ISTANBUL TECHNICAL UNIVERSITY ★ EARTHQUAKE ENGINEERING AND DISASTER MANAGEMENT INSTITUTE

FATIGUE ANALYSIS OF SIMPLE SPAN RIVETED STEEL RAILWAY BRIDGES

M.Sc. THESIS

Aylin ERTİK

Earthquake Engineering and Disaster Management Institute

Earthquake Engineering Program

SEPTEMBER 2019

ISTANBUL TECHNICAL UNIVERSITY ★ EARTHQUAKE ENGINEERING AND DISASTER MANAGEMENT INSTITUTE

FATIGUE ANALYSIS OF SIMPLE SPAN RIVETED STEEL RAILWAY BRIDGES

M.Sc. THESIS

Aylin ERTİK
501111206

Earthquake Engineering and Disaster Management Institute

Earthquake Engineering Program

Thesis Advisor: Assoc. Prof. Dr. Barlas Özden ÇAĞLAYAN

SEPTEMBER 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ DEPREM MÜHENDİSLİĞİ VE AFET YÖNETİMİ ENSTİTÜSÜ

TEK AÇIKLIKLI PERÇİNLİ ÇELİK DEMİRYOLU KÖPRÜLERİNDE YORULMA ANALİZİ

YÜKSEK LİSANS TEZİ

Aylin ERTİK
501111206

Deprem Mühendisliği ve Afet Yönetimi Enstitüsü

Deprem Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Barlas Özden ÇAĞLAYAN

EYLÜL 2019

Aylin Ertik, a **M.Sc.** student of ITU **Earthquake Engineering and Disaster Management Institute**, student ID 501111206, successfully defended the **thesis** entitled **"FATIGUE ANALYSIS OF SIMPLE SPAN RIVETED STEEL RAILWAY BRIDGES"**, which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Barlas Özden ÇAĞLAYAN** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Filiz PİROĞLU** Istanbul Technical University	.....
	Prof. Dr. Bilge DORAN Yildiz Technical University	.....

Date of Submission : 13 September 2019
Date of Defense : 13 September 2019

To my family,

FOREWORD

First of all, I would like to thank my respectful thesis advisor Assoc. Prof. Dr. Barlas Özden Çağlayan for being that much helpful, humble, patient, and sharing his experiences and wisdom in both lessons, Bachelor's and Master's theses.

I am really grateful to Uğur Usuğ for his moral and technical support during this thesis process. As a MSc computer engineer, he used his knowledge in Python, spent a lot of time and made great effort on the codes used in the analysis, which made most of the numerical analysis possible, more precise and easier. I appreciate the help of Burcu Nerkis Kaçaroğlu, Emre Kaçaroğlu and Mustafa Yiğit Battal in running half of my SAP2000 models by their high-performance computers and sending me the results very soon, even if they are living in different cities and even countries. They made me save a half week in a very busy schedule. I would also like to thank Tijen Bayer for caring about me and always offering help.

Finally, I would like to thank my family who devoted their limited conditions to my education and encouraged me during my whole life.

September 2019

Aylin ERTİK
Civil Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

AASHTO	**:** American Association of State Highway and Transportation Officials
FEM	**:** Finite Elements Method
LEFM	**:** Linear Elastic Fracture Mechanics
TCDD	**:** Türkiye Cumhuriyeti Devlet Demiryolları (Turkish State Railways Administration in Turkish)
3D	**:** Three dimensional

SYMBOLS

A	**:** Coefficient for fatigue detail categories
d	**:** Fatigue damage for $\Delta\sigma$ stress range
d_i	**:** Fatigue damage for $\Delta\sigma_i$ stress range
D	**:** Total fatigue damage
n	**:** Number of cycles for $\Delta\sigma$ stress range
n_i	**:** Number of cycles for $\Delta\sigma_i$ stress range
n_j	**:** Number of cycles for $\Delta\sigma_j$ stress range
n_k	**:** Number of cycles for $\Delta\sigma_k$ stress range
N	**:** Fatigue life cycle at stress range $\Delta\sigma$
N_i	**:** Fatigue life cycle at stress range $\Delta\sigma_i$
N_j	**:** Fatigue life cycle at stress range $\Delta\sigma_j$
N_k	**:** Fatigue life cycle at stress range $\Delta\sigma_k$
S_N, $\Delta\sigma$, $(\Delta\sigma)_n$	**:** Stress range
S-N curve	**:** Table showing the relationship between stress ranges S and number of fatigue life cycles N in log-log scales

LIST OF TABLES

Page

LIST OF FIGURES

Page

FATIGUE ANALYSIS OF SIMPLE SPAN RIVETED STEEL RAILWAY BRIDGES

SUMMARY

Most of the steel railway bridges have been constructed over the past century and a great amount of them completed more than half of their service lives. This is caused by the neglection of fatigue effects in the design of these bridges and the increase of vehicle loads and traffic density in today's world which shortens the remaining fatigue life of the bridges more over. Today, it is known that fatigue failure is one of the main reasons for the sudden collapse of many old bridges. Under these circumstances, it is clearly seen that the fatigue safety of current old bridges should be investigated for public safety and they should be strengthened or replaced if necessary. Thereupon, determination of the fatigue damage accumulation and prediction of the remaining life of the bridges have become one of the most common topics of bridge engineering.

In this thesis study, fatigue analysis of an existing single span, riveted and plane girder steel railway bridge which was built in 1900's, is carried out. Stress-based approach is applied, which is the most common method in fatigue analysis. The chosen bridge is modeled with frame elements in a finite element program. Train load profiles and traffic history tables for the time period which the bridge was in service are constructed according to the railway vehicle and load statistics of the Turkish State Railways Administration (TCDD). The live loads are applied to the structural model. After running the analysis in the finite element program, the frame element axial stresses are maintained. The critical stress points and sections of the critical elements for fatigue analysis are chosen by making comparisons. Stress graphs for the chosen critical stress points showing stress-load case interaction in each train profile passage over the bridge are drawn. Afterwards, the full cycles of stress ranges in those stress graphs are counted using rainflow cycle method. The number of full cycles of stress ranges are plotted to the stress histograms. Every single histogram which is obtained for any element in one passage of any train profile over the bridge is multiplied by the traffic history data and result histograms are obtained for each critical bridge element. Finally, fatigue damage accumulations according to the detail categories in AASHTO, were calculated from these histograms using Miner's Rule. The total fatigue damage over the critical elements are obtained. The analysis results show that, the stinger midspan is the only place where fatigue damage accumulation occurred and the bridge spent approximately one fifth of its fatigue life. The results are evaluated and recommendations are given.

TEK AÇIKLIKLI PERÇİNLİ ÇELİK DEMİRYOLU KÖPRÜLERİNDE YORULMA ANALİZİ

ÖZET

Ülkemizde kullanılan çelik demiryolu köprülerinin çoğu geçtiğimiz yüzyılın başlarında yapılmıştır. Bu köprülerin çoğu servis ömürlerinin yarıdan fazlasını tamamlamış durumdadırlar. Bunun sebepleri, bu köprülerin yapıldığı dönemlerde tasarımda yorulma etkilerinin göz ardı edilmesi, günümüzde taşıt yüklerinin ve trafik yoğunluğunun artması ve köprülerin yapıldıkları tarihten bugüne çevresel etkilere maruz kalmalarıdır. Tüm bu sebeplerden, günümüzde köprülerin güvenilirliği ilgi duyulan bir konu haline gelmiştir. Köprülerde oluşan yorulma hasarı birikimlerinin ve köprülerin kalan ömürlerinin tayini üzerine çalışmalar yapılması köprü mühendislerinin temel ihtiyaçlarından biri haline gelmiştir.

Günümüzde sıklıkla kullanılan üç farklı yorulma analizi yöntemi vardır. Bunlar gerilme esaslı yaklaşım, şekil değiştirme esaslı yaklaşım ve lineer elastik kırılma mekaniği (LEFM) yaklaşımlarıdır.

Bu tez çalışmasında, 1900'lü yıllarda inşa edilmiş mevcut bir tek açıklıklı, perçinli, düzlem kirişli çelik demiryolu köprüsü üzerinde yorulma analizleri yapılmıştır. Yorulma analizi yöntemi olarak, en sık kullanılan yaklaşım olan gerilme esaslı yaklaşım seçilmiştir. Seçilen köprü sonlu elemanlar yöntemiyle çalışan SAP2000 yapısal analiz programında çubuk eleman yöntemiyle modellenmiştir. Köprünün üzerinden geçen yükleri elde edebilmek için Türkiye Cumhuriyeti Devlet Demiryolları'nın demiryolu yük ve demiryolu araçları istatistikleri ile trafik geçmişi bilgileri elde edilerek buna uygun katar yükü profilleri oluşturulmuştur. Bu katar yüklerinin hepsinin tekil yüklerini ve bu yüklerin konumlarını elle girerek oluşturmak hem çok zaman alacak hem de hataya mahal verebilecektir. Bu sebeple Python programı kullanılarak, lokomotif ve vagonların dingil yük ve mesafelerinin bilgilerini içeren bir text dosyası ile oluşturulan yük profillerinin hangi lokomotif ve vagonların kombinasyonundan oluştuğu bilgilerini içeren başka bir text dosyasını giriş verisi olarak kullanıp bu yük profillerini tüm tekil yük ve konumları ile çıktı olarak veren bir kod hazırlanmıştır. Boylamaların serbest mesafesinin orta noktalarında ve enlemelerle birleşim noktalarında yük bulunması durumunda moment diyagramları en elverişsiz durumları vereceğinden, bu yük profillerinin, her adımda, açıklığın yarısı kadar ilerletilmesine karar verilmiştir. Bu amaçla da bir Basic programı olan QB64 ile, bu yük profillerinin köprüye girip köprüden çıkana kadar her adımda dingillerin hangi çubuk elemanlar üzerinde hangi konumlarda olduğunu çıktı olarak veren bir kod hazırlanmıştır. Bu kodun sonucunda elde edilen text dosyası .csv formatında bir dosyaya dönüştürülmüştür, böylece SAP2000'de "Interactive Database Editing" sekmesindeki Excel formatındaki verilerle aynı formatta veri elde edilmiş ve bu veriler programa aktarılabilmiştir. Böylece bir yük profilini köprü üzerinde, açıklığın yarısı kadar adımlarla (bu köprü için 80 cm) gezdirmek için tüm yük durumları SAP2000

modeline girilmiştir. Programı çalıştırdıktan sonra eleman eksenel gerilme sonuçları tablosu Excel'e çıkarılmıştır. Bu veri hem Python'da yazılan kodla hem de mühendislik nosyonu kullanılarak elenerek her elemanda yorulma için en elverişsiz durumun oluştuğu kesit ve o kesitteki ilgili nokta (alt ya da üst lifin köşe noktaları) belirlenmiştir. Daha sonra her kritik eleman tipinde seçilen bu noktalarda her bir yük profili altında eksenel gerilme değerleri birer gerilme grafiğine işlenmiştir. Bu gerilme grafiklerindeki gerilme aralıklarının tam tekrarı yorulma analizinde kullanılacak veridir, ancak bunu harmonik olmayan bu grafikler üzerinden hesaplamak önemli bir problemdir. Bu problemi çözmek için geliştirilmiş çeşitli yöntemlerden "rainflow" çevrim sayma metodu kullanılmıştır. Bu nümerik analizi yapabilmek için Python için hazırlanmış "rainflow" kod kütüphanesinden yararlanılmıştır. Girdisi gerilme – yük adımı numarası değerleri olan bu kodun çıktısı olarak her bir kritik köprü elemanın seçilen kesitinin seçilen noktasında her bir yük profilinin köprüden bir defa geçişi altında oluşan gerilme aralıklarının tam tekrarlarını gösteren histogramlar elde edilmiştir. Bu histogramlar, yine bir Python kodu yardımıyla trafik geçmişi bilgisi ile çarpılarak, geçmişten incelenen tarihe kadarki tüm süre boyunca her bir yük profilinin ilgili defalarca köprünün üzerinden geçmesi sonucu, her bir kritik elemanın seçilen kritik noktasında oluşan toplam histogramlar elde edilmiştir. Son olarak da her kritik eleman için tüm yük profillerinin toplamından oluşan sonuç histogramı çıkarılmıştır. Daha sonra bu histogramlardaki veriler kullanılarak, AASHTO'daki detay kategorilerine göre verilen yorulma sınırını gösteren gerilme-çevrim grafiği üzerinden, Miner Kuralı kullanılarak elemanlardaki toplam yorulma hasar birikimleri lineer hasar birikimi yöntemiyle hesaplanmıştır. Böylece, yapılışından incelenen tarihe kadar köprünün kritik elemanlarının seçilen kritik noktalarında, yorulma ömrünün ne kadarının tamamlandığı bulunmuştur. Analiz sonuçlarına göre, sadece boylama serbest açıklık ortalarında yorulma ömrünün yaklaşık beşte birinin tamamlandığı görülmüştür. Sonuçlar değerlendirilmiş ve öneriler sunulmuştur.

1. INTRODUCTION

Infrastructure and transportation systems are considered to be the heart of Turkish economy and leading to economic improvement. Its importance has been described as following “the infrastructure supporting human activities includes complex and interrelated physical, social, ecological, economic, and technological systems such as transportation, energy production and distribution; water resources management; waste management; facilities supporting urban and rural communities; communications; sustainable resources development; and environmental protection.” [1]. The aspects of the transportation systems affect both the conditions of life due to lags and gridlocks, and the safety of millions of passengers every day. Bridges become the critical component of any transportation systems by providing vital connection to roads across valleys or other natural barriers.

One of the main challenges for today’s bridges is the requirement of frequent repair and maintenance to keep necessary serviceability due to constant condition deterioration, material degradation and traffic growth. The increment of heavy traffic over recent years further makes it worse. This is crucial for many current bridges, which have been designed according to the old standards.

Railway bridges are subjected to dynamic forces due to moving trains. The repetitive loads, even though ordinarily could not lead to strength failure, may produce damage accumulation due to fatigue on bridges [2]. Fatigue, as described in materials science, is the continuous and regional structural damage that appears when material is imposed to cyclic or alternating strains at nominal stresses much less than the ultimate tensile stress limit [3]. The most used approach to assess the fatigue damage is the S-N curves approach [4]. The S-N curves shows the relationship between stress ranges and load cycles. Stress ranges and load cycles are marked in logarithmic scales, where stress ranges are in the vertical axis and the load cycles are in the horizontal axis. The S-N curves approach is generally employed to obtain the number of cycles to failure generated by constant stress amplitude [5]. When the structure is subjected to complex

loading, the cumulative damage due to fatigue is usually estimated by the Palmgren-Miner linear damage hypothesis, which will be explained in detail afterwards.

In this study, it is aimed to indicate the time-varying dynamic loads caused by the casual traffic. In order to achieve this, firstly, a three dimensional (3D) structural model of the bridge is developed using the commercial finite-elements-method (FEM)-based structural analysis program SAP2000 [6]. Casual traffic loads over the bridge are converted to a SAP2000-suitable form using BASIC Language with QB64 [7], a BASIC compiler. It consists of load cases for a passage of every train load profile over the bridge. The axial stress results in each load case are filtered from the analysis results for each critical element and plotted to stress-load case graphs for each load profile. After that, rainflow cycle counting numerical studies are carried out on the stress graphs and the resultant cycle counts for each stress graph are shown in histograms for each critical bridge element. These histograms are multiplied with the traffic history data, thus total histograms for each critical bridge member on the given traffic history data is obtained. Lastly, using the total histogram data, fatigue damage accumulation is obtained using linear damage accumulation theory (Miner's Rule) according to the AASHTO S-N curves table for detail categories [8].

The thesis has five chapters: Chapter 2 explains the fatigue analysis theory and approaches in detail; structural model and load profiles are given in Chapter 3; analysis results and calculations are given in Chapter 4; the conclusions and recommendations in Chapter 5 concludes the study.

2. FATIGUE ANALYSIS

Fatigue concept first took place in literature in 19th century. Fatigue is defined as the attitude of structures under the effect of repeated loads. It is an interior, continuous, and permanent structural transformation in the material. Repeated load cycles cause some micro-cracks in steel elements. At first, when the loads started to be applied, the propagation of microcracks are rather slow. As loading process continues, the propagation of the microcracks speeds up by time. Those cracks turn into macro-cracks over long periods. The macro-cracks specify the remaining fatigue life caused by the alternating stress till failure arises. The repeated loads causing fatigue failure is less than the ultimate tensile stress limit and fatigue failure is sudden and brittle.

Fatigue analysis could be assessed in three different ways which are the crack propagation method or Linear Elastic Fracture Mechanics (LEFM) approach, the strain-based approach, and the stress-based approach or S-N curve method. The fatigue analysis in this thesis study is based on the stress-based approach.

2.1 Stress-Based Approach

In the past century, the first studies aimed to understand and measure fatigue used a stress-based approach which is alternatively called as S-N curve method. Since then it is the standard method for fatigue design and applications.

To create the S-N curve, a cyclic loading having constant amplitude is applied to the experiment samples until failure is observed and the amount of cycles to failure are counted. The stress range and the number of cycles is plotted in a logarithmic scaled graph as exemplified in Figure 2.1. This curve is only valid when the stress-time values of the sample have a constant amplitude. It is crucial to increase the number of samples and numbers of cycles to failure in order to gather statistically reasonable information for each curve.

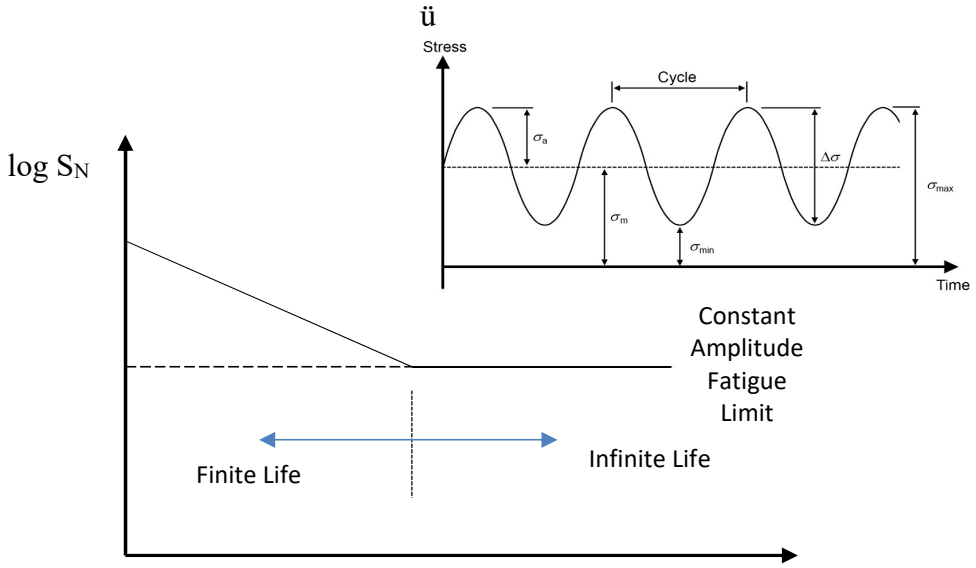


Figure 2.1 : S-N Curve showing finite-infinite life.

