

UNIVERSITY OF BAHRAIN

College of Engineering



Department of Civil Engineering

Optimization of Prestressed Concrete AASHTO Solid and Voided Slab Beams using a Modified Harmony Search Algorithm

A Thesis Submitted in Partial Fulfilment of the Requirements for the
M.Sc. Degree in Civil Engineering

Submitted by

ALAA ZUHAIR A.REDHA MOHAMED MAKI ALBAHARNA
Student's ID: 2015 5033

Supervised by

Dr. Semih Erhan
Assistant Professor

University of Bahrain

May 2025

Defense Committee's Approval

UNIVERSITY OF BAHRAIN
Deanship of Graduate Studies & Scientific Research



جامعة البحرين
عمادة الدراسات العليا والبحث العلمي

Thesis Discussion Committee Decision

Form Code: UDNGS013

Student Name:	ALAA ZUHAIR A.REDHA MOHAMED MAKI ALBAHARNA
Student ID. No.	20145033
Department:	Civil Engineering Department
College:	College of Engineering
Program:	Master of Science in Civil Engineering
Student GPA:	3.96
Thesis Title:	Optimization of Prestressed Concrete AASHTO Solid and Voided Slab Beams using a Modified Harmony Search Algorithm

The Discussion Committee met on 18.05.2025 in Building S40 upon discussing the student whose details are stated above in public. The discussion committee recommends the following:

The student is awarded the grade PASS in the thesis course CENG599.

☐ Without making any changes to the thesis.

☒ After making the modifications attached with this decision.

Discussion Committee Members Approval:

1.	Supervisor Name:	Dr. Semih Erhan	Sign.:	
2.	Internal Examiner:	Dr. Mahmoud Jahjouh	Sign.:	
3.	External Examiner:	Prof. Serdar ÇARBAŞ	Sign.:	

Abstract

This study examines the overall cost optimization of simply supported bridges with prestressed solid and voided slab beams. The goal of this research is to identify the ideal SI friendly and cost-efficient cross-sectional characteristics of prestressed solid and voided slab beams such as width and height beams as well as dimensions of voids. Design variables include the span length, total bridge width, the area of the prestressed steel, and the dimensions of the solid & voided slab beams' cross sections.

AASHTO Standard Specifications for Highway Bridges (2020) are followed in the design of the prestressed solid and voided slab beams. To obtain the most cost-efficient dimensions of the considered beams, a Modified adaptive Harmony search algorithm (MARSHAL) is applied. Geometric constraints, working stress, and ultimate strength are taken into account. The bridge's total cost is used as the optimality criterion. Numerical examples from the application are designed and an optimal design computer program is coded. In addition, multiple design charts are formulated based on the optimum sections obtained with respect to the corresponding bridge span length. It is concluded that the total cost optimization of prestressed concrete bridges with solid and voided slab beams can be successfully achieved by utilizing the The modified adaptive Harmony search algorithm with cost reduction ranging from (12-21) % compared to AASHTO standard products.

Dedication

I dedicate this thesis to my parents, whose unwavering belief and support have nurtured my love for knowledge and exploration. To my husband and sons, whose love and encouragement inspire me every day.

In loving memory of my older brother, whose hard work and perseverance continue to inspire me.

I am also grateful to my supervisor, Dr. Semih Erhan, for igniting my passion for structural engineering during my academic journey.

Lastly, to all my family and friends, thank you for your love and support throughout this journey.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, Dr. Semih Erhan, for his invaluable guidance and support throughout my research. His insights and encouragement were instrumental in shaping my work. I extend my thanks to Dr. Mahmoud Jahjouh for his assistance and expertise in the optimization process. I am grateful for the feedback from my committee members, which helped refine my thesis.

Finally, I am deeply grateful to my family and friends. To my parents, whose belief in me has been unwavering; to my husband, who has continually supported me; and to my sons, who inspire me to strive for excellence every day.

Table of Contents

Defense Committee’s Approval	A
Abstract	B
Dedication	C
Acknowledgements.....	D
Table of Contents	E
List of Tables	G
List of Figures	H
List of Abbreviations	J
Chapter 1	2
Introduction.....	2
1.1. Overview	2
1.2. Background	3
1.3. Objectives.....	5
1.4. Significance of Study	7
Chapter 2.....	9
Literature Review	9
2.1. Introduction.....	9
2.2. Prestressed Concrete AASHTO Solid Slab Beams.....	10
2.3. Prestressed Concrete AASHTO Voided Slab Beams	12
2.4. Optimization Algorithms.....	15
Chapter 3	22
Research Methodology	22
3.1. Introduction.....	22
3.2. Research Method	22
3.2.1. Properties of AASHTO solid and voided slab beams	25
3.2.2. Optimization Problem Formulation.....	26
3.3 Design Procedure	29
3.4. Research Tools.....	36
3.4.1. The modified adaptive harmony search algorithm (MARSHAL) and search space reduction.....	36

3.5. Numerical Example	39
Chapter 4.....	42
Empirical Findings and Analysis.....	42
4.1. Introduction.....	42
4.2. Study Results	42
4.2.1. Optimization runs section properties and design output	43
4.3. Study Analysis.....	47
4.4. Results Discussion	54
Chapter 5.....	58
Conclusion and Recommendations.....	58
5.1. Introduction.....	58
5.2. Summary and Empirical Results	58
5.3. Future Research	59
References.....	61
Appendix A Solid and Voided Slab Bridges Design Code Integrated with (MARSHAL) Code.....	64
Appendix B Optimization Runs Convergence Graphs.....	76
Appendix C Solid and Voided Slab Bridge Design Charts Code	87
Appendix D AASHTO/PCI vs Optimum Sections Design Charts and Tables	92
المخلص.....	K

List of Tables

Table 3.1 AASHTO/PCI solid and voided slab beams dimensions (mm).....	26
Table 3.2 AASHTO/PCI solid and voided slab beams geometric properties.....	26
Table 3.3 Constraints used for bridge girder optimization.....	27
Table 3.4 Design parameters of the bridge.....	39
Table 3.5 Unit prices.....	39
Table 4.1 Optimum Sections dimensions (mm).....	43
Table 4.2 Optimum sections geometric properties.....	43
Table 4.3 Optimization runs input and output values.....	45
Table 4.4 Best optimum section vs best AASHTO/PCI section and cost reduction % relative to bridge span length.....	55

List of Figures

Figure 2.1 Prestressed concrete solid slab beams bridge deck (Oldcastleinfrastructure.com, 2025).	12
Figure 2.2 Prestressed concrete voided slab beams section (Halbe & Joyce & Roberts-Wollmann & Cousins., 2017)..	14
Figure 2.3 Prestressed concrete voided slab beams bridge (California Department of Transportation, 2022).	14
Figure 3.1 Methodology flowchart.	22
Figure 3.2 Design truck and tandem loads (AASHTO, 2020).	24
Figure 3.3 geometric properties of Solid and Voided Slab Beams.....	25
Figure 3.4 MARSHAL's flowchart (Jahjouh MM, Nackenhorst U, 2016)	38
Figure 3.5 Convergence history of the MARSHAL algorithm during the optimization of a voided slab beam section.....	38
Figure 4.1 Span length vs Number of Strands for Optimum Sections (1-4).	47
Figure 4.2 Span length vs Number of Strands for Optimum Sections (5-8).	48
Figure 4.3 Span length vs Number of Strands for Optimum Sections (9-12).	48
Figure 4.4 Span length vs Number of Strands for Optimum Sections (13-17).	49
Figure 4.5 Span length vs cost for Optimum Sections (1-4).	49
Figure 4.6 Span length vs Cost for Optimum Sections (5-8).	50
Figure 4.7 Span length vs Cost for Optimum Sections (9-12).	50
Figure 4.8 Span length vs cost for Optimum Sections (13-17).	51
Figure 4.9 Span length vs Number of Strands for AASHTO (36) sections.	52

Figure 4.10 Span length vs Number of Strands for AASHTO (48) sections.	52
Figure 4.11 Span length vs cost for AASHTO (36) sections.	53
Figure 4.12 Span length vs cost for AASHTO (48) sections.	53
Figure 4.13 Span length vs cost for best optimum solution sections.	54
Figure 4.14 AASHTO/PCI sections vs OPT sections cost comparison. ..	56

List of Abbreviations

AASHTO	American Association of State Highway and Transportation Officials
PCI	Precast Concrete Institute
PS	Prestressing Steel
SI	The International System of Units
MARSHAL	The modified adaptive Harmony search algorithm
GA	Genetic Algorithm
ACO	Ant Colony Optimization
LRFD	Load and Resistance Factor Design
LLDF	Live Load Distribution Factor
HL-93	Highway vehicular loading proposed by AASHTO in 1993

Chapter 1

Introduction

Chapter 1

Introduction

1.1. Overview

Bridge engineering plays a crucial role in infrastructure development and promoting connectivity between different regions in the Kingdom of Bahrain. It is of utmost importance to optimize bridge designs and construction processes to ensure that the built structures meet the highest standards of safety, durability, and cost-effectiveness. The use of optimization techniques, like the Modified Harmony Search (MARSHAL) algorithm, can greatly assist in achieving these objectives.

In the Kingdom of Bahrain, bridge engineering plays a critical role in establishing efficient transportation networks and facilitating economic growth. With a focus on safety, durability, and cost-effectiveness, AASHTO solid and voided slab beams can serve as viable solutions in bridge construction. The optimization of these beams using the MARSHAL can contribute to the development of sustainable and efficient bridge infrastructure in Bahrain.

In order to achieve the highest level of effectiveness in design and ensure optimal structural integrity, it is absolutely crucial to thoroughly optimize the design and possess a deep understanding of material strength and properties. Conventional design techniques and approaches might not fully unlock the true potential of a structure, making it essential to explore advanced optimization algorithms such as the MARSHAL (Gupta & Khosravy & Patel & Dey., 2020). By leveraging this algorithm, designers can exceed project requirements without unnecessary wastage, thereby

conserving materials and leading to tremendous long-term cost savings. It is therefore of utmost importance to explore strategies that maximize the effectiveness of design processes, ultimately generating significant economic benefits.

1.2. Background

Bridge engineering involves the design, construction, and maintenance of bridges, which are essential structures that connect two points over obstacles such as rivers, valleys, or roads. The background of bridge engineering dates back thousands of years, with early civilizations creating basic structures to cross natural barriers. Over time, advancements in materials, techniques, and technology have revolutionized bridge construction and allowed for the development of more efficient and durable structures (Tyrrell, 1911).

One notable advancement in bridge engineering is the use of prestressed concrete beams. Prestressed concrete is a type of concrete that is reinforced by pre-tensioning or post-tensioning steel wires or strands within the concrete. This process creates a state of compression in the concrete, which helps to enhance its load-carrying capacity and reduce the effects of external forces, such as tension and bending moments (Dinges, 2009).

Prestressed concrete beams offer several advantages in bridge construction. They have a higher strength-to-weight ratio compared to traditional reinforced concrete beams, allowing for longer spans and lighter structures. This reduces the number of piers or supports required, decreases construction cost and time, and minimizes disruption to traffic during construction.

Furthermore, prestressed concrete beams have enhanced resistance to cracking and corrosion, leading to longer service life and reduced

maintenance requirements. Their robustness and durability make them highly suitable for withstanding the harsh environmental conditions and heavy loads typically encountered in bridge engineering.

The applications of prestressed concrete beams in bridge engineering are widespread. They are used in various bridge types, such as beam bridges, arch bridges, and cable-stay bridges. Depending on the specific requirements and constraints of each project, engineers choose the most appropriate type of prestressed concrete beam, considering factors like span length, traffic volume, and aesthetic considerations.

The field of bridge engineering has a rich history and has evolved significantly over time. The introduction of prestressed concrete beams has revolutionized the construction of bridges, offering numerous benefits in terms of strength, durability, and cost-effectiveness. The applications of prestressed concrete beams in bridge engineering are extensive, contributing to the development of efficient and reliable structures that connect people and places (Ding & Ning, 2023).

One exceptional tool that can be employed to tackle complex design issues is a heuristic optimization approach. This approach offers a unique and innovative perspective, enabling designers to navigate and resolve intricate design complexities effectively. Taking inspiration from the realm of music, the MARSHAL empowers designers to uncover and exploit the full possibilities hidden within their designs (Dubey et al., 2021). Its remarkable adaptability and customization capabilities make it an ideal choice for resolving complex design challenges with ease. By employing such advanced optimization methods, designers can create truly outstanding and sustainable designs that truly push the boundaries of what is attainable.

However, it is important to note that expertise is paramount when utilizing these advanced optimization methods. Combining the power of these methods with the knowledge and experience of skilled designers and engineers leads to remarkable achievements and groundbreaking advancements in the field of design and engineering. By harnessing the potential of advanced optimization algorithms like MARSHAL, researchers and practitioners can revolutionize the field of design and engineering, paving the way for innovative solutions and advancements.

1.3. Objectives

The objective of this study is to effectively optimize and enhance the performance of prestressed concrete solid and voided slab bridges. To achieve this, a highly efficient and innovative MARSHAL will be employed. The aim is to develop an SI friendly design that ensures structural integrity and strength of the bridge, while significantly reducing overall costs associated with concrete and steel materials.

Throughout the study, all necessary design code requirements will be meticulously considered and adhered to, prioritizing safety, longevity, and reliability. Compliance with both local and international standards will be maintained to ensure the bridge can withstand a variety of environments and scenarios. By incorporating these standards into the optimization process, a bridge design will be created showcasing engineering excellence and practical feasibility.

The focus of the approach will be to achieve a high level of cost-effectiveness without compromising the overall quality and durability of the bridge. Financial efficiency is important, and a well-designed bridge should not only meet the required design standards but also be SI friendly

and economically viable. Therefore, striking a harmonious balance between these factors is of utmost significance in the study.

The integration of advanced optimization algorithms in the design process brings forth numerous benefits. Firstly, it allows designers to effectively explore a wide range of design possibilities. The MARSHAL, with its robust capabilities, enables designers to systematically evaluate and analyze various design configurations, allowing for comprehensive optimization. Through this process, designers can identify the most efficient and optimal design solutions, thus ensuring cost-effectiveness and resource conservation (Dubey et al., 2021).

Moreover, the application of advanced optimization methods significantly reduces the risk of design failure. By thoroughly considering material strength and properties, as well as utilizing powerful algorithms like MARSHAL, designers can identify potential weaknesses or flaws in the design early on. This early detection enables designers to make necessary adjustments and modifications, preventing costly and time-consuming errors in the construction or manufacturing stage.

Additionally, the use of advanced optimization algorithms promotes innovation in design. By exploring unconventional design configurations and pushing the boundaries of traditional design techniques, designers can create structures that are not only functional but also visually striking. The MARSHAL facilitates this innovative design process by allowing designers to experiment with different parameters and variables, ultimately leading to unique and groundbreaking solutions.

Furthermore, the employment of advanced optimization methods promotes sustainability in design. By optimizing the use of materials and reducing wastage, designers can contribute to the preservation of natural resources

and the environment. The MARSHAL aids in achieving this goal by uncovering design solutions that minimize material consumption while fulfilling project requirements. This sustainable approach not only benefits the present generation but also ensures a better future for the generations to come.

1.4. Significance of Study

The integration of advanced optimization algorithms, such as MARSHALs, in the design process is crucial for achieving the highest level of effectiveness and optimal structural integrity. These advanced methods allow designers to explore a wide range of design possibilities, minimize the risk of design failure, foster innovation, and promote sustainability. By combining the power of advanced optimization algorithms with the expertise and ingenuity of skilled designers and engineers, the field of design and engineering can reach new heights and accomplish feats previously deemed impossible.

By successfully optimizing the prestressed concrete solid and voided slab bridge design through the MARSHAL method, a blueprint will be created that offers unparalleled engineering excellence, maximized cost-effectiveness, and meticulous adherence to the highest design standards. This study will contribute to the advancement of bridge design practices, promoting sustainable and efficient infrastructure development in both local and international contexts.

Chapter 2

Literature Review

Chapter 2

Literature Review

2.1. Introduction

The AASHTO design specifications for highway bridge design (2020) are widely adopted in the Kingdom of Bahrain. The provisions for pretensioned members in concrete bridge design follow the ultimate strength design. The ultimate moment (M_n or M_u) is computed according to AASHTO (2020) rules, and the actual prestress contribution at service load is determined using the same methods based on gross areas. In AASHTO (2020), no consideration is given to the tensile capacity of the concrete, and the upper 20% of the section is not considered to provide tensile resistance, with the specific value depending on the amount of prestressing.

Prestressed concrete is a method for overcoming concrete's natural weakness in tension. It can be added to a concrete member during new construction or to repair an existing member by applying internal forces in the concrete to the member using steel strands (Zhang et al., 2022). The member serves to neutralize or reduce the effects of "live" or dynamic loadings on the frame or structure, making the member perform better structurally (increased strength and serviceability). Partial safety factor design is used to provide extra safety to the structure compared to working stress design. This method uses a structural member subjected to a series of loads throughout its design life. The mean anticipated load effects during the specified design life should be calculated, as well as the characteristic load effects during the same period. Load combinations should then be performed with the load effect and their appropriate factors. The load effect

can be a bending moment, shear force, etc. The absolute values of the individual load effects in the most unfavorable load combination forming part of the series should be added together to give an ultimate load effect. A full analysis should be done with the mean load effects and these calculated extreme load effects and their effects on the structure. The aim should be to maintain a serviceable structure with the absolute values of the load effects having a higher than 90% chance of occurring for a specified limit state. This method differentiates the prescribed mean loads from the variable live loads and takes into account the combination of different types of loads and uncertainties in their effects, into a partial safety factor applied to the load effect.

The structural benefits related to the use of prestressing in pretensioned beams include higher ultimate load-carrying capacity, larger deflection control, longer spans, and improved fatigue strength of the system. Ultimate loads can be increased because of the application of additional compression across the concrete section. The improved flexural behavior provided by the prestress is used to design light structural beam elements, especially suitable for long-span constructions such as conventional bridges and trapezoidal and box bridges (Harajli, 1993).

2.2. Prestressed Concrete AASHTO Solid Slab Beams

Prestressed Concrete AASHTO Solid Slab Beams are an extensively used construction material that provides significant benefits in terms of strength and longevity. Its exceptional performance characteristics make it a preferred choice for various applications in the construction industry.

One of the key advantages of Prestressed Concrete AASHTO Solid Slab Beams is the ability to withstand heavier loads and resist cracking. By introducing compression forces into the concrete before external loads are

applied, it effectively counters the tensile stresses that can cause cracks. This enhances the overall strength of the structure and extends its service life, making it highly durable (Hueste et al., 2015).

Additionally, Prestressed Concrete AASHTO Solid Slab Beams exhibit exceptional resistance to corrosion, which is a major concern in structural materials. By utilizing a combination of high-strength concrete and prestressing steel (PS), the concrete is protected from the harmful effects of moisture and other corrosive elements. This makes Prestressed Concrete AASHTO Solid Slab Beams a reliable choice even in harsh environments, such as coastal areas or industrial settings.

Moreover, the versatility in design offered by Prestressed Concrete AASHTO Solid Slab Beams is noteworthy. Architects and engineers can explore their creative vision while maintaining structural integrity, thanks to the ability to create a wide range of shapes and forms. This flexibility opens up endless possibilities in architectural design and enables the construction of unique and aesthetically pleasing structures.

Another advantage is the speedy installation process associated with Prestressed Concrete AASHTO Solid Slab Beams. As the precast elements are manufactured off-site, construction time is significantly reduced, leading to cost savings and efficient project completion. This makes Prestressed Concrete AASHTO Solid Slab Beams an attractive choice for time-sensitive projects or areas with limited construction access.

Lastly, the sustainability aspect of Prestressed Concrete AASHTO Solid Slab Beams is worth mentioning. By optimizing material usage and recycling methods, the environmental impact of structures made with this material can be minimized. Additionally, the long lifespan of Prestressed

Concrete AASHTO Solid Slab Beams reduces the need for frequent repairs or replacements, further contributing to sustainability.

In conclusion, Prestressed Concrete AASHTO Solid Slab Beams are reliable, durable, and versatile construction materials that offer numerous advantages in terms of strength, durability, corrosion resistance, design flexibility, speed of installation, and sustainability. They continue to play a crucial role in modern construction, shaping city skylines and providing safe and long-lasting structures.



Figure 2.1 Prestressed concrete solid slab beams bridge deck (Oldcastleinfrastructure.com, 2025).

2.3. Prestressed Concrete AASHTO Voided Slab Beams

The Prestressed Concrete AASHTO voided slab beams have gained significant acclaim and recognition in the field of civil engineering due to their exceptional structural efficiency, remarkable durability, and remarkable cost-effectiveness (California Department of Transportation, 2022). The incorporation of voids in these beams not only assists in reducing the weight of the structures but also offers an enhanced load-carrying capacity, making them highly desirable for a wide range of

construction projects. Moreover, these beams exhibit excellent resistance to various environmental factors such as corrosion and seismic events, ensuring their longevity and reliability. The innovative design and construction techniques employed in the manufacturing of these beams further contribute to their widespread popularity and adoption. With continuous advancements and refinements, the Prestressed Concrete AASHTO voided slab beams have proven to be a cutting-edge solution that meets the stringent demands of modern civil engineering, setting new standards for structural excellence and efficiency.

These exceptional features are a testament to the groundbreaking nature of the Prestressed Concrete AASHTO voided slab beams. Not only do they provide exceptional strength and stability, but they also deliver cost-effective solutions for the construction industry. The incorporation of voids within the beams brings a revolutionary approach to structural design, bringing about a reduction in weight without compromising on load-bearing capacity.

The longevity and reliability of these beams make them highly sought after in the realm of civil engineering. With their resistance to corrosion and seismic events, they can withstand the harshest of environmental conditions, ensuring the safety and integrity of the structures they support. This durability translates into long-term cost savings, as the need for frequent repairs or replacements is significantly reduced.

The popularity and adoption of these beams can be attributed to their innovative design and construction techniques. The manufacturing process involves intricate procedures that result in beams of unparalleled quality. The use of prestressed concrete further enhances their performance, providing added strength and reinforcement.

As advancements and refinements continue to shape the Prestressed Concrete AASHTO voided slab beams, they continue to surpass expectations and set new industry standards. Their ability to meet the ever-evolving demands of modern civil engineering is a testament to their excellence and effectiveness. The importance and impact of these beams cannot be understated, as they pave the way for a more efficient, sustainable, and resilient built environment.



Figure 2.2 Prestressed concrete voided slab beams section (Halbe & Joyce & Roberts-Wollmann & Cousins., 2017).



Figure 2.3 Prestressed concrete voided slab beams bridge (California Department of Transportation, 2022).

2.4. Optimization Algorithms

Harmony Search algorithm was originally presented as a method for optimizing water reservoir system operations. The concept of the algorithm is inspired by a musician's spontaneous creation of a musical piece. Over the years, the algorithm has found applications in various engineering design and optimization problems primarily because of its simplicity and reliable convergence. The algorithm's effectiveness lies in its ability to strike a balance between exploration and exploitation, enabling it to search for the best possible solution across an unexplored landscape. This is achieved through the improvisation of new ideas and the refinement of existing ones based on a harmony memory, which represents candidate solutions. The harmony search algorithm starts with the creation of a random harmony memory that provides an initial set of solutions. Subsequently, through pitch adjustment and harmony memory consideration, a new harmony improvisation is generated, ideally surpassing the previous generation. This process is repeated for a specific number of iterations until a predetermined stopping condition is met.

Incorporating the characteristics of the best few solutions to improve the next generation's search performance is a desired requirement in most optimization problems. Recent research has shown that the harmony search algorithm, whose performance is known to be inferior to other evolutionary algorithms, can be enhanced to obtain significant improvements. In this chapter, a review and analysis of recent improvements to the harmony search algorithm are presented, where modifications have been made to memory consideration, pitch adjustment, and the harmony memory considering rate. The harmony search algorithm is a music-based algorithm where a set of solutions to an optimization problem is considered a song. Individually, each solution has considerations to decision variables as a

musician has a note that he can play with a specific instrument (Dubey et al., 2021).

In the comprehensive literature review conducted, the primary focus was to provide an in-depth discussion and analysis of three widely used optimization algorithms - the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and the Modified adaptive Harmony search algorithm (MARSHAL). These algorithms, due to their versatility and adaptability, have gained significant popularity and found extensive applications in various diverse fields such as engineering, computer science, economics, and numerous others. The Genetic Algorithm is a highly potent and influential optimization algorithm that draws inspiration from the intriguing process of natural selection (Pham & Hong, 2022). By incorporating principles derived from genetics and evolution, this algorithm progressively moves towards identifying and attaining the most optimal solution. It achieves this through the integration of several crucial steps, including selection, crossover, mutation, and adaptation of potential solutions. The Genetic Algorithm has consistently demonstrated its robustness and efficiency across a wide range of domains, making it particularly suitable for optimization problems characterized by highly complex constraints and vast high-dimensional search spaces. On the other hand, Ant Colony Optimization represents a highly effective and proven metaheuristic algorithm that closely mimics the efficient foraging behavior exhibited by ants. Its exceptional competency lies in its ability to solve problems associated with combinatorial optimization, which involves identifying the optimal combination of elements from an extensive set (Sachan & Singh, 2021). The algorithm successfully simulates the efficient communication that occurs among ants through the exchange of pheromones, ultimately allowing them to collectively uncover the shortest

path to a food source. With remarkable success, ACO has found extensive applications in domains such as routing, scheduling, data mining, transportation, telecommunications, and various other real-world scenarios. Additionally, the Harmony Search algorithm takes inspiration from the art of musical improvisation and adopts an innovative metaheuristic approach. The optimization process simulates the search for optimal harmony, thereby facilitating efficient and effective optimization. Initially designed to address continuous optimization problems, the harmony search algorithm has been thoughtfully modified and expanded to effectively tackle discrete and combinatorial problems as well. By establishing a flexible framework that caters to the specific characteristics of different problems, the modified version of the harmony search algorithm excels in exploring and exploiting diverse search spaces. This adaptability allows for enhancements tailored to specific problem characteristics, thereby striving to improve the overall performance and accuracy of the algorithm when dealing with complex optimization problems across a multitude of domains (Dubey et al., 2021). The literature references consulted provided a comprehensive overview of the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and the Modified adaptive Harmony search algorithm (MARSHAL). These valuable sources highlight the extensive range of applications these algorithms possess across a spectrum of fields, including engineering, computer science, economics, and beyond. Time and again, these algorithms have consistently demonstrated their efficacy and versatility as proficient tools in the realm of optimization problem-solving. Moreover, the references also shed light on potential areas where further improvement of these algorithms is required. As technology continues to advance and novel challenges arise, it becomes imperative to enhance these optimization algorithms to ensure their ability to deliver accurate and efficient solutions.

