcoord[282]+0.02, ycoord[282], zcoord[282]), ), ((
            xcoord[272]+0.02, ycoord[272], zcoord[272]),
786     ), ((xcoord[262]+0.02, ycoord[262], zcoord[262]), ), ((
            xcoord[252]+0.02, ycoord[252], zcoord[252]),
787     ), ((xcoord[242]+0.02, ycoord[242], zcoord[242]), ), ((
            xcoord[232]-0.02, ycoord[232], zcoord[232]),
788     ), ((xcoord[222]-0.02, ycoord[222], zcoord[222]), ), ((
            xcoord[212]-0.02, ycoord[212], zcoord[212]),
789     ), ((xcoord[202]-0.02, ycoord[202], zcoord[202]), ), ((
            xcoord[192]-0.02, ycoord[192], zcoord[192]),
790     ), ), name='Coarse Mesh - Lower')
791
792 mdb.models['CFRP Bridge'].Part(dimensionality=THREE_D, name='
        GFRP Beam',
793     type=DEFORMABLE_BODY)
794 mdb.models['CFRP Bridge'].parts['GFRP Beam'].ReferencePoint(
        point=(0.0, 0.0,
795     0.0))
796 for i in range(390, 423):
797     mdb.models['CFRP Bridge'].parts['GFRP Beam']
            .DatumPointByOffset(point=
798     mdb.models['CFRP Bridge'].parts['GFRP Beam']
            .referencePoints[1], vector=(
799     xcoord[i], ycoord[i], zcoord[i]))
800 for i in range(452, 485):
801     ini=int(inipoint[i])+1-390
802     end=int(endpoint[i])+1-390
```

```
803     mdb.models['CFRP Bridge'].parts['GFRP Beam'].WirePolyLine(
            mergeWire=OFF,
804     meshable=ON, points=((
805     mdb.models['CFRP Bridge'].parts['GFRP Beam'].datums[ini],
806     mdb.models['CFRP Bridge'].parts['GFRP Beam'].datums[end]),
            ))
807
808 mdb.models['CFRP Bridge'].parts['GFRP Beam'].BlendFaces(method=
        SHORTEST_PATH,
809     side1=(mdb.models['CFRP Bridge'].parts['GFRP Beam']
            .edges.findAt((xcoord[401]+(xcoord[402]-xcoord[401])/2,
810     ycoord[402]+(ycoord[401]-ycoord[402])/2, zcoord[402]+(
            zcoord[401]-zcoord[402])/2), ), ), side2=(
811     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[412]+(xcoord[413]-xcoord[412])/2,
812     ycoord[412]+(ycoord[413]-ycoord[412])/2, zcoord[412]+(
            zcoord[413]-zcoord[412])/2), ), ))
813 mdb.models['CFRP Bridge'].parts['GFRP Beam'].BlendFaces(method=
        SHORTEST_PATH,
814     side1=(mdb.models['CFRP Bridge'].parts['GFRP Beam']
            .edges.findAt((xcoord[390]+(xcoord[400]-xcoord[390])/2,
815     ycoord[390]+(ycoord[400]-ycoord[390])/2, zcoord[390]+(
            zcoord[400]-zcoord[390])/2),), ), side2=(
816     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[390]+(xcoord[391]-xcoord[390])/2,
817     ycoord[390]+(ycoord[391]-ycoord[390])/2, zcoord[390]+(
            zcoord[391]-zcoord[390])/2), ), mdb.models['CFRP Bridge'
            ].parts['GFRP Beam'].edges.findAt((xcoord[391]+(xcoord
            [392]-xcoord[391])/2,
818     ycoord[391]+(ycoord[392]-ycoord[391])/2, zcoord[392]+(
            zcoord[391]-zcoord[392])/2),),
819     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[392]+(xcoord[393]-xcoord[392])/2,
```

```
820     ycoord[392]+(ycoord[393]-ycoord[392])/2, zcoord[393]+(
            zcoord[392]-zcoord[393])/2), ), mdb.models['CFRP Bridge'
            ].parts['GFRP Beam'].edges.findAt((xcoord[394]+(xcoord
            [393]-xcoord[394])/2,
821     ycoord[394]+(ycoord[393]-ycoord[394])/2, zcoord[394]+(
            zcoord[393]-zcoord[394])/2),),
822     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[395]+(xcoord[394]-xcoord[395])/2,
823     ycoord[395]+(ycoord[394]-ycoord[395])/2, zcoord[395]+(
            zcoord[394]-zcoord[395])/2), ), mdb.models['CFRP Bridge'
            ].parts['GFRP Beam'].edges.findAt((xcoord[396]+(xcoord
            [395]-xcoord[396])/2,
824     ycoord[396]+(ycoord[395]-ycoord[396])/2, zcoord[396]+(
            zcoord[395]-zcoord[396])/2),),
825     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[397]+(xcoord[396]-xcoord[397])/2,
826     ycoord[397]+(ycoord[396]-ycoord[397])/2, zcoord[397]+(
            zcoord[396]-zcoord[397])/2), ), mdb.models['CFRP Bridge'
            ].parts['GFRP Beam'].edges.findAt((xcoord[398]+(xcoord
            [397]-xcoord[398])/2,
827     ycoord[398]+(ycoord[397]-ycoord[398])/2, zcoord[398]+(
            zcoord[397]-zcoord[398])/2),),
828     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[399]+(xcoord[398]-xcoord[399])/2,
829     ycoord[399]+(ycoord[398]-ycoord[399])/2, zcoord[399]+(
            zcoord[398]-zcoord[399])/2), ), mdb.models['CFRP Bridge'
            ].parts['GFRP Beam'].edges.findAt((xcoord[400]+(xcoord
            [399]-xcoord[400])/2,
830     ycoord[400]+(ycoord[399]-ycoord[400])/2, zcoord[399]+(
            zcoord[400]-zcoord[399])/2),)))
831 mdb.models['CFRP Bridge'].parts['GFRP Beam'].BlendFaces(method=
        SHORTEST_PATH,
832     side1=(mdb.models['CFRP Bridge'].parts['GFRP Beam']
            .edges.findAt((xcoord[401]+(xcoord[403]-xcoord[401])/2,
```

```
833     ycoord[401]+(ycoord[403]-ycoord[401])/2, zcoord[401]+(
            zcoord[403]-zcoord[401])/2),),
834     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[403]+(xcoord[404]-xcoord[403])/2,
835     ycoord[403]+(ycoord[404]-ycoord[403])/2, zcoord[403]+(
            zcoord[404]-zcoord[403])/2), ),
836     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[404]+(xcoord[405]-xcoord[404])/2,
837     ycoord[404]+(ycoord[405]-ycoord[404])/2, zcoord[404]+(
            zcoord[405]-zcoord[404])/2), ),
838     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[405]+(xcoord[406]-xcoord[405])/2,
839     ycoord[405]+(ycoord[406]-ycoord[405])/2, zcoord[405]+(
            zcoord[406]-zcoord[405])/2), ),
840     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[406]+(xcoord[407]-xcoord[406])/2,
841     ycoord[406]+(ycoord[407]-ycoord[406])/2, zcoord[406]+(
            zcoord[407]-zcoord[406])/2), ),
842     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[407]+(xcoord[408]-xcoord[407])/2,
843     ycoord[407]+(ycoord[408]-ycoord[407])/2, zcoord[407]+(
            zcoord[408]-zcoord[407])/2), ),
844     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[408]+(xcoord[409]-xcoord[408])/2,
845     ycoord[408]+(ycoord[409]-ycoord[408])/2, zcoord[408]+(
            zcoord[409]-zcoord[408])/2), ),
846     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[409]+(xcoord[410]-xcoord[409])/2,
847     ycoord[409]+(ycoord[410]-ycoord[409])/2, zcoord[409]+(
            zcoord[410]-zcoord[409])/2), ),
848     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[410]+(xcoord[411]-xcoord[410])/2,
849     ycoord[410]+(ycoord[411]-ycoord[410])/2, zcoord[410]+(
            zcoord[411]-zcoord[410])/2),),
```

```
850     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[411]+(xcoord[402]-xcoord[411])/2,
851     ycoord[411]+(ycoord[402]-ycoord[411])/2, zcoord[411]+(
            zcoord[402]-zcoord[411])/2), )), side2=(
852     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[412]+(xcoord[414]-xcoord[412])/2,
853     ycoord[412]+(ycoord[414]-ycoord[412])/2, zcoord[412]+(
            zcoord[414]-zcoord[412])/2), ),
854     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[414]+(xcoord[415]-xcoord[414])/2,
855     ycoord[414]+(ycoord[415]-ycoord[414])/2, zcoord[414]+(
            zcoord[415]-zcoord[414])/2),  ),
856     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[415]+(xcoord[416]-xcoord[415])/2,
857     ycoord[415]+(ycoord[416]-ycoord[415])/2, zcoord[415]+(
            zcoord[416]-zcoord[415])/2),),
858     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[416]+(xcoord[417]-xcoord[416])/2,
859     ycoord[416]+(ycoord[417]-ycoord[416])/2, zcoord[416]+(
            zcoord[417]-zcoord[416])/2),  ),
860     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[417]+(xcoord[418]-xcoord[417])/2,
861     ycoord[417]+(ycoord[418]-ycoord[417])/2, zcoord[417]+(
            zcoord[418]-zcoord[417])/2),  ),
862     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[418]+(xcoord[419]-xcoord[418])/2,
863     ycoord[418]+(ycoord[419]-ycoord[418])/2, zcoord[418]+(
            zcoord[419]-zcoord[418])/2),  ),
864     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[419]+(xcoord[420]-xcoord[419])/2,
865     ycoord[419]+(ycoord[420]-ycoord[419])/2, zcoord[419]+(
            zcoord[420]-zcoord[419])/2),  ),
866     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[420]+(xcoord[421]-xcoord[420])/2,
```

```
867     ycoord[420]+(ycoord[421]-ycoord[420])/2, zcoord[420]+(
            zcoord[421]-zcoord[420])/2),),
868     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[421]+(xcoord[422]-xcoord[421])/2,
869     ycoord[421]+(ycoord[422]-ycoord[421])/2, zcoord[421]+(
            zcoord[422]-zcoord[421])/2),),
870     mdb.models['CFRP Bridge'].parts['GFRP Beam'].edges.findAt((
            xcoord[422]+(xcoord[413]-xcoord[422])/2,
871     ycoord[422]+(ycoord[413]-ycoord[422])/2, zcoord[422]+(
            zcoord[413]-zcoord[422])/2),)))
872
873 mdb.models['CFRP Bridge'].parts['GFRP Beam'].Set(faces=
874     mdb.models['CFRP Bridge'].parts['GFRP Beam'].faces.findAt
            (((5.004167,
875     0.30303, -0.316345), ), ((4.995833, -0.30303, -0.316345), )
            , ), name=
876     'Top Flange')
877 mdb.models['CFRP Bridge'].parts['GFRP Beam'].Set(faces=
878     mdb.models['CFRP Bridge'].parts['GFRP Beam'].faces.findAt
            (((4.997917,
879     0.707071, -0.384366), ), ((4.997917, 0.505051, -0.438782),
            ), ((4.997917,
880     0.252525, -0.484695), ), ((4.997917, 0.126263, -0.496174),
            ), ((4.997917,
881     0.050505, -0.49915), ), ((4.997917, -0.025253, -0.499575),
            ), ((4.997917,
882     -0.10101, -0.497449), ), ((4.997917, -0.20202, -0.489797),
            ), ((4.997917,
883     -0.40404, -0.459188), ), ((4.997917, -0.707071, -0.384366),
             ), ((5.002083,
884     -0.707071, -0.384366), ), ((5.002083, -0.505051, -0.438782)
            , ), ((5.002083,
885     -0.252525, -0.484695), ), ((5.002083, -0.126263, -0.496174)
            , ), ((5.002083,
```

```
886     -0.050505, -0.49915), ), ((5.002083, 0.025253, -0.499575),
            ), ((5.002083,
887     0.10101, -0.497449), ), ((5.002083, 0.20202, -0.489797), ),
             ((5.002083,
888     0.40404, -0.459188), ), ((5.002083, 0.707071, -0.384366), )
            , ), name=
889     'Bottom Flange')
890 mdb.models['CFRP Bridge'].parts['GFRP Beam'].Set(faces=
891     mdb.models['CFRP Bridge'].parts['GFRP Beam'].faces.findAt
            (((5.0, -0.702777,
892     -0.33335), ), ((5.0, -0.498474, -0.33335), ), ((5.0, -0
            .197864, -0.346104),
893     ), ((5.0, -0.098803, -0.346742), ), ((5.0, -0.024614, -0
            .346954), ), ((5.0,
894     0.024614, -0.346954), ), ((5.0, 0.098803, -0.346742), ),
            ((5.0, 0.197864,
895     -0.346104), ), ((5.0, 0.498474, -0.33335), ), ((5.0, 0
            .702777, -0.33335),
896     ), ), name='Web')
897
898
899 mdb.models['CFRP Bridge'].Part(dimensionality=THREE_D, name='
        GFRP Plate',
900     type=DEFORMABLE_BODY)
901 mdb.models['CFRP Bridge'].parts['GFRP Plate'].ReferencePoint(
        point=(0.0, 0.0,
902     0.0))
903 for i in range(423, 427):
904     mdb.models['CFRP Bridge'].parts['GFRP Plate']
            .DatumPointByOffset(point=
905     mdb.models['CFRP Bridge'].parts['GFRP Plate']
            .referencePoints[1], vector=(
906     xcoord[i], ycoord[i], zcoord[i]))
907 for i in range(485, 487):
```

```
908     ini=int(inipoint[i])+1-423
909     end=int(endpoint[i])+1-423
910     mdb.models['CFRP Bridge'].parts['GFRP Plate'].WirePolyLine(
            mergeWire=OFF,
911     meshable=ON, points=((
912     mdb.models['CFRP Bridge'].parts['GFRP Plate'].datums[ini],
913     mdb.models['CFRP Bridge'].parts['GFRP Plate'].datums[end]),
             ))
914
915 mdb.models['CFRP Bridge'].parts['GFRP Plate'].BlendFaces(method
        =SHORTEST_PATH,
916     side1=(mdb.models['CFRP Bridge'].parts['GFRP Plate']
            .edges.findAt((xcoord[423]+(xcoord[424]-xcoord[423])/2,
917     ycoord[423]+(ycoord[424]-ycoord[423])/2, zcoord[423]+(
            zcoord[424]-zcoord[423])/2),), ), side2=(
918     mdb.models['CFRP Bridge'].parts['GFRP Plate'].edges.findAt
            ((xcoord[425]+(xcoord[426]-xcoord[425])/2,
919     ycoord[425]+(ycoord[426]-ycoord[425])/2, zcoord[425]+(
            zcoord[426]-zcoord[425])/2),), ))
920
921 mdb.models['CFRP Bridge'].parts['GFRP Plate'].Set(faces=
922     mdb.models['CFRP Bridge'].parts['GFRP Plate'].faces.findAt
            (((xcoord[423]+(xcoord[426]-xcoord[423])/2,
923     ycoord[423]+(ycoord[426]-ycoord[423])/2, zcoord[423]+(
            zcoord[426]-zcoord[423])/2), )), name='Plate')
924
925
926 mdb.models['CFRP Bridge'].Material(name='GFRP - Stiffeners')
927 mdb.models['CFRP Bridge'].materials['GFRP - Stiffeners']
        .Density(table=((1.7,
928     ), ))
929 mdb.models['CFRP Bridge'].materials['GFRP - Stiffeners']
        .Elastic(table=((
930     e[3], 0.3), ))
```

```
931 mdb.models['CFRP Bridge'].Material(name='GFRP - Plate')
932 mdb.models['CFRP Bridge'].materials['GFRP - Plate'].Density(
        table=((1.7,
933      ), ))
934 mdb.models['CFRP Bridge'].materials['GFRP - Plate'].Elastic(
        table=((
935      e[4], 0.3), ))
936
937
938 mdb.models['CFRP Bridge'].HomogeneousShellSection(idealization=
        NO_IDEALIZATION,
939      integrationRule=SIMPSON, material='GFRP - Stiffeners', name
             ='Section - GFRP',
940      numIntPts=5, poissonDefinition=DEFAULT, preIntegrate=OFF,
             temperature=
941      GRADIENT, thickness=b[3], thicknessField='',
             thicknessModulus=None,
942      thicknessType=UNIFORM, useDensity=OFF)
943 mdb.models['CFRP Bridge'].HomogeneousShellSection(idealization=
        NO_IDEALIZATION,
944      integrationRule=SIMPSON, material='GFRP - Plate', name='
             Section - GFRP Plate',
945      numIntPts=5, poissonDefinition=DEFAULT, preIntegrate=OFF,
             temperature=
946      GRADIENT, thickness=b[4], thicknessField='',
             thicknessModulus=None,
947      thicknessType=UNIFORM, useDensity=OFF)
948
949
950 mdb.models['CFRP Bridge'].parts['GFRP Beam'].SectionAssignment(
        offset=0.0,
951      offsetField='', offsetType=BOTTOM_SURFACE, region=
952      mdb.models['CFRP Bridge'].parts['GFRP Beam'].sets['Top
             Flange'],
```

```
953     sectionName='Section - GFRP', thicknessAssignment=
954     FROM_SECTION)
955 mdb.models['CFRP Bridge'].parts['GFRP Beam'].SectionAssignment(
        offset=0.0,
956     offsetField='', offsetType=MIDDLE_SURFACE, region=
957     mdb.models['CFRP Bridge'].parts['GFRP Beam'].sets['Web'],
958     sectionName='Section - GFRP', thicknessAssignment=
959     FROM_SECTION)
960 mdb.models['CFRP Bridge'].parts['GFRP Beam'].SectionAssignment(
        offset=0.0,
961     offsetField='', offsetType=TOP_SURFACE, region=
962     mdb.models['CFRP Bridge'].parts['GFRP Beam'].sets['Bottom
            Flange'],
963     sectionName='Section - GFRP', thicknessAssignment=
964     FROM_SECTION)
965 mdb.models['CFRP Bridge'].parts['GFRP Plate'].SectionAssignment
        (offset=0.0,
966     offsetField='', offsetType=MIDDLE_SURFACE, region=
967     mdb.models['CFRP Bridge'].parts['GFRP Plate'].sets['Plate'
            ],
968     sectionName='Section - GFRP Plate', thicknessAssignment=
969     FROM_SECTION)
970
971
972 mdb.models['CFRP Bridge'].Part(dimensionality=THREE_D, name='
        GFRP Stiffener',
973     type=DEFORMABLE_BODY)
974 mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
        .ReferencePoint(point=(0.0, 0.0,
975     0.0))
976 for i in range(427, 451):
977     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
            .DatumPointByOffset(point=
```