The major factors governing fatigue strength are the number of stress cycles, the magnitude of stress range, the type of stress range and the type of construction details. The stress ranges higher than the constant amplitude fatigue limit creates fatigue damage, thus this region is named as finite life. The stress ranges below this limit are not considered in fatigue life calculations, therefore this region is called infinite life.

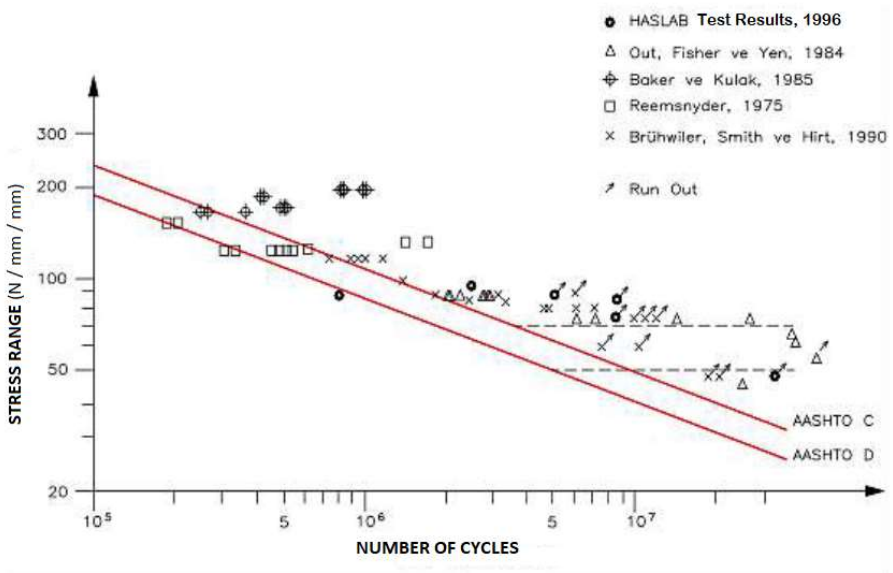


Figure 2.2 : S-N Curve showing full scale test results [9-13].

The knowledge of fatigue behavior is derived from the experimental data. A collection of the full-scale test results experimental studies in literature [9-13], are depicted in Figure 2.2.

2.2 Cycle Counting

A load cycle is a closed loop in load range over time or load cases. If the loading is harmonic, a cycle is observed as starting from a certain load value, passes one max and min value and comes back to the same initial value as shown in Figure 2.3. The amplitude and mid values define the load cycle.

When the load path is not harmonic as shown in Figure 2.4, it is more complicated to determine a load cycle. Since the stress graphs are not harmonic and different amplitudes in these graphs are crucial in fatigue analysis, a numerical analysis method which takes the different amplitudes into account, called rainflow cycle counting [14], is going to be used.

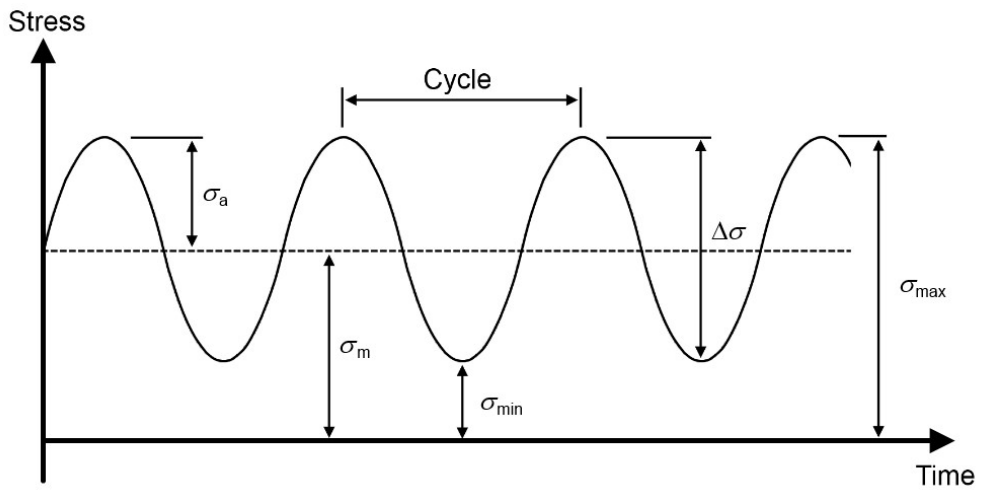


Figure 2.3 : Harmonic cycle.

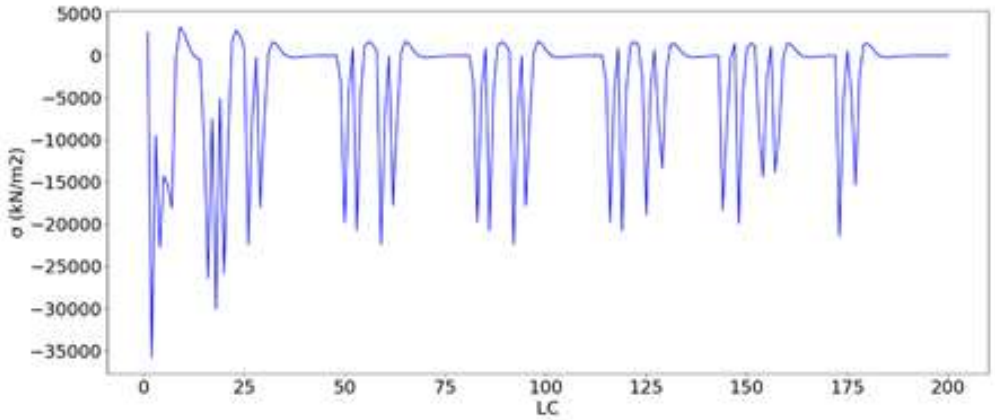


Figure 2.4 : Non-harmonic cycle.

The non-harmonic stress data should be simplified for as to have a comparable form with the constant amplitude data. This process is called as cycle counting. Among

several methods for cycle counting, one of the common and easy methods is preferred, which is rainflow cycle counting method [14].

2.2.1 Rainflow counting

Rainflow counting method is given in ASTM (1986) standards [15] and developed by Downing and Socie (1982) [16].

To be able to implement this procedure, a stress-time graph should be drawn as the stress is the ordinates and the time is the axis of the graph. In this form of graph, stress values create peaks called “Pagoda roofs”. Cycles are then defined by the manner in which rain is allowed to “drip” or “fall” down the roofs. There are some rules for the dripping rain to constitute closed cycles:

- The stress time history should be started and finished in the greatest stress value to prevent the half cycles to be counted.
- A rainflow which starts in the previous stress reversal should stop when:
    a) The flow began at a peak, falls opposite another peak greater than the peak it starts
    b) The flow began at a valley falls opposite another valley greater than the valley it starts
    c) It merges with a flow that started at an earlier rainflow.

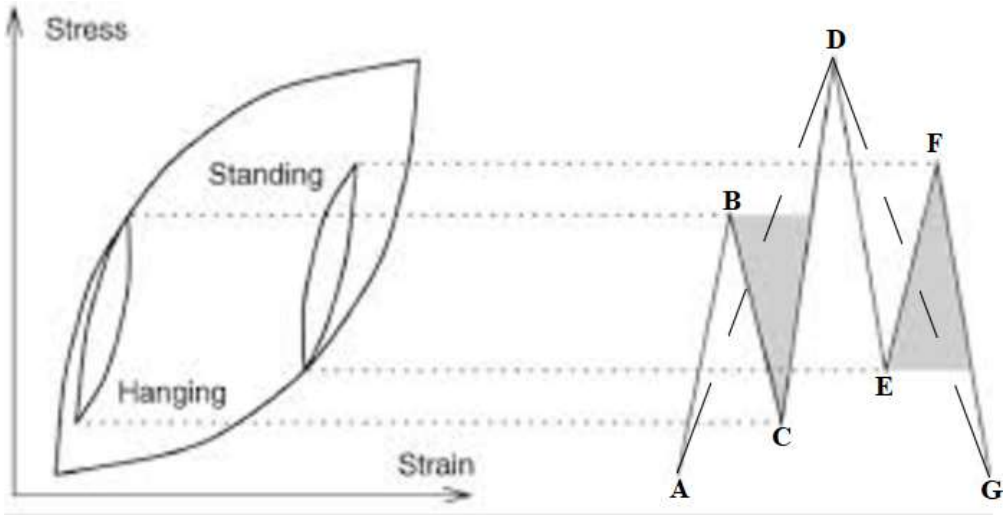


Figure 2.5 : Example to show rainflow counting [17].

As seen in Figure 2.3, the given stress –time history starts and finishes at the biggest stress value in magnitude (point A). Rainflow is started at each turnaround in the stress history. Steps of the algorithm on the example is given below:

1. Rainflow from A continues over B and D till the end of the history since there was no condition to stop the rainflow.

2. Rainflow starting from B, passes over C and stops in the level of D, as both B and D are peaks and the is a higher peak. (rule a)

3. Rainflow from C stops upon meeting the rainflow from A (rule c)

4. Rainflow from D goes over E and G, continues to the end of the history since no rules for stopping are satisfied.

5. Rainflow from E continues over F and stops at the level of G, as both E and G are valleys and G is a deeper valley than E (rule b)

In this example, A-D and D-A are combined to form a full cycle. Event B-C combines with event C-B (of stress range C-D) to form an additional cycle. Similarly, another cycle is formed at E-F.

2.3 Linear Damage Rule

The linear damage rule was first proposed by Palmgren in 1924 and was further developed by Miner in 1945 [18] and presented by J.M. Barsom and S.T. Rolfe [19]. Today, this method is commonly known as Miner's Rule. The rule states that, fatigue damage fraction d is defined as the used life for a stress level and shown as in (**2.1**).

$$d = \frac{n}{N} \tag{2.1}$$

where n is number of cycles at stress range Δσ and N is the fatigue life cycle at stress range Δσ. According to the equation above, when one cycle of loading is applied, $\frac{1}{N}$ of the fatigue life is consumed.

$$D = \sum_{i=0}^{k} d_i = \frac{n_i}{N_i} + \frac{n_j}{N_j} + \cdots + \frac{n_k}{N_k} \geq 1 \tag{2.2}$$

Fatigue failure is assumed to occur when D, the summation of damage fractions d_i, equals to or bigger than 1 as shown in **(2.2)**.

3. STRUCTURAL MODEL AND LOAD PROFILES

3.1 Structural Model

Structural analysis of the considered bridge is processed using SAP2000 version 20.2.0. A 3D structural computer model with frame elements having the same cross-sections and lengths of the current bridge elements is developed as seen in Figure 3.1.

To simulate the support conditions, fixed supports are defined in one edge and sliding supports are defined in the other edge of the frames resembling the main girders. All of the frame elements are divided in the intersections of the elements except the stringers. Because the train loads are applied to the stringers, the mid sections of each free span of the stringers are moment-critical sections. For this reason, the stringers are divided to two parts in each free span. The main girders are assigned as frame elements in the centroid line of the actual main girders. These elements are connected to dummy frame elements which resembles the projection of the main girder in the stringer-cross beam plane, and to cross beams by rigid links with no mass and weight, rotational inertias in three directions are 10^6, and fixed in all directions. Cross beams and stringers are defined in the same plane, the distance between their centers are defined with the insertion point definition in the cross-beam sections. The bracings are defined in the centroid of the actual bracings, and connected to the dummy frames of main beam by rigid links. Another group of dummy elements are also defined connecting the edges of the bracings to prevent the movement of the links towards each other. All of the mentioned properties of the model can be seen in Figure 3.2.

The dimensions of the bridge in plan view and the frame element numbers are shown in Figure 3.3. The main girders are plate girders which have different numbers of top and bottom plates in different sections. The red, yellow and dark blue sections of the main beam are in Figure 3.4 (a) in the same order. To define these sections in the model, section designer is used. The cross-sections of the cross beams and stringers are shown in Figure 3.4. (b) and (c) respectively. The cross-section of the bracings is two 80-80-8 (mm) angle profiles, placed back to back.

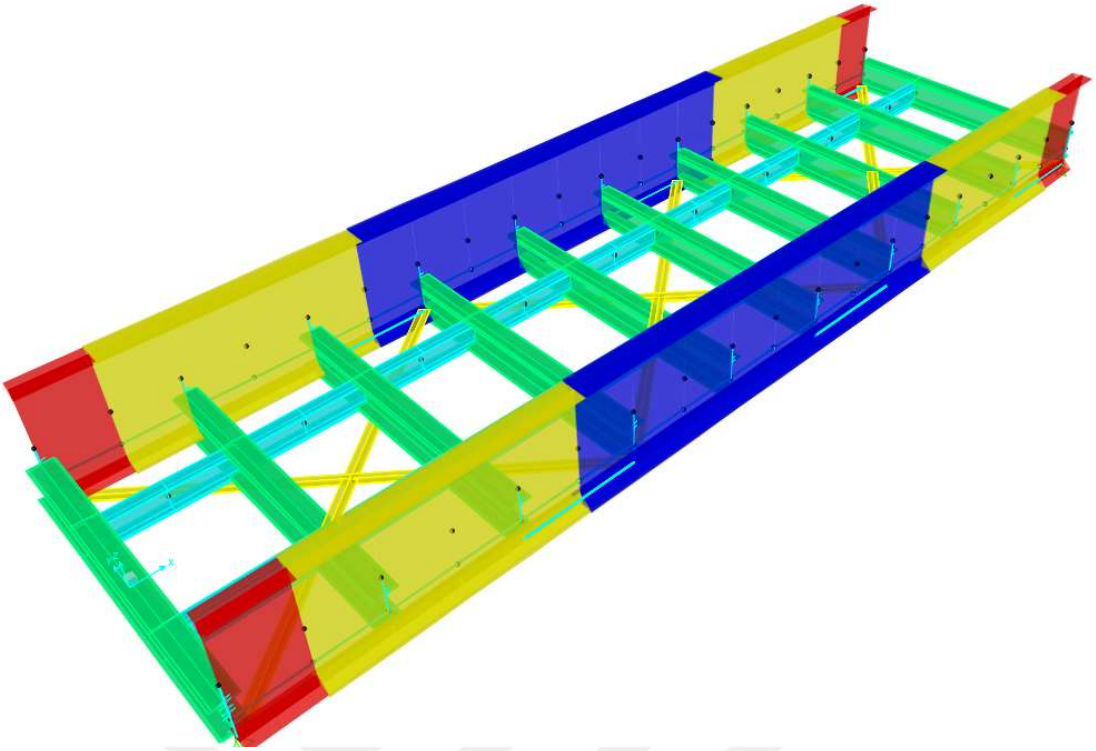


Figure 3.1: 3D extruded view of the bridge.

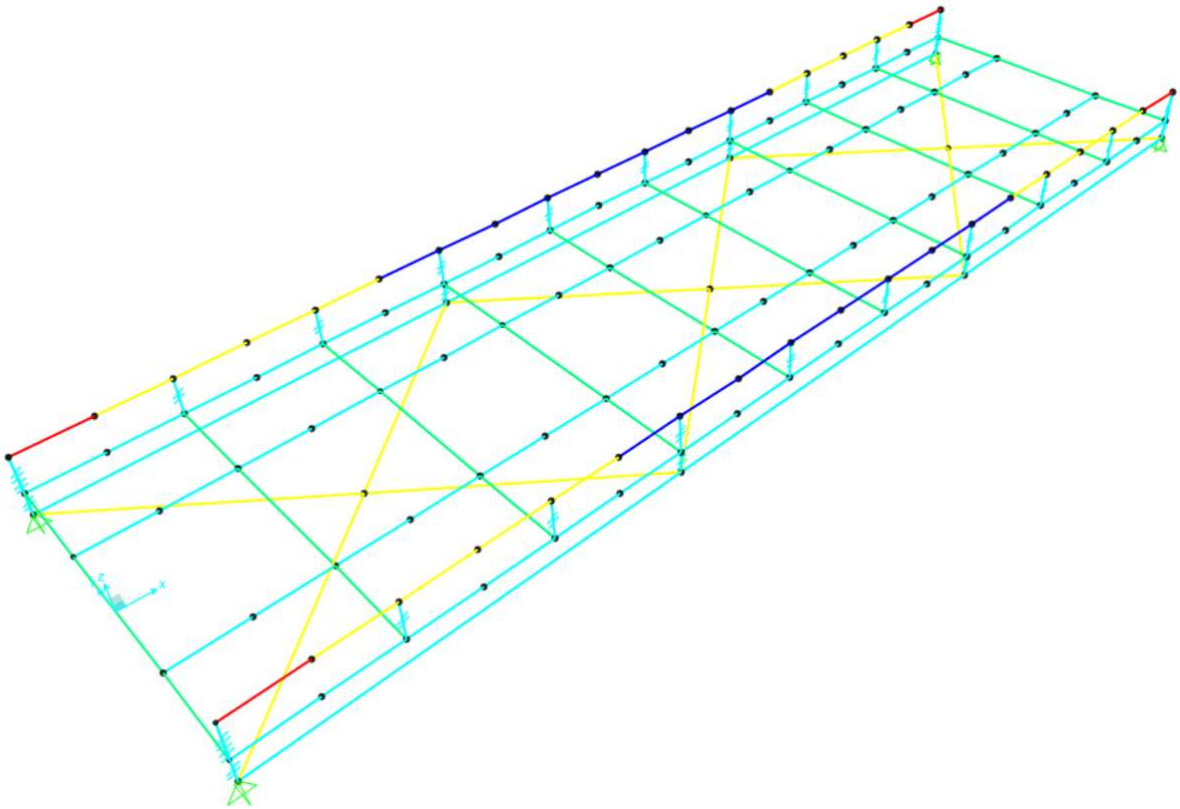


Figure 3.2: 3D model with frame elements.

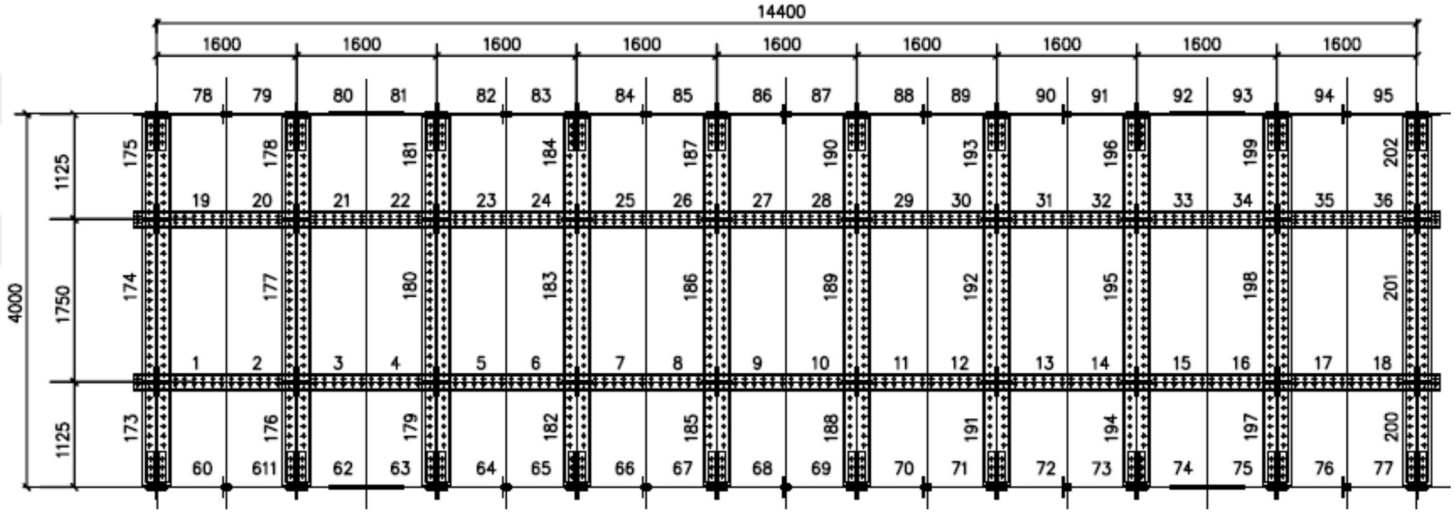


Figure 3.3: Bridge dimensions and frame element numbers.