The literature points towards avenues for future research and development, encouraging innovation and the exploration of novel techniques and adaptations. These avenues of exploration aim to significantly enhance the performance and accuracy of the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and the Modified adaptive Harmony search algorithm (MARSHAL). By actively addressing these identified areas, researchers can effectively push the boundaries of optimization algorithms, ultimately unlocking new possibilities for the resolution of complex real-world problems. The field of optimization algorithms is continuously evolving, and it is of paramount importance to stay updated with the latest advancements and findings. By staying abreast of the ongoing research and development, individuals can harness the knowledge and insights gained from the literature to propel their own endeavors in optimization problem-solving. The shared wisdom and collective efforts within the academic community foster an environment of collaboration, innovation, and progress. Through collaborative exploration and the exchange of ideas, researchers can pave the way for groundbreaking advancements in the field of optimization, contributing to the overall improvement of society as a whole. As the optimization field continues to expand, researchers and practitioners must remain vigilant in their efforts to push the boundaries of current algorithms while finding new ways to enhance their performance. By examining the latest literature, individuals can gain valuable insights into the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and the Modified adaptive Harmony search algorithm (MARSHAL), understanding their effectiveness, limitations, and potential areas for improvement. The comprehensive analysis and review of these algorithms highlight their versatility and applicability across various domains. Through genetic algorithms, researchers can leverage the principles of natural selection and evolution to find optimal solutions to complex

problems. The Genetic Algorithm (GA) harnesses the power of selection, crossover, mutation, and adaptation to explore vast search spaces and identify highly desirable outcomes. This algorithm's robustness and efficiency make it a valuable tool for addressing optimization problems characterized by complex constraints and high-dimensional search spaces. In contrast, Ant Colony Optimization (ACO) takes inspiration from the efficient foraging behavior of ants to solve combinatorial optimization problems (Sachan & Singh, 2021). By simulating ant communication through pheromone exchange, ACO effectively navigates and uncovers the best combinations from a large set of elements. Its successful applications in routing, scheduling, data mining, transportation, and telecommunications demonstrate its ability to tackle real-world challenges. The Modified adaptive Harmony search algorithm (MARSHAL) introduces a unique approach by drawing inspiration from musical improvisation. This innovative metaheuristic algorithm searches for optimal harmony, enabling efficient optimization techniques. Originally designed for continuous optimization problems, the MARSHAL exhibits adaptability and versatility in solving discrete and combinatorial problems. By tailoring its framework to specific problem characteristics, the MARSHAL explores diverse search spaces, aiming to improve overall performance and accuracy in complex optimization scenarios. The literature references consulted provide a comprehensive overview of the versatility and widespread applications of the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and the Modified adaptive Harmony search algorithm (MARSHAL). These algorithms have repeatedly demonstrated their efficacy and adaptability in solving optimization problems across various fields, including engineering, computer science, economics, and beyond. Their consistent performance in diverse domains makes them indispensable tools for optimization problem-solving

endeavors (Vié, 2020). Furthermore, the literature also highlights areas that would benefit from continued research and development to further enhance these algorithms. As technology advances and new challenges arise, it becomes crucial to refine and innovate optimization algorithms to ensure accurate and efficient solutions. The identified avenues for future exploration encourage researchers to pioneer novel techniques and adaptations, thereby pushing the boundaries of the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and the Modified adaptive Harmony search algorithm (MARSHAL). These endeavors strive to significantly improve the performance and applicability of these algorithms, unlocking their potential to solve complex real-world problems. As the optimization field continues to evolve, it is essential to stay up to date with the latest advancements and findings. By actively engaging with ongoing research and development, individuals can harness knowledge and insights gained from literature to advance their own optimization problem-solving endeavors. The collaborative spirit within the academic community fosters innovation, progress, and the exchange of ideas. Through collective exploration, researchers can propel groundbreaking advancements in optimization, contributing to the betterment of society as a whole. Remaining receptive to new ideas and embracing collaboration allows for continuous improvements and paves the way for solving increasingly complex real-world problems using optimization algorithms (Sachan and Singh, 2021).

Chapter 3

Research Methodology

Chapter 3

Research Methodology

3.1. Introduction

This study utilizes a modified Harmony Search Algorithm to solve the cost minimization problem with diverse design constraints and provides design procedures as well as design transformation procedures for practical practices. The accuracy of the optimization design is verified by solving examples. It is found that the MARSHAL works effectively and stably for the optimization of bridge girders.

This chapter aims to explain the process used to conduct this study. In addition, to explain the code formulation and previous studies tools which were used to develop the final optimization model and design charts.

3.2. Research Method

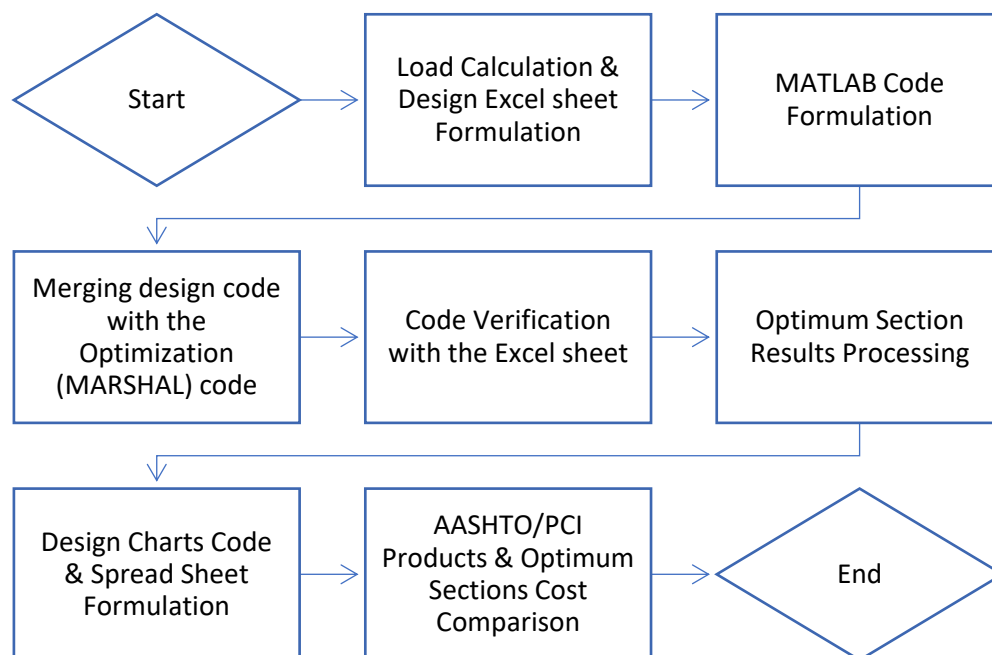


Figure 3.1 Methodology flowchart.

This parametric study aims to develop more cost-effective solid and voided slab beam sections. A more detailed summary of the research approach is provided in Figure 3.1. In the beginning, an excel sheet is formulated to calculate the loads, prestressing steel (PS) area and configuration, prestressed losses, check the service stresses limit state, and the flexural design criteria. Subsequently, a MATLAB code is formulated in order to be used along with the optimization code. The same bridge configuration which is developed in the excel sheet is used in the optimization code in order to come up with an optimum design section ‘OPT sol.1’ and the same is also used in the excel sheet to verify the code. After that, optimization code along with the objective function are used to collect optimum section results. The next step is to formulate a design charts code to compare the optimum sections ‘OPT sol.2’ to ‘OPT sol.17’ and design charts are developed as a last stage. Finally, a cost comparison is conducted between AASHTO/PCI sections as described in the Precast Concrete Institute (PCI, 2014) and the optimum sections obtained in this study.

The bridges considered in this research study are limited to single span bridges. Thus, a simply supported slab beam is assumed. Self-weight and superimposed loads make up the dead loads. Additionally, live loads are computed using the AASHTO LRFD design standards (2020). The HL-93 vehicular live load is considered in the live load analysis. Specifically, the components of the HL-93 live load model (LL) are as follows:

- The design truck load
- The design tandem load
- The design lane load (9.3 kN/m)

HL-93 live load components, truck load and tandem load are displayed in the following Figure 3.2.

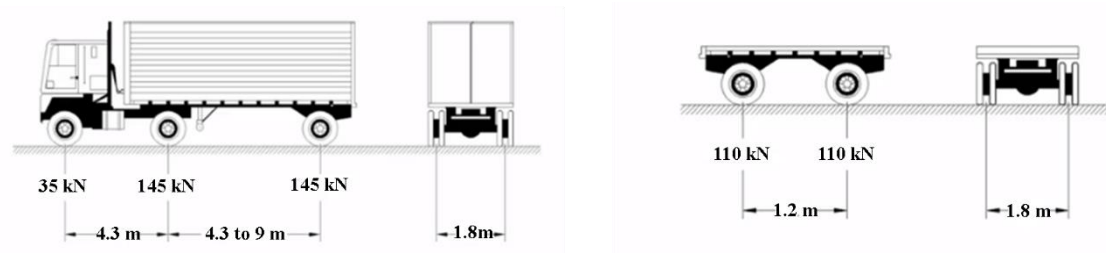


Figure 3.2 Design truck and tandem loads (AASHTO LRDF Bridge Design Specifications, 2020).

There is no requirement for finite element software in the structural analysis of such beams under the specified loads. The internal forces are calculated by using simple supported beam analysis. However, the design and optimization are handled by a programming code. The produced slab beam sections in this study, resulting in cost optimization, are compared with AASHTO products.

Precast pretensioned solid and voided slab beams are typically used for relatively short span structures. Parametric research is carried out to examine the structural efficiency of AASHTO solid and voided slab beams using the Modified adaptive Harmony search algorithm (MARSHAL). Numerous geometric characteristics, including span length, overall bridge width, and different slab beam dimensions, are taken into account in parametric research. The span lengths in this study range from 9 to 24 meters, which corresponds to the practical span lengths of short span bridges. The optimization technique optimizes the slab dimensions, which are then contrasted with AASHTO slab beams that are frequently utilized in bridge design. Seven constraints are applied to the optimization process to guarantee compliance with the AASHTO LRFD design specification (2020). The slab beam cost is the optimization criterion.

The primary goal is to reduce costs within the parameters specified in the AASHTO LRFD Bridge design specification (2020). These costs are computed to include the cost of prestressing steel (PS) and concrete.

3.2.1. Properties of AASHTO solid and voided slab beams

The geometric properties of currently used AASHTO slab beam sections are presented in this section. There are eight types of AASHTO/PCI sections, and the geometric configuration is illustrated in Figure 3.3. The dimensions of AASHTO/PCI slab beams are presented in Table 3.1. The geometric properties of the slab beams are then calculated using these dimensions and presented in Table 3.2 where Area is the gross cross-sectional area, y_b is the centroid axis from bottom, Inertia is the moment of inertia, Torsional is the torsional moment of inertia and S_b is section modulus from the bottom. These geometric properties are then used to calculate structural efficiency of these sets of bridges to compare them to newly developed slab beam sections in this study.

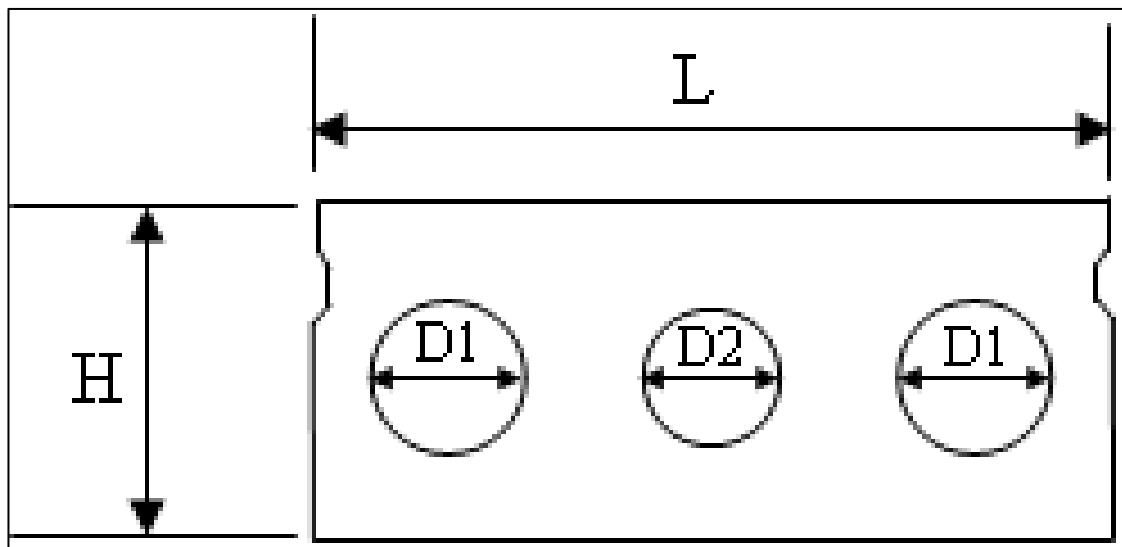


Figure 3.3 geometric properties of Solid and Voided Slab Beams.

Table 3.1 AASHTO/PCI solid and voided slab beams dimensions (mm).

Type	L	H	L1	L2	No. of Voids	D1	D2
SI-36	914.4	304.8	0	0	0	0	0
SII-36	914.4	381	266.7	190.5	2	203.2	0
SIII-36	914.4	457.2	266.7	190.5	2	254	0
SIV-36	914.4	533.4	254	203.2	2	304.8	0
SI-48	1219.2	304.8	0	0	0	0	0
SII-48	1219.2	381	254	355.6	3	203.2	203.2
SIII-48	1219.2	457.2	241.3	368.3	3	254	254
SIV-48	1219.2	533.4	254	355.6	3	304.8	254

Table 3.2 AASHTO/PCI solid and voided slab beams geometric properties.

Type	Area (mm ²)	y _b (mm)	Inertia (mm ⁴)	Torsional (mm ⁴)	S _b (mm ³)
SI-36	278709	152.4	2157743710	69910896214	14158423
SII-36	283528	190.5	4046966597	39920249093	21243919
SIII-36	316722	228.6	6873750776	36598236724	30068901
SIV-36	341809	266.7	10716813724	31842660654	40183029
SI-48	371612	152.4	2876991614	165714716952	18877898
SII-48	367227	190.5	5368059365	84695988237	28178789
SIII-48	405406	228.6	9096895327	74234789772	39793943
SIV-48	453719	266.7	14367215833	73741914383	53870326

3.2.2. Optimization Problem Formulation

The structural design optimization of bridge girder can be formulated as an optimization problem with the objective of minimizing the girder material cost. To achieve this, it is required to find a solution x that minimizes an objective function $F(x)$ where x is a vector that describes the design of the bridge girder and $F(x)$ is the cost of the material defined as,

$$F(x) = n \times (C_c + C_s) \quad (3.1)$$

where n is the number of slab beams, C_c and C_s are the cost components of the concrete, and strands of each slab beam.

Table 3.3 Constraints used for bridge girder optimization.

#	Constraint	Type	Limit
1	Maximum number of strands (n_s)	Max	Based on minimum 50 mm center to center strands spacing and 60 mm edge distance
2	Minimum strands spacing	Min	$\min(1.5 (m), \frac{1}{8} \times \text{slab thick.})$
3	Compressive stress in service	Max	$0.60 f_c'$ MPa
4	Tensile stress in service	Max	$0.50\sqrt{f_c'}$ MPa
5	Strands stress in service	Max	$0.8 \times 0.9 f_{pu}$
6	Strength Requirement	Min	M_u kN.m
7	Reinforcement	Min	$\min(1.3 M_u, M_{cr})$

To include the constraints with regard to strength and service limit states of the AASHTO LRFD (2020) code, the objective function defined in Equation (3.1) has to be modified to include those constraints. Thus, the constrained optimization problem is converted into an unconstrained problem via the penalty method. The penalized objective function, used in this study, is defined as:

$$\Phi(x) = F(x)(1 + 10 \sum_{i=1}^n c_i)^2 \quad (3.2)$$

where $\Phi(x)$ the penalized objective function $F(x)$ is the unconstrained objective function, c_i is the i^{th} constraint. In total, seven constraints were

used in this study, summarized in Table 3.3. To further elaborate, two types of constraints are considered in this study, a minimum constraint (min) and a maximum constraint (max). A minimum constraint requires the value not to be smaller than a predefined limit, as follows:

$$c = \frac{x_{min} - x}{x_{min}} \geq 0 \quad (3.3)$$

Furthermore, a maximum constraint (max) requires the value not to be larger than a predefined limit, as follows:

$$c = \frac{x - x_{max}}{x_{max}} \geq 0 \quad (3.4)$$

As previously mentioned, the solution vector x represents the section shape, design and girder arrangement required to fulfill the design requirements of a given bridge geometry. The definition of the solution vector is as follows:

$$x = \{L, H, D1, D2\} \quad (3.5)$$

A graphical representation of the solution vector is shown in Figure 3.3.

All optimization runs and iterations require the bridge girders to be analyzed and calculated. The dead loads are determined by adding all loads of permanent nature, such as the own weight of members, the deck slab weight, asphalt layers, road barriers, etc. As for the live load, it was determined using the Live Load Distribution Factor (LLDF) method as per the AASHTO LRFD code (2020). The analysis of internal loads was done assuming the bridge was simply supported.

3.3 Design Procedure

A single-span solid and voided slab bridge design process includes a number of critical steps to guarantee both structural integrity and performance.

The span length and necessary bridge width are first established to build the structural geometry. Next, the slab type and depth are chosen using span-to-depth ratios. The right materials are then selected, including the choice of mild steel and prestressing steel reinforcement, as well as the concrete strengths for the slabs and overlays. After that, gross section properties are computed, taking into consideration the slab's area and moment of inertia, which includes keyways and voids.

The process then moves on to load determination. Permanent loads are based on the unit weight of concrete which include the weight of prestressing strands, rebar and forms that remain inside the section, in addition to the barrier and wearing surface loads. Moreover, the vehicular live loads are computed in accordance with HL-93.

Unfactored bending moment and shear are then calculated using structural analysis for simply supported single-span bridges via the following equations:

$$Vx = w(0.5L - x) \quad (3.6)$$

$$Mx = 0.5wx(L - x) \quad (3.7)$$

Where:

$w = \text{uniform dead load}$

$x = \text{distance from the left end of the slab}$

$L = \text{span length between bearings}$

The live load distribution factor for moments (DFM) for an interior slab is governed by the larger value for one design lane versus two or more design lanes loaded are calculated as follows:

$$DFM = k \left(\frac{b}{33.3L} \right)^{0.5} \left(\frac{I}{J} \right)^{0.35} \quad (3.8)$$

$$DFM = k \left(\frac{b}{305 * 25.4} \right)^{0.6} \left(\frac{b}{12.0 L} \right)^{0.2} \left(\frac{I}{J} \right)^{0.06} \quad (3.9)$$

Provided the ranges are met:

$$889 \leq b \leq 1524 \quad b = \text{effective slab width (mm)}$$

$$6.1 \leq L \leq 36.6 \quad L = \text{span length (m)}$$

$$5 \leq N_b \leq 20 \quad N_b = \text{number of slabs}$$

Where:

$$k = 2.5(N_b^{-0.2}) \geq 1.5 \quad (3.10)$$

The necessary prestressing force and number of strands are then calculated. The main purpose in the case of pre-stressed girder design is to balance the tensile stresses due to self-weight, slab-weight, superimposed dead load and live load with the compression stresses produced by the strands. The tensile stresses due to the dead loads and live loads and balancing compression stresses are calculated as follows:

$$f_{bt} = \frac{M_{DC1} + M_{DC2} + M_{DW} + \alpha M_{LL}}{S_b} \quad (3.11)$$

$$f_{bc} = \frac{P_P}{A_g} + \frac{P_P * e}{S_b} \quad (3.12)$$

Where:

$$M_{DC1} = \text{unfactored moment due to slab self-weight.}$$

M_{DC2} = unfactored moment due to barrier weight.

M_{DW} = unfactored moment due to overlay weight.

S_b = section modulus for the bottom extreme fiber.

$\alpha = 1$ for service limit state I and 0.8 for service limit state III.

e : Distance between neutral axis of girder and center of geometry of small strands.

P_p : Prestressing force

A_g : Area of girder

f_{bt} : tensile stress on concrete (at bottom fiber)

f_{bc} : compression stress on concrete (at top fiber)

The final tensile stress at the bottom of the girder section should be less than fallow that is allowable tensile stress of concrete. Then total tensile stress f_{bt} required to be balanced with the compression stress f_{pr} produced by the pre-stressing strands. f_{bt} and f_{pr} are calculated as follows:

$$f_{btn} = f_{bt} - f_{allow} \quad (3.13)$$

$$f_{pr} = -f_{bc} = -\frac{P_p}{A_g} - \frac{P_p * e}{S_b} \quad (3.14)$$

$$f_{allow} = 0.5\sqrt{f_c} \quad (3.15)$$

$$P_p = \left(\frac{A_g * S_b}{S_b + A_g * e}\right) f_{btn} \quad (3.16)$$

$$P_{p-strand} = f_{PJ} * A_{strand} * \left(1 - \frac{loss\%}{100}\right) \quad (3.17)$$

Where:

A_{strand} : Single area of strand (mm^2)

f_{pj} : Jacking stress $0.75*f_{pu}$ (MPa)

Loss%: Pre-stressed loss

$$n_{strand} = \frac{P_p}{P_{p-strand}} \quad (3.18)$$

Prestress losses over the service life are then calculated. According to AASHTO standards (2020), total losses in prestressing strands are assumed to be the sum of immediate and long-term losses. Immediate losses for strands in PC slab are due to elastic shortening. Long-term losses are primarily due to concrete creep and shrinkage as well as steel relaxation (AASHTO, 2020).

$$\Delta f_{pT} = \Delta f_{pES} + \Delta f_{pLT} \quad (3.19)$$

where:

Δf_{pES} = change in stress due to elastic shortening loss.

Δf_{pLT} = losses due to long-term shrinkage and creep of concrete and relaxation of prestressing steel.

Calculation of elastic shortening loss is an alternative procedure since its formula includes f_{cgp} which is implicitly dependent on shortening loss. AASHTO LRFD (2020) uses the following equation to calculate elastic shortening in pre-tensioned members:

$$\Delta f_{pES} = \frac{E_p}{E_{ct}} * f_{cgp} \quad (3.20)$$

$$f_{cgp} = \frac{P_i}{A_g} + \frac{M_i * e}{I_g} - \frac{M_{gdl} * e}{I_g} \quad (3.21)$$

Where:

E_p = Elastic modulus of pre-stressing strands

E_{ct} = Elastic modulus of concrete just immediately after jacking

$P_i = 0.75 * f_{pu} * A_{strand} * (1 - \text{initial loss}\%)$

f_{pu} = ultimate strength of strand (MPa)

A_{strand} = Total area of strands (mm²)

P_i = initial prestressing force (N)

A_g = Area of girder (mm²)

I_g = Moment of inertia (mm⁴)

M_i = Moment due to eccentricity between neutral axis and center of gravity of strands ($M_i = P_i * e$) (N.mm)

M_{gdl} = Moment due to self-weight of girder (N.mm)

e = distance between neutral axis of girder and center of geometry of strands (mm)

For strand precast or pre-tensioned member, the long term (time dependent) loss Δf_{pLT} due to creep of concrete, shrinkage of concrete, and relaxation of steel shall be estimated using the following formula:

$$\Delta f_{pLT} = 10 * \frac{f_{pi} * A_{ps}}{A_g} * \gamma_h * \gamma_{st} + 83\gamma_h\gamma_{st} + \Delta f_{pr} \quad (3.22)$$

$$\gamma_{st} = \frac{35}{(7 + f_{ci})} \quad (3.23)$$

Where:

f_{pi} = pre-stressing steel stress just before transfer (MPa)

A_{ps} = area of strands (mm²)

A_g = area of girder (mm²)

$\gamma_h = 1.7 - 0.01H$ where H is percentage of humidity. (80%)

γ_{st} = correction factor for specified concrete strength at pre-stress transfer

f_{ci} = concrete strength at transfer of prestressing (MPa) $f_{ci} = 0.8 * f_{cu}$

Δf_{pr} = prestressing steel relaxation (17 MPa) for low relaxation strands

In order to ensure that serviceability requirements like deflection and cracking are fulfilled and to check structural strength under calculated loads, the design must then meet both service and strength limit states.

The girder stresses calculation and limits can be determined using the following formulas:

$$f_{top} = -\frac{P_p}{A_g} + \frac{P_p * e}{S_t} - \frac{(M_{DC1} + M_{DC2} + M_{DW})}{S_t} \quad (3.24)$$

$$f_{top} < 0.6f'_c \quad (3.25)$$

$$f_{bot} = -\frac{P_p}{A_g} - \frac{P_p * e}{S_t} + \frac{(M_{DC1} + M_{DC2} + M_{DW})}{S_t} \quad (3.26)$$

$$f_{bottom} < 0.5\sqrt{f'_c} \quad (3.27)$$

Last but not least, the factored flexural resistance is checked at the ultimate limit state (Strength I) as follows:

$$\phi M_n > M_u \quad (3.28)$$

For pre-stressed concrete section, $\phi=1.0$ (Article 5.5.4.2) (AASHTO, 2020)

$$M_u = 1.25 * M_{DC1} + 1.5 * M_{DC2} + 1.75 * M_{DW} \quad (3.29)$$

$$M_n = A_{ps} * f_{ps} \left(d_p - \frac{a}{2} \right) + A_s * f_s \left(d_s - \frac{a}{2} \right) \quad (3.30)$$

$$M_n = A_{ps} * f_{ps} \left(d_p - \frac{a}{2} \right) + A_s * f_s \left(d_s - \frac{a}{2} \right) - A_{s'} * f_{s'} \left(d_{s'} - \frac{a}{2} \right) + 0.85 f_{c'} (b - b_w) h_f \left(\frac{a}{2} - \frac{h_f}{2} \right) \quad (3.31)$$

In addition, minimum number of tension reinforcement should be provided to prevent brittle failure. Accordingly, the following requirements should be satisfied:

A factored flexural resistance, M_R , at least equal to the lesser of:

- 1.2 times the cracking moment M_{cr}
- 1.33 times the factored moment capacity, M_u .

Where:

$$M_{cr} = \gamma_3 \left[(\gamma_1 f_r + \gamma_2 f_{cpe}) S_c - M_{dnc} \left(\frac{S_c}{S_{nc}} - 1 \right) \right] \quad (3.32)$$

where:

f_r = modulus of rupture of concrete specified in Article 5.4.2.6 (AASHTO, 2020)

f_{cpe} = compressive stress in concrete due to effective prestress forces only (after allowance for all prestress losses), at extreme fiber of section where tensile stress is caused by externally applied load.