```
978     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
            .referencePoints[1], vector=(
979     xcoord[i], ycoord[i], zcoord[i]))
980 for i in range(487, 499):
981     ini=int(inipoint[i])+1-427
982     end=int(endpoint[i])+1-427
983     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
            .WirePolyLine(mergeWire=OFF,
984     meshable=ON, points=((
985     mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].datums[
            ini],
986     mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].datums[
            end]), ))
987
988 mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].BlendFaces(
        method=
989     SHORTEST_PATH, side1=
990     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
            .edges.findAt(((xcoord[429]+(xcoord[430]-xcoord[429])/2,
991     ycoord[429]+(ycoord[430]-ycoord[429])/2, zcoord[429]+(
            zcoord[430]-zcoord[429])/2), ), ), side2=
992     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
            .edges.findAt(((xcoord[431]+(xcoord[432]-xcoord[431])/2,
993     ycoord[431]+(ycoord[432]-ycoord[431])/2, zcoord[431]+(
            zcoord[432]-zcoord[431])/2),),))
994 mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].BlendFaces(
        method=
995     SHORTEST_PATH, side1=
996     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
            .edges.findAt(((xcoord[435]+(xcoord[436]-xcoord[435])/2,
997     ycoord[435]+(ycoord[436]-ycoord[435])/2, zcoord[435]+(
            zcoord[436]-zcoord[435])/2), ), ), side2=
998     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
            .edges.findAt
```

```
999 (((xcoord[437]+(xcoord[438]-xcoord[437])/2,
1000     ycoord[437]+(ycoord[438]-ycoord[437])/2, zcoord[437]+(
             zcoord[438]-zcoord[437])/2),), ))
1001 mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].BlendFaces(
         method=
1002     SHORTEST_PATH, side1=
1003     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
             .edges.findAt(((xcoord[441]+(xcoord[442]-xcoord[441])/2,
1004     ycoord[441]+(ycoord[442]-ycoord[441])/2, zcoord[441]+(
             zcoord[442]-zcoord[441])/2), ), ), side2=
1005     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
             .edges.findAt
1006 (((xcoord[443]+(xcoord[444]-xcoord[443])/2,
1007     ycoord[443]+(ycoord[444]-ycoord[443])/2, zcoord[443]+(
             zcoord[444]-zcoord[443])/2),), ))
1008 mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].BlendFaces(
         method=
1009     SHORTEST_PATH, side1=
1010     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
             .edges.findAt(((xcoord[447]+(xcoord[448]-xcoord[447])/2,
1011     ycoord[447]+(ycoord[448]-ycoord[447])/2, zcoord[447]+(
             zcoord[448]-zcoord[447])/2), ), ), side2=
1012     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
             .edges.findAt
1013 (((xcoord[449]+(xcoord[450]-xcoord[449])/2,
1014     ycoord[449]+(ycoord[450]-ycoord[449])/2, zcoord[449]+(
             zcoord[450]-zcoord[449])/2),), ))
1015 mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].BlendFaces(
         method=
1016     SHORTEST_PATH, side1=
1017     mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
             .edges.findAt(((xcoord[429]+(xcoord[430]-xcoord[429])/2,
1018     ycoord[429]+(ycoord[430]-ycoord[429])/2, zcoord[429]+(
             zcoord[430]-zcoord[429])/2), ), ), side2=
```

```
1019      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .edges.findAt
1020  (((xcoord[441]+(xcoord[442]-xcoord[441])/2,
1021      ycoord[441]+(ycoord[442]-ycoord[441])/2, zcoord[441]+(
              zcoord[442]-zcoord[441])/2),), ))
1022  mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].BlendFaces(
          method=
1023      SHORTEST_PATH, side1=
1024      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .edges.findAt(((xcoord[431]+(xcoord[432]-xcoord[431])/2,
1025      ycoord[431]+(ycoord[432]-ycoord[431])/2, zcoord[431]+(
              zcoord[432]-zcoord[431])/2), ), ), side2=
1026      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .edges.findAt
1027  (((xcoord[443]+(xcoord[444]-xcoord[443])/2,
1028      ycoord[443]+(ycoord[444]-ycoord[443])/2, zcoord[443]+(
              zcoord[444]-zcoord[443])/2),), ))
1029  mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].BlendFaces(
          method=
1030      SHORTEST_PATH, side1=
1031      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .edges.findAt(((xcoord[435]+(xcoord[436]-xcoord[435])/2,
1032      ycoord[435]+(ycoord[436]-ycoord[435])/2, zcoord[435]+(
              zcoord[436]-zcoord[435])/2), ), ), side2=
1033      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .edges.findAt
1034  (((xcoord[447]+(xcoord[448]-xcoord[447])/2,
1035      ycoord[447]+(ycoord[448]-ycoord[447])/2, zcoord[447]+(
              zcoord[448]-zcoord[447])/2),), ))
1036  mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].BlendFaces(
          method=
1037      SHORTEST_PATH, side1=
1038      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .edges.findAt(((xcoord[437]+(xcoord[438]-xcoord[437])/2,
```

```
1039      ycoord[437]+(ycoord[438]-ycoord[437])/2, zcoord[437]+(
              zcoord[438]-zcoord[437])/2), ), ), side2=
1040      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .edges.findAt
1041  (((xcoord[449]+(xcoord[450]-xcoord[449])/2,
1042      ycoord[449]+(ycoord[450]-ycoord[449])/2, zcoord[449]+(
              zcoord[450]-zcoord[449])/2),), ))
1043
1044  mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].Set(faces=
1045      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .faces.findAt(((4.5,
1046      -0.456629, -0.448564), ), ((4.0, -0.452462, -0.449406), ),
              ((4.5, 0.456629,
1047      -0.448564), ), ((4.0, 0.452462, -0.449406), ), ), name='
              Bottom Flange')
1048  mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].Set(faces=
1049      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .faces.findAt(((4.5,
1050      -0.458712, -0.316345), ), ((4.0, -0.450379, -0.316345), ),
              ((4.5, 0.458712,
1051      -0.316345), ), ((4.0, 0.450379, -0.316345), ), ), name='Top
               Flange')
1052  mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].Set(faces=
1053      mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
              .faces.findAt(((4.0,
1054      -0.442045, -0.428983), ), ((4.5, -0.467045, -0.424774), ),
              ((4.0, 0.442045,
1055      -0.428983), ), ((4.5, 0.467045, -0.424774), ), ), name='Web
              ')
1056
1057  mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
         .SectionAssignment(offset=0.0,
1058      offsetField='', offsetType=BOTTOM_SURFACE, region=
```

```
1059      mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].sets['Top
              Flange'],
1060      sectionName='Section - GFRP', thicknessAssignment=
1061      FROM_SECTION)
1062  mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
          .SectionAssignment(offset=0.0,
1063      offsetField='', offsetType=MIDDLE_SURFACE, region=
1064      mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].sets['Web
              '],
1065      sectionName='Section - GFRP', thicknessAssignment=
1066      FROM_SECTION)
1067
1068  mdb.models['CFRP Bridge'].parts['GFRP Stiffener']
          .SectionAssignment(offset=0.0,
1069      offsetField='', offsetType=TOP_SURFACE, region=
1070      mdb.models['CFRP Bridge'].parts['GFRP Stiffener'].sets['
              Bottom Flange'],
1071      sectionName='Section - GFRP', thicknessAssignment=
1072      FROM_SECTION)
1073
1074  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Surface(name='
          Bridles',
1075      side12Faces=
1076      mdb.models['CFRP Bridge'].parts['CFRP Bridge']
              .faces.getSequenceFromMask((
1077      '[#9e79e7ff #e79e79e7 #79e79e79 #9e79e79e #1fff ]', ), ))
1078
1079  mdb.models['CFRP Bridge'].rootAssembly.DatumCsysByDefault(
          CARTESIAN)
1080  mdb.models['CFRP Bridge'].rootAssembly.Instance(dependent=OFF,
          name=
1081      'GFRP Beam-1', part=mdb.models['CFRP Bridge'].parts['GFRP
              Beam'])
```

```
mdb.models['CFRP Bridge'].rootAssembly.LinearInstancePattern(direction1=(-1.0,
    0.0, 0.0), direction2=(0.0, 1.0, 0.0), instanceList=('GFRP Beam-1', ),
    number1=2, number2=1, spacing1=1.5, spacing2=1.81818)
mdb.models['CFRP Bridge'].rootAssembly.LinearInstancePattern(direction1=(-1.0,
    0.0, 0.0), direction2=(0.0, 1.0, 0.0), instanceList=('GFRP Beam-1-lin-2-1',
    ), number1=15, number2=1, spacing1=0.5, spacing2=1.81818)
mdb.models['CFRP Bridge'].rootAssembly.LinearInstancePattern(direction1=(-1.0,
    0.0, 0.0), direction2=(0.0, 1.0, 0.0), instanceList=('GFRP Beam-1', ),
    number1=2, number2=1, spacing1=10.0, spacing2=1.81818)
mdb.models['CFRP Bridge'].rootAssembly.Instance(dependent=OFF, name=
    'GFRP Stiffener-1', part=mdb.models['CFRP Bridge'].parts['GFRP Stiffener'])
mdb.models['CFRP Bridge'].rootAssembly.LinearInstancePattern(direction1=(-1.0,
    0.0, 0.0), direction2=(0.0, 1.0, 0.0), instanceList=('GFRP Stiffener-1', ),
    number1=2, number2=1, spacing1=8.5, spacing2=0.934091)
mdb.models['CFRP Bridge'].rootAssembly.Instance(dependent=OFF, name=
    'GFRP Plate-1', part=mdb.models['CFRP Bridge'].parts['GFRP Plate'])
mdb.models['CFRP Bridge'].rootAssembly.InstanceFromBooleanMerge(domain=GEOMETRY
    , instances=(
    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam-1'],
```

```
1101    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1'],
1102    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-2-1'],
1103    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-3-1'],
1104    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-4-1'],
1105    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-5-1'],
1106    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-6-1'],
1107    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-7-1'],
1108    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-8-1'],
1109    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-9-1'],
1110    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-10-1'],
1111    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-11-1'],
1112    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-12-1'],
1113    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-13-1'],
1114    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-14-1'],
1115    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-lin-15-1'],
1116    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Beam
            -1-lin-2-1-1'],
1117    mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP
            Stiffener-1'],
```

```
1118     mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP
             Stiffener-1-lin-2-1'],
1119     mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP
             Plate-1']), name=
1120     'GFRP Deck', originalInstances=DELETE)
1121 del mdb.models['CFRP Bridge'].rootAssembly.features['GFRP Deck
        -1']
1122
1123 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(name='Points
        ', vertices=
1124     mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .vertices.findAt(((xcoord[109],
1125     ycoord[109], zcoord[109]), ), ((xcoord[230], ycoord[230],
             zcoord[230]), ), ((xcoord[234], ycoord[234],
1126     zcoord[234]), ), ((xcoord[233], ycoord[233], zcoord[233]),
             ), ((xcoord[232], ycoord[232], zcoord[232]), ), ((xcoord
             [231], ycoord[231], zcoord[231]), ), ((xcoord[220],
             ycoord[220], zcoord[220]), ), ((xcoord[224], ycoord
             [224], zcoord[224]), ), ((xcoord[223], ycoord[223],
             zcoord[223]),
1127     ), ((xcoord[222], ycoord[222], zcoord[222]), ), ((xcoord
             [221], ycoord[221], zcoord[221]),
1128     ), ((xcoord[210], ycoord[210], zcoord[210]), ), ((xcoord
             [215], ycoord[215], zcoord[215]), ), ((
1129     xcoord[213], ycoord[213], zcoord[213]), ), ((xcoord[212],
             ycoord[212], zcoord[212]), ), ((
1130     xcoord[211], ycoord[211], zcoord[211]), ), ((xcoord[200],
             ycoord[200], zcoord[200]), ), ((
1131     xcoord[204], ycoord[204], zcoord[204]), ), ((xcoord[203],
             ycoord[203], zcoord[203]), ), ((
1132     xcoord[202], ycoord[202], zcoord[202]), ), ((xcoord[201],
             ycoord[201], zcoord[201]), ), ((
1133     xcoord[190], ycoord[190], zcoord[190]), ), ((xcoord[194],
             ycoord[194], zcoord[194]), ), ((xcoord[193], ycoord
```

```
        [193], zcoord[193]), ), ((xcoord[192], ycoord[192],
        zcoord[192]), ), ((xcoord[191], ycoord[191], zcoord
        [191]), ), ((xcoord[172], ycoord[172], zcoord[172]), ),
        ((xcoord[113], ycoord[113], zcoord[113]), ), ((xcoord
        [86], ycoord[86], zcoord[86]), ), ((xcoord[74], ycoord
        [74], zcoord[74]), ), ((xcoord[65], ycoord[65], zcoord
        [65]), ), ((xcoord[0], ycoord[0], zcoord[0]), ), ))
1134
1135 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(edges=
1136     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[0]-0.02, ycoord[0], zcoord[0]), ), ((-xcoord
        [0]+0.02, -ycoord[0], zcoord[0]), ), ((-xcoord[0]+0.02,
        ycoord[0], zcoord[0]), ), ((xcoord[0]-0.02, -ycoord[0],
        zcoord[0]), ), ), name=
1137     'Bridles Supports')
1138
1139 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(edges=
1140     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
        (((xcoord[381]+0.02,
1141     ycoord[381], zcoord[381]), ), ((xcoord[371]+0.02, ycoord
        [371], zcoord[371]), ), ((
1142     xcoord[361]+0.02, ycoord[361], zcoord[361]), ), ((xcoord
        [351]+0.02, ycoord[351], zcoord[351]), ),
1143     ((xcoord[341]+0.02, ycoord[341], zcoord[341]), ), ((xcoord
        [331]-0.02, ycoord[331], zcoord[331]),
1144     ), ((xcoord[321]-0.02, ycoord[321], zcoord[321]), ), ((
        xcoord[311]-0.02, ycoord[321], zcoord[321]),
1145     ), ((xcoord[301]-0.02, ycoord[301], zcoord[301]), ), ((
        xcoord[291]-0.02, ycoord[291], zcoord[291]),
1146     ), ((xcoord[281]+0.02, ycoord[281], zcoord[281]), ), ((
        xcoord[271]+0.02, ycoord[271], zcoord[271]),
1147     ), ((xcoord[261]+0.02, ycoord[261], zcoord[261]), ), ((
        xcoord[251]+0.02, ycoord[251], zcoord[251]),
```

```
1148     ), ((xcoord[241]+0.02, ycoord[241], zcoord[241]), ), ((
             xcoord[231]-0.02, ycoord[231], zcoord[231]),
1149     ), ((xcoord[221]-0.02, ycoord[221], zcoord[221]), ), ((
             xcoord[211]-0.02, ycoord[211], zcoord[211]),
1150     ), ((xcoord[201]-0.02, ycoord[201], zcoord[201]), ), ((
             xcoord[191]-0.02, ycoord[191], zcoord[191]),
1151     ), ), name='Fine Mesh')
1152
1153 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(edges=
1154     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             (((xcoord[380]+0.02,
1155     ycoord[380], zcoord[380]), ), ((xcoord[370]+0.02, ycoord
             [370], zcoord[370]), ), ((
1156     xcoord[360]+0.02, ycoord[360], zcoord[360]), ), ((xcoord
             [350]+0.02, ycoord[350], zcoord[350]), ),
1157     ((xcoord[340]+0.02, ycoord[340], zcoord[340]), ), ((xcoord
             [330]-0.02, ycoord[330], zcoord[330]),
1158     ), ((xcoord[320]-0.02, ycoord[320], zcoord[320]), ), ((
             xcoord[310]-0.02, ycoord[320], zcoord[320]),
1159     ), ((xcoord[300]-0.02, ycoord[300], zcoord[300]), ), ((
             xcoord[290]-0.02, ycoord[290], zcoord[290]),
1160     ), ((xcoord[280]+0.02, ycoord[280], zcoord[280]), ), ((
             xcoord[270]+0.02, ycoord[270], zcoord[270]),
1161     ), ((xcoord[260]+0.02, ycoord[260], zcoord[260]), ), ((
             xcoord[250]+0.02, ycoord[250], zcoord[250]),
1162     ), ((xcoord[240]+0.02, ycoord[240], zcoord[240]), ), ((
             xcoord[230]-0.02, ycoord[230], zcoord[230]),
1163     ), ((xcoord[220]-0.02, ycoord[220], zcoord[220]), ), ((
             xcoord[210]-0.02, ycoord[210], zcoord[210]),
1164     ), ((xcoord[200]-0.02, ycoord[200], zcoord[200]), ), ((
             xcoord[190]-0.02, ycoord[190], zcoord[190]),
1165     ), ), name='Coarse Mesh - Upper')
1166
1167 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].Set(edges=
```

```
1168     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].edges.findAt
             (((xcoord[382]+0.02,
1169     ycoord[382], zcoord[382]), ), ((xcoord[372]+0.02, ycoord
             [372], zcoord[372]), ), ((
1170     xcoord[362]+0.02, ycoord[362], zcoord[362]), ), ((xcoord
             [352]+0.02, ycoord[352], zcoord[352]), ),
1171     ((xcoord[342]+0.02, ycoord[342], zcoord[342]), ), ((xcoord
             [332]-0.02, ycoord[332], zcoord[332]),
1172     ), ((xcoord[322]-0.02, ycoord[322], zcoord[322]), ), ((
             xcoord[312]-0.02, ycoord[312], zcoord[312]),
1173     ), ((xcoord[302]-0.02, ycoord[302], zcoord[302]), ), ((
             xcoord[292]-0.02, ycoord[292], zcoord[292]),
1174     ), ((xcoord[282]+0.02, ycoord[282], zcoord[282]), ), ((
             xcoord[272]+0.02, ycoord[272], zcoord[272]),
1175     ), ((xcoord[262]+0.02, ycoord[262], zcoord[262]), ), ((
             xcoord[252]+0.02, ycoord[252], zcoord[252]),
1176     ), ((xcoord[242]+0.02, ycoord[242], zcoord[242]), ), ((
             xcoord[232]-0.02, ycoord[232], zcoord[232]),
1177     ), ((xcoord[222]-0.02, ycoord[222], zcoord[222]), ), ((
             xcoord[212]-0.02, ycoord[212], zcoord[212]),
1178     ), ((xcoord[202]-0.02, ycoord[202], zcoord[202]), ), ((
             xcoord[192]-0.02, ycoord[192], zcoord[192]),
1179     ), ), name='Coarse Mesh - Lower')
1180
1181 mdb.models['CFRP Bridge'].parts['GFRP Deck'].Set(faces=
1182     mdb.models['CFRP Bridge'].parts['GFRP Deck']
             .faces.getSequenceFromMask((
1183     '[#0:7 #c0000000 #3ffffc0c #0:7 #c0000000 #3ffffc0c ]', ),
             ), name=
1184     'Deck Supports')
1185 mdb.models['CFRP Bridge'].parts['GFRP Deck'].Surface(name='
        Bottom Flanges',
1186     side2Faces=
```