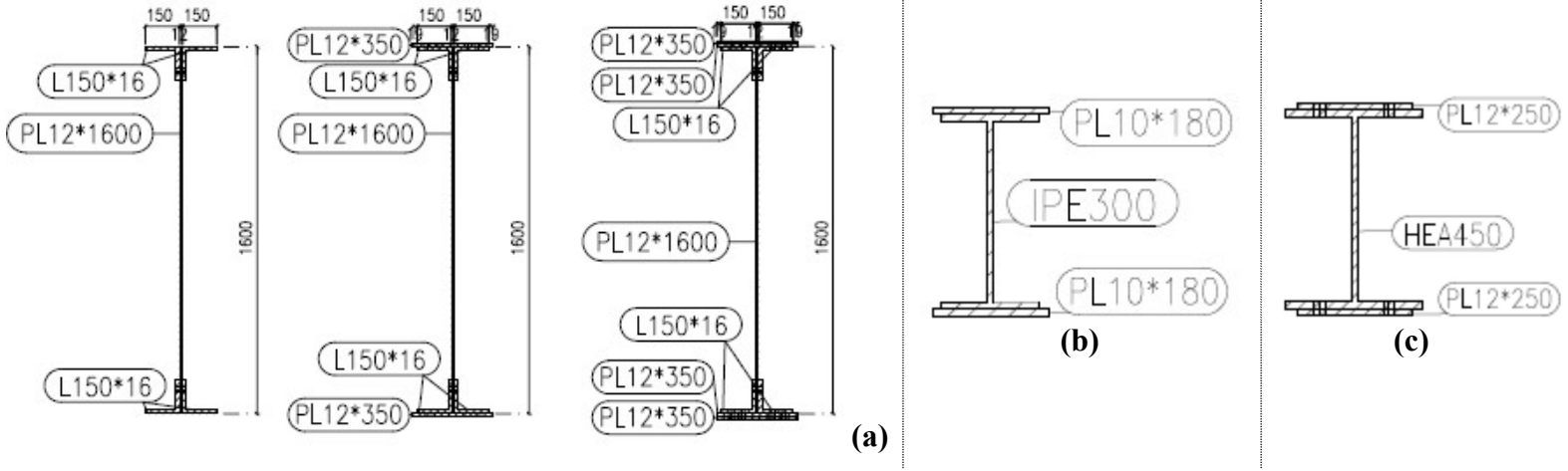


Figure 3.4: Cross sections of the elements (a)Main beam. (b) Stringer. (c)Cross beam.

As seen in Figure 3.5, the cross-section area of the dummy elements is taken as the same area of the bottom flange of the main beam and stiffness modifiers for the moment of inertia in both 2 and 3 directions are given as 10^{-5}, in this way it simulates the axial rigidities provided by the main girder bottom flange.

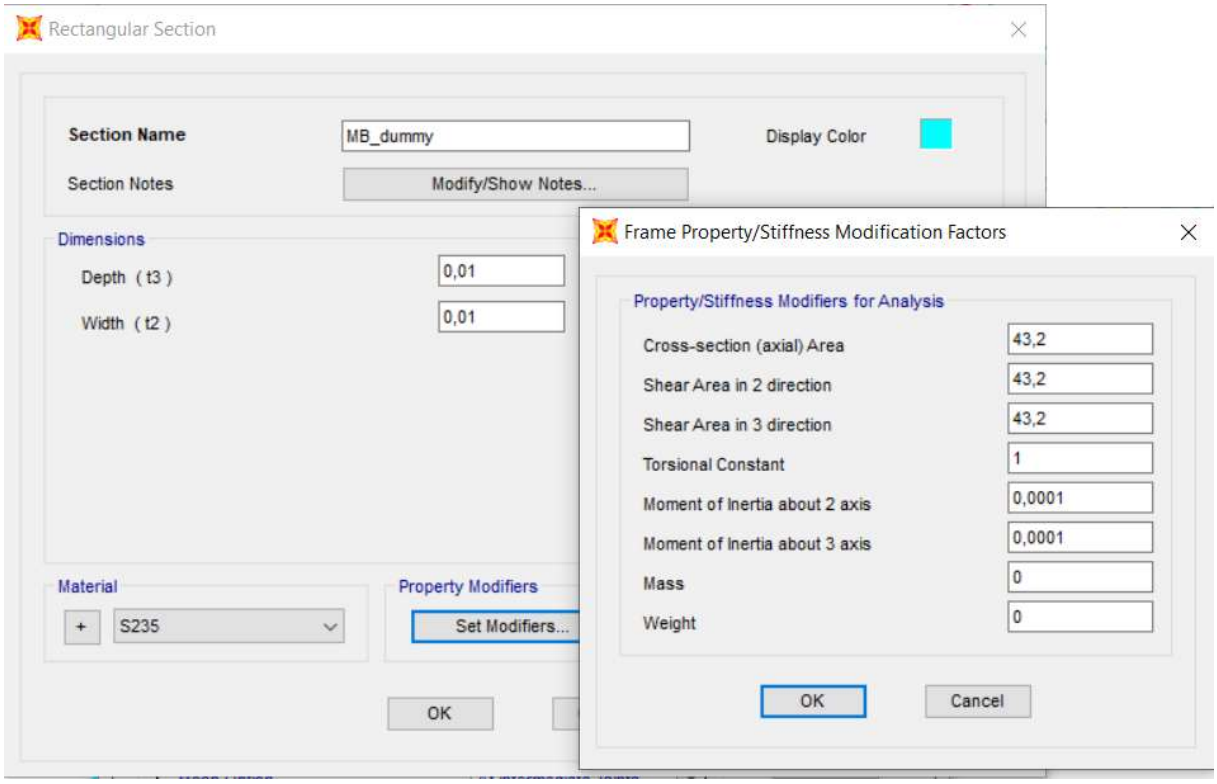


Figure 3.5: Properties of the dummy frame element.

The connections of cross beams to main girders and stringers to cross beams are fixed in all directions except M2, therefore the connections are rigid in their strong directions.

3.2 Load Profiles Applied to the Structural Model

In order to simulate the fatigue life of the considered bridge, traffic history data containing the types of trains, the axle loads and the number of passages over the bridge is required. Daily passenger, freight and suburban train passages, their axle loads and distances have been taken from TCDD [20, 21] for the active years of the bridge. Depending on the information obtained from TCDD, fatigue load profiles are produced. The locomotive and wagon combinations of these fatigue load profiles are demonstrated in Table 3.1. The axle loads and axle distances of the locomotives and wagons which constructs the fatigue load profiles are shown in Figure 3.6

below. Because of the changes in traffic mass over time, the traffic history is divided into five periods. There had been different types of passenger or freight trains in the same time periods. As a result of this convergence, the traffic history data for the produced fatigue load profiles, which is given in Table 3.2, is obtained by multiplying the traffic history data for train types, which is given in Table 3.3, with fractional coefficients

Table 3.1 : Fatigue load profiles.

Load Profiles	Load Profile
Type A	DE24000-3A-2B
Type B	DE21500-3Fas-10K-3Sg-3Rs
Type C	DE24000-3Fas-10K-3Sg-3Rs
Type D	E43000-4WSPm-2B
Type E	DE24000-2Fas-4Fal-3Sg
Type F	E43000-2Fas-4Fal-3Sg-5G
Type G	BL-5Ea-2Rs
Type H	4E8000

Table 3.2 : Traffic history data dissipated over fatigue load profiles.

Load Profiles	1959-1970	1971-1984	1985-1990	1991-2004	2005
A	-	-	16	-	-
B	-	14 x 0.4	2 x 0.4	-	-
C	-	14 x 0.6	2 x 0.6	-	-
D	-	-	-	15	10
E	-	-	-	17 x 0.3	11 x 0.3
F	-	-	-	17 x 0.7	11 x 0.7
G	16	-	-	-	-
H	72	72	108	108	114

Table 3.3 : Traffic history data.

	1959-1970	1971-1984	1985-1990	1991-2004	2005
Passenger Train (A, D) Passage per day	-	-	16	15	10
Freight Train (B, C, E, F) passage per day	16	14	2	17	11
Suburban Train (H) passage per day	72	72	108	108	114

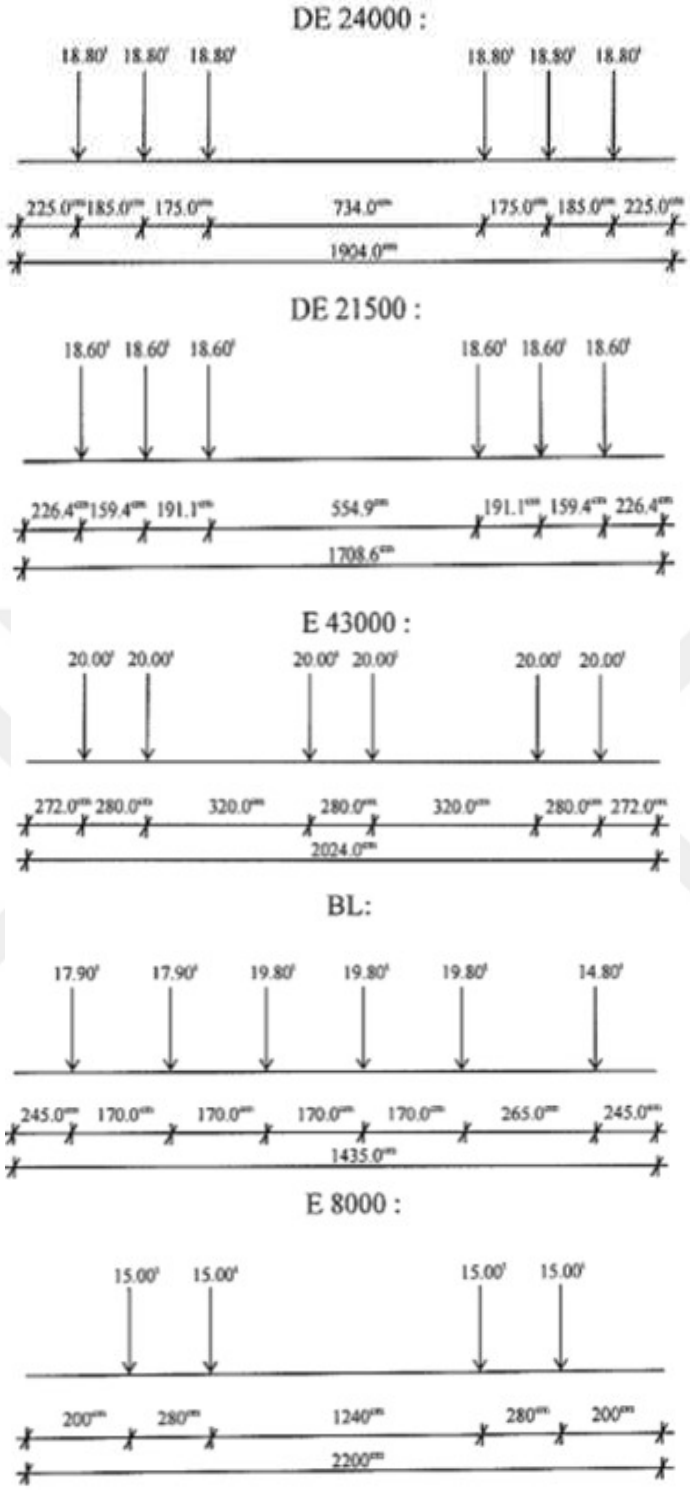


Figure 3.6: Locomotive and wagon axle loads and distances.

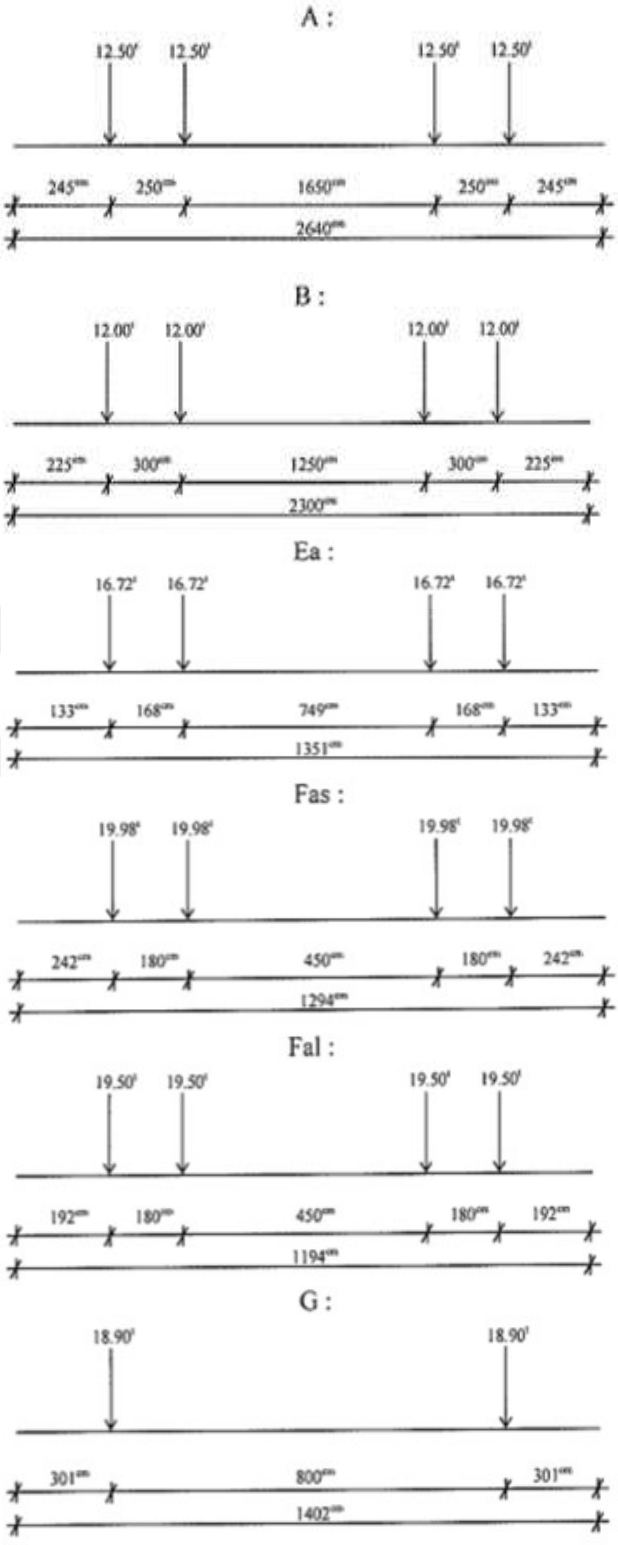


Figure 3.6 (Continued): Locomotive and wagon axle loads and distances.

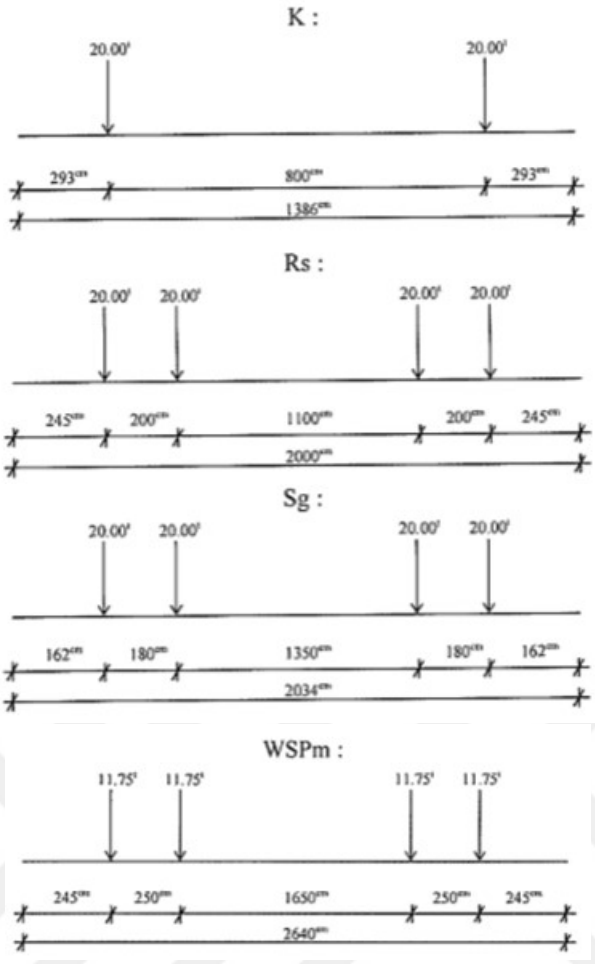


Figure 3.6 (Continued): Locomotive and wagon axle loads and distances.

4. ANALYSIS RESULTS AND CALCULATIONS

4.1 Application of the Loads to the Structural Model

The fatigue load profiles which is obtained in the previous chapter, should be moved over the structural bridge model. To be able to do this, step by step load cases are needed to be generated which simulates the passage of every fatigue load profile over the bridge from the reach of the first axle loads till the leave of the last axle loads. In every step, the load profile stands in different frame parts in different positions. Writing down the load profiles axle loads and distances by hand would most probably cause mistakes and a great loss of time. Furthermore, it would be very complicated to find in which frame element number and in which position the axle loads would stand in each step. To conduct the complex analysis and big data, Python open source coding program is used.

First of all, a Python code is created which takes the information file about the locomotive and wagon types and their combinations in each load profile and generates the load profiles defined by the axle loads and distances in text files. This code is given in the Appendix, Figure A.1. The step distance for the load profile to move on the bridge is chosen as 0,8 m which is the half of the free-span distance of the stringers, because the maximum and minimum stresses would occur when the load is in the middle of the span and when the load is near the supports. To be able to apply the moving load profiles over the bridge with a step distance of a half-span, another code of a Basic program QB64 is used which can be seen in Figure A.2. The output of the code gave us the load cases which shows the loads on the frame numbers and the positions over the frame in each step of the load profile passing through the bridge. The format of the output is turned from a text file to a .csv format file so as to be recognizable by SAP2000 interactive database editing tables format. By editing the load patterns, load cases and frame load assignments tables using the interactive database editing section, the load cases for each load profile are applied to the bridge model.