M_{dnc} = total unfactored dead load moment acting on the non-composite section.

S_c = section modulus for the extreme fiber of the composite section where tensile is caused by externally applied loads.

γ_1 = flexural cracking variability factor 1.6 for other than PC segmental structures.

γ_2 = prestress variability factor 1.1 for bonded tendons.

γ_3 = ratio of specified minimum yield strength to ultimate tensile strength of reinforcement 1.0 for prestressed concrete structures.

This comprehensive approach guarantees that the bridge meets all necessary safety and performance standards throughout its operational lifespan.

3.4. Research Tools

Computer programming tools have been developed and utilized in the formulation of this research study. As explained in the methodology flowchart in Figure 3.1. Microsoft Excel is the starting point for performing a full bridge design example using AASHTO voided slab (Type SIV). In addition, MATLAB software is utilized for converting the design spreadsheet into a code and further to combine it with the modified adaptive harmony search algorithm (MARSHAL) code which can be found in Appendix A.

3.4.1. The modified adaptive harmony search algorithm (MARSHAL) and search space reduction

This study makes use of the MARSHAL, a modified form of the Harmony Search algorithm that was introduced by Jahjouh and Nackenhorst (2016) to overcome the weaknesses of the metaheuristic optimization algorithms.

The process of the MARSHAL is shown in Figure 3.4. The updated approach makes use of an adaptive scheme for the control parameters HMCR and PAR.

Additionally, MARSHAL employs an adaptive search space reduction strategy, which will be used to some extent in this study. After a predetermined number of rounds, the search space is reduced to avert an early convergence.

The adaptive reduction process is initiated after a specified number of iterations, usually about 30% of the total to avoid early convergence and promote initial exploration. The bounds of the search space for each design variable x_i are adjusted according to the statistical distribution of values maintained in the Harmony Memory:

$$x_{i,lb} = \mu(x_i) - 3\sigma(x_i) \quad (3.33)$$

$$x_{i,ub} = \mu(x_i) + 3\sigma(x_i) \quad (3.34)$$

Where:

- $x_{i,lb}$ and $x_{i,ub}$ represent the updated lower and upper bounds for the variable x_i
- $\mu(x_i)$ is the average of all (x_i) values in the harmony memory,
- $\sigma(x_i)$ denotes the standard deviation of (x_i) in the harmony memory.

This statistical adjustment of bounds guarantees that the algorithm maintains diversity while concentrating on a more refined search area around the most promising solutions discovered thus far.

The performance of the MARSHAL algorithm in optimizing the voided slab beam geometry can be further illustrated using the convergence graph shown in Figure 3.5. This graph represents the evolution of the penalized objective function value over successive iterations for a typical optimization run. At the start of the optimization process (0–10,000 iterations), there is a steep drop in the objective function, indicating rapid exploration and initial improvement. After that, the objective function stabilizes with minimal changes, indicating that the algorithm has converged to a near-optimal solution.

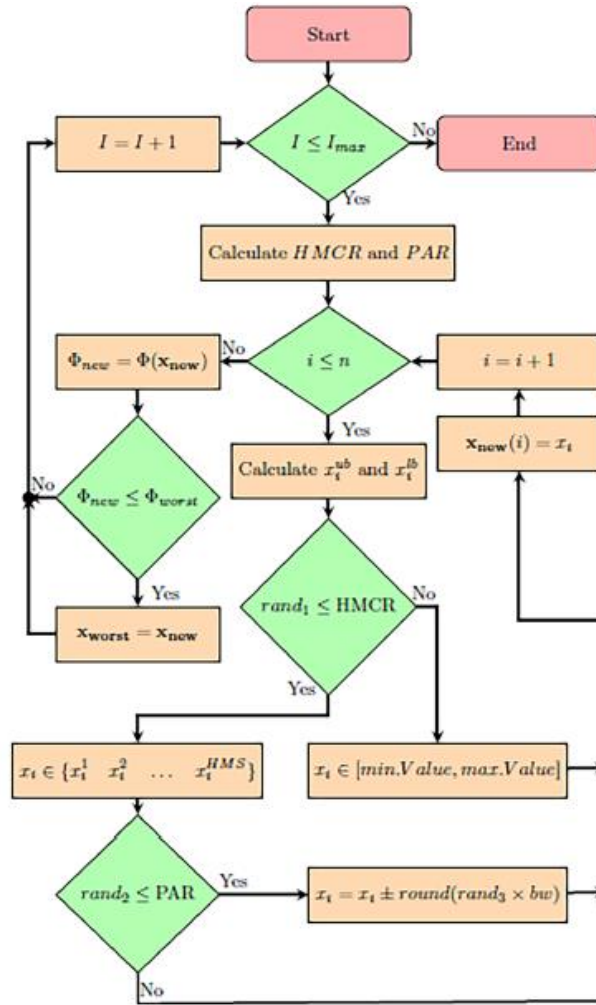


Figure 3.4 MARSHAL's flowchart (Jahjough MM, Nackenhorst U, 2016)

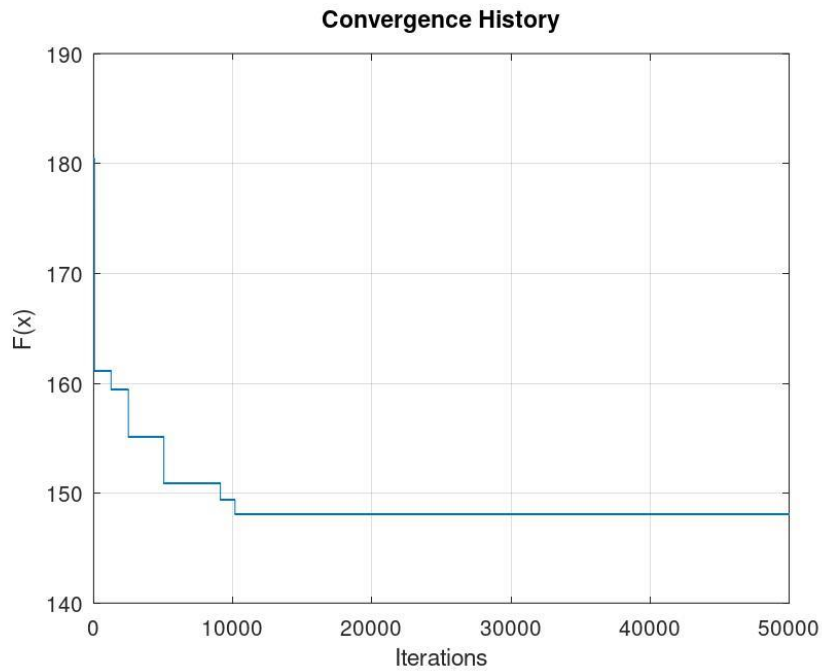


Figure 3.5 Convergence history of the MARSHAL algorithm during the optimization of a voided slab beam section

3.5. Numerical Example

The first excel sheet mentioned in the methodology flowchart is conducted to convert the design of the voided slab bridge examples represented in Bridge Design Practice (2022) to SI friendly spreadsheet. Design parameters of the bridge for this example are given in Table 3.4. The unit prices of the productions are taken according to the Kingdom of Bahrain unit price criteria and given in Table 3.5.

Table 3.4 Design parameters of the bridge.

General properties of the bridge	Total length of bridge	14.63 m
	Total width of bridge	13.41 m
Material Properties	Precast concrete	C35
	Reinforcing steel	S460
	Ultimate stress of PS	1860
	Modulus of elasticity of PS	197 MPa
	Diameter of PS strands	15.24 mm
Loads	Barrier load	3.65 kN/m
	Asphalt thickness	75 mm
	vehicular live load	HL-93

Table 3.5 Unit prices.

Production	Unit	Unit price (BHD)
Concrete	m ³	33
Prestressing Steel (PS)	ton	360

Several numerical examples are then obtained from the optimization runs by changing the problem input “Span length” values from 9 m to 24 m and setting the bridge width value to 13.45 m with reference to Shaikh Khalifa bin Salman & Shaikh Isa bin Salman Highway new left turn flyover width (2 lanes bridge) to come up with multiple optimum sections and further to

prepare design charts with the best section in terms of the minimum cost criteria.

Chapter 4

Empirical Findings and Analysis

Chapter 4

Empirical Findings and Analysis

4.1. Introduction

The goal of the design methodology is to select values of critical parameters of the solid and voided slabs in order to construct the most economical and practical beams. This goal is achieved by generating realistic initial populations where critical parameters take values within the allowable limits to demonstrate the suitability of the proposed method. Naturally, the number of active numbers for the voided slab resulted in an apparent volume grade reduction. Results of the proposed method demonstrate the ability of the modified harmony search (MARSHAL) algorithm to optimize the selected solid section shape and prestress force appropriately. The proposed method outperforms traditional optimization methods by using fewer parameters and demonstrating better search ability and convergence behavior. The designed voided slabs are more economical and practical than the traditional AASHTO/PCI sections. Therefore, the described optimization method provides a useful guide to the design of practical precast prestressed voided slabs and can be further extended to the design of other prestressed concrete cross sections.

4.2. Study Results

The first optimization run (OPT sol.1) is conducted with setting the bridge span length and total bridge width, the same as the numerical example which is indicated in Table 3.4. The results are compared with AASHTO SIV-48 product and verified using the same design excel sheet. The

subsequent optimization runs (OPT sol.2 - OPT sol.17) are obtained by changing the problem input “Span length” values from 9 m to 24 m and setting the bridge width value to 13.45 m.

4.2.1. Optimization runs section properties and design output

The geometric properties of optimum solution slab beam sections are presented in this section. There are seventeen optimum sections, and the geometric configuration is illustrated in Figure 3.2. The dimensions of the optimum slab beams are presented in Table 4.1. The geometric properties of the slab beams are then calculated using these dimensions and presented in Table 4.2 where Area is the gross cross-sectional area, y_b is the centroid axis from bottom, Inertia is the moment of inertia, Torsional is the torsional moment of inertia and S_b is section modulus from the bottom. In addition, the optimization runs are shown in Table 4.3. Where (ns) is the total number of strands and Cost being the section cost calculated per meter length. In addition, the optimization runs solution convergence graphs are shown in Appendix B.

Table 4.1 Optimum Sections dimensions (mm).

Type	L	H	L1	L2	No. of Voids	D1	D2
OPT sol.1	1460	520	285	445	3	370	370
OPT sol.2	1070	390	220	315	3	240	240
OPT sol.3	1070	390	220	315	3	240	240
OPT sol.4	1120	420	235	325	3	270	230
OPT sol.5	1100	400	225	325	3	250	250
OPT sol.6	1310	470	260	395	3	320	320

Type	L	H	L1	L2	No. of Voids	D1	D2
OPT sol.7	1220	430	270	370	3	280	280
OPT sol.8	1430	510	280	435	3	360	360
OPT sol.9	1430	510	280	435	3	360	360
OPT sol.10	1490	530	290	455	3	380	380
OPT sol.11	1490	530	290	455	3	380	380
OPT sol.12	1220	620	335	275	2	470	0
OPT sol.13	1370	690	370	315	2	540	0
OPT sol.14	1340	680	365	305	2	530	0
OPT sol.15	1340	680	365	305	2	530	0
OPT sol.16	1480	750	400	340	2	600	0
OPT sol.17	1420	720	385	325	2	570	0

Table 4.2 Optimum sections geometric properties.

Type	Area (mm ²)	y _b (mm)	Inertia (mm ⁴)	Torsional (mm ⁴)	S _b (mm ³)
OPT sol.1	436637	260.0	14347376778	63335840845	55182218
OPT sol.2	281583	195.0	4800697011	32738792385	24618959
OPT sol.3	281583	195.0	4800697011	32738792385	24618959
OPT sol.4	314341	210.0	6255772389	39018028251	29789392
OPT sol.5	292738	200.0	5291423871	34696281956	26457119
OPT sol.6	374426	235.0	9789855212	50191048483	41658958
OPT sol.7	339874	215.0	7178055991	46473645102	33386307
OPT sol.8	423937	255.0	13334138772	60559338419	52290740
OPT sol.9	423937	255.0	13334138772	60559338419	52290740

Type	Area (mm ²)	y _b (mm)	Inertia (mm ⁴)	Torsional (mm ⁴)	S _b (mm ³)
OPT sol.10	449466	265.0	15414944612	66188653752	58169602
OPT sol.11	449466	265.0	15414944612	66188653752	58169602
OPT sol.12	409411	310.0	19439397714	36132301009	62707735
OPT sol.13	487256	345.0	29156921793	48331066054	84512817
OPT sol.14	469963	340.0	27365111726	44565481732	80485623
OPT sol.15	469963	340.0	27365111726	44565481732	80485623
OPT sol.16	544513	375.0	39307799753	55910749396	104820799
OPT sol.17	512048	360.0	33804350252	50840643649	93900973

Table 4.3 Optimization runs input and output values.

Type	Input		Output						
	Span Length (m)	Width (m)	L (mm)	H (mm)	No. of Voids	D1 (mm)	D2 (mm)	ns	Cost (BHD/m)
SIV-48	14.600	13.400	1219	533	3	305	254	12	219.277
OPT sol.1	14.600	13.400	1460	520	3	370	370	14	186.200
OPT sol.2	9.000	13.450	1070	390	3	240	240	6	148.100
OPT sol.3	10.000	13.450	1070	390	3	240	240	8	158.970
OPT sol.4	11.000	13.450	1120	420	3	270	230	8	164.710
OPT sol.5	12.000	13.450	1100	400	3	250	250	10	168.690
OPT sol.6	13.000	13.450	1310	470	3	320	320	12	178.190
OPT sol.7	14.000	13.450	1220	430	3	280	280	14	188.180
OPT sol.8	15.000	13.450	1430	510	3	360	360	14	186.770
OPT sol.9	16.000	13.450	1430	510	3	360	360	16	194.890
OPT sol.10	17.000	13.450	1490	530	3	380	380	18	202.350
OPT sol.11	18.000	13.450	1490	530	3	380	380	18	226.120

	Input		Output						
Type	Span Length (m)	Width (m)	L (mm)	H (mm)	No. of Voids	D1 (mm)	D2 (mm)	ns	Cost (BHD/m)
OPT sol.12	19.000	13.450	1220	620	2	470	0	16	222.580
OPT sol.13	20.000	13.450	1370	690	2	540	0	16	223.680
OPT sol.14	21.000	13.450	1340	680	2	530	0	18	231.370
OPT sol.15	22.000	13.450	1340	680	2	530	0	20	240.020
OPT sol.16	23.000	13.450	1480	750	2	600	0	22	247.550
OPT sol.17	24.000	13.450	1420	720	2	570	0	24	256.180

4.3. Study Analysis

The optimum solution sections are then evaluated and compared with the AASHTO/PCI sections in terms of bridge span length, number of strands and the section cost per meter using the developed design charts code in Appendix C.

The design output tables for the AASHTO/PCI sections and the Optimum sections are presented in Appendix D. However, the design charts which are obtained from the design runs are shown as follows:

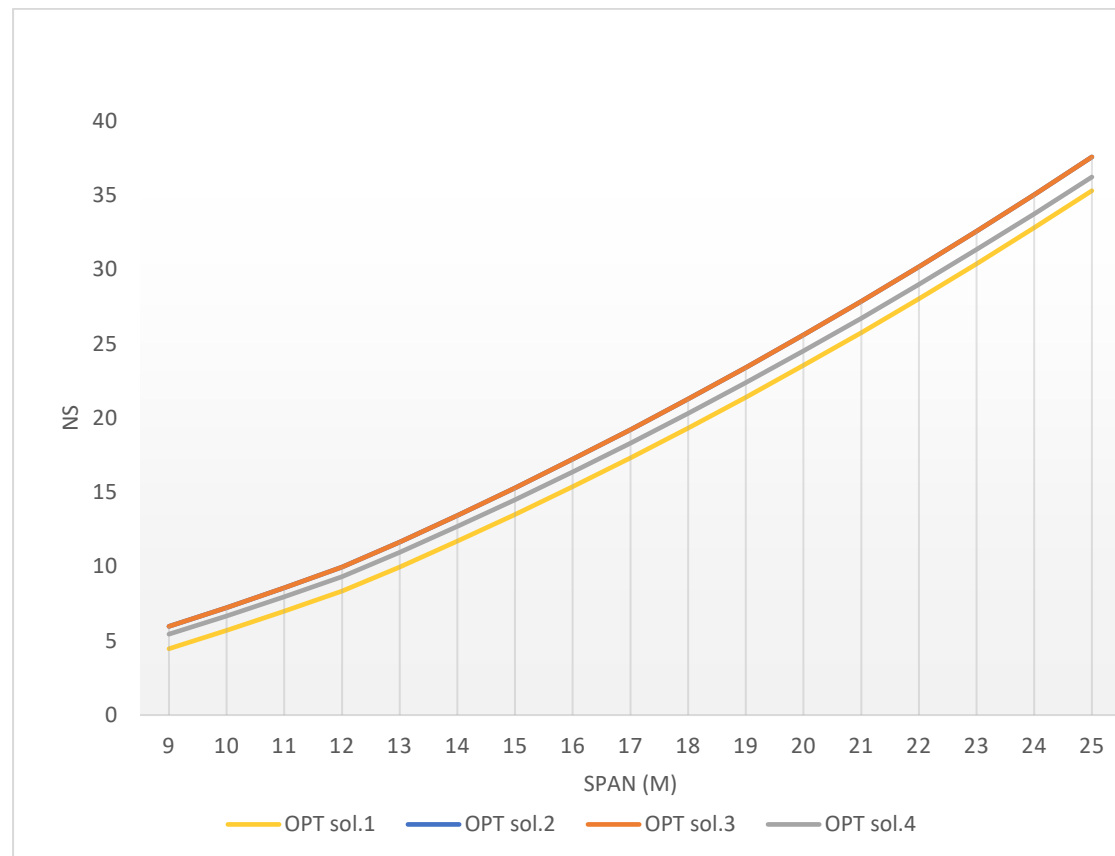


Figure 4.1 Span length vs Number of Strands for Optimum Sections (1-4).

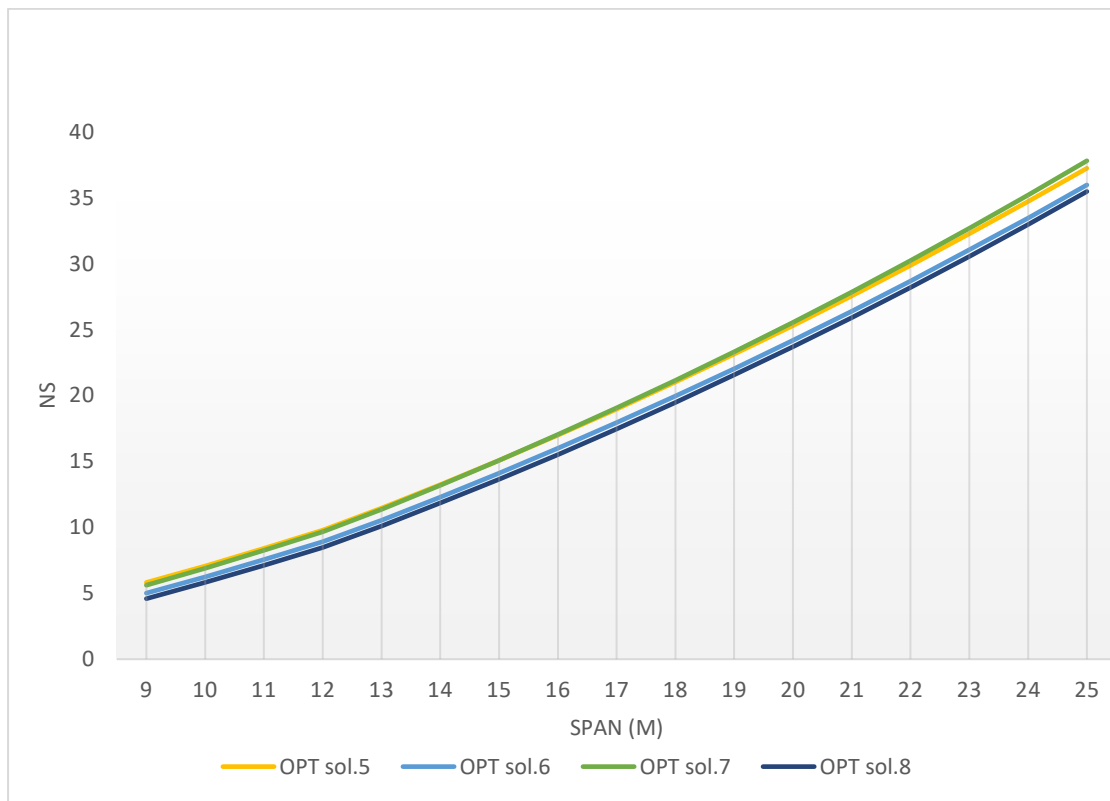


Figure 4.2 Span length vs Number of Strands for Optimum Sections (5-8).

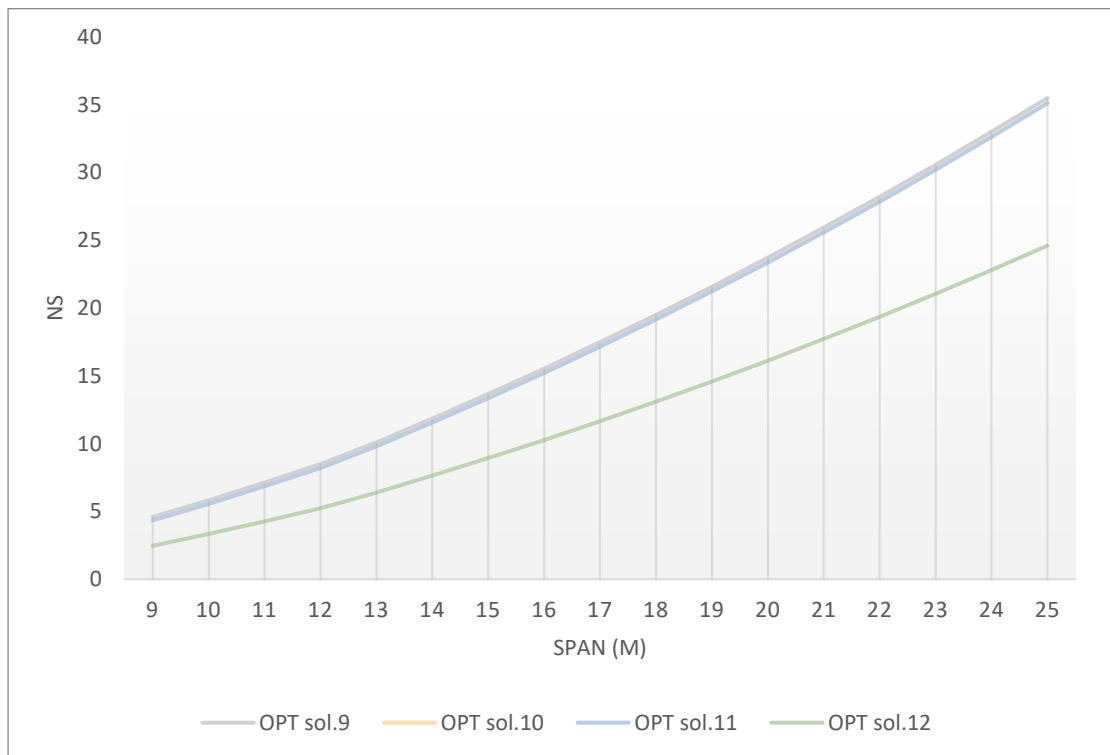


Figure 4.3 Span length vs Number of Strands for Optimum Sections (9-12).

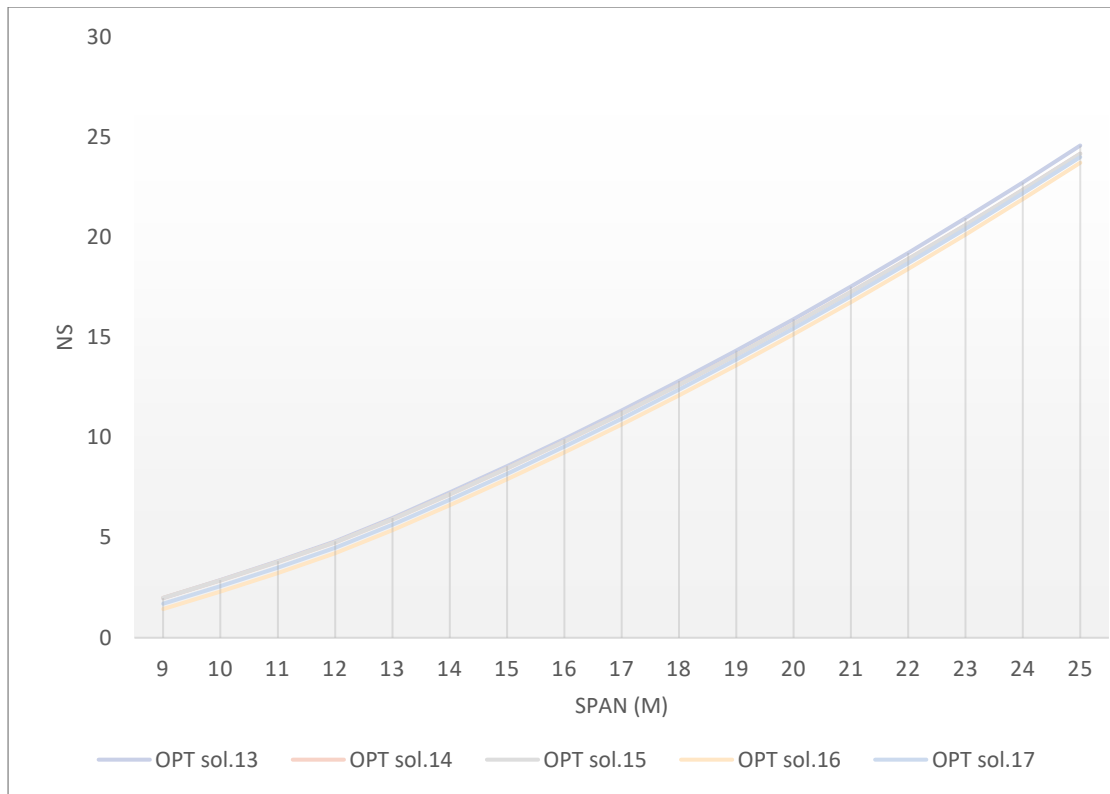


Figure 4.4 Span length vs Number of Strands for Optimum Sections (13-17).