```
1187     mdb.models['CFRP Bridge'].parts['GFRP Deck']
             .faces.getSequenceFromMask((
1188     '[#fff00000 #fff000ff:6 #c00000ff #3ffffc0c #ffe0cc00 #
             ffe001ff:3 #e3e001ff',
1189     ' #ffe001ff:2 #c00001ff #3ffffc0c #ffe0cc00 #1ff ]'), ))
1190 mdb.models['CFRP Bridge'].parts['GFRP Deck'].Surface(name='
        Plate', side2Faces=
1191     mdb.models['CFRP Bridge'].parts['GFRP Deck']
             .faces.getSequenceFromMask((
1192     '[#fffff ]', ), ))
1193
1194 mdb.models['CFRP Bridge'].parts['CFRP Bridge']
        .MaterialOrientation(
1195     additionalRotationType=ROTATION_NONE, axis=AXIS_1,
             fieldName='', localCsys=
1196     None, orientationType=GLOBAL, region=Region(
1197     faces=mdb.models['CFRP Bridge'].parts['CFRP Bridge']
             .faces.getSequenceFromMask(
1198     mask=('[#ffffffff:4 #1ffff ]', ), )))
1199 mdb.models['CFRP Bridge'].parts['GFRP Deck']
        .MaterialOrientation(
1200     additionalRotationType=ROTATION_NONE, axis=AXIS_1,
             fieldName='', localCsys=
1201     None, orientationType=GLOBAL, region=Region(
1202     faces=mdb.models['CFRP Bridge'].parts['GFRP Deck']
             .faces.getSequenceFromMask(
1203     mask=('[#ffffffff:18 #1fffffff ]', ), )))
1204
1205 mdb.models['CFRP Bridge'].rootAssembly.Instance(dependent=ON,
        name=
1206     'CFRP Bridge-1', part=mdb.models['CFRP Bridge'].parts['CFRP
              Bridge'])
1207 mdb.models['CFRP Bridge'].rootAssembly.Instance(dependent=ON,
        name=
```

```
1208     'GFRP Deck-1', part=mdb.models['CFRP Bridge'].parts['GFRP
             Deck'])
1209
1210 mdb.models['CFRP Bridge'].StaticStep(name='Installation',
         nlgeom=ON, previous=
1211     'Initial')
1212 mdb.models['CFRP Bridge'].StaticStep(name='Traffic Load',
         previous=
1213     'Installation')
1214
1215 mdb.models['CFRP Bridge'].Tie(adjust=ON, master=
1216     mdb.models['CFRP Bridge'].rootAssembly.instances['CFRP
             Bridge-1'].surfaces['Bridles']
1217     , name='Bridles-Bottom Flanges', positionToleranceMethod=
             COMPUTED, slave=
1218     mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Deck
             -1'].surfaces['Bottom Flanges']
1219     , thickness=ON, tieRotations=ON)
1220
1221 mdb.models['CFRP Bridge'].Tie(adjust=OFF, master=
1222     mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Deck
             -1'].surfaces['Plate']
1223     , name='Bridles-Plate', positionToleranceMethod=COMPUTED,
             slave=
1224     mdb.models['CFRP Bridge'].rootAssembly.instances['CFRP
             Bridge-1'].surfaces['Bridles']
1225     , thickness=ON, tieRotations=ON)
1226
1227 mdb.models['CFRP Bridge'].PinnedBC(createStepName='Initial',
         localCsys=None,
1228     name='Bridles Supports', region=
1229     mdb.models['CFRP Bridge'].rootAssembly.instances['CFRP
             Bridge-1'].sets['Bridles Supports'])
```

```
1230 mdb.models['CFRP Bridge'].PinnedBC(createStepName='Initial',
        localCsys=None,
1231      name='Deck Supports', region=
1232      mdb.models['CFRP Bridge'].rootAssembly.instances['GFRP Deck
             -1'].sets['Deck Supports'])
1233 mdb.models['CFRP Bridge'].Gravity(comp3=-9.81, createStepName='
        Installation',
1234      distributionType=UNIFORM, field='', name='Gravity')
1235
1236 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].seedEdgeByNumber
        (constraint=
1237      FIXED, edges=
1238      mdb.models['CFRP Bridge'].parts['CFRP Bridge'].
             edges.findAt(((xcoord[381]+0.02,
1239      ycoord[381], zcoord[381]), ), ((xcoord[371]+0.02, ycoord
             [371], zcoord[371]), ), ((
1240      xcoord[361]+0.02, ycoord[361], zcoord[361]), ), ((xcoord
             [351]+0.02, ycoord[351], zcoord[351]), ),
1241      ((xcoord[341]+0.02, ycoord[341], zcoord[341]), ), ((xcoord
             [331]-0.02, ycoord[331], zcoord[331]),
1242      ), ((xcoord[321]-0.02, ycoord[321], zcoord[321]), ), ((
             xcoord[311]-0.02, ycoord[321], zcoord[321]),
1243      ), ((xcoord[301]-0.02, ycoord[301], zcoord[301]), ), ((
             xcoord[291]-0.02, ycoord[291], zcoord[291]),
1244      ), ((xcoord[281]+0.02, ycoord[281], zcoord[281]), ), ((
             xcoord[271]+0.02, ycoord[271], zcoord[271]),
1245      ), ((xcoord[261]+0.02, ycoord[261], zcoord[261]), ), ((
             xcoord[251]+0.02, ycoord[251], zcoord[251]),
1246      ), ((xcoord[241]+0.02, ycoord[241], zcoord[241]), ), ((
             xcoord[231]-0.02, ycoord[231], zcoord[231]),
1247      ), ((xcoord[221]-0.02, ycoord[221], zcoord[221]), ), ((
             xcoord[211]-0.02, ycoord[211], zcoord[211]),
1248      ), ((xcoord[201]-0.02, ycoord[201], zcoord[201]), ), ((
             xcoord[191]-0.02, ycoord[191], zcoord[191]),
```

```
1249     ), ), number=int(mn[0]))
1250 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].seedEdgeByNumber
        (constraint=
1251     FIXED, edges=
1252     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].
            edges.findAt(((xcoord[380]+0.02,
1253     ycoord[380], zcoord[380]), ), ((xcoord[370]+0.02, ycoord
            [370], zcoord[370]), ), ((
1254     xcoord[360]+0.02, ycoord[360], zcoord[360]), ), ((xcoord
            [350]+0.02, ycoord[350], zcoord[350]), ),
1255     ((xcoord[340]+0.02, ycoord[340], zcoord[340]), ), ((xcoord
            [330]-0.02, ycoord[330], zcoord[330]),
1256     ), ((xcoord[320]-0.02, ycoord[320], zcoord[320]), ), ((
            xcoord[310]-0.02, ycoord[320], zcoord[320]),
1257     ), ((xcoord[300]-0.02, ycoord[300], zcoord[300]), ), ((
            xcoord[290]-0.02, ycoord[290], zcoord[290]),
1258     ), ((xcoord[280]+0.02, ycoord[280], zcoord[280]), ), ((
            xcoord[270]+0.02, ycoord[270], zcoord[270]),
1259     ), ((xcoord[260]+0.02, ycoord[260], zcoord[260]), ), ((
            xcoord[250]+0.02, ycoord[250], zcoord[250]),
1260     ), ((xcoord[240]+0.02, ycoord[240], zcoord[240]), ), ((
            xcoord[230]-0.02, ycoord[230], zcoord[230]),
1261     ), ((xcoord[220]-0.02, ycoord[220], zcoord[220]), ), ((
            xcoord[210]-0.02, ycoord[210], zcoord[210]),
1262     ), ((xcoord[200]-0.02, ycoord[200], zcoord[200]), ), ((
            xcoord[190]-0.02, ycoord[190], zcoord[190]),
1263     ), ), number= int(mn[1]))
1264 mdb.models['CFRP Bridge'].parts['CFRP Bridge'].seedEdgeByNumber
        (constraint=
1265     FIXED, edges=
1266     mdb.models['CFRP Bridge'].parts['CFRP Bridge'].
            edges.findAt(((xcoord[382]+0.02,
1267     ycoord[382], zcoord[382]), ), ((xcoord[372]+0.02, ycoord
            [372], zcoord[372]), ), ((
```

```
1268      xcoord[362]+0.02, ycoord[362], zcoord[362]), ), ((xcoord
              [352]+0.02, ycoord[352], zcoord[352]), ),
1269      ((xcoord[342]+0.02, ycoord[342], zcoord[342]), ), ((xcoord
              [332]-0.02, ycoord[332], zcoord[332]),
1270      ), ((xcoord[322]-0.02, ycoord[322], zcoord[322]), ), ((
              xcoord[312]-0.02, ycoord[312], zcoord[312]),
1271      ), ((xcoord[302]-0.02, ycoord[302], zcoord[302]), ), ((
              xcoord[292]-0.02, ycoord[292], zcoord[292]),
1272      ), ((xcoord[282]+0.02, ycoord[282], zcoord[282]), ), ((
              xcoord[272]+0.02, ycoord[272], zcoord[272]),
1273      ), ((xcoord[262]+0.02, ycoord[262], zcoord[262]), ), ((
              xcoord[252]+0.02, ycoord[252], zcoord[252]),
1274      ), ((xcoord[242]+0.02, ycoord[242], zcoord[242]), ), ((
              xcoord[232]-0.02, ycoord[232], zcoord[232]),
1275      ), ((xcoord[222]-0.02, ycoord[222], zcoord[222]), ), ((
              xcoord[212]-0.02, ycoord[212], zcoord[212]),
1276      ), ((xcoord[202]-0.02, ycoord[202], zcoord[202]), ), ((
              xcoord[192]-0.02, ycoord[192], zcoord[192]),
1277      ), ), number=int(mn[2]))
1278  mdb.models['CFRP Bridge'].parts['CFRP Bridge'].generateMesh()
1279
1280  mdb.models['CFRP Bridge'].parts['GFRP Deck'].seedPart(
          deviationFactor=0.1,
1281      minSizeFactor=0.1, size=ms[1])
1282  mdb.models['CFRP Bridge'].parts['GFRP Deck'].generateMesh()
1283
1284  mdb.models['CFRP Bridge'].Pressure(amplitude=UNSET,
          createStepName=
1285      'Traffic Load', distributionType=UNIFORM, field='',
              magnitude=2.0, name=
1286      'Traffic Load', region=Region(
1287      side1Faces=mdb.models['CFRP Bridge'].rootAssembly.instances
              ['GFRP Deck-1'].faces.getSequenceFromMask(
1288      mask=('[#9fffa ]', ), )))
```

```
1289
1290 mdb.Job(atTime=None, contactPrint=OFF, description='',
        echoPrint=OFF,
1291      explicitPrecision=SINGLE, getMemoryFromAnalysis=True,
            historyPrint=OFF,
1292      memory=90, memoryUnits=PERCENTAGE, model='CFRP Bridge',
            modelPrint=OFF,
1293      multiprocessingMode=DEFAULT, name='CFRP_Bridge',
            nodalOutputPrecision=
1294      SINGLE, numCpus=1, queue=None, scratch='', type=ANALYSIS,
            userSubroutine=''
1295      , waitHours=0, waitMinutes=0)
1296 mdb.jobs['CFRP_Bridge'].submit(consistencyChecking=OFF)
1297 # mdb.jobs['CFRP_Bridge'].waitForCompletion()
1298
1299 odb=openOdb(path='CFRP_Bridge.odb')
1300 step=odb.steps['Traffic Load']
1301 step0=odb.steps['Installation']
1302 frame=step.frames[-1]
1303 frame0=step0.frames[-1]
1304 dispField=frame.fieldOutputs['U']
1305 dispField0=frame0.fieldOutputs['U']
1306 rotaField=frame.fieldOutputs['UR']
1307 rotaField0=frame0.fieldOutputs['UR']
1308 stressField=frame.fieldOutputs['S']
1309 nodes=odb.rootAssembly.instances['CFRP BRIDGE-1'].nodeSets['
        POINTS']
1310 central=odb.rootAssembly.instances['CFRP BRIDGE-1'].elementSets
        ['CENTRAL AXIS']
1311 exterior=odb.rootAssembly.instances['CFRP BRIDGE-1']
        .elementSets['EXTERIOR BRIDLES']
1312
1313 interior=odb.rootAssembly.instances['CFRP BRIDGE-1']
        .elementSets['INTERIOR BRIDLES']
```

```
1314 disp_at_nodes=dispField.getSubset(region=nodes)
1315 disp_at_nodes0=dispField0.getSubset(region=nodes)
1316 rota_at_nodes=rotaField.getSubset(region=nodes)
1317 rota_at_nodes0=rotaField0.getSubset(region=nodes)
1318 stress_at_central=stressField.getSubset(region=central)
1319 stress_at_exterior=stressField.getSubset(region=exterior)
1320 stress_at_interior=stressField.getSubset(region=interior)
1321 outFile=open('displacements.txt', 'w')
1322 for i in range(0, 32):
1323         dispNode = disp_at_nodes.values[i].data
1324         outFile.write( '\n' )
1325         for j in range( 3 ):
1326             outFile.write( str( dispNode[j] ) + ' ' )
1327 outFile.close()
1328 outFile=open('rotations.txt', 'w')
1329 for i in range(0, 32):
1330         rotaNode = rota_at_nodes.values[i].data
1331         outFile.write( '\n' )
1332         for j in range( 3 ):
1333             outFile.write( str( rotaNode[j] ) + ' ' )
1334     # write point data
1335 outFile.close()
1336 outFile=open('displacements0.txt', 'w')
1337 for i in range(0, 32):
1338         dispNode = disp_at_nodes0.values[i].data
1339         outFile.write( '\n' )
1340         for j in range( 3 ):
1341             outFile.write( str( dispNode[j] ) + ' ' )
1342 outFile.close()
1343 outFile=open('rotations0.txt', 'w')
1344 for i in range(0, 32):
1345         rotaNode = rota_at_nodes0.values[i].data
1346         outFile.write( '\n' )
1347         for j in range( 3 ):
```

```
1348             outFile.write( str( rotaNode[j] ) + ' ' )
1349     # write point data
1350 outFile.close()
1351 outFile = open( 'stressesc.txt' , 'w' )
1352 for i in stress_at_central.values:
1353       stressNode=i.mises
1354       outFile.write( '\n' )
1355       outFile.write( str( stressNode ) + ' ')
1356 # write point data
1357 outFile.close()
1358 outFile = open( 'stressese.txt' , 'w' )
1359 for i in stress_at_exterior.values:
1360       stressNode=i.mises
1361       outFile.write( '\n' )
1362       outFile.write( str( stressNode ) + ' ')
1363 # write point data
1364 outFile.close()
1365 outFile = open( 'stressesi.txt' , 'w' )
1366 for i in stress_at_interior.values:
1367       stressNode=i.mises
1368       outFile.write( '\n' )
1369       outFile.write( str( stressNode ) + ' ')
1370 # write point data
1371 outFile.close()
1372 odb.close()
```

Appendix B

Surrogate Model Adjustment Codes

```
%Created by Julio Rodríguez Sáchez

clear; clc;

d11=dlmread('d11.txt');
d1=d11(1:690,:);
d22=dlmread('d22.txt');
d2=d22(1:690,:);
parametersm=dlmread('parameters1.txt');
variablesv=[2 3 4 5 7 11 12 13];
min_range=[ 0.5 0.2 0.3  1    2    1.2  0.60 3  3.8  1.5  0.02
     0.02  0.02];
max_range=[ 0.5 0.4 0.6  1.5  2.5  1.2  1.75 3  3.8  1.5  0.10
     0.10  0.10];
parametersm(1:10,:)=parametersm(1:10,:)*10/66;
for i=1:size(variablesv,2)
    variablesm(i,:)=log(parametersm(variablesv(i),:)/min_range
        (1,variablesv(i)))/log(max_range(1,variablesv(i))/
```

```
        min_range(1,variablesv(i)));
end
freqm1=dlmread('modes.txt');
freqm=freqm1(:,1:8)/100;


for i=1:size(d1,2)
    yc{i,1}=d1(:,i)-d2(:,i);
end
fprintf(' Data Read  ======> ');
for i=1:size(d1,2)
    for j=1:size(freqm,2)
        for k=1:size(variablesm,1)
            Ac{i,k+size(variablesm,1)*(j-1)}=variablesm(k,i)*
                freqm(:,j);
        end
    end
end
fprintf(' A Matrix Created  ======> ');

A=cell2mat(Ac);
y=cell2mat(yc);
coeff=(A'*A)\(A'*y);
res=y-A*coeff;

for i=1:size(d1,2)
    r(:,i)=res(1+size(d1,1)*(i-1):size(d1,1)*i,1);
    yd(:,i)=y(1+size(d1,1)*(i-1):size(d1,1)*i,1);
end

for i=1:size(d1,1)
    rmax(i,1)=max(r(i,:));
    rmean(i,1)=mean(r(i,:));
    ydmean(i,1)=mean(yd(i,:));
```