4.2 Analysis Options and Collecting the Results

It is very important that, the station spacing of the frames should be chosen to feed the need of the accuracy of the calculations. It specifies the number of sections the analysis results will be given. When the number of load cases are big as in the case of this study and most of the fatigue analysis loadings based on a big history data, it should be indicated very carefully, otherwise the analysis results would take too much time, even exceeding the Running Memory of the computers, and there would be very much data to be used. Figure 4.1 shows the menu where maximum station spacing is adjusted, which can be reached by a right click to the frame elements. The desired spacing could be written by hand, station at element intersections and station at concentrated loads could be chosen as Yes or No.

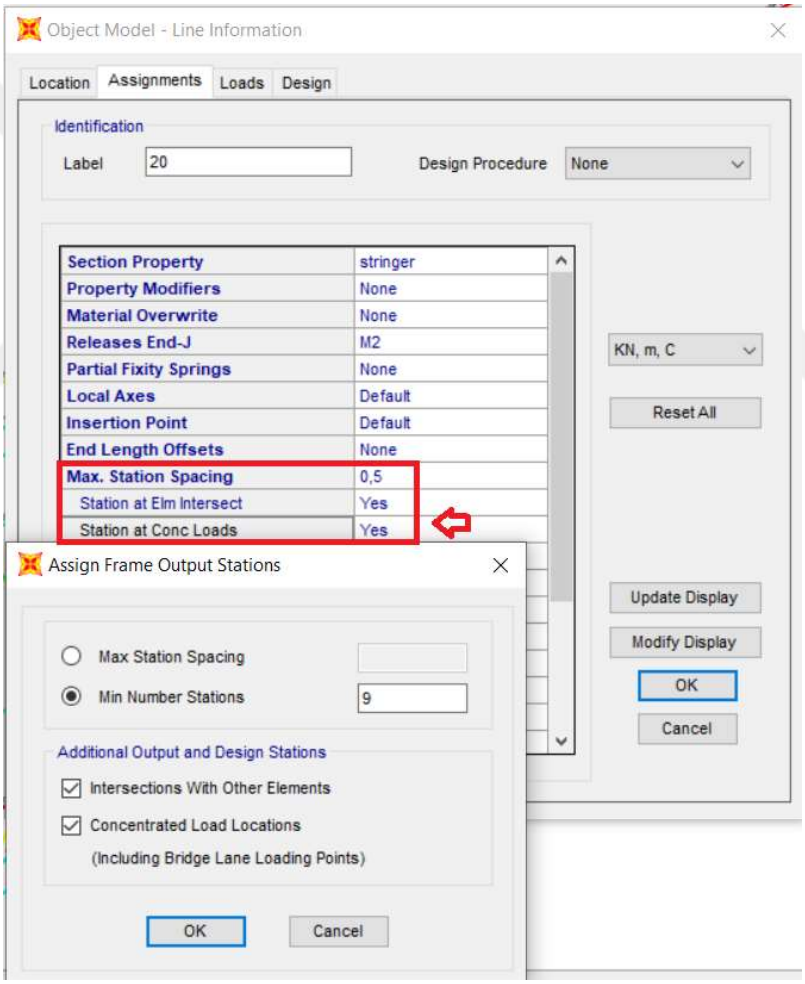


Figure 4.1: Maximum station spacing adjustment.

By double clicking to the marked section in Figure 4.1, the below menu will appear which also allows to give a minimum number of stations. Since there are variable the axle load distances

which are closed to each other, the load positions on the frames are not similar to each other. When the maximum station spacing menu is adjusted as seen in Figure 4.1, it is observed that the analysis results are given in almost every 5 cm for each element of the bridge. Some of the load profile results even exceeded the maximum line limit of Excel. In the beginning of the analysis, the concentrated load locations section could be unchecked and the minimum number of stations and maximum station spacing could be adjusted or an SQL code could be used to filter the data using the menu reached by following Display-Show Tables-Analysis Results-Format Filter Sort-Filter Table-Advanced. This menu is shown in Figure 4.2.

Figure 4.2: Advanced filter menu for the analysis results.

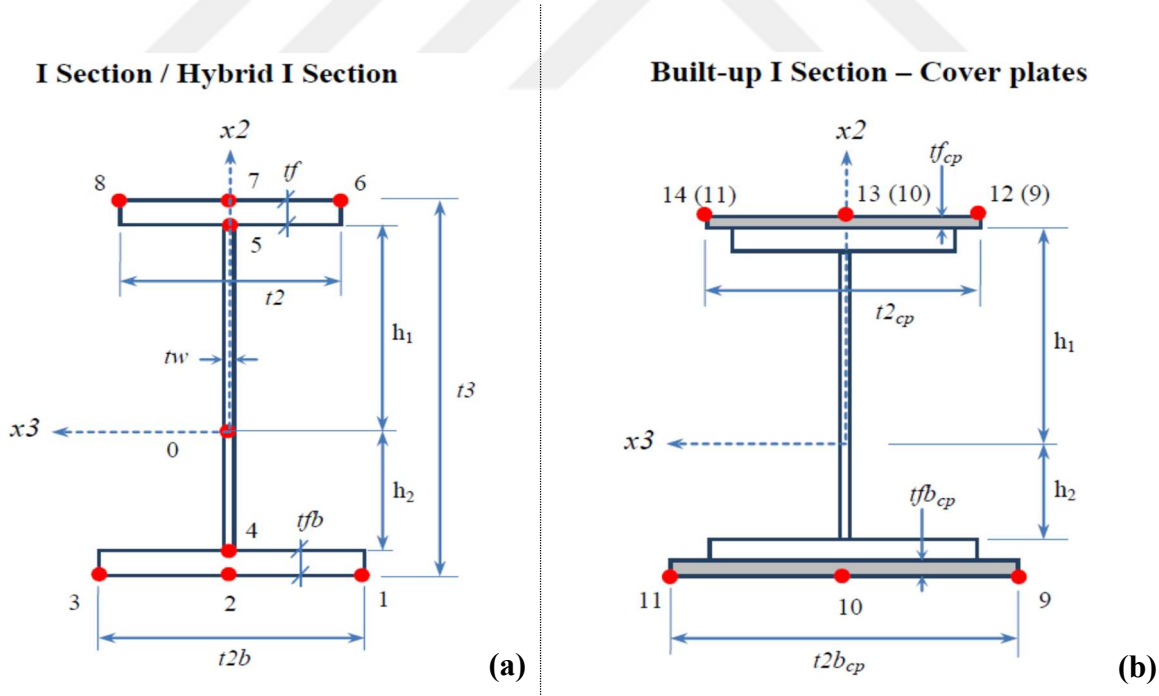


Figure 4.3: Stress points in SAP2000 for different sections. (a) I section. (b) I section with cover plates [22].

After running the analysis, the frame element stress output tables are examined, it is seen that there are six kinds of S values which are S11, S12, S13, Smax, Smin and SVM. Those stress types are defined in a manual of SAP2000 [22] and it states that S11 is axial stress, S12 and S13 are shear stresses, Smax and Smin are principal stresses and SVM is the von-Mises Stress. In fatigue analysis, axial stress is considered, accordingly S11 values from the results will be considered. The stress results of the program give the results in a range of point values. These stress points are shown in Figure 4.3. The points in parenthesis in Figure 4.3 (b) is valid when the cover plate is only in the top flange. For the sections which are defined using the section designer, the stress point could be picked by the user. For the main beam which is designed in the section designer, the default stress point was defined in the centroid and the corners of the bounding box of the section. In the light of this knowledge, the analysis results are filtered meanly in the Filter section of the program and then exported from SAP2000 to Excel.

4.3 Derivation of the Stress Graphs

Analysis results data is filtered in detail and examined using the Python codes given in Figure A.5. By the help of the code, stress graphs for any point and any position in an element is drawn and compared. The top and bottom point stress results for any element are approximately symmetrical in in horizontal axis as shown in an example in Figure 4.4. Because the axial forces are very small, and the stress is mostly governed by the moments which results in linear stress distribution where the neutral axis is in the centroid for the symmetrical sections. It is seen that in every step of a load case, the image on the stress graph seems to shift to right without a change in the number of stress ranges and small changes in the amplitude are also observed as can be seen in Figure 4.5. Therefore, it can be understood that the section and point with the maximum stress range would give the bigger fatigue damage accumulation since the number of the peaks and valleys is same throughout the element under the same load profile. The changes in the stress values were not very big in the sections with close positions. That is why, the data is filtered to show the station 0.0 m values for every frame. In the light of all the information gathered, the critical stress points and sections of the critical elements for fatigue analysis are chosen. The critical points are chosen as the top flange of frame 2 for stringer midspan for load profiles A and D and frame 14 for the other load profiles, bottom flange of frame 3 for stringer connections, top flange of frame 91 for the main beams and the bottom

flange of frame 199 for the cross beams at station 0.0 m for each frame. The maximum stress points in the top or bottom flanges are chosen by the code. For each critical element of the bridge, the stress-load case graphs of each fatigue load profile are plotted by the code given in Figure A.3. The stress graphs for stringer midspan, in each fatigue load profile is given in Figure 4.6. The stress graphs for main beam, cross beam and stringer connection elements are given in Figure A.8, Figure A.9 and Figure A.10 respectively.

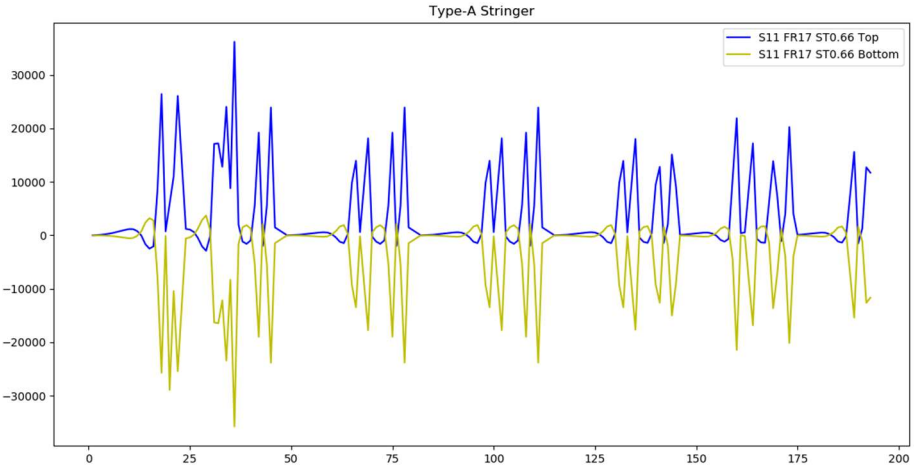


Figure 4.4: Stress graph comparison for top and bottom flanges of frame 17.

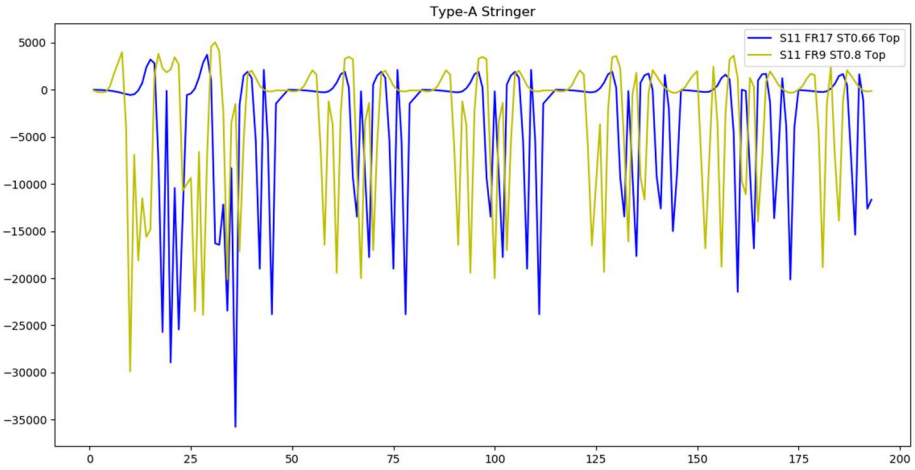


Figure 4.5: Stress graph comparison for the top flanges of the sections in station 0.66 and 0.8 m of frame 17.

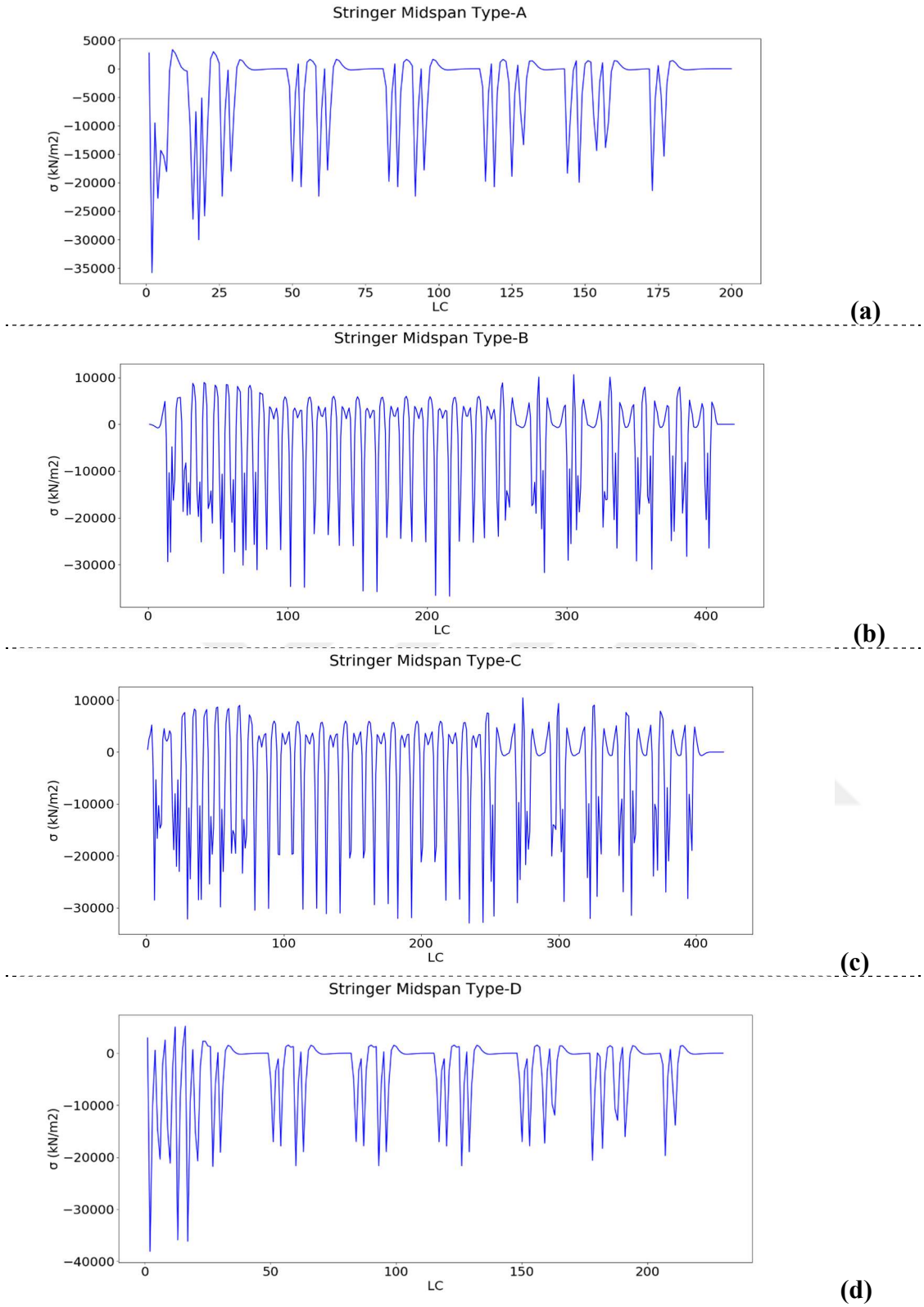
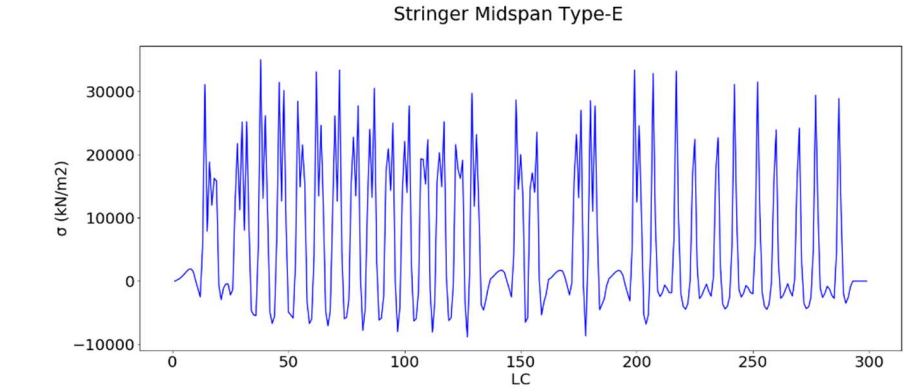
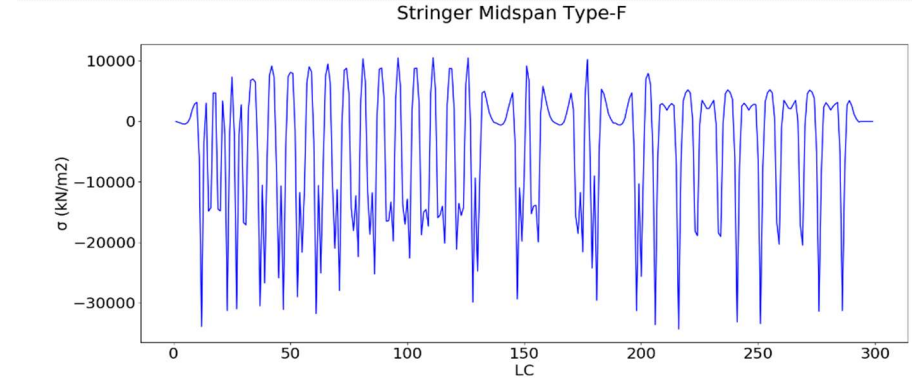


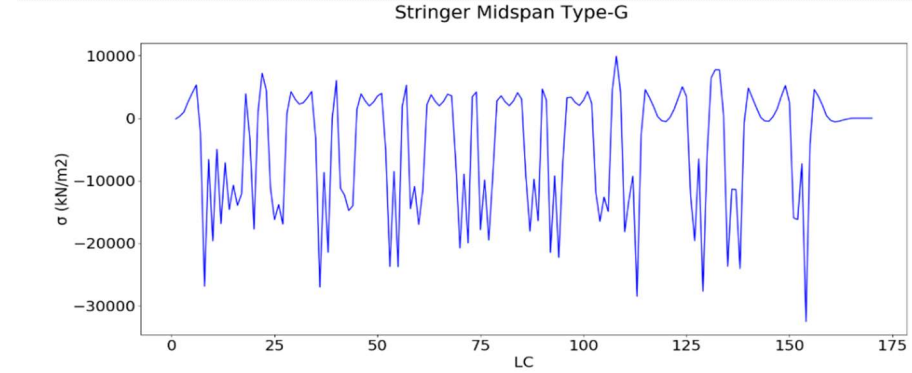
Figure 4.6: Stringer midspan stress graphs for: (a)Type A. (b) Type B. (c) Type C. (d) Type D.