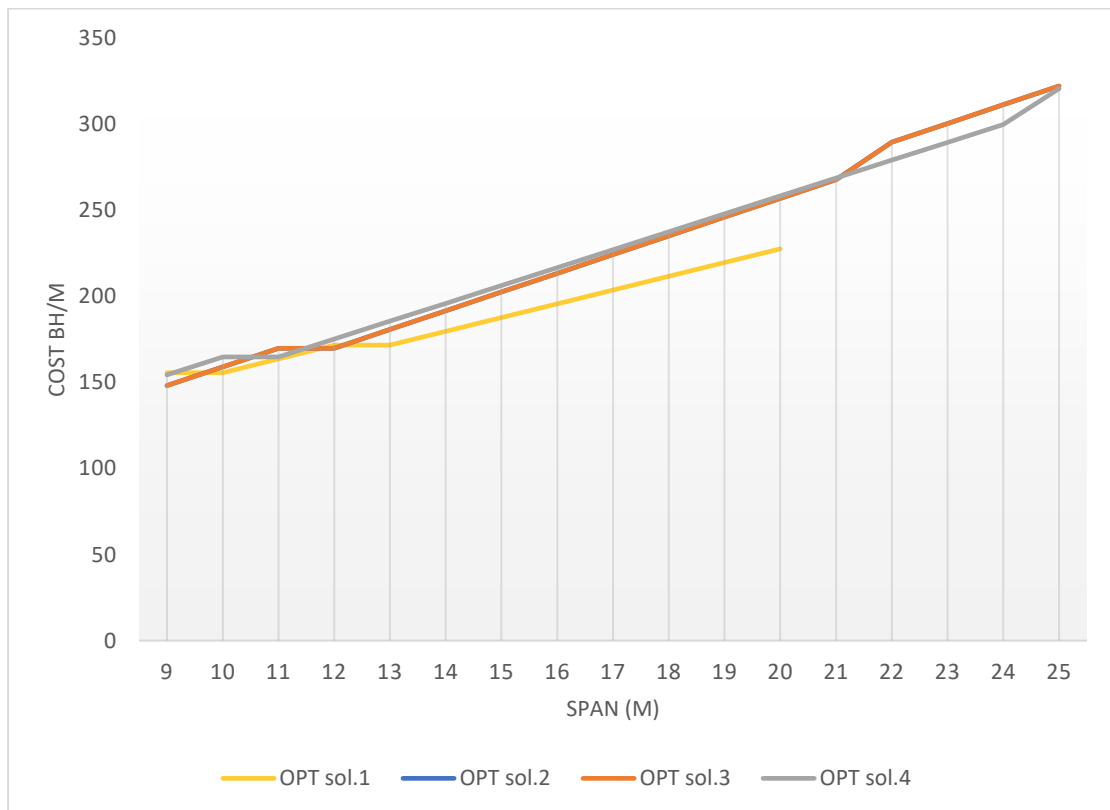


Figure 4.5 Span length vs cost for Optimum Sections (1-4).

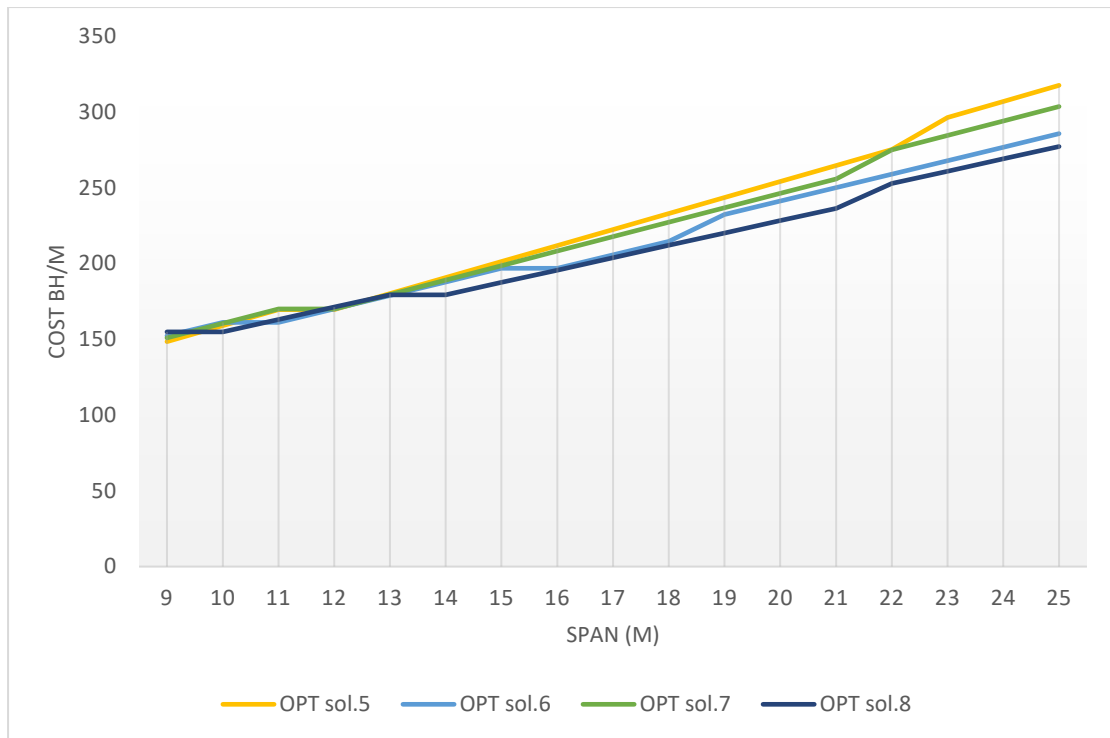


Figure 4.6 Span length vs Cost for Optimum Sections (5-8).

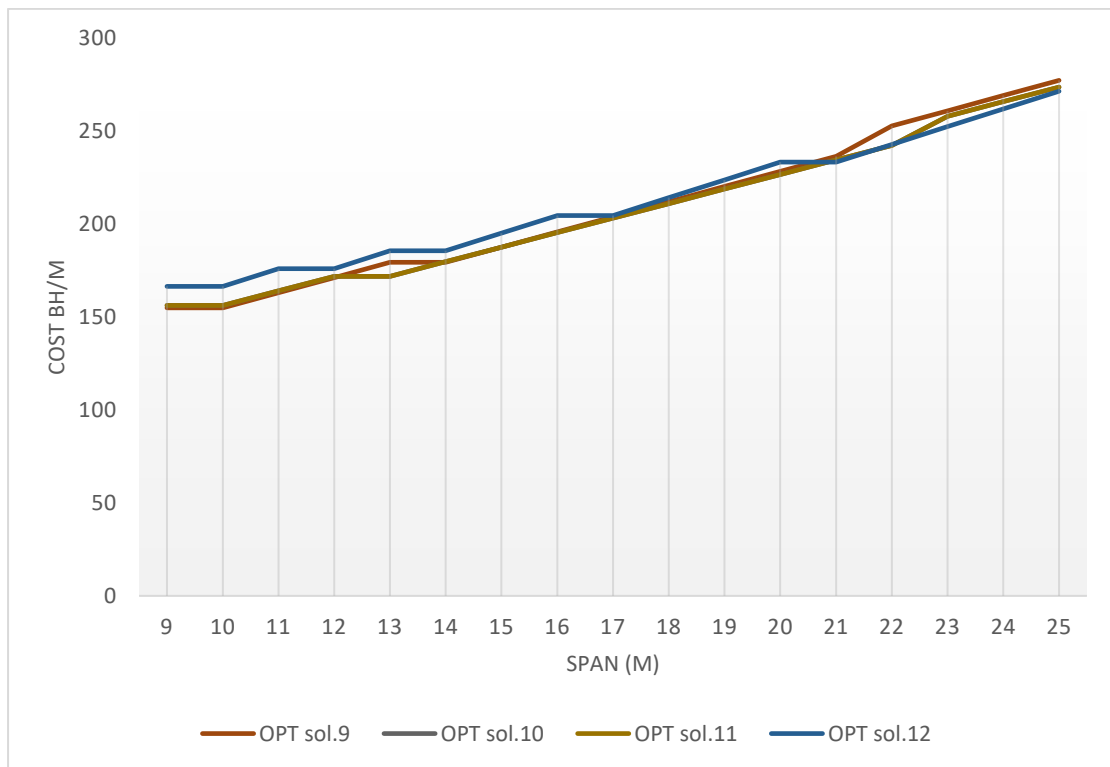


Figure 4.7 Span length vs Cost for Optimum Sections (9-12).

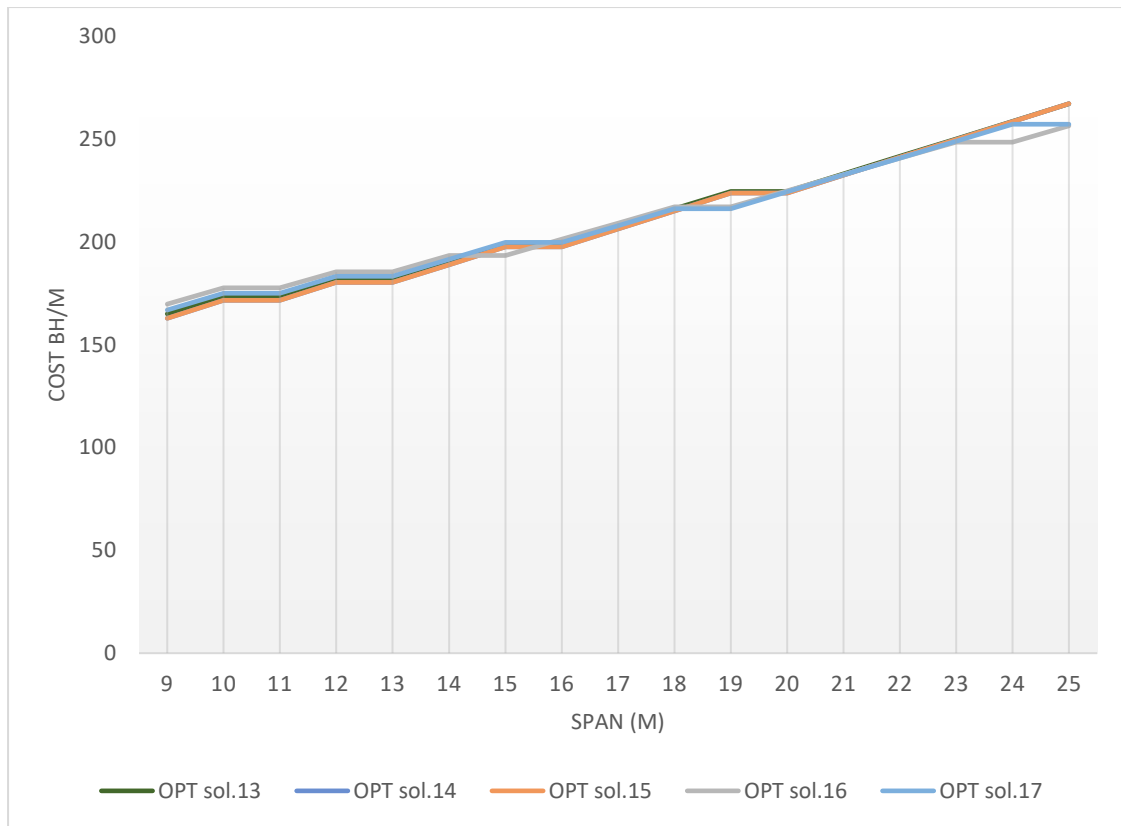


Figure 4.8 Span length vs cost for Optimum Sections (13-17).

Figures 4.1-4.4 show the required number of strands versus the span length for the optimum sections, and Figures 4.5-4.8 show the cost calculated per meter length for the optimum sections versus the span length. Based on the cost criteria, it is noticed that the best optimum sections are OPT sol.1, OPT sol.10, OPT sol.14, and OPT sol.16.

Figures 4.9-4.10 show the required number of strands versus the span length for the AASHTO/PCI sections. In addition, the best optimum sections are compared with the AASHTO/PCI sections in Figures 4.11-4.12 in terms of cost.

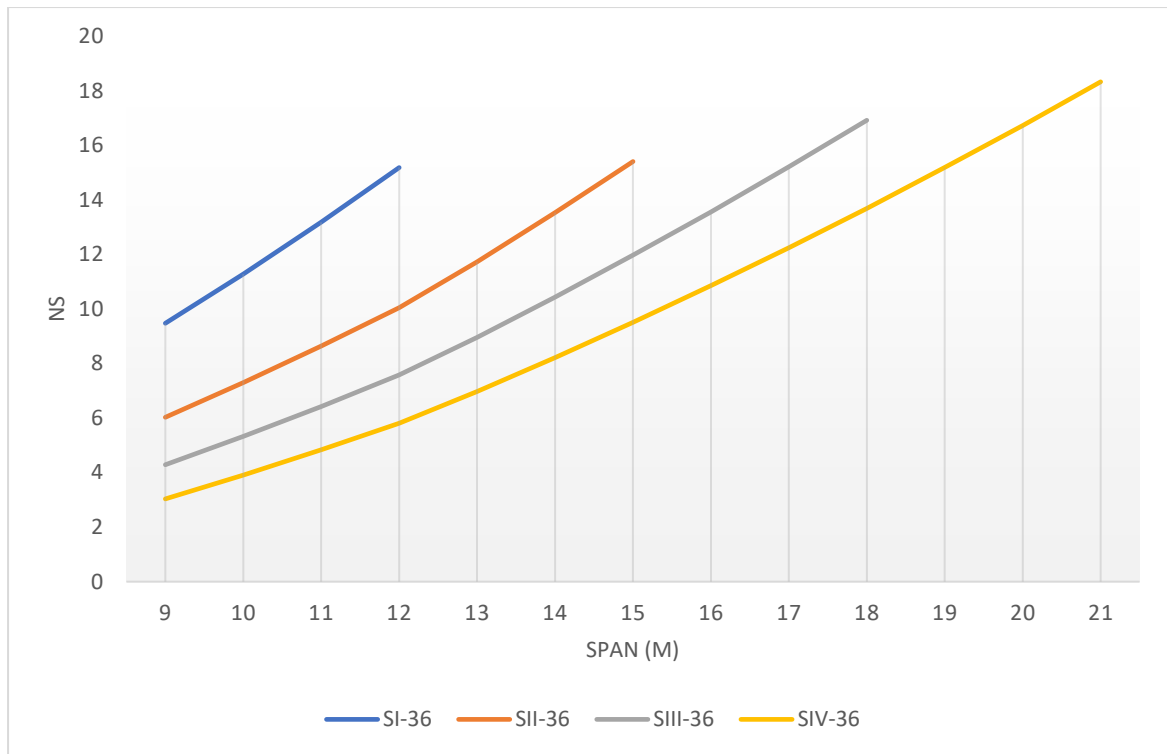


Figure 4.9 Span length vs Number of Strands for AASHTO (36) sections.

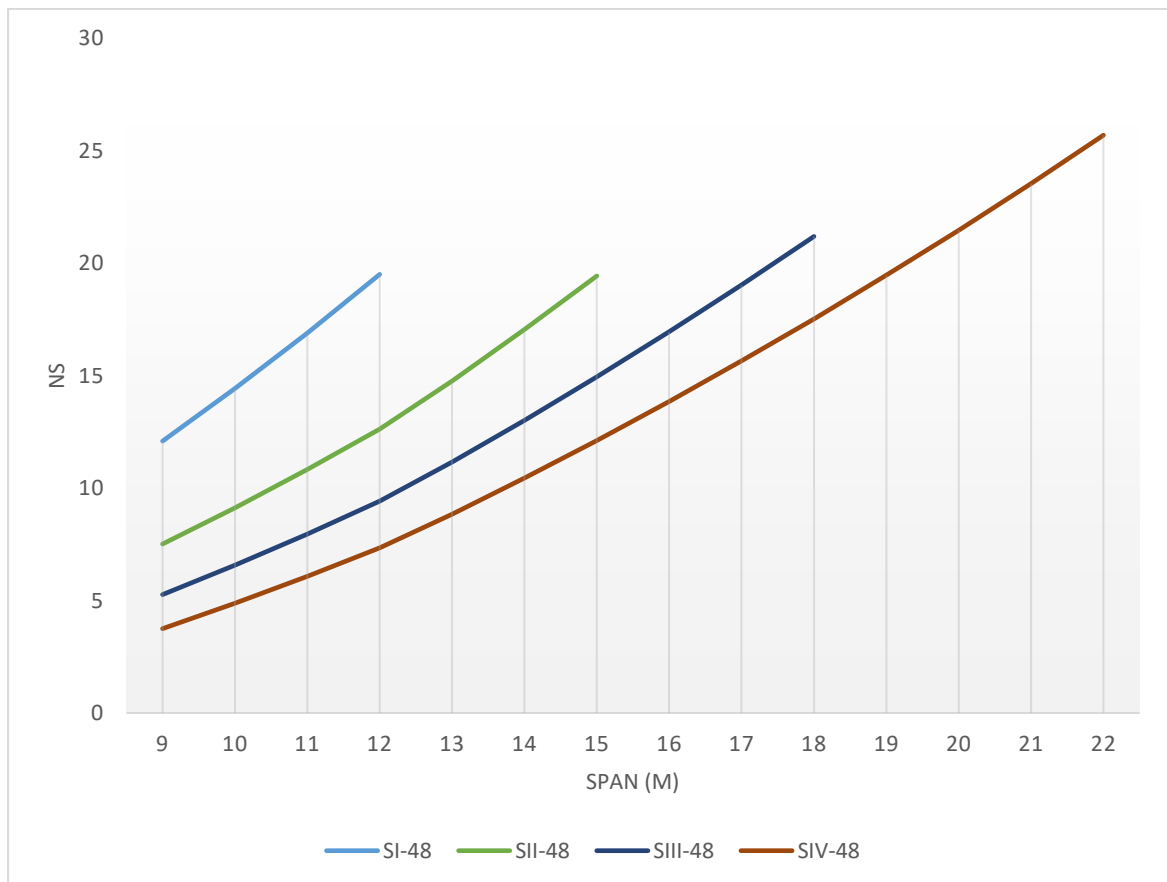


Figure 4.10 Span length vs Number of Strands for AASHTO (48) sections.

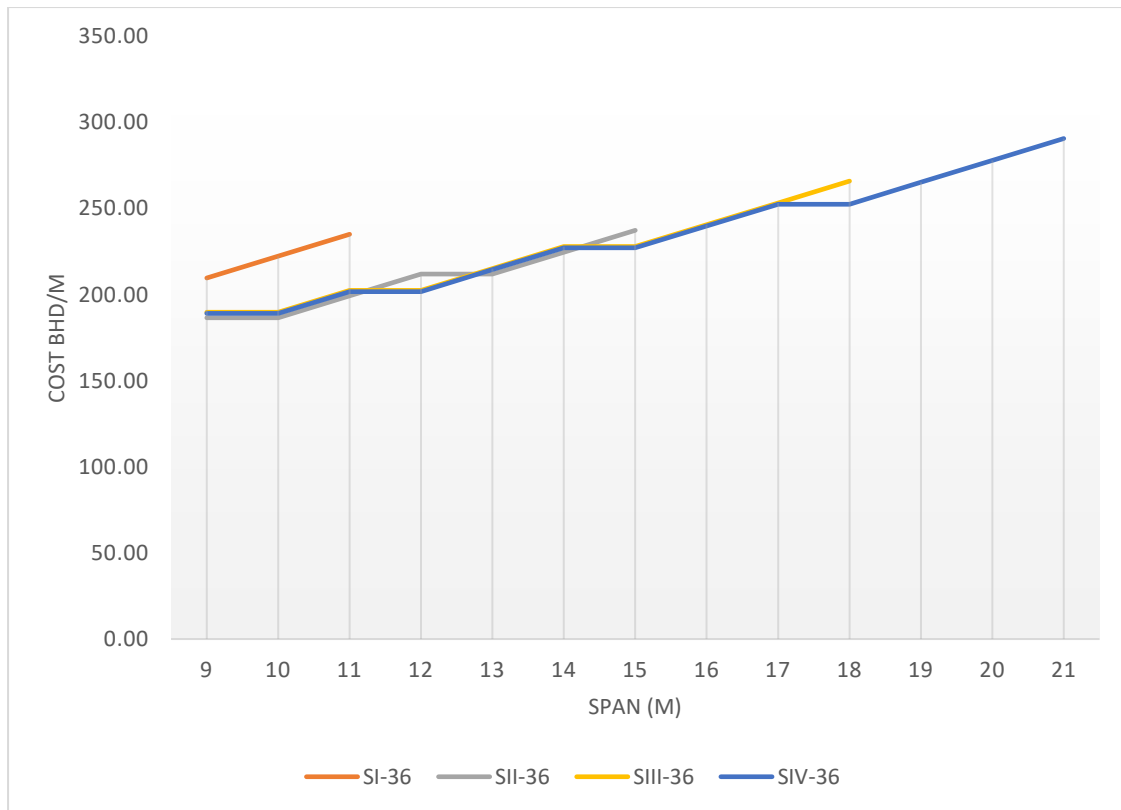


Figure 4.11 Span length vs cost for AASHTO (36) sections.

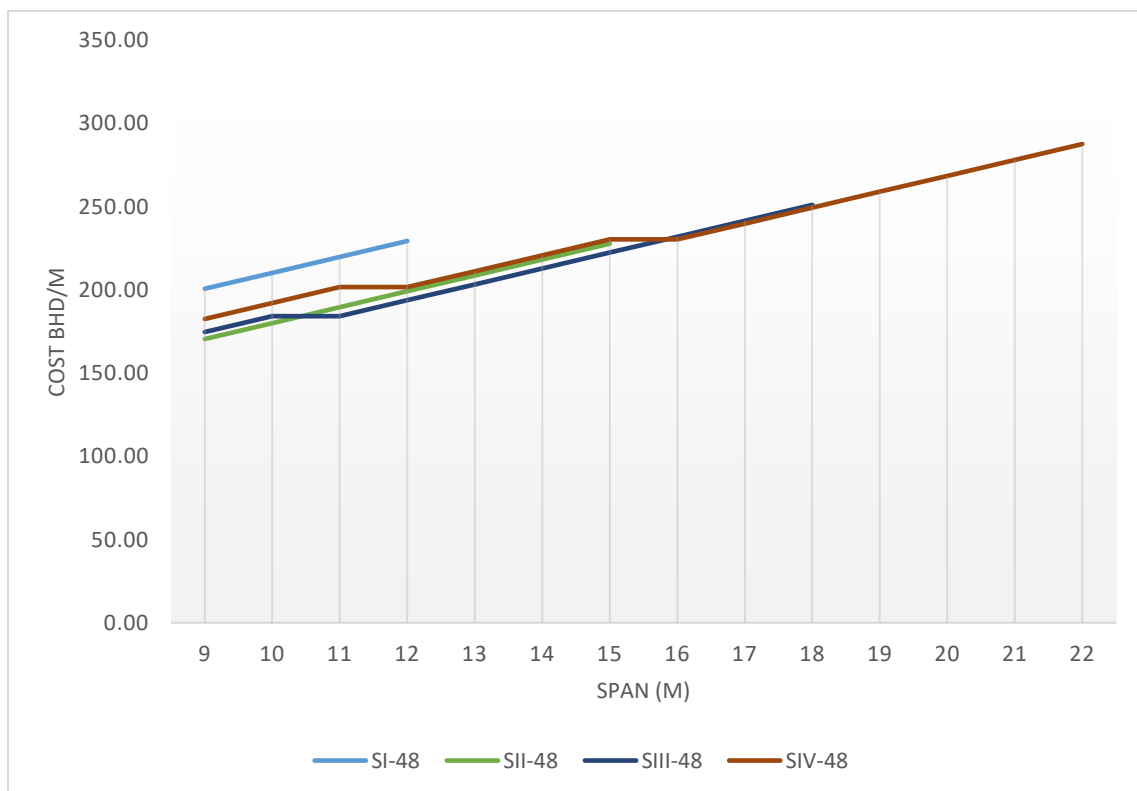


Figure 4.12 Span length vs cost for AASHTO (48) sections.

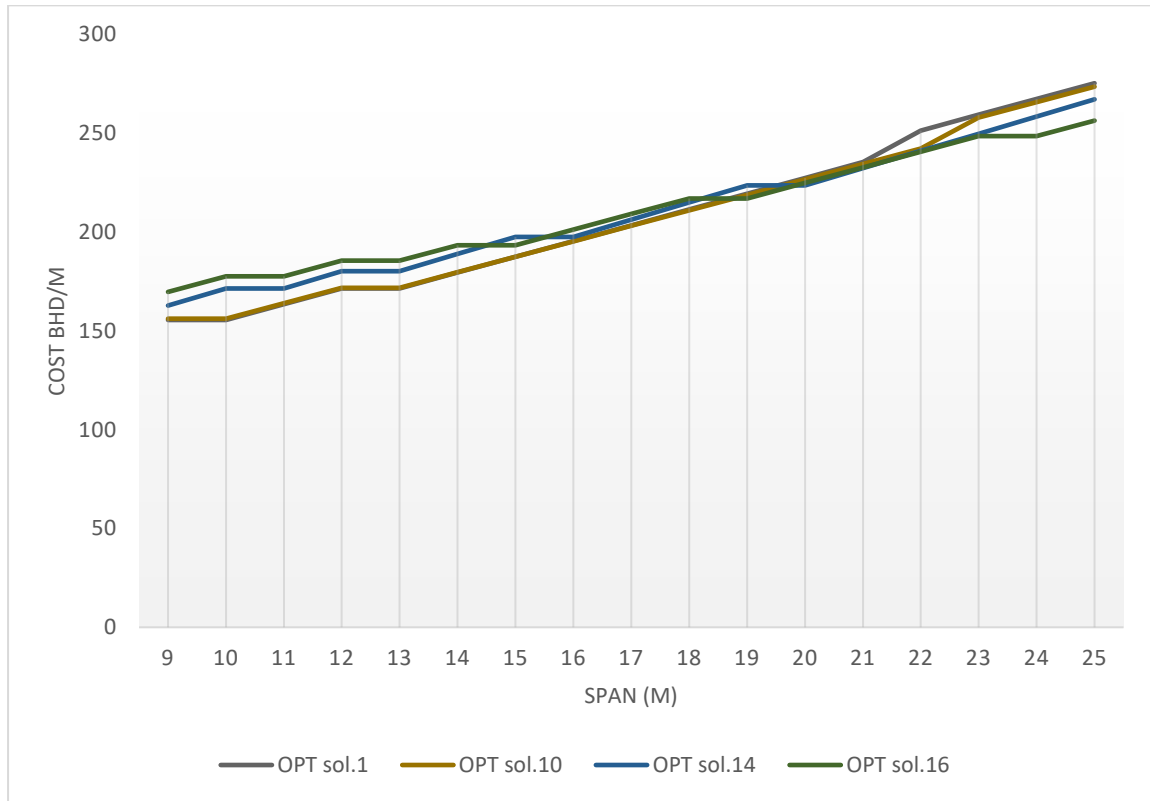


Figure 4.13 Span length vs cost for best optimum solution sections.