```
48      yfemean(i,1)=mean(d1(i,:));
49      rstdv(i,1)=std(r(i,:));
50  end
51  resnorm=norm(rmean);
52  ydnorm=norm(ydmean);
53  yfenorm=norm(yfemean);
54
55  fprintf(' Residues Computed  ======> ');
56
57  save -ascii 'ucoefficients.txt' coeff;
58
59
60  % for i=1:size(an,1)
61  %       for j=1:size(At,2)
62  %       sum=0;
63  %           for k=1:size(At,1)
64  %           sum=sum+an{i,k}\At{k,j};
65  %           end
66  %       h2{i,j}=sum;
67  %       end
68  % end
69  %
70  % fprintf(' h2 Matrix Done  ======> ');
71  %
72  % for i=1:size(A,1)
73  %       for j=1:size(h2,2)
74  %       sum=0;
75  %           for k=1:size(h2,1)
76  %           sum=sum+A{i,k}.*h2{k,j};
77  %           end
78  %       H{i,j}=sum;
79  %       end
80  % end
81  %
```

```
% fprintf(' H Matrix Done  ======> ');
%
% for i=1:size(A,1)
%       for j=1:size(h2,2)
%          H1{i,j}=H{i,j}-ones(size(d1,1),1);
%       end
% end
%
% for i=1:size(H1,1)
%       for j=1:size(y,2)
%       sum=0;
%           for k=1:size(y,1)
%           sum=sum+H1{i,k}.*y{k,j};
%           end
%       r{i,j}=sum;
%       end
% end
```

```
%Created by Julio Rodríguez Sáchez

clear; clc;

s1=dlmread('s1.txt');
s2=dlmread('s2.txt');
parametersm=dlmread('parameters1.txt');
variablesv=[2 3 4 5 7 11 12 13];
min_range=[ 0.5 0.2 0.3  1    1.8  1.2  0.60 3  3.8  1.5  0.02
    0.02  0.02];
max_range=[ 0.5 0.4 0.6  1.5  2.5  1.2  1.75 3  3.8  1.5  0.10
    0.10  0.10];
parametersm(1:10,:)=parametersm(1:10,:)*10/66;
for i=1:size(variablesv,2)
```

```
13      variablesm(i,:)=log(parametersm(variablesv(i),:)/min_range
            (1,variablesv(i)))/log(max_range(1,variablesv(i))/
            min_range(1,variablesv(i)));
14  end
15  for i=1:size(s1,2)
16      for j=1:3
17          if s2(j,i)>8e05
18              s1(:,i)=s1(:,i)*0;
19              s2(:,i)=s2(:,i)*0;
20              parametersm(:,i)=parametersm(:,i)*0;
21          end
22      end
23  end
24  n=-1; s11=s1; s22=s2; parametersm1=parametersm;
25  for i=1:size(s1,2)
26      if mean(s2(:,i))==0
27          n=n+1;s11(:,i-n)=[]; s22(:,i-n)=[]; parametersm1(:,i-n)
                =[];
28      end
29  end
30  s1=s11;s2=s22;parametersm=parametersm1;
31  for i=1:size(s1,2)
32      yc{1,i}=s1(:,i)-s2(:,i);
33  end
34  fprintf(' Data Read  ======> ');
35  id=[1 0 0; 0 1 0; 0 0 1];
36  for i=1:size(s1,2)
37      Ac1(1+3*(i-1):3+3*(i-1),1:3)=id*variablesm(1,i);
38      Ac2(1+3*(i-1):3+3*(i-1),1:3)=id*variablesm(2,i);
39      Ac3(1+3*(i-1):3+3*(i-1),1:3)=id*variablesm(3,i);
40      Ac4(1+3*(i-1):3+3*(i-1),1:3)=id*variablesm(4,i);
41      Ac5(1+3*(i-1):3+3*(i-1),1:3)=id*variablesm(5,i);
42      Ac6(1+3*(i-1):3+3*(i-1),1:3)=id*variablesm(6,i);
43      Ac7(1+3*(i-1):3+3*(i-1),1:3)=id*variablesm(7,i);
```

```
44      Ac8(1+3*(i-1):3+3*(i-1),1:3)=id*variablesm(8,i);
45  end
46  A(:,1:3)=Ac1;A(:,4:6)=Ac2;A(:,7:9)=Ac3;A(:,10:12)=Ac4;A
        (:,13:15)=Ac5;A(:,16:18)=Ac6;A(:,19:21)=Ac7;A(:,22:24)=Ac8;
47
48
49  fprintf(' A Matrix Created  ======> ');
50
51  y=cell2mat(yc');
52  coeff=(A'*A)\(A'*y);
53  res=y-A*coeff;
54
55  for i=1:size(s1,2)
56      r(:,i)=res(1+size(s1,1)*(i-1):size(s1,1)*i,1);
57      yd(:,i)=y(1+size(s1,1)*(i-1):size(s1,1)*i,1);
58  end
59
60  for i=1:size(s1,1)
61      rmax(i,1)=max(r(i,:));
62      rmean(i,1)=mean(r(i,:));
63      ydmean(i,1)=mean(yd(i,:));
64      yfemean(i,1)=mean(s1(i,:));
65      rstdv(i,1)=std(r(i,:));
66  end
67  resnorm=norm(rmean);
68  ydnorm=norm(ydmean);
69  yfenorm=norm(yfemean);
70
71  fprintf(' Residues Computed  ======> ');
72
73  save -ascii 'scoefficients.txt' coeff;
74
75
76  % for i=1:size(an,1)
```

```
77 %       for j=1:size(At,2)
78 %       sum=0;
79 %           for k=1:size(At,1)
80 %           sum=sum+an{i,k}\At{k,j};
81 %           end
82 %       h2{i,j}=sum;
83 %       end
84 % end
85 %
86 % fprintf(' h2 Matrix Done  ======> ');
87 %
88 % for i=1:size(A,1)
89 %       for j=1:size(h2,2)
90 %       sum=0;
91 %           for k=1:size(h2,1)
92 %           sum=sum+A{i,k}.*h2{k,j};
93 %           end
94 %       H{i,j}=sum;
95 %       end
96 % end
97 %
98 % fprintf(' H Matrix Done  ======> ');
99 %
100 % for i=1:size(A,1)
101 %       for j=1:size(h2,2)
102 %          H1{i,j}=H{i,j}-ones(size(d1,1),1);
103 %       end
104 % end
105 %
106 % for i=1:size(H1,1)
107 %       for j=1:size(y,2)
108 %       sum=0;
109 %           for k=1:size(y,1)
110 %           sum=sum+H1{i,k}.*y{k,j};
```

```
111 %             end
112 %         r{i,j}=sum;
113 %         end
114 % end
```

Appendix C

Parametric Lifetime MA Model of the Bridge Codes

```
%Created by Julio Rodríguez Sáchez

tic
clc, clear
time=tic;
warning off

%% Number of Samples
n_cycles=5;
esamples=2;

%rnd_range=[ m1  m2  m3  m4    m5   m6   m7   m8  l1  l2    b1
      b2    b3]
min_range=[ 0.5 0.2 0.3  1   2    1.2  0.60 3  3.8  1.5  0.02
    0.02  0.02];
max_range=[ 0.5 0.4 0.6  1.5 2.5  1.2  1.75 3  3.8  1.5  0.10
    0.10  0.10];

```

```
16  min_range(1,1:10)=min_range(1,1:10)*66/10;
17  max_range(1,1:10)=max_range(1,1:10)*66/10;
18
19  acc_y=1;
20  acc_n=4;
21  us_y=1;
22  us_n=12;
23  noise=1;
24  cycle_cost=0;
25
26 timel=tic;
27
28 %% Variable Range Definition
29  lamcal_gfrp;
30  lamcal_cfrp;
31  geomcal;
32  parameters(:,1)=[m1 m2 m3 m4 m5 m6 m7 m8 l1 l2 b1 b2 b3 b4 b5
        b6 ...
33       h1 h2 h3 h4 h5 YM1 YM2 YM3 YM4 YM5]';
34  parametersm(:,1)=parameters(:,1);
35  parametersm(1:10,1)=parametersm(1:10,1)*10/66;
36
37 %% Section Variables
38 % [Central Axis; Exterior Bridles; Interior Bridles]
39 section=[b1; b2; b3; b4; b5; b6; b7];
40 depth=[h1; h2; h3; h4; h5];
41
42 for cycle=1:n_cycles
43     stop_crck=0;
44     for ymdist=1:esamples
45     %% Material Properties
46     if cycle==1
47         mprop=[YM1(ymdist) SM1(ymdist); YM2(ymdist) SM2(ymdist)
              ; ...
```

```
48              YM3(ymdist) SM3(ymdist); YM4 SM4; YM5 SM5];
49      end
50          parameters(22:24,1)=[mprop(1,1) mprop(2,1) mprop(3,1)
                ]';
51          parametersm(:,1)=parameters(:,1);
52          parametersm(1:10,1)=parametersm(1:10,1)*10/66;
53
54      %% Matrix Analysis Model
55          timema=tic;
56          ma;
57          fprintf(' MBM Analysis Completed  ======> ');
58          toc(timema)
59          maxdisp(ymdist,cycle)=d2(657,1);
60          maxstrs(ymdist,cycle)=s2max(1,1);
61          ymev(1:3,cycle)=mprop(1:3,1);
62          useful_life(ymdist)=cycle;
63          fprintf(' Cycle ======> %d , Sample ======> %d\n',
                cycle, ymdist);
64      end
65      stiffness_evolution;
66      if stop_crck==1
67          useful_life(ymdist)=cycle; break
68      end
69      %%   Probability of Failure
70      prob_failure;
71      %%   Cost Function
72      c_cost;
73      %% End
74      fprintf(' Cycles Completed  ======> %d\n', cycle);
75      fprintf(' Time  ======> %d\n', toc(time));
76  end
77  mass=(cfrp_volume*1700+gfrp_volume*2000);
78  material_cost=mass*70;construction_cost=1*material_cost;
79  initial_cost=(material_cost+construction_cost)/0.8+10000+20000;
```

```
lc_cost=(initial_cost+sum(cycle_cost))/min(useful_life);
toc
```

```
%Created by Julio Rodríguez Sáchez

%% Laminate Calculator

%%
 % First it is needed to determine the plies that makes the
    laminate.
 % Here we will consider just 4 families of plies (plies with
    different
 % fiber orientations)
 % Set the characteristics of every ply in the command lines
    below
 % Units are in m, GPa, degrees
 % Call the calculator this way: "sol=lamcalsym([m n p q r+ r-
    h b e d])
 % where m,n,p and q are the fiber orientations of plies 1,2,3
    and 4,
 % in the workspace directions (m,n p and q are written in
    degrees),r+ and
 % r- are the stacking sequence index above and below the
    midplane,
 % respectively (r+ and r- must be a positive number or zero),
    h is the total
 % thickness of the laminate, b is the with of the section and
    e is the
 % thickness of the flanges the section might have
 % Variable d is stated for decision purposes between 3-ply
    laminates or
 % other else. If a 3-ply laminate has to be calculated d has
    to be equal
```

```
20  % to 1. For other cases (homogeneous, 2-ply or 4-ply laminates
        ) d has to
21  % be not equal to one
22  % For calculation purposes this code works with normalized
        axial forces
23  % and bending moments, wich makes everything easier to compute
         (stacking
24  % sequence-flexural stress interaction can be messy, so we
        will just work
25  % with normalized stresses and thus we can keep an easier
        calculation
26  % going on)
27
28
29  %%%PLY DEFINITION (STANFORD)
30
31  Ex=41.7e9;                    %[Pa] On-axis (in-plane) (Young's
        modulus)
32  Ey=13e9;                   %[Pa] On-axis (in-plane) (Transverse
        Young's modulus
33  Es=3.4e9;                   %[Pa] Shear modulus (in-plane)
34  nux=0.300;                   %[no units] in plane Poisson's
        ratio
35  nuy=nux*(Ey/Ex);             %[no units] in- plane plane Poisson
        's ratio
36  nuyz=0.42;                   %[no units] out of plane Poisson's
        ratio
37  Gyz=Ey/(2*(1+nuyz));         %[Pa] out of plane shear modulus
38  plythick=0.203e-3;          %[m] Ply Thickness
39  G_d0=9.07e13;                %[Pa/m] this is a purely
        experimental parameter. See Ogihara1995, Table 3.
40  G_d0=2.07e11;                %[Pa/m] this is one trial by Manuel
41  G_c=150.609;                 %[J/m^-2] intralaminar critical
        energy release rate  150.609
```

```
42
43
44 %%%LAMINATE DEFINITION (GUDMUNDSON)
45 lam_type='x-ply';
46 StackSeq=[0 45 -45 90 90 -45 45 0];          %total laminate.
       From top to bottom
47 Sub_s_StackSeq=[0];                     %sublaminate 1 (Remember:
        only one of the 0° Ply stack)
48 Sub_90_StackSeq=[45 -45 90 90 -45 45];          %sublaminate 2
49 Laminate.StackSeq=StackSeq;
50 h=numel(Laminate.StackSeq)*plythick/2;        %[m] Laminate half
       -thickness
51 B=0.001;
52
53 path(path,'lib')
54 stiffness;
55
56 b4=0.05;h4=size(StackSeq,2)*plythick;YM4=E_0*10^(-3);SM4=Gyz
       *10^(-6);
57
58 %%%LAMINATE DEFINITION (GUDMUNDSON)
59 lam_type='x-ply';
60 StackSeq=[0 0 45 45 -45 -45 90 90 90 90 -45 -45 45 45 0 0];
              %total laminate. From top to bottom
61 Sub_s_StackSeq=[0];                     %sublaminate 1 (Remember:
        only one of the 0° Ply stack)
62 Sub_90_StackSeq=[45 -45 90 90 -45 45];          %sublaminate 2
63 Laminate.StackSeq=StackSeq;
64 h=numel(Laminate.StackSeq)*plythick/2;        %[m] Laminate half
       -thickness
65 B=0.001;
66
67 path(path,'lib')
68 stiffness;
```

```
69
70 b5=2;h5=size(StackSeq,2)*plythick;YM5=E_0*10^(-3);SM5=Gyz
       *10^(-6);
71 b6=2;
72 b7=0.05;
```

```
1  %Created by Julio Rodríguez Sáchez
2
3  %% Laminate Calculator
4
5  %%
6   % First it is needed to determine the plies that makes the
       laminate.
7   % Here we will consider just 4 families of plies (plies with
       different
8   % fiber orientations)
9   % Set the characteristics of every ply in the command lines
       below
10  % Units are in m, GPa, degrees
11  % Call the calculator this way: "sol=lamcalsym([m n p q r+ r-
       h b e d])
12  % where m,n,p and q are the fiber orientations of plies 1,2,3
       and 4,
13  % in the workspace directions (m,n p and q are written in
       degrees),r+ and
14  % r- are the stacking sequence index above and below the
       midplane,
15  % respectively (r+ and r- must be a positive number or zero),
       h is the total
16  % thickness of the laminate, b is the with of the section and
       e is the
17  % thickness of the flanges the section might have
```

```
18 % Variable d is stated for decision purposes between 3-ply laminates or
19 % other else. If a 3-ply laminate has to be calculated d has to be equal
20 % to 1. For other cases (homogeneous, 2-ply or 4-ply laminates) d has to
21 % be not equal to one
22 % For calculation purposes this code works with normalized axial forces
23 % and bending moments, wich makes everything easier to compute (stacking
24 % sequence-flexural stress interaction can be messy, so we will just work
25 % with normalized stresses and thus we can keep an easier calculation
26 % going on)
27
28
29 for k=1:esamples
30
31 %%%PLY DEFINITION (STANFORD)
32
33    Ex=normrnd(127.553e9,127.553e8);                %[Pa] On-axis (in-plane) (Young's modulus)
34    Ey=normrnd(8.411e9,8.411e8);                    %[Pa] On-axis (in-plane) (Transverse Young's modulus
35    Es=normrnd(6.205e9,6.205e8);                    %[Pa] Shear modulus (in-plane
36    nux=normrnd(0.309,0.0309);                      %[no units] in plane Poisson's ratio
37    nuy=nux*(Ey/Ex);              %[no units] in- plane plane Poisson's ratio
38    nuyz=0.49;                    %[no units] out of plane Poisson's ratio
```

```
39      Gyz=Ey/(2*(1+nuyz));          %[Pa] out of plane shear
            modulus
40      plythick=0.1524e-3;           %[m] Ply Thickness
41      G_d0=9.07e13;                 %[Pa/m] this is a purely
            experimental parameter. See Ogihara1995, Table 3.
42      G_d0=2.07e11;                 %[Pa/m] this is one trial by
            Manuel
43      G_c=150.609;                  %[J/m^-2] intralaminar critical
             energy release rate  150.609
44
45      %%%LAMINATE DEFINITION (GUDMUNDSON)
46      lam_type='x-ply';
47      StackSeq=[0 0 90 90 90 90 90 90 90 90 0 0];  %total
            laminate. From top to bottom
48      Sub_s_StackSeq=[0 0];             %sublaminate 1 (Remember:
            only one of the 0º Ply stack)
49      Sub_90_StackSeq=[90 90 90 90 90 90 90 90];   %sublaminate 2
50      Laminate.StackSeq=StackSeq;
51      h=numel(Laminate.StackSeq)*plythick/2;       %[m] Laminate
            half-thickness
52      B=0.07;                                %[m] Laminate half-
            width (if variable section, choose the critical one)
53
54      cd('lib')
55      stiffness;
56
57      cd('../')
58
59      YM1(k)=E_0*10^(-3);YM2(k)=E_0*10^(-3);YM3(k)=E_0*10^(-3);
60      SM1(k)=Gyz*10^(-3);SM2(k)=Gyz*10^(-3);SM3(k)=Gyz*10^(-3);
61
62  end
63
64      b1=min_range(1,11)+(max_range(1,11)-min_range(1,11))*rand;
```

```
65      b2=min_range(1,12)+(max_range(1,12)-min_range(1,12))*rand;
66      b3=min_range(1,13)+(max_range(1,13)-min_range(1,13))*rand;
67      h1=plythick*size(StackSeq,2);h2=plythick*size(StackSeq,2);
            h3=plythick*size(StackSeq,2);
```

