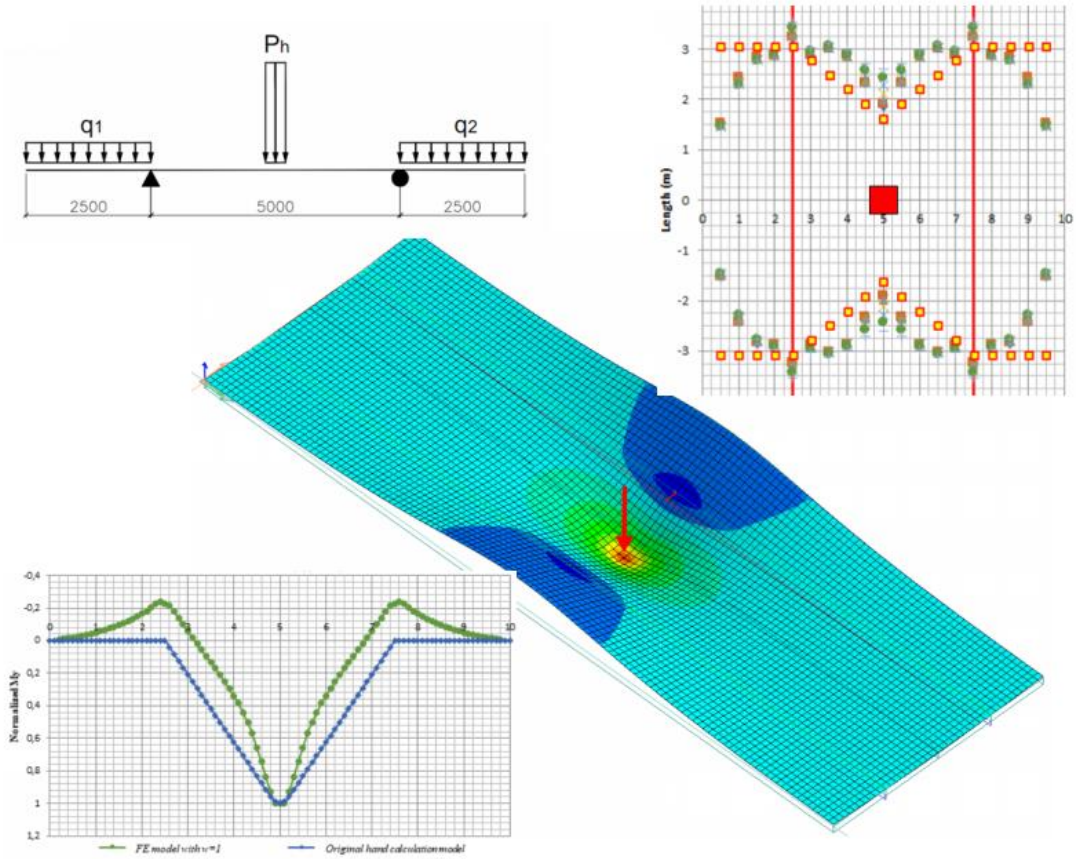


Distribution widths for concrete slabs

Master's thesis in the master's program Structural engineering and building technology

AHMED ALANBARI
DA YUAN

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING
DIVISION OF STRUCTURAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020
www.chalmers.se

MASTER'S THESIS ACEX30

Distribution widths for concrete slabs

Master's thesis in the master's program Structural engineering and building technology

AHMED ALANBARI
DA YUAN

Department of Architecture and Civil Engineering
Division of Structural Engineering
Concrete Structures
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2020

Department of Architecture and Civil Engineering
AHMED ALANBARI
DA YUAN

© AHMED ALANBARI, DA YUAN, 2020

Examensarbete ACEX30
Institutionen för arkitektur och samhällsbyggnadsteknik
Chalmers tekniska högskola, 2020

Department of Architecture and Civil Engineering
Division of Structural Engineering
Concrete Structures
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Distribution widths for concrete slabs
Master's thesis in the master's program Structural engineering and building technology.

AHMED ALANBARI

DA YUAN

Department of Architecture and Civil Engineering
Division of Division Name
Research Group Name
Chalmers University of Technology

ABSTRACT

The distribution width of sectional forces and moments for a one-way concrete bridge deck highly influences the amount of reinforcement required for the deck. In engineering practice, two calculation methods are used: simplified analytical calculations and finite element (FE) analysis. Compared to FE analysis, the analytical calculation is more convenient to use but due to the lack of accurate guidelines, it is too conservative and hence overestimates the reinforcement. This thesis focuses on providing a more accurate expression of distribution width for the analytical calculation by utilizing FE parametric study to enhance the accuracy of the analytical calculation.

The approach to determine the distribution width in this thesis is by comparing the sectional force from a linear 2D shell FE model to an analytical calculation model. The ratio between sectional forces of these two models was the approach to finding the new expression. Based on this approach a full parametric study was required where multiple models were analyzed, and results were collected. Following that, the expressions of the distribution widths were established based on all the results collected from these various models.

To simplify the task, a specific model was chosen representing a common bridge deck, and this model was referred to as the "base model". The whole approach was tested on the base model first, and several problems and limitations were identified. One of them was the constrain caused by the cantilever in the FE model for a load at the mid-span, which led to a mismatch of the zero-moment position between the two models. Consequently, a "dummy load" was introduced into the analytical calculation model. Another problem was the extensive data processing to include the torsional moment in the design;this was avoided by choosing the average width as zero.

Graphical curve fitting was utilized to create the expressions of the distribution widths and other related calibration variables such as dummy loads. Furthermore, results comparison envelopes between the new expression and the raw FE results were used to check the accuracy of the expressions.

Finally, recommendations for use in engineering practice is given, and several issues are discussed such as the "cutoff" region for shear force expression, and the influence of the support stiffness and slab orthotropy on the new expression.

Keywords: linear 2D shell FE model, analytical calculation model, isotropic element, dummy load, averaging width, equivalent width, distribution width, local effect, cutoff region, punching shear region, envelope.

Contents

1 INTRODUCTION 1
1.1 Background 1
1.2 Purpose 1
1.3 Method 2
1.4 Limitations 3
1.5 Outline of the report 3

2 LITERATURE STUDY 5
2.1 Introduction 5
2.2 Bestämmelser för betongkonstruktioner B7 [Statens Betongkommitté (1968)] 6
2.2.1 Introduction 6
2.2.2 Moment distribution 6
2.2.3 Shear force distribution 9
2.3 Boverkets handbok om betongkonstruktioner BBK 10
2.3.1 Introduction 10
2.3.2 Moment distribution 10
2.3.3 Shear force distribution in BBK 04 [Boverket,(2004)] 10
2.3.4 Shear force distribution in BBK 79 [Statens Betongkommitté (1979)] 11
2.4 Structural use of concrete BS 8110-1:1997 [BSI (1997)] 12
2.4.1 Introduction 12
2.4.2 Moment distribution 12
2.4.3 Shear force distribution 12
2.5 *fib* Model Code 2010 [fib (2013)] 13
2.5.1 Introduction 13
2.5.2 Moment distribution 13
2.5.3 Shear force distribution 13
2.6 Krav Bärighetsberäkning v6.0 [Trafikverket (2019a)] 14
2.6.1 Introduction 14
2.6.2 Moment distribution 14
2.6.3 Shear force distribution 14
2.7 Råd Bärighetsberäkning v6.0 [Trafikverket (2019b)] 15
2.7.1 Introduction 15
2.7.2 Moment distribution 15
2.7.3 Shear force distribution 15
2.8 Recommendations for finite element analysis for the design of reinforced concrete slabs [Pacoste *et al.* (2012)] 16
2.8.1 Introduction 16
2.8.2 Moment averaging width 16
2.8.3 Shear force averaging width 17
2.8.4 The relation between averaging width and compression zone height 17

2.8.5 Moment averaging width for cantilever slabs 18
2.8.6 Shear force averaging width in cantilever slab 19

2.9 Other guidelines 21
2.9.1 Introduction 21
2.9.2 Moment distribution 21
2.9.3 Shear force distribution 21

2.10 Summary of existing guidelines 23

3 BASE MODEL ANALYSIS 27

3.1 Introduction 27

3.2 FE model description 27
3.2.1 Software 27
3.2.2 Geometry 27
3.2.3 Boundary conditions 28
3.2.4 Load 28
3.2.5 Element and mesh 29
3.2.6 Analysis 29
3.2.7 Analysis results 30
3.2.8 Processed analysis results 32

3.3 Analytical calculations 37
3.3.1 The original analytical calculation model 37
3.3.3 Result comparison with the FE model 39
3.3.4 The modified analytical calculation model 40
3.3.5 Equivalent width 44

3.4 Discussion and conclusion of the base model study and prediction for full parametric study. 47
3.4.1 Dummy load 47
3.4.2 Influence and dependency of the averaging width. 48
3.4.3 Influence of torsional moment 48
3.4.4 The selected averaging width for parametric study 49

4 PARAMETRIC ANALYSIS 50

4.1 Introduction 50

4.2 General process 50
4.2.1 Selecting the base model variations 50
4.2.2 FE analyzing and data processing 51
4.2.3 Curves fitting and expression 52
4.2.4 Envelope verification 52

4.3 Program controlling and data processing by Python scripting 53

5 RESULTS 56

5.1 Introduction 56

5.2 The expression of distribution width (b_{eff}) 56
5.2.1 The distribution width for the moment 56

5.2.2 The distribution width for shear force 57
5.3 Expression of the dummy load 60
5.3.1 Dummy load for bridge slab with constant effective depth 60
5.4 Result verification 61
5.4.1 Base model envelope comparison 61
5.5 Comparison with existing guidelines for distribution width 65
5.5.1 The distribution width for the moment 65
5.5.2 The distribution width for shear force 66

6 DISCUSSION 69
6.1 Introduction 69
6.2 General method 69
6.3 Difficulties 69
6.4 Consideration for using the recommendations 70
6.4.1 The updated analytical calculation model 70
6.4.2 Punching shear region as the "cutoff" region 70
6.4.3 Influence of prestressing, support stiffness and element orthotropy 71
6.4.4 Optimal design 71

7 CONCLUSIONS 72
7.1 Introduction 72
7.2 Conclusion remarks 72
7.3 Recommendations for future studies 73

8 REFERENCES 74

9 APPENDIX A: CONVERGENCE STUDY 77

10 APPENDIX B: BASE MODEL ANALYSIS RESULTS 78

11 APPENDIX C: PARAMETRIC STUDY RESULTS 129
11.1 First Variation Constant thickness $\boldsymbol{t = 0.45m}$ 129
11.2 The second Variation changing the concrete type to C12/15 154
11.3 Third Variation $\boldsymbol{lm + 2lk = 20m}$ 175
11.4 Fourth Variation $\boldsymbol{\alpha = 0.25,\ lm = 5}$ 200
11.5 Fifth Variation $\boldsymbol{\alpha = 0.75,\ lm = 5}$ 218
11.6 Sixth Variation $\boldsymbol{t = [0.15, 0.3, 0.3]m}$ 249
11.7 Seventh Variation $\boldsymbol{t = [0.15, 0.3, 0.45]m}$ 274
11.8 Eighth Variation $\boldsymbol{t = [0.15, 0.3, 0.15]m}$ 299

12 APPENDIX D: PYTHON SCRIPT 324
12.1 Directory structure: 324
12.2 _coor.bsc 325
12.3 _M2.bsc 326
12.4 _My.bsc , _Mxy.bsc, and _Vy.bsc 328
12.5 Main.py 330
12.6 Plate.py 333
12.7 Parser.py 335
12.8 x_section_w.py 338
12.9 y_intergration_w.py 342
12.10 handcal.py 344
12.11 exp_env.py 350
12.12 XLSX.py 353
12.13 fdAPI_wrapper_mod.py 363

Preface

This Master's Thesis focuses on providing a guideline for the distribution of concentrated loads on concrete slabs. Such a guideline is needed to get more accurate analytical calculations of design forces without the need to do case-specific FE analysis. The guideline will be specific for concrete bridge decks supported on two longitudinal bridge girders.

The work was performed during 2020, from January to June, at Norconsult AB, as a final part of the master's program Structural Engineering and Building Technology at Chalmers University of Technology, Sweden. The thesis was written in collaboration with the Department of Civil and Environmental Engineering, Division of Structural Engineering, Concrete Structures at Chalmers University of Technology.

We would like to show our gratitude to our supervisor, Emanuel Trolin, for his excellent guidance and valuable support throughout the work of this Thesis. We would also like to thank our examiner, Professor Mario Plos, Chalmers, for his guidance, and insightful comments.

AHMED ALANBARI

DA YUAN

Göteborg, 2020

Notations

The notations in the below tables are used generally throughout this project. It should be noted that other cases or reference specific notations are used in chapter 2 where each of these cases has its local notation definition that is not related to these notations.

Greek upper-case letters

Greek lower-case letters

α	The ratio of cantilever length to the length of the span. (Unitless)
μ	Constant that governs the ratio of the torsional moment in the design moment.

Roman upper-case letters

E	Young's modulus of elasticity (GPa)
FE	Finite element
CSV	Comma-separated values file
M_h	Analytical calculation moment value (kN.m)
M_{av}	Averaged internal moment value (kN.m/m)
M_{ry}	Design moment value (kN.m/m)
P_{FE}	Finite element distributed load (kN/m^2)
P_h	Analytical calculation point load value (kN)
T_{xy}	Torsional moment value (kN.m)
V_{yz}	Averaged internal shear force value (kN/m)
V_h	Analytical calculation shear force value (kN)
V_{av}	Averaged shear force value (kN/m)
XML	Extensible Markup Language file
XLS	Excel Microsoft Office Open XML Format Spreadsheet file

Roman lower-case letters

b_{eq}	Equivalent distribution width (m).
b_{eff}	Distribution width (m).
b_{load}	Width of load(m).
b_{sup}	Width of support(m).
d	The effective depth (m).
d_{edge}	The effective depth at the edge of the cantilever (m).
d_{supp}	The effective depth at the support(m).
d_{mid}	The effective depth at the midspan(m).
h	The total thickness of the cantilever slab(m).
l_m	Span length (center to center) (m).
l_k	Cantilever length (edge to center) (m).
l_s	Slab length (m).
l_{tot}	Total width of the slab (sum of span length and two cantilevers) (m).
l_{load}	Length of load(m).

q_1, q_2	Dummy loads (kN/m)
t_c	The thickness of the cover(m).
t_f	The thickness of the filling(m).
w	Averaging width (m)
x	Distance between center load to the center of the nearest support (m).
x_{crit}	Distance between the center of the load to the critical section(m).

1 Introduction

1.1 Background

One of the most common types of bridge decks is a concrete slab supported by two longitudinal bridge girders. The slab is mainly carrying loads in the direction perpendicular to the girders but, for concentrated loads, it distributes the load in both directions. For a concrete bridge deck, two methods are generally used for the calculations of sectional forces and moments in engineering practice: Analytical simplified calculations and numerical analysis with the finite element method (FEM).

The finite element (FE) analysis is normally made using slab or shell elements and is based on the condition of linear material response. From the FE analysis, the moment to be used in reinforcement design is determined by averaging the design moments from the linear analysis over a certain redistribution width perpendicular to the reinforcement direction. This width is governed by the plastic redistribution capacity of the slab, which in turn depends on the amount of reinforcement. As compared to an analytical simplified calculation, a linear FE analysis gives a more accurate estimation of the force distribution. However, it is more time-consuming, and it is not convenient to use in all cases.

In an analytical calculation, the reinforcement moment is achieved by approximating the slab to act as beam segments supported by the bridge girders. The concentrated loads are distributed over a predetermined width, the distribution width. Since this is not yet treated in Eurocode 2 [CEN (2004)], the engineers must find out a suitable distribution width from other guidelines and recommendations. The choice of the distribution width highly affects the results of the analytical calculation. Currently, conservative load distribution widths are used in engineering practice, resulting in a higher amount of reinforcement than required according to FE analysis. This rise the need for developing a guideline with better accuracy for determining the distribution width of common types of bridge decks to enhance the results of the analytical calculations.

1.2 Purpose

The purpose of this thesis is to create a guideline for analytical calculations with recommendations for the distribution width of concentrated loads. To achieve this targeted guideline, the following aspects will be addressed in the project:

- Explanation of the methods and theories that are used in this thesis.
- Find out how and in which magnitude each parameter will affect the result from a linear FE analysis.
- Raise out the most interesting parameters that will be examined in a parametric study.
- Organize and visualize results from FE analysis for relevant parameters.
- Present a method to determine the distribution width for analytical calculation, based on the results of the parametric FE analysis.
- List the limitation and basic assumptions of the method presented.
- Compare the results from the method presented to existing guidelines.
- Due to the limitations of the presented method give, if possible, suggestions for further calibrations to cover more cases.

- Give recommendations for which aspects must be further investigated to extend and enhance the presented method.

Addressing the above will result in a guideline that can provide a more optimized design based on analytical calculations, in better agreement with linear FE analysis but without the need for case-specific FE analysis.

1.3 Method

Literature study: A literature study was performed that focused on the distribution of load effects (sectional moments and shear forces) in concrete slabs and how they are treated in codes/guidelines.

Simplified FE analysis: Here, a full study was done on a simplified model (Base model). This study started by presenting the FE and analytical models and ended by initiating a simplified expression to evaluate the equivalent width for the base model. During this step, results obtained from FE analysis and analytical calculation were presented. The conclusion of this step highlighted issues or aspects that had to be handled during the following parametric analysis. FE software and other calculation software such as Mathcad or Excel were utilized during this step.

Parametric analysis: In this step, the parameters of the simplified model (Base model) were modified into several variations, and results were collected to find the effect of changing each parameter. Changed parameters included geometries, thicknesses, load positions, and redistribution widths (averaging widths) based on common bridge decks practices. In this step, Python scripting was utilized to automatically run and collect the data from the FE analysis and the analytical analysis to be output in spreadsheets. At the end of this step, all the highlighted aspects from the previous step were studied.

Evaluation of results: Here, the data obtained from the previous steps were evaluated and embedded into the initial equivalent width expression. This step included a comparison of the results with results obtained by the alternative methods found during the first step (Literature study)

The method is summarized in Figure 1.1

Figure 1.1 The method of the study consisted of four subsequent steps starting with the Literature study and ends with Results evaluation

1.4 Limitations

This thesis utilized a simplified FE model for parametric analysis. The reason is that, when using a simplified model, the influence of each parameter can be easily captured. Due to this, limitations are introduced for some aspects regarding the FE model. To simulate a slab carrying the loads predominantly in one direction, supported by two bridge girders, the slab was modeled as an infinitely long slab supported by two stiff longitudinal line supports. Transverse beams or edge beams were not included in the model. The FE analysis was limited to linear elastic analysis for an isotropic plate (stiffness is equal in both main directions) with one individual point load, and the results concerning the design of the main reinforcement in the slab (in the transversal direction of the bridge) were extracted. Geometric parameters were selected to represent the most common geometries in practice.

Due to the above limitations, aspects such as the influence of the edge beam or orthotropic plate were not treated by this project. Non-linear FE analysis was not used to study the non-linear response of the slab. Based on that, the results are usable only in the ultimate limit state.

1.5 Outline of the report

Chapter 1 Introduction.

Chapter 2 This chapter introduces existing methods for estimating the distribution width for analytical calculation from different guidelines.

Chapter 3 This chapter presents a full study on one selected model geometry which was defined as the base model. In this chapter, the approach used to determine the equivalent width is introduced in addition to the FE and analytical calculation models. The basic assumptions and parameters such as description, boundary conditions, loading conditions, etc. were presented together with corresponding results. By the end of this chapter, the approach to calculate the equivalent width was introduced and discussed as this was the initialization step for parametric study in chapter 4.

Chapter 4 This chapter is a continuation and iteration of chapter 3. It focuses on explaining the general process of full parametric analysis and presents the approaches used for establishing and verifying the generated guideline of equivalent width for the analytical calculation model. The last section presents the used automation method and the script structure that was utilized in the parametric study to produce the models, extract data, and verify the results.

Chapter 5 This chapter presents and discusses the results of the parametric study, equivalent width, and dummy loads expressions with their limitations and conditions. It provides an overview of the results of the parametric study and observations on the effect of the changes in each parameter.

Following that, a result comparison between FE analysis and expression-based analytical calculation was initiated using the shear force and moment envelops, and the differences were highlighted.

Chapter 6 This chapter discusses several interesting topics which are the most important from a designer's point of view, and how and when this guideline could be applied. It begins with a discussion of the general method used for this guideline. After that, the difficulties that were faced in the research and considerations of using this guideline are discussed. At the end of this chapter, a discussion around achieving an optimal design is made.

Chapter 7 This chapter layout all the important conclusions of this thesis. In the end, suggestions for further research are given.

2 Literature Study

2.1 Introduction

This chapter describes and summarizes guidelines and codes that treat the subject of *distribution width* for slabs, for both moment and shear force in analytical calculations and FE analysis. Some of these guidelines are not in use today. Still, they are mentioned here to describe the development of the current approaches and provide a result comparison between old practices and the results obtained in this thesis.

It should be noticed that different terms were used to describe the distribution width in these guidelines, like effective width b_{ef} or equivalent width b_{eq}. To avoid confusion as this report listed all these approaches, the term *Distribution Width* replaces the above-mentioned terms to describe the distribution of moment and shear forces in the analytical calculation. As for the FE analysis guidelines, the term *Averaging width* is used. Each of these guidelines has its own set of complementary parameters that were defined locally within each section. All these parameters are unified into one set of parameters in the summary section 2.10

All the approaches presented hereafter are categorized into two main categories, which are related to the type of analysis, namely: *analytical calculation* or *FE analysis*. Each approach has subcategories, according to each reference. Below is a list of the approaches that are presented in detail in this chapter.

Guidelines for distribution widths in analytical calculations

- Bestämmelser för betongkonstruktioner, B7: Allmänna konstruktionsbestämmelser. [Statens Betongkommitté (1968)]:

  This is an old Swedish code for concrete structures, which is often referred to as only "B7"

- Boverkets handbok om betongkonstruktioner BBK 04, [Boverket (2004)]:

  This was the acting code for concrete design in Sweden before it was replaced by Eurocode. It replaced a previous code with the same name:

  Bestämmelser för betongkonstruktioner BBK 79, [Statens Betongkommitté (1979)]

- BS 8110-1:1997 Structural use of concrete. Code of practice for design and construction. [BSI (1997)]

  This is an old British code for "Structural use of concrete".

- *fib* model code for concrete structures 2010 [fib (2013)]:

  The *fib* model code was developed by the International Federation for Structural Concrete, fib. Previous model codes have had a great influence on the current version of Eurocode 2, and model code is an important reference in the ongoing revision of the Eurocodes.

- Krav TRVK Bärighetsberäkning av vägbroar TDOK 2013:0267 [Trafikverket (2019a)] and, Råd TRVK Bärighetsberäkning av vägbroar TDOK 2013:0273 [Trafikverket (2019b)]:

These are codes provided by the Swedish Traffic Administration (Trafikverket) and consist of requirements for the determination and assessment of the structural capacity of existing bridges.

- Other guidelines:
  This section presented other approaches that were studied in the research paper "Transverse load redistribution and effective shear force width in reinforced concrete slabs" [Lantsoght *et al*. (2013)], by Delft University of Technology.

Guidelines for averaging widths in FE analysis.

- Recommendations for finite element analysis for the design of reinforced concrete slabs [Pacoste *et al*. (2012)].

2.2 Bestämmelser för betongkonstruktioner B7 [Statens Betongkommitté (1968)]

2.2.1 Introduction

This is a general code issued by the "Statens Betongkommitte" the year 1969 as a part of the series publication started the year 1949. This series of books provided guidelines for concrete construction.

2.2.2 Moment distribution

B7 stated that the distribution width of the moment for a one-way slab can be evaluated in two directions of the slab as shown in Figure 2.2. The distribution width r in transverse direction (perpendicular to the primary reinforcement) as shown in *Figure 2.1* shall be calculated using:

$$r \leq \frac{3}{4} l + v + f + t \tag{2.1}$$

With a maximum value of:

$$r = 2.5\ \mathrm{m} + v + f + t \tag{2.2}$$

r	:	Moment distribution width perpendicular to the primary reinforcement.
v	:	Load width perpendicular to the primary reinforcement.
l	:	The theoretical span of a one-way slab.
f	:	The thickness of filling (pavement) etc.
t	:	The thickness of the concrete cover to the top main reinforcement.

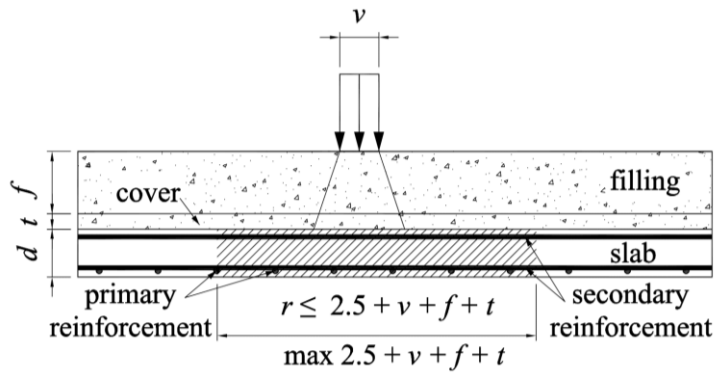


Figure 2.1 Cross-section perpendicular to primary reinforcement. Redrawn from [Statens Betongkommitté. (1968)].

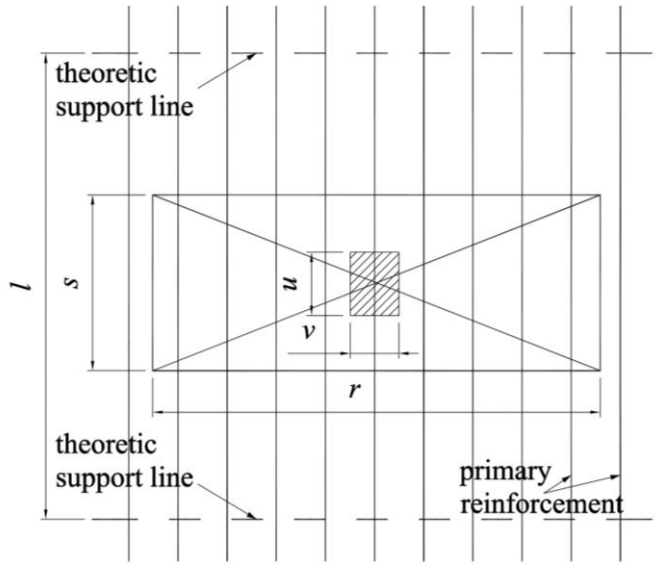


Figure 2.2 Plan of the one-way slab. Redrawn from [Statens Betongkommitté (1968)].

In longitudinal direction (parallel to the main reinforcement), the load may be assumed distributed to the top of the slab with distribution width s as shown in Figure 2.3:

$$s = u + 2(f + t) \tag{2.3}$$

u : Load width parallel to primary reinforcement.

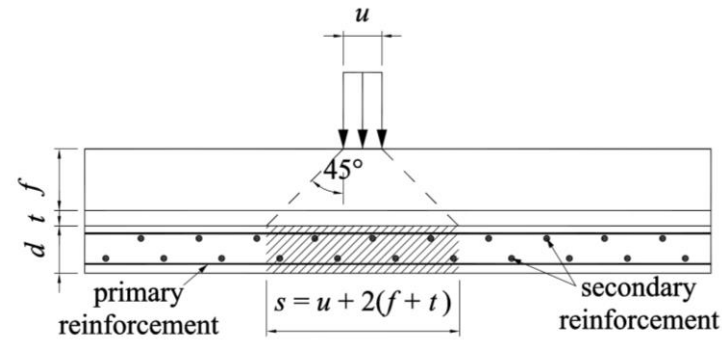


Figure 2.3 Cross-section parallel to primary reinforcement. Redrawn from [Statens Betongkommitté (1968)].

In the case of a concentrated load P acting on a cantilever slab stiffened by an edge beam, when calculating bending stresses in the slab, the distribution width r will be:

$$r = \frac{2}{n} \tag{2.4}$$

$$n = \sqrt[4]{\frac{c}{4EI_1}}$$
$$and \quad c = \frac{3EI_2}{a^3} \tag{2.5}$$

Where:

E	:	Young's modulus of elasticity.
I_1	:	Moment inertia of the edge beam and adjacent portion of the slab up to the load acting point see Figure 2.4.
c	:	Coefficient equal to load intensity parallel to the edge beam divided by the corresponding deflection of the cantilever slab at the load acting point.
I_2	:	Moment inertia of the slab per unit length of the support. See Figure 2.4.
a	:	Distance between the center of the load and the edge of the support.

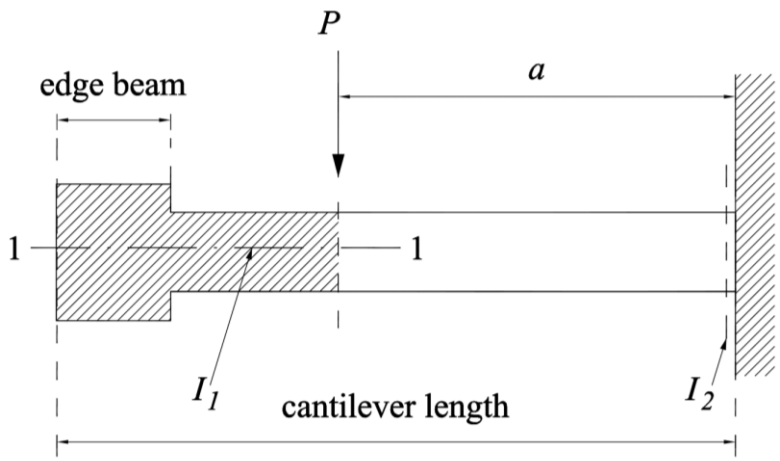


Figure 2.4 Cantilever slab with edge beam. Redrawn from [Statens Betongkommitté (1968)].

The edge beam should be designed for a positive moment:

$$M_k = \frac{P}{4n} \tag{2.6}$$

and designed for a negative moment:

$$M' = -\frac{P}{19n} \tag{2.7}$$

2.2.3 Shear force distribution

For the calculation of distributed shear force, two cases are considered. First when a concentrated load is acting on a slab at the edge of the support. The shear force, in this case, is assumed to be distributed uniformly laterally with a distribution width r_1 as shown in Figure 2.5:

$$r_1 = max\begin{cases} v + 2(f + t + d_{min}) \\ 5d_{min} \end{cases} \tag{2.8}$$

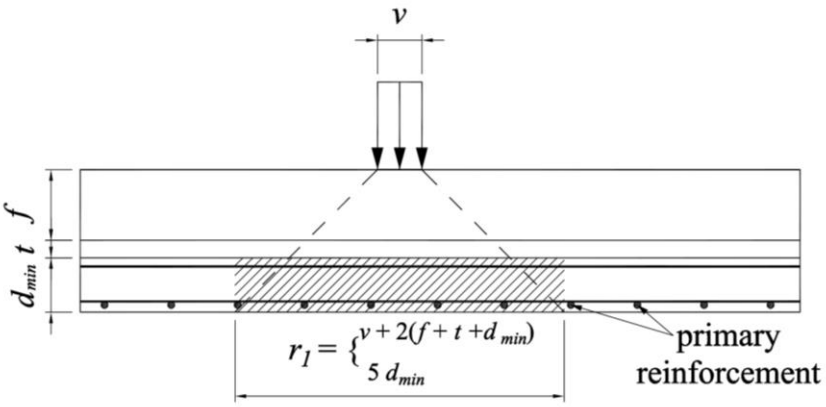


Figure 2.5 Transverse section at the edge of the support. Redrawn from [Statens Betongkommitté (1968)].

Second when a concentrated load at a distance x from the edge of the support, in this case, the distribution width r_1 can be increased by $\frac{3}{4}x$ as shown in Figure 2.6:

$$r_1 + \frac{3}{4}x \tag{2.9}$$

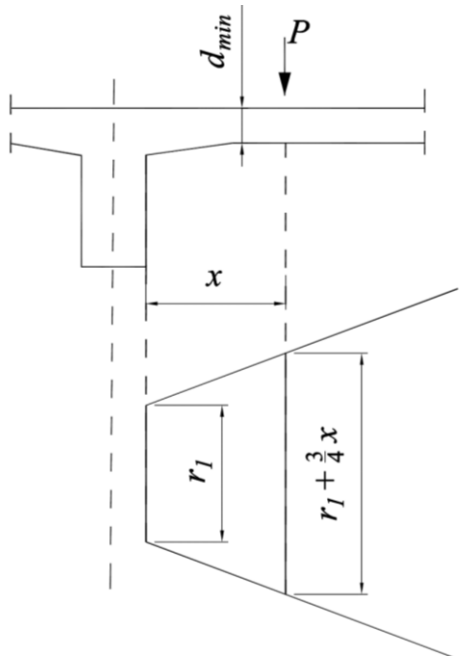


Figure 2.6 Representation of the increment of r_1 value with respect to the location of the concentrated load. Redrawn from [Statens Betongkommitté (1968)].

2.3 Boverkets handbok om betongkonstruktioner BBK

2.3.1 Introduction

BBK04 was the acting regulations for concrete design in Sweden before it was replaced by Eurocode. Several versions existed for these regulations like BBK 94 and BBK 79. They are not used anymore in the design process in Sweden.

2.3.2 Moment distribution

There is no recommendation for moment distribution in BBK.

2.3.3 Shear force distribution in BBK 04 [Boverket,(2004)]

BBK04 stated that the shear force distribution width can be determined by:

$$b_{ef} = max\begin{cases} 7d + b + t \\ 10d + 1.3x \end{cases} \tag{2.10}$$

Where:

b_{ef} : Distribution width of shear force.
b : Load width.
d : Effective height of the slab.
t : The thickness of filling etc.
x : Distance from the load center to the calculation section, which is considered at the distance $d/2$ outside the limit of the load distribution closest to the support. See Figure 2.7.

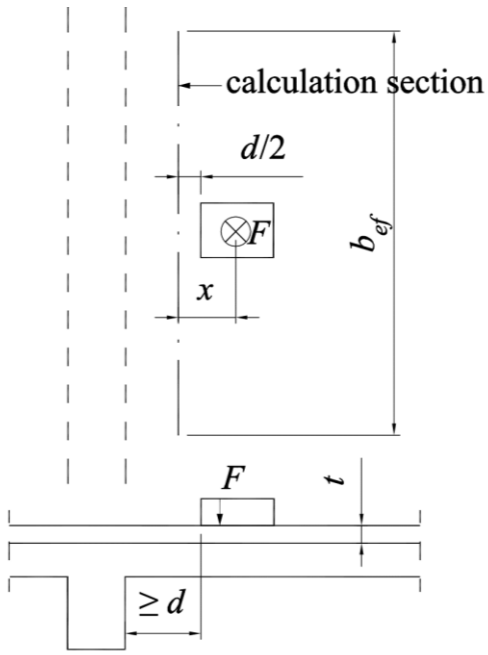


Figure 2.7 Distribution width b_{ef} for a concentrated load on the slab close to the support. Redrawn from [Boverket,(2004)].

For two closely situated concentrated loads, so close to each other that their distribution width b_{ef} overlap with each other as in Figure 2.8, the distributed shear force per unit length v at the selected section can be determined as described in the following.

When the location of the resultant $R(F_1, F_2)$ to F_1 and F_2 is determined. Distribution width for $R(F_1, F_2)$ should be set to $(b_{ef} + 2l_{res})$ where b_{ef} is the distribution width of the larger load and l_{res} is the distance between $R(F_1, F_2)$ and the larger load. The calculated value of v shall not be lower than the value of v that is caused by the largest individual load. If any, distributed shear force in the selected section that is caused by other loads on the slab, ex. self-weight and other concentrated loads (ex. F3 and F4 in Figure 2.8) shall be added to the value of v of $R(F_1, F_2)$.

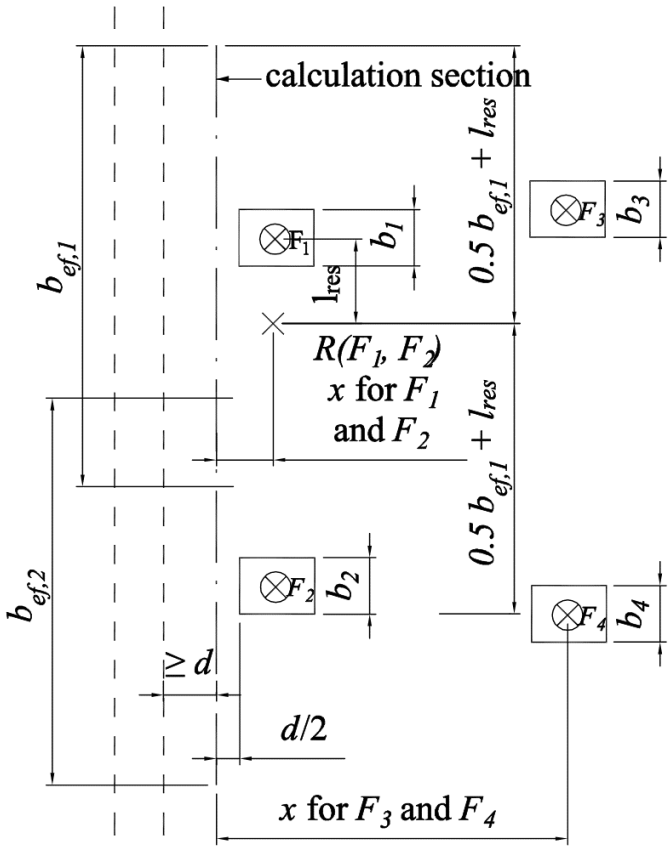


Figure 2.8 Distribution width $(b_{ef.1} + 2l_{res})$ for two loads close to the support in different intensities. $F_1 \geq F_2$. Redrawn from [Boverket,(2004)].

2.3.4 Shear force distribution in BBK 79 [Statens Betongkommitté (1979)]

An old approach for determining the distribution width was:

$$b_{ef} = max\begin{cases} 7d + b \\ 10.65d + 0.65(b + length\ of\ load) \end{cases} \quad (2.11)$$

2.4 Structural use of concrete BS 8110-1:1997 [BSI (1997)]

2.4.1 Introduction

This is the old British code "BS 8110-1:1997 Structural use of concrete. Code of practice for design and construction" was the predecessor code to Eurocode that was used in the UK until it was withdrawn and replaced with Eurocode in 2010.

2.4.2 Moment distribution

According to BS 8110-1:1997, if a slab is simply supported on two opposite edges and carries one or more concentrated loads in a line in the direction of the span, the maximum design bending moment caused by the loading system shall be accounted for. This bending moment may be assumed to be resisted by a specific distribution width of the slab (parallel to the supports) which can be estimated according to the below cases.

For simply supported solid slabs, the distribution of concentrated loads on them was estimated according to the position of the load on the bridge deck. Thus, the distribution width can be calculated as:

$$b_{ef} = 2.4x\left(1 - \frac{x}{l}\right) + Load\ width \qquad (2.12)$$

x : Distance to the nearest support.
l : Span length.

For the unsupported edge, the distribution width shall not exceed the value evaluated in equation (2.12*)* nor half the value plus the distance of the center of the load from the supported edge as in the below Figure 2.9.

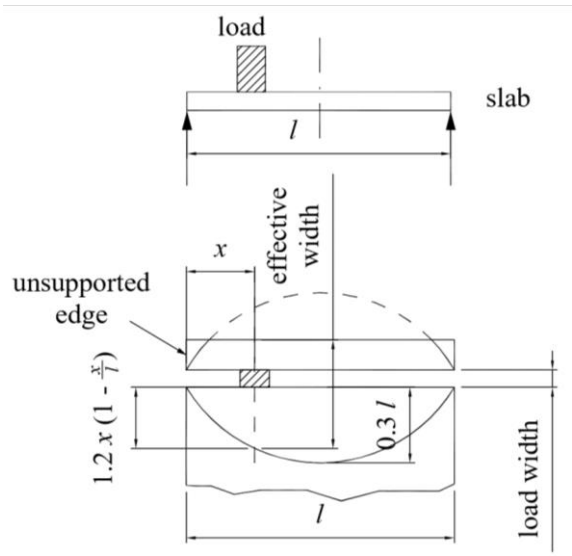


Figure 2.9 Effective width of solid slab carrying a concentrated load near an unsupported edge. Redrawn from [BSI, B.S., 8110-1. (1997)].

2.4.3 Shear force distribution

There is no recommendation for shear force distribution in this guideline.

2.5 *fib* Model Code 2010 [fib (2013)]

2.5.1 Introduction

The *fib* Model Code was initiated by *fib* predecessors CEB (Comité Euro-International du Béton) and FIP (Fédération Internationale de la Précontrainte) at a time when there were hardly any international codes. Since the Model Code 1990, the series intention aimed to serve as an important basis for the successive versions of Eurocode 2. The main aim of this book is to contribute to the development of improved design methods and the application of improved structural materials.

2.5.2 Moment distribution

There is no recommendation for moment distribution in this guideline.

2.5.3 Shear force distribution

In fib Model Code, the distribution width is not discussed in detail, rather just referred to in the slab shear force design section that explains the way to choose a shear force control section. The recommendation is that it shall be chosen with the least distance of either d or $\frac{a_v}{2}$ from the support as in Figure 2.10.

For simply supported one-way slabs

$$b_w = 2(a_v + b - d)\tan(60°) + b \tag{2.13}$$

For loads at a clamped end

$$b_w = 2(a_v + b - d) + b \tag{2.14}$$

Where

b : The load width and length (assuming square load).

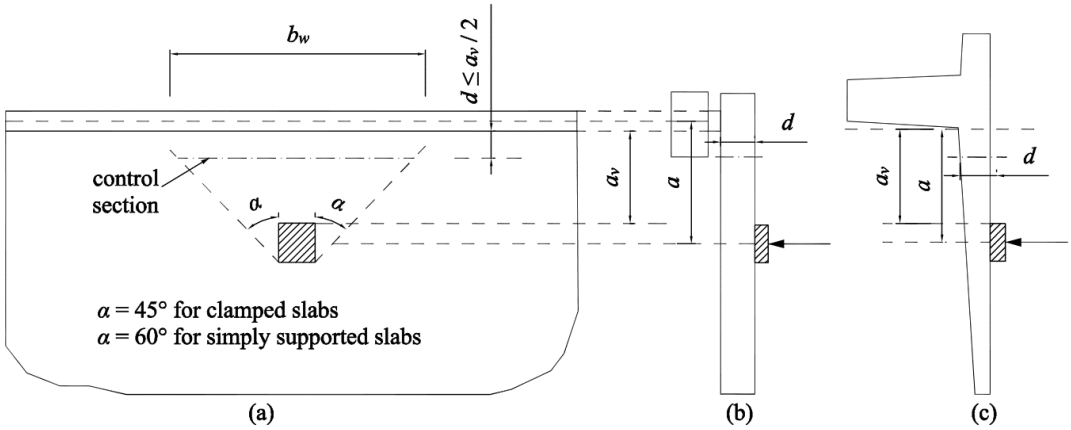


Figure 2.10 (a) Distribution width for shear force indicated for two cases (b) simply supported edge, and (c) clamped slab edge. Redrawn from [fib, (2013)].

2.6 Krav Bärighetsberäkning v6.0 [Trafikverket (2019a)]

2.6.1 Introduction

This document provided by The Swedish Traffic Administration (Trafikverket) 2019 presents requirements for the assessment of existing bridges, support walls, boat or ferry docks, pile decks, and tunnels that come across roads, railways, and pedestrian and cycle paths. This is the acting document version for bridge assessment in Sweden at the time of this study.

2.6.2 Moment distribution

This regulation adopts an approach similar to the approach found in B7 that was mentioned in section 2.2. The results are the following expressions:

$$r = 0.75L + b + f + t \tag{2.15}$$

But highest:

$$r = 2.7 + f + t \tag{2.16}$$

r : Distributed width
L : Center distance of the beams
b : Load width in the longitudinal direction of the slab
f : Filling thickness
t : Concrete cover layer thickness

2.6.3 Shear force distribution

There is no recommendation for shear force distribution in this guideline.

2.7 Råd Bärighetsberäkning v6.0 [Trafikverket (2019b)]

2.7.1 Introduction

This guideline published by The Swedish Traffic Administration (Trafikverket) 2019 provides advice for the assessment of existing bridges.

2.7.2 Moment distribution

There is no recommendation for shear force distribution in this guideline.

2.7.3 Shear force distribution

This document adopts the same approach as in section 2.3.3 in BBK 04 for the distribution width of a single concentrated load. In addition, it gives a simpler approach to deal with the distribution width for three closely situated concentrated loads.

For three equal concentrated loads (whose distribution widths overlap with each other), as in Figure 2.11, the distribution width for each individual load can be determined as b_{ef} by equation (2.10*)* and increased by a distance of 2*c* between the outer loads.

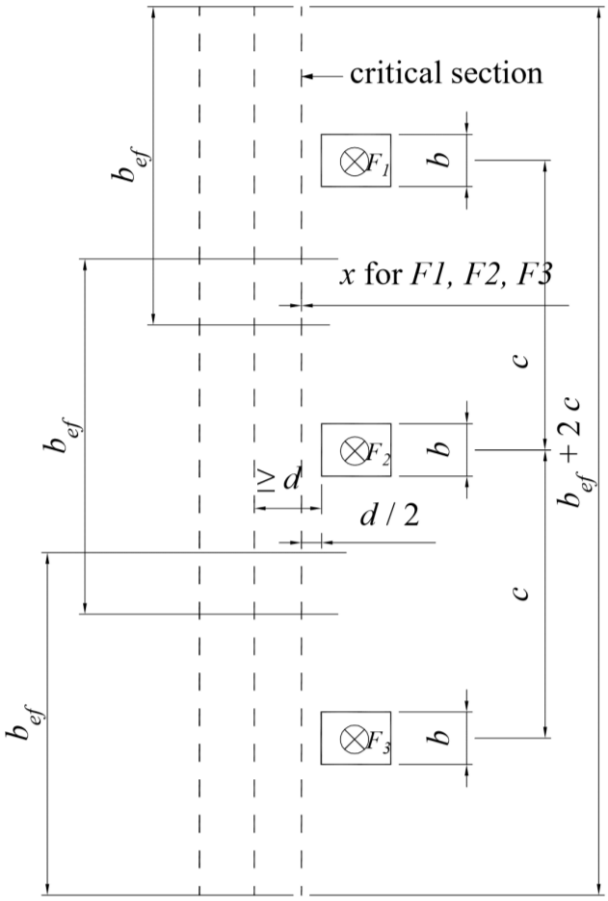


Figure 2.11 Distribution with of three closely situated concentrated loads. Redrawn from Råd Bärighetsberäkning v6.0 [Trafikverket (2019b)]

2.8 Recommendations for finite element analysis for the design of reinforced concrete slabs [Pacoste *et al.* (2012)]

2.8.1 Introduction

This document provides simplifications and guidance on how to implement linear FE analysis and interpret the results in a slab designing process in both the ultimate limit state and service limit state. The recommendations were preliminary intended for bridge and tunnel structures. This document was created to address three aspects in which one of them was the choice of averaging width for moment and shear force.

2.8.2 Moment averaging width

The averaging width of a reinforced concrete slab depends on its ability to redistribute the load, which can be represented by $\frac{x_u}{d}$. The recommendations for selecting the averaging width are:

$$w = min\left(3h, \frac{L_c}{10}\right) \quad for\ \frac{x_u}{d} = 0.45\ (0.35\ for\ C\ classes \geq C55\ /\ 67) \tag{2.17}$$

$$w = min\left(5h, \frac{L_c}{5}\right) \quad for\ \frac{x_u}{d} = 0.30\ (0.23\ for\ C\ \ classes \geq C55\ /\ 67) \tag{2.18}$$

$$w = \frac{L_C}{4} \quad for\ \frac{x_u}{d} = 0.25\ (0.15\ for\ C\ classes \geq C55\ /\ 67) \tag{2.19}$$

$$w = \frac{L_C}{2} \quad for\ \frac{x_u}{d} = 0.15\ (0.10\ for\ C\ classes \geq C55\ /\ 67) \tag{2.20}$$

$$w = min\left(5h, \frac{L_c}{5}\right) \quad for\ \frac{x_u}{d} = 0.00 \tag{2.21}$$

In most of the bridge deck designs, equations (2.20*)* and (2.21*)* are the governing equations and w value commonly is not less than $1m$.

Since the value w depends on the redistribution capability of the slab, which in turn is affected by the amount of reinforcement, the design process is iterative. Generally, the averaging width should not be less than w_{min} :

$$w \geq w_{min} = 2h + a \tag{2.22}$$

h : Height of the section.
x_u : Depth of the neutral axis in ULS after redistribution.
L_c : Characteristic span width.
a : The dimension of the support in the considered direction.
d : The effective depth of the section.

2.8.3 Shear force averaging width

The recommendation for the shear force averaging width is based on the assumption that a reinforced concrete slab has good capabilities for plastic redistribution in the ultimate limit state. Thus, the same rules for choosing moment averaging width in section 2.8.2 is deemed taking into consideration the following limitations for w value:

$$w < 5h \tag{2.23}$$

And

$$|\alpha_i - \alpha_0| \leq 45^{\circ}$$

The averaging width shall be restricted to a variation of the angle α of less than 45° which is represented as w_{eff} as in Figure 2.12 below, where α is the principal shear force v_0 direction angle with w_{eff}.

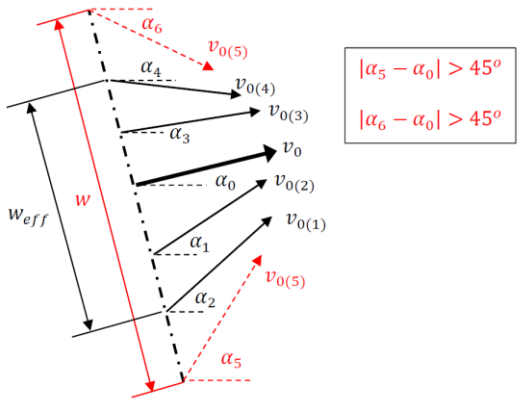


Figure 2.12 Variation of the direction of the resultant shear force within a distribution. From [Pacoste, .et al. (2012)].

2.8.4 The relation between averaging width and compression zone height

The averaging width w is dependent on the value of $\frac{x_u}{d}$ in a similar pattern to the relation between the rotation capacity of cross-section on $\frac{x_u}{d}$ as can be seen in Figure 2.13.

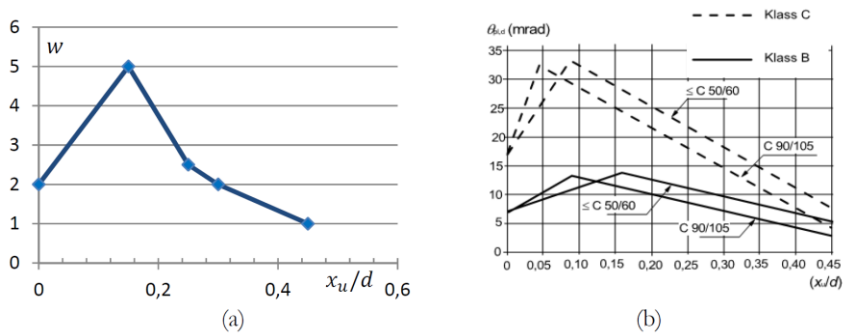


Figure 2.13 (a) Relation between the averaging width w and $\frac{x_u}{d}$, and (b) Relation between rotation capacity and $\frac{x_u}{d}$. From [Pacoste, .et al. (2012)].

2.8.5 Moment averaging width for cantilever slabs

For single load F as in Figure *2.14*, the averaging width w_x for ultimate limit states given by :

$$w_x = min\begin{cases} 7d + b + t \\ 10d + 1.3y_{cs} \end{cases} \quad (2.24)$$

$$For \;\; 0.25 \geq \frac{x_u}{d} \geq 0.15$$

For $\frac{x_u}{d}$ values outside the above limit the distribution will be taken as:

$$w_x = 2h + b + t \quad (2.25)$$

x_u	:	The depth of the neutral axis at the ultimate limit state after redistribution. X_u should be evaluated for the section with the highest reinforcement ratio.
d	:	The effective depth at the critical cross-section.
h	:	The height of the cantilever at the critical cross-section.
b	:	The width of the load (see Figure *2.14*).
t	:	The thickness of the surface (see Figure *2.14*).
y_{cs}	:	The distance from the center of the load to the critical cross-section.

For multiple closely positioned concentrated loads with overlapping distribution widths, the resultant distribution width w_{xR} for the whole group of forces can be determined according to the rule shown in Figure *2.14*.

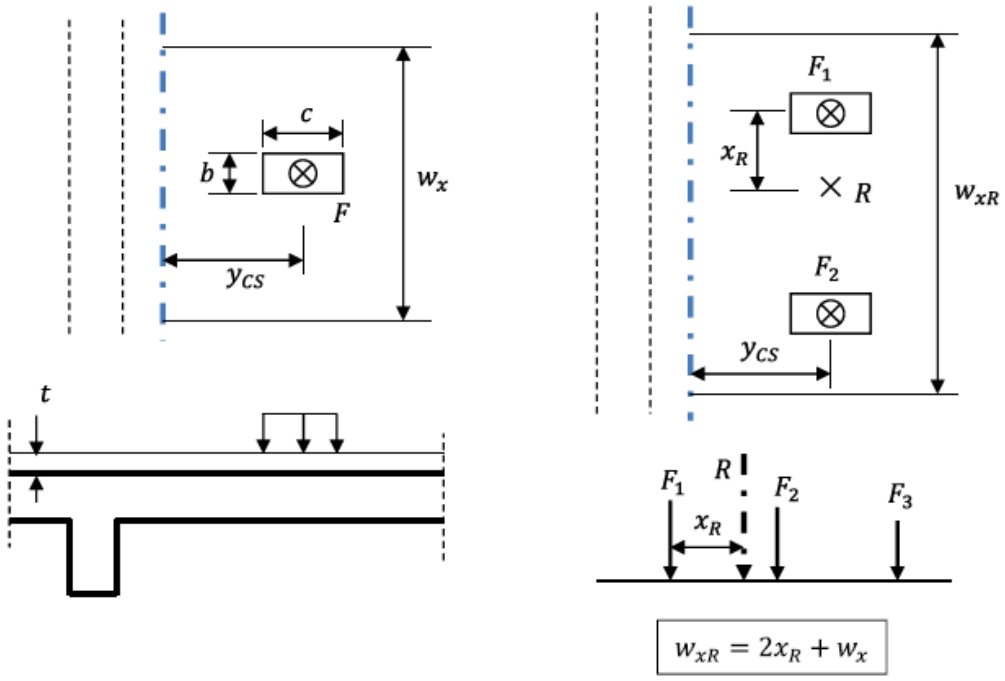


Figure 2.14 Averaging width for the reinforcement moment in a cantilever slab, in case of one concentrated load (left), and two or several concentrated forces (right). From [Pacoste et al. (2012)].

2.8.6 Shear force averaging width in cantilever slab

For a single force or a group of forces situated in a row, the critical cross-section for shear forces is always placed at a distance $y_{cs} = \frac{c+d}{2}$ from the concentrated load, where c and d are defined in Figure 2.15. and Figure 2.16.

For the case of two rows of concentrated loads, the critical cross-section is chosen according to the principle illustrated in Figure 2.16.

Depending on the position, y, of the critical cross-section on the cantilever, the averaging width $w_{x(v_0)}$ can be determined through linear interpolation between the max and min values shown below:

$$w_{max} = max\begin{cases} 7d + b + t \\ 10d + 1.3y_{cs} \end{cases} \qquad for\ \ y = 0 \tag{2.26}$$

$$w_{min} = min\begin{cases} 7d + b + t \\ 10d + 1.3y_{cs} \end{cases} \qquad for\ \ y = y_{max} \tag{2.27}$$

Where $y_{\max}$ is defined according to Figure 2.15.

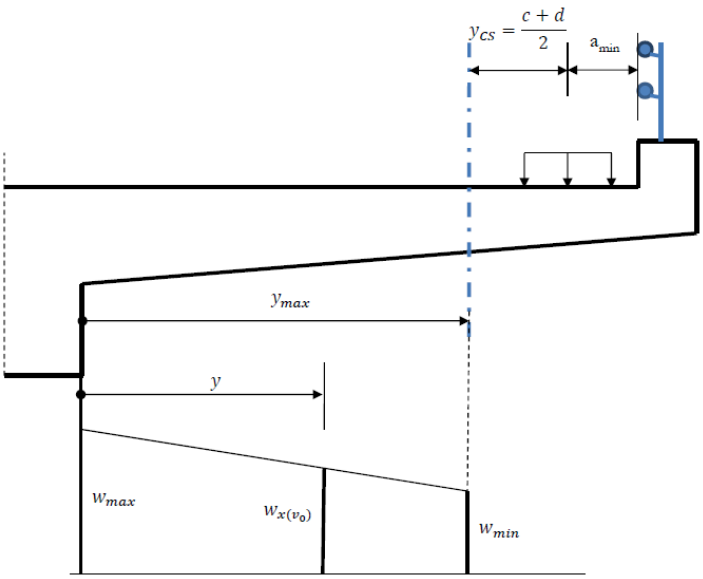


Figure 2.15 Averaging widths for shear forces in a cantilever slab, in case of a single load. From [Pacoste, .et al. (2012)].

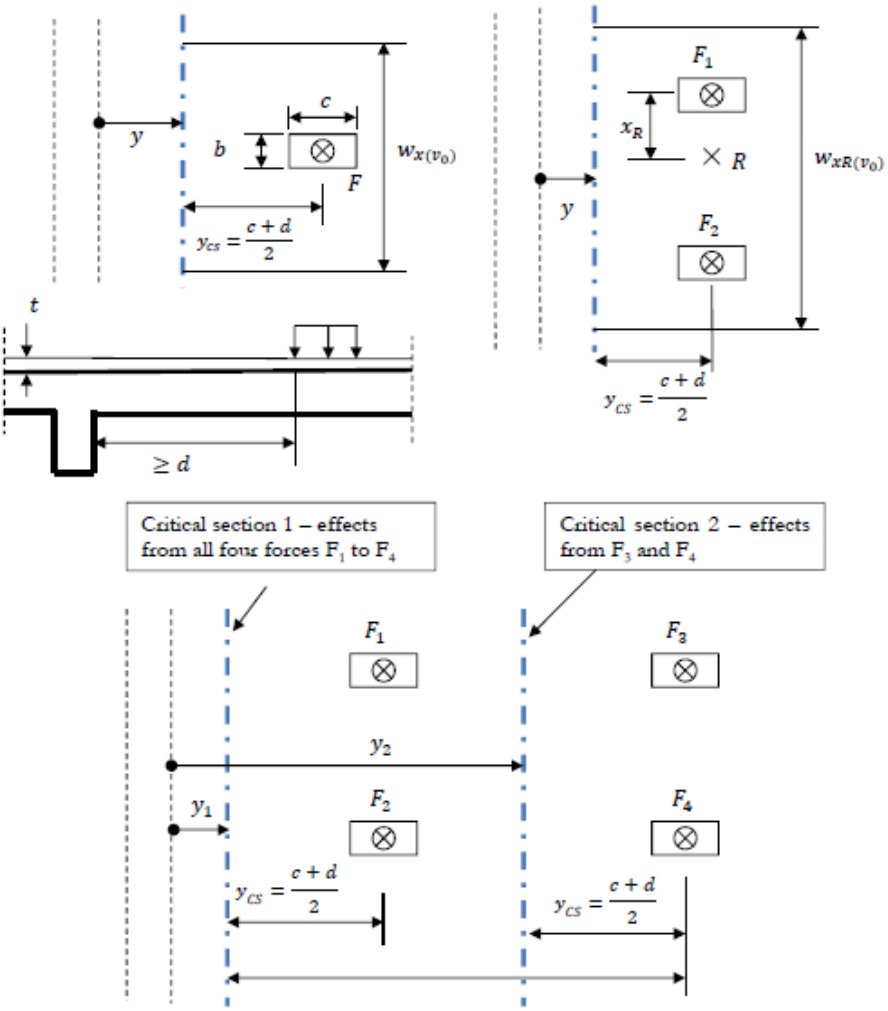


Figure 2.16 Critical sections for shear forces for cantilever slabs, for a single force (upper left), several forces in a row (upper right), and a group of four forces (lower center). From [Pacoste, .et al. (2012)].

2.9 Other guidelines

2.9.1 Introduction

Multiple approaches are presented in this section. These approaches were found in a research paper by [Lantsoght *et al.* (2015)]. The authors conducted an experimental program to determine the shear capacity of slabs under concentrated load close to the support. The study was conducted after several existing reinforced concrete bridges, designed according to the Dutch national codes, were found to have limited shear capacity when assessed. is the problem being due to the development of transportation and vehicles which resulted in shorter axle distances with higher loads.

2.9.2 Moment distribution

No recommendation for moment distribution is provided in this section.

2.9.3 Shear force distribution

The published paper presented multiple approaches to calculate the distribution width for shear force. All symbols definitions for equations are mentioned at the end of this section and illustrated in Figure 2.18.

The "Dutch approach" [Lantsoght *et al.* (2015)], where horizontal load spreading is assumed under a 45° angle from the center of the load towards the support as in Figure 2.17(a)

$$b_{eff} = 2x_{crit} \tag{2.28}$$

The "French approach" [Chauvel, et al. (2007)], where the load is also spread with an angle of 45° but from the far corners of the loaded area as in Figure 2.17(b)

$$b_{eff} = 2\left(x_{crit} + \frac{b_{load}}{2}\right) + b_{load} \tag{2.29}$$

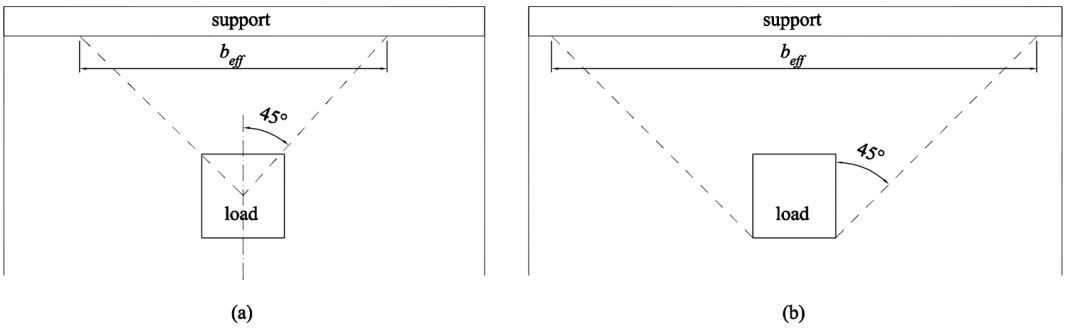


Figure 2.17 Distribution width (a) Dutch Practice. (b) French Practice. Redrawn from [Lantsoght et al. *(2015)].*

Lubell, Bentz, and Collins [Lubell *et al.* (2008)] specified a reduction factor β_L on the slab width b

$$\beta_L = 0.7 + 0.3\kappa \tag{2.30}$$

$$\kappa = min\ (\frac{b_{load}}{b}; \frac{b_{sup}}{b})$$

$$b_{eff} = \beta_L b \qquad (2.31)$$

Grasser and Thielen [Grasser *et al.* (1991)] estimated the distribution width which was used by the German practice for simply supported one-way slabs:

$$b_{eff} = t_y + 0.5x \leq b \qquad (2.32)$$

$$t_y = b_{load} + d_l \qquad and \qquad t_x = l_{load} + d$$

The above equation is only valid with the below conditions:

$$0 < x < l\ , \qquad t_y \leq 0.8l \qquad and \qquad t_x \leq l$$

For loads at a clamped end:

$$b_{eff} = t_y + 0.3x \qquad (2.33)$$

$$0.2 < x < l\ , \qquad t_y \leq 0.4l \qquad and \qquad t_x \leq 0.2l$$

Zheng [Zheng *et al.* (2010)] estimated the distribution width as follows:

$$b_{eff} = l_{load} + l(1 - r_{cp})tan\Phi \qquad (2.34)$$

$$r_{cp} = \frac{b_{load}}{l} \leq 0.4$$

$$\Phi = 23.3r_{cp} + 35.1 \qquad where \quad \Phi \ \ in\ degrees$$

Where:

x	:	Center to center distance between the load and the support.
x_{crit}	:	Center to center distance between the load and the critical section.
d_l	:	The effective depth to the longitudinal reinforcement.
t_x, t_y	:	Size of the wheel print which is distributed to mid-depth on the concrete slab.
l	:	Span length.
b	:	Slab width.
b_{load}	:	Width of the load.
l_{load}	:	Length of the load.
b_{sup}	:	Width of support.

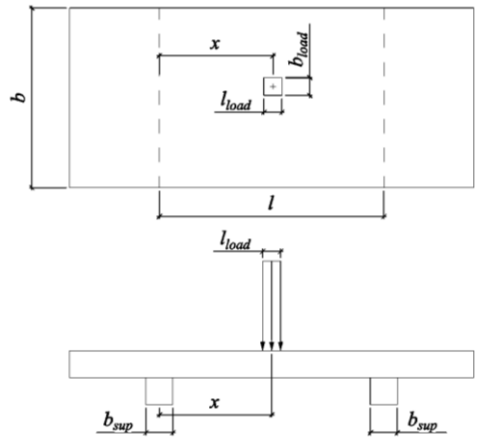


Figure 2.18 Illustrative figure to clear the parameters related to the mentioned expressions.

2.10 Summary of existing guidelines

The previously stated guidelines were summarized and compared in this section. To facilitate the comparison between mathematical expressions and parameters that governs the distribution in each expression, all the variable names are unified and converted to the listed variables in the notation chapter. The guidelines are categorized according to distribution parameters (moment and shear force) and the location of the load whether it is located in the span or in the cantilever. All the expressions are listed below in Table 2.1 to Table 2.4.

Table 2.1 Guidelines that treated moment distribution in cantilever slab.

Distribution of moment at the cantilever slab		
B7 [Statens Betongkommitté (1968)]		assuming no edge beam, and without end supports: $b_{eff} = 0.54(x^3 l_k - x^4)^{0.25}$

Table 2.2 Guidelines that treated moment distribution in the span.

Distribution of the moment at the span		
B7 [Statens Betongkommitté (1968)]	(2.1) (2.2)	Perpendicular to reinforcement: $b_{eff} = 0.75\, l_m + b_{load} + t_f + t_c$ $b_{eff} \leq 2.5 + b_{load} + t_f + t_c$
Krav Bärighetsberäkning. [Trafikverket (2019a)]	(2.15) (2.16)	$b_{eff} = 0.75 l_m + b_{load} + t_f + t_c$ $b_{eff} \leq 2.7 + t_f + t_c$
BS 8110-1:1997 [BSI (1997)]	(2.12)	$b_{eff} = 2.4x\left(1 - \frac{x}{l_m}\right) + b_{load}$

Table 2.3 Guidelines that treated shear force distribution in cantilever slab.

Distribution of shear force at the cantilever		
fib Model Code 2010 [fib (2013)]	(2.14)	$b_{eff} = 2\left(x_{crit} + \frac{b_{load}}{2}\right) + b_{load}$
Grasser and Thielen [Grasser *et al.* (1991)]		$b_{eff} = l_{load} + d + 0.3x$ $0.2 < x < l_m$, $l_{load} + d \leq 0.4 l_m$ *and* $b_{load} + d \leq 0.2 l_m$

Table 2.4 Guidelines that treated shear force distribution in the span.

Distribution of shear force at the span		
BBK 79 [Statens Betongkommitté (1979)]	(2.11)	$b_{eff} = max\begin{cases} 7d + b_{load} \\ 10.65d + 0.65(b_{load} + l_{load}) \end{cases}$
BBK 04 [Boverket (2004)]	(2.10)	$b_{eff} = max\begin{cases} 7d + b_{load} + t_f \\ 10d + 1.3x_{crit} \end{cases}$
B7 [Statens Betongkommitté (1968)]	(2.8) (2.9)	$b_{eff} = \frac{3}{4}x + max\begin{cases} b_{load} + 2(t_f + t_c + d) \\ 5d \end{cases}$
Dutch approach [Lantsoght et al. (2015)]		$b_{eff} = 2x_{crit}$
French approach [Chauvel, et al. (2007)]		$b_{eff} = 2\left(x_{crit} + \frac{b_{load}}{2}\right) + b_{load}$
Lubell, Bentz and Collins [Lubell *et al.* (2008)]		$b_{eff} = min\begin{cases} 0.7\ l_m + 0.3b_{load} \\ 0.7\ l_m + 0.3b_{sup} \end{cases}$
Grasser and Thielen [Grasser *et al.* (1991)]		$b_{eff} = b_{load} + d\ + 0.5x \leq l_s$ $0 < x < l_m\ ,\quad b_{load} + d\ \leq 0.8l_m$ $and \quad l_{load} + d \leq l_m$
ACI Structural Journal. [Zheng *et al.* (2010)]		$b_{eff} = l_{load} + (l_m - b_{load})tan(23.3\frac{b_{load}}{l_m} + 35.1)$ $and \quad \frac{b_{load}}{l_m} \leq 0.4$
fib Model Code 2010 [fib (2013)]	(2.13)	$b_{eff} = 2\left(x_{crit} + \frac{b_{load}}{2}\right)\tan(60°) + b_{load}$

To compare the results of each expression, the initial conditions and parameters used in this study were applied to each expression and the results are shown in Figure 2.19 to Figure *2.22* below.

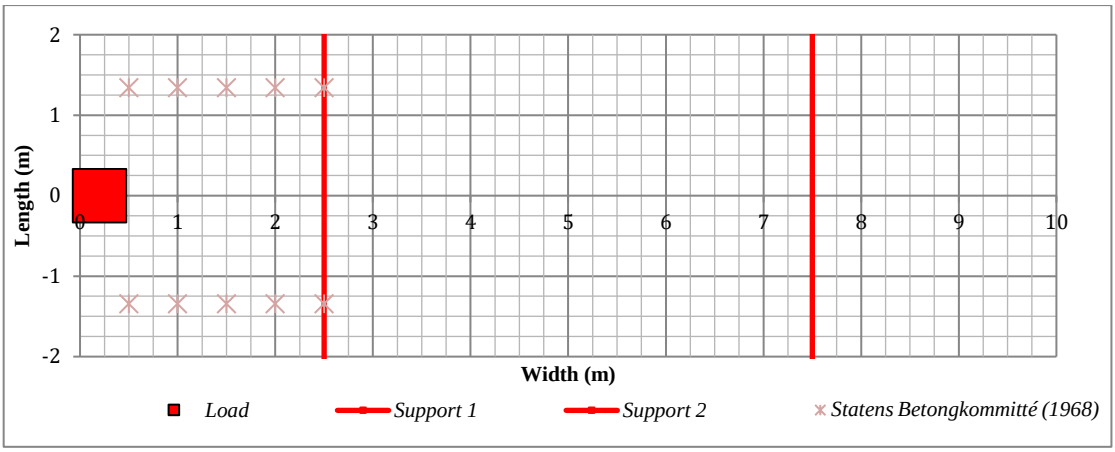


Figure 2.19 Moment distribution width at the cantilever.

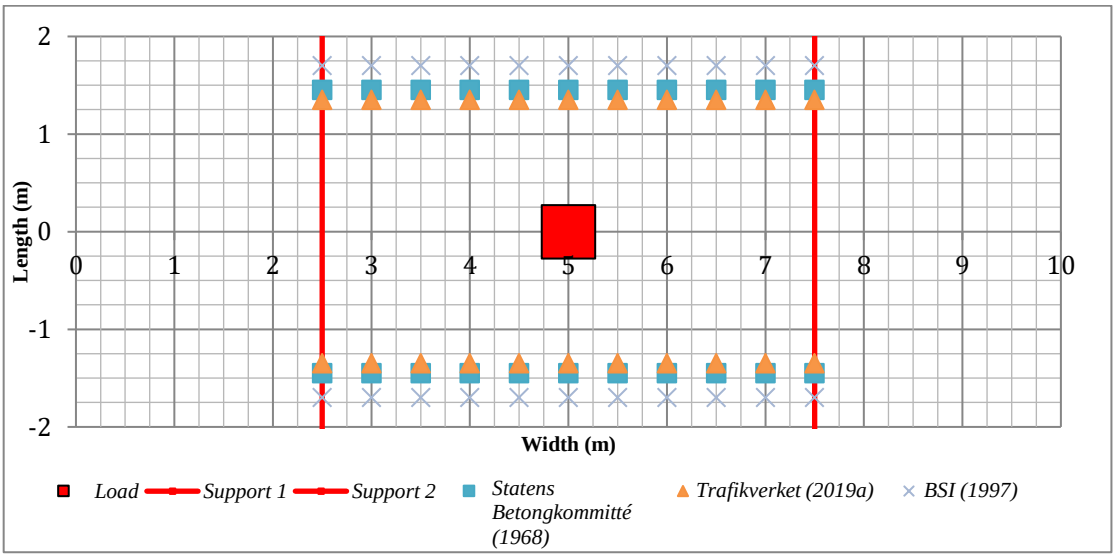


Figure 2.20 Moment distribution width at the span.

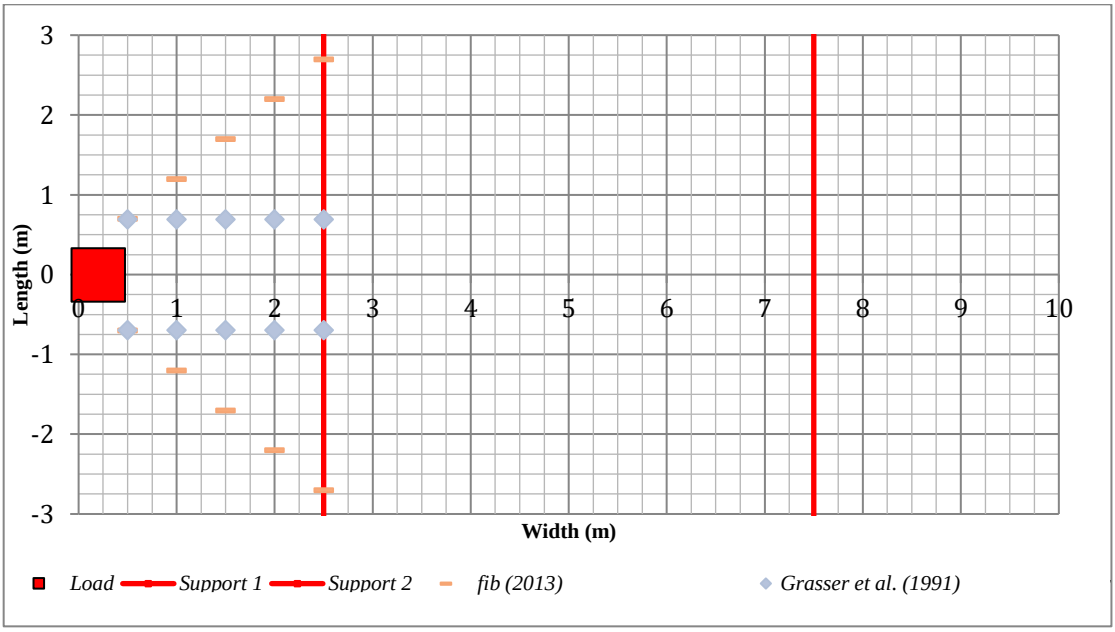


Figure 2.21 Shear force distribution width at the cantilever.

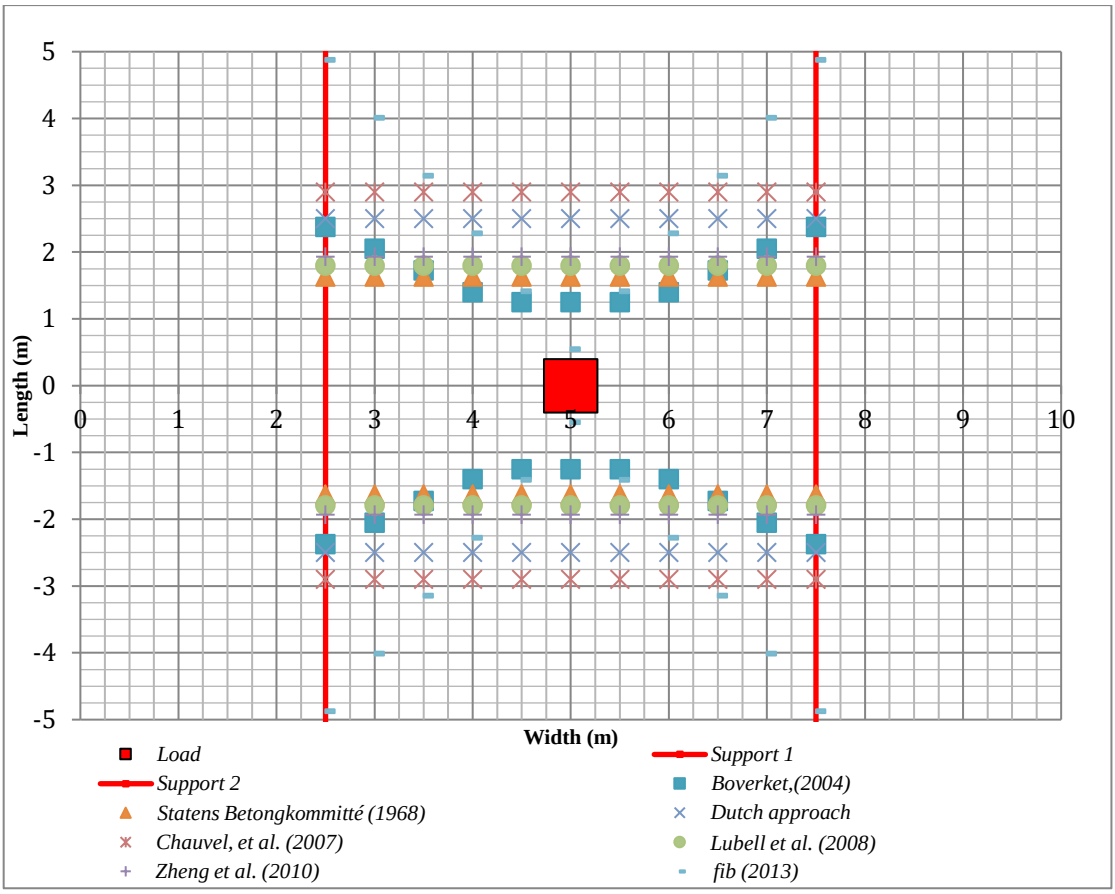


Figure 2.22 Shear force distribution width at the span.

3 Base model analysis

3.1 Introduction

This chapter presents a full study on one selected geometry. It was considered as the base model, to which the approach used to determine the equivalent width was introduced. This chapter is divided into five sections.

The first two sections describe the two models: the FE and the analytical calculation model. The basic assumptions and parameters such as geometry, boundary conditions, loading conditions, etc. were presented together with corresponding results.

In the third section, a result comparison between FE analysis and analytical calculation is made, where the differences are highlighted. Accordingly, the possibility of enhancing the analytical calculation model is discussed, and several modified models for the analytical calculations are introduced and compared. Based on this, the most accurate alternative was chosen.

In the fourth section, by using the chosen analytical calculation model, the results of the equivalent width calculation process are presented and discussed.

At the end of this chapter, the most important issues that are addressed in the full parametric study (chapter 4) are discussed. These issues were identified with respect to how they can influence the results. Strategies to capture them in the full parametric study were suggested.

3.2 FE model description

This section presents the FE model characteristics. A predefined geometry was selected, as shown in Figure 3.1.

3.2.1 Software

The analysis was made with Strusoft FEM Design [Strusoft (2020)].

3.2.2 Geometry

The model resembled a section of a bridge deck. The length of the bridge was defined to be 30 m in order to achieve the effect of an infinitely long slab, as verified in Appendix A. The main span length l_m and cantilever length l_k were selected in the ratio of $\alpha = 0.5$, i.e. the cantilever length is equal to the main span length since $\alpha = \frac{l_k}{l_m}$. The slab thickness was chosen to be 0.3 m constant through the slab, see Figure 3.1.

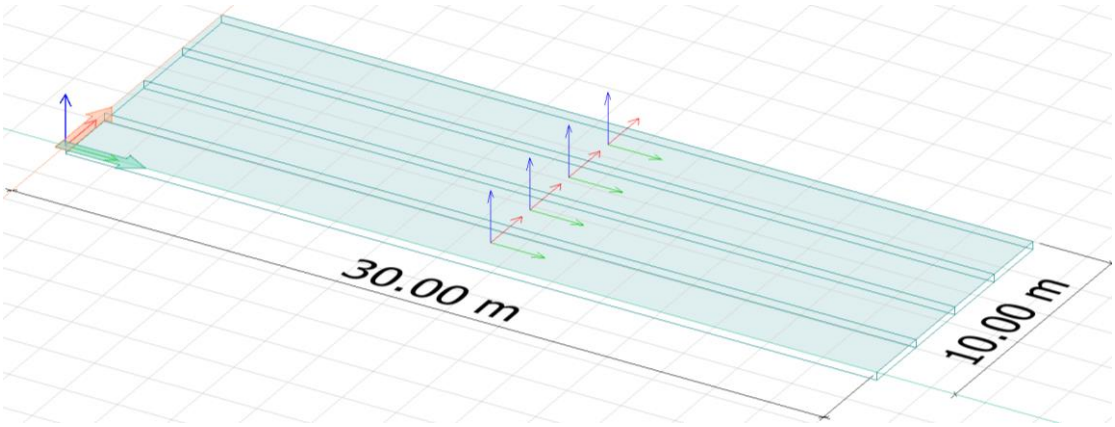


Figure 3.1 Geometry of the bridge deck slab used in the base model analysis.

3.2.3 Boundary conditions

The deck was assumed to be simply supported on two line supports in the longitudinal direction, as illustrated in Figure 3.2. Poisson's ratio was set to zero.

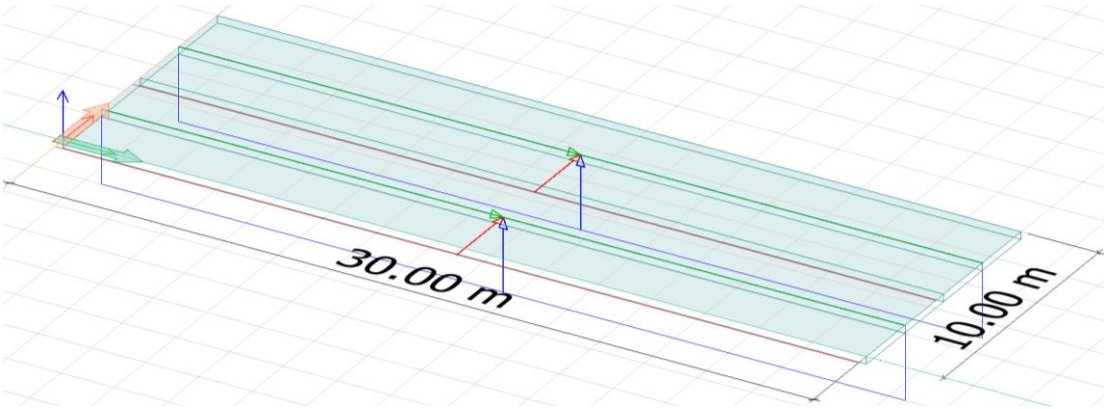


Figure 3.2 Boundary conditions for the base model. The blue rectangles represent the two line supports with constrained vertical displacement.

3.2.4 Load

One concentrated load was applied on the bridge deck for each analysis run. The concentrated load was distributed on an area resembling a vehicle wheel, where the load print dimensions were taken according to Eurocode to be (0.4 x 0.4 m), as shown in Figure 3.3. For each analysis run, the load location was shifted in the width direction, across the slab. Applied load intensity was chosen to be $P_{FE} = 6.25\ \mathrm{kN/m^2}$ which resulted in a total load of 1 kN.

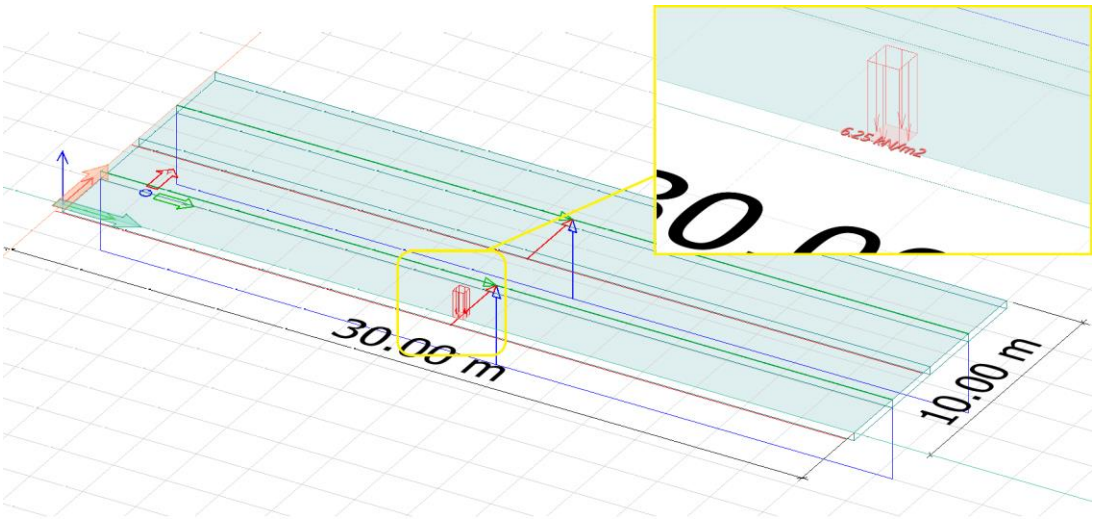


Figure 3.3 Distributed load on the edge of the slab for the base model. The highlighted red arrows represent the load of one wheel.

3.2.5 Element and mesh

The mesh size used for this analysis was 0.25 m, as shown in Figure 3.4, which was found to give accurate enough results for our analysis requirement, as indicated in Appendix A, showing mesh convergence results. 9 node quadratic shell elements were used to represent the slab with isotropic properties.

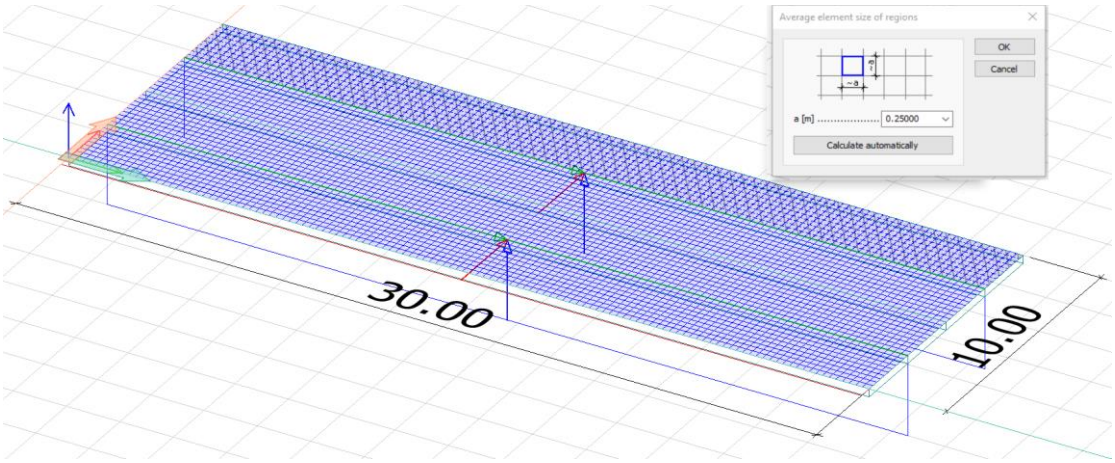


Figure 3.4 FE mesh with 0.25 m x 0.25 m 9 node quadratic shell elements, used for the base model.

3.2.6 Analysis

Linear elastic analysis for static live load case was conducted for the isotropic shell model. The results presented hereafter were based on two load locations: the load at the middle of the span, and the load at the end of the cantilever.

3.2.7 Analysis results

The results of interest in this study were the internal moment M_y and torsional moment T_{xy} which are required to evaluate the reinforcement moment, and the shear force V_{yz} all in the traverse direction. This section presents a summary of the FE analysis results for the two load cases according to section 3.2.6.

3.2.7.1 Concentrated load at the end of the cantilever

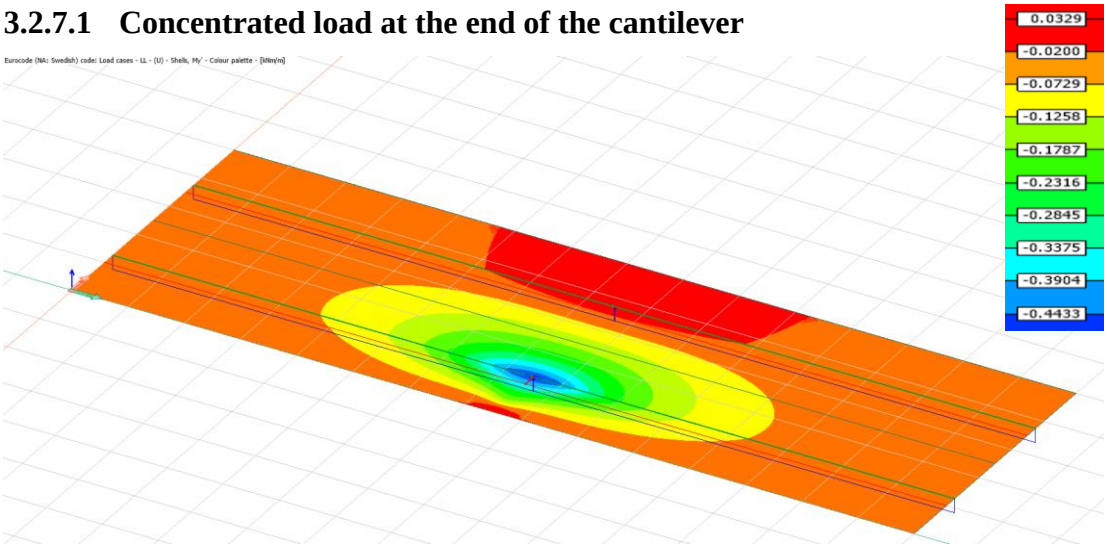


Figure 3.5 The moment in Y direction (My) for a concentrated load at the end of the cantilever, for the base model.

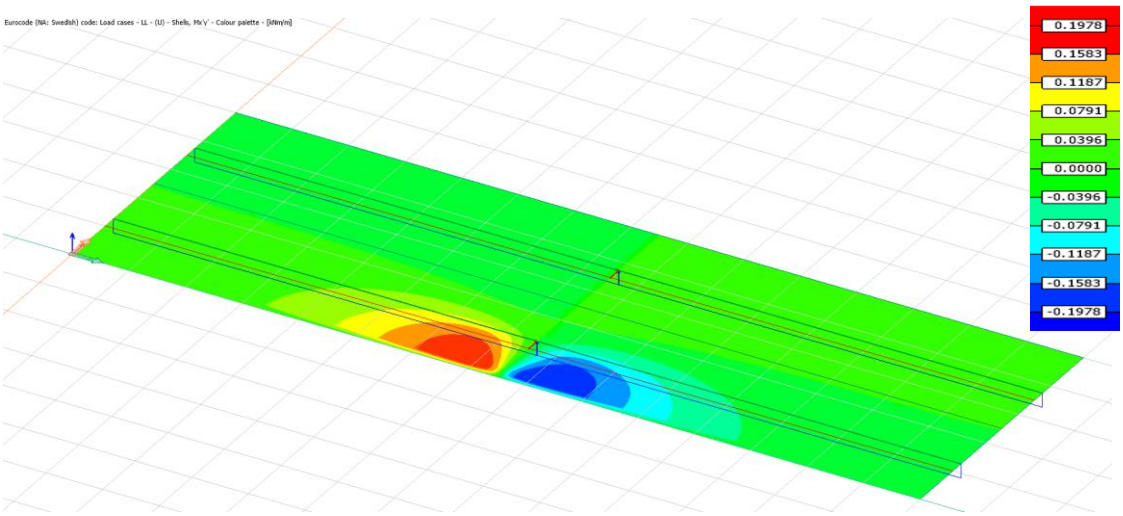


Figure 3.6 The torsional moment (Txy) for a concentrated load at the end of the cantilever, for the base model.

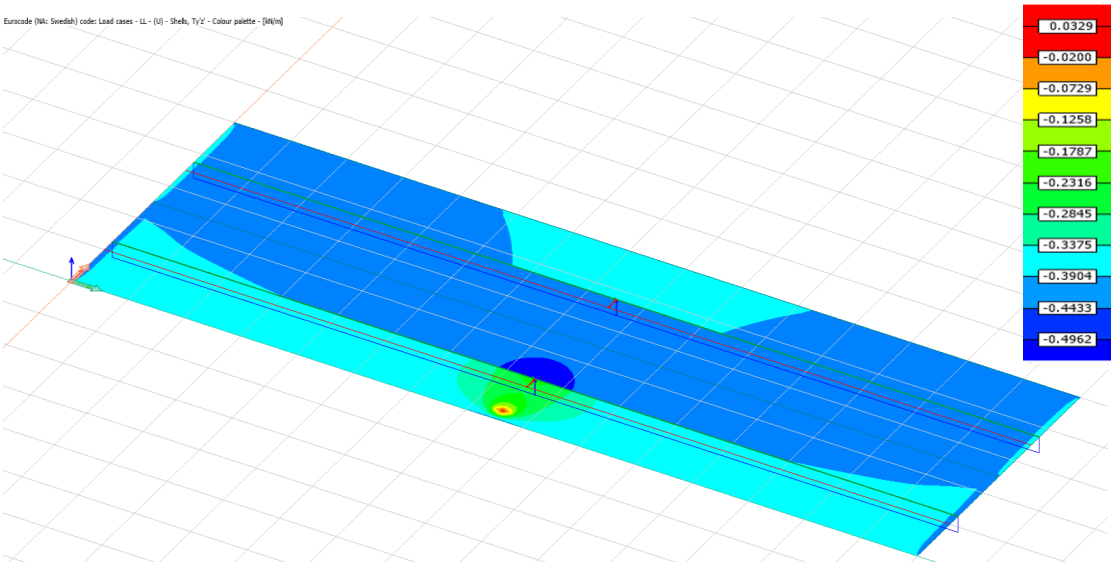


Figure 3.7 The shear force in Y (Vyz) for a concentrated load at the end of the cantilever, for the base model.

3.2.7.2 Concentrated load in the middle of the mid span

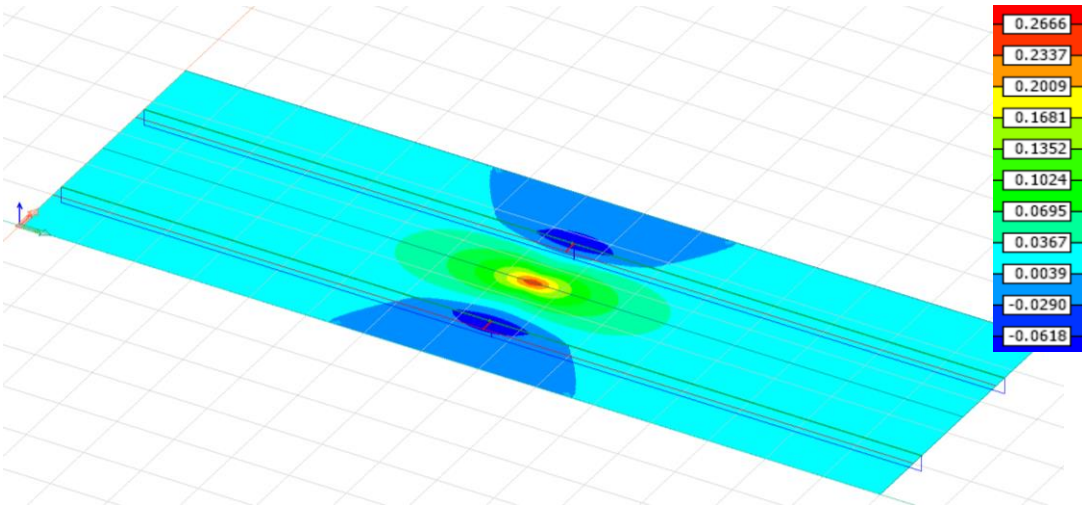


Figure 3.8 The moment in Y direction (My) for a concentrated load at the middle of the main span, for the base model.

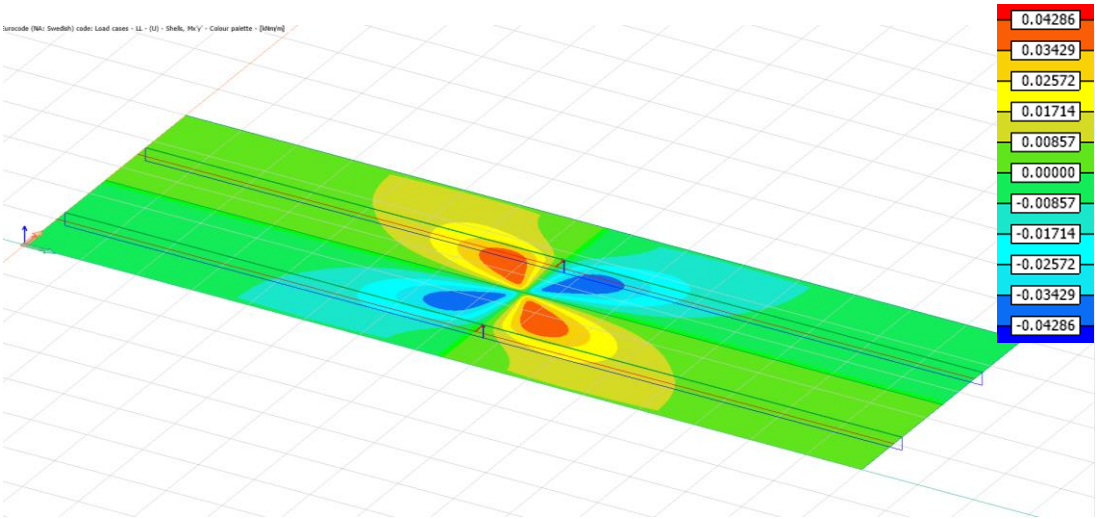


Figure 3.9 The torsional moment (Txy) for a concentrated load at the middle of the main span, for the base model.

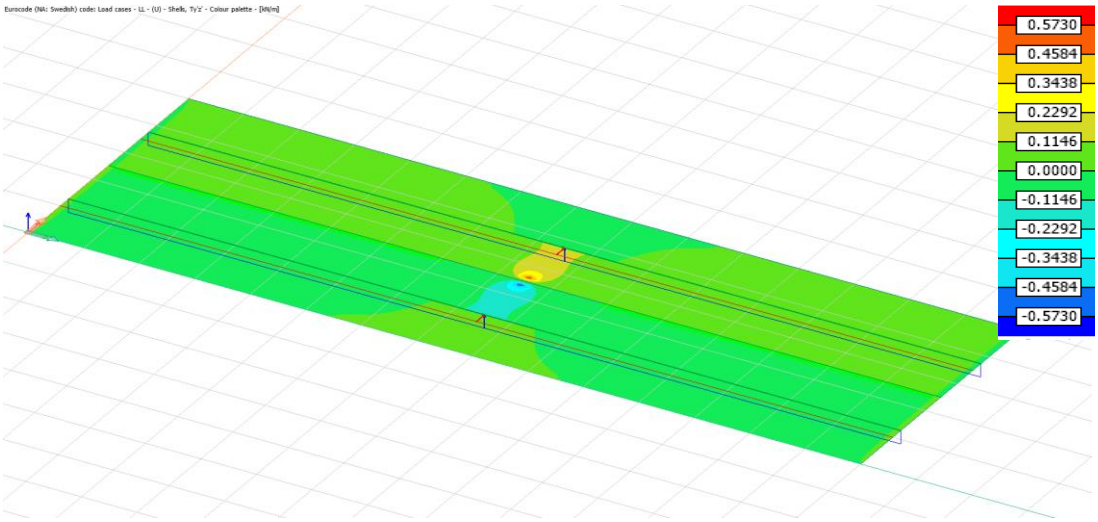


Figure 3.10 The shear force in Y (Vyz) for a concentrated load at the middle of the main span, for the base model.

3.2.8 Processed analysis results

This subchapter presents diagrams with extracted FE analysis sectional results. The results are extracted for different longitudinal sections through the slab automatically using a python script. These results are for shell internal shear force and moment. Figure 3.11 below, shows the locations of the selected sections in the longitudinal direction of the bridge for which the results were extracted.

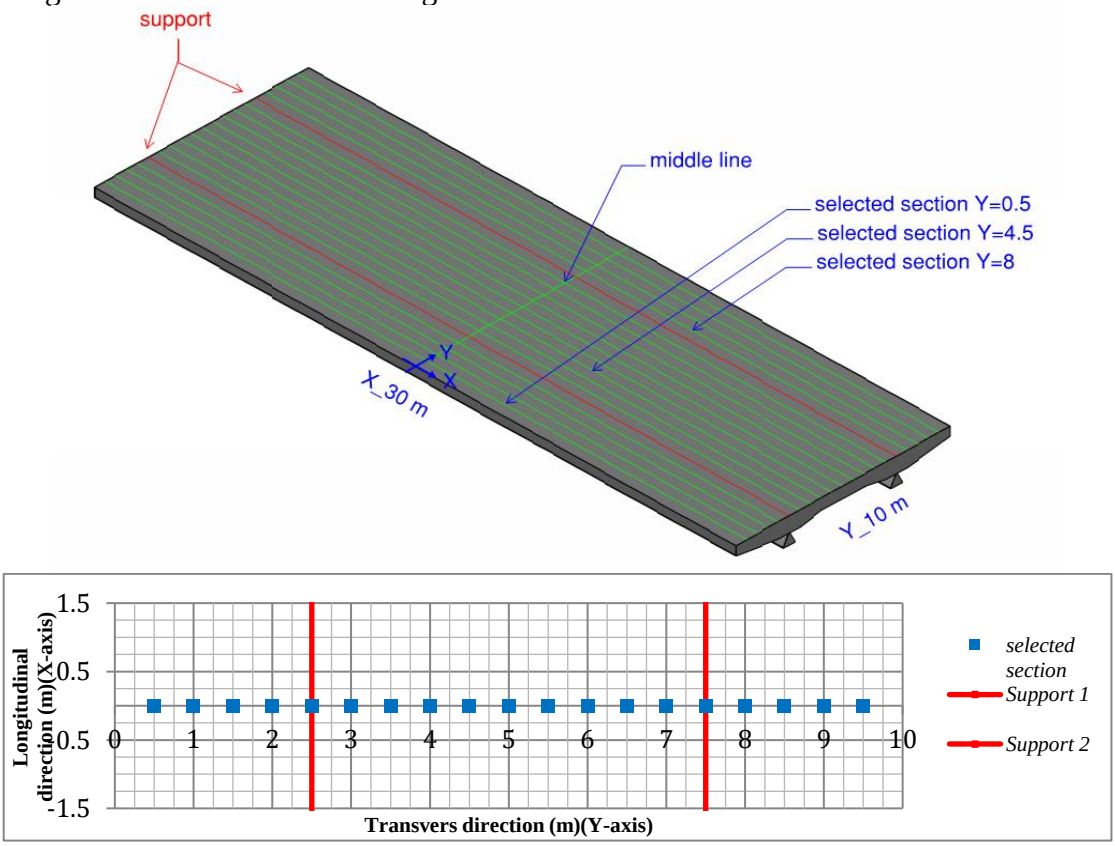


Figure 3.11 3D and plan view of the slab model with green lines (blue squares in the plan view) are the selected sections

In Figure 3.12 and Figure 3.13. results are presented for the load case when the load is applied at the middle of the span. Other results can be found in appendix B.

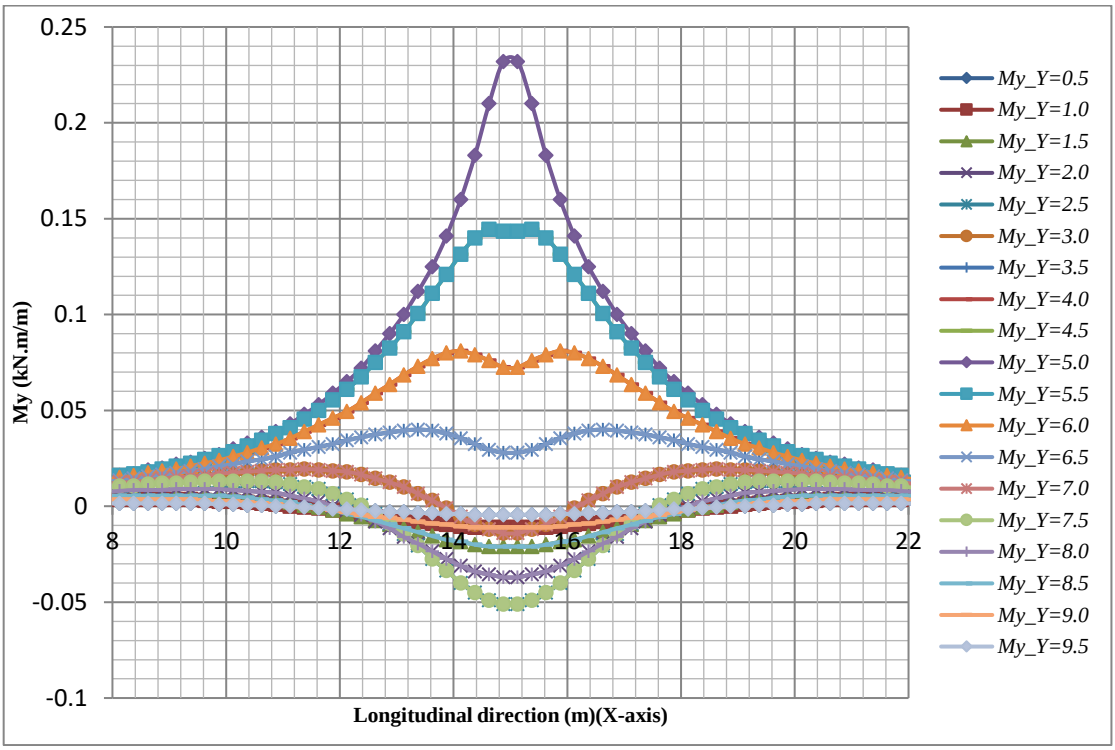


Figure 3.12 The moment (M_y) in the selected longitudinal sections parallel to X-axis, for a concentrated load applied at the middle of the span. The legend indicates M_y curves for sections located at distance Y from the origin.

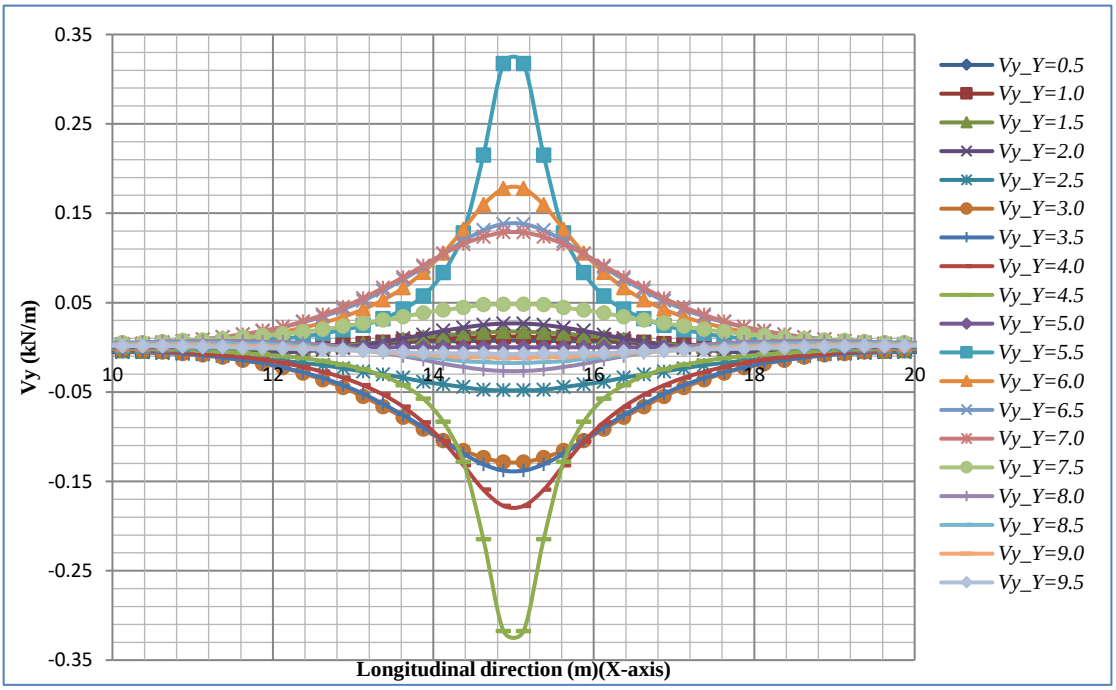


Figure 3.13 The shear force (V_y) in the selected longitudinal sections parallel to X-axis, for a concentrated load applied at the middle of the span. The legend indicates V_y curves for sections located at distance Y from the origin.

Based on the sectional shear force and moment distributions shown above, the average moment M_{av} and the average shear force V_{av} were evaluated according to equations (3.1) and (3.2), respectively.

$$M_{av} = \frac{\int_{-\frac{w}{2}}^{\frac{w}{2}} M_{FE}\, dy}{w} \tag{3.1}$$

$$V_{av} = \frac{\int_{-\frac{w}{2}}^{\frac{w}{2}} V_{FE}\, dy}{w} \tag{3.2}$$

The values for the averaging width w were preselected to represent the common range of values found in the design recommendations for bridge decks, taking into consideration the upper and lower margins within the range of the equations stated in section 2.8.2. The resulting average moments and shear forces, for different preselected averaging widths, w, were illustrated in Figure 3.14 and Figure 3.15 respectively.

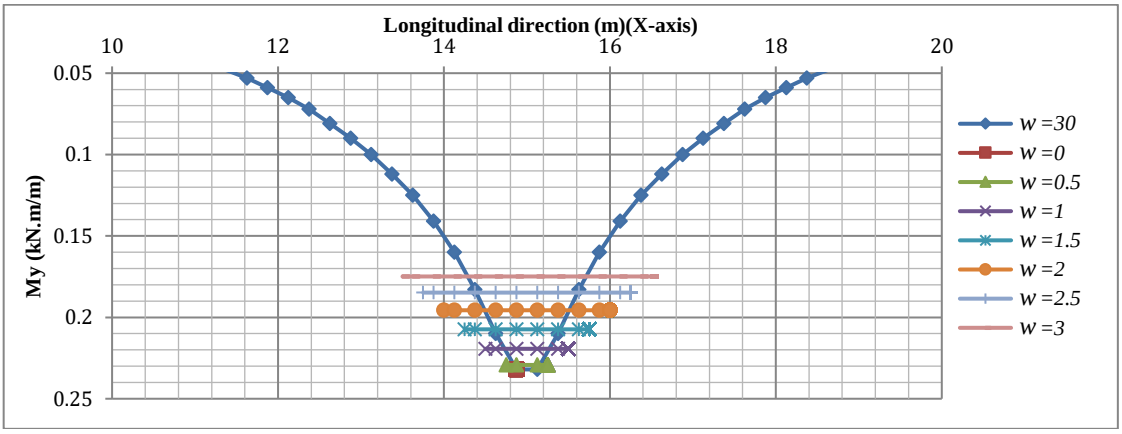


Figure 3.14 Moment variation and averaged moment values for section Y = 5 m *(Midspan). The legend marks refer to the original non-averaged moment (w = 30) and to the average moment corresponding to a chosen averaging width, w.*

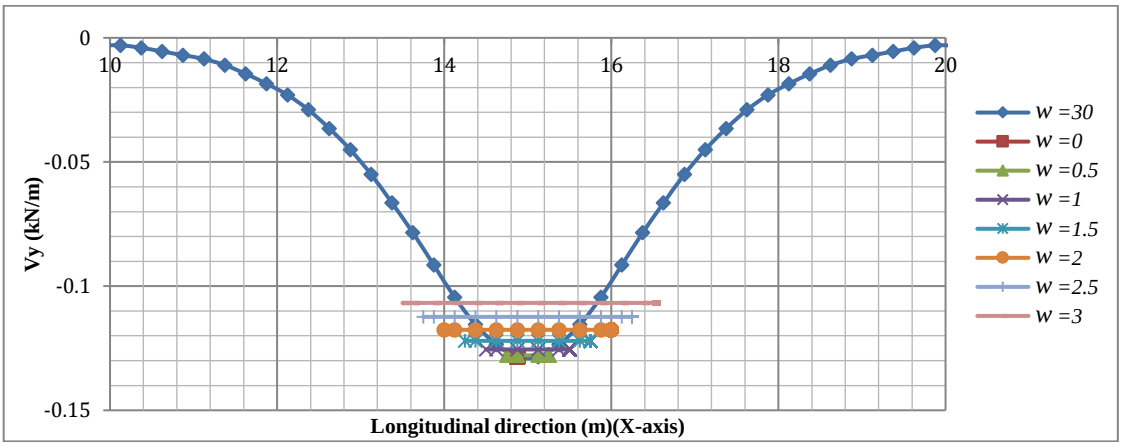


Figure 3.15 Shear force variation and averaged shear force values for section Y = 5 m *(Midspan). The legend marks refer to the original non-averaged moment (w = 30) and to the average shear force corresponding to a chosen averaging width, w.*

The moment and shear force diagrams for an averaging width of w = 1 m are shown below in Figure *3.16* and Figure 3.17 below. Each point in these two figures represents averaged sectional result, i.e. the average moment and shear force for w = 1 m in Figure 3.14 and Figure 3.15. Corresponding averaged values for each Y-section are combined into one plot to create the shear force and moment diagrams. Using different averaging widths, w, the moment and shear force diagrams across the slab in the load centerline will be different.

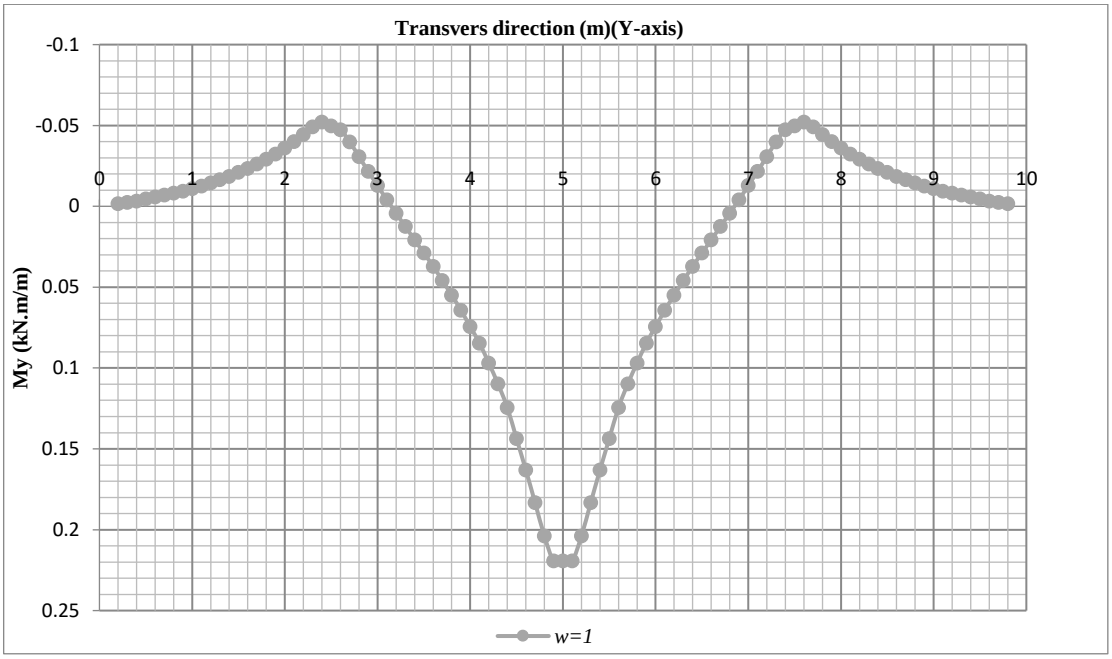


Figure 3.16 *The averaged moment diagram* in the load centerline, for a concentrated load applied at the middle of the span, *based on averaging width* w=1.

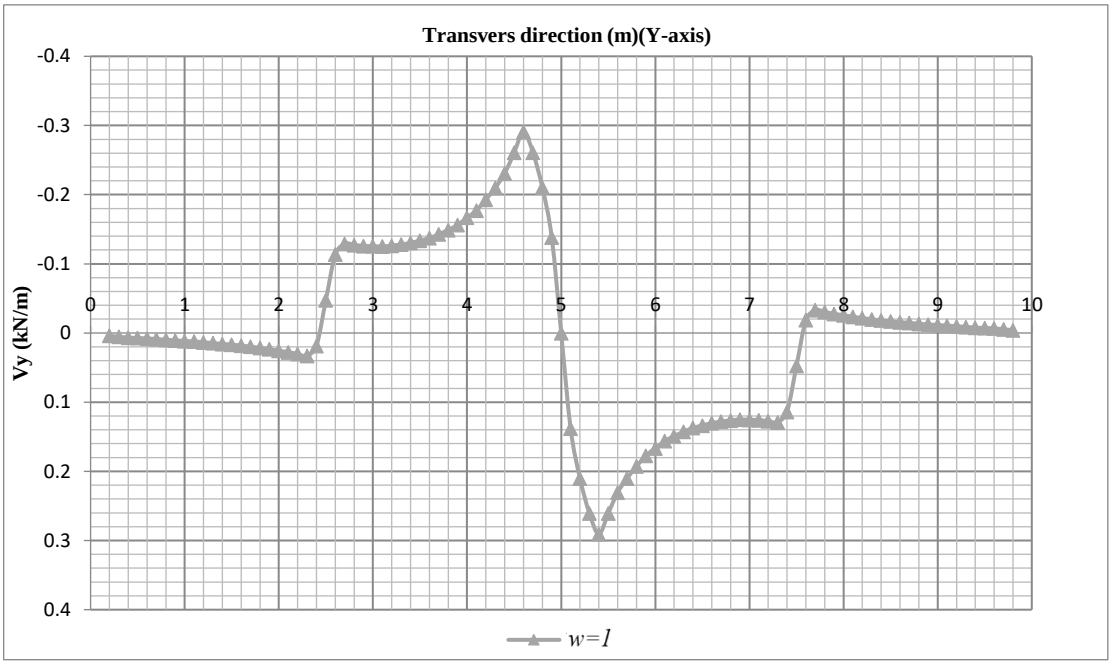


Figure 3.17 *The averaged shear force diagram in the load centerline, for a concentrated load applied at the middle of the span, based on averaging width* w=1.

Using the same procedure, when the load was applied at the edge of the cantilever, the moment and shear force diagrams are shown in Figure 3.18 and Figure 3.19 respectively.

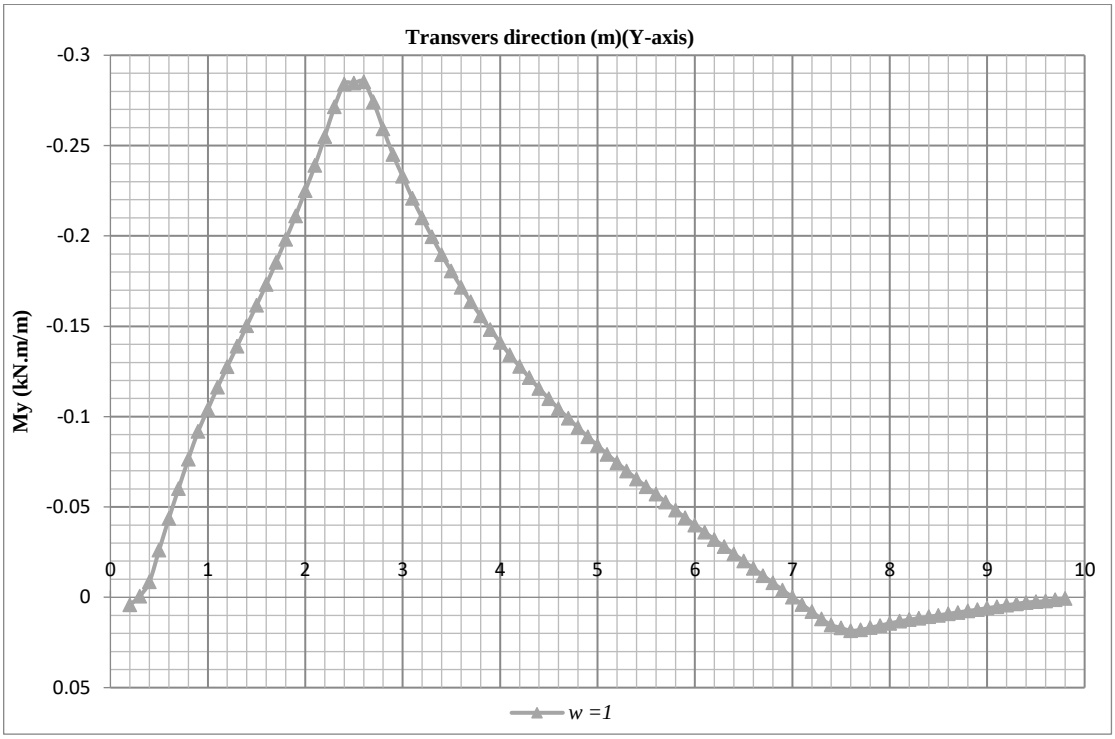


Figure 3.18 The averaged moment diagram in the load centerline, for a concentrated load applied at the edge of the cantilever, based on averaging width w=1.

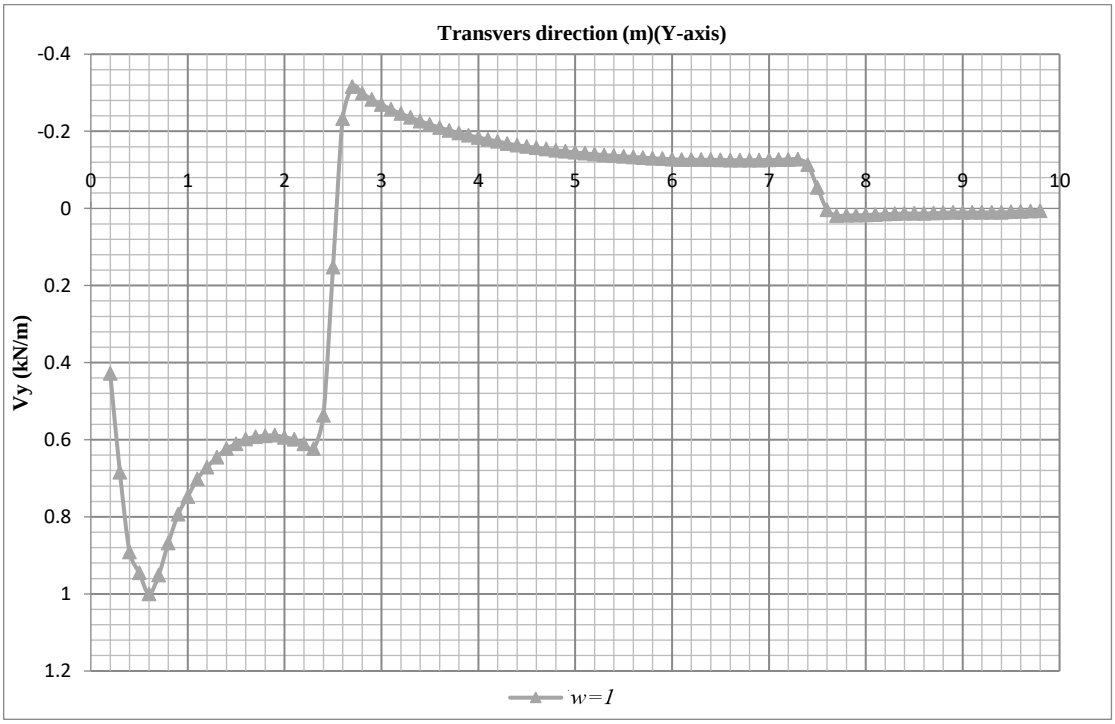


Figure 3.19 The averaged shear force diagram in the load centerline, for a concentrated load applied at the edge of the cantilever, based on averaging width w=1.

3.3 Analytical calculations

The analytical calculation model was chosen according to current engineering practice, which was a 1D simply supported beam model.

3.3.1 The original analytical calculation model

To represent the transverse section of the analyzed bridge deck, a simply supported beam of one span and two cantilevers was chosen as shown in Figure 3.20. The total length of the beam matched the FE model (10 m), both cantilevers are 2.5 m, with the same cantilever to main span ratio, $\alpha = 0.5$. The concentrated load was distributed over 0.4 m, the same as in the FE model.

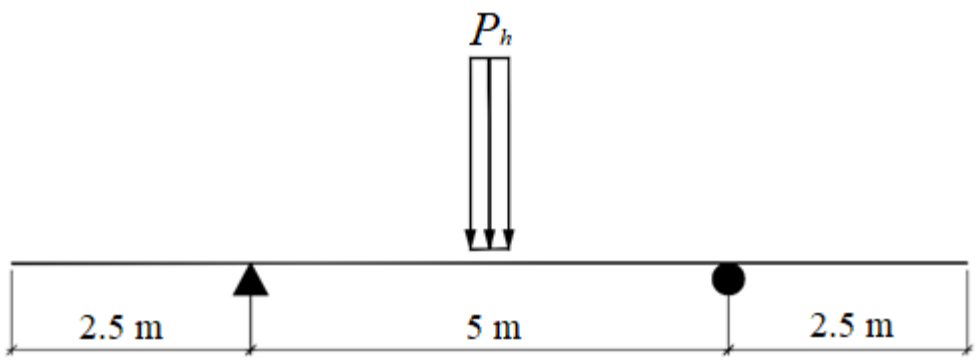


Figure 3.20 Analytical calculation 1D beam model which represents a transverse section in the slab.

Similar to the FE model, the total load of the analytical calculation model was 1 kN. Thus, the load for the analytical calculation can be determined by equation (3.3).

$$P_h = \frac{1\ \text{kN}}{b_{load}} = \frac{1\ \text{kN}}{0.4\ \text{m}} = 2.5\ \text{kN/m} \qquad (3.3)$$

Mathcad v15 [PTC (2020)] was used to plot the moment and the shear force diagrams for two different load positions, i.e. with a concentrated load at the mid-span, see Figure 3.21 and Figure 3.23, and at the edge of the cantilever, see Figure 3.22 and Figure 3.24.

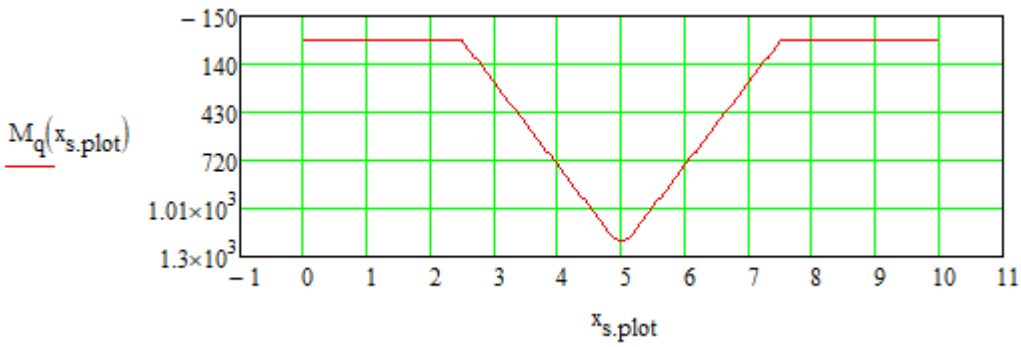


Figure 3.21 Moment diagram for the load acting at the middle of the span.

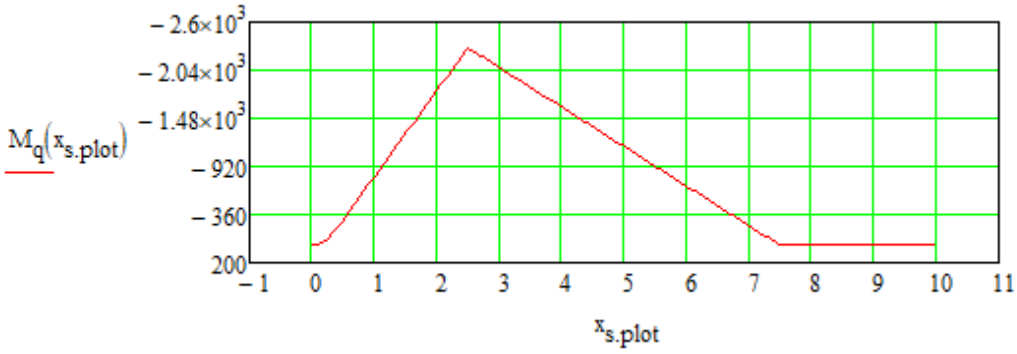


Figure 3.22 Moment diagram for the load acting at the end of the cantilever.

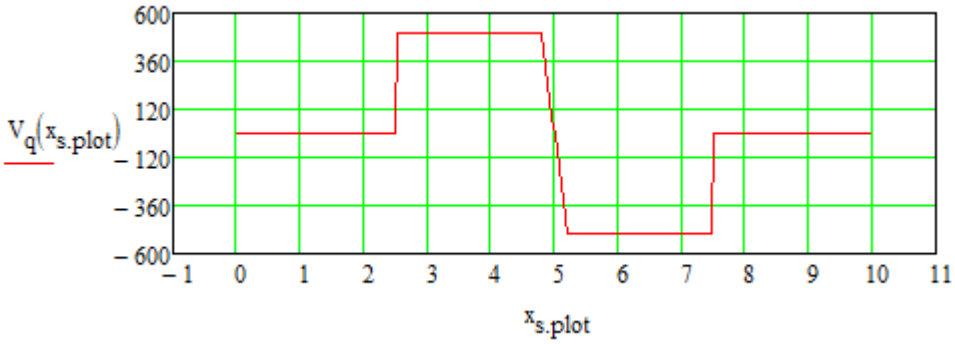


Figure 3.23 Shear force diagram for the load is acting at the middle of the span.

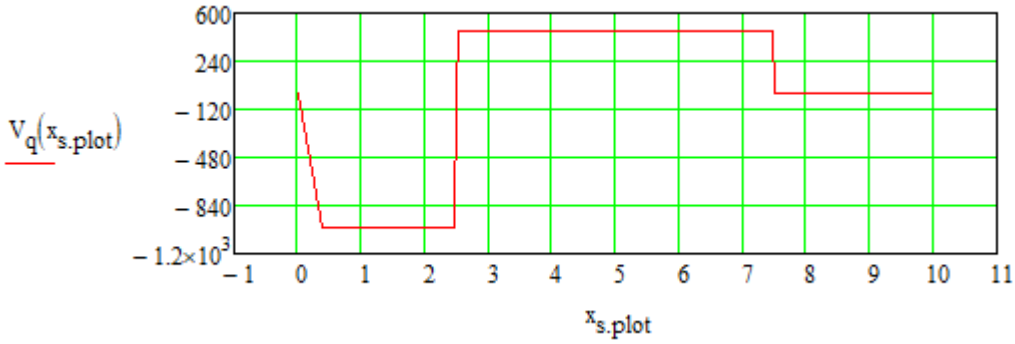


Figure 3.24 Shear force diagram for the load is acting at the edge of the cantilever.

3.3.3 Result comparison with the FE model

For the load positioned at midspan, a result comparison between analytical calculations and the FE model was carried out. The differences highlighted the need to enhance and calibrate the analytical calculation model to better represent the behavior of the bridge deck, with its extension perpendicular to the 1D model.

Since the units of internal forces of analytical calculation and FE model were not identical, the results of the two models were normalized. i.e. the results of each model were divided by its absolute maximum value. In this way, the behavior and pattern could be easily compared.

The normalized moment and shear force diagrams acquired from the FE model and the analytical calculation are shown in Figure 3.25 and Figure *3.26*.

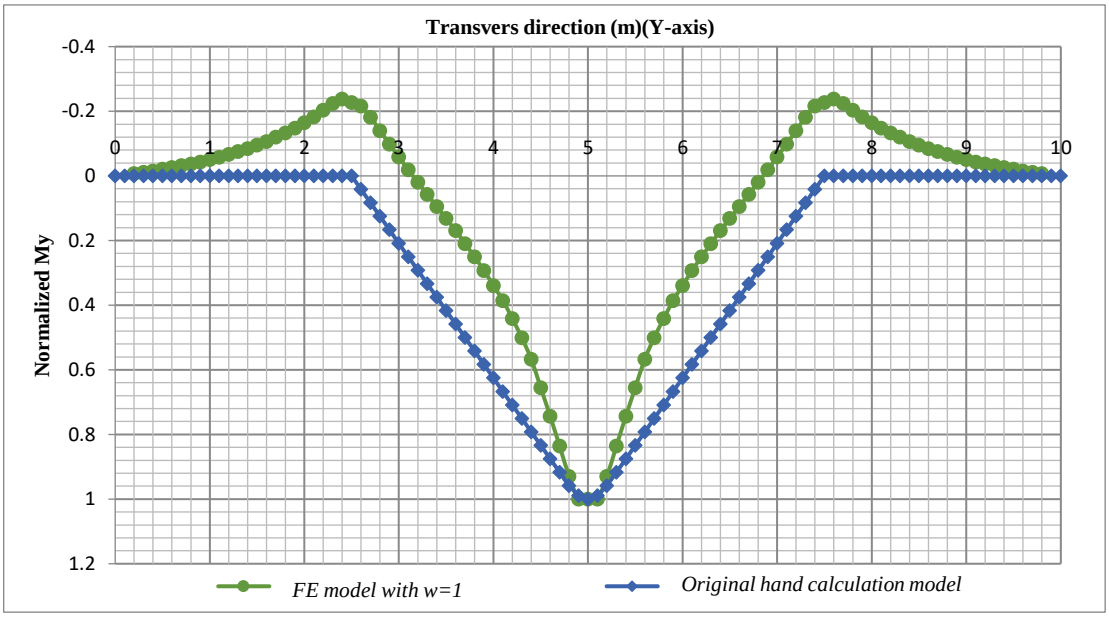


Figure 3.25 *Comparison of normalized moment diagrams from FE analysis and original analytical calculation with load position at the center of the main span* ($y = 5\ m$).