4.4. Results Discussion

The optimum sections show a relatively high-cost reduction compared to the best AASHTO/PCI sections relative to span lengths. In general, the voided slab beams are dominant in this criterion taking into consideration the reduced material cost and accordingly reduction in the applied self-weight loadings. Table 4.4 and Figure 4.14 show the most suitable optimum section for each bridge span length value, the optimum section cost, minimum AASHTO/PCI section cost, and the cost difference percentage between the optimum sections and AASHTO/PCI sections. It was found that the price difference ranges between 12% up to 21%.

Table 4.4 Best optimum section vs best AASHTO/PCI section and cost reduction % relative to bridge span length.

Span length	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Optimum Section	OPT sol.2	OPT sol.8	OPT sol.6	OPT sol.5	OPT sol.1	OPT sol.8	OPT sol.1	OPT sol.10	OPT sol.10	OPT sol.10	OPT sol.17	OPT sol.14	OPT sol.14	OPT sol.16	OPT sol.16	OPT sol.16
min. Opt. Sec. cost (BHD/m)	148	155	161	170	172	179	188	195	203	211	216	224	232	241	249	249
min PCI sec. cost (BHD/m)	171	180	184	194	203	213	222	230	240	249	259	26	278	288	297	316
Price difference %	13%	14%	12%	12%	16%	16%	16%	15%	15%	15%	17%	17%	16%	16%	16%	21%

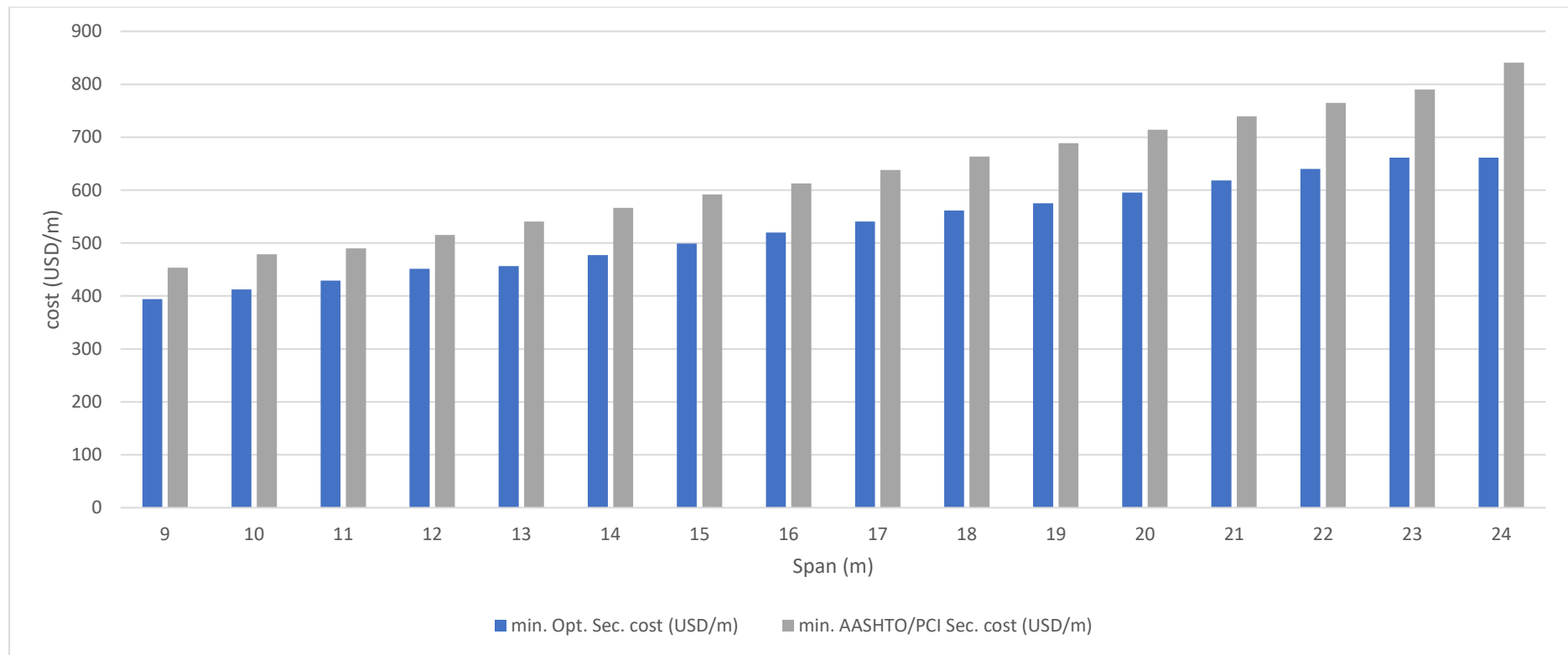


Figure 4.14 AASHTO/PCI sections vs OPT sections cost comparison.

Chapter 5

Conclusion and Recommendations

Chapter 5

Conclusion and Recommendations

5.1. Introduction

In the present study, a philosophy is proposed by taking advantage of the simplicity and bi-criterion nature of different objectives. The employed harmony search algorithm is proposed to be a powerful generic optimization tool for a variety of practical and relevant civil and geotechnical engineering applications. The required information regarding the problem addressed and value ranges for design parameters would enable civil engineers to model the optimization problem. The corporate efforts of research and development can develop decision aids for designers. Rapid advances in computer hardware development often permit conducting complex or multiple design scenarios within the reach of conventional working conditions. Abundant civil, geotechnical, environmental, and structural engineering applications in connection with the developed models facilitate the spreading usage and update of the provided optimizations.

5.2. Summary and Empirical Results

This paper presents a study on the optimization of seventeen prestressed concrete voided and solid slab bridge beams using the Modified Harmony Search (MARSHAL) algorithm. The optimized beams and their respective structural analysis showed superior structural performance characteristics in terms of economy and strength parameters, among which the voided slab bridge beam showed great promise.

The optimization code runs resulted in 17 optimum sections, which are then compared with each other based on the minimum cost criteria for each span length. The results are presented in Table 4.4.

Bridge with span length between 9-12 m showed different best optimum solutions, the same are compared with the best AASHTO/PCI section with 12-14 % cost reduction. Bridge with span length 13-15 m shows that optimum sections 1 and 8 are the best with 16 % cost reduction. Moreover, optimum section 10 is the best for span length 16-18 m with 15% cost reduction. While bridge with span length 19-21 m optimum sections 17 and 14 are the best with 16-17 % cost reduction. Lastly, optimum section 16 is the best for bridges with larger span lengths from 22-25 m. Therefore, the table shows that the most repeated sections are optimum sections 1, 10, 14, and 16 which are expressed as the best optimum sections.

On the other hand, Figure 4.14 represents a clear cost comparison between the best optimum sections compared with the best AASHTO/PCI section. Bridges with span length 9-12 m length shows that the cost reduction is the least compared with larger span lengths reaching the peak in the cost reduction for bridges with span length 24-25 m.

The findings of the present study illustrate that the presented MARSHAL can decide on a safe and economic design of prestressed concrete slab beams that fulfill a given set of the (AASHTO, 2020) code's governing restrictions.

5.3. Future Research

It can be stated that the modified harmony search algorithm has several advantages over its reason for not only being able to obtain good quality solutions but also having a small number of control inputs, thus making it easier to modify or to adapt to a certain engineering application. However,

it is proposed to use the proposed algorithm to optimize bridges with a larger number of design variables and constraints to test its potential solutions. It is strongly recommended to carry out real world case studies in order to strengthen the practical implementation of the optimal designs and confirm the study's conclusions. These case studies ought to concentrate on current bridges that make use of AASHTO solid and voided slab beams so that the effectiveness of the optimized designs under real-world circumstances may be evaluated. Real-world case studies offer a great chance to improve the research by adding empirical support to the conclusions and increasing the work's applicability. By advancing bridge engineering techniques, this strategy will eventually boost structural integrity and safety in the industry.

References

- Tyrrell, H. G. (1911). *History of bridge engineering*, Retrieved from <https://archive.org/details/historyofbridgee00tyrrrich/page/481/mod e/1up>
- Dinges, T. (2009). *The history of prestressed concrete: 1888 to 1963*. Kansas State University.
- Ding, Y., & Ning, X. (2023). *Prestressed concrete*. In *Reinforced concrete: Basic theory and standards*. (pp. 355–401). Springer, Singapore.
- AASHTO. (2020). *AASHTO LRFD Bridge Design Specifications (9th ed.)*. Washington: American Association of State Highway and Transportation Officials.
- Zhang, Y., Huang, S., Zhu, Y., Hussein, H. H., & Shao, X. (2022). Experimental validation of damaged reinforced concrete beam strengthened by pretensioned prestressed ultra-high-performance concrete layer. *Engineering Structures*, vol. 260, 114251.
- Harajli, M. H. (1993). Strengthening of concrete beams by external prestressing. *PCI Journal*, vol. 38(6), pp. 76-88.
- Hueste, M. B. D., Mander, J. B., Terzioglu, T., Jiang, D., & Petersen-Gauthier, J. (2015). *Spread prestressed concrete slab beam bridges*. Texas A&M Transportation Institute.
- Dubey, M., Kumar, V., Kaur, M., & Dao, T. P. (2021). *A systematic review on harmony search algorithm: Theory, literature, and applications*.
- Pham, T. D., & Hong, W. K. (2022). Genetic algorithm using probabilistic-based natural selections and dynamic mutation ranges in optimizing precast beams. *Computers & Structures*, 258, 106681, Retrieved from <https://doi.org/10.1016/j.compstruc.2021.106681>

- Sachan, R., & Singh, K. D. (2021). Nature-inspired optimization algorithms: Research direction and survey. arXiv, Retrieved from <https://doi.org/10.48550/arXiv.2102.04013>
- Vié, A. (2020). Qualities, challenges, and future of genetic algorithms: A literature review. arXiv, Retrieved from <https://doi.org/10.48550/arXiv.2011.05277>
- PCI. (2014). *PCI Bridge Design Manual* (3rd ed., 2nd release). PCI, Chicago, IL, Retrieved from <https://www.kerkstra.com/wp-content/uploads/2020/03/Bridge-Design.pdf>
- Jahjouh, M. M., & Nackenhorst, U. (2016). A modified harmony search approach on structural identification and damage detection of wind turbine supporting structures. *Journal of Vibroengineering*, vol. 18, pp. 103–118.
- California Department of Transportation. (2022). Chapter 5.5: Precast post tensioned voided slabs. In *Bridge Design Practice* (pp. 1-84).
- oldcastleinfrastructure.com. (2025). Retrieved: March 24, 2025, from: <https://oldcastleinfrastructure.com/product/bridge-decks/>
- Halbe, Joyce, Roberts-Wollmann, & Cousins. (2017). Development of improved connection details for adjacent prestressed member bridges.

Appendix A

Solid and Voided Slab Bridges Design Code Integrated with (MARSHAL) Code

Appendix A Solid and Voided Slab Bridges Design

Code Integrated with (MARSHAL) Code

In this appendix, the code of the design optimization of the solid and voided slab bridge is presented.

A.1. Define the optimization & problem inputs

```
1 close all
2 clc
3 clear
4
5
6 % Optimization Inputs
7
8 AlgorithmInputs.LocalBandWidthRatio = 0.05; %Ratio Search Space for Local
9 % This controls the % which is used by the MARSHAL when it wants to select
10 % a neighboring element during the PAR process.
11
12 AlgorithmInputs.MaxIter = 50000;
13 % This is the maximum number of iterations that the algorithm is going to
14 % use to try find the optimum solution.
15
16 AlgorithmInputs.ReportInterval = 10000;
17 % This is the number of iterations that are done before matlab shows the
18 % user the updated status of his optimization run.
19
20 AlgorithmInputs.HMCRinit = 0.5;
21 % MARSHAL will adjust the HMCR automatically, but it still needs a starting
22 % value to be adjusted. This is the starting value. MARSHAL will be
23 % modifying it very soon.
24
25 AlgorithmInputs.PARinit = 0.35;
26 % Similar to HMCR, but for PAR.
27
28 AlgorithmInputs.Gamma = 0.25; %learning rate [0.25,0.50] according to Saka
29 % and hasancebi. This is used to calculate the new HMCR each iteration
30
31 AlgorithmInputs.HMS = 30;
32 % This is the harmony memory size. The brain of the MARSHAL.
33
34 AlgorithmInputs.UseSSRMAfterIterRatio = 0.3;
35 % MARSHAL will shrink the search space after a certain number of iterations
36 % to enhance its ability to find the optimum solutions. This is the ratio
37 % of iterations elapsed before MARSHAL starts shrinking the search space.
38
39 AlgorithmInputs.BWSigmaSSRM = 2;
40 % This controls how the shrunk search space is calculated. If you have a
41 % parameter (x) stored 30 times in the Memory of MARSHAL, then the upper
42 % bound of x is calculated as upper = avg(x) + BWSigmaSSRM * std(x)
43
44
45 AlgorithmInputs.MinSpaceRatio = 0.5; % You must keep at least this ratio
46 % This is forcing MARSHAL not to allow the full collapse of the search
47 % space due to its reduction. It means that MARSHAL can only collapse the
48 % search space until it becomes the given ratio.
```

```

49
50 AlgorithmInputs.ConstraintMultiplier = 10;
51 AlgorithmInputs.ConstraintPower = 2;
52 % Constraint multiplier and power are used in the objective function
53 % penalty calculation. Phy = (1 + multiplier * c ) ^ power
54
55
56 %% Problem Inputs
57
58 ProblemInputs.S=27; %m
59 ProblemInputs.W=13.45; %m
60
61 %The bounds are determined by you. The number of bounds should equal the
62 %number of variables in your problem; because each variable must have
63 %bounds and step sizes.
64 %X=[L H D1 D2];
65 ProblemInputs.LowerBounds = [900 150 0 0];
66 ProblemInputs.UpperBounds = [1500 800 800 800];
67 ProblemInputs.StepSizes = [10 10 10 10];
68
69 % Now, we are about to start Autopilot.
70 close all
71
72 % Here, the magic of optimization happens. We are going to call MARSHAL on
73 % those problem inputs and algorithm inputs.
74 %Xopt gives you the best solution
75
76 %Fhistory gives you the best objective function for each iteration (for
77 %plotting and documentation purposes)
78
79 %HMCrhistory, PARhistory ... are similar to Fhistory, those help plotting
80 %and producing cool graphs for your work.
81 [Xopt, Fhistory, HM, F, HMCrhistory, PARhistory, Chistory, Xbesthistory, lowerBoundshistory,
upperBoundshistory] = OptimizationRun(ProblemInputs, AlgorithmInputs);
82 save('Run');
83 fprintf('Best solution is:\n');
84 fprintf('L = %5.3f\n', Xopt(1));
85 fprintf('H = %5.3f\n', Xopt(2));
86 fprintf('D1 = %5.3f\n', Xopt(3));
87 fprintf('D2 = %5.3f\n', Xopt(4));
88 [FObj, constraintViolation, ns, L1, L2, cost] = ObjectiveFunction(Xopt, ProblemInputs, AlgorithmInputs);
89 fprintf('L1 = %5.3f\n', L1);
90 fprintf('L2 = %5.3f\n', L2);
91 fprintf('Number of strands is: %5.3f\n', ns);
92 fprintf('cost per m span length = %5.3f\n', cost);
93 fprintf('Its Objective Function Value is: %5.3f\n', FObj);
94 fprintf('Its constraint violation (should be 0) = %5.3f\n', constraintViolation);
95 figure
96 plot(1:AlgorithmInputs.MaxIter, Fhistory);
97 title('Convergence History')
98 xlabel('Iterations');
99 ylabel('F(x)');

```

A.2. Optimization Run

```
1 function [Xopt, Fhistory, HM, F, HMCRhistory, PARhistory, Chistory, Xbesthistory, lowerBoundshistory,
upperBoundshistory] = OptimizationRun(ProblemInputs, AlgorithmInputs)
2 % The meaning of the inputs and outputs of this function have been
3 % explained in the file "startHere.m". Look those up please.
4
5
6
7 % Initialization
8
9 % Since we are going to store some histories, let's initialize those to
10 % zero.
11 Xbesthistory = zeros(AlgorithmInputs.MaxIter, length(ProblemInputs.LowerBounds));
12 lowerBoundshistory = zeros(size(Xbesthistory));
13 upperBoundshistory = zeros(size(Xbesthistory));
14 Fhistory = zeros(1, AlgorithmInputs.MaxIter);
15 Chistory = zeros(1, AlgorithmInputs.MaxIter);
16 HMCRhistory = zeros(1, AlgorithmInputs.MaxIter);
17 PARhistory = zeros(1, AlgorithmInputs.MaxIter);
18
19 % Now, let's initialize the Harmony Memory HM, objective function F
20 HM = zeros(AlgorithmInputs.HMS, length(ProblemInputs.LowerBounds));
21 F = zeros(1, AlgorithmInputs.HMS);
22
23 % Also, for MARSHAL, we need to initialize the HMCR vector and PAR vector
24 % that are used to calculate a new HMCR and PAR value each iteration.
25 HMCR_Storage = ones(AlgorithmInputs.HMS, 1) * AlgorithmInputs.HMCRinit; % Initialize HMCR vector
26 PAR_Storage = ones(AlgorithmInputs.HMS, 1) * AlgorithmInputs.PARinit; % Initialize PAR vectors
27
28 % Finally, let's initialize the constraint violation vector, because why
29 % not.
30 C = zeros(1, AlgorithmInputs.HMS);
31
32 % Starting the initialization phase. Here, the memory is filled with random
33 % values from the search space. The way it is programmed looks cool. You
34 % could simplify if you want.
35 for solIdx = 1:AlgorithmInputs.HMS
36 % For each Solution
37 for elemIdx = 1:length(ProblemInputs.LowerBounds)
38 % For each element
39 element = rand * (ProblemInputs.UpperBounds(elemIdx) - ProblemInputs.LowerBounds(elemIdx)) +
ProblemInputs.LowerBounds(elemIdx);
40 element = round(element / ProblemInputs.StepSizes(elemIdx)) * ProblemInputs.StepSizes(elemIdx);
41 % The element (variable) is created, let's store it in the solution
42 % in the HM
43 HM(solIdx, elemIdx) = element;
44 end
45
46 % The solution has been created, let's evaluate its objective function
47 % value.
48 [FObj, c] = ObjectiveFunction(HM(solIdx,:), ProblemInputs, AlgorithmInputs);
49
50 % Let's store those values in their corresponding vectors
51 F(solIdx) = FObj;
52 C(solIdx) = c;
53 end
54 % END OF INITIALIZATION PHASE
55
56
57 % Now let's go and do the iterative search
```

```

58 for iter = 1:AlgorithmInputs.MaxIter
59 % for each iteration
60
61 % Let's initialize the lower and upper bounds for THIS iteration. It
62 % might be changed, it might not be changed, it depends on the
63 % settings.
64 lowerBounds = ProblemInputs.LowerBounds;
65 upperBounds = ProblemInputs.UpperBounds;
66
67 % Let's ask the question, did enough iteration elapse to start reducing
68 % the search space??
69 % THIS IS MARSHAL'S FIRST IMPROVEMENT ON Original HS
70 if iter > AlgorithmInputs.MaxIter * AlgorithmInputs.UseSSRMAfterIterRatio
71 % yes, so the magic of search space reduction happens here. It is
72 % written in a cool way. You can simplify it if you want.
73 means = mean(HM);
74 means = round(means./ProblemInputs.StepSizes).*ProblemInputs.StepSizes;
75 stds = std(HM);
76 stds = round(stds./ProblemInputs.StepSizes).*ProblemInputs.StepSizes;
77 for ii = 1:length(lowerBounds)
78 lbx = means(ii)- AlgorithmInputs.BWSigmaSSRM * stds(ii);
79 lbx = max(ProblemInputs.LowerBounds(ii),lbx);
80 ubx = means(ii)+ AlgorithmInputs.BWSigmaSSRM * stds(ii);
81 ubx = min(ProblemInputs.UpperBounds(ii),ubx);
82 stepsTotal = (ProblemInputs.UpperBounds(ii) -
ProblemInputs.LowerBounds(ii))/ProblemInputs.StepSizes(ii);
83 stepsFromlbxToOrig = (lbx-ProblemInputs.LowerBounds(ii))/ProblemInputs.StepSizes(ii);
84 stepsFromubxToOrig = (ProblemInputs.UpperBounds(ii) - ubx)/ProblemInputs.StepSizes(ii);
85 stepsFromlbxToubx = (ubx - lbx)/ProblemInputs.StepSizes(ii);
86 if stepsFromlbxToubx < AlgorithmInputs.MinSpaceRatio * stepsTotal
87 % Our search space is too small. Override.
88 dSteps = AlgorithmInputs.MinSpaceRatio * stepsTotal - stepsFromlbxToubx;
89 if stepsFromlbxToOrig > round(0.50*dSteps) && stepsFromubxToOrig > round(0.50*dSteps)
90 lbx = lbx - round(0.50*dSteps)*ProblemInputs.StepSizes(ii);
91 ubx = ubx + round(0.50*dSteps)*ProblemInputs.StepSizes(ii);
92 else
93 if(stepsFromlbxToOrig > stepsFromubxToOrig)
94 ubx = ProblemInputs.UpperBounds(ii);
95 lbx = lbx - round(dSteps - stepsFromubxToOrig)*ProblemInputs.StepSizes(ii);
96 else
97 lbx = ProblemInputs.LowerBounds(ii);
98 ubx = ubx + round(dSteps - stepsFromlbxToOrig)*ProblemInputs.StepSizes(ii);
99 end
100 end
101 end
102 %Final Double Check
103 lbx = max(ProblemInputs.LowerBounds(ii),lbx);
104 ubx = min(ProblemInputs.UpperBounds(ii),ubx);
105
106 % The new bounds have been calculated, let's put them into the
107 % bound vector.
108 lowerBounds(ii) = lbx;
109 upperBounds(ii) = ubx;
110 end
111 end
112 % We finished modifying our bounds.
113
114
115 if(mod(iter,10) == 0)
116 % That is some timer thingy. Might not be needed.
117 tic;
118 end
119

```

```

120 %MARSHAL's second improvement on the original HS (this is taken from
121 %the adaptive harmony search) (see saka hasancebi's paper)
122
123 %Let's calculate the HMCR and PAR needed for this iteration. Note that
124 %we are going to use the HMCR and PAR vector that we have initialized.
125 HMCR = (1+ (1-mean(HMCR_Storage))/mean(HMCR_Storage)*exp(-
AlgorithmInputs.Gamma*normrnd(0,1)))^1;
126 PAR = (1+ (1-mean(PAR_Storage))/mean(PAR_Storage)*exp(-AlgorithmInputs.Gamma*normrnd(0,1)))^1;
127
128 % Ok now we are ready to assume a new solution in the iterative search
129 %Let's initialize the solution
130 Xnew = zeros(1,length(lowerBounds));
131
132
133 % Since there might be a PAR happening, we need to convert the
134 % "LocalBandwidthRatio" into number of steps.
135
136 % Calculate Local Bandwidth from bounds
137 numSteps = (upperBounds-lowerBounds)./ProblemInputs.StepSizes;
138 bWs = ceil(AlgorithmInputs.LocalBandWidthRatio * numSteps);
139
140
141 % OK, NOW LETS GET THE XNEW variable by variable
142 for elemIdx = 1:length(lowerBounds)
143 % Check the HMCR, are we going to take the variable from inside or
144 % outside the HM?
145
146 if rand < HMCR
147 % From Inside HM
148 % Some cool magic trick happens here to get the variable from
149 % inside.
150 rowIdx = round(rand * (AlgorithmInputs.HMS-1) + 1);
151 xi = HM(rowIdx,elemIdx);
152
153 % Check the PAR, are we going to switch the variable with one
154 % of its neighbors?
155 if rand < PAR
156 % Yes, we are going to switch that thing.
157 xi = xi - bWs(elemIdx)*ProblemInputs.StepSizes(elemIdx) +
2*rand*bWs(elemIdx)*ProblemInputs.StepSizes(elemIdx);
158 xi = round(xi/ProblemInputs.StepSizes(elemIdx))*ProblemInputs.StepSizes(elemIdx);
159 xi = max(min(upperBounds(elemIdx),xi),lowerBounds(elemIdx));
160 end
161 Xnew(elemIdx) = xi;
162 else %rand > HMCR
163 % Well, we are not going to select anything from inside HM, so
164 % let's select something randomly.
165 xi = rand *(upperBounds(elemIdx) - lowerBounds(elemIdx)) + lowerBounds(elemIdx);
166 xi = round(xi / ProblemInputs.StepSizes(elemIdx))*ProblemInputs.StepSizes(elemIdx);
167 Xnew(elemIdx) = xi;
168 end
169 end
170
171 % We are ready with our new Xnew.. Let's get its objective function.
172 % THIS IS WHERE EACH PROBLEM IS DIFFERENT, IN THE OBJECTIVE FUNCTION
173 [Fnew, cNew] = ObjectiveFunction(Xnew,ProblemInputs, AlgorithmInputs);
174
175 % Let's get the worst solution inside the HM
176 [Fworst, idxWorst] = max(F);
177
178 % Is our solution better than the worst?
179 if (Fnew < Fworst)
180 %Yes, our solution is better ( < ) than the worst. Let's do some

```

```

181 %replacement
182
183 % Remember, not only is the solution replaced, but its F, C, HMCR
184 % and PAR.
185 F(idxWorst) = Fnew;
186 C(idxWorst) = cNew;
187 HM(idxWorst,:) = Xnew;
188 HMCR_Storage(idxWorst) = HMCR;
189 PAR_Storage(idxWorst) = PAR;
190 end
191
192 %Let's do some data storage
193
194 %Get the best solution
195 [Fbest, idxBest] = min(F);
196 Xbest = HM(idxBest,:);
197
198 %Store the best solution
199 Xbesthistory(iter,:) = Xbest;
200
201 % Store the bounds and so on.
202 lowerBoundshistory(iter,:) = lowerBounds;
203 upperBoundshistory(iter,:) = upperBounds;
204 Fhistory(iter) = Fbest;
205 Chistory(iter) = C(idxBest);
206 HMCRhistory(iter) = HMCR;
207 PARhistory(iter) = PAR;
208
209 % Check if we want to provide any info in the console of matlab?
210 if(mod(iter,AlgorithmInputs.ReportInterval) == 0)
211 % Yes we want.
212 clc
213 fprintf('%i iteration out of %i\n', iter, AlgorithmInputs.MaxIter)
214 % FEEL FREE TO REPORT ANYTHING IN THE CONSOLE
215 end
216 [~,idxBest] = min(F);
217 Xopt = HM(idxBest,:);
218
219
220 %We finished this iteration, let's do it over and over and over again.
221 end