```
1  function [a_art,eng_cnst]=LamTheory(PlyData,StackSeq)
2
3  %This function calculates the normalized A and a stiffness
       matrices of a
4  %symmetric or unsymmetric laminate. Note that if unsymmetric
       laminate, the
5  %stiffness and engineering constants are valid for in-plane
       loads only,
6  %wich is the typical case in fatigue.
7
8
9  % %prueba
10 % clear all
11 %  StackSeq=[90 90 90 90]; %total laminate. From top to bottom
12 %  nplies=numel(StackSeq);  %Total number of plies
13 % families=[0,90]; %N° of fiber angle orientation
14 % norientfamil=numel(families); %N° of orientation angles
15 % %
16 % Ex=127.553e9;               %[Pa] On-axis (in-plane) (Young's
       modulus)
17 % Ey=8.411e9;                 %[Pa] On-axis (in-plane) (
       Transverse Young's modulus
18 % Es=6.205e9;                 %[Pa] Shear modulus (in-plane)
19 % nux=0.309;                  %[no units] in plane Poisson's
       ratio
20 % nuy=nux*(Ey/Ex);            %[no units] in- plane plane
       Poisson's ratio
```

```
% nuyz=0.49;                    %[no units] out of plane Poisson's ratio
% Gyz=Ey/(2*(1+nuyz));          %[Pa] out of plane shear modulus
% plythick=0.135e-3;            %[m] Ply Thickness
% G_d0=9.07e13;                 %[Pa/m] this a purely experimental parameter. See Ogihara1995, Table 3.
%
% PlyData=[Ex,Ey,Es,nux,nuy,nuyz,Gyz,plythick];    % characterizing vector


Ex=PlyData(1);
Ey=PlyData(2);
Es=PlyData(3);
nux=PlyData(4);
nuy=PlyData(5);
plythick=PlyData(8);

nplies=numel(StackSeq);

Qxx=Ex/(1-nux*nuy);
Qyy=Ey/(1-nux*nuy);
Qxy=nuy*Qxx;
Qyx=nux*Qyy;
Qss=Es;

Qonx=[Qxx, Qxy, 0;...
   Qyx Qyy 0;...
   0 0 Qss];
```

```
52 %Acc_Soffx={};          %Initialize a storage matrix with Soffx
       for each ply
53 %aux_effstiff={};        %Initialize an auxiliary storage
      matrix
54 %A={};                   %Initialize  "A" storage matrix
55 %acc_effstiff=zeros(3,3);       %Initializa a auxiliary storage
       matrix
56 %betamatrix={};
57 Acc_Qoffx=zeros(3,3);
58
59 for i=1:nplies
60     theta=StackSeq(i)*(pi/180);     %fiber angle: degree*(degree
          ->rad)= [rad]
61     m=cos(theta);
62     n=sin(theta);
63
64
65               tranfmatrix=[m^4 n^4 2*(m^2)*(n^2) 4*(m^2)*(n^2);
                     ...  %Transformation matrix
66                    n^4 m^4 2*(m^2)*(n^2) 4*(m^2)*(n^2);...
67                    (m^2)*(n^2) (m^2)*(n^2) (m^4)+(n^4) -4*(m^2)*(
                         n^2);...
68                    (m^2)*(n^2) (m^2)*(n^2) -2*(m^2)*(n^2) ((m^2)
                         -(n^2))^2;...
69                    (m^3)*n -m*(n^3) m*(n^3)-(m^3)*n 2*(m*(n^3)-(m
                         ^3)*n);...
70                    m*(n^3) -(m^3)*n (m^3)*n-m*(n^3) 2*((m^3)*n-m
                         *(n^3))];
71
72               Qoffx=tranfmatrix*[Qonx(1,1);Qonx(2,2);Qonx(2,1);
                     Qonx(3,3)];
73
74
75
```

```
76      %Soffx=tranfmatrix*[Sonx(1,1) Sonx(2,2) Sonx(2,1) Sonx(3,3)
            ]';
77      %Soffx=[Soffx(1) Soffx(3) Soffx(5);...
78          %Soffx(3) Soffx(2) Soffx(6);...
79          %Soffx(5) Soffx(6) Soffx(4)];
80
81          Qoffx=[Qoffx(1) Qoffx(3) Qoffx(5);...
82          Qoffx(3) Qoffx(2) Qoffx(6);...
83          Qoffx(5) Qoffx(6) Qoffx(4)];
84
85      Acc_Qoffx=Acc_Qoffx+Qoffx;
86      %Acc_Qoffx{i,1}=Qoffx;
87  end
88
89  %normalized laminate stiffness matrix
90  A_norm=Acc_Qoffx./nplies;
91
92  %a_norm=inv(A_norm);
93
94
95  dtA=det(A_norm);
96  %normalized laminate compliance matrix
97  a_art.a_11=(A_norm(2,2)*A_norm(3,3)-(A_norm(2,3))^2)/dtA;
98  a_art.a_22=(A_norm(1,1)*A_norm(3,3)-(A_norm(1,3))^2)/dtA;
99  a_art.a_66=(A_norm(1,1)*A_norm(2,2)-(A_norm(1,2))^2)/dtA;
100 a_art.a_12=(-A_norm(1,2)*A_norm(3,3)+A_norm(1,3)*A_norm(2,3))/
        dtA;
101 a_art.a_16=(A_norm(1,2)*A_norm(2,3)-A_norm(2,2)*A_norm(1,3))/
        dtA;
102 a_art.a_26=(A_norm(1,2)*A_norm(1,3)-A_norm(1,1)*A_norm(2,3))/
        dtA;
103
104 %a_art=[a_11,a_12,a_16;a_12,a_22,a_26;a_16,a_26,a_66];
105
```

```
106 %Laminate engineering constants
107
108 eng_cnst.E_1=1/a_art.a_11;
109 eng_cnst.E_2=1/a_art.a_22;
110 eng_cnst.E_6=1/a_art.a_66;
111 eng_cnst.G=eng_cnst.E_6;
112 eng_cnst.nu_12=-a_art.a_12/a_art.a_22;
113 eng_cnst.nu_21=-a_art.a_12/a_art.a_11;
114 eng_cnst.nu_61=a_art.a_16/a_art.a_11;
115 eng_cnst.nu_16=a_art.a_16/a_art.a_66;
116 eng_cnst.nu_62=a_art.a_26/a_art.a_22;
117 eng_cnst.nu_26=a_art.a_26/a_art.a_66;
118
119 %eng_cnst=[E_1,E_2,E_6,G,nu_12,nu_21,nu_61,nu_16,nu_62,nu_26];
120
121
122 %end
```

```
1 %Created by Julio Rodríguez Sáchez
2
3 m1=min_range(1,1)+(max_range(1,1)-min_range(1,1));          %
      deck thickness
4 m2=min_range(1,2)+(max_range(1,2)-min_range(1,2))*rand;     %
      distance between directrix line and outline joints (type 2.4
      )
5 m3=min_range(1,3)+(max_range(1,3)-min_range(1,3))*rand;     %
      distance between center and first nearest joint (type 1.6)
6 m4=min_range(1,4)+(max_range(1,4)-min_range(1,4))*rand;     %
      distance between center and second joint (type 1.4)
7 m5=min_range(1,5)+(max_range(1,5)-min_range(1,5))*rand;     %
      distance between center and third joint (type 1.3)
8 m6=min_range(1,6)+(max_range(1,6)-min_range(1,6));          %
      distance between extreme line of deck and pylons
```

```
m7=min_range(1,7)+(max_range(1,7)-min_range(1,7))*rand;      %
    pylons height
m8=min_range(1,8)+(max_range(1,8)-min_range(1,8));          %
    distance between center and fourth joint (type 1.4)
l1=min_range(1,9)+(max_range(1,9)-min_range(1,9));          %
    middle of total lenght of deck
l2=min_range(1,10)+(max_range(1,10)-min_range(1,10));       %
    middle of total widht of deck
```

```
% Matrix analysis
% Guillermo Rus Carlborg        2005-07-03
% Alejandro Rodríguez Sánchez   2012-07
% Julio Rodríguez Sánchez       2012-12


 %%
 %% Geometric Linear Calculation (START)

 ma_linear;

  % Rearranging displacements
 for i=1:115
    for j=1:6
        d2(6*(i-1)+j,1)=displacements(i,j);
    end
 end
   % Rearranging stresses
 s2=stress(1:115,1);

delete('coord.mat')
```

```
%Created by Julio Rodríguez Sáchez
```

```
2
3  %% Geometric Linear Calculation (START)
4
5
6  % Initialization of Calculation (START)
7  % Certain variables will be set up in this part of the code
8  %% Searching for the scripts needed in the calculation
9   path(path,'lib');path(path,'dat');
10 % Model for calculation
11  pm_laminae;
12
13  name='modelmsa';
14   disp(['MA ' name]); eval(name); % read
15  nn=size(coord,1); nb=size(conect,1); nd=size(coac,1);
16
17 % Initialization of stiffness matrices (natural stiffness
      matrix k,
18 % geometrical stiffness matrix kg and higher-order stiffness
      matrix kho)
19  k=zeros(nn*size(coac,2),nn*size(coac,2));
20  kg=zeros(nn*size(coac,2),nn*size(coac,2));
21  kho=zeros(nn*size(coac,2),nn*size(coac,2));
22  km=zeros(nn*size(coac,2),nn*size(coac,2));
23 % Initialization of forces vectors f, f0, fbl and fnodes
24  f=zeros(nn*size(coac,2),1);
25  f0=zeros(nn*size(coac,2),1);
26  fbl=zeros(nb*size(coac,2),2);
27
28 % Initialization of displacements vector u
29  u=zeros(nn*size(coac,2),1);
30 % Initialization of Calculation (OVER)
31 %  for b=1:nn
32 %      f(size(coac,2)*b-(size(coac,2)-1):size(coac,2)*b,1)=fn(b
      ,1:size(coac,2));
```

```
%  end
% Calculation Core (START)
 linear_calculation;

 %% Deformation Energy
 fblgen;
%  energy;
 stress_cal;
 structural_volume;

%% Last Deformation
% Deforming previous state
%  nn=size(coord,1);
%  displacements=zeros(nn,m);
%  for b=1:size(displacements,1)
%     displacements(b,1:6)=u(b*m-(m-1):b*m,1);
%  end
%  pm_laminaegnl;
%  load coord
%% Calculation Core (OVER)
%% Display of results

% Setting up the display of displacements vector udisplay
%  m=size(coac,2);
%  coord_plot=coord-coord_plot;
%  udisplay=zeros(m,size(f,1)/m);
%  for icount=1:size(f,1)/m
%      udisplay(1:m,icount)=coord_plot(icount,1:m);
%  end
%  fdisplay=zeros(m,size(f,1)/m);
%
%  for icount=1:size(f,1)/m
%      fdisplay(1:m,icount)=f(m*icount-(m-1):m*icount,1);
%  end
```

```
67
68 % Displaying total stiffness matrix [Kt], displacements vector
       {u} and
69 % nodal forces vector {f}
70 %  disp(' ');
71 % disp('Stiffness Matrix'); disp(' '); disp(kt);
72 % disp('Flexibility Matrix'); disp(' '); disp(kt^-1);
73 % disp('Displacements'); disp(' '); disp(udisplay);
74 % disp('Forces (reac)'); disp(' '); disp(fdisplay');
75 %   disp('Maximum displacement'); disp(' ');disp(min((
       displacements(:,3))));
76 % Setting up the display of forces in bars vector
77 % Displaying forces in bars vector
78 %  disp('Stress'); disp(fbldisplay);
79 %% Conversion for plot
80 %  nn=size(coord,1);
81 %  up=zeros(size(coac,2),nn);
82 %  for b=1:nn
83 %      up(1:size(coac,2),b)=u(size(coac,2)*b-(size(coac,2)-1):
       size(coac,2)*b,1);
84 %  end
85 %  u=up;
86 %
87 % % Calling plotting program
88 %  coord_plot=zeros(size(coord,1),3);
89 %  coord_plot(1:size(coord,1),1:3)=coord(1:size(coord,1),1:3);
90 %  coord=coord_plot;
91 %  ma_plot;
```

```
1 %Created by Julio Rodríguez Sáchez
2
3 % Laminae definition
4 % This is the main body of bridge frame model
```

```
% Every variable regarding space form of the bridge will be defined and
% declared here

%
```

```
% These are the variables that control the parametric form of the bridge

 m1=parametersm(1,1)*66/10;
 m2=parametersm(2,1)*66/10;
 m3=parametersm(3,1)*66/10;
 m4=parametersm(4,1)*66/10;
 m5=parametersm(5,1)*66/10;
 m6=parametersm(6,1)*66/10;
 m7=parametersm(7,1)*66/10;
 m8=parametersm(8,1)*66/10;
 l1=parametersm(9,1)*66/10;
 l2=parametersm(10,1)*66/10;
%
```

```

% Model True Construction I (Right Nodes Position)
m3_probe=0; m4_probe=1; m5_probe=2; m8_probe=3;

while (m4<m3 || m5<m4 || m8<m5)
if m4<m3
    m3_probe=m3; m3=m4; m4=m3_probe;
end
if m5<m4
    m4_probe=m4; m4=m5; m5=m4_probe;
end
```

```
33 if m8<m5
34     m5_probe=m5; m5=m8; m8=m5_probe;
35 end
36 end
37
38 % Model True Construction II (Tune Node Position)
39 if m3==m4
40     m4=m4+0.01;
41 end
42 if m4==m5
43     m5=m5+0.01;
44 end
45 if m5==m8
46     m8=m8+0.01;
47 end
48
49 % Definition of nodes
50 % Planar position of nodes (x & y axis) are defined in the
       following matrix
51 % "l_coor"
52
53 a1=l1+m6-m8;
54 a2=l1+m6-m5;
55 a3=l1+m6-m4;
56 a4=l1+m6-m3;
57 a5=l1+m6;
58 b1=l2+l2*m7/m1/2;
59 b2=l2+l2*m7/m1/2-m2;
60 t1=atan(b1/a1);
61 t2=atan(b1/a2);
62 t3=atan(b1/a3);
63 t4=atan(b1/a4);
64 t5=atan(b2/a5);
65 t6=atan(m2/m3);
```

```
66 t7=atan(b1/a5);
67 st=tan(t5)+tan(t6);
68 ts=tan(t4)+tan(t6);
69
70 % xcoordinate ycoordinate zcoordinate %#node
71 l_coor=[  l1+m6           l2+m7*l2/m1/2    0 0 0 0 ... %   1
       % Support #1
72         ; (m8+a1/b1*10)    10               0 0 0 0 ... %   2
              % Bridle #1
73         ; (m8+a1/b1*6)      6               0 0 0 0 ... %   3
              % Bridle #1
74         ; (m8+a1/b1*4)      4               0 0 0 0 ... %   4
              % Bridle #1
75         ; (m8+a1/b1*2)      2               0 0 0 0 ... %   5
              % Bridle #1
76         ; (m8+a1/b1*1)      1               0 0 0 0 ... %   6
              % Bridle #1
77         ; (m5+a2/b1*10)    10               0 0 0 0 ... %   7
              % Bridle #2
78         ; (m5+a2/b1*6)      6               0 0 0 0 ... %   8
              % Bridle #2
79         ; (m5+a2/b1*4)      4               0 0 0 0 ... %   9
              % Bridle #2
80         ; (m5+a2/b1*2)      2               0 0 0 0 ... %  10
              % Bridle #2
81         ; (m5+a2/b1*1)      1               0 0 0 0 ... %  11
              % Bridle #2
82         ; (m4+a3/b1*10)    10               0 0 0 0 ... %  12
              % Bridle #3
83         ; (m4+a3/b1*6)      6               0 0 0 0 ... %  13
              % Bridle #3
84         ; (m4+a3/b1*4)      4               0 0 0 0 ... %  14
              % Bridle #3
```

```
85        ; (m4+a3/b1*2)      2               0 0 0 0 ... %  15
              % Bridle #3
86        ; (m4+a3/b1*1)      1               0 0 0 0 ... %  16
              % Bridle #3
87        ; (m3+a4/b1*10)    10               0 0 0 0 ... %  17
              % Bridle #4
88        ; (m3+a4/b1*6)      6               0 0 0 0 ... %  18
              % Bridle #4
89        ; (m3+a4/b1*4)      4               0 0 0 0 ... %  19
              % Bridle #4
90        ; (m3+a4/b1*2)      2               0 0 0 0 ... %  20
              % Bridle #4
91        ; (m3+a4/b1*1)      1               0 0 0 0 ... %  21
              % Bridle #4
92        ; (a5/b2*(10-m2))  10               0 0 0 0 ... %  22
              % Bridle #5
93        ; (a5/b2*(6-m2))    6               0 0 0 0 ... %  23
              % Bridle #5
94        ; (a5/b2*3)        m2+3             0 0 0 0 ... %  24
              % Bridle #5
95        ; (a5/b2*2)        m2+2             0 0 0 0 ... %  25
              % Bridle #5
96        ; (a5/b2*1)        m2+1             0 0 0 0 ... %  26
              % Bridle #5
97        ];
98
99    for i=1:26
100       l_coor(i+26,1)=l_coor(i,1);
101       l_coor(i+26,2)=(-1)*l_coor(i,2);
102       l_coor(i+26*2,1)=(-1)*l_coor(i,1);
103       l_coor(i+26*2,2)=l_coor(i,2);
104       l_coor(i+26*3,1)=(-1)*l_coor(i,1);
105       l_coor(i+26*3,2)=(-1)*l_coor(i,2);
106   end
```

```
107
108 l_coor(105:115,:)= [ (m8+a1/b1*0)      0               0 0 0 0
        ... % 105   % Central Axis #1
109                     ; (m5+a2/b1*0)      0               0 0 0 0
                            ... % 106   % Central Axis #2
110                     ; (m4+a3/b1*0)      0               0 0 0 0
                            ... % 107   % Central Axis #3
111                     ; (m3+a4/b1*0)      0               0 0 0 0
                            ... % 108   % Central Axis #4
112                     ;  0                0               0 0 0 0
                            ... % 109   % Middle Point Central Axis
113                     ;-(m3+a4/b1*0)      0               0 0 0 0
                            ... % 110   % Central Axis #5
114                     ;-(m4+a3/b1*0)      0               0 0 0 0
                            ... % 111   % Central Axis #6
115                     ;-(m5+a2/b1*0)      0               0 0 0 0
                            ... % 112   % Central Axis #7
116                     ;-(m8+a1/b1*0)      0               0 0 0 0
                            ... % 113   % Central Axis #8
117                     ; (a5/b2*0)         m2+0            0 0 0 0
                            ... % 114   % Square
118                     ; (a5/b2*0)        -m2+0            0 0 0 0
                            ... % 115   % Square
119                     ];
120
121 l_coor(116:129,:)= [  a5    6              0 0 0 0 ... %   1
       % Bridle #1R (RIGHT)
122                      ; a5    4             0 0 0 0 ... %   2
                            % Bridle #1R (RIGHT)
123                      ; a5    2             0 0 0 0 ... %   3
                            % Bridle #1R (RIGHT)
124                      ; a5    1             0 0 0 0 ... %   4
                            % Bridle #1R (RIGHT)
```