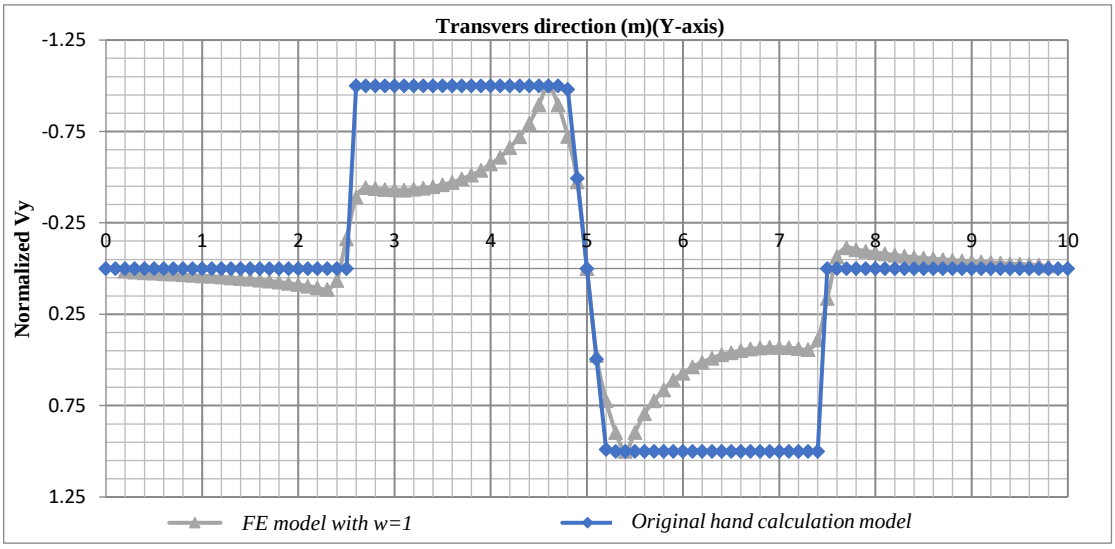


Figure 3.26 Comparison of normalized *shear force diagrams from the FE model and analytical calculation with load position at the center of the main span* ($y = 5$ m).

Through the moment diagram comparison showed in Figure 3.25, a clear pattern mismatch between the two models could be identified. In the analytical calculation results, the moment was considered constant in the cantilever zones with a value equal to zero. On the other hand, in FE analysis results, a negative moment was observed which increase from zero at the edge to the maximum at the support. This support moment is a result of that the slab in the FE model extends in the direction perpendicular to the studied section, and that adjacent parts of the slab contribute to carry the load. When the ends of the cantilevers want to deform upwards in the loaded section, the adjacent parts of the slab counteract this, causing a negative support moment.

A local effect was also observed in the FE model which leads to sharp changes in the moment values near the load position. This was not considered as a mismatch, but rather a local effect which was addressed during the evaluation of equivalent width in the upcoming section 3.3.5.

3.3.4 The modified analytical calculation model

As observed in section 3.3.3, the extra negative moment at the support resulted in a mismatch of the zero-moment section's position of FE and analytical calculation models. Ignoring this mismatch would result in inaccurate calculations of equivalent width. To avoid this, an extra constrain was given at the support to match the zero-moment sections' positions.

To achieve the constrain effect, two different approaches that would not highly complicate the analytical calculation model were evaluated. In the first one, which is the simplest, a negative moment was added directly at the support, as shown in Figure 3.27.

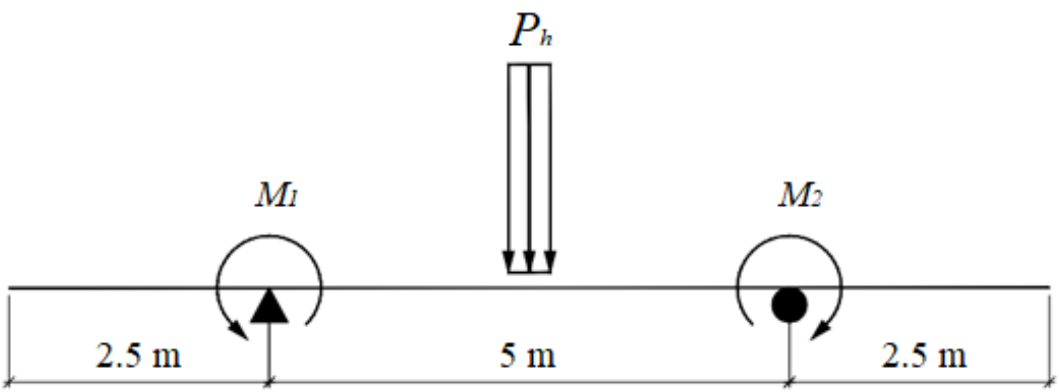


Figure 3.27 The analytical calculation model with concentrated dummy moments at the supports, when the load at $y = 5\ m$.

This will give good result representation for the moment distribution in the span. But, since the moment value is zero in the cantilevers, the models will still have a mismatch.

The second approach was to add a dummy load on the edge of the cantilever in such a way both effects can be achieved. As shown in Figure *3.28* and Figure 3.29, the dummy load can be chosen as a concentrated load or a distributed load, since both of them are sufficient to express the increment of the negative moment along the cantilever.

As they are both easy to utilize, further investigation was done by checking the shear force diagram as the shear force diagram is the first-order derivative of the moment, which can be noticed from Figure *3.26*. The shear force was approximately increasing linearly which matches the distributed dummy load and hence would give a more accurate pattern than the point load. As a conclusion, the modified analytical calculation model was chosen according to Figure 3.29.

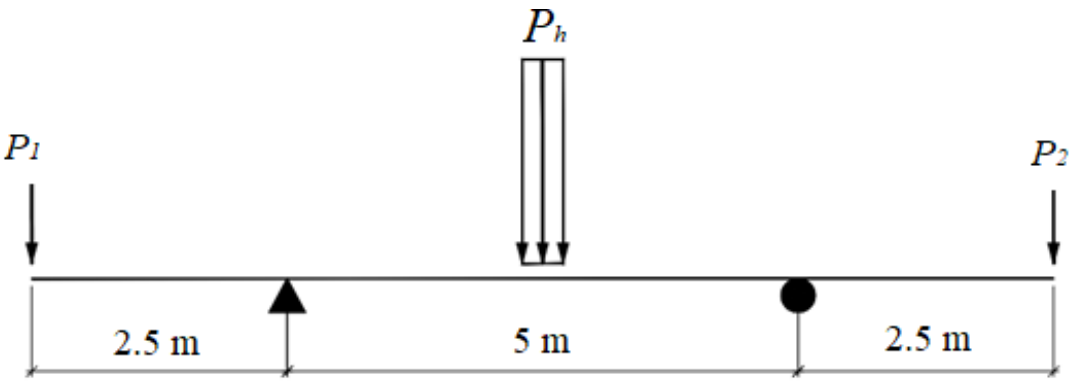


Figure 3.28 *The analytical calculation model with concentrated dummy loads when the load is at* $y = 5\ m$.

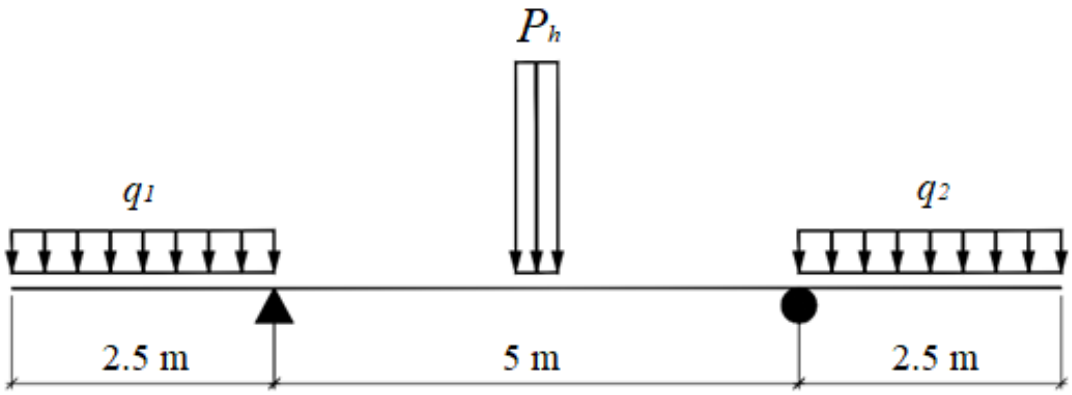


Figure 3.29 The analytical calculation model with distributed dummy loads when the load is at $y = 5\ m$.

The magnitude of the distributed dummy load q_1 and q_2 were determined by matching the zero-moment section's position in the analytical calculation model to the FE model. This process was done by python scripting (see appendix C). The comparison between the moment and shear force diagrams of the modified analytical calculation model with the previous unmodified model are shown in Figure 3.30 and Figure 3.31.

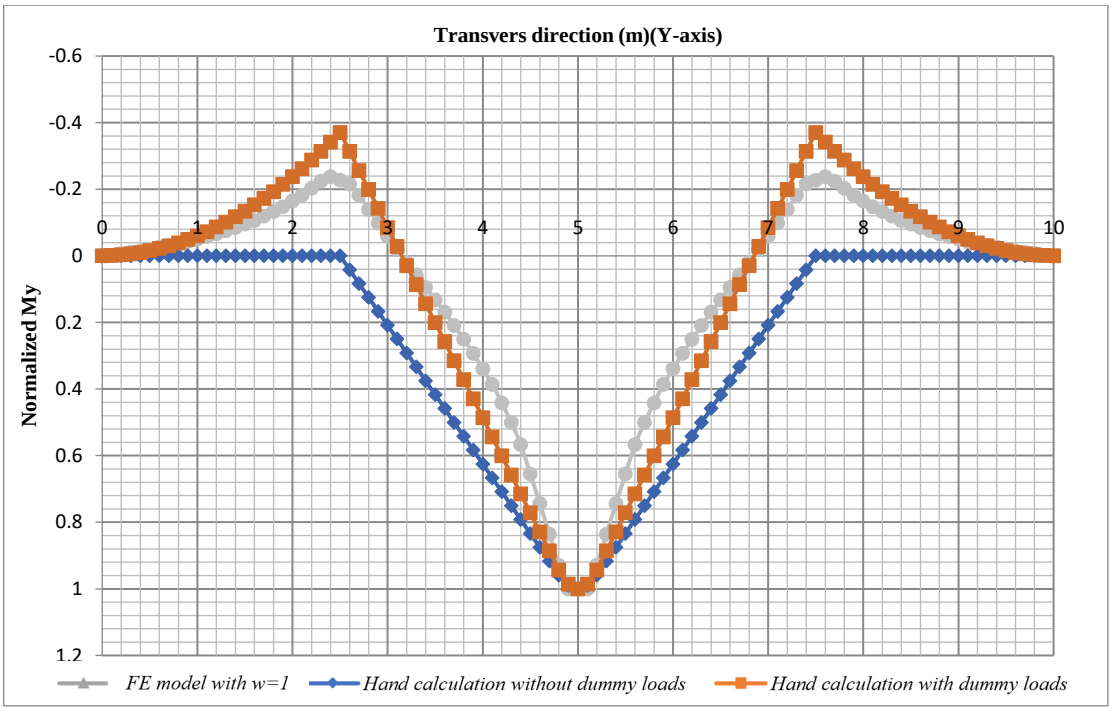


Figure 3.30 Moment diagram of the FE model, analytical calculation with and without dummy load when the load is at $y = 5\ m$.

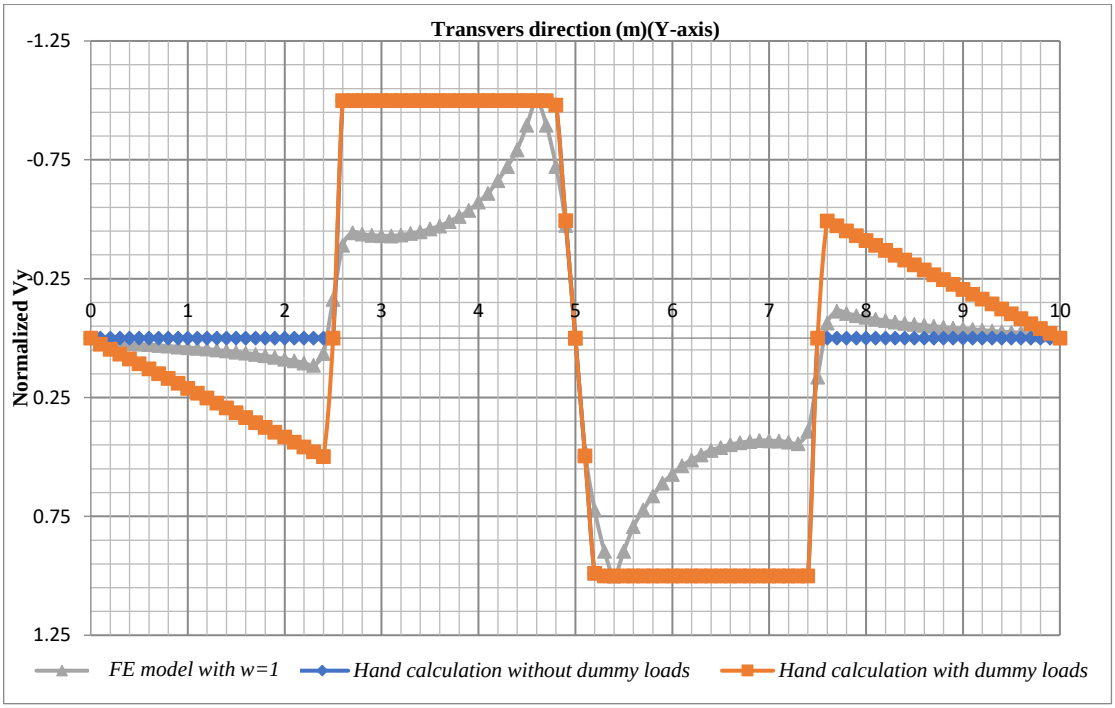


Figure 3.31 Shear force diagram of the FE model, analytical calculation with and without dummy load when the load is at $y = 5\ m$.

In the case where the load is located on the cantilever, it was observed that the constrain caused by the extension of the slab would cause a small positive moment on the support opposite to the load location, which affected the equivalent width values in the span close to the support. For the case with the load on the cantilever, a dummy load directed upwards was instead placed on the cantilever of the opposite side to the main load location as shown in Figure 3.32. The result comparison is shown in Figure 3.33 and Figure 3.34.

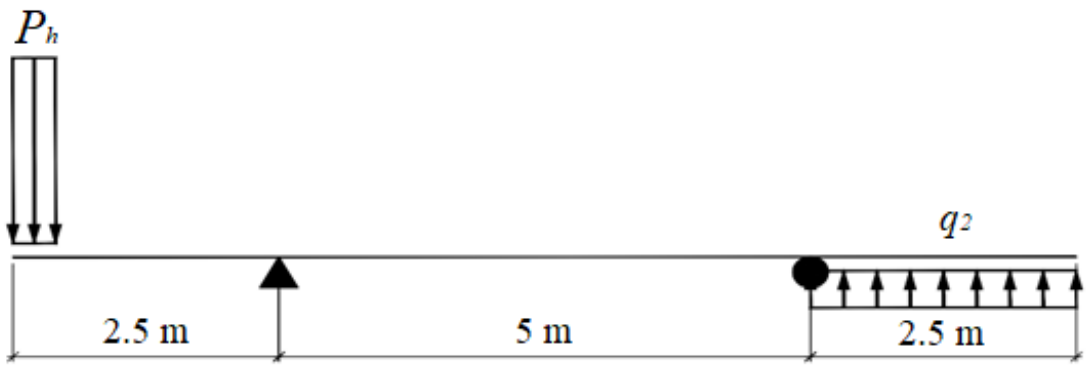


Figure 3.32 Analytical calculation model with distributed dummy loads when $x = 0.2\ m$.

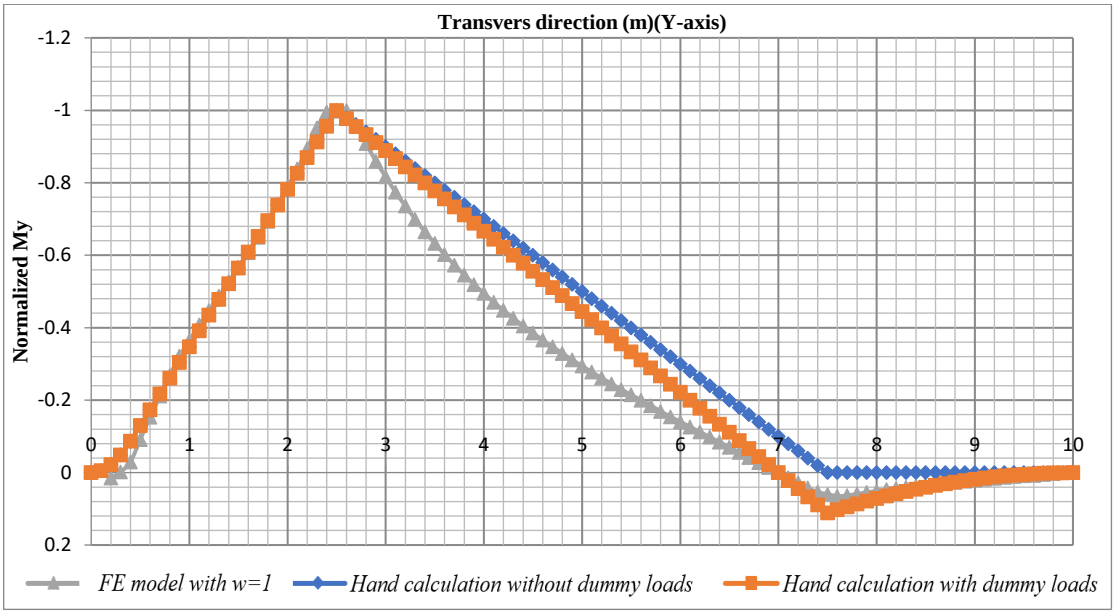


Figure 3.33 Moment diagram of the FE model, and analytical calculation with and without dummy load. when the load is at $y = 0.2\ m$.

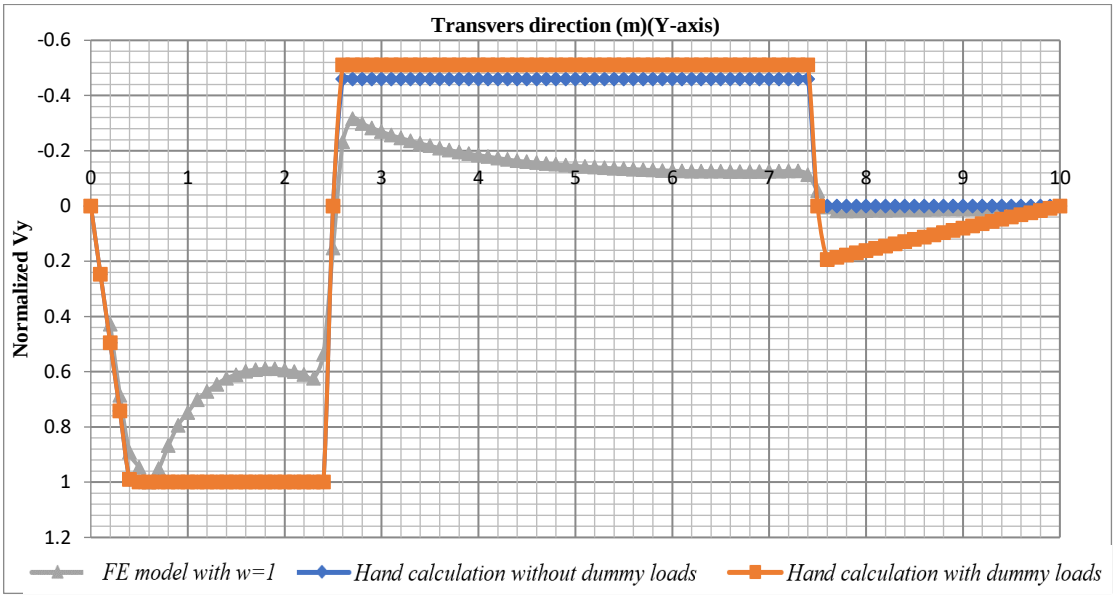


Figure 3.34 Shear force diagram of the FE model, analytical calculation with and without dummy load. when the load is at $y = 0.2\ m$.

3.3.5 Equivalent width

The main purpose of this thesis was to find a relationship between the FE model and the analytical calculation in order to determine an expression for the equivalent width, b_{eq}, to be used to distribute the sectional forces from the analytical calculation. Before going further, it is very important to mention that the origin was shifted to the centerline of the load keeping the X, Y, and Z directions unchanged to make the scripting process and the representation easier.

At any selected section, the equivalent width for moment and shear force can be determined by dividing the analytically calculated sectional force with corresponding average value from the FE analysis, as in equations (3.4) and (3.5).

$$b_{eq}(w) = \frac{M_h}{M_{av}(w)} \tag{3.4}$$

$$b_{eq}(w) = \frac{V_h}{V_{av}(w)} \tag{3.5}$$

For the modified analytical calculation model, with the dummy loads according to section 3.3.4 included, the equivalent width of moment and shear forces based on different averaging widths for different load positions are shown in Figure 3.35 to Figure 3.37. In these figures, the equivalent width was represented by the distance between two points on a line parallel to the longitudinal X-axis i.e. parallel to the supports.

It can be seen in Figure 3.35 the equivalent width for the moment varies, when the load is applied at the mid-span, in intervals: In the span, at the region surrounding the load position (the local load effect region), the equivalent width increases approximately linearly towards the supports until it becomes approximately constant outside this region. The local load effect region covers approximately half of the span length. At the cantilever, the distribution is governed by the negative moment caused by the dummy load; the equivalent width decreases approximately linearly from the supports towards the edges.

An interesting observation was that the averaging width used for the FE analysis results, only affect the equivalent width at the local load effect region, the influence at all other parts was negligible.

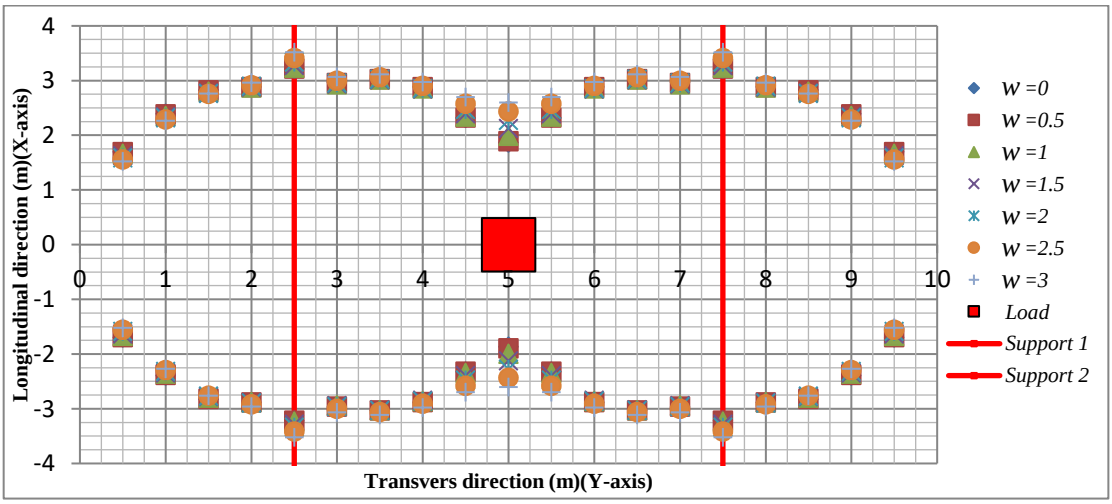


Figure 3.35 The equivalent width of the moment when the load is at $y = 5\,m$, for different averaging widths, w.

The equivalent width for shear force was more influenced by the averaging widths, see Figure 3.36. The equivalent width changes more than for the moment within the local effect region at mid-span. However, the pattern for the change in equivalent width is similar to the change in equivalent width for the moment: it increases linearly from the load position towards the supports within the local load effect region and then becomes almost constant (or has a much smaller inclination). At the cantilever, the equivalent width shows a bilinear behavior. Moving from the supports towards the edges, it first increases and then decreases. It was also observed that the equivalent with for shear force varied more than that for the moment, it was smaller in the span, while it was larger in the cantilever.

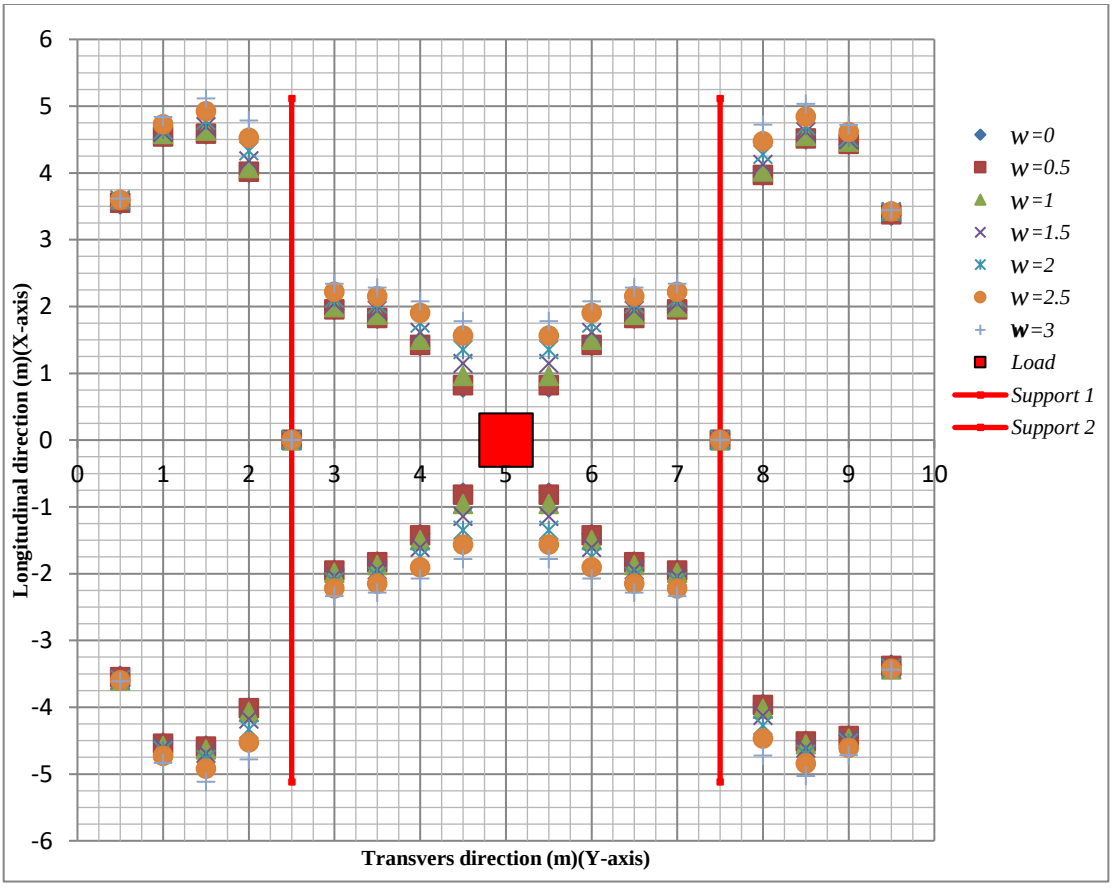


Figure 3.36 Equivalent width of shear force when the load is at $y = 5\ m$, for different averaging widths, w.

When applying the load at the edge of one cantilever, as shown in **Error! Reference source not found.**, the equivalent width of both moment and shear force show similar patterns at the span and the opposite cantilever (without the main load). At the span, in the direction away from the main load, the equivalent width first increases but ends up with a constant value. At the opposite cantilever, it decreases in the same direction. At both mentioned regions, the equivalent width shows negligible dependence on the averaging widths used for the FE analysis results. However, at the cantilever subjected to the load, the influence of averaging width is significant. For the moment, in the direction away from the load, if the averaging width was chosen as $w = 1.5$ m, the equivalent width was almost constant. If instead $w < 1.5$ m it it increases, and oppositely it decreases when $w > 1.5$ m. For shear force the equivalent width increases from the load towards the support, and the inclination is inversely proportional to the averaging widths.

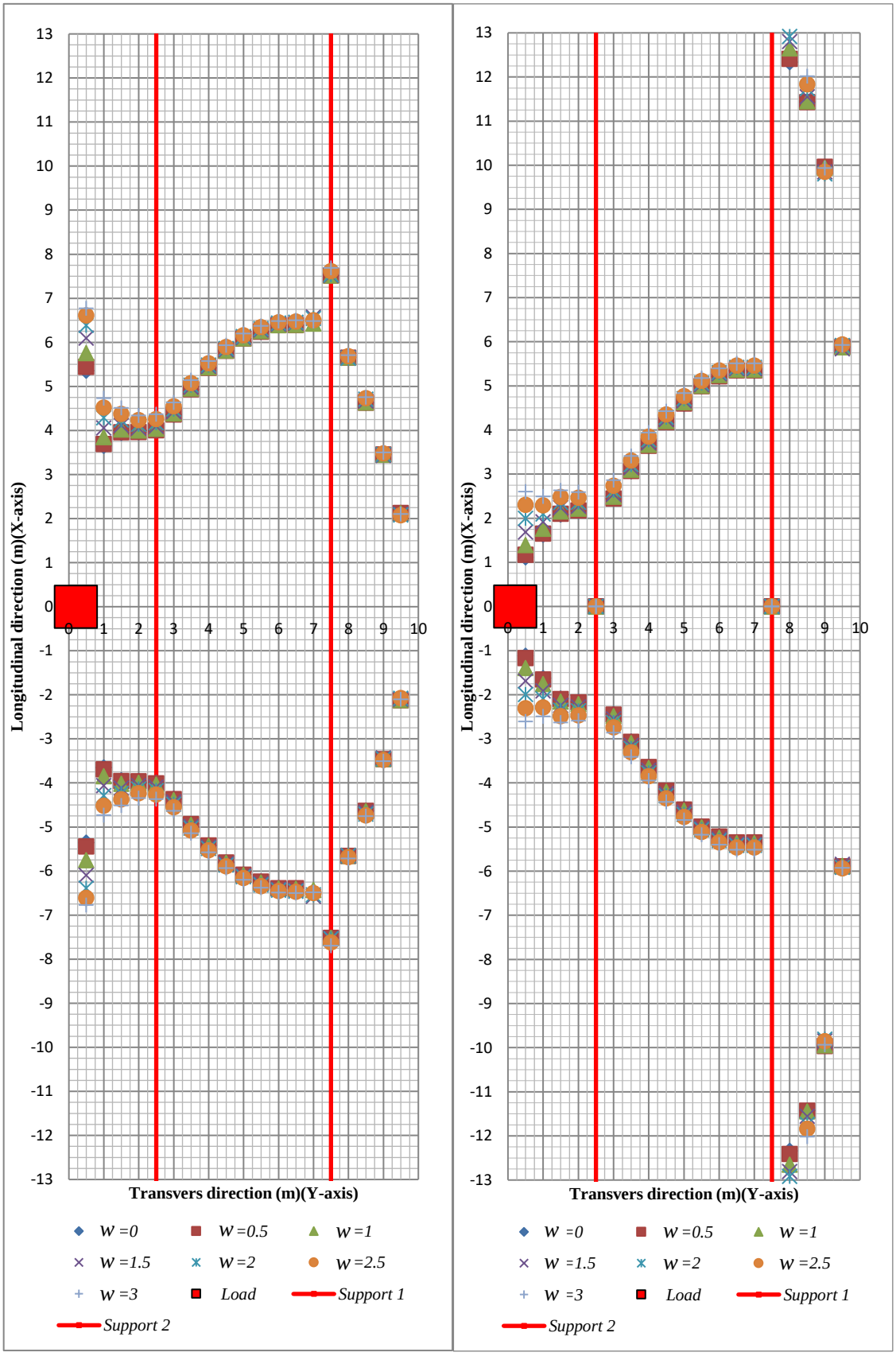


Figure 3.37 *(Left) Equivalent width of the moment when the load is at* $y = 0.2$ m, *(Right) Equivalent width of the shear force when the load is at* $y = 0.2$ m. *Shown for different averaging widths, w.*

3.4 Discussion and conclusion of the base model study and prediction for full parametric study.

To conclude this chapter, several aspects were highlighted and discussed. Furthermore, the influence of these aspects on the results was clarified as the base model was the starting point for the parametric study in the next chapter.

3.4.1 Dummy load

As explained in section 3.3.4, the dummy loads were used for compensating the mismatch between the FE and analytical calculation models. The mismatch was because the 1D beam model did not have an extension in the longitudinal direction of the bridge and, consequently, could not reflect the constrain from other parts of the slab, as the 2D FE model did.

The averaging width w represents a chosen width of the slab, over which the sectional force or moment from the FE analysis was averaged. However, in the FE analysis, the load was not resisted only by the part of the slab within this width; the remaining part of the slab also contributed to the resistance and obtained internal forces at both sides of the chosen width. Consequently, the dummy load would not be needed if the whole bridge length was selected as the averaging width.

It is noticeable that the intensity of the dummy load was inversely proportional to the averaging width w. Or to be precise, it was inversely proportional to the ratio of the stiffness of the slab within the chosen averaging width in relation to the whole slab's stiffness, since the stiffness is the parameter directly related to the contribution of resistance.

The study of the base model showed that, when the load was located at the cantilever, the direction of the dummy load was always upward, as in Figure 3.32. On the contrary to the main load effect, from self-weight and distributed traffic load, the dummy load created a positive moment in the cantilever and at the support. However, this is not critical for design since the main load effect at these positions always is the negative moment. In this case, the contribution of the dummy load counteracts the main load effect. Hence, to obtain a conservative design, the dummy load should be neglected. Consequently, the dummy load was not included for the load case with the main load located at the cantilever.

When the load was located at the span, the dummy loads created negative moments in the cantilevers and at the supports. This contribution increases the critical moment at the supports and can therefore not be neglected. For the base model study, the intensity of the dummy load versus load location in the span is shown in Figure 3.38. To determine the intensity of dummy load, the effect of other parameters such as the slab thickness, span length, cantilever to span ratio, etc. must be studied and hence a full parametric study will be carried out in Chapter 4. As a conclusion, a simplified relation between the dummy loads and related parameters is necessary to be provided as a guideline for designers.

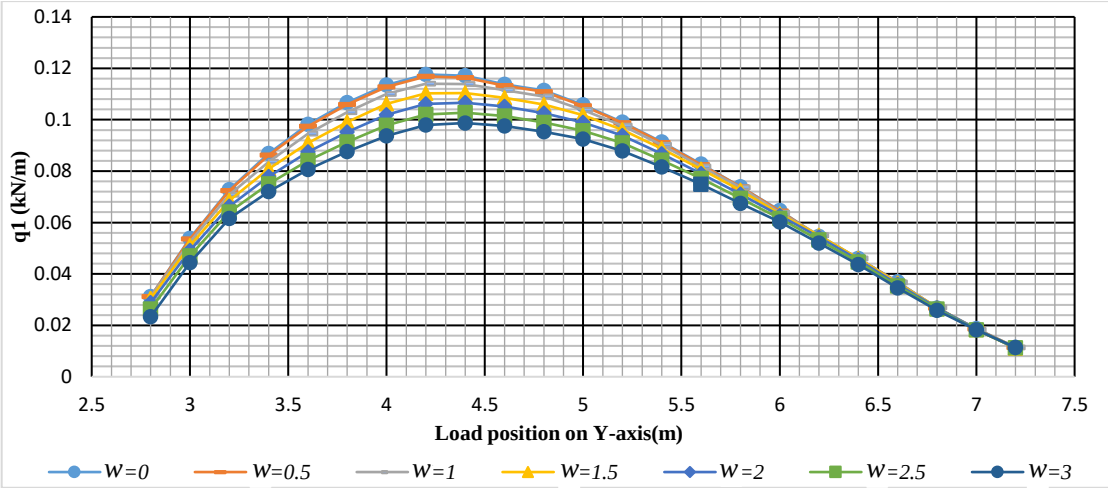


Figure 3.38 Relation between q_1 value and the load position in the span.

3.4.2 Influence and dependency of the averaging width.

The equivalent width b_{eq} , that was to be determined in this study, is related to the choices of averaging width w used for the FE analysis results. The limitation of choosing the averaging width for both moment and shear force was treated in [Pacoste *et al.* (2012)].

Based on the results of the base model, it was found that a more conservative equivalent width b_{eq} was obtained by choosing a smaller averaging width w. This can be seen from Figure 3.35 to Figure 3.37. It was also determined that the value of b_{eq} was not highly affected by the average width. This indicated that results for $w = 0$ were accurate enough to establish conservative recommendations for b_{eq}.

3.4.3 Influence of torsional moment

As mentioned in section 3.2.7, the moment values that were extracted from the FE shell analysis of the base model study were shell element internal moment in the y-direction M_y. Hence, the equivalent width values shown in Figure 3.35 and Figure 3.37 were based on the internal moment.

On the other hand, for the design of reinforced concrete slabs, the "design moment" M_{ry} is required. Consequently, the equivalent width should rather relate to the design moment instead. The design moment M_{ry}, can be calculated from the internal shell moment M_y and the torsional moment T_{xy} according to equation *(3.6)* [Engström B. (2011)]. The factor μ can be taken as 1 to have a conservative assumption.

$$M_{ry} = M_y \pm \mu|T_{xy}| \qquad (3.6)$$

In Figure 3.39 and Figure 3.40, a comparison was done between M_{ry} and M_y for two typical load positions using averaging width $w = 0$. For the bottom reinforcement design, where the positive moment is critical, a positive contribution from the torsional moment was accounted for ($+|T_{xy}|$); while for the top reinforcement design a negative effect was included ($-|T_{xy}|$).

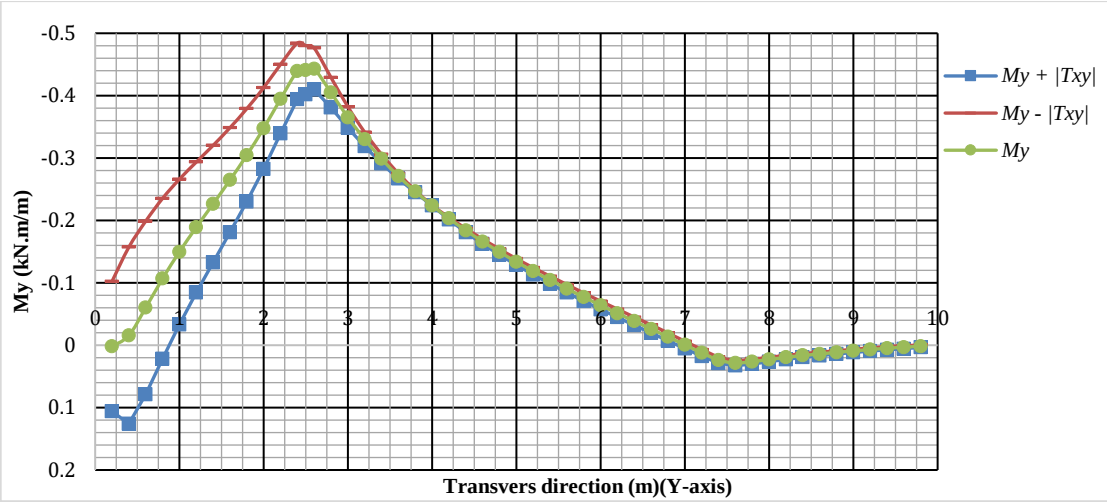


Figure 3.39 *Illustration of the effect of the torsional moment in comparison to the moment M_y obtained from the shell FE analysis when the load is located at $y = 0.2$ m.*

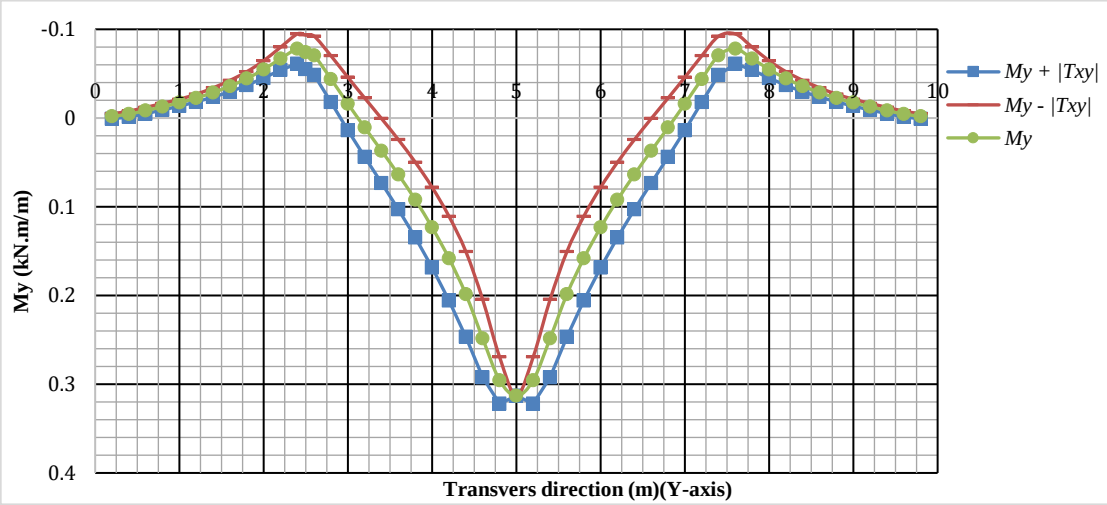


Figure 3.40 *Illustration of the effect of the torsional moment in comparison to the moment M_y obtained from the shell FE analysis when the load is located at y=5 m.*

To include the torsional moment in the calculation for equivalent width would have complicated and would have required more data from the FE model to be processed. However, as suggested in 3.4.2, the conservative approximation is to use $w = 0$, and hence the torsional moment is zero.

3.4.4 The selected averaging width for parametric study

As a conclusion from the discussions of section 3.4.2 and 3.4.3, the averaging width was chosen to zero for the rest of this study. This selection was considered conservative, yet accurate enough not to introduce any large approximations for the equivalent width for analytical calculations.

4 PARAMETRIC ANALYSIS

4.1 Introduction

This chapter focus on the general process of a full parametric analysis in order to explain the approaches used for establishing and verifying the generated guideline for choosing equivalent width in the analytical calculation model. For each part of the process, the results are addressed in the next chapter or in the appendixes.

The 1st section describes the parametric study process and how the base model variations were chosen. Other study process steps are presented, such as the FE model processing data extraction, formats and verifications, respectively.

The 2nd and the last section present the used automation method and the script structure that was utilized to produce the models, extract data, and verify the results. This section enables the user to understand and upgrade or modify the script code for future use, as a tool in related research studies or in a continuation of this project.

4.2 General process

The parametric study was carried out based on the conclusion made in the convergence study and the base model study. The general process of parametric study is illustrated in Figure 4.1.

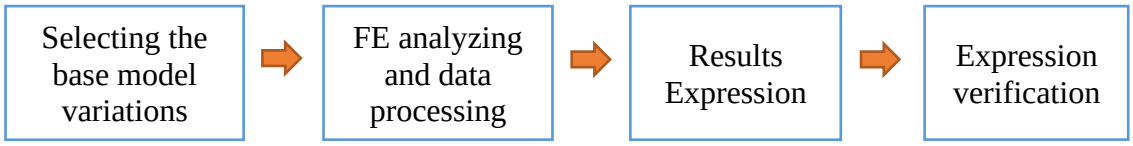


Figure 4.1 The parametric study process.

4.2.1 Selecting the base model variations

The purpose was to study the influence of some geometrical and material properties. The main idea was to change one parameter of the base model at a time and compare the results.

The following parameters were selected, and the specified variations were given for each parameter, as shown in Figure 4.2:

- The total deck width, $(l_m + 2l_k) = 10$ m in the base model, was here increased to 20 m.
- The cantilever to span ration, $\alpha = \frac{l_k}{l_m} = 0.5$ in the base model, was here changed to 0.25 and 0.75, respectively.
- The constant slab thickness, t = 0.3 m in the base model, was here changed to variable thickness, see Figure *4.3*. To achieve this, three different thickness combinations in the red dashed boxes were examined. The thickness in the arrays mentioned in Figure 4.2 refers to the effective depth of each slab part in the sequence of $t = [d_{edge}, d_{supp}, d_{mid}]$, as shown in Figure *4.3*. The slab is symmetrical about its mid-span section.
- Young's modulus, E = 35 GPa in the base model corresponding to concrete strength C40/50, was here changed to 27 GPa corresponding to concrete strength C12/15. This variation was not expected to influence the results.

- The constant slab thickness, t = 0.3 m in the base model, was here changed to 0.45 m. This variation was not expected to influence the results.

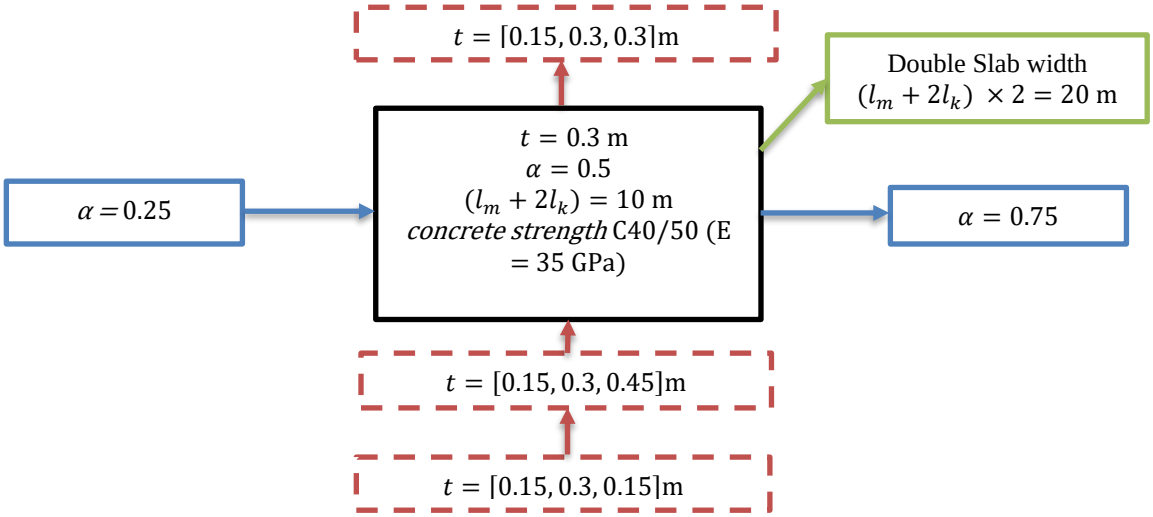


Figure 4.2 Base model variations.

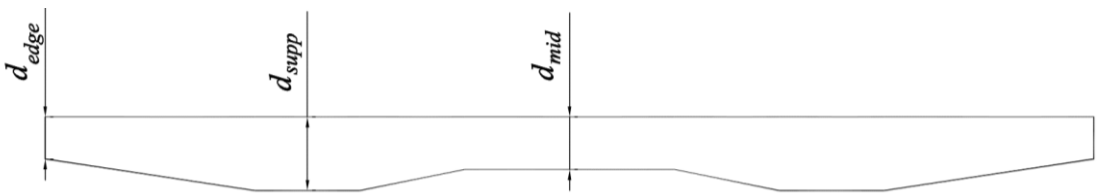


Figure 4.3 Effective depth of the bridge deck transverse section.

Where:

d_{edge}	The effective depth at the edge of the cantilever (m).
d_{supp}	The effective depth at the support(m).
d_{mid}	The effective depth at the midspan(m).
l_m	Span length (center to center) (m).
l_k	Cantilever length (edge to center) (m).
E	Young's modulus of elasticity (GPa)
α	The ratio of cantilever length to the length of the span. (Unitless)

4.2.2 FE analyzing and data processing

By automating the FE analysis with python scripting, the FEM design software [Strusoft (2020)] was controlled to run analyses with each variation of the parameters automatically. Subsequently, the FE results were sorted and processed together with the analytical calculation model. Finally, the dummy loads and the equivalent widths for both shear force and moment were stored in spreadsheet databases in Microsoft Excel [Microsoft Corporation (2020)] and plotted in graphs. The details of the script structure used for this process are introduced in section section **Error! Reference source not found.**. The plots of dummy loads and equivalent width are presented in appendix B and appendix C.

4.2.3 Curves fitting and expression

In order to create the distribution width expression that provides the best approximation for the equivalent width and dummy loads, and conservatively cover the base model and its variations explained in section 4.2.1, a graphical curve fitting was carried out using graphical linear regression. The curve fitting was done on the generated results (charts) of equivalent widths that were mentioned in section 3.3.5. The graphical curve fitting was implemented manually for each of the result charts.

The curve fitting started by fitting simple linear or bilinear relations for all the equivalent width results acquired from the base model and all the variations. These fittings were then represented mathematically by piece-wise linear functions based on the linear equation:

$$y = mx + b \tag{4.1}$$

Where y (m) is distribution width and represents the approximated equivalent width value at a section distance x (m) from a specific origin, and m (unitless) is the slope of the fitted line, and b (m) is the initial (minimum or maximum distribution width).

The final step was to specify the intervals for bilinear fitting and find a conservative relation between the constants m and b and the base model variations.

For the dummy loads, the expression could not be expressed linearly as the required fitting accuracy was high, and hence was more complicated in terms of pure graphical curve fitting since the relationship was non-linear. To simplify the fitting, multiple mechanical relations were tested such as the equation below:

$$M_{support} = \frac{P_h(x_{crit})(l_m - x_{crit})^2}{l_m^2} \tag{4.2}$$

Equation 4.2 gave the best fit to the base model dummy loads results. To improve the accuracy, it was calibrated graphically by using constants C_1 and C_2.

All the expressions of equivalent width and dummy load are presented in chapter 5. After this step, the guideline of the distribution width for analytical calculation was established.

4.2.4 Envelope verification

To verify the new guideline for analytical calculation, it was crucial to compare its results with the pure FE model results. In an analytical bridge slab design, the equivalent width is used for the calculation of envelopes for the moment and shear force. The retrieved results should then be equivalent to the FE results. As the envelope expresses the most critical sectional forces at each position of the bridge deck, for all possible load position of the load applied, it is the most important result to compare.

To calculate the envelope for analytical calculation, a python script according to section 0 was used. In this script, the expression of the equivalent width presented in section 0 was utilized.

For the FE model, the envelope was established by using the existing sectional results from FE analysis. Based on the conclusion in chapter 3, the envelope for the FE model was created using the averaging width $w = 0$. The comparison of the two envelopes is presented in chapter 5.

4.3 Program controlling and data processing by Python scripting

The script utilized multiple python Application programming interfaces (APIs)such as Pandas [The pandas development team. (2020)], Nympy [Harris *et al.* (2020)], XLSXWriter [John McNamara (2020)], and Strusoft FEM Design 19 wrapper [Strusoft (2019)] that automatically generated FE models, run the analysis, and collected results. The Strusoft FEM Design API was modified to fit our purpose in terms of data extraction, and a copy can be found in appendix D. The FE results were ingested by the script, and the final processed results were extracted in Microsoft Excel [Microsoft Corporation (2020)] sheets with multiple charts that showed the results.

The automation process could be summarized in the block diagram shown in Figure *4.4*. Each block includes multiple modules that execute a specific task. All the inputs related to the models' geometrical or physical characteristics and FE analysis parameters were embedded in the script. Thus, running the script resulted in an Excel file that contained multiple sheets with titles that represented the load moving from the edge to the middle of the deck in a specified step. Each sheet included the following:

- Moment and Shear force-related data which are averaged values for various averaging widths (w),
- FE model Moment and Shear force sectional values,
- Analytical beam model Moment and Shear force sectional values, and
- calculated equivalent width values

All these values were presented as data and in terms of various charts on the same sheet.

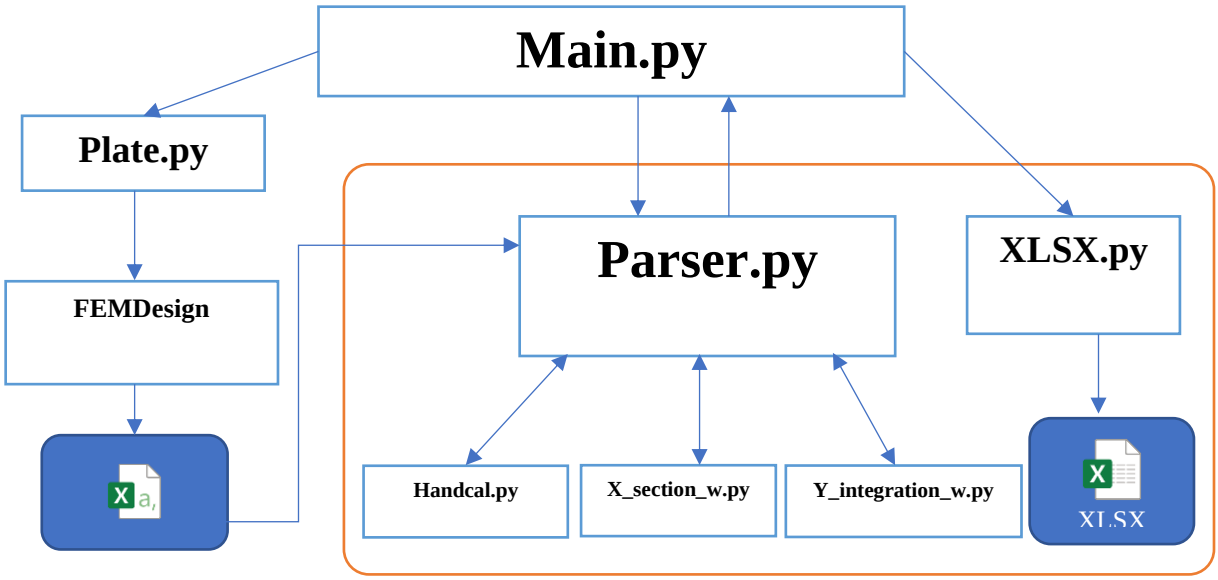


Figure 4.4 Block diagram for the Python script

Main.py

This was the main control script which calls all the other main scripts and iterates through all the main modules “Deck”, “Analyze”, and “xlsExcel”. It included all the inputs for all the other functions in terms of required parameters, model

characteristics, etc. The majority of the variables were stored in two dictionaries FEMIn and ParserIn.

Plate.py

It contained one module called “Deck” which Creates the XML model file according to the input dictionaries “FEMIn” and “ParserIn”, then run the required analysis and command FEMDesign software to extract predefined results files. It utilized FEMDesign API wrapper [Strusoft (2019)] in the process.

Parser.py

It included various modules that were controlled by the “Analyze” module. These modules would process and extract the sectional data from the CSV files according to the section coordinates. Further calculations such as, averaged moment and shear according to averaging width (w), and the required analytical calculation values for that section were done by utilizing the below tree blocks. The block export fully analyzed data back to the main block in terms of a dictionary “Para_dict”.

Handcal.py

It included modules that calculated the moment and shear force of the analytical calculation model. Furthermore, included the functions based on the FE moment results to calculate the dummy loads for each iteration.

X_section_w.py

It was a script that used the coordinate and FE sectional results to interpolate the required section between two longitudinal sections of the FE model results, and subsequently, selected the interpolated FE result within the given averaging width of that longitudinal section.

Y_integration_w.py

This script integrated the interpolated section and calculated the averaged moment and shear values according to the list of averaging width for each section.

XLSX.py

This script formatted, wrote, and created charts for the data in the dictionary “Para_dict” in a sheet that was named based on the load position.

Envelope calculation.py

This was a calculation script that utilized the expressions of equivalent widths and the dummy loads for the analytical calculation model to generate with the help of Handcal.py the envelope for the analytical calculation model and stored results in a CSV file. the results in the CSV file were later copied manually to the excel that contained the FE envelopes to be compared and verified.

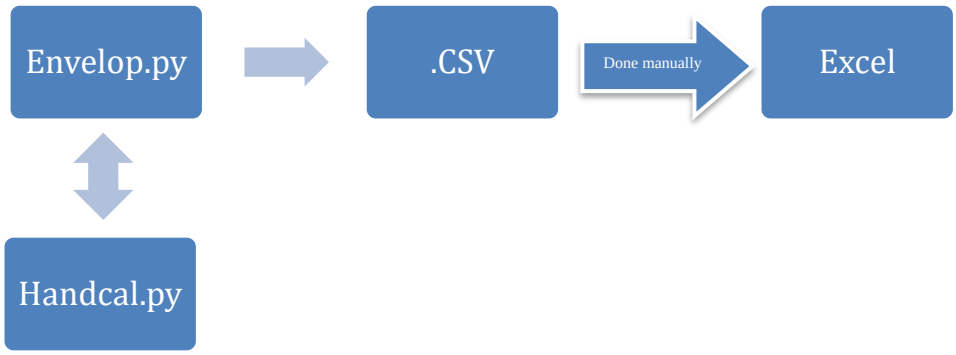


Figure 4.5 Envelop calculation.py

5 RESULTS

5.1 Introduction

This chapter presents and discusses the results of the parametric study, distribution width, and dummy loads expressions.

In the beginning, the expressions of the distribution width and dummy loads that were generated based on the parametric study are presented, with their conditions and limitations.

Following that, a result comparison between FE analysis and analytical calculation is presented using the moment and shear force envelops, and the differences are discussed.

At the end of this chapter, is a comparison of the expressions presented here to those in other guidelines, presented in chapter 2.

5.2 The expression of distribution width (b_{eff})

In this section, the expressions that were generated from curve fitting are presented, see Figure *5.1*. These expressions were generated based on the limited number of variations that were made in the parametric study as discussed in chapter 4, and where the results of the fitting can be found in Appendix B and Appendix C. The generated expressions cover all the variations in a conservative and accurate approach.

5.2.1 The distribution width for the moment

The distribution width for the moment can be expressed by the following equations (5.1*)* to (5.5*)*.

5.2.1.1 When the load is positioned at the cantilever

The distribution width at the same cantilever is:

$$b_{eff} = l_m \cdot \alpha + 2x \cdot \frac{d_{edge}}{d_{supp}} \tag{5.1}$$

The distribution width at the span is:

$$b_{eff} = l_m \cdot \alpha + 2x \cdot \frac{d_{edge}}{d_{supp}} + 2(x_{crit} - x) \cdot \tan 30° \tag{5.2}$$

Where:

α	The ratio of cantilever length to the length of the span. (Unitless)
b_{eff}	Distribution width (m).
d_{edge}	The effective depth at the edge of the cantilever (m).
d_{supp}	The effective depth at the support(m).
l_m	Span length (center to center) (m).
x	Distance between center load to the center of the nearest support (m).
x_{crit}	Distance between the center of the load to the critical section(m).

5.2.1.2 When the load is positioned at the span

The distribution width at the span is:

$$b_{eff} = 2\text{ m} + 0.1 \cdot l_m \cdot x \cdot \text{m}^{-1} + 2x_{crit} \cdot \tan 30° \quad (5.3)$$

The distribution width at the cantilever which is closest to the load:

$$b_{eff} = 2\text{ m} + 0.1 \cdot l_m x \cdot \text{m}^{-1} + 2x \cdot \tan 30° \quad (5.4)$$

The distribution width at the cantilever which is furthest from the load:

$$b_{eff} = 2\text{ m} + 0.1 \cdot l_m \cdot x \cdot \text{m}^{-1} + 2(l_m - x) \cdot tan\ 30° \quad (5.5)$$

Where:

α	The ratio of cantilever length to the length of the span. (Unitless)
b_{eff}	Distribution width (m).
d_{edge}	The effective depth at the edge of the cantilever (m).
d_{supp}	The effective depth at the support(m).
l_m	Span length (center to center) (m).
x	Distance between center load to the center of the nearest support (m).
x_{crit}	Distance between the center of the load to the critical section(m).

5.2.2 The distribution width for shear force

The distribution width (m) for shear force can be expressed by the following equations (5.6*)* to (5.9*)*.

5.2.2.1 When the load is positioned at the cantilever

The distribution width at the same cantilever and the span is:

$$b_{eff} = max \begin{cases} \frac{2}{3} x \cdot \frac{d_{edge}}{d_{supp}} + 2x_{crit} \cdot \tan 30° \\ b_{load} \end{cases} \text{ where } x \geq 0.5 b_{load} + d \quad (5.6)$$

Where:

α	The ratio of cantilever length to the length of the span. (Unitless)
b_{eff}	Distribution width (m).
b_{load}	Width of load(m).
d	The effective depth (m).
d_{edge}	The effective depth at the edge of the cantilever (m).
d_{supp}	The effective depth at the support(m).
l_m	Span length (center to center) (m).
x	Distance between center load to the center of the nearest support (m).
x_{crit}	Distance between the center of the load to the critical section(m).

5.2.2.2 When the load is positioned at the span

The distribution width at the span:

$$b_{eff} = max\begin{cases} 0.5x + 2x_{crit} \cdot \tan\ 30° \\ b_{load} \end{cases} \text{ where } x \geq 0.5b_{load} + d \tag{5.7}$$

The distribution width at the cantilever which is closest to the load:

$$b_{eff} = max\begin{cases} 0.5x + 2x \cdot \tan\ 30° \\ b_{load} \end{cases} \text{ where } x \geq 0.5b_{load} + d \tag{5.8}$$

The distribution width at the cantilever which is furthest from the load:

$$b_{eff} = max\begin{cases} 0.5x + 2(l_m - x) \cdot \tan 30\,° \\ b_{load} \end{cases} \text{ where } x \geq 0.5b_{load} + d \tag{5.9}$$

Where:

α	The ratio of cantilever length to the length of the span. (Unitless)
b_{eff}	Distribution width (m).
b_{load}	Width of load(m).
d	The effective depth (m).
d_{edge}	The effective depth at the edge of the cantilever (m).
d_{supp}	The effective depth at the support(m).
l_m	Span length (center to center) (m).
x	Distance between center load to the center of the nearest support (m).
x_{crit}	Distance between the center of the load to the critical section(m).

The limit $(x \geq 0.5b_{load} + d)$ is the limit of the "cutoff" region which is defined and discussed in section 5.4.1.2.

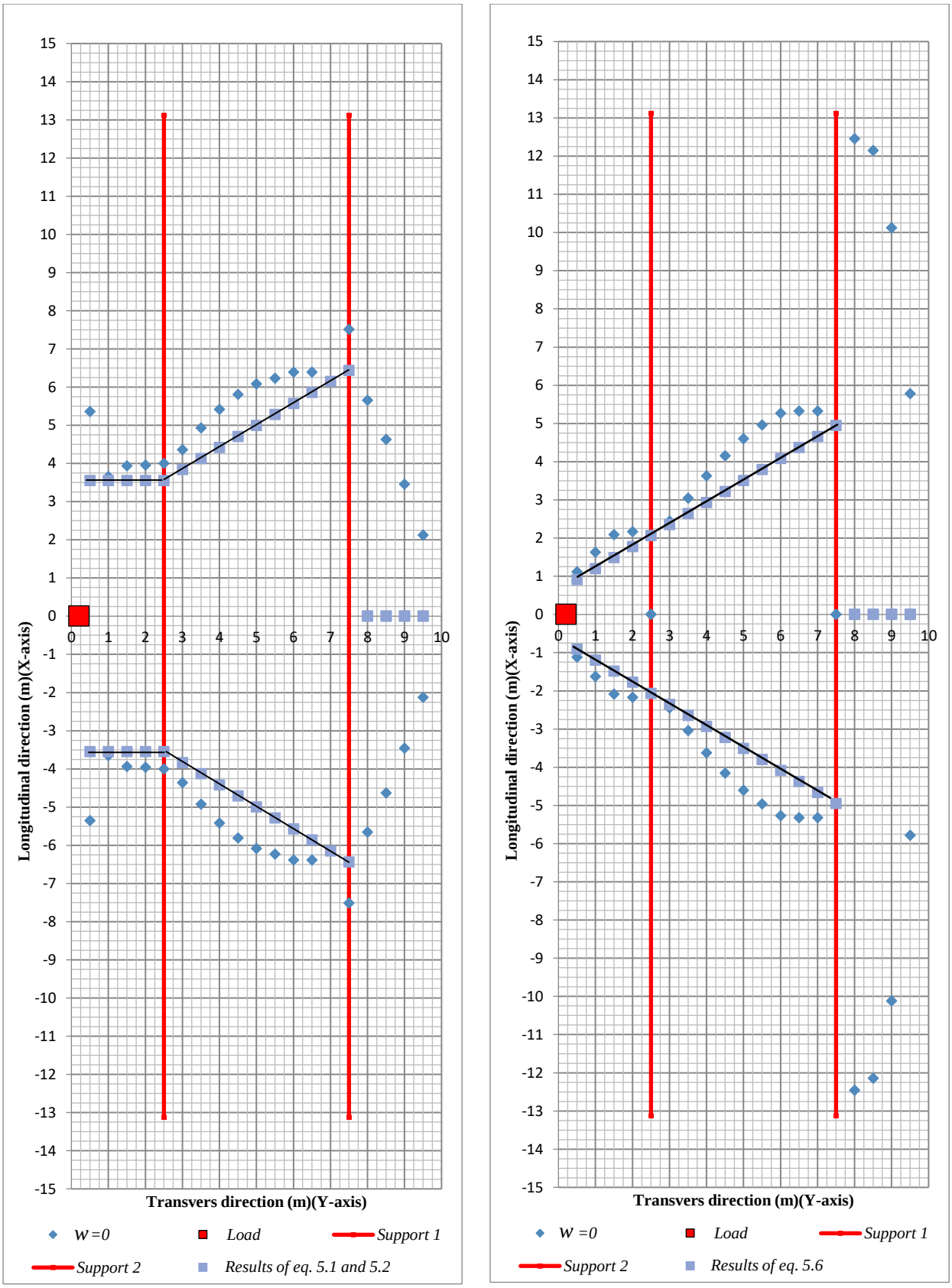


Figure 5.1 *Proposed expressions for distribution widths (Equations 5.1, 5.2, and 5.6) compared to calculated distribution widths for the base model (Chapter 3). Shown for the load at y=*0.2 m *and different averaging widths, w. (Left) Distribution width for the moment distribution and (Right) Distribution width of the shear force distribution.*

5.3 Expression of the dummy load

As explained in chapter 3, dummy loads on the cantilevers are needed for correct estimation of moment and shear force by analytical calculation when the load is positioned at the span. Unlike the distribution width, the expression for the dummy load was more complicated to determine. Due to that, a case-specific expression was determined only for slabs with constant thicknesses; to determine a common expression that would fit all the cases, also with varying slab thickness, was found to be too complicated.

Through the parametric study, it was noticed that the constants C_1 and C_2 were case-specific, as shown in Table 5.1, and that several valid combinations of constants C_1 and C_2 were possible.

5.3.1 Dummy load for bridge slab with constant effective depth

5.3.1.1 When the load is positioned at the span

The intensity of the dummy load at the cantilever which is closest to the load:

$$q_1 = C_1 \cdot x \cdot \frac{2P_h \cdot b_{load} \cdot x \cdot (l_m - x) \cdot (l_m - x + C_2 \cdot \mathrm{m})}{l_k^2 \cdot l_m^2} \tag{5.10}$$

The intensity of the dummy load at the cantilever which is furthest from the load:

$$q_2 = C_1 \cdot x \cdot \frac{2P_h \cdot b_{load} \cdot x \cdot (l_m - x) \cdot (x + C_2 \cdot \mathrm{m})}{l_k^2 \cdot l_m^2} \tag{5.11}$$

Where:

l_m	Span length (center to center) (m).
l_s	Slab length (m).
P_h	Analytical calculation point load value (kN)
l_{tot}	Total width of the slab (sum of span length and two cantilevers) (m).
b_{load}	Width of load(m).
q_1, q_2	Dummy loads (kN/m)
x	Distance between center load to the center of the nearest support (m).

For the base model variations in Figure 4.2, except for varying effective depth, the constants C_1 and C_2 may be chosen according to Table 5.1.

Table 5.1 Constant value for specific cases

Parameters Combination	C_1	C_2
Base model	0.35	1.3
Variation with changed thickness [$d = 0.45$]	0.35	1.3
Variation with changed Young's modules [$E = 27\ GPa$]	0.35	1.3
Variation with changed span ratio [$\alpha = 0.25$]	0.3	1.3
Variation with changed span ratio [$\alpha = 0.75$]	0.37	1.3
Variation with changed slab width [$w_{tot} \times 2$]	0.4	1.3

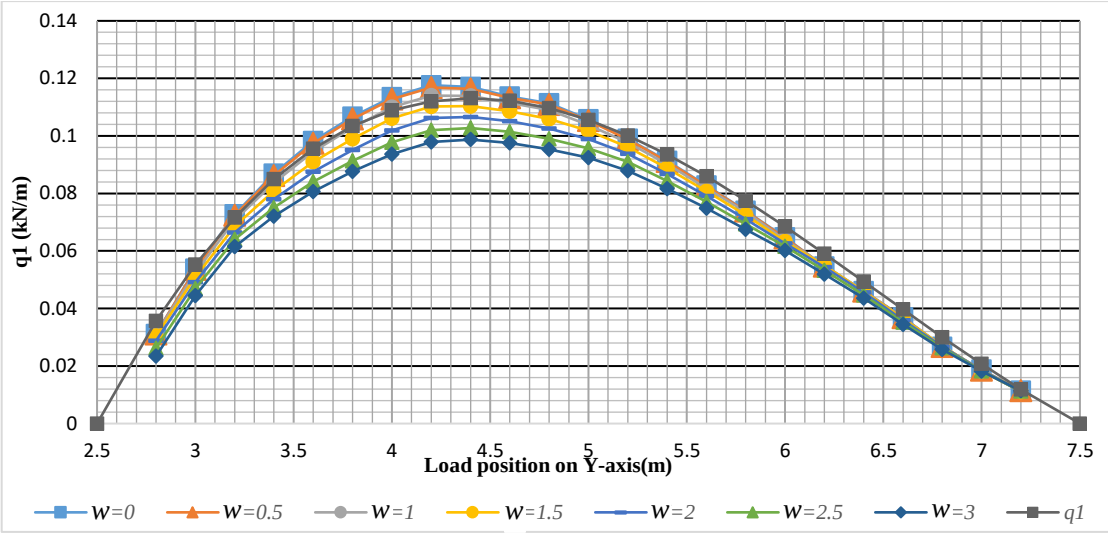


Figure 5.2 Relation between q1 value and the load position in the span.

5.4 Result verification

As mentioned in section 4.2.4, a comparison of envelopes of critical moments and shear forces was made to verify the accuracy of the expressions for distribution width and dummy load. Since the expression for the dummy load was case-specific and did not provide accurate enough results for all the cases mentioned in Figure 4.2, the comparison was only carried out for the base model.

5.4.1 Base model envelope comparison

The envelope comparison of the base model is discussed hereafter.

5.4.1.1 Envelope for moment

The envelope comparison between the analytical calculation, with the expressions in section 0, and FE analysis is illustrated in Figure 5.3

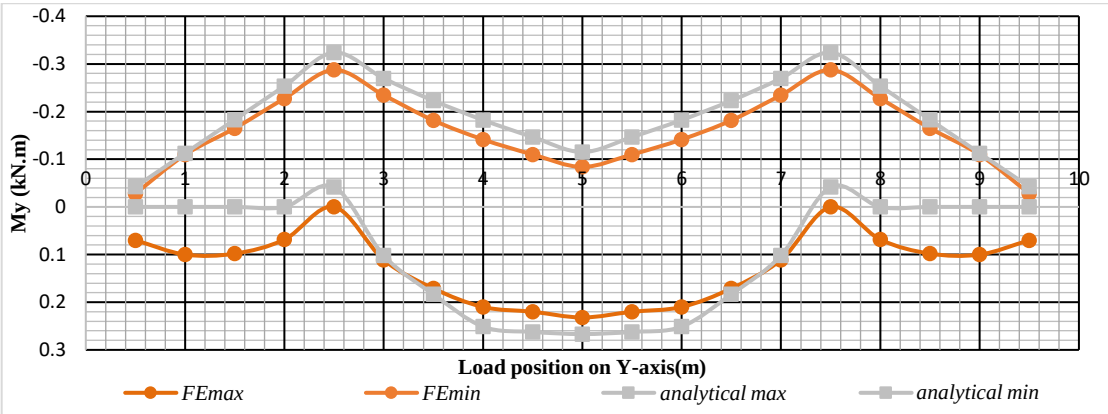


Figure 5.3 Comparison of the envelopes of the moment for multiple load positions along the Y-axis from edge to edge of the slab. (FEmax/FEmin represents Maximum/Minimum values from FE analysis, respectively. Analytical max/analytical min represents Maximum/Minimum values from analytical calculations, respectively).

From the figure, it is clear that the expressions of the distribution width and dummy load developed, the analytical calculation gives conservative values for the critical moment with small marginal in most positions. The only mismatch is for a positive moment at the cantilever. However, as discussed already in chapter 3, in design, the positive moment at the cantilever should not be considered as a critical sectional force.

As a conclusion, the expressions of the distribution width and corresponding dummy load will, when used with the updated analytical calculation model, give conservative and highly accurate results.

5.4.1.2 Envelope for shear

The comparison of the envelope for shear was much more complicated than that for moments. It can be noticed that in all expressions for shear in section 5.2.2, the distance between the studied section and the load has a lower limit of $(0.5b_{load} + d)$. According to fib model code 2010 [fib (2013)]and as in Figure 2.10, the prevailing failure mode in this region is punching failure rather than one-way shear failure, a type of failure not treated in this thesis. The other reason that made the comparison more complicated was the local effect in FE analysis. When the studied section was close to the load, the equivalent width determined from the division between the shear forces from FE analysis and analytical calculation was much bigger than the distribution width determined by the expression in section 5.2.2. Consequently, the obtained averaged shear force by the expression is over-conservative. Due to these two reasons, for all the load positions, the averaged shear force at the studied sections within this region should not be included in the envelope comparison, therefore a “cutoff” was made.

An envelope records both the maximum and minimum sectional force at all the studied sections for each load position. To make a “cutoff” means here that the obtained average shear force was replaced by zero at the sections within the region $(x < 0.5b_{load} + d)$. In this way, the influence of the averaged shear force on the envelope was eliminated within this region. The extent of the region where “cutoff” was applied $(x < 0.5b_{load} + d)$ was in this thesis defined as the region for which punching shear limits the capacity.

The “cutoff” influences the average shear force obtained from both FE and analytical calculation model. However, to see the influence of this “cutoff”, three envelope comparisons were carried out.

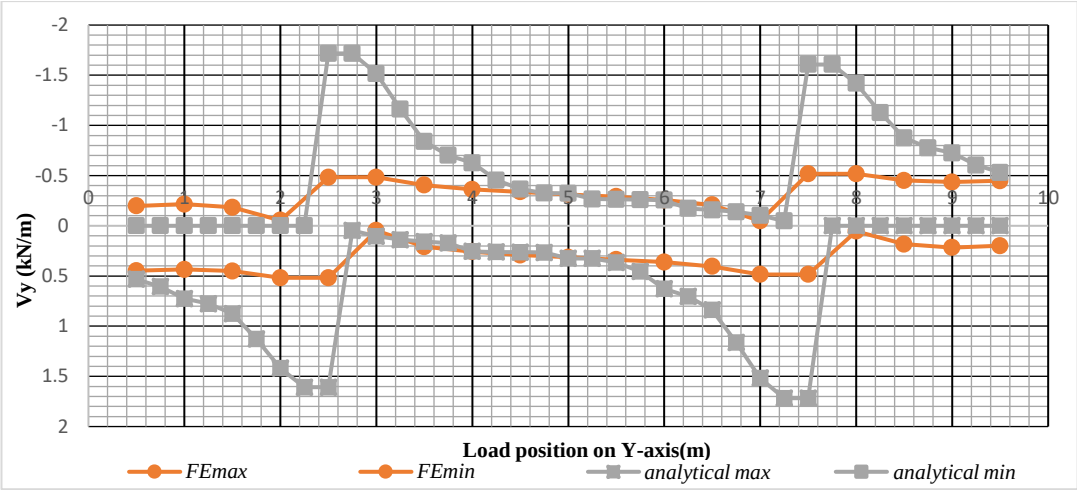


Figure 5.4 *First Comparison of the envelopes of averaged shear force for multiple load positions along the Y-axis from edge to edge of the slab between FE and analytical calculation model (both without "cutoff"). (FEmax/FEmin represents Maximum/Minimum values in FE model shear force diagrams respectively, analytical max / analytical min represents Maximum/Minimum values in analytical calculation model shear force diagrams respectively).*

In Figure 5.4, niether the FE or analytical calculation model used any "cutoff". Generally, the average shear force obtained from analytical calculation was too conservative compare to that from the FE model. However, the figure also shows that the results from the analytical calculation model is unconservative at both cantilever, where the analytical calculation gives zero shear force. As explained in section 3.4.1, this load effect at the cantilevers of the FE model was caused by restraint given by the cantilever when the load was positioned at the opposite cantilever. It was also concluded in section 3.4.1 that this load effect can be neglected and still obtain a conservative design.

The envelope comparison showed a bigger difference in the positions close to the support. The reason was that, for load positions close to the supports, the expression in section 5.2.2 gave very small distribution width, while the FE results were governed by the local effect, which gave a much bigger equivalent width in this region.

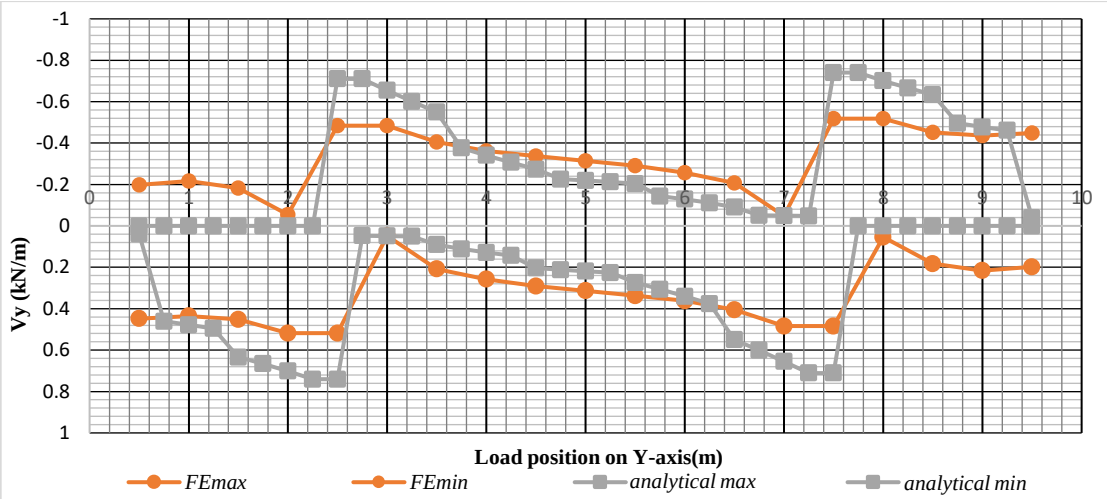


Figure 5.5 *Second comparison of the envelopes of averaged shear force for multiple load positions along the Y-axis from edge to edge of the slab between FE (without "cutoff") and analytical calculation model (with "cutoff"). (FEmax / FEmin represents Maximum/Minimum values in FE model shear force diagrams respectively, analytical max/ analytical min represents Maximum/Minimum values in analytical calculation model shear force diagrams respectively).*

In Figure 5.5, the analytical calculation model used the "cutoff" while the FE model did not. Compared to Figure 5.4, the shear force from the analytical calculation model became much smaller at the positions closer to the support. However, there is still in a big difference to the FE results.

In the span, it was noticed that the analytical calculation now was unconservative. The reason was that the "cutoff" was only used for the analytical calculation model, but the FE model still included the local effect within the "cutoff" region.

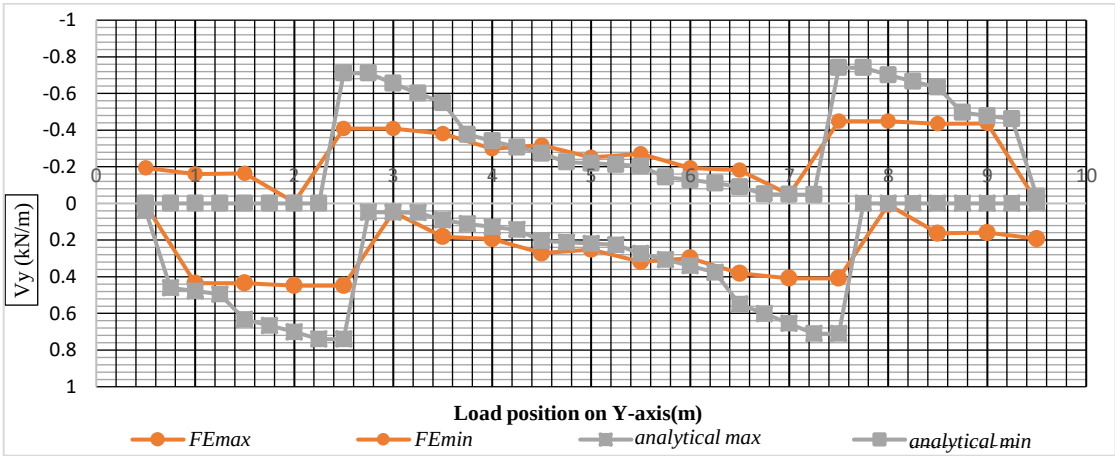


Figure 5.6 *Third comparison of the envelopes of averaged shear force for multiple load positions along the Y-axis from edge to edge of the slab between FE and analytical calculation model (both with "cutoff"). (FEmax/FEmin represents Maximum/Minimum values in FE model shear force diagrams respectively, analytical max /analytical min represents Maximum/Minimum values in analytical calculation model shear force diagrams respectively).*

The best agreement between the envelopes were obtained when the "cutoff" was made for both analytical calculation and FE model, as shown in Figure 5.6. The envelopes from the two models got a better match, but still, the difference was considarable at the positions close to the support.

As a conclusion, the expression of the distribution width for shear force was over-conservative at positions close to the support, hence further optimization can be made to enhance the accuracy.

5.5 Comparison with existing guidelines for distribution width

As stated in chapter 1, this thesis aimed to create a more accurate guideline for distribution width for analytical calculation. Based on the findings in this study, new expressions for distribution widths to be used with the proposed method for analytical calculations were presented in sections 5.2. and 5.3. In this section, a result comparison with other guidelines, presented in chapter 2, is carried out.

5.5.1 The distribution width for the moment

For the moment, the most critical sections were over the support where the maximum negative moment is located and at mid-span where the maximum positive moment is located, hence these two sections were chosen for comparison of the distribution width. The comparison was made for three base model variations, as shown in Table 5.2. to Table 5.4.

Table 5.2 Comparison distribution width for the moment, first base model variation.

$t = 0.3$ m , $\alpha = 0.3,\ and\ \ (l_m + 2l_k)\ \times 2 = 8$ m		
	load at cantilever; B_{eq} at support	load at midspan; B_{eq} at midspan
This thesis	4.1 m	3.25 m
B7 [Statens Betongkommitté (1968)]	0.44 m	2.9 m
Krav Bärighetsberäkning. [Trafikverket (2019a)]	N/A	2.7 m
BS 8110-1:1997 [BSI (1997)]	N/A	3.4 m

Table 5.3 Comparison distribution width for the moment, second base model variation.

$t = 0.3$ m , $\alpha = 0.4,\ and\ \ (l_m + 2l_k)\ \times 2 = 10$ m		
	load at cantilever; B_{eq} at support	load at midspan; B_{eq} at midspan
This thesis	6.267 m	3.54 m
B7 [Statens Betongkommitté (1968)]	0.612 m	2.9 m
Krav Bärighetsberäkning. [Trafikverket (2019a)]	N/A	2.7 m
BS 8110-1:1997 [BSI (1997)]	N/A	3.73 m

Table 5.4 Comparison distribution width for the moment, third base model variation.

$t = 0.3$ m , $\alpha = 0.5,\ and\ \ (l_m + 2l_k)\ \times 2 = 12$ m		
	load at cantilever; B_{eq} at support	load at midspan; B_{eq} at midspan
This thesis	8.6 m	3.8 m
B7 [Statens Betongkommitté (1968)]	0.782 m	2.9 m
Krav Bärighetsberäkning. [Trafikverket (2019a)]	N/A	2.7 m
BS 8110-1:1997 [BSI (1997)]	N/A	4 m

For the case when the load is applied at the cantilever the distribution width is only treated in B7 [Statens Betongkommitté (1968)]. As compared to that, the distribution width determined in this thesis is much bigger. This leads to considerably smaller design moments and a reduced amount of reinforcement.

For the case with the load at the mid-span, the distribution width determined in this thesis was bigger than according to [Trafikverket (2019a)], which is the guideline that is currently used in Sweden for structural assessment, this shows that Trafikveket's guideline is conservative. However, the comparison also shows that the distribution width determined in this thesis was slightly conservative compared with the one obtained by [BSI (1997)].

For the load positioned in the span, the analytical calculation approach of this thesis also accounts for the negative moment over the support through the influence of the dummy load. This negative moment reduces the positive moment at the mid-span. When this effect was included, the averaged moment at the span determined according to this thesis was even closer to one obtained by [BSI (1997)].

5.5.2 The distribution width for shear force

The comparison of the distribution width for shear force was carried out using the same base model variations as for the moment. The chosen critical sections for comparison were the sections located slightly outside the punching shear region, $(0.5b_{load} + d)$ *away from the load. The comparisons are presented in*

Table 5.5. to Table 5.7.

Table 5.5 Comparison distribution width for the shear force, first base model variation.

$t = 0.3$ m , $\alpha = 0.3,\ and\ \ (l_m + 2l_k)\ \times 2 = 8$ m		
	load at cantilever, $x_{crit} = 0.5$ m	load at midspan, $x_{crit} = 0.5$ m
This thesis	1.44 m	1.83 m
fib Model Code 2010 [fib (2013)]	1.8 m	1.8 m (45°) to 2.82 m(60°)
BBK 04 [Boverket (2004)]	N/A	3.65 m
B7 [Statens Betongkommitté (1968)]	N/A	3.375 m
Dutch approach [Lantsoght et al. (2015)]	N/A	1 m
French approach [Chauvel, et al. (2007)]	N/A	1.8 m
Lubell, Bentz and Collins [Lubell *et al.* (2008)]	N/A	3.5 m
Grasser and Thielen [Grasser *et al.* (1991)]	1.19 m	1.95 m
ACI Structural Journal. [Zheng *et al.* (2010)]	N/A	3.86 m

Table 5.6 Comparison distribution width for the shear force, second base model variation.

$t = 0.3$ m , $\alpha = 0.4,\ and\ \ (l_m + 2l_k)\ \times 2 = 10$ m		
	load at cantilever, $x_{crit} = 0.5$ m	load at midspan, $x_{crit} = 0.5$ m
This thesis	1.93 m	1.97 m
fib Model Code 2010 [fib (2013)]	1.8 m	1.8 m (45°) to 2.82 m (60°)
BBK 04 [Boverket (2004)]	N/A	3.65 m
B7 [Statens Betongkommitté (1968)]	N/A	3.58 m
Dutch approach [Lantsoght et al. (2015)]	N/A	1 m
French approach [Chauvel, et al. (2007)]	N/A	1.8 m
Lubell, Bentz and Collins [Lubell *et al.* (2008)]	N/A	3.89 m
Grasser and Thielen [Grasser *et al.* (1991)]	1.41 m	2.09 m
ACI Structural Journal. [Zheng *et al.* (2010)]	N/A	4.25 m

Table 5.7 Comparison distribution width for the shear force, third base model variation.

$t = 0.3$ m , $\alpha = 0.5$, *and* $(l_m + 2l_k) \times 2 = 12$ m		
	load at cantilever, $x_{crit} = 0.5$ m	load at midspan, $x_{crit} = 0.5$ m
This thesis	2.44 m	2.08 m
fib Model Code 2010 [fib (2013)]	1.8 m	1.8 m (45°) to 2.82 m (60°)
BBK 04 [Boverket (2004)]	N/A	3.65 m
B7 [Statens Betongkommitté (1968)]	N/A	3.75 m
Dutch approach [Lantsoght et al. (2015)]	N/A	1 m
French approach [Chauvel, et al. (2007)]	N/A	1.8 m
Lubell, Bentz and Collins [Lubell *et al.* (2008)]	N/A	4.2 m
Grasser and Thielen [Grasser *et al.* (1991)]	1.64 m	2.2 m
ACI Structural Journal. [Zheng *et al.* (2010)]	N/A	4.57 m

Through the comparison, it was noticed that the distribution width determined in this thesis was conservative compared to several of the references found. This was true even though the the expression was limited to exclude one-way shear failure within the punching shear region.

6 Discussion

6.1 Introduction

This chapter discusses the most important questions a designer would have when utilizing this guideline. It begins with a discussion of the general method used in this thesis. After that, the difficulties that were faced in the research study and considerations of using the recommendations developed in this thesis are discussed. At the end of this chapter, a discussion about how an optimal design can be achieved is made.

6.2 General method

The general methodology used in this thesis was to determine recommendations for the analytical calculation by comparing the results of linear 2D shell FE models with the 1D beam model. In other words, the expressions for distribution width, to be used together with the analytical 1D beam model, reflects approximately the 2D shell analysis results. By using the proposed analytical calculation model, the recommendations given in this thesis gives an approximation of the 2D she.ll FE result.

As the 2D shell FE modeling is one of the most common design methods for a one-way bridge deck design and is usually considered to be of high accuracy, it was reasonable to use this to calibrate the analytical calculation model.

6.3 Difficulties

For calculation of the moment, a simple analytical calculation model, consisting just of a simply supported beam with cantilevers, does not express the constrain from the slab cantilever that causes a negative moment at the supports. With such a model, there will be no support moments for a load in the midspan, and there will be a mismatch of the zero-moment positions, compared to the shell FE analysis. In order to reflect the effect of the constraint from the slab cantilever, dummy loads were introduced for the beam model. The magnitude of the dummy loads on the cantilevers were obtained by matching the zero-moment positions of the two models.

To obtain accurate results by analytical calculation, the size of the dummy loads must be accurately determined for all different base model variations. However, to generate a general expression for the dummy load became the largest challenge of this thesis due to the three-dimensional behavior of the slab, which cannot be approximated to 2D beam response without sacrificing accuracy. Thus, instead of a common expression, a case-specific expression valid for constant slab thickness was determined, without sacrificing the simplicity of the expression.

Two approaches to overcome this challenge were considered. The approach, used in this thesis, was to determine an expression for the magnitude of the dummy load. However, this approach would have required a deeper study of the constraint effect to find a general expression. The other approach considered was to model the cantilever and the simply supported span separately in two different models, in such a way that the dummy load at the cantilever can be avoided. When using the second approach for a bridge deck with a cantilever, the negative support moment caused by a load in the mid-span of the simply supported beam would be missing. Consequently, in this case, when a load is applied at mid-span in combination with a load at the cantilever, the

obtained accumulated negative support moment from the beam and the cantilever becomes unconservative. A simple way to compensate for this missing negative moment can be achieved by reducing the distribution width at the cantilever; this way the obtained averaged negative moment is scaled up. Using such an approach requires further studies to obtain the magnitude of reduction of the distribution width at the cantilever.

For analyzing the shear force, the biggest challenge was the local effect on the equivalent width at positions close to the load (according to section 5.4.1.2). The expression of distribution width could not reflect this local effect correctly. Hence a big mismatch was shown in Figure 5.4 at envelope comparison. The largest difference between the analytical expression the shell FE results occurred at positions close to the supports. It was noticed that the expression matched well when the analyzed section was at a relatively far distance from the load, so a limitation of the distance between the studied section and the load was introduced. By using the extent of the punching shear region, as defined by section 5.4.1.2 , as the "cutoff" distance, the accuracy of the expression increased significantly.

6.4 Consideration for using the recommendations

When using the recommendations presented in this thesis, the user must keep in mind that they are based on the 2D shell model FE analysis with a linear material response, isotropic elements, and rigid supports for the slab. For calculation of the moment, the analytical calculation model was an updated analytical calculation model with a dummy load. For calculation of the shear force, the presented expressions are limited to sections outside the punching shear region.

6.4.1 The updated analytical calculation model

In contrast to the earlier guidelines described in chapter two, the analytical calculation model in this thesis is an updated model including both the span and the cantilevers, and a dummy load to compensate for the restraint effect from the slab cantilever. When using these recommendations, the designer needs to determine the dummy load that is used to capture the negative moment at the support caused by the constraint, when the load is located at the main span.

The expression for the dummy load determined in this thesis is case-specific for slabs with constant thickness. However, the recommendations could still be used without including the dummy load. In this case, the obtained average moment at the main span becomes larger and conservative. Furthermore, the negative moment at the support and the cantilever would be missing and, consequently, the designer needs to determine the negative moment at the support through other guidelines.

6.4.2 Punching shear region as the "cutoff" region

As described at the end of section 5.5.2, the the expressions for distribution widths are valid for sections at a distance of at least a "cutoff" region from the load. The recommended "cutoff" region size of this thesis is based on model code 2010 [fib model code 2010, (2013)]. If the designer uses a different "cutoff" region size according to other guidelines that are more suitable for the specific case, it is important to keep in mind that the smaller "cutoff" region, the more conservative result will be achieved.

6.4.3 Influence of prestressing, support stiffness and element orthotropy

In the study, the effect of prestressing was not included. Consequently, the recommendations given should be used with some care for prestressed bridge decks.

This thesis is based on linear models with isotropic elements for the bridge deck. Consequently, the influence of nonlinear response due to e.g. cracking, and the orthotropy in the bridge slab this causes, is not taken into account.

Furthermore,, the supports were modeled as two rigid beams without deformations in the longitudinal direction. However, in reality, the bridge deck would follow the beams' deformation, causing different stiffnesses at different positions in the longitudinal direction due to the cracking of the concrete deck.

In the longitudinal direction, as the bridge deck follows the deformation of the beams, at the mid-span of the bridge, the deck would be subjected to a positive moment causing a compression force in the slab in the longitudinal direction. In this way, its stiffness in the longitudinal direction becomes larger, and hence, the load is distributed more in the longitudinal direction. As the load is relatively less distributed in the transverse direction of the bridge, the distribution width for the transverse moment becomes larger.

The opposite effect is obtained at the supports of the girders in a continuous bridge. At this position, the deck is subjected to a negative moment which gives a tensile force in the deck. In this way, its stiffness in the longitudinal direction becomes smaller than that at the zero-moment position and a smaller part of the load is distributed in the longitudinal direction. As the result, the distribution width for the transverse reinforcement becomes smaller.

Regarding the influence of orthotropy, a more detailed discussion can be found in [Lindelöf and Walhelm (2014)].

6.4.4 Optimal design

The recommendations in this thesis generally gave a bigger distribution width than other guidelines for the moment in the transverse direction. This results in a reduced amount of transverse reinforcement. However, the thesis only studied the distribution width for sectional forces in the transverse direction. It is therefore not shown that this necessarily means that a more economic design is achieved.

To achieve an optimal design, also the capacity in the longitudinal direction must be checked. In most cases, for the type of bridges studied, it is more efficient to carry the load on the slab mainly in the transverse direction, rather than the longitudinal direction. A bigger distribution width that reduces the amount of reinforcement in the transverse direction will then lead to a more effective design, even if it increases the reinforcement in the longitudinal direction.

However, the recommendations in this thesis can be seen as providing a bigger range of distribution widths for selection; also, smaller distribution widths can be chosen. Consequently, to make sure the most optimal design is achieved, the distribution of load in the two directions need to be iterated.

7 Conclusions

7.1 Introduction

As the last chapter of this thesis, all the important conclusions are summarized here. In the end, recommendations for further studies are given.

7.2 Conclusion remarks

This thesis studied the distribution width of a reinforced concrete bridge deck in a single span with cantilevers that are subjected to a single concentrated load. By comparing the obtained averaged sectional force of a 2D shell FE model with a 1D beam analytical calculation model, expressions for distribution widths for moments and shear forces were generated. These expressions are presented in sections 5.2.1, 5.2.2. Some of these expressions require that the analytical calculation are made with a dummy load on the cantilever. The expressions for the dummy load is presented in section 5.3.1.

Through a base model study, it was noticed that a negative moment at the support and cantilever was caused by the load in the mid-span. This load effect resulted in a mismatch of the zero-moment position between the shell FE model and the analytical model. In order to solve this problem, a dummy load in the shape of a distributed load on the cantilever was introduced into the updated analytical calculation model. Equivalent widths for both moment and shear force was determined as the ratio between the results from the analytical and shell FE models.

It was also realized that different averaging widths of the FE model gave a marginal effect on the equivalent width, so the average width was chosen to zero in the parametric study to obtain a conservative equivalent width. This resulted in reduced amount of data handling since the torsional moment is always zero in the transversal section of the applied load, i.e. when the averaging width is zero.

A fully parametric study for a number of base model variations was carried out after the base model study, and equivalent widths were obtained and processed. General expressions of distribution widths for both moment and shear force were then determined from the processed equivalent widths through graphical curve fitting.

To verify the expressions of distribution width for both moment and shear force, an envelope comparison was carried out. It was observed that the determined expressions of the distribution width is conservative and highly accurate when used together with the updated analytical calculation model with the corresponding dummy load,. The expression of the distribution width for shear force is over-conservative and inaccurate at positions close to the supports. In order to tackle the inaccuracy of the distribution width for shear force, a “cutoff” region was introduced to exclude the local effects from the FE analysis and the region where punching shear will be critical from the expression. This led to an expression that is valid piecewise in specific intervals across the span.

The thesis also discusses how the recommendations in this thesis be utilized. For the distribution width of shear force, a designer should note and understand that the choice of the “cutoff” region can affect the accuracy of the results. If a designer would choose a “cutoff” region according to another guideline it is important to

realise that asmaller “cutoff” will result in more conservative results, overestimating the shear force.

For both moment and shear analysis, a bigger distribution width is recommended by this thesis, compared to other guidelines. However, also the capacity in the perpendicular (longitudinal) direction needs to be checked. The best approach is to iterate the design for multiple distribution widths for internal forces in these two directions, so that the most optimal design can be achieved.

7.3 Recommendations for future studies

Two types of research studies can be suggested as a direct continuation of this thesis. The first is the investigation and development of a general expression for the dummy load that is valid for more variations of the geometry. In particular, it would be of interest to include also slabs with variable thickness.

The second is to develop a more accurate expression for distribution width for the shear force, valid also for the sections closer to the load position.

Research studies with different basic assumptions can also be interesting. For instance, it would be of interest to determine expressions for distribution width based on non-linear FE modeling. Furthermore, it would be interesting to study the influence on the distribution width from non-rigid supports and orthotropic element response, taking the changed stiffness due to cracking in different directions into account. Besides this, the distribution width for multiple spans bridge decks, prestressed concrete bridge slabs, and support from multiple girders would also be of high interest.

8 References

Boverket (2004): *Boverkets handbok om betongkonstruktioner BBK 04*, Boverket, Byggavdelningen, Karlskrona, Sweden.

Statens Betongkommitté (1979): *Bestämmelser för betongkonstruktioner*, BBK 79. Konstruktion, Svensk Byggtjänst, Stockholm.

Bro, B.V. (2004): *Banverkets ändringar och tillägg till Vägverkets Bro 2004*. Banverket, Borlänge, Sweden.

Statens Betongkommitté, (1968): *B7, Bestämmelser för betongkonstruktioner: Allmänna konstruktionsbestämmelser*, Stockholm, Sweden.

BSI (1997): *B.S., 8110-1:1997, Structural use of concrete–Part 1: Code of practice for design and construction*. The British Standard Institution, BSI, United Kingdom.

CEN (2004b). *EN 1992-2 Eurocode 2: Design of concrete structures - part 2: Concrete bridges. Design and detailing rules*. Brussels, Belgium: CEN European Committee for Standardization.

CEN (2004a). EN 1992-1-1 Eurocode 2: Design of concrete structures - part 1-1: General rules and rules for buildings. Brussels, Belgium: CEN European Committee for Standardization.

Chauvel, D., Thonier, H., Coin, A. and Ile, N. (2007): *Shear Resistance of slabs not provided with shear reinforcement* CEN/TC 250/SC 02 N 726.

Engström B (2011): Design and analysis of slabs and flat slabs, Report 2011-5. Division of Structural Engineering, Chalmers University of Technology, Göteborg

Fardis, M.N. (2013): *Performance-and displacement-based seismic design and assessment of concrete structures in fib Model Code 2010*. Structural Concrete, 14(3), pp.215-229.

fib (2013): *Model Code for Concrete Structures 2010*. Lausanne: International Federation for Structural Concrete (fib).

Grasser, E. and Thielen G. (1991): *Design aid for the calculation of sectional forces and deformations of reinforced concrete structures*. Deutscher Ausschuss für Stahlbeton, 240: 1-86. (in German)

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2. (https://www.nature.com/articles/s41586-020-2649-2/), 2020.

John McNamara (2020): *XlsxWriter (1.2.7)* [Software]. [2020], (https://xlsxwriter.readthedocs.io/) , 2020.

Lantsoght, E.O.L., Van Der Veen, C., De Boer, A. and Walraven, J. (2015): *Transverse load redistribution and effective shear width in reinforced concrete slabs*. Heron, 60(3), pp.145-180.

Lantsoght, E.O.L, Van Der Veen, C. and Walraven, J.C. (2013): *Shear in One-Way Slabs under Concentrated Load Close to Support*. ACI Structural Journal, 110(2).

Lubell, A.S., Bentz E., and Collins M.P. (2008): *One-Way Shear in Wide Concrete Beams with Narrow Support*, ASCE Structures Congress, Vancouver, Canada.

Lim, S. (2013): *Redistribution of force concentrations in reinforced concrete cantilever slab using 3D non-liner FE analyses* (Master's thesis). Division of Structural Engineering, Chalmers University of Technology, Göteborg

Lindelöf, A. and Walhelm, B. (2014): *Moment distribution and cumulative plastic rotation in reinforced concrete slabs subjected to concentrated forces* (Master's thesis). Division of Structural Engineering, Chalmers University of Technology, Göteborg

Microsoft Corporation. (2020): *Office 365 student*, [Software]. [2020], (https://office.microsoft.com), 2020

Pacoste, C., Plos, M. and Johansson, M. (2012): *Recommendations for finite element analysis for the design of concrete slabs*. Royal Institute of Technology, KTH, Stockholm

PTC. (2020): Mathcad (15.0). [Software]. [2020]. (https://www.mathcad.com/), 2020.

Python Software Foundation. (2018): *Python (3.7 - 64bit)* [Software]. [2020], (http://www.python.org/), 2020.

Strusoft. (2020): *FEMDesign (19.00.001).* [Software]. [2020], (https://strusoft.com/), 2020.

Strusoft. (2019): *Python API wrapper beta (1.01).* [Software]. [2020], (https://wiki.fem-design.strusoft.com/), 2020.

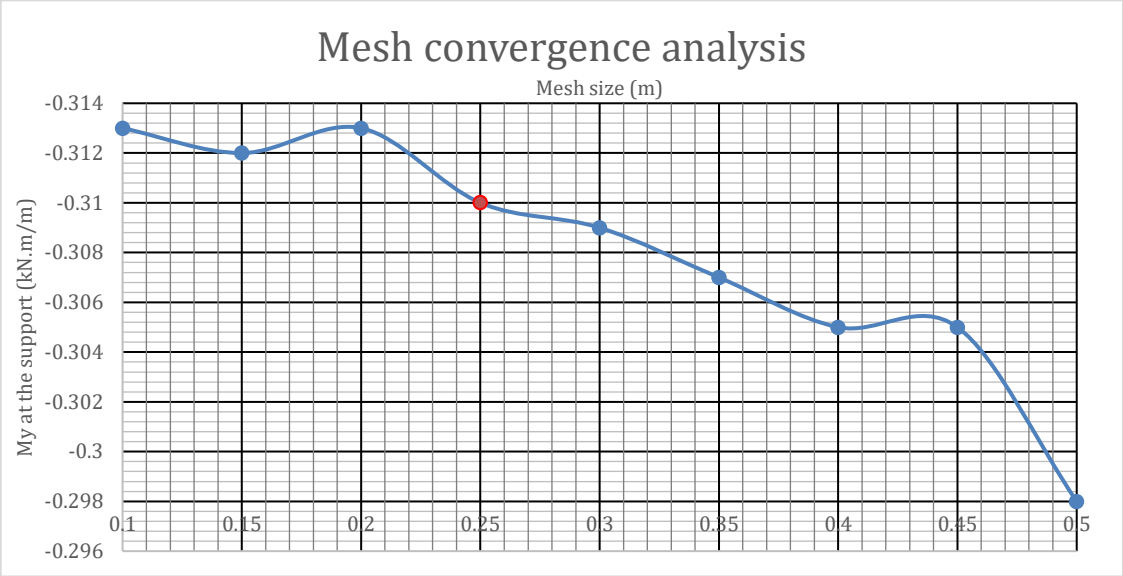
The pandas development team. (2020): *Pandas (1.01)* [Software]. [2020], Zenodo, (https://pandas.pydata.org/), 2020.

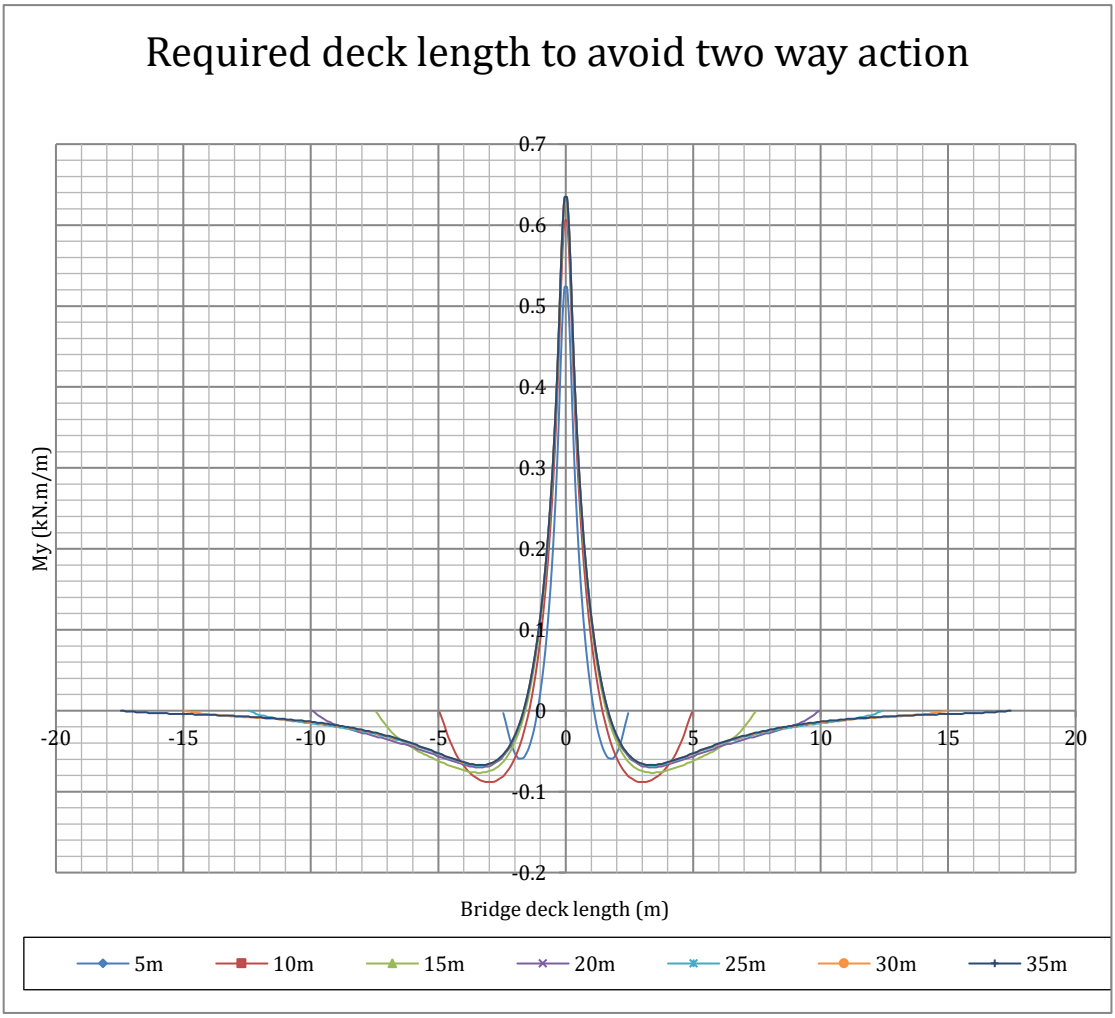
Trafikverket (2019a): *TDOK 2013:0267, Krav TRVK Bärighetsberäkning av vägbroar* (eng. Requirements for load rating of road bridges), Trafikverket, (Swedish Transport Administration), (in Swedish).

Trafikverket (2019b): *TDOK 2013:0273, Råd TRVK Bärighetsberäkning av vägbroar* (eng. Recommendations for load rating of road bridges), Trafikverket, (Swedish Transport Administration), (in Swedish).

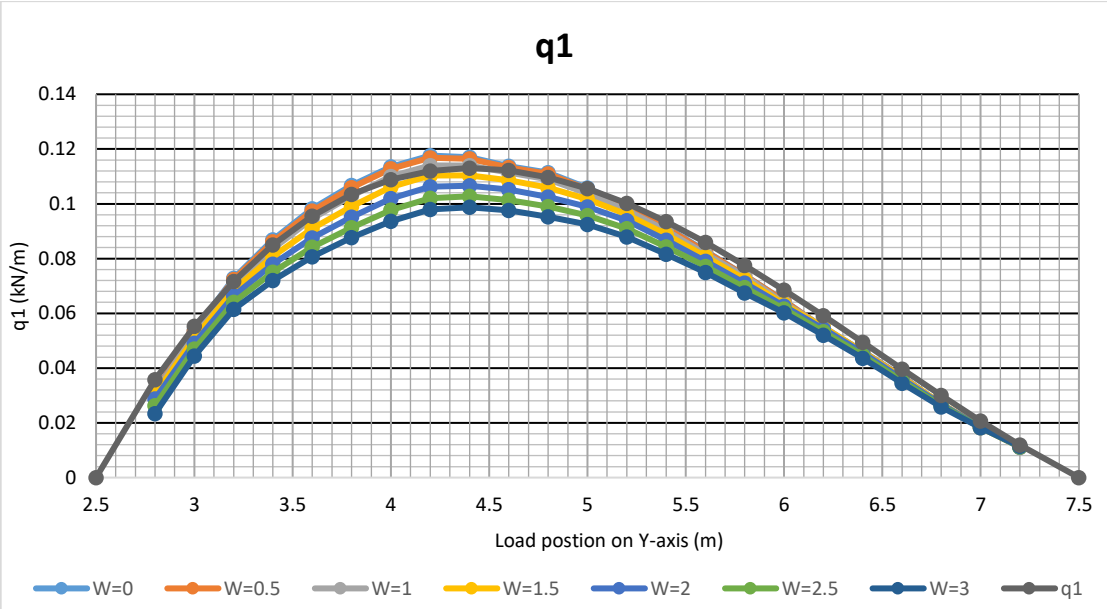
Zheng, Y., Taylor, S., Robinson, D. and Cleland, D. (2010): *Investigation of Ultimate Strength of Deck Slabs in Steel-Concrete Bridges*, ACI Structural Journal. 107: 82-91.

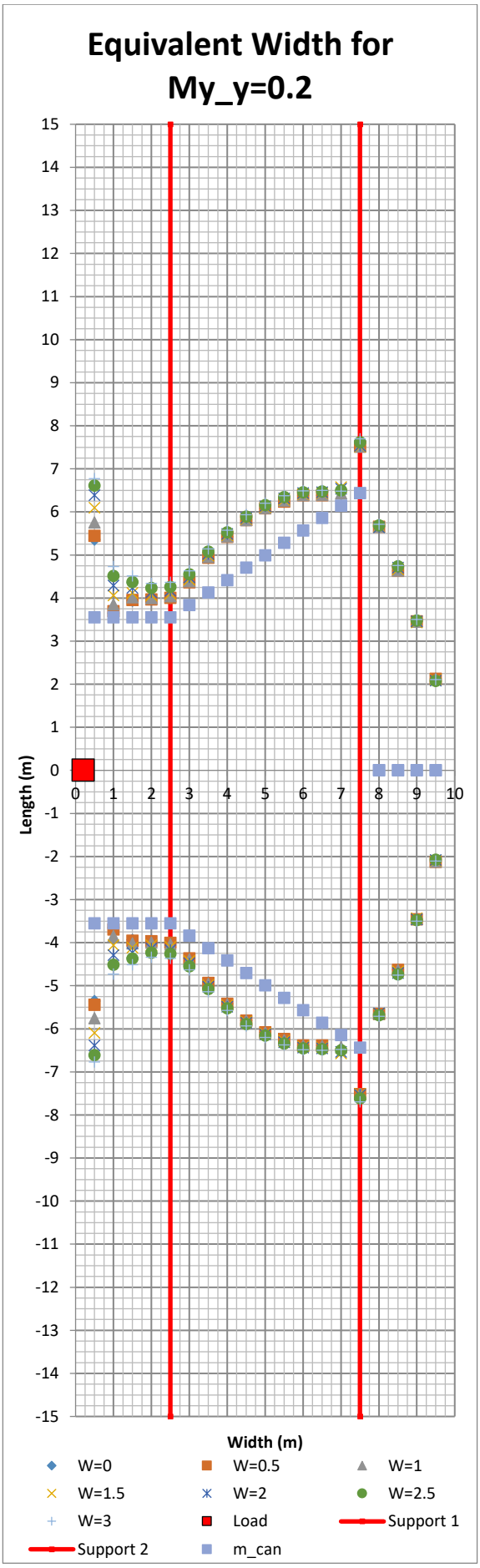
9 APPENDIX A: Convergence study

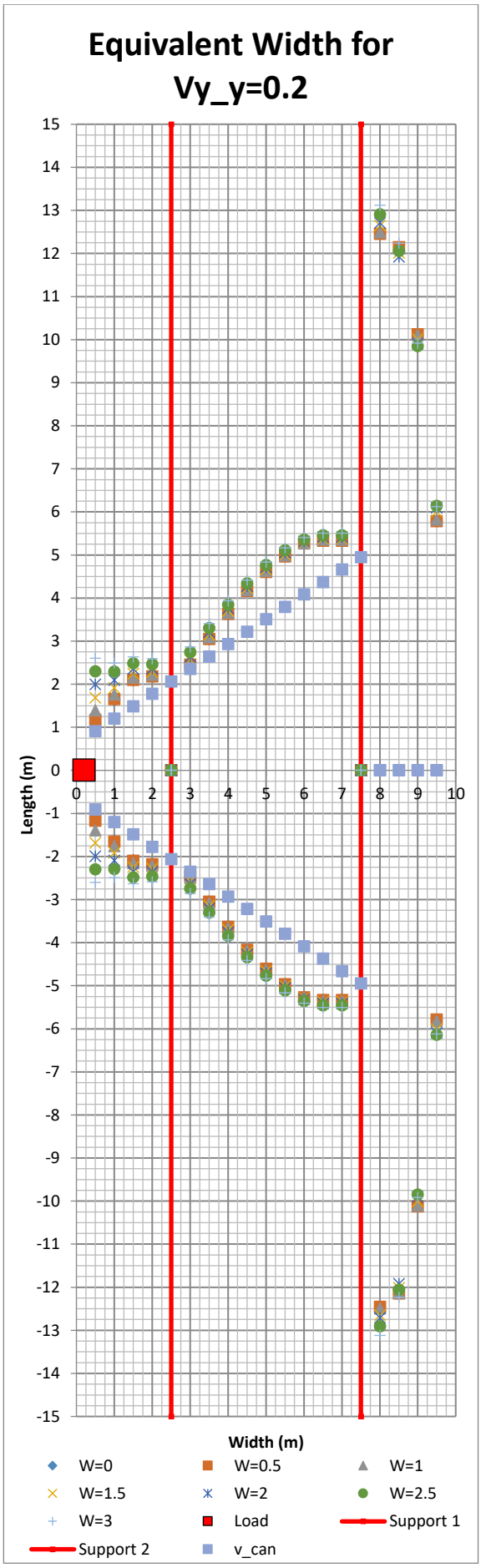


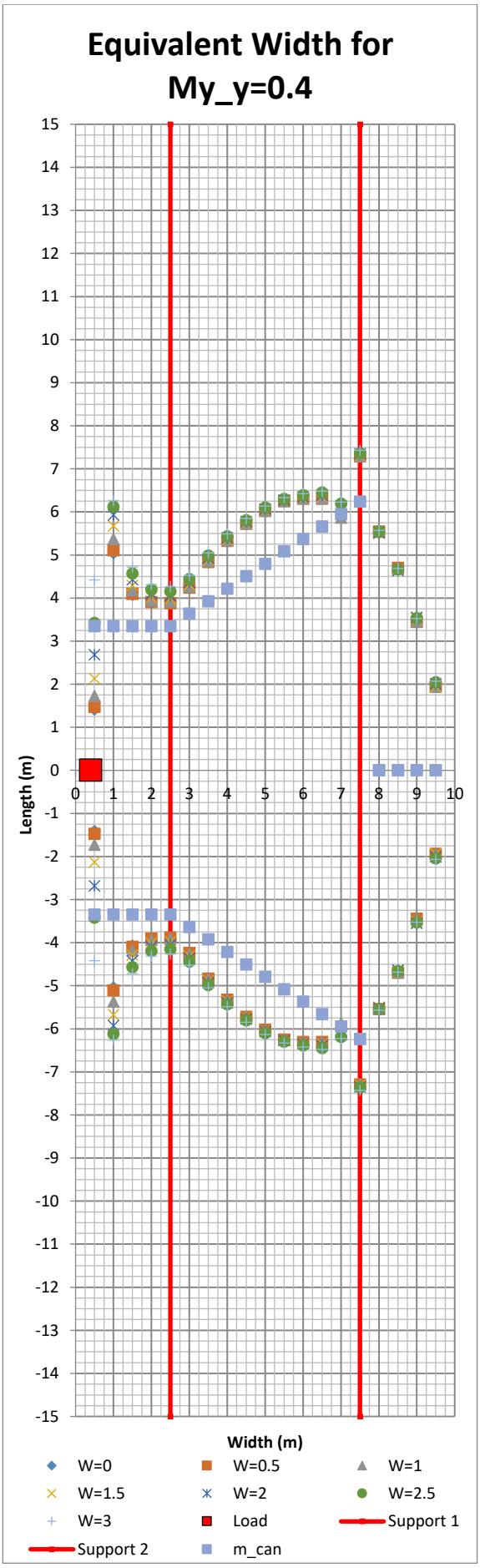


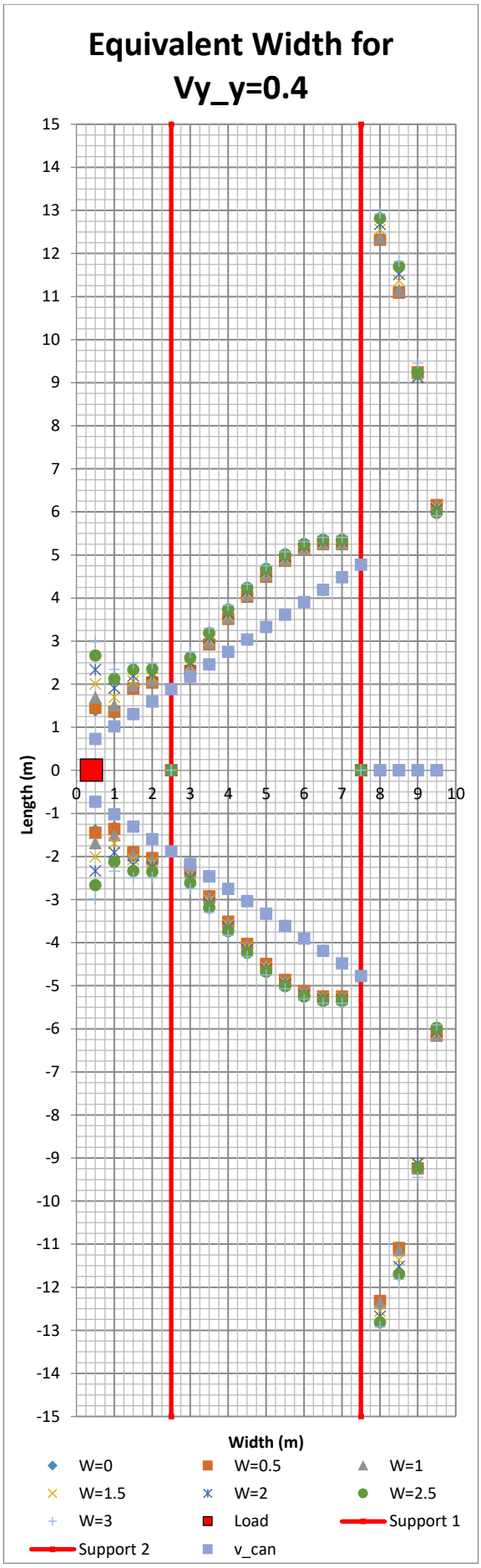
10 APPENDIX B: Base model analysis Results