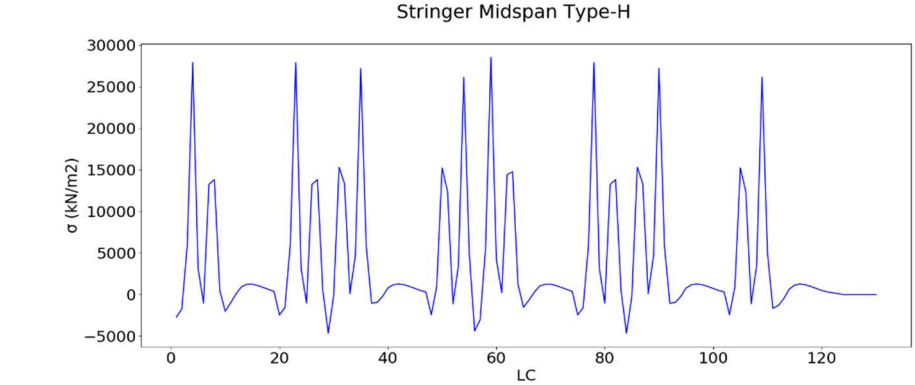
(e)



(f)



(g)



(h)

Figure 4.6 (Continued): Stringer midspan stress graphs for: (e)Type E. (f) Type F. (g) Type G. (h) Type H.

4.4 Application of the Rainflow Counting

In the next step, a rainflow cycle counting algorithm written using PyPI "rainflow 2.1.2" library [23] is used for cycle counting of the stress graphs, the code is given in Figure A.6. Then, the results are plotted to stress histograms, using the code given in Figure A.4. An example S-N histogram for stringer midspan for one pass of Load Type B is shown in Figure 4.7. An example histogram of S-N/year for between period 1971-1984 for stringer midspan for Load Type B is shown in Figure 4.8. An example S-N histogram for stringer midspan with all traffic data of Load Type B (without impact factor) is shown in Figure 4.9. Every single histogram which is obtained for one passage of any load type over any element is multiplied by the traffic history data by the help of the coding in Python and result histograms under the traffic history data is obtained for each critical bridge element as shown in Figure 4.10.

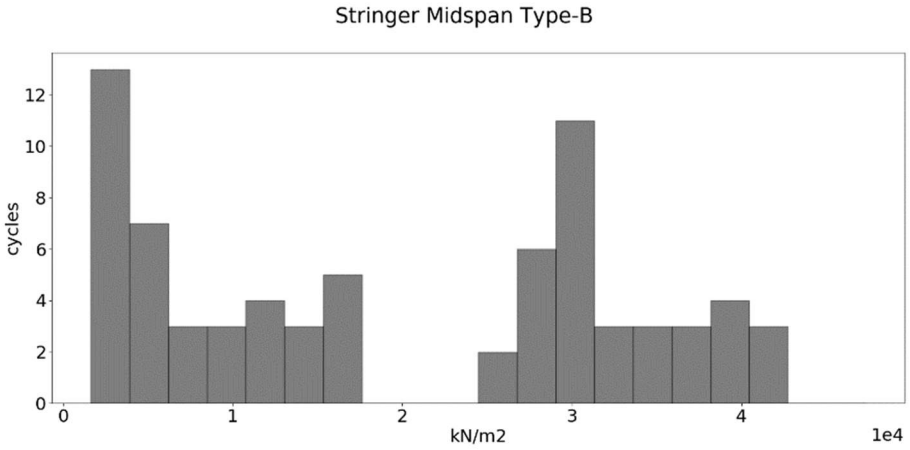


Figure 4.7: Histogram for stringer midspan for one pass of Load Type B.

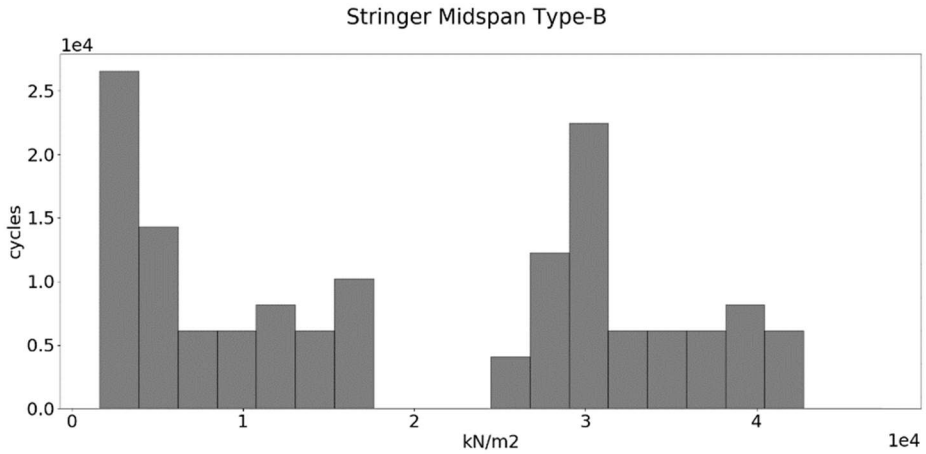


Figure 4.8: Histogram for stringer midspan for one year of Load Type B (1971-1984).

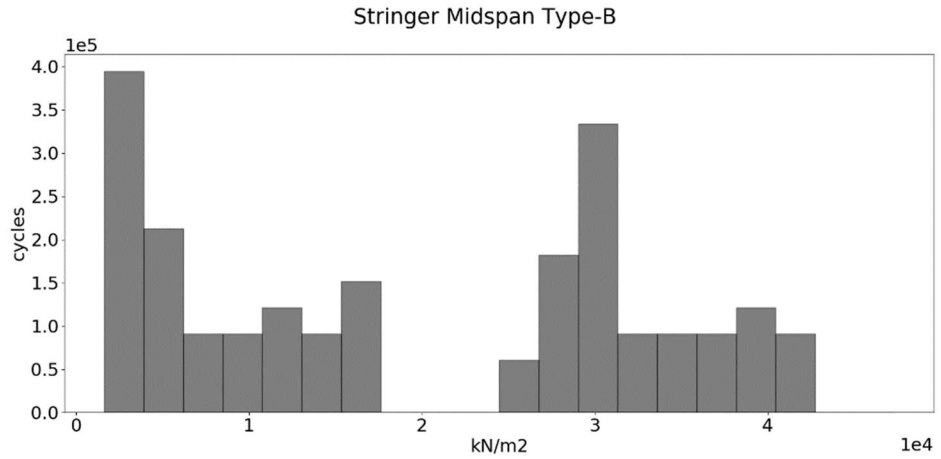


Figure 4.9: Histogram for stringer midspan with all traffic data of Load Type B.

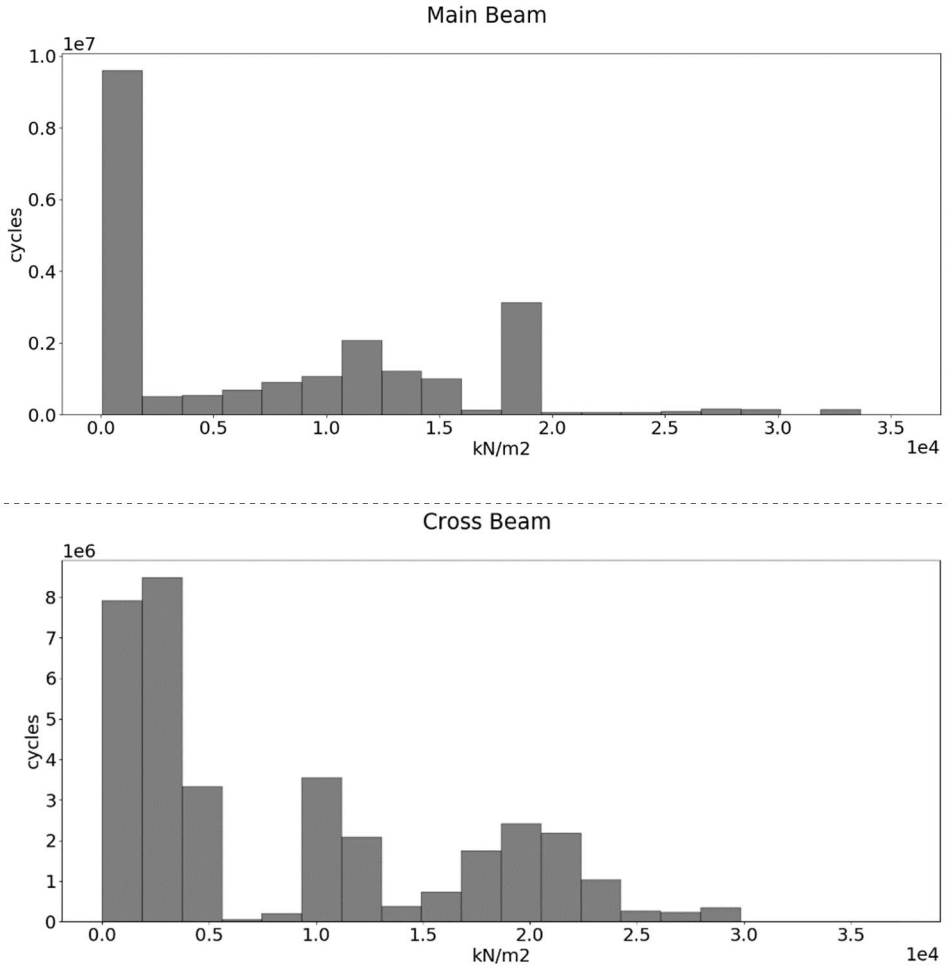
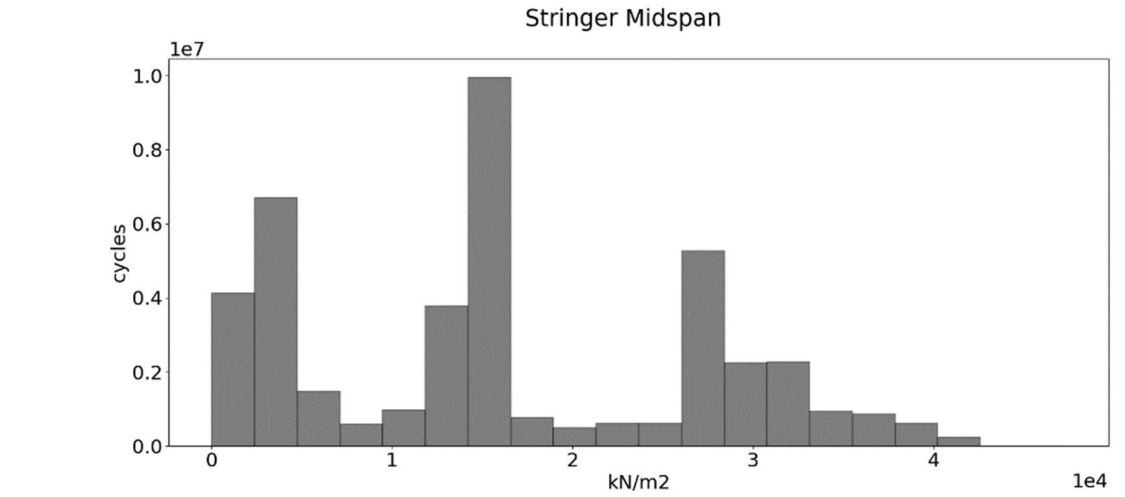
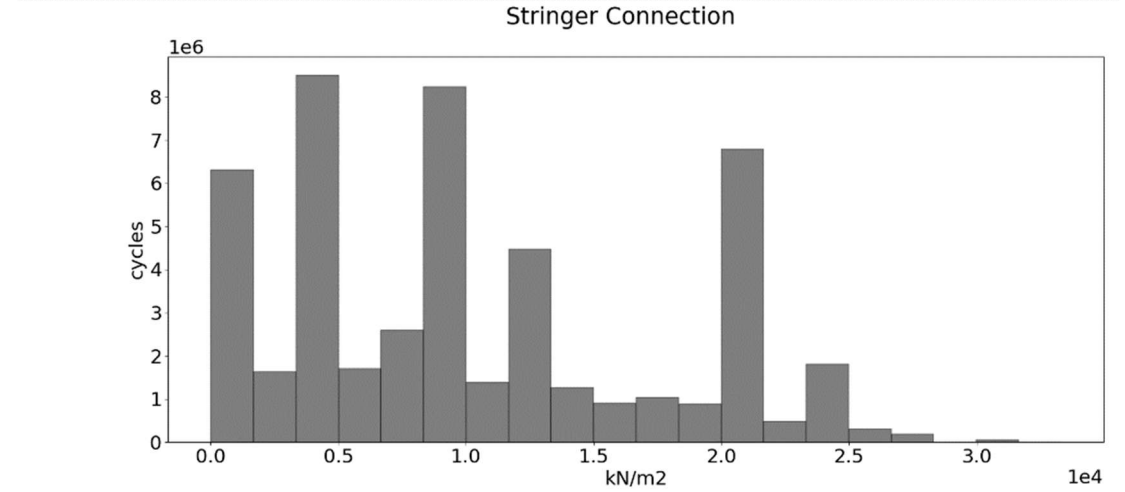


Figure 4.10: Result histograms for critical elements. (a) Main beam (b)Cross beam.



(c)



(d)

Figure 4.10 (Continued): Result histograms for critical elements. (c) Stringer midspan (d) Stringer connection.

4.5 Calculation of the Total Damage Accumulation

In the final step of the analysis, fatigue damage accumulations according to the detail categories in AASHTO [8], were calculated from these histograms using Miner's Rule and the total fatigue damage over the critical elements are obtained by the sum of the fatigue damage accumulations. The loads have been taken without any coefficient up to this point. In the previously conducted series of field test data given in the TU-Bridges research reports prepared during the NATO Science for Stability Program, the selected impact factors for each element have been given, as listed in Table 4.1. The stress ranges in the histogram data are multiplied by the impact factors.

The multiplied stress ranges are used to calculate the fatigue damage accumulation. As there are not enough field tests for the considered bridge and the riveted bridges in general, the total damage calculations are done for both of the detail categories which are AASHTO D (ECCS 71) and AASHTO C (ECCS 90) which are mostly used for riveted bridges. AASHTO detail category table, showing stress range versus number of cycles is shown in Figure 4.11.

Table 4.1 : Impact factors.

Element	ϕ
Main Beam	1.23
Cross Beam	1.28
Stringer (midspan)	1.3
Stringer (Connection)	1.3

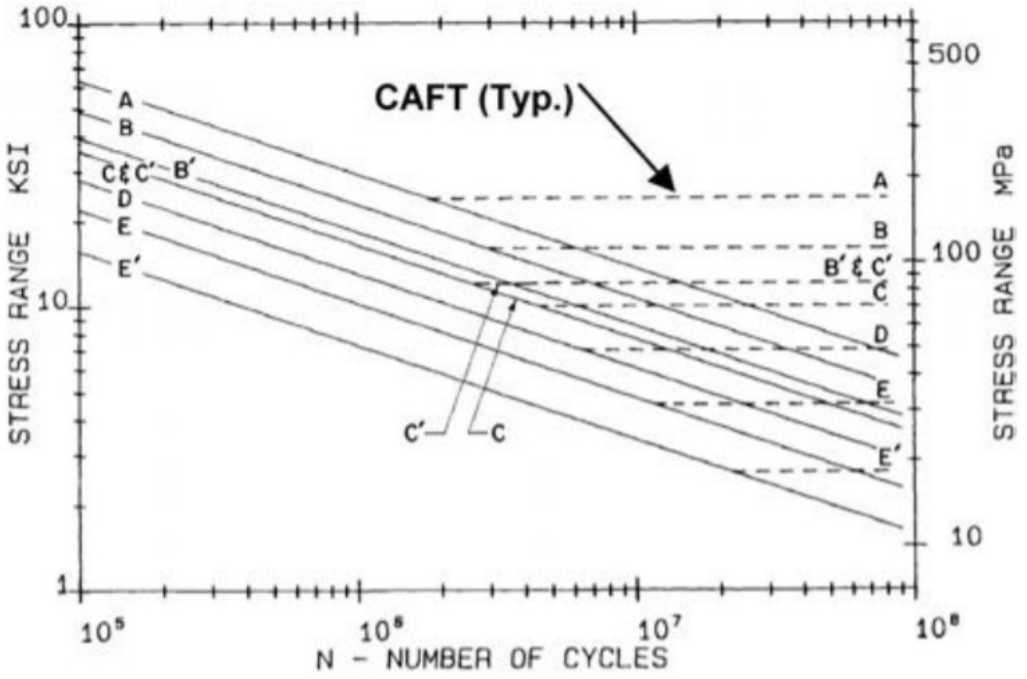


Figure 4.11: AASHTO S-N curve for detail categories [8].

The equation of the curves in the S-N graph is given in **(4.1).**

$$(\Delta\sigma)_n = \left(\frac{A}{N}\right)^{1/3} \quad \textbf{(4.1)}$$

$(\Delta\sigma)_n$ is the stress range

A is a coefficient for the detail categories

N is the number of cycles in $(\Delta\sigma)_n$ stress range

Here, $(\Delta\sigma)_n$ is the multiplied $\Delta\sigma$ values by the impact factors ϕ. The detail category constant A is given in AASHTO LRFD as $22x10^8$ ksi^3 for detail category D and $44x10^8$ ksi^3 for category C, the constant amplitude fatigue threshold for category D is 7 ksi and for category C, 10 ksi [8]. The number of cycles to failure, for the C and D detail categories corresponding to each multiplied stress range value in the histograms is calculated using **(4.1)**. The cycle number counted for the same stress range $(\Delta\sigma)_n$ is divided to the N values and the result values, d_i, shows the damage accumulation values for each stress range, as given in **(2.1)**. According to the S-N curve method, when the stress range $(\Delta\sigma)_n$ is below the constant amplitude fatigue threshold, the element is in the infinite life zone, therefore no damage accumulation occurs. The operation is done for all of the stress range values in the stress histogram and the sum of the damage accumulations, gives the total fatigue damage as indicated in **(2.2)**. Calculated total damages for each critical bridge member are shown below in the tables Table 4.2, Table 4.3, Table 4.4 and Table 4.5. The summary of the total damages is given in Table 4.7.

Table 4.2 : Total damage calculation for main girder.

		Category D		Category C	
(Φ=1.23)xStress range (kN/m^2)	Number of cycles	Number of cycle to failure	Damage	Number of cycle to failure	Damage
ΦxΔσ	**n**	**N**	**D**	**N**	**D**
…	…	-	-	-	-
34960	80300	-	-	-	-
39256	30368	-	-	-	-
40919	30368	-	-	-	-
40923	63620	-	-	-	-
41263	27266	-	-	-	-
		Total Damage (%)	**0.0**	Total Damage (%)	**0.0**

Table 4.3 : Total damage calculation for cross beam.