```
125                         ; a5    0               0 0 0 0 ... %   5
                                % Bridle #1R (RIGHT)
126                         ; a5   -1               0 0 0 0 ... %   6
                                % Bridle #1R (RIGHT)
127                         ; a5   -2               0 0 0 0 ... %   7
                                % Bridle #1R (RIGHT)
128                         ; a5   -4               0 0 0 0 ... %   8
                                % Bridle #1R (RIGHT)
129                         ; a5   -6               0 0 0 0 ... %   9
                                % Bridle #1R (RIGHT)
130                         ; a5   -4               0 0 0 0 ... %  10
                                % Bridle #1R (RIGHT)
131                         ; a5   -2               0 0 0 0 ... %  11
                                % Bridle #1R (RIGHT)
132                         ; a5    0               0 0 0 0 ... %  12
                                % Bridle #1R (RIGHT)
133                         ; a5    2               0 0 0 0 ... %  13
                                % Bridle #1R (RIGHT)
134                         ; a5    4               0 0 0 0 ... %  14
                                % Bridle #1R (RIGHT)
135                         ];
136
137
138 % Definition of bars
139 % Since nodes were defined earlier, matrix "l_conc" contains in
        which way
140 % nodes are connected
141 % #node1 #node2 %#bar
142 l_conc=[  1   2 ... %  1    % Bridle #1 (START)
143         ;  2   3 ... %  2
144         ;  3   4 ... %  3
145         ;  4   5 ... %  4
146         ;  5   6 ... %  5
147         ;  6 105 ... %  6    % Bridle #1 (OVER)
```

```
148         ;  1   7 ... %  7    % Bridle #2 (START)
149         ;  7   8 ... %  8
150         ;  8   9 ... %  9
151         ;  9  10 ... % 10
152         ; 10  11 ... % 11
153         ; 11 106 ... % 12    % Bridle #2 (OVER)
154         ;  1  12 ... % 13    % Bridle #3 (START)
155         ; 12  13 ... % 14
156         ; 13  14 ... % 15
157         ; 14  15 ... % 16
158         ; 15  16 ... % 17
159         ; 16 107 ... % 18    % Bridle #3 (OVER)
160         ;  1  17 ... % 19    % Bridle #4 (START)
161         ; 17  18 ... % 20
162         ; 18  19 ... % 21
163         ; 19  20 ... % 22
164         ; 20  21 ... % 23
165         ; 21 108 ... % 24    % Bridle #4 (OVER)
166         ;  1  22 ... % 25    % Bridle #5 (START)
167         ; 22  23 ... % 26
168         ; 23  24 ... % 27
169         ; 24  25 ... % 28
170         ; 25  26 ... % 29
171         ; 26 114 ... % 30    % Bridle #5 (OVER)
172         ];
173
174
175 for i=1:30
176         l_conc(i+30+0*30,1)=l_conc(i,1)+1*26;
177         l_conc(i+30+0*30,2)=l_conc(i,2)+1*26;
178         l_conc(i+60+0*30,1)=l_conc(i,1)+2*26;
179         l_conc(i+60+0*30,2)=l_conc(i,2)+2*26;
180         l_conc(i+90+0*30,1)=l_conc(i,1)+3*26;
181         l_conc(i+90+0*30,2)=l_conc(i,2)+3*26;
```

```
182 end
183
184     l_conc(36,2)=105;
185     l_conc(42,2)=106;
186     l_conc(48,2)=107;
187     l_conc(54,2)=108;
188     l_conc(60,2)=115;
189     l_conc(66,2)=113;
190     l_conc(72,2)=112;
191     l_conc(78,2)=111;
192     l_conc(84,2)=110;
193     l_conc(90,2)=114;
194     l_conc(96,2)=113;
195     l_conc(102,2)=112;
196     l_conc(108,2)=111;
197     l_conc(114,2)=110;
198     l_conc(120,2)=115;
199
200    l_conc(121:132,:)=[ 105 106 ... % 121    % Central Axis (
           START)
201                       ; 106 107 ... % 122
202                       ; 107 108 ... % 123
203                       ; 108 109 ... % 124
204                       ; 109 110 ... % 125
205                       ; 110 111 ... % 126
206                       ; 111 112 ... % 127
207                       ; 112 113 ... % 128
208                       ; 108 114 ... % 129
209                       ; 114 110 ... % 130
210                       ; 110 115 ... % 131
211                       ; 115 108 ... % 132    % Central Axis (OVER
                              )
212                       ];
213
```

```
214   l_conc(133:146,:)=[ 116 117 ... % 133
215                       ; 117 118 ... % 134
216                       ; 118 119 ... % 135
217                       ; 119 120 ... % 136
218                       ; 120 121 ... % 137
219                       ; 121 122 ... % 138
220                       ; 122 123 ... % 139
221                       ; 123 124 ... % 140
222                       ; 124 125 ... % 141
223                       ; 125 126 ... % 142
224                       ; 126 127 ... % 143
225                       ; 127 128 ... % 144
226                       ; 128 129 ... % 145
227                       ; 129 116 ... % 146
228                       ];
229
230 % Z Coordinate Generation
231 % Parabolic
232 for b=1:124
233     if abs(l_coor(b,2))<l2
234         l_coor(b,3)=-m1*(1-(l_coor(b,2)/l2)*(l_coor(b,2)/l2));
235     else
236         l_coor(b,3)=m1*(abs(l_coor(b,2))-l2)/l2*2;
237     end
238 end
239 for b=125:size(l_coor,1)
240     l_coor(b,3)=-m1*(1-(l_coor(3,2)/l2)*(l_coor(3,2)/l2));
241 end
242
243 for i=1:14
244     l_coor(129+i,1)=l_coor(115+i,1)-9.9;
245     l_coor(129+i,2)=l_coor(115+i,2);
246     l_coor(129+i,3)=l_coor(115+i,3);
247 end
```

```
for j=1:14
    for i=1:14
        l_coor(143+i+14*(j-1),1)=l_coor(129+i,1)-3.3*j;
        l_coor(143+i+14*(j-1),2)=l_coor(129+i,2);
        l_coor(143+i+14*(j-1),3)=l_coor(129+i,3);
    end
end
for i=1:14
    l_coor(339+i,1)=-1*l_coor(115+i,1);
    l_coor(339+i,2)=l_coor(115+i,2);
    l_coor(339+i,3)=l_coor(115+i,3);
end

% Deck Flanges
for j=1:16
    for i=1:14
        l_conc(146+i+14*(j-1),1)=l_conc(132+i,1)+j*14;
        l_conc(146+i+14*(j-1),2)=l_conc(132+i,2)+j*14;
    end
end
% Deck Webs
l_conc(371:375,:)=[ 117 129 ... % 147
                  ; 118 128 ... % 148
                  ; 120 127 ... % 149
                  ; 122 126 ... % 150
                  ; 123 125 ... % 151
                  ];
for j=1:16
    for i=1:5
        l_conc(375+i+5*(j-1),1)=l_conc(370+i,1)+14*j;
        l_conc(375+i+5*(j-1),2)=l_conc(370+i,2)+14*j;
    end
end
% Deck Plate
```

```
for j=1:5
    for i=1:16
        l_conc(455+i+16*(j-1),1)=130-j+14*(i-1);
        l_conc(455+i+16*(j-1),2)=130-j+14*i;
    end
end
for i=1:16
        l_conc(535+i,1)=116+14*(i-1);
        l_conc(535+i,2)=116+14*i;
end
for i=1:16
        l_conc(551+i,1)=124+14*(i-1);
        l_conc(551+i,2)=124+14*i;
end
  coun=0;
for j=1:15
    for i=1:9
        coun=coun+1;
        vd(1,coun)=129+i+14*(j-1);
    end
end
radii=0.5;
count=567;
for j=1:size(vd,2)
    for i=1:115
        li=sqrt((l_coor(vd(1,j),1)-l_coor(i,1))^2+(l_coor(vd(1,
            j),2)-l_coor(i,2))^2+(l_coor(vd(1,j),3)-l_coor(i,3))
            ^2);
        if li-radii<0&&li>0.005
            count=count+1;
            l_conc(count,1)=vd(1,j); l_conc(count,2)=i;
        end
    end
end
```

```
314
315 % Conversion and saving for MSA Calculation
316 % Idealization of structure as a 3D frame
317 % "coord" is a matrix for bridge nodes
318 % "conect" is a matrix for bridge nodes connection
319 coord = l_coor;
320 coord_plot=l_coor;
321 conect = l_conc;
322 coord = l_coor*10/66;
323 % deck_load;
```

```
1
2 %% Element local stiffness matrix
3 % Guillermo Rus Carlborg        2005-05-17
4 % Alejandro Rodríguez Sánchez   2012-07
5 % Julio Rodríguez Sánchez
6
7 %% Element Local Stiffness Matrix Calculator
8 % For any given case, elemental local stiffness matrix is
      calculated, as
9 % with corresponding elemental rotation matrix, for each bar
10
11 % Material & geometric properties
12  l=properties(b,1); e=properties(b,4); g=properties(b,5);
13  a=properties(b,2)*properties(b,3); ix=properties(b,7); iy=
       properties(b,8); iz=properties(b,9); j=properties(b,11);
14
15  co=coord(conect(b,1),1:3); ci=coord(conect(b,2),1:3);
16
17  plane_x=ci(1)-co(1); plane_y=ci(2)-co(2); plane_z=ci(3)-co(3);
        plane_xy=sqrt((ci(1)-co(1))^2+(ci(2)-co(2))^2);
18  theta=atan(plane_y/plane_x); phi=atan(plane_z/plane_x);
19  if plane_z==0
```

```
20      if plane_x==0
21          phi=0;
22      end
23  end
24   if plane_y==0
25      if plane_x==0
26          theta=0;
27      end
28  end
29  if plane_xy==0
30     theta=0;
31     phi=pi/2*abs(plane_z)/plane_z;
32  end
33  if plane_x<0
34     theta=theta+pi;
35     phi=phi+pi;
36  end
37
38  v1=plane_x;v2=plane_y;v3=plane_z;mv1=sqrt(v1^2+v2^2+v3^2);
39  v4=(-1)*(v2+v3*0)/v1;v5=1;v6=0;mv2=sqrt(v4^2+v5^2+v6^2);
40  v7=v2*v6-v3*v5;v8=v3*v4-v1*v6;v9=v1*v5-v2*v4;mv3=sqrt(v7^2+v8
        ^2+v9^2);
41  l1=v1/mv1;m1=v2/mv1;n1=v3/mv1;
42  l2=v4/mv2;m2=v5/mv2;n2=v6/mv2;
43  l3=v7/mv3;m3=v8/mv3;n3=v9/mv3;
44
45  if v1==0
46      l1=0;m1=v2/mv1;n1=v3/mv1;
47      l2=abs(v2/v2);m2=0;n2=0;
48      l3=0;m3=l2*v3/mv1;n3=-l2*v2/mv1;
49      if v2==0
50          l1=0;m1=0;n1=1;
51          l2=1;m2=0;n2=0;
52          l3=0;m3=1;n3=0;
```

```
53      end
54   end
55
56   ld=[l1 m1 n1; l2 m2 n2; l3 m3 n3];
57
58
59   %  Ry=[cos(phi) 0  sin(phi);0 1 0 ; -sin(phi) 0 cos(phi)];
60   %  Rz=[cos(theta) sin(theta) 0; -sin(theta) cos(theta) 0; 0 0
        1];
61   %  ld=(Ry*Rz);
62
63   % Natural elastic stiffness matrix definition
64
65   kbl=[ e*a/l      0             0             0         0
                  0           -e*a/l        0             0
                 0        0            0           ...
66         ; 0        12*e*iz/l^3  0             0         0
                         6*e*iz/l^2   0           -12*e*iz/l^3   0
                        0        0            6*e*iz/l^2  ...
67         ; 0        0            12*e*iy/l^3  0        -6*e*iy/l^2
                 0            0            0            -12*e*iy/l
              ^3  0       -6*e*iy/l^2   0           ...
68         ; 0        0            0            g*j/l     0
                         0            0            0             0
                       -g*j/l    0            0           ...
69         ; 0        0           -6*e*iy/l^2   0         4*e*iy/l
                  0            0            0             6*e*iy/l
              ^2   0        2*e*iy/l     0           ...
70         ; 0        6*e*iz/l^2   0            0         0
                         4*e*iz/l     0           -6*e*iz/l^2    0
                        0        0            2*e*iz/l    ...
71         ; -e*a/l   0            0            0         0
                         0            e*a/l        0             0
                        0        0            0           ...
```

```
72        ; 0       -12*e*iz/l^3  0            0         0
                        -6*e*iz/l^2  0             12*e*iz/l^3   0
                        0        0           -6*e*iz/l^2  ...
73        ; 0        0           -12*e*iy/l^3  0         6*e*iy/l^2
                0            0            0             12*e*iy/l
             ^3  0        6*e*iy/l^2   0           ...
74        ; 0        0            0           -g*j/l     0
                         0            0            0             0
                        g*j/l    0            0           ...
75        ; 0        0           -6*e*iy/l^2   0         2*e*iy/l
                  0            0            0              6*e*iy/l
             ^2   0        4*e*iy/l     0           ...
76        ; 0        6*e*iz/l^2   0            0         0
                         2*e*iz/l     0           -6*e*iz/l^2    0
                        0        0            4*e*iz/l    ...
77        ];
78
79    mbl=density*l*[ 1/3      0                    0
                              0         0                      0
                               1/6      0                    0
                              0         0                      0
                                  ...
80        ; 0        13/35+6*iz/(5*a*l)    0                    0
                     0                     11*l/210+iz/(10*a*l)  0
                    9/70-6*iz/(5*a*l^2)  0                    0
                     0                    -13*l/420+iz/(10*a*l)
             ...
81        ; 0        0                    13/35+6*iy/(5*a*l)    0
                    -11*l/210-iy/(10*a*l)  0                      0
                    0                    9/70-6*iz/(5*a*l^2)  0
                     13*l/420-iy/(10*a*l)  0
             ...
82        ; 0        0                    0                    j
             /(3*a)    0                      0                      0
```

```
0 0 j
/(6*a) 0 0
...
; 0 0 -11*l/210-iy/(10*a*l) 0
l^2/105+2*iy/(15*a) 0 0
0 -13*l/420+iz/(10*a*l) 0
-l^2/140-iy/(30*a) 0
...
; 0 11*l/210+iz/(10*a*l) 0 0
0 l^2/105+2*iz/(15*a) 0
13*l/420-iz/(10*a*l) 0 0
0 -l^2/140-iz/(30*a)
...
; 1/6 0 0 0
0 0
1/3 0 0 0
0 0
...
; 0 9/70-6*iz/(5*a*l^2) 0 0
0 13*l/420-iz/(10*a*l) 0
13/35+6*iz/(5*a*l) 0 0
0 -11*l/210-iz/(10*a*l)
...
; 0 0 9/70-6*iz/(5*a*l^2) 0
-13*l/420+iz/(10*a*l) 0 0
0 13/35+6*iy/(5*a*l) 0
-11*l/210-iy/(10*a*l) 0
...
; 0 0 0 j
/(6*a) 0 0 0
0 0 j
/(3*a) 0 0
...
```

```
89        ; 0         0                      13*l/420-iy/(10*a*l) 0
                    -l^2/140-iy/(30*a)      0                      0
                    0                    -11*l/210-iy/(10*a*l) 0
                     l^2/105+2*iy/(15*a)    0
             ...
90        ; 0        -13*l/420+iz/(10*a*l) 0                      0
                     0                     -l^2/140-iz/(30*a)      0
                    -11*l/210-iz/(10*a*l) 0                      0
                     0                      l^2/105+2*iz/(15*a)
             ...
91        ];
```

```
1
2  %% Distributed forces
3  % Guillermo Rus Carlborg        2005-07-05
4  % Alejandro Rodríguez Sánchez   2012-07
5  % Julio Rodríguez Sánchez
6
7  %% Forces on a bar are converted to forces at each of its nodes
8
9  % vector is developed for mount defects and axial temperature-
       related
10 % forces; matrix is developed for distributed loads on bars
11  vector=[0 0 0 0 0 0 0 0 0 fb(b,14:16)]+[0 0 0 0 0 0 0 0 0 fb(b
       ,13)*properties(b,6)*properties(b,1) 0 0];
12  l=properties(b,1);
13  matrix=-[ l/2  0       0       1/2  0    0     0 0 0 0 0 0 ...
         qx    Fx1
14          ; 0    1/2     0       0    1/2  0     0 0 0 0 0 0 ...
                qy    Fy1
15          ; 0    0       1/2     0    0    1/2   0 0 0 0 0 0 ...
                qz    Fz1
```

```
16          ; 0    0       0       0    0    0     0 0 0 0 0 0 ...
                 fx    Mx1
17          ; 0    0       l^2/12  0    0    l/8   0 0 0 0 0 0 ...
                 fy    My1
18          ; 0    l^2/12  0       0    l/8  0     0 0 0 0 0 0 ...
                 fz    Mz1
19          ; l/2  0       0       1/2  0    0     0 0 0 0 0 0 ...
                 0     Fx2
20          ; 0    l/2     0       0    1/2  0     0 0 0 0 0 0 ...
                 0     Fy2
21          ; 0    0       l/2     0    0   -1/2   0 0 0 0 0 0 ...
                 0     Fz2
22          ; 0    0       0       0    0    0     0 0 0 0 0 0 ...
                 0     Mx2
23          ; 0    0      -l^2/12  0    0    l/8   0 0 0 0 0 0 ...
                 0     My2
24          ; 0   -l^2/12  0       0   -l/8  0     0 0 0 0 0 0 ...
                 0     Mz2
25          ;];
26  % Pseudo-forces elemental vector
27  f0l=-kbl*vector'-matrix*fb(b,1:12)' ;
28
29  if max(coac(conect(b,1),1:6))==1
30      for j=1:6
31          f0l(j,1)=f0l(j,1)*coac(conect(b,1),j);
32      end
33  end
34  if max(coac(conect(b,2),1:6))==1
35      for j=1:6
36          f0l(6+j,1)=f0l(6+j,1)*coac(conect(b,2),j);
37      end
38  end
```