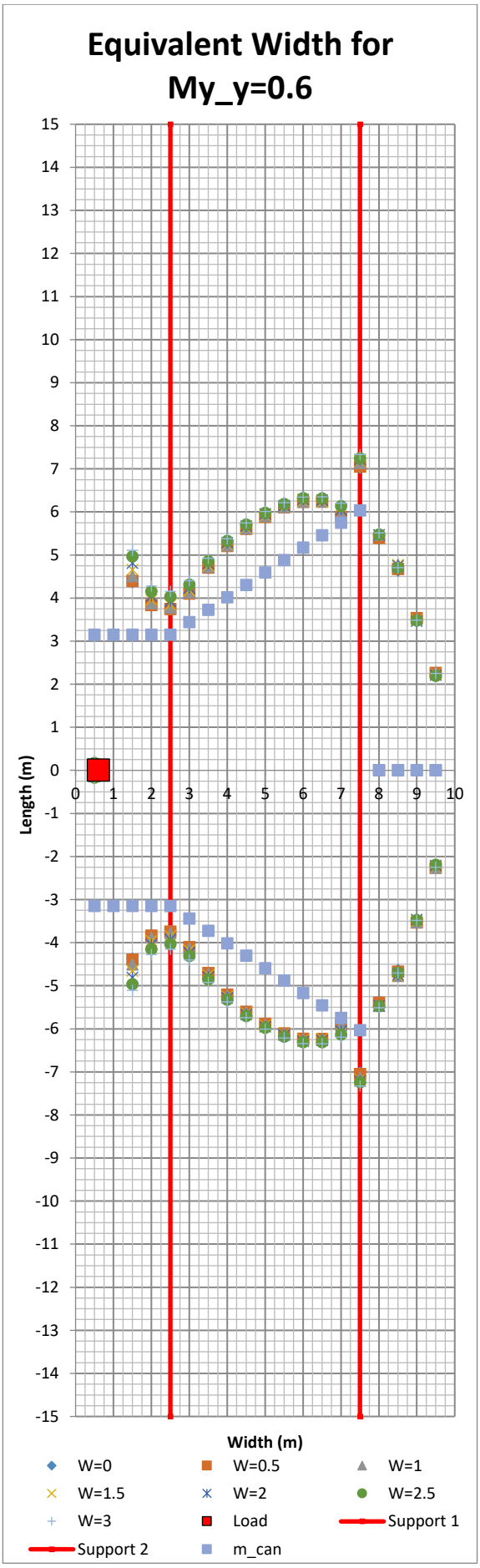


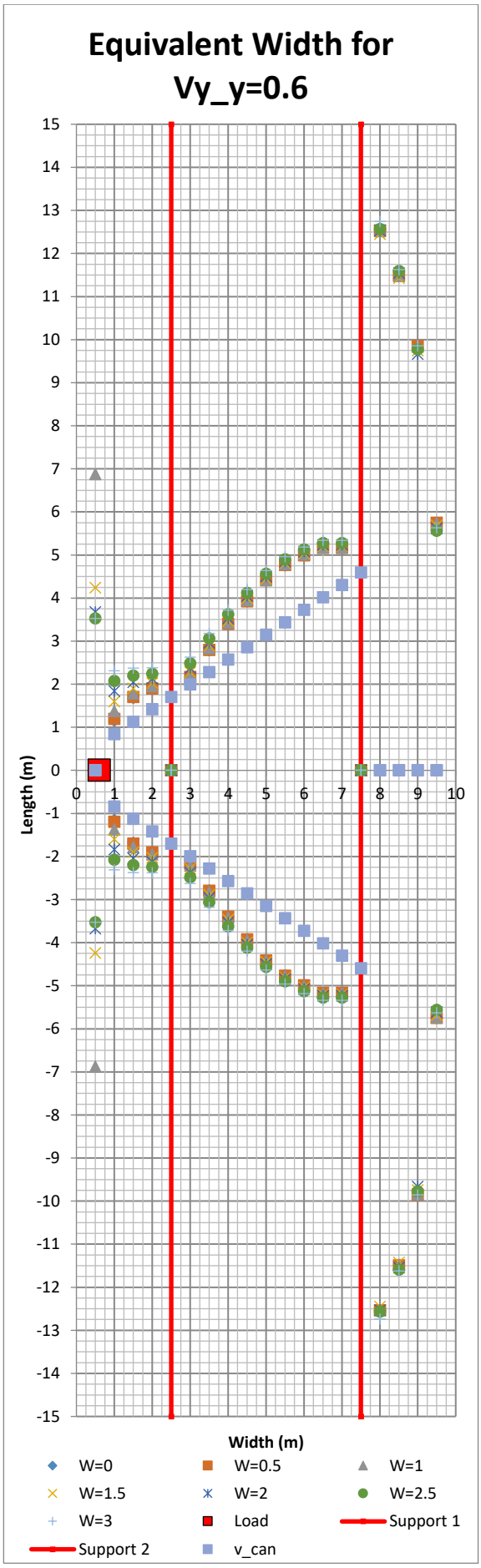

Equivalent Width for My_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

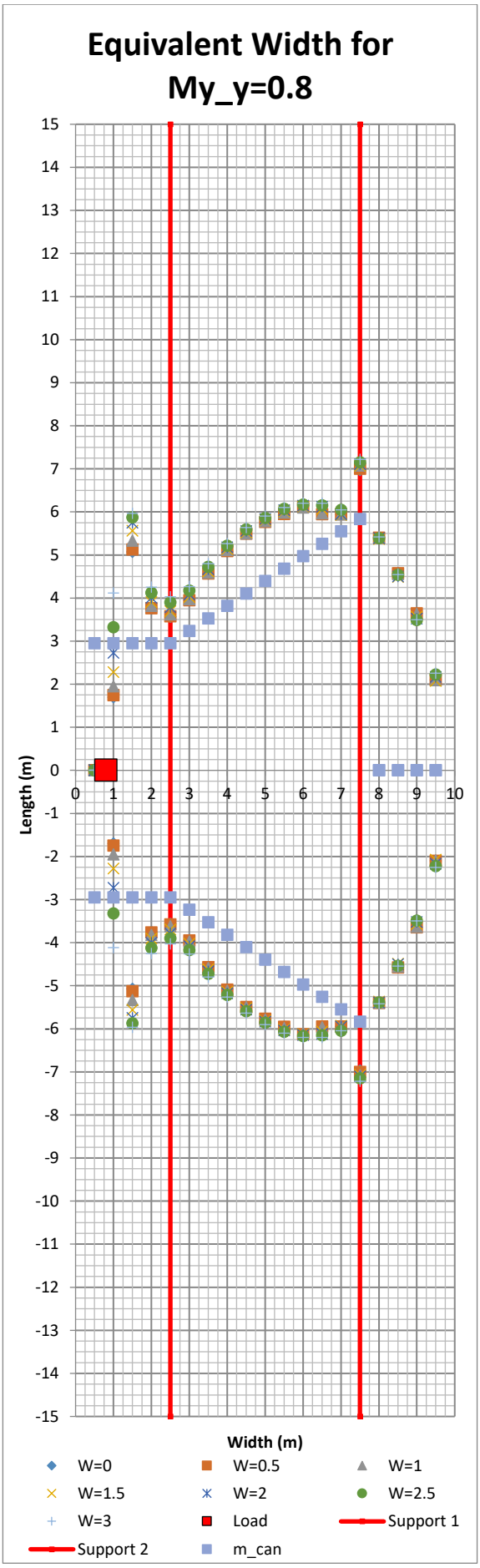

Equivalent Width for Vy_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

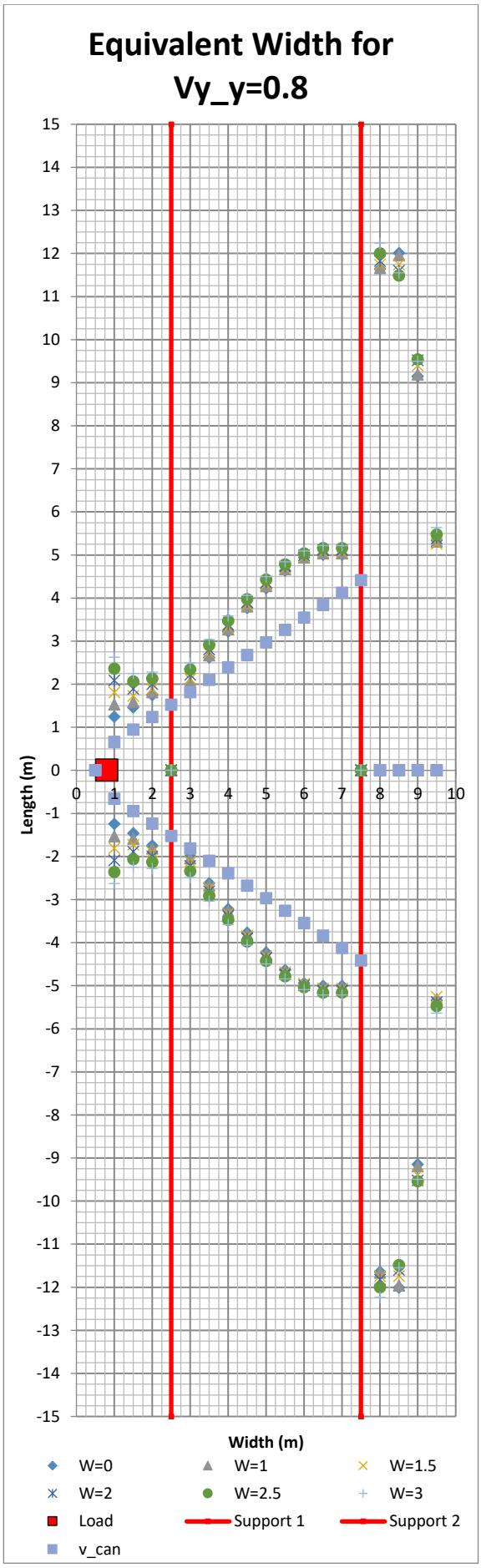

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

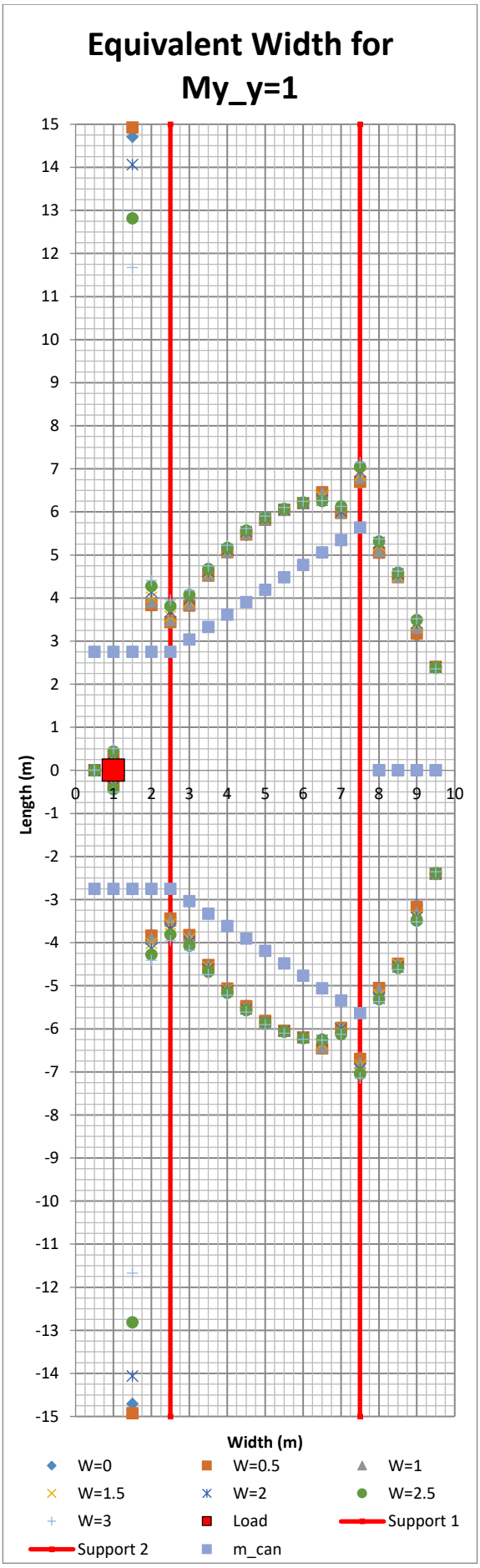

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

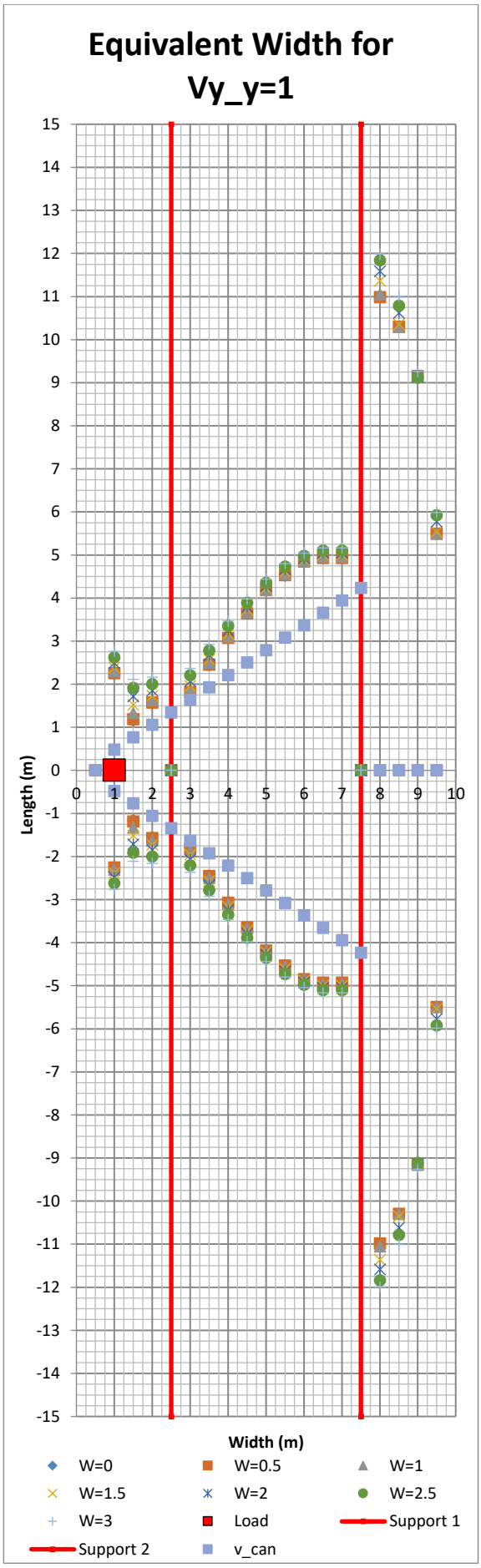

Equivalent Width for My_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

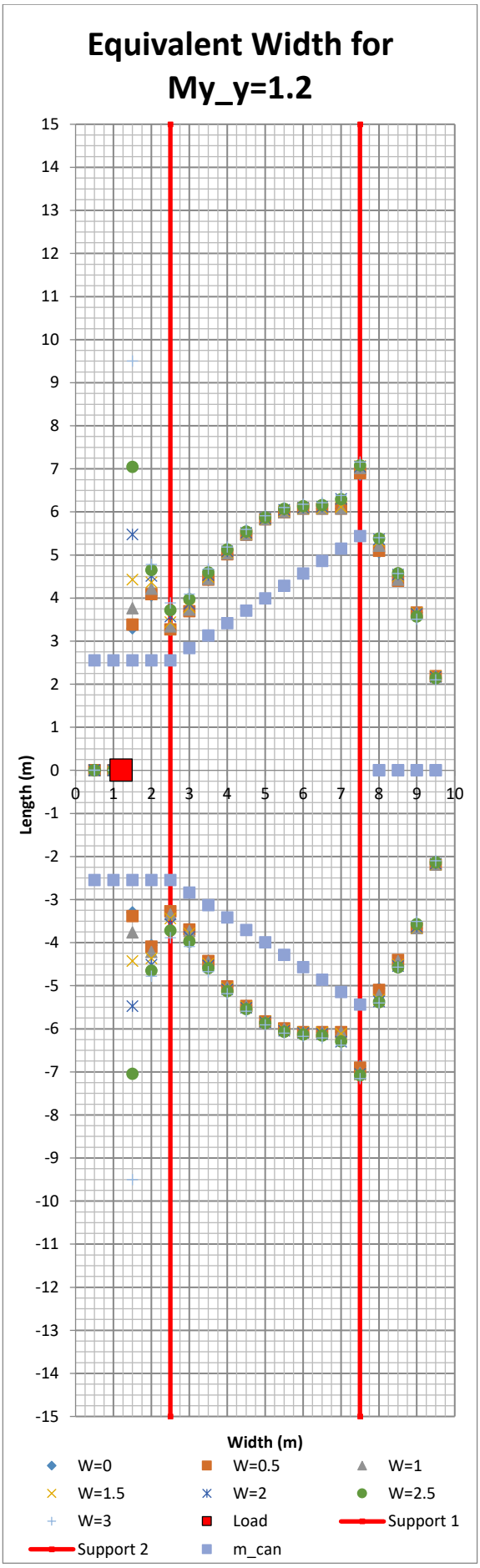

Equivalent Width for Vy_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

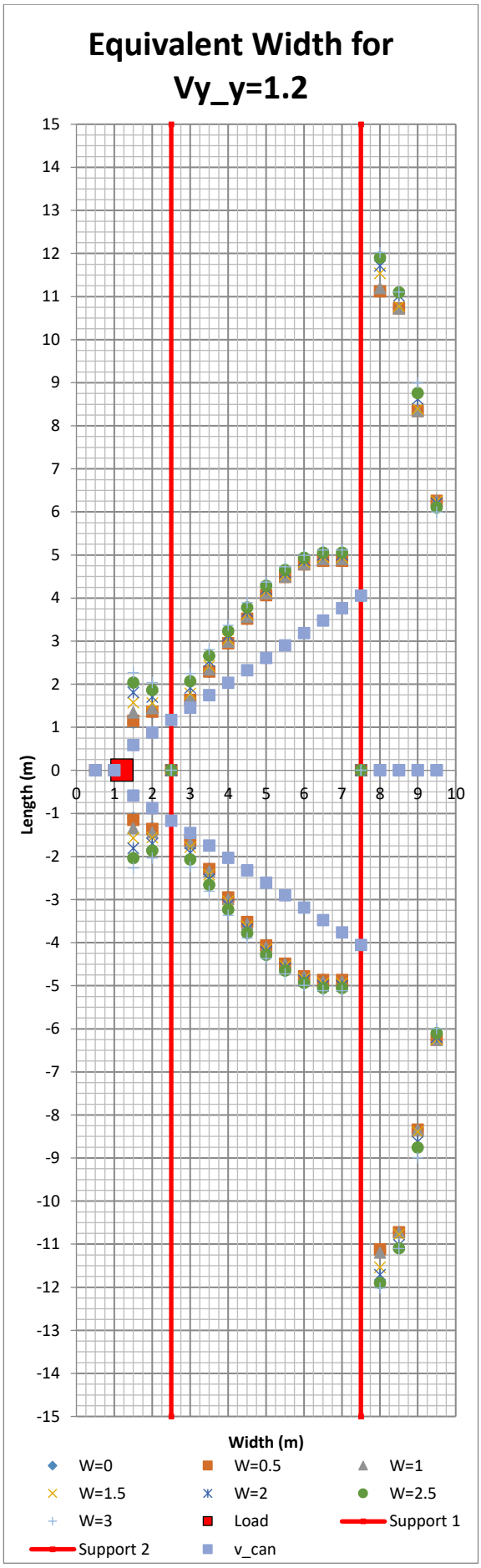

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

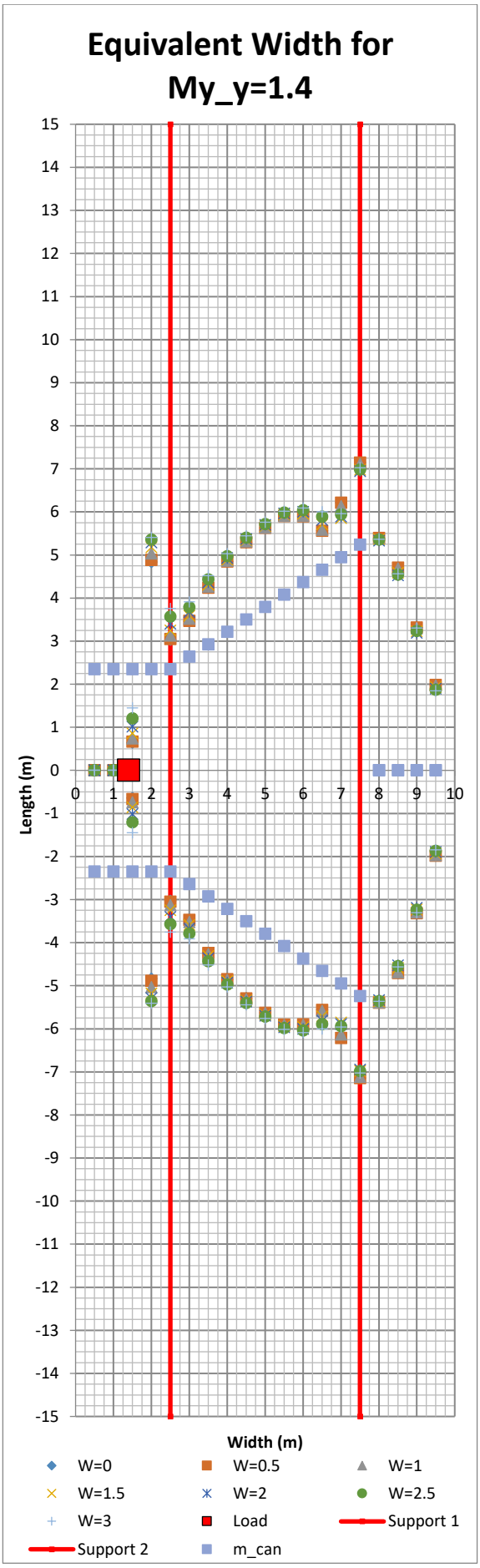

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

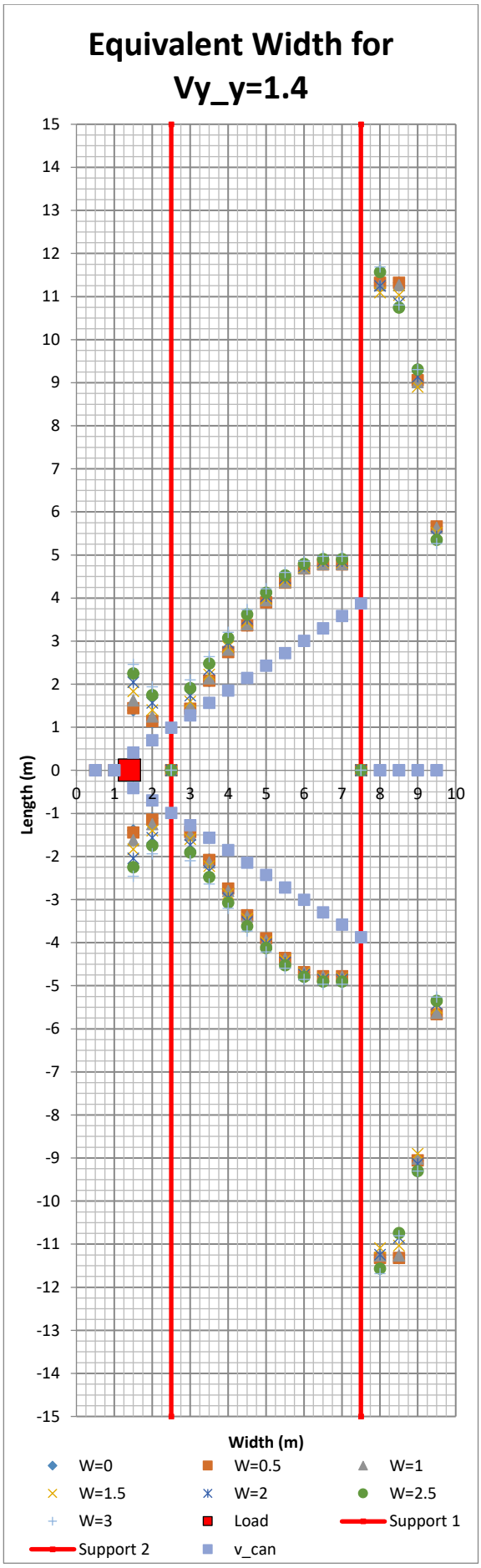

Equivalent Width for My_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

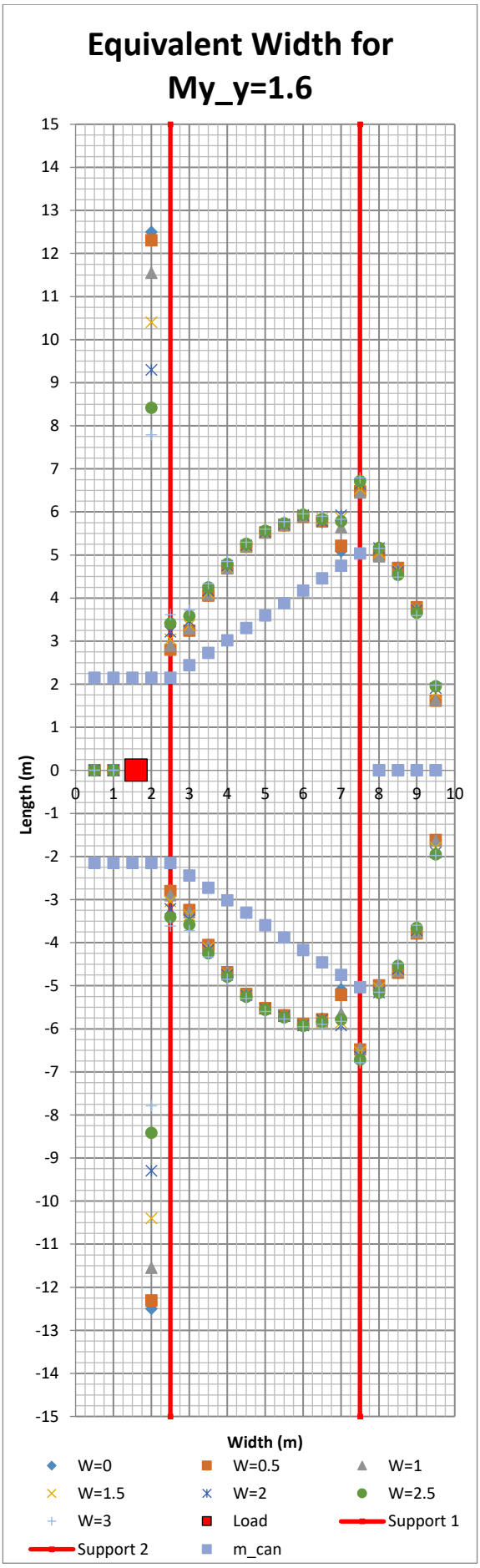

Equivalent Width for Vy_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

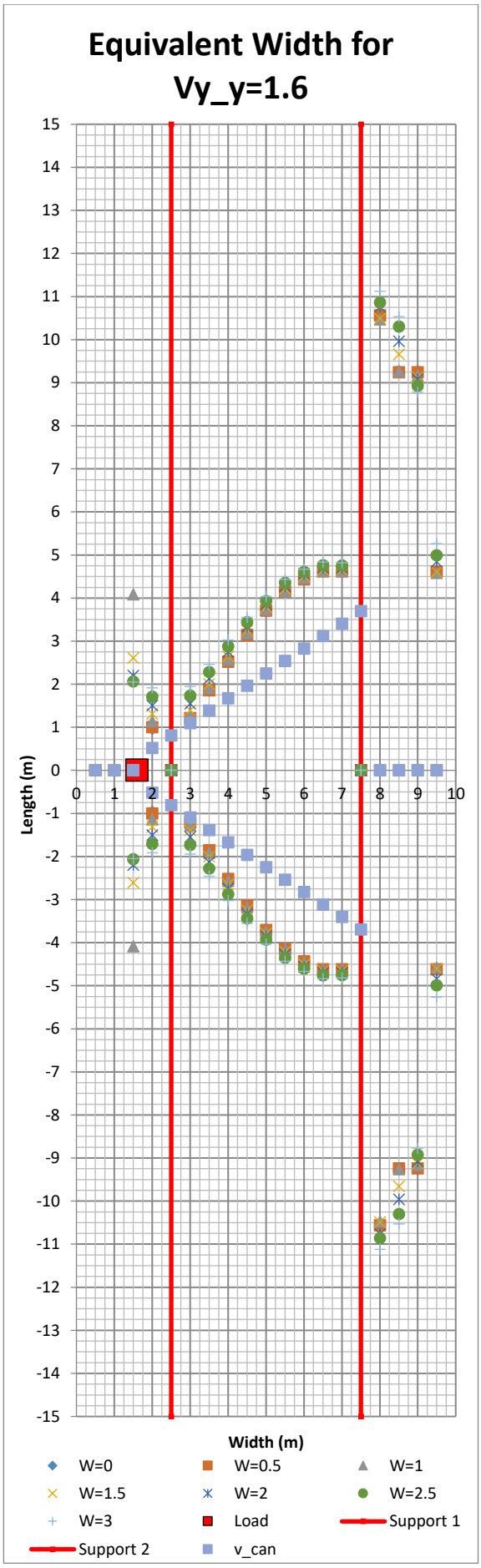

Equivalent Width for My_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

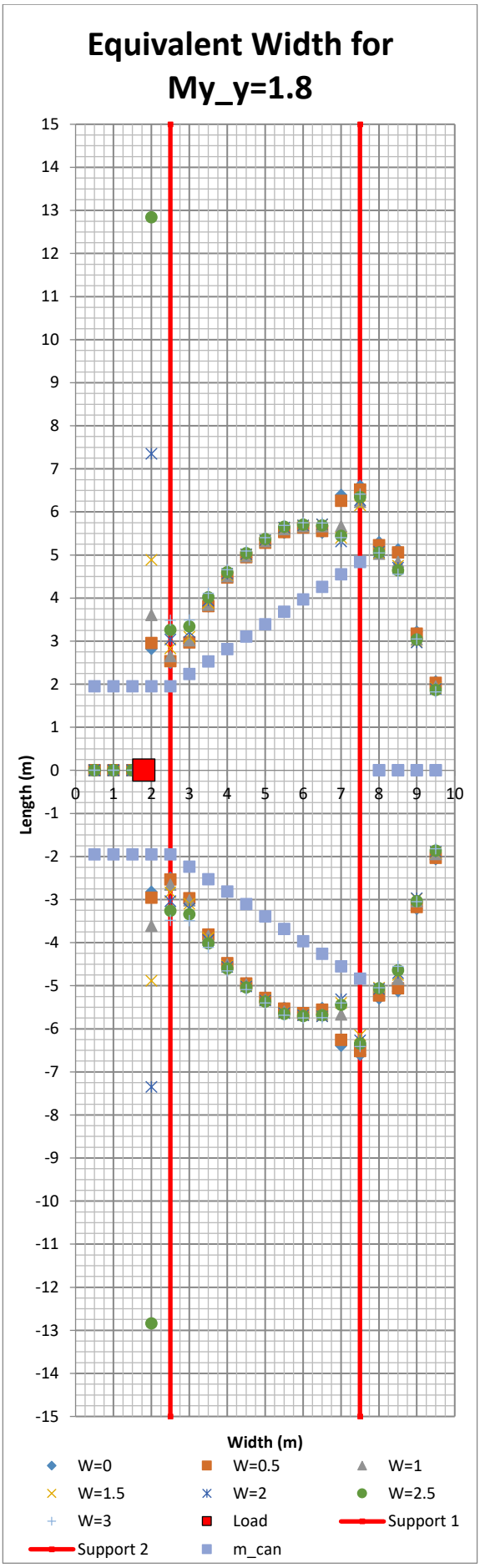

Equivalent Width for Vy_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

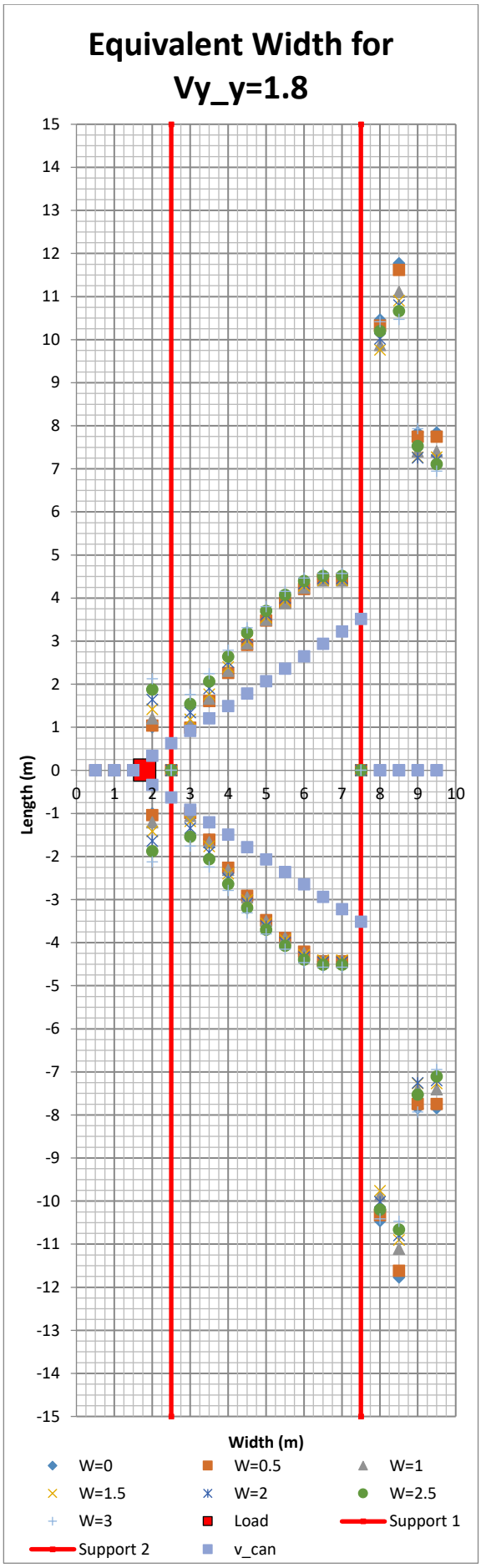

Equivalent Width for My_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

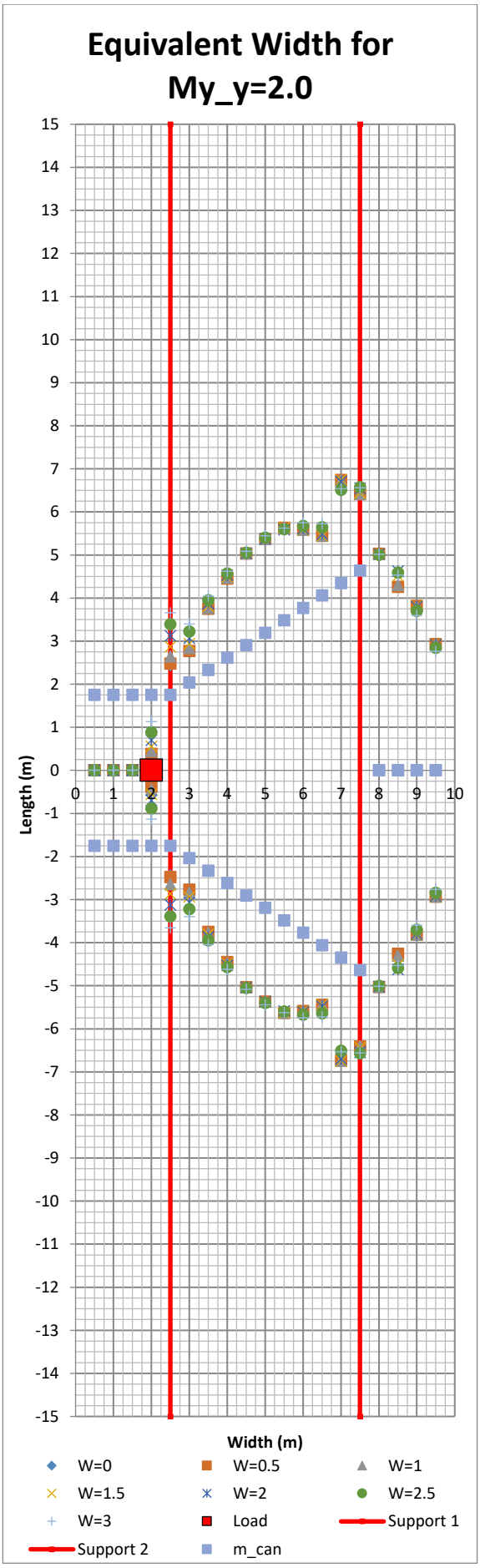

Equivalent Width for Vy_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

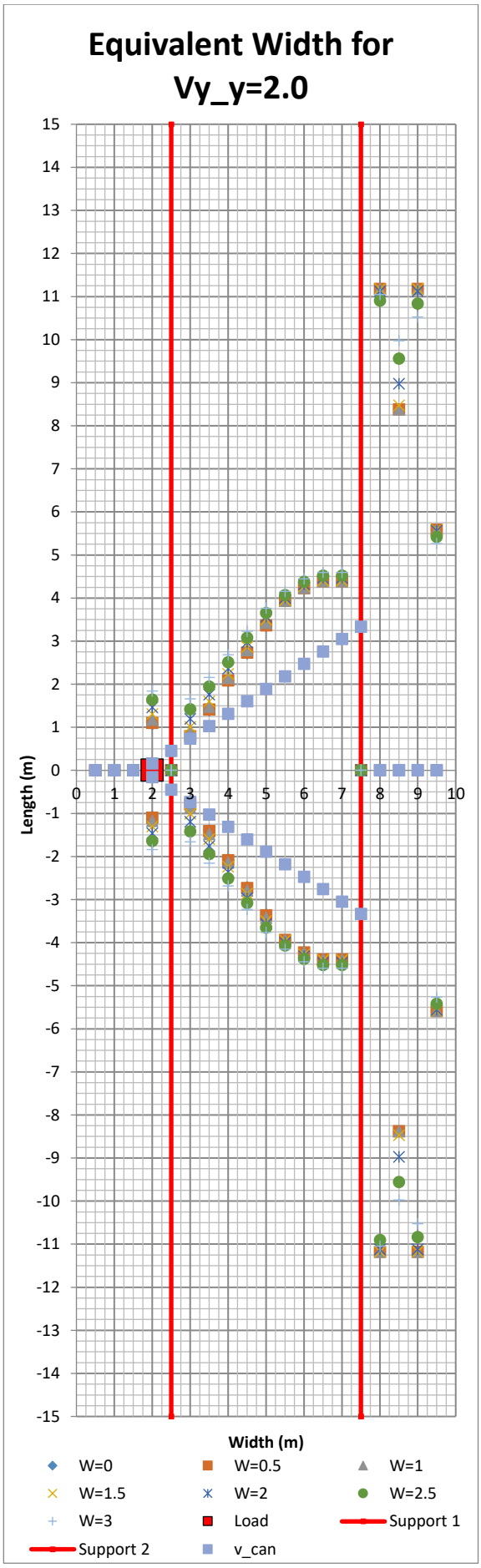

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

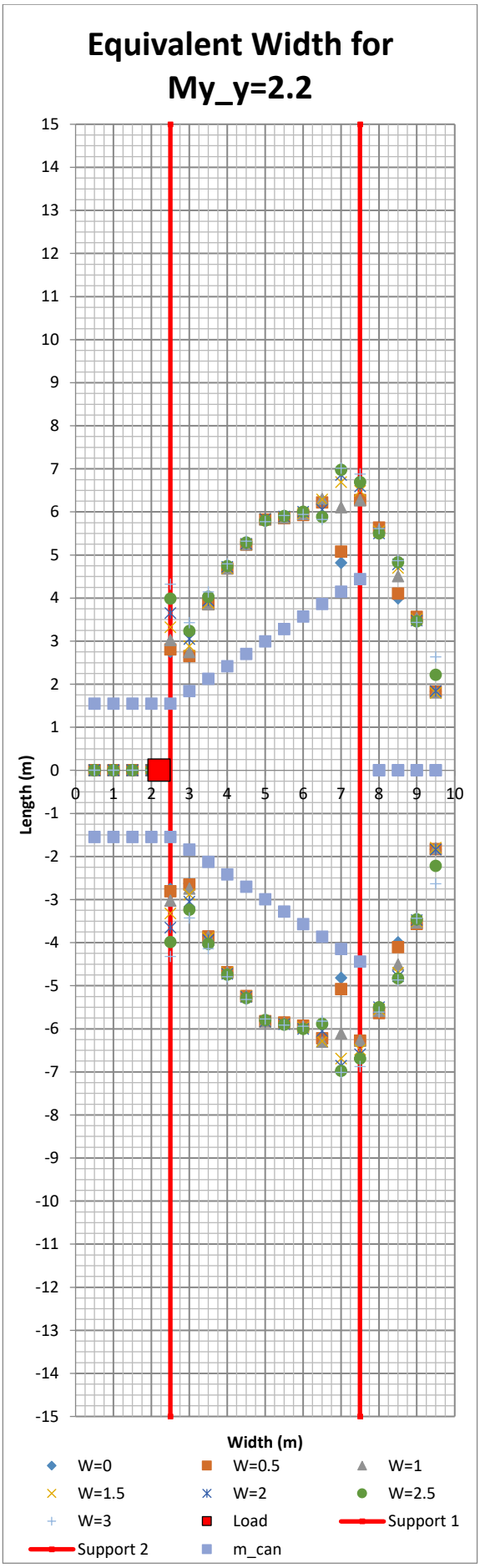

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

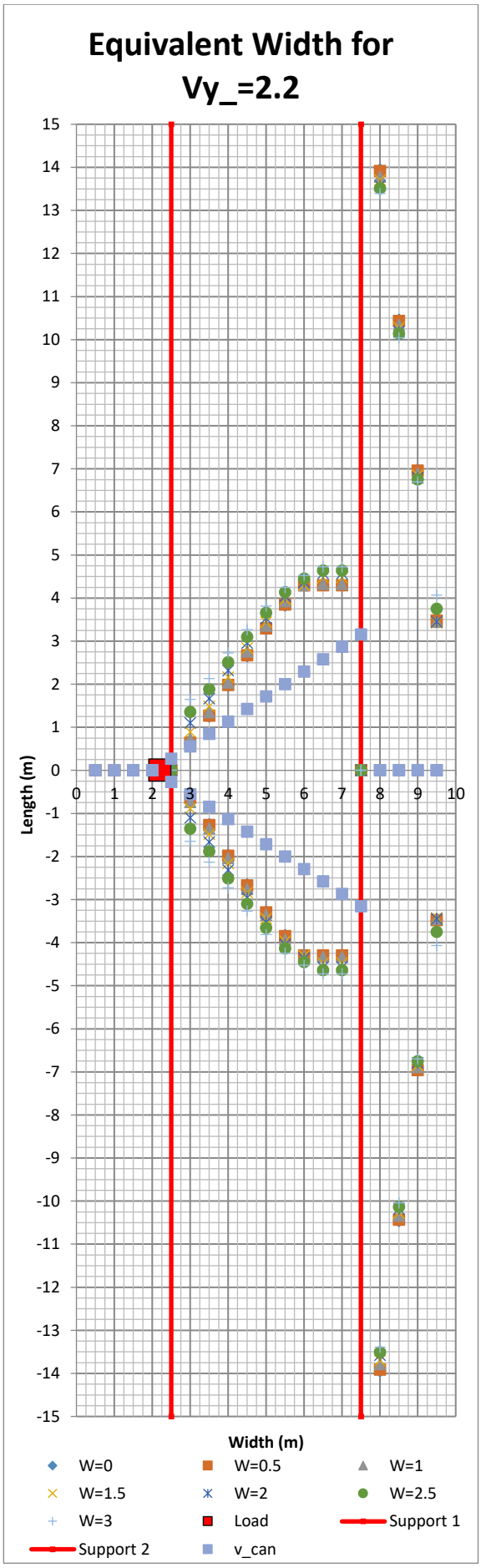

Equivalent Width for My_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

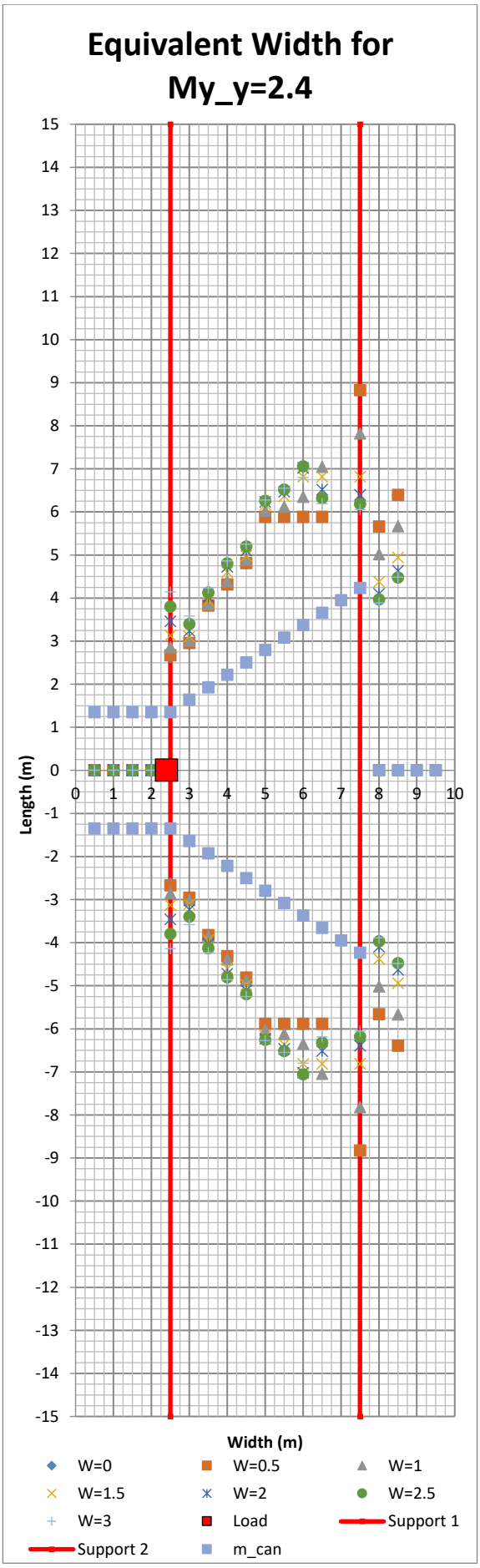

Equivalent Width for Vy_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

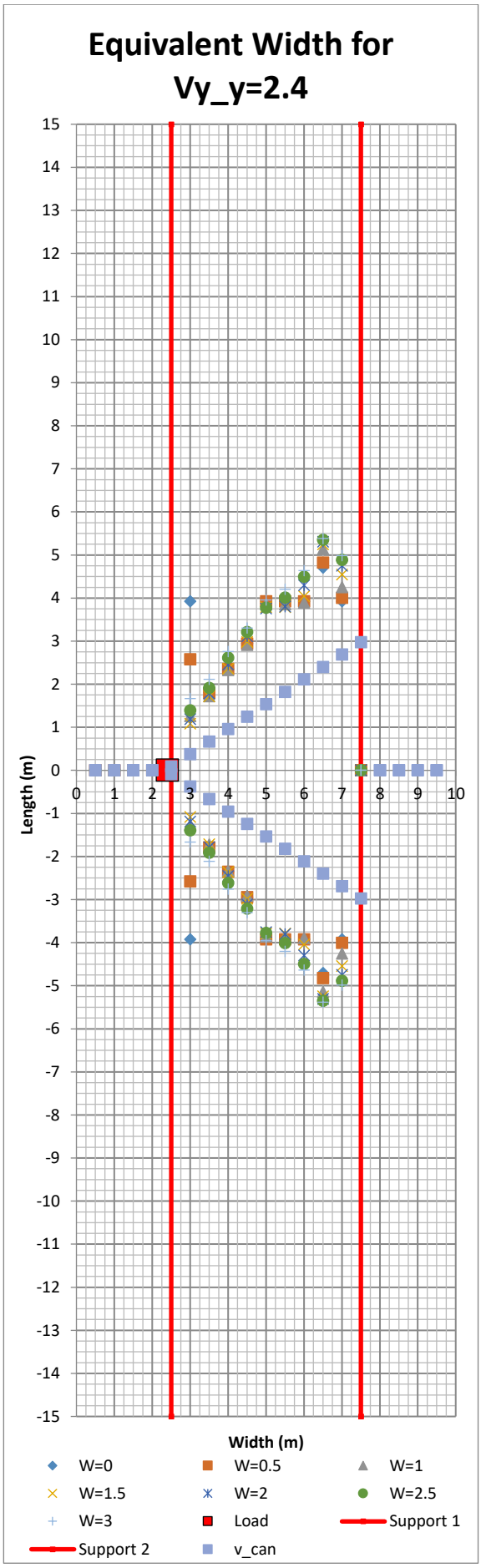

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

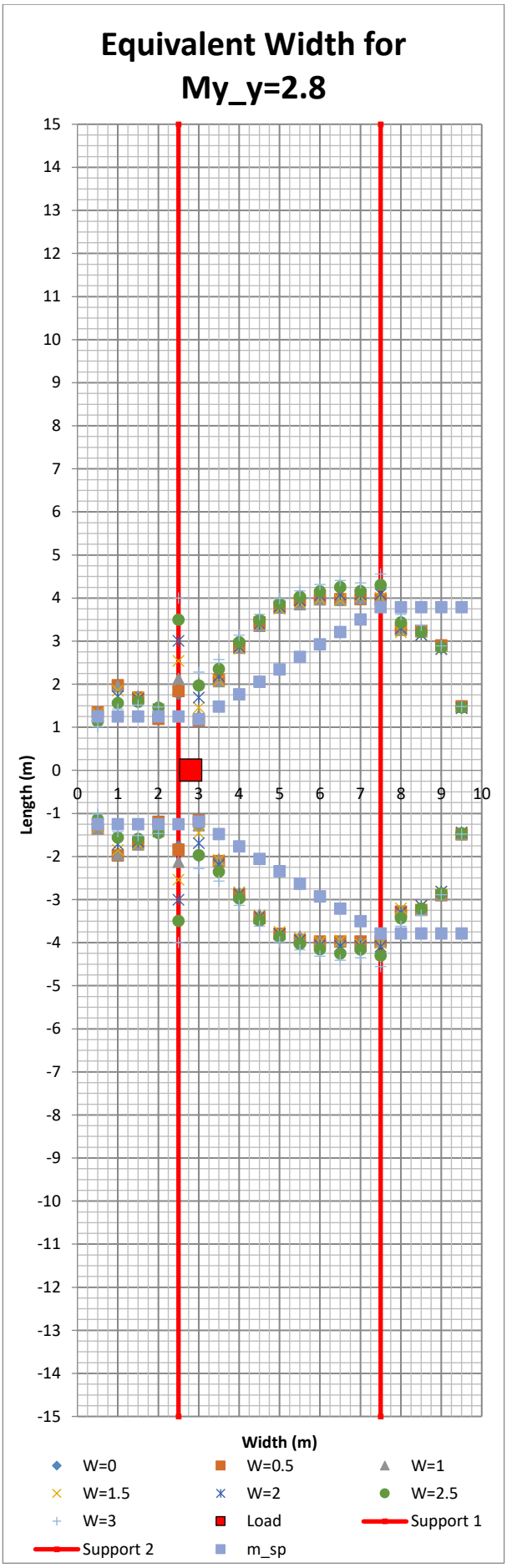

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

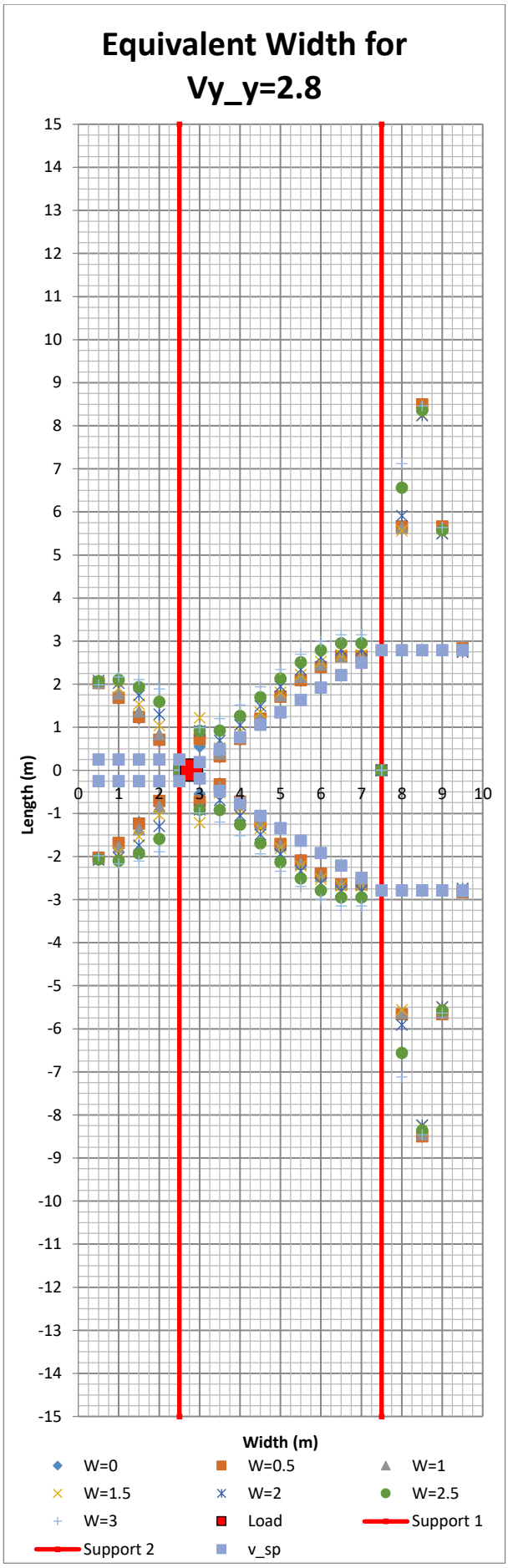

Equivalent Width for My_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

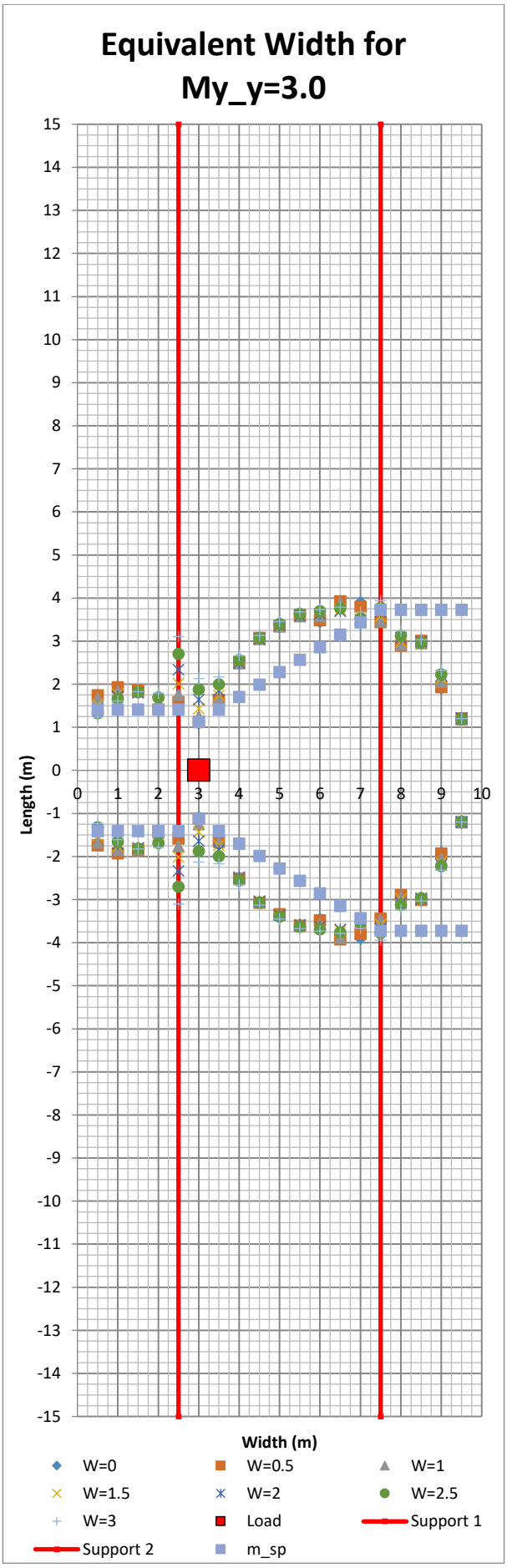

Equivalent Width for Vy_=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

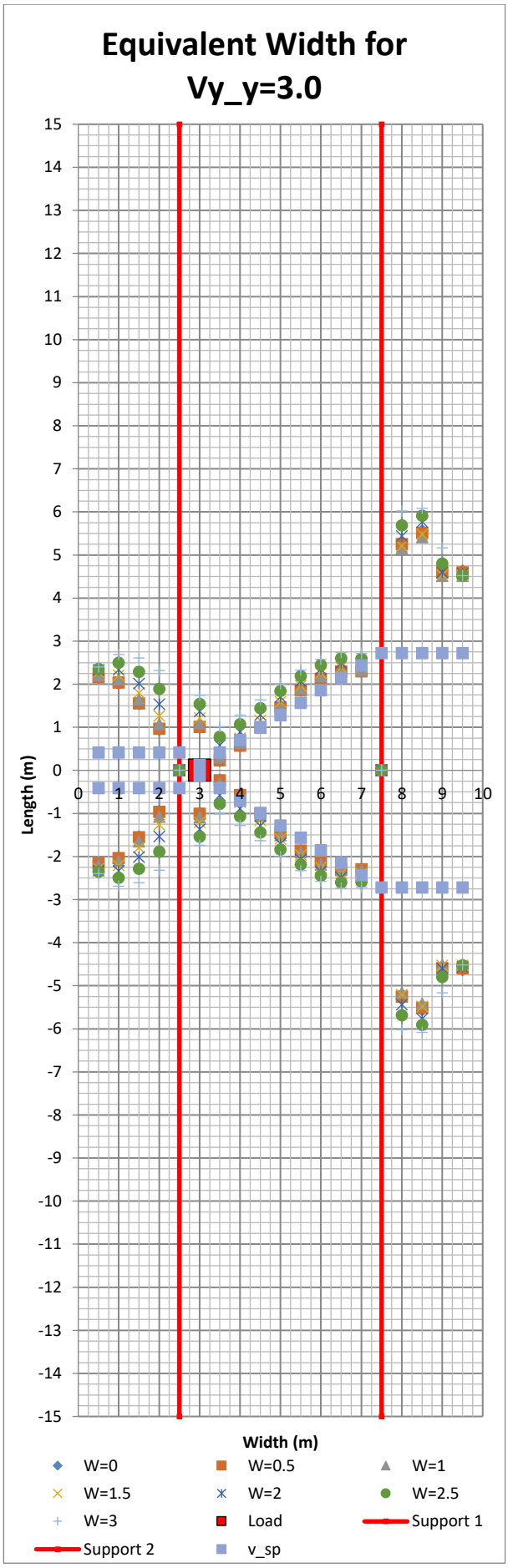

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

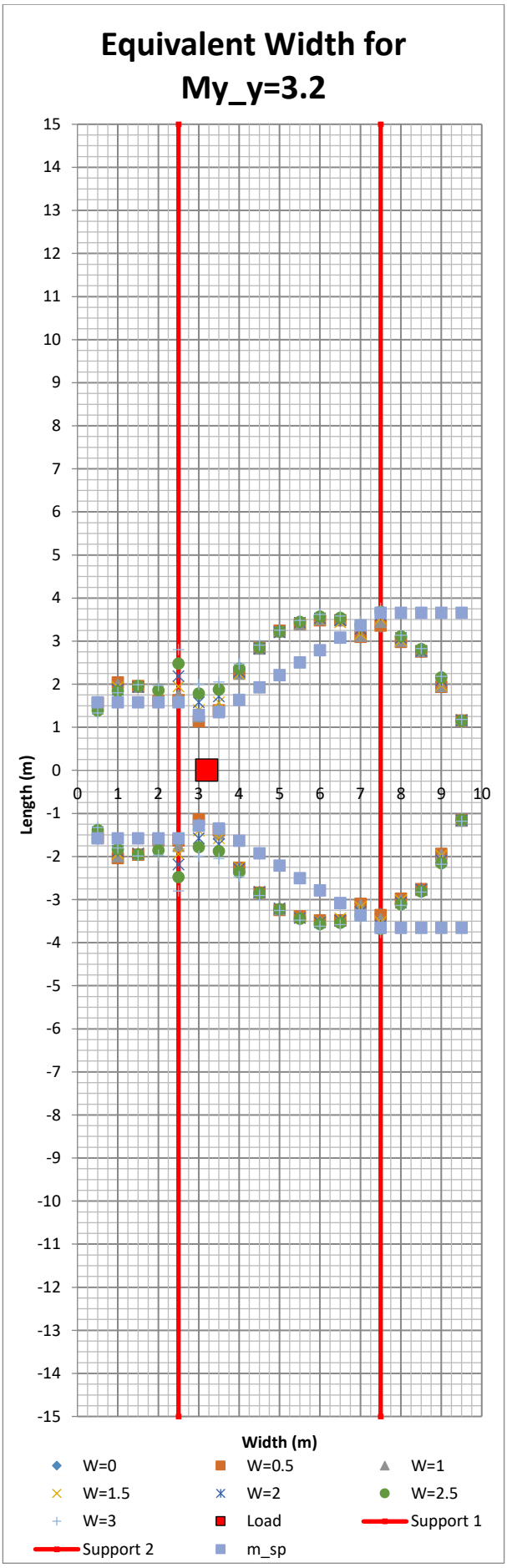

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

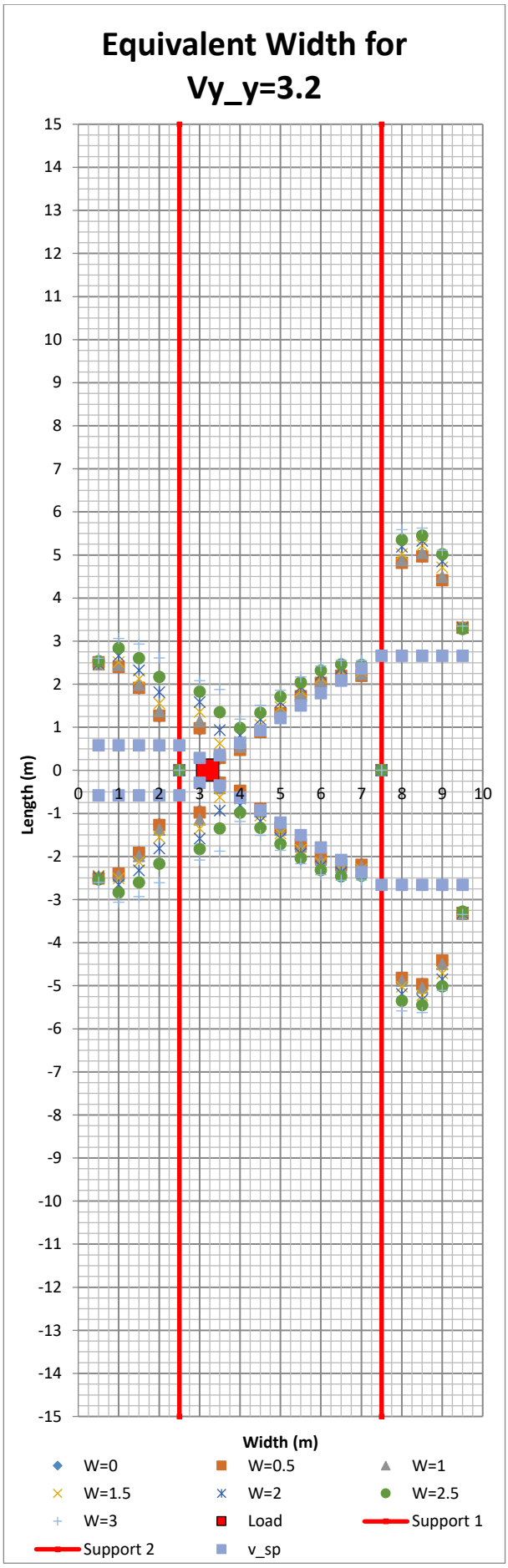

Equivalent Width for My_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

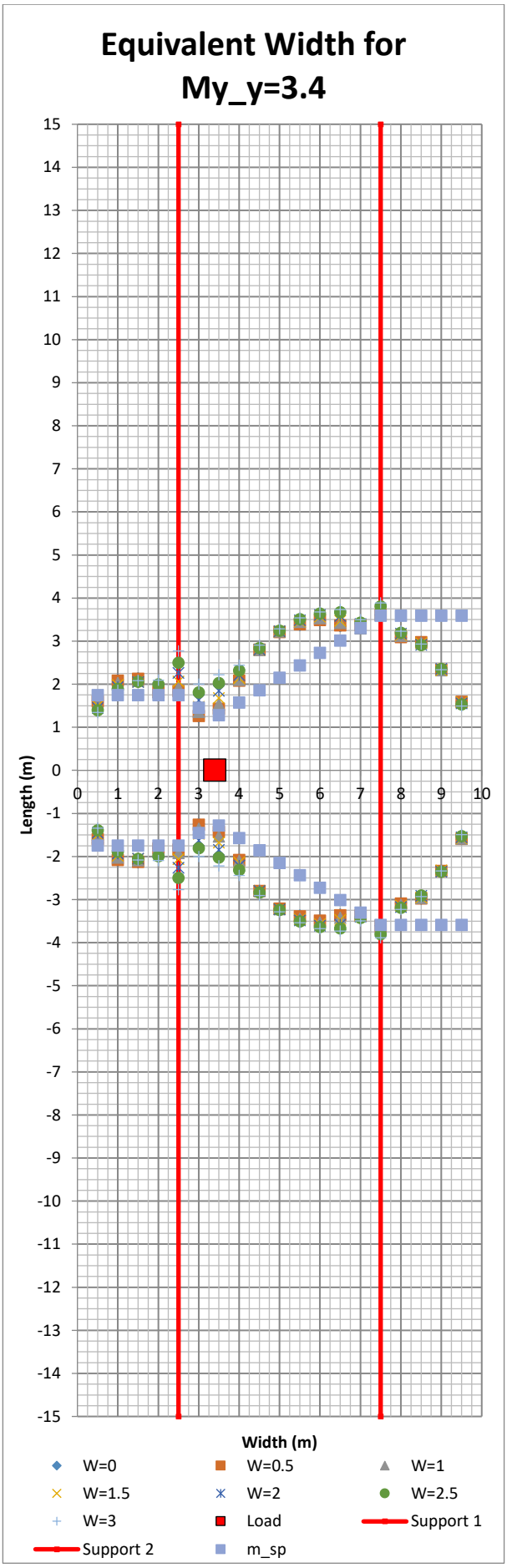

Equivalent Width for Vy_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

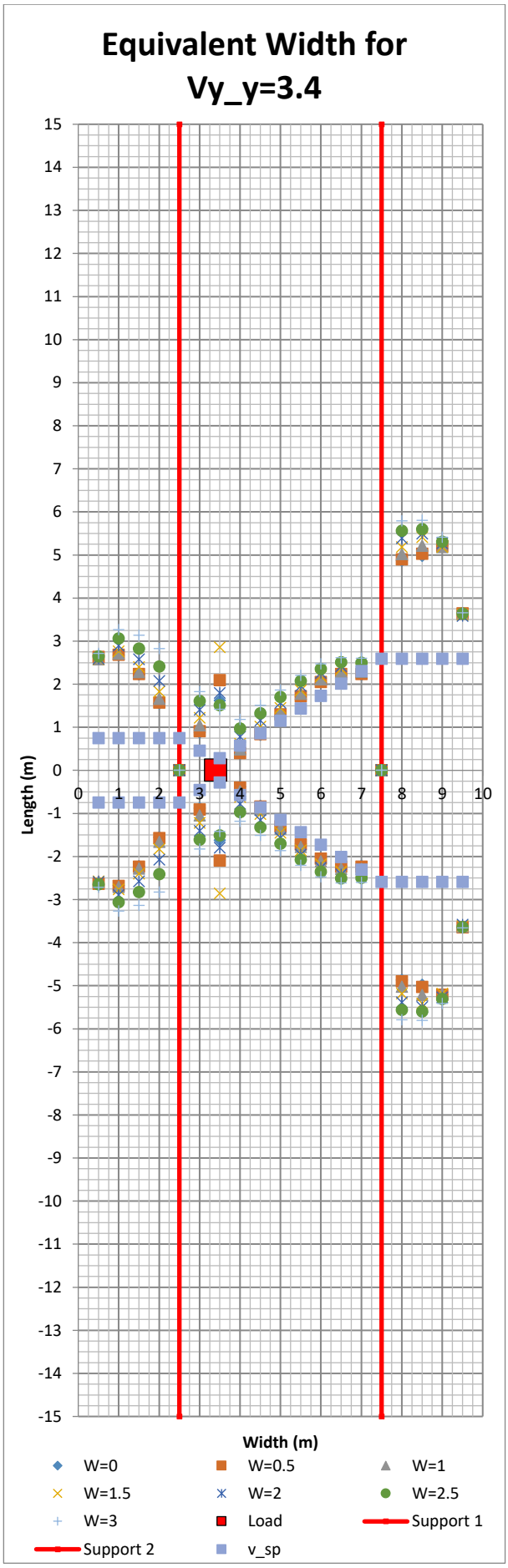

Equivalent Width for My_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

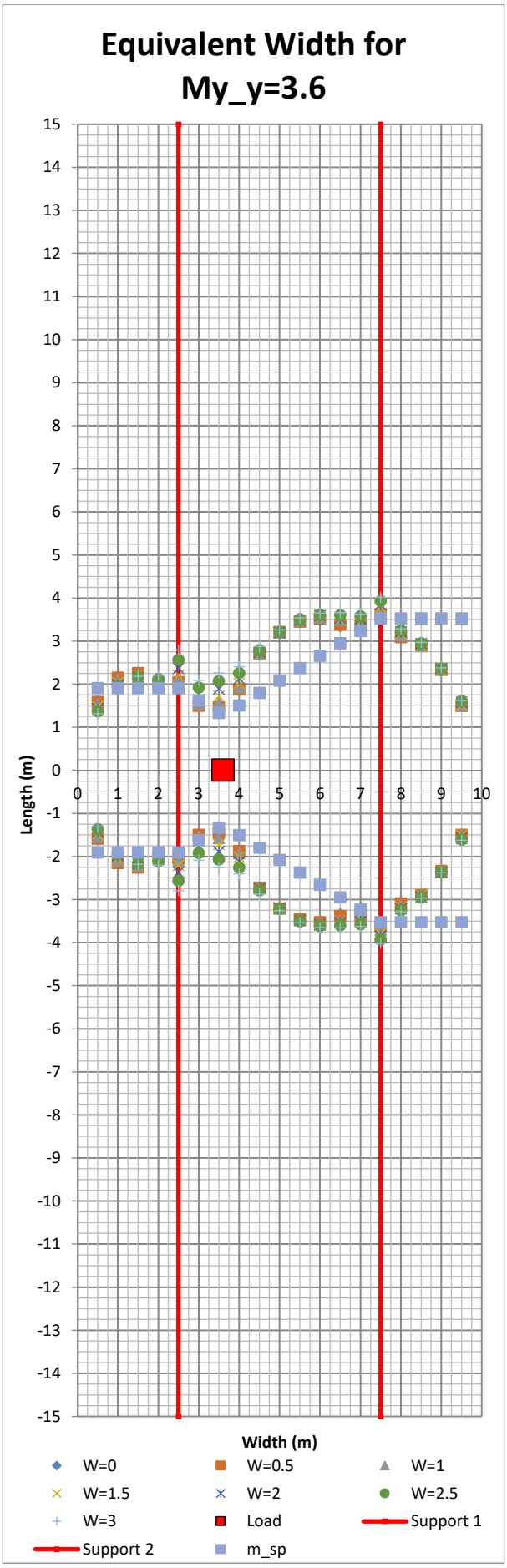

Equivalent Width for Vy_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

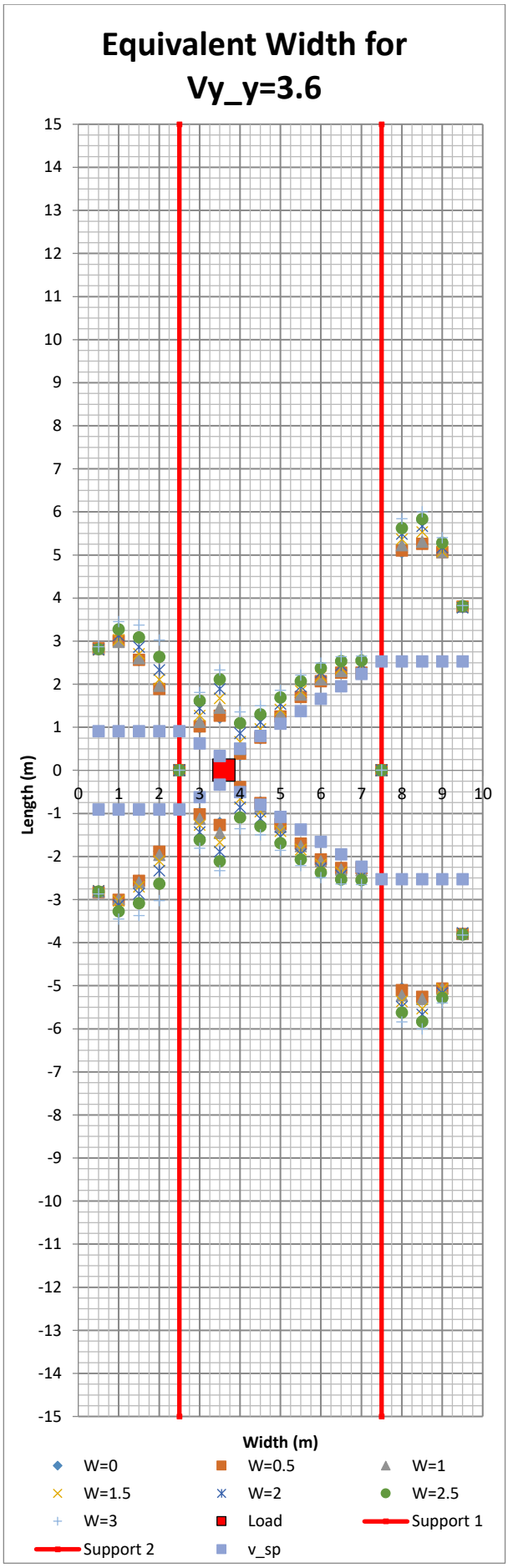

Equivalent Width for My_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

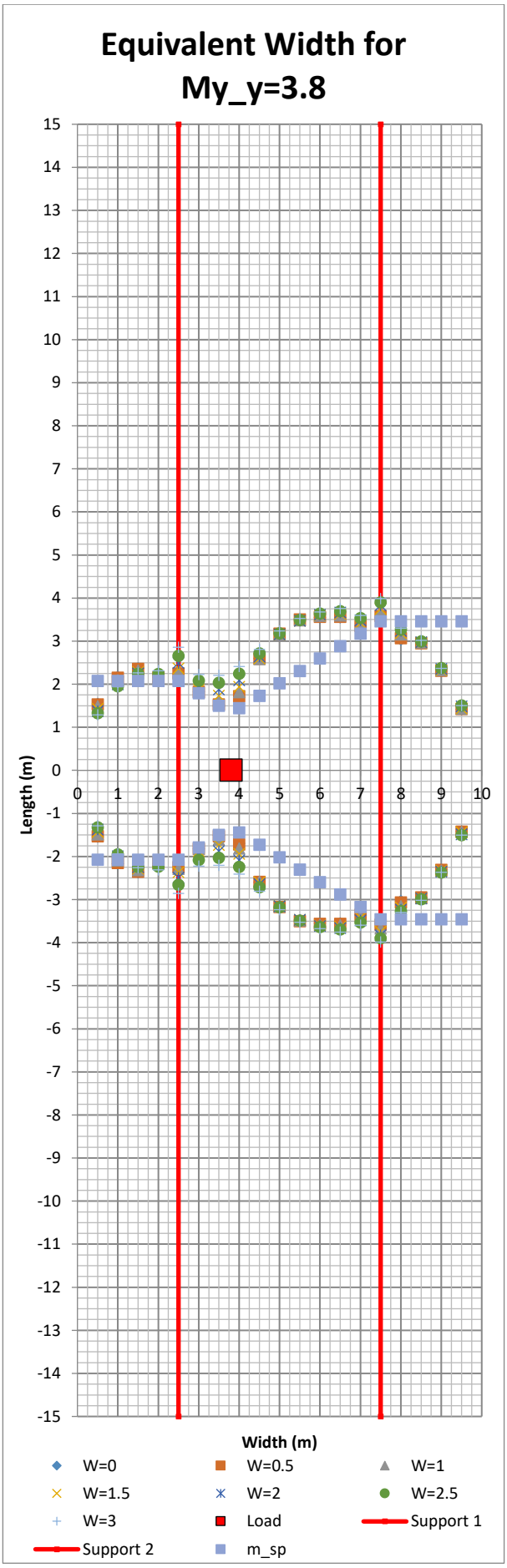

Equivalent Width for Vy_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

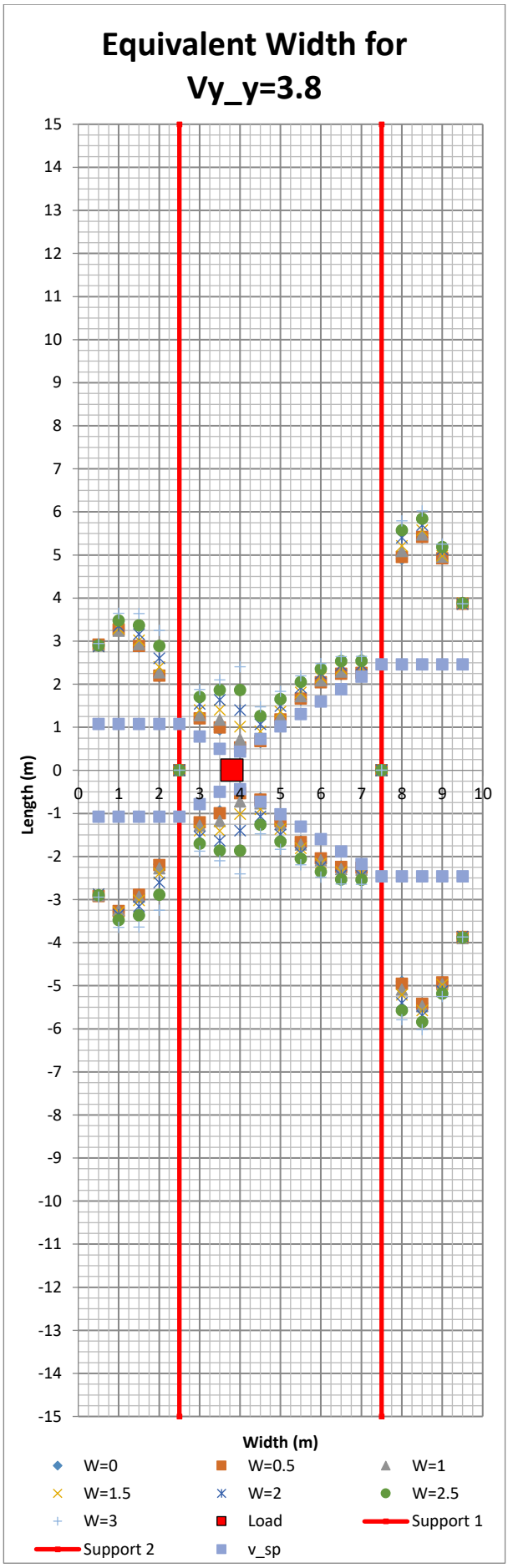

Equivalent Width for My_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

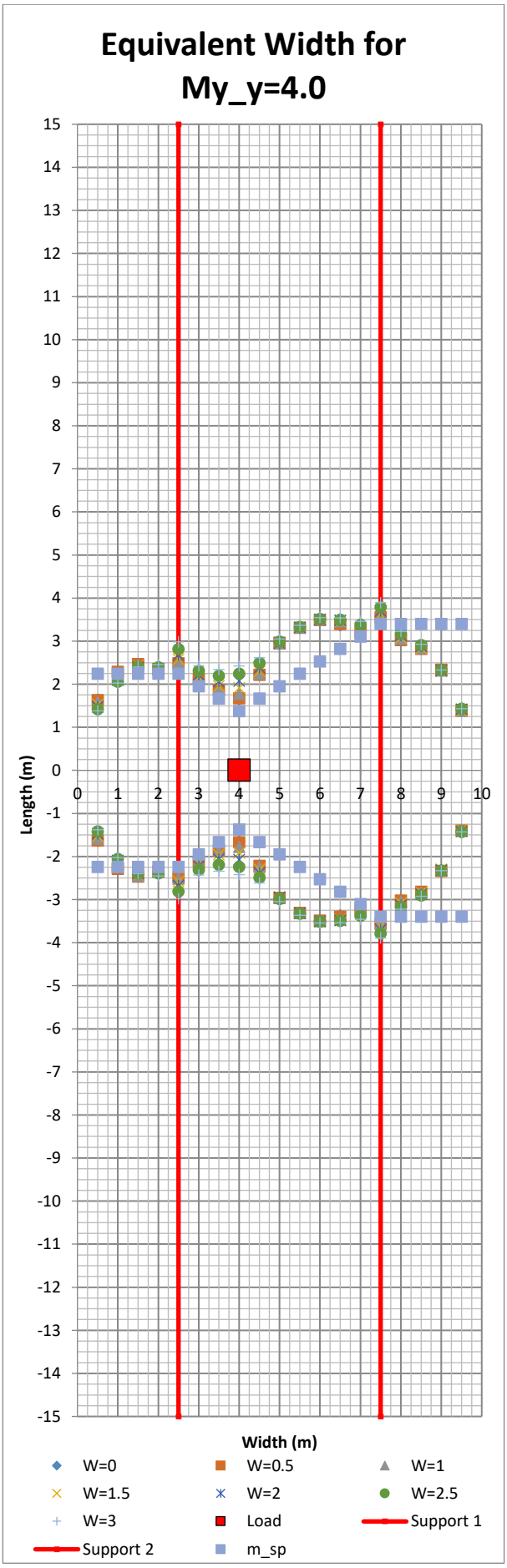

Equivalent Width for Vy_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

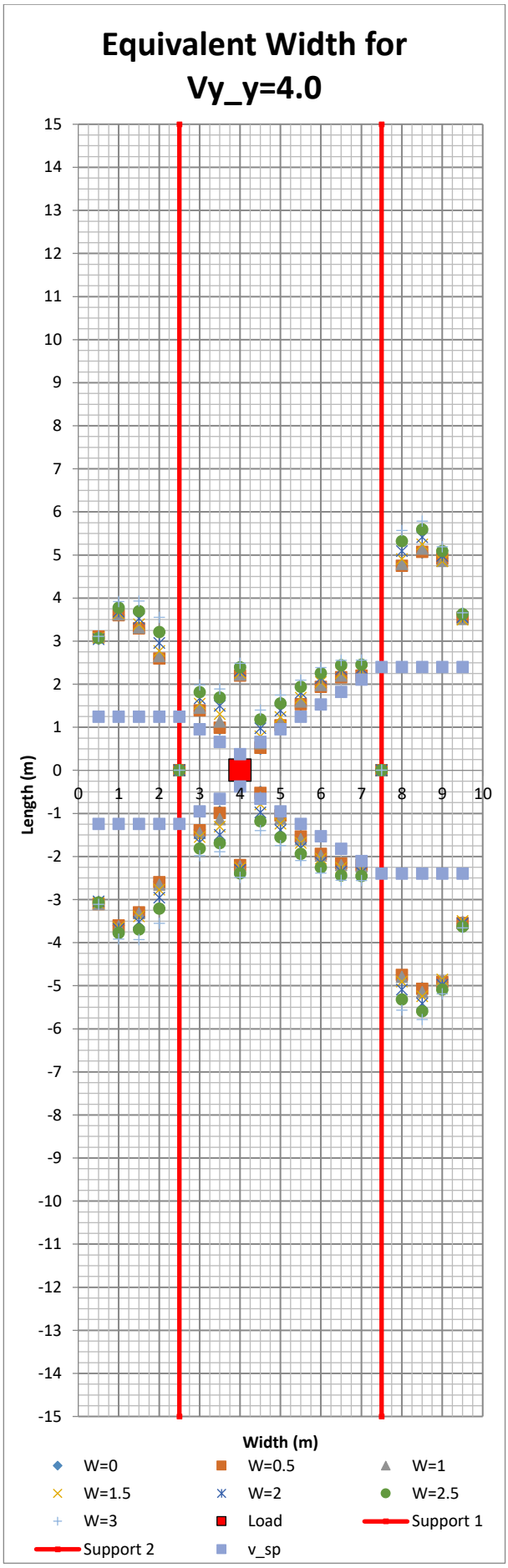

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

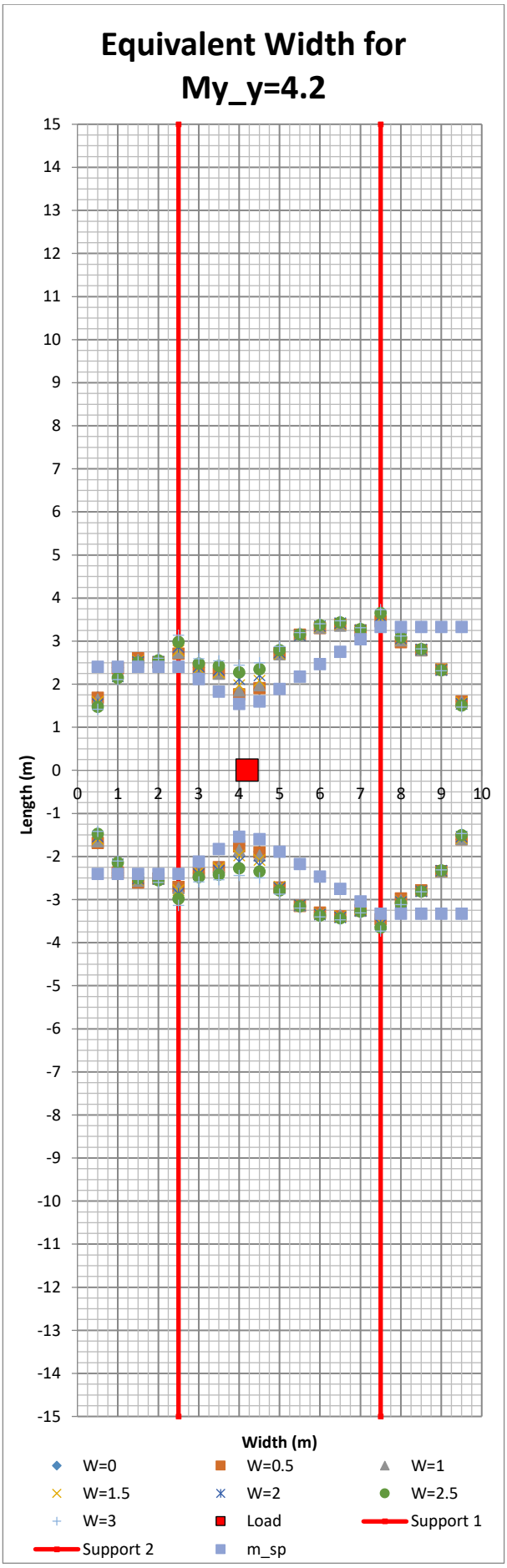

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

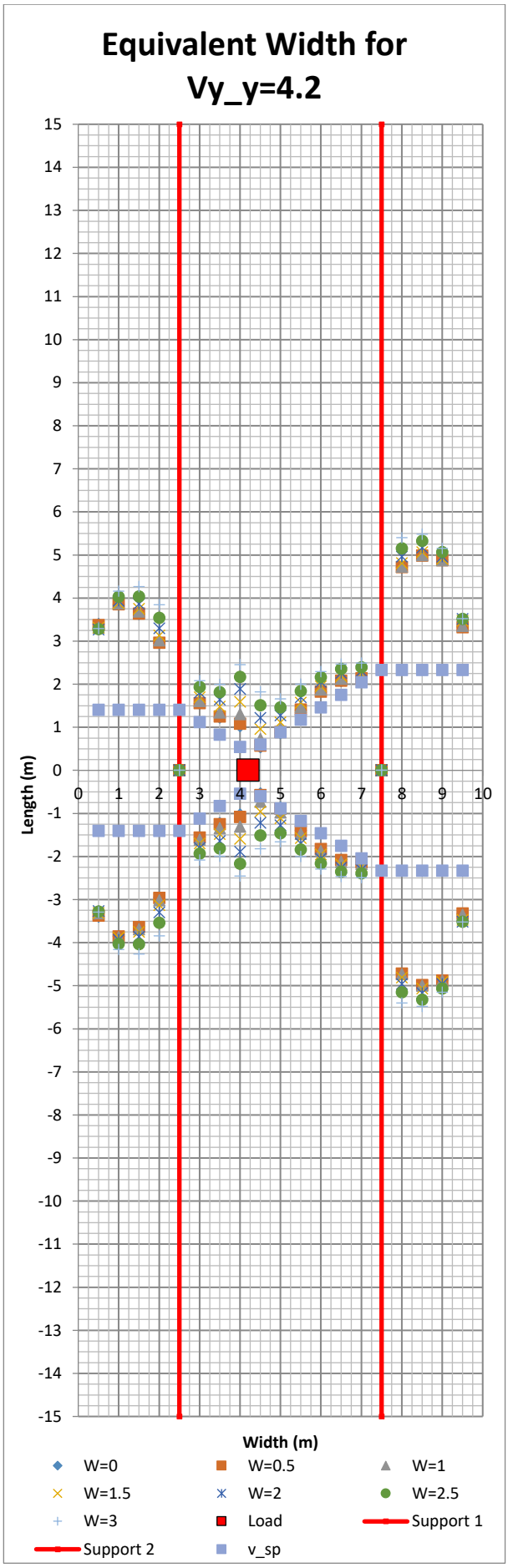

Equivalent Width for My_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

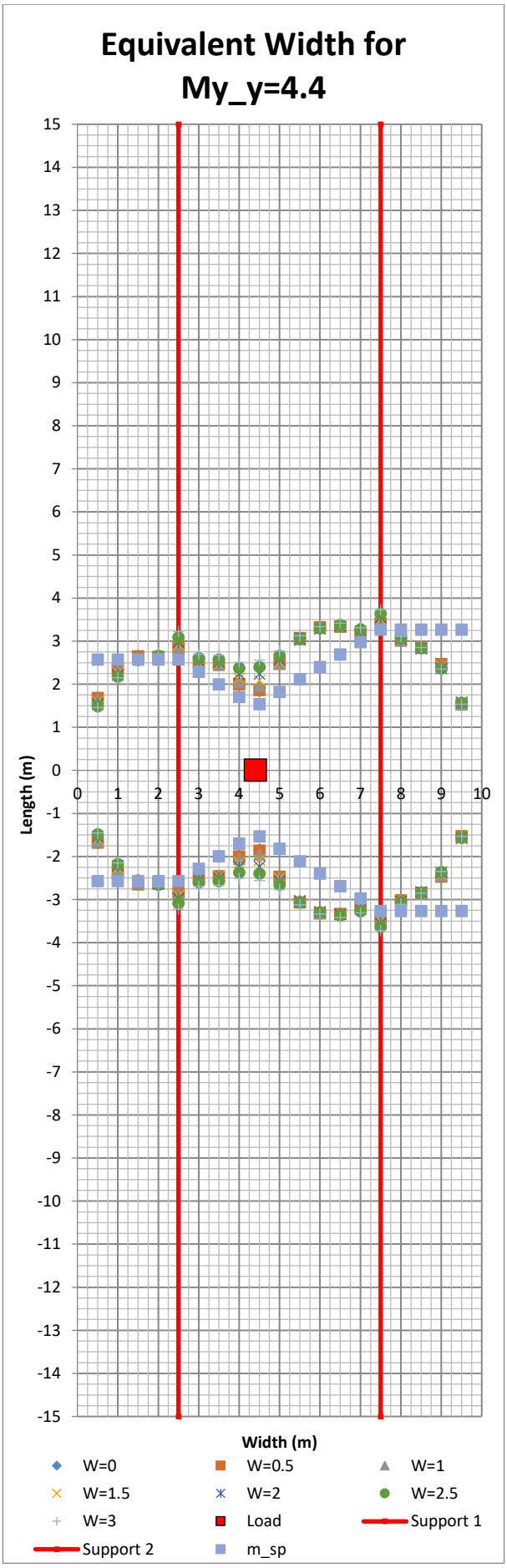

Equivalent Width for Vy_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

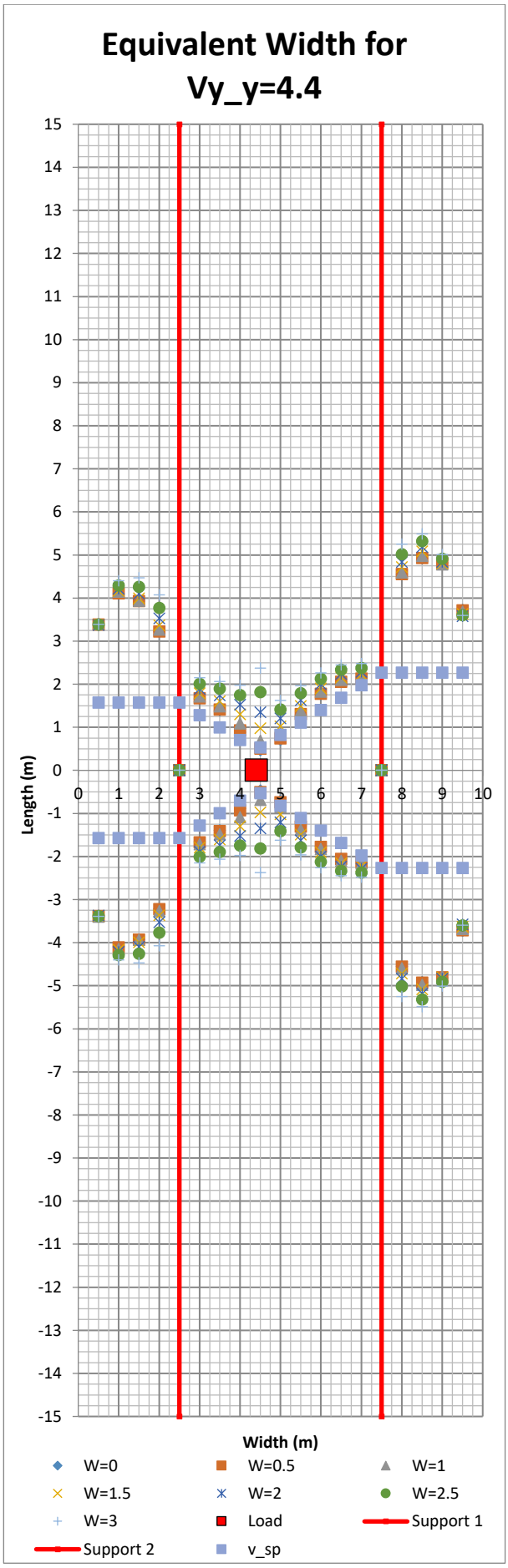

Equivalent Width for My_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

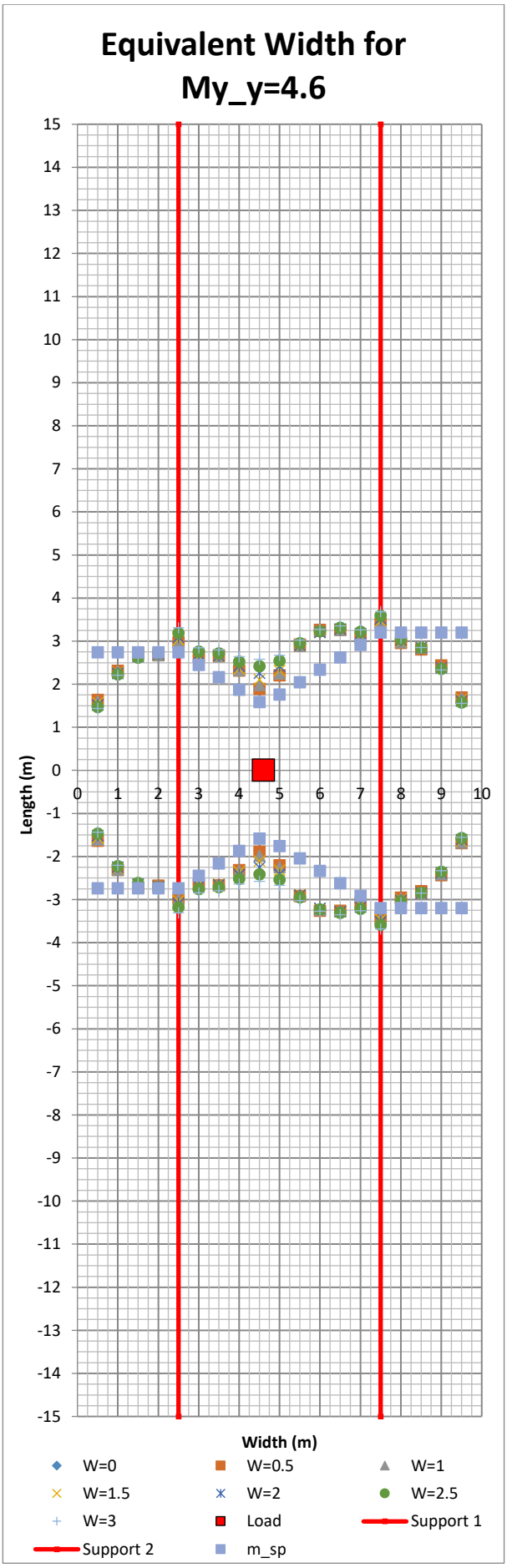

Equivalent Width for Vy_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

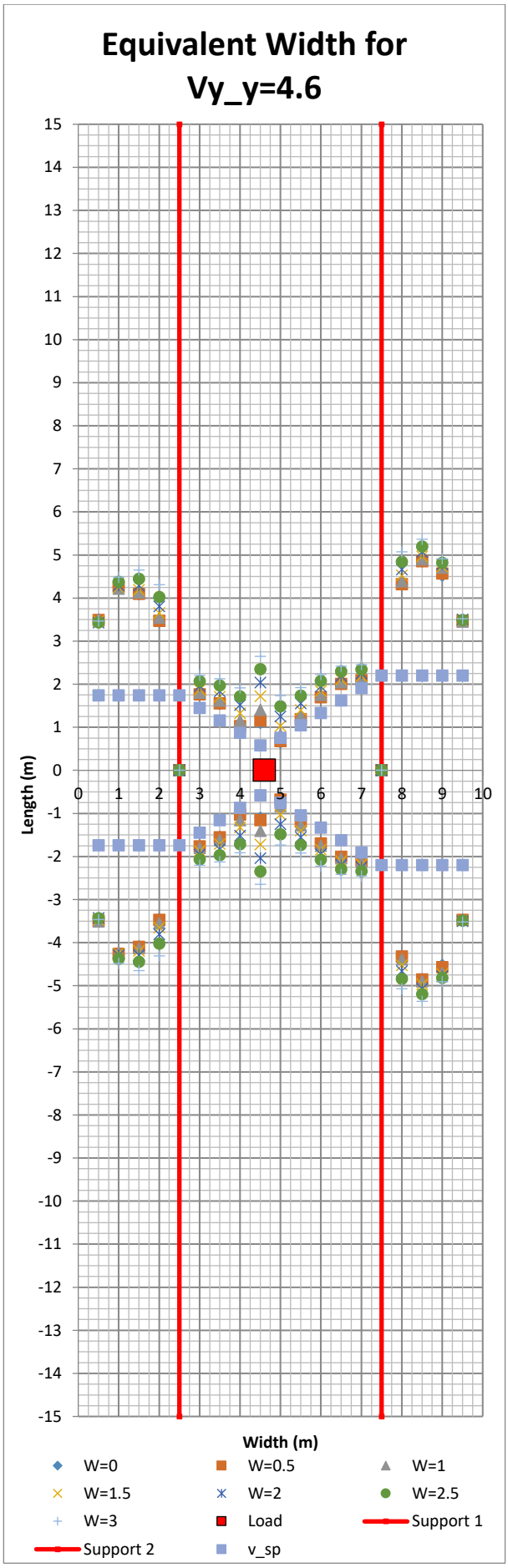