		Category D		Category C	
(Φ=1.28)xStress range (kN/m^2)	Number of cycles	Number of cycle to failure	Damage	Number of cycle to failure	Damage
ΦxΔσ	**n**	**N**	**D**	**N**	**D**
…	…	-	-	-	-
37088	70080	-	-	-	-
37327	70080	-	-	-	-
37402	70080	-	-	-	-
37405	70080	-	-	-	-
37453	70080	-	-	-	-
		Total Damage (%)	**0.0**	Total Damage (%)	**0.0**

Table 4.4 : Total damage calculation for stringer connection.

		Category D		Category C	
(Φ=1.3)xStress range (kN/m^2)	Number of cycles	Number of cycle to failure	Damage	Number of cycle to failure	Damage
ΦxΔσ	**n**	**N**	**D**	**N**	**D**
…	…	-	-	-	-
34813	27266	-	-	-	-
34813	45552	-	-	-	-
36058	63620	-	-	-	-
36305	63620	-	-	-	-
39297	63620	-	-	-	-
		Total Damage (%)	**0.0**	Total Damage (%)	**0.0**

Table 4.5 : Total damage calculation for stringer midspan.

		Category D		Category C	
(Φ=1.3)xStress range (kN/m^2)	Number of cycles	Number of cycle to failure	Damage	Number of cycle to failure	Damage
ΦxΔσ	**n**	**N**	**D**	**N**	**D**
…	…	-	-	-	-
48191	63620	-	-	-	-
48304	45552	6398375	0.007	-	-
48332	30368	6387225	0.005	-	-
48484	63620	6327478	0.010	-	-
49091	63620	6095769	0.010	-	-
49183	63620	6061373	0.010	-	-
49242	30368	6039664	0.005	-	-
49295	45552	6020160	0.008	-	-
49295	27266	6020160	0.005	-	-
49473	45552	5955430	0.008	-	-
49572	30368	5920082	0.005	-	-
49744	27266	5858892	0.005	-	-
49898	63620	5804572	0.011	-	-
49950	30368	5786422	0.005	-	-
50492	27266	5602289	0.005	-	-
50742	30368	5519764	0.006	-	-
50809	63620	5497883	0.012	-	-
50828	45552	5491792	0.008	-	-
51159	27266	5385874	0.005	-	-
51479	45552	5286017	0.009	-	-
51495	27266	5281212	0.005	-	-
51926	30368	5150867	0.006	-	-
52286	27266	5045234	0.005	-	-
52323	63620	5034414	0.013	-	-
52455	30368	4996362	0.006	-	-
53056	30368	4828729	0.006	-	-
53325	63620	4755822	0.013	-	-
54275	30368	4510602	0.007	-	-
54306	30368	4502786	0.007	-	-
		Total Damage (%)	**20.6**	Total Damage (%)	**0.0**

Table 4.6 : Total Fatigue Damages.

Element	**Total damage (%):**	
	Cat. D	Cat. C
Main Beam	0	0
Cross Beam	0	0
Stringer (midspan)	20.58	0
Stringer (Connection)	0	0

5. CONCLUSIONS AND RECOMMENDATIONS

A single span riveted railway bridge which is taken into consideration was built in 1900s and had a very busy traffic load since then. Furthermore, there had been some visible severe collision damages in some elements. In 2005, there had been a project in Istanbul Technical University Civil Engineering Faculty, to evaluate the condition of this bridge and decide whether the bridge should be strengthened of replaced. For this purpose, field and laboratory tests and analysis were conducted for this bridge. It was found that, it is not economical to strengthen the bridge because of the collision damages.

In this study, fatigue analysis is carried out using the design data and test results of the bridge. Stress-based approach is adopted by using the load history records and rainflow counting method is used for cycle counting. Based on the maintained cycle counting values, total fatigue damages on each critical member were calculated using linear damage accumulation method. The results show that, fatigue damage is only obtained in the stringer midspan for detail category D and the total damage value is %20.58. This means that, between 1959 and 2005, this element used nearly one fifth of its fatigue life which is quite satisfying. However, the results are more conservative than expected. The reasons of this is listed below:

- The design train load of the bridges which were built in that period was the standard S1950 train which is much heavier than the trains in service of the Turkish Railway Network. As a result of this, the stress values under the service loads are rather small.
- Another reason is the rigid connection definitions between the elements in the structural model. The connections in the bridge are more likely to behave as semi-rigid and semi-rigid connections would transfer less moment in the connection, which will result in bigger moments in the midspans when compared to the rigid connections. As the results show that the highest damage accumulation occurred in the stringer midspans. If the connections would be defined as semi-rigid, the total damage accumulation in the fatigue-critical elements would increase. For a more realistic model, the rigidities of the semi-rigid connections should be determined.

- The structural model is not refined according to the field test data; thus, the material is defined to have no loss in the cross-sections because of corrosion or material defects. The cross-sections of the elements in tension would decrease because of the rivet holes, however this is neglected in the analysis.

On the other hand, by using programming languages, the stress graphics and the selection of the fatigue-critical points have become more accurate.

Recommendations for further researches are listed below according the evaluation of the results are listed below:

- As can be seen from the results of the study, the impact factor has a direct and great affect in the total damage accumulation. For this reason, it is a must to indicate the impact factors clearly and trustfully.
- There are not adequate studies for the AASHTO C and D detail categories. There is a big difference between the total damage calculation results for each category which are suggested for riveted members. The tests should be verified and more accurate fatigue limit categories should be determined.

REFERENCES

[1] **Wright, J. R.** (1995). Aims & Scope of the Journal. Journal of Infrastructure Systems, 1.

[2] **Tong, X. C.** (2011). Advanced Materials for Thermal Management of Electronic Packaging (Vol. 30). Springer.

[3] **Bond, A., & Harris, A.** (2008). Decoding Eurocode 7.

[4] **Rigo, A.** (2011). Scantling Optimization of Ship Structures Considering Fatigue at the Early Design Stage. Advances in Marine Structures, 569–579.

[5] **Siemes, A. J.** (1982). Miner's Rule with Respect to Plain Concrete Variable Amplitude Tests. ACI, 75, 343-372.

[6] **SAP2000**, (2000). Structural Analysis Program, Computers and Structures Inc., Berkeley, California, USA.

[7], **QB64,** version.1.3, Open source BASIC Compiler Program.

[8] **American Association of State Highway and Transportation Officials**, (2017). Standard Specifications for Highway Bridges, 17th Edition.

[9] **Out, J.M.M., Fisher, J.M., Yen, B.T.,** (1984). "Fatigue Strength of Weathered and Deteriorated Riveted Members", Fritz Engineering Laboratory Report 483-3C84.

[10] **Baker, K.A., Kulak, G.L.,** (1985). "Fatigue of Riveted Connections", Canadian J. Civil Engineering, Vol. 12.

[11] **Reemsnyder, H.S.,** (1975). "Fatigue Life Extension of Riveted Connections", ASCE Journal of Structures Division No.101, pp.2591.

[12] **Brühwiler, E., Smith, I.F.C. and Hirt, M.A.,** (1988). "Fatigue and Fracture of Riveted Bridge Members", Journal of Structural Engineering, Vol. 116 No 1.

[13] **Uzgider, E., Çağlayan, B. Ö., & Kaya, H**. (2005), "Gerçek Boyutlu Çelik Köprü Elemanlarının Yorulma Testi Ve Artık Yorulma Ömrü Tespiti"., TMH (Türkiye Mühendislik Haberleri), Vol. 436.

[14] **I. Rychlik**, (1987). "A new definition of the rainflow cycle counting method," Internationally Journal of Fatigue, vol. 9, pp. 119–121.

[15] **American Society for Testing and Materials,** (1986). Annual Book of ASTM Standards, Section 3, Vol.03.01, Philadelphia.

[16] **Downing, S.D., and Socie, D.F.,** (1982). "Simple rainflow counting algorithms", International journal of fatigue, 4(1), 31-40.

[17] **Rychlik, I.** (2013). Fatigue Cycle Counting. Encyclopedia of Tribology, 1032-1041.

[18] **Miner, M.A.,** (1945). "Cumulative Damage in Fatigue", J. Appl. Mech., Vol.12, Trans. ASME, Vol.67, pp. A- 159-A164.

[19] **Barsom, J.M. and Rolfe, S.T.,** (1987). Fracture and Fatigue Control in Structures, Prentice Hall Inc., Englewood cliffs, New Jersey.

[20] **TCDD (Turkish General Directorate of Railways),** (2005). TCDD Genel Müdürlüğü İstatistik Yıllığı**,** Ankara.

[21] **TCDD (Turkish General Directorate of Railways),** (2005). TCDD Yük ve Yolcu Vagonları.

[22] **Url-1** < *http://docs.csiamerica.com/manuals/sap2000/Technical%20Notes/S-TN-FRA-001.pdf* >, date retrieved 30.08.2019.

[23] **PyPi. Rainflow Library**, version 2.1.2. Retrieved August 13, 2019, from https://pypi.org/project/rainflow/.

APPENDICES

APPENDIX A: Analysis codes and results

APPENDIX A

```
working_dir = 'data'
f = open(working_dir + '/wagon_models.txt', "r")
lines = f.readlines()

wagon_dists = {}
wagon_loads = {}
trains = {}

for line in lines:
    print("Parsing line: " + line)
    words = line.split('\t')
    if len(words) == 3:
        if(words[1] == 'DIST'):
            dists = list(map(float, words[2].split('-')))
            wagon_dists[words[0]] = dists
        elif(words[1] == 'LOAD'):
            loads = list(map(float, words[2].split('-')))
            wagon_loads[words[0]] = loads
        else:
            print("Error at the line : unknown keyword other than DIST or
LOAD")
            break
    else:
        print("Error at the line: no 3 items")
        break
f.close()

g = open(working_dir + '/loads.txt', "r")
lines = g.readlines()

for line in lines:
    print("Parsing line: " + line)
    words = line.split('\t')
    if len(words) == 2:
        type_words = words[1].split()
        if len(type_words) == 2:
            if(type_words[0] == 'Type'):
                trains[type_words[1]] = list(words[0].split('-'))
            else:
                print("Error at the line: no Type keyword")
                break
        else:
            print("Error at the line: no 2 items")
            break
    else:
        print("Error at the line: no 2 items")
        break

g.close()
```

Figure A.1 : Python code for load profiles.

```
coment_keyword = "REM"
load_keyword = "LOADF"

for train_type in trains:
    load_number = 1
    coordinate = 0
    print("producing output " + train_type + ".txt file for Type " +
train_type)
    h = open(working_dir + '/' + train_type +'.txt', "w")
    for wagon_with_count in trains[train_type]:
        print("adding wagon(s) " + wagon_with_count)

        intersect_point = 0
        wagon_count = 1
        wagon_name = wagon_with_count
        for i in range(len(wagon_with_count)):
            if(1 == wagon_with_count[i].isdigit()):
                intersect_point = i+1
            else:
                break
        if( 0 != intersect_point):
            wagon_count = int (wagon_with_count[:intersect_point])
            wagon_name = wagon_with_count[intersect_point:]

        for i in range(wagon_count):
            print("adding wagon " + wagon_name)
            if(wagon_count == 1):
                h.write('\r%s - %s\r' % (coment_keyword, wagon_name))
            else:
                h.write('\r%s - %d%s\r' % (coment_keyword, i+1,
wagon_name))

            loads = wagon_loads[wagon_name]
            coordinates = wagon_dists[wagon_name]

            if(load_number != 1):
                coordinate += coordinates[0]

            for i in range(len(loads)):
                load = -1 * loads[i]/2
                h.write('%s   %d   %.1f  %.2f   %.2f\r' % (load_keyword,
load_number, coordinate, load, load))
                load_number += 1
                coordinate += coordinates[i+1]
    h.close()
```

Figure A.1 (Continued) : Python code for load profiles.

```
DECLARE SUB Calc (Ykoor!, ElemanBoy!(), ElemNo!, oran!, Adet)
DECLARE SUB ComLine (Cl$, Args$(), NumArgs!, MaxArgs!)
DECLARE SUB Logo ()
DECLARE SUB FileNames (CommandLine$, DataFileName$, OutFileName$)
DECLARE SUB Control (DataFileName$, OutFileName$, true!)
DECLARE SUB GetInOut (DataFileName$, OutFileName$)

DIM SHARED Args$(100)
CLS
Logo
FileNames COMMAND$, DataFileName$, OutFileName$
GetInOut DataFileName$, OutFileName$
PRINT
PRINT "islem tamam !"
END

SUB Calc (Ykoor, ElemanBoy(), ElemNo, oran, Adet)
    oran = -2: TopYer = 0: say = 0: ElemNo = 0
    IF Ykoor < 0 THEN ElemNo = 0: EXIT SUB
    DO
        say = say + 1
        IF ElemanBoy(say) = 0 AND say <= Adet THEN
            PRINT ElemanBoy(say - 1), say
            PRINT "hatali data girisi bulundu"
            PRINT " eleman boyu sifir (0) verilmis"
            PRINT " programa son verildi, datayi duzeltip tekrar
calistiriniz"
            END
        END IF
        IF say > Adet THEN oran = -1: EXIT SUB
        TopYer = TopYer + ElemanBoy(say)
        IF TopYer >= Ykoor THEN
            ElemNo = say
            A = TopYer - Ykoor
            oran = 1 - (A / ElemanBoy(say))
            oran = INT(oran * 10000) / 10000
            EXIT DO
        END IF
    LOOP
END SUB

SUB ComLine (Cl$, Args$(), NumArgs, MaxArgs) STATIC
    CONST true = -1, false = 0
    ERASE Args$
    NumArgs = 0: in = false
    l = LEN(Cl$)
    'Go through the command line a character at a time.
    FOR i = 1 TO l
        C$ = MID$(Cl$, i, 1)
        'Test for character being a blank or a tab.
        IF (C$ <> " " AND C$ <> CHR$(9) AND C$ <> ",") THEN
```

Figure A.2 : QB64 code for load cases.

```
            'Neither blank nor tab. Test if you're already inside an
argument.
            IF NOT in THEN
                'You've found the start of a new argument.
                'Test for too many arguments.
                IF NumArgs = MaxArgs THEN EXIT FOR
                NumArgs = NumArgs + 1
                in = true
            END IF
            'Add the character to the current argument.
            Args$(NumArgs) = Args$(NumArgs) + C$
        ELSE
            'Found a blank or a tab.
            'Set "Not in an argument" flag to FALSE.
            in = false
        END IF
    NEXT i

END SUB

SUB FileNames (CommandLine$, DataFileName$, OutFileName$)
    true = 1
    ComLine CommandLine$, Args$(), NumArgs, 2
    IF NumArgs = 2 THEN
        DataFileName$ = Args$(1)
        OutFileName$ = Args$(2)
        PRINT "Data  Dosyasi Adi =  "; Args$(1)
        PRINT "Cikis Dosyasi Adi =  "; Args$(2)
    ELSEIF NumArgs = 1 THEN
        DataFileName$ = Args$(1)
        PRINT "Data  Dosyasi Adi =  "; Args$(1)
        INPUT "€ikis Dosyasi Adi =  ", OutFileName$
    ELSEIF NumArgs = 0 THEN
        INPUT "Data  Dosyasi Adi =  ", DataFileName$
        INPUT "€ikis Dosyasi Adi =  ", OutFileName$
    END IF
    IF DataFileName$ = "" OR OutFileName$ = "" THEN
        PRINT: PRINT
        PRINT "Dosya isimleri dogru girilmedi"
        PRINT "program durduruldu !"
        PRINT "dosya isimlerini dogru girip yeniden calistirabilirsiniz !"
        END
    END IF
END SUB

SUB GetInOut (DataFileName$, OutFileName$)
    DIM Elem1No(500), Elem2No(500), Elem1Boy(500), Elem2Boy(500),
LoadDingil1(200)
    DIM PosDingil(200), LoadDingil2(200), LoadType(100), LengthDingil(200)
    DIM LoadDingily(200)
    DIM Ext1(500): DIM Ext2(500)
    IMPL = 0
```

Figure A.2 (Continued) : QB64 code for load cases.

```
    REM tekil yuk
    REM Sirala1$ = "&  ##### , &  #.##### ,  #####.##### ,   ##### , 1"
    Sirala1$ = "&;&;&;&;&;&;#.#####;#.#####;#####.#####"
    p = 0: l = 0: durum = 0
    limit1 = 0: limit2 = 0
    OPEN DataFileName$ FOR INPUT AS #1

    WHILE NOT EOF(1)
        LINE INPUT #1, A$
        PRINT A$
        ComLine A$, Args$(), Num, 30
        SELECT CASE UCASE$(Args$(1))
            CASE "OTHER"
                IMPL = 1
                pp = 0
                REDIM DummyNo(500), DummyElBoy(500)
                FOR i = p TO 1 STEP -1
                    pp = pp + 1
                    DummyNo(pp) = Elem1No(i)
                    DummyElBoy(i) = Elem1Boy(i)
                NEXT
                FOR i = 1 TO p
                    Elem1No(i) = DummyNo(i)
                    Elem1Boy(i) = DummyElBoy(i)
                NEXT
                pp = 0
                ERASE DummyNo, DummyElBoy
                REDIM DummyNo(500), DummyElBoy(500)
                FOR i = l TO 1 STEP -1
                    pp = pp + 1
                    DummyNo(pp) = Elem2No(i)
                    DummyElBoy(pp) = Elem2Boy(i)
                NEXT
                FOR i = 1 TO l
                    Elem2No(i) = DummyNo(i)
                    Elem2Boy(i) = DummyElBoy(i)
                NEXT

            CASE "LC"
                LcNum = VAL(Args$(2))
            CASE "POINT"
                IYK = VAL(Args$(2))
            CASE "STEP"
                RunStep = VAL(Args$(2))
            CASE "LANE1"
                N1 = VAL(Args$(2))
                N1s = VAL(Args$(3))
                Bl1 = VAL(Args$(4))
                FOR i = N1 TO N1s
                    p = p + 1
                    Elem1No(p) = i
                    Elem1Boy(p) = Bl1
```