```

A.3. Objective function

```
1 function [FObj,constraintViolation,ns,L1,L2,cost] = ObjectiveFunction(X,ProblemInputs,AlgorithmInputs)
2 L = X(1);
3 H = X(2);
4 D1 = X(3);
5 D2 = X(4);
6 edged=100;
7 innerd=(L-2*D1-D2-edged*2)/2;
8 A=L*H-pi()/4*D1^2-pi()/4*D2^2;
9 Yb=H/2;
10 I=L*H^3/12-pi()*D1^4/64*2-pi()*D2^4/64;
11 J=A^4/(40*I);
12 Sb=I/(H/2);
13 S = ProblemInputs.S;
14 W = ProblemInputs.W;
15 %voids dimensions limits
16 if (H-max(D1,D2))/2<75
17 FObj=inf;
18 constraintViolation=inf;
19 return;
20 %elseif (pi()/4*D1^2+pi()/4*D2^2)/A>0.4
21 % FObj=inf;
22 % constraintViolation=inf;
23 % return;
24 %elseif max(D1,D2)/H>0.6
25 % FObj=inf;
26 % constraintViolation=inf;
27 % return;
28 elseif innerd<75 && D2~=0
29 FObj=inf;
30 constraintViolation=inf;
31 return;
32 elseif innerd*2<75 && D2==0
33 FObj=inf;
34 constraintViolation=inf;
35 return;
36 end
37 n=ceil(W*1000/L);
38 if n<5
39 FObj=inf;
40 constraintViolation=inf;
41 return;
42 elseif n>20
43 FObj=inf;
44 constraintViolation=inf;
45 return;
46 end
47 %The gap should be at least 1.5 times the sweep, for span >= 12 m g=0.75 in, for span <= 12 m g=0.5 in
48 if S>=12
49 g=0.75*25.4*(n-1);
50 elseif S<12
51 g=0.5*25.4*(n-1);
52 end
53 %Distribute dead loads assuming a uniform distribution across the width using the following
54 %dead load distribution factor DFDL = (Slab width + gap) / (total bridge width)
55 DFDL=(L+0.75*25.4)/(W*1000);
56 %DEAD LOAD ANALYSIS
57 DC1= 24*A*10^-6;
58 VDC1=DC1*S/2;
59 MDC1=DC1*S^2/8;
```

```

60 DC2=2*3.65*DFDL;
61 VDC2=DC2*S/2;
62 MDC2=DC2*S^2/8;
63 DW=22.07*0.075*(W-2*0.61)*DFDL;
64 VDW=DW*S/2;
65 MDW=DW*S^2/8;
66 %LIVE LOAD ANALYSIS
67 Vtruck = max((145*0+145*4.3+35*8.6)/S,145*2+35-(145*0+145*4.3+35*8.6)/S);
68 Mtruck = (145*2+35-((S/2-4.3)*145+145*S/2+35*(S/2+4.3))/S)*S/2-145*(S/2-(S/2-4.3));
69 Vtandem = max((110*0+110*1.2)/S,110*2-(110*0+110*1.2)/S);
70 Mtandem = (110*2-(110*S/2+110*(S/2+1.2))/S)*S/2-110*(S/2-S/2);
71 Vlane= 9.3*S/2;
72 Mlane= 9.3*S^2/8;
73 Vmax=max(Vtruck*1.33+Vlane,Vtandem*1.33+Vlane);
74 Mmax=max(Mtruck*1.33+Mlane,Mtandem*1.33+Mlane);
75 IJ=0.54*H/L+0.16;
76 if(2.5*n^-0.2>=1.5)
77 k=2.5*n^-0.2;
78 else
79 k=1.5;
80 end
81 DFM1=k*((L+0.75*25.4)/(33.3*S*1000/12))^0.5*IJ^0.25;
82 DFM2=k*((L+0.75*25.4)/(305*25.4))^0.6*((L+0.75*25.4)/(12*S*1000/12))^0.2*(IJ)^0.06;
83 DFM=max(DFM1,DFM2);
84 if I>=40000*25.4^4 && I<=610000*25.4^4 && J>=25000*24.4^4 && J<=610000*25.4^4
85 DFV1=((L+0.75*25.4)/(130*S*1000/12))^0.15*IJ^0.05;
86
87 DFV2=((L+0.75*25.4)/(156*25.4))^0.4*((L+0.75*25.4)/(12*S*1000/12))^0.1*IJ^0.05*max(1,(L+0.75*25.4)/(
48*25.4));
87 DFV=max(DFV1,DFV2);
88 else
89 DFV=DFM;
90 end
91 VLL=Vmax*DFV;
92 MLL=Mmax*DFM;
93 %NO. OF STRANDS
94 fpu=1860;
95 fc=35;
96 %Assume 38 mm cover, 12mm stirrups, 15.24 mm strands, offset from bottom is calculated as follows:
97 offset=round((38.1+12+15.24/2)/5)*5;
98 e=Yb-offset;
99 fpj=0.75*fpu;
100 loss=20;
101 Astrand=140;
102 fallow=0.5*sqrt(fc');
103 fbt=((MDC1+MDC2+MDW+1*MLL)/Sb)*10^6;
104 fbtn=fbt-fallow;
105 Pp=((Sb*A)/(Sb+e*A))*fbtn;
106 Ppstr=fpj*Astrand*(1-loss/100);
107 nmin=Pp/Ppstr;
108 if mod(ceil(nmin),2)==0
109 ns=ceil(nmin);
110 else
111 ns=ceil(nmin)+1;
112 end
113 %CONSTRAINTS CALC _ C1 & C2 _ MAX NO OF STRANDS & SPACING
114 %consider 50 mm c/c spacing of strands, 60mm from girder end to center of strands
115 nsmax=floor((L-70)/50);
116 if ns<=nsmax
117 c1=0;
118 else
119 c1=abs((ns-nsmax)/nsmax);
120 end

```

```

121 %spacing of tendons or groups of tendons in at least one direction shall not exceed the smaller of
eight times the slab thickness and 1.5 m.
122 if (L-60*2)/(ns-1)<min(8*H,1500);
123 c2=0;
124 else
125 c2=abs((ns-min(8*H,1500))/min(8*H,1500));
126 end
127 Aps=140*ns;
128 Pj=Aps*fpj/1000;
129 %PRESTRESS LOSSES
130 Ep=197;
131 Ect=4800*sqrt(fc)/1000;
132 initial=5;
133 Pi=0.75*fpu*Aps*(1-initial/100);
134 Mi=Pi*e;
135 fcgp=Pi/A+Mi*e/I-MDC1*106*e/I;
136 Dfpes=Ep/Ect*fcgp;
137 calc=Dfpes*100/(fpu*0.75);
138 %to stop the while loop to run to infinity
139 iter = 0;
140 while (abs(initial - calc) > 0.1 && iter <20)
141 initial=calc;
142 Pi=0.75*fpu*Aps*(1-initial/100);
143 Mi=Pi*e;
144 fcgp=Pi/A+Mi*e/I-MDC1*106*e/I;
145 Dfpes=Ep/Ect*fcgp;
146 calc=Dfpes*100/(fpu*0.75);
147 iter = iter + 1;
148 end
149 if(abs(initial - calc) > 0.1)
150 % did not converge. Explode
151 FObj=inf;
152 constraintViolation=inf;
153 return;
154 end
155 fpi=0.75*fpu;
156 fci=0.8*fc;
157 gst=35/(7+fci);
158 %gh=1.0 For 70% humidity
159 Dfplt=10*fpi*Aps/A*1*gst+83*1*gst+17;
160 Dfpt=Dfpes+Dfplt;
161 Df=Dfpt/fpi*100;
162 %SERVICE LIMIT STATE
163 Pp=(0.75*fpu-Dfpes-Dfplt)*Aps;
164 ftop=-1*Pp/A+Pp*e/Sb-(MDC1+MDC2+MDW+MLL)*106/Sb;
165 %CONSTRAINTS CALC _ C3,C4,C5 _ SERVICE LIMIT STATE STRESSES
166 if abs(ftop)<0.6*fc
167 %disp("top stress is within the allowable");
168 c3=0;
169 else
170 %disp("top stress is exceeding the allowable");
171 c3=abs((ftop-0.6*fc))/abs(0.6*fc);
172 end
173 fbot=-1*Pp/A-Pp*e/Sb+(MDC1+MDC2+MDW+MLL)*106/Sb;
174 if abs(fbot)<0.5*sqrt(fc)
175 %disp("bottom stress is within the allowable");
176 c4=0;
177 else
178 %disp("bottom stress is exceeding the allowable");
179 c4=abs((fbot-0.5*sqrt(fc))/abs(0.5*sqrt(fc)));
180 end
181 fstrand=fpi-Dfpt;
182 if abs(fstrand)<0.8*(0.9*1860)

```

```

183 %disp("strands stress is within the allowable");
184 c5=0;
185 else
186 %disp("strands stress is exceeding the allowable");
187 c5=abs((fstrand-(0.8*0.9*1860)))/abs(0.8*0.9*1860);
188 end
189 %FLEXURAL DESIGN
190 %38 mm cover, 12 mm stirrups, 16 mm reinf. dia
191 ds=2*Yb-25-12-16/2;
192 if fc<=28
193 B1=0.85;
194 else
195 B1=max(0.85-0.05*(fc-28)/7,0.65);
196 end
197 dp=Yb+e;
198 hv=Yb-D1/2;
199 fs=460;
200 %assuming As= 0 / Asmin of bonded reinforcement, As'=0;
201 As=0;
202 fr=0.63*sqrt(fc);
203 %k= 0.28 for low- relaxation steel, (fpy/fpu) = 0.9
204 %calculating c assuming rectangular section
205 c=(Aps*fpu+As*fs-0)/(0.85*fc*B1*L+0.28*Aps*fpu/dp);
206 if c<hv
207 %disp("rectangular section assumption is applicable");
208 fpe=fstrand;
209 fps=fpu*(1-0.28*c/dp);
210 Mn=(Aps*fps*(dp-c*B1/2)+As*fs*(ds-c*B1/2))/10^6;
211 else
212 %disp("rectangular section assumption is not applicable, recalculate c for T-section behavior");
213 c=(Aps*fpu+As*fs-0.85*fc*(L-(L-2*D1-D2)))/(0.85*fc*B1*L+0.28*Aps*fpu/dp);
214 fpe=fstrand;
215 fps=fpu*(1-0.28*c/dp);
216 Mn=(Aps*fps*(dp-c*B1/2)+As*fs*(ds-c*B1/2)-0.85*fc*(L-(L-2*D1-D2))*hv*(c*B1/2-hv/2))/10^6;
217 end
218 %phi=1 for flexural capacity of section
219 Mr=1*Mn;
220 Mu=1.25*(MDC1+MDC2)+1.5*(MDW)+1.75*(MLL);
221 %CONSTRAINTS CALC _ C6,C7 _ ULTIMATE LIMIT STATE FLEXURAL
222 if Mr>Mu
223 %disp("moment capacity check is confirmed");
224 c6=0;
225 else
226 %disp("moment capacity is exceeded, As required calculation shall be considered");
227 c6=abs((Mu-Mr))/abs(Mu);
228 end
229 fcpe=Pp/A+Pp*e/Sb;
230 Mdnc=MDC1+MDC2;
231 Mcr=(Sb*(1.6*fr+1.1*fcpe)-Mdnc*10^6*(Sb/Sb-1))/10^6;
232 if Mr>min(1.33*Mu,Mcr)
233 %disp("minimum reinforcement check is confirmed");
234 c7=0;
235 else
236 %disp("minimum reinforcement check is not confirmed");
237 c7=abs((min(1.33*Mu,Mcr)-Mr))/abs(min(1.33*Mu,Mcr));
238 end
239 % All above this line, is what you should program. You must program how the
240 % algorithm calculates all constraints and all the necessary values to
241 % achieve those constraints.
242
243 % ALOT OF THINGS MUST BE CALCULATED TO BE ABLE TO CALCULATE YOUR
244 % CONSTRAINTS
245

```

```

246 % This is where you calculate c1 c2 c3 c4
247 constraintViolation = c1+c2+c3+c4+c5+c6+c7;
248
249 % cost calculated per m length (concrete cost per cubic meter=33 BHD/m3 * x-sectional area (m2))
250 % strands weight per m length = (strands cost per kg = 377 BHD/kg * ns * 1.102 kg/m)
251 L1=edged+D1/2;
252 L2=innerd+D1/2+D2/2;
253
254 cost = ((W*1000-g)/L)*(A/1000^2*33+ns*1.102*360/907+(As*360/907*7850/1000^2));
255
256 FObj = cost * (1 +
AlgorithmInputs.ConstraintMultiplier*constraintViolation)^AlgorithmInputs.ConstraintPower;
257 end

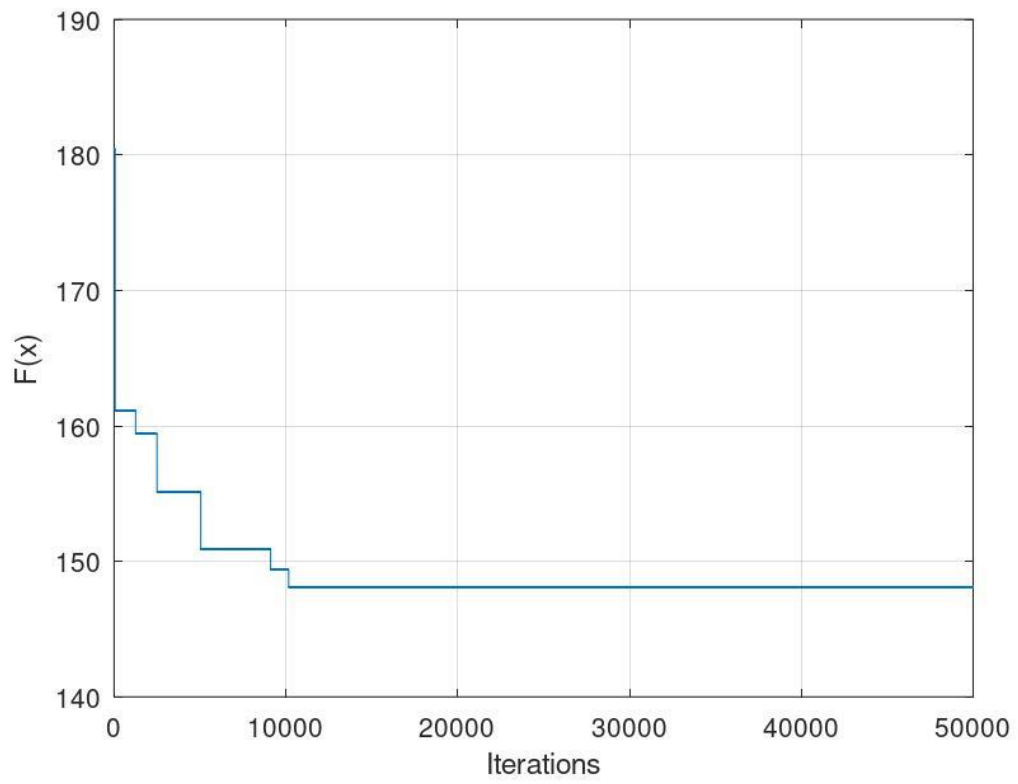
```