Equivalent Width for My_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

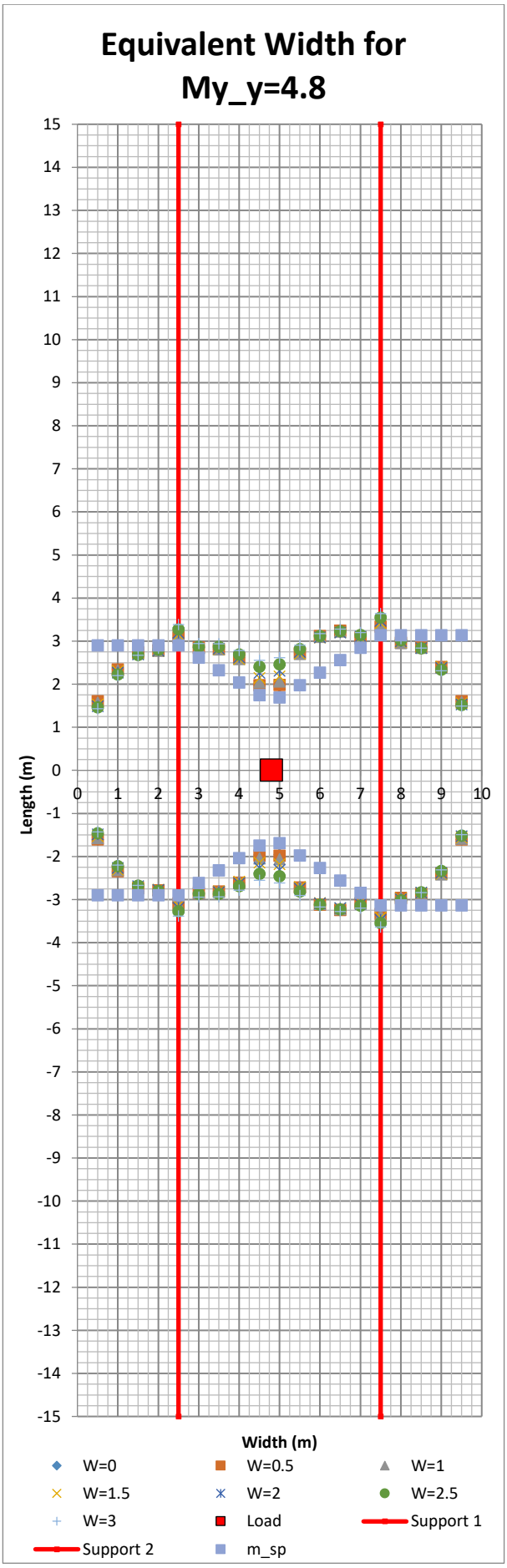

Equivalent Width for Vy_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

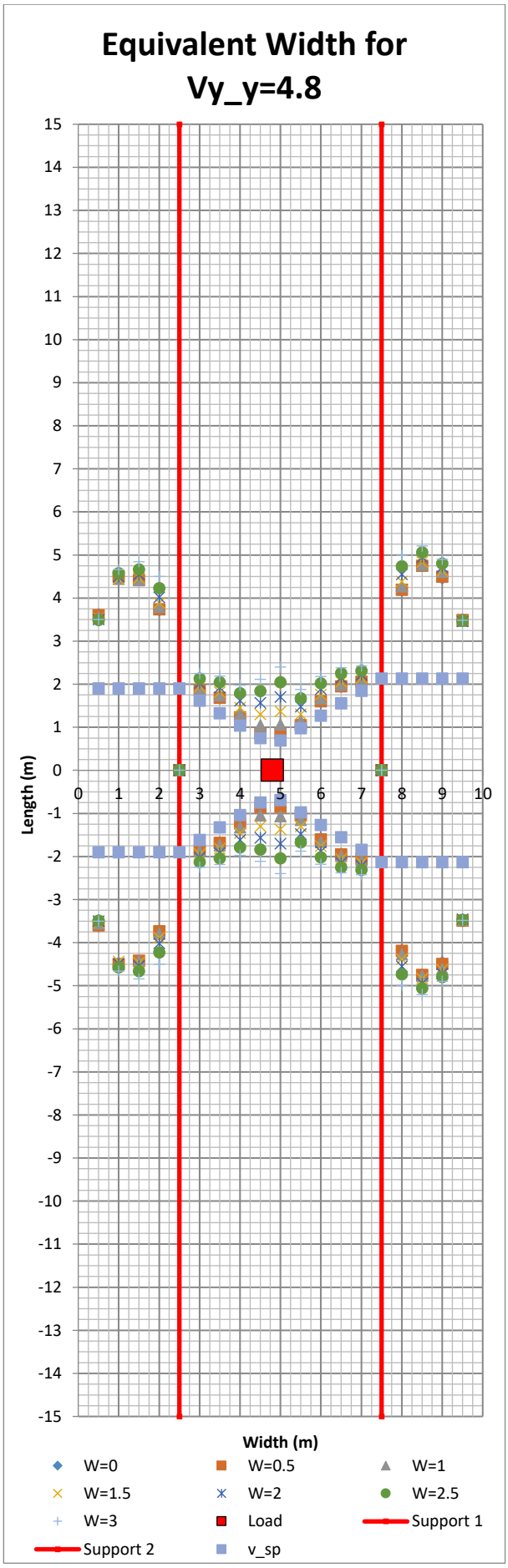

Equivalent Width for My_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

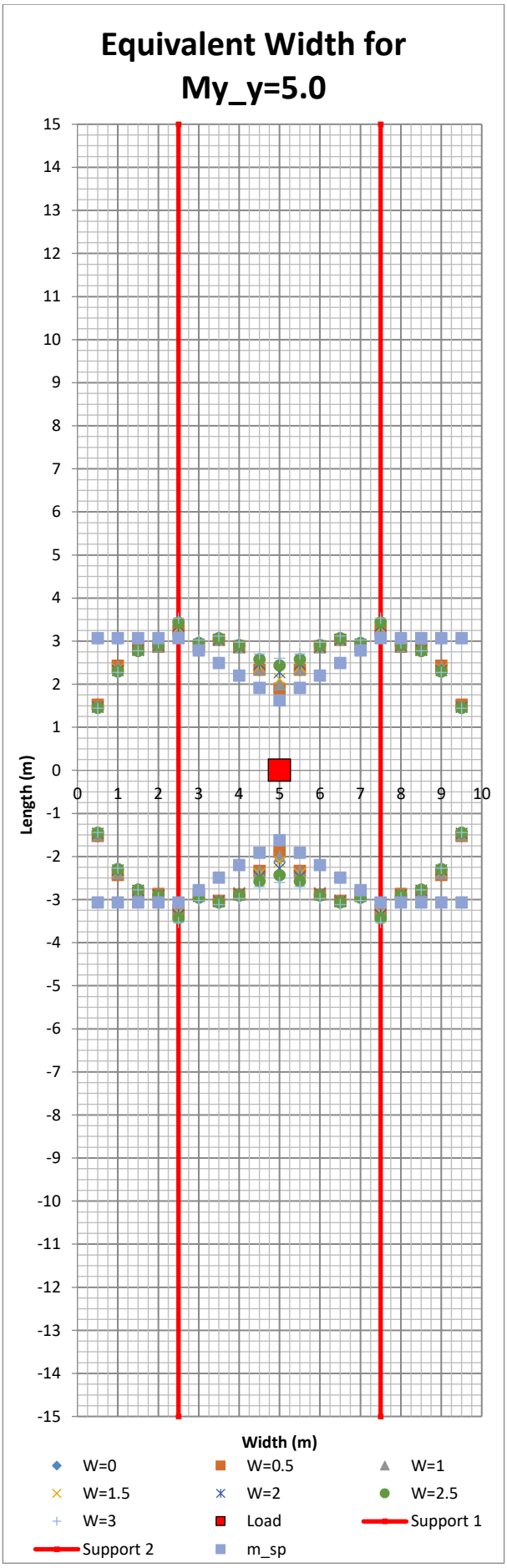

Equivalent Width for Vy_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

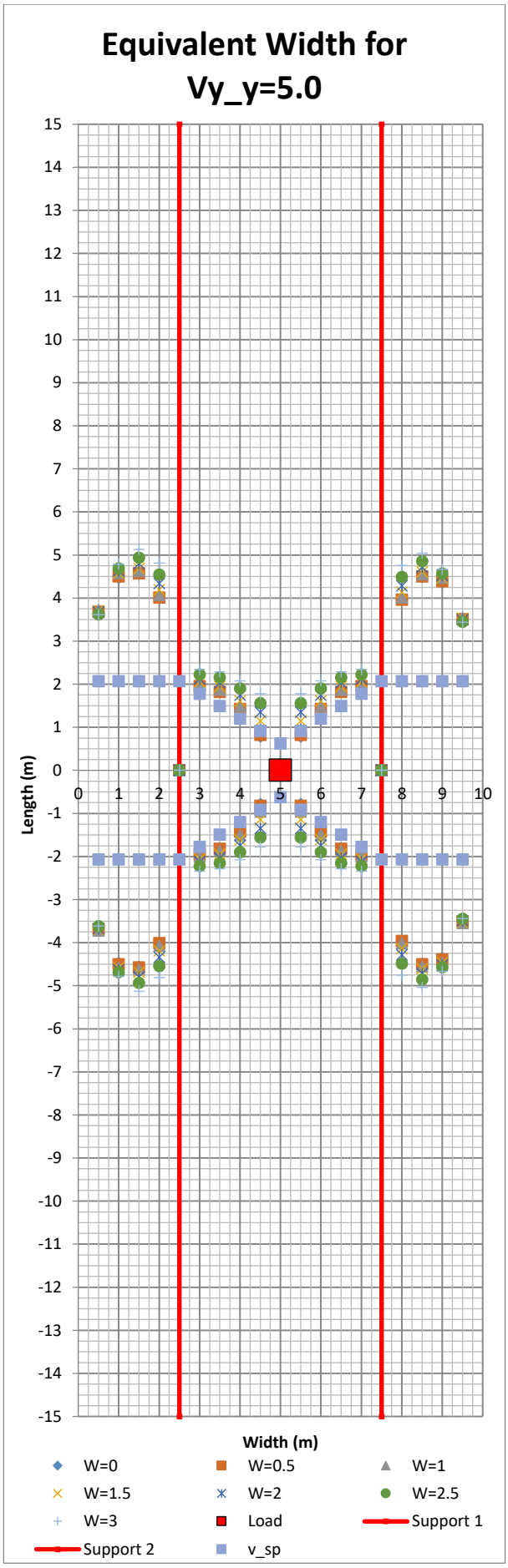

Equivalent Width for My_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

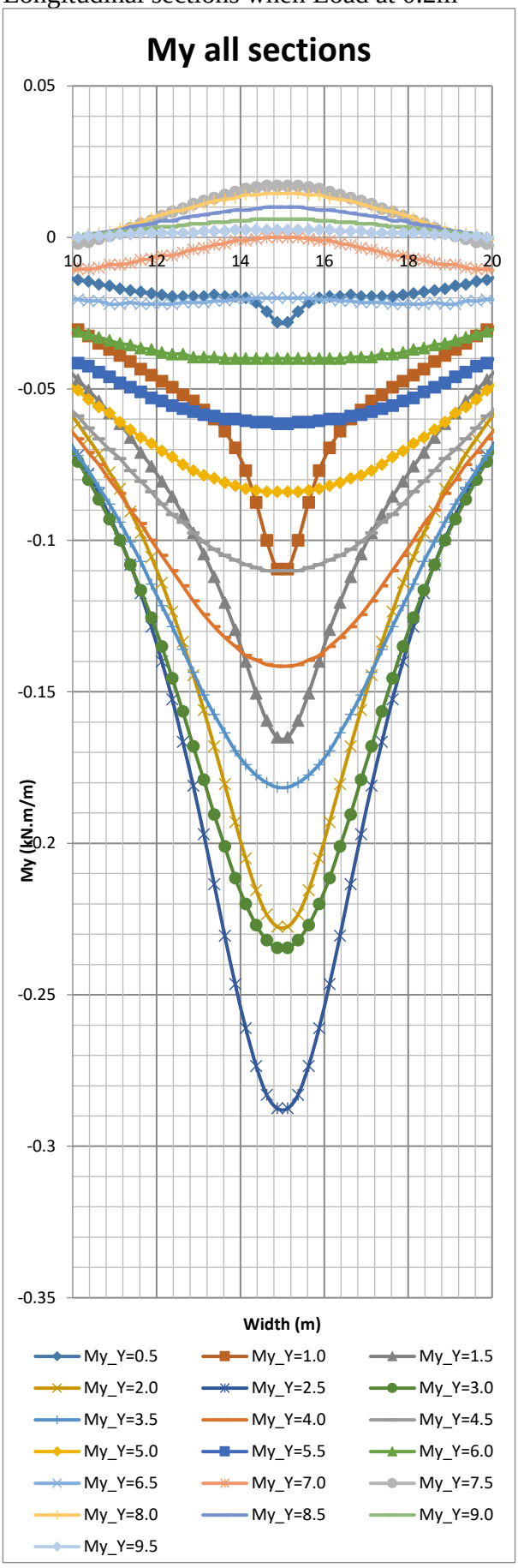

Equivalent Width for My_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

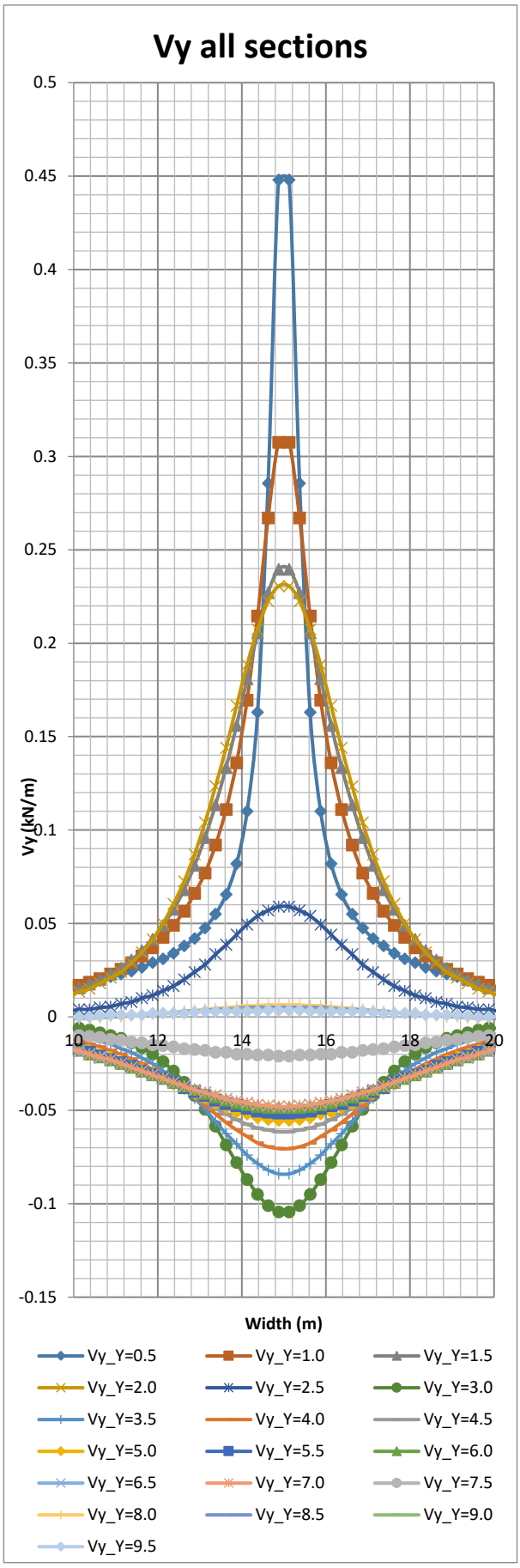

Equivalent Width for Vy_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp


Equivalent Width for My_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

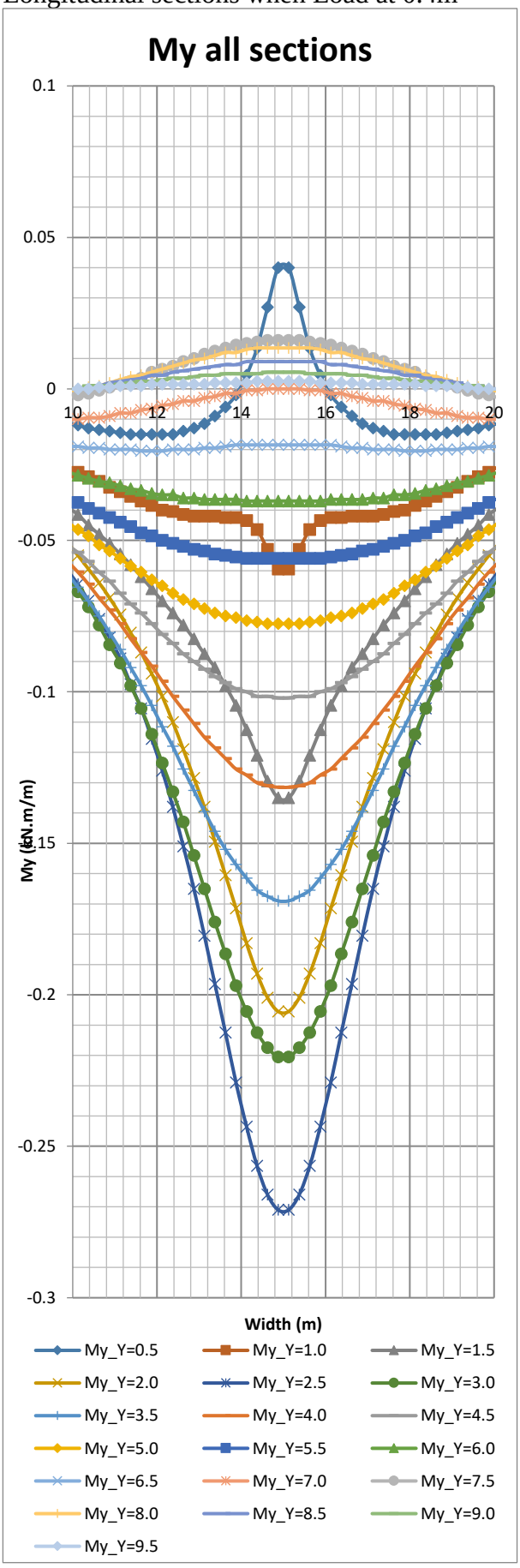

Equivalent Width for Vy_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

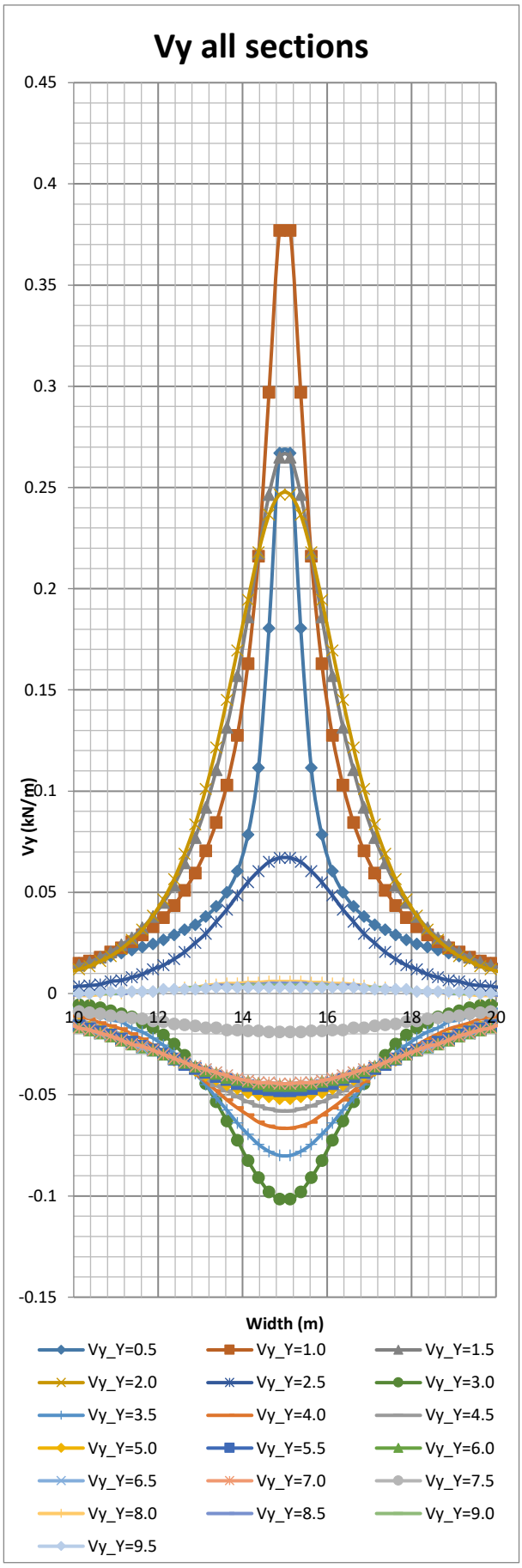
Longitudinal sections when Load at 0.2m



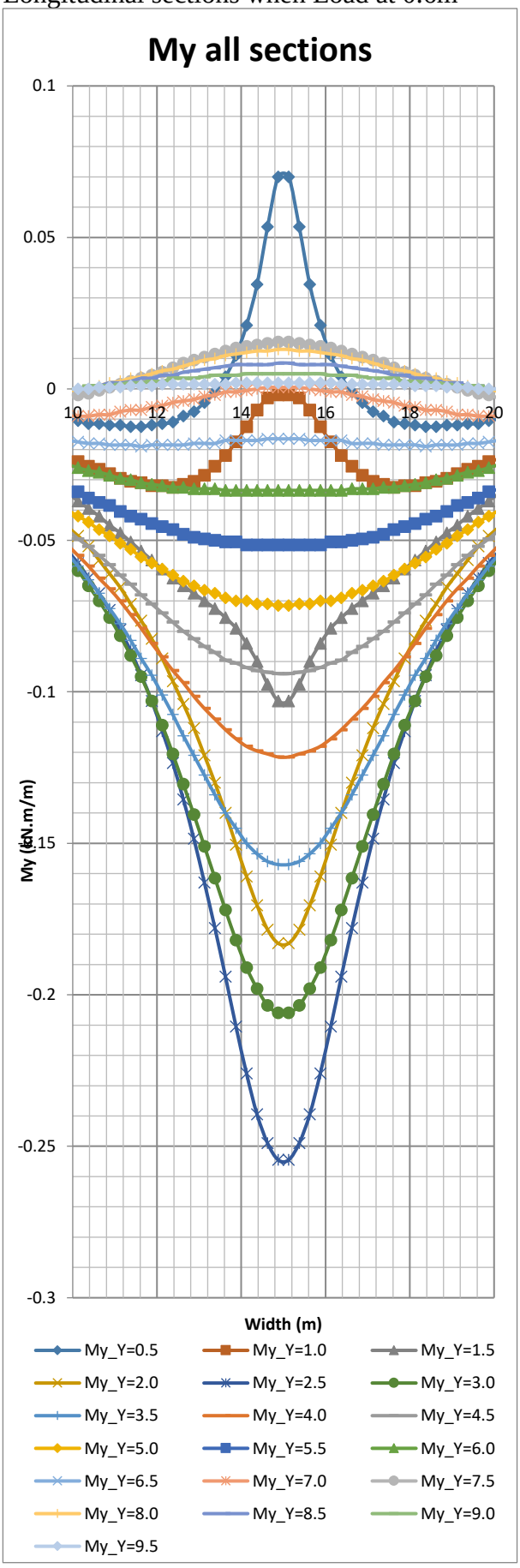


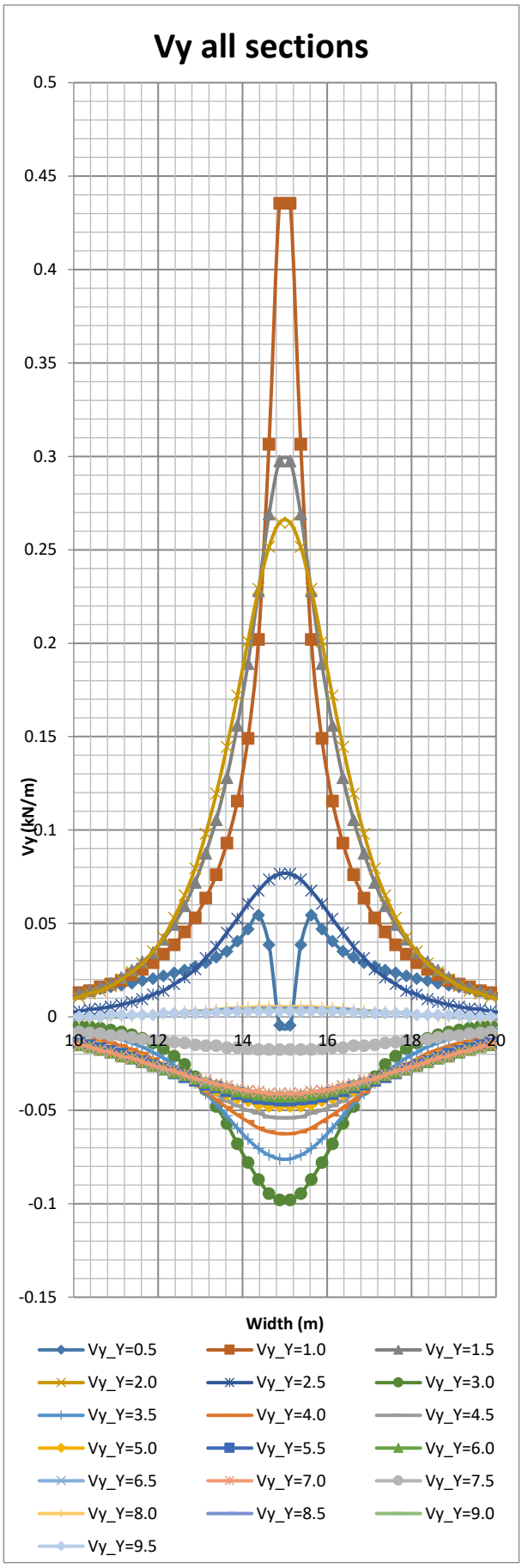
Longitudinal sections when Load at 0.4m



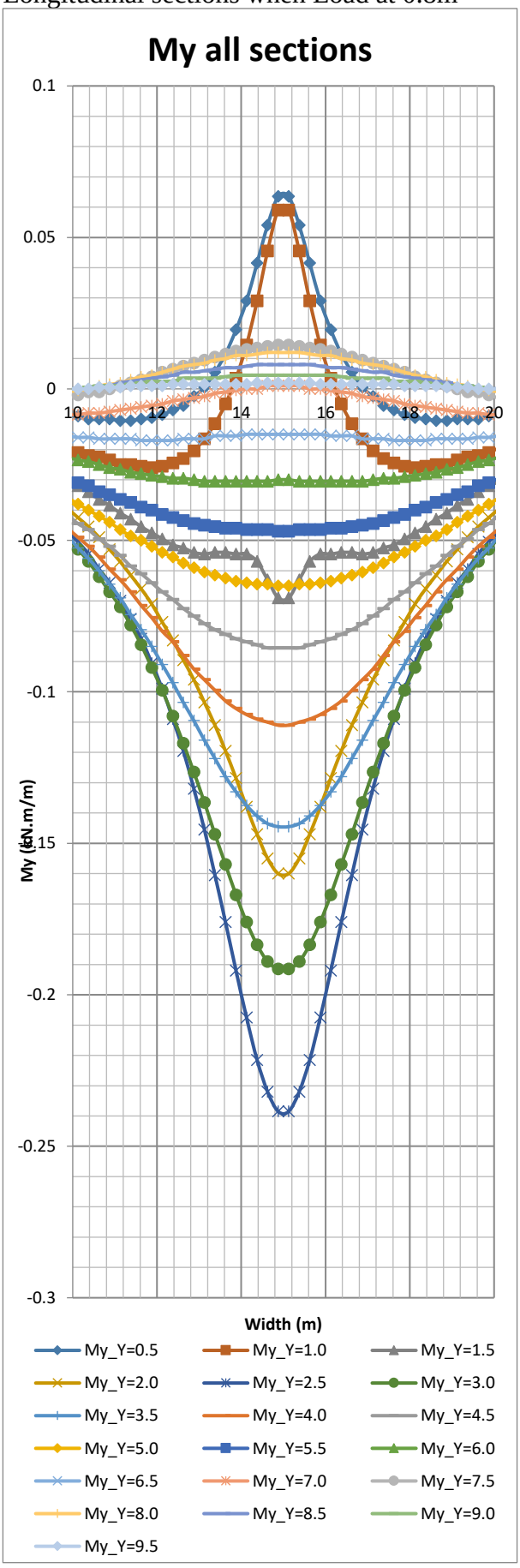


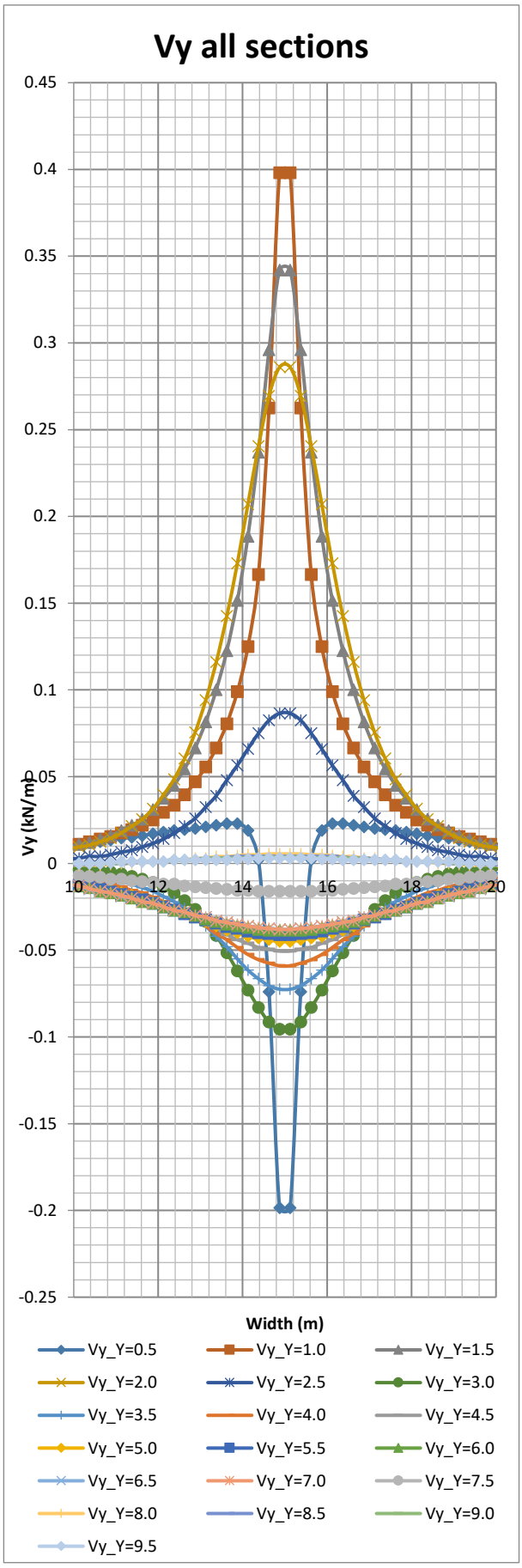
Longitudinal sections when Load at 0.6m



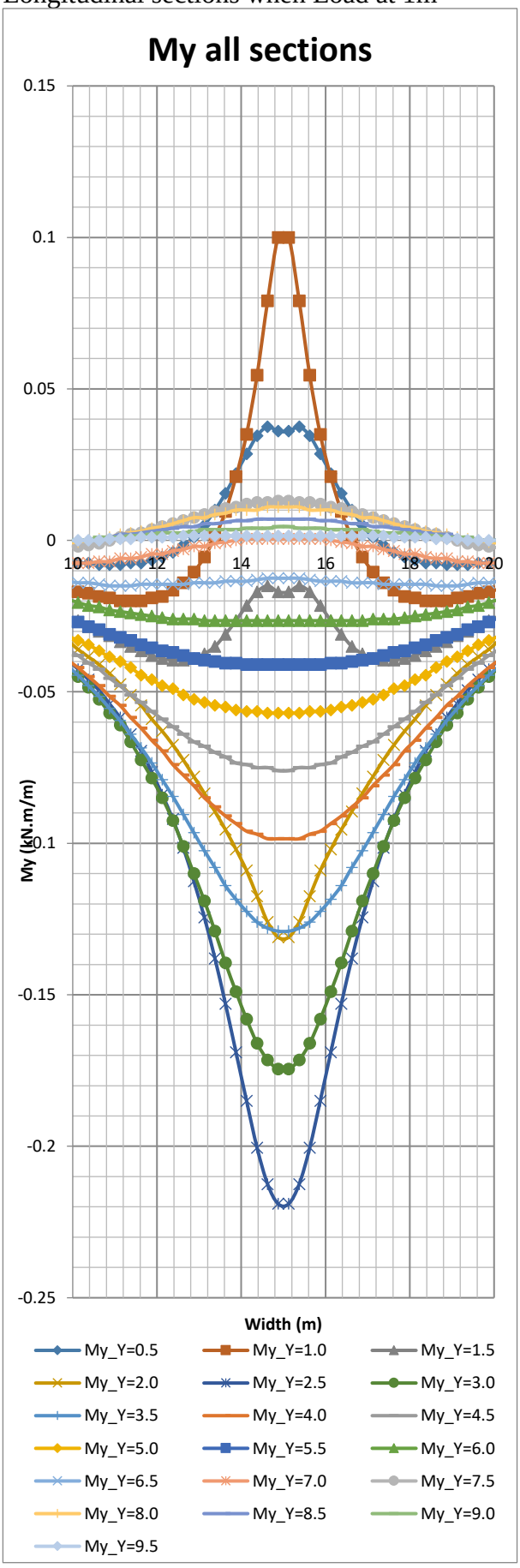


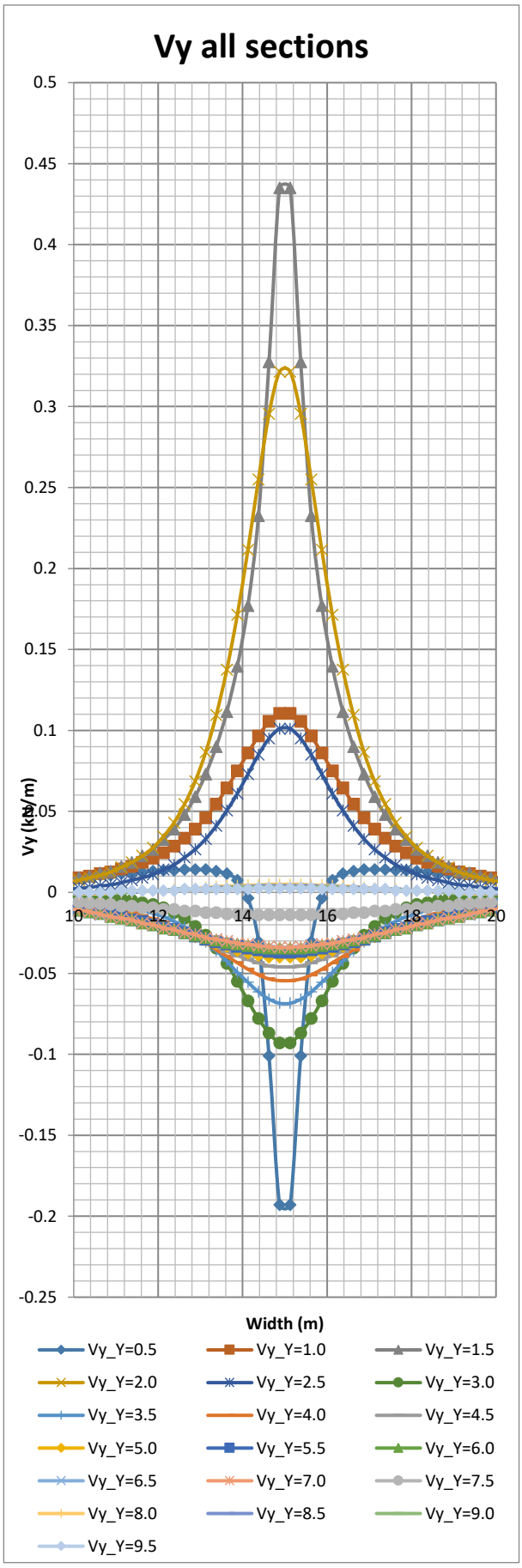
Longitudinal sections when Load at 0.8m



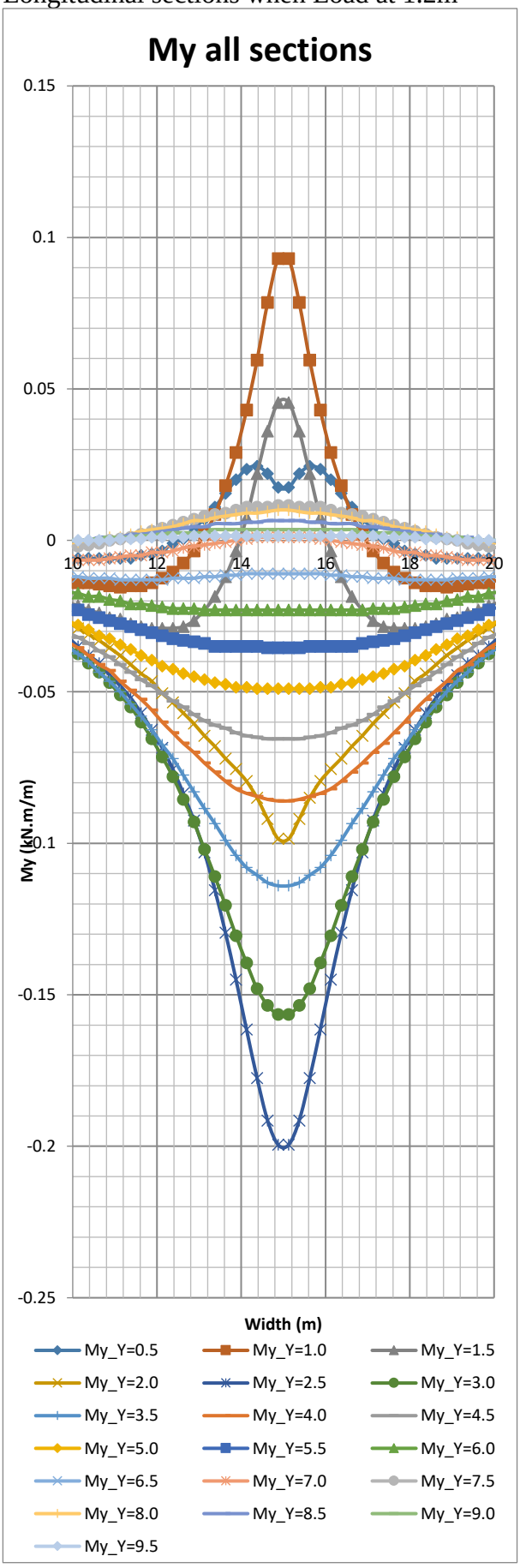


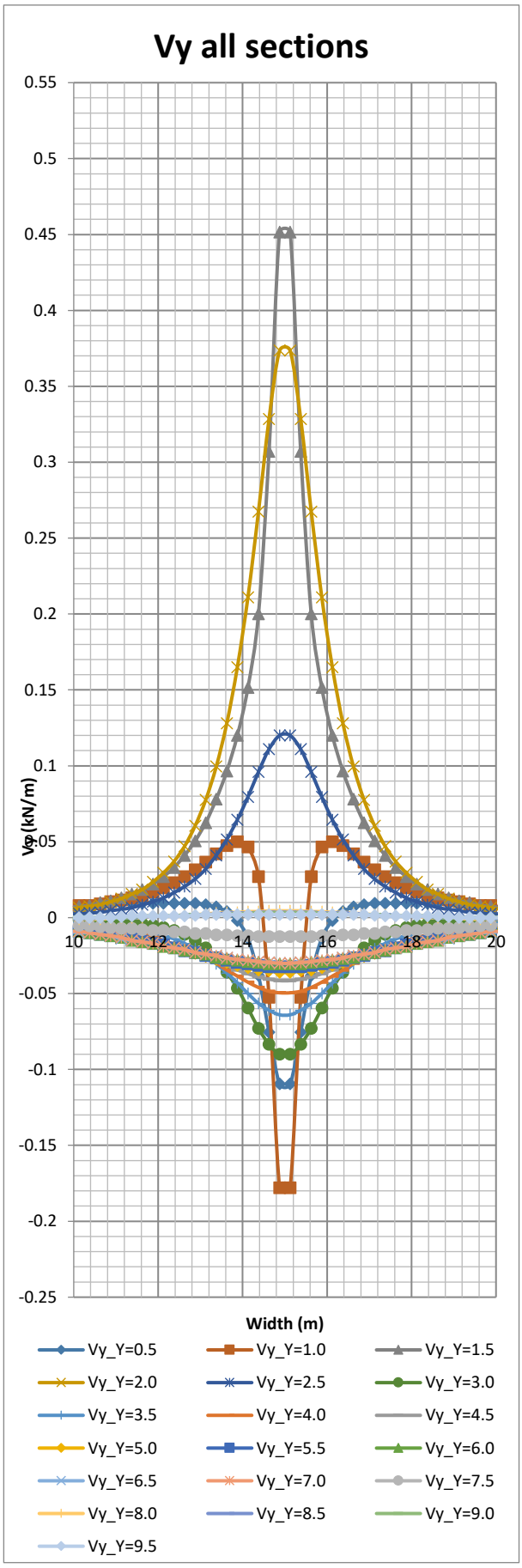
Longitudinal sections when Load at 1m



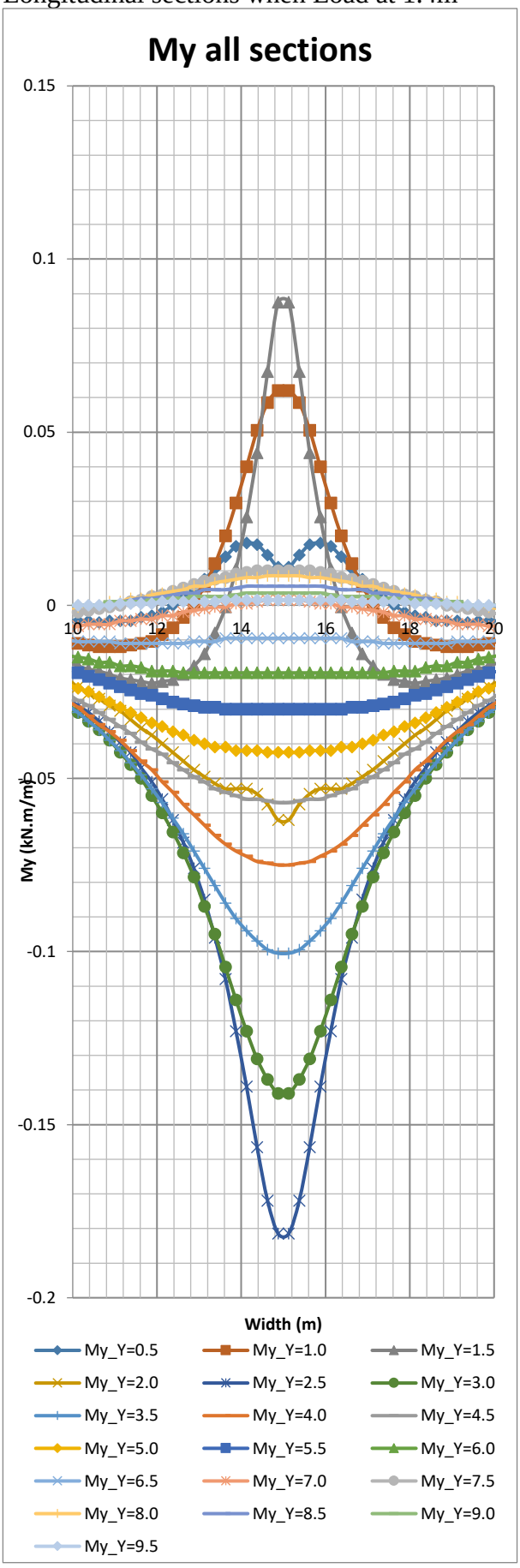


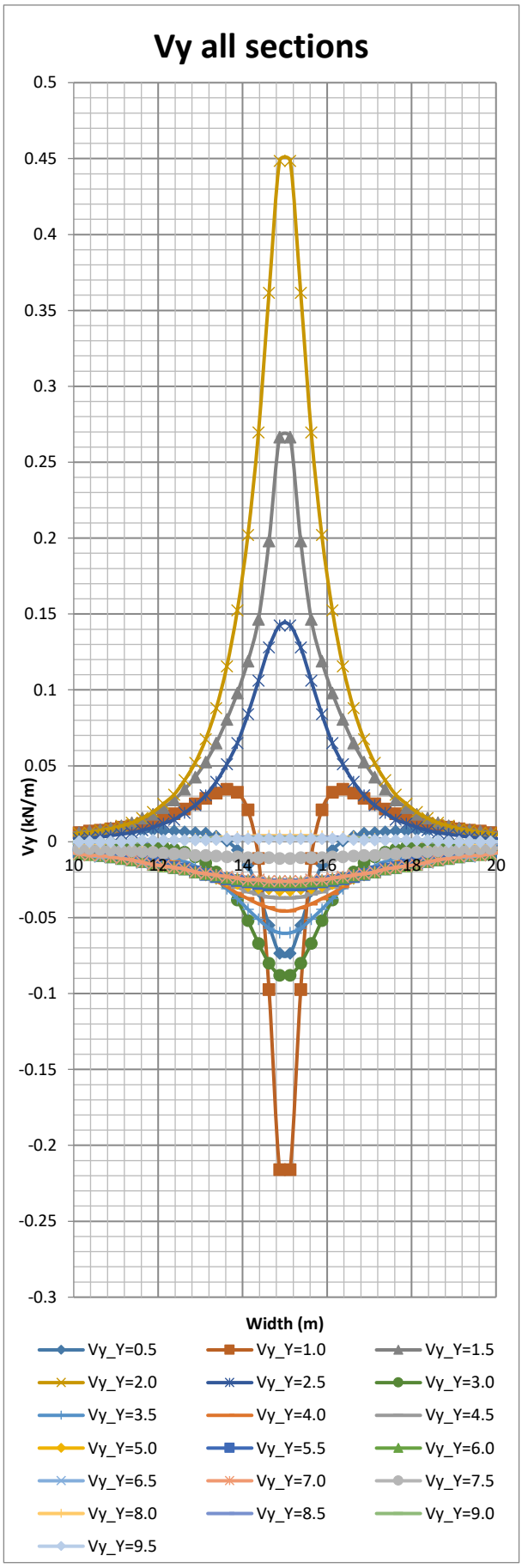
Longitudinal sections when Load at 1.2m



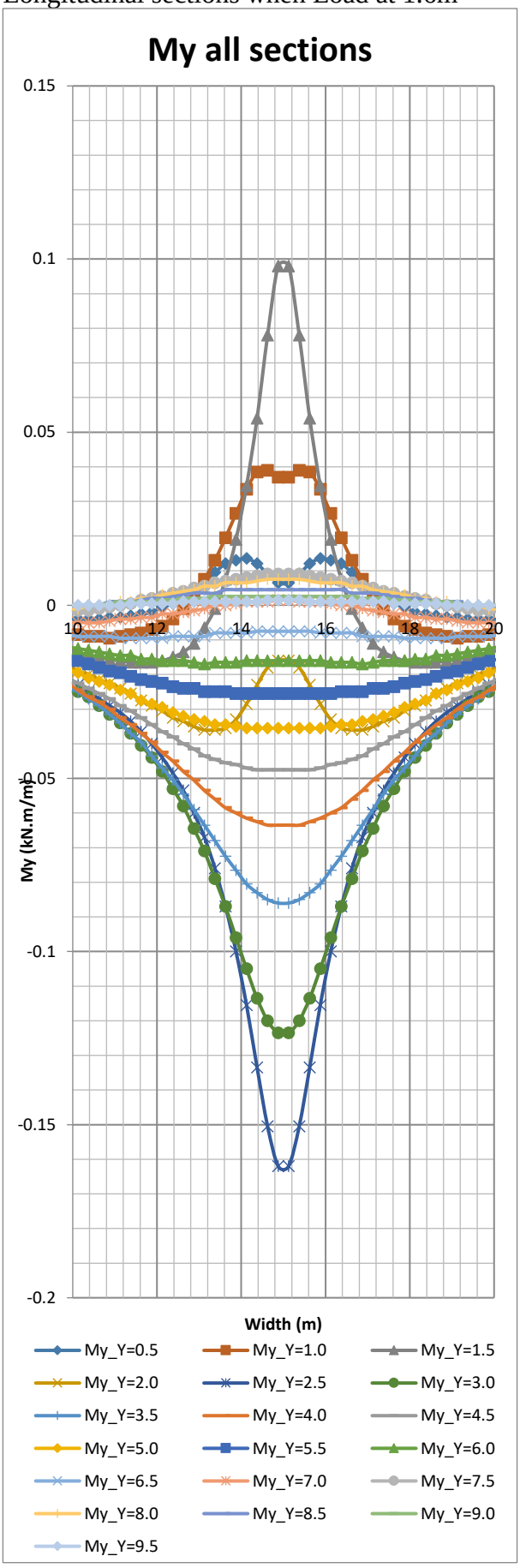


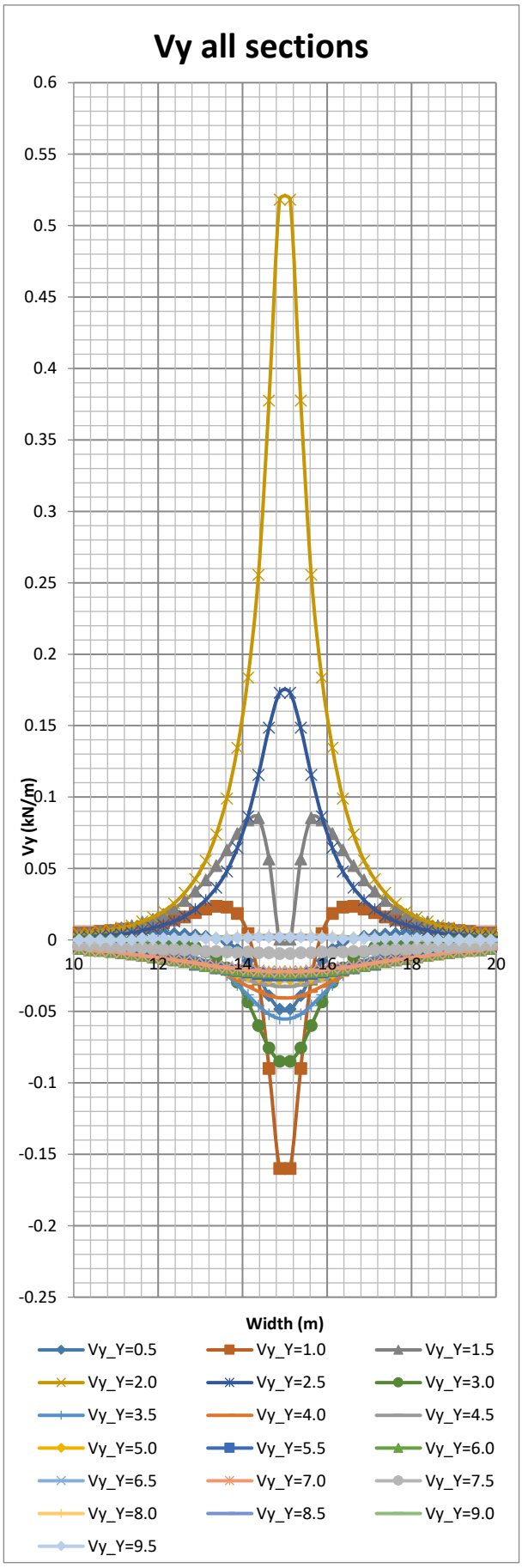
Longitudinal sections when Load at 1.4m



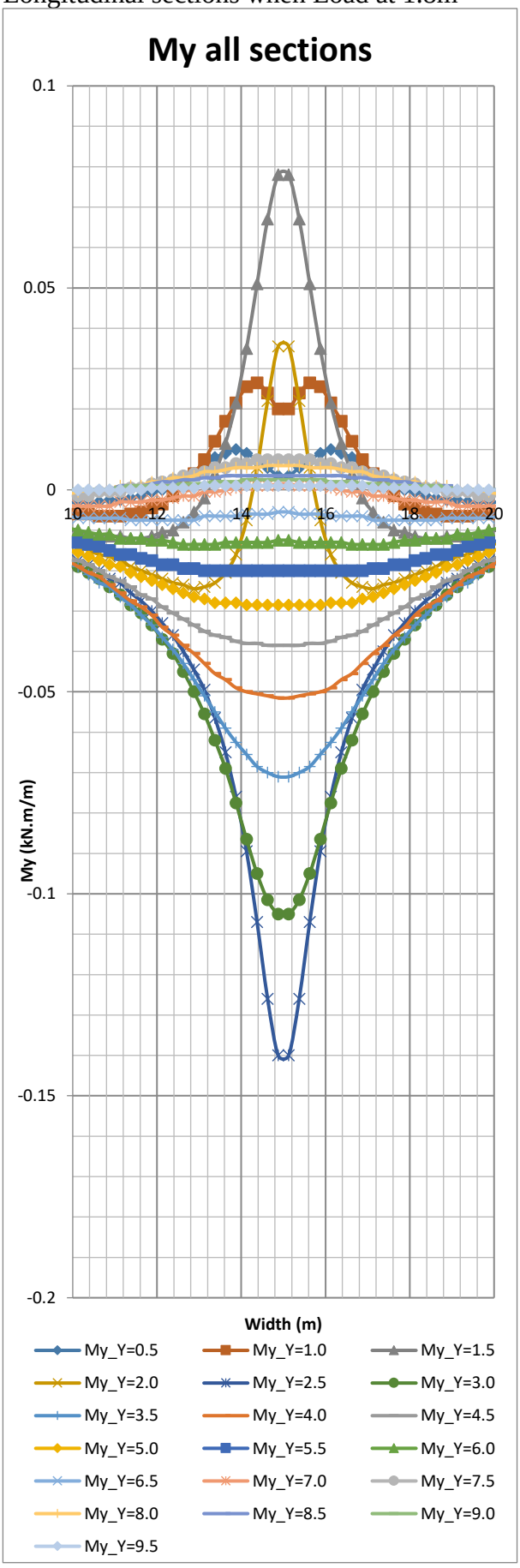


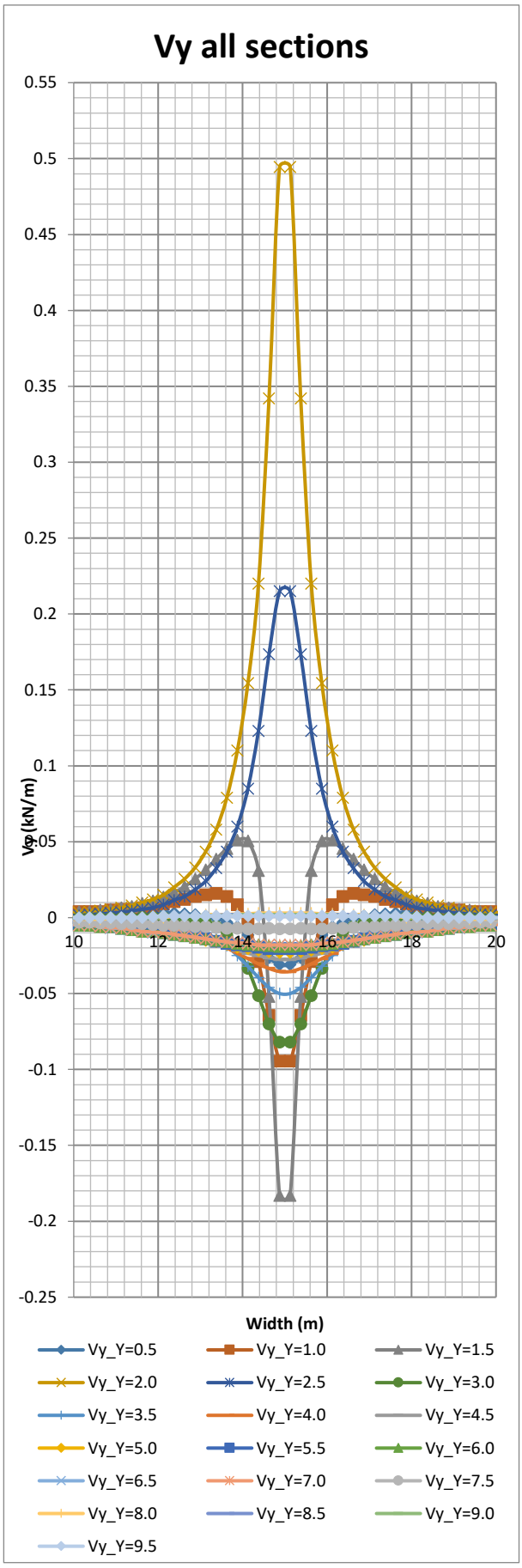
Longitudinal sections when Load at 1.6m



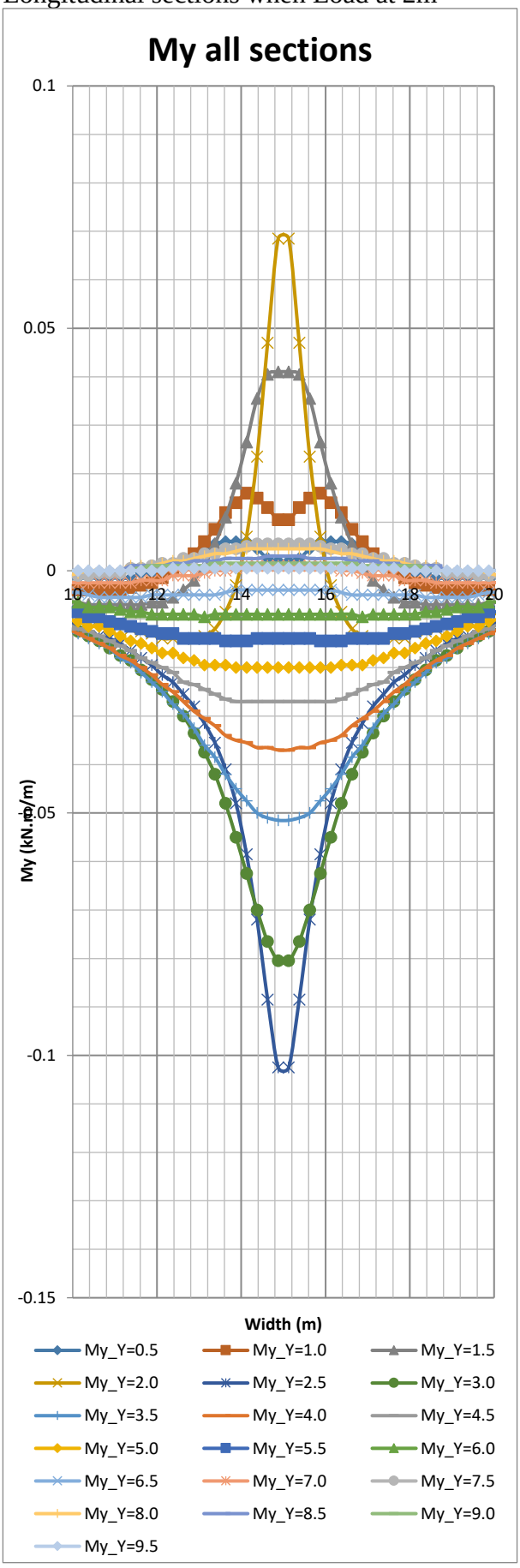


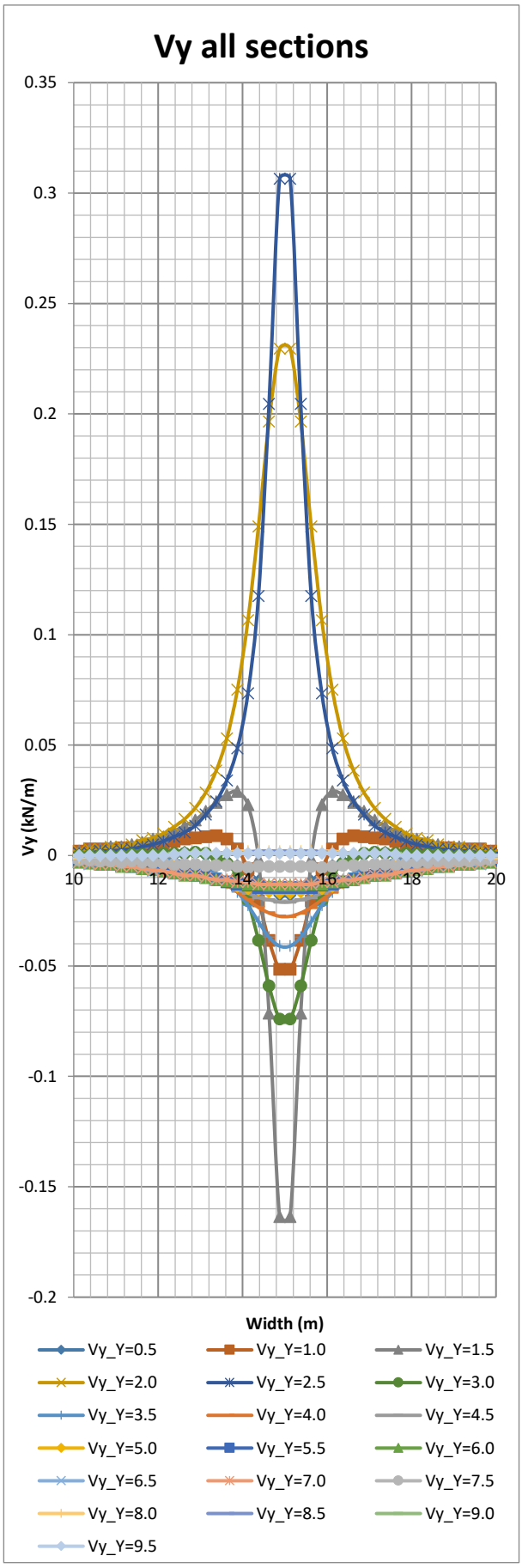
Longitudinal sections when Load at 1.8m



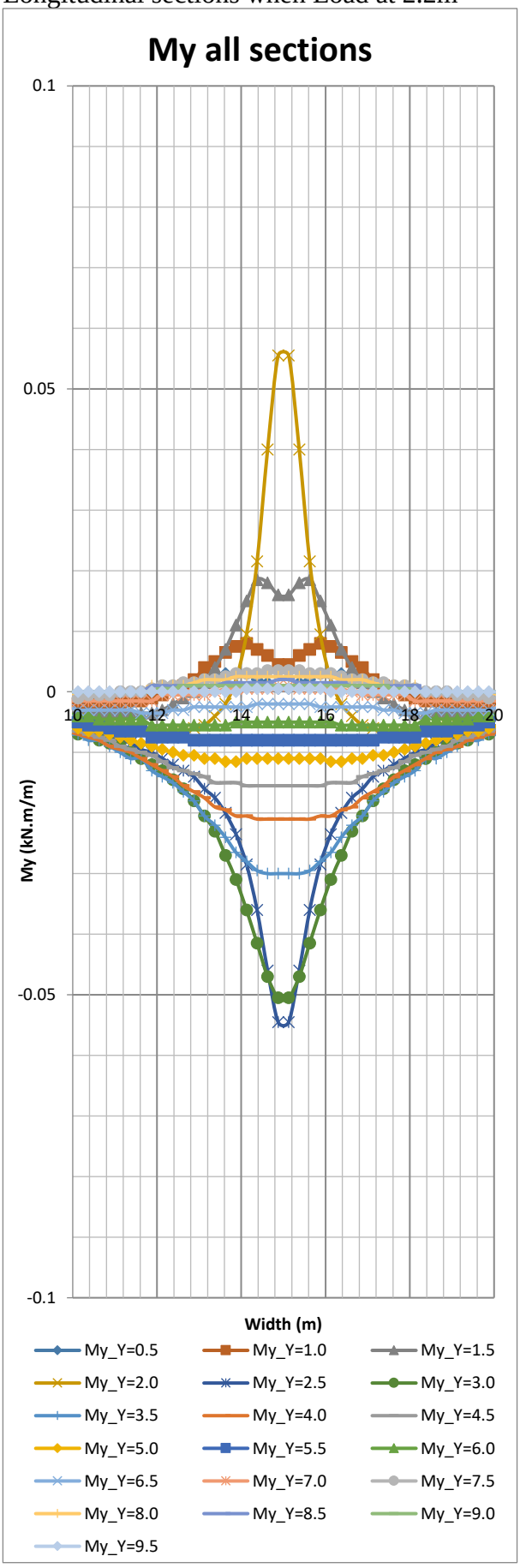


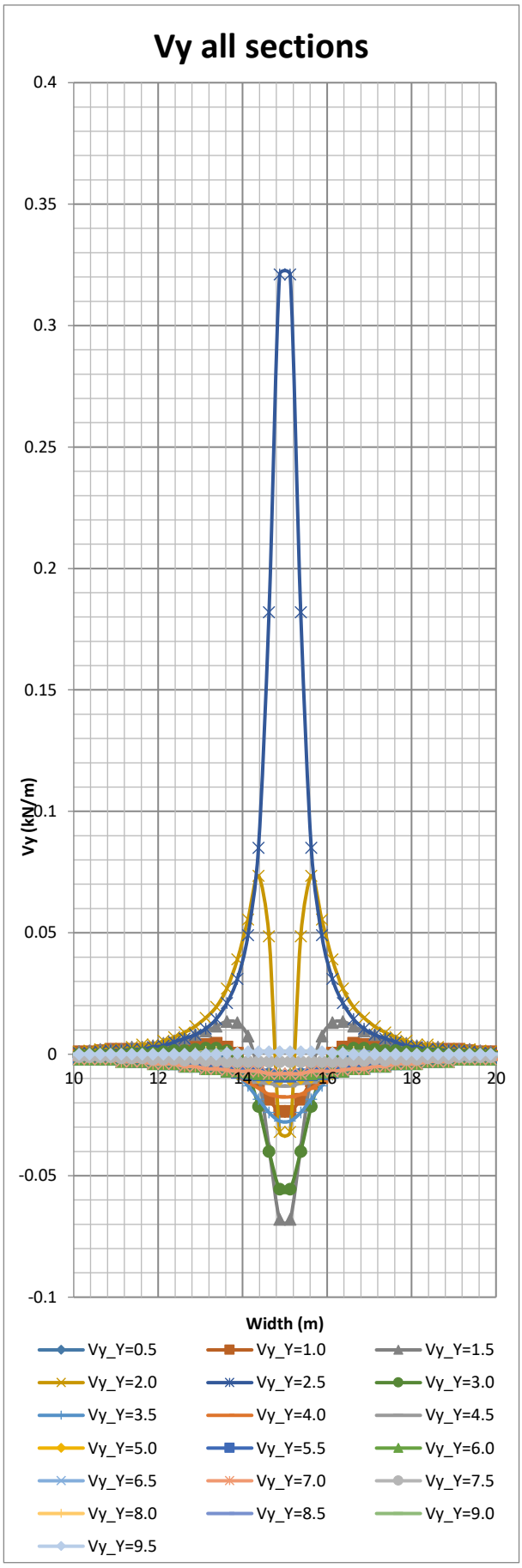
Longitudinal sections when Load at 2m



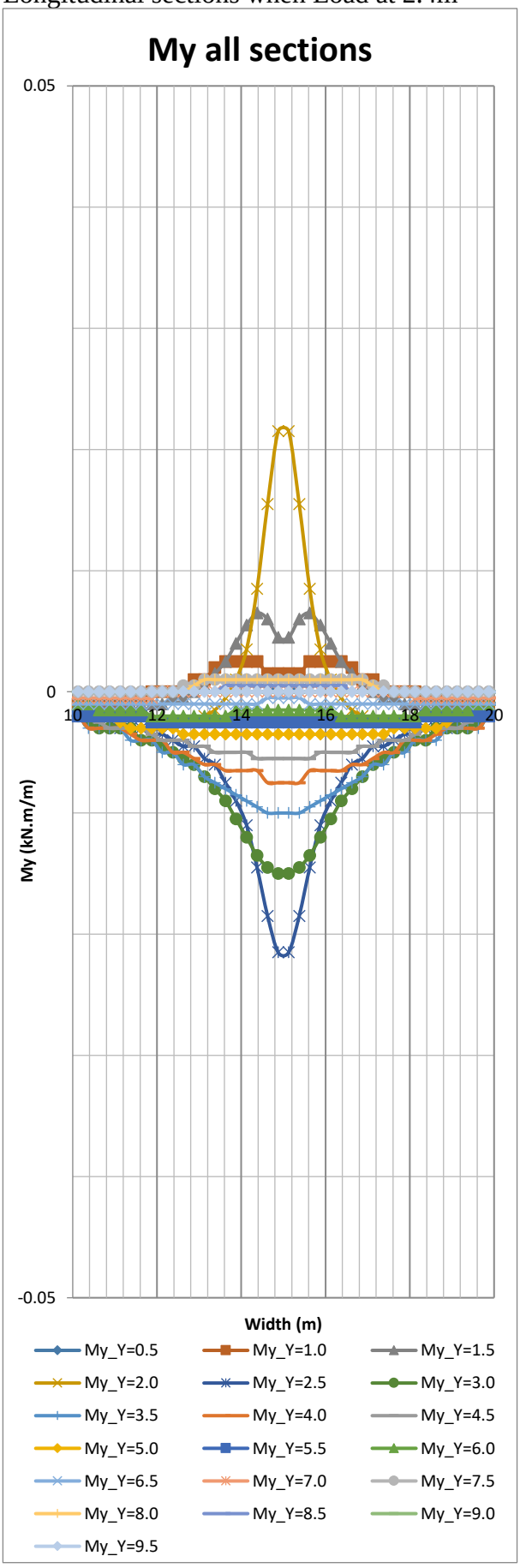


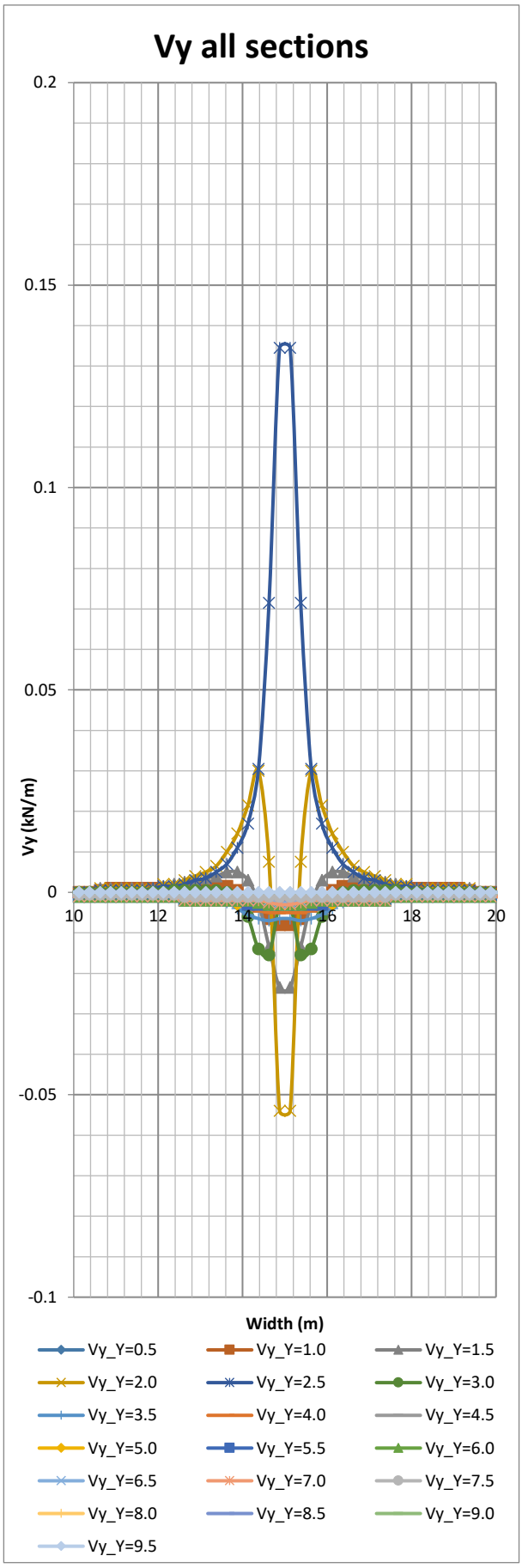
Longitudinal sections when Load at 2.2m



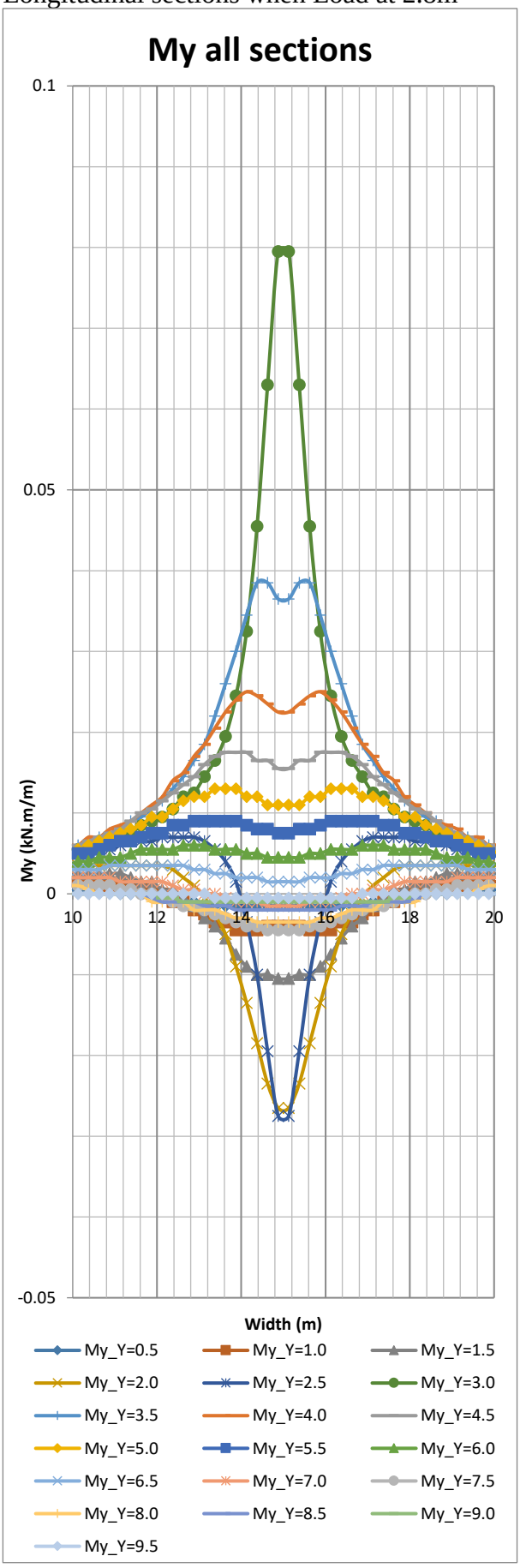


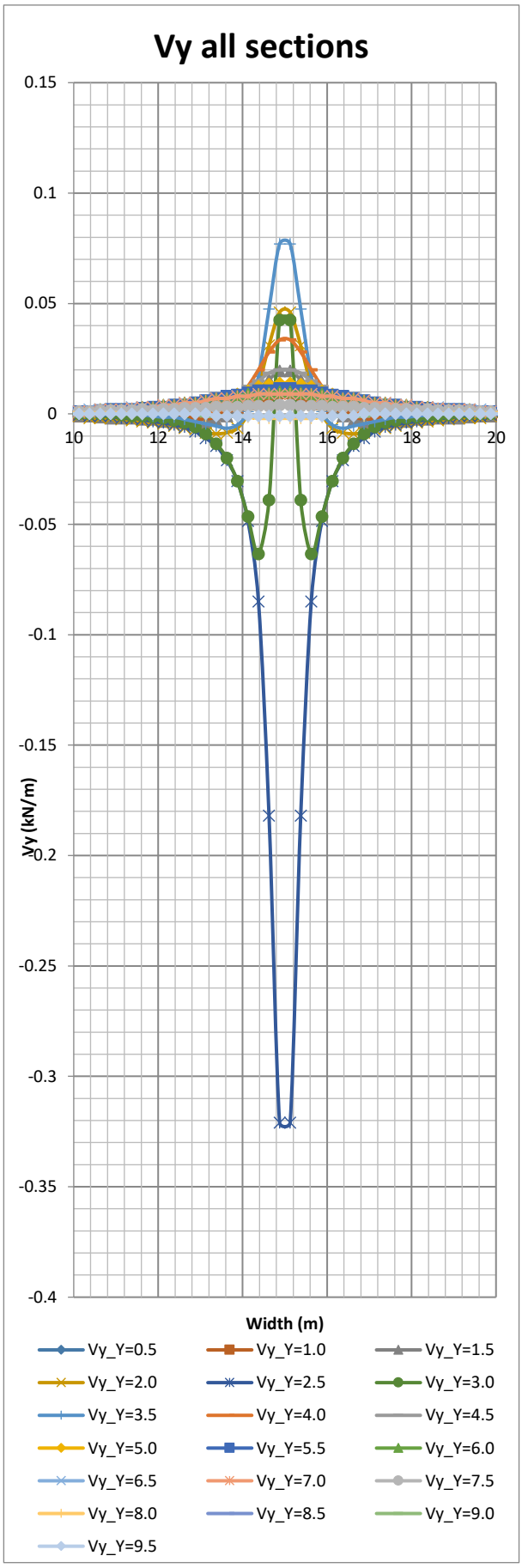
Longitudinal sections when Load at 2.4m



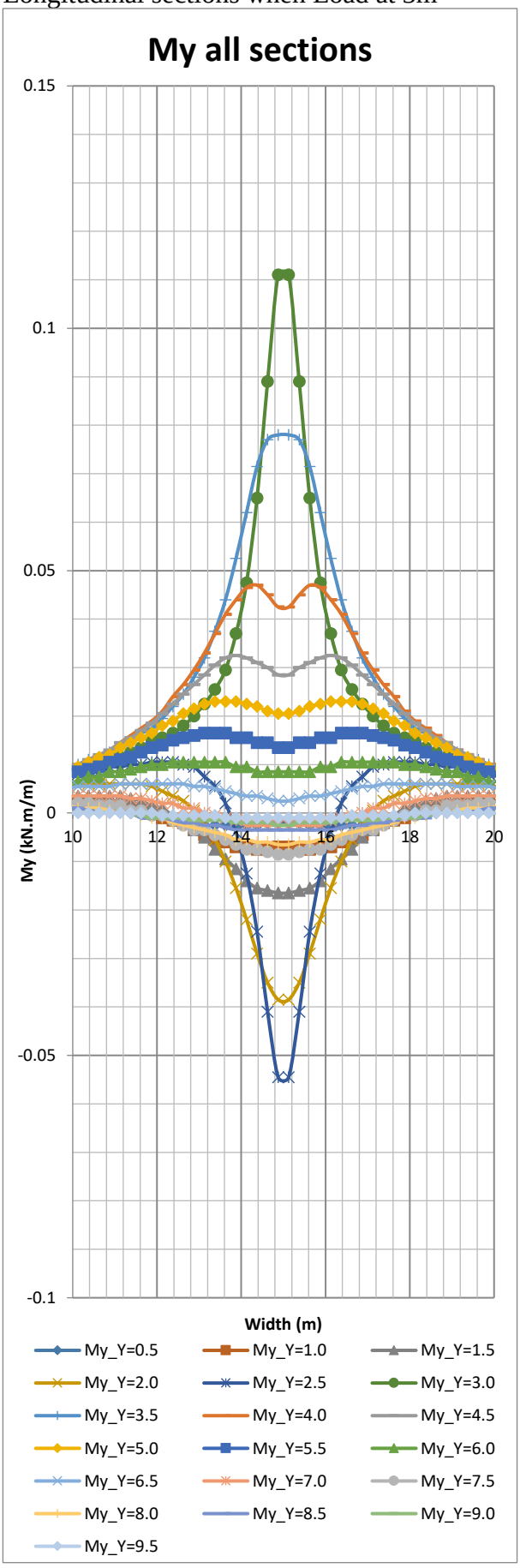


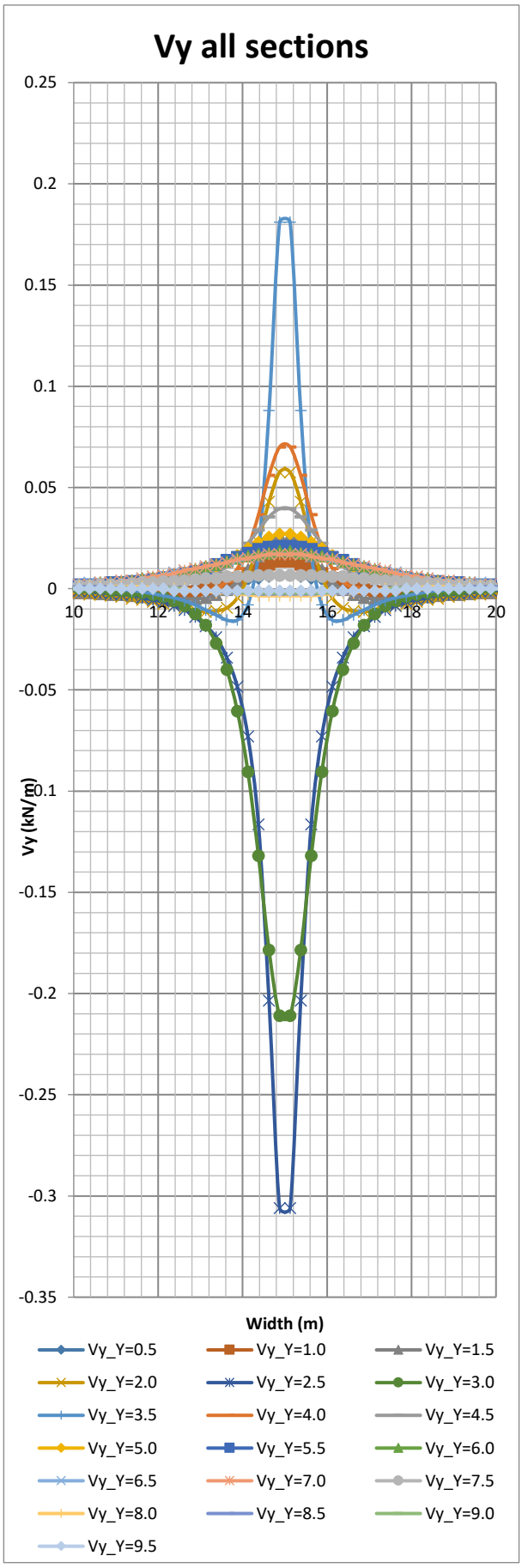
Longitudinal sections when Load at 2.8m



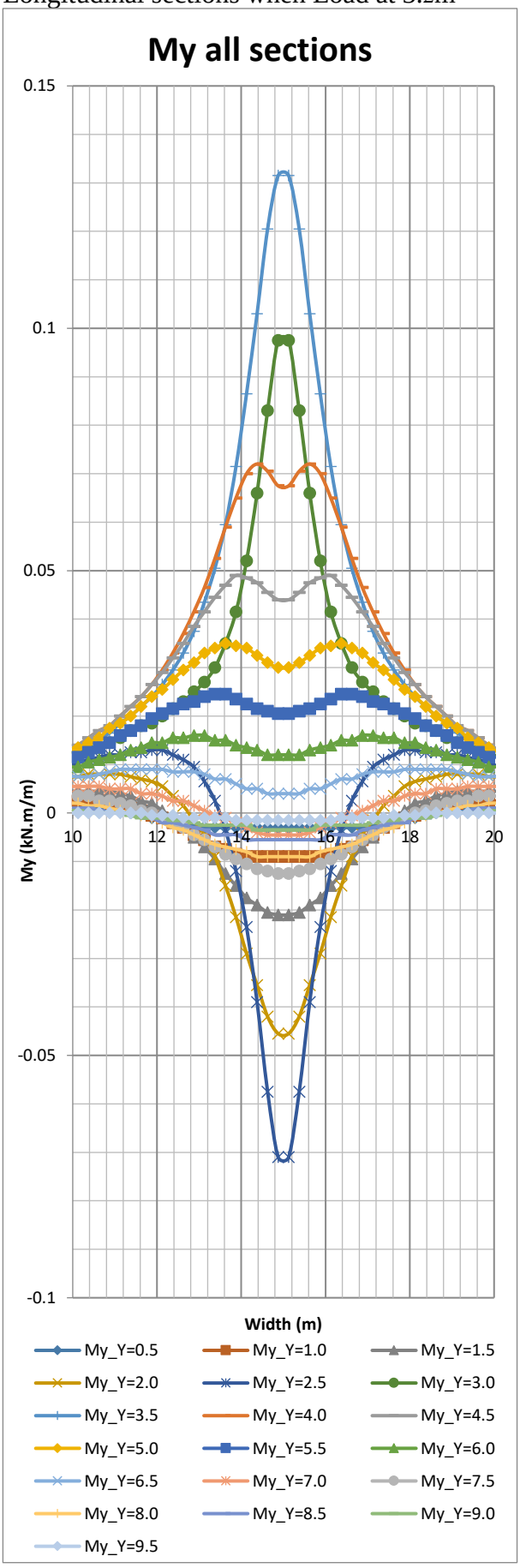


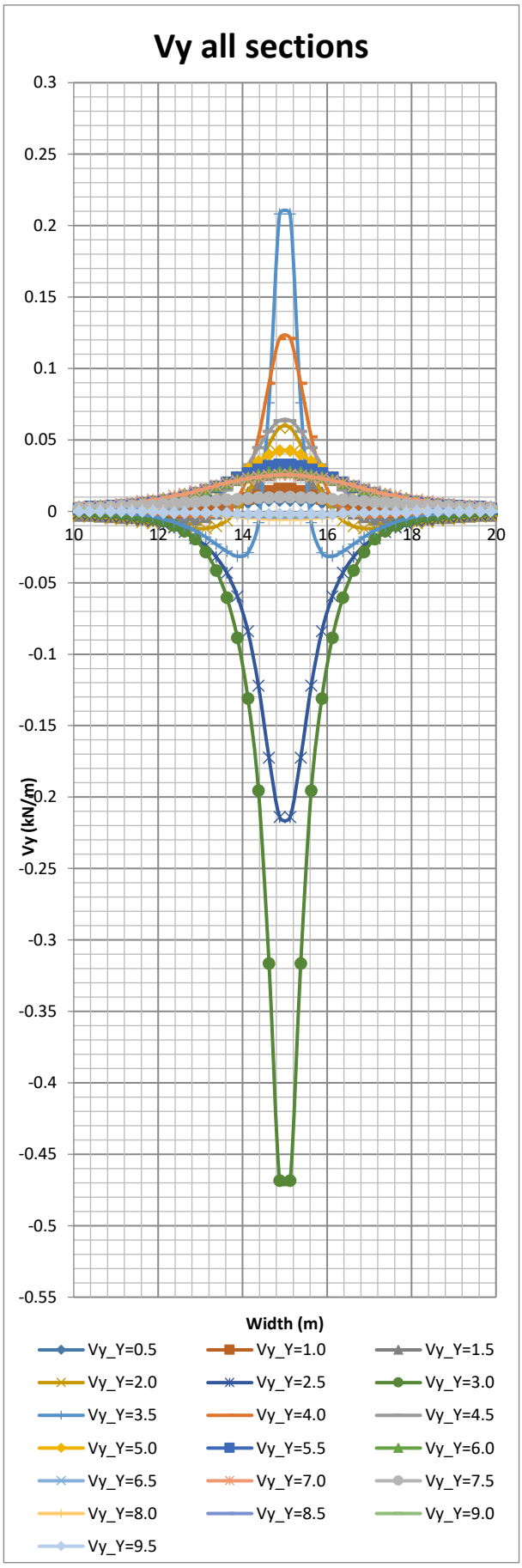
Longitudinal sections when Load at 3m



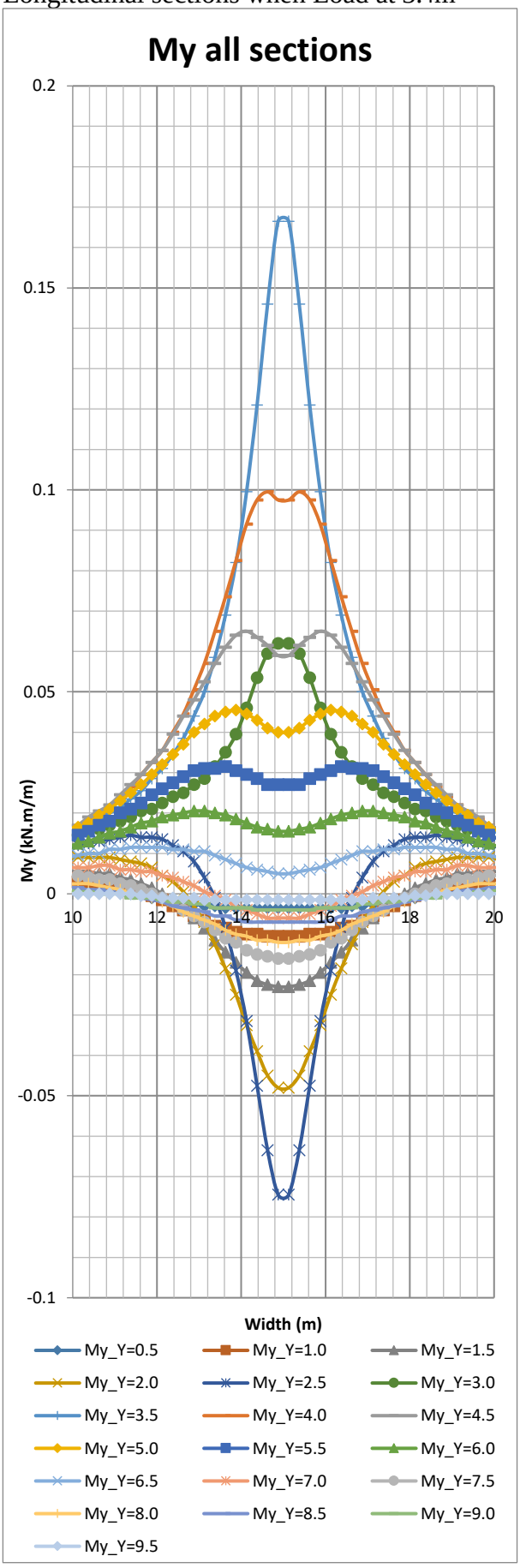


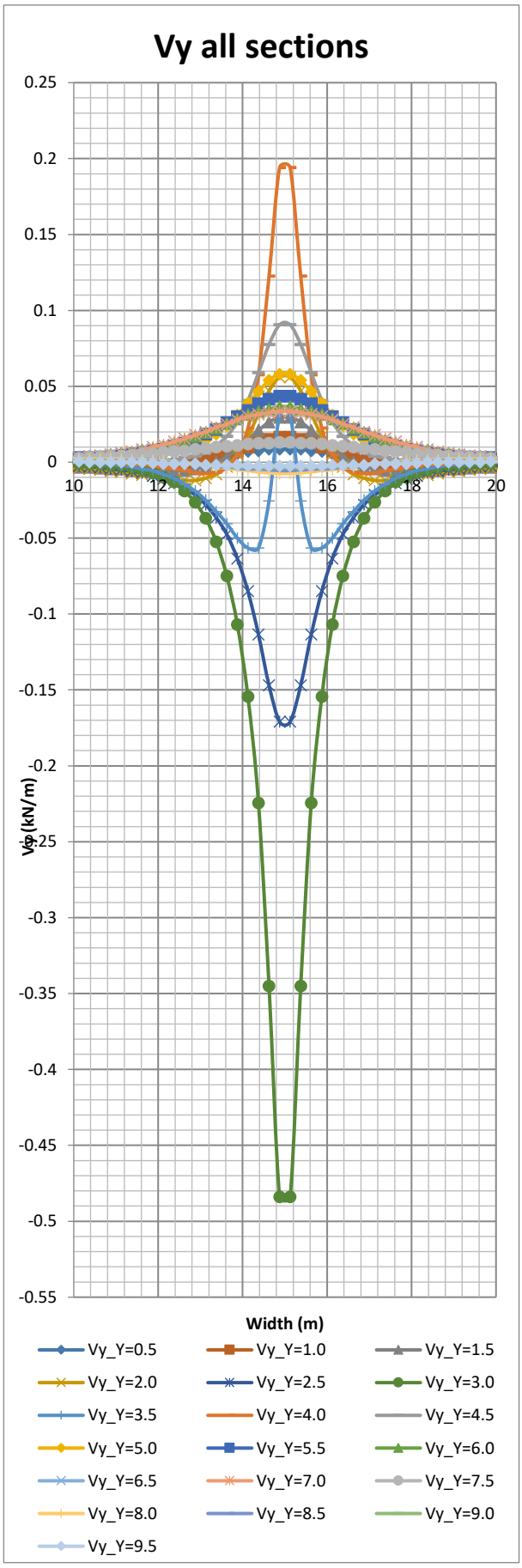
Longitudinal sections when Load at 3.2m



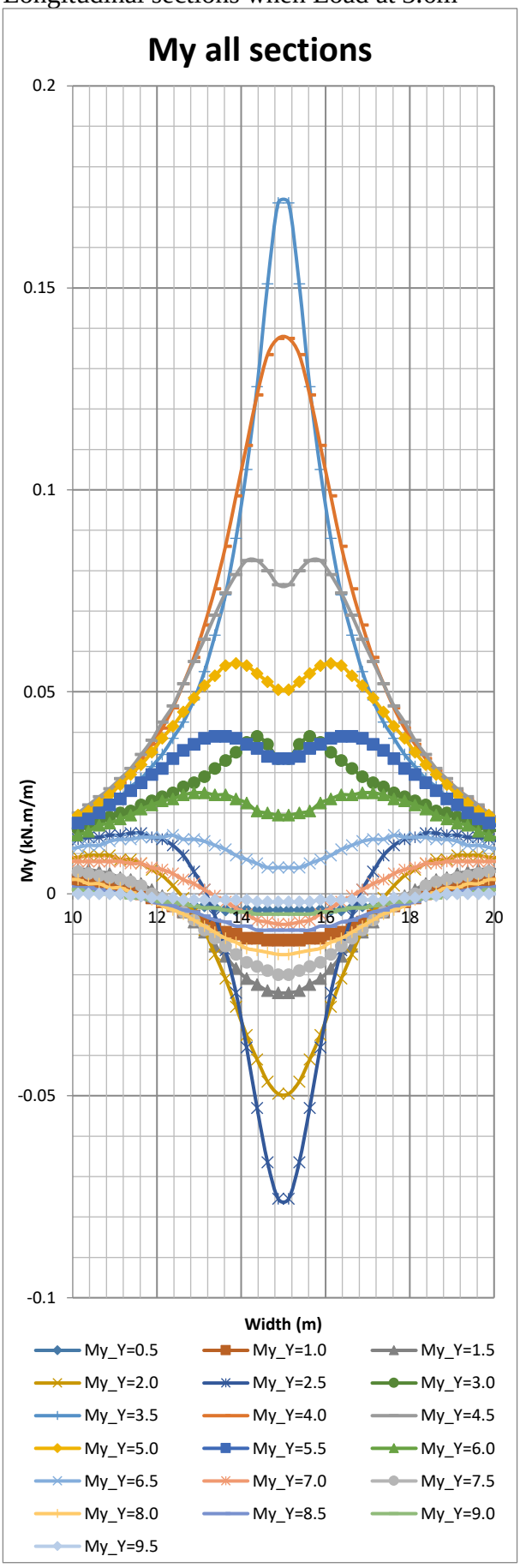


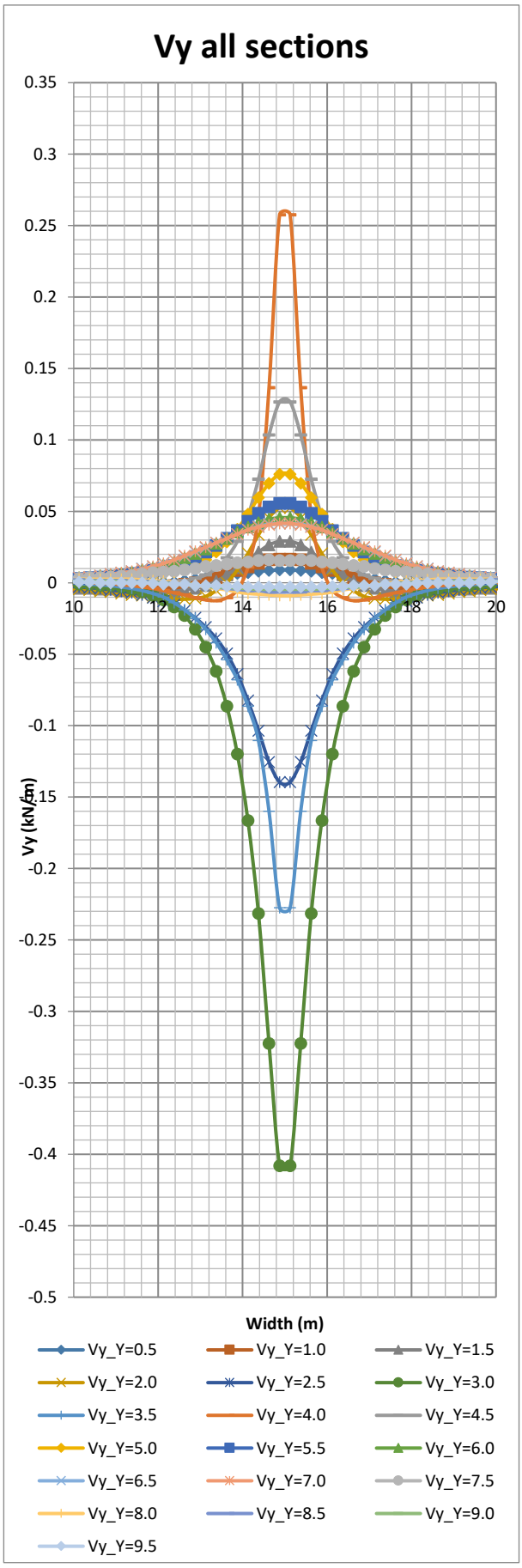
Longitudinal sections when Load at 3.4m



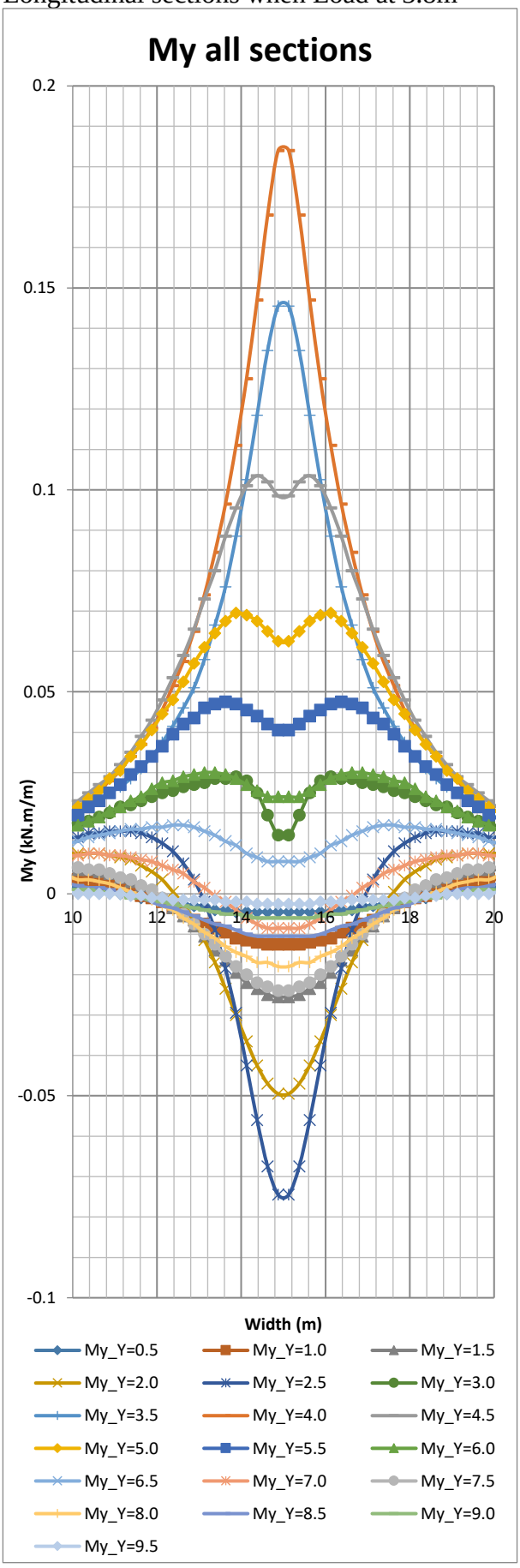


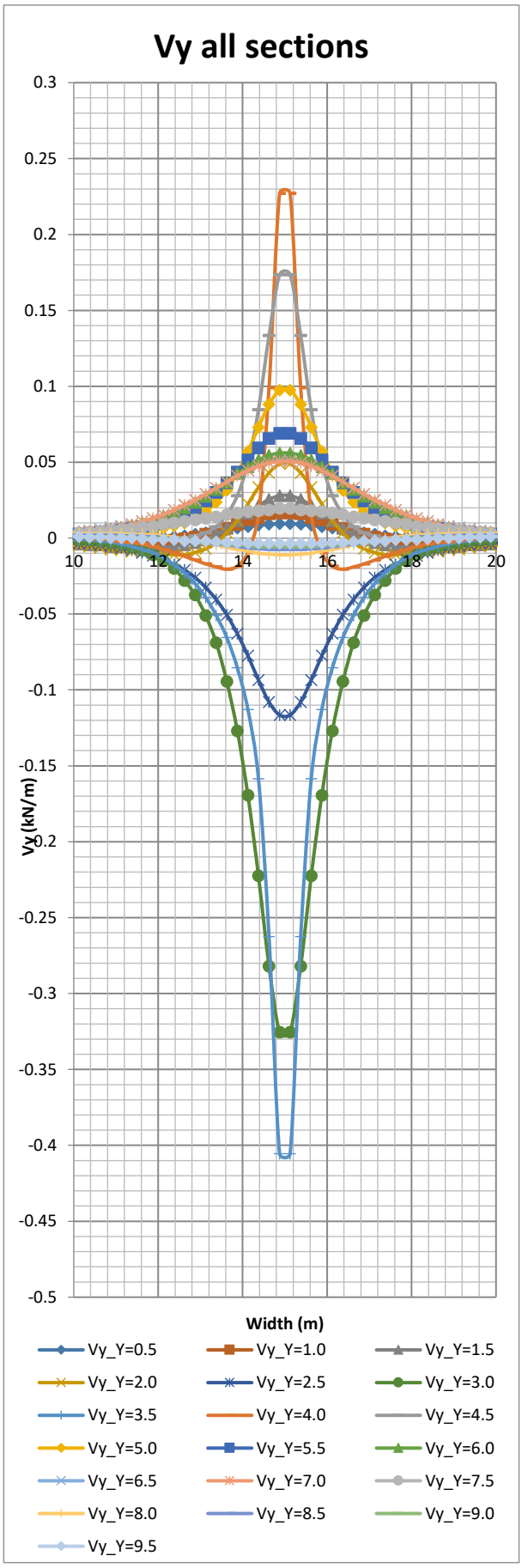
Longitudinal sections when Load at 3.6m



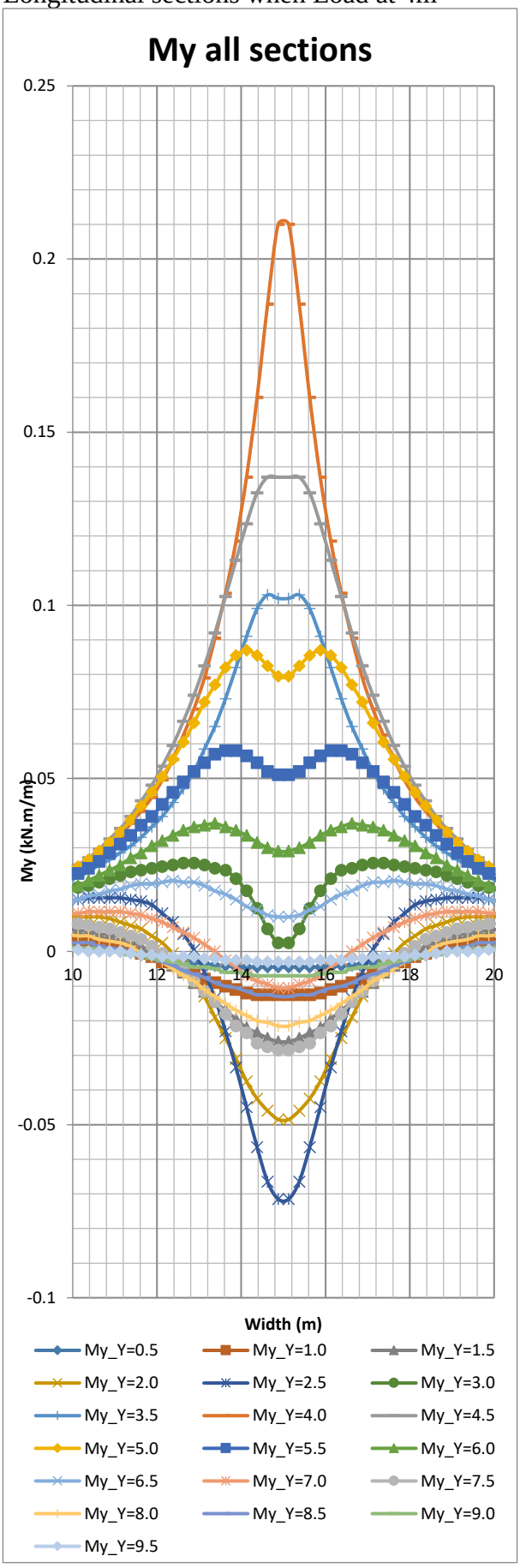


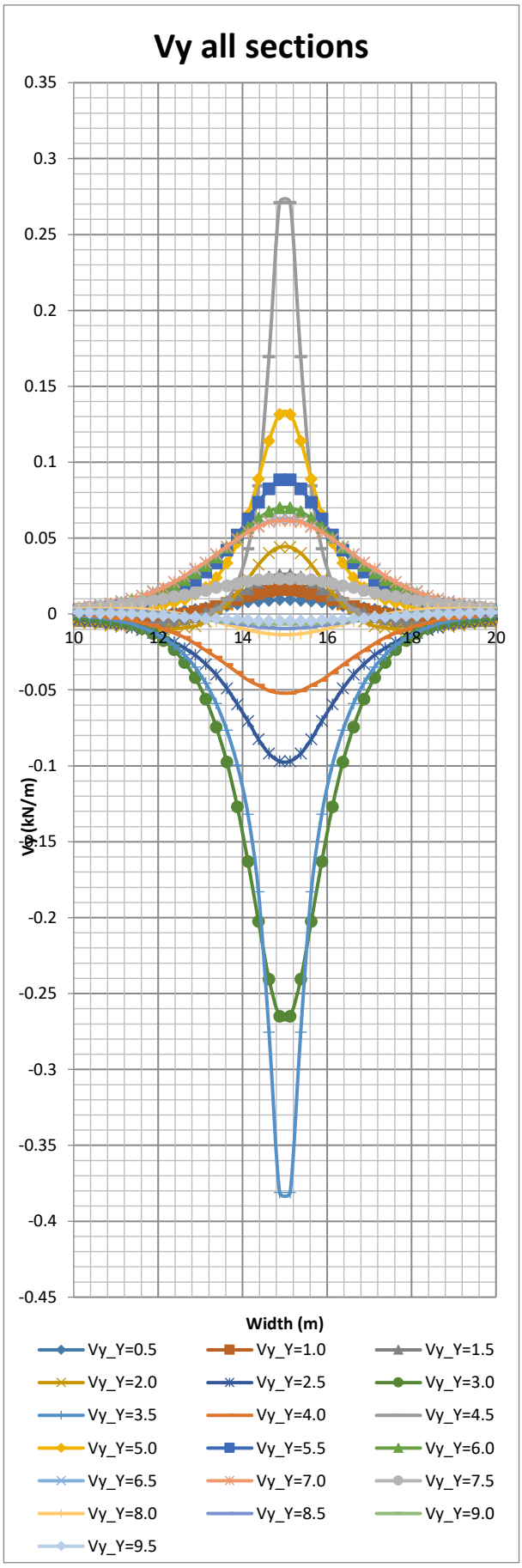
Longitudinal sections when Load at 3.8m



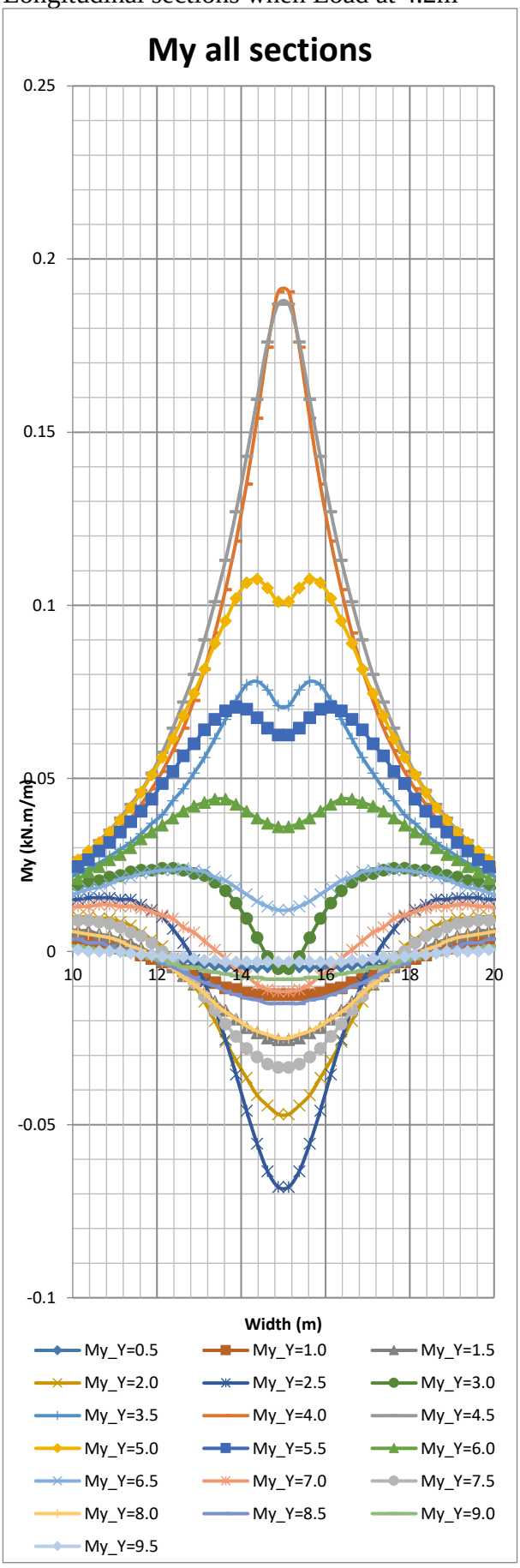


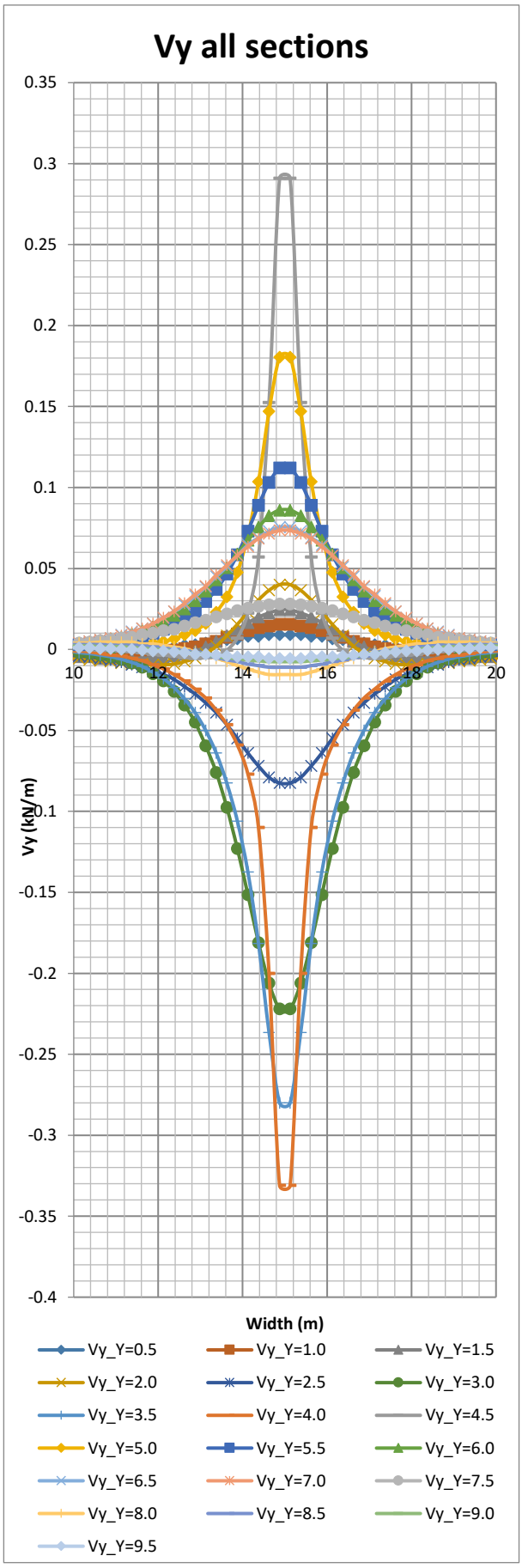
Longitudinal sections when Load at 4m



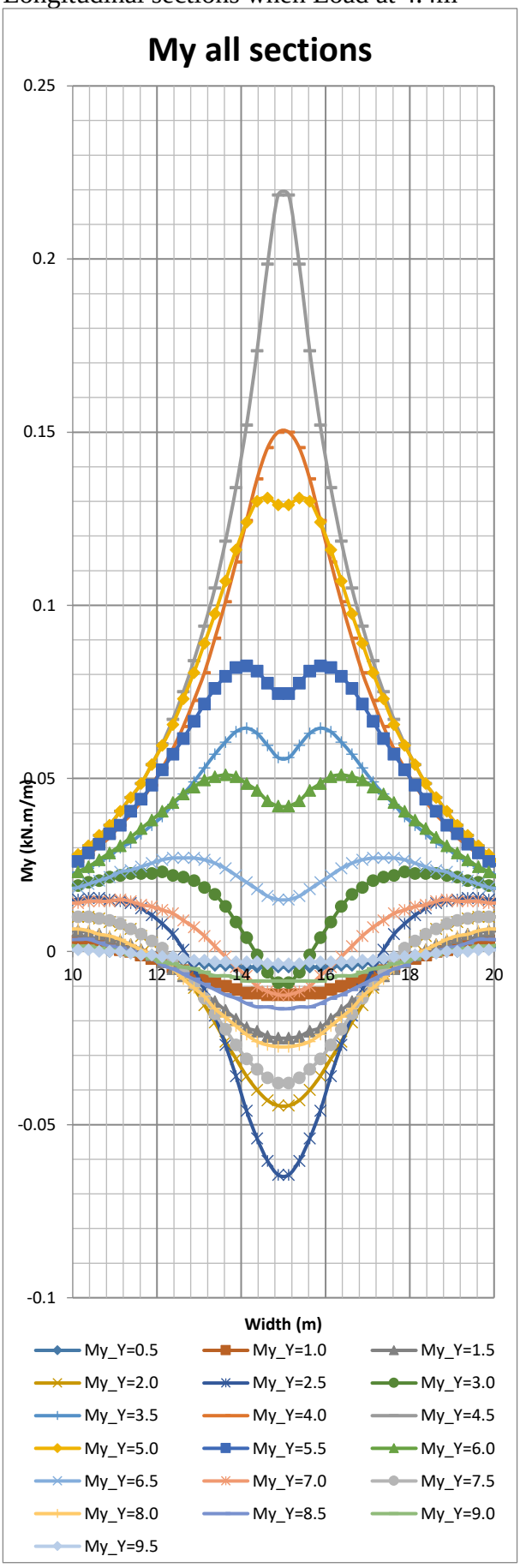


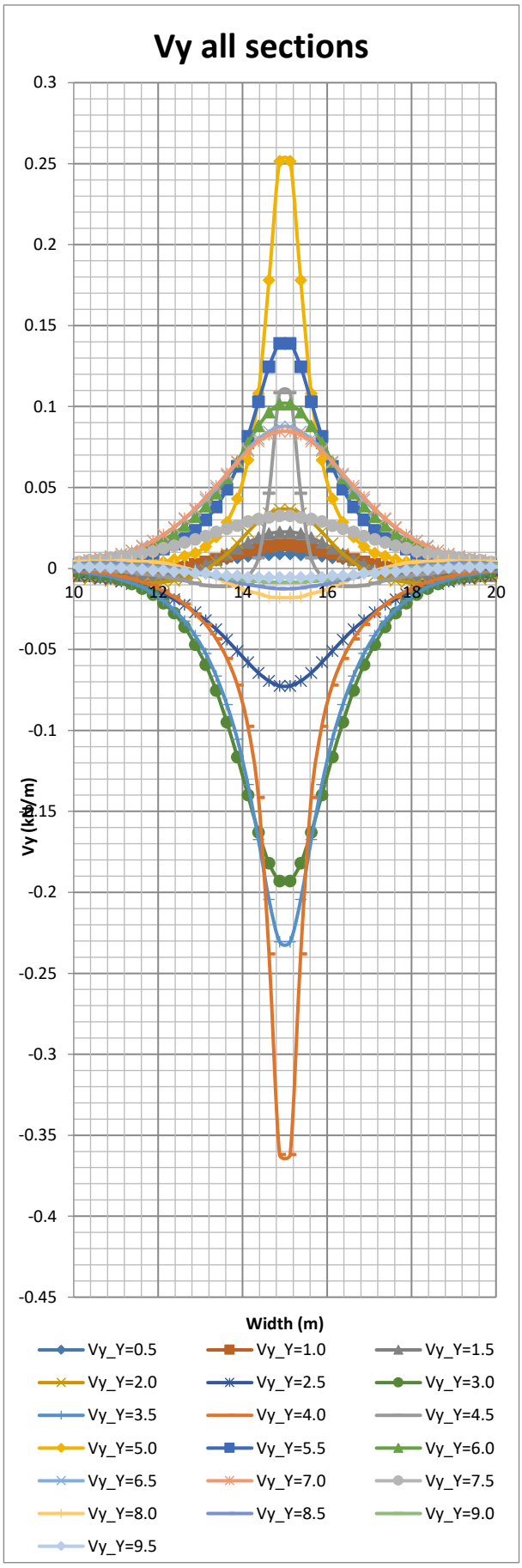
Longitudinal sections when Load at 4.2m



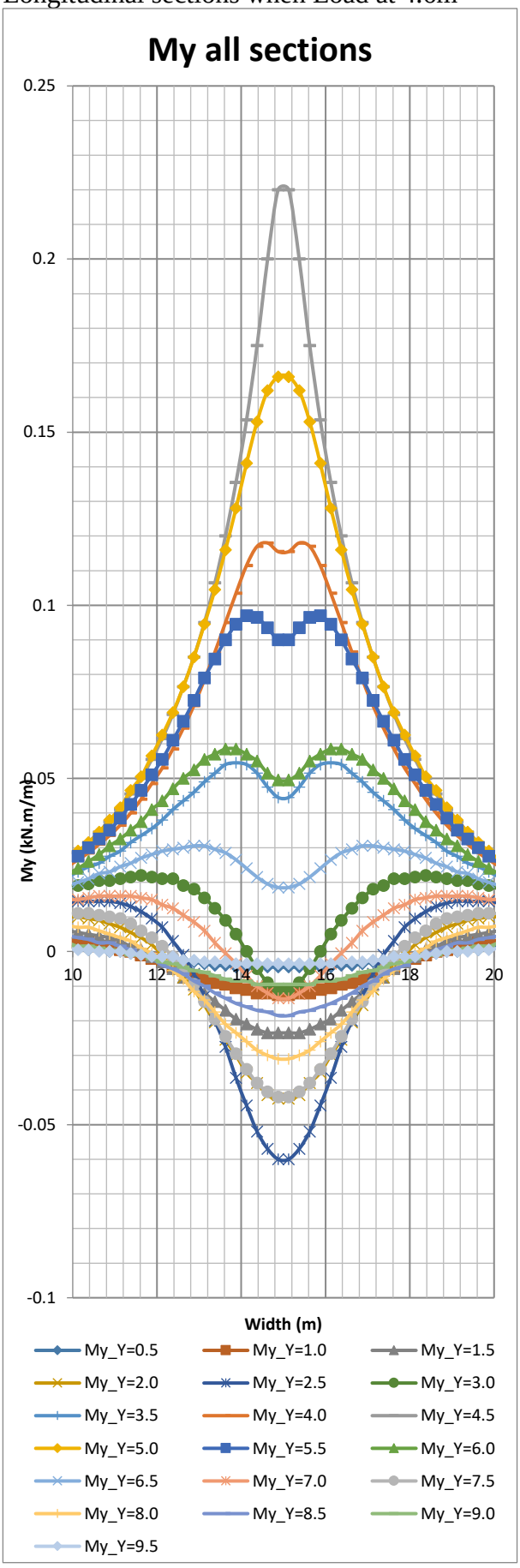


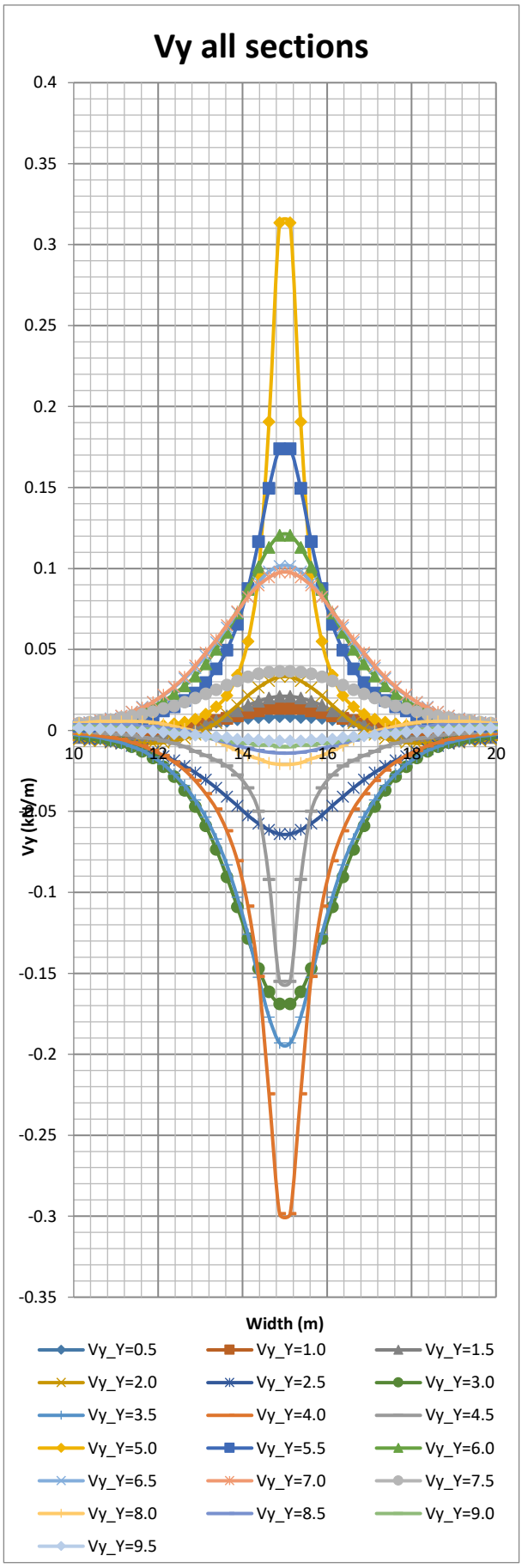
Longitudinal sections when Load at 4.4m



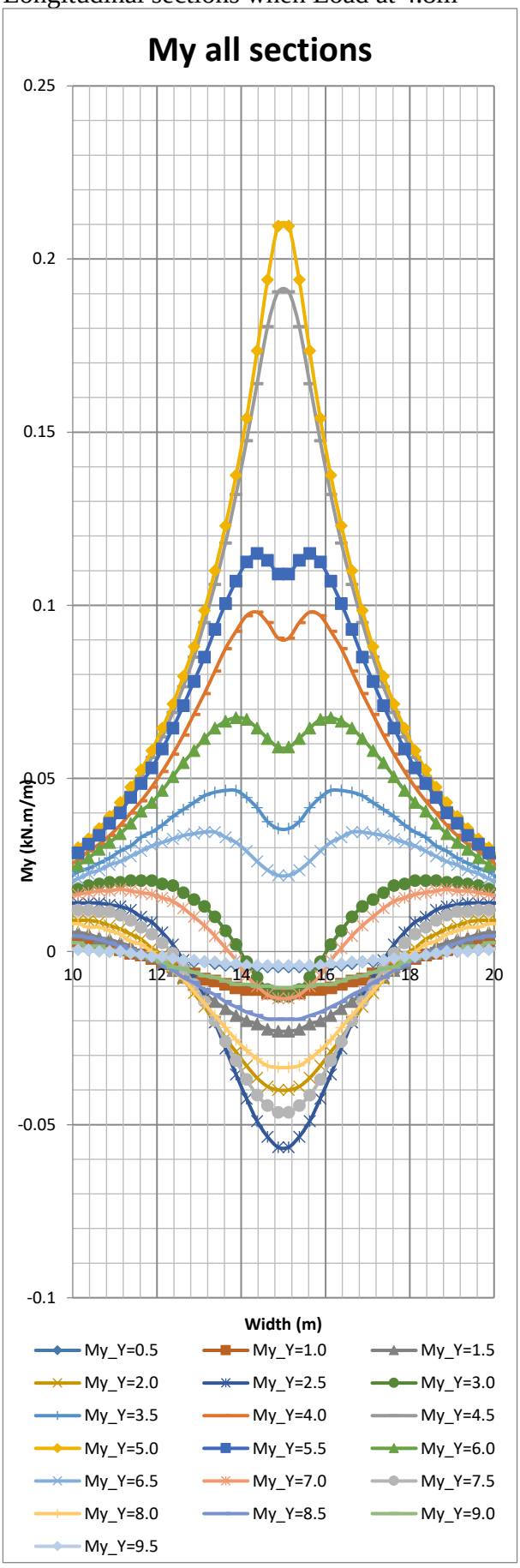


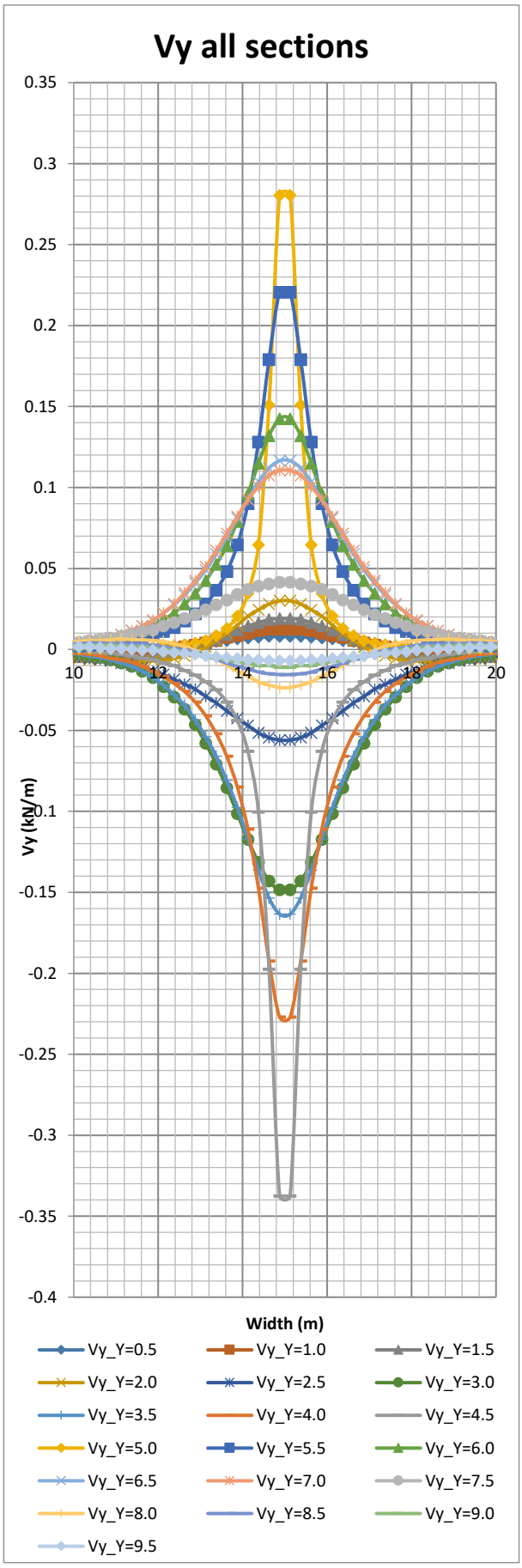
Longitudinal sections when Load at 4.6m



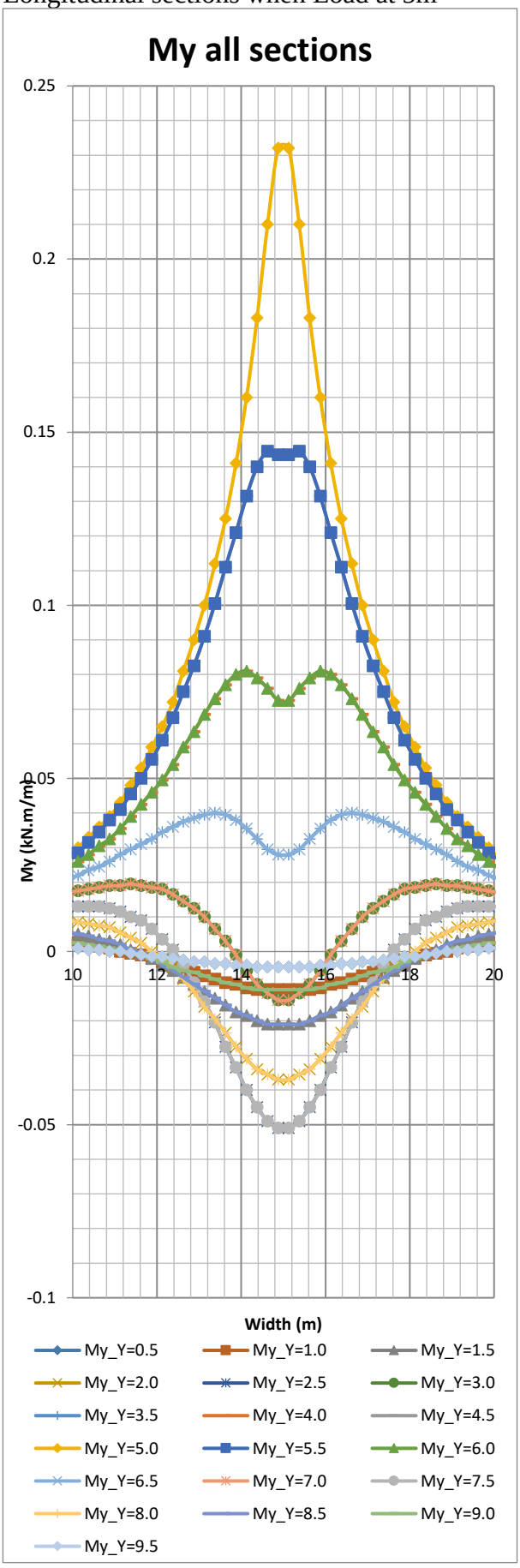


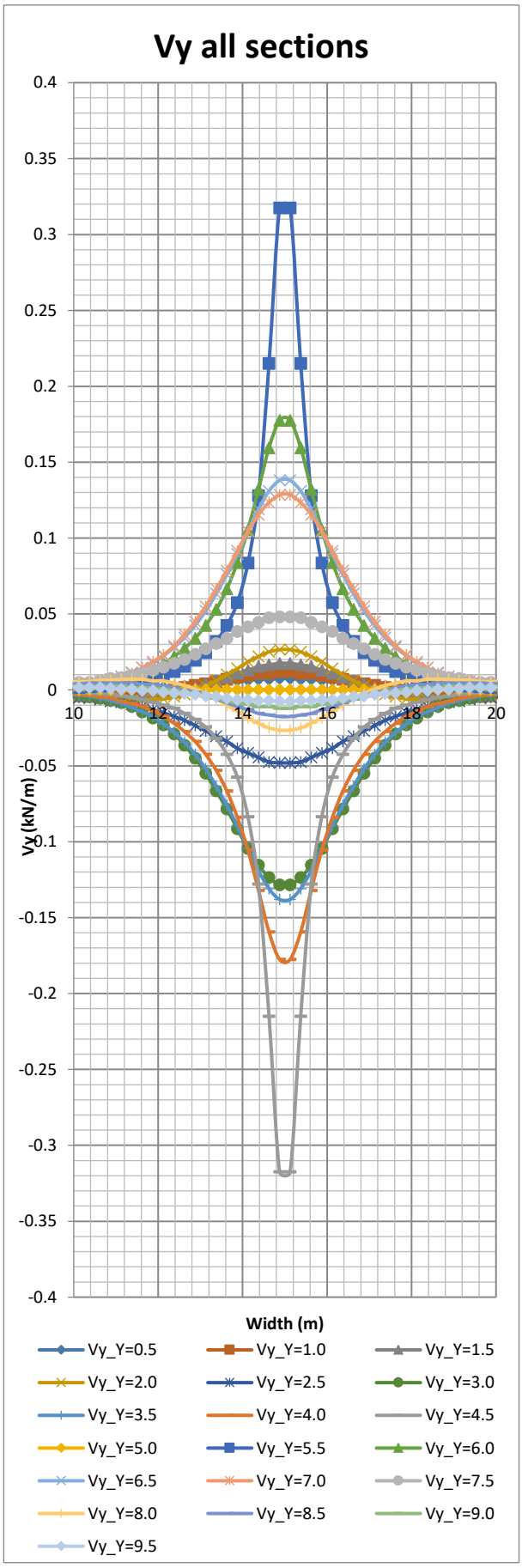
Longitudinal sections when Load at 4.8m



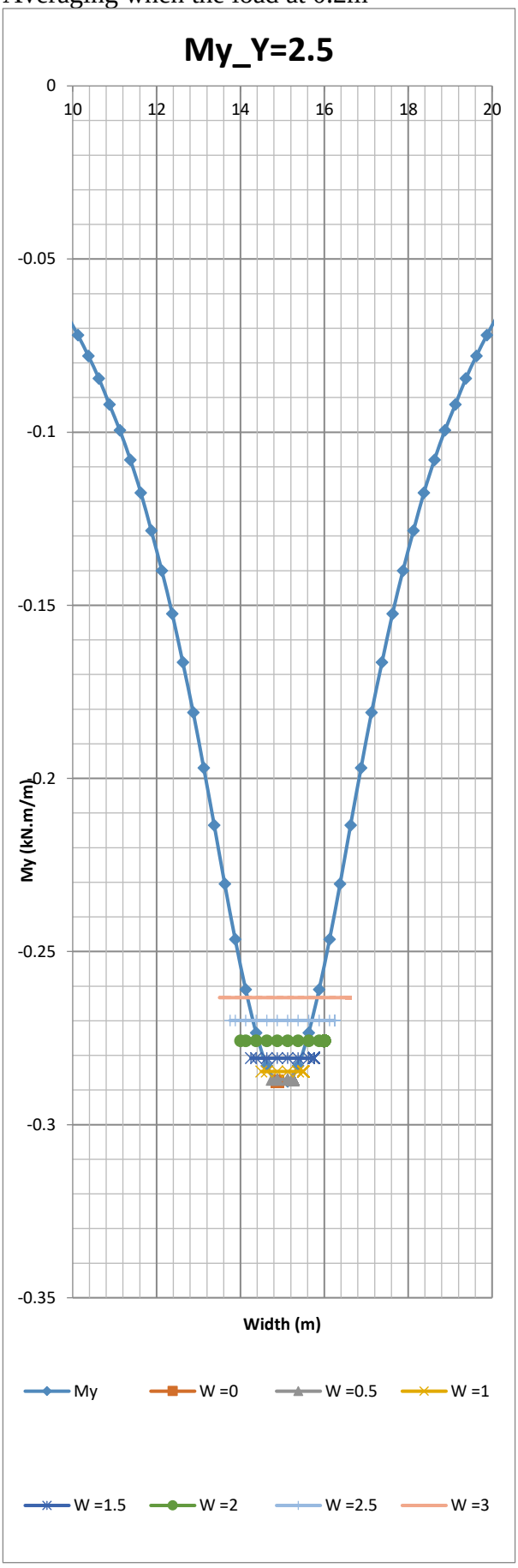


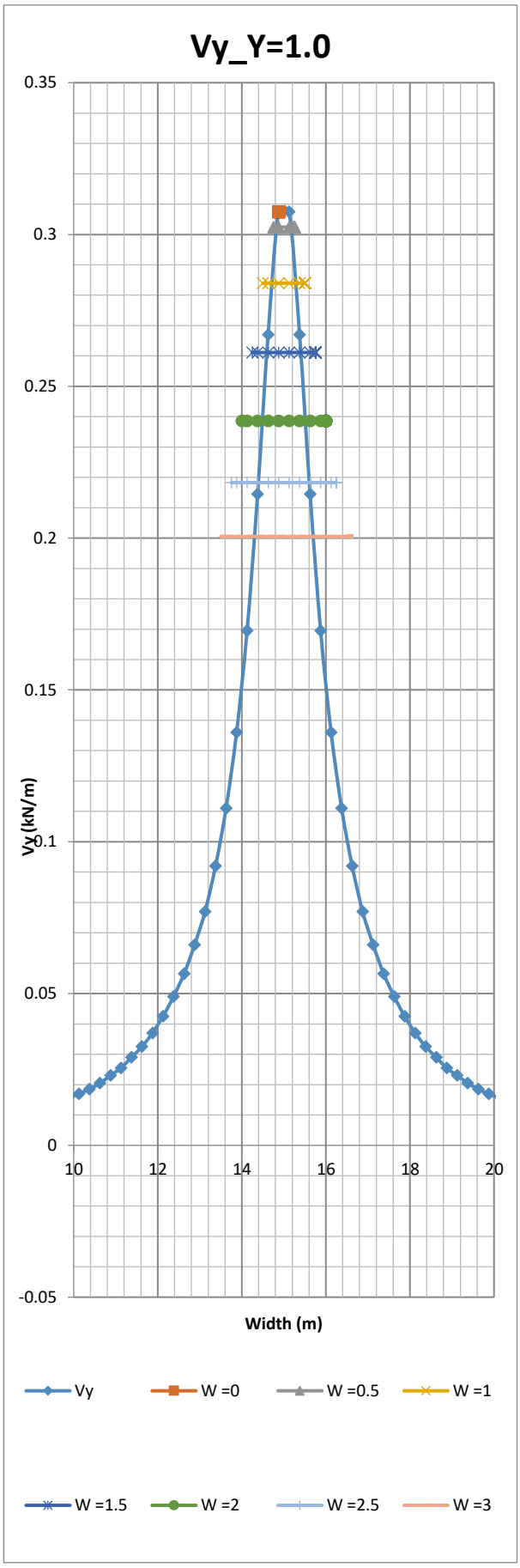
Longitudinal sections when Load at 5m



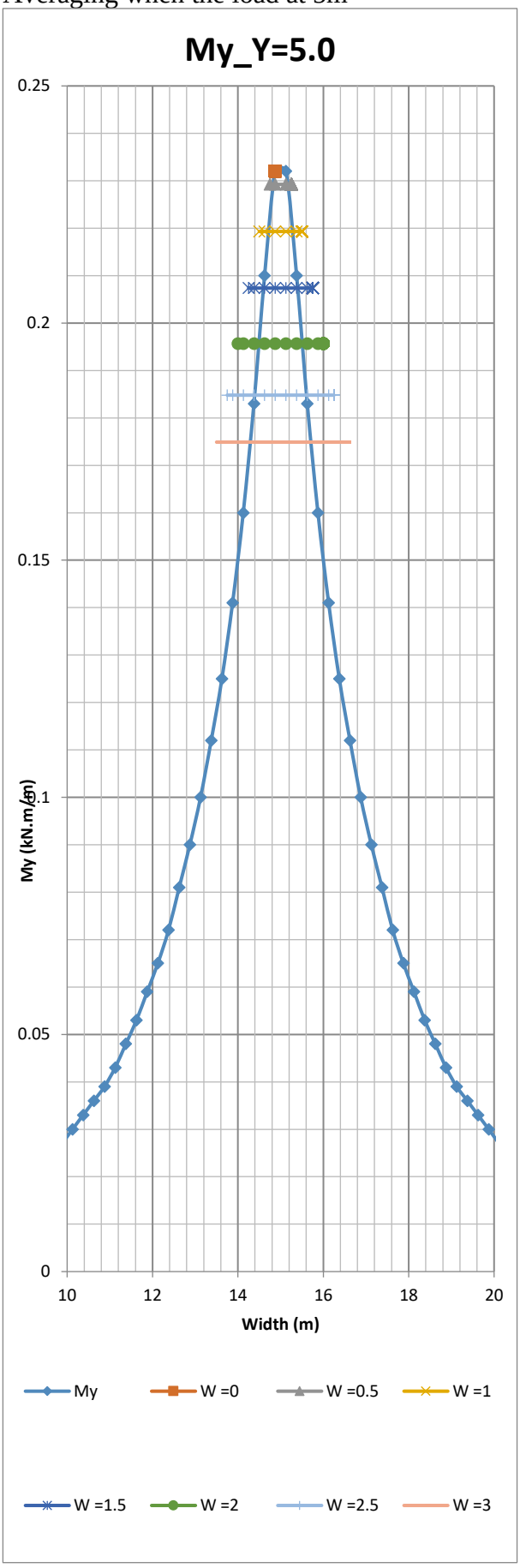


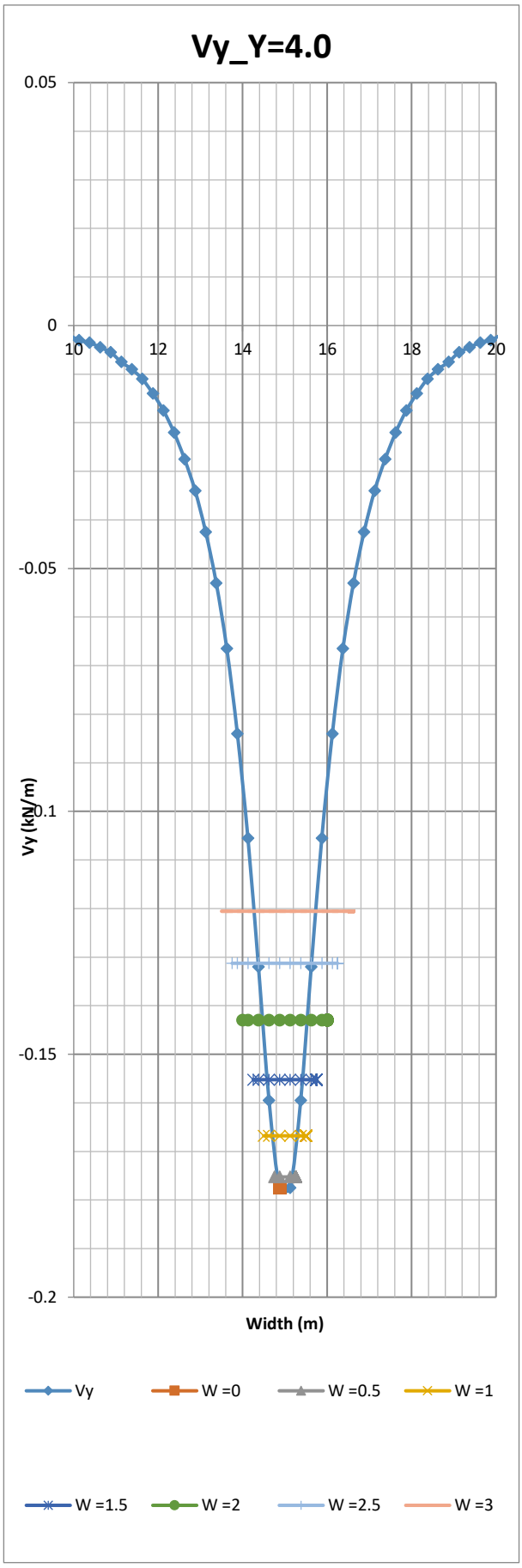
Averaging when the load at 0.2m





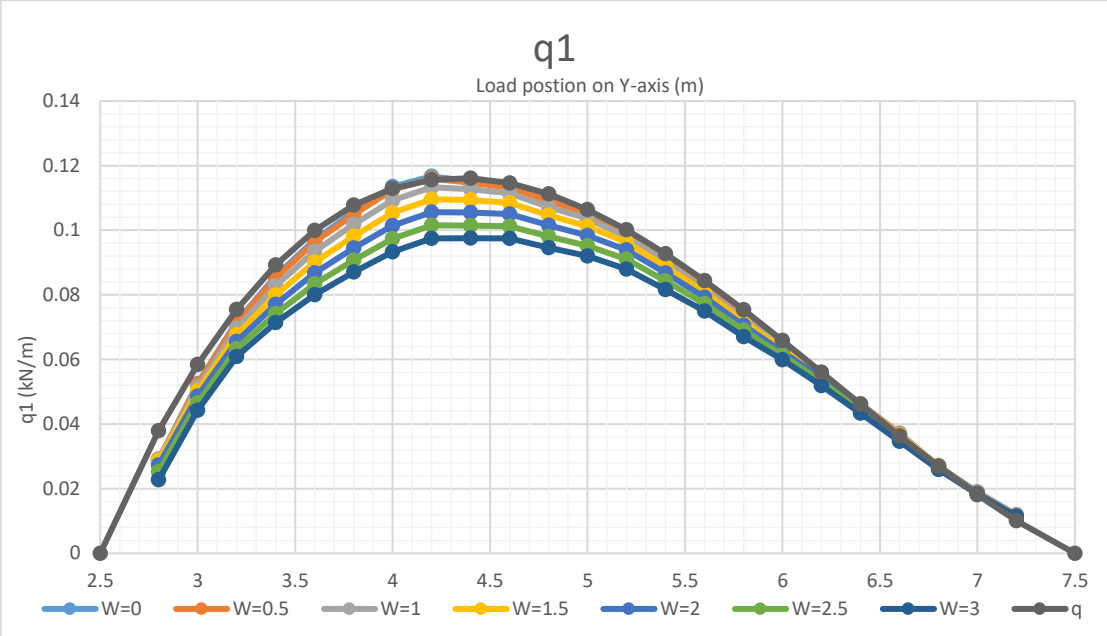
Averaging when the load at 5m

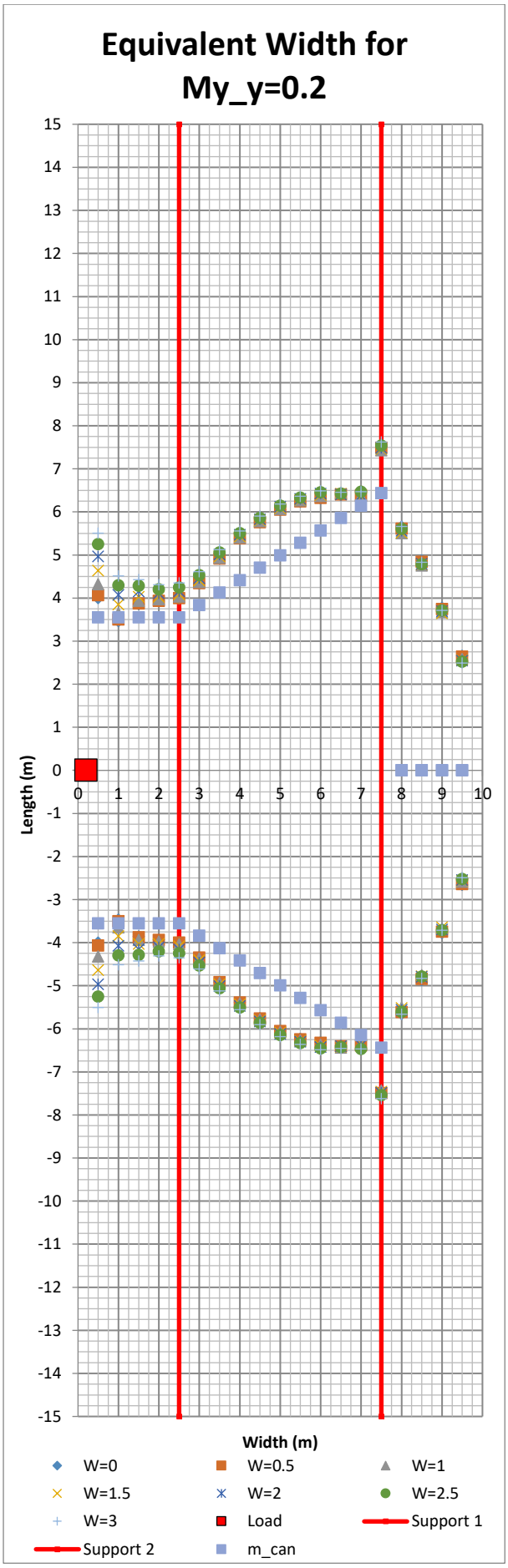


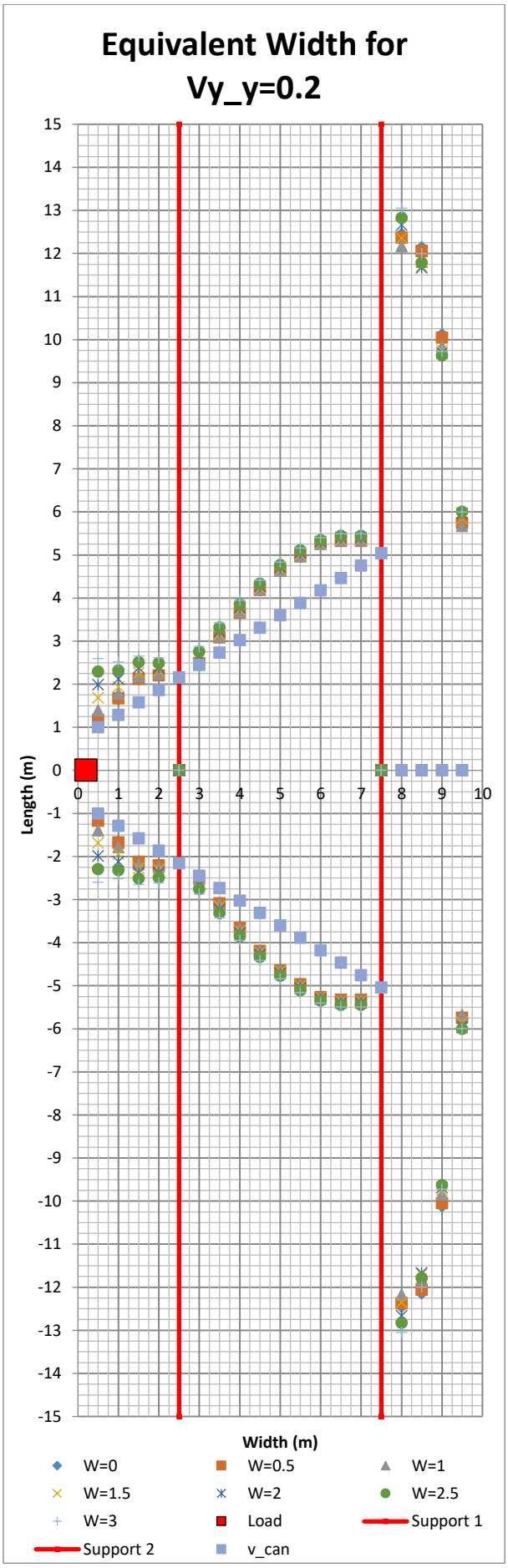


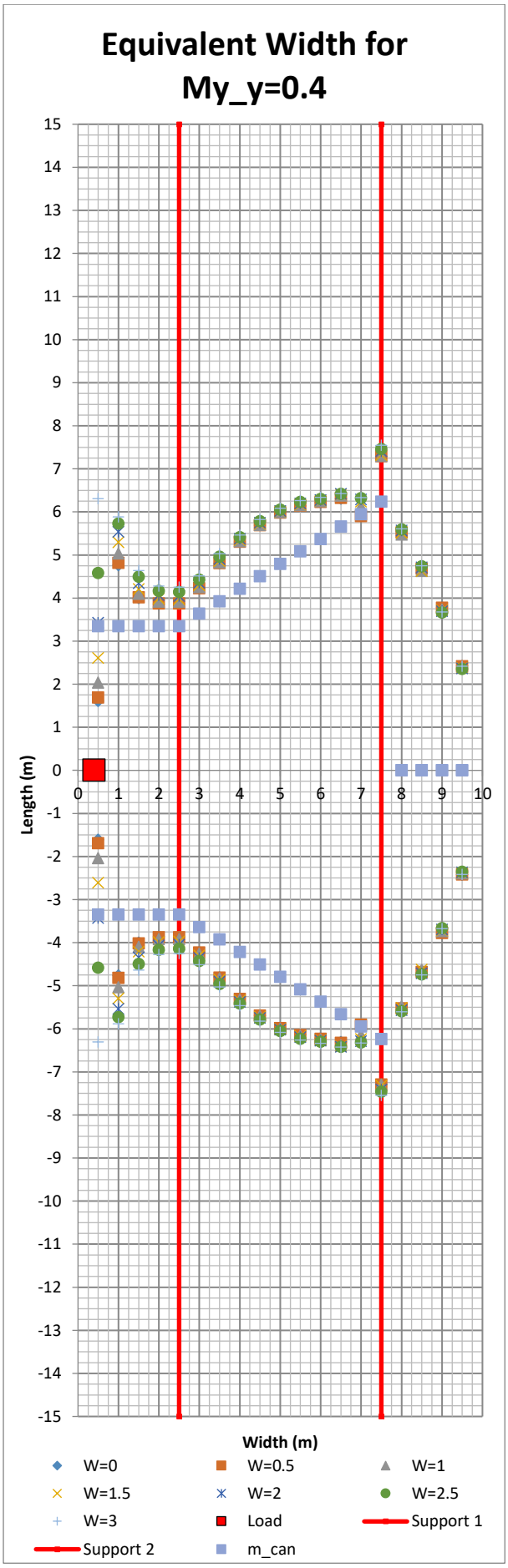
11 APPENDIX C: Parametric study results

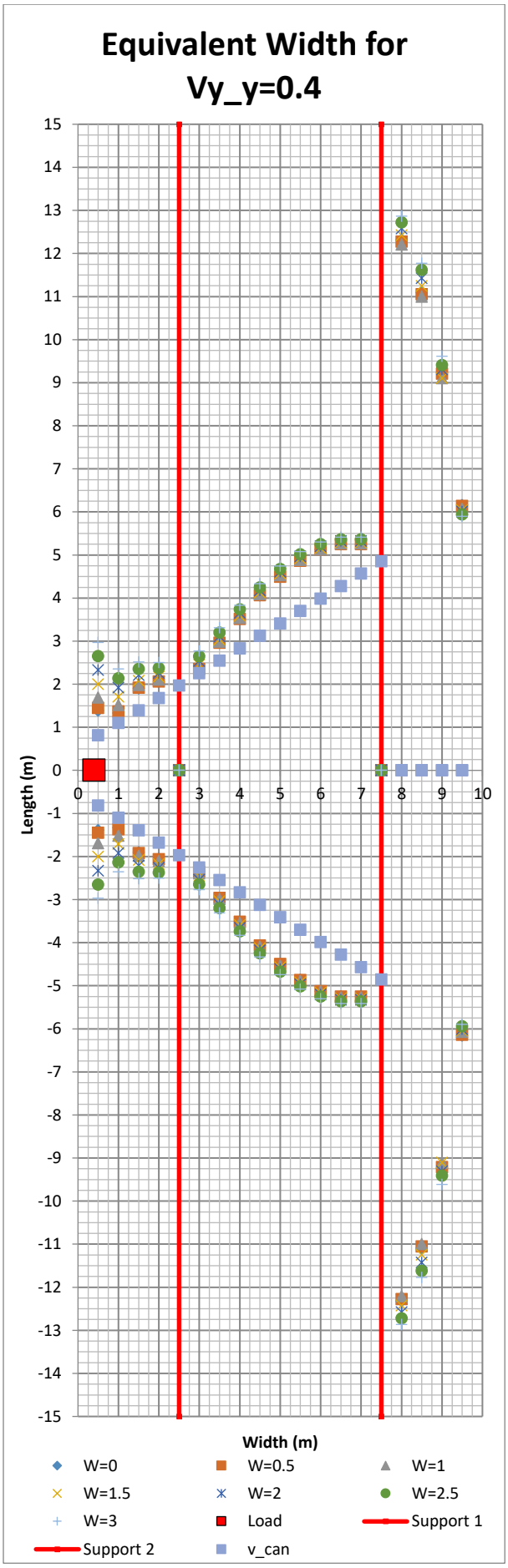
11.1 First Variation Constant thickness $t = 0.45m$