Figure A.2 (Continued) : QB64 code for load cases.

```
                NEXT
            CASE "LANE2"
                N2 = VAL(Args$(2))
                N2s = VAL(Args$(3))
                Bl2 = VAL(Args$(4))
                FOR i = N2 TO N2s
                    l = l + 1
                    Elem2No(l) = i
                    Elem2Boy(l) = Bl2
                NEXT

            CASE IS = "LOADF", "F"
                Yer = VAL(Args$(2))
                LoadType(Yer) = 1
                PosDingil(Yer) = VAL(Args$(3))
                LoadDingil1(Yer) = VAL(Args$(4))
                LoadDingil2(Yer) = VAL(Args$(4))
                IF Num = 6 THEN
                    LoadDingil2(Yer) = VAL(Args$(5))
                    LoadDingily(Yer) = VAL(Args$(6))
                END IF
        END SELECT
    WEND
    CLOSE #1
    OPEN OutFileName$ FOR OUTPUT AS #1

    DataFileNamePrefix$ = LEFT$(DataFileName$, INSTR(DataFileName$, ".") -
1)
    FOR i = 1 TO LcNum
        REM PRINT #1, "ACTIVE, LC, "; i
        FOR j = 1 TO Yer
            Ykoor = IYK + (i - 1) * RunStep - PosDingil(j)
            IF i = LcNum AND j = 1 THEN PRINT Ykoor; " cm. ilk elemanin
yeri"
            IF Ykoor < 0 THEN EXIT FOR
            Calc Ykoor, Elem1Boy(), El1, Oran1, p
            Calc Ykoor, Elem2Boy(), El2, Oran2, l
            IF LoadType(j) = 1 THEN
                IF IMPL = 1 THEN
                    Oran1 = 1 - Oran1
                    Oran2 = 1 - Oran2
                END IF
                IF Oran1 >= 0 AND Elem1No(El1) > 0 THEN
                    REM PRINT #1, USING Sirala1$; "FB ,"; Elem1No(El1);
"FY,"; Oran1; LoadDingil1(j); Elem1No(El1)
                    REM PRINT #1, USING Sirala1$; "FB ,"; Elem1No(El1);
"FZ,"; Oran1; LoadDingily(j); Elem1No(El1)
                    PRINT #1, USING Sirala1$; LTRIM$(STR$(Elem1No(El1)));
DataFileNamePrefix$ + LTRIM$(STR$(i)); "GLOBAL"; "Force"; "Gravity";
"RelDist"; Oran1; RunStep / 100; LoadDingil1(j) * -10
                END IF
```

Figure A.2 (Continued) : QB64 code for load cases.

```
                IF Oran2 >= 0 AND Elem2No(El2) > 0 THEN
                    REM PRINT #1, USING Sirala1$; "FB ,"; Elem2No(El2);
"FY,"; Oran2; LoadDingil2(j); Elem2No(El2)
                    REM PRINT #1, USING Sirala1$; "FB ,"; Elem2No(El2);
"FZ,"; Oran2; LoadDingily(j); Elem2No(El2)
                    PRINT #1, USING Sirala1$; LTRIM$(STR$(Elem2No(El2)));
DataFileNamePrefix$ + LTRIM$(STR$(i)); "GLOBAL"; "Force"; "Gravity";
"RelDist"; Oran2; RunStep / 100; LoadDingil2(j) * -10
                END IF
            END IF
        NEXT j
    NEXT i
    CLOSE
END SUB
```

Figure A.2 (Continued) : QB64 code for load cases.

```
train_type = "B"
element = "Stringer Beam"
fiber = "TopFiber"
stations_filter = [0.0]
frame = 18
df = model.load_data(train_type)
points = model.get_points(train_type)
points_filter = points[element][fiber]
svalues, load_cases = ps.filter_params(df, frame, points_filter,
stations_filter)
print(svalues.shape)
print(load_cases.shape)
print(load_cases)
points_filter = points[element][fiber]
frame = 9
svalues2, load_cases2 = ps.filter_params(df, frame, points_filter,
stations_filter)
title = element + " Type-"+train_type
ps.plot_data(title, load_cases, svalues, None, None)
ps.plot_data(title, load_cases, svalues, load_cases2, svalues2)
```

Figure A.3 : Python code for stress graphs.

```
import histogram as hist
#hist.plot_train_pass_by_position("A", 17, [0,12,13,14], [0.0],
num_of_bins = 20, flatten="No")
#hist.plot_all_train_pass_by_position(17, [12,14], [0.0], num_of_bins =
20, flatten="No")
#hist.plot_train_pass("B", "Stringer Midspan", [0.0], num_of_bins = 20,
flatten="Yes")
#hist.plot_all_train_pass_on_element("Stringer Midspan", [0.0],
num_of_bins = 20, flatten="Yes")
#hist.plot_all_train_pass_on_element("Cross Beam", [0.0], num_of_bins =
20, flatten="Yes")
hist.plot_all_train_pass([0.0], num_of_bins = 20, flatten="Yes")
```

Figure A.4 : Python code for histograms.

```
import numpy as np
import plot
import model

stations_filter = [0.0]
for train_type in model.trains:
    print("Train ", train_type)

    df = model.load_data(train_type)
    points = model.get_points(train_type)

    for element in model.bridge_elements:
        #print(element)
        max_delta = 0
        max_fiber = ""
        max_frame = -1
        for frame in model.bridge_elements[element]:
            for fiber in points[element]:
                points_filter = points[element][fiber]
                svalues, load_cases = model.filter_params(df, frame,
points_filter, stations_filter)
                maxval = np.amax(svalues)
                minval = np.amin(svalues)
                std = np.std(svalues)
                diff = maxval - minval
                if max_delta < diff:
                    max_delta = diff
                    max_fiber = fiber
                    max_frame = frame
        print(element + ": Frame", max_frame, max_fiber,
              " MaxDelta ", "{0:.2f}".format(round(max_delta,2)))

        points_filter = points[element][max_fiber]
        svalues, load_cases = model.filter_params(df, max_frame,
points_filter, stations_filter)
        title = element + " Type-"+train_type#+" " + 'Fr'+ str(max_frame)
        plot.plot_data(title, load_cases, svalues, None, None)
```

Figure A.5 : Python code for selecting critical section.

```
"""
Implements rainflow cycle counting algorythm for fatigue analysis
according to section 5.4.4 in ASTM E1049-85 (2011).
"""
__version__ = "2.1.2"

from collections import deque, defaultdict
import functools

def _get_round_function(ndigits=None):
    if ndigits is None:
        def func(x):
            return x
    else:
        def func(x):
            return round(x, ndigits)
    return func

def reversals(series, left=False, right=False):
    """Iterate reversal points in the series.

    A reversal point is a point in the series at which the first
derivative
    changes sign. Reversal is undefined at the first (last) point because
the
    derivative before (after) this point is undefined. The first and the
last
    points may be treated as reversals by setting the optional parameters
    `left` and `right` to True.

    Parameters
    ----------
    series : iterable sequence of numbers
    left: bool, optional
        If True, yield the first point in the series (treat it as a
reversal).
    right: bool, optional
        If True, yield the last point in the series (treat it as a
reversal).

    Yields
    ------
    float
        Reversal points.
    """
    series = iter(series)

    x_last, x = next(series), next(series)
    d_last = (x - x_last)

    if left:
        yield x_last
```

Figure A.6 : Python code for rainflow counting.

```
    for x_next in series:
        if x_next == x:
            continue
        d_next = x_next - x
        if d_last * d_next < 0:
            yield x
        x_last, x = x, x_next
        d_last = d_next
    if right:
        yield x_next

def _sort_lows_and_highs(func):
    "Decorator for extract_cycles"
    @functools.wraps(func)
    def wrapper(*args, **kwargs):
        for low, high, mult in func(*args, **kwargs):
            if low < high:
                yield low, high, mult
            else:
                yield high, low, mult
    return wrapper

@_sort_lows_and_highs
def extract_cycles(series, left=False, right=False):
    """Iterate cycles in the series.

    Parameters
    ----------
    series : iterable sequence of numbers
    left: bool, optional
        If True, treat the first point in the series as a reversal.
    right: bool, optional
        If True, treat the last point in the series as a reversal.

    Yields
    ------
    cycle : tuple
        Each tuple contains three floats (low, high, mult), where low and
high
        define cycle amplitude and mult equals to 1.0 for full cycles and
0.5
        for half cycles.
    """
    points = deque()

    for x in reversals(series, left=left, right=right):
        points.append(x)
        while len(points) >= 3:
            X = abs(points[-2] - points[-1])
            Y = abs(points[-3] - points[-2])
            if X < Y:
                break
```

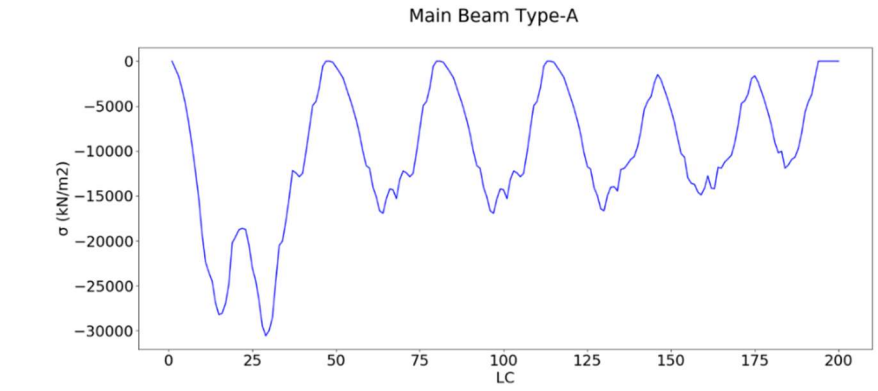
Figure A.6 (Continued) : Python code for rainflow counting.

```
            elif len(points) == 3:
                yield points[0], points[1], 0.5
                points.popleft()
            else:
                yield points[-3], points[-2], 1.0
                last = points.pop()
                points.pop()
                points.pop()
                points.append(last)
    else:
        while len(points) > 1:
            yield points[0], points[1], 0.5
            points.popleft()

def count_cycles(series, ndigits=None, left=False, right=False):
    """Count cycles in the series.

    Parameters
    ----------
    series : iterable sequence of numbers
    ndigits : int, optional
        Round cycle magnitudes to the given number of digits before
counting.
    left: bool, optional
        If True, treat the first point in the series as a reversal.
    right: bool, optional
        If True, treat the last point in the series as a reversal.

    Returns
    -------
    A sorted list containing pairs of cycle magnitude and count.
    One-half cycles are counted as 0.5, so the returned counts may not be
    whole numbers.
    """
    counts = defaultdict(float)
    round_ = _get_round_function(ndigits)

    for low, high, mult in extract_cycles(series, left=left, right=right):
        delta = round_(abs(high - low))
        counts[delta] += mult
    return sorted(counts.items())
```

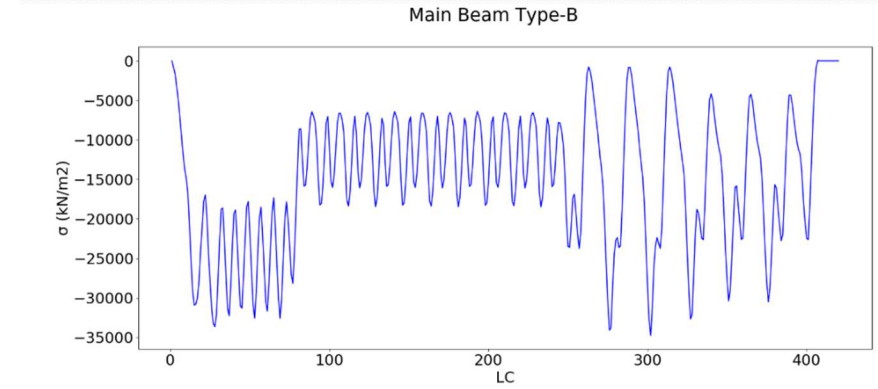
Figure A.6 (Continued) : Python code for rainflow counting.

```
import histogram as hist
import numpy as np
import sncurve
import pandas as pd
def make_damage_table(title, delta_sigma_fi, cycle_data,
                      fatigue_life_cycle_data=None, damage=None):
    df = pd.DataFrame(columns=['ΦΔσ', 'n', 'N', 'D'])
    df['ΦΔσ'] = delta_sigma_fi
    df['n'] = cycle_data
    if fatigue_life_cycle_data is not None:
        df['N'] = fatigue_life_cycle_data
    if damage is not None:
        df['D'] = damage
    df = df.sort_values(by=['ΦΔσ'])
    print(df)
    df.to_excel("./out/"+title+".xlsx")

fi_table = {
     "Stringer Midspan" : 1.3,
     "Stringer Connection" :1.3,
     "Main Beam" : 1.23,
     "Cross Beam" : 1.28
}

for category in sncurve.fatigue_category:
    limit = sncurve.get_fatigue_limit(category)
    print("Calculating Total Damages of each element for
category",category)
    for fi_element in fi_table:
        fi = fi_table[fi_element]
        print(fi_element, "Φ=", fi)
        delta_sigma, cycle_data =
hist.get_all_train_pass_on_element(fi_element, [0.0], flatten="Yes")
        delta_sigma_fi = delta_sigma * fi
        make_damage_table("Unfiltered Damage-"+category+"-"+fi_element,
                          delta_sigma_fi,
                          cycle_data)
        boolArr = delta_sigma_fi >= limit
        filtered_delta_sigma_fi = delta_sigma_fi[boolArr]
        filtered_cycle_data = cycle_data[boolArr]
        total_damage = 0
        if filtered_delta_sigma_fi.size != 0:
            fatigue_life_cycle_data =
sncurve.get_fatigue_life_cycle(filtered_delta_sigma_fi, category)
            damage = filtered_cycle_data / fatigue_life_cycle_data
            make_damage_table("Filtered Damage-"+category+"-"+fi_element,
                              filtered_delta_sigma_fi,
                              filtered_cycle_data,
                              fatigue_life_cycle_data,
                              damage)
            total_damage = np.sum(damage)
        print("Total Damage for category",category, ":", total_damage)
```

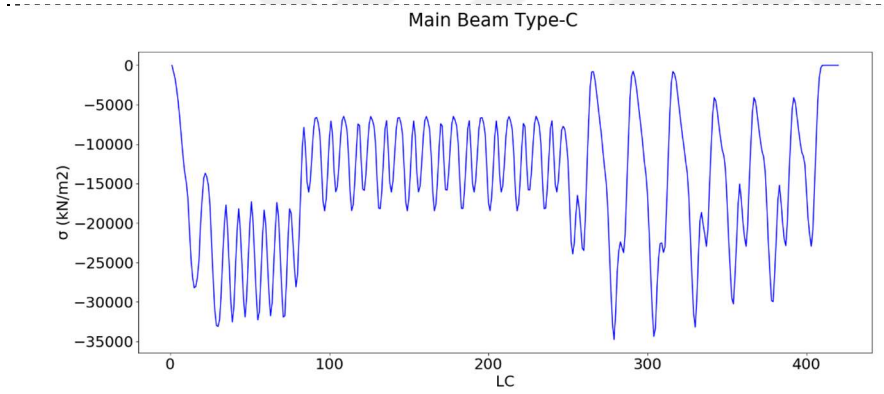
Figure A.7 : Python code for total damage calculation.



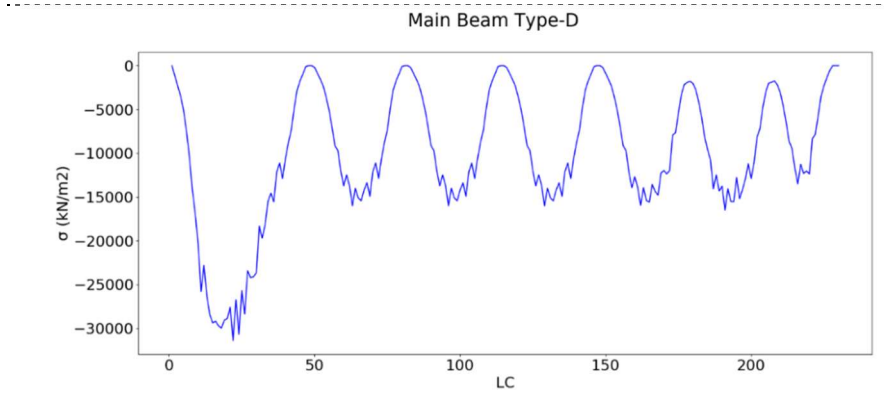
(a)



(b)

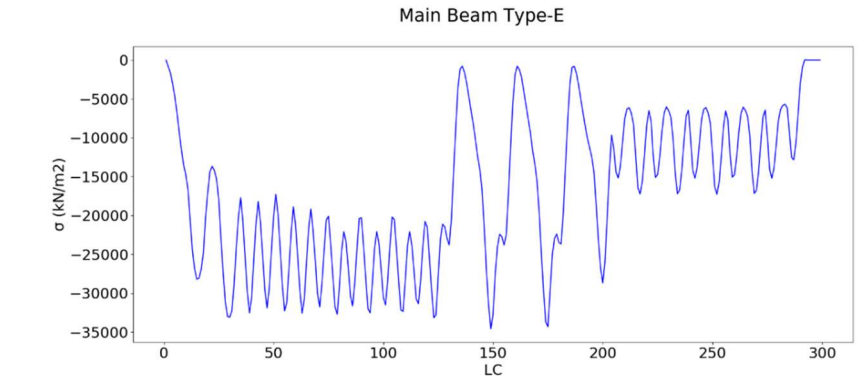


(c)

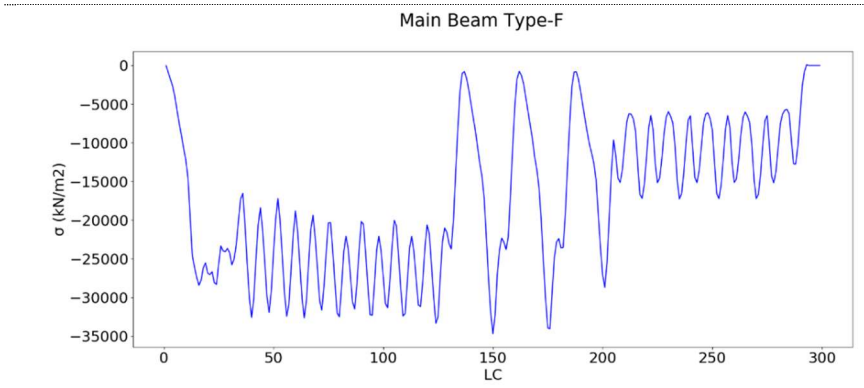


(d)

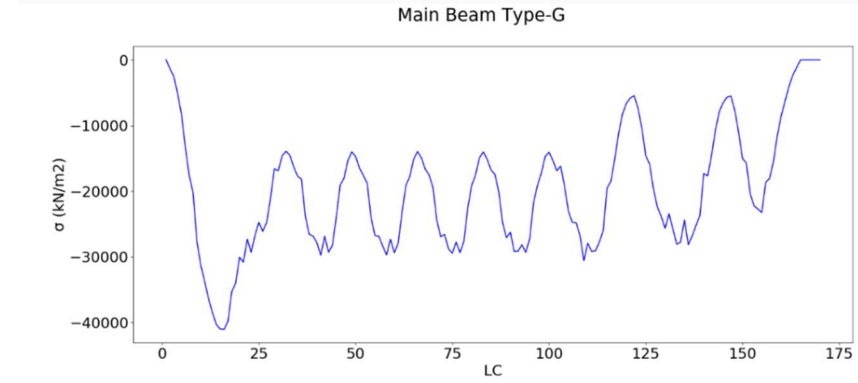
Figure A.8 : Main beam stress graphs for: (a)Type A. (b) Type B. (c) Type C. (d) Type D.