Appendix B

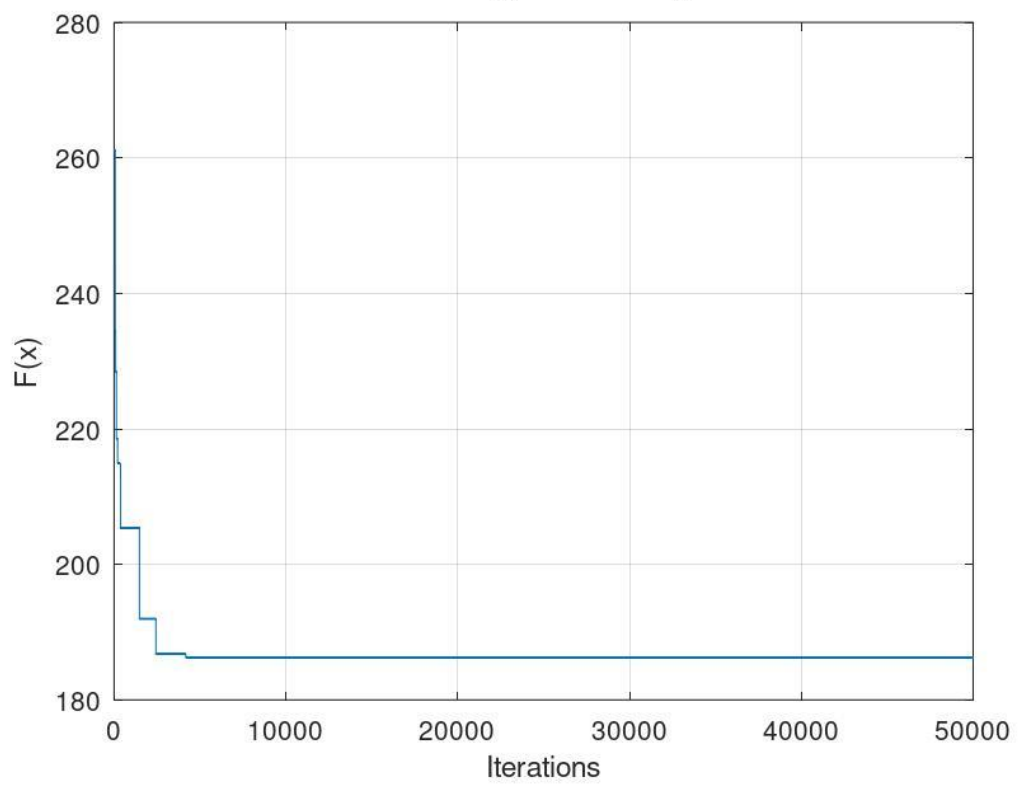
Optimization Runs Convergence Graphs

Appendix B Optimization Runs Convergence Graphs

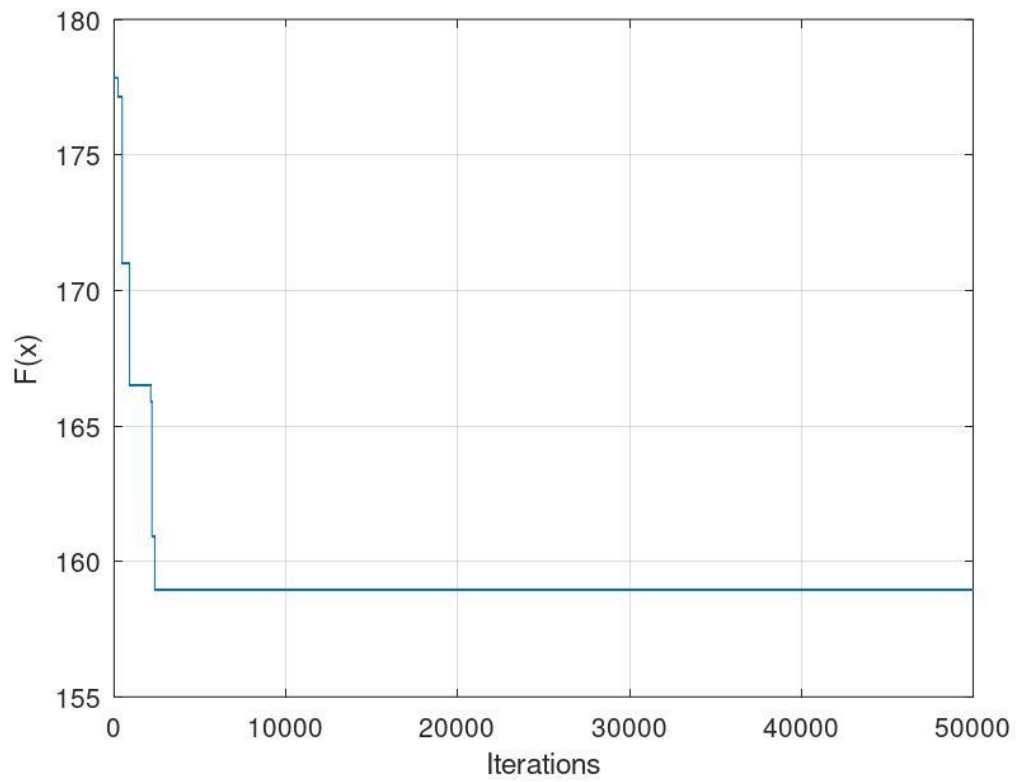
Convergence History



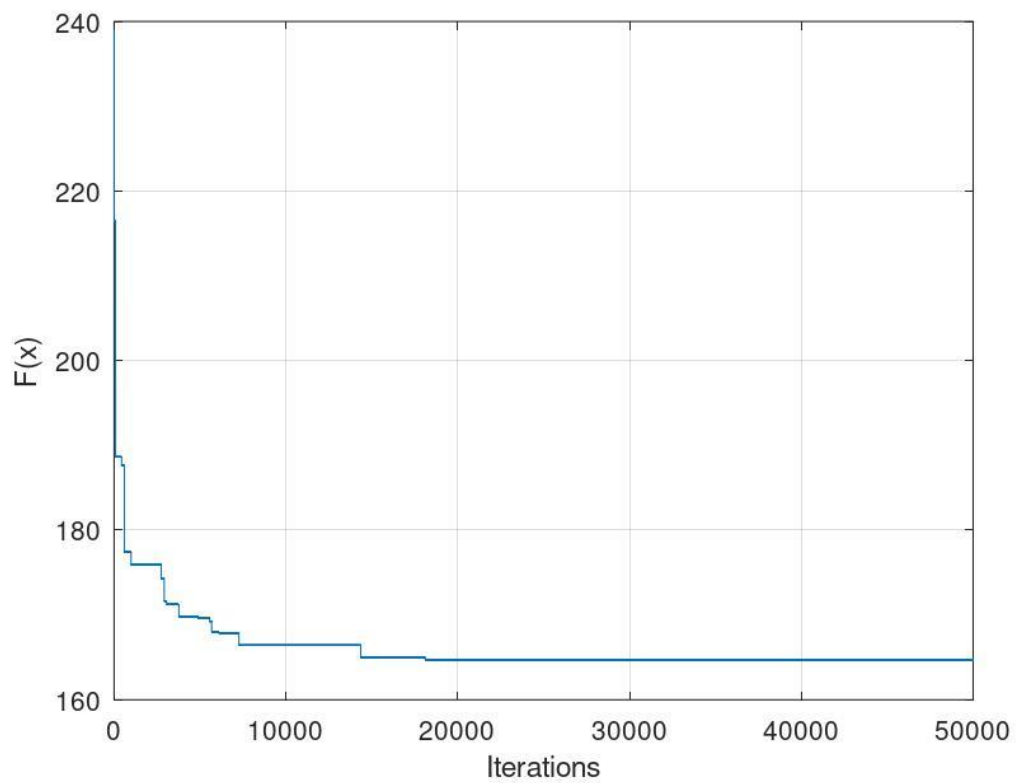
Convergence History



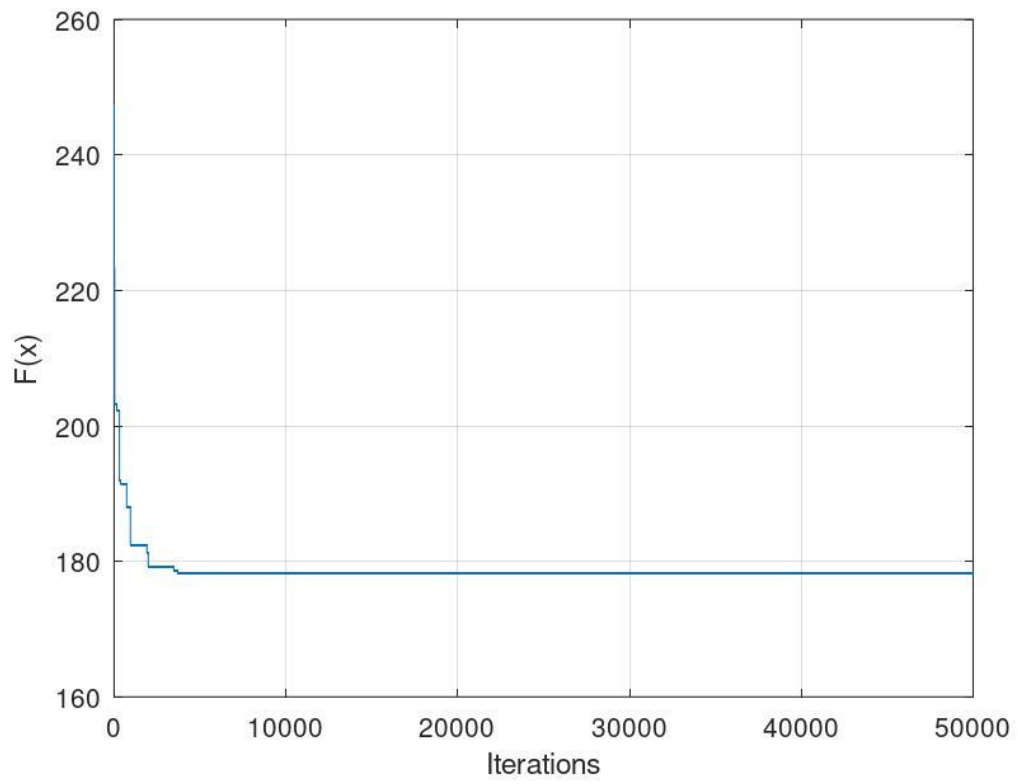
Convergence History



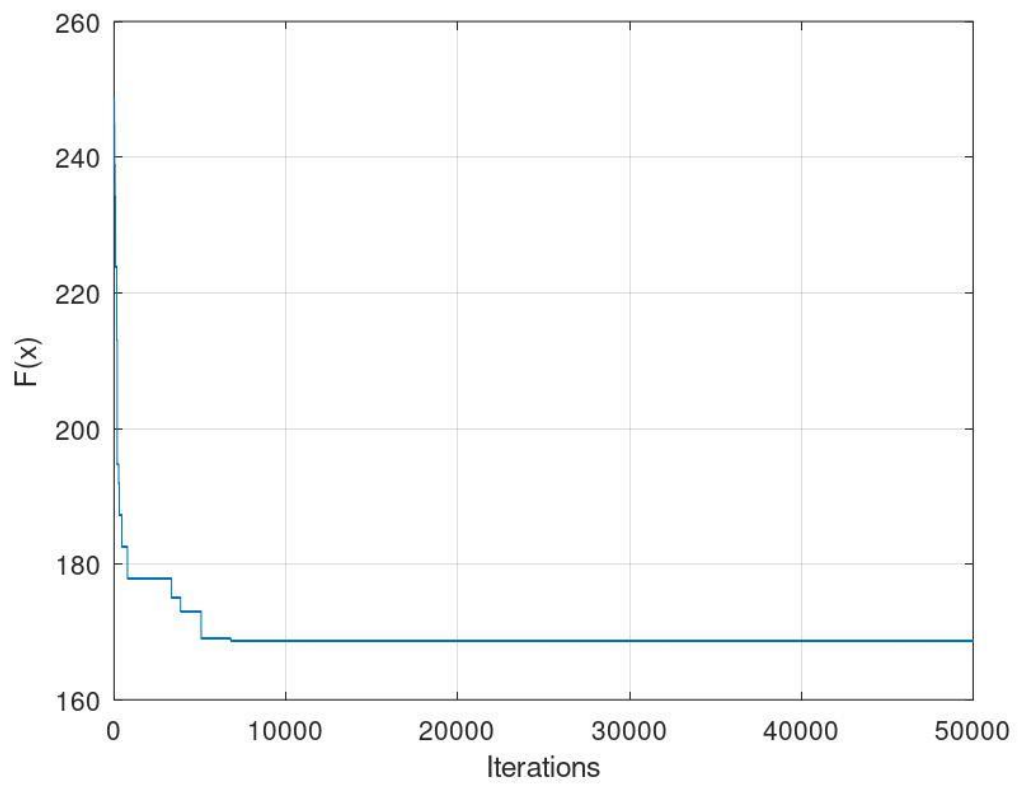
Convergence History



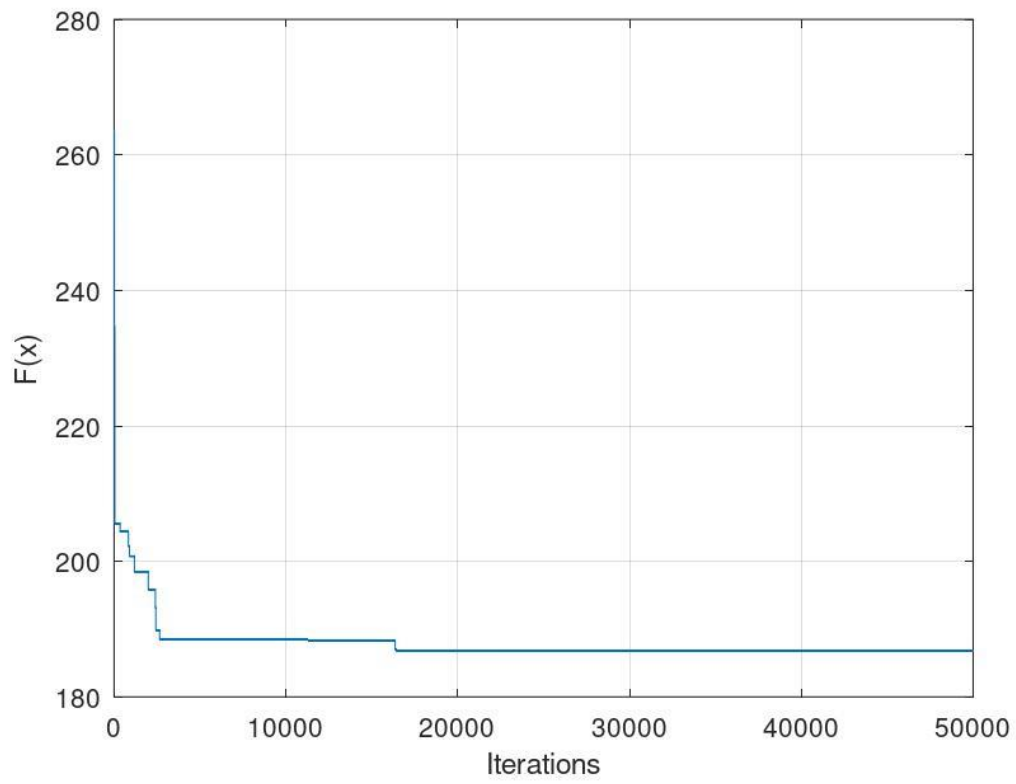
Convergence History



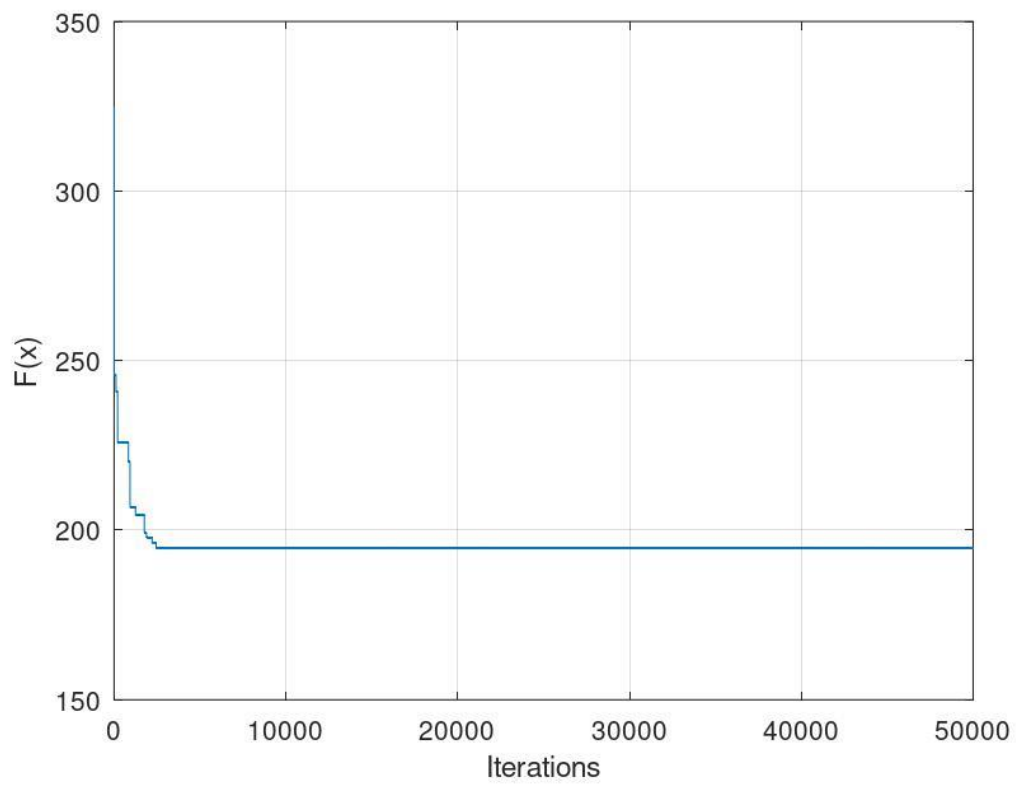
Convergence History



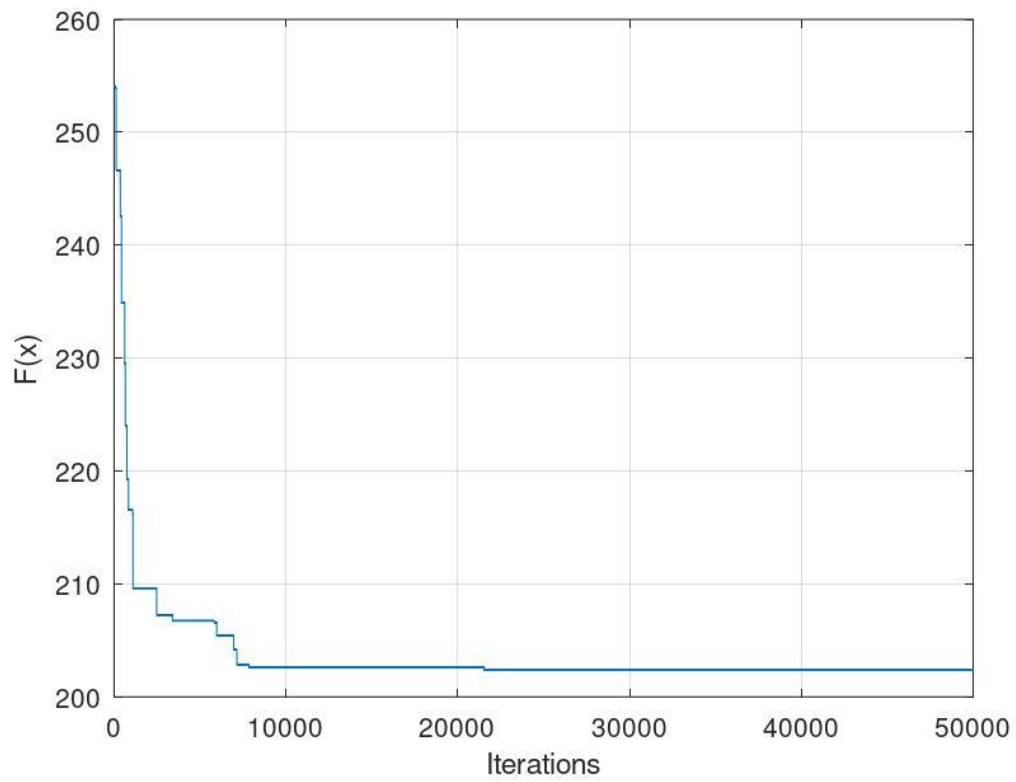
Convergence History



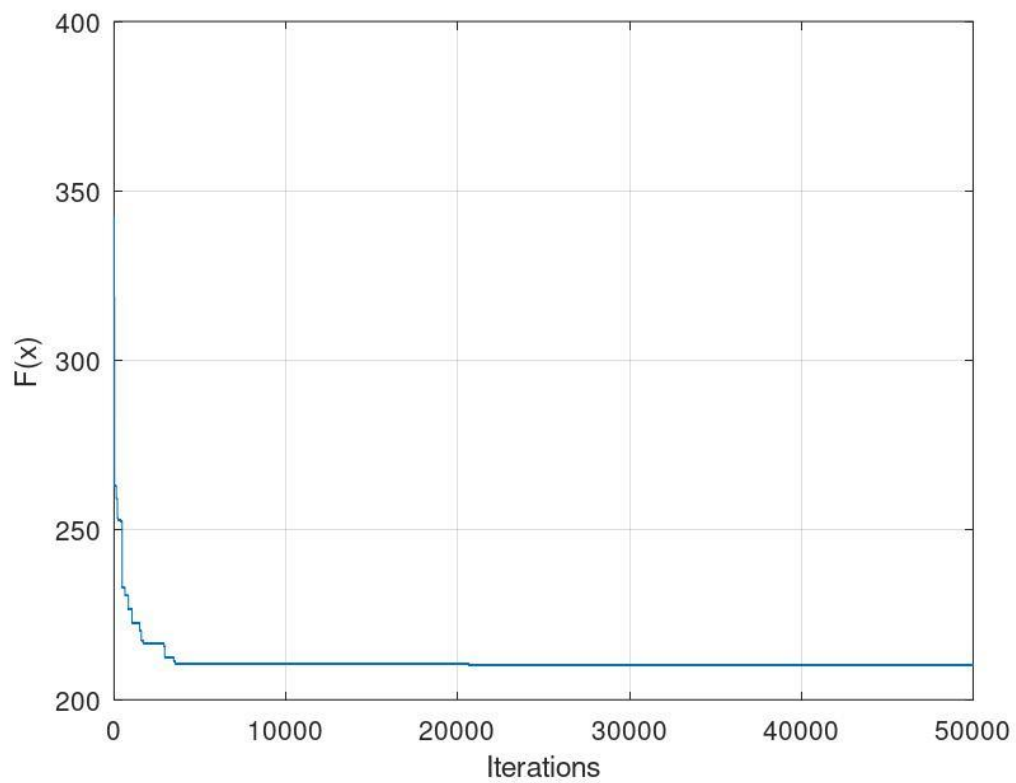
Convergence History

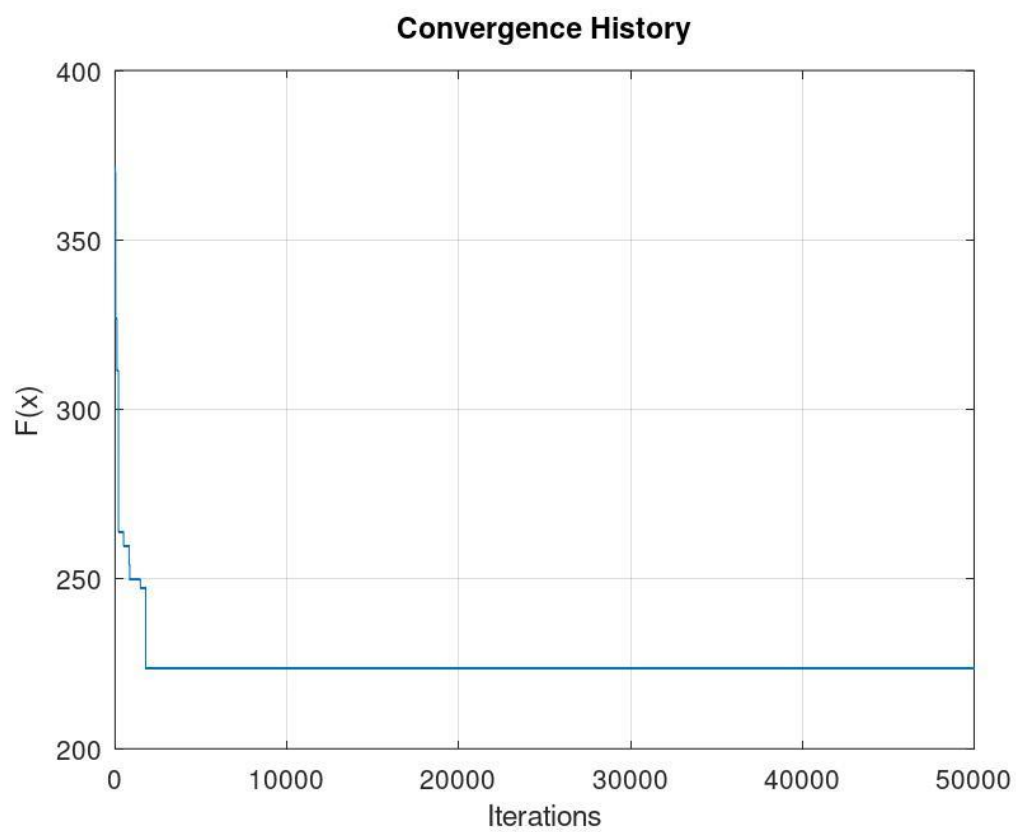
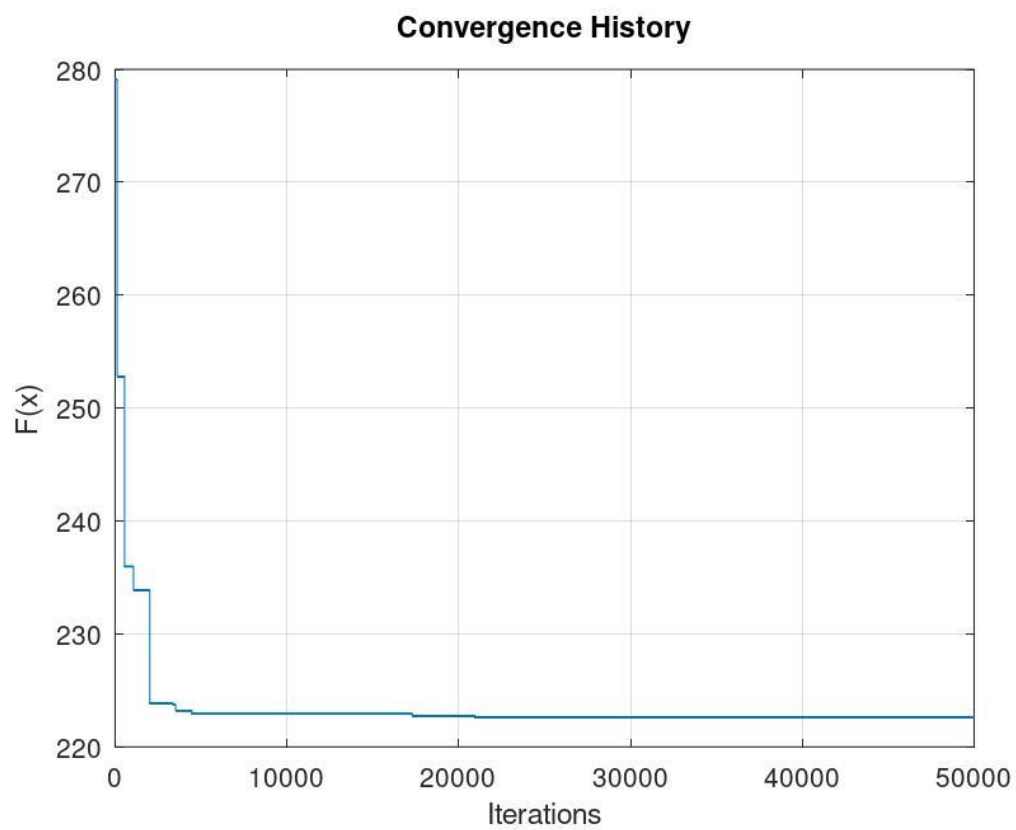


Convergence History

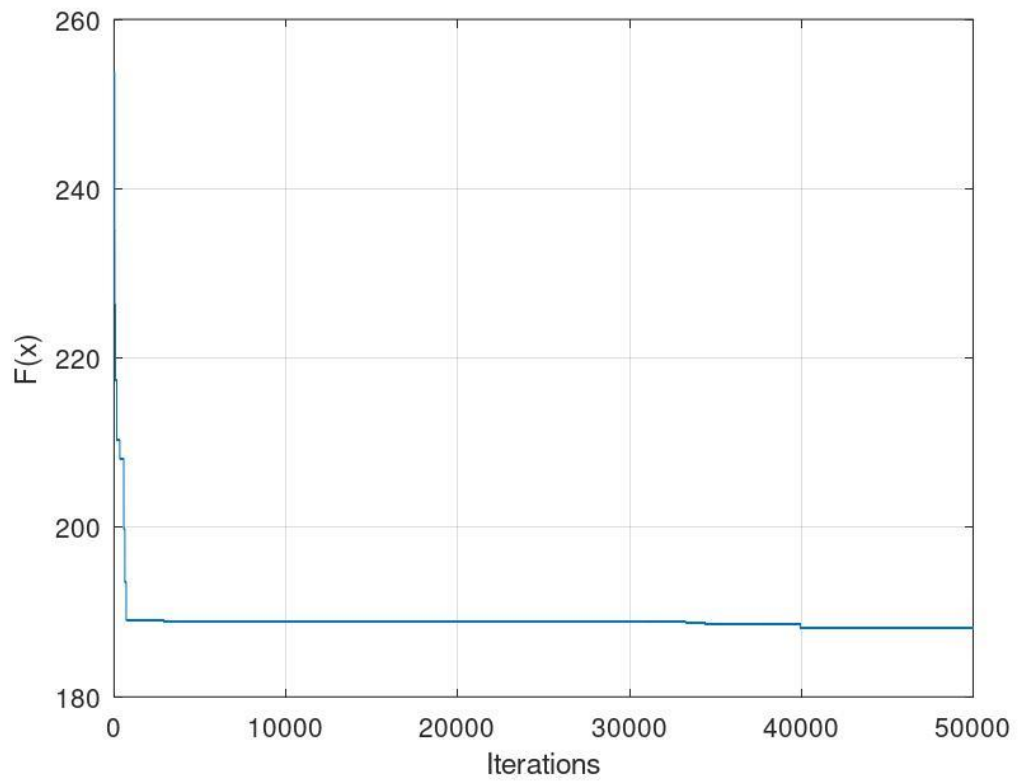


Convergence History

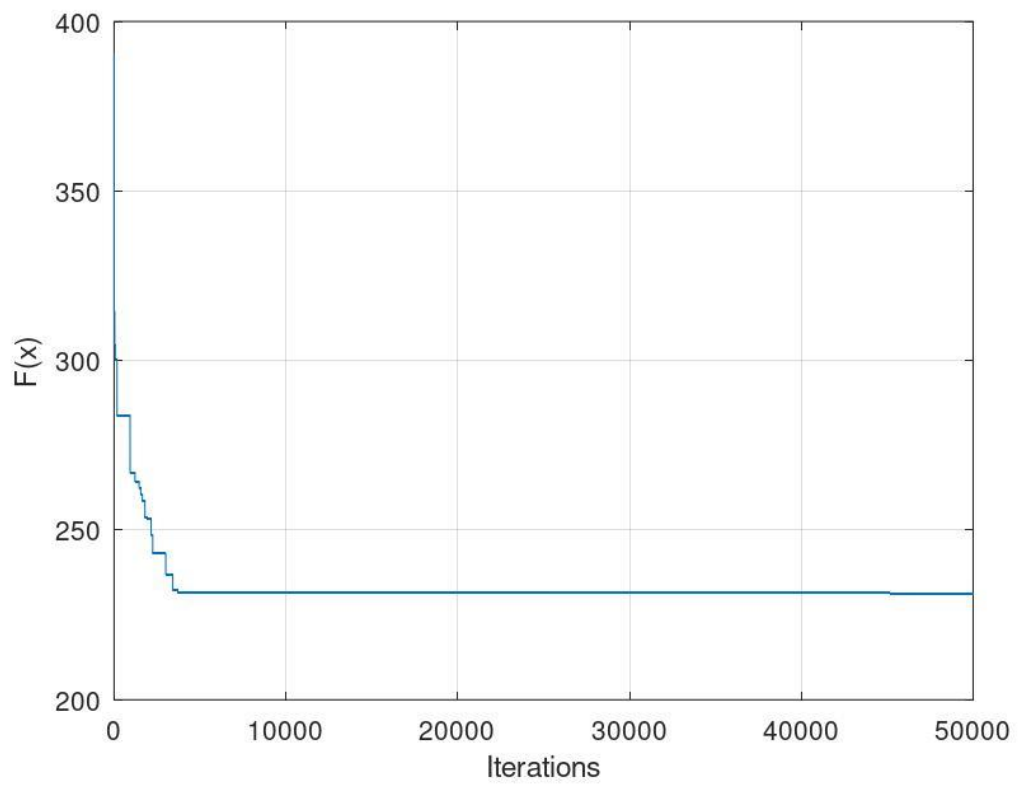


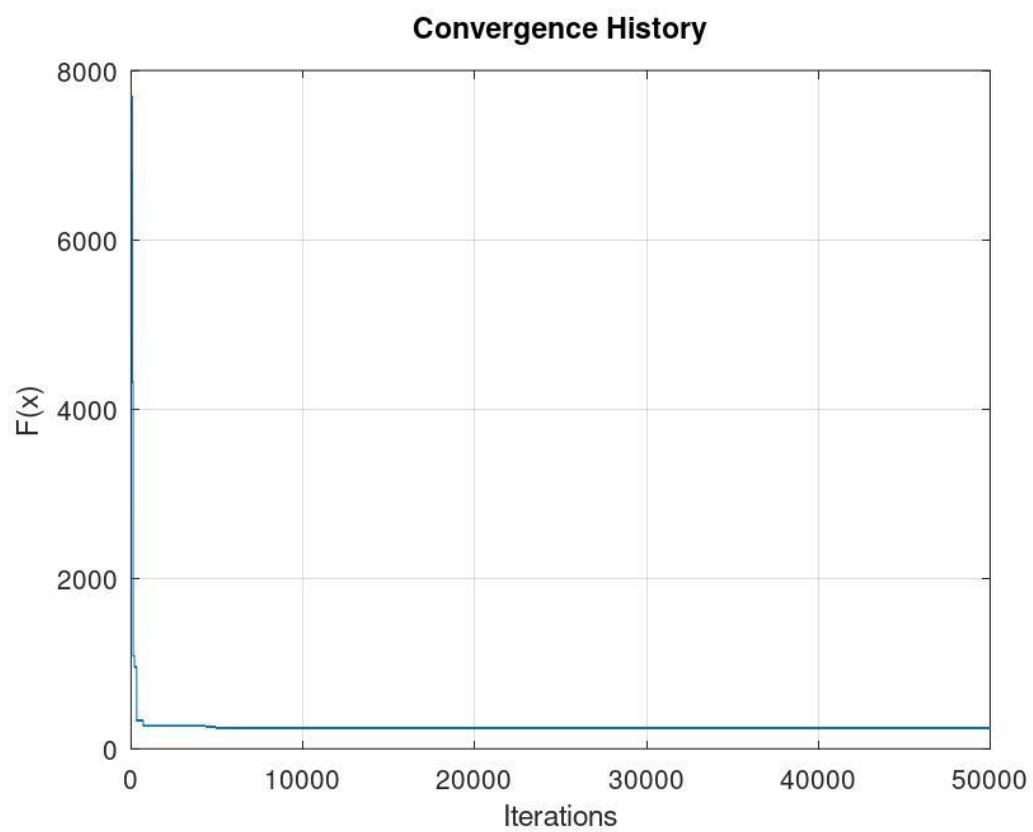
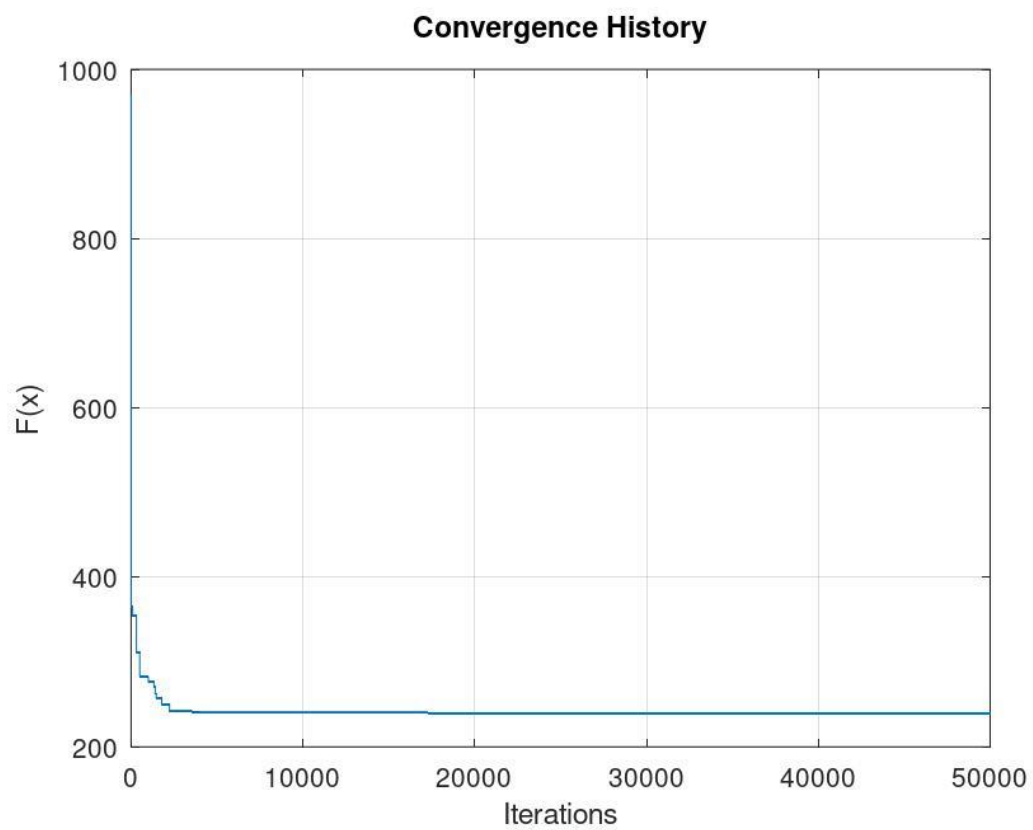