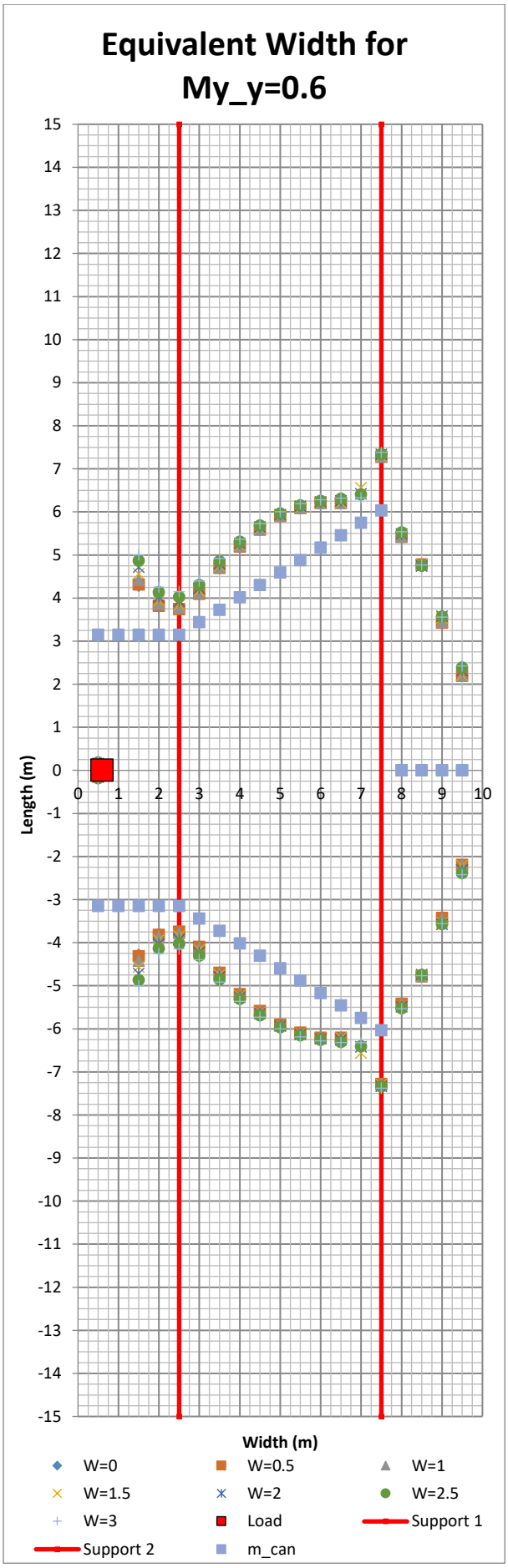


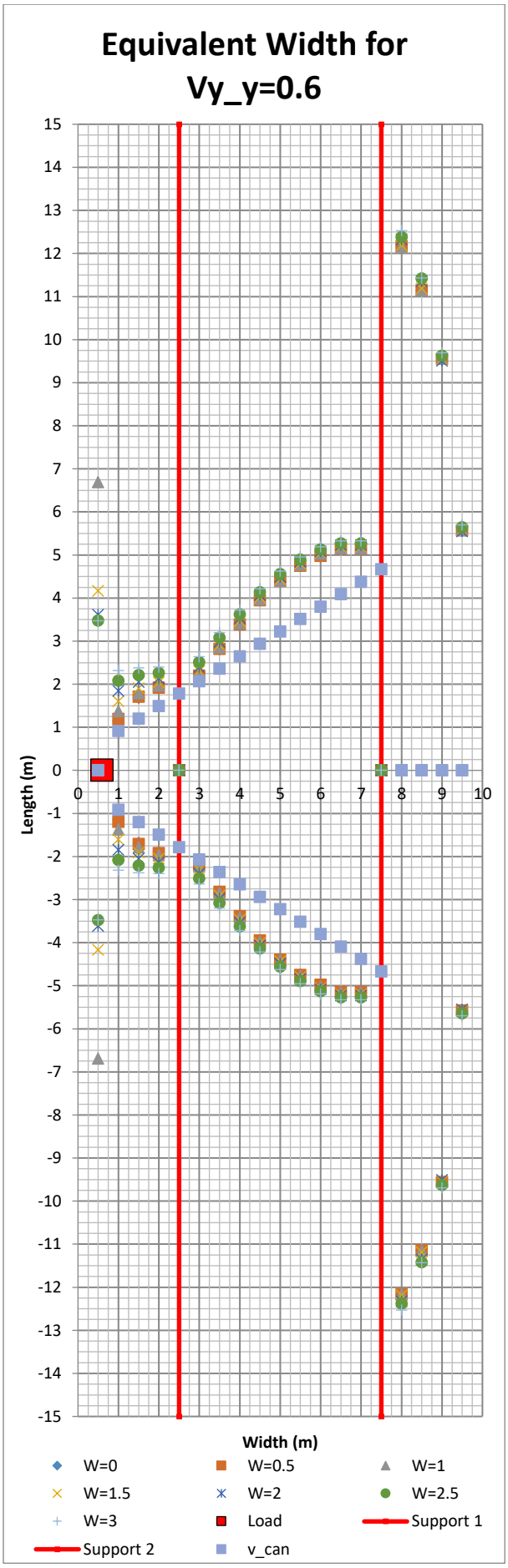

Equivalent Width for My_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

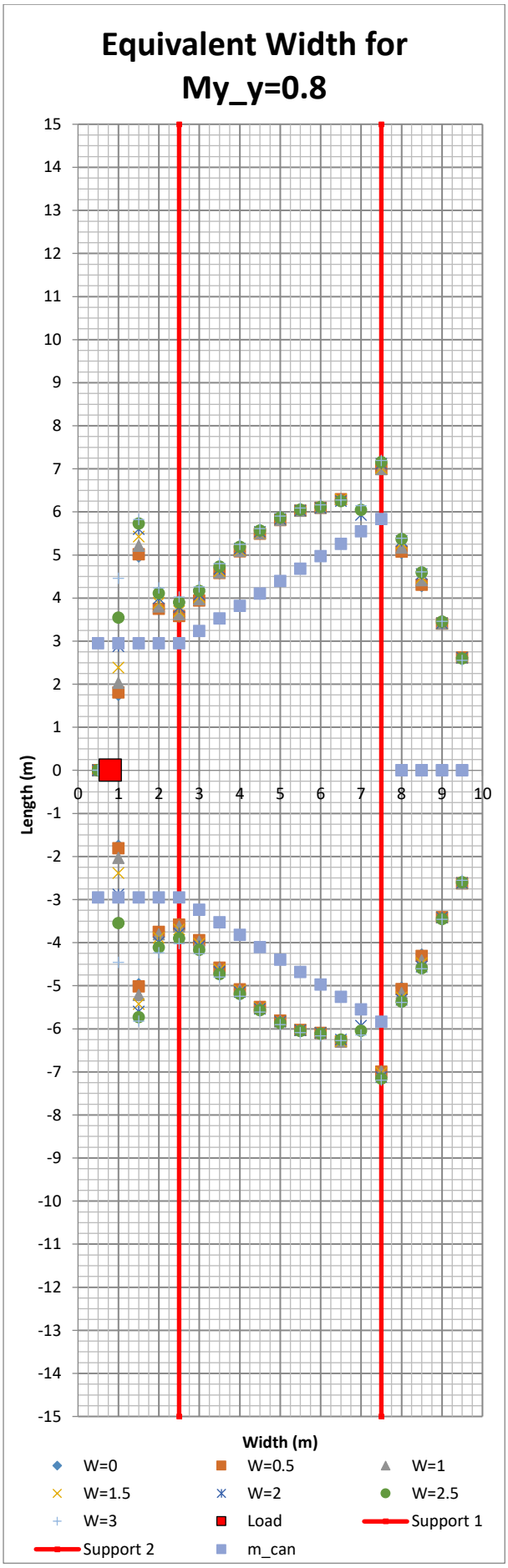

Equivalent Width for Vy_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

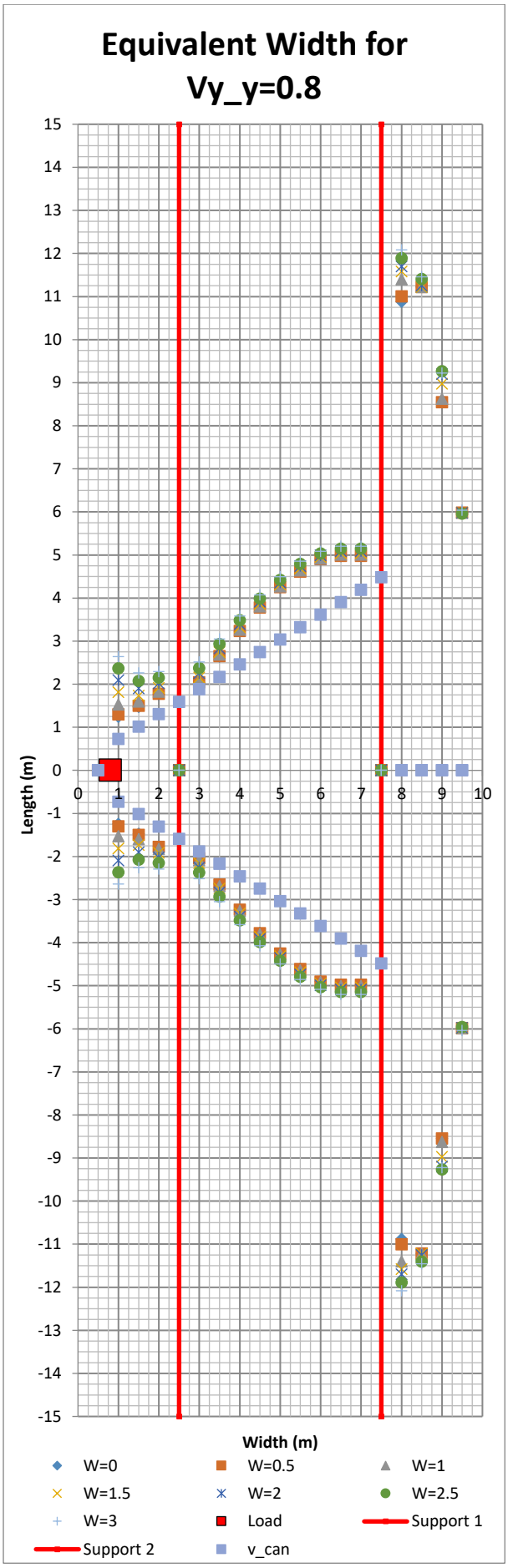

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

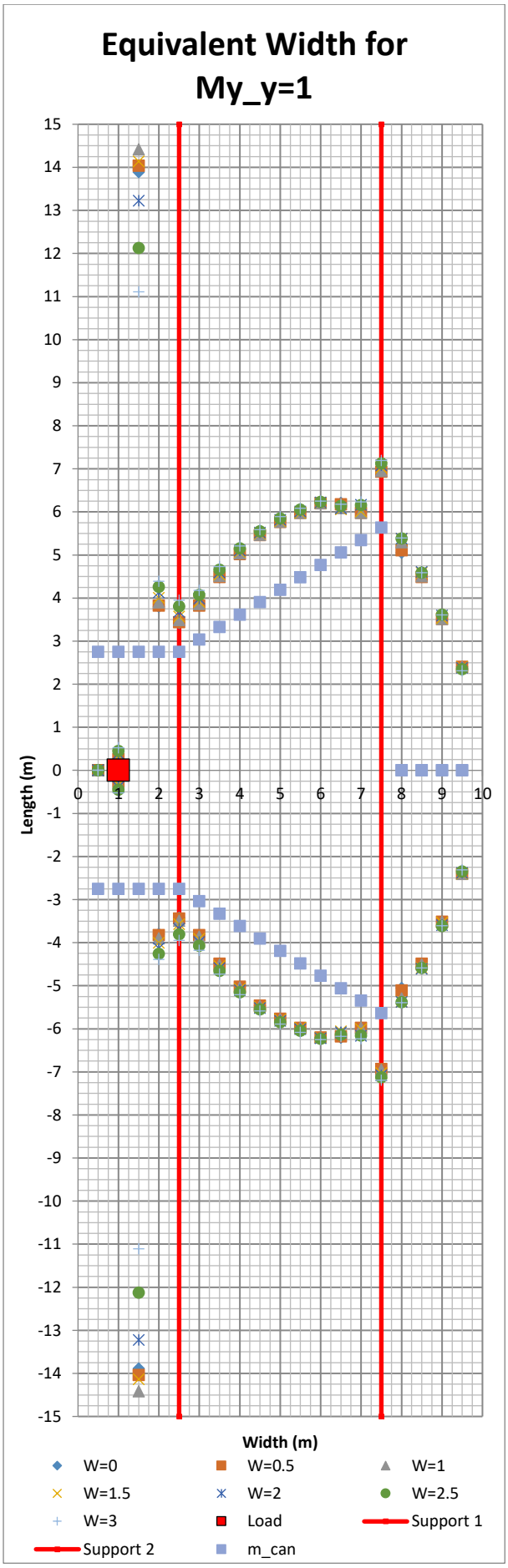

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

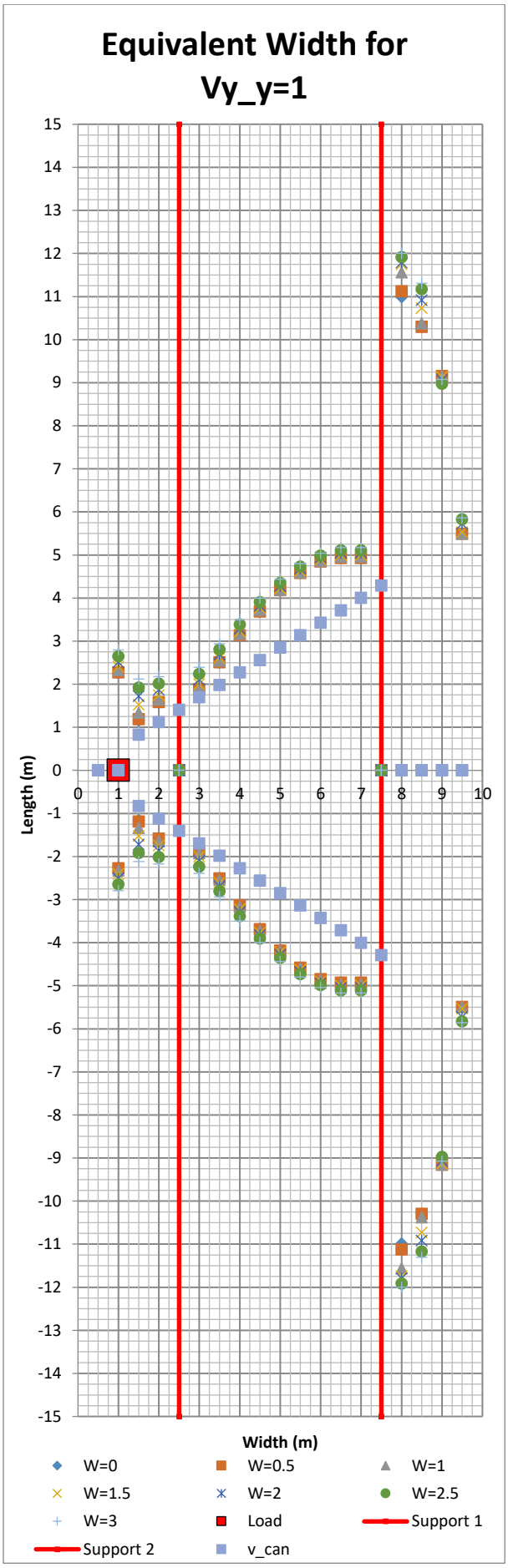

Equivalent Width for My_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

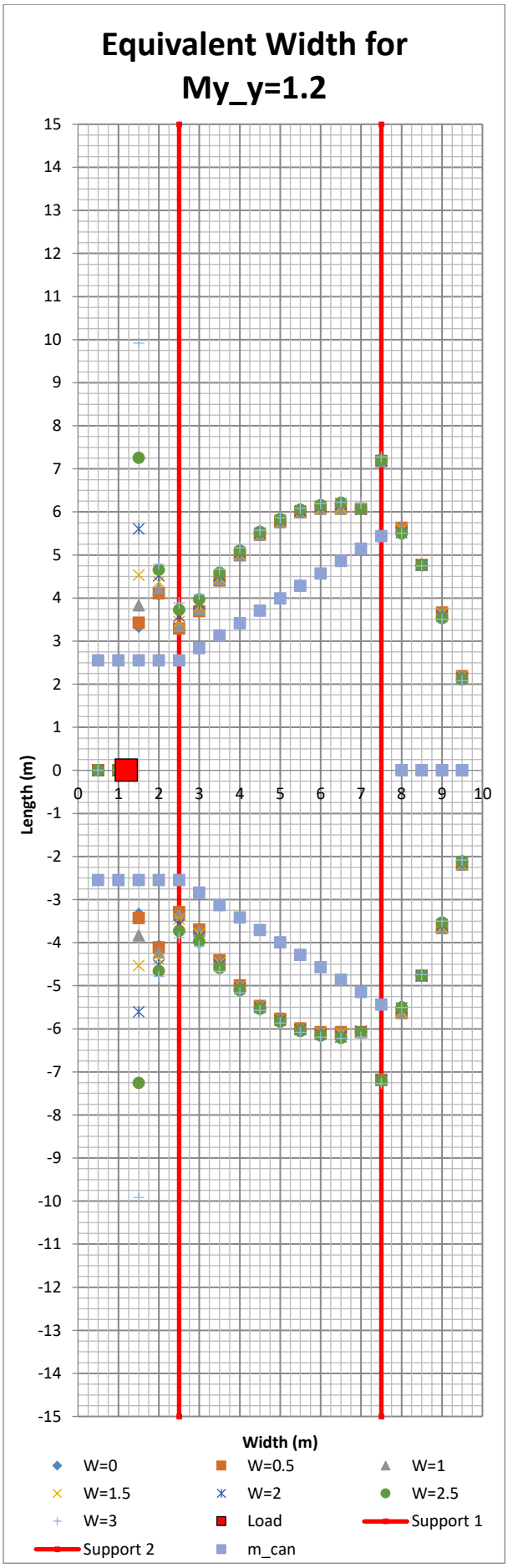

Equivalent Width for Vy_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

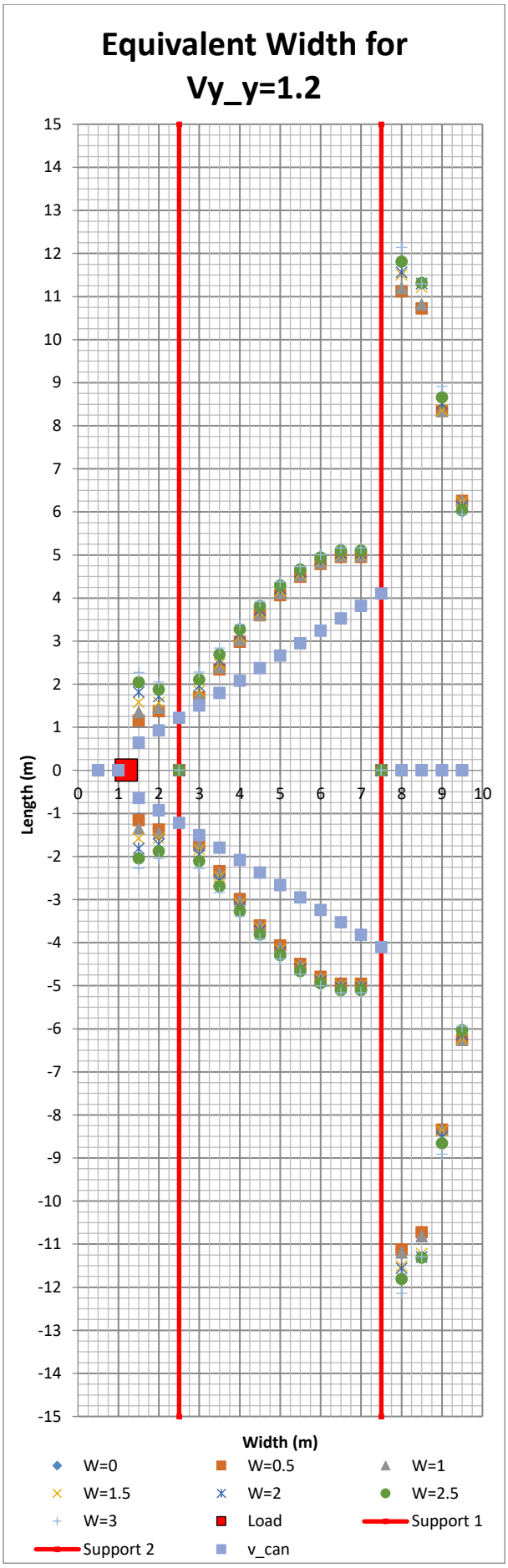

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

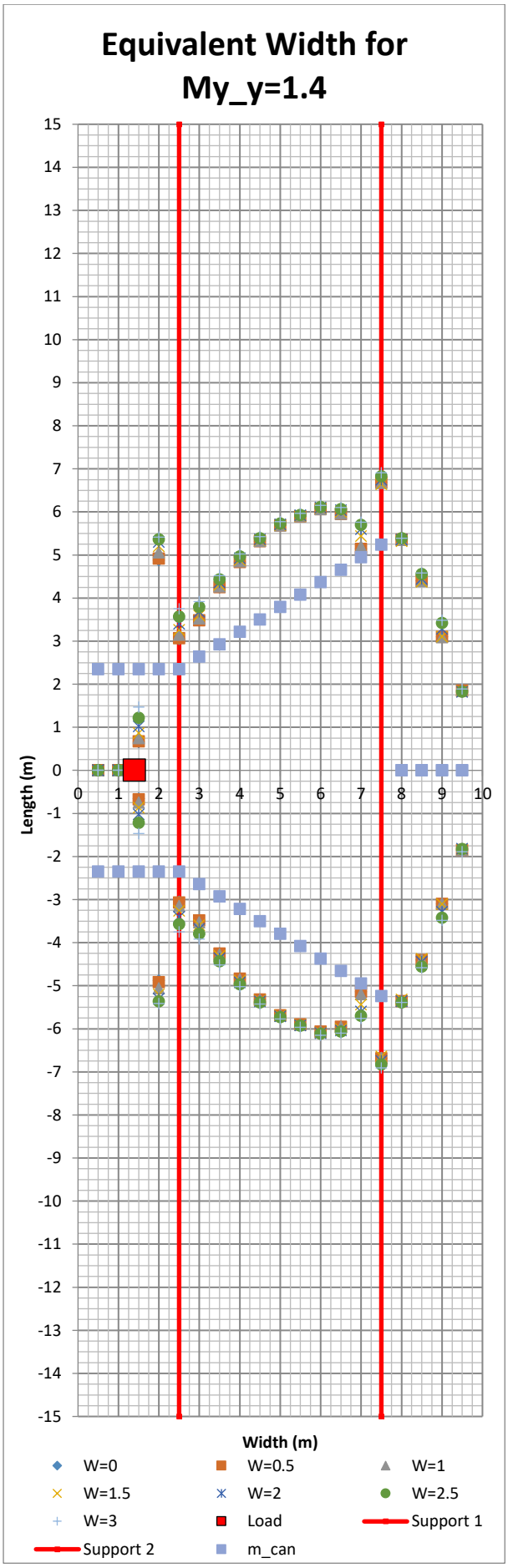

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

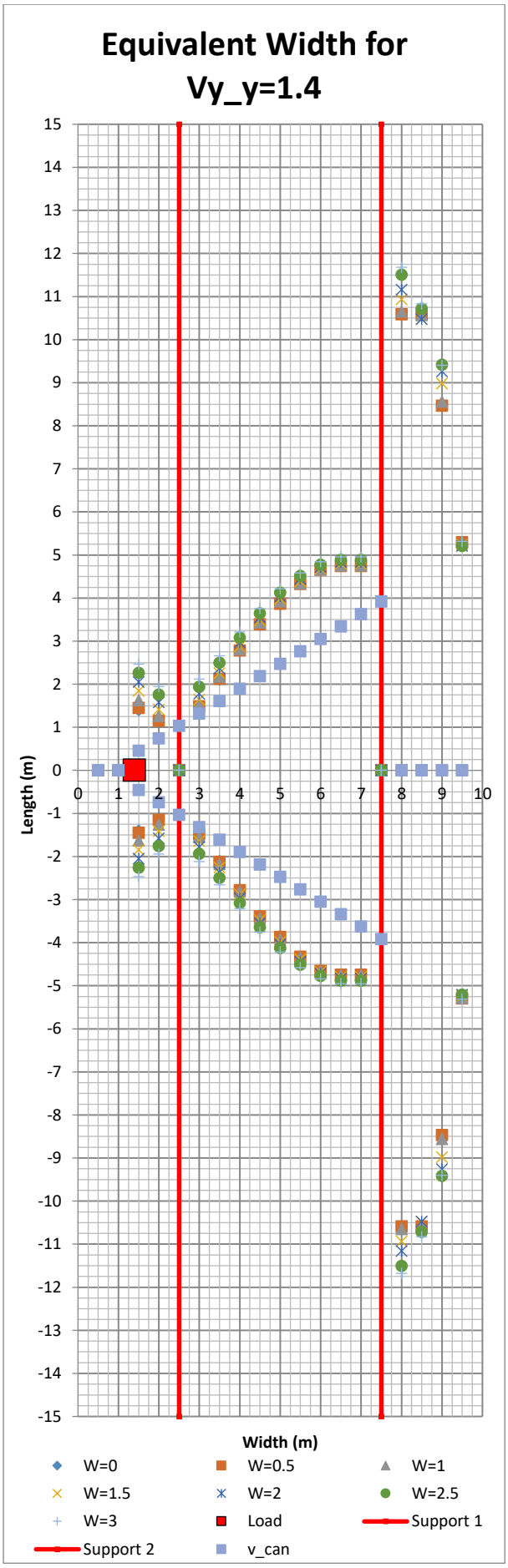

Equivalent Width for My_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

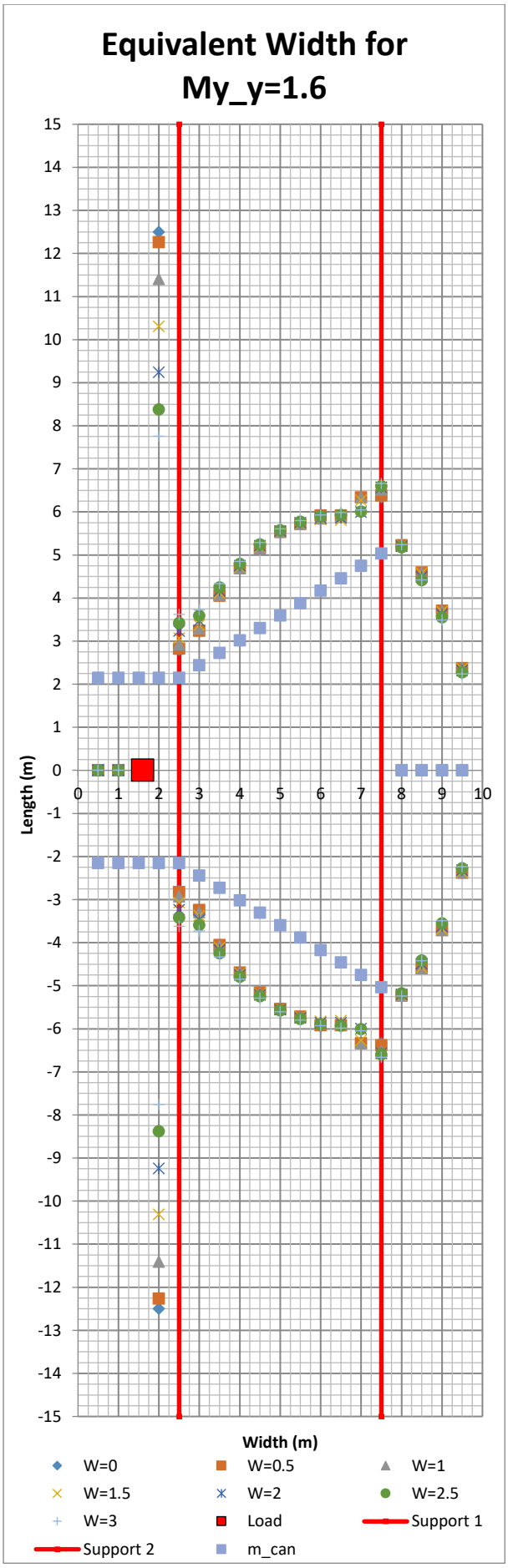

Equivalent Width for Vy_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

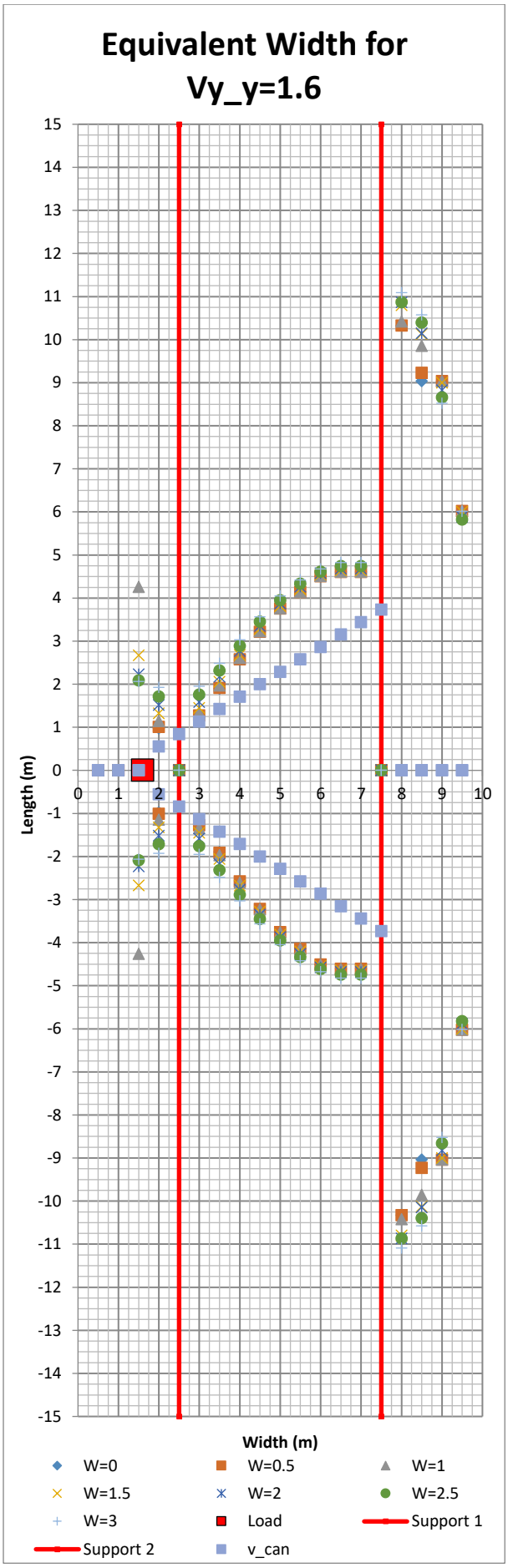

Equivalent Width for My_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

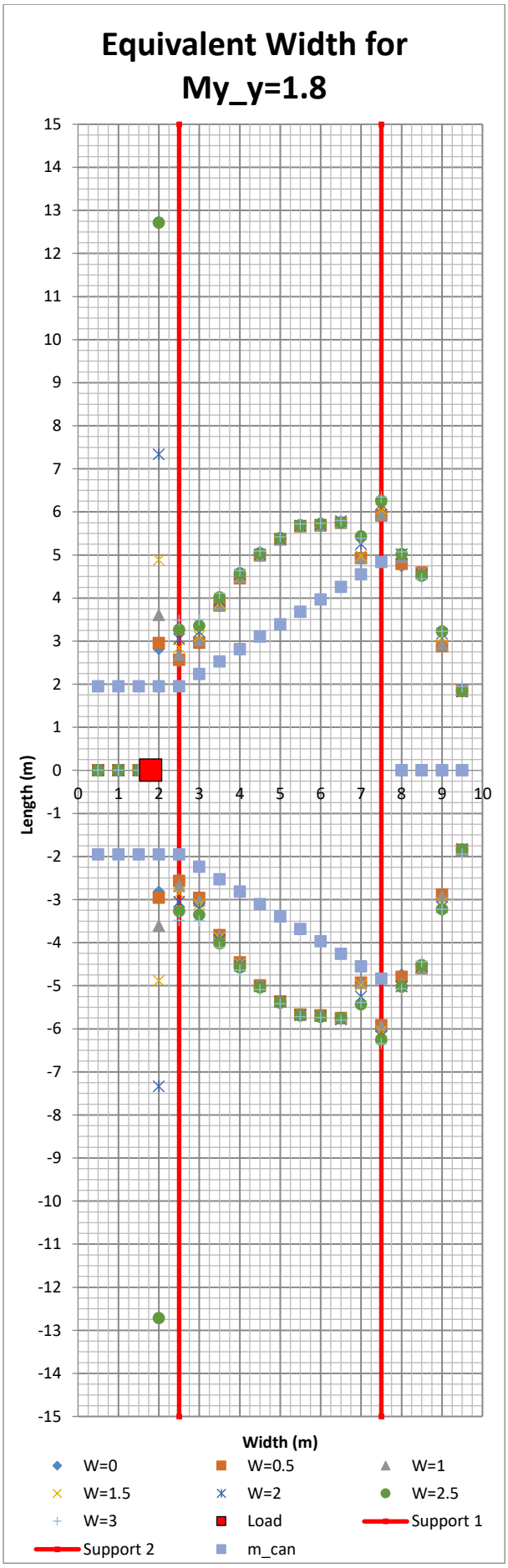

Equivalent Width for Vy_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

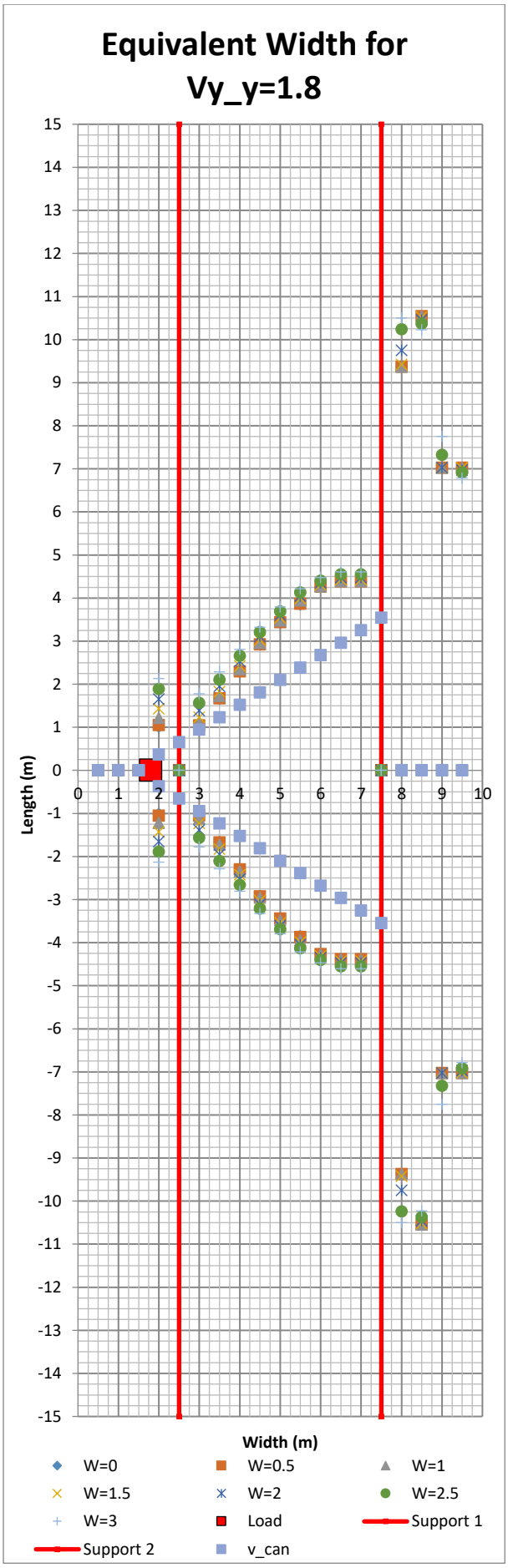

Equivalent Width for My_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

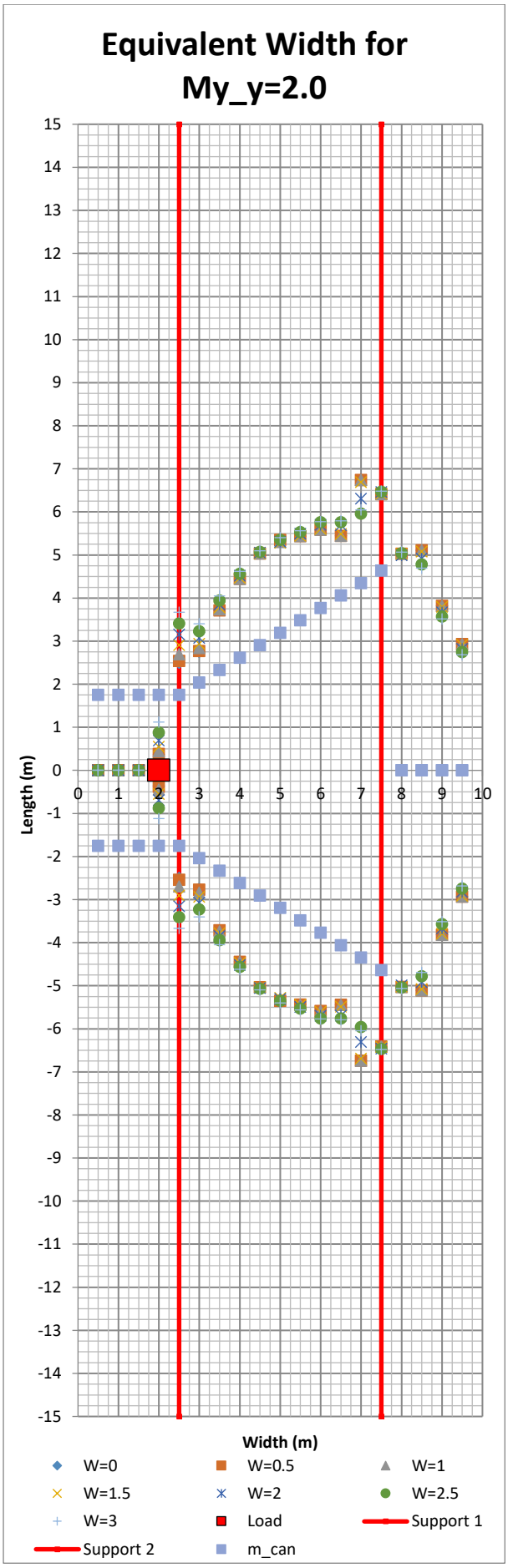

Equivalent Width for Vy_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

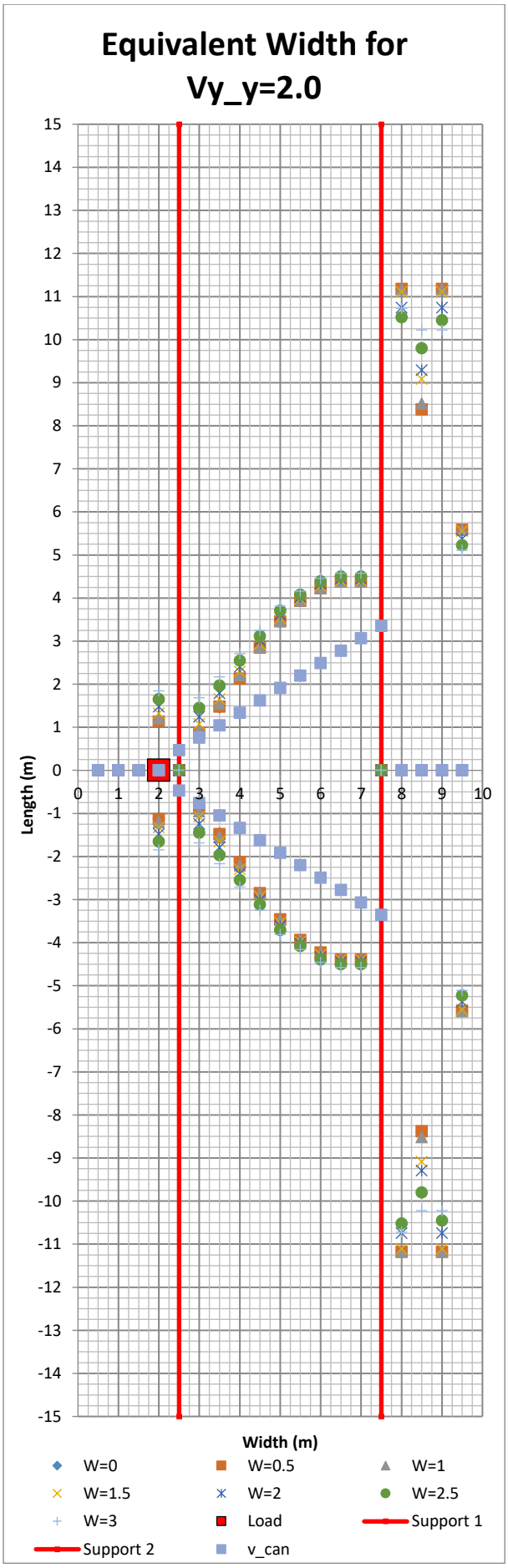

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

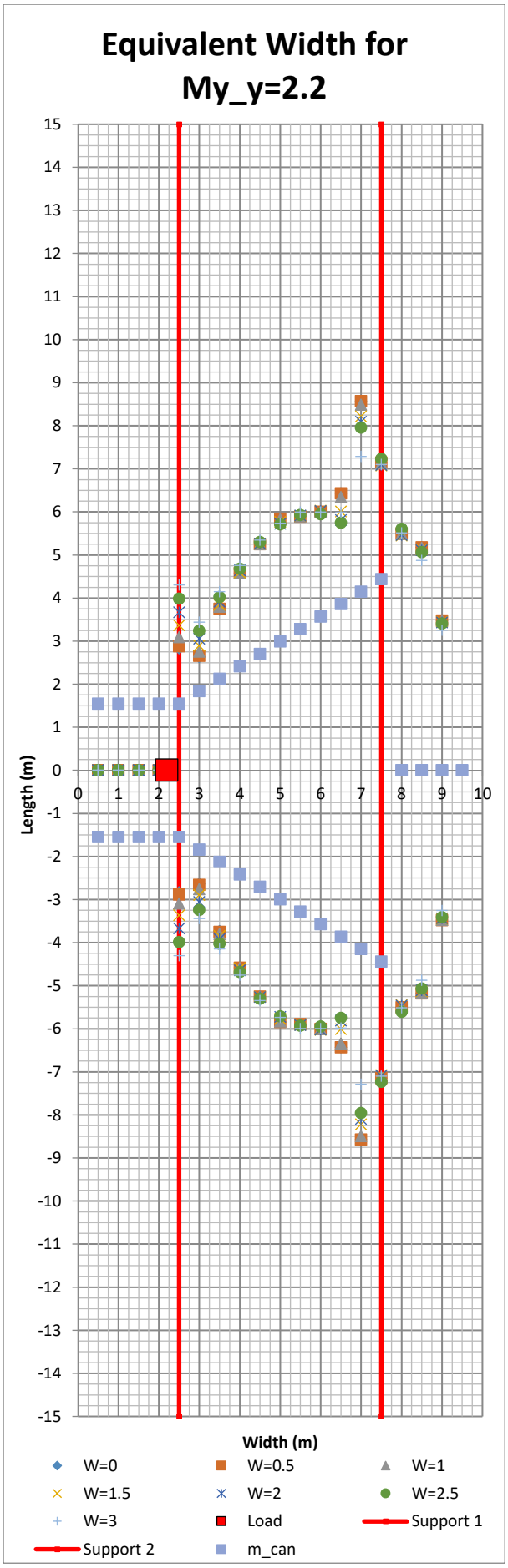

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

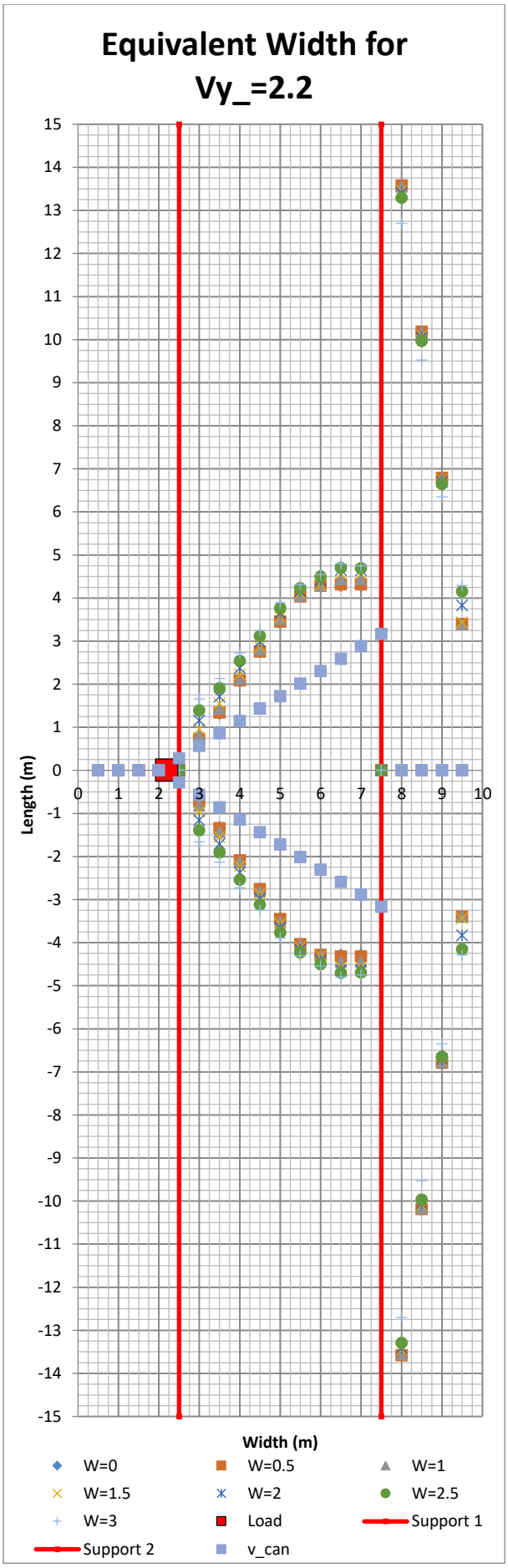

Equivalent Width for My_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

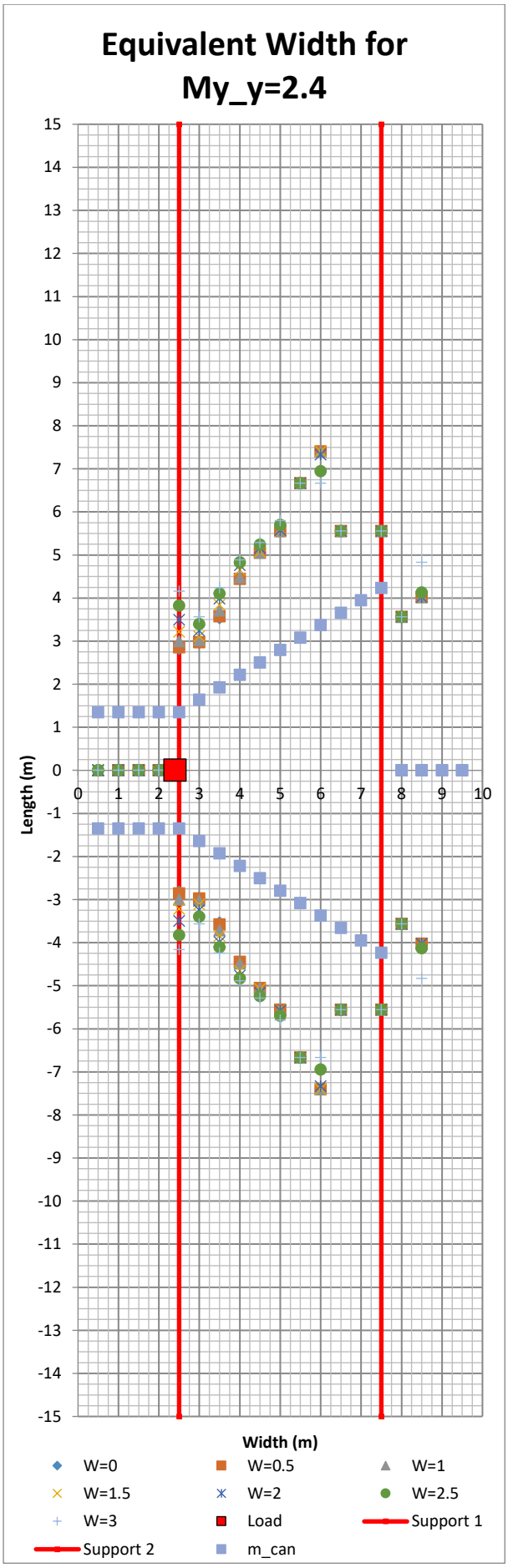

Equivalent Width for Vy_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

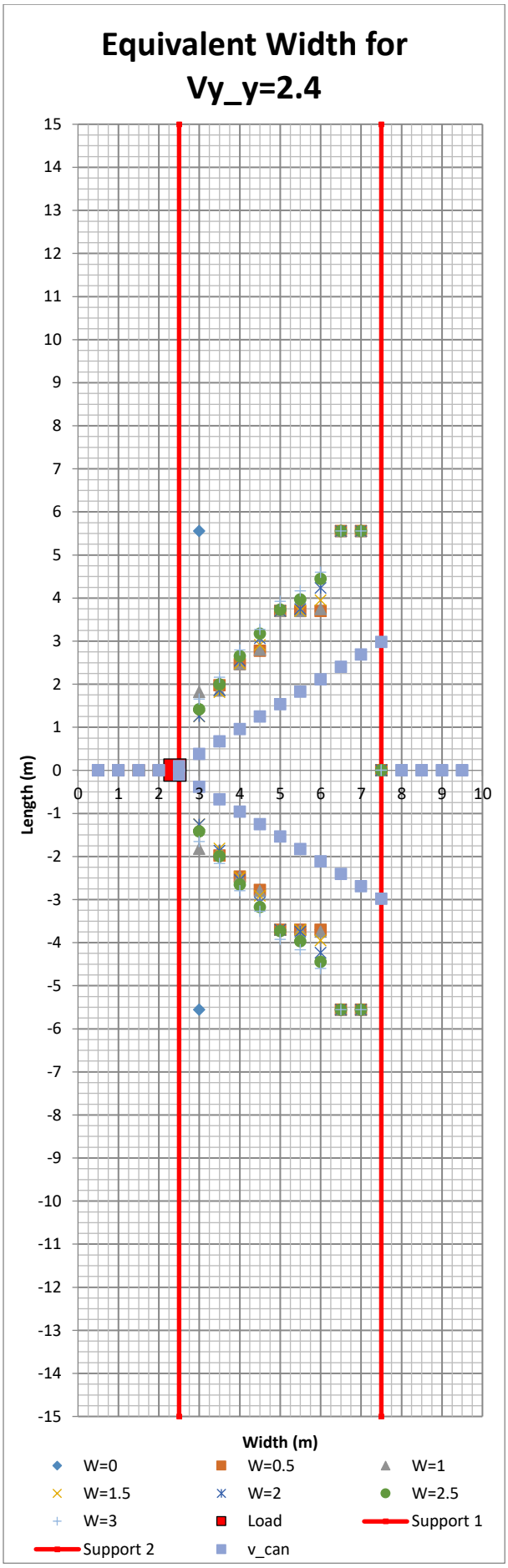

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

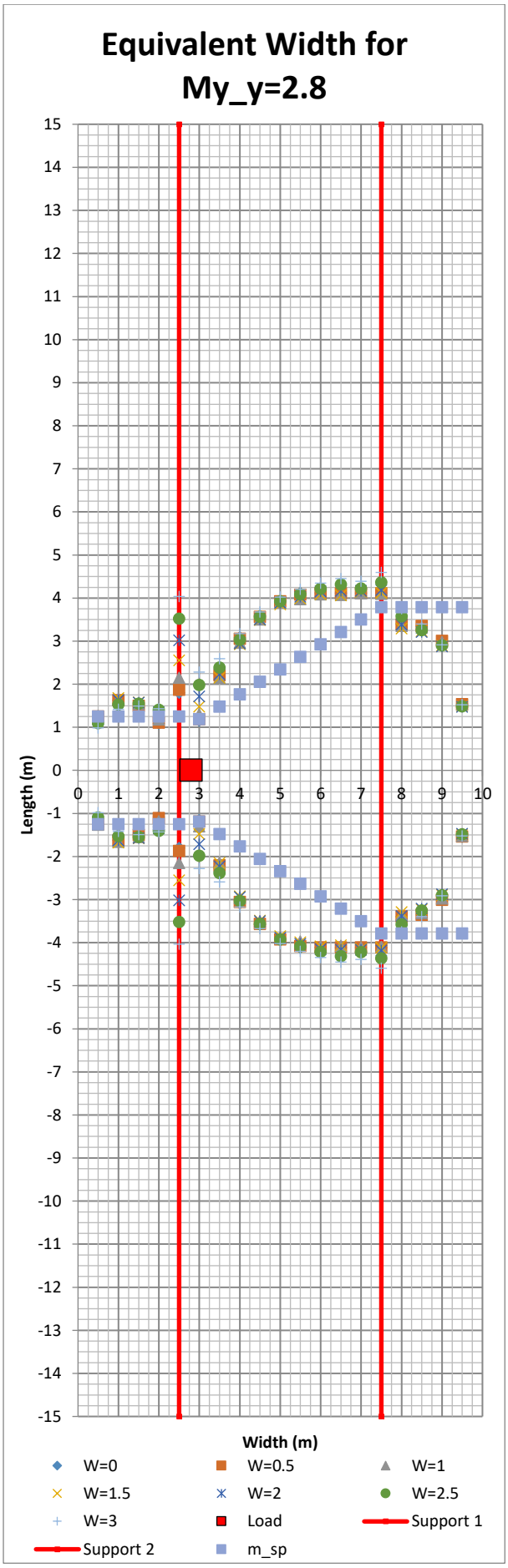

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

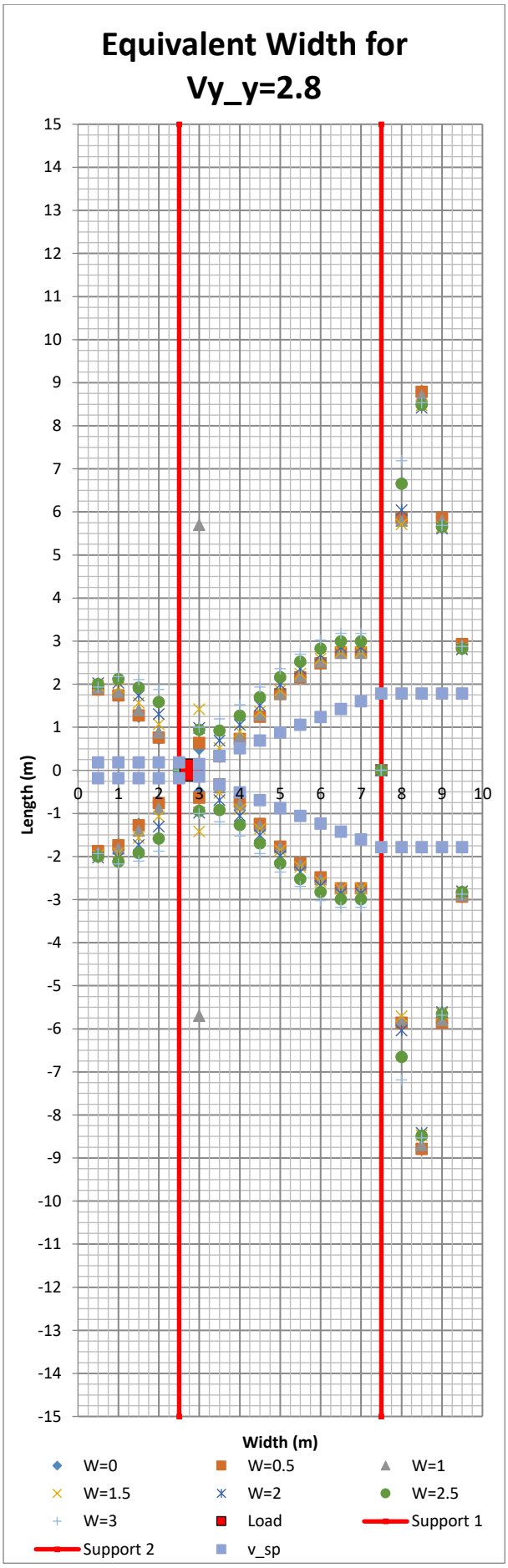

Equivalent Width for My_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

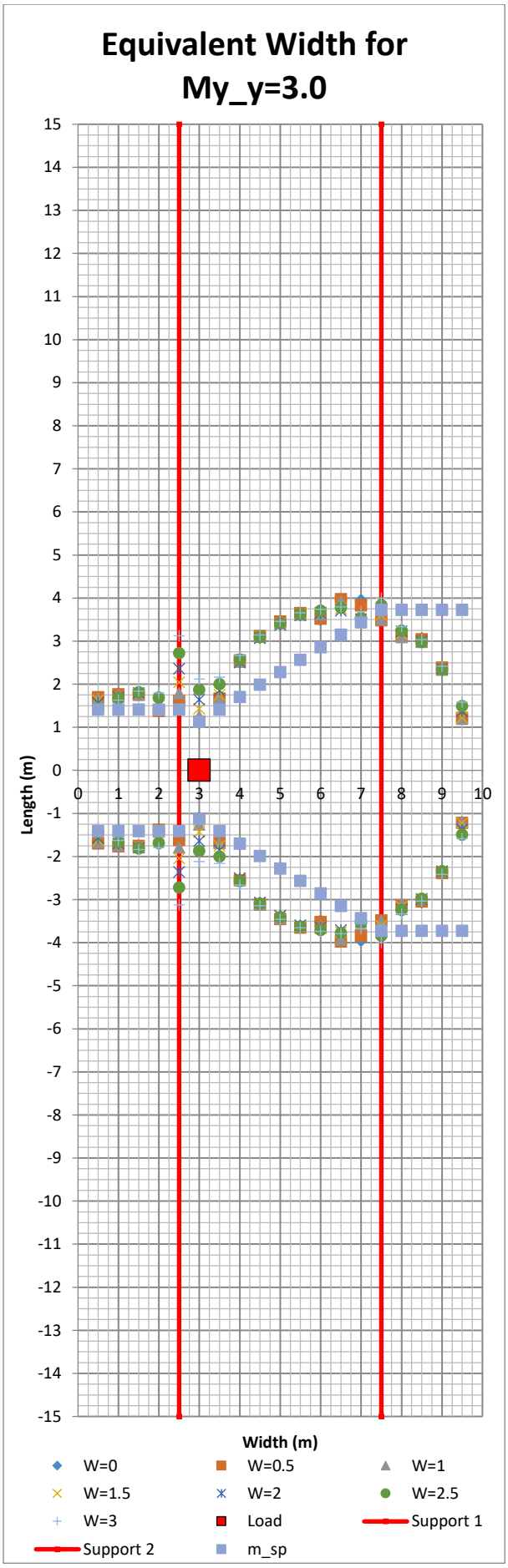

Equivalent Width for Vy_=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

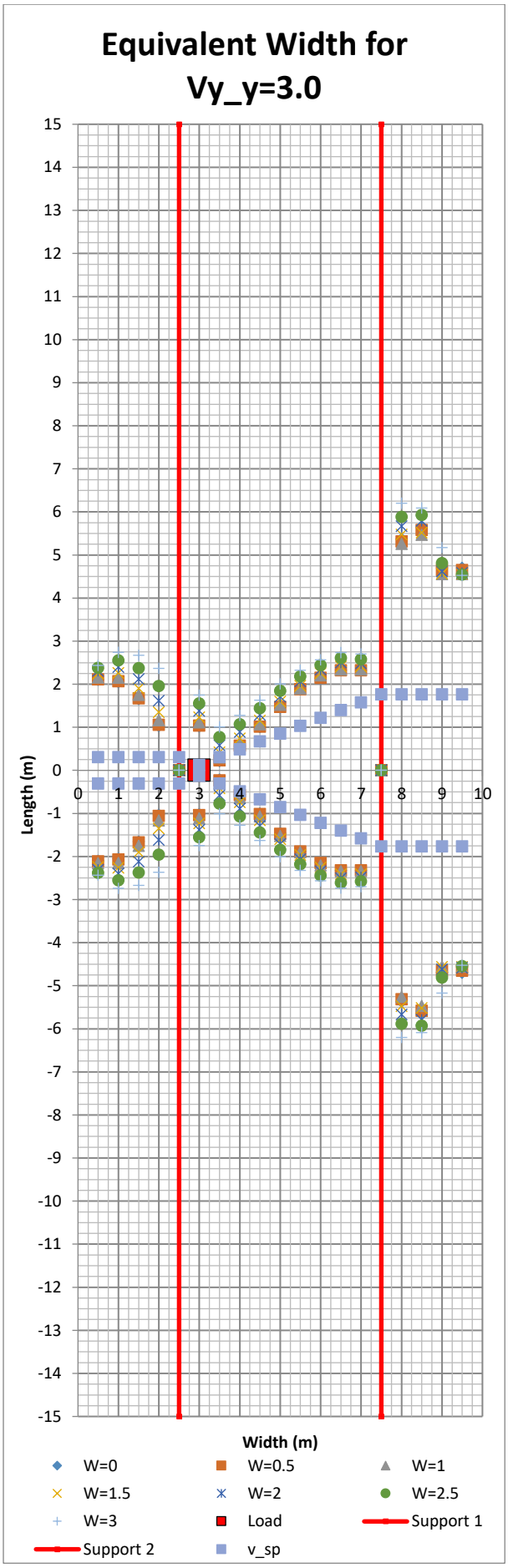

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

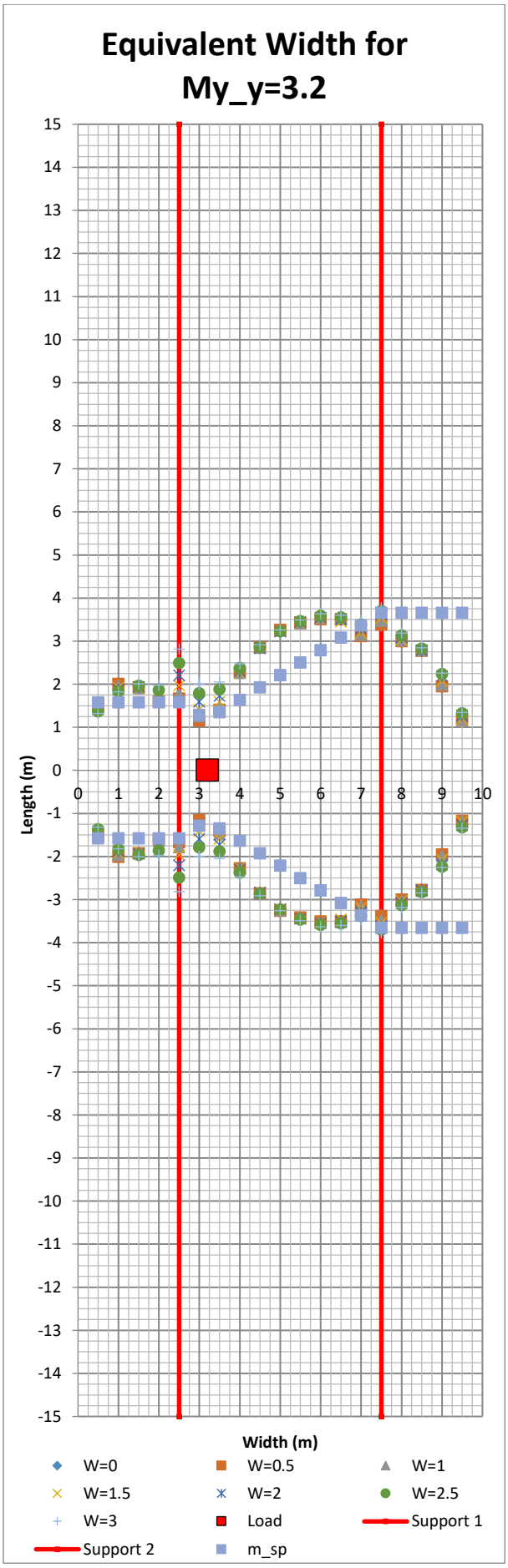

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

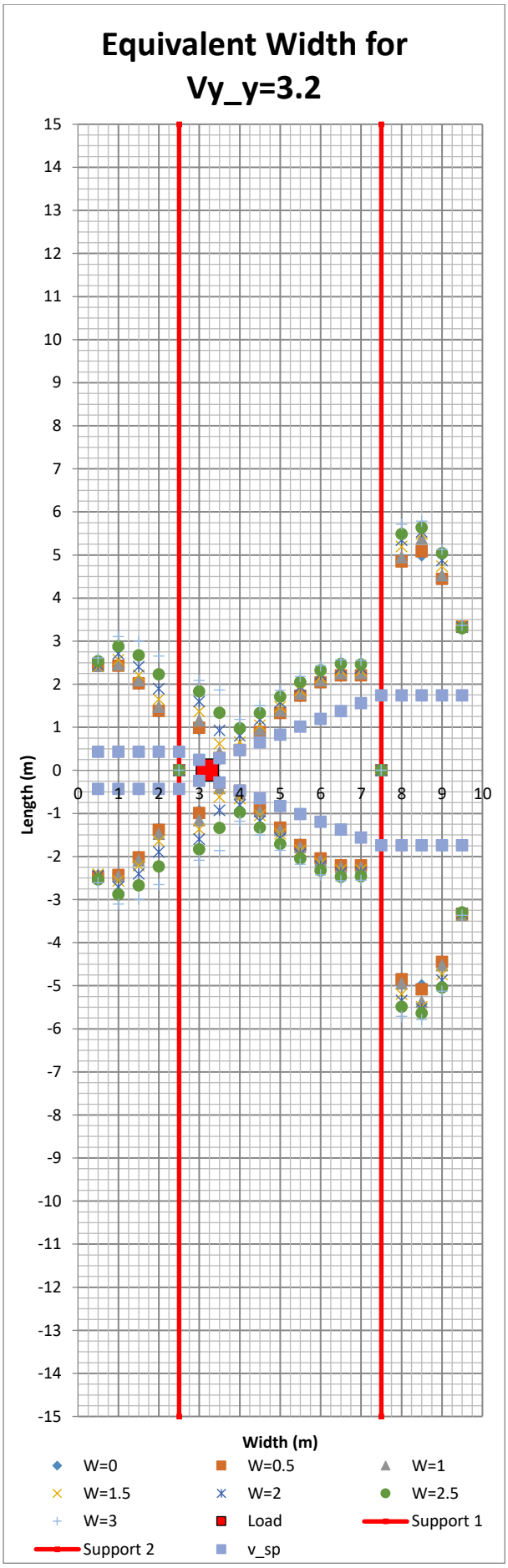

Equivalent Width for My_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

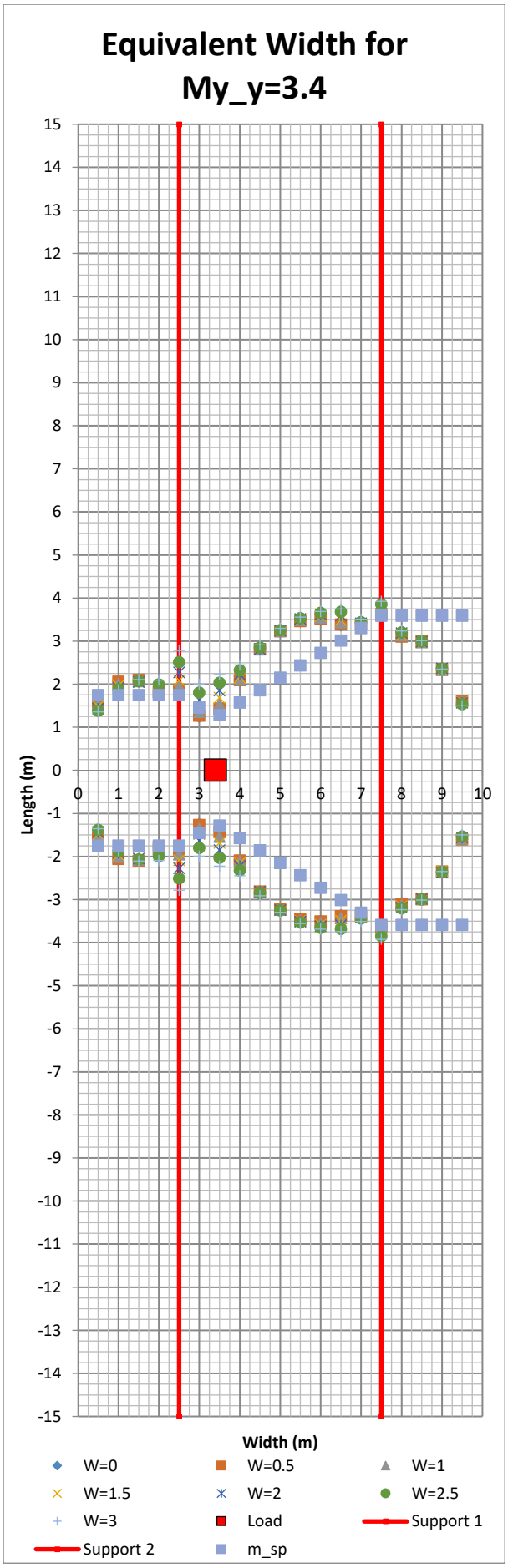

Equivalent Width for Vy_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

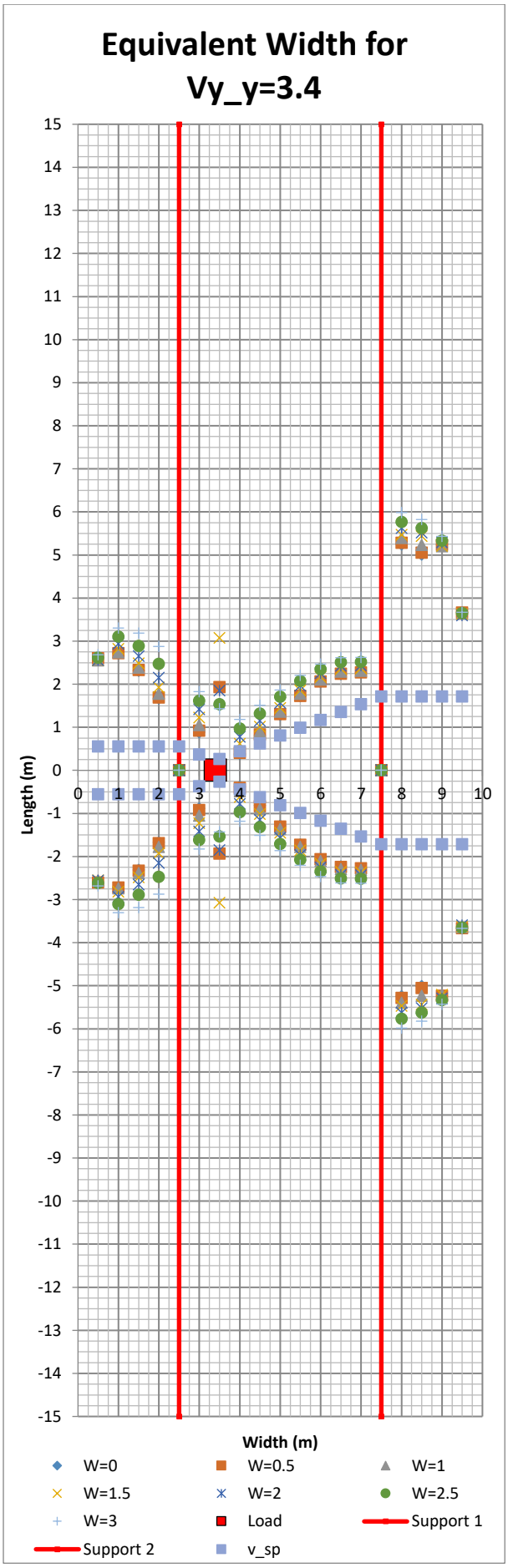

Equivalent Width for My_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

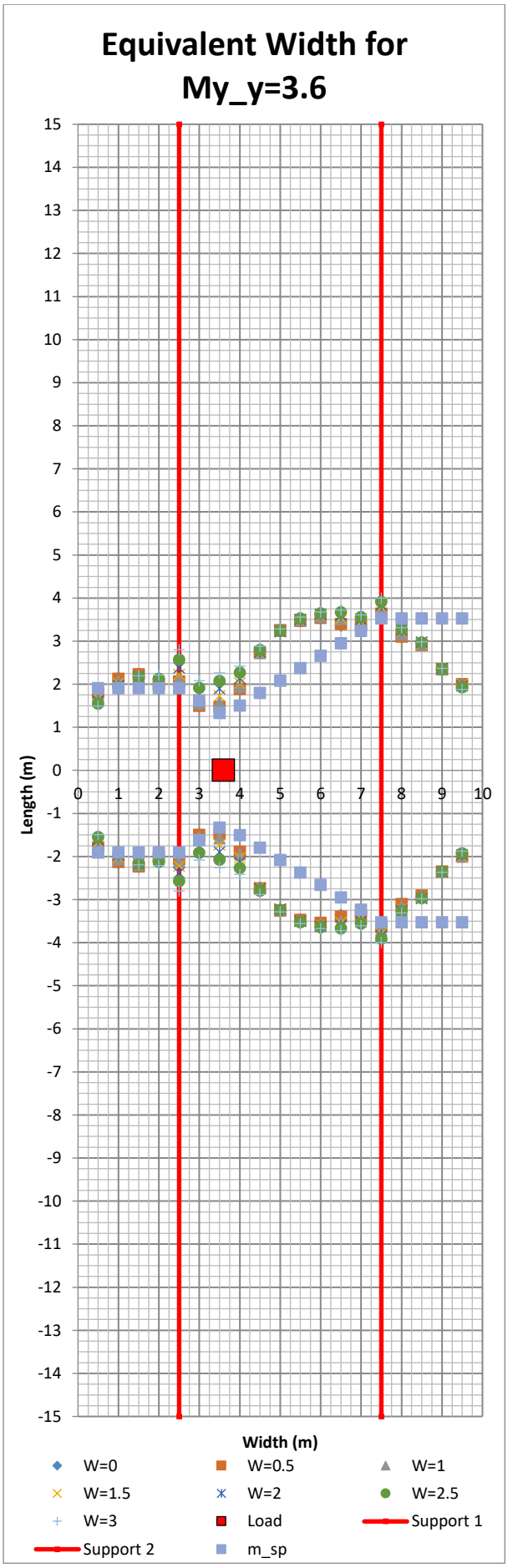

Equivalent Width for Vy_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

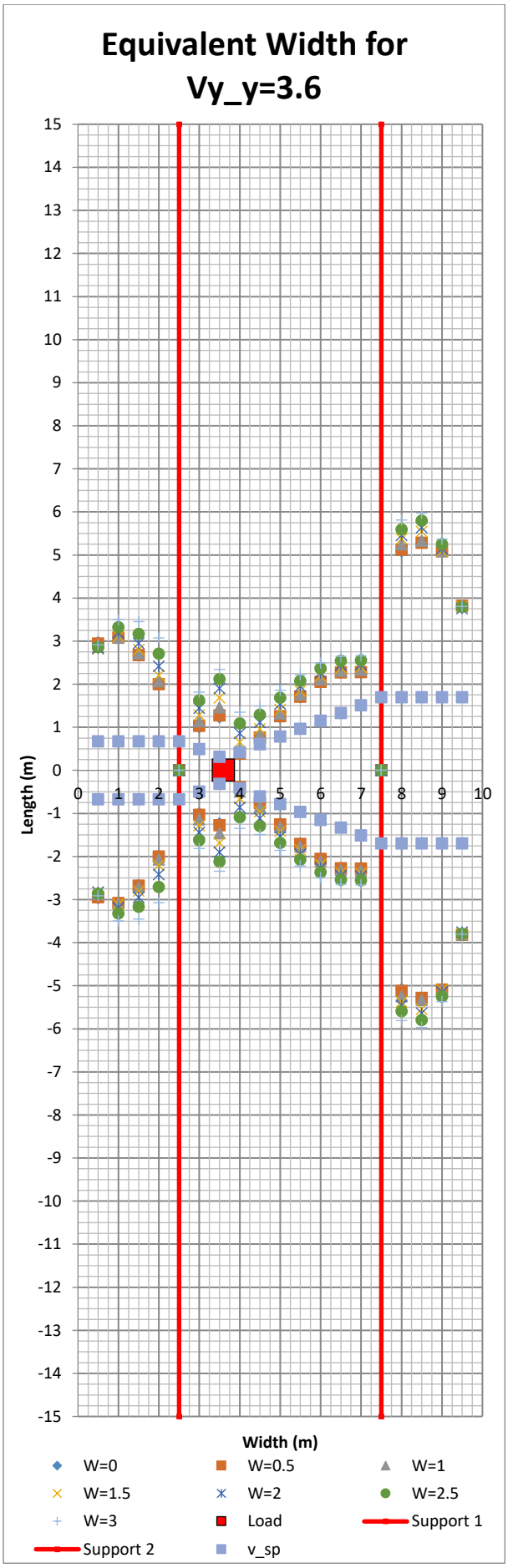

Equivalent Width for My_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

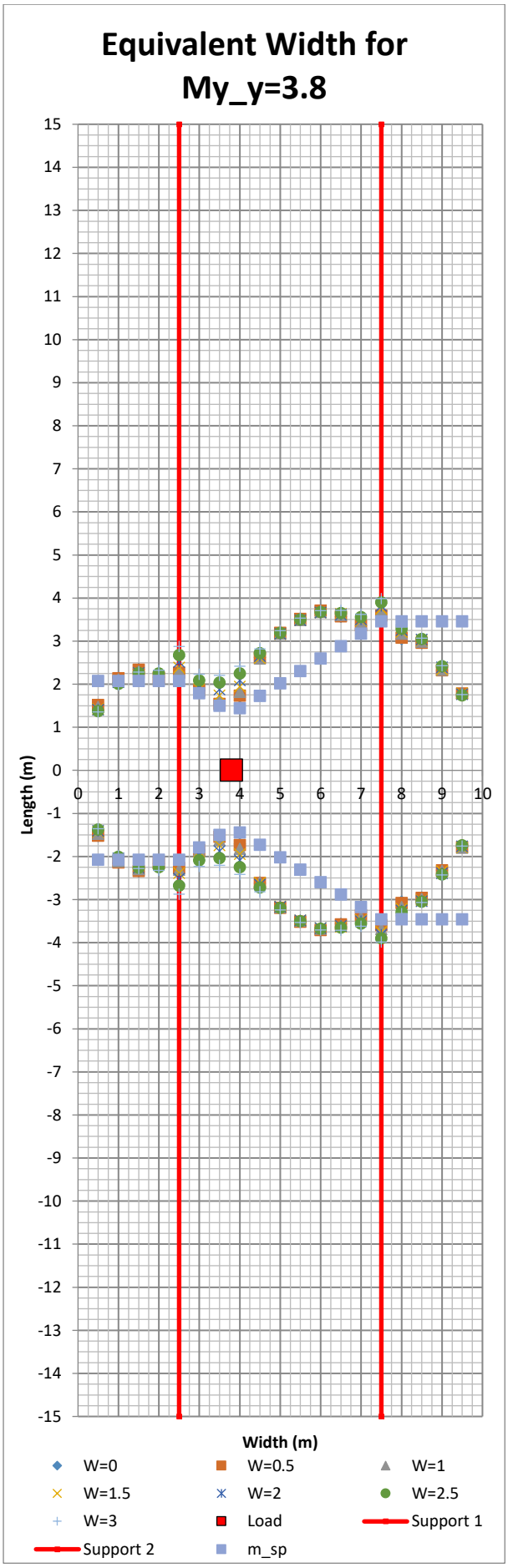

Equivalent Width for Vy_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

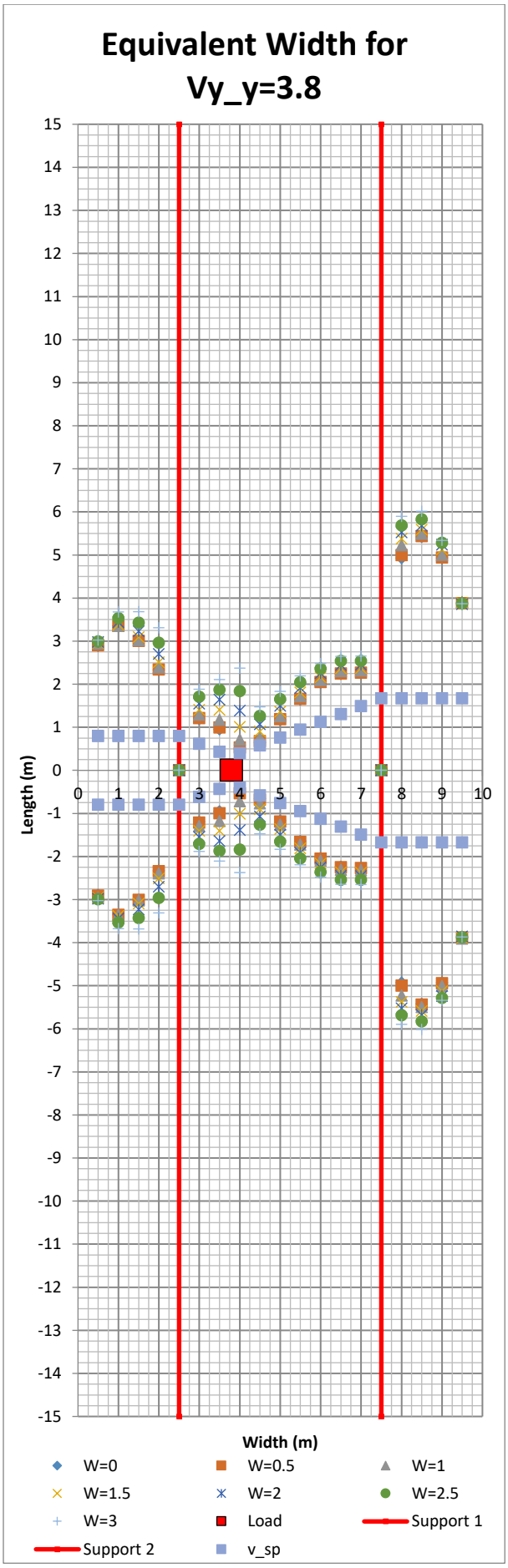

Equivalent Width for My_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

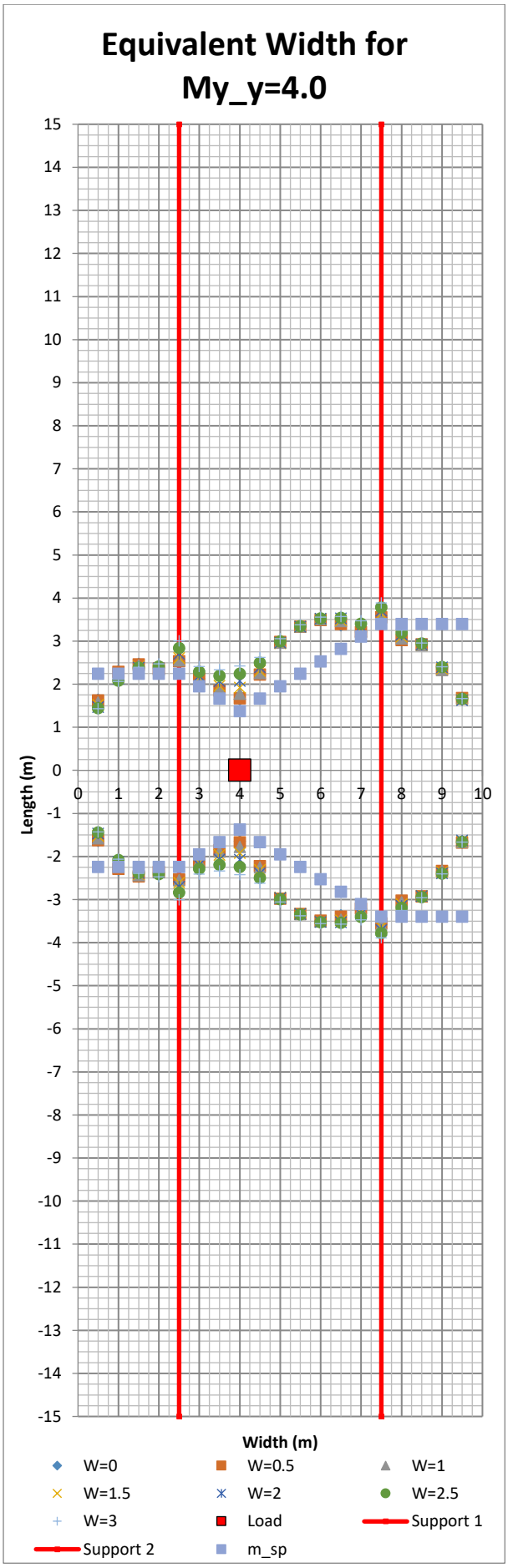

Equivalent Width for Vy_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

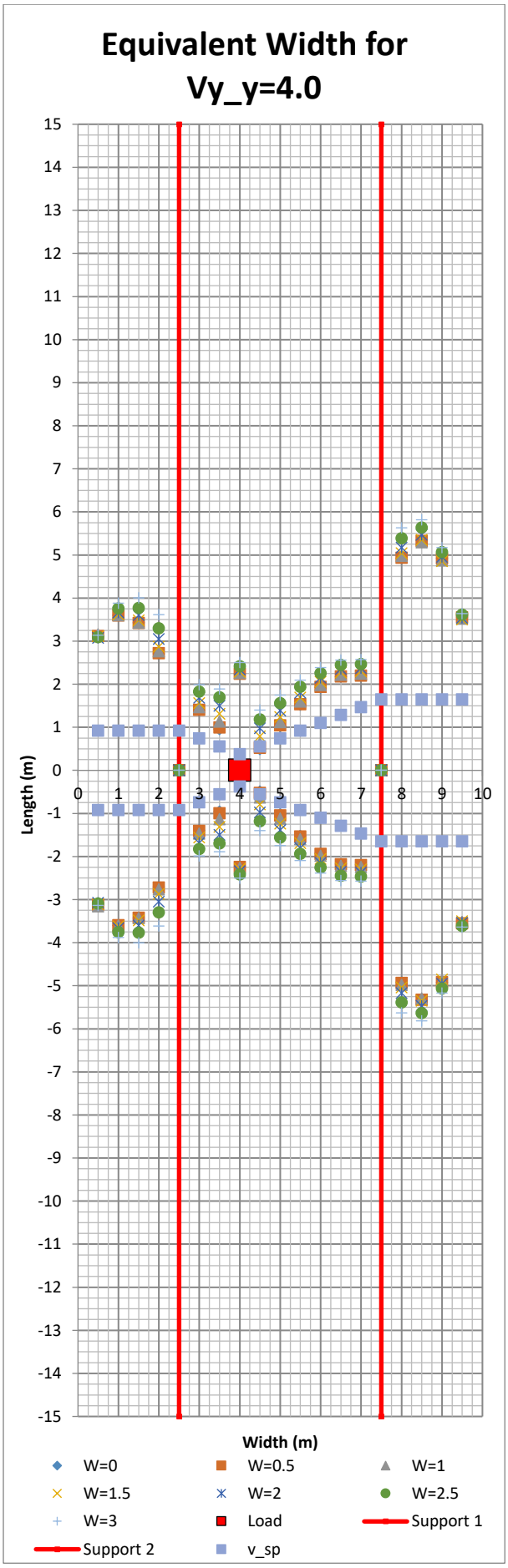

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

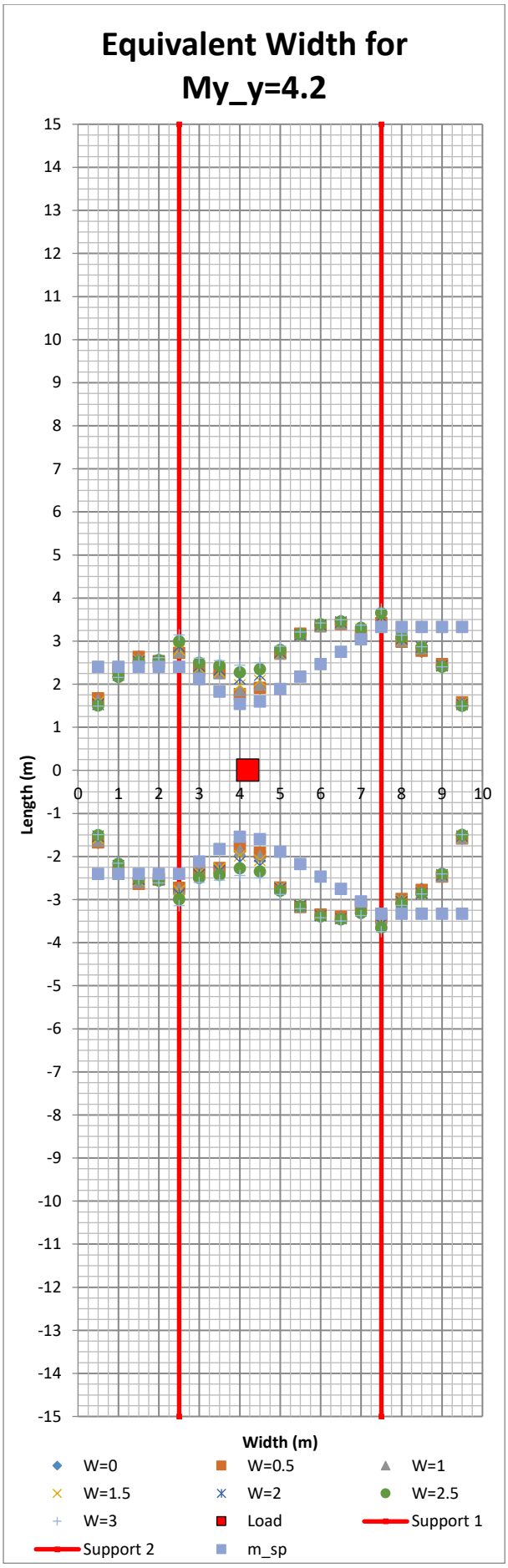

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

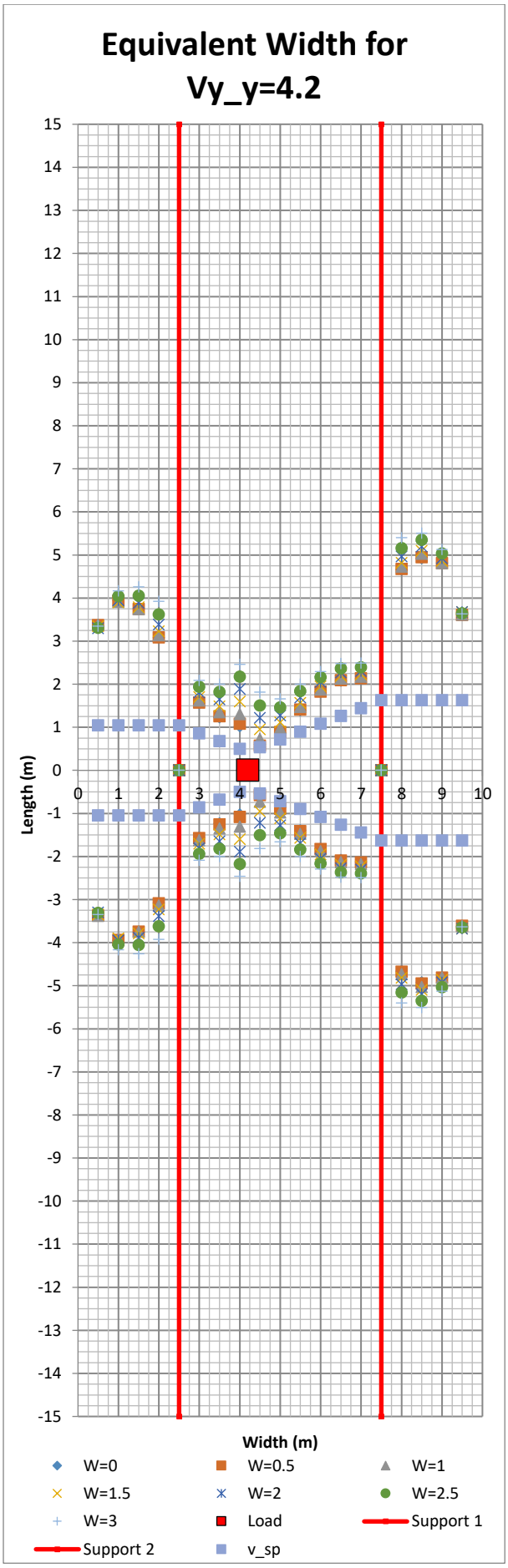

Equivalent Width for My_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

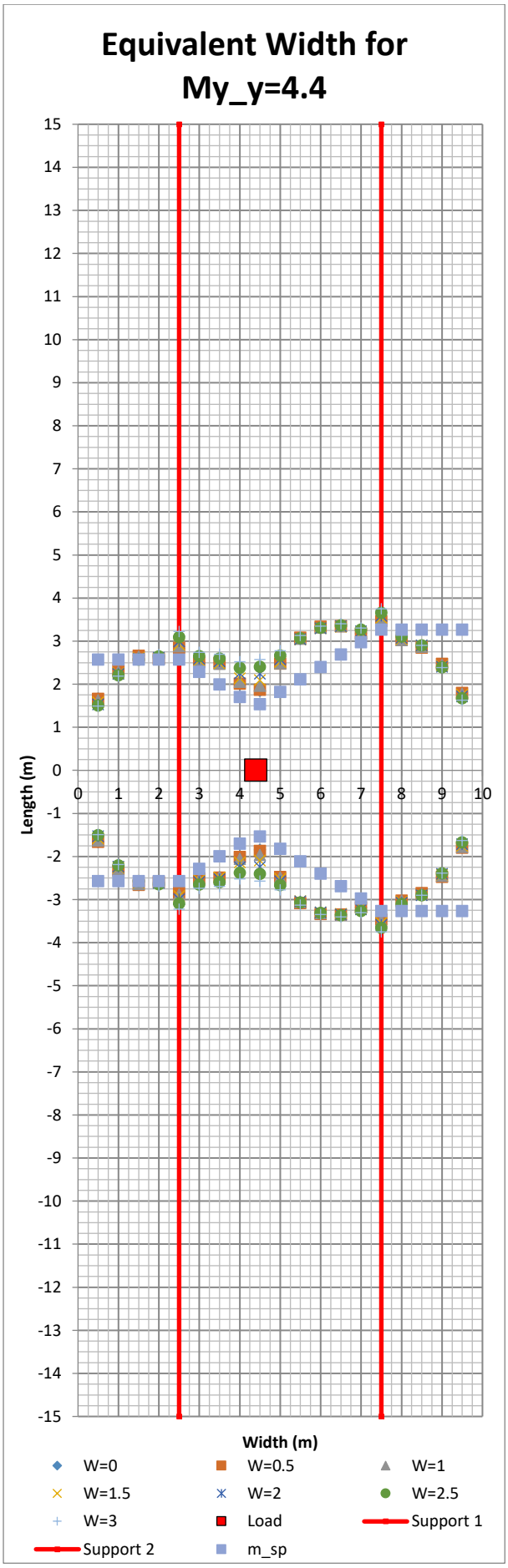

Equivalent Width for Vy_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

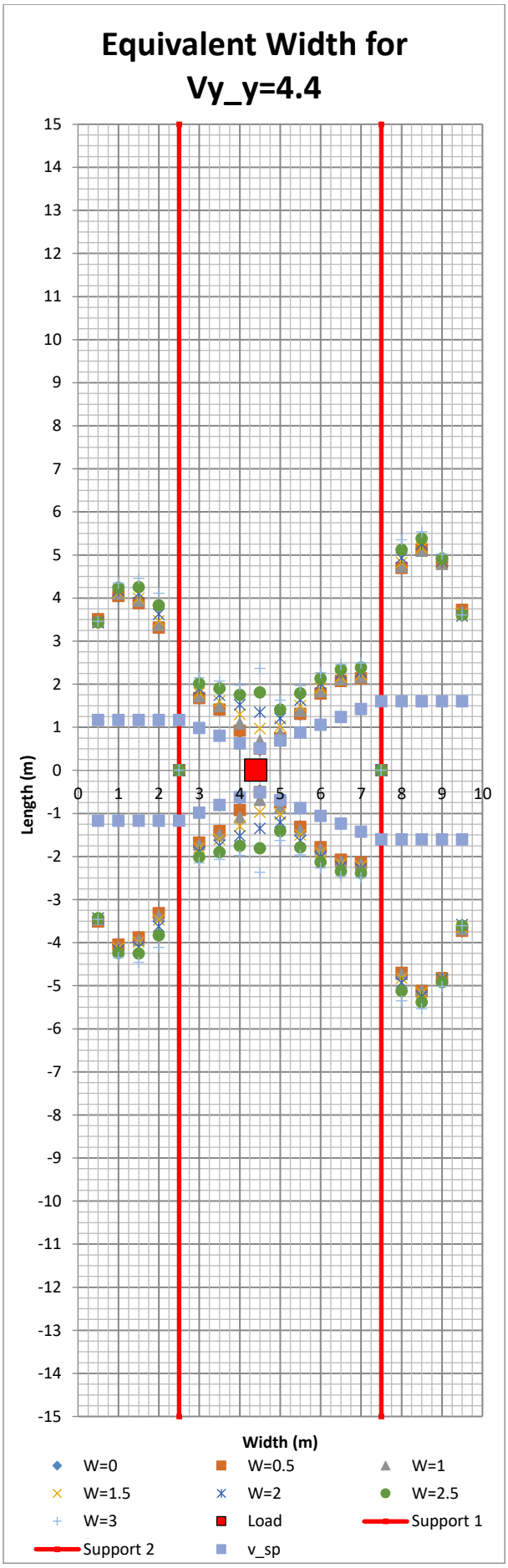

Equivalent Width for My_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

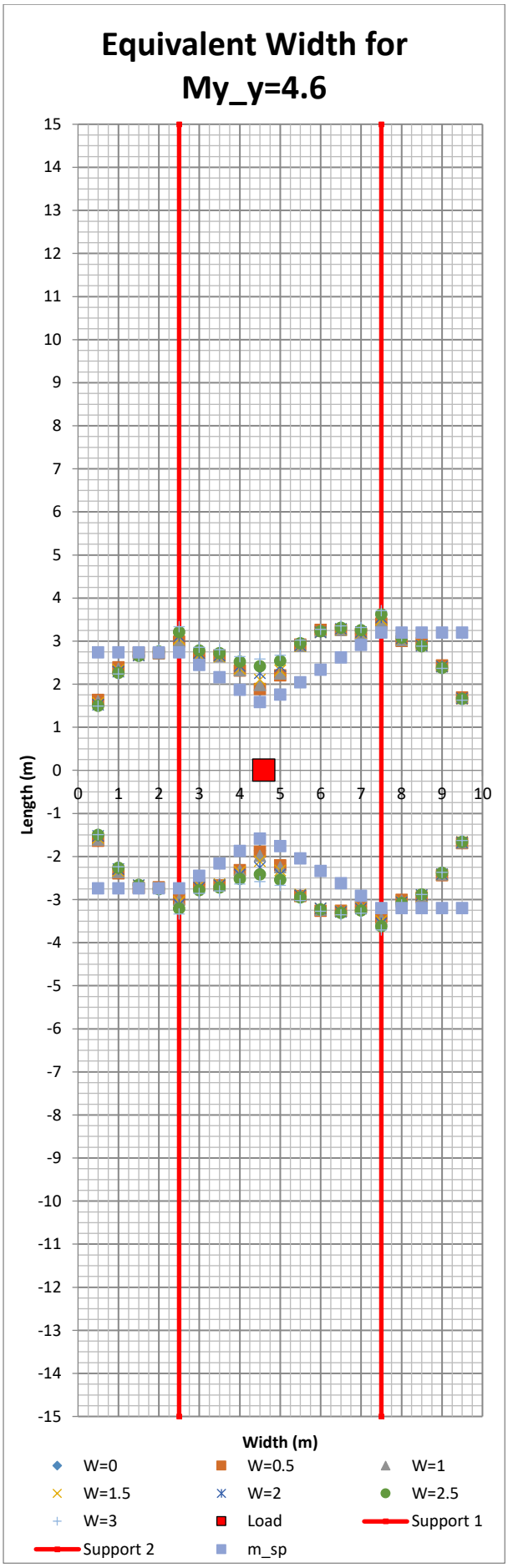

Equivalent Width for Vy_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

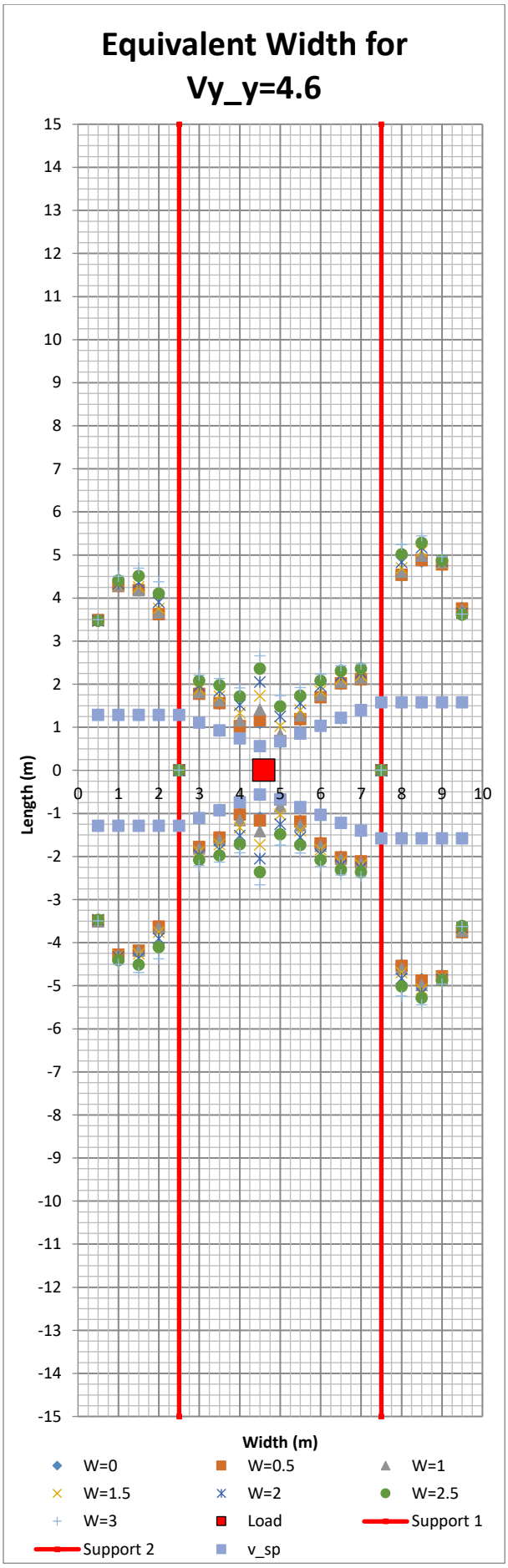

Equivalent Width for My_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

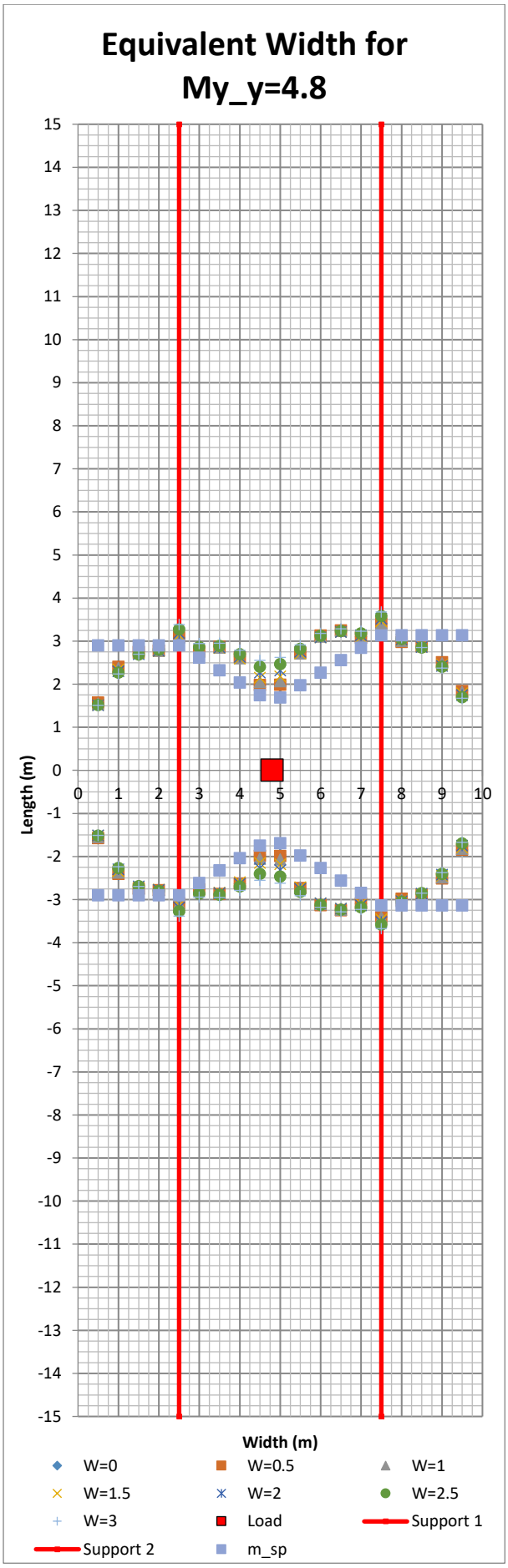

Equivalent Width for Vy_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

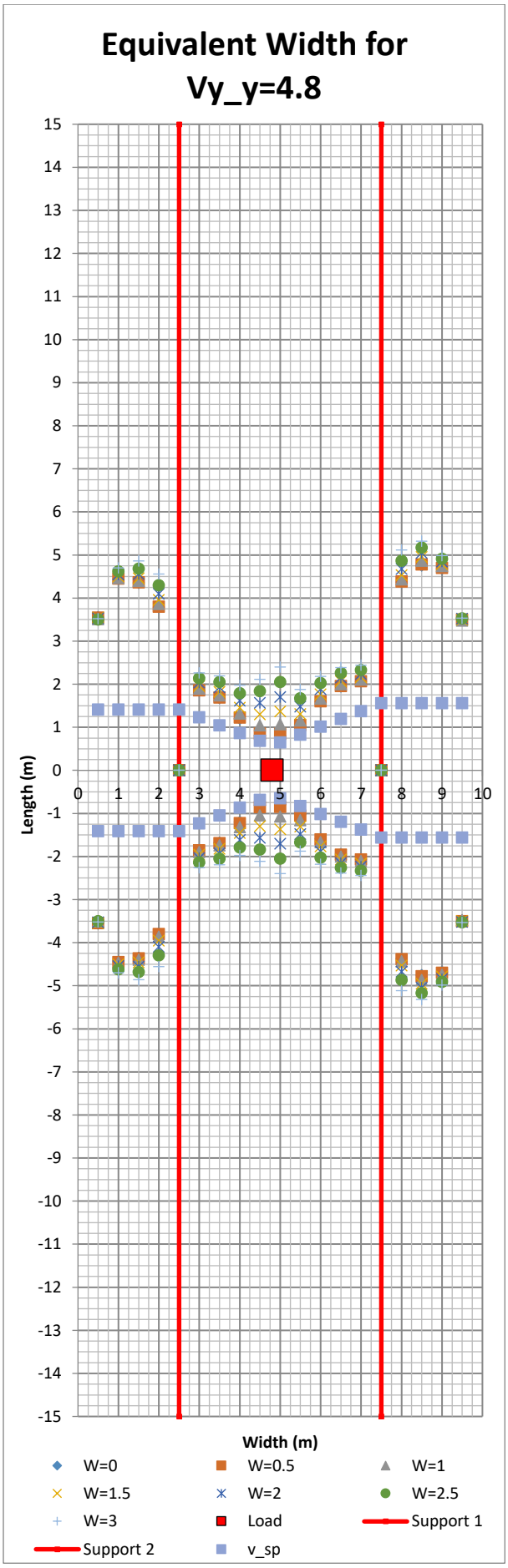

Equivalent Width for My_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

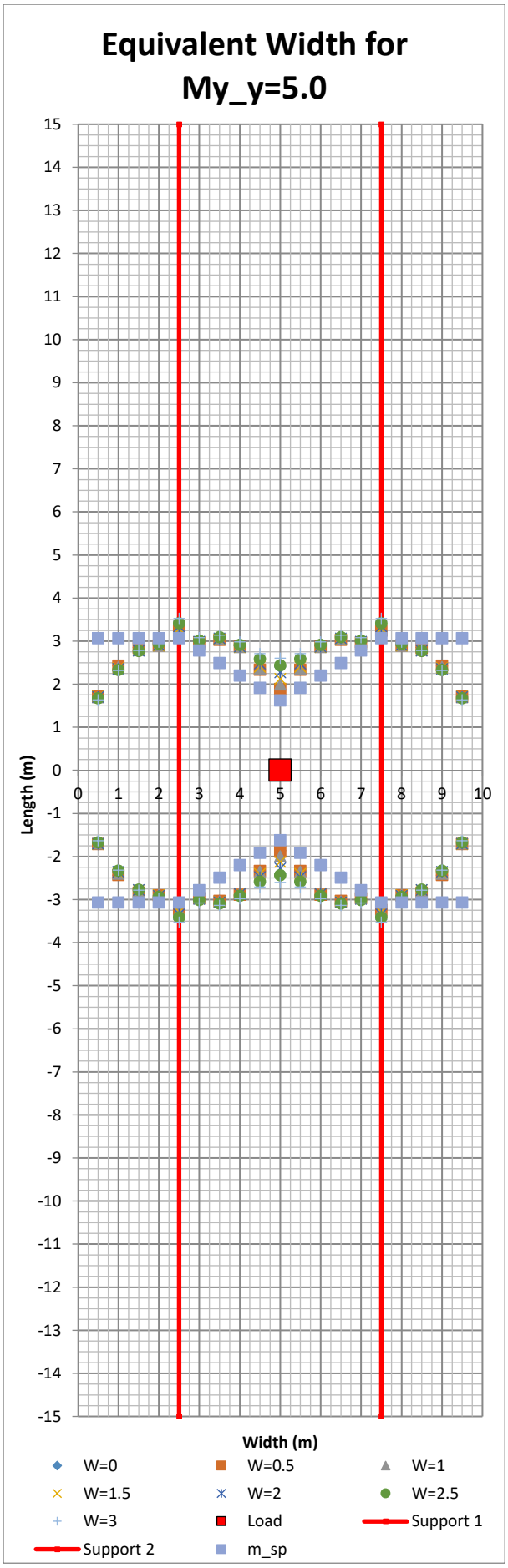

Equivalent Width for Vy_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

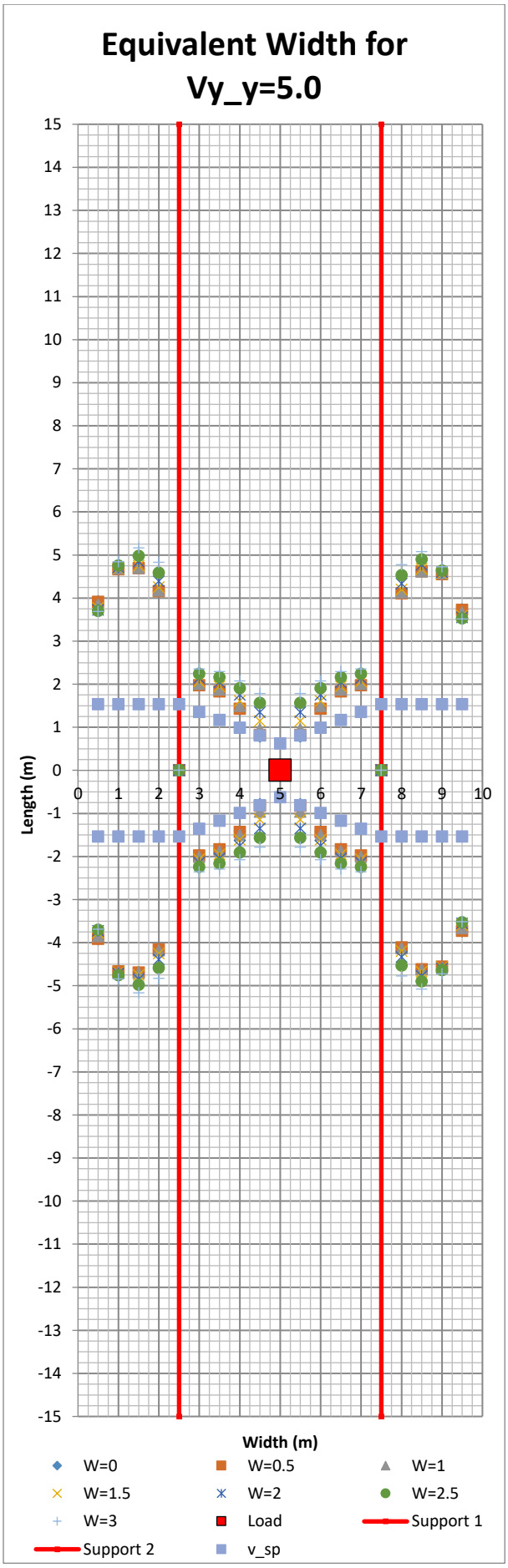

Equivalent Width for My_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

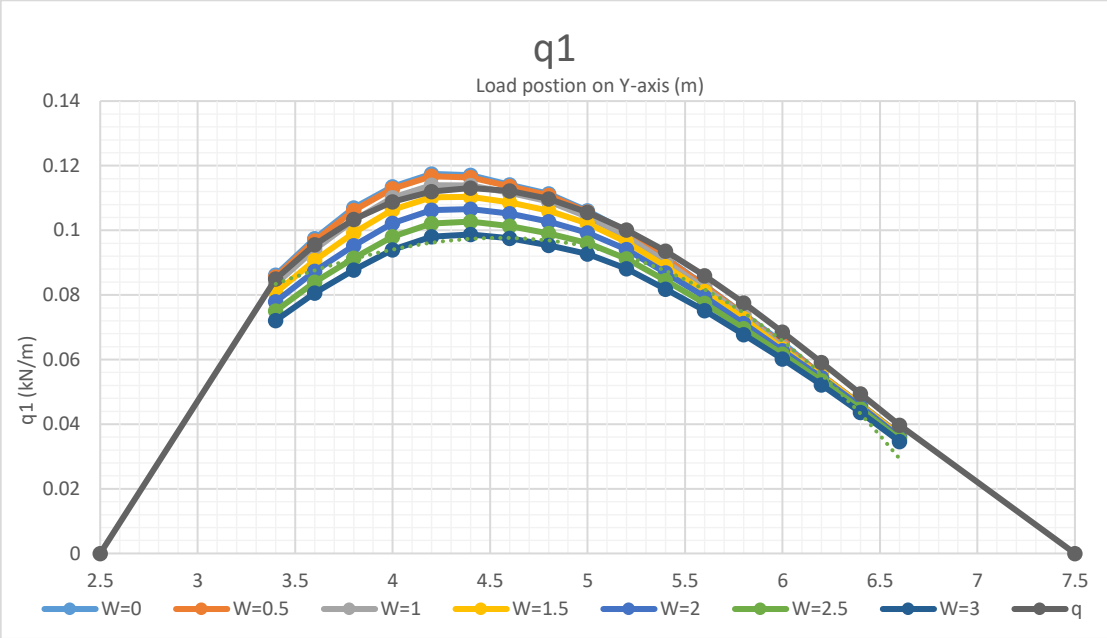

Equivalent Width for My_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

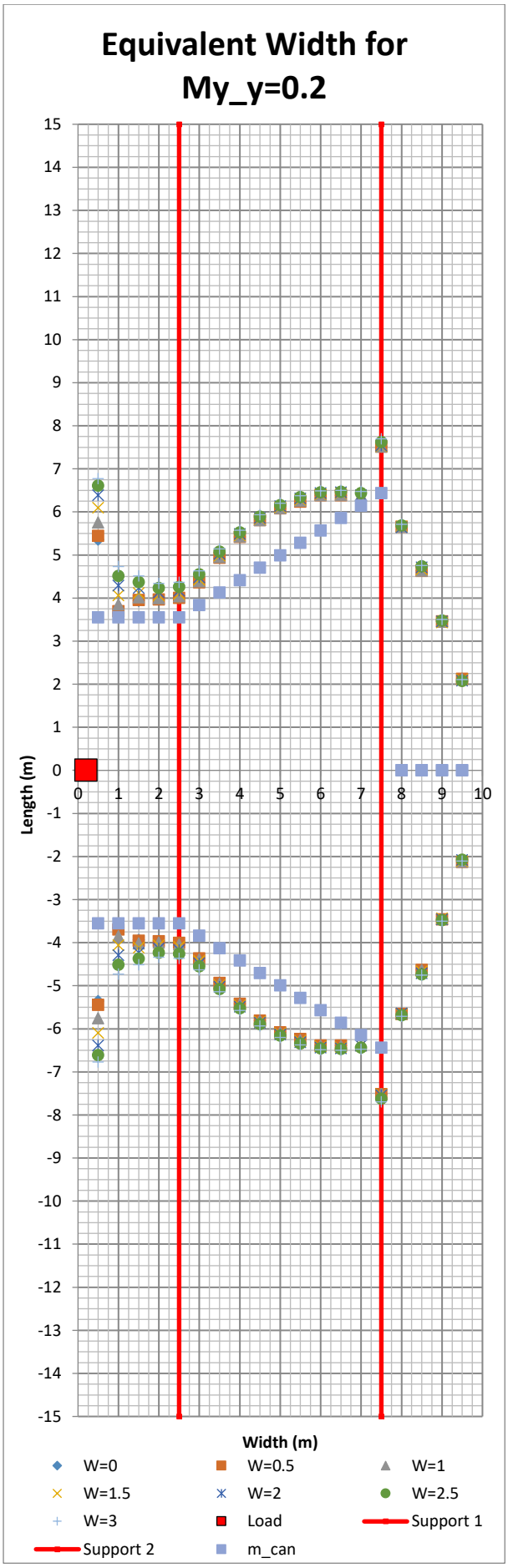

Equivalent Width for Vy_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

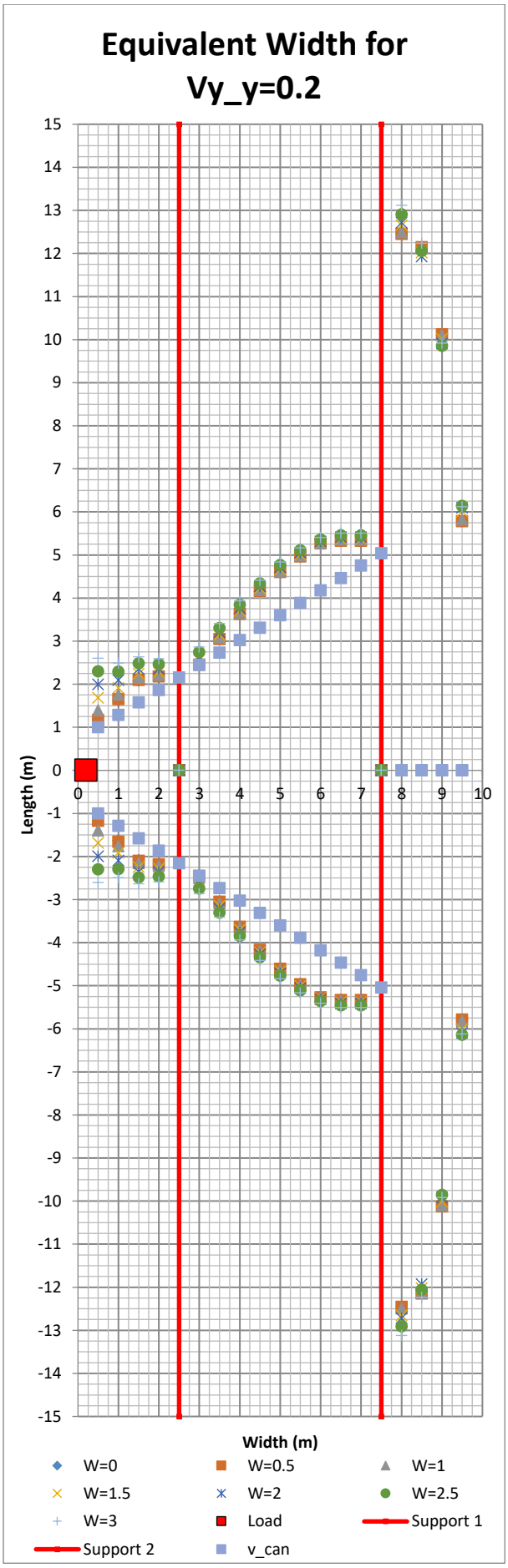

Equivalent Width for My_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

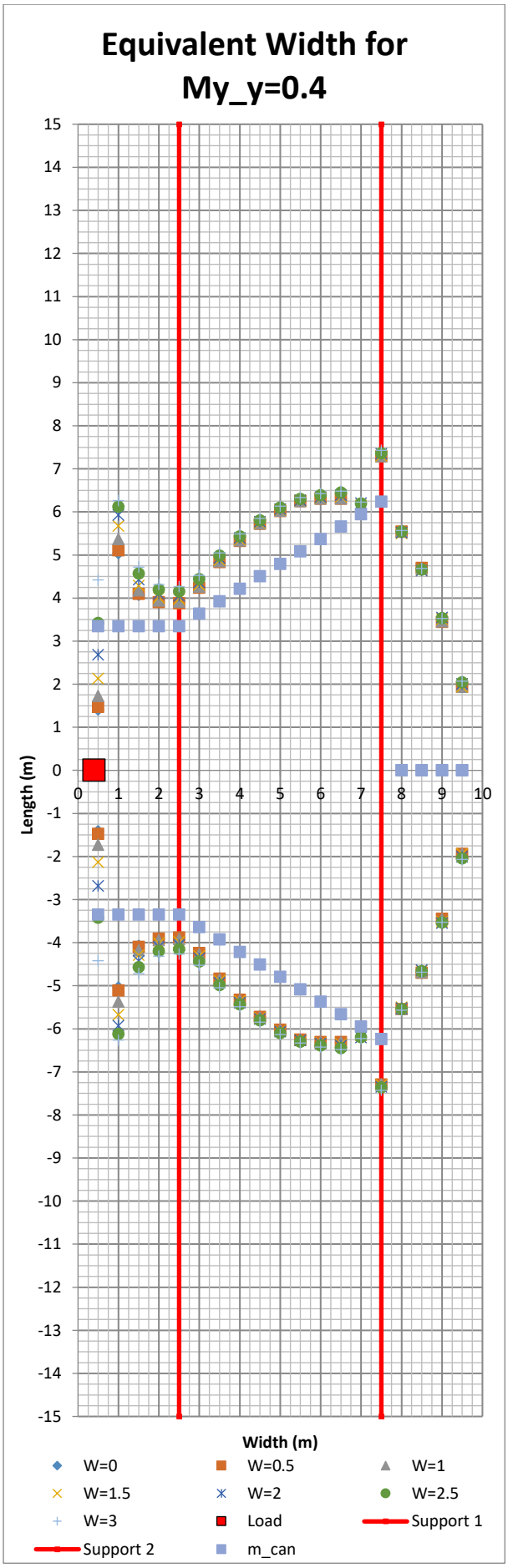

Equivalent Width for Vy_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

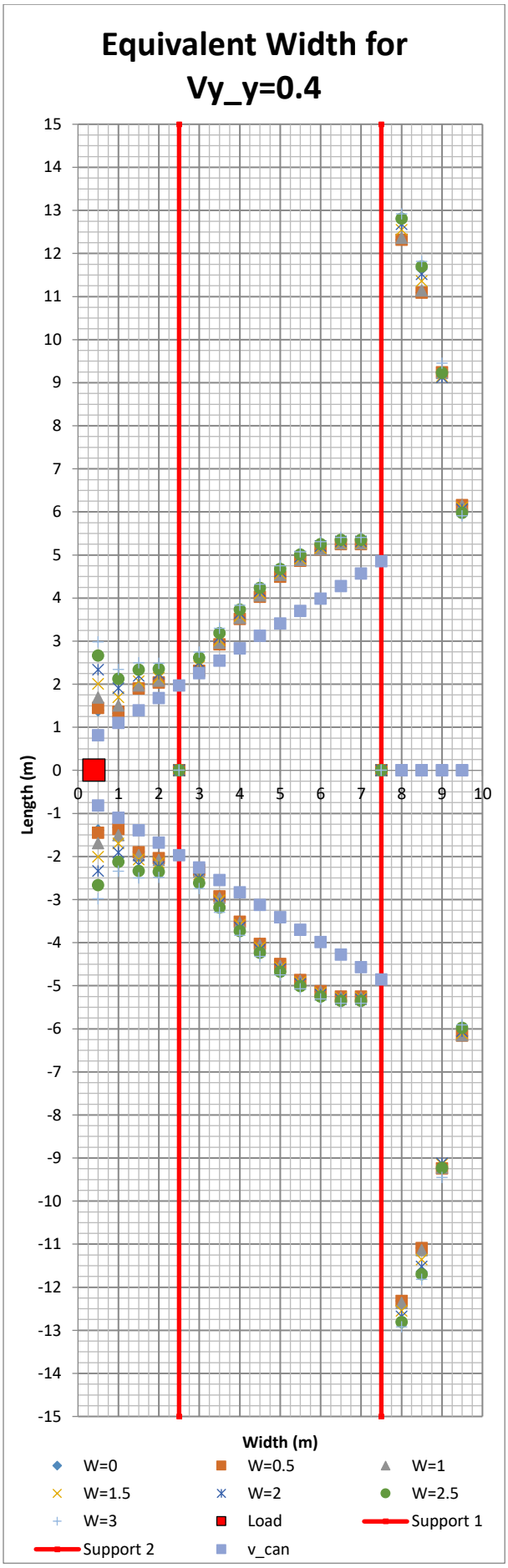
11.2 The second Variation changing the concrete type to C12/15