```
1
2  %% Rotate element stiffness matrix
3  % Guillermo Rus Carlborg        2005-07-04
4  % Alejandro Rodríguez Sánchez   2012-07
5  % Julio Rodríguez Sánchez
6
7  %%
8  % Each bar has an element stiffness matrix, and all them will
       be treated
9  % and assembled in the global stiffness matrix
10 % The first step is a condensation (for the bridge model this
       is not done
11 % because it is not needed)
12 % Second, a global rotation matrix is developed for each  bar,
       thus
13 % yielding the global stiffness submatrix for each bar
14 % Finally, each submatrix is assembled in the global matrix
15
16 %% Condensation
17 %  [kbl,f0l]=condense(kbl,f0l,con(b,:));
18
19 %% Global stiffness submatrix generation and initialization of
       assembling process
20 % Rotation matrix
21  lc=[ld zeros(3:3) zeros(3:3) zeros(3:3);zeros(3:3) ld zeros
       (3:3) zeros(3:3);zeros(3:3) zeros(3:3) ld zeros(3:3);zeros
       (3:3) zeros(3:3) zeros(3:3) ld];
22 %  lcf=[ldf zeros(3:3) zeros(3:3) zeros(3:3);zeros(3:3) ldf
       zeros(3:3) zeros(3:3);zeros(3:3) zeros(3:3) ldf zeros(3:3);
       zeros(3:3) zeros(3:3) zeros(3:3) ldf];
23 % Rotation of pseudo-forces vector and elemental natural
       stiffness matrix
24  f0n=lc'*f0l;  kb=lc'*kbl*lc; mb=lc'*mbl*lc;
25 % Variables for plotting purposes
```

```
26  rows=[(conect(b,1)-1)*nd+(1:nd)  (conect(b,2)-1)*nd+(1:nd)];
27  i=conect(b,1);   j=conect(b,2);   m=size(coac,2);
28 %% Assembling stiffness matrix
29 % This changes submatrices inside the global matrix
30 % Natural elastic stiffness matrix
31  k(m*i-(m-1):m*i,m*i-(m-1):m*i)=k(m*i-(m-1):m*i,m*i-(m-1):m*i)+
        kb(1:m,1:m);
32  k(m*i-(m-1):m*i,m*j-(m-1):m*j)=k(m*i-(m-1):m*i,m*j-(m-1):m*j)+
        kb(1:m,m+1:2*m);
33  k(m*j-(m-1):m*j,m*i-(m-1):m*i)=k(m*j-(m-1):m*j,m*i-(m-1):m*i)+
        kb(m+1:2*m,1:m);
34  k(m*j-(m-1):m*j,m*j-(m-1):m*j)=k(m*j-(m-1):m*j,m*j-(m-1):m*j)+
        kb(m+1:2*m,m+1:2*m);
35 % Natural elastic mass matrix
36  km(m*i-(m-1):m*i,m*i-(m-1):m*i)=km(m*i-(m-1):m*i,m*i-(m-1):m*i
        )+mb(1:m,1:m);
37  km(m*i-(m-1):m*i,m*j-(m-1):m*j)=km(m*i-(m-1):m*i,m*j-(m-1):m*j
        )+mb(1:m,m+1:2*m);
38  km(m*j-(m-1):m*j,m*i-(m-1):m*i)=km(m*j-(m-1):m*j,m*i-(m-1):m*i
        )+mb(m+1:2*m,1:m);
39  km(m*j-(m-1):m*j,m*j-(m-1):m*j)=km(m*j-(m-1):m*j,m*j-(m-1):m*j
        )+mb(m+1:2*m,m+1:2*m);
40 %% Assembling pseudo-forces vector
41  f0(m*i-(m-1):m*i,1)=f0(m*i-(m-1):m*i,1)+f0n(1:m,1);
42  f0(m*j-(m-1):m*j,1)=f0(m*j-(m-1):m*j,1)+f0n(m+1:2*m,1);
```

```
1 % Matrix analysis
2 % Guillermo Rus Carlborg        2005-07-03
3 % Alejandro Rodríguez Sánchez   2012-07
4 % Julio Rodríguez Sánchez
5
6 %% Geometric Linearity Calculation (START)
7 % This is an elastic first order analysis
```

```
% Geometric non-linearity is not taken into account in this step

% Stiffness matrices and forces vectors for every bar will be calculated
% here, then all them will be combined in an assembling process
 for b=1:nb;
  ema;f0ma;rota;
 end;

% Forces vectors and stiffness matrices are changed here
% The general equation we are solving is: [Kt]{u}+{f0}={f}
% Here variables will be changed just to get a more welcome code

 f=(f-f0);
 kt=k;

%  c=find(con_m);
%  keq=kt-kt(:,c)*(kt(c,c))^(-1)*kt(c,:);
%  feq=f-(kt(:,c)*(kt(c,c))^(-1)*f(c));

% Now the system is: [K]{u}={f}

% Solving the system [K]{u}={f}
% Solving this system can be tricky because of some numerical problems
% Often, for big sized stiffness matrix, it appears that inversion of the
% stiffness matrix can be a source of error
% Dealing with this usually makes us to try algebraic apparatus in order to
% develop an accurate solution for the system
```

```
35 % A quick and easy way to solve for numerical implications is
       an L U
36 % decomposition of the system, which has been developed here
37 % Possibly, direct method works faster than L U decomposition
       method, so it
38 % is encouraged to use the first one if no errors are appearing
        during the
39 % solving process of the system
40
41 % Solving via Direct Method (START)
42  direct;
43
44 % Solving via Direct Method (OVER)
45
46 % Solving via L U Decomposition (START)
47 %  lu_decomposition;
48 % Solving via L U Decomposition (OVER)
49 % Geometric Linearity Calculation (OVER)
50
51 % Deforming previous state
52 displacements=zeros(nn,m);
53
54 for b=1:size(displacements,1)
55     displacements(b,1:6)=u(b*m-(m-1):b*m,1);
56 end
57 if displacements(109,3)>0
58     u=-1*u;
59 end
60
61 variablesv=[2 3 4 5 7 11 12 13];
62 min_range=[ 0.5 0.2 0.3  1    2    1.2  0.60 3  3.8  1.5  0.02
       0.02  0.02];
63 max_range=[ 0.5 0.4 0.6  1.5  2.5  1.2  1.75 3  3.8  1.5  0.10
       0.10  0.10];
```

```
64 for i=1:size(variablesv,2)
65     variablesm(i,:)=log(parametersm(variablesv(i),1)/min_range
           (1,variablesv(i)))/log(max_range(1,variablesv(i))/
           min_range(1,variablesv(i)));
66 end
67 freqm1=dlmread('modes.txt');
68 freqm=freqm1(:,1:8)/100;
69 ucoefficients=dlmread('ucoefficients.txt');
70 summ=0;
71 for i=1:size(freqm,2)
72     for j=1:size(variablesm,1)
73         summ=summ+variablesm(j,1)*ucoefficients(j+size(
               variablesm,1)*(i-1),1)*freqm(:,i);
74     end
75 end
76 u(1:690,1)=u(1:690,1)+summ;
77
78  fadjusted=kt(1:690)*u(1:690);
```

```
1
2 %% Equation System Direct Solution
3 % Guillermo Rus Carlborg
4 % Alejandro Rodríguez Sánchez   07-2012
5 % Julio Rodríguez Sánchez
6
7 % Free DOFs and fixed DOFs are distinguished here
8 fr=find(1-coac_m(:)); fx=find(coac_m(:));
9
10 % Solution for unknown displacements
11 u(fr)=kt(fr,fr)\(f(fr)-kt(fr,fx)*u(fx));
12
13 % Solution for unknown forces (reactions)
14 f(fx)=kt(fx,fr)*u(fr)+kt(fx,fx)*u(fx)+f0(fx);
```

```
%Created by Julio Rodríguez Sáchez

%% Stress Calculator

%%

stress_bars=zeros(size(l_conc,1));
for b=1:size(conect,1)
    co=coord(conect(b,1),1:3); ci=coord(conect(b,2),1:3);
    do=displacements(conect(b,1),1:3); di=displacements(conect(
        b,2),1:3);
    mo=co+do;mi=ci+di;
    strain(b,1)=(sqrt((mi(1)-mo(1))^2+(mi(2)-mo(2))^2+(mi(3)-mo
        (3))^2)-properties(b,1))/properties(b,1);
    stress_bars(b,1)=strain(b,1)*properties(b,4);
end
stress_nodes=zeros(2,size(l_coor,1));
for b=1:size(stress_bars,1)
    stress_nodes(1,l_conc(b,1))=stress_nodes(1,l_conc(b,1))+
        stress_bars(b,1);
    stress_nodes(2,l_conc(b,1))=stress_nodes(2,l_conc(b,1))+
        stress_bars(b,1);
end

for i=1:115
    stress(i,1)=max(stress_nodes(:,i));
end

cav=[27 28 29 30 31 32];
ebv=[1 2 3 4 5 6 22 23 24 25 26];
ibv=[7 8 9 10 11 12 13 14 15 16 17 18 19 20 21];

for i=1:size(cav,2)
    scav(i,1)=stress(cav(i),1);
```

```
31 end
32 for i=1:size(ebv,2)
33     sebv(i,1)=stress(ebv(i),1);
34 end
35 for i=1:size(ibv,2)
36     sibv(i,1)=stress(ibv(i),1);
37 end
38
39 s2max=[max(scav); max(sebv); max(sibv)];
40
41 variablesv=[2 3 4 5 7 11 12 13];
42 min_range=[ 0.5 0.2 0.3  1    1.8  1.2  0.60 3  3.8  1.5  0.02
       0.02  0.02];
43 max_range=[ 0.5 0.4 0.6  1.5  2.5  1.2  1.75 3  3.8  1.5  0.10
       0.10  0.10];
44 % parametersm(1:10,o)=parametersm(1:10,o)*10/66;
45 for i=1:size(variablesv,2)
46     variablesm(i,:)=log(parametersm(variablesv(i),1)/min_range
           (1,variablesv(i)))/log(max_range(1,variablesv(i))/
           min_range(1,variablesv(i)));
47 end
48 sadm=[1 0 0; 0 1 0; 0 0 1];
49 scoefficients=dlmread('scoefficients.txt');
50 for i=1:8
51     for j=1:3
52     scoefficientsm(j,i)=scoefficients(j+3*(i-1),1);
53     end
54 end
55 summ=0;
56 for i=1:size(scoefficientsm,2)
57     summ=summ+variablesm(i,1)*sadm*scoefficientsm(:,i);
58 end
59 s2max(:,:)=s2max(:,:)+(summ);
```

```
%Created by Julio Rodríguez Sáchez

%% Volume Calculator

%%

cfrp_volume=0;

for b=1:132
    cfrp_volume=cfrp_volume+properties(b,1)*properties(b,2)*
        properties(b,3);
end

gfrp_volume=0;

for b=1:17
    gfrp_volume=gfrp_volume+1.8*0.05*0.0036*2+0.25*1.8*0.0036;
end
gfrp_volume=gfrp_volume+0.0036*1.8*10;
```

```
%%%Julio Rodríguez Sáchez
%%%Manuel & Juan Chiachio
%%%NASA Ames Research Center
%%%April 2013

%plotting module
format compact
global Ply
global Damage
global Laminate
global Loads

 path(path,'lib')
```

```

%%%PLY DEFINITION (STANFORD)

Ex=127.553e9;                %[Pa] On−axis (in−plane) (Young's modulus)
Ey=8.411e9;                  %[Pa] On−axis (in−plane) (Transverse Young's modulus
Es=6.205e9;                  %[Pa] Shear modulus (in−plane)
nux=0.309;                   %[no units] in plane Poisson's ratio
nuy=nux*(Ey/Ex);             %[no units] in− plane plane Poisson's ratio
nuyz=0.49;                   %[no units] out of plane Poisson's ratio
Gyz=Ey/(2*(1+nuyz));         %[Pa] out of plane shear modulus
plythick=0.1524e−3;          %[m] Ply Thickness
G_d0=9.07e13;                %[Pa/m] this is a purely experimental parameter. See Ogihara1995, Table 3.
G_d0=2.07e12;                %[Pa/m] this is one trial by Manuel
G_c=150.609;                 %[J/m^−2] intralaminar critical energy release rate  150.609


PlyData=[Ex,Ey,Es,nux,nuy,nuyz,Gyz,plythick];   %characterizing vector

Ply=struct('Ex',Ex,'Ey',Ey,'Es',Es,'nux',nux,'nuy',nuy,'nuyz',nuyz,'Gyz',Gyz,'plythick',plythick,'G_d0',G_d0);

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %LAMINATE DEFINITION (STANFORD)
```

```
37 % lam_type='x-ply';
38 % StackSeq=[0 0 90 90 90 90 90 90 90 90 0 0];  %total laminate.
       From top to bottom
39 % Sub_s_StackSeq=[0 0];                          %sublaminate 1
40 % Sub_90_StackSeq=[90 90 90 90 90 90 90 90];    %sublaminate 2
41 % Laminate.StackSeq=struct('StackSeq',StackSeq);
42 % h=numel(Laminate.StackSeq)*plythick/2;        %[m] Laminate
       half-thickness
43 % B=0.044;                                  %[m] Laminate half-width
       (if variable section, choose the critical one)
44
45
46 %LAMINATE DEFINITION (GUDMUNDSON)
47 lam_type='x-ply';
48 StackSeq=[0 0 90 90 90 90 90 90 90 90 0 0];  %total laminate.
       From top to bottom
49 Sub_s_StackSeq=[0 0];            %sublaminate 1 (Remember: only
       one of the 0° Ply stack)
50 Sub_90_StackSeq=[90 90 90 90 90 90 90 90];   %sublaminate 2
51 Laminate.StackSeq=StackSeq;
52 h=numel(Laminate.StackSeq)*plythick/2;       %[m] Laminate half
       -thickness
53 B=0.044;                                %[m] Laminate half-width (
       if variable section, choose the critical one)
54
55
56
57 %for [0m/90n]s laminates (Takeda and Nairm)
58 n_90=numel(Sub_90_StackSeq);               % number of 90 plies
       in the central sublaminate
59 m_0=numel(Sub_s_StackSeq);                 %number of plies in
       each 0 sublaminate
60 t_90=(n_90/2)*plythick;          %[90n] sublaminate half-
       thickness
```

```
t_s=m_0*plythick;               %[0m] sublaminate total thickness (each)

Laminate.Size=struct('n_90', n_90, 'm_0', m_0, 't_s', t_s,'t_90', t_90,'h',h,'B',B);

[a_art_0,eng_cnst_0]=LamTheory(PlyData,StackSeq);     %[Pa] undamaged laminate stiffness modulus (longitudinal off-axis)
E_0=eng_cnst_0.E_1;
[a_art_S,eng_cnst_S]=LamTheory(PlyData,Sub_s_StackSeq);    %[Pa] undamaged s] sublaminate stiffness modulus ((longitudinal off-axis)
E_Sub_s=eng_cnst_S.E_1;
[a_art_90,eng_cnst_90]=LamTheory(PlyData,Sub_90_StackSeq);  %[Pa] undamaged s] sublaminate stiffness modulus ((longitudinal off-axis)
%E_Sub_90=1/a_art_90(1,1);
E_Sub_90=eng_cnst_90.E_1;

[a_art_90_onax,eng_cnst_90_onax]=LamTheory(PlyData,0*Sub_90_StackSeq);  %[Pa] undamaged 90 sublaminate stiffness modulus (longitudinal on-axis)

Laminate.Stiffness=struct('E_0', E_0,'E_Sub_s',E_Sub_s,'E_Sub_90',E_Sub_90,'G_c',G_c,...
    'a_art_0',a_art_0,'a_art_S',a_art_S,'a_art_90_onax',a_art_90_onax,'eng_cnst_0',eng_cnst_0,'eng_cnst_S',eng_cnst_S,'eng_cnst_90_onax',eng_cnst_90_onax);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
81
82 %DAMAGE FEATURES
83 %cracks and local delamination
84
85 if cycle==1
86 %rhoGud=(1/0.0254)*10.6251*(2*Laminate.Size.t_90); %%(Entrada
       densidad microgrietas)
87 %l=0.0254*0.5*(1/(10.6251));
88 l=0.01;  %[m] Averge half-spacing
89
90
91 crck_den=1/(2*l);                              %[m^-1] crack
       density (minimum, 0.025 #/mm^-1)
92
93 rhoGud=Laminate.Size.t_90/l;
94
95
96 %for [0m/90n]s laminates
97 %rho=1/(2*crck_den*t_90);
98 rho=l/Laminate.Size.t_90;   %dimensionless half spacing
99 ad_crck_den=1/(2*rho); %adimensional crack density
100
101 delam_rt=0;                                    %[0-1] local
       delamination ratio (prop to l)
102 delam=delam_rt*l;                              %[m] local
       delamination half-lenght.
103 Δ=delam/Laminate.Size.t_90;                % dimensionaless
       delamination ratio
104
105
106 %for general laminates
107
108 %%
109
```

```
110         %undamlampoiss=-a_art_0(2,1)/a_art_0(1,1);
111         %undamlampoiss=eng_cnst_0.nu_21;
112        % damlampoiss=0.6*undamlampoiss;%-a_art_S(2,1)/a_art_S
             (1,1);
113
114         fact1=(1-Laminate.Stiffness.eng_cnst_90_onax.nu_12*
               Laminate.Stiffness.eng_cnst_0.nu_12)/...
115             (1-Laminate.Stiffness.eng_cnst_90_onax.nu_12*
                  Laminate.Stiffness.eng_cnst_90_onax.nu_21);
116         fact2=1+Laminate.Stiffness.eng_cnst_S.nu_12*(
               Laminate.Stiffness.a_art_S.a_12*Laminate.Size.t_90+
               ...
117             Laminate.Stiffness.a_art_90_onax.a_12*
                  Laminate.Size.t_s)/(
                  Laminate.Stiffness.a_art_S.a_22*
                  Laminate.Size.t_90+
                  Laminate.Stiffness.a_art_90_onax.a_11*
                  Laminate.Size.t_s);
118
119
120         %fact1=(1-Ply.nux*undamlampoiss)/(1-Ply.nux*Ply.nuy);
121         %fact2=damlampoiss*(1+(a_art_S(1,2)*Laminate.Size.t_90
               +a_art_90(1,2)*Laminate.Size.t_s)/(a_art_S(2,2)*
               Laminate.Size.t_90+a_art_90(1,1)*Laminate.Size.t_s)
               );
122
123         a=(eng_cnst_90_onax.E_2*Laminate.Size.t_90)/(
               eng_cnst_S.E_1*Laminate.Size.t_s)*fact1*fact2;
124
125         lambda=Laminate.Size.t_s/Laminate.Size.t_90;
126
127         %a=(Laminate.Stiffness.E_Sub_90/(lambda*
               Laminate.Stiffness.E_Sub_s))*1;%fact1*fact2;
128
```