Convergence History

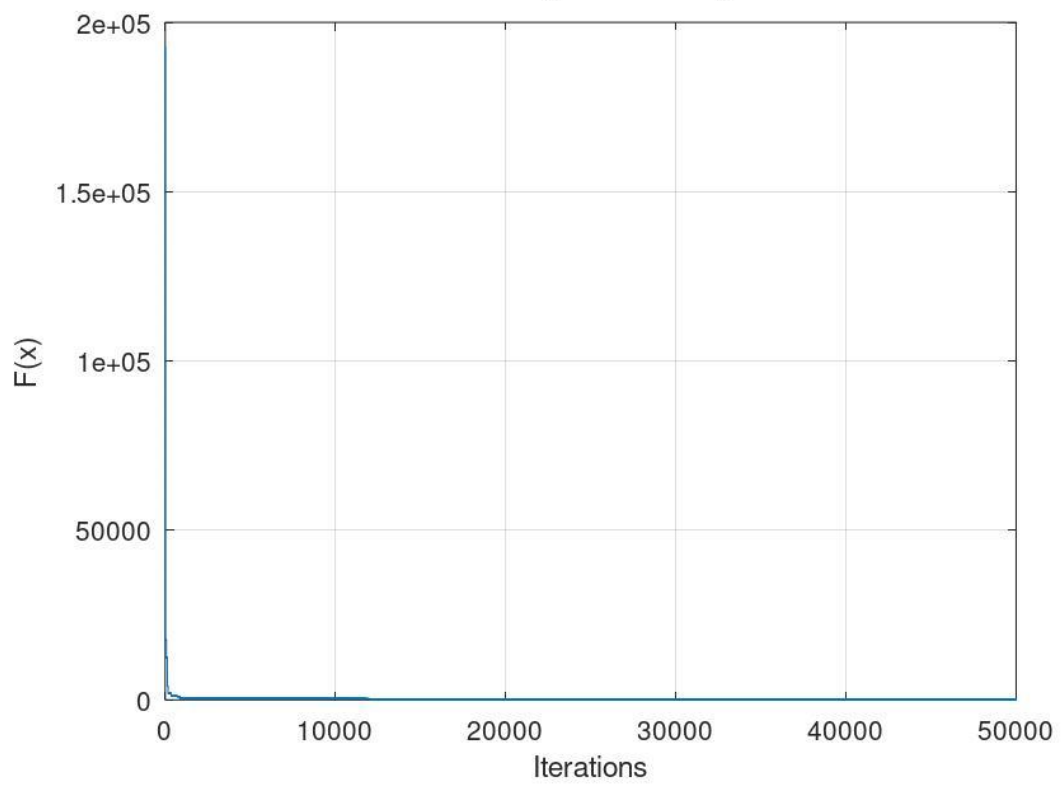


Convergence History





Convergence History



Appendix C

Solid and Voided Slab Bridge Design Charts Code

Appendix C Solid and Voided Slab Bridge Design

Charts Code

```
1 close all;
2 clc;
3 clear;
4 L = input("Enter Section Width(mm): ");
5 H = input("Enter Section Thickness(mm): ");
6 D1 = input("Enter D1(mm): ");
7 D2 = input("Enter D2(mm): ");
8 edged=100;
9 innerd=(L-2*D1-D2-edged*2)/2;
10 A=L*H-pi()/4*D1^2-pi()/4*D2^2;
11 Yb=H/2;
12 I=L*H^3/12-pi()/4*D1^4/64-pi()/4*D2^4/64;
13 J=A^4/(40*I);
14 Sb=I/(H/2);
15 W = 13.45;
16 %voids dimensions limits
17 if (H-max(D1,D2))/2<75
18 disp("voids dimensions limit 1 violated");
19 return;
20 %elseif (pi()/4*D1^2+pi()/4*D2^2)/A>0.4
21 % disp("voids dimensions limit 2 violated");
22 % return;
23 %elseif max(D1,D2)/H>0.6
24 % disp("voids dimensions limit 2 violated");
25 % return;
26 elseif innerd<75 && D2~=0
27 disp("voids clc limit 1 violated");
28 return;
29 elseif innerd*2<75 && D2==0
30 disp("voids clc limit 2 violated");
31 return;
32 end
33 n=ceil(W*1000/L);
34 if n<5
35 disp("number of slabs is below limit, increase bridge width");
36 W = input("Enter Bridge Width(m): ");
37 elseif n>20
38 disp("number of slabs is above limit, reduce bridge width");
39 W = input("Enter Bridge Width(m): ");
40 endif
41 i=1;
42 S=9;
43 while S(i) <= 25
44 %The gap should be at least 1.5 times the sweep, for span >= 12 m g=0.75 in, for span <= 12 m g=0.5 in
45 if S>=12
46 g=0.75*25.4*(n-1);
47 elseif S<12
48 g=0.5*25.4*(n-1);
49 end
50 %Distribute dead loads assuming a uniform distribution across the width using the following
51 %dead load distribution factor DFDL = (Slab width + gap) / (total bridge width)
52 DFDL=(L+0.75*25.4)/(W*1000);
53 %DEAD LOAD ANALYSIS
54 DC1= 24*A*10^-6;
55 VDC1(i)=DC1*S(i)/2;
56 MDC1(i)=DC1.*S(i).^2/8;
57 DC2=2*3.65*DFDL;
```

```

58 VDC2(i)=DC2.*S(i)/2;
59 MDC2(i)=DC2.*S(i).^2/8;
60 DW=22.07*0.075*(W-2*0.61)*DFDL;
61 VDW(i)=DW.*S(i)/2;
62 MDW(i)=DW.*S(i).^2/8;
63 %LIVE LOAD ANALYSIS
64 Vtruck(i) = max((145*0+145*4.3+35*8.6)./S(i),145*2+35-(145*0+145*4.3+35*8.6)./S(i));
65 Mtruck(i) =
(145*2+35-((S(i)./2-4.3)*145+145.*S(i)/2+35.*(S(i)/2+4.3))./S(i)).*S(i)/2-145.*(S(i)./2-(S(i)./2-4.3));
66 Vtandem(i) = max((110*0+110*1.2)./S(i),110*2-(110*0+110*1.2)./S(i));
67 Mtandem(i) = (110*2-(110.*S(i)/2+110.*(S(i)./2+1.2))./S(i)).*S(i)./2-110.*(S(i)./2-S(i)./2);
68 Vlane(i)= 9.3.*S(i)./2;
69 Mlane(i)= 9.3.*S(i).^2/8;
70 Vmax(i)=max(Vtruck(i) .*1.33+Vlane(i),Vtandem(i) .*1.33+Vlane(i));
71 Mmax(i)=max(Mtruck(i) .*1.33+Mlane(i),Mtandem(i) .*1.33+Mlane(i));
72 IJ=0.54*H/L+0.16;
73 if(2.5*n^-0.2>=1.5)
74 k=2.5*n^-0.2;
75 else
76 k=1.5;
77 end
78 DFM1(i)=k.*((L+0.75*25.4)./(33.3.*S(i).*1000./12)).^0.5*IJ^0.25;
79 DFM2(i)=k.*((L+0.75*25.4)./(305*25.4)).^0.6.*((L+0.75*25.4)./(12.*S(i).*1000./12)).^0.2*(IJ)^0.06;
80 DFM(i)=max(DFM1(i),DFM2(i));
81 if I>=40000*25.4^4 && I<=610000*25.4^4 && J>=25000*24.4^4 && J<=610000*25.4^4
82 DFV1(i)=((L+0.75*25.4)./(130.*S(i).*1000/12)).^0.15*IJ^0.05;
83 DFV2(i)=((L+0.75*25.4)./(156*25.4))^0.4*((L+0.75*25.4)./(12.*S(i).*1000/12)).^0.1.*IJ
.^0.05*max(1,(L+0.75*25.4)/(48*25.4));
84 DFV(i)=max(DFV1(i),DFV2(i));
85 else
86 DFV(i)=DFM(i);
87 end
88 VLL(i)=Vmax(i).*DFV(i);
89 MLL(i)=Mmax(i).*DFM(i);
90 %NO. OF STRANDS
91 fpu=1860;
92 fc=35;
93 %Assume 38 mm cover, 12mm stirrups, 15.24 mm strands, offset from bottom is calculated as follows:
94 offset=round((38.1+12+15.24/2)/5)*5;
95 e=Yb-offset;
96 fpj=0.75*fpu;
97 loss=20;
98 Astrand=140;
99 fallow=0.5*sqrt(fc');
100 fbt(i)=((MDC1(i)+MDC2(i)+MDW(i)+1*MLL(i))./Sb).*10^6;
101 fbtn(i)=fbt(i)-fallow;
102 Pp(i)=((Sb*A)/(Sb+e*A)).*fbtn(i);
103 Ppstr(i)=fpj*Astrand*(1-loss/100);
104 nmin(i)=Pp(i)./Ppstr(i);
105 if mod(ceil(nmin(i)),2)==0
106 ns(i)=ceil(nmin(i));
107 else
108 ns(i)=ceil(nmin(i))+1;
109 end
110 %consider 50 mm c/c spacing of strands, 60mm from girder end to center of strands
111 nsmax=floor((L-70)/50);
112 if ns(i)>nsmax
113 disp("number of strands is exceeding the allowable for span length");
114 disp(S(i));
115 end
116 %spacing of tendons or groups of tendons in at least one direction shall not exceed the smaller of
eight times the slab thickness and 1.5 m.
117 if (L-60*2)./(ns-1)<min(8*H,1500);

```

```

118 disp('spacing of strands is exceeding the maximum for span length');
119 disp(S(i));
120 end
121 Aps(i)=140.*ns(i);
122 Pj(i)=Aps(i).*fpj./1000;
123 %PRESTRESS LOSSES
124 Ep=197;
125 Ect=4800*sqrt(fc)/1000;
126 initial(i)=5;
127 Pi(i)=0.75.*fpu.*Aps(i).*(1-initial(i)/100);
128 Mi(i)=Pi(i).*e;
129 fcgp(i)=Pi(i)./A+Mi(i).*e./I-MDC1(i).*10^6.*e./I;
130 Dfpes(i)=Ep./Ect.*fcgp(i);
131 calc(i)=Dfpes(i).*100./(fpu*0.75);
132 %to stop the while loop to run to infinity
133 iter(i) = 0;
134 while (abs(initial(i) - calc(i)) > 0.1 && iter < 20)
135 initial(i)=calc(i);
136 Pi(i)=0.75.*fpu.*Aps(i).*(1-initial(i)/100);
137 Mi(i)=Pi(i)*e;
138 fcgp(i)=Pi(i)/A+Mi(i)*e/I-MDC1(i).*10^6*e/I;
139 Dfpes(i)=Ep/Ect*fcgp(i);
140 calc(i)=Dfpes(i)*100/(fpu*0.75);
141 iter(i) = iter(i) + 1;
142 end
143 if(abs(initial(i) - calc(i)) > 0.1)
144 disp('initial losses does not match calculated');
145 end
146 fpi=0.75*fpu;
147 fci=0.8*fc;
148 gst=35/(7+fci);
149 %gh=1.0 For 70% humidity
150 Dfplt(i)=10.*fpi.*Aps(i)./A.*1.*gst+83*1.*gst+17;
151 Dfpt(i)=Dfpes(i)+Dfplt(i);
152 Dfp(i)=Dfpt(i)./fpi.*100;
153 %SERVICE LIMIT STATE
154 Pp(i)=(0.75*fpu-Dfpes(i)-Dfplt(i)).*Aps(i);
155 ftop(i)=-1.*Pp(i)./A+Pp(i).*e./Sb-(MDC1(i)+MDC2(i)+MDW(i)+MLL(i)).*10^6./Sb;
156 if abs(ftop(i))>0.6*fc
157 disp('top stress is exceeding the allowable for span length');
158 disp(S(i));
159 end
160 fbot(i)=-1.*Pp(i)./A-Pp(i).*e./Sb+(MDC1(i)+MDC2(i)+MDW(i)+MLL(i)).*10^6./Sb;
161
162 if abs(fbot(i))>0.5*sqrt(fc)
163 disp('bottom stress is exceeding the allowable for span length');
164 disp(S(i));
165 end
166 fstrand(i)=fpi-Dfpt(i);
167 if abs(fstrand(i))>0.8*(0.9*1860)
168 disp('strands stress is exceeding the allowable for span length');
169 disp(S(i));
170 end
171 %FLEXURAL DESIGN
172 %38 mm cover, 12 mm stirrups, 16 mm reinf. dia
173 ds=2*Yb-25-12-16/2;
174 if fc<=28
175 B1=0.85;
176 else
177 B1=max(0.85-0.05*(fc-28)/7,0.65);
178 end
179 dp=Yb+e;
180 hv=Yb-D1/2;

```

```

181 fs=460;
182 %assuming As= 0, As'=0
183 As=0;
184 fr=0.63 *sqrt(fc);
185 %k= 0.28 for low- relaxation steel, (fpy/fpu) = 0.9
186 %calculating c assuming rectangular section
187 c(i)=(Aps(i) .*fpu+As .*fs-0)./(0.85*fc*B1*L+0.28 .*Aps(i) .*fpu ./dp);
188 if c(i)<hv
189 %disp("rectangular section assumption is applicable");
190 fpe(i)=fstrand(i);
191 fps(i)=fpu .* (1-0.28 .*c(i) ./dp);
192 Mn(i)=(Aps(i) .*fps(i) .* (dp-c(i)*B1/2)+As .*fs .* (ds-c(i)*B1/2)) ./10^6;
193 else
194 %disp("rectangular section assumption is not applicable, recalculate c for T-section behavior");
195 c(i)=(Aps(i).*fpu+As .*fs-0.85*fc*(L-(L-2*D1-D2)))./(0.85*fc*B1*L+0.28 .*Aps(i) .*fpu ./dp);
196 fpe(i)=fstrand(i);
197 fps(i)=fpu .* (1-0.28 .*c(i) ./dp);
198 Mn(i)=(Aps(i)
.*fps(i) .* (dp-c(i)*B1/2)+As .*fs .* (ds-c(i)*B1/2)-0+0.85*fc*(L-(L-2*D1-D2))*hv*(c(i)*B1/2-hv/2))./10^6
;
199 end
200 %phi=1 for flexural capacity of section
201 Mr(i)=1*Mn(i);
202 Mu(i)=1.25.*(MDC1(i)+MDC2(i))+1.5.*(MDW(i))+1.75.*(MLL(i));
203 if Mr(i)<Mu(i)
204 disp("moment capacity is exceeded, As required calculation shall be considered for span length");
205 disp(S(i));
206 end
207 fcpe(i)=Pp(i) ./A+Pp(i) .*e ./Sb;
208 Mdnc(i)=MDC1(i)+MDC2(i);
209 Mcr(i)=(Sb*(1.6*fr+1.1 .*fcpe(i))-Mdnc(i) .*10^6*(Sb/Sb-1))./10^6;
210 if Mr(i)<min(1.33 .*Mu(i),Mcr(i))
211 disp("minimum reinforcement check is not confirmed for span length");
212 disp(S(i));
213 end
214 cost(i) = ((W .*1000- g ) ./ L) .* (A ./1000 .^2 .*33+ns(i) .*1.102 .*360 ./907);
215 S(i+1)= 9 + i;
216 i=i+1;
217 end

```

Appendix D

AASHTO/PCI vs Optimum Sections Design Charts and Tables

Appendix D AASHTO/PCI vs Optimum Sections

Design Charts and Tables

SI-36																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	9	11	13	15	18	20	23	26	28	31	34	38	41	44	48	51	55
cost BHD/m	196.98	209.68	222.38	235.08	247.77	273.17	285.87	298.56	323.96	336.65	362.05	374.75	400.14	425.54	438.23	463.63	489.02
SII-36																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	6	7	9	10	12	14	15	17	19	21	24	26	28	30	33	35	38
cost BHD/m	186.59	186.59	199.29	211.99	211.99	224.69	237.38	250.08	262.78	275.48	288.17	300.87	326.27	338.96	351.66	364.36	377.06
SIII-36																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	4	5	6	8	9	10	12	14	15	17	19	21	22	24	26	28	31
cost BHD/m	189.80	189.80	202.49	202.49	215.19	227.89	227.89	240.59	253.28	265.98	278.68	291.38	304.07	316.77	329.47	342.17	354.86
SIV-36																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	3	4	5	6	7	8	10	11	12	14	15	17	18	20	22	23	25
cost BHD/m	189.12	189.12	201.81	201.81	214.51	227.21	227.21	239.91	252.60	252.60	265.30	278.00	290.69	290.69	303.39	316.09	328.79
SI-48																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	12	14	17	20	23	26	29	33	37	40	44	49	53	57	62	66	71
cost BHD/m	200.73	210.28	219.83	229.38	248.48	258.03	277.14	296.24	315.34	334.44	353.54	372.64	391.74	410.84	429.94	458.59	477.69
SII-48																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	8	9	11	13	15	17	19	22	24	27	30	33	36	39	42	45	48
cost BHD/m	170.50	180.05	189.60	199.15	208.70	218.25	227.80	237.35	256.46	266.01	275.56	294.66	304.21	323.31	332.86	351.96	371.06
SIII-48																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	5	7	8	9	11	13	15	17	19	21	23	26	28	31	33	36	38
cost BHD/m	174.71	184.26	184.26	193.81	203.36	212.91	222.46	232.01	241.56	251.11	260.66	270.21	289.31	298.86	308.41	317.96	337.06
SIV-48																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	4	5	6	7	9	10	12	14	16	18	19	21	24	26	28	30	33
cost BHD/m	182.56	192.11	201.66	201.66	211.21	220.76	230.31	230.31	239.86	249.41	258.96	268.51	278.07	287.62	297.17	316.27	325.82

OPT sol.1																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	4	6	7	8	10	12	14	15	17	19	21	24	26	28	30	33	35
cost BHD/m	155.58	155.58	163.57	171.56	171.56	179.56	187.55	195.54	203.53	211.52	219.51	227.5	235.49	251.47	259.46	267.45	275.44
OPT sol.2																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	6	7	9	10	12	13	15	17	19	21	23	26	28	30	33	35	38
cost BHD/m	148.1	158.97	169.84	169.84	180.71	191.58	202.45	213.33	224.2	235.07	245.94	256.81	267.68	289.43	300.3	311.17	322.04
OPT sol.3																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	6	7	9	10	12	13	15	17	19	21	23	26	28	30	33	35	38
cost BHD/m	148.1	158.97	169.84	169.84	180.71	191.58	202.45	213.33	224.2	235.07	245.94	256.81	267.68	289.43	300.3	311.17	322.04
OPT sol.4																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	5	7	8	9	11	13	15	16	18	20	22	25	27	29	31	34	36
cost BHD/m	154.32	164.71	164.71	175.09	185.48	195.86	206.25	216.64	227.02	237.41	247.8	258.18	268.57	278.96	289.34	299.73	320.5
OPT sol.5																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	6	7	8	10	11	13	15	17	19	21	23	25	28	30	32	35	37
cost BHD/m	148.51	159.08	169.66	169.66	180.23	190.81	201.38	211.96	222.53	233.11	243.68	254.26	264.83	275.41	296.56	307.13	317.71
OPT sol.6																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	5	6	8	9	11	12	14	16	18	20	22	24	26	29	31	33	36
cost BHD/m	152.35	161.25	161.25	170.15	179.05	187.94	196.84	196.84	205.74	214.63	232.43	241.32	250.22	259.12	268.01	276.91	285.81
OPT sol.7																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	6	7	8	10	11	13	15	17	19	21	23	26	28	30	33	35	38
cost BHD/m	151	160.54	170.09	170.09	179.63	189.17	198.72	208.26	217.81	227.35	236.9	246.44	255.98	275.07	284.62	294.16	303.7
OPT sol.8																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	5	6	7	8	10	12	14	16	17	19	22	24	26	28	31	33	35
cost BHD/m	154.94	154.94	163.1	171.26	179.41	179.41	187.57	195.73	203.89	212.05	220.2	228.36	236.52	252.84	260.99	269.15	277.31

OPT sol.9																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	5	6	7	8	10	12	14	16	17	19	22	24	26	28	31	33	35
cost BHD/m	154.94	154.94	163.1	171.26	179.41	179.41	187.57	195.73	203.89	212.05	220.2	228.36	236.52	252.84	260.99	269.15	277.31
OPT sol.10																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	4	6	7	8	10	12	13	15	17	19	21	23	26	28	30	33	35
cost BHD/m	156.24	156.24	164.07	171.9	171.9	179.73	187.56	195.39	203.22	211.05	218.88	226.71	234.54	242.37	258.02	265.85	273.68
OPT sol.11																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	4	6	7	8	10	12	13	15	17	19	21	23	26	28	30	33	35
cost BHD/m	156.24	156.24	164.07	171.9	171.9	179.73	187.56	195.39	203.22	211.05	218.88	226.71	234.54	242.37	258.02	265.85	273.68
OPT sol.12																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	2	3	4	5	6	8	9	10	12	13	15	16	18	19	21	23	25
cost BHD/m	166.49	166.49	176.03	176.03	185.58	185.58	195.12	204.67	204.67	214.21	223.75	233.3	233.3	242.84	252.39	261.93	271.47
OPT sol.13																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	2	3	4	5	6	7	9	10	11	13	14	16	18	19	21	23	25
cost BHD/m	165.03	173.55	173.55	182.06	182.06	190.58	199.1	199.1	207.61	216.13	224.64	224.64	233.16	241.67	250.19	258.7	267.22
OPT sol.14																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	2	3	4	5	6	7	8	10	11	13	14	16	17	19	21	22	24
cost BHD/m	162.89	171.59	171.59	180.29	180.29	188.99	197.69	197.69	206.38	215.08	223.78	223.78	232.48	241.17	249.87	258.57	267.27
OPT sol.15																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	2	3	4	5	6	7	8	10	11	13	14	16	17	19	21	22	24
cost BHD/m	162.89	171.59	171.59	180.29	180.29	188.99	197.69	197.69	206.38	215.08	223.78	223.78	232.48	241.17	249.87	258.57	267.27
OPT sol.16																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	1	2	3	4	5	7	8	9	11	12	14	15	17	18	20	22	24
cost BHD/m	169.79	177.68	177.68	185.56	185.56	193.44	193.44	201.32	209.21	217.09	217.09	224.97	232.85	240.74	248.62	248.62	256.5
OPT sol.17																	
Span	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ns	2	3	4	4	6	7	8	10	11	12	14	15	17	19	20	22	24
cost BHD/m	166.91	175.12	175.12	183.34	183.34	191.55	199.77	199.77	207.98	216.2	216.2	224.42	232.63	240.85	249.06	257.28	257.28

الملخص

يهدف هذا البحث الى تحسين التكلفة الإجمالية لجسور الدعامات البسيطة ذات الألواح الخرسانية المجهدة مسبقاً الصلبة والمفرغة من خلال تحديد خصائص المقطع العرضي المثالية والفعالة من حيث التكلفة والمتوافقة مع النظام الدولي للوحدات (SI) ، مثل عرض وارتفاع الألواح الخرسانية، بالإضافة إلى عدد وأبعاد الفراغات. تشمل متغيرات التصميم طول الامتداد، والعرض الكلي للجسر، ومساحة الفولاذ المجهد مسبقاً، وأبعاد المقاطع العرضية للألواح الخرسانية المجهدة مسبقاً. تُتبع مواصفات AASHTO القياسية لجسور الطرق السريعة في تصميم الألواح الخرسانية المجهدة مسبقاً والمفرغة.

تُطبق خوارزمية البحث المتناغم المعدلة مع الأخذ في الاعتبار القيود الهندسية، وإجهاد العمل، والقوة القصوى للحصول على الأبعاد الأكثر فعالية من حيث التكلفة للألواح المدروسة. وتُستخدم التكلفة الإجمالية للجسر كمعيار للأمتلية. يتم تصميم أمثلة حسابية، ويُبرمج كود حاسوبي للحصول على التصميم الأمثل. بالإضافة إلى ذلك، تم تنفيذ مخططات تصميم متعددة بناءً على المقاطع المثلى المُحصل عليها في مقابل طول امتداد الجسر.

خلصت الدراسة إلى أنه يمكن تحقيق تحسين التكلفة الإجمالية لجسور الخرسانة المُجهدة مسبقاً ذات الألواح الخرسانية الصلبة والمفرغة بنجاح باستخدام خوارزمية البحث المتناغم المعدلة ، حيث تم تحقيق خفض للتكلفة بنسبة تتراوح بين (12-21)% مقارنةً بمنتجات AASHTO القياسية.

جامعة البحرين



قسم الهندسة المدنية

كلية الهندسة

**تحسين جسور الألواح الصلبة والمجوفة من الخرسانة سابقة
الإجهاد وفقاً لمعايير AASHTO باستخدام خوارزمية البحث
المتناغم المعدلة**

أطروحة مقدمة كجزء من متطلبات الحصول على درجة
الماجستير في الهندسة المدنية

إعداد

آلاء زهير عبدالرضا محمد مكي البحارنة
الرقم الجامعي: 20145033

إشراف

د. سميح إرهان
أستاذ مساعد
جامعة البحرين

مملكة البحرين
مايو 2025