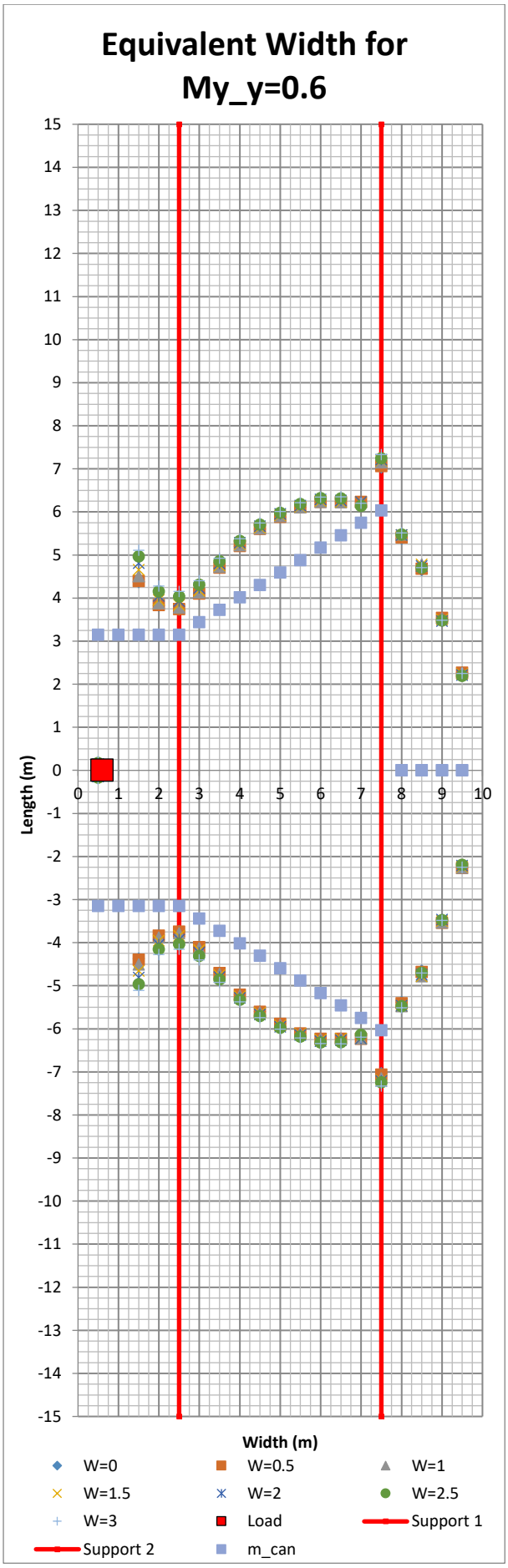


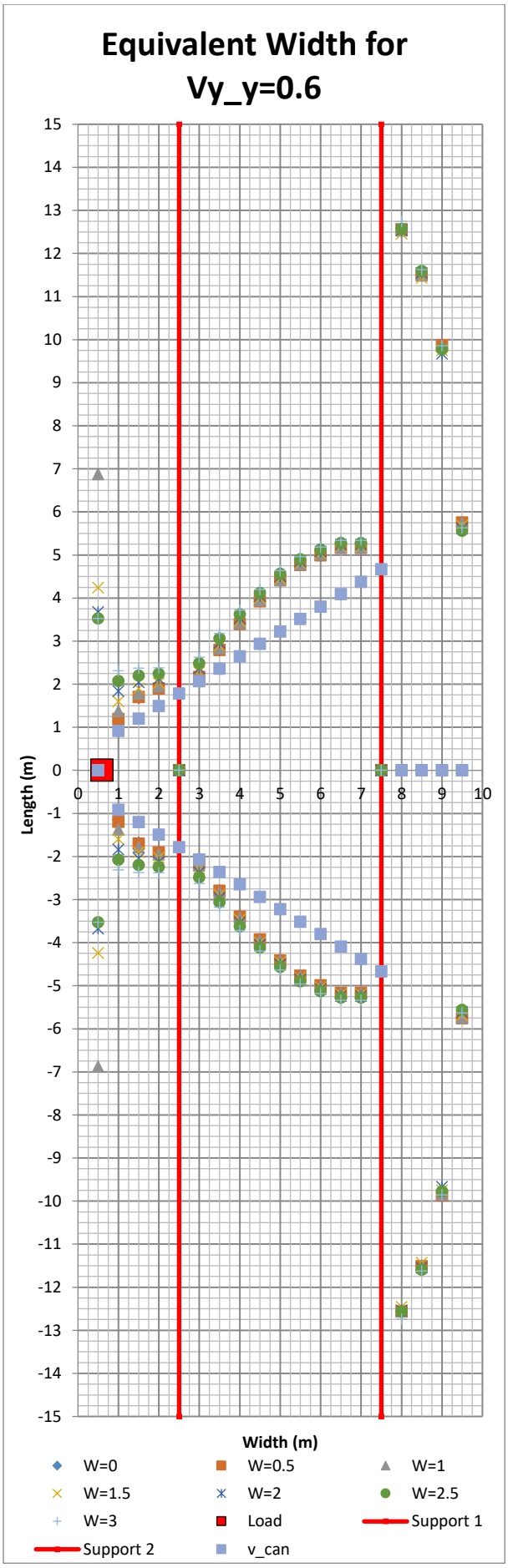

Equivalent Width for My_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

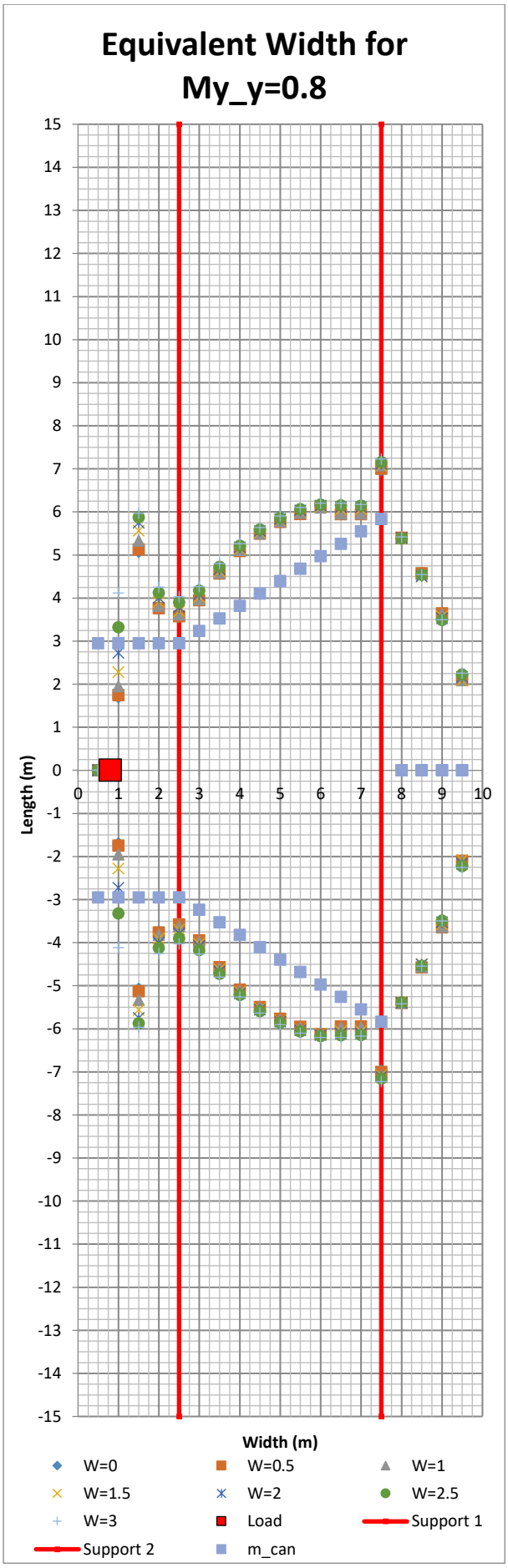

Equivalent Width for Vy_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

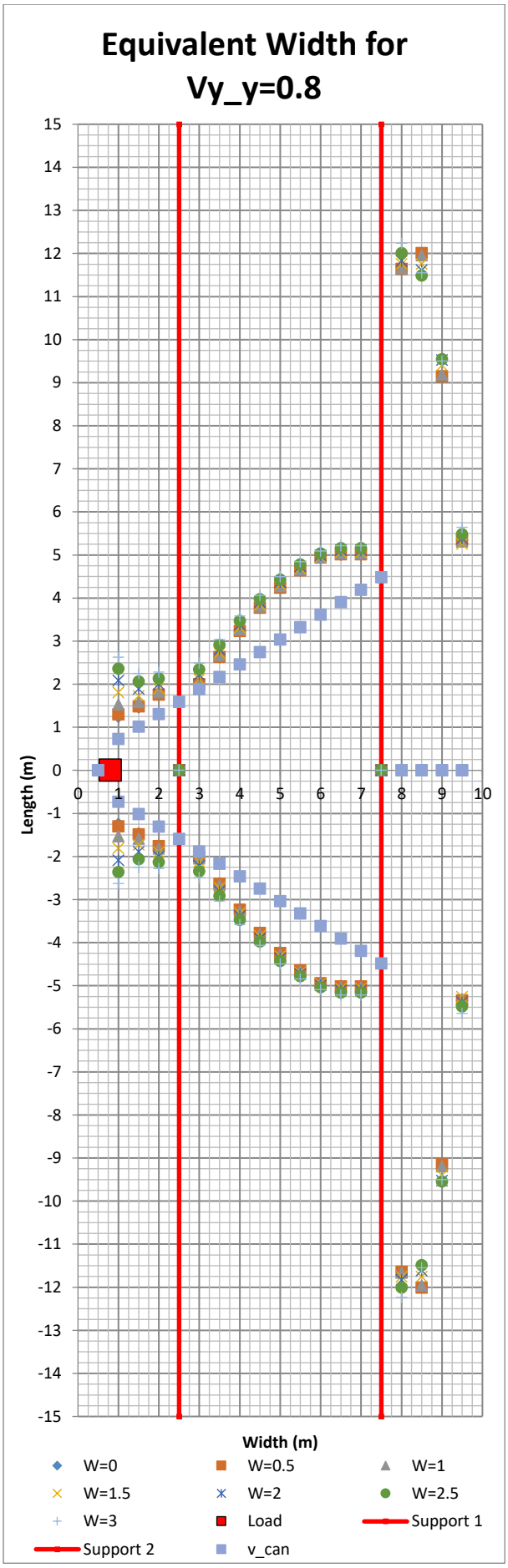

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

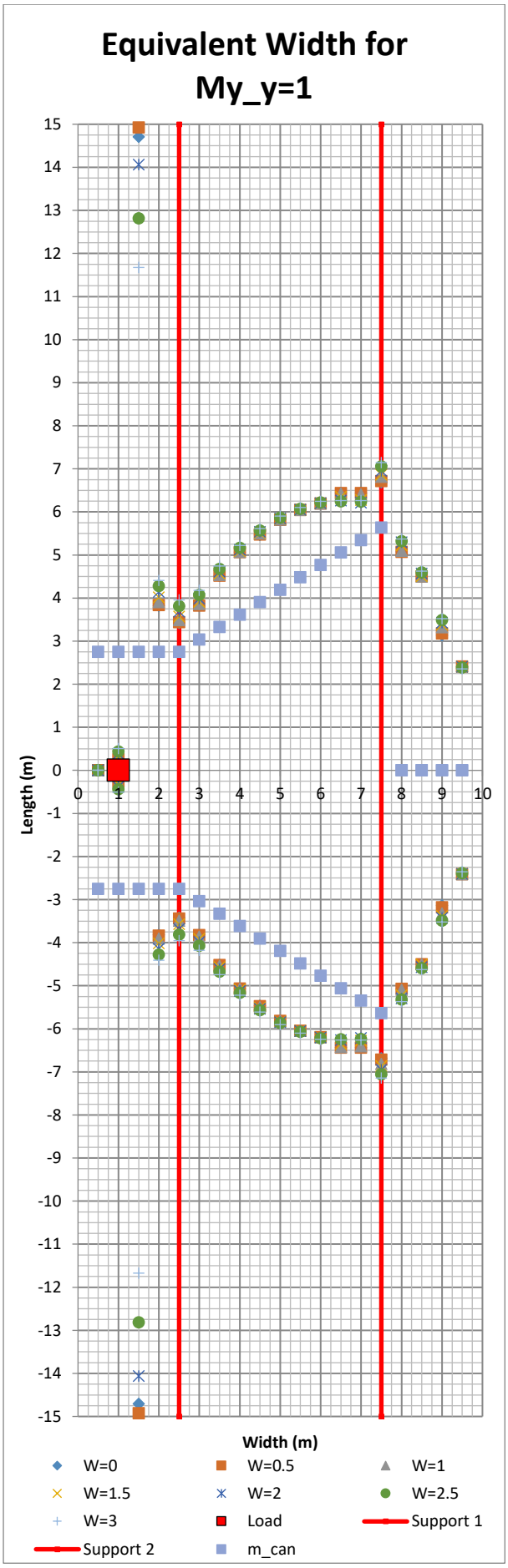

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

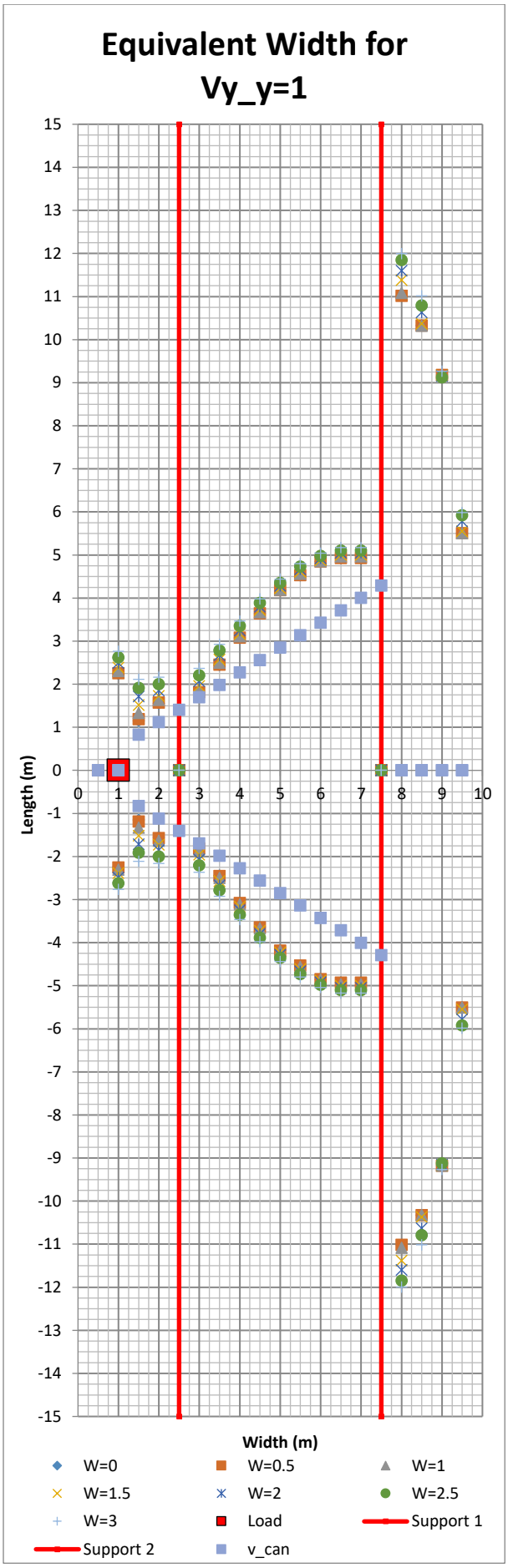

Equivalent Width for My_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

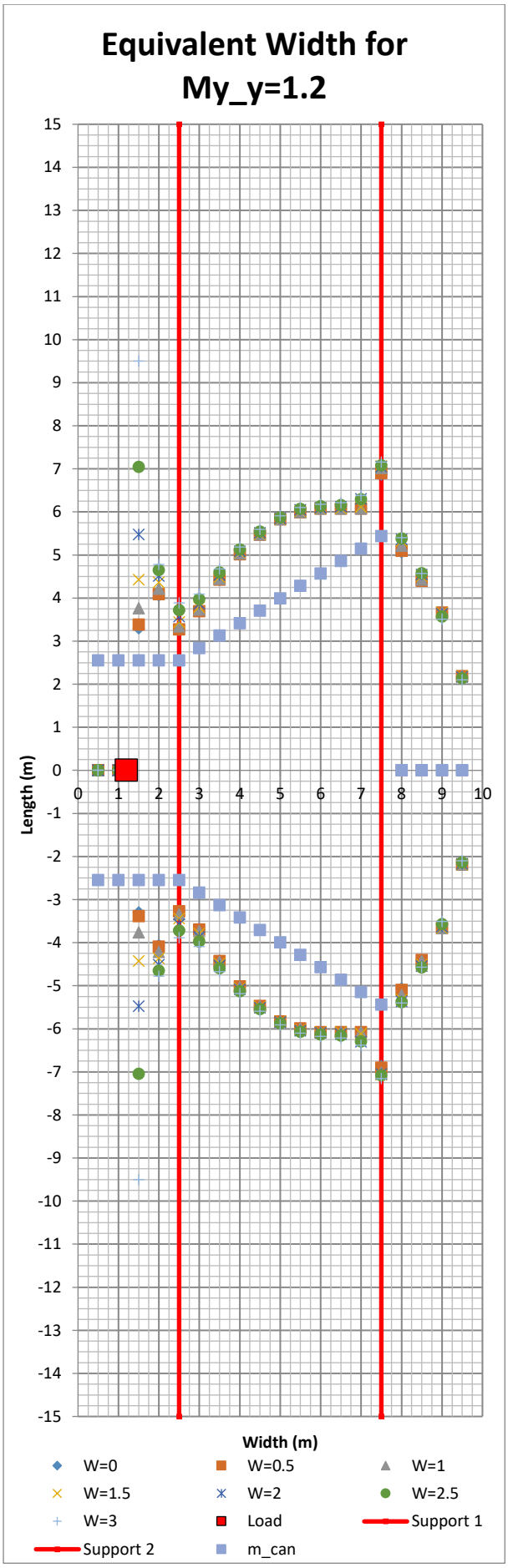

Equivalent Width for Vy_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

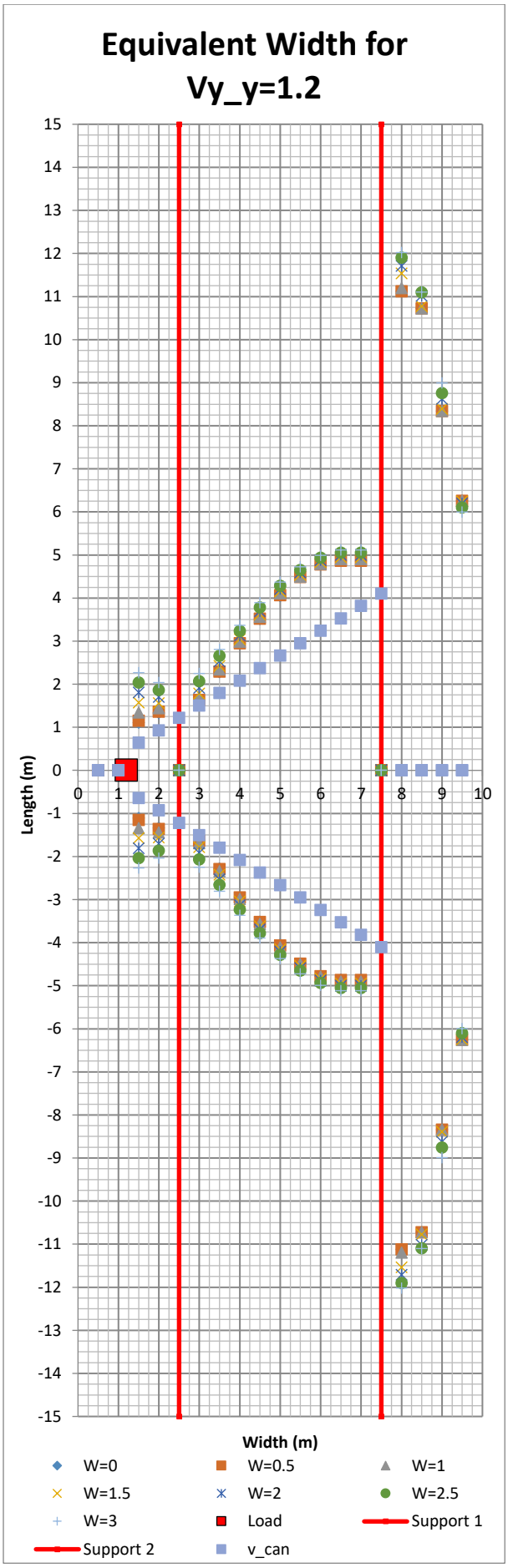

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

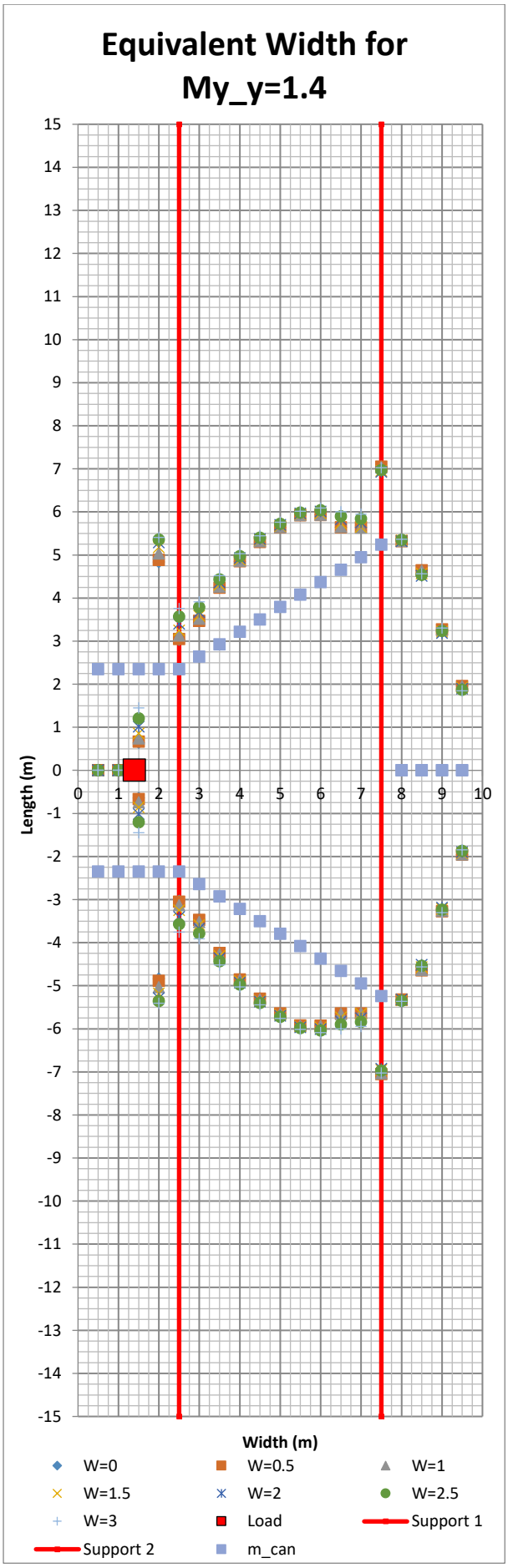

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

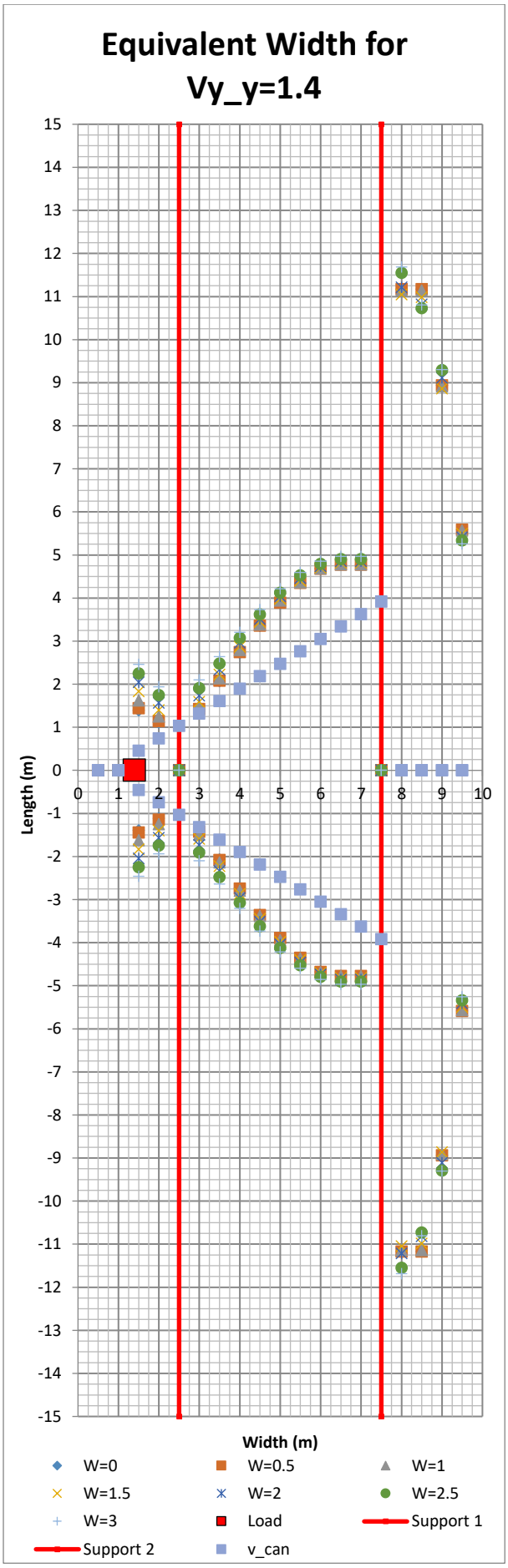

Equivalent Width for My_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

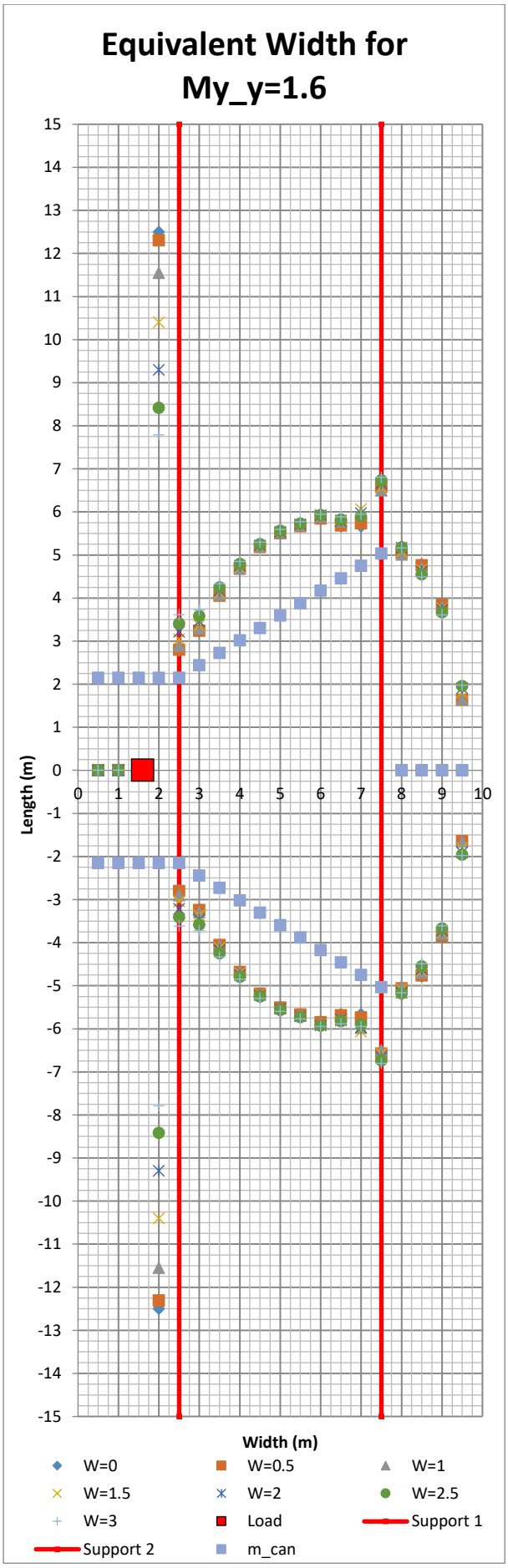

Equivalent Width for Vy_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

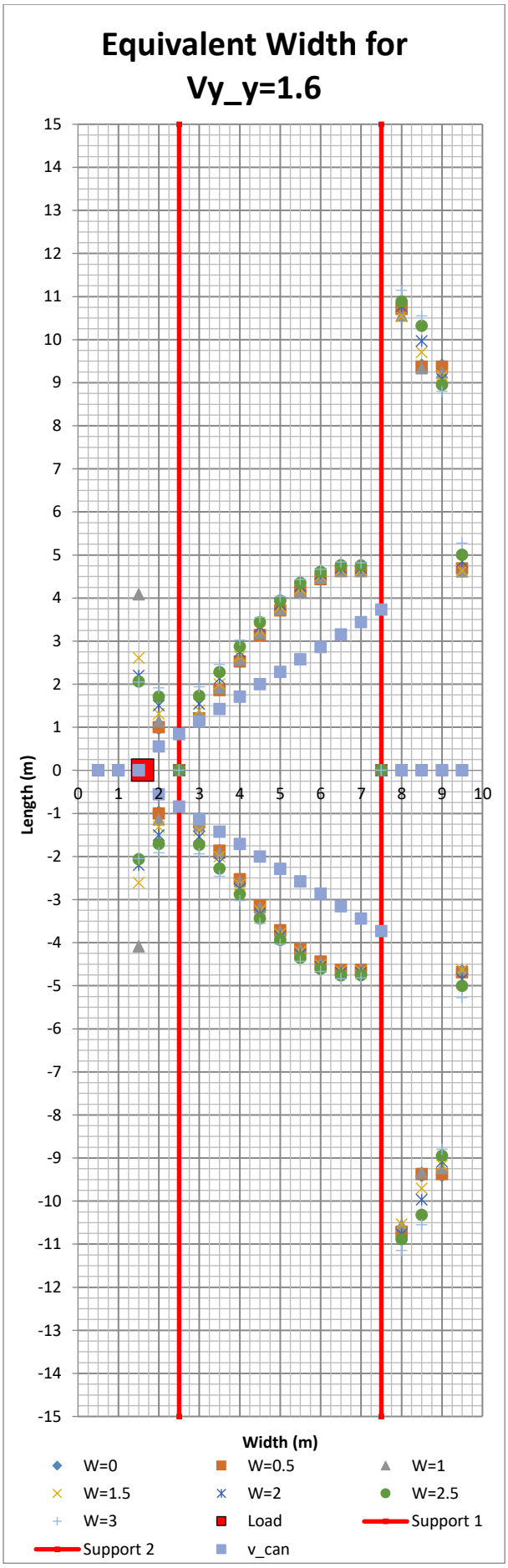

Equivalent Width for My_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

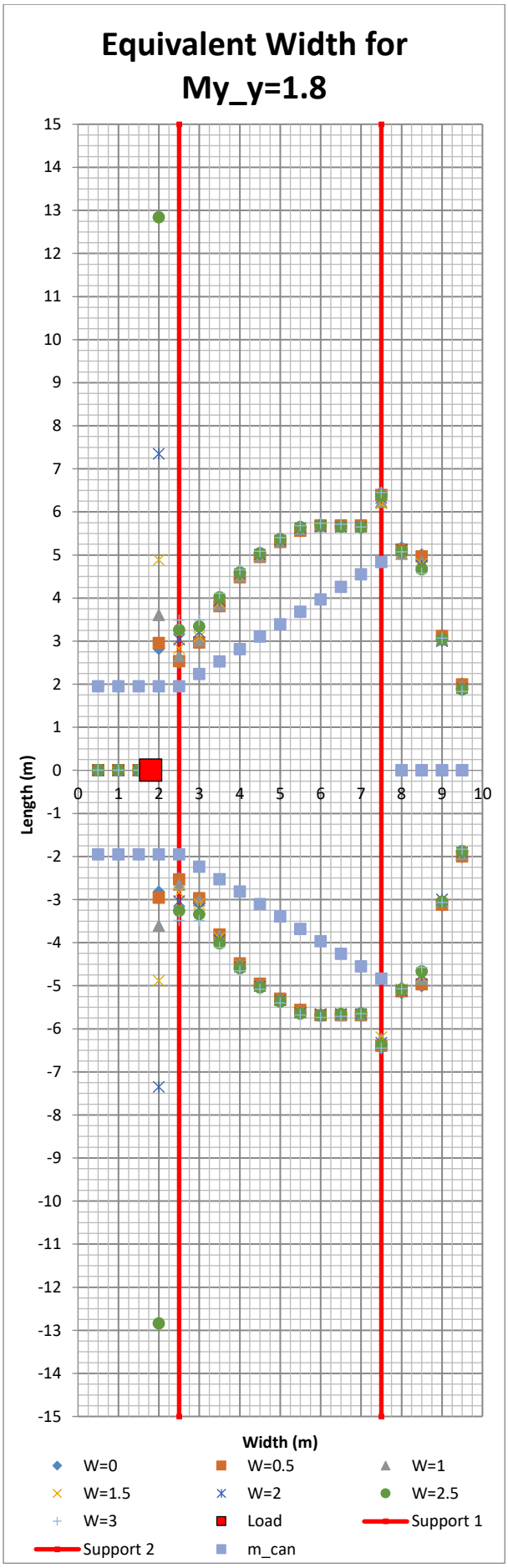

Equivalent Width for Vy_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

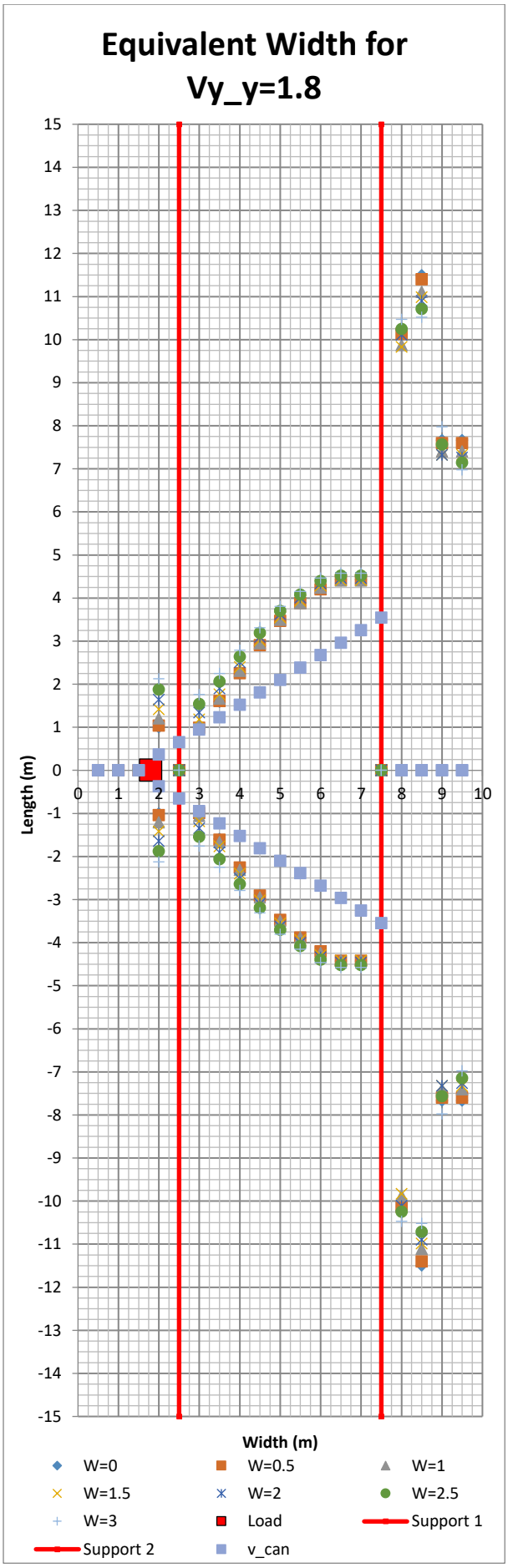

Equivalent Width for My_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

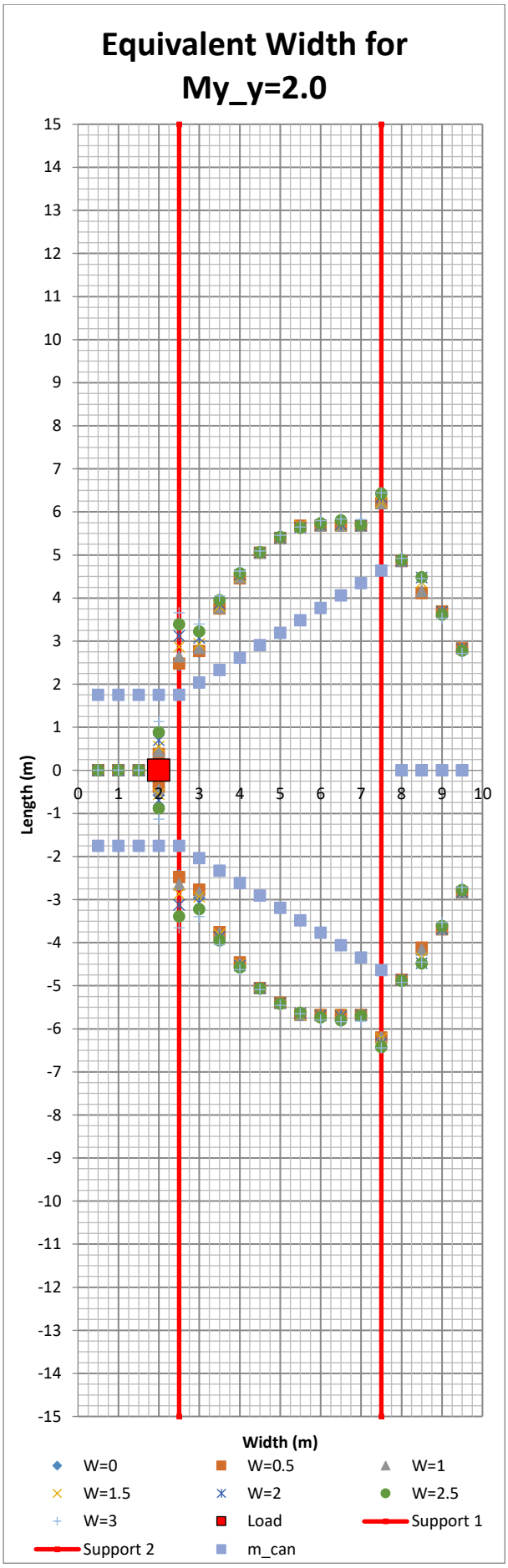

Equivalent Width for Vy_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

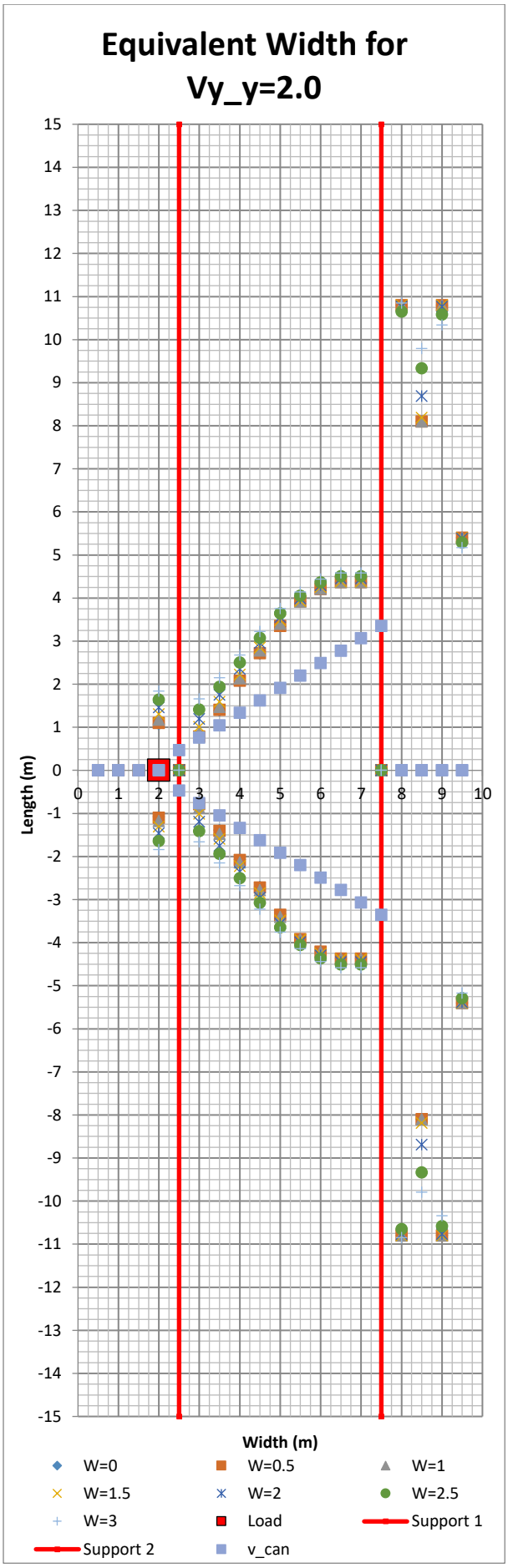

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

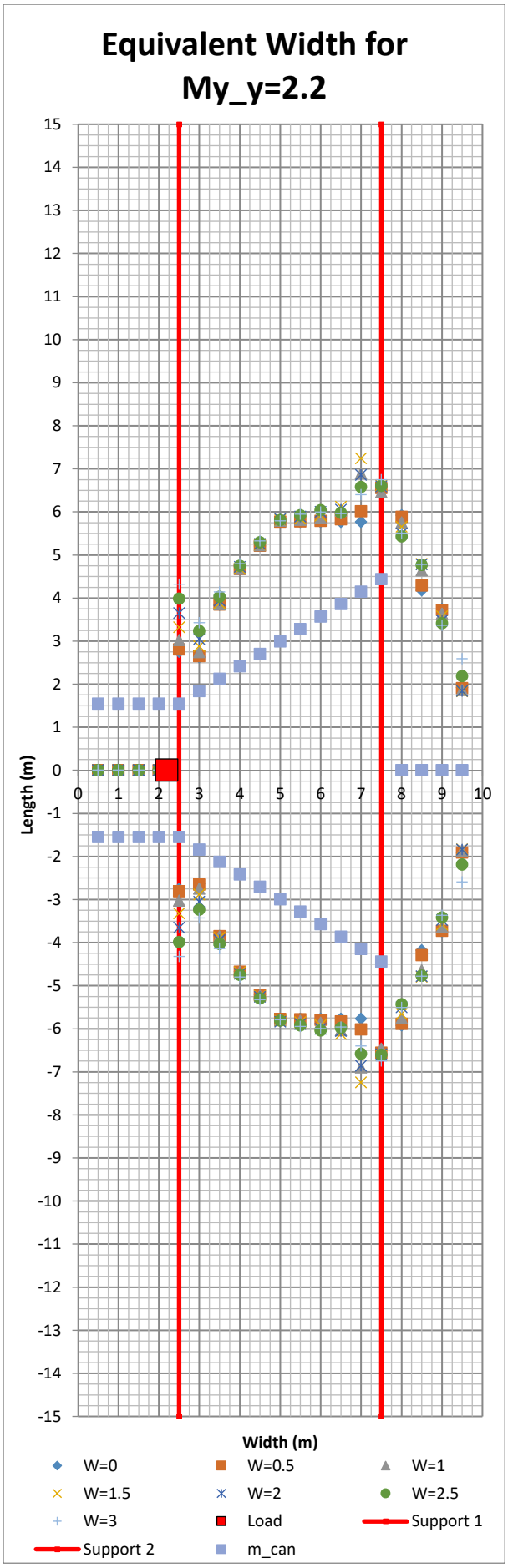

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

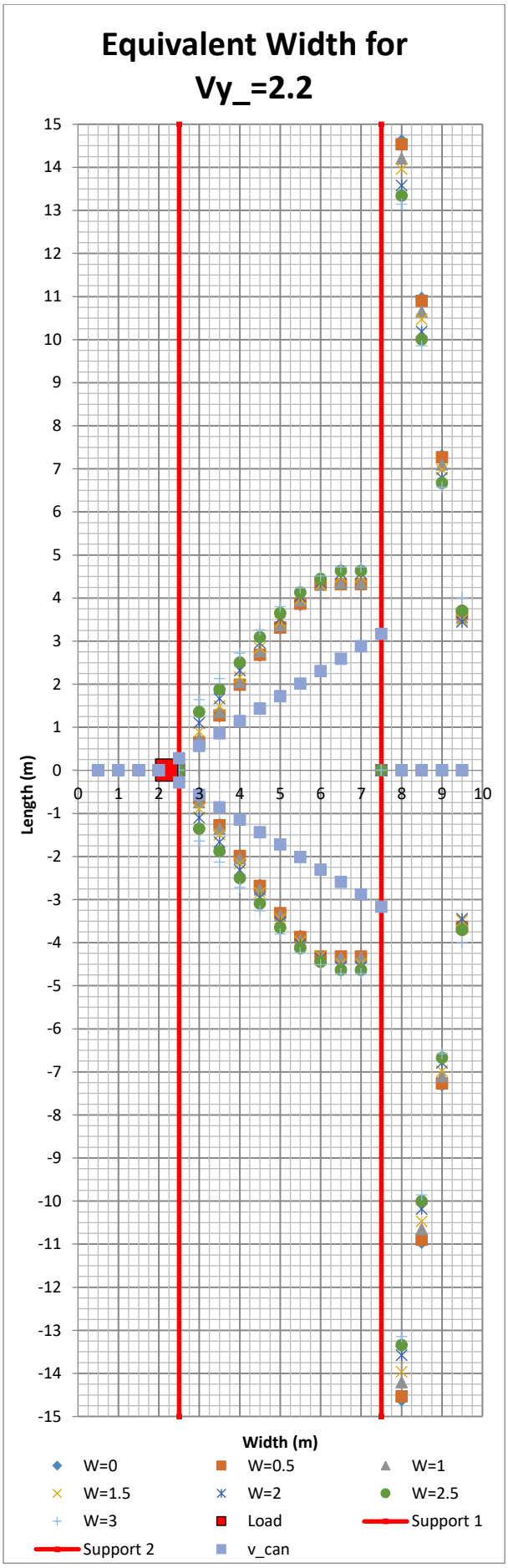

Equivalent Width for My_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

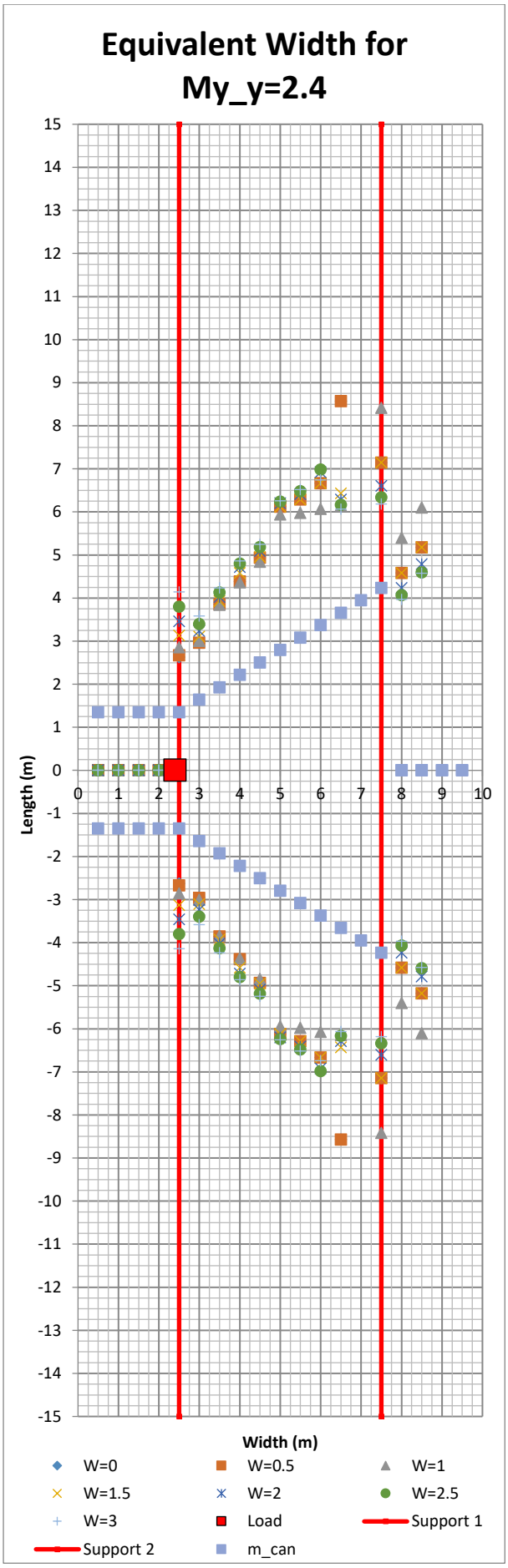

Equivalent Width for Vy_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

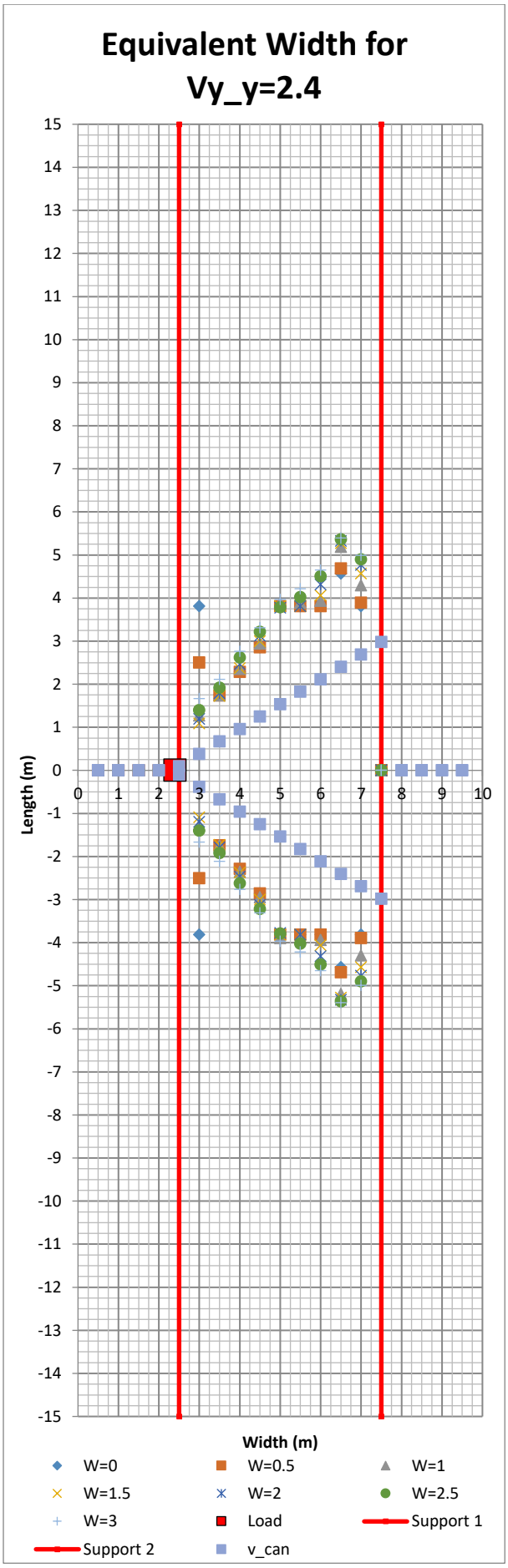

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

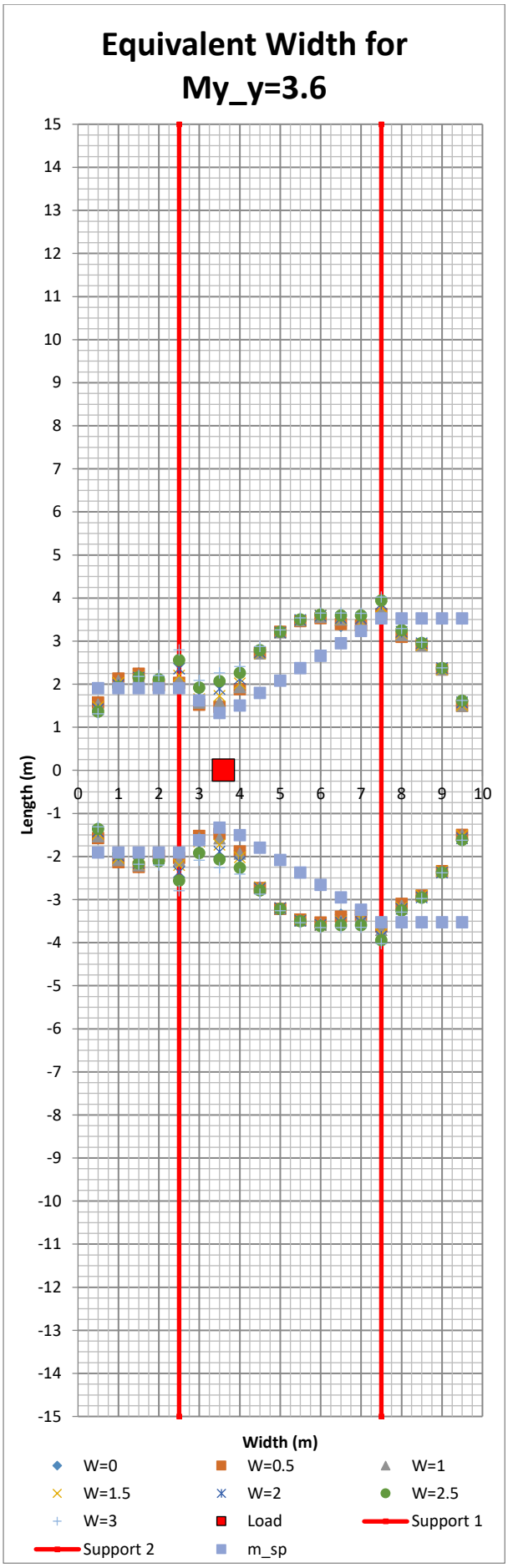

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

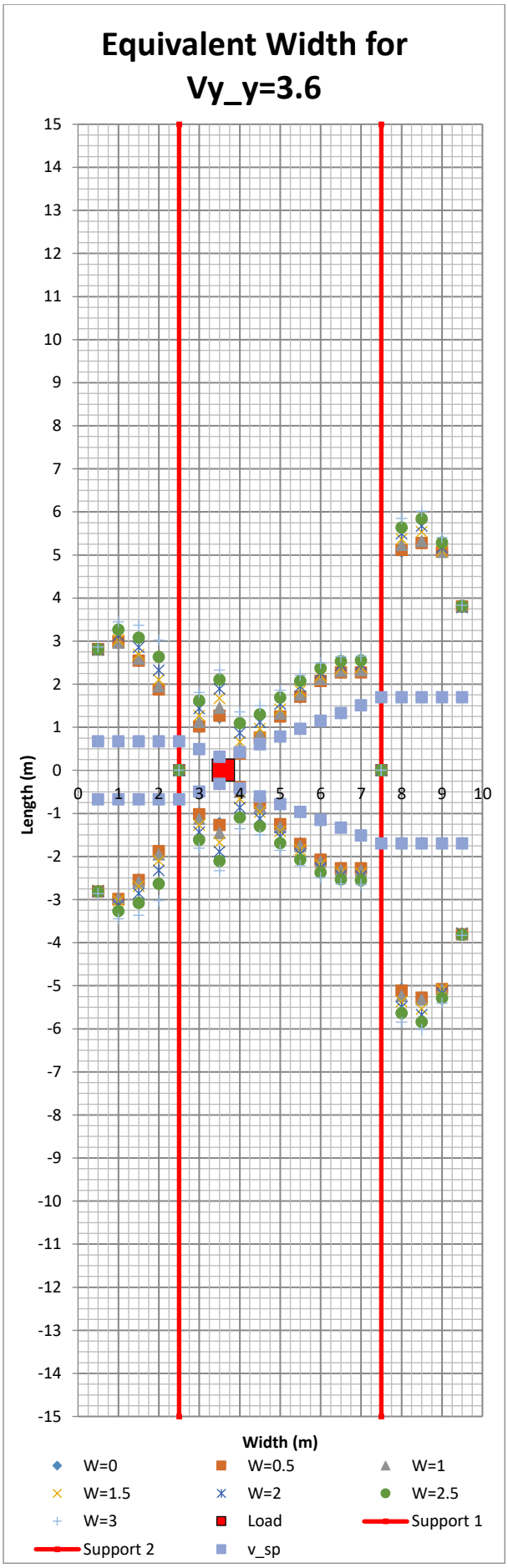

Equivalent Width for My_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

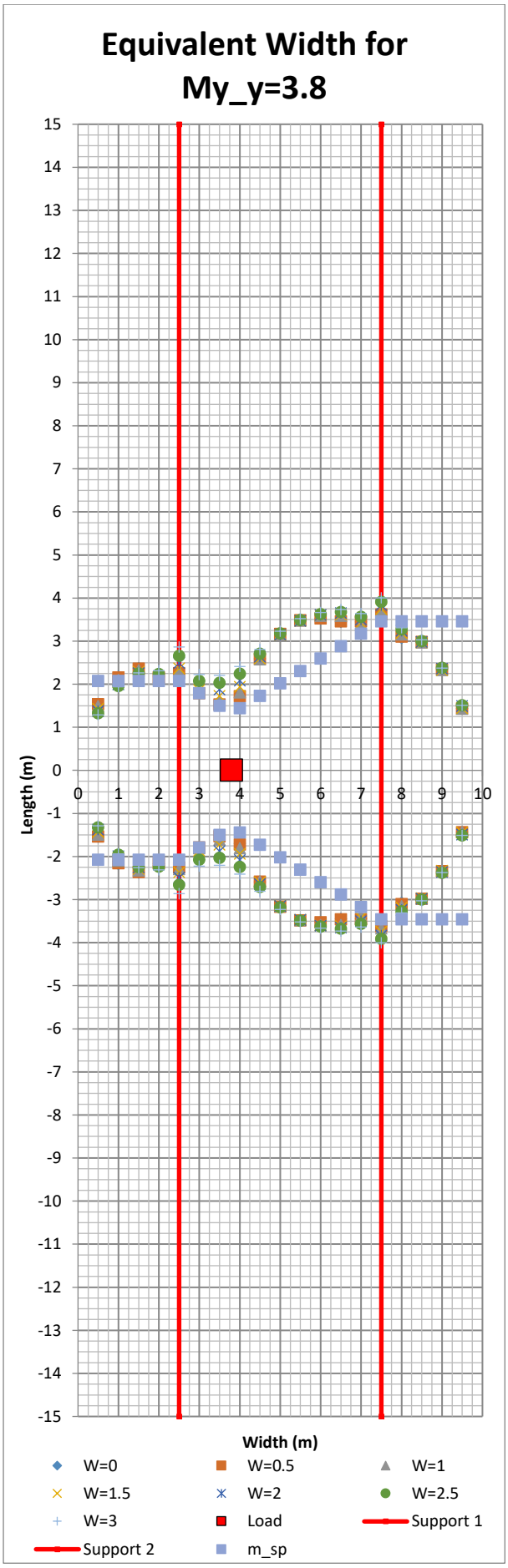

Equivalent Width for Vy_=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

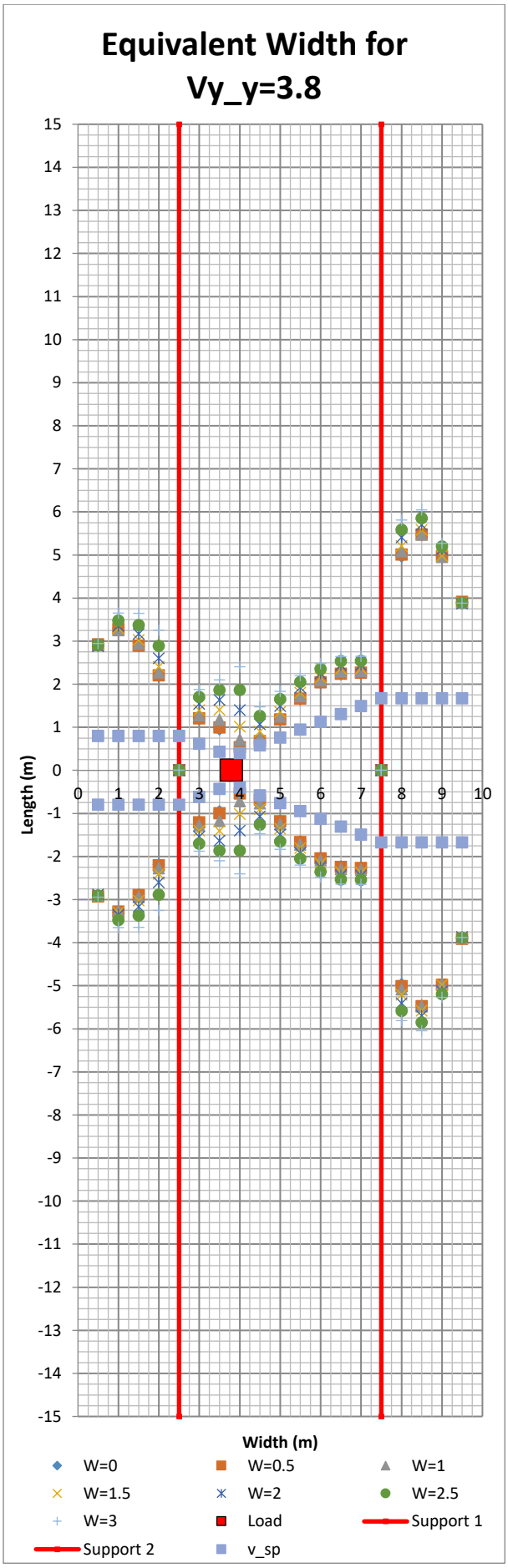

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

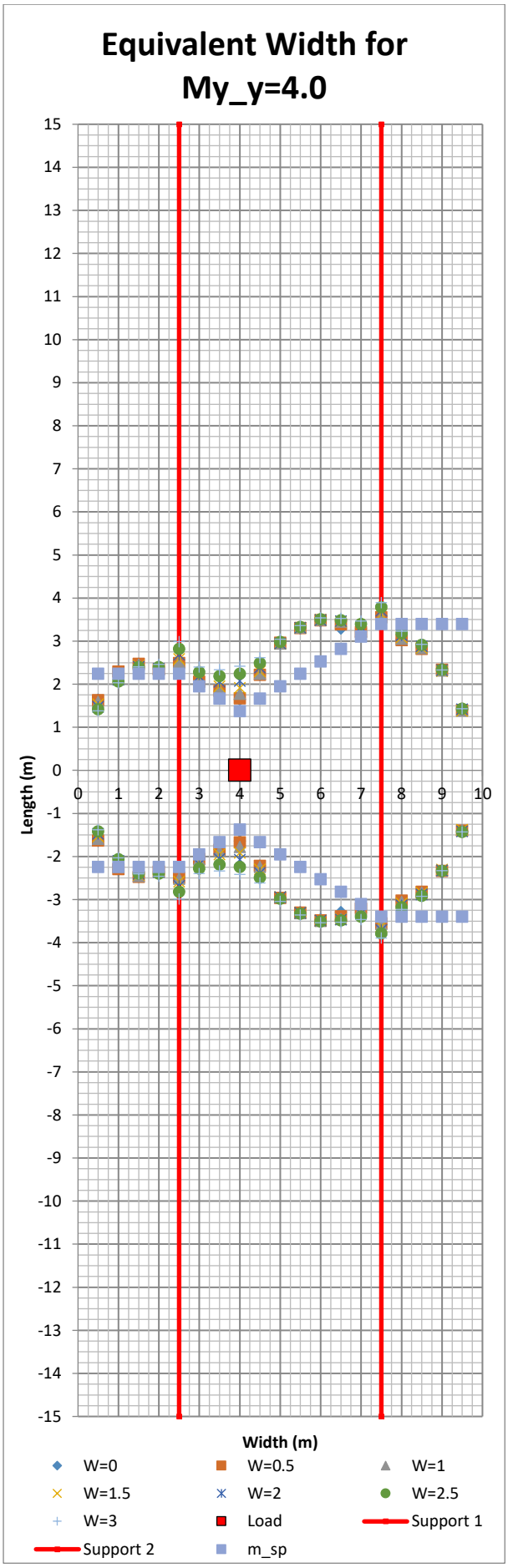

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

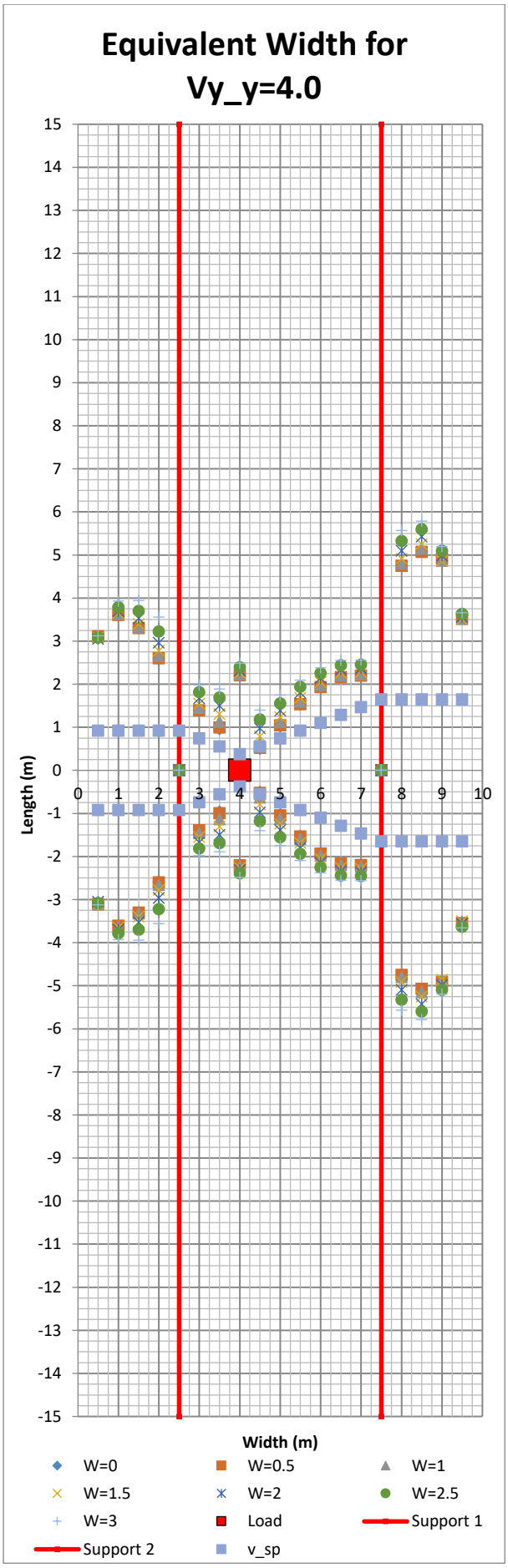

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

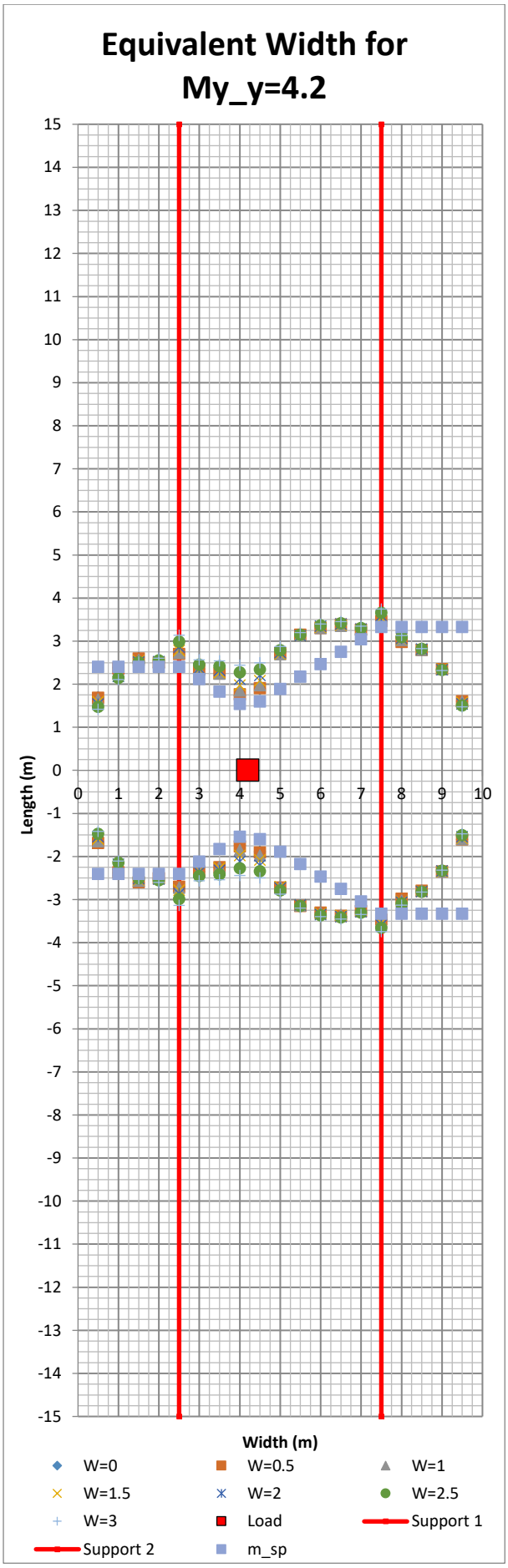

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

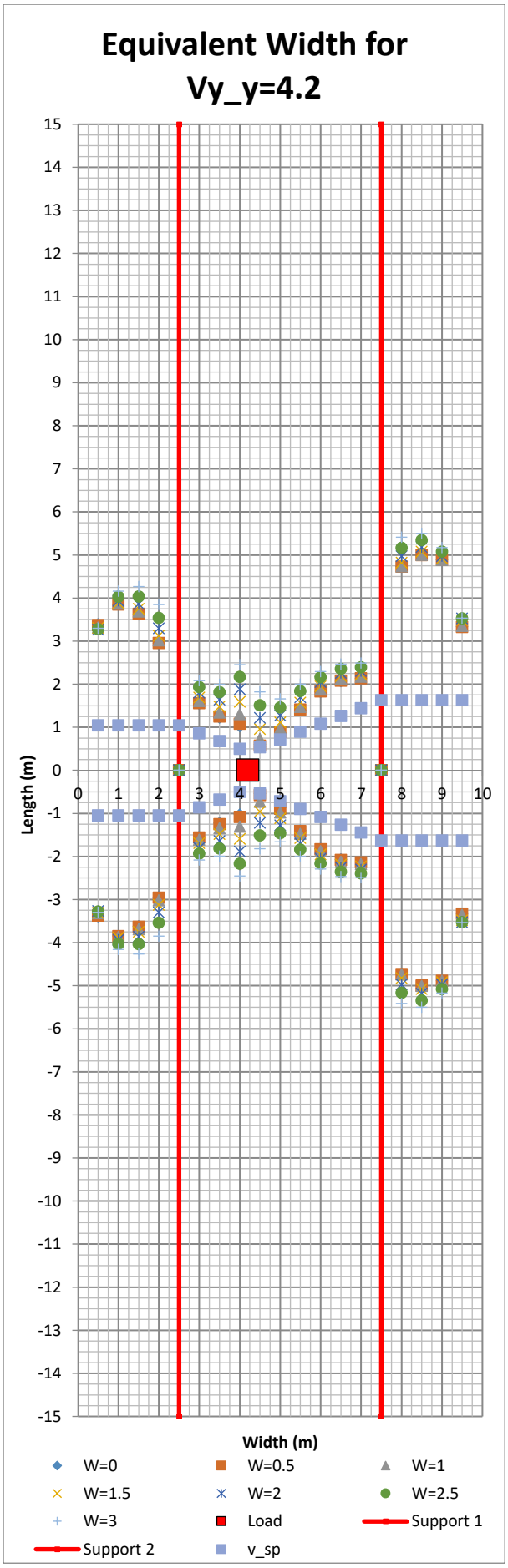

Equivalent Width for My_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

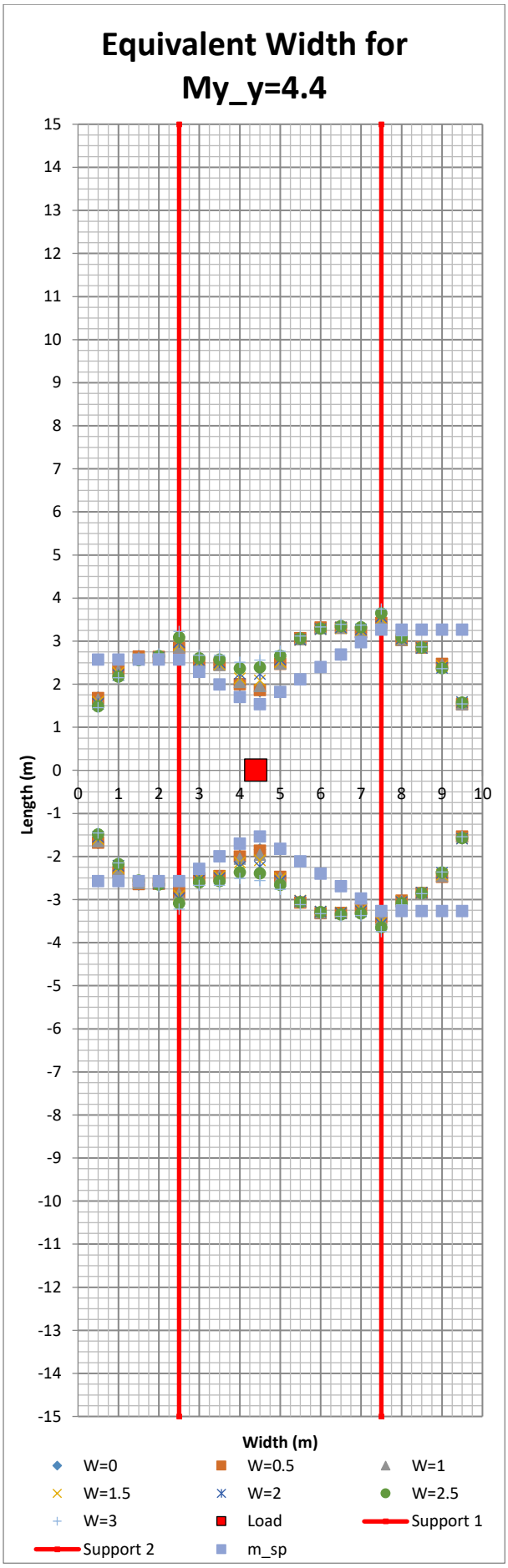

Equivalent Width for Vy_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

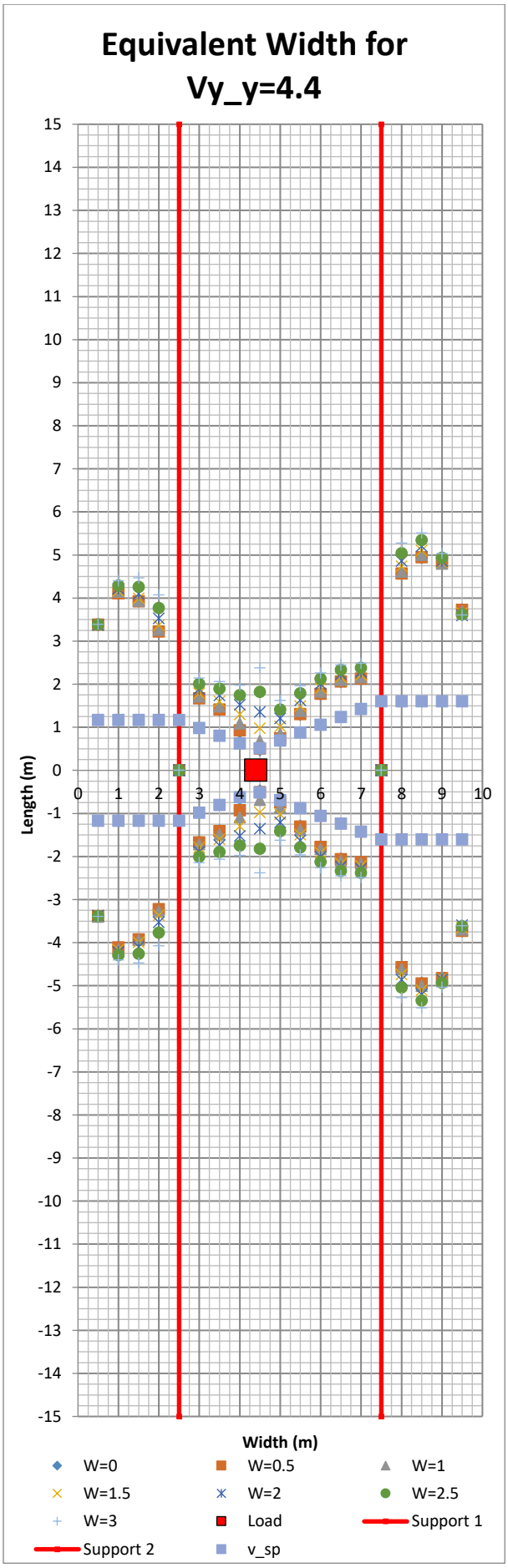

Equivalent Width for My_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

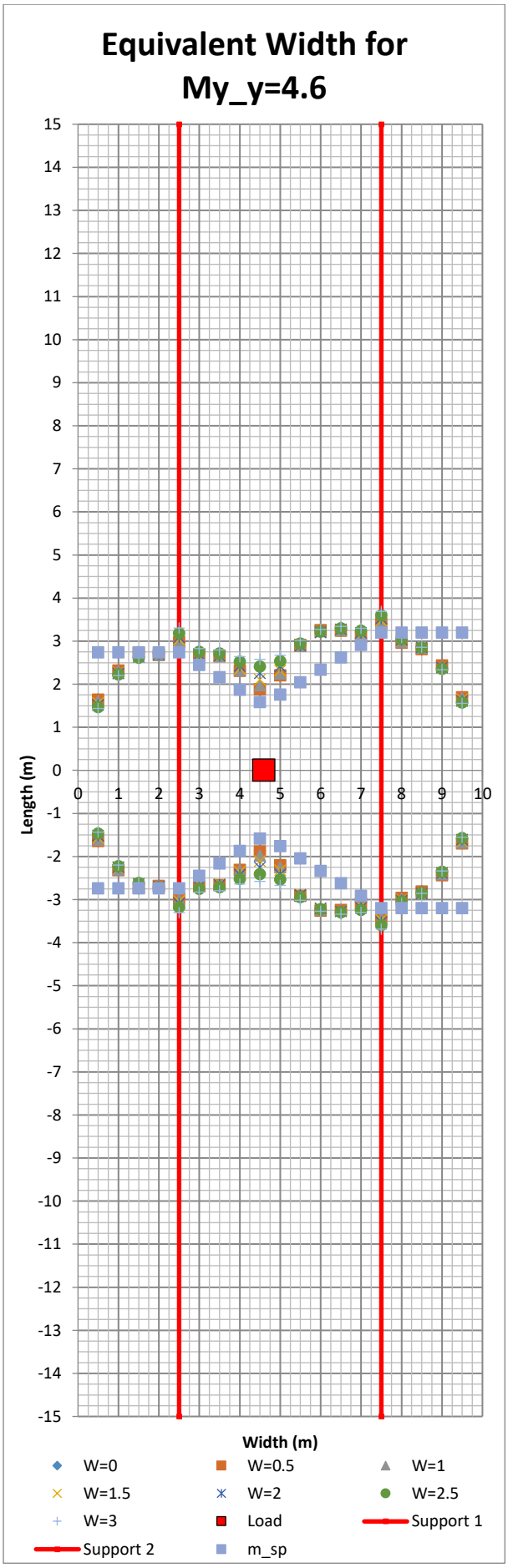

Equivalent Width for Vy_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

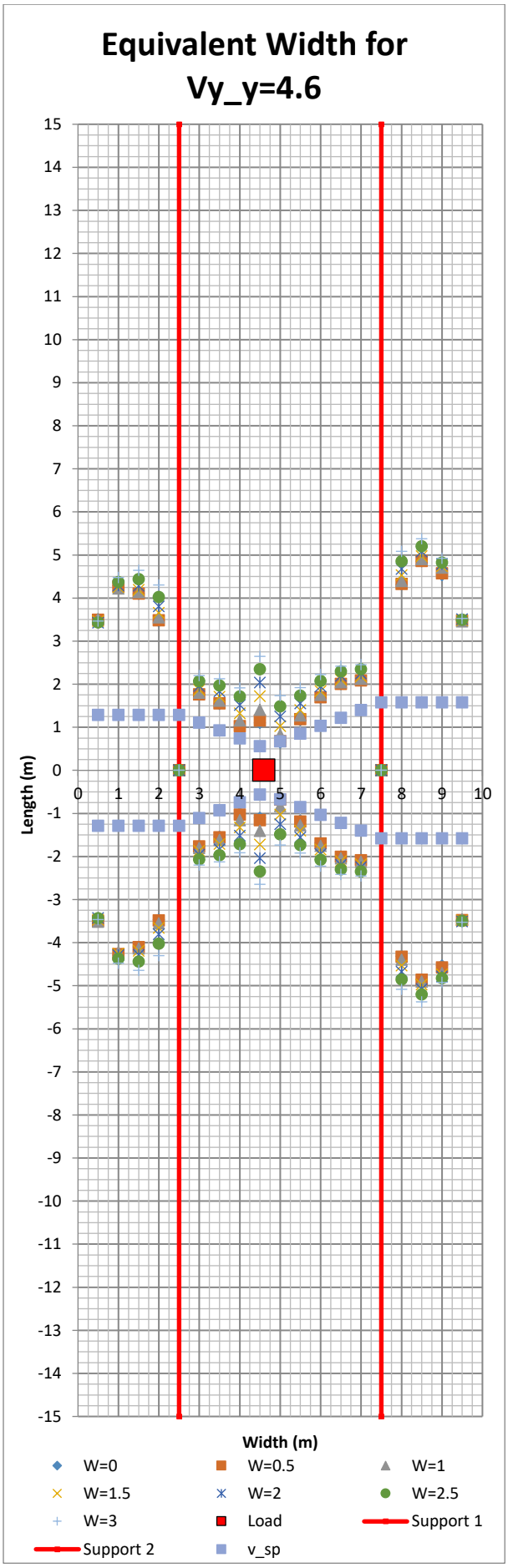

Equivalent Width for My_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

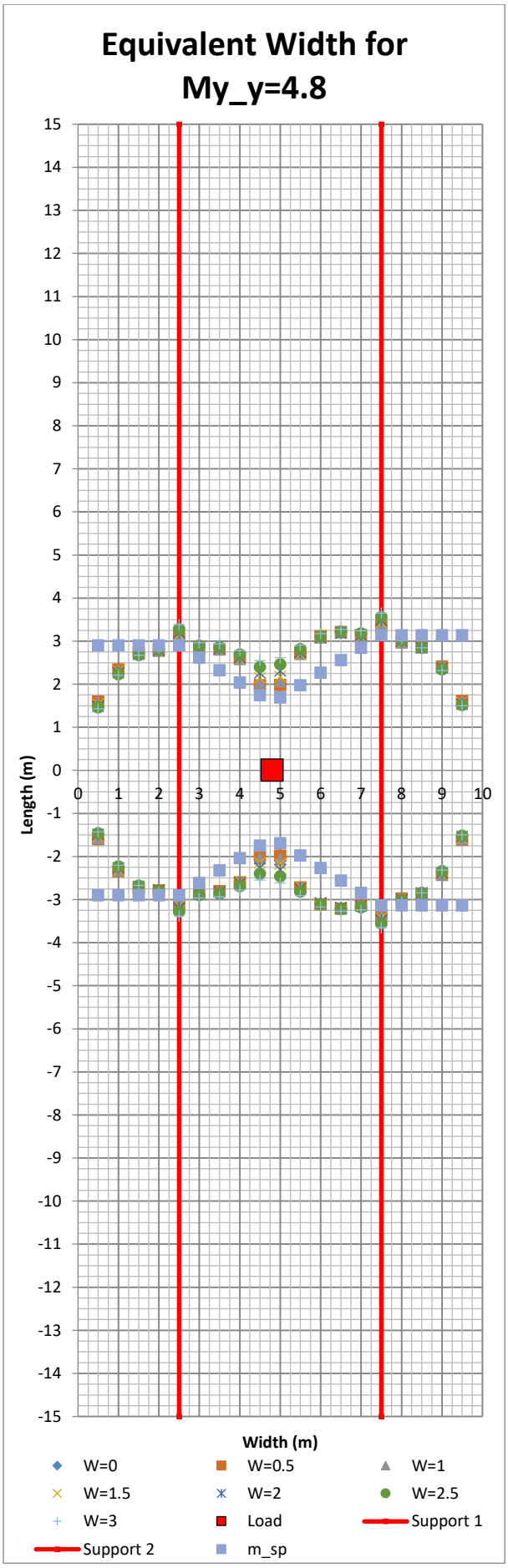

Equivalent Width for Vy_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

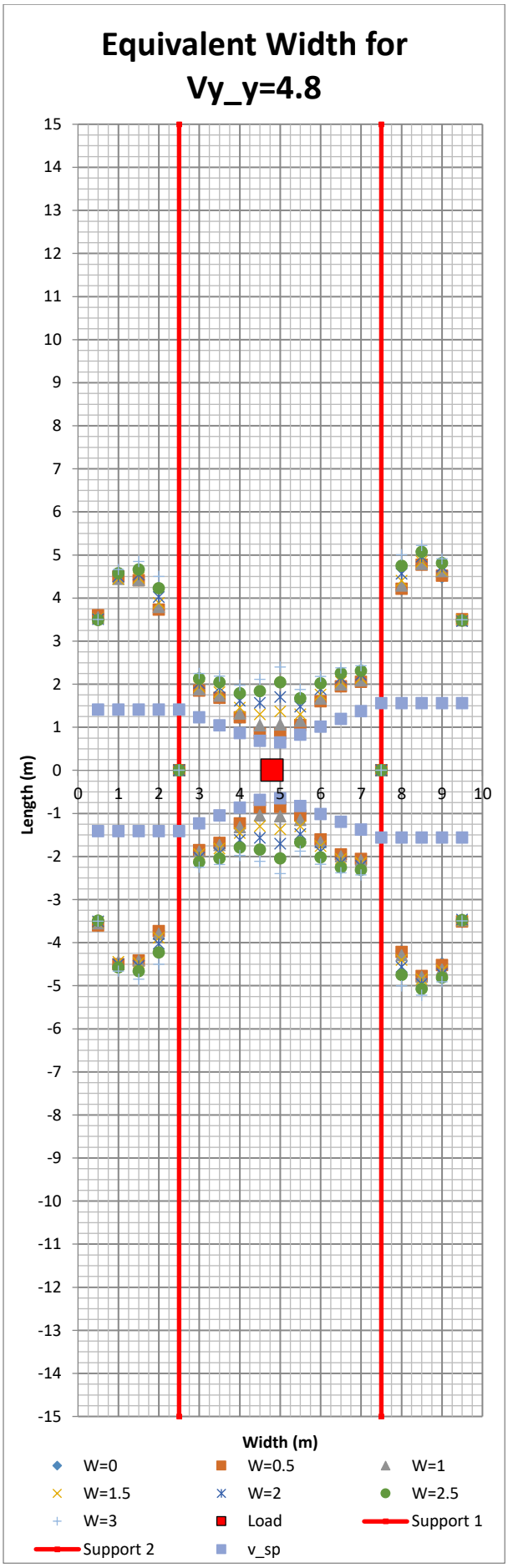

Equivalent Width for My_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

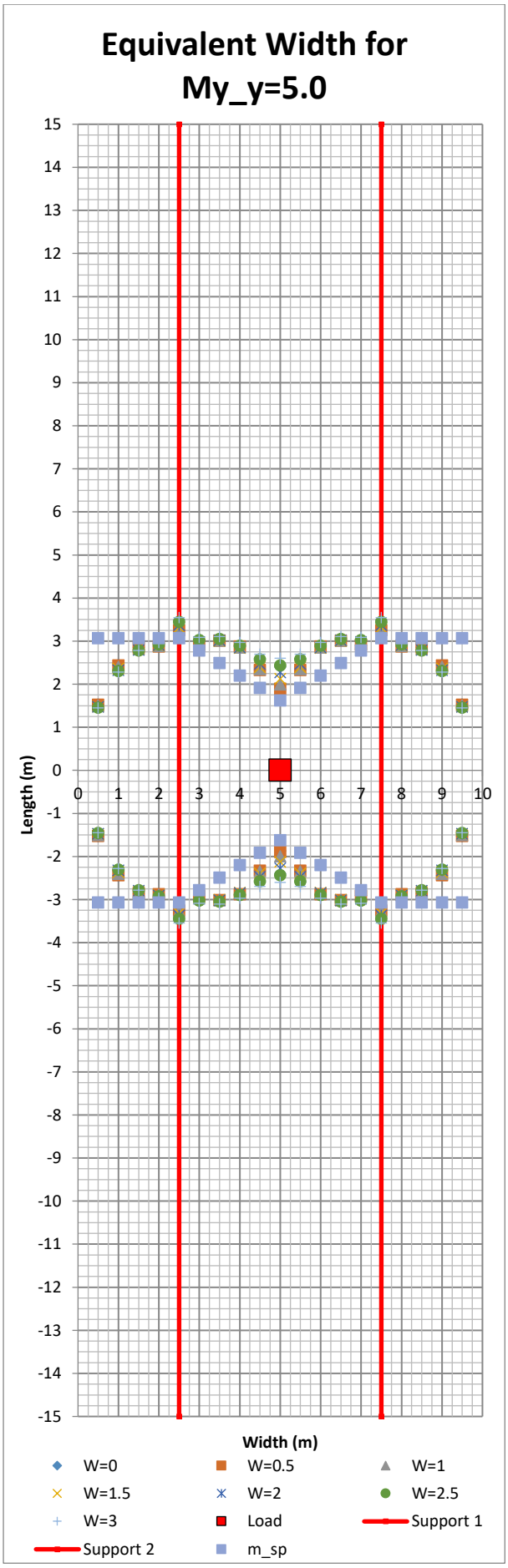

Equivalent Width for Vy_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

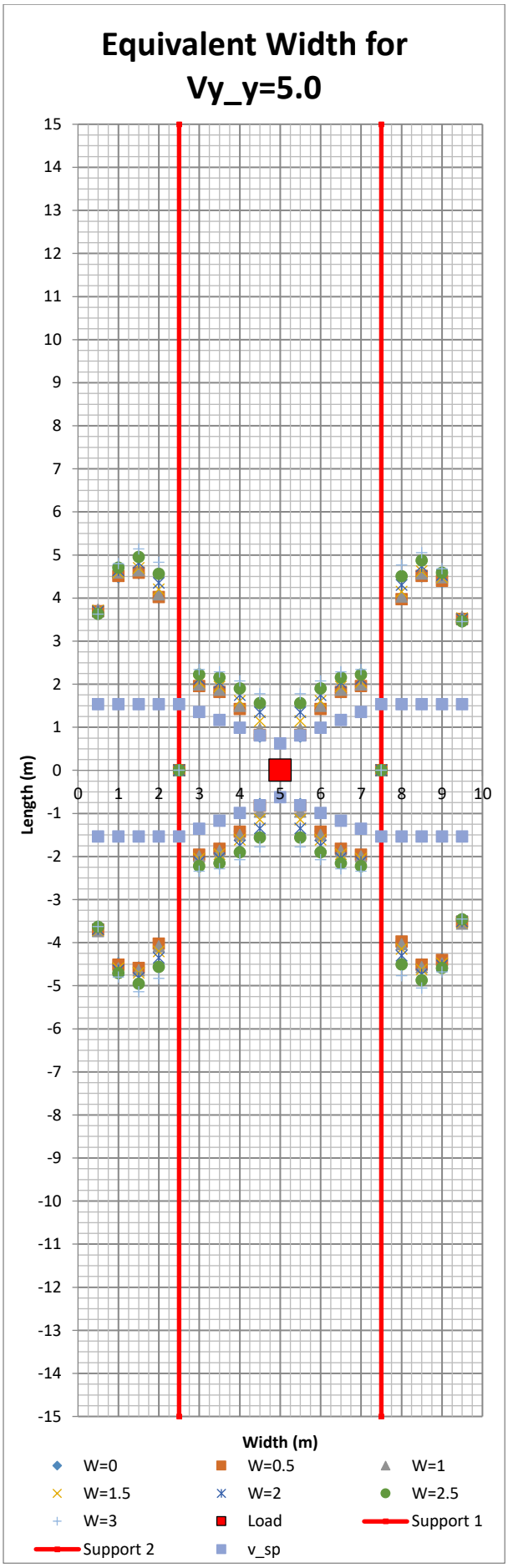

Equivalent Width for My_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

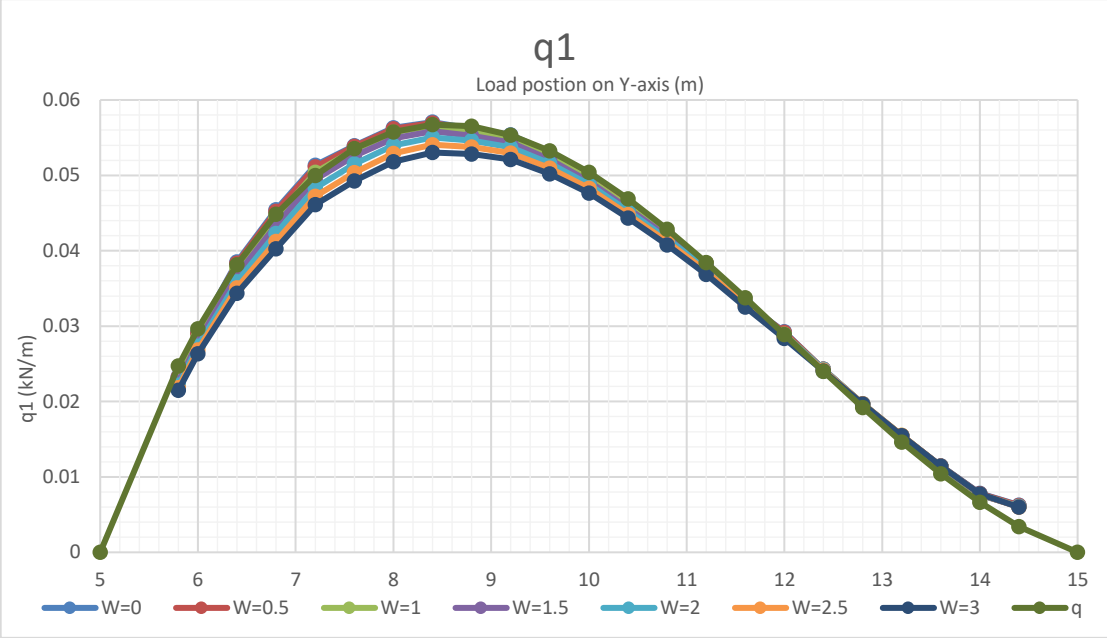

Equivalent Width for My_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

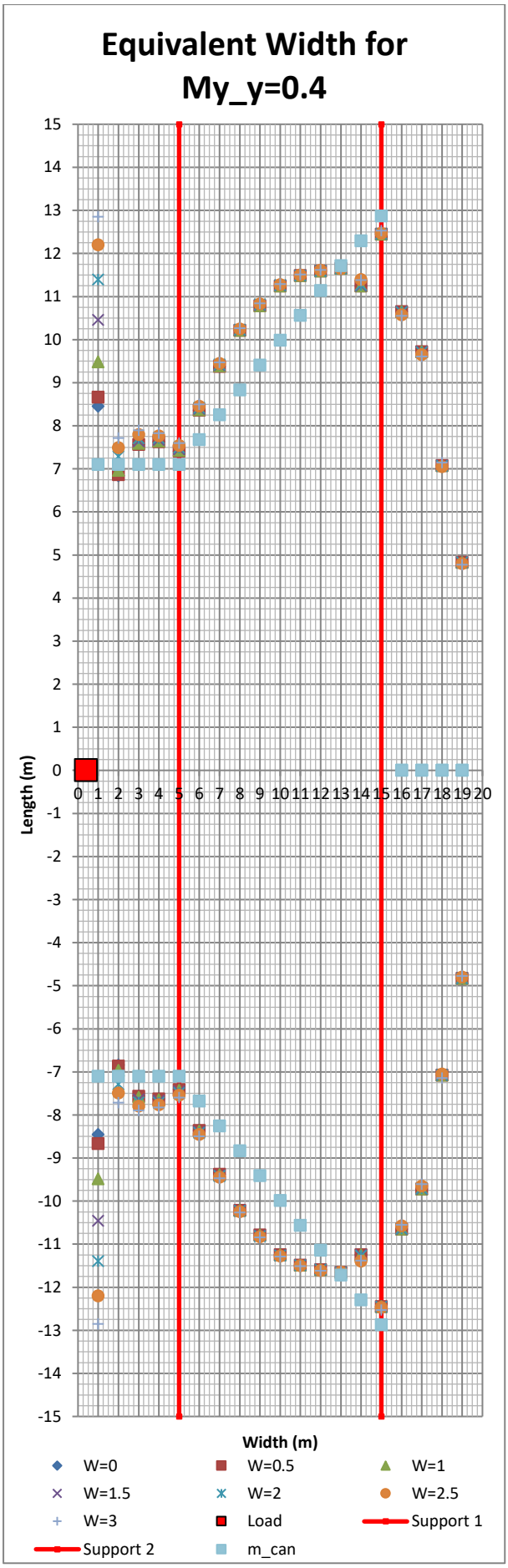

Equivalent Width for Vy_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

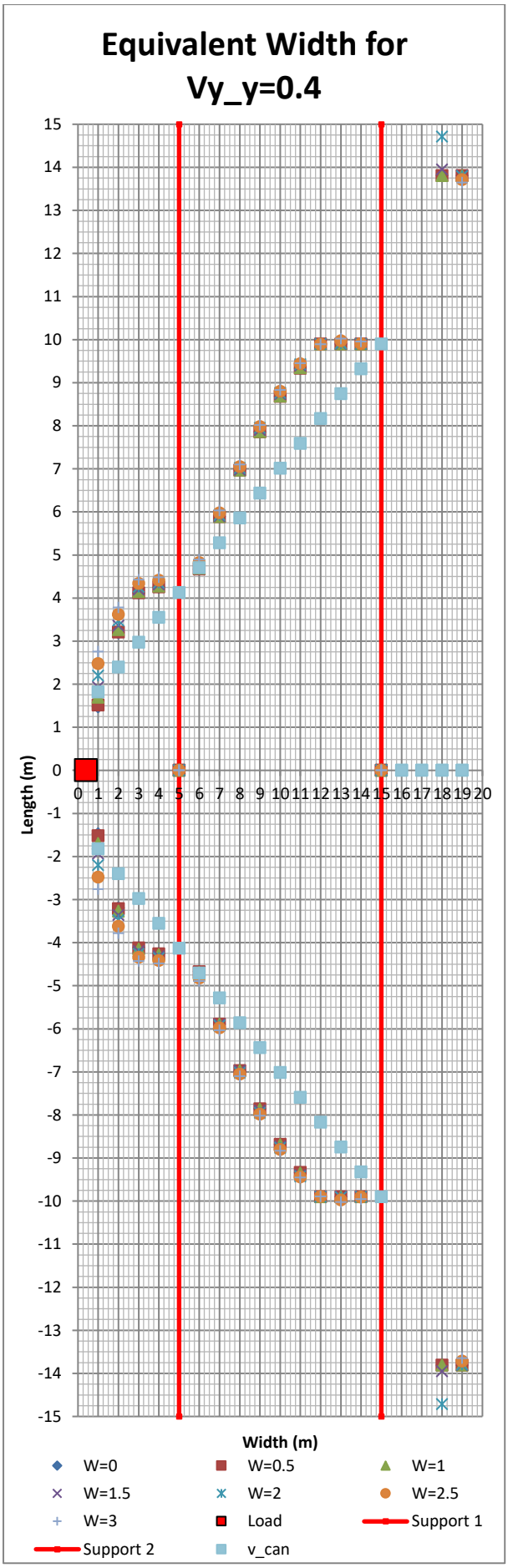

Equivalent Width for My_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

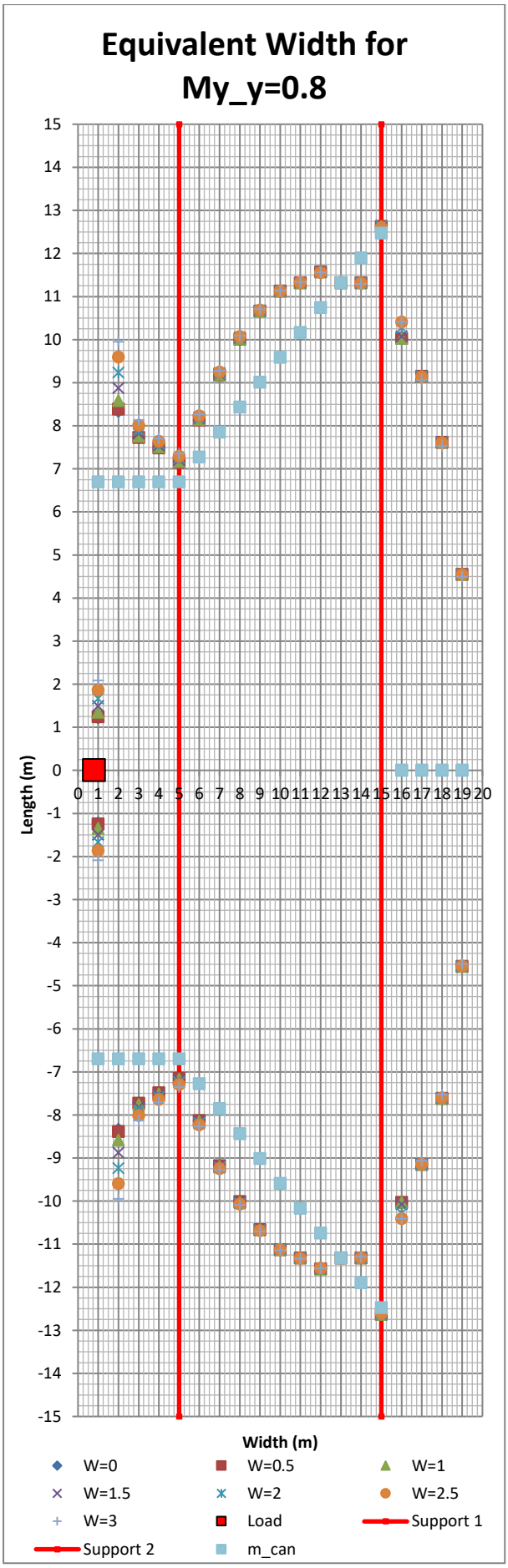

Equivalent Width for Vy_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

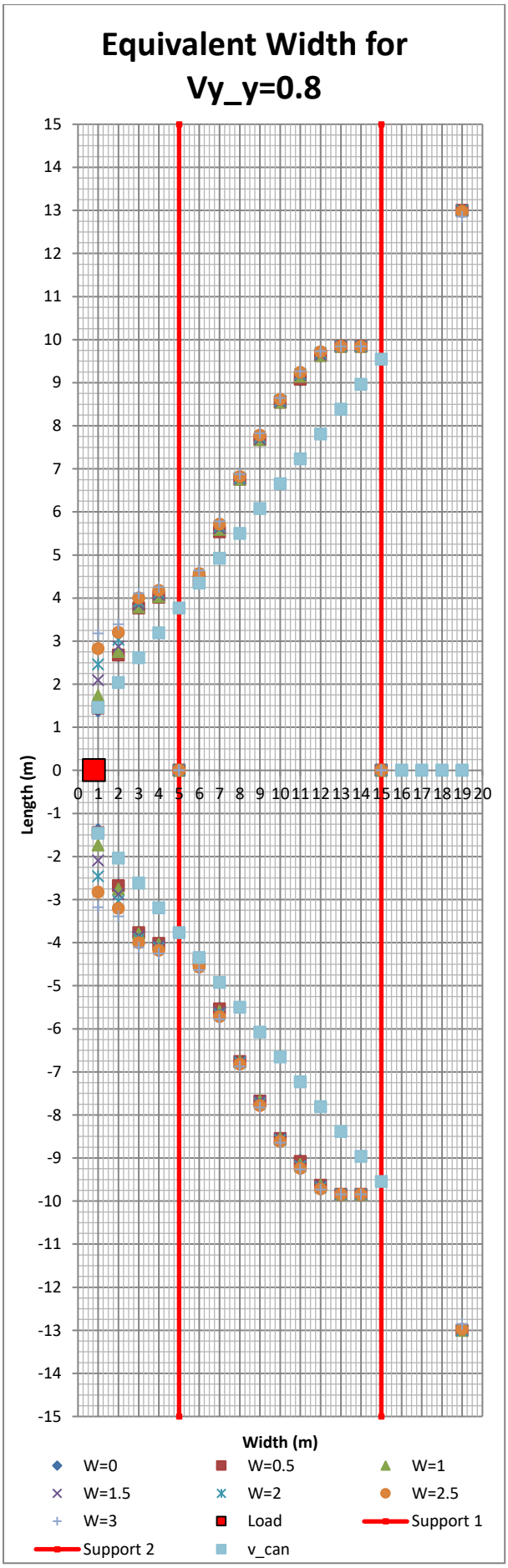
11.3 Third Variation $l_m + 2l_k = 20m$

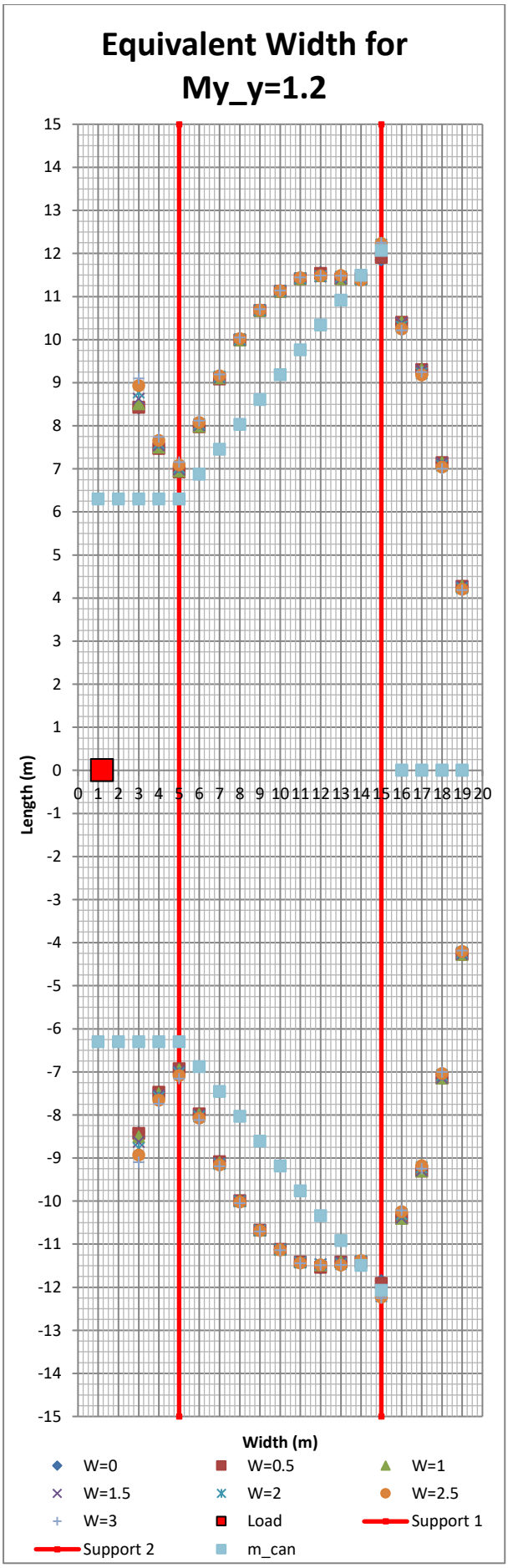


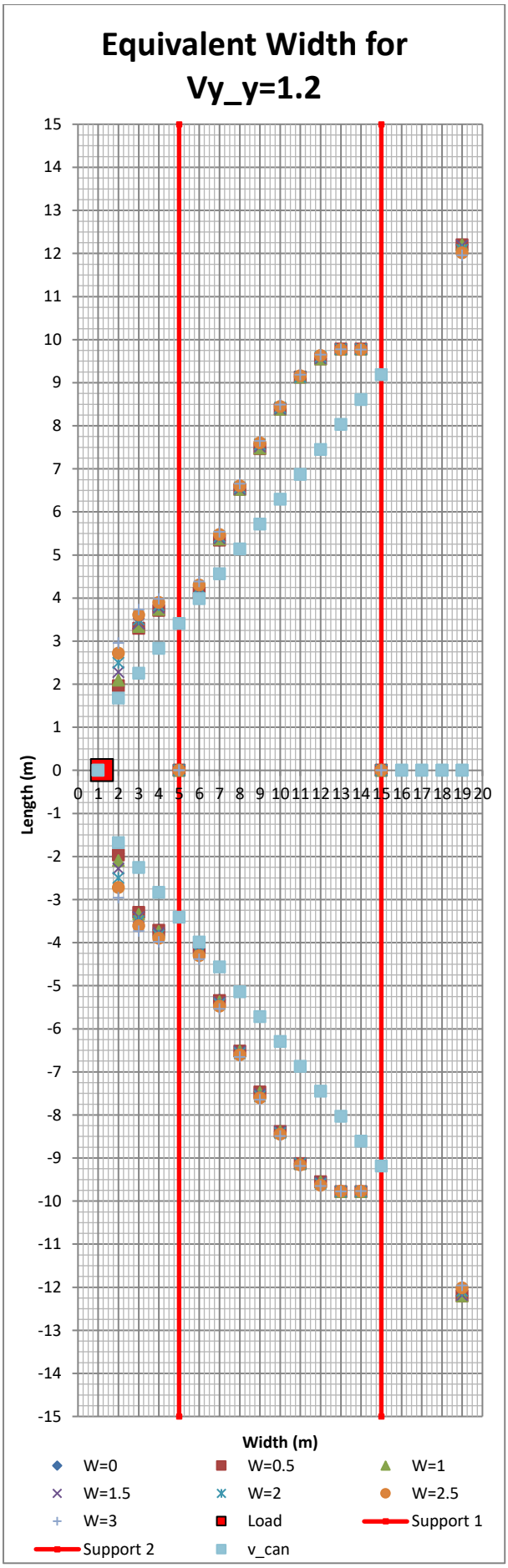

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

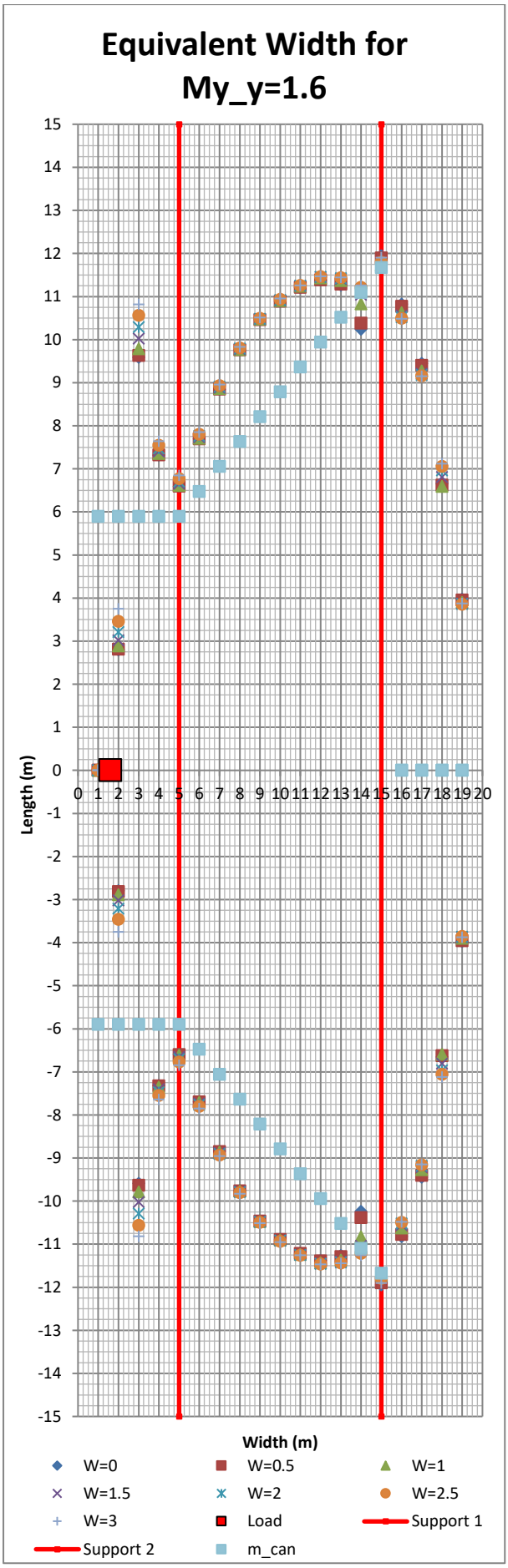

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

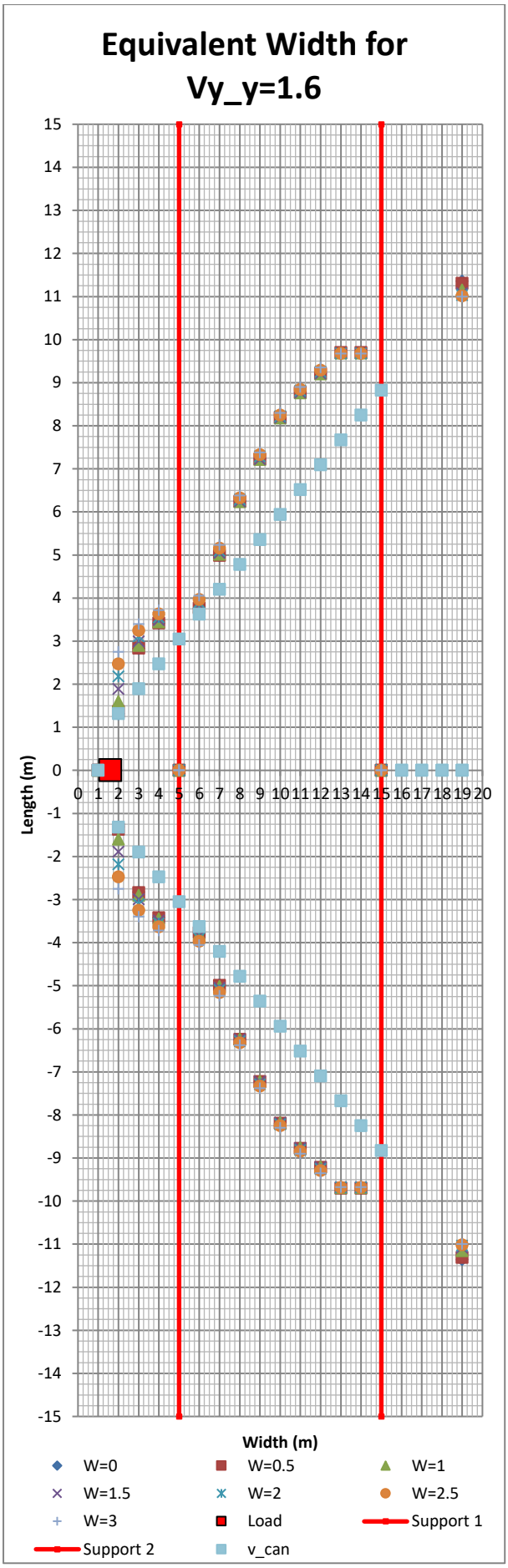

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

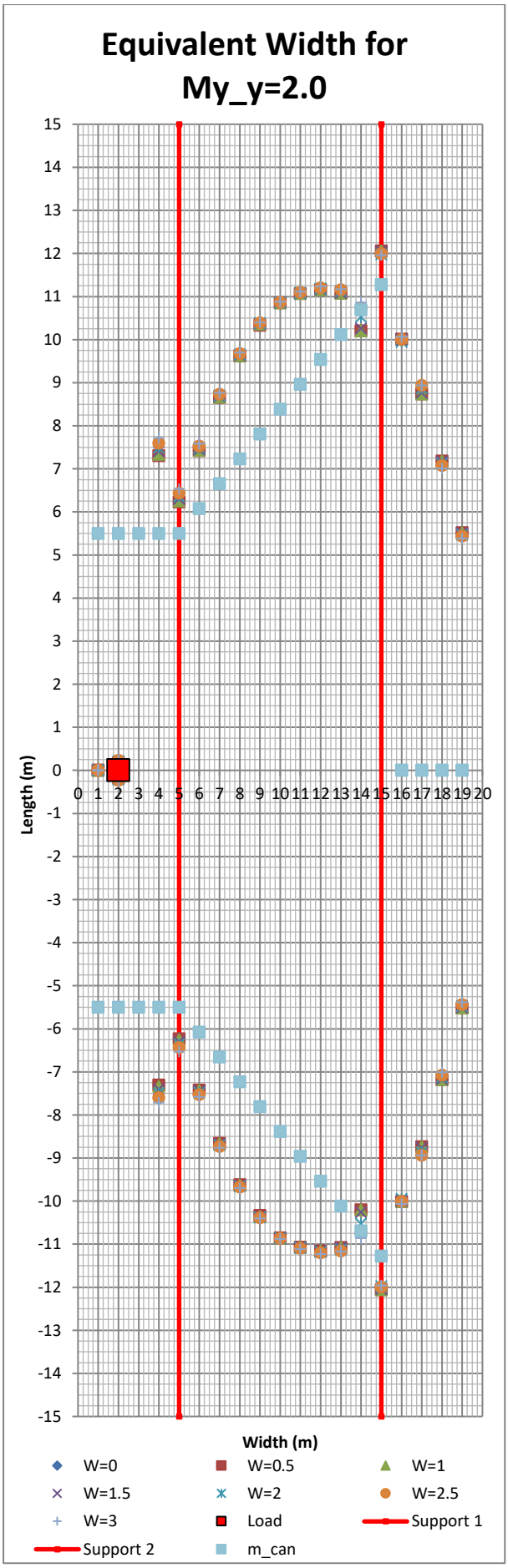

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

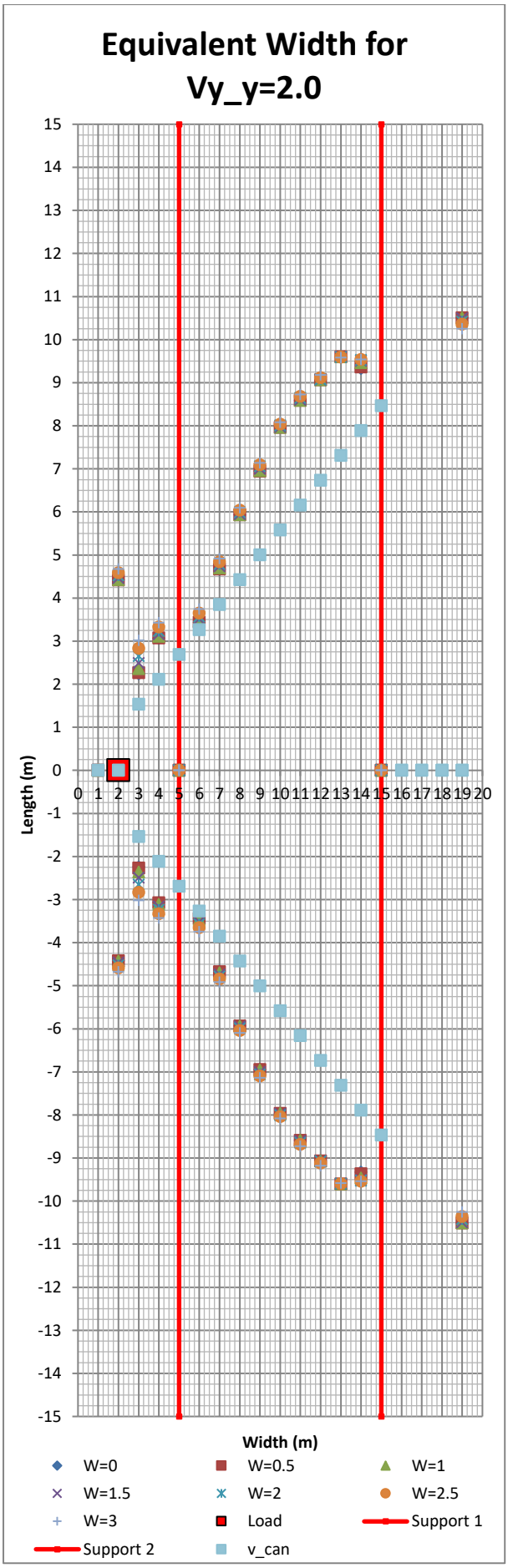

Equivalent Width for My_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

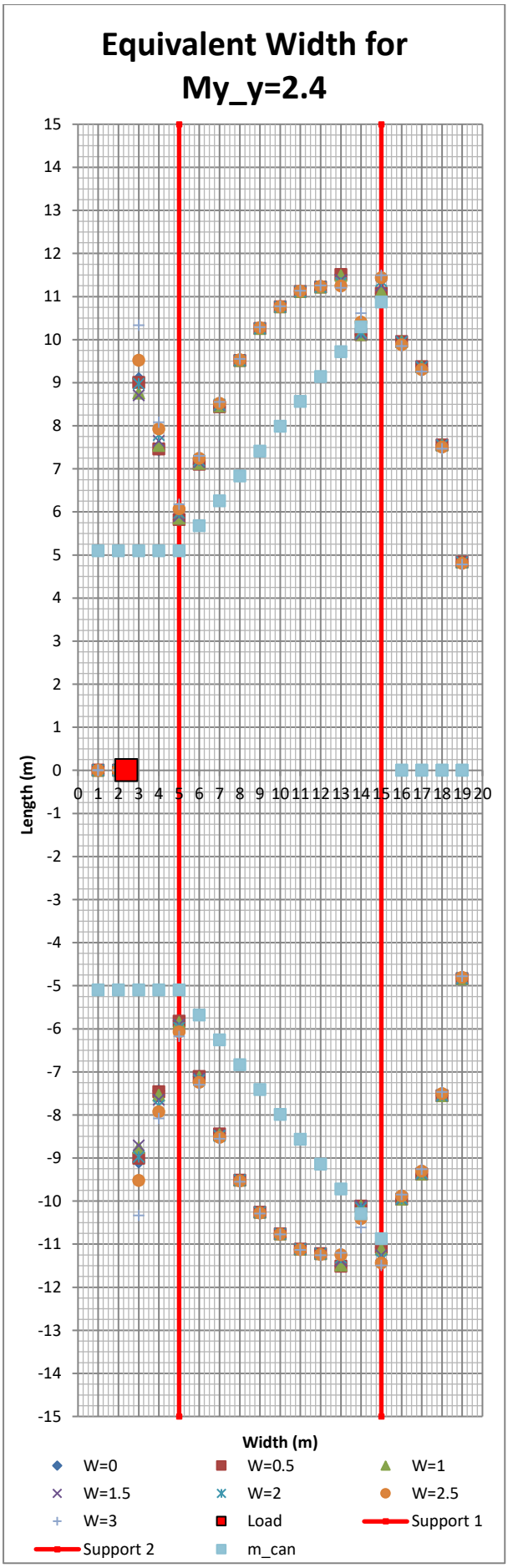

Equivalent Width for Vy_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

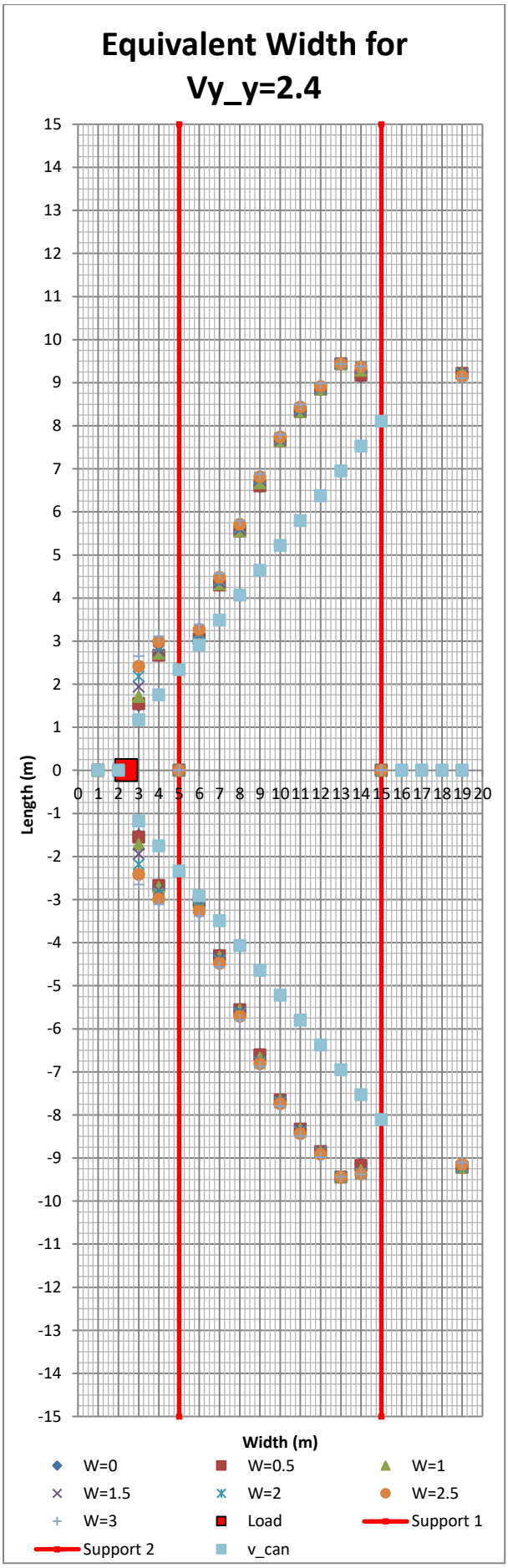

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

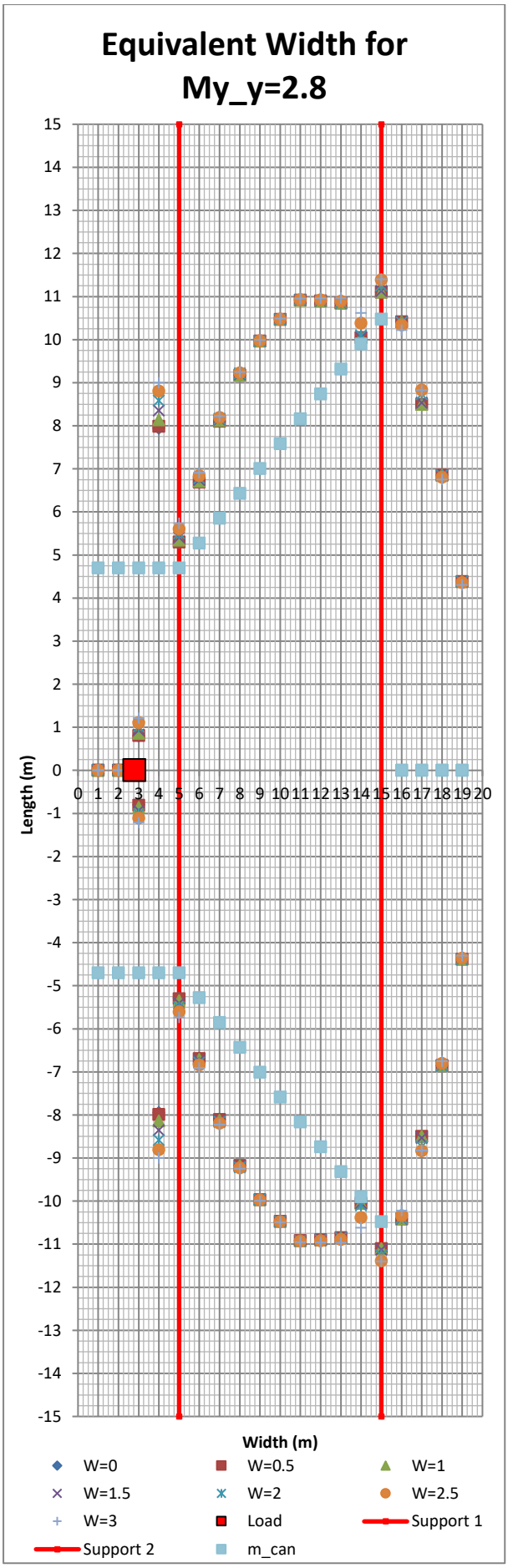

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

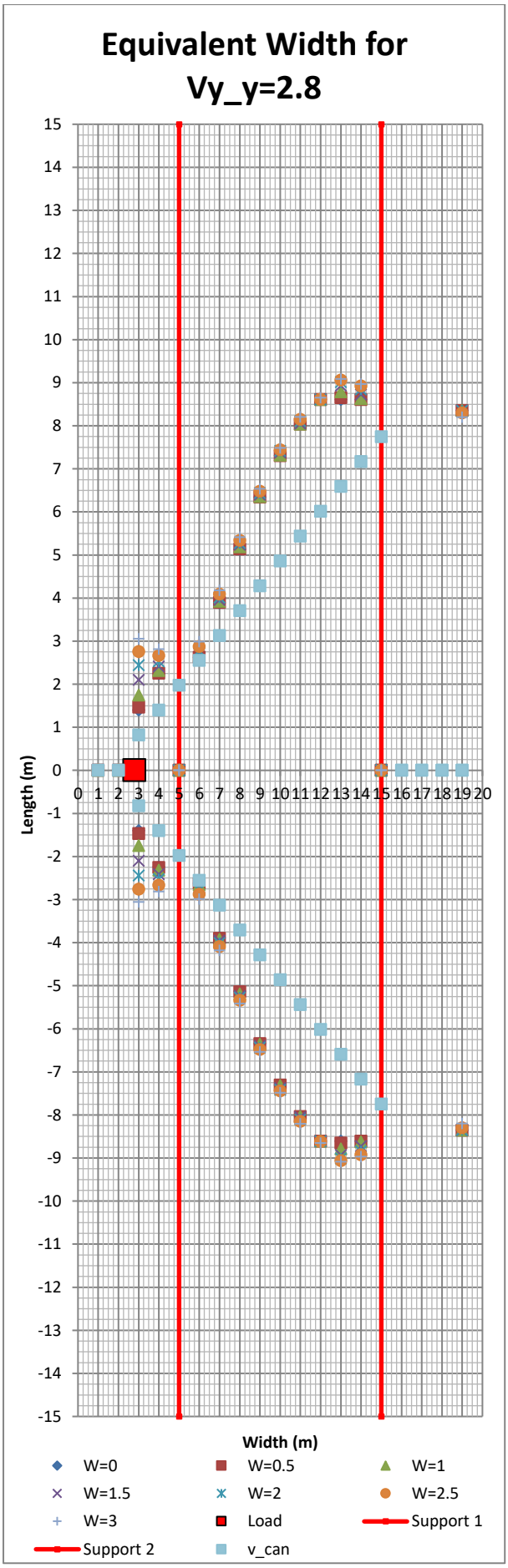

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

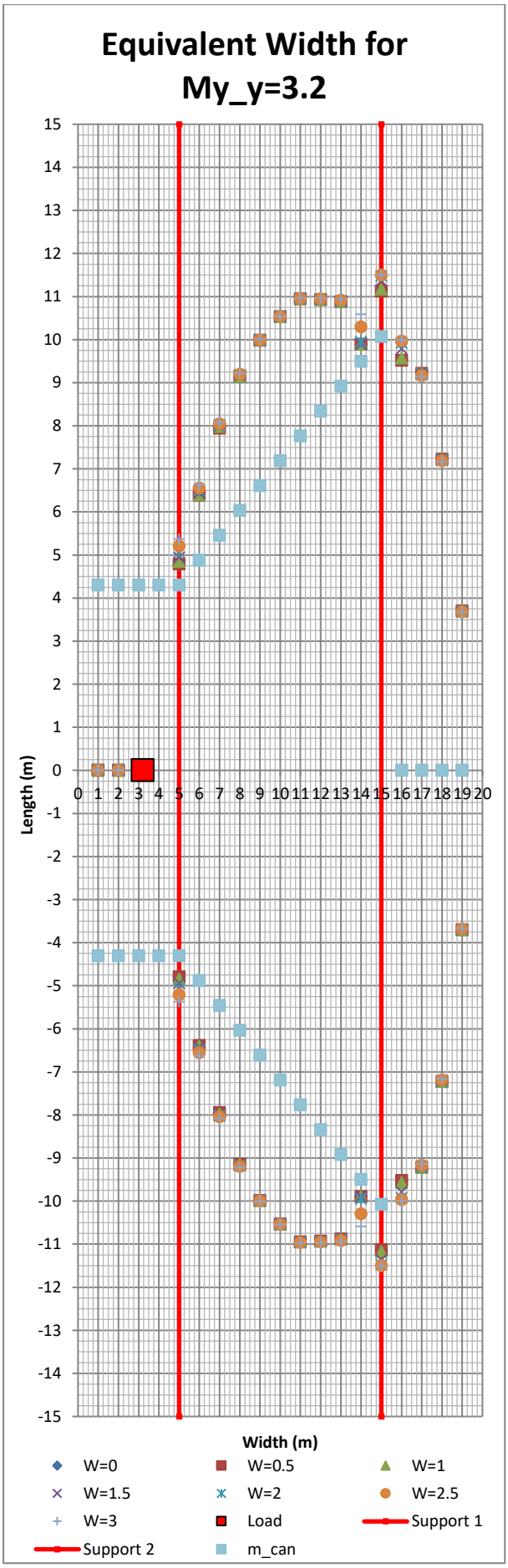

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

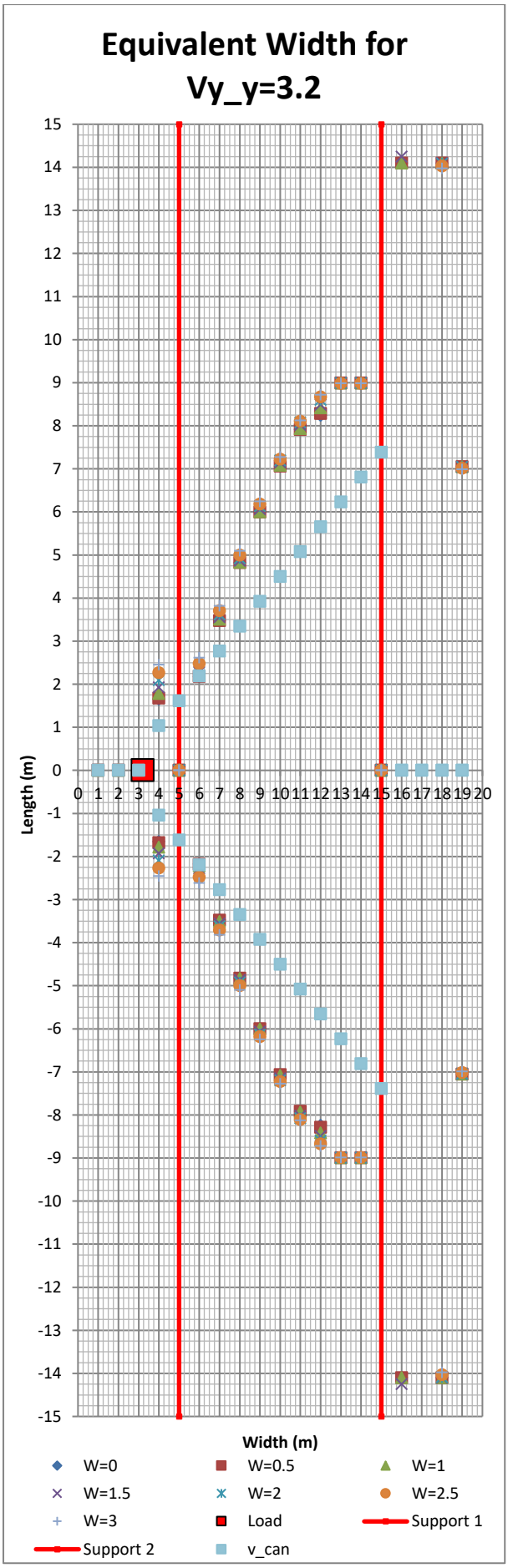

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

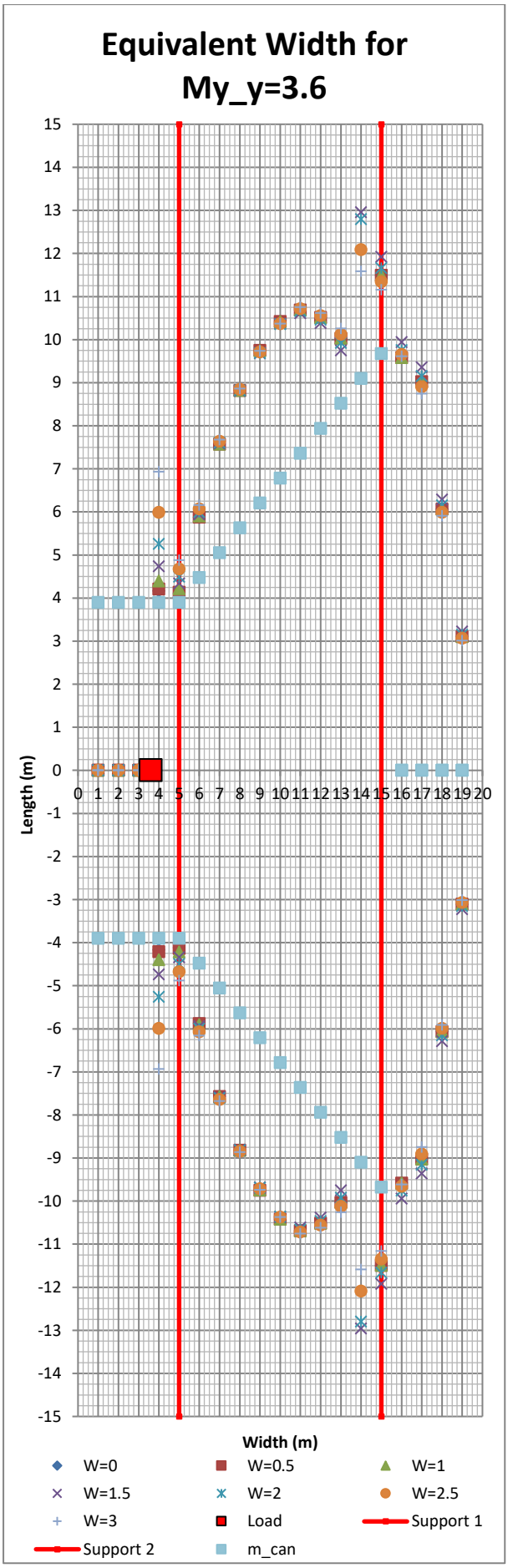

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

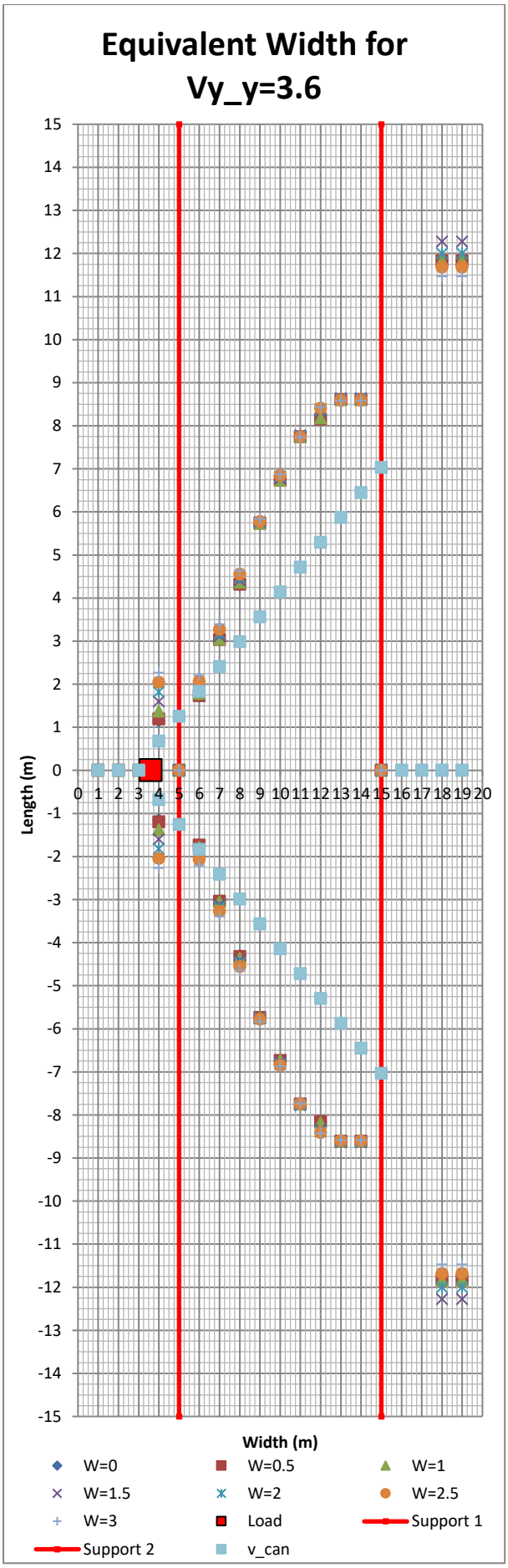

Equivalent Width for My_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

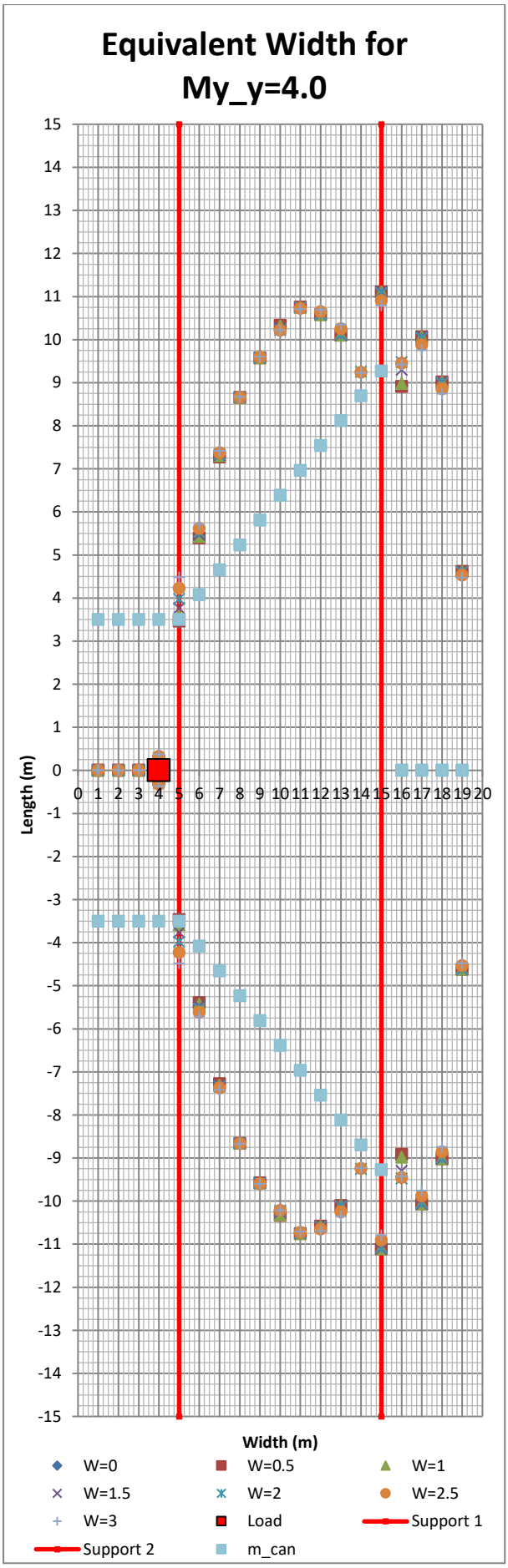

Equivalent Width for Vy_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

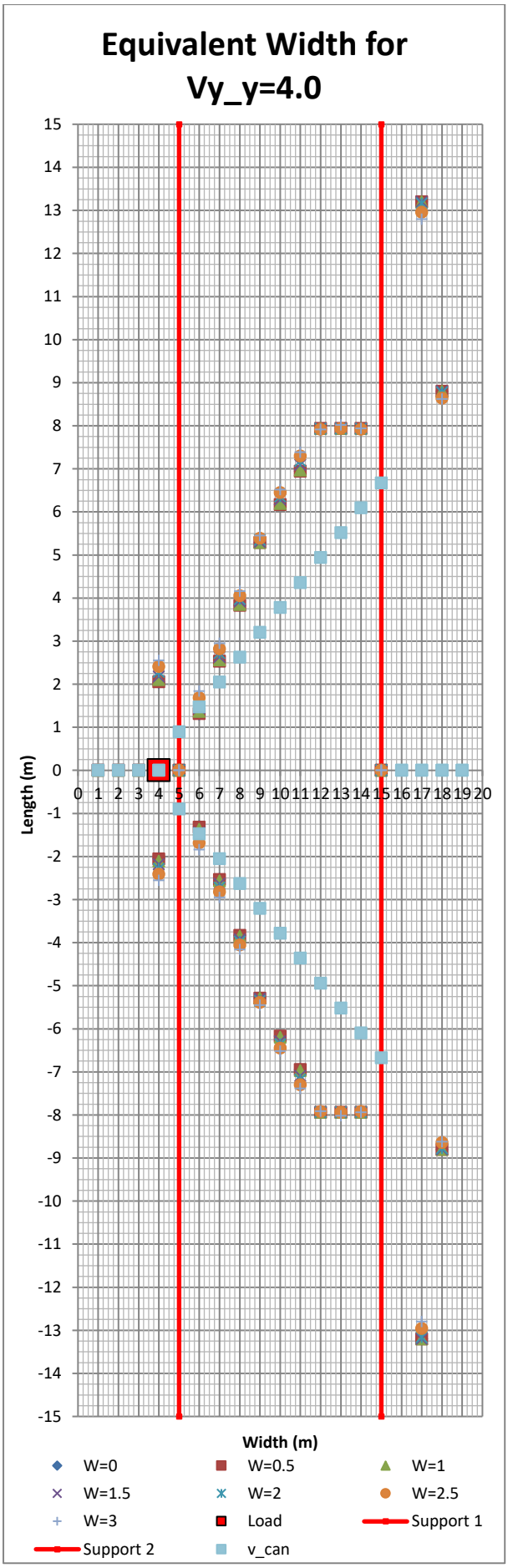

Equivalent Width for My_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

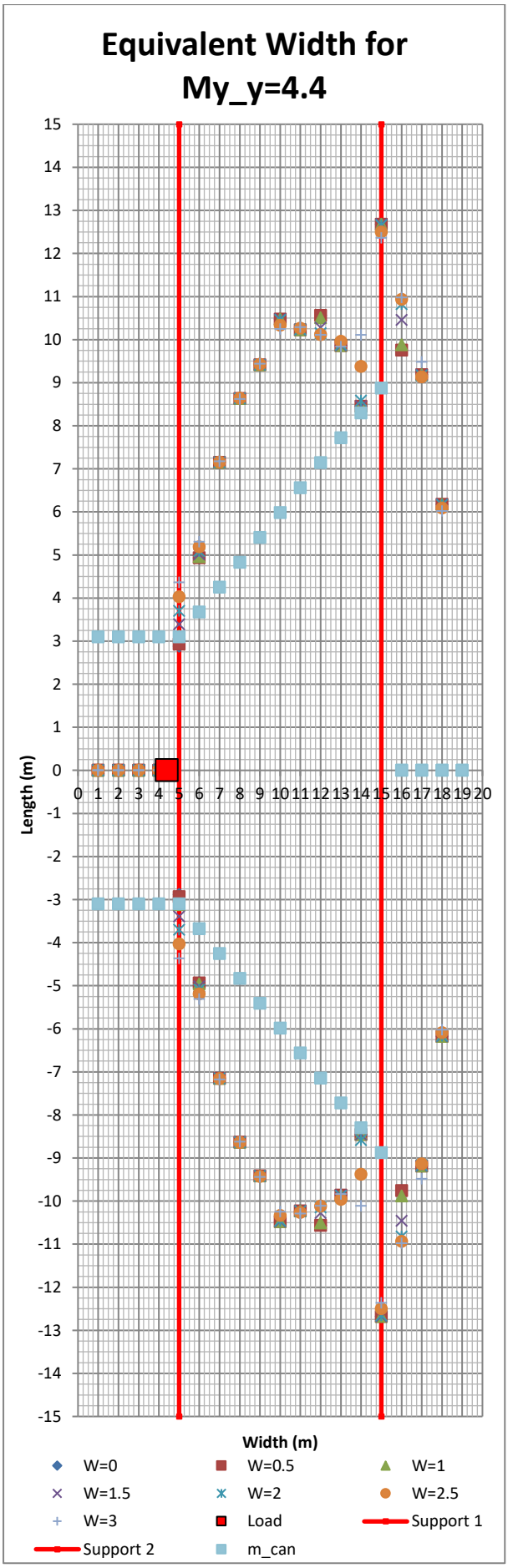

Equivalent Width for Vy_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

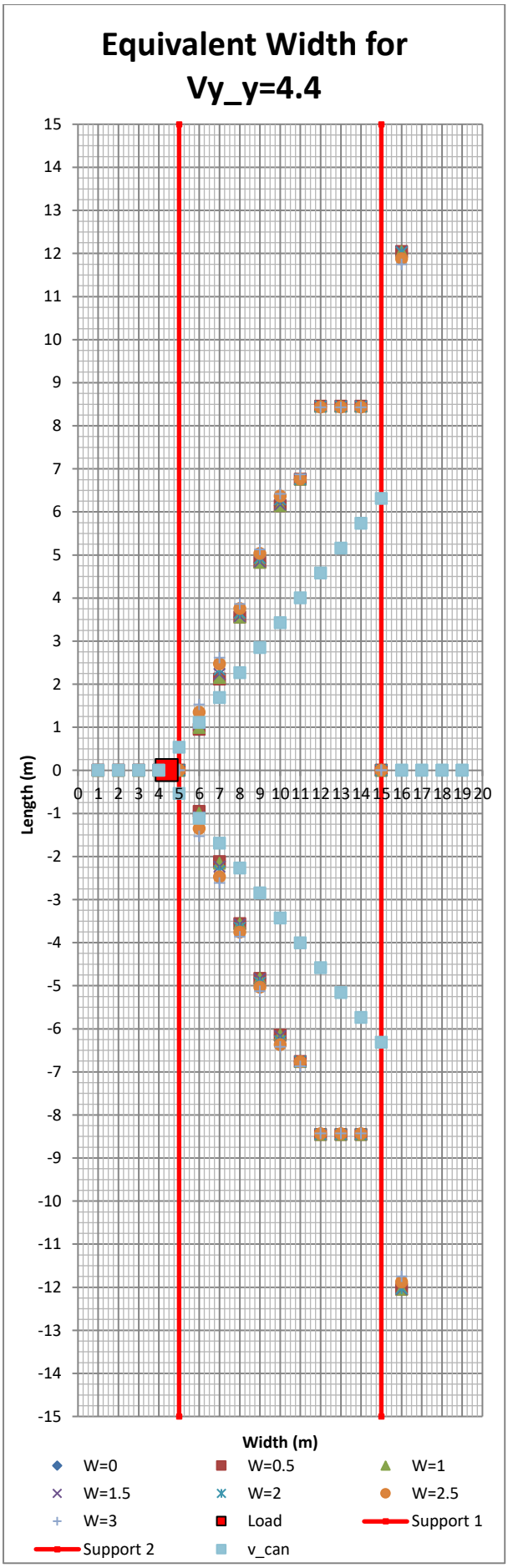

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

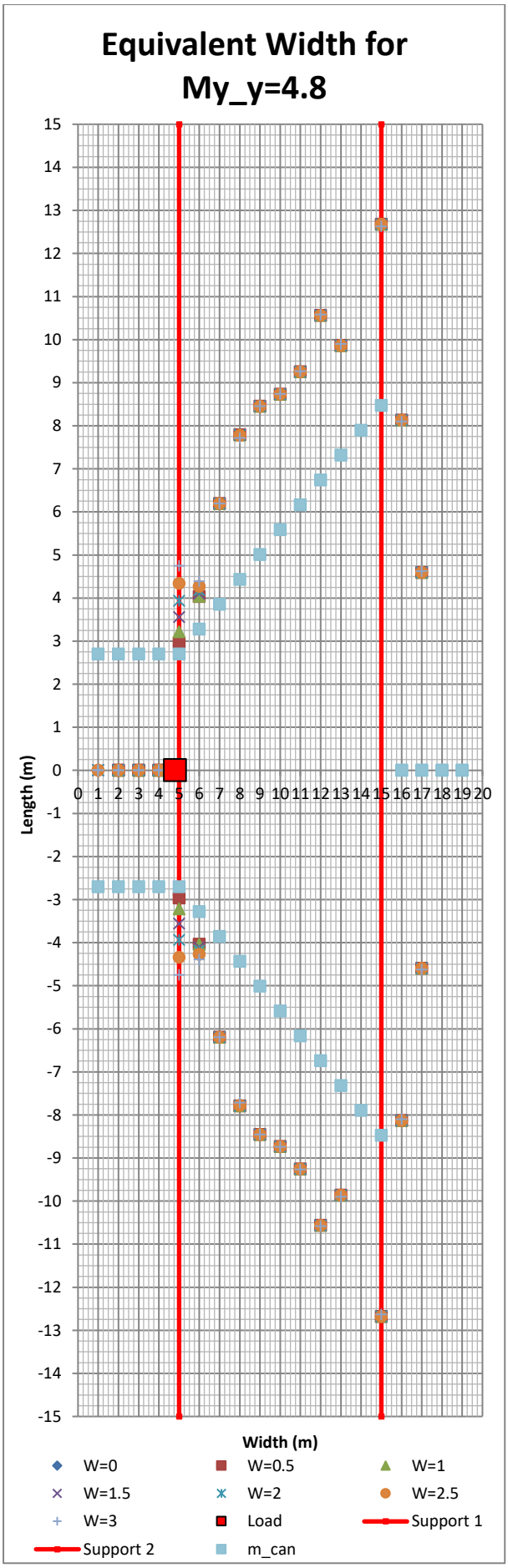

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

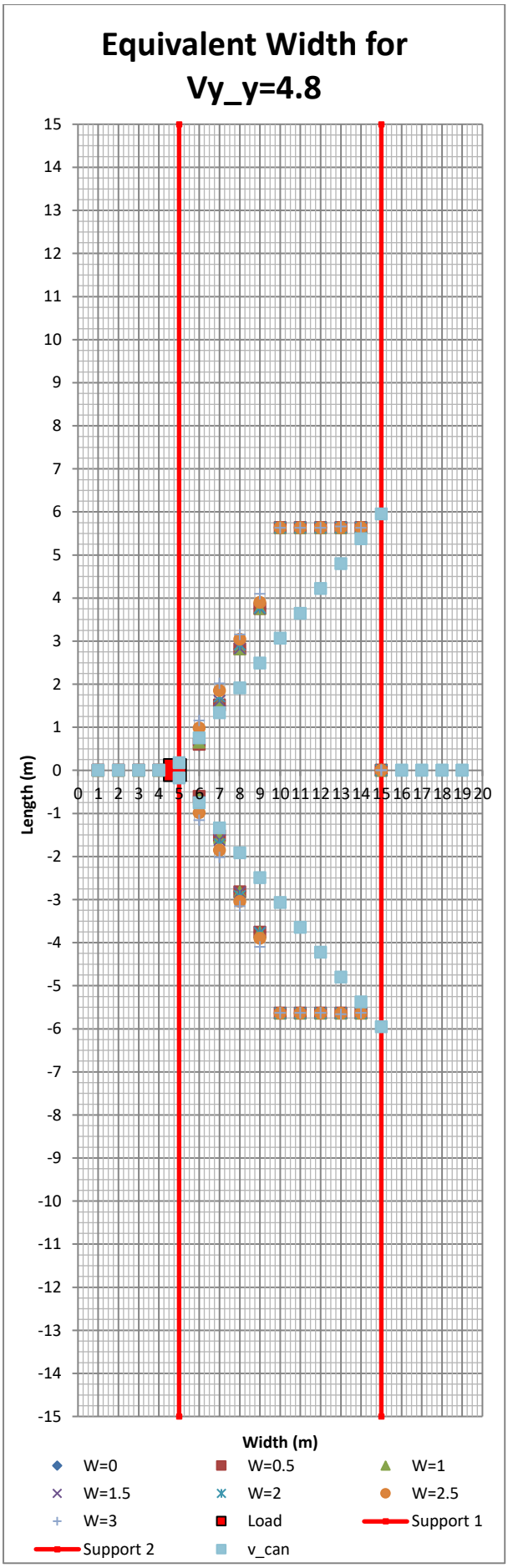

Equivalent Width for My_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

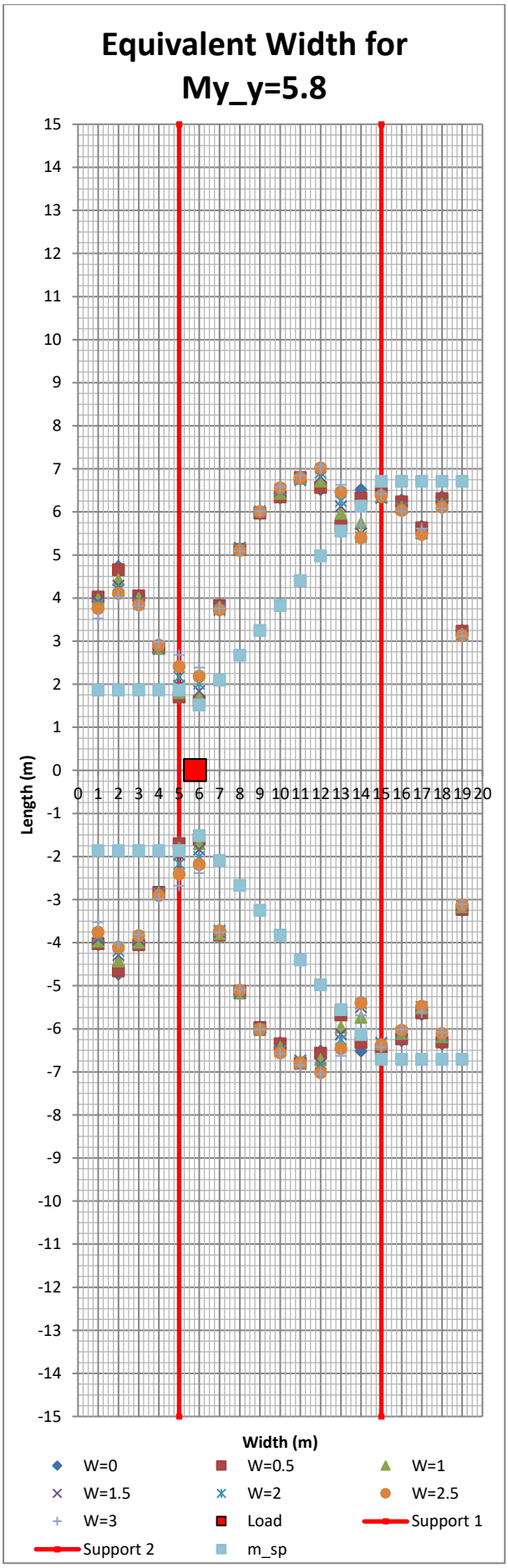

Equivalent Width for Vy_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

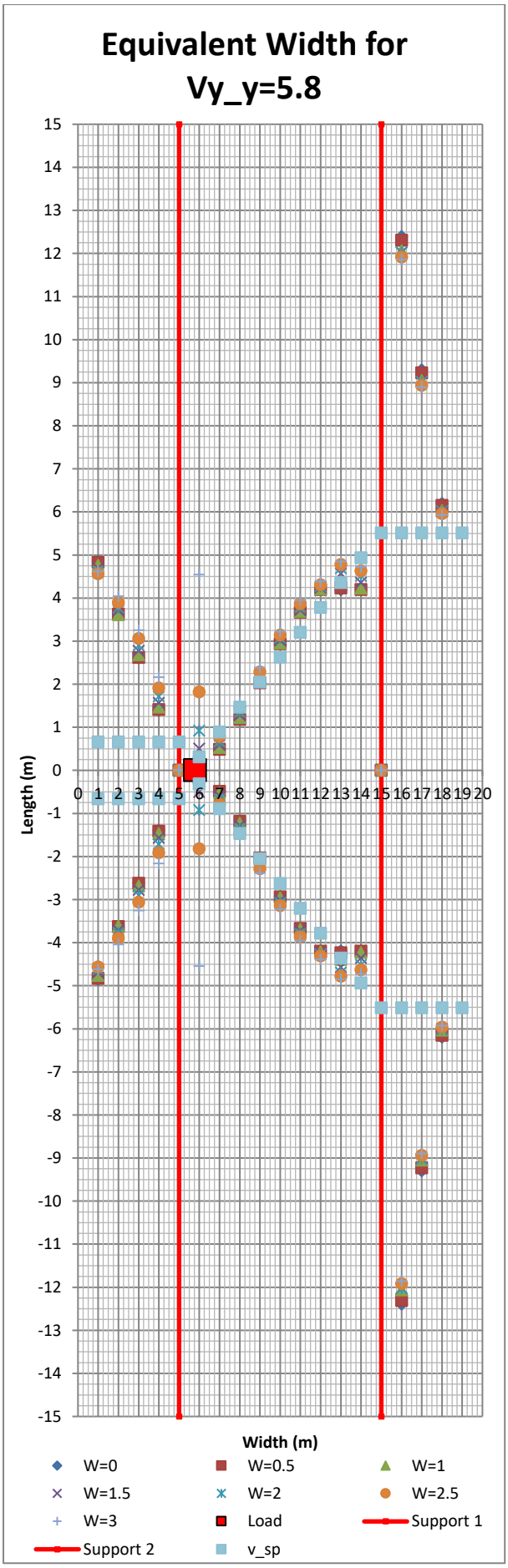

Equivalent Width for My_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

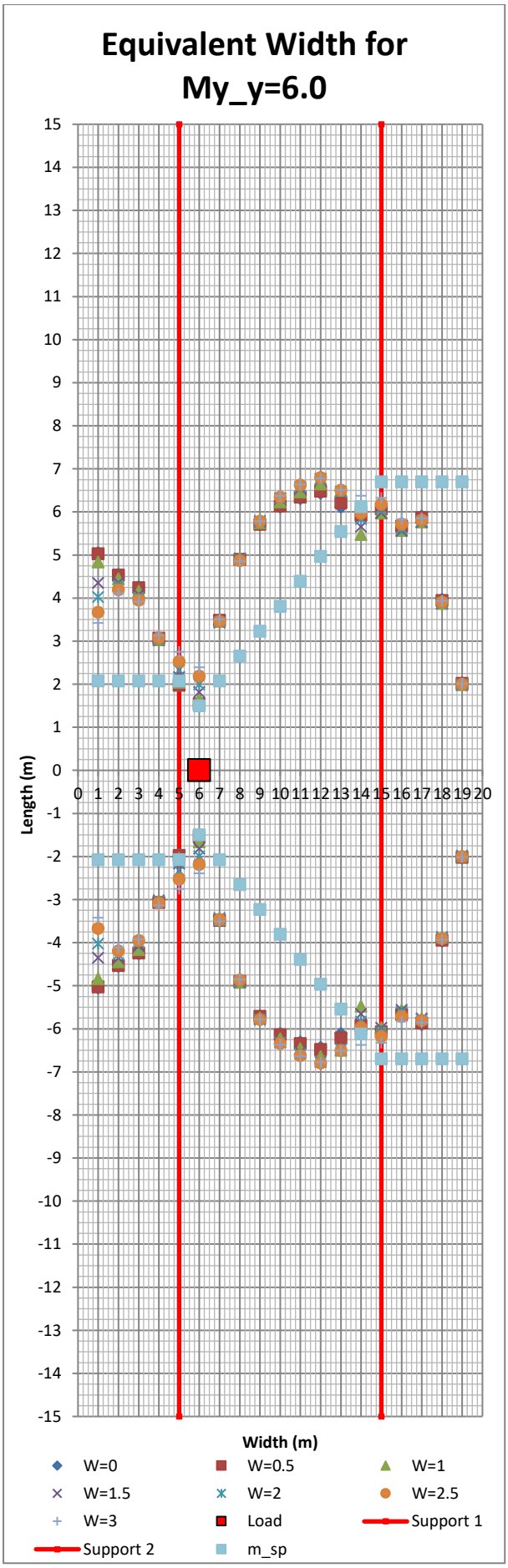

Equivalent Width for Vy_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

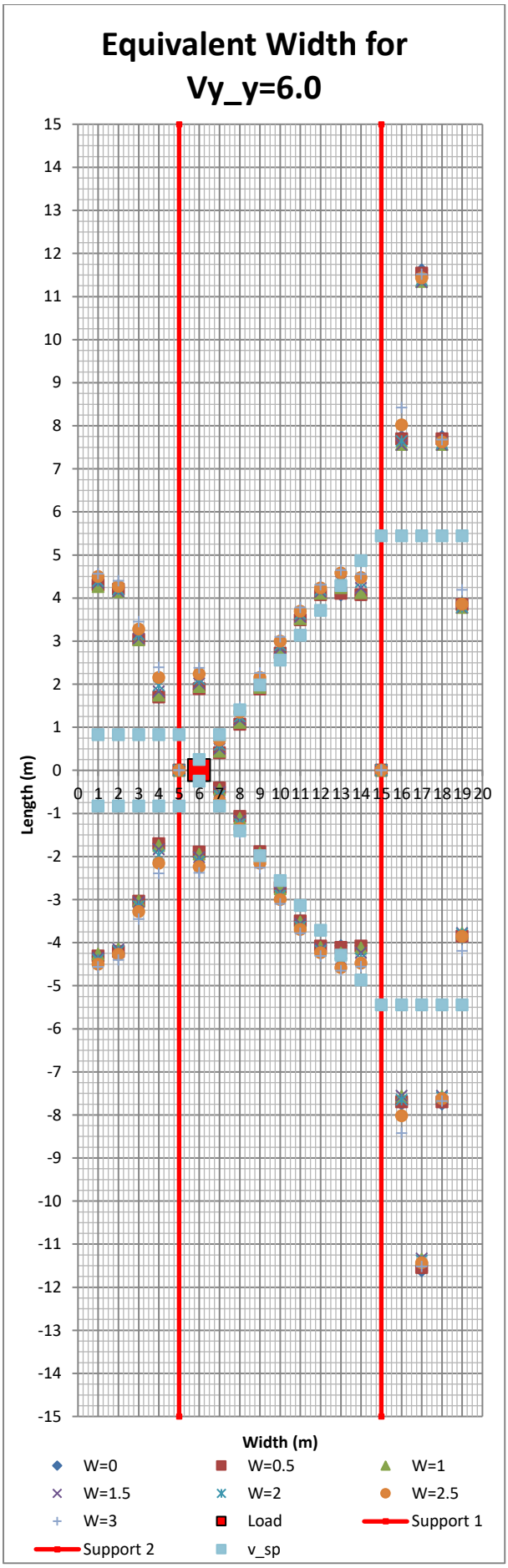

Equivalent Width for My_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

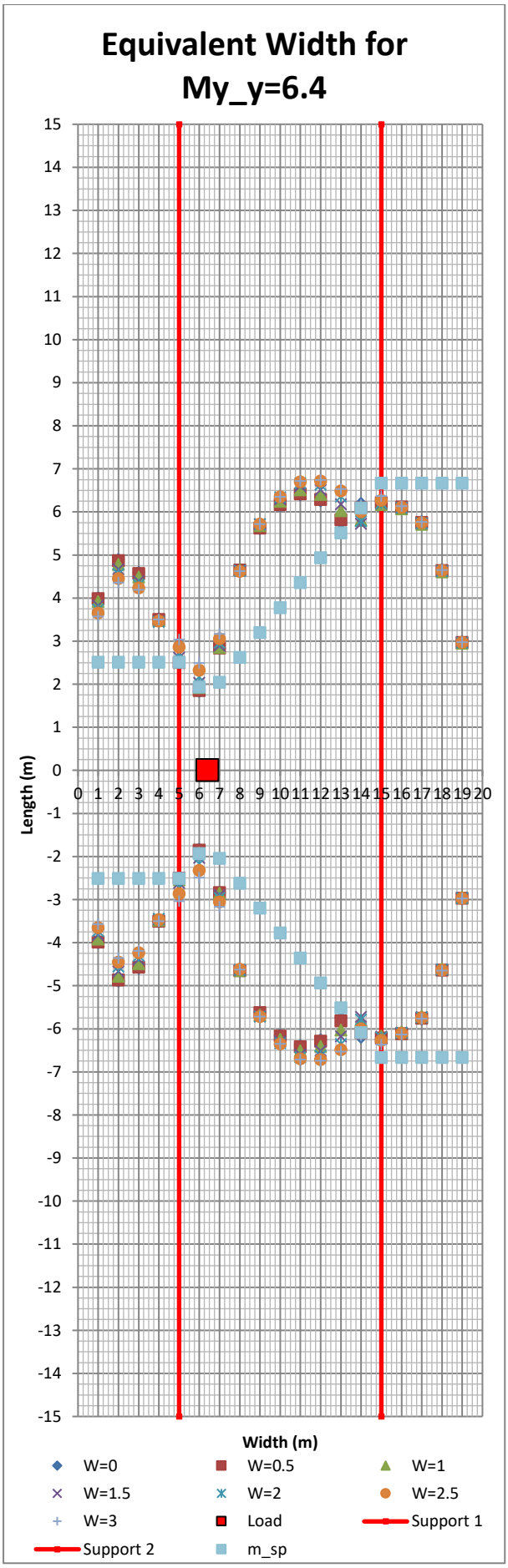

Equivalent Width for Vy_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

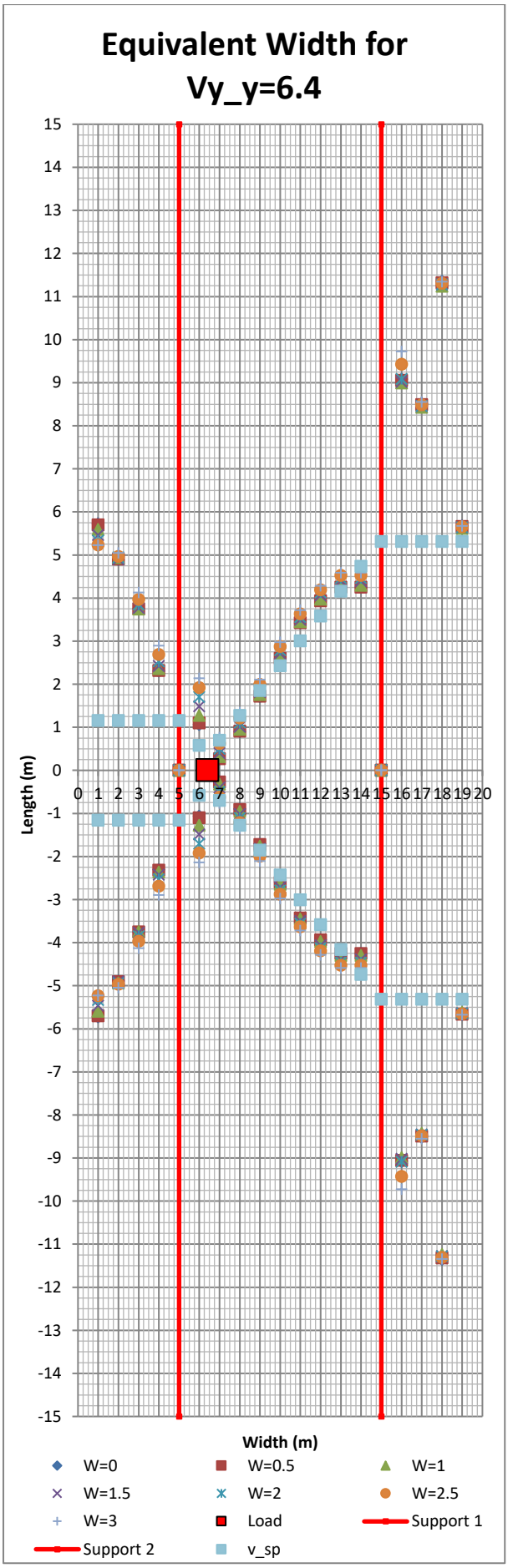

Equivalent Width for My_y=5.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