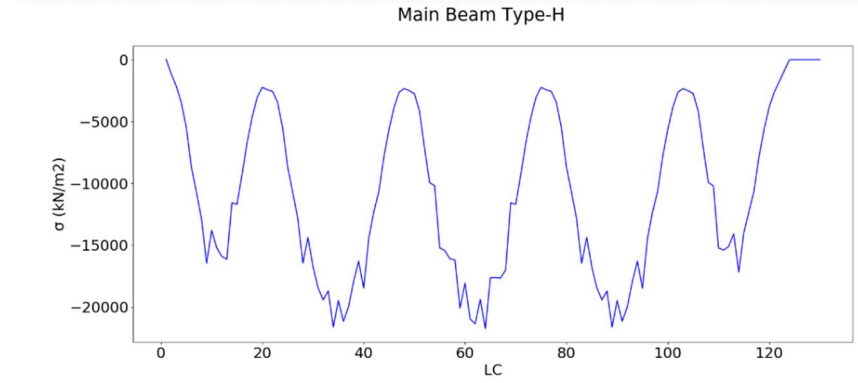
(e)



(f)



(g)



(h)

Figure A.8 (Continued): Main beam stress graphs for: (e)Type E. (f) Type F. (g) Type G. (h) Type H.

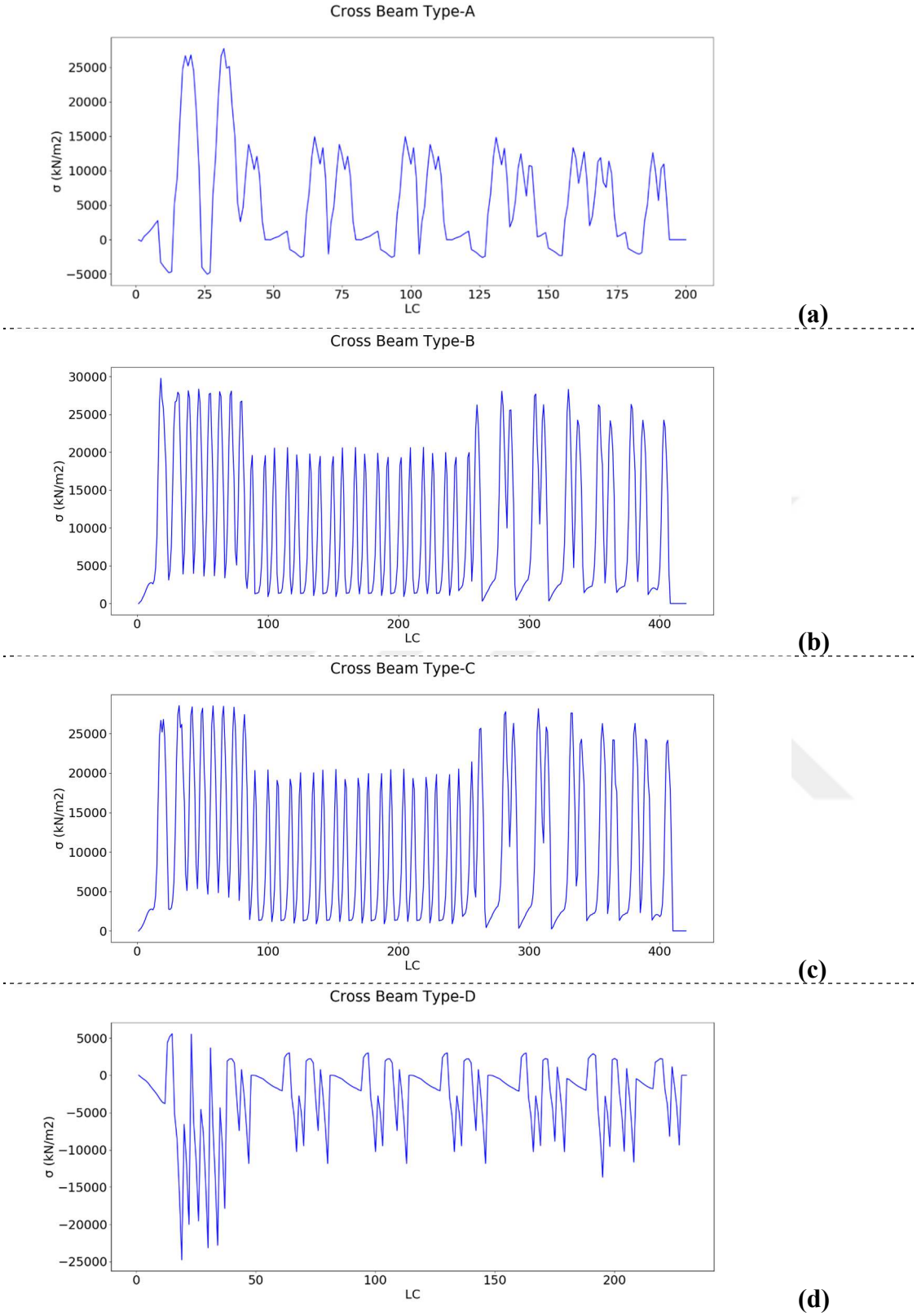
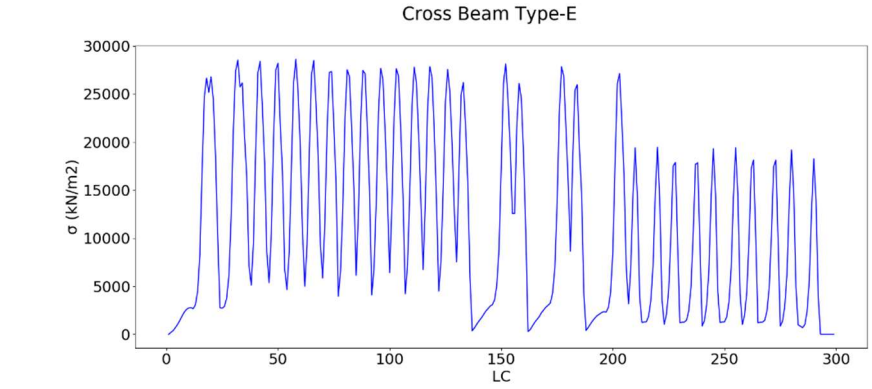
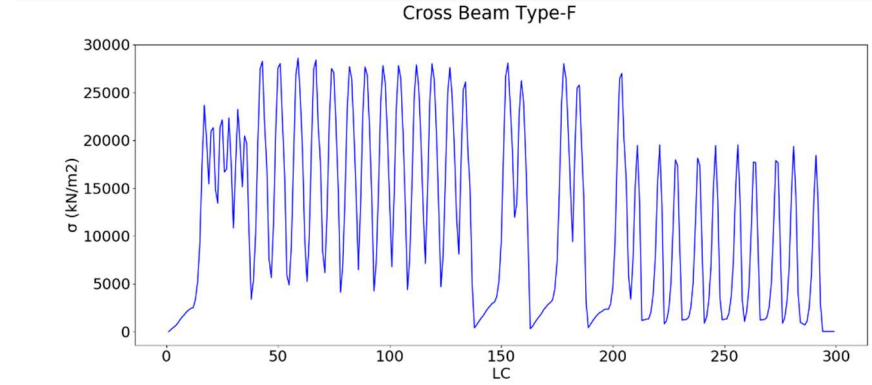


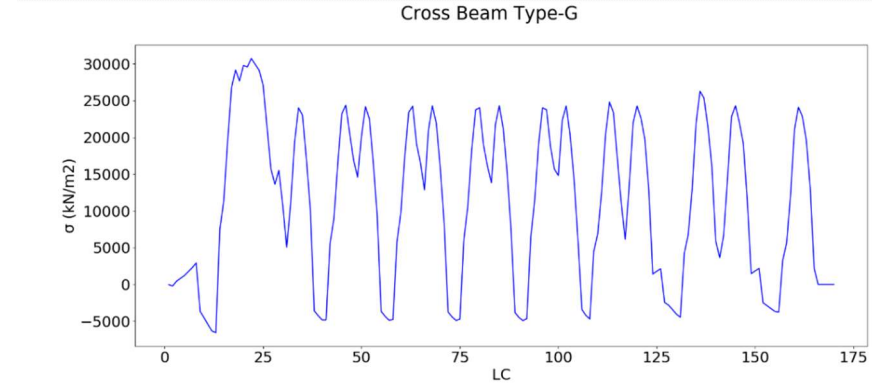
Figure A.9 : Cross beam stress graphs for: (a)Type A. (b) Type B. (c) Type C. (d) Type D.



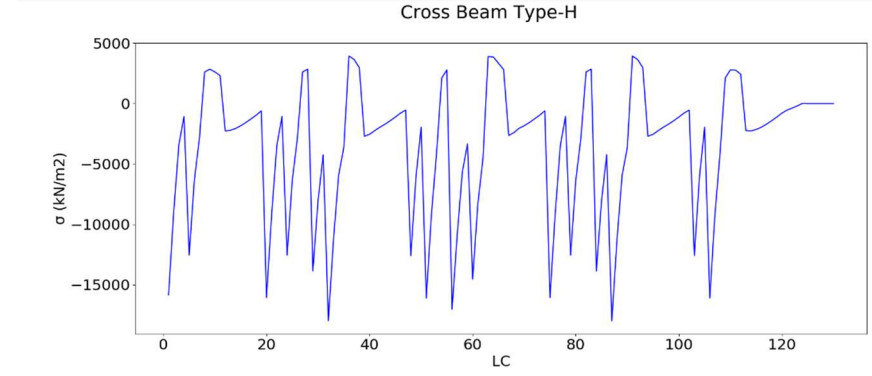
(e)



(f)



(g)



(h)

Figure A.9 (Continued): Cross beam stress graphs for: (e)Type E. (f) Type F. (g) Type G. (h) Type H.

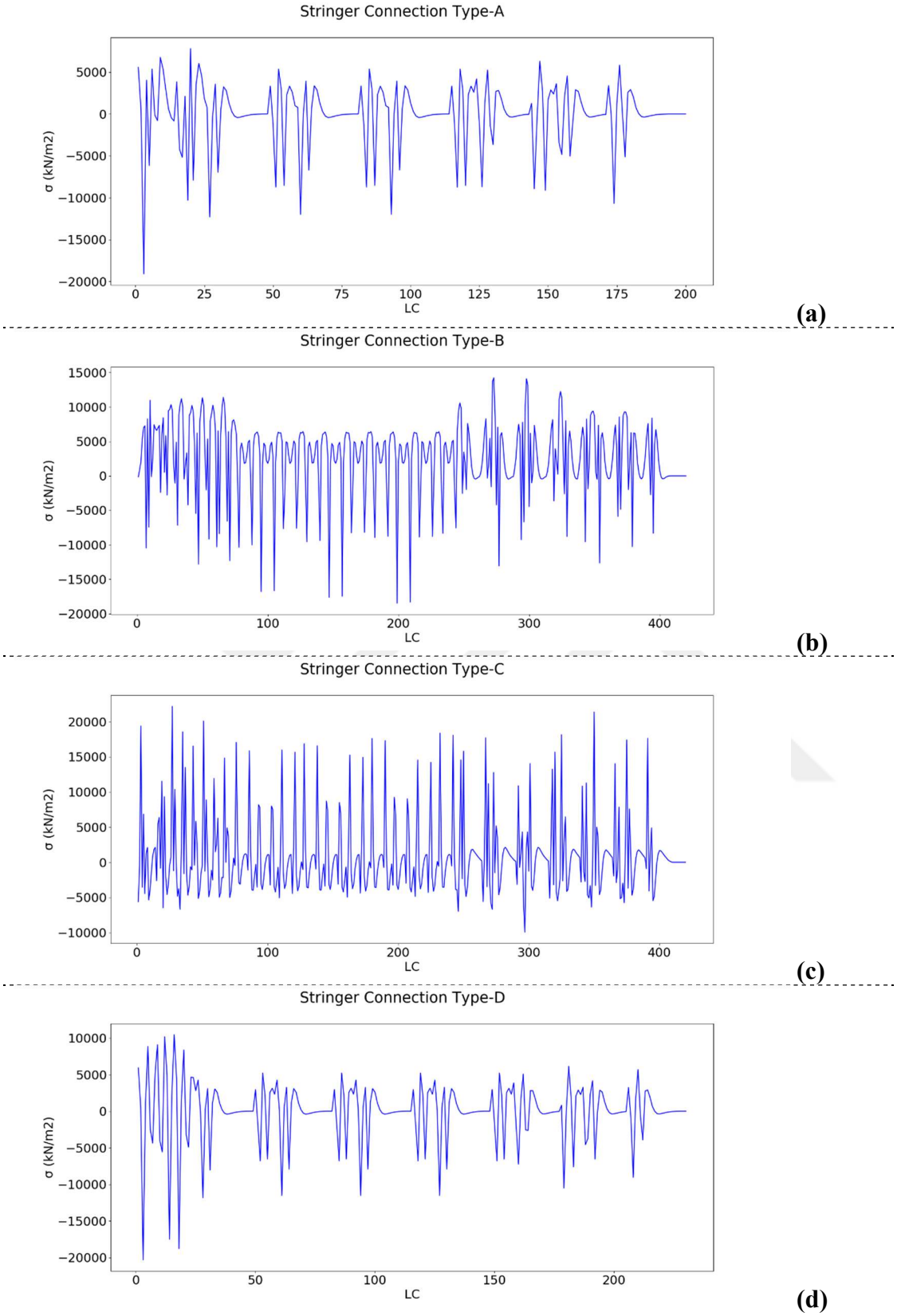
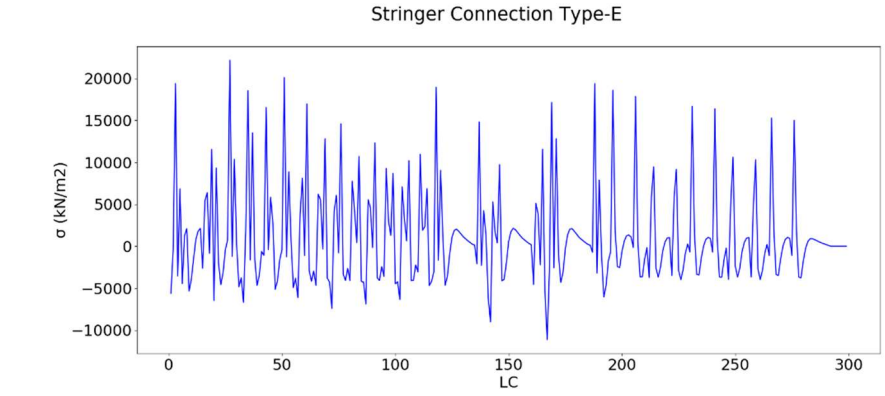
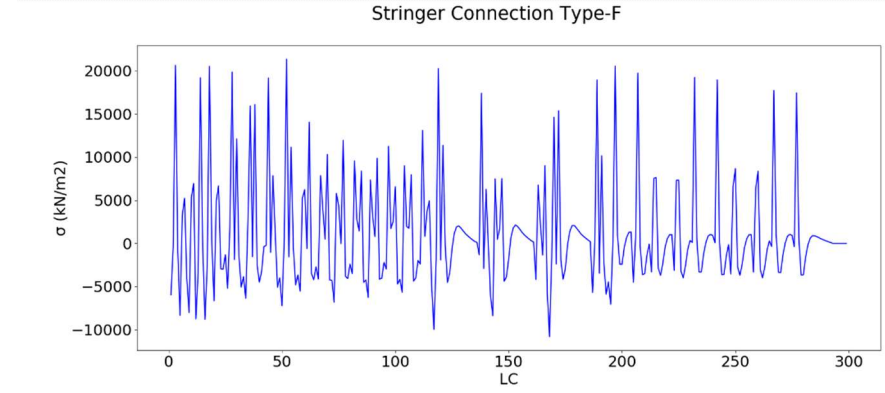


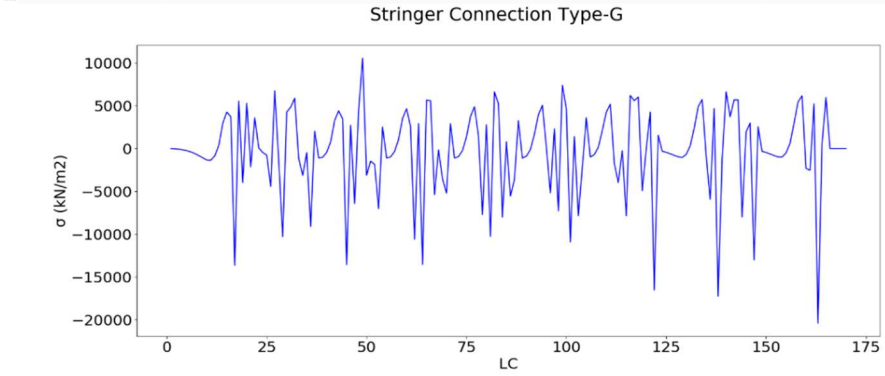
Figure A.10 : Stringer connection stress graphs for: (a)Type A. (b) Type B. (c) Type C. (d) Type D.



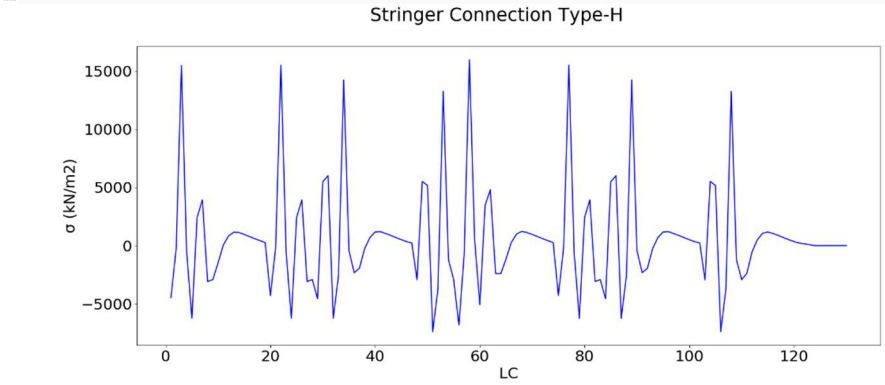
(e)



(f)



(g)



(h)

Figure A.10 (Continued): Stringer connection stress graphs for: (e)Type E. (f) Type F. (g) Type G. (h) Type H.

CURRICULUM VITAE

Name Surname **:** Aylin Ertik

Place and Date of Birth **:** İskenderun / 07.01.1987

E-Mail **:** aylinertik@gmail.com

B.Sc. **:** 2011, Istanbul Technical University, Civil Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- FBM Engineering and Consulting Corporation

  03.2011-12.2014 / Structural Engineer

- Grontmij Engineering Consulting and Design Limited Company

  12.2014-06.2015 / Structural-Bridge Engineer

- Prota Engineering Design and Consultancy Limited Company

  07.2015-12.2015 / Structural&Bridge Engineer

- Amec Foster Wheeler Bimaş Construction and Engineering Company

  05.2016-12.2016 / Structural Engineer

- Yüksel Proje International Incorporated Company

  09.2017-Current / Structural-Bridge Engineer