```
Damage.Cracks=struct('l',l,'crck_den',crck_den,'rho',rho,'
    rhoGud',rhoGud);
Damage.LocalDel=struct('delam_rt',delam_rt,'delam',delam,'Δ',Δ)
    ;


%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%Relative Stiffness Reduction E/E_0



 mdl='variational';

        lambda=Laminate.Size.t_s/Laminate.Size.t_90;

C_1=1/Laminate.Stiffness.E_Sub_90+1/(lambda*
    Laminate.Stiffness.E_Sub_s);
C_2=Ply.nuyz*(lambda+2/3)/Laminate.Stiffness.E_Sub_90-lambda*
    Ply.nux/(3*Laminate.Stiffness.E_Sub_s);
C_3=(lambda+1)*(3*lambda^2+12*lambda+8)/(60*
    Laminate.Stiffness.E_Sub_90);
C_4=1/3*(1/Ply.Gyz+lambda/Ply.Es);
p=(C_2-C_4)/C_3; q=C_1/C_3;
alpha=1/2*sqrt(2*sqrt(q)-p); beta=1/2*sqrt(2*sqrt(q)+p);
end

for i=1:3
    Loads.MaxStress=s2max(i,1)*1000;
    [crck_den_n En IncStiff]=fmodel_crack_energy([2.908e-4 1
        .6821],mdl,crck_den);
    crck_den=crck_den_n; rho=1/(2*crck_den_n*Laminate.Size.t_90
        );
```

```
154     [eff_stiffness RSR]=fstiffness(Ply,Laminate,rho,mdl);
155     mprop(1,1)=YM1(ymdist)*RSR;mprop(2,1)=YM2(ymdist)*RSR;mprop
            (3,1)=YM3(ymdist)*RSR;
156     rsrv(i,cycle)=RSR;
157     if crck_den≥450
158         stop_crck=1;
159     end
160 end
```

```
1  function [crck_den_1 En IncStiff]=fmodel_crack_energy(V,mdl2,
       crck_den)
2
3  global Ply
4  global Laminate
5  global Loads
6  global Damage
7
8
9  % and the one before the last is the exponent.
10 %const=exp(th(1)); %comment this line if the multiplicative
       constant A is not a Jeffrey's parameter
11 const=V(1);
12
13 expnt=V(2);  %comment this line if the exponent is not a
       Jeffrey's parameter
14
15
16 %DC_times=linspace(1,max_times,1650/2); %times grid in which
       model is evaluated
17 %DC_times=linspace(1,max_times,1651);
18
19 %     convert_crack_param(crck_den)  %this function updates the
        crack parameters providing ant_data (crack density [m^-1])
```

```
20
21      [G IncStiff]=Crack_Energy(Ply,Laminate,Loads,Damage,mdl2);
22      En=G;
23      crck_den_1=crck_den+(const*(G)^expnt);
24
25
26  end
```

```
1
2   function [eff_stiffness RSR]=fstiffness(Ply,Laminate,rho,mdl)
3
4
5    fact1=(1-Laminate.Stiffness.eng_cnst_90_onax.nu_12*
        Laminate.Stiffness.eng_cnst_0.nu_12)/...
6               (1-Laminate.Stiffness.eng_cnst_90_onax.nu_12*
                    Laminate.Stiffness.eng_cnst_90_onax.nu_21);
7     fact2=1+Laminate.Stiffness.eng_cnst_S.nu_12*(
         Laminate.Stiffness.a_art_S.a_12*Laminate.Size.t_90+...
8               Laminate.Stiffness.a_art_90_onax.a_12*
                    Laminate.Size.t_s)/(
                    Laminate.Stiffness.a_art_S.a_22*
                    Laminate.Size.t_90+
                    Laminate.Stiffness.a_art_90_onax.a_11*
                    Laminate.Size.t_s);
9
10
11          a=(Laminate.Stiffness.eng_cnst_90_onax.E_2*
                Laminate.Size.t_90)/...
12               (Laminate.Stiffness.eng_cnst_S.E_1*
                    Laminate.Size.t_s)*fact1*fact2;
13
14          lambda=Laminate.Size.t_s/Laminate.Size.t_90;
15
```

```
         %a=(Laminate.Stiffness.E_Sub_90/(lambda*
            Laminate.Stiffness.E_Sub_s))*1;%fact1*fact2;

ad_crck_den=1/(2*rho);
crck_den=ad_crck_den/Laminate.Size.t_90;

%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%Relative Stiffness Reduction E/E_0



 %mdl='COD';

 switch mdl

     case 'shearlag'


K1=Ply.G_d0;
K2=Ply.Gyz/Laminate.Size.t_90;
K3=3*Ply.Gyz/Laminate.Size.t_90;

shpar=(1/(Laminate.Stiffness.eng_cnst_90_onax.E_2)+...
    1/(lambda*Laminate.Stiffness.eng_cnst_S.E_1));

         beta1=sqrt(K1*Laminate.Size.t_90*shpar); %shear lag
            parameter (we can use many models for this
            parameter. See Talreja book)
         beta2=sqrt(K2*Laminate.Size.t_90*shpar);
         beta3=sqrt(K3*Laminate.Size.t_90*shpar);

         R=(2/beta3)*tanh(beta3*rho);
```

```
45          RSR=1/(1+a*ad_crck_den*R);
46
47
48      case 'variational'
49
50
51  C_1=1/Laminate.Stiffness.E_Sub_90+1/(lambda*
        Laminate.Stiffness.E_Sub_s);
52  C_2=Ply.nuyz*(lambda+2/3)/Laminate.Stiffness.E_Sub_90-...
53      lambda*Ply.nux/(3*Laminate.Stiffness.E_Sub_s);
54  C_3=(lambda+1)*(3*lambda^2+12*lambda+8)/(60*
        Laminate.Stiffness.E_Sub_90);
55  C_4=1/3*(1/Ply.Gyz+lambda/Ply.Es);
56
57  p=(C_2-C_4)/C_3; q=C_1/C_3;
58
59
60
61  if 4*q/p^2 ≥1
62  alpha=1/2*sqrt(2*sqrt(q)-p); beta=1/2*sqrt(2*sqrt(q)+p);
63
64  R=(4*alpha*beta/(alpha^2+beta^2))*(cosh(2*alpha*rho)-cos(2*beta
        *rho))/...
65      (beta*sinh(2*alpha*rho)+alpha*sin(2*beta*rho));
66  else
67      disp('JCHR warning: 4*q/p^2 <1')
68  end
69          RSR=1/(1+a*ad_crck_den*R);
70
71
72       case 'COD'
73       %[complrelation,complwdamage,compdegrad,Eff_compl]=
             Gudmunson_stiff(Ply,Laminate,Damage);
74        [complrelation,~,~,~]=Gudmunson2(Ply,Laminate,1/(2*rho));
```

```
75
76      RSR=complrelation;
77      %Eff_compl;
78
79      case 'potential'
80          a=5.454e-3;
81          b=0.5114;
82
83          RSR=1-a*crck_den^b;
84
85  end
86
87  eff_stiffness=Laminate.Stiffness.E_0*RSR;
```

```
1  %Created by Julio Rodríguez Sáchez
2
3  for k=1:ymdist
4      G_ELS(k,cycle)=maxdisp(ymdist,cycle)+10/600;
5  end
6
7  pf_els(:,cycle)=G_ELS(:,cycle)≤0;
8  nf_els(cycle,1)=mean(sum(pf_els(:,cycle))/ymdist);
9
10 for k=1:ymdist
11     G_ELU(k,cycle)=maxstrs(ymdist,cycle)-1200000;
12 end
13
14 pf_elu(:,cycle)=G_ELU(:,cycle)≥0;
15 nf_elu(cycle,1)=mean(sum(pf_elu(:,cycle))/ymdist);
16
17 % fprintf('Num. muestras: %d\n', useful_life);
18 % fprintf('Probabilidad de fallo - ELS: %12.4e\n',mean(nf_els))
     ;
```

```
19 % fprintf('Coef. de variación: %12.4e\n',std(nf_els)/mean(
     nf_els));
20 % fprintf('Probabilidad de fallo — ELU: %12.4e\n',mean(nf_elu))
     ;
21 % fprintf('Coef. de variación: %12.4e\n',std(nf_elu)/mean(
     nf_elu));
```

```
1  %Created by Julio Rodríguez Sáchez
2
3
4  mass=(cfrp_volume*1700+gfrp_volume*2000);
5  material_cost=mass*70;
6  construction_cost=1*material_cost;
7  initial_cost=(material_cost+construction_cost)/0.8+10000+20000;
8  revision_cost=0.05*construction_cost;
9
10 acc_n=acc_n*acc_y;
11 us_n=us_n*us_y;
12 if noise==1,   noise=noise+1;   end;
13
14 acc_cost=1000*acc_n+5000;
15 us_cost=4000+2000*(noise-1);
16
17 xus=1-min(rsrv(:,cycle));
18 muus=0.025*noise;
19 sigmaus=0.035*noise;
20 xacc=1-min(rsrv(:,cycle));
21 muacc=0.025;
22 sigmaacc=0.035;
23
24 pod_us=normcdf(xus,muus,sigmaus);
25 pod_acc=normcdf(xacc,muacc,sigmaacc);
26
```

```
mafail_m=[1-nf_els(cycle,1)-nf_elu(cycle,1);
    nf_els(cycle,1); nf_elu(cycle,1)];
mofail_m=[1-pod_us*pod_acc pod_acc pod_us];

scost_m=[0 revision_cost revision_cost;...
         0 0.1*initial_cost 0.1*initial_cost; ...
         initial_cost 0.2*initial_cost 0.2*initial_cost];
pcost_m=mafail_m*mofail_m;

cost_m=scost_m.*pcost_m;

cycle_cost(cycle,1)=sum(sum(cost_m));
```

List of Figures

1.1 Global structural system optimization flow chart. 2

1.2 3D Model of the geometry of the CFRP Bridge 6

1.3 Sketch of the geometry of the CFRP Bridge 7

2.1 Bridge optimization process flow chart. 10

2.2 Damage evolution algorithm flow chart. 18

2.3 Cost function computation algorithm flow chart. 21

3.1 Error in stresses versus logarithm of element size. 23

3.2 Stresses in kPa for Ultimate Limit State loading. 24

3.3 Vertical displacements in m for Serviceability Limit State loading. 24

3.4 First global mode shape for Vibrations Limit State loading. . 25

3.5 Value of Coefficient α_{51} vs. Number of Samples. 26

3.6 Value of Coefficient α_5 vs. Number of Samples. 26

3.7 Value of error vector norm vs. number of mode shapes. . . . 27

3.8 Value of mean error in maximum displacement vs. number of mode shapes. 27

3.9 Cost distribution of optimum bridge. 30

3.10 Sensitivity analysis of the optimum CFRP Bridge 31

3.10 Sensitivity analysis of the optimum CFRP Bridge 32

3.10 Sensitivity analysis of the optimum CFRP Bridge 33

3.10 Sensitivity analysis of the optimum CFRP Bridge 34

3.10 Sensitivity analysis of the optimum CFRP Bridge 35

3.11 Geometry of the optimum bridge. 36

3.12 Stresses in kPa for Ultimate Limit State loading. 36

3.13 Vertical displacements in m for Serviceability Limit State loading. 37

3.14 First global mode shape for Vibrations Limit State loading. . 37

List of Tables

1.1 Value ranges for the geometric parameters of the bridge. . . 7

2.1 Value ranges for the sectional properties of the bridge. . . . 11

2.2 Value ranges for the optimization variables. 19

3.1 Value of maximum stress in the bridge and yielding stress of CFRP. 24

3.2 Value of maximum vertical displacement of the bridge and maximum vertical displacement prescribed by the IAP-11. . 24

3.3 Value of first global mode shape frequency and minimum frequency allowed by the IAP-11. 25

3.4 First set of optimum values for the geometric parameters and sectional properties of the bridge. 28

3.5 First set of optimum values for monitoring equipment and lifetime of the bridge. 29

3.6 Second set of optimum values for the geometric parameters and sectional properties of the bridge. 29

3.7 Second set of optimum values for monitoring equipment and lifetime of the bridge. 30

3.8 Value of maximum stress in the bridge and yielding stress of CFRP. 36

3.9 Value of maximum vertical displacement of the bridge and maximum vertical displacement prescribed by the IAP-11. . 37

3.10 Value of first global mode shape frequency and minimum frequency allowed by the IAP-11. 37

Bibliography

[1] B. J. Bichon, J. M. McFarland, S. Mahadevan, Efficient surrogate models for reliability analysis of systems with multiple failure modes, Reliability Engineering & System Safety 96 (10) (2011) 1386–1395.

[2] D. M. Frangopol, A. Strauss, K. Bergmeister, Lifetime cost optimization of structures by a combined condition-reliability approach, Engineering Structures 31 (7) (2009) 1572–1580.

[3] V. M. Karbhari, Fiber reinforced composite bridge systems-transition from the laboratory to the field, Composite Structures 66 (1-4) (2004) 5–16.

[4] Z. K. Awad, T. Aravinthan, Y. Zhuge, F. Gonzalez, A review of optimization techniques used in the design of fibre composite structures for civil engineering applications, Materials & Design 33 (2012) 534–544.

[5] L. C. Bank, T. Gentry, B. P. Thompson, J. S. Russell, A model specification for FRP composites for civil engineering structures, Construction and Building Materials 17 (6-7) (2003) 405–437.

[6] M. Chiachio, J. Chiachio, G. Rus, Reliability in composites–A selective review and survey of current development, Composites Part B: Engineering 43 (3) (2012) 902–913.

[7] M. C. G. Rus, J. Chiachío, Estructura autotensada para puente de

material compuesto, patent number: P200802147 (z00894600000516) (02 2011).

[8] R. Burgueño, A. M. Asce, J. Wu, Membrane-Based Forms for Innovative FRP Bridge Systems through Structural Optimization (October) (2006) 453–461.

[9] E. J. Barbero, D. H. Cortes, A mechanistic model for transverse damage initiation, evolution, and stiffness reduction in laminated composites, Composites Part B: Engineering 41 (2) (2010) 124–132.

[10] A. Hosoi, K. Takamura, N. Sato, H. Kawada, Quantitative evaluation of fatigue damage growth in CFRP laminates that changes due to applied stress level, International Journal of Fatigue 33 (6) (2011) 781–787.

[11] M. L. Ribeiro, V. Tita, D. Vandepitte, A new damage model for composite laminates, Composite Structures 94 (2) (2012) 635–642.

[12] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P. Kevin Tucker, Surrogate-based analysis and optimization, Progress in Aerospace Sciences 41 (1) (2005) 1–28.

[13] N. Jansson, W. Wakeman, J.-a. Må nson, Optimization of hybrid thermoplastic composite structures using surrogate models and genetic algorithms, Composite Structures 80 (1) (2007) 21–31.

[14] Instrucción sobre las acciones a considerar en el proyecto de puentes de carretera, IAP-11, Dirección General de Carreteras, Ministerio de Fomento, Madrid, 2011.

[15] Eurocode 3: Design of steel structures : Part 2 - Steel bridges, AFNOR, Paris, 2007.

[16] A. Chopra, Dynamics of Structures: Theory and Applications to Earthquake Engineering, Prentice Hall International Series in Civil Engineering And, Pearson/Prentice Hall, 2007.

[17] a. Olsson, G. Sandberg, O. Dahlblom, On Latin hypercube sampling for structural reliability analysis, Structural Safety 25 (1) (2003) 47–68.

[18] R. McGuire, W. & Gallagher, Matrix Structural Analysis, John Wiley & Sons, 2000.

[19] C. Y. Song, J. Lee, Reliability-based design optimization of knuckle component using conservative method of moving least squares metamodels, Probabilistic Engineering Mechanics 26 (2) (2011) 364–379.

[20] A. I. Forrester, A. J. Keane, Recent advances in surrogate-based optimization, Progress in Aerospace Sciences 45 (1-3) (2009) 50–79.

[21] H. Jeffreys, Theory of probability, Oxford University Press, London.

[22] R. Talreja, C. Singh, Damage and Failure of Composite Materials, Damage and Failure of Composite Materials, Cambridge University Press, 2012.

[23] J. A. Nairn, 2.12 - matrix microcracking in composites, in: E. in Chief: Anthony Kelly, C. Zweben (Eds.), Comprehensive Composite Materials, Pergamon, Oxford, 2000, pp. 403 – 432.

[24] R. Joffe, J. Varna, Analytical modeling of stiffness reduction in symmetric and balanced laminates due to cracks in 90° layers, Composites Science and Technology 59 (11) (1999) 1641 – 1652.

[25] L. J. R. Saxena, Goebel, Chang, Accelerated aging experiments for prognostics of damage growth in composites materials, The 8th In-

ternational Workshop on Structural Health Monitoring, F.-K. Chang, Editor., Stanford, CA , (2011).

[26] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsgaii, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 181–197.

[27] S.-Y. Lee, G. Rus, T. Park, Detection of stiffness degradation in laminated composite plates by filtered noisy impact testing, Computational Mechanics 41 (1) (2007) 1–15.

[28] G. Rus, S. Y. Lee, S. Y. Chang, S. C. Wooh, Optimized damage detection of steel plates from noisy impact test, International Journal for Numerical Methods in Engineering 68 (7) (2006) 707–727.

[29] T. H. P. S. Y. Lee, G. Rus, Quantitative nondestructive evaluation of thin plate structures using the complete frequency from impact testing, Structural Engineering and Mechanics, Vol 28, 5(2008).