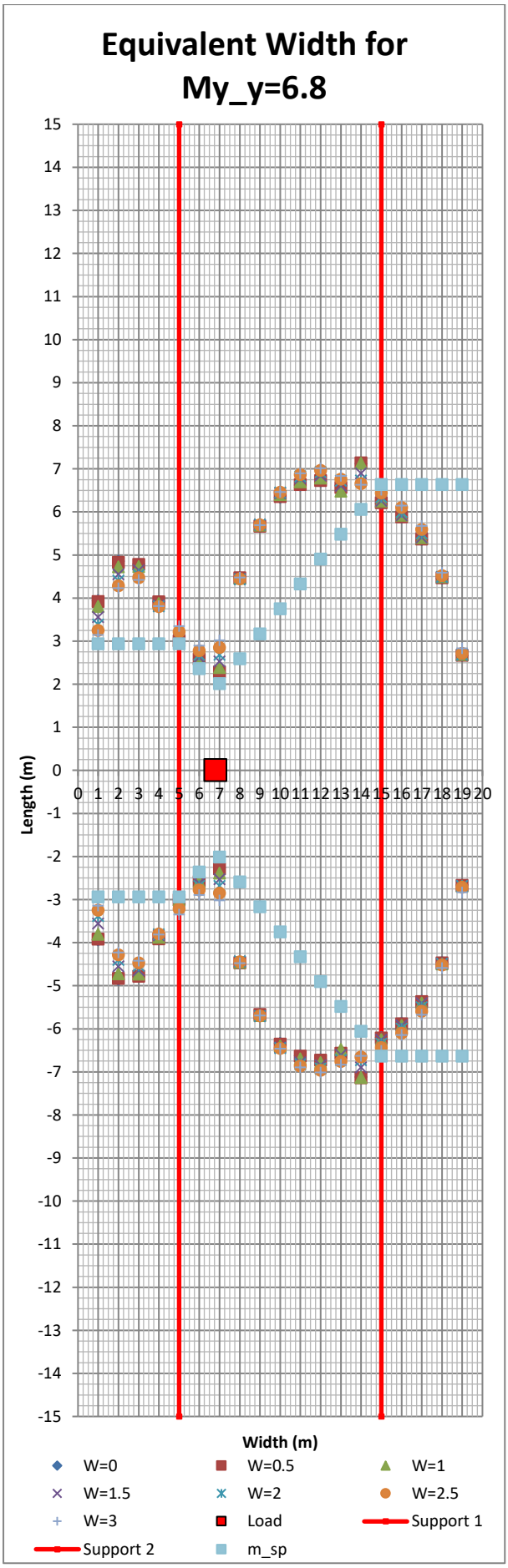

Equivalent Width for Vy_y=5.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

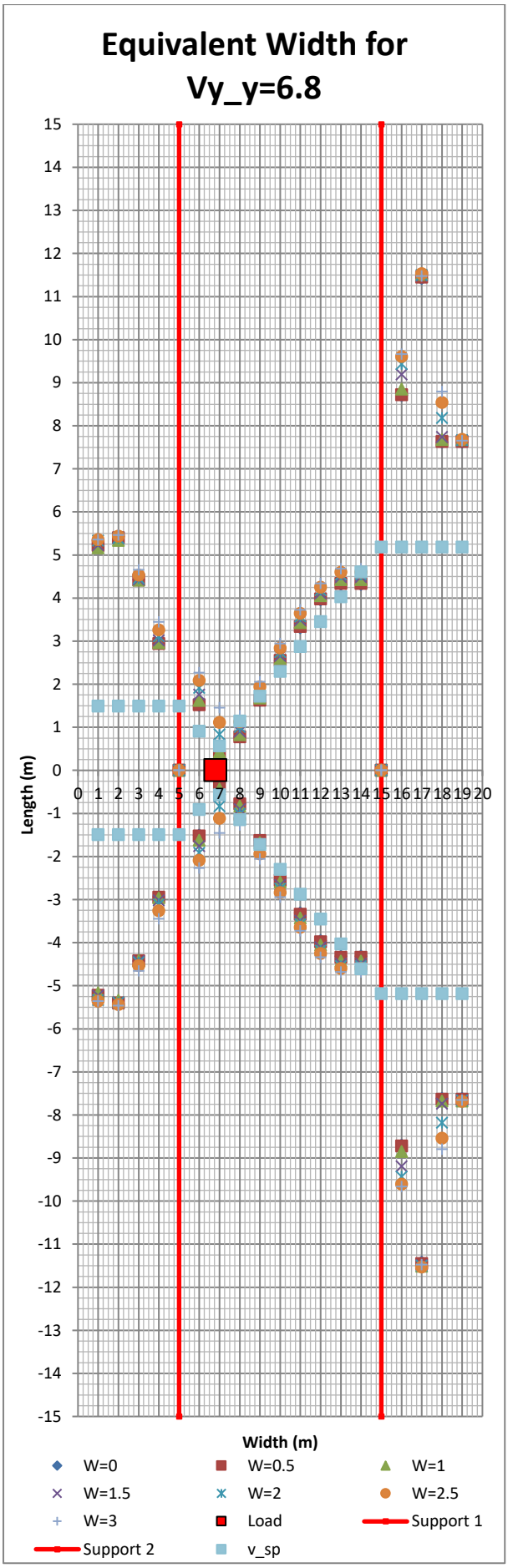

Equivalent Width for My_y=6.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

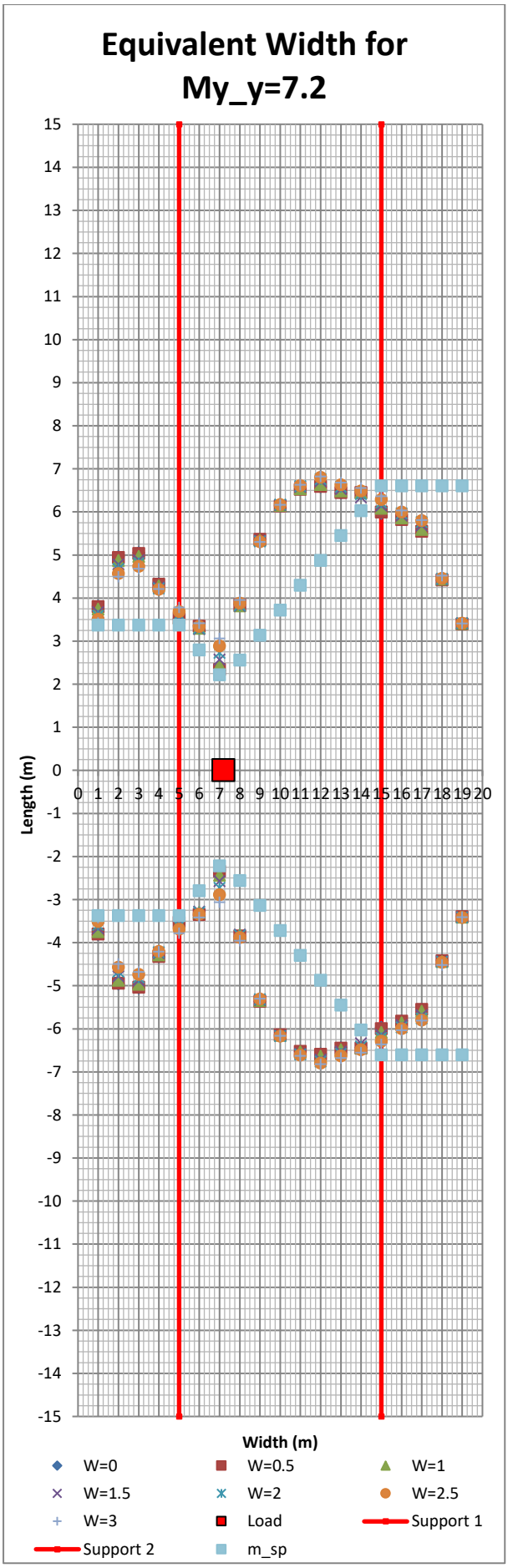

Equivalent Width for Vy_y=6.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

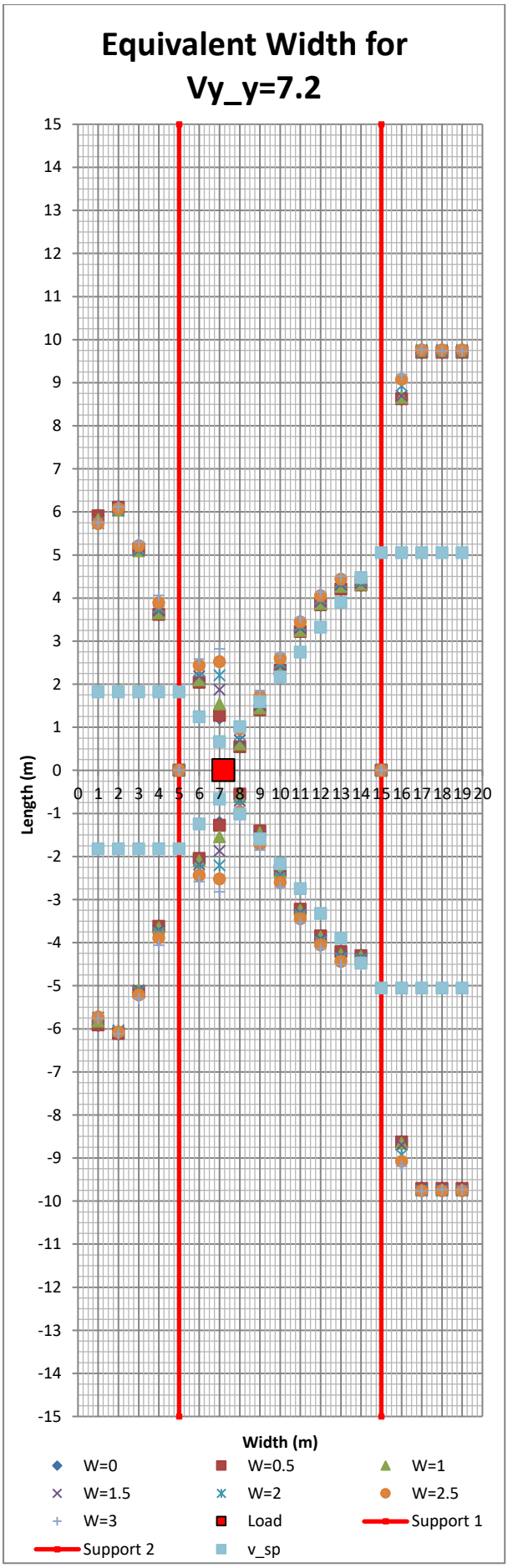

Equivalent Width for My_y=6.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

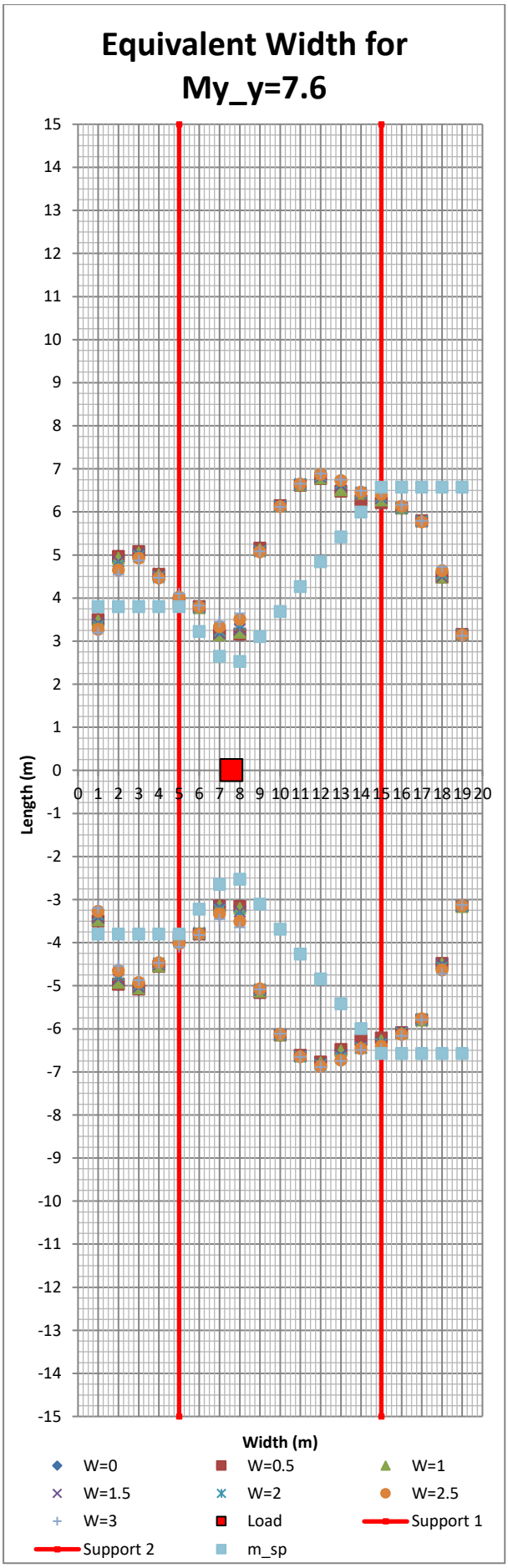

Equivalent Width for Vy_y=6.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

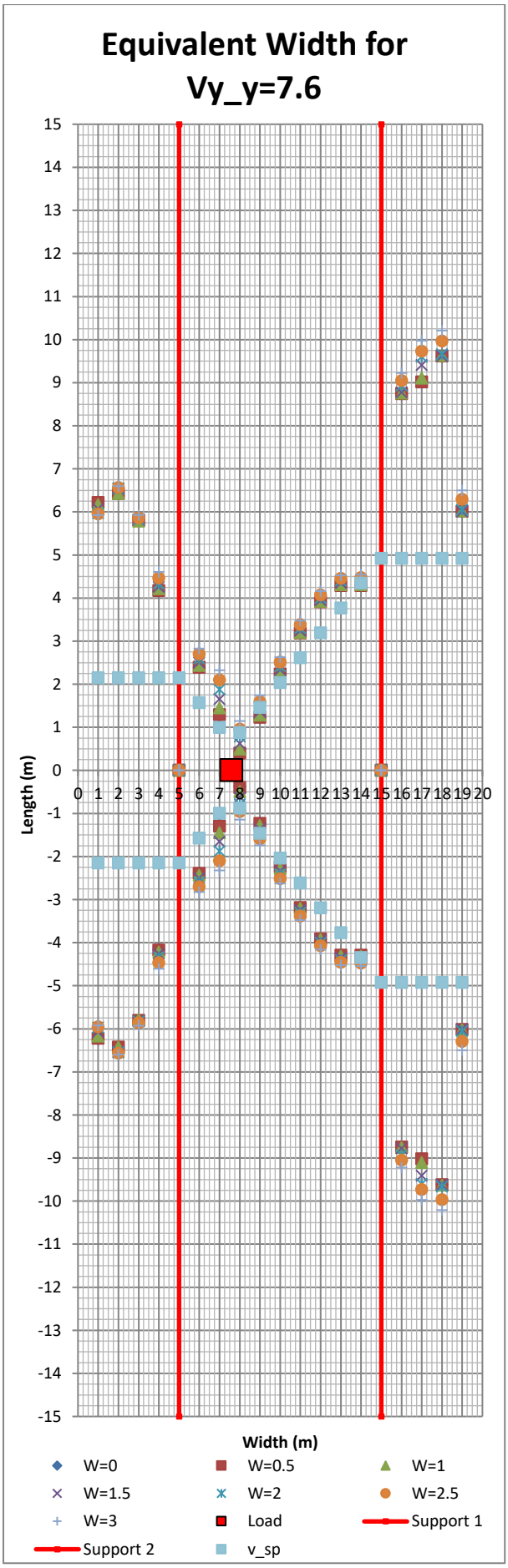

Equivalent Width for My_y=6.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

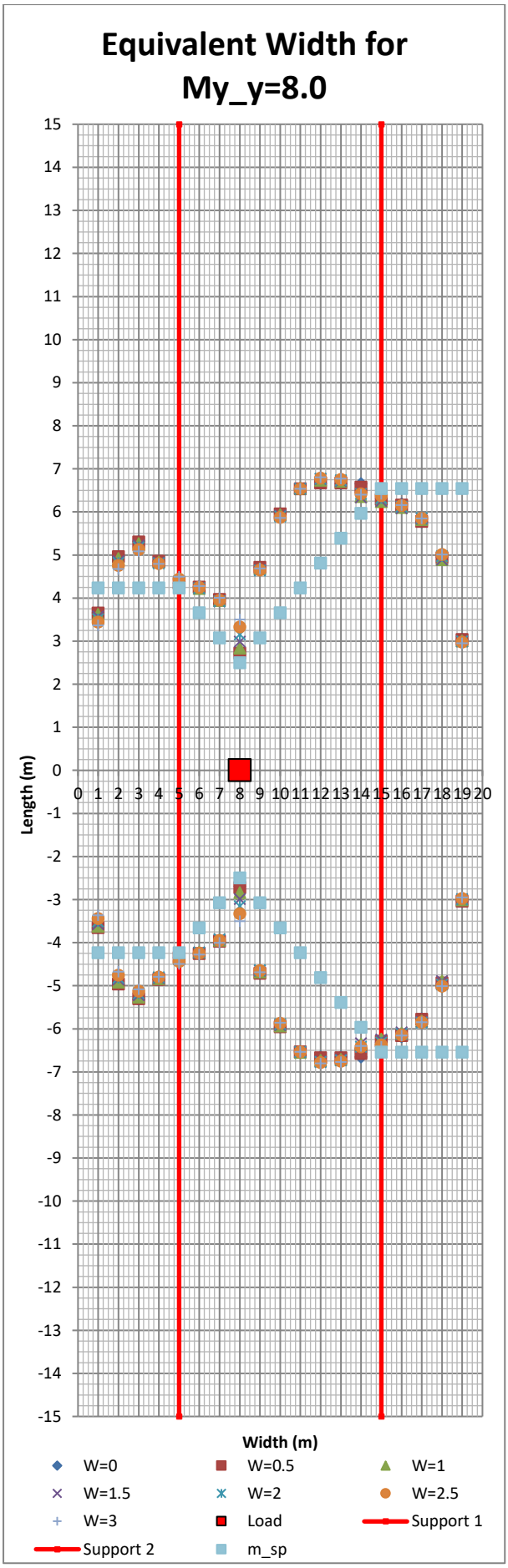

Equivalent Width for Vy_y=6.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

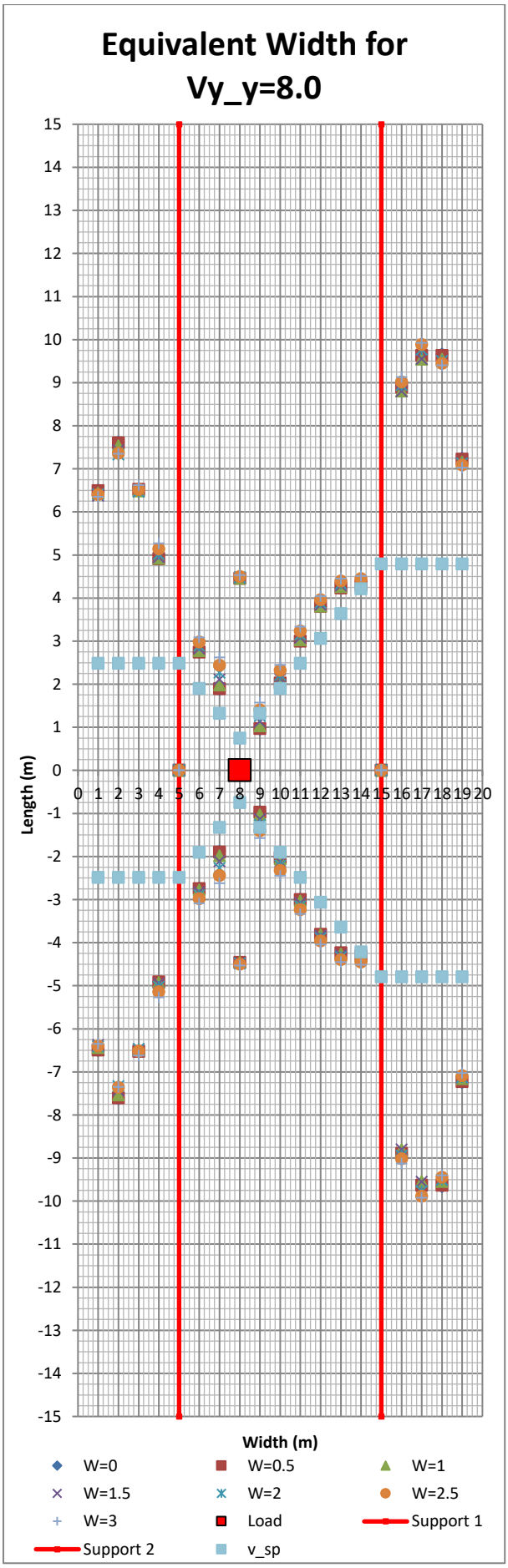

Equivalent Width for My_y=7.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

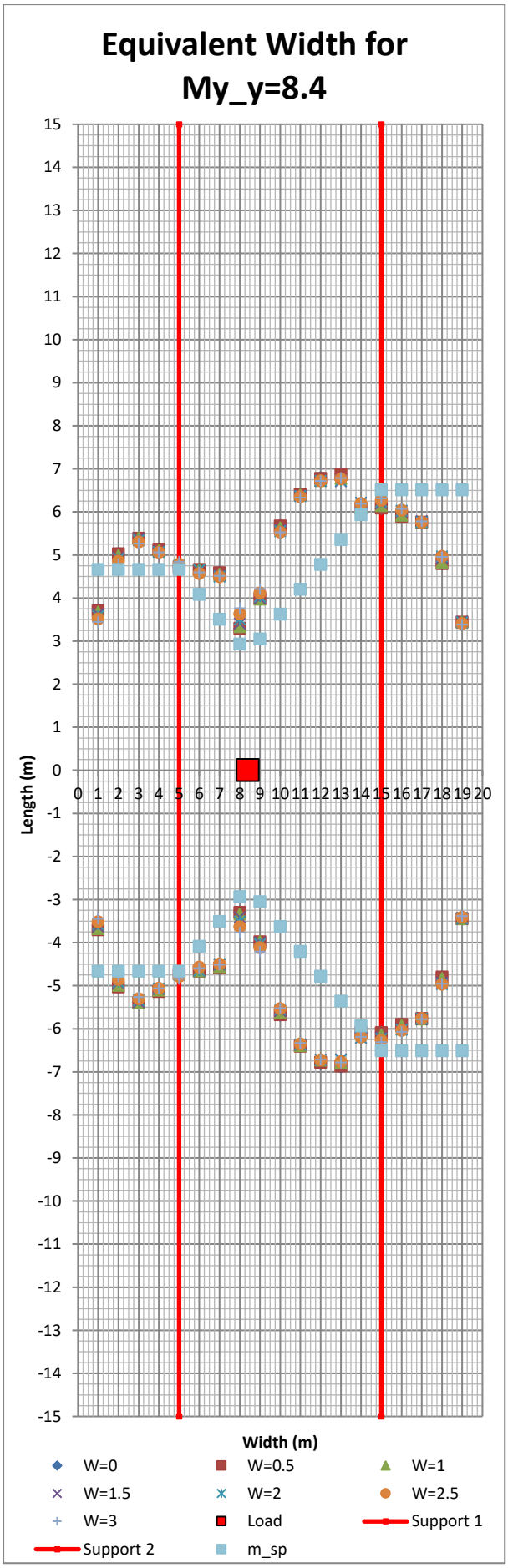

Equivalent Width for Vy_y=7.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

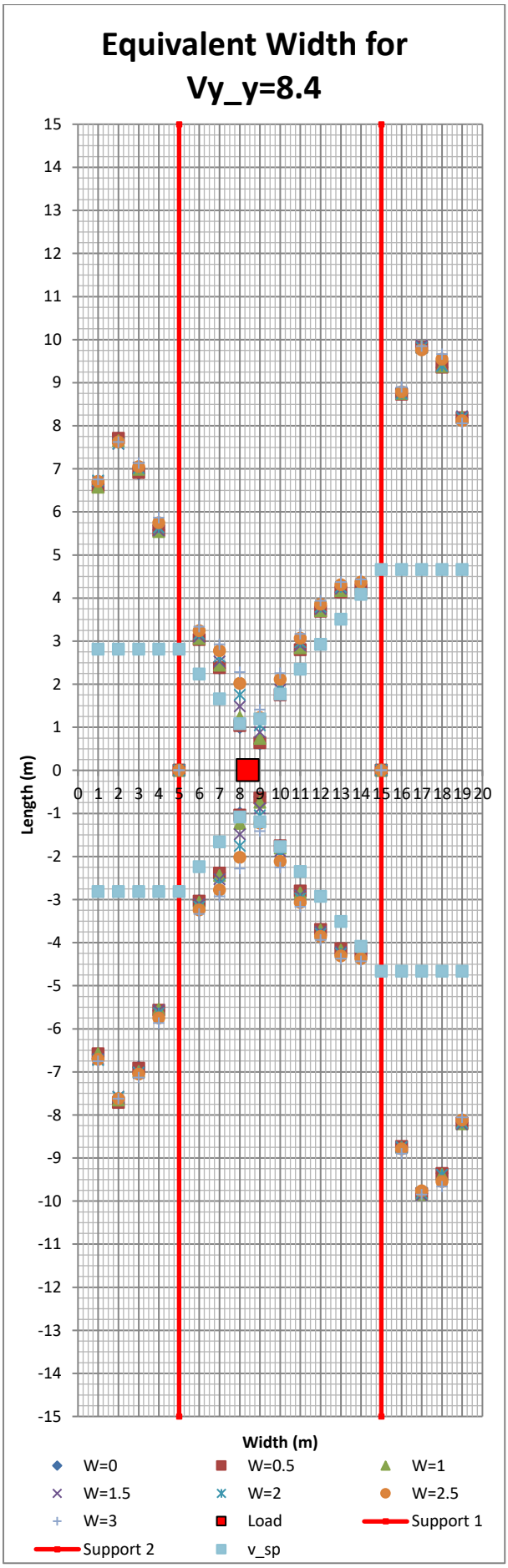

Equivalent Width for My_y=7.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

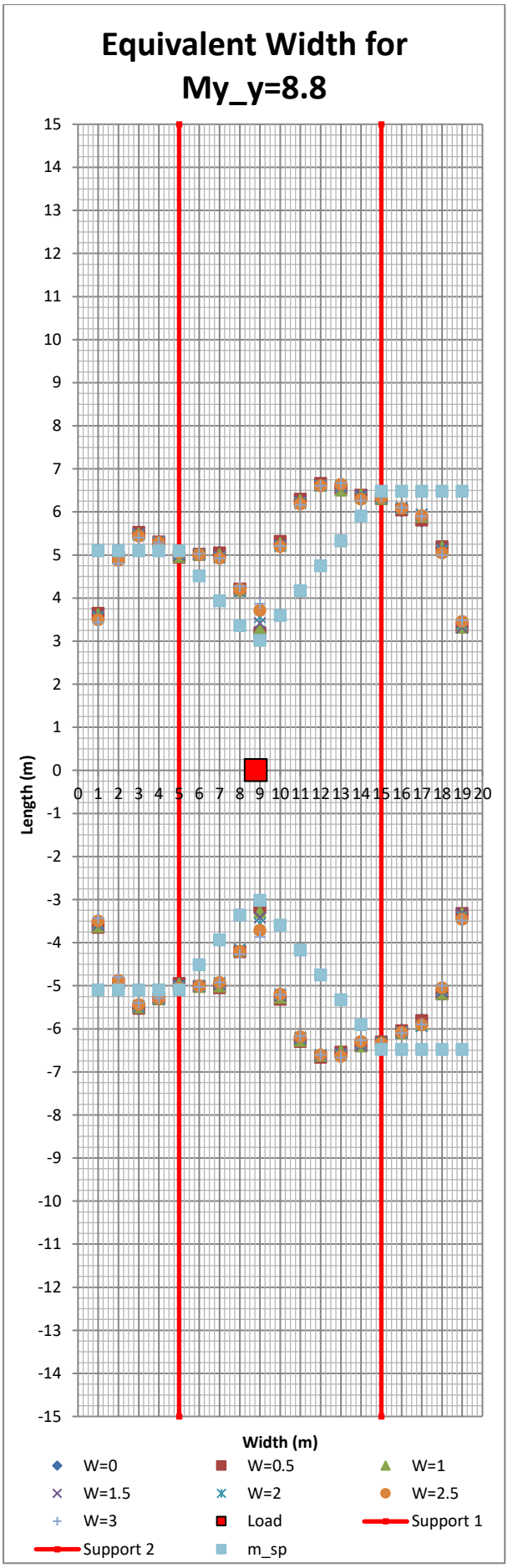

Equivalent Width for Vy_y=7.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

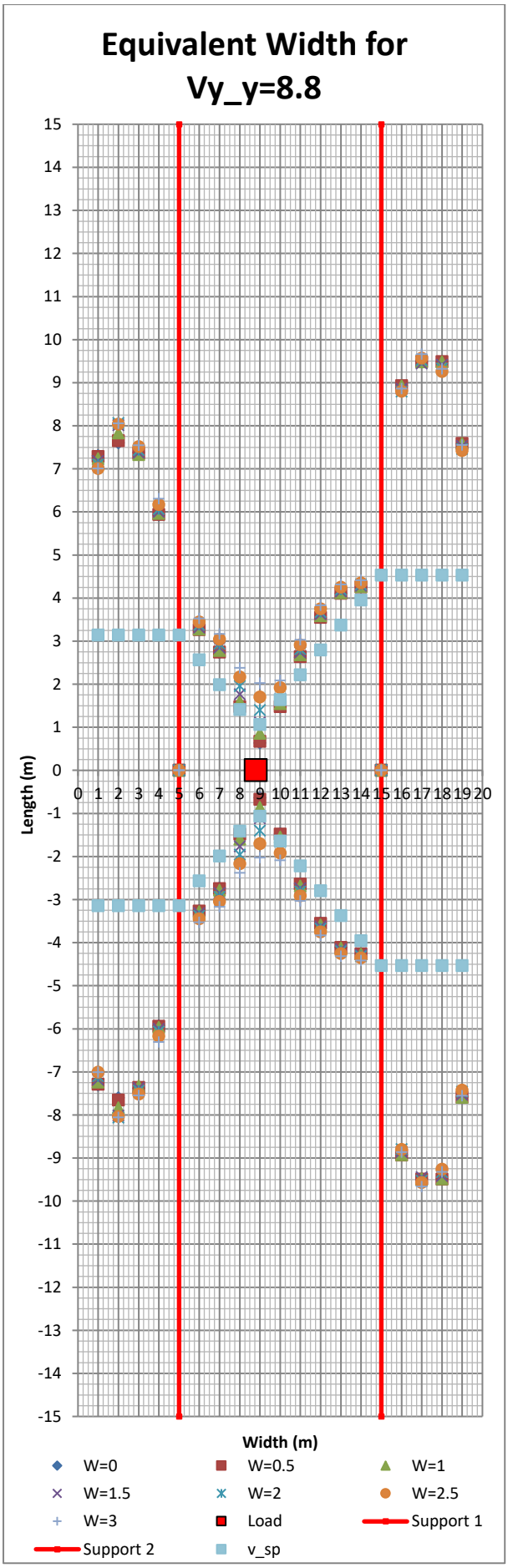

Equivalent Width for My_y=8.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

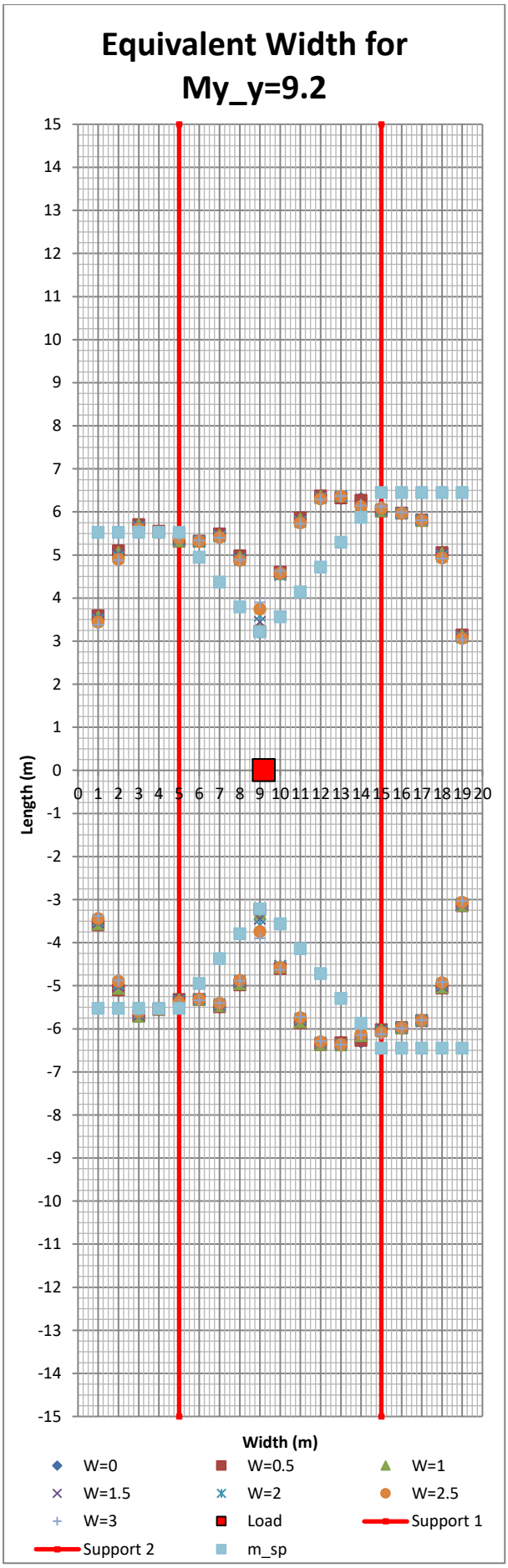

Equivalent Width for Vy_y=8.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

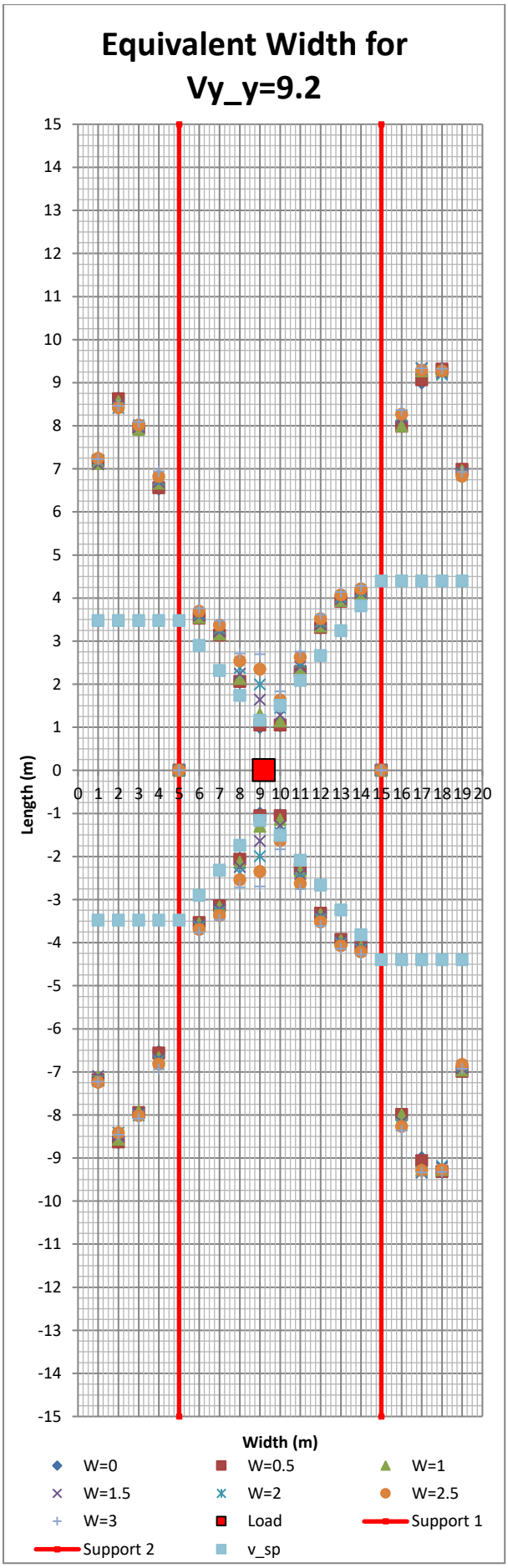

Equivalent Width for My_y=8.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

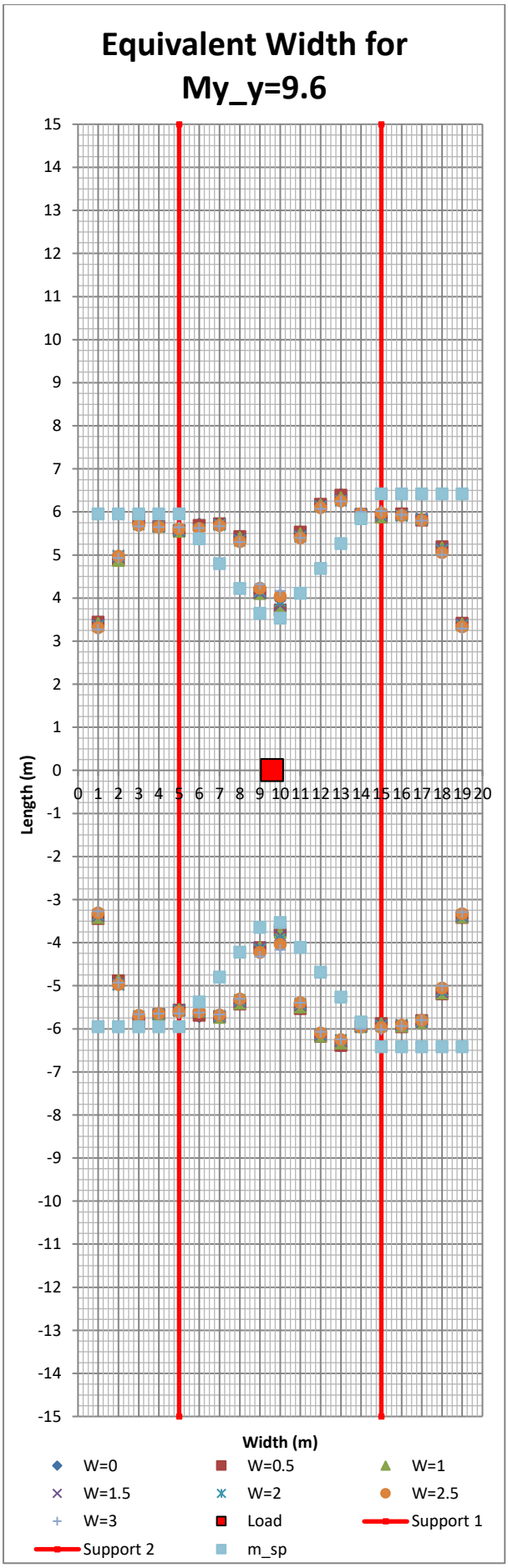

Equivalent Width for Vy_y=8.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

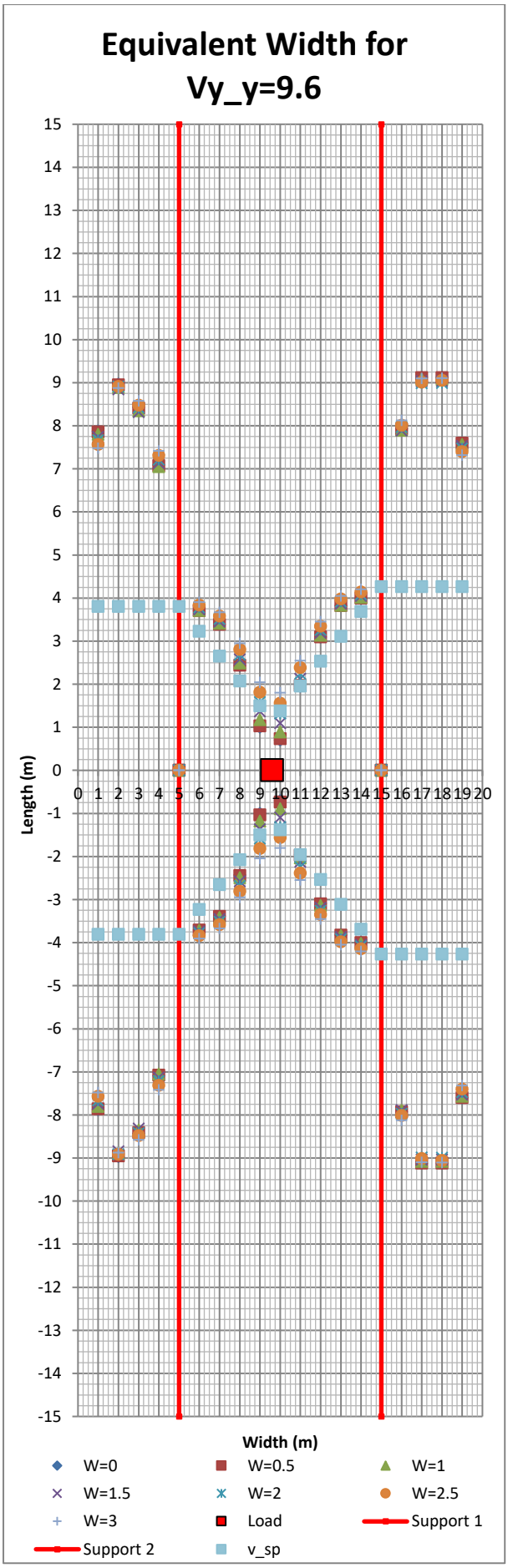

Equivalent Width for My_y=8.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

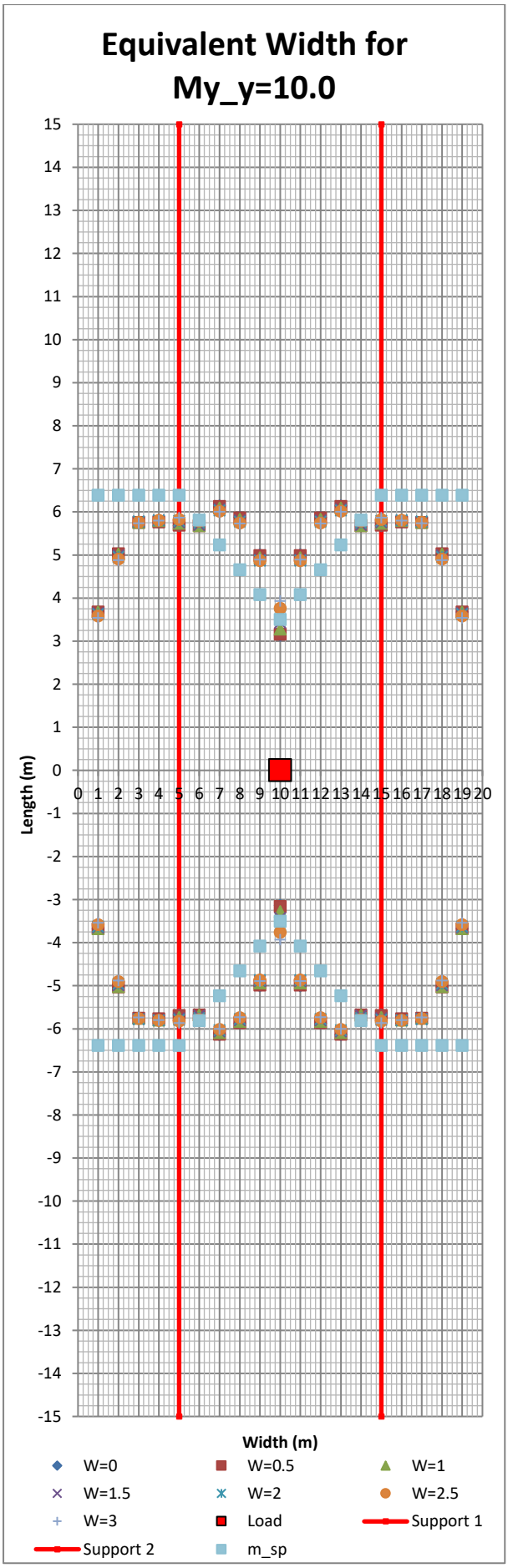

Equivalent Width for Vy_y=8.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

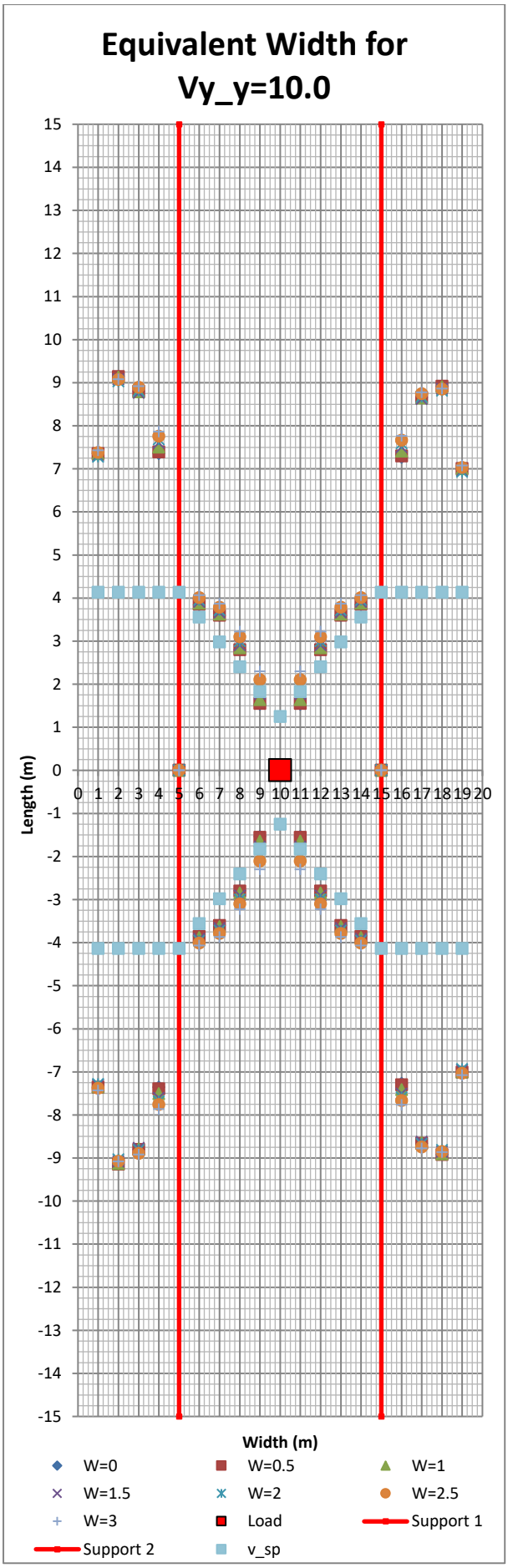

Equivalent Width for My_y=9.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=9.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

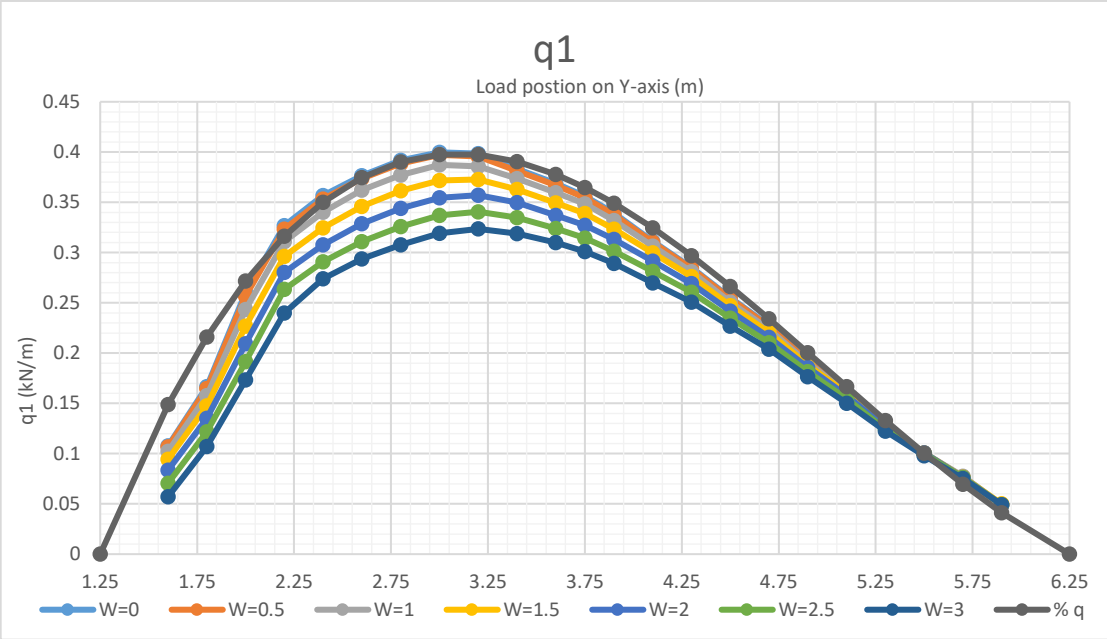

Equivalent Width for My_y=9.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

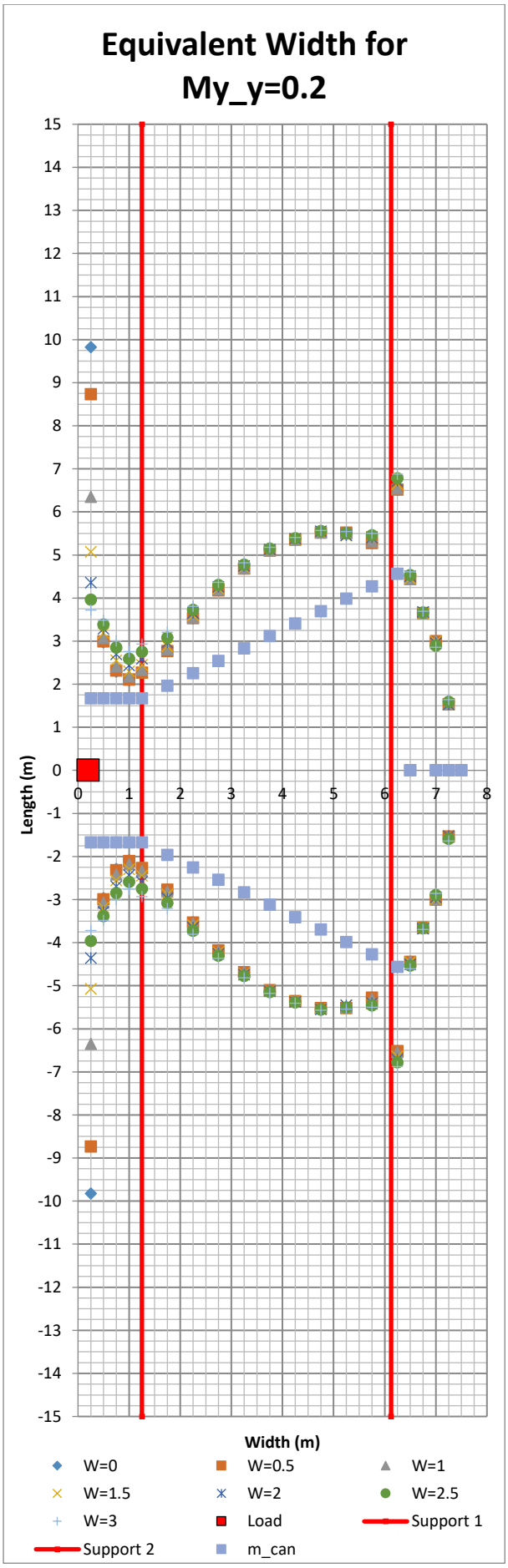

Equivalent Width for Vy_y=9.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

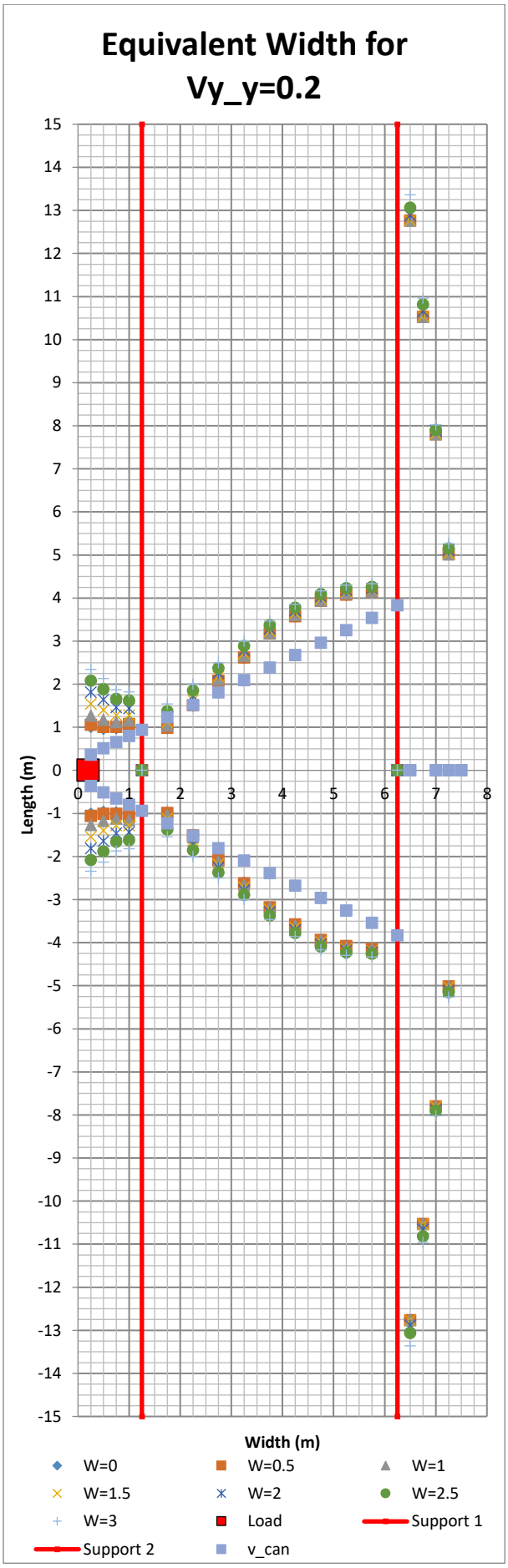

Equivalent Width for My_y=10.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

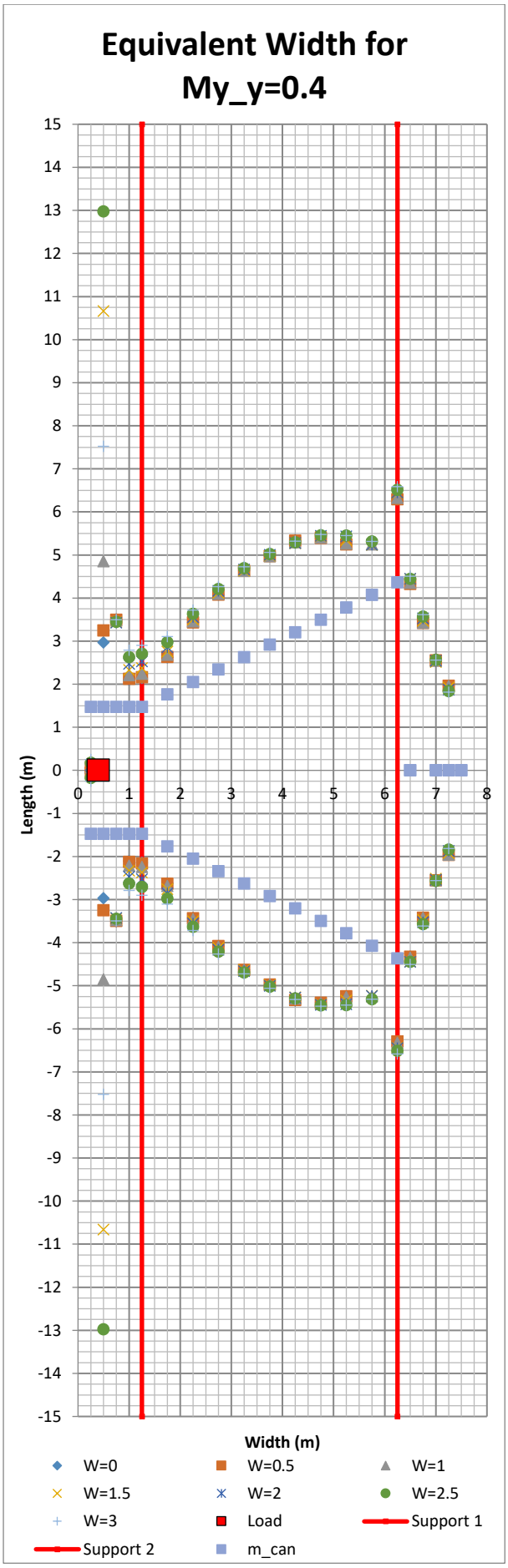

Equivalent Width for Vy_y=10.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

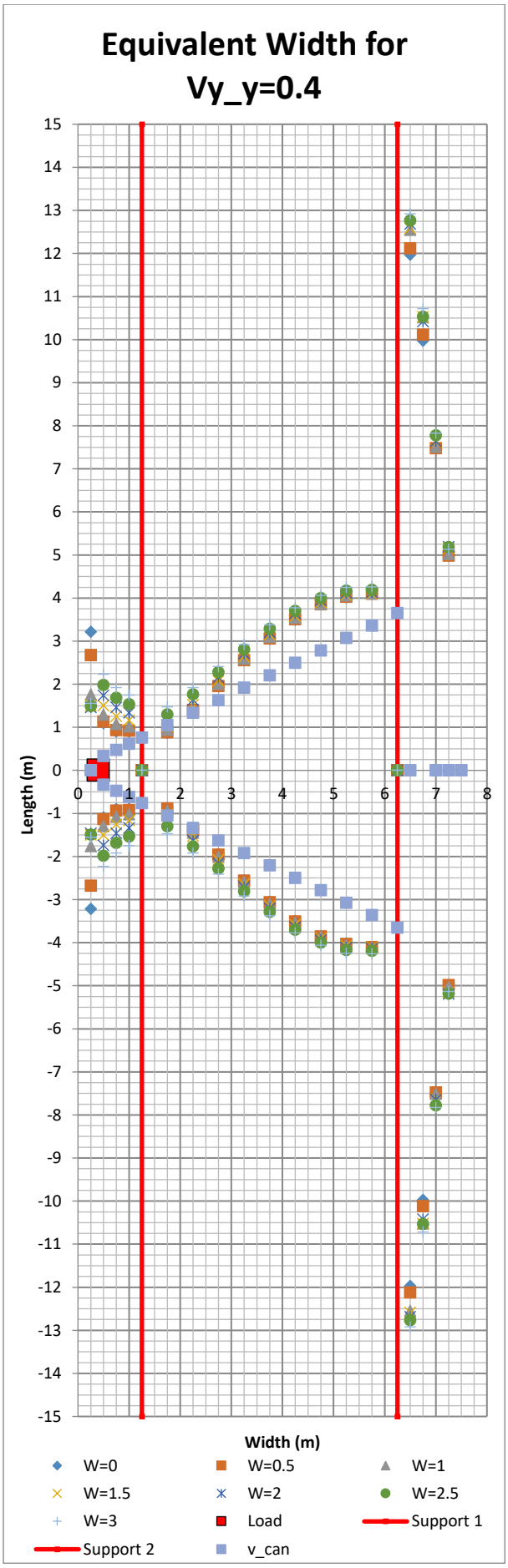
11.4 Fourth Variation $\alpha = 0.25,\ l_m = 5$

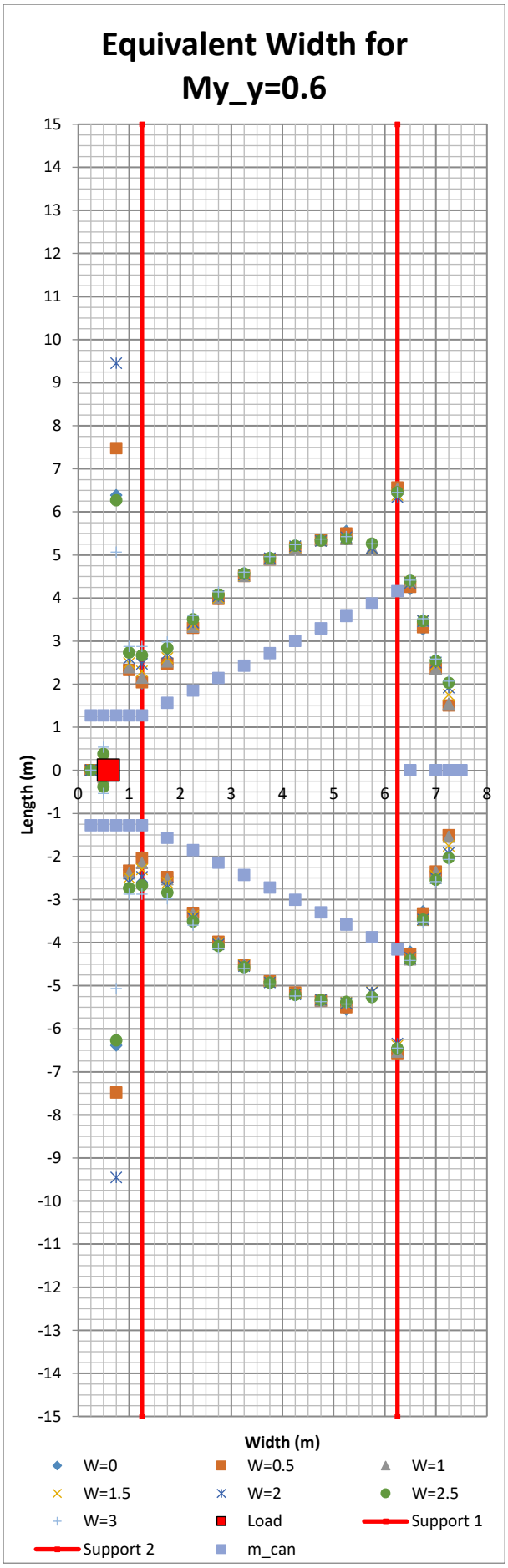


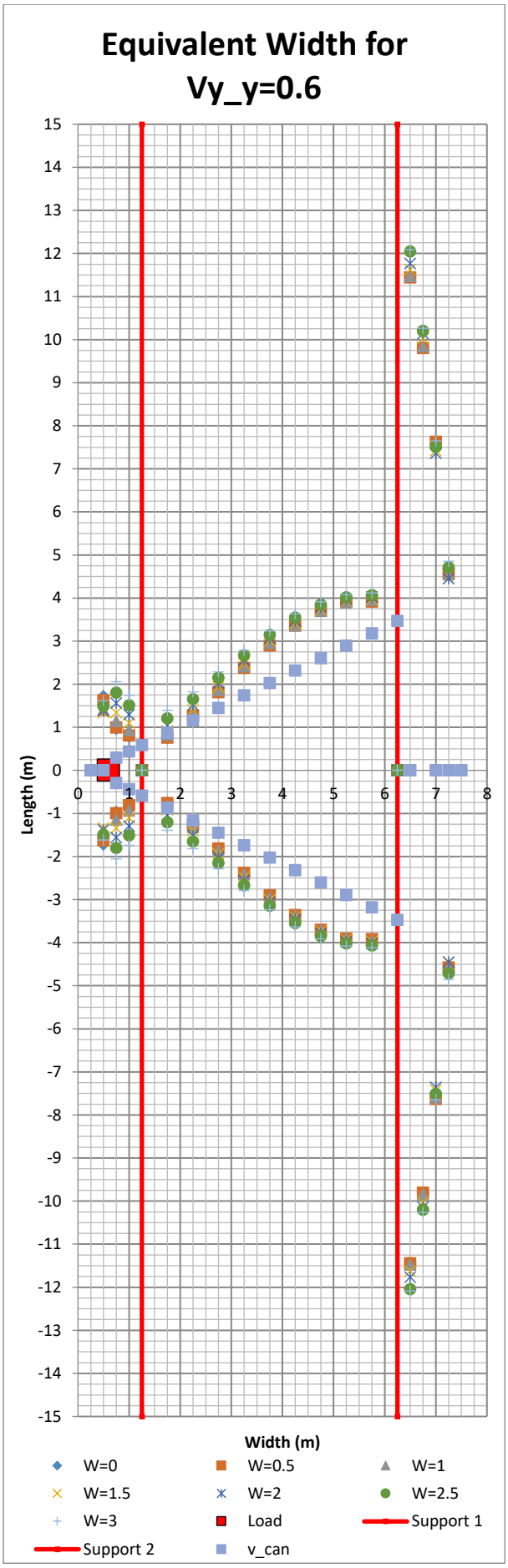

Equivalent Width for My_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

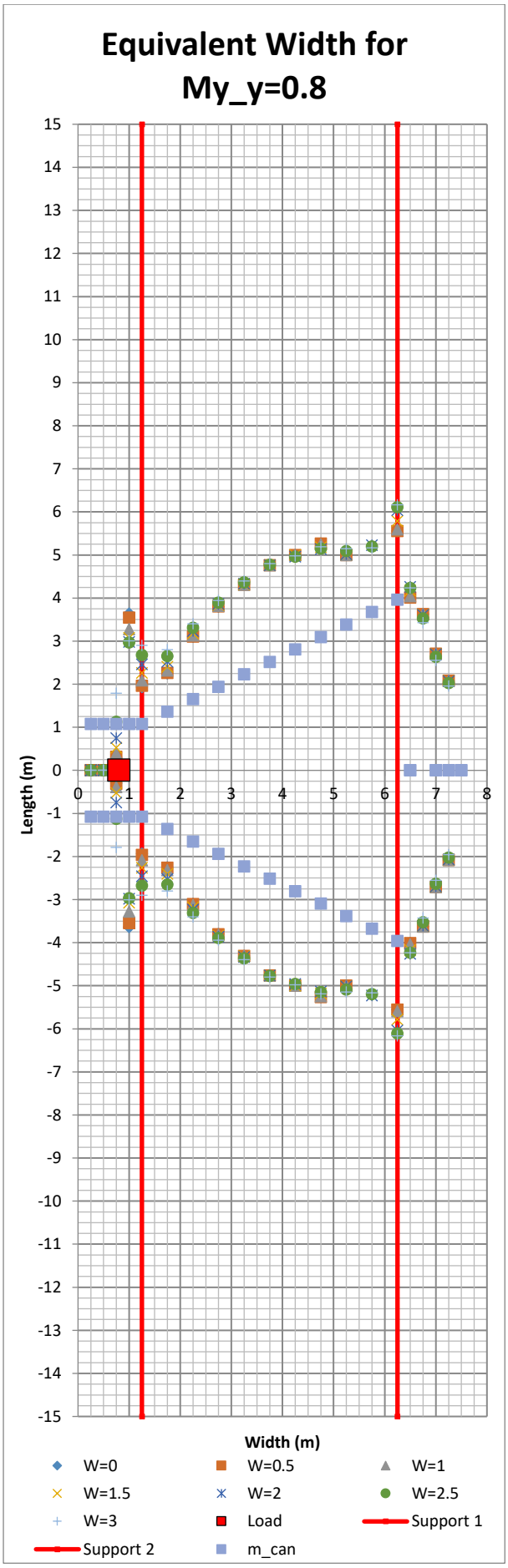

Equivalent Width for Vy_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

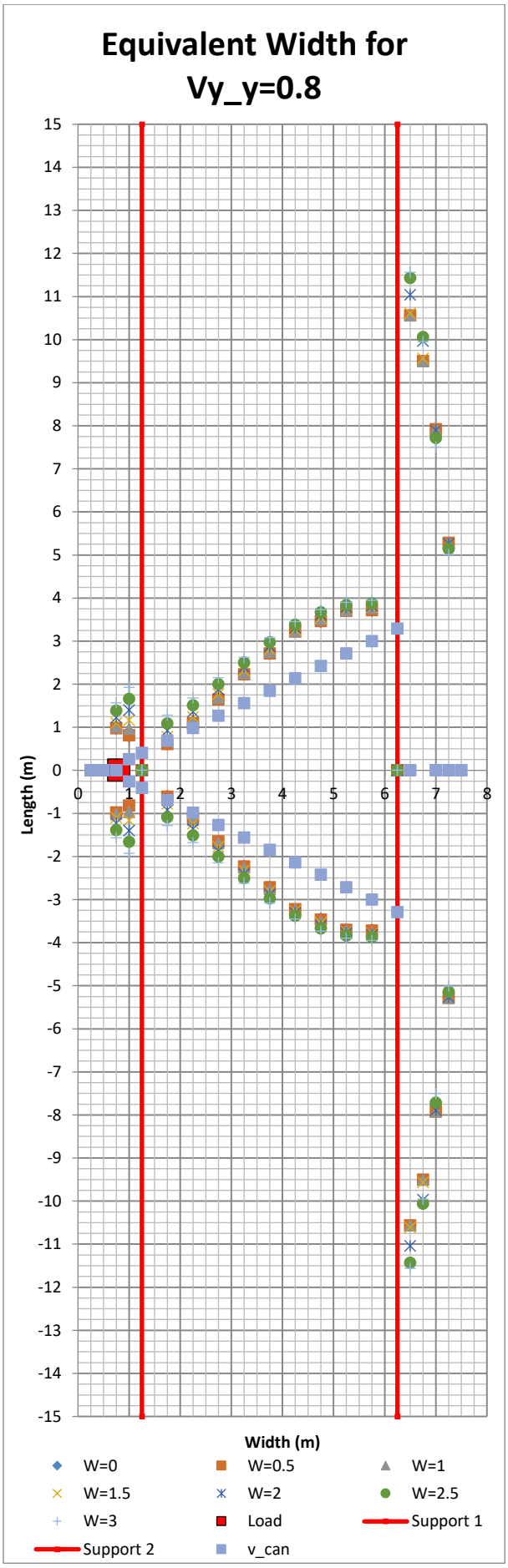

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

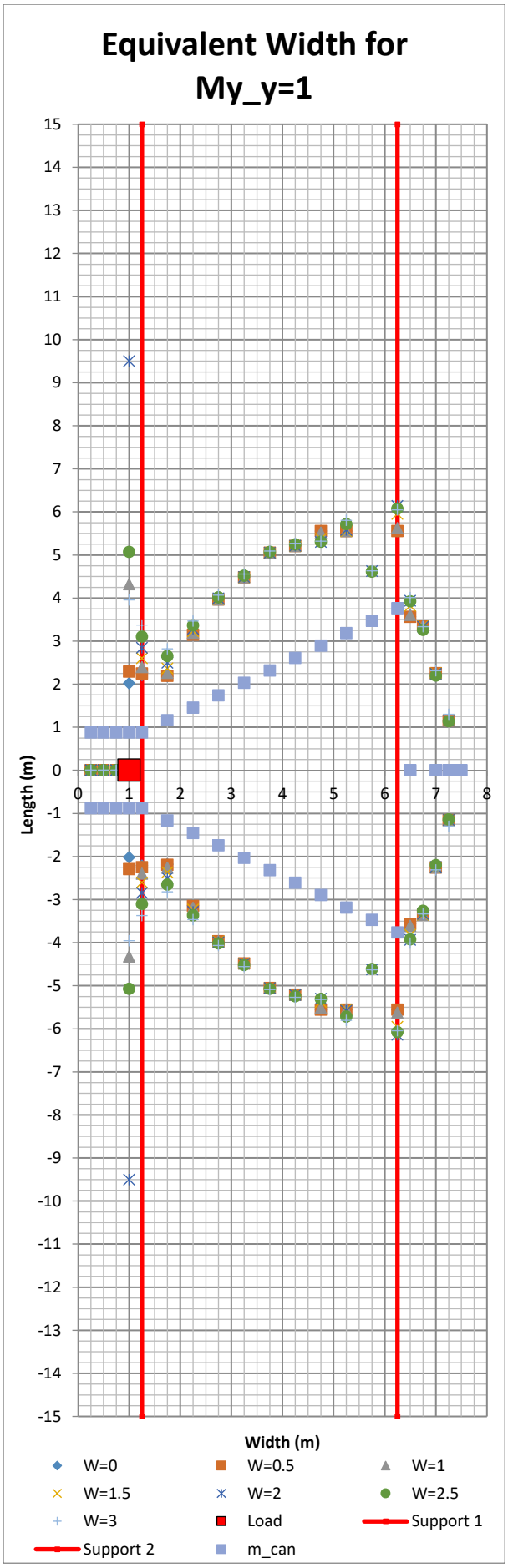

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

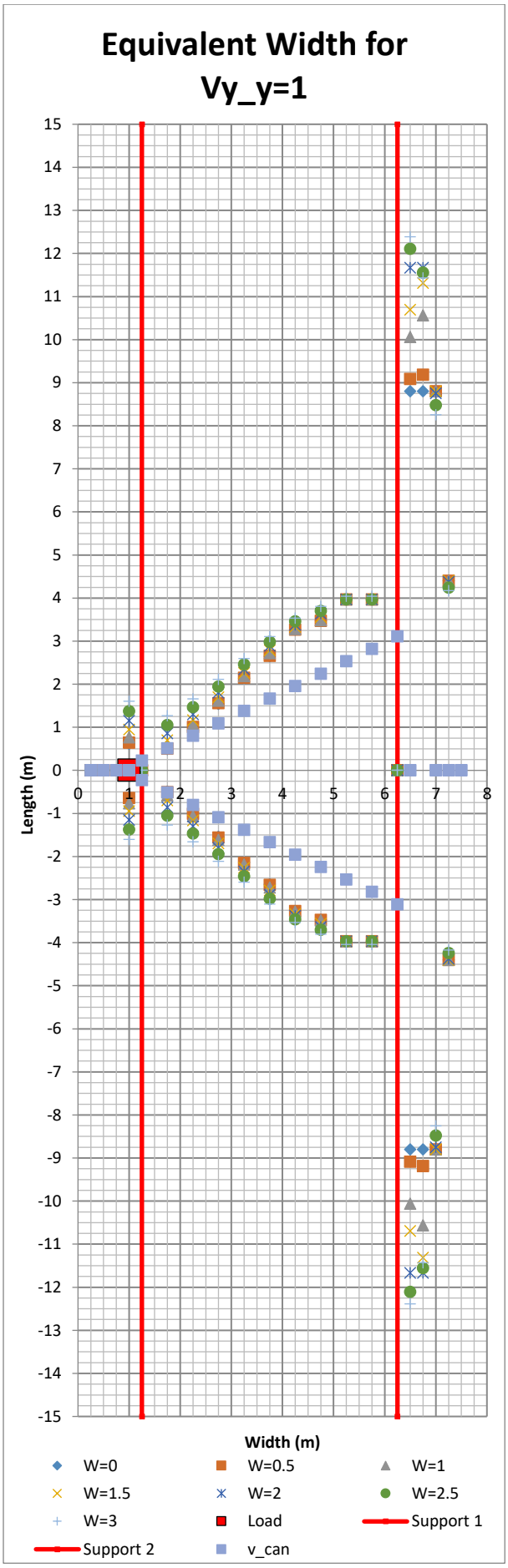

Equivalent Width for My_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

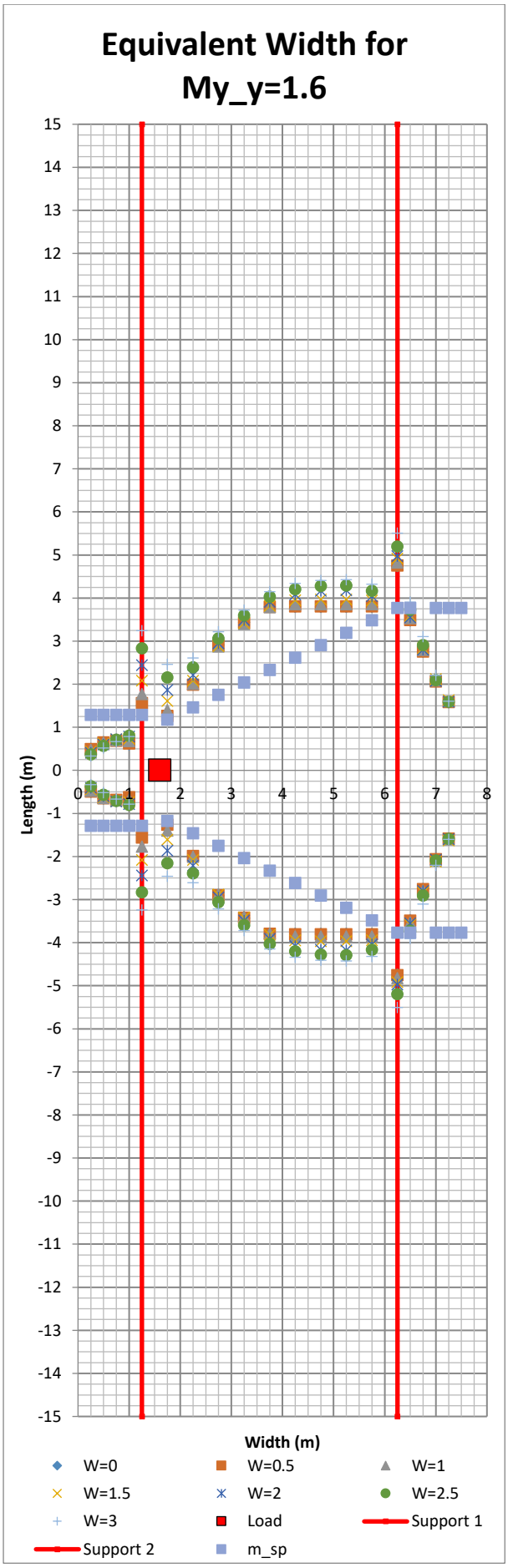

Equivalent Width for Vy_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

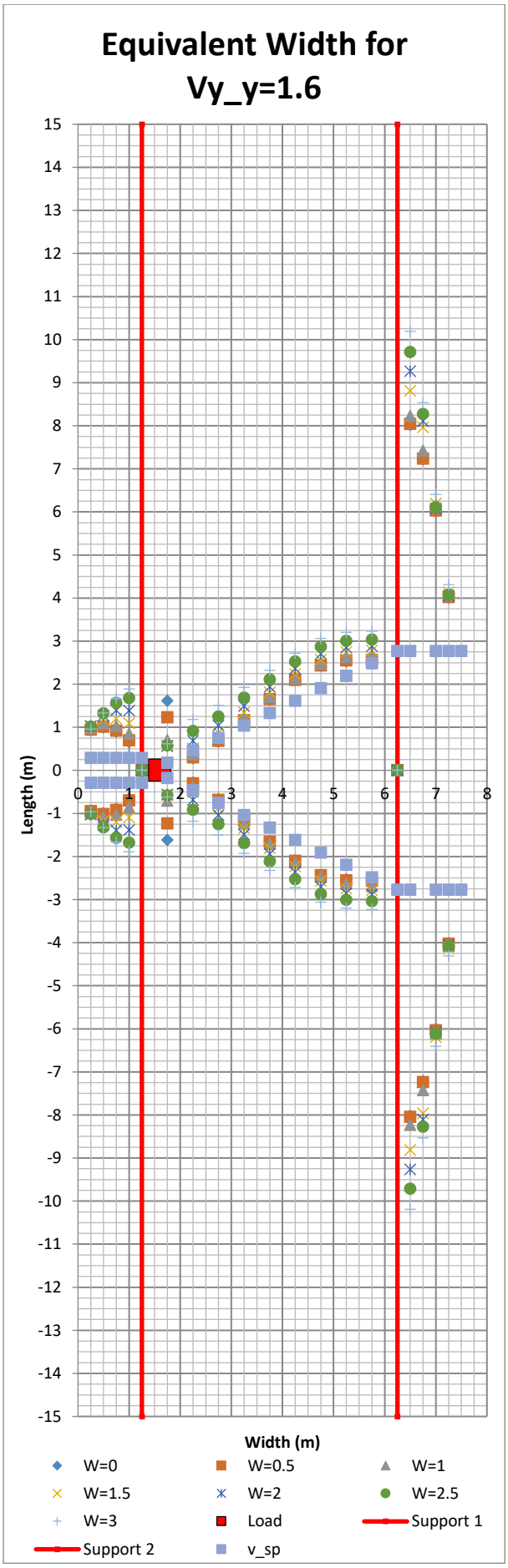

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

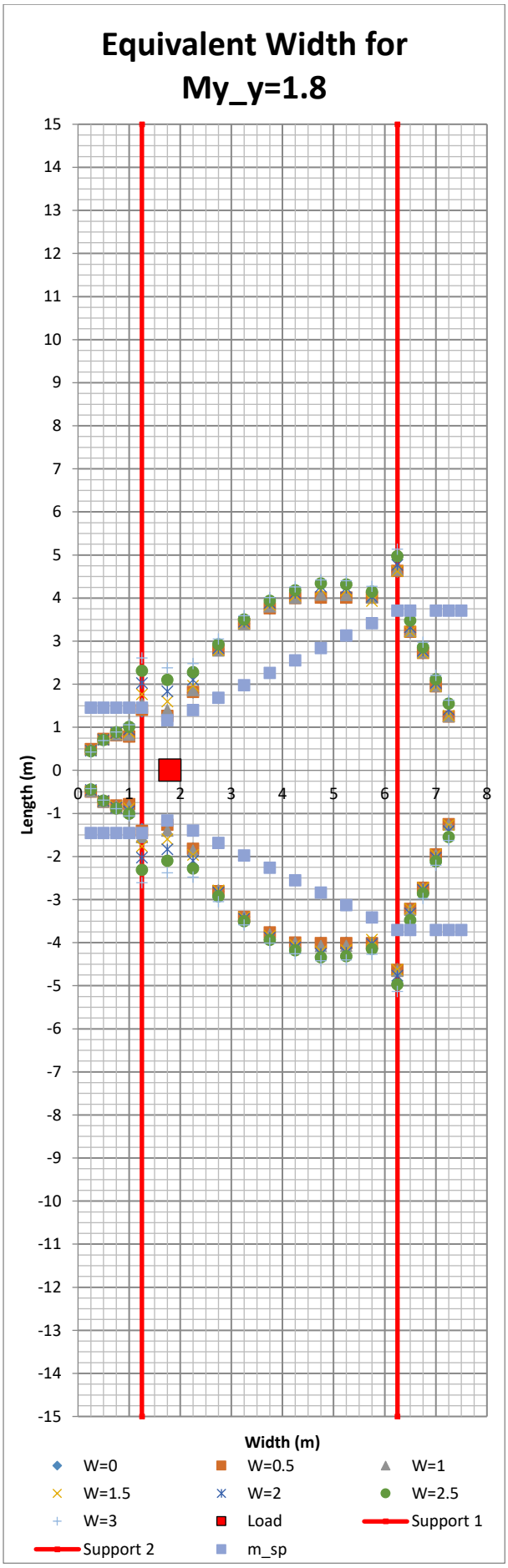

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

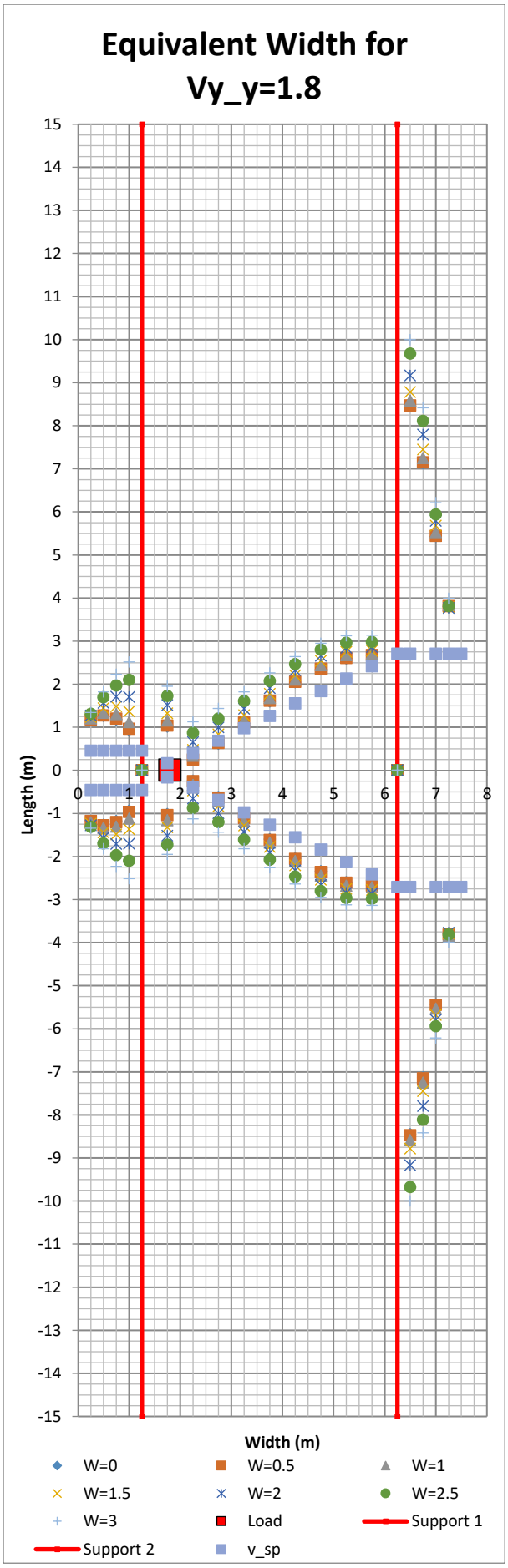

Equivalent Width for My_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

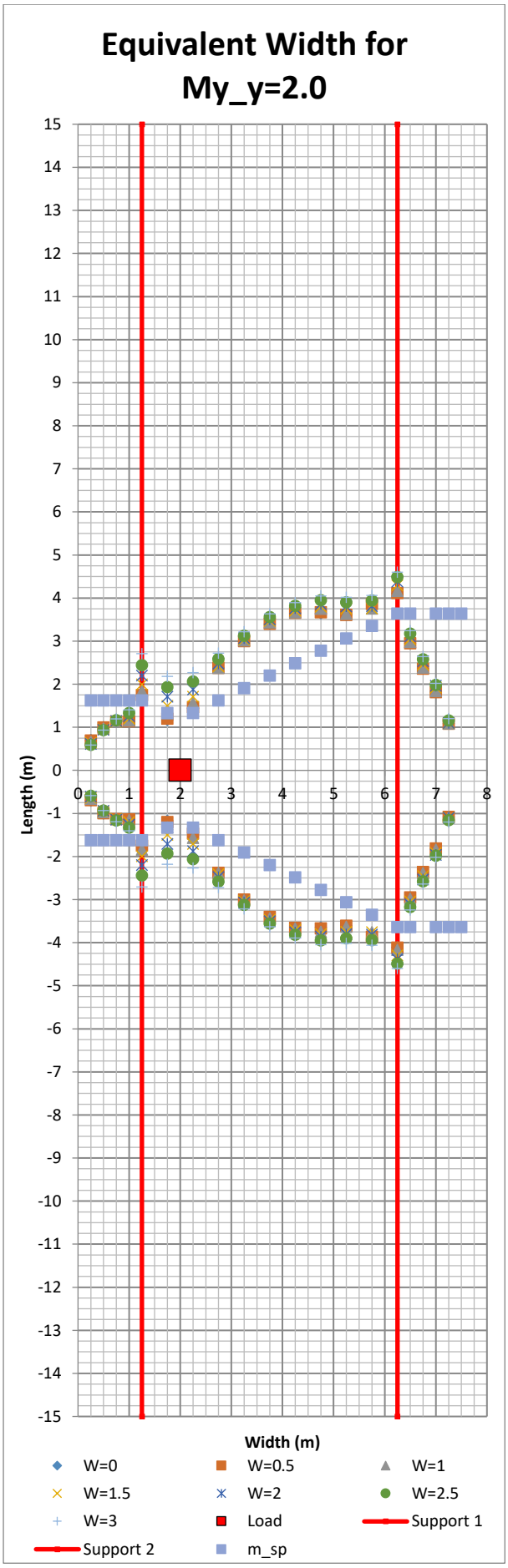

Equivalent Width for Vy_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

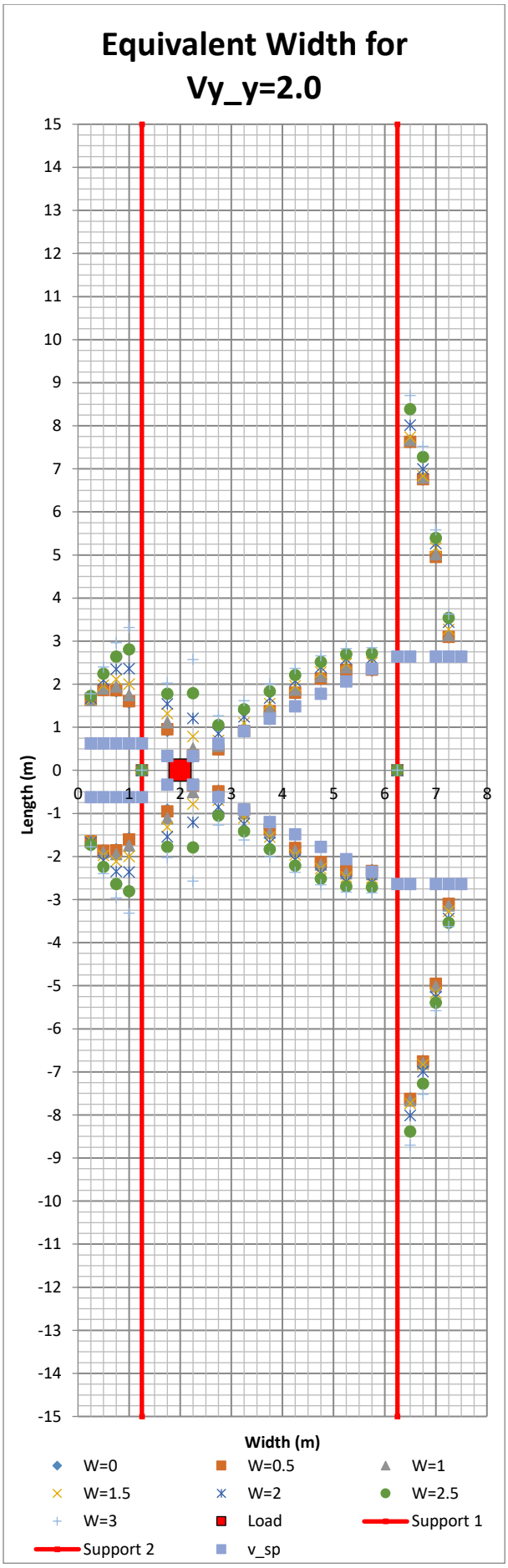

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

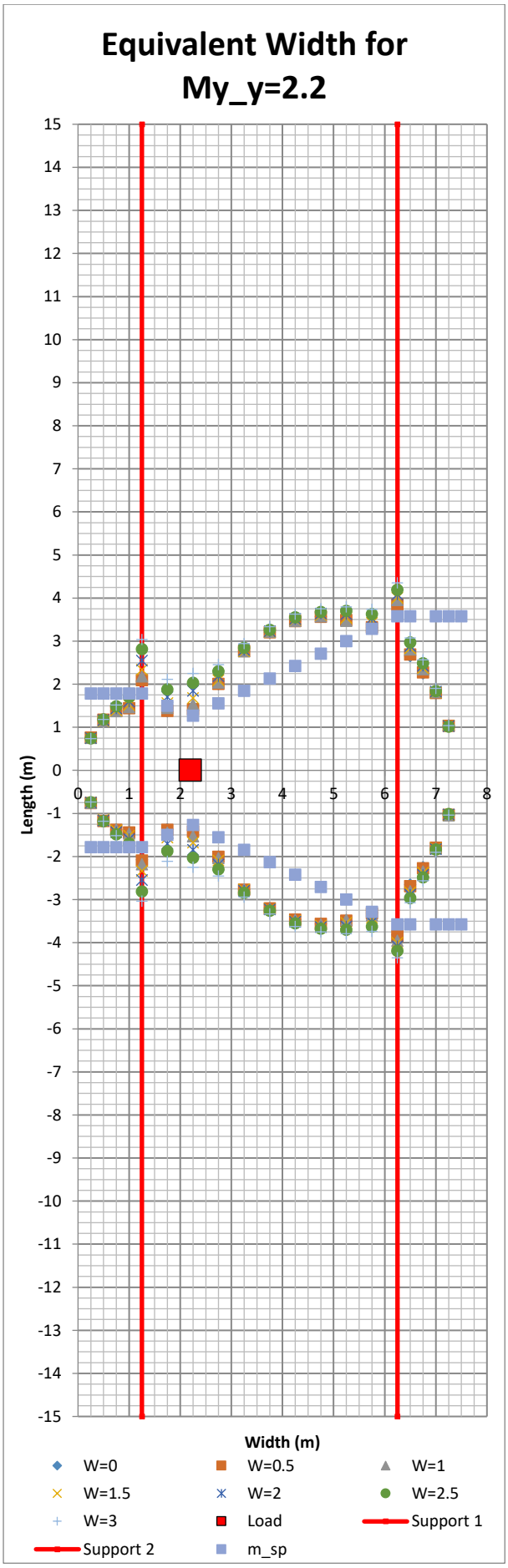

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

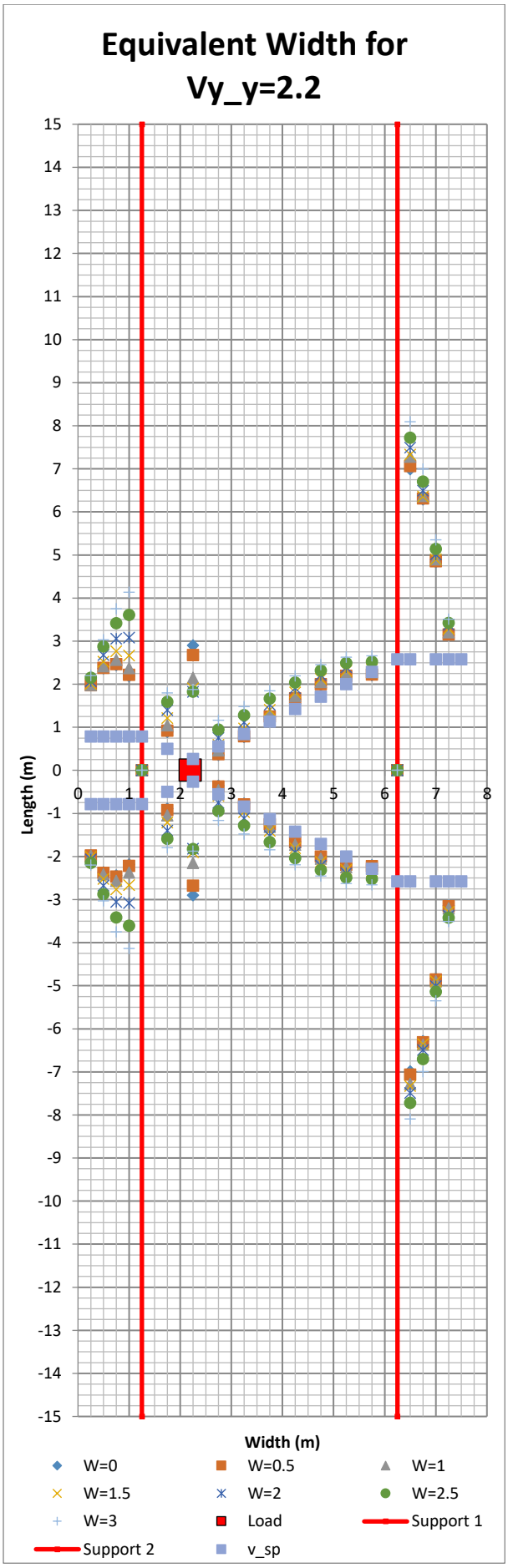

Equivalent Width for My_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

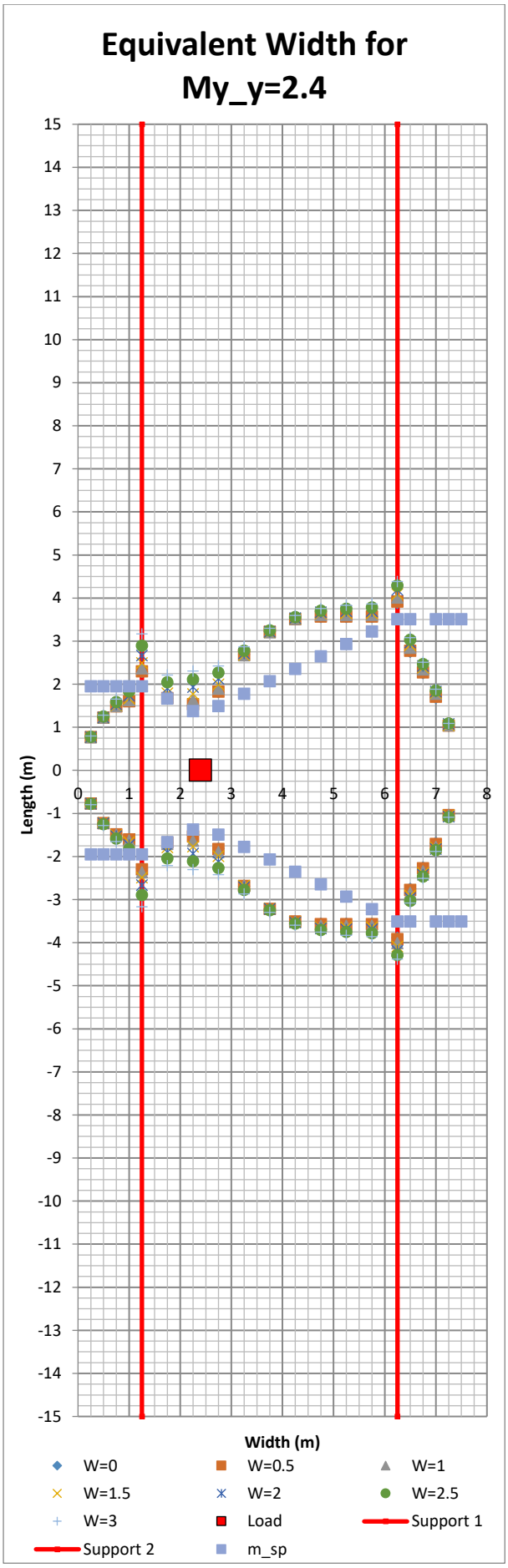

Equivalent Width for Vy_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

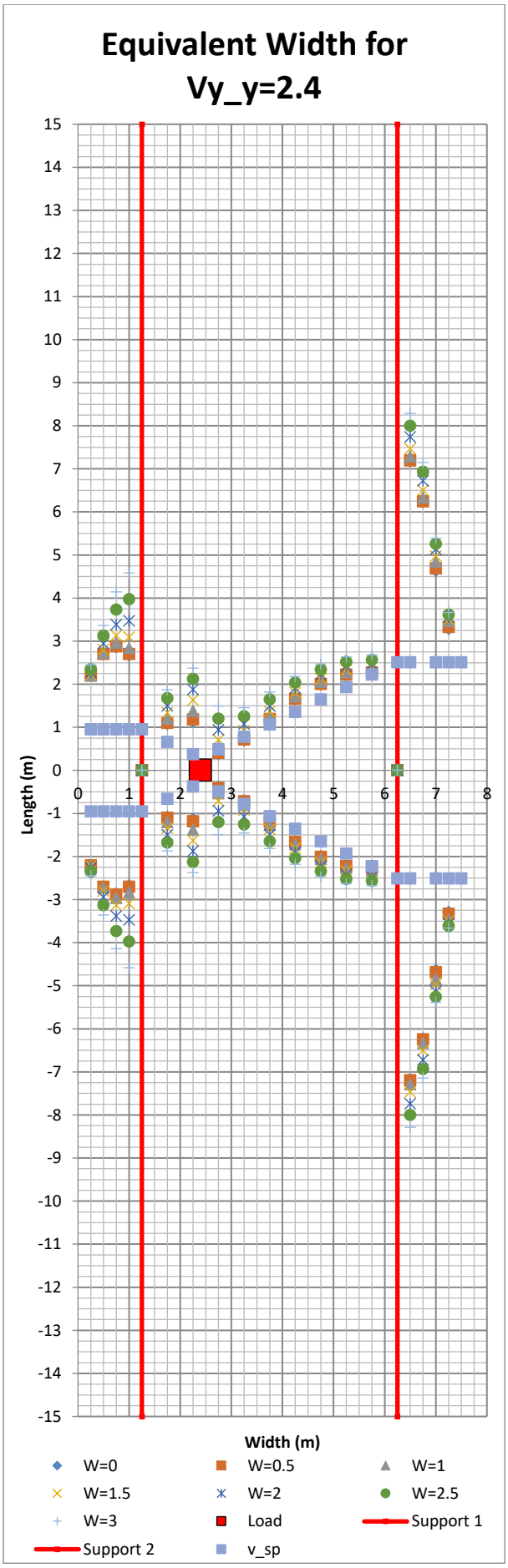

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

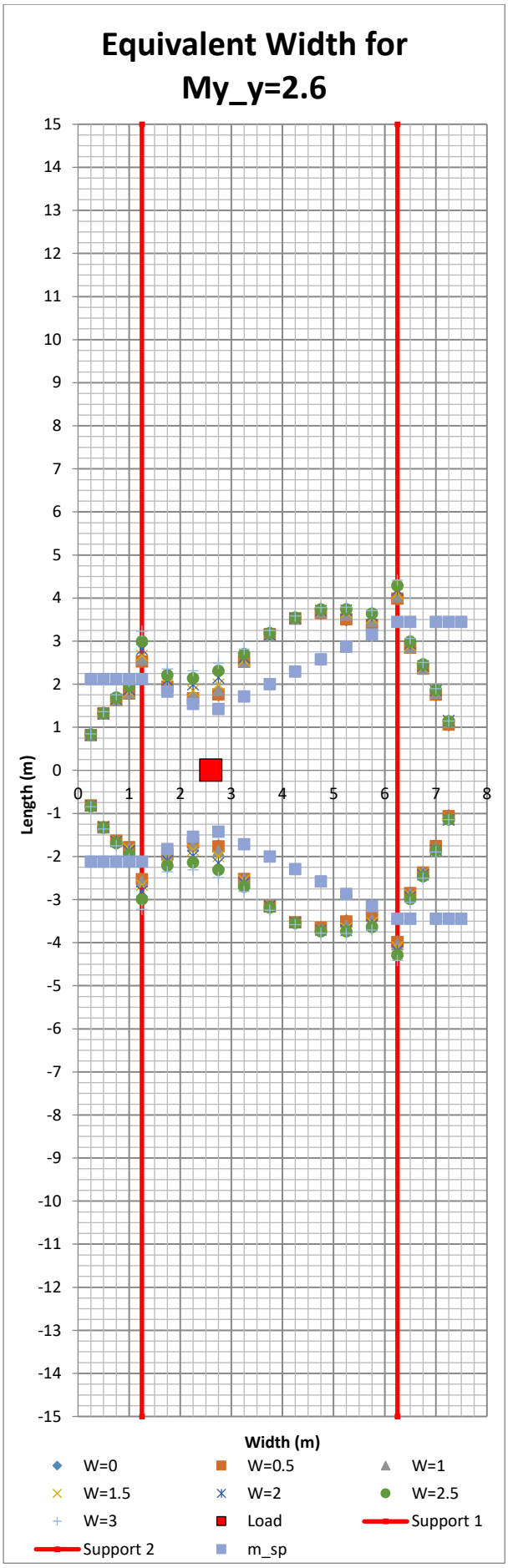

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

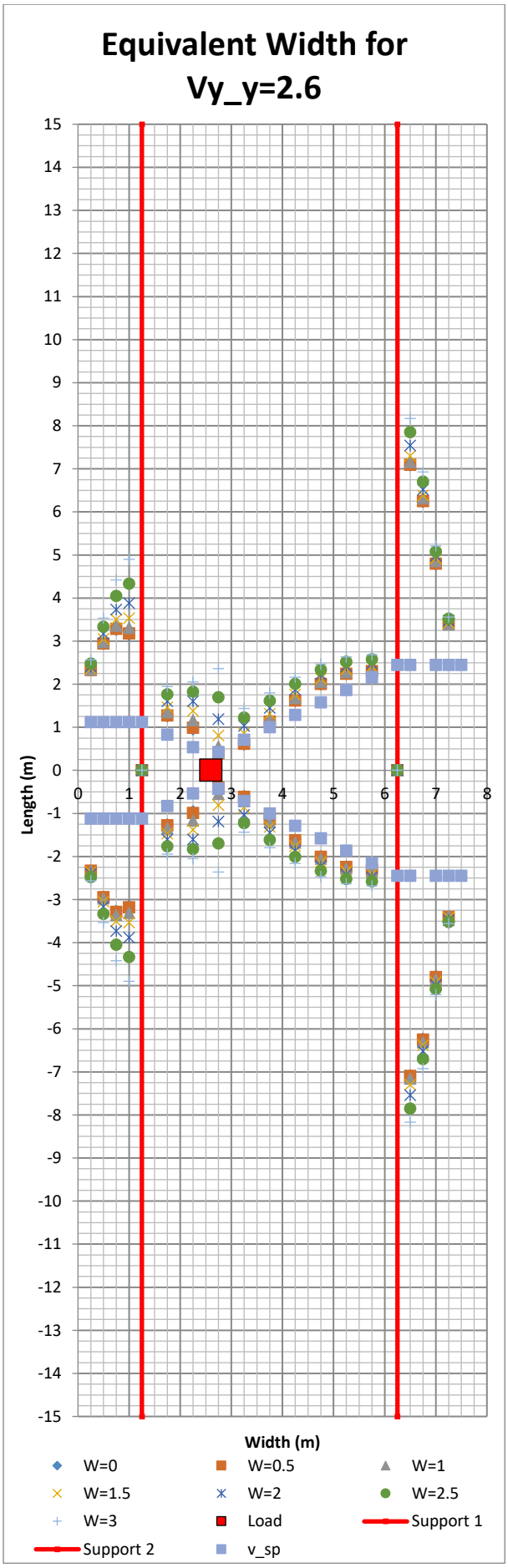

Equivalent Width for My_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

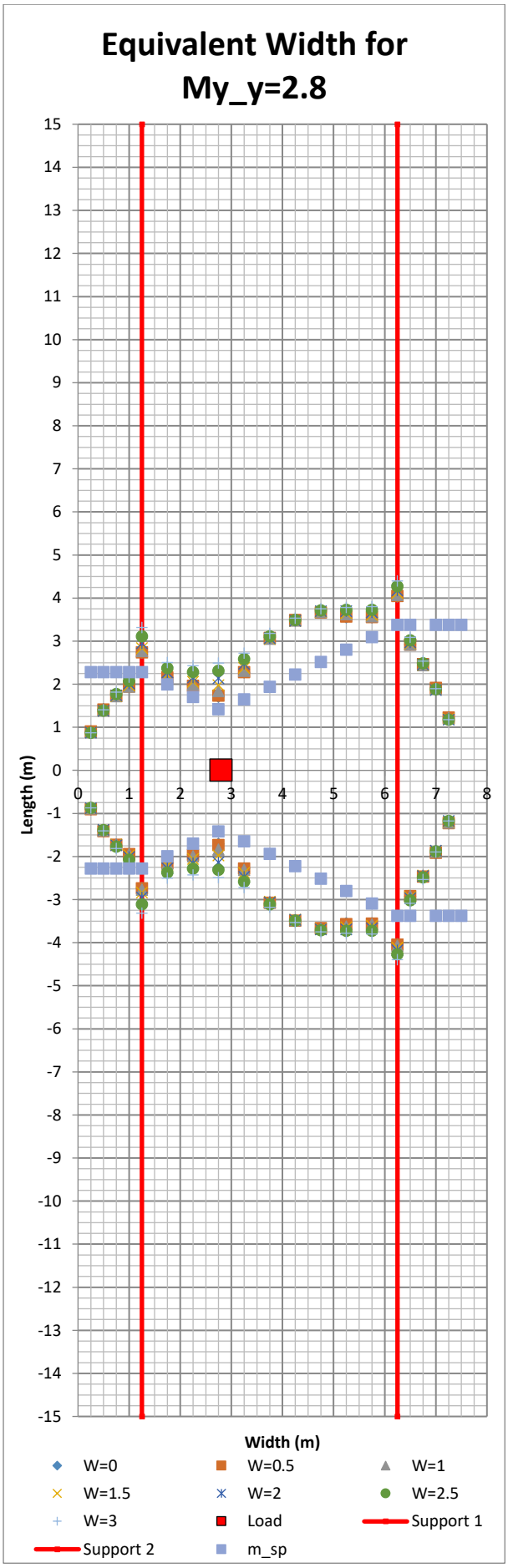

Equivalent Width for Vy_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

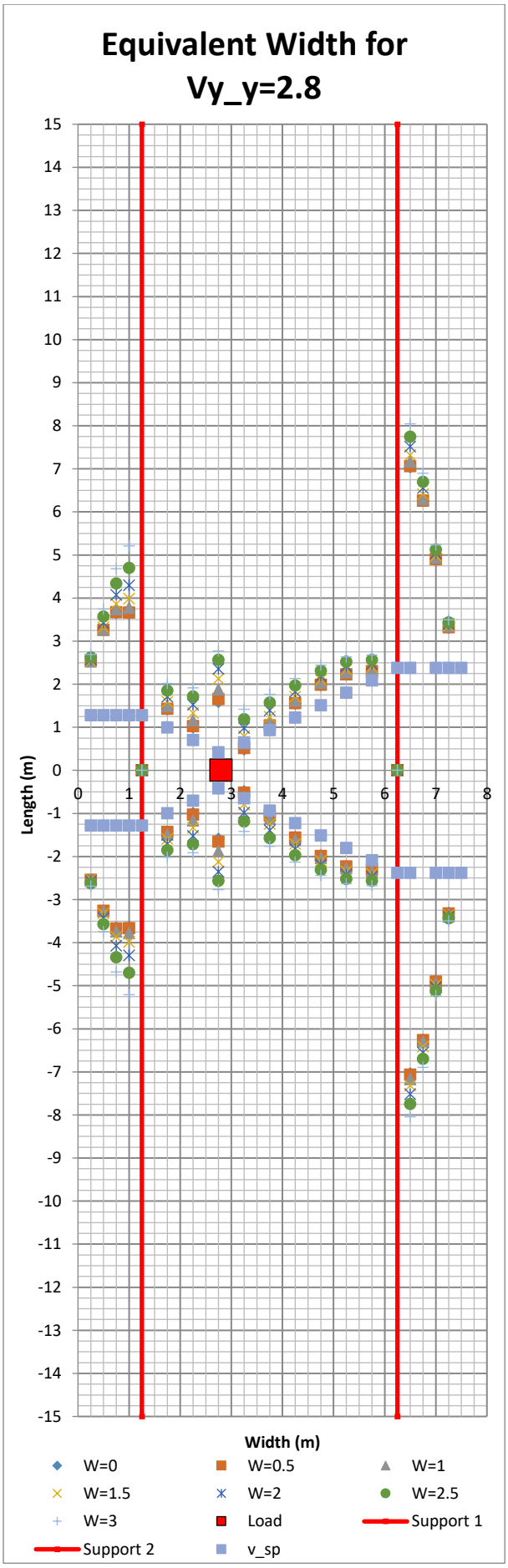

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

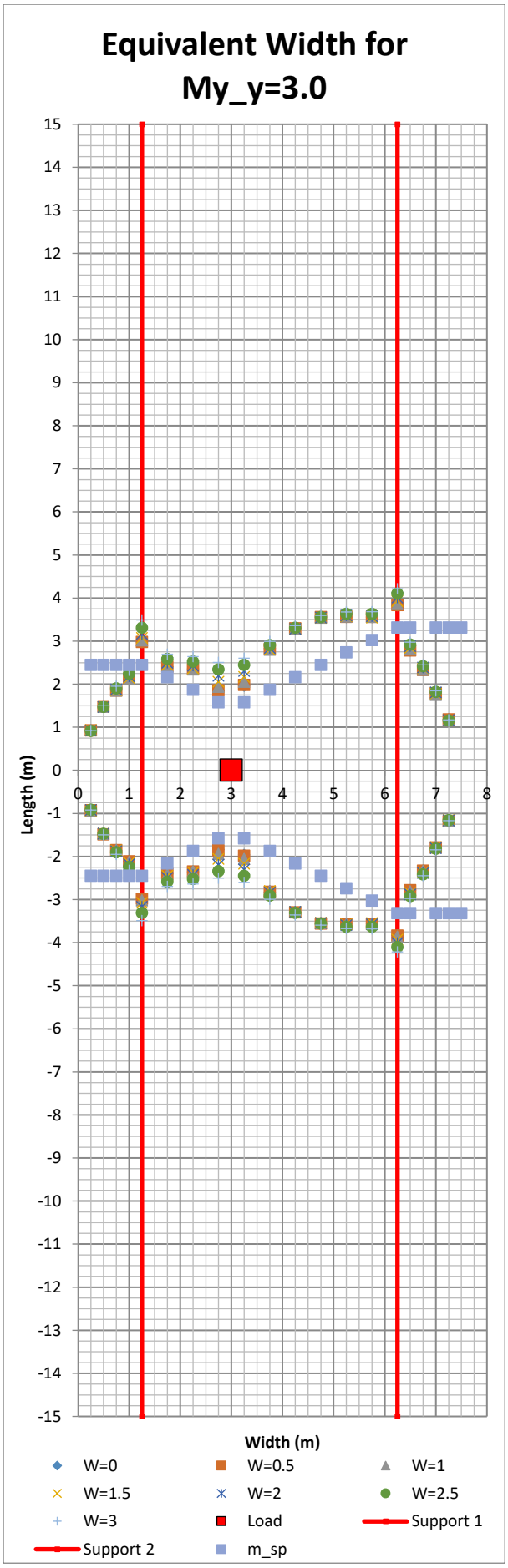

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

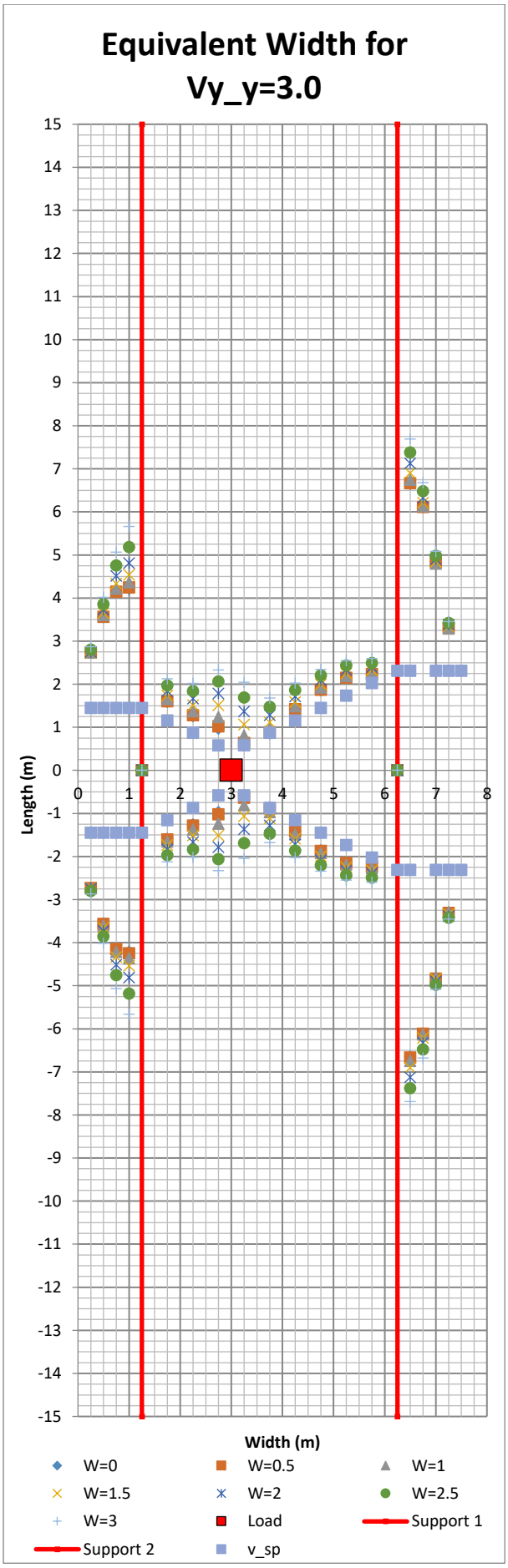

Equivalent Width for My_y=2.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

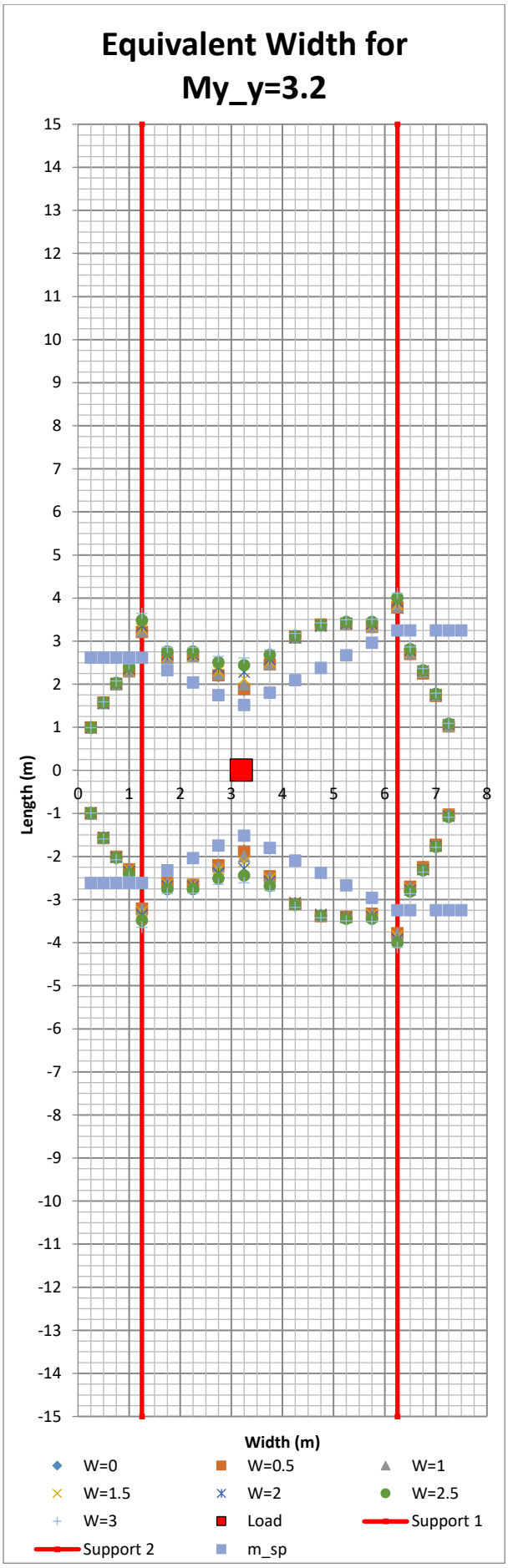

Equivalent Width for Vy_y=2.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

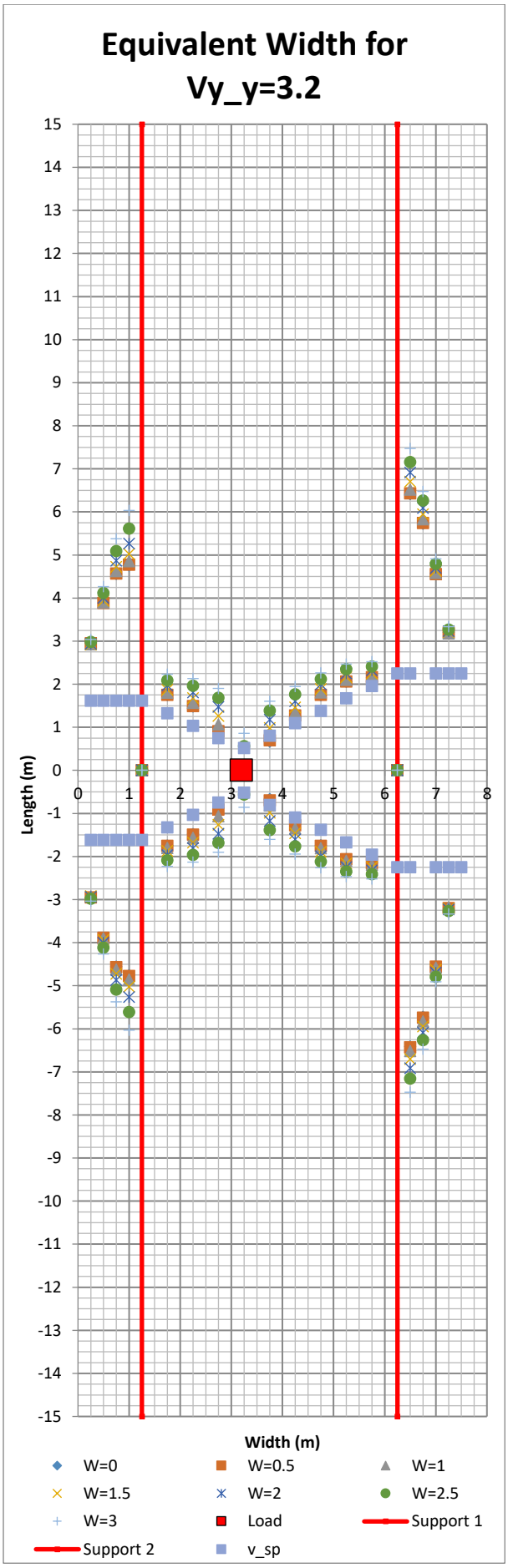

Equivalent Width for My_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

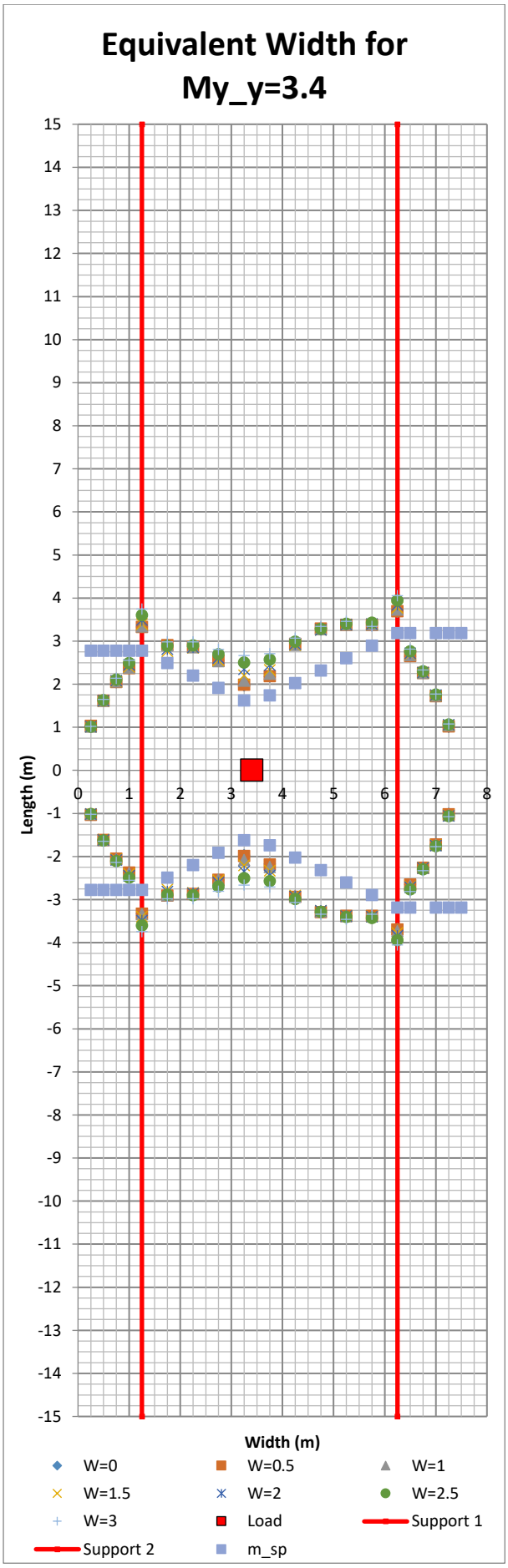

Equivalent Width for Vy_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

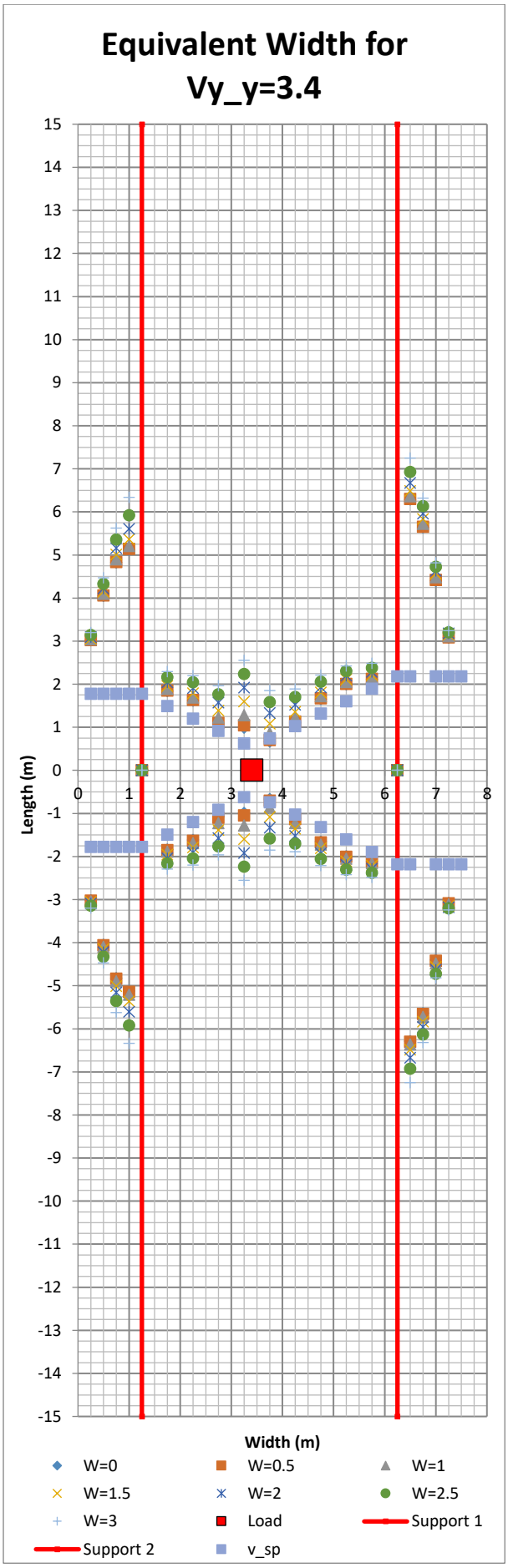

Equivalent Width for My_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

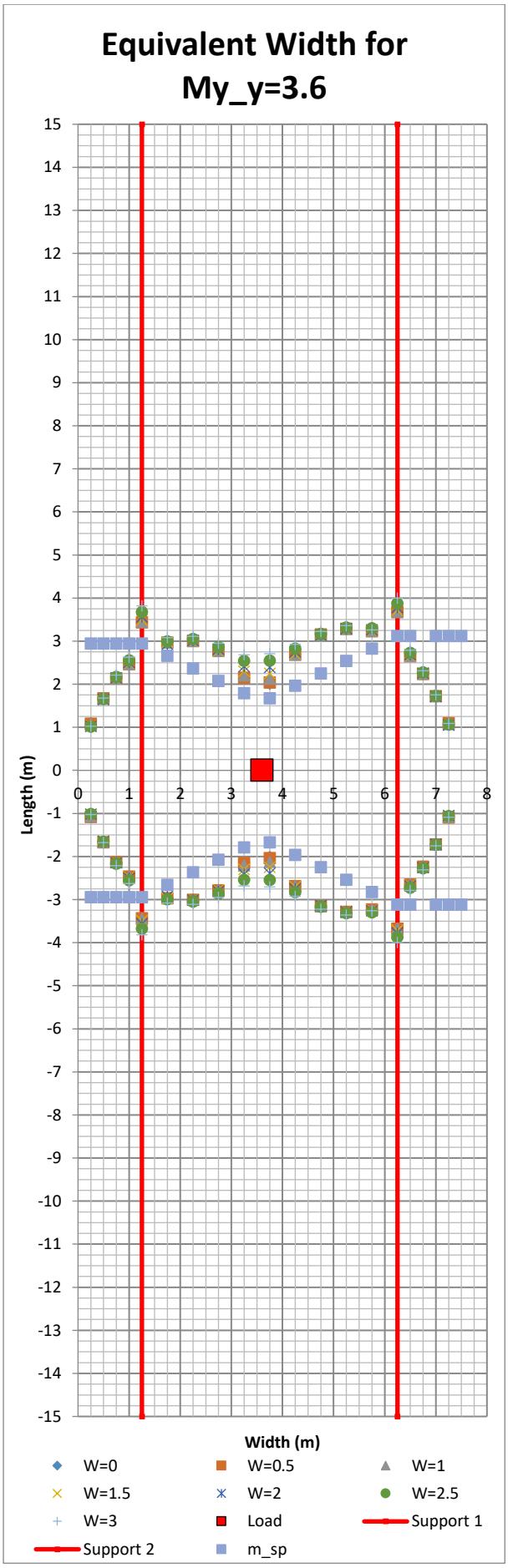

Equivalent Width for Vy_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

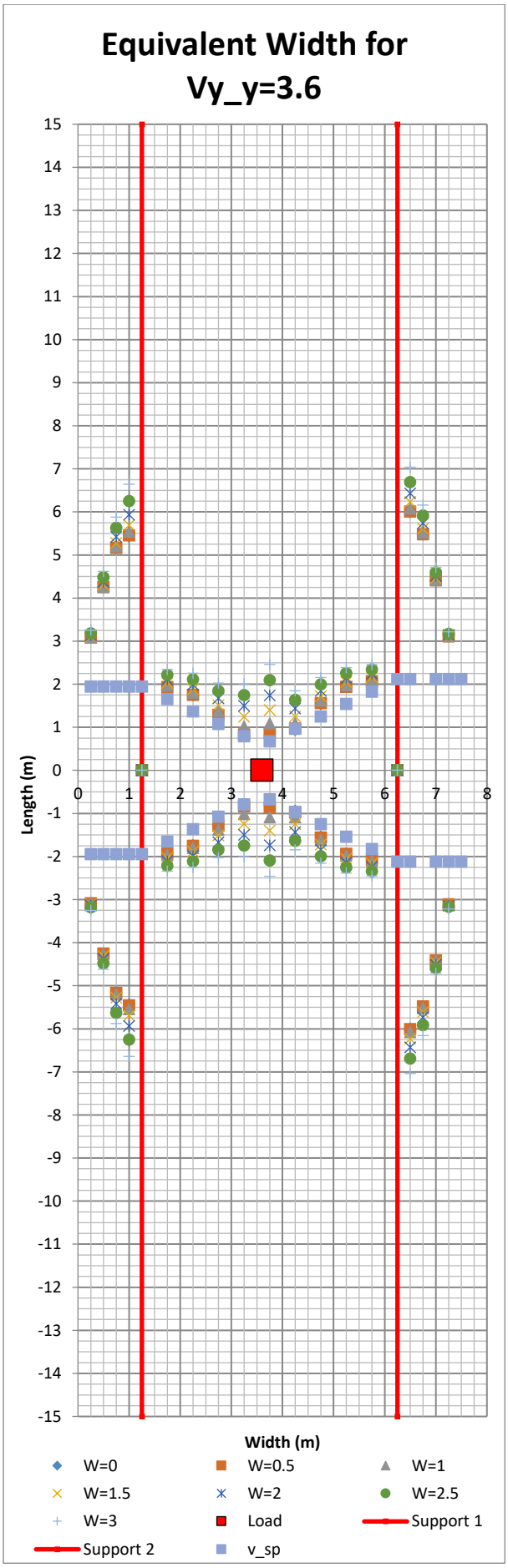

Equivalent Width for My_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

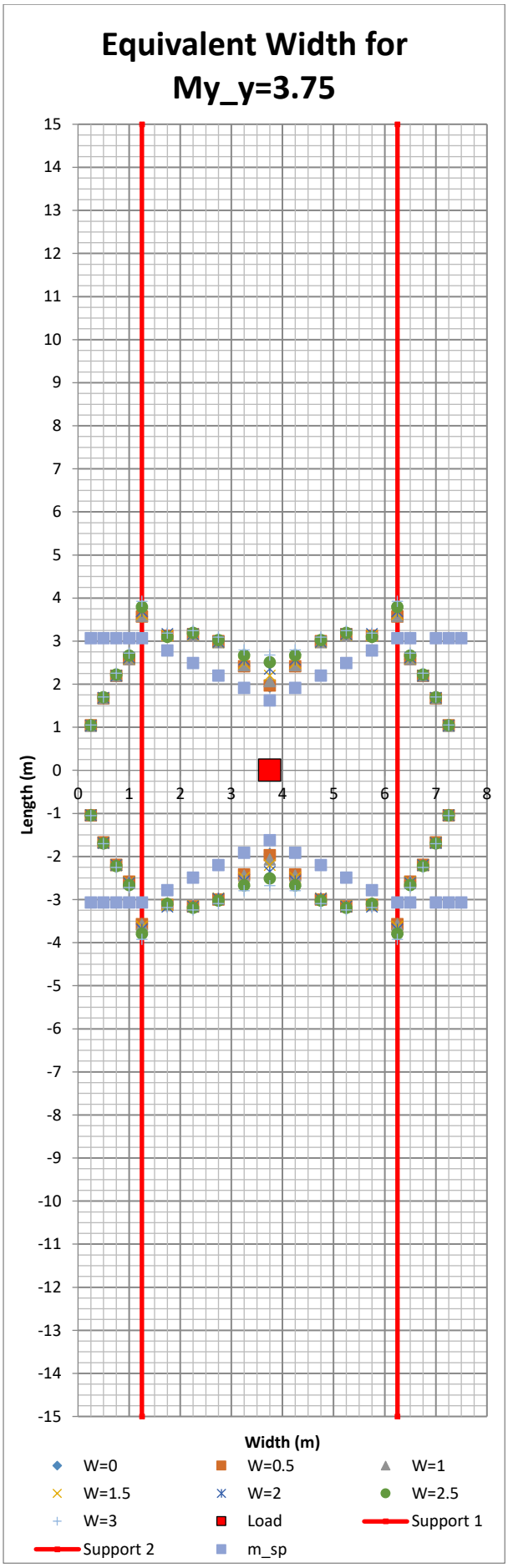

Equivalent Width for Vy_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

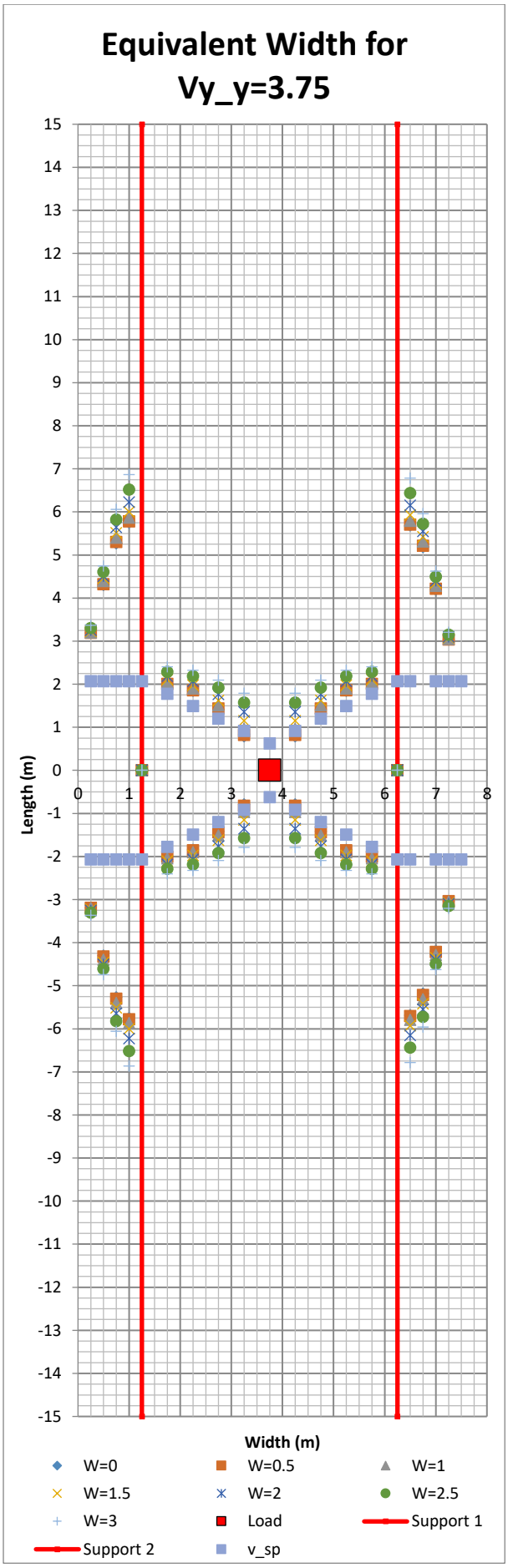

Equivalent Width for My_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

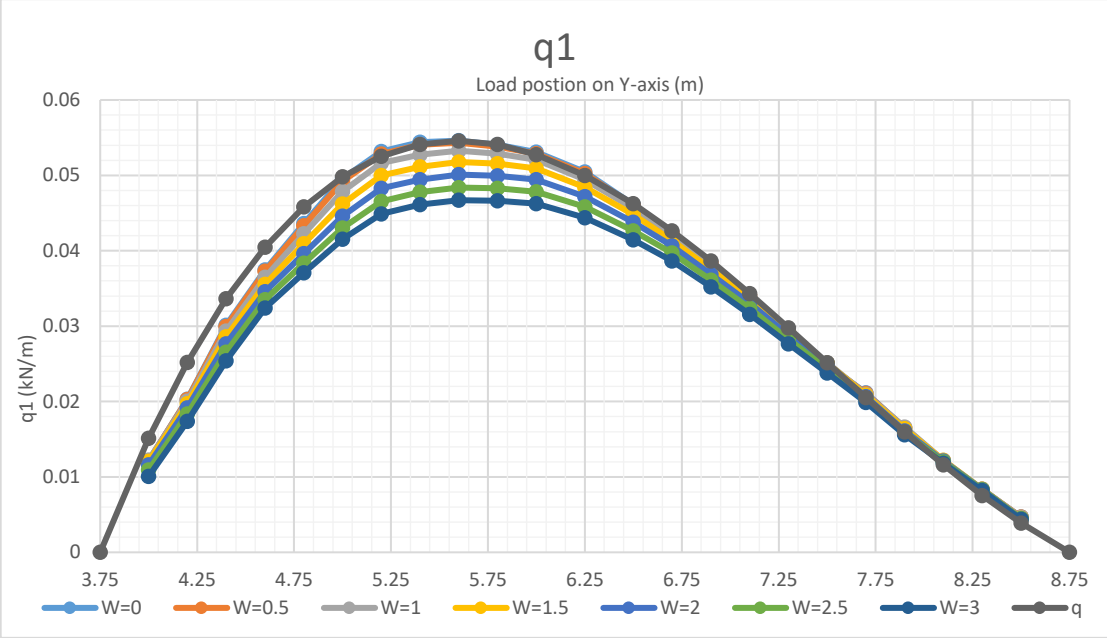

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

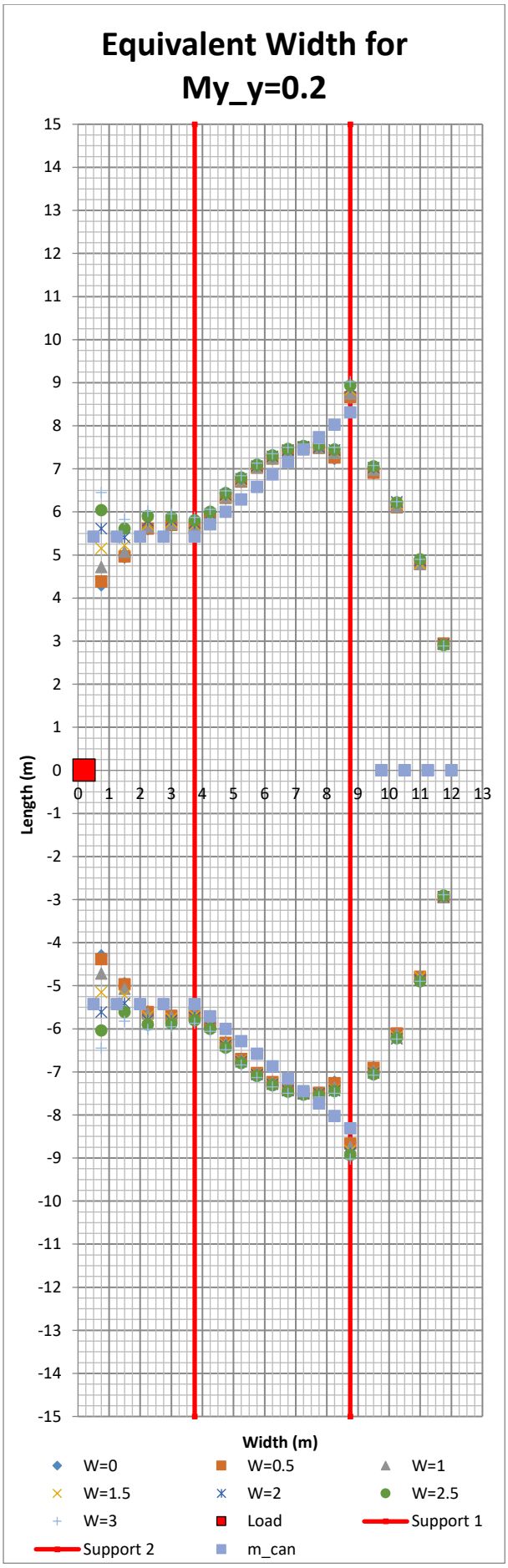

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

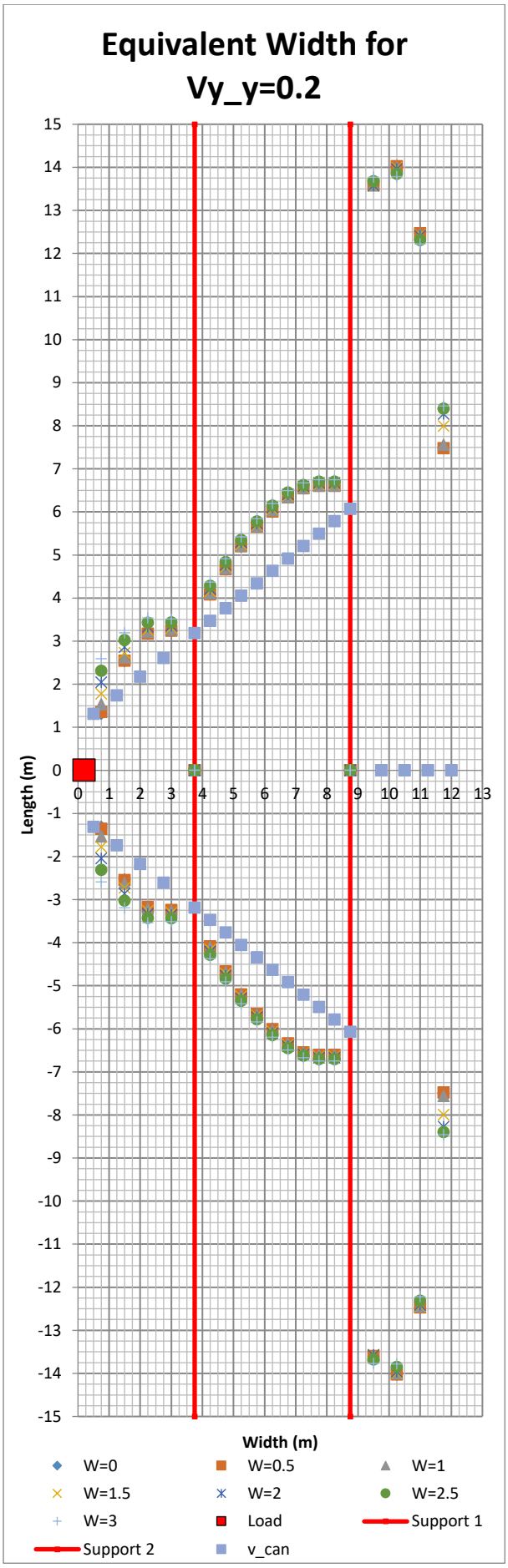

Equivalent Width for My_y=3.75
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

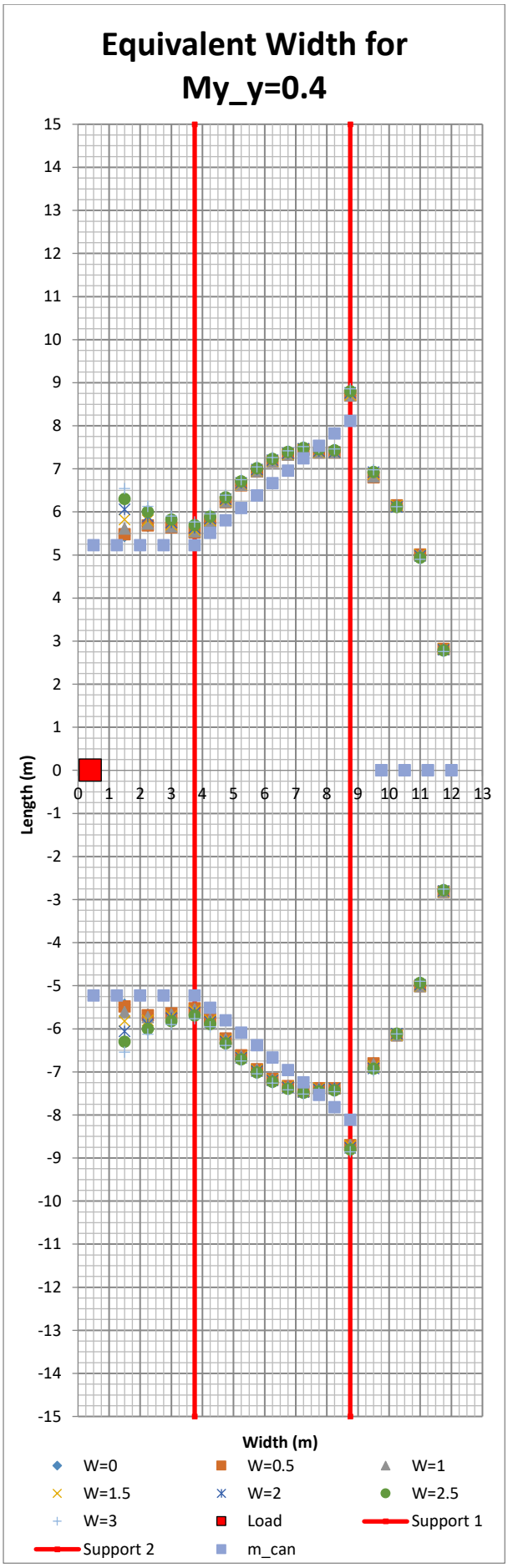

Equivalent Width for Vy_y=3.75
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

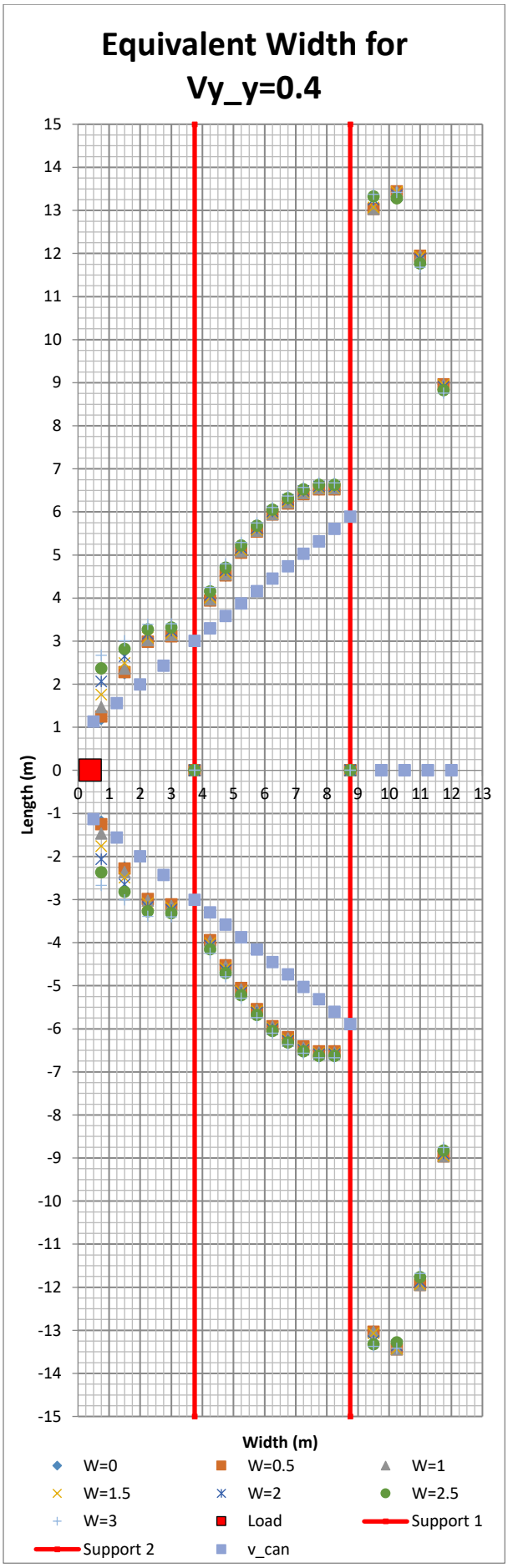
11.5 Fifth Variation $\alpha = 0.75,\ l_m = 5$

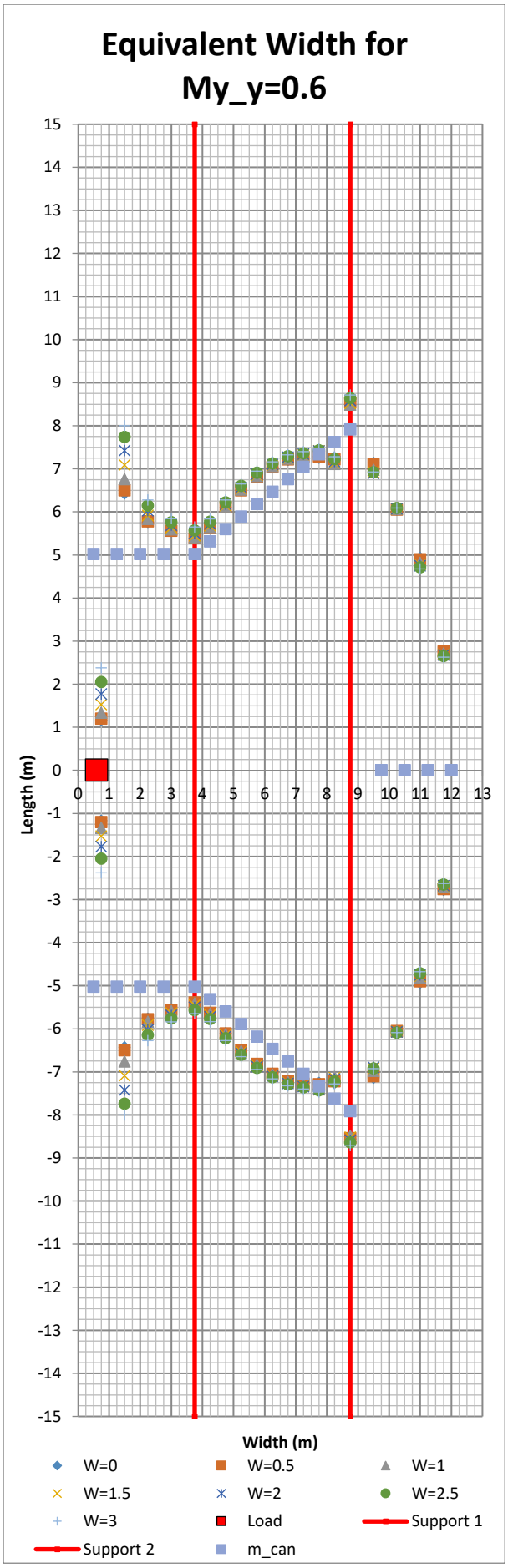


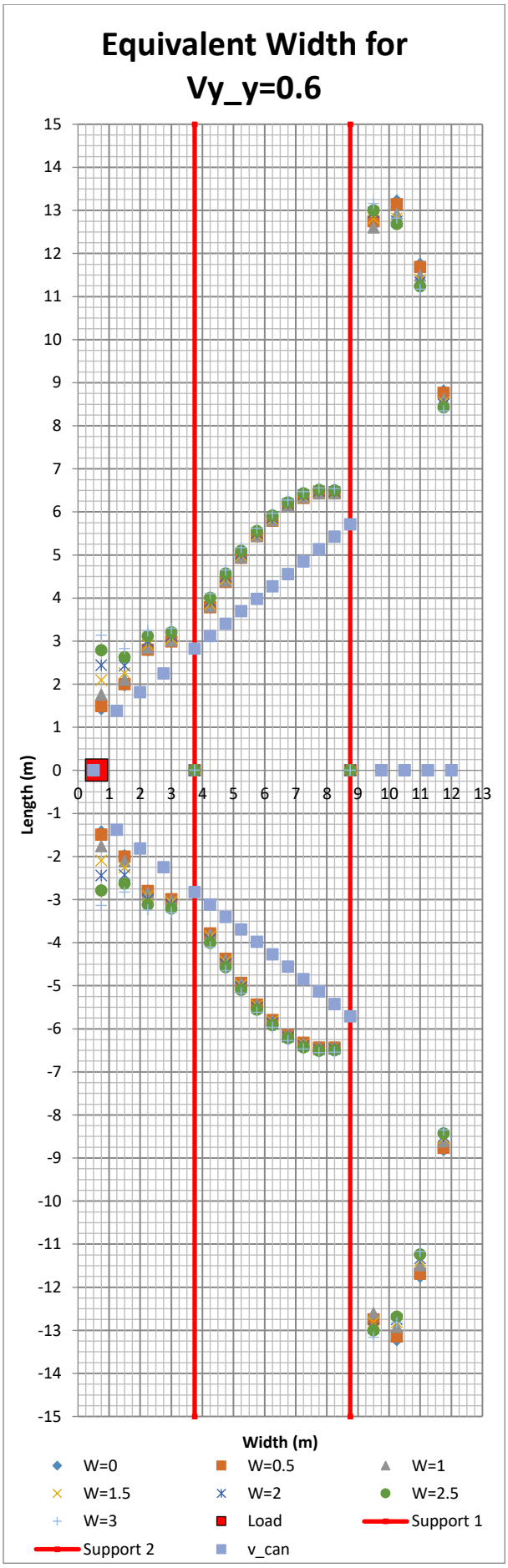

Equivalent Width for My_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

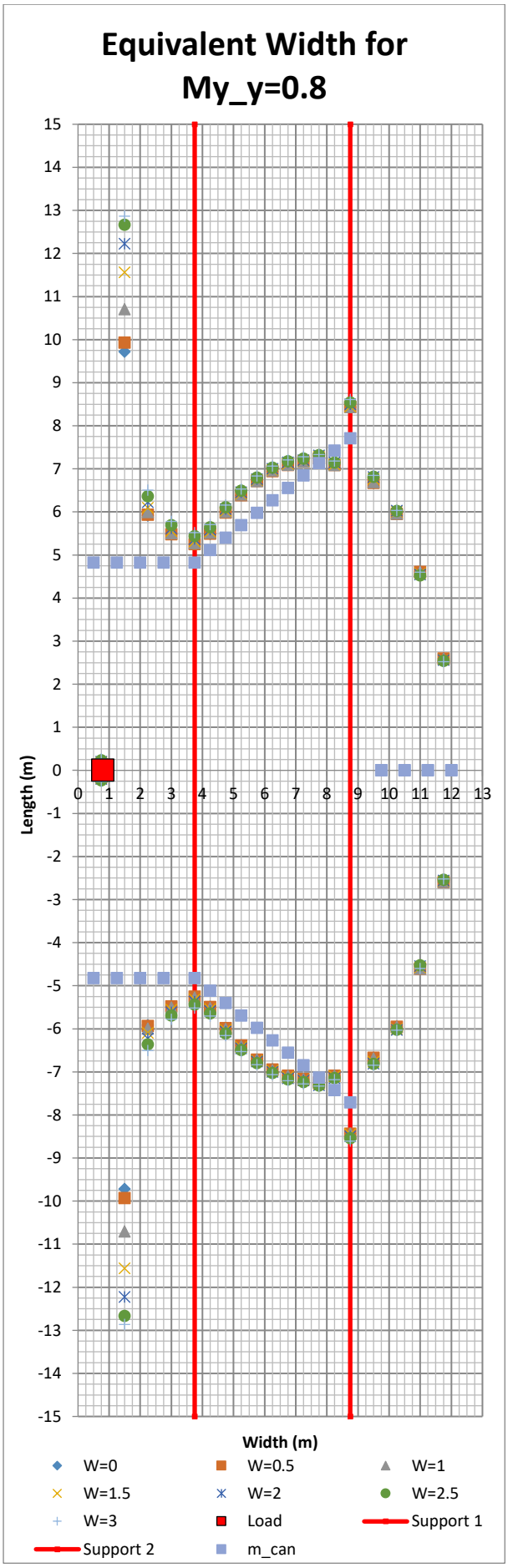

Equivalent Width for Vy_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

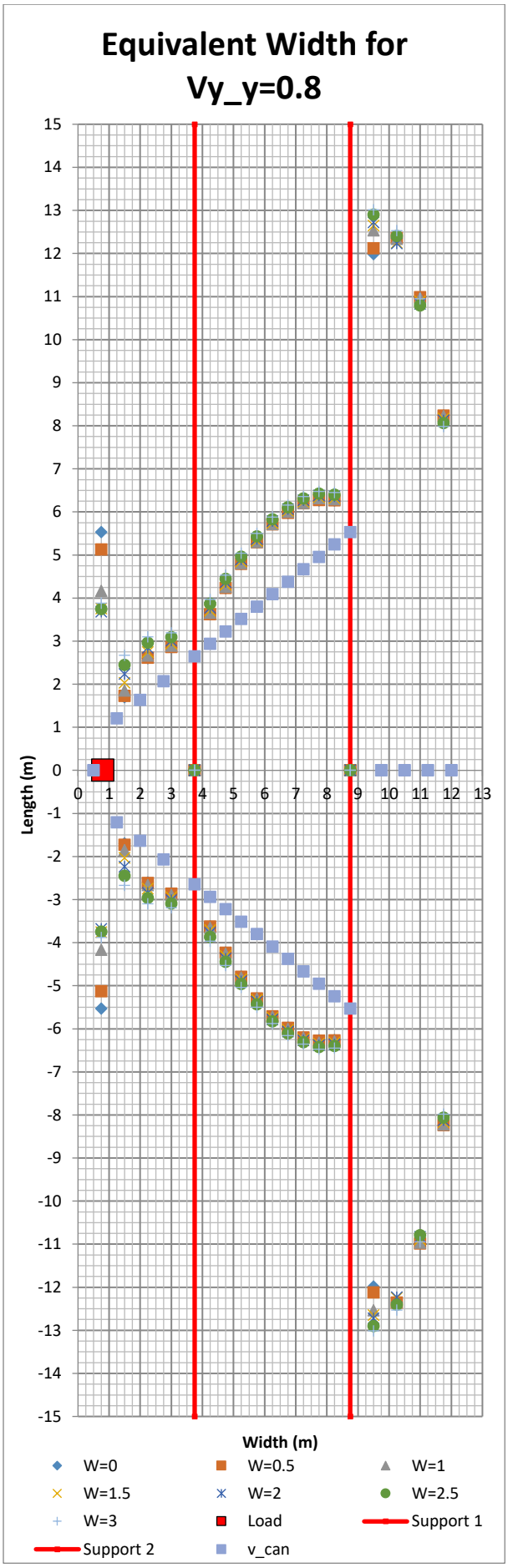

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

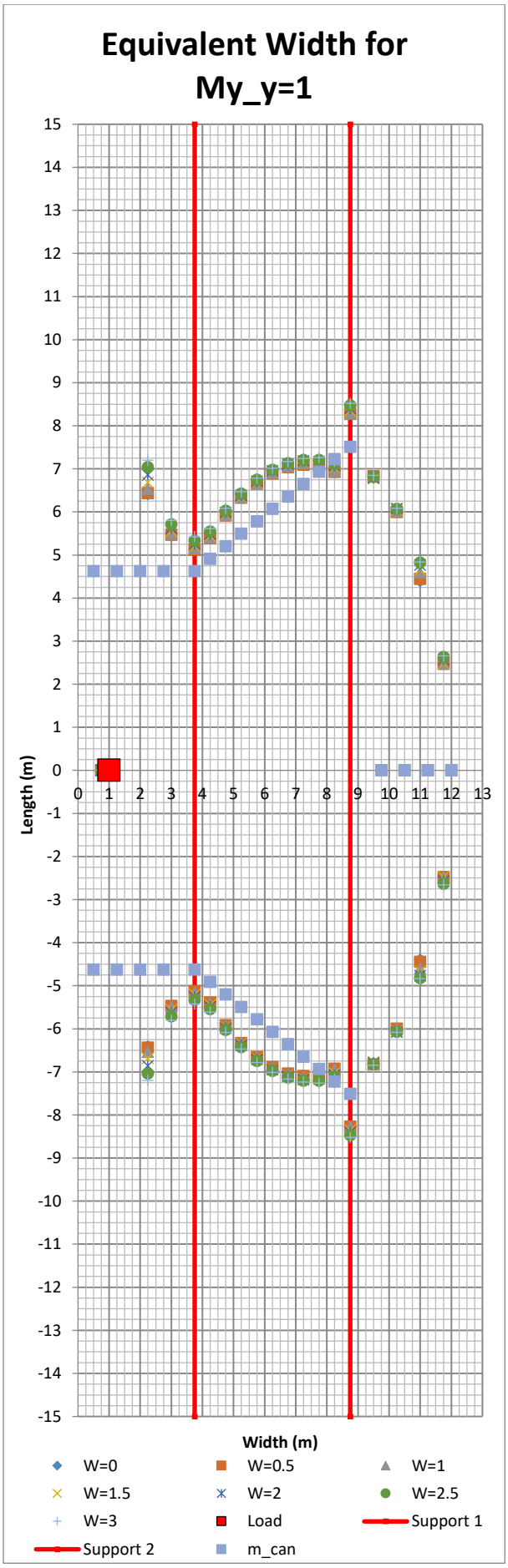

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

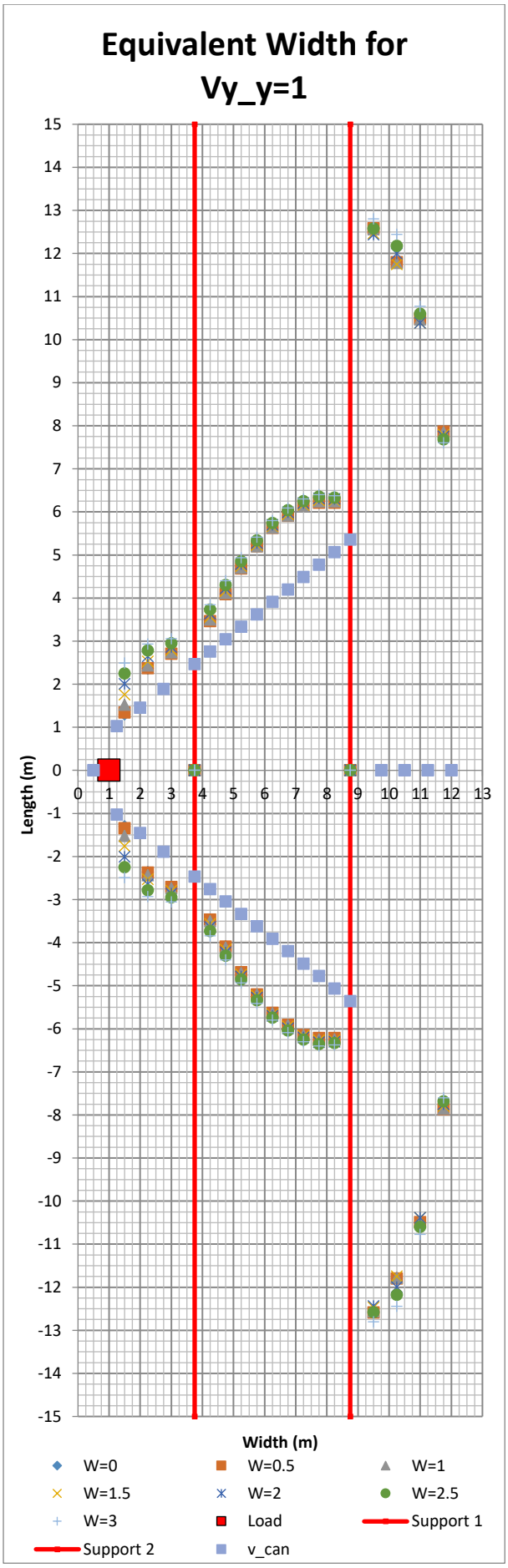

Equivalent Width for My_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

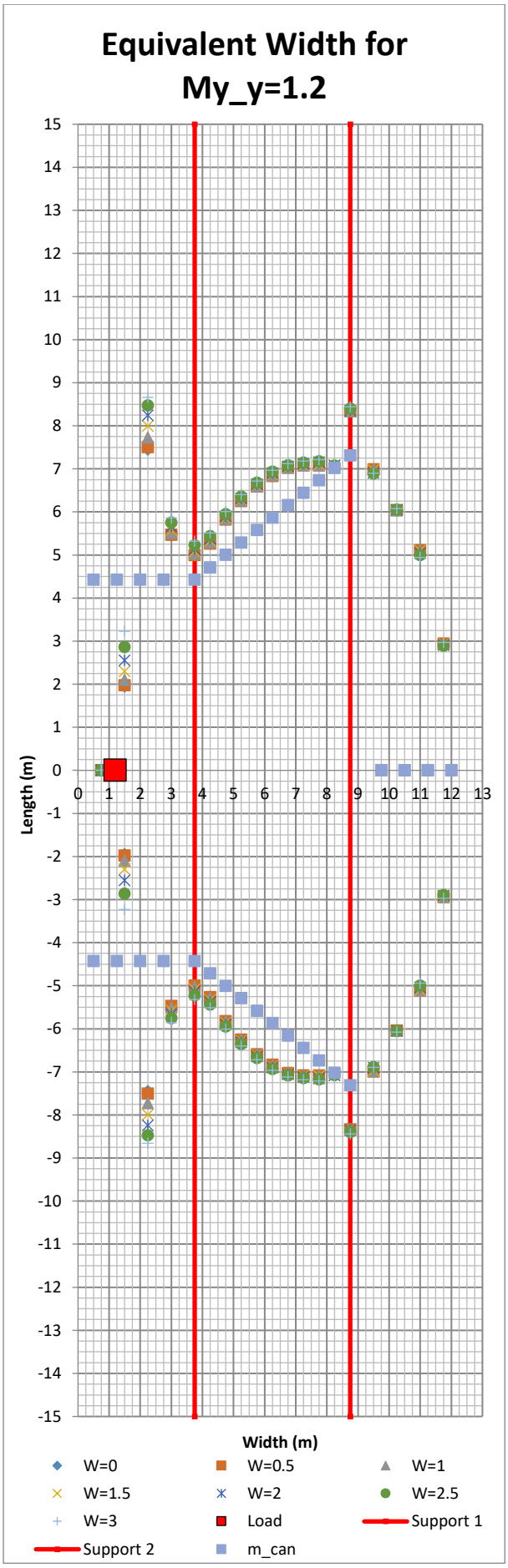

Equivalent Width for Vy_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

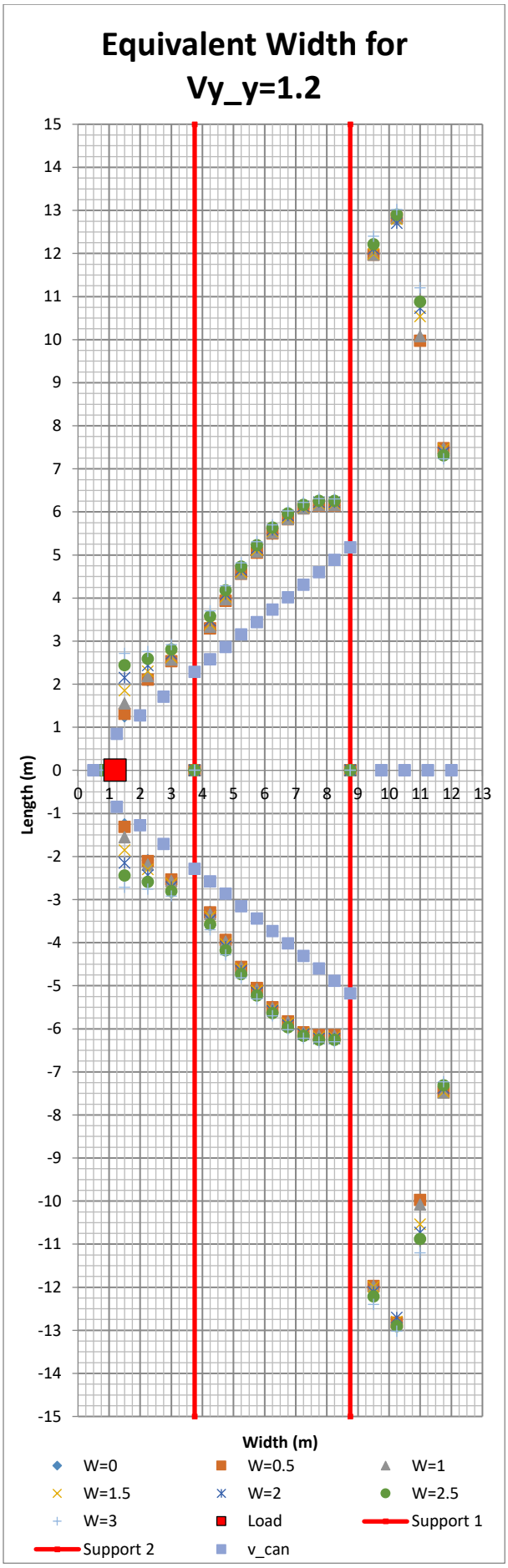

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

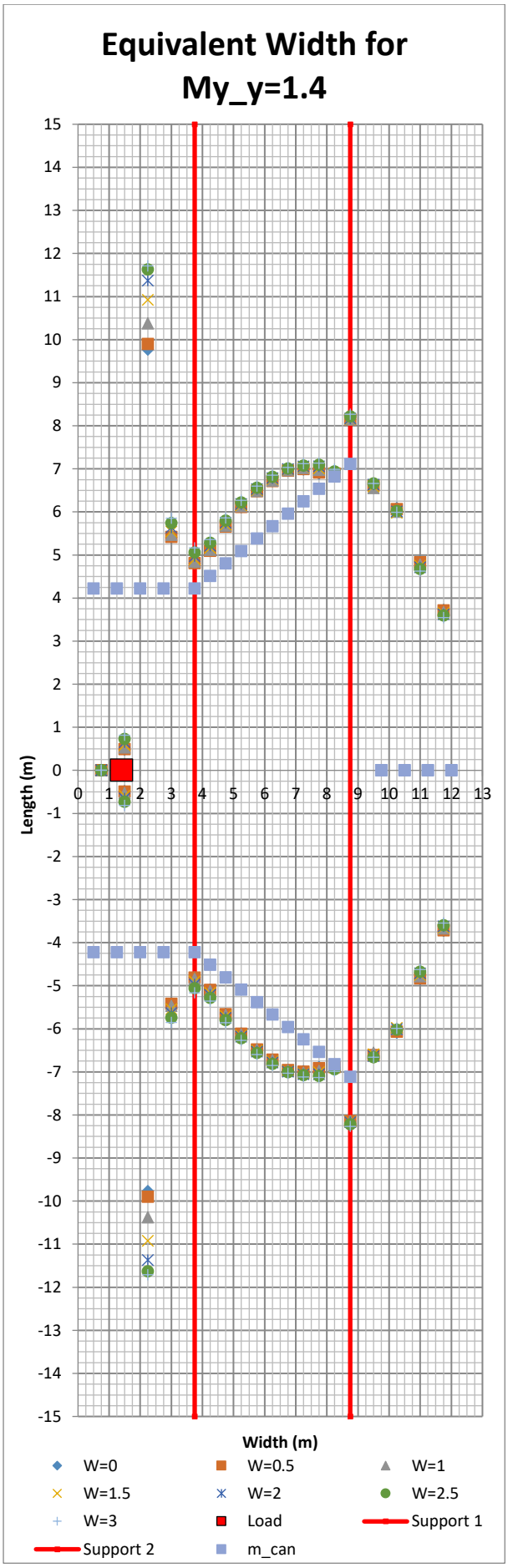

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

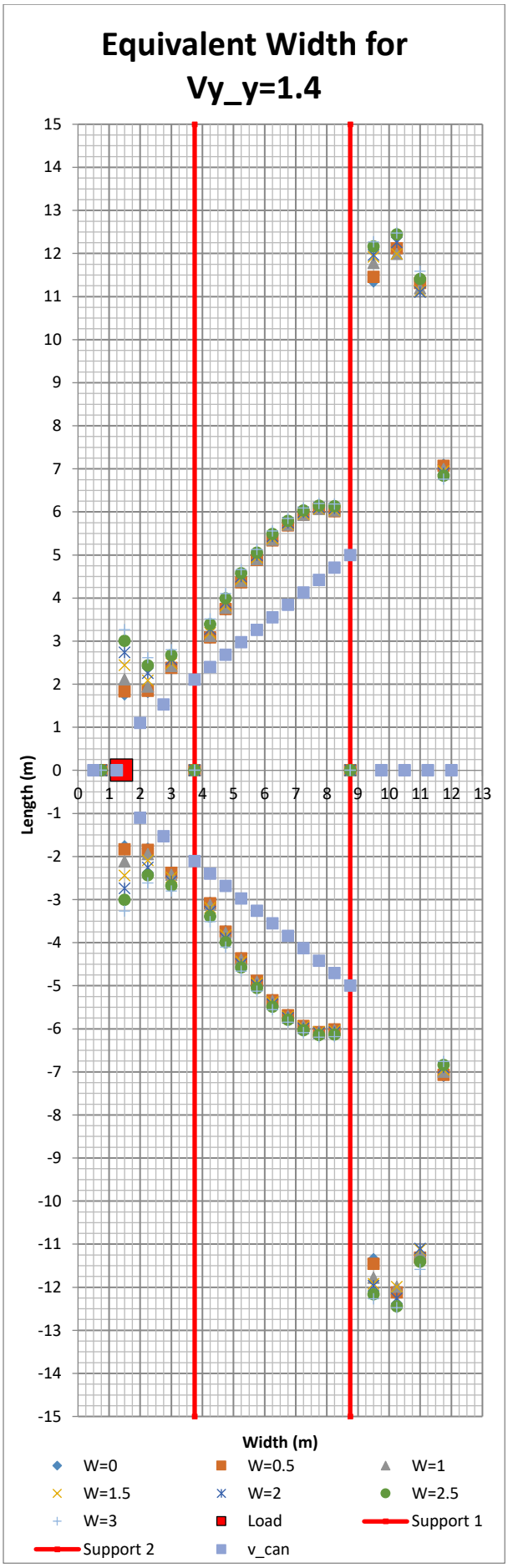

Equivalent Width for My_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

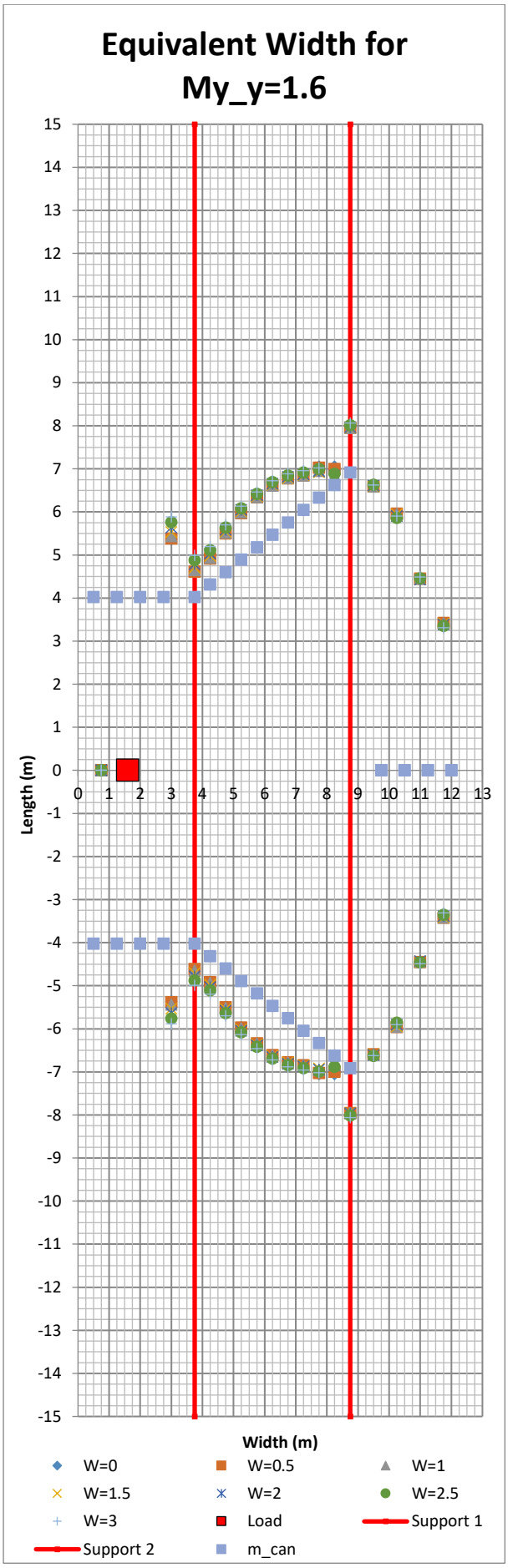

Equivalent Width for Vy_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

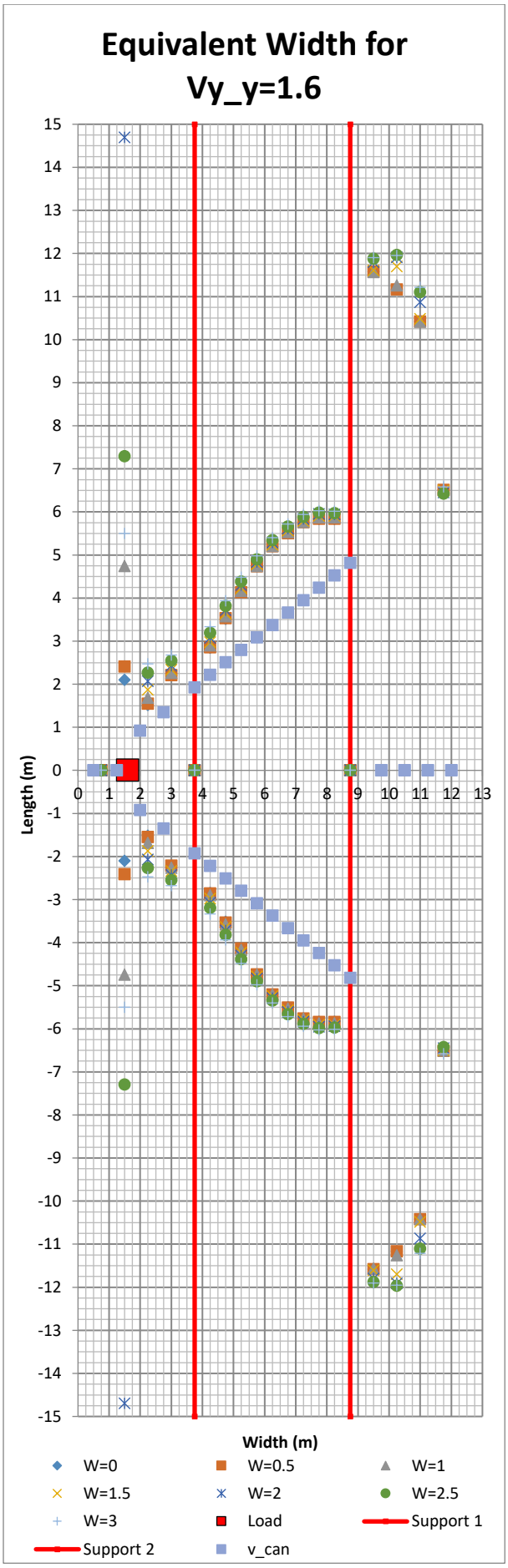

Equivalent Width for My_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

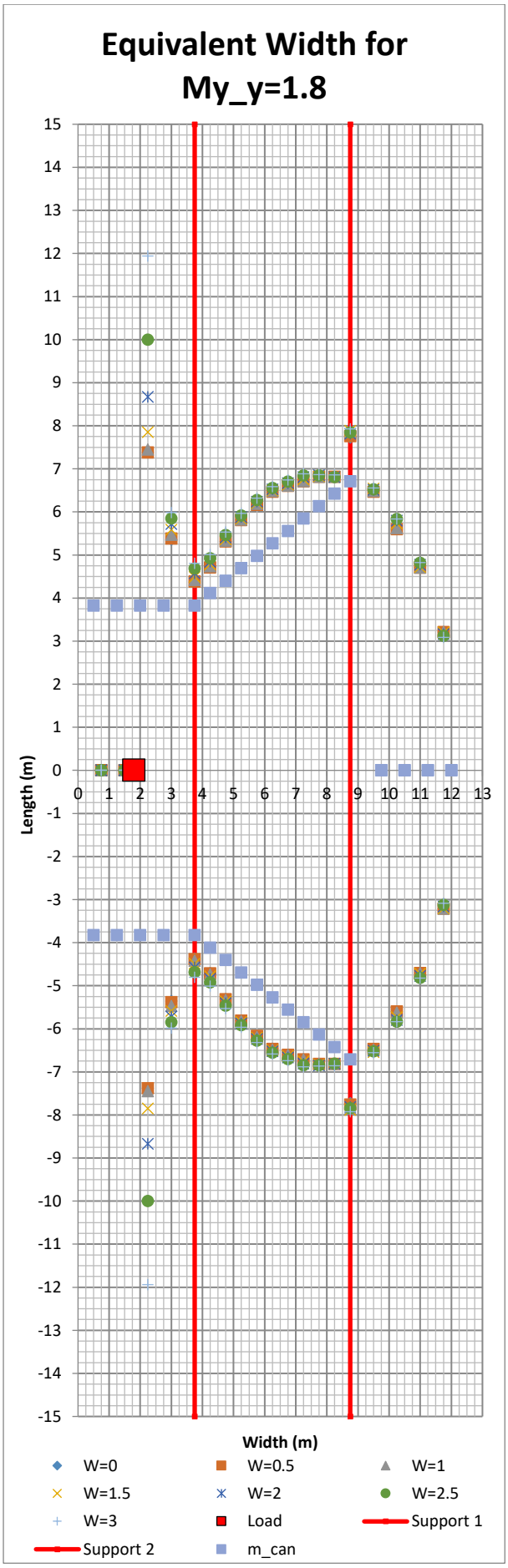

Equivalent Width for Vy_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

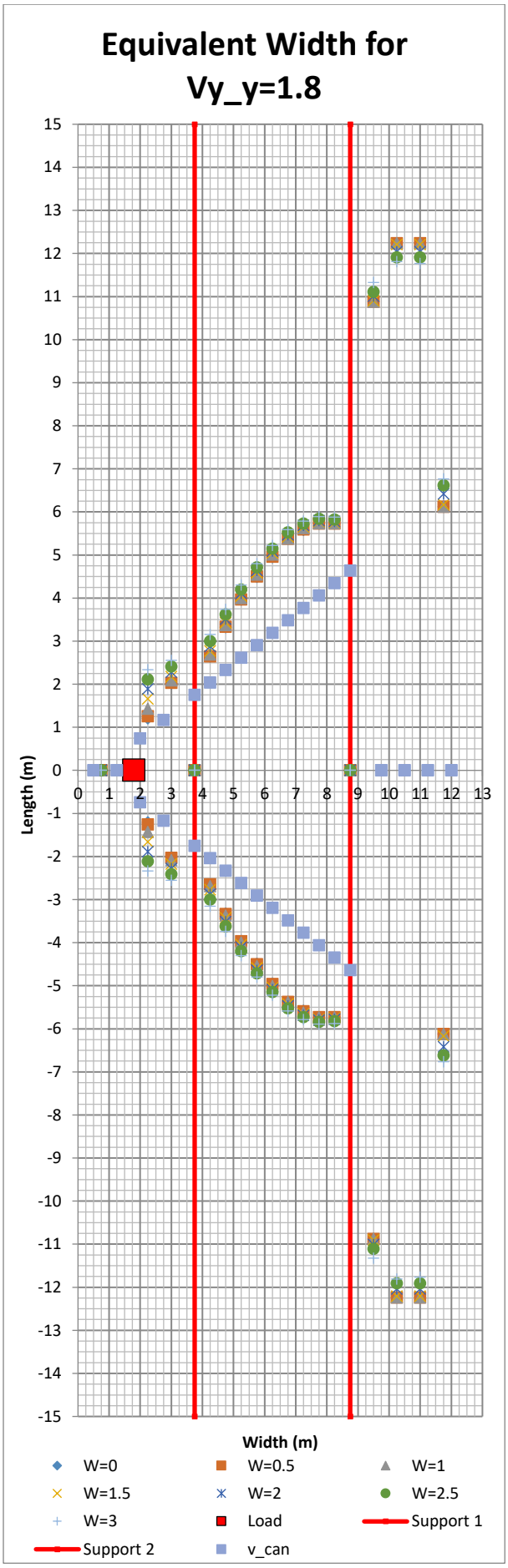

Equivalent Width for My_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

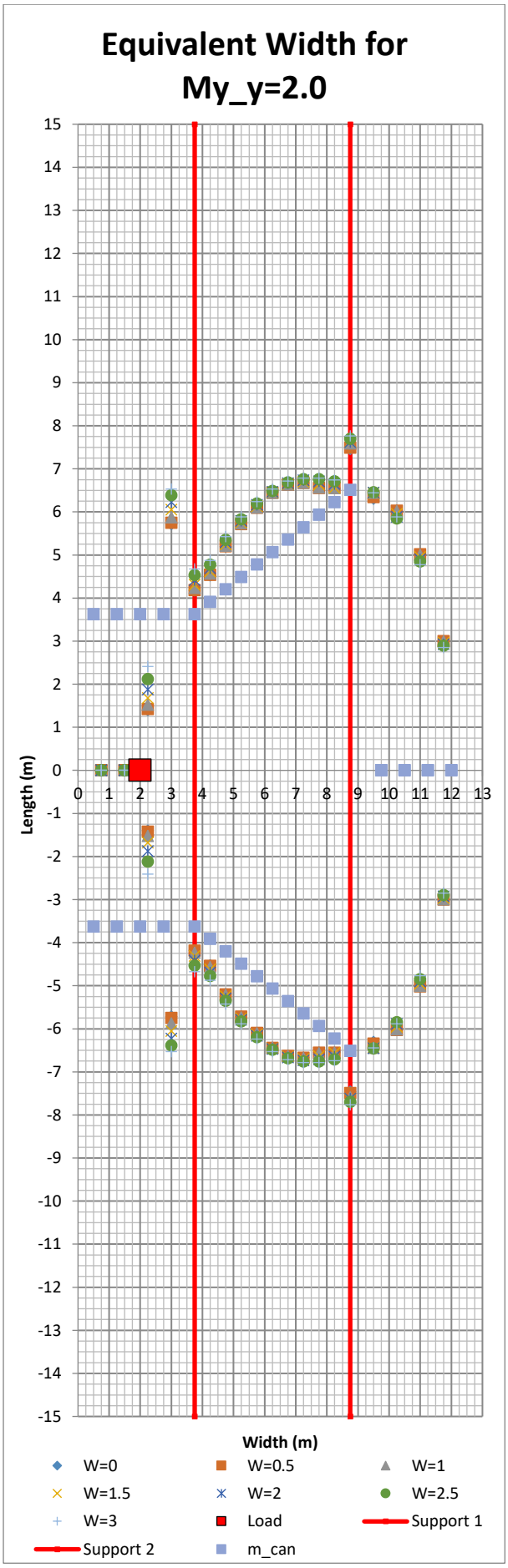

Equivalent Width for Vy_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

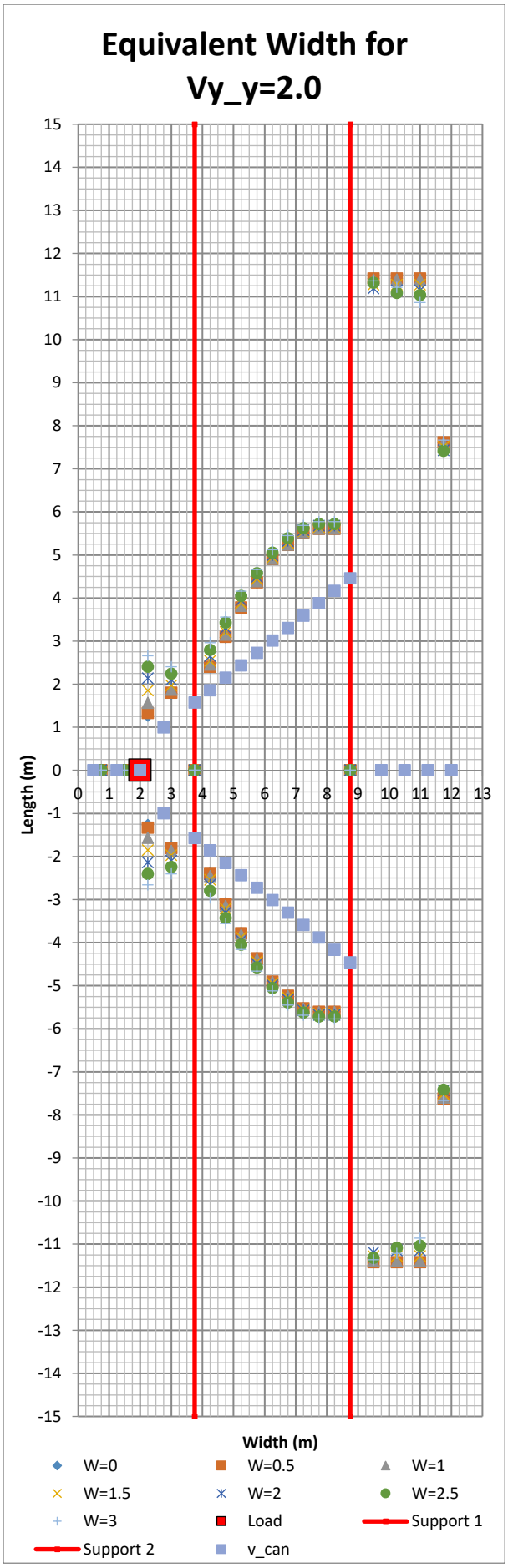

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

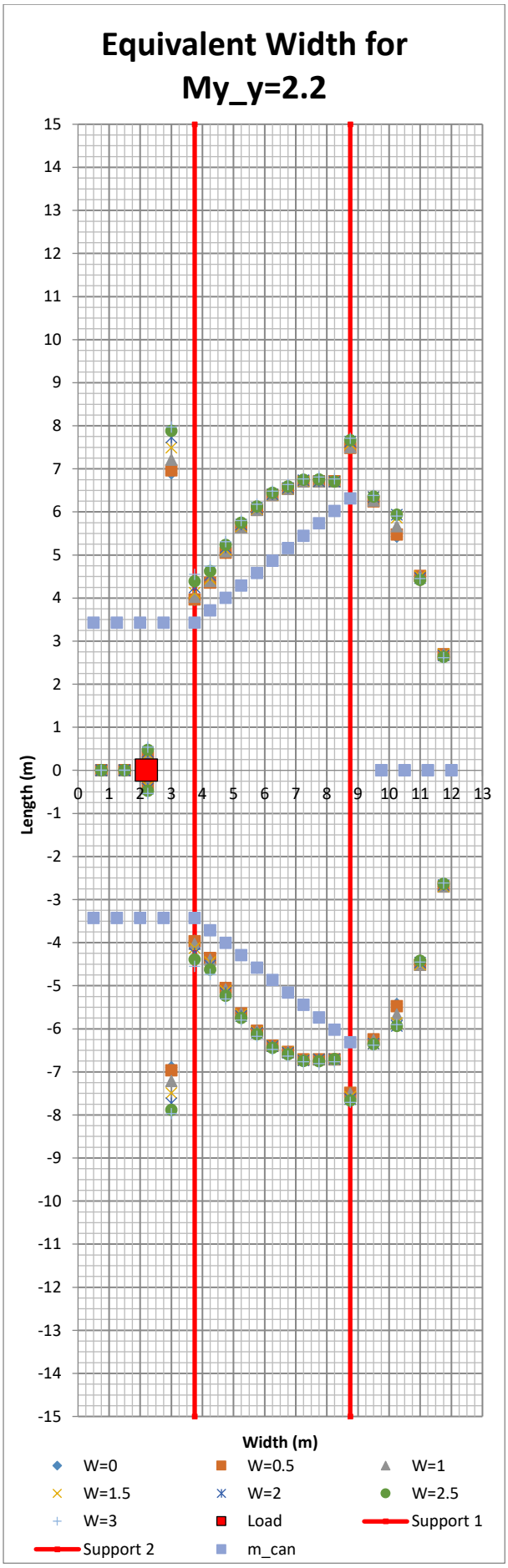

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

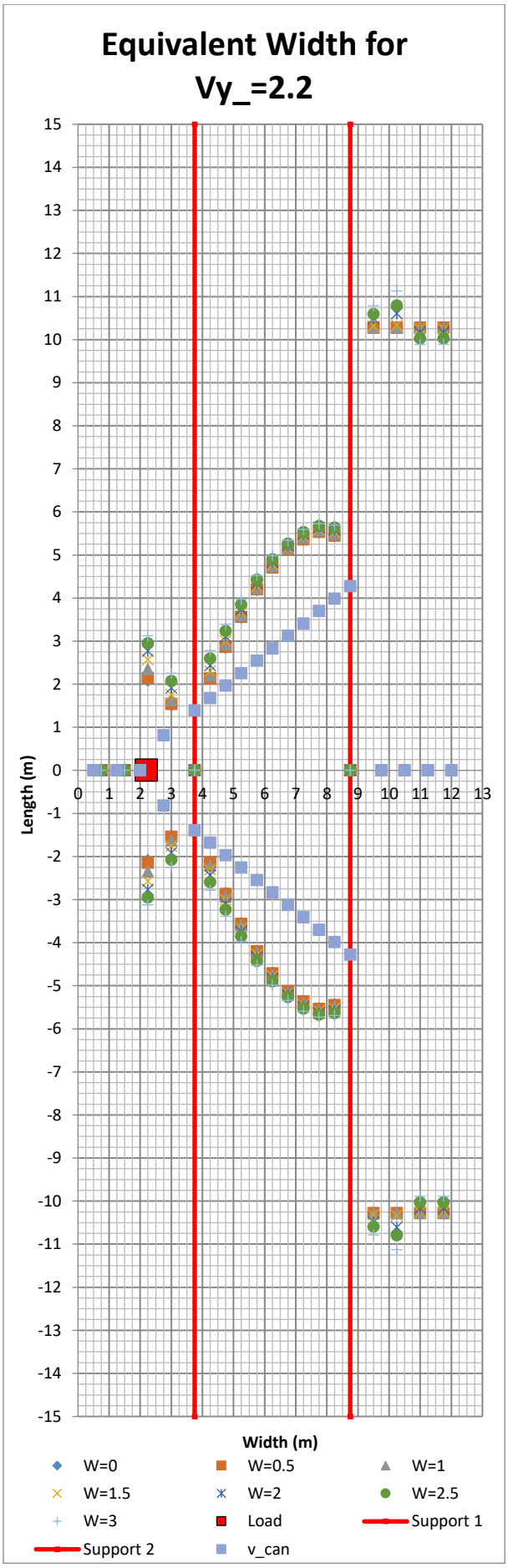

Equivalent Width for My_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

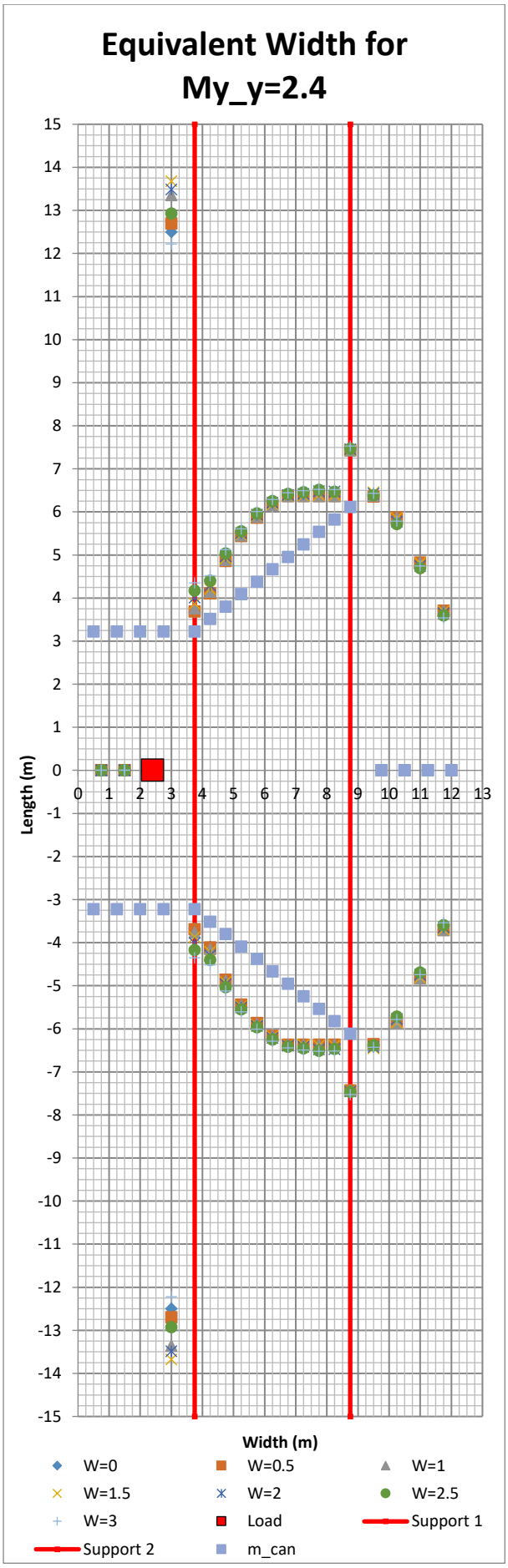

Equivalent Width for Vy_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

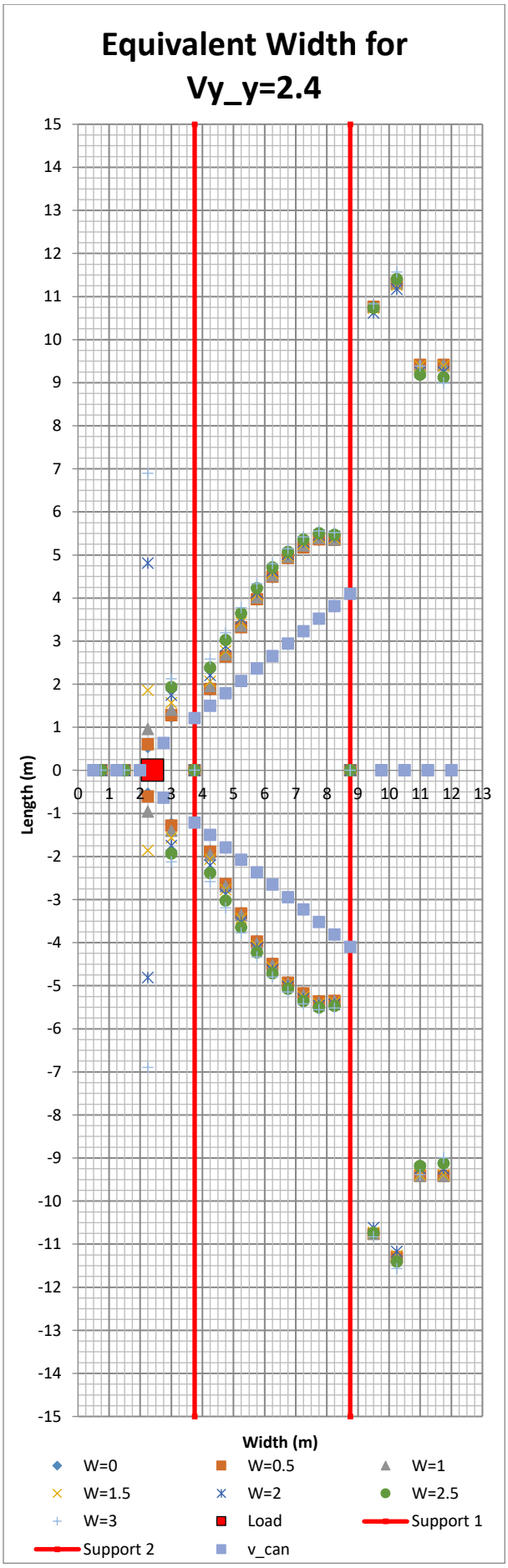

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

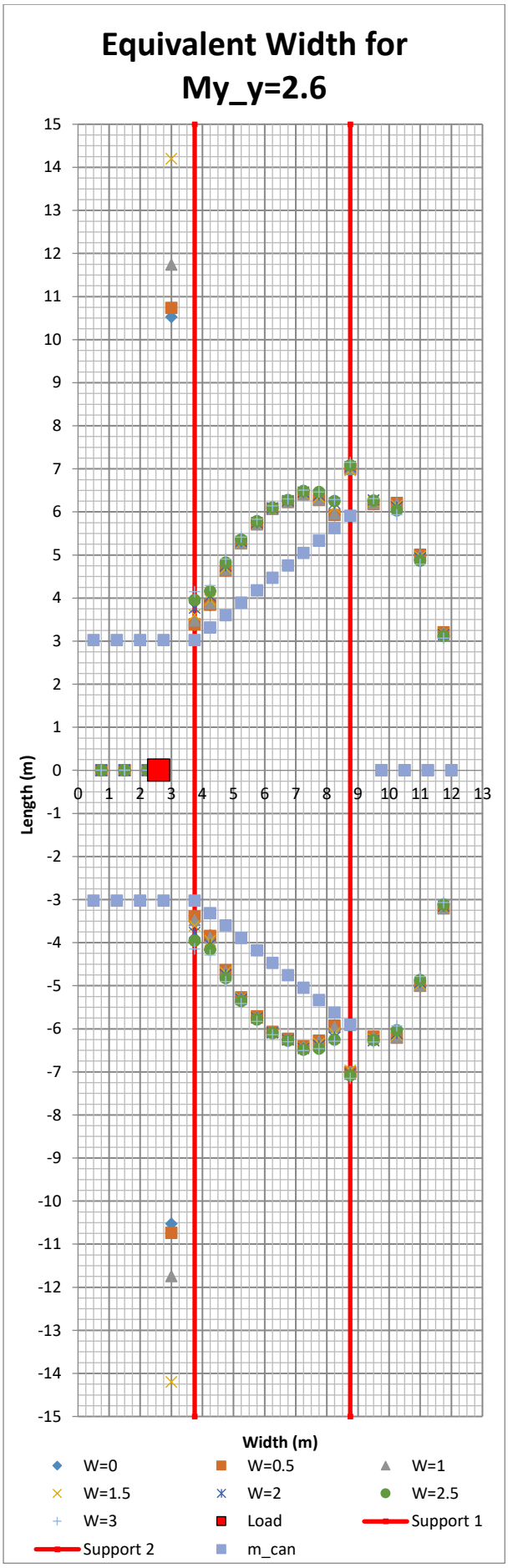

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

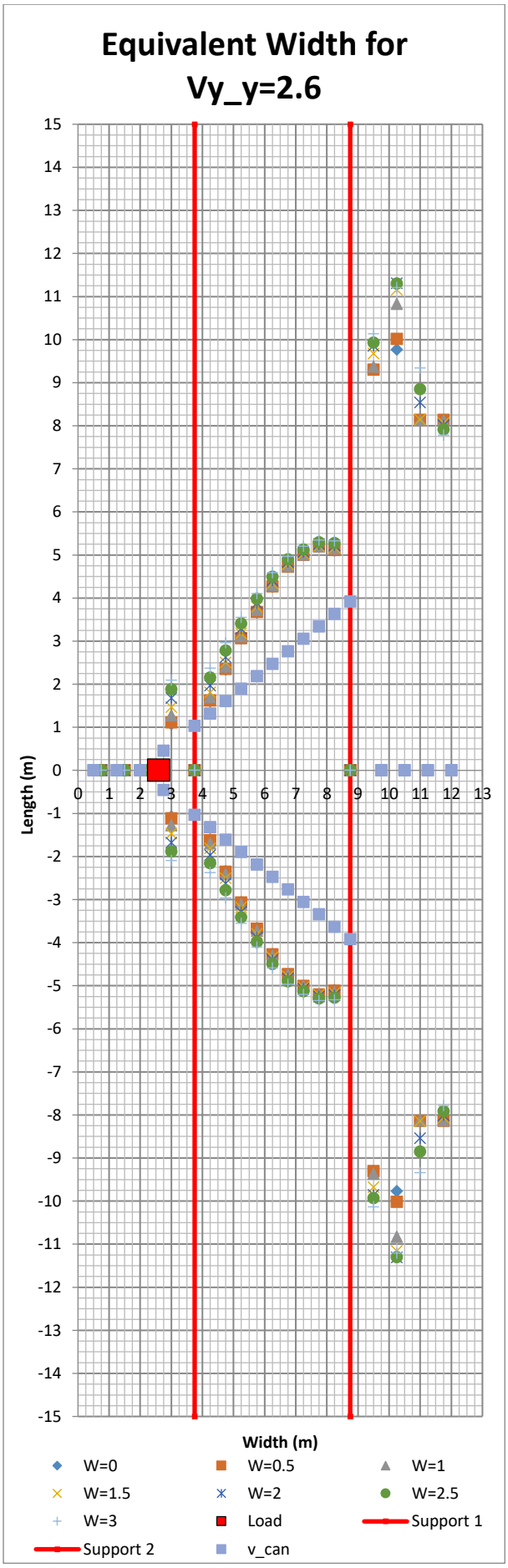

Equivalent Width for My_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

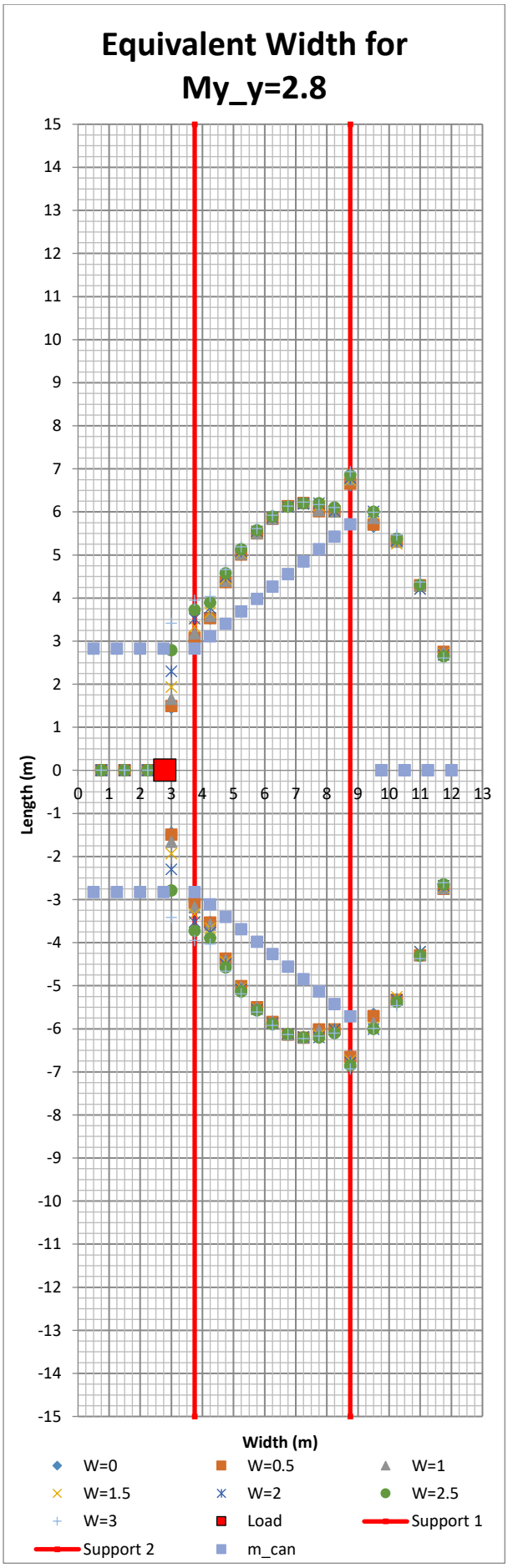

Equivalent Width for Vy_=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

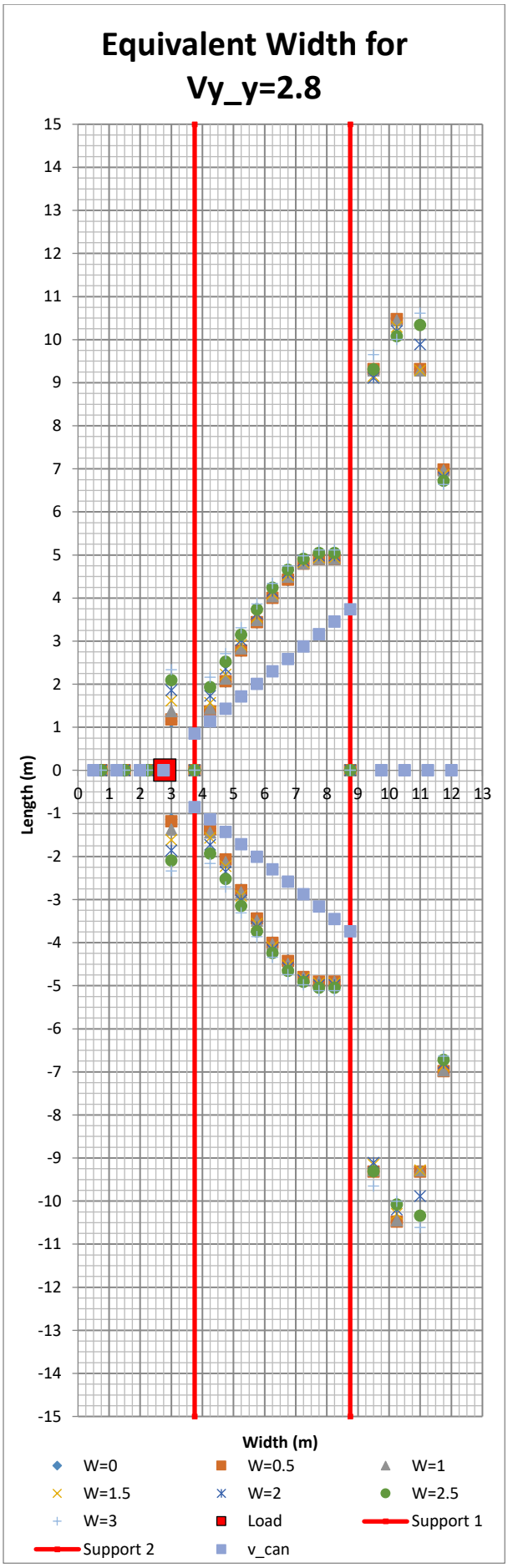

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

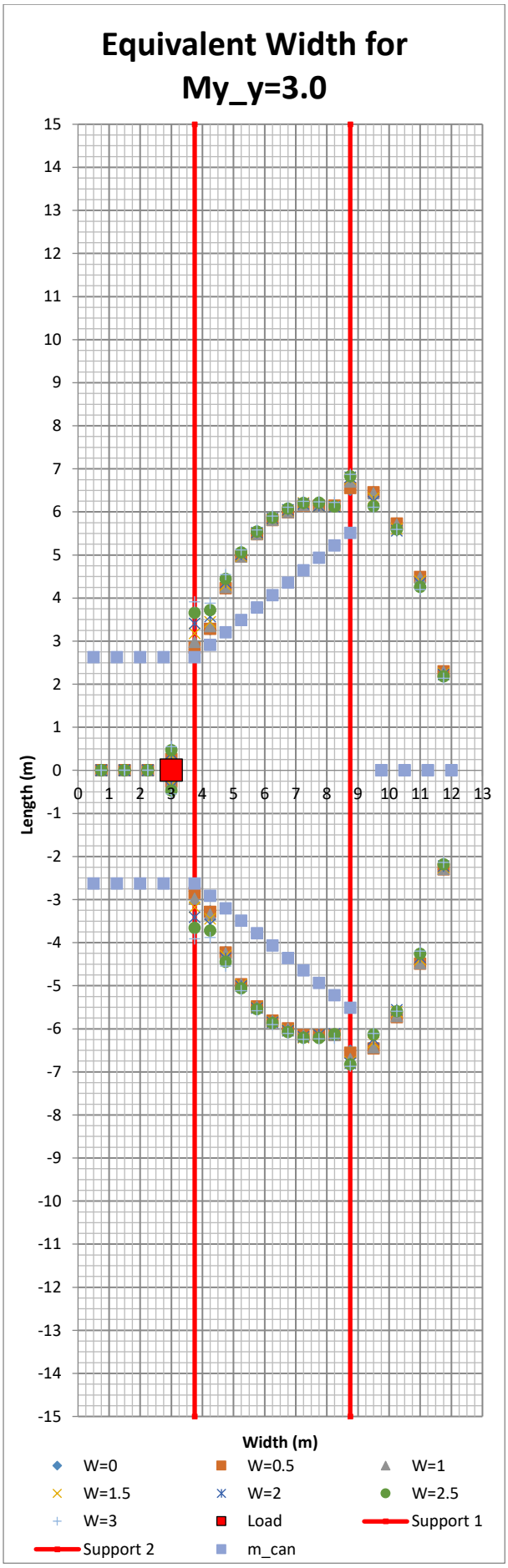

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

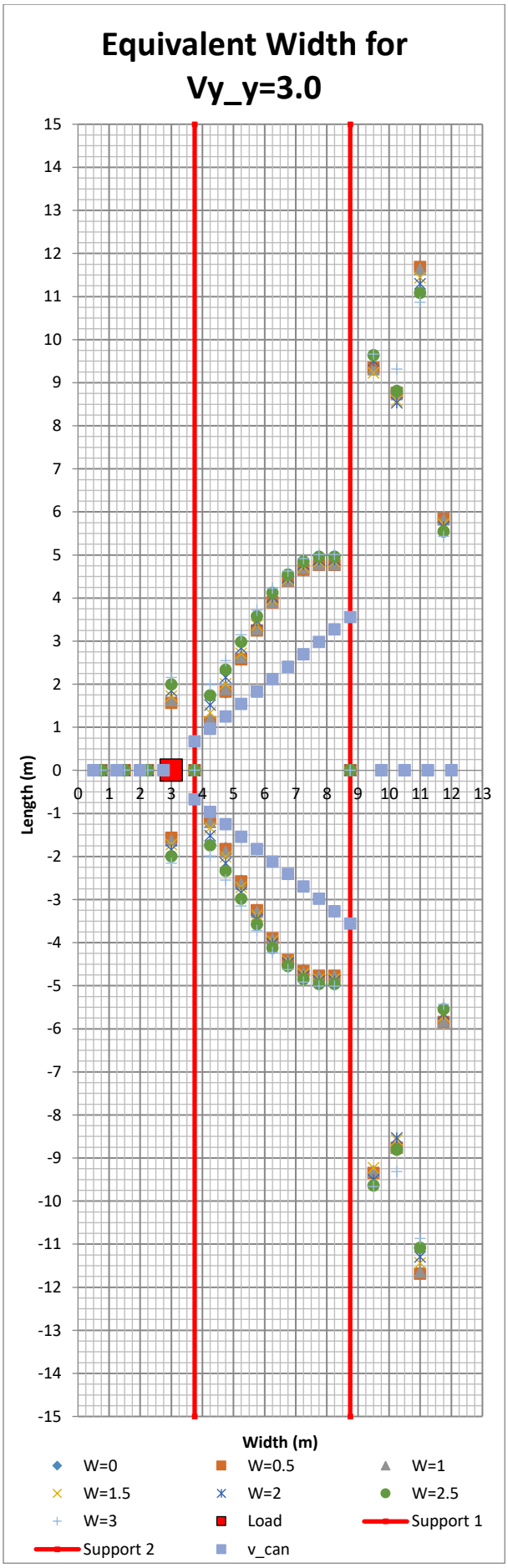

Equivalent Width for My_y=2.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

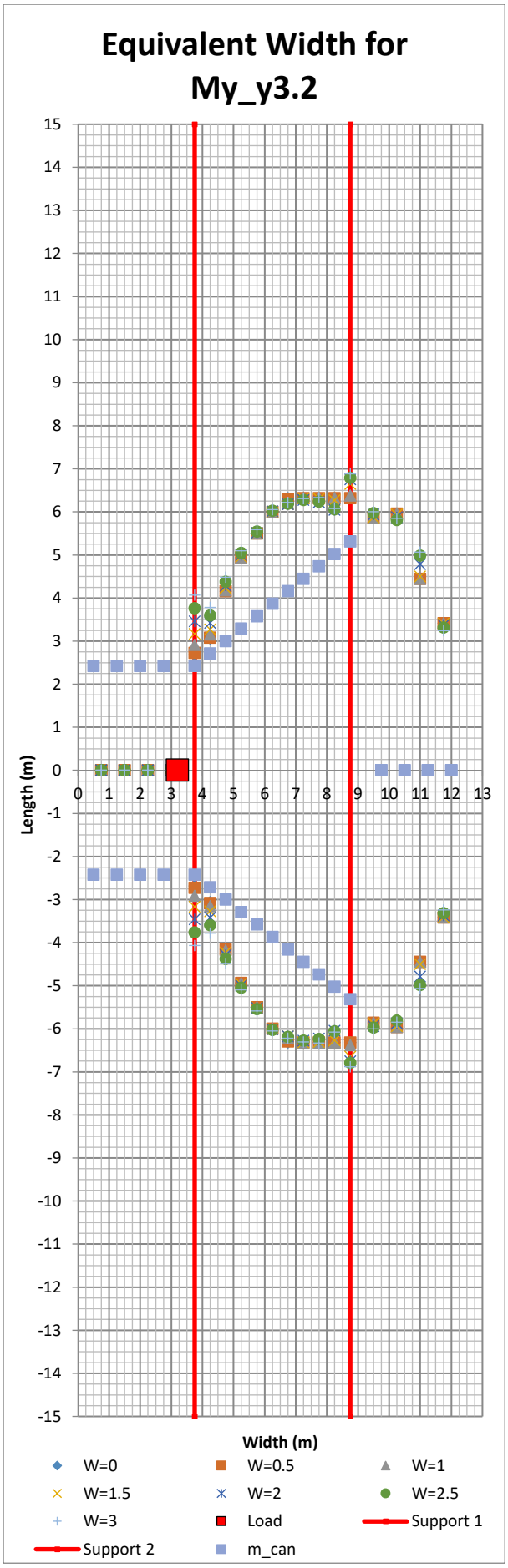

Equivalent Width for Vy_y=2.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

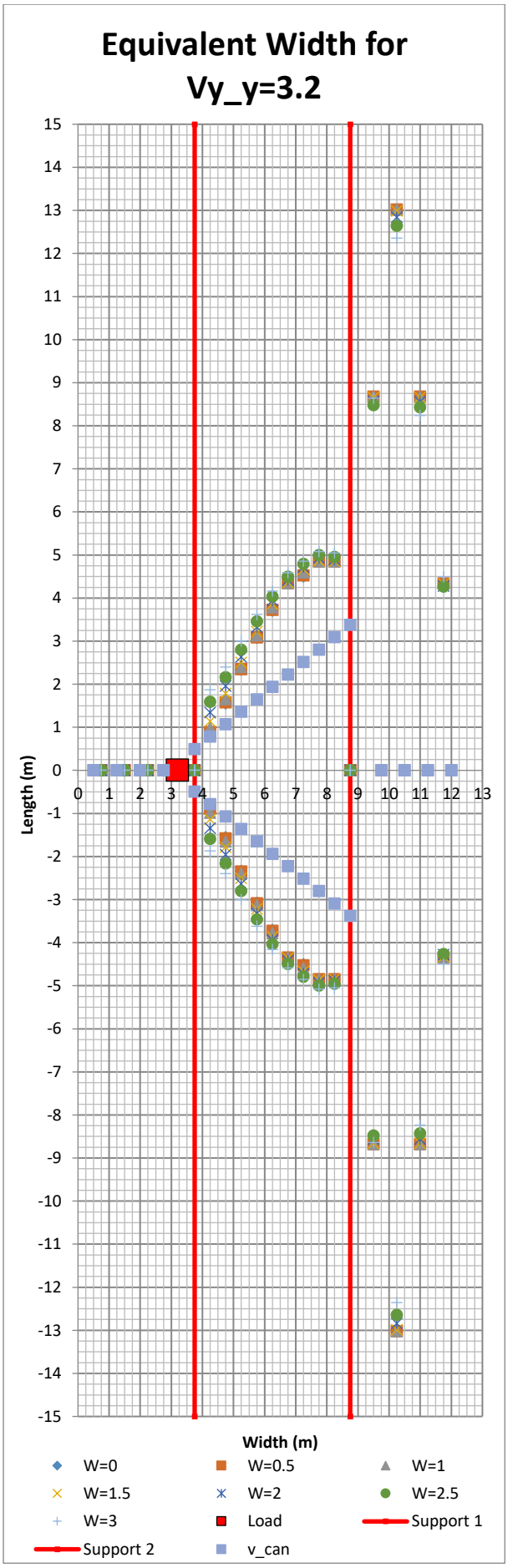

Equivalent Width for My_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

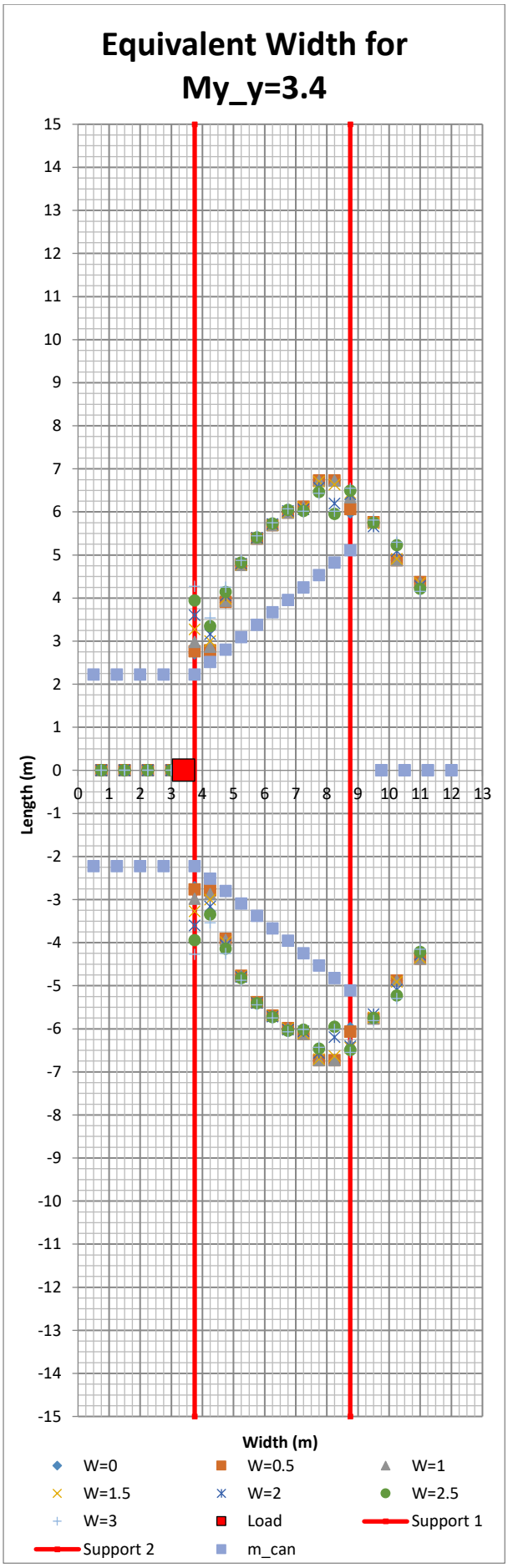

Equivalent Width for Vy_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

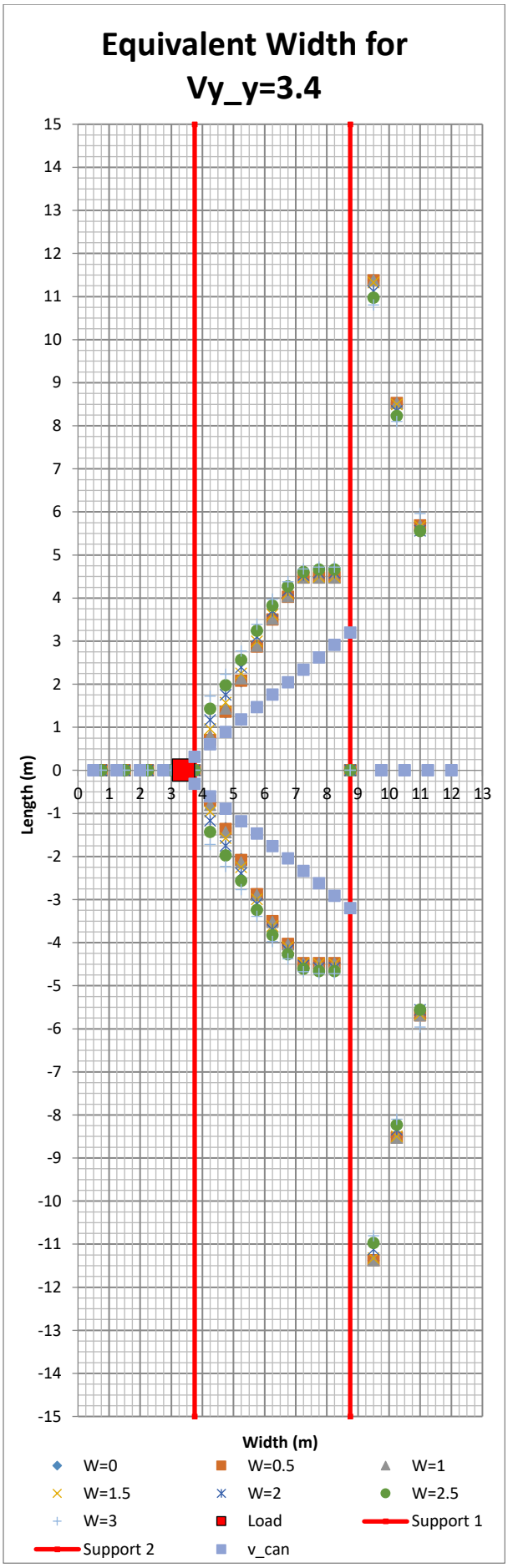

Equivalent Width for My_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

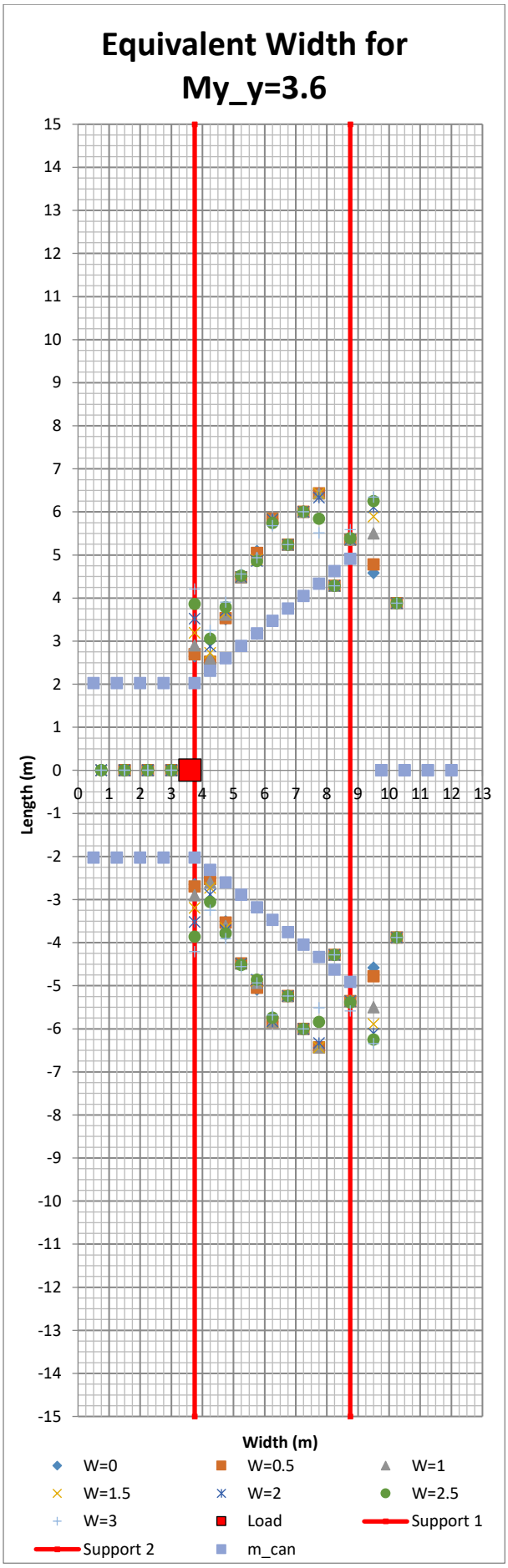

Equivalent Width for Vy_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

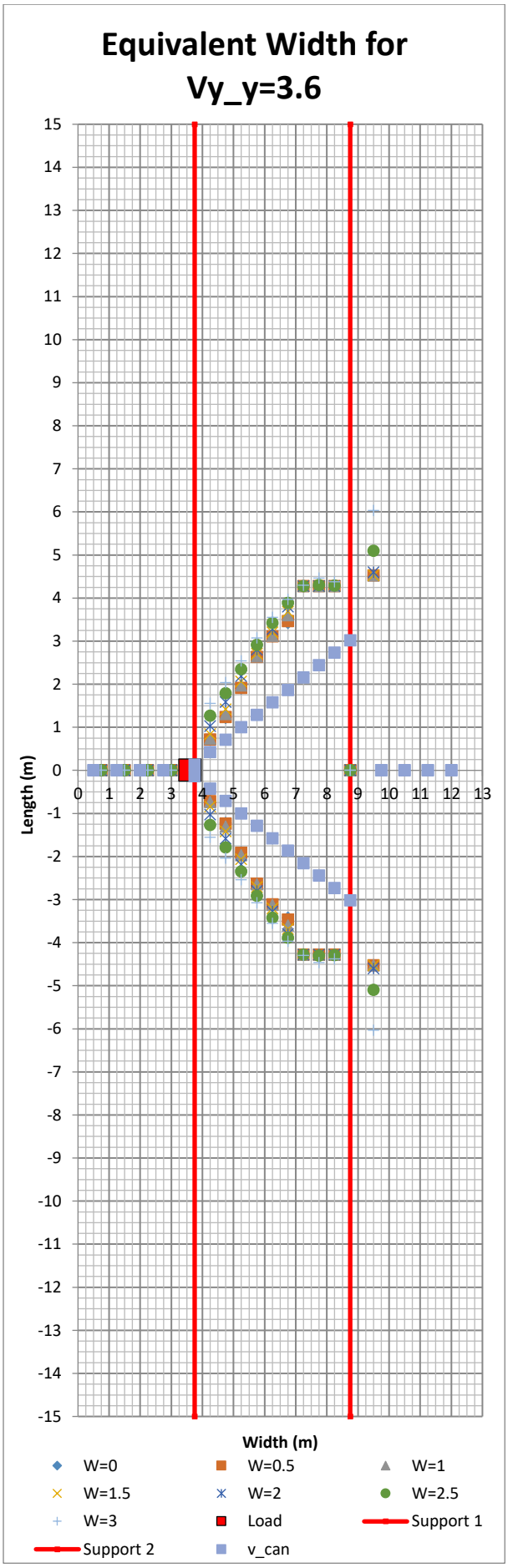

Equivalent Width for My_y3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

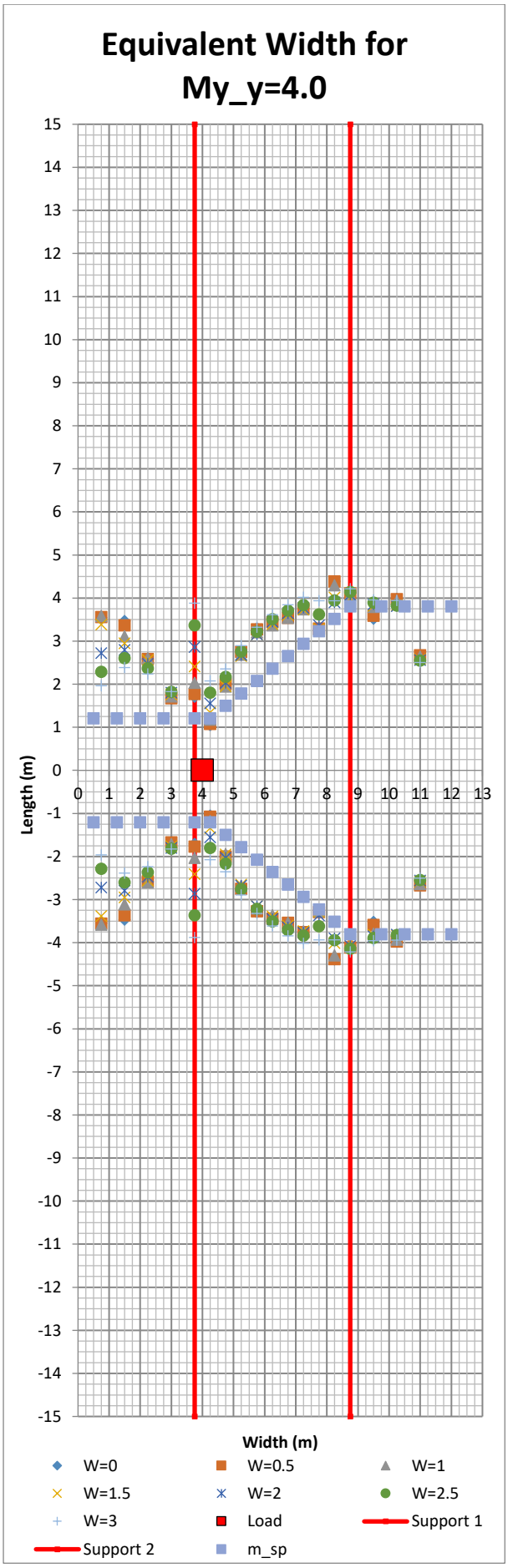

Equivalent Width for Vy_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

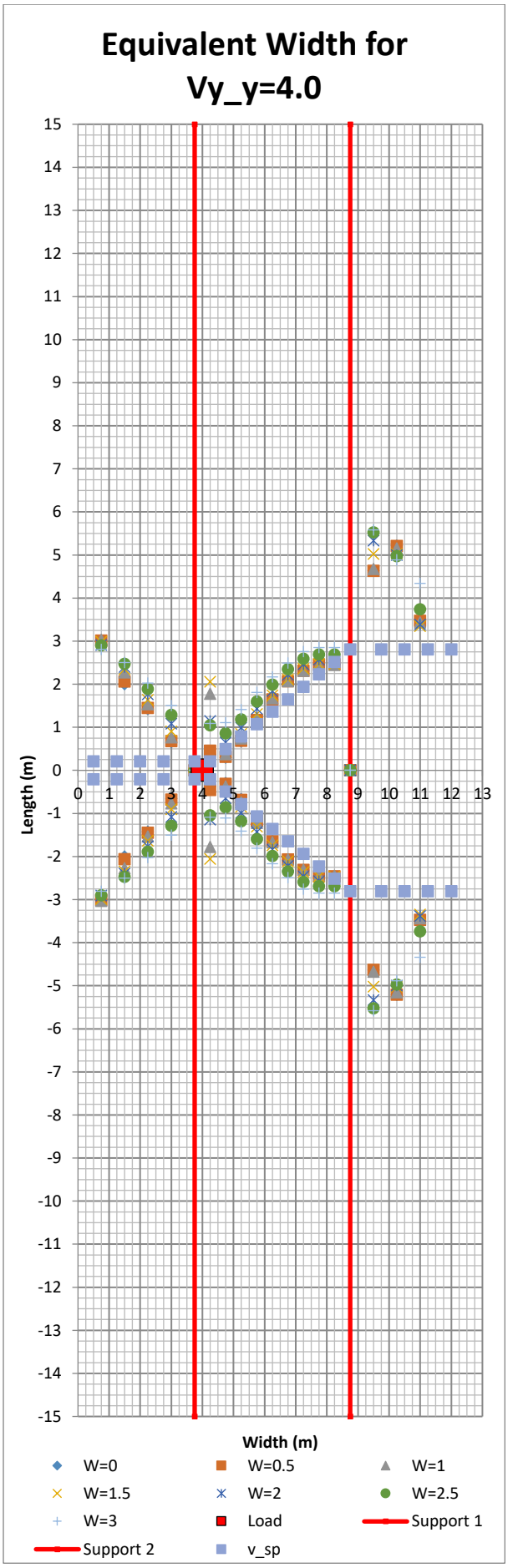

Equivalent Width for My_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

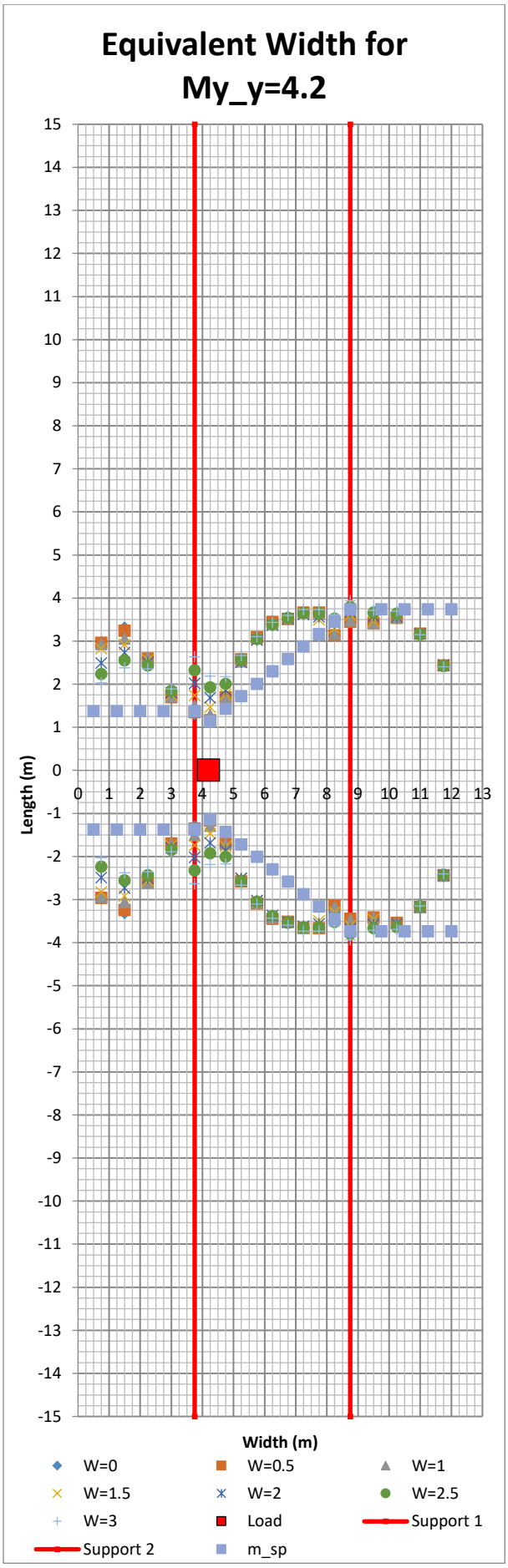

Equivalent Width for Vy_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

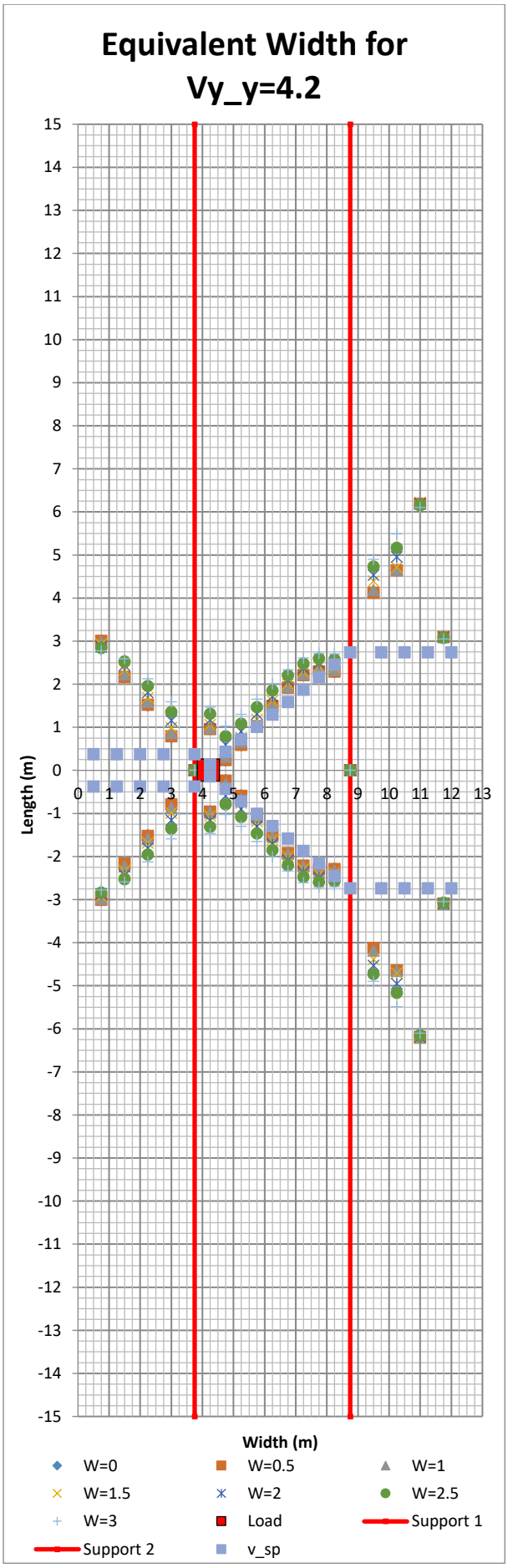

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

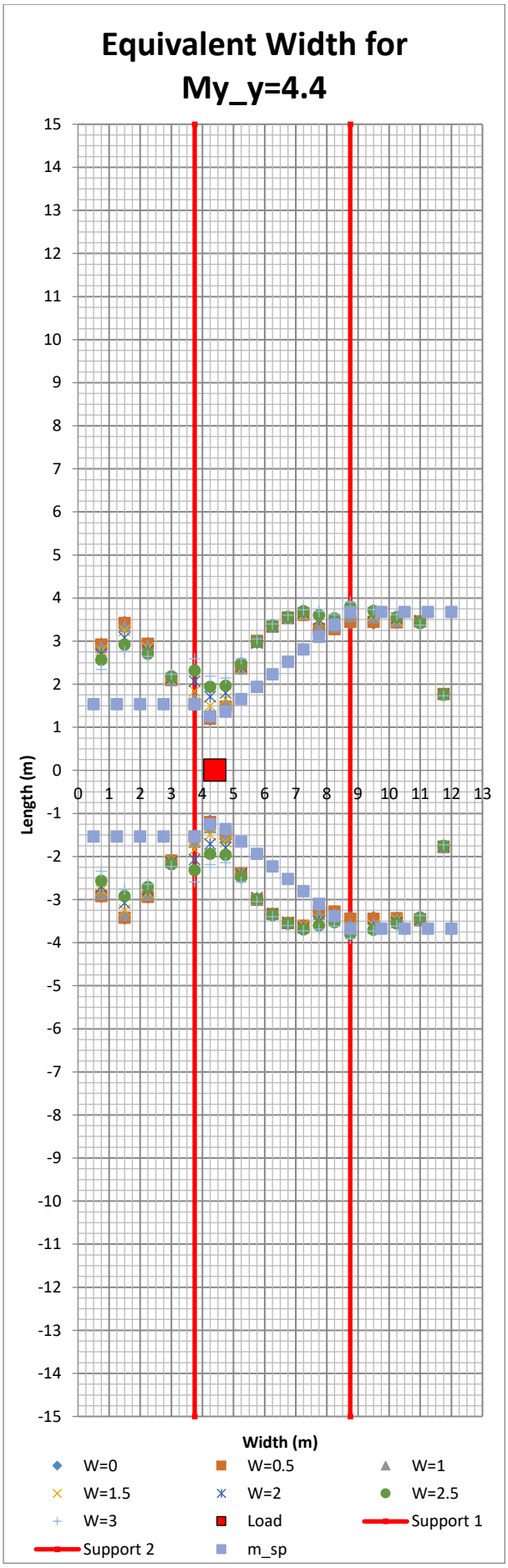

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

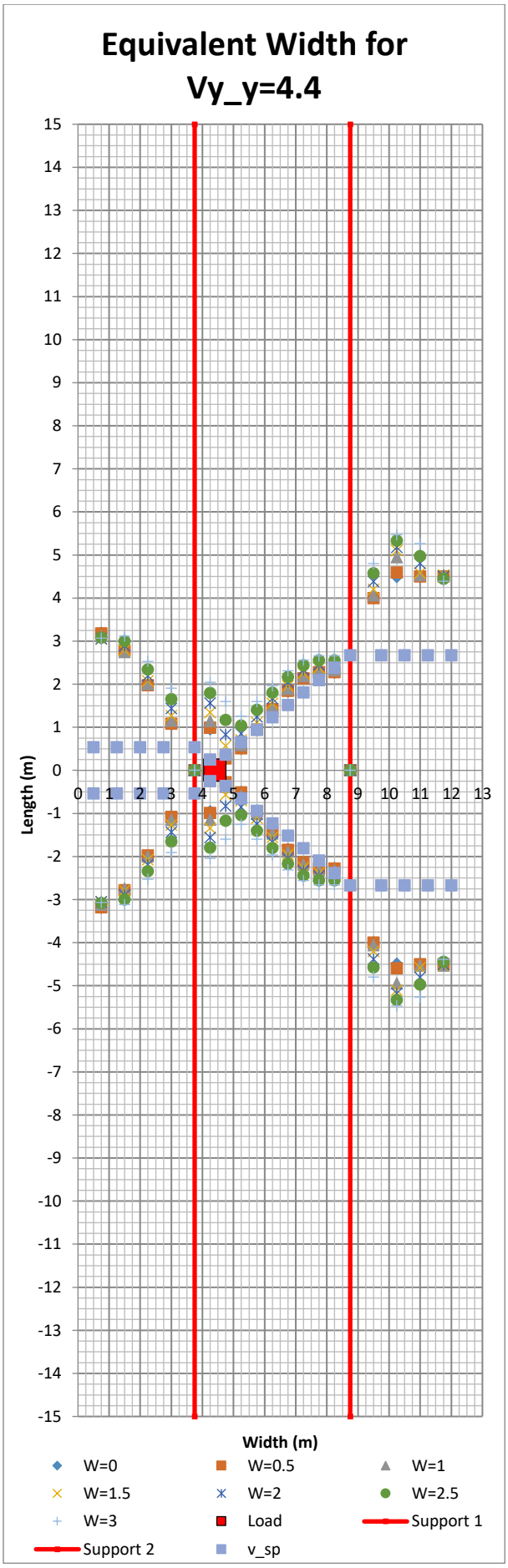

Equivalent Width for My_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

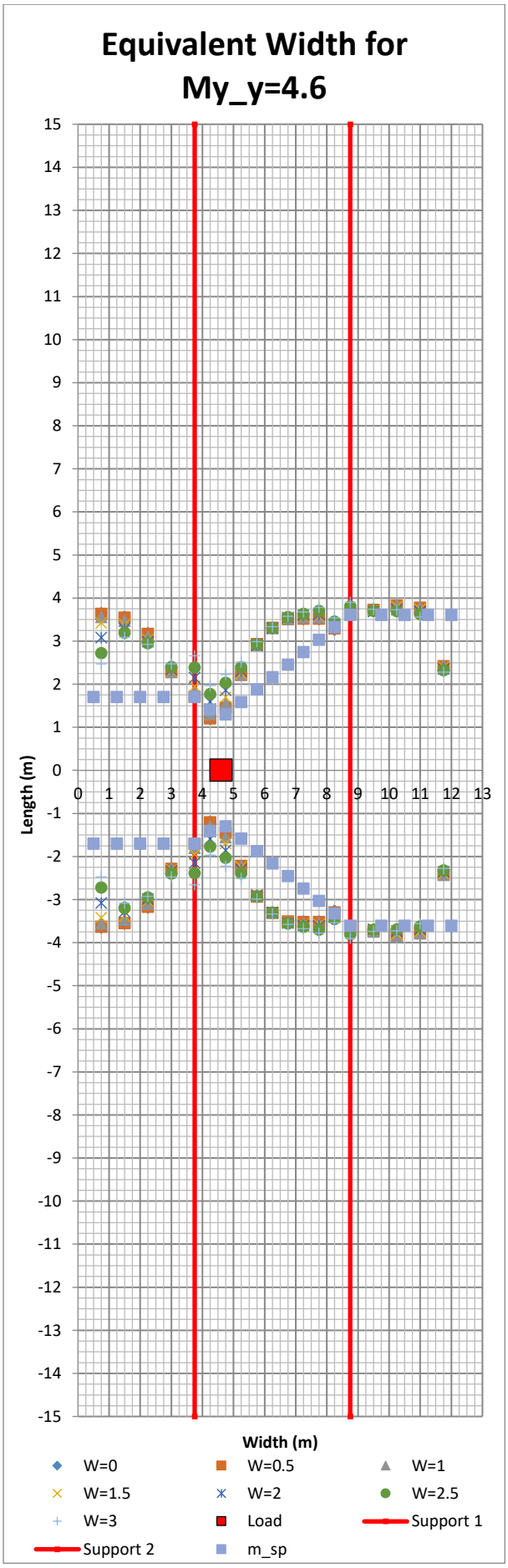

Equivalent Width for Vy_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

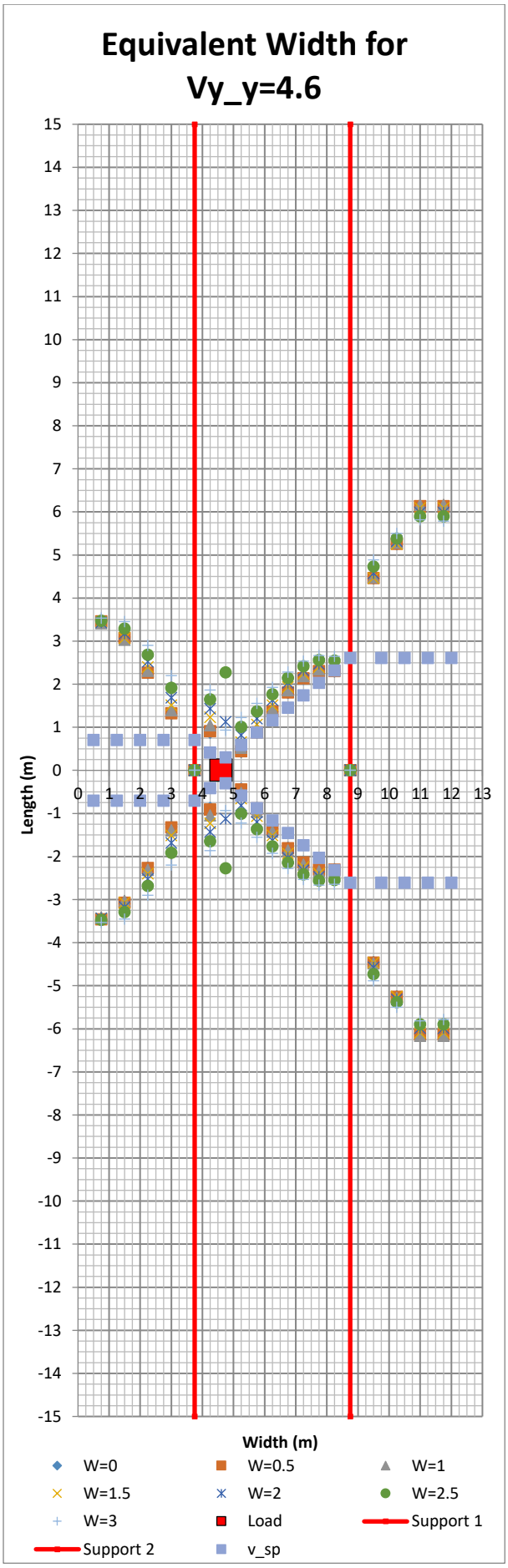

Equivalent Width for My_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

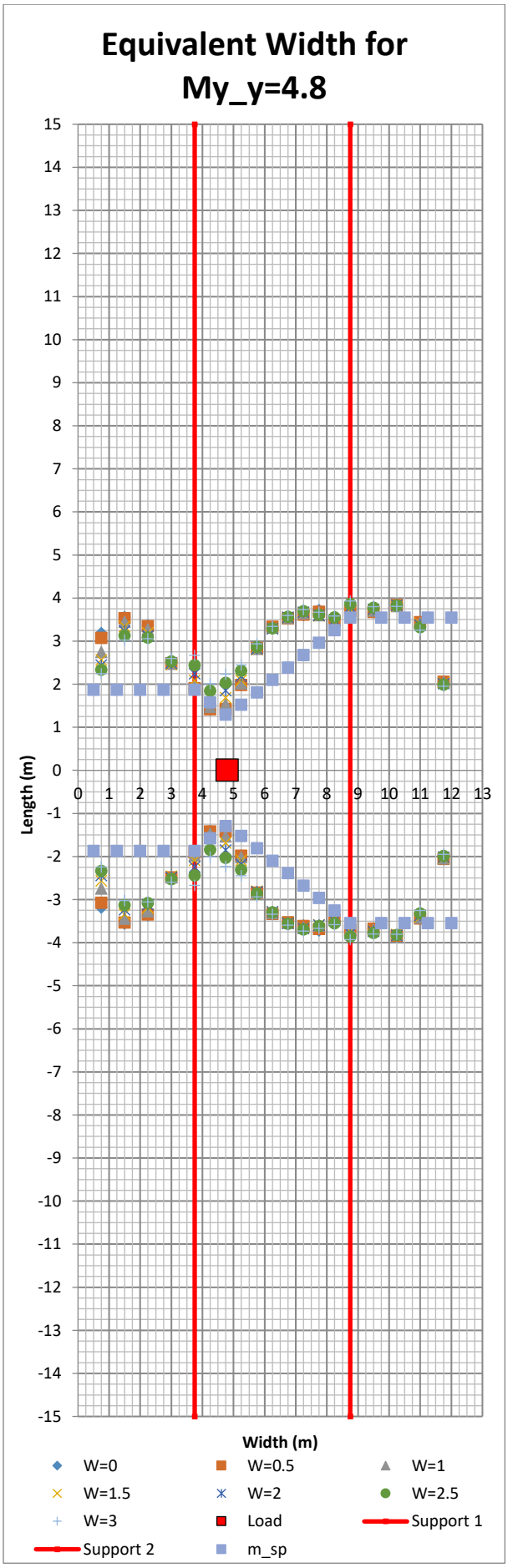

Equivalent Width for Vy_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

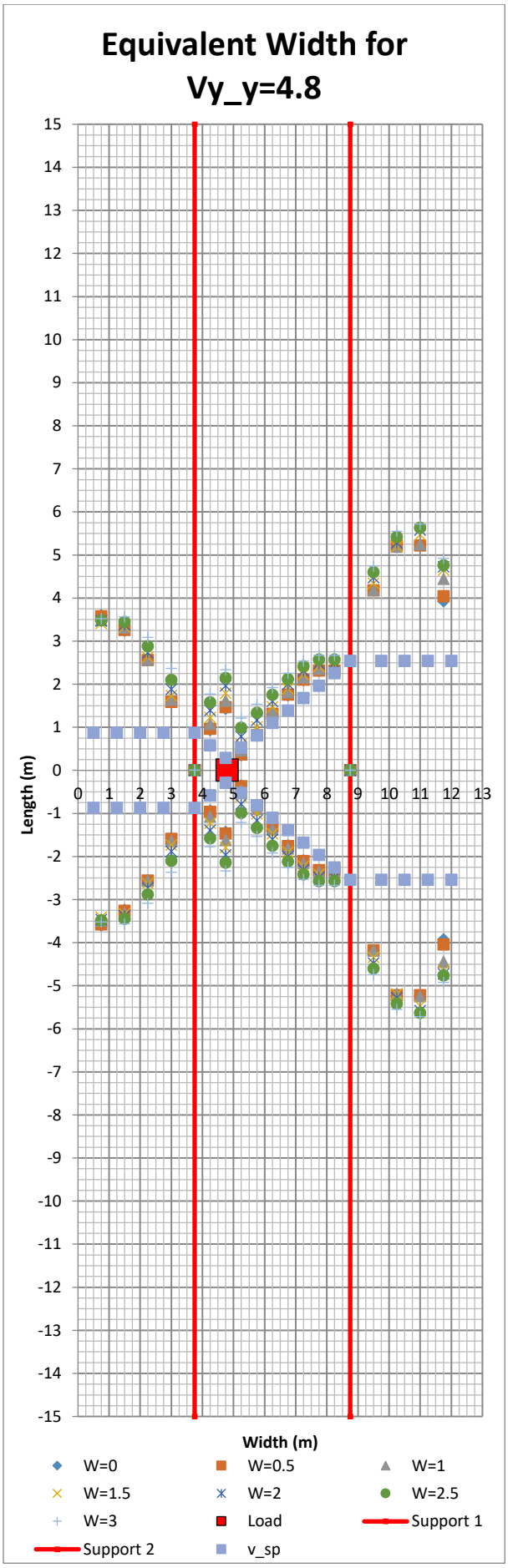

Equivalent Width for My_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

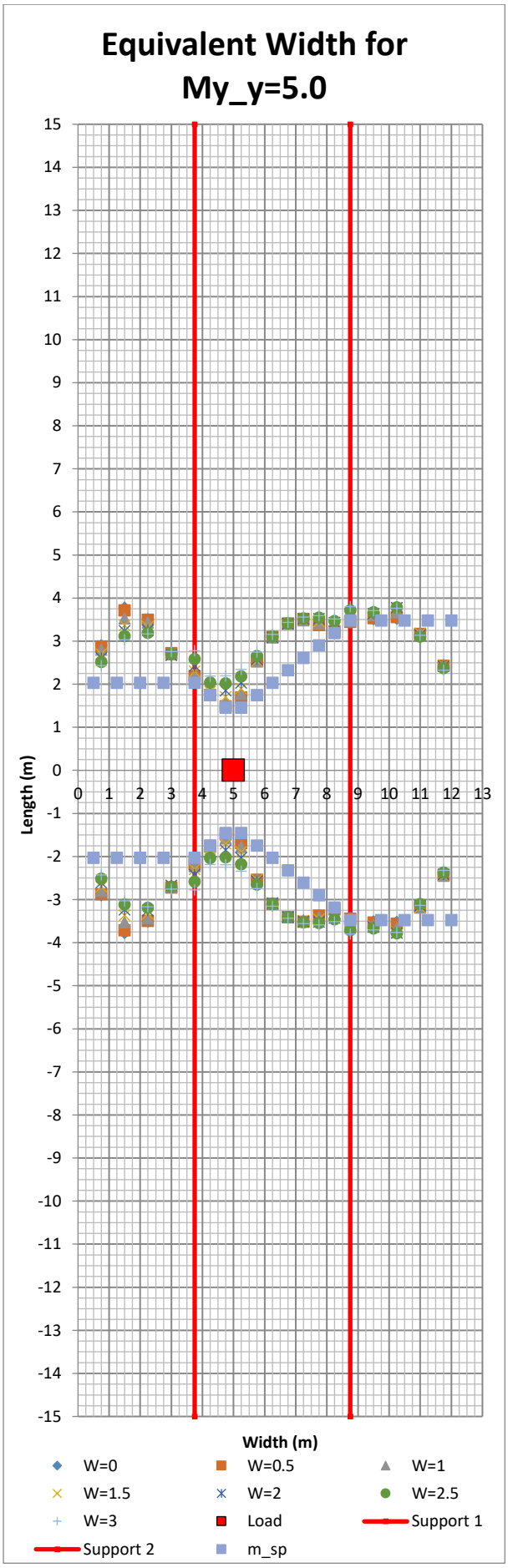

Equivalent Width for Vy_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

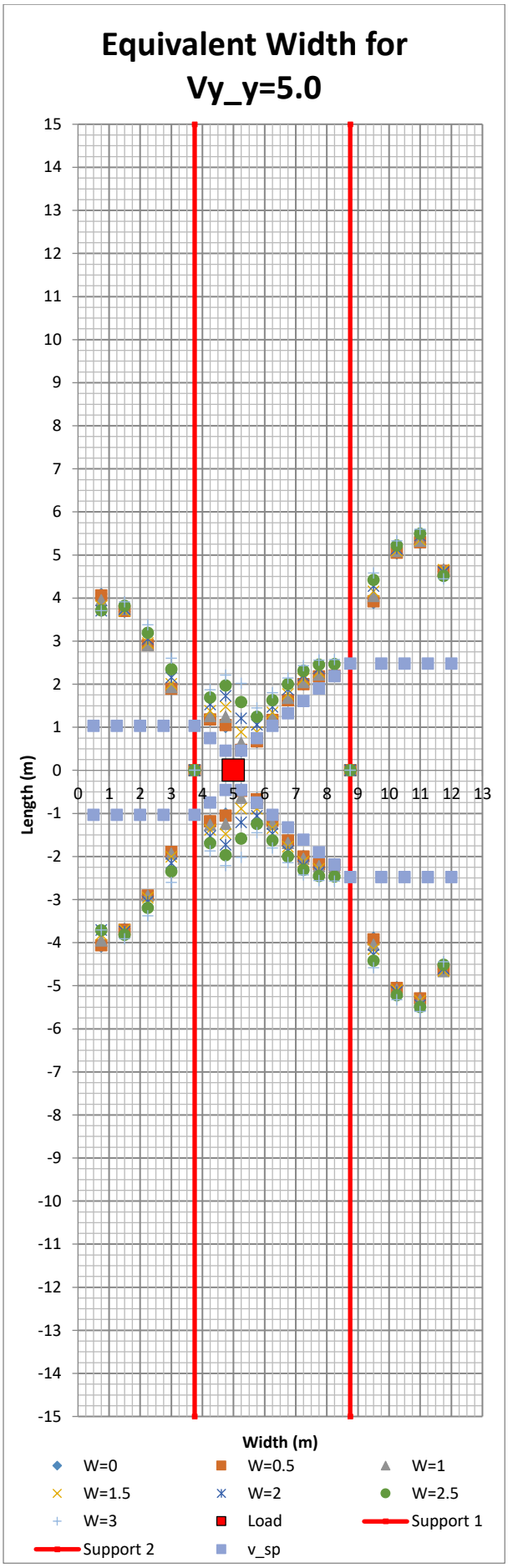

Equivalent Width for My_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

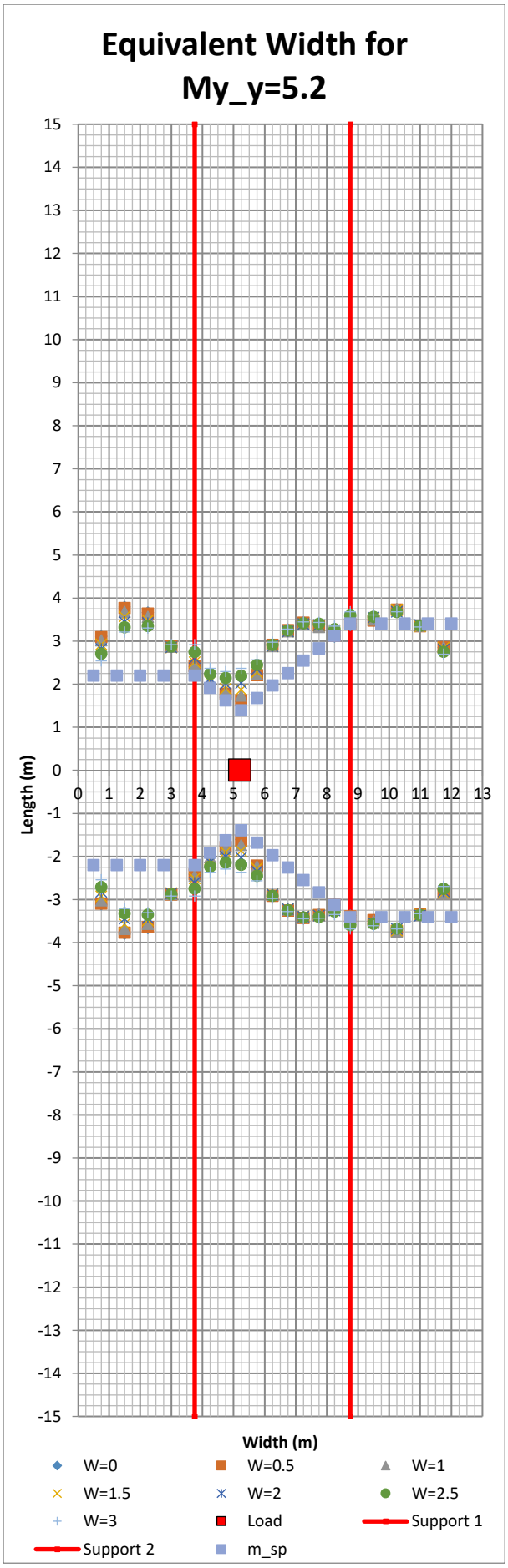

Equivalent Width for Vy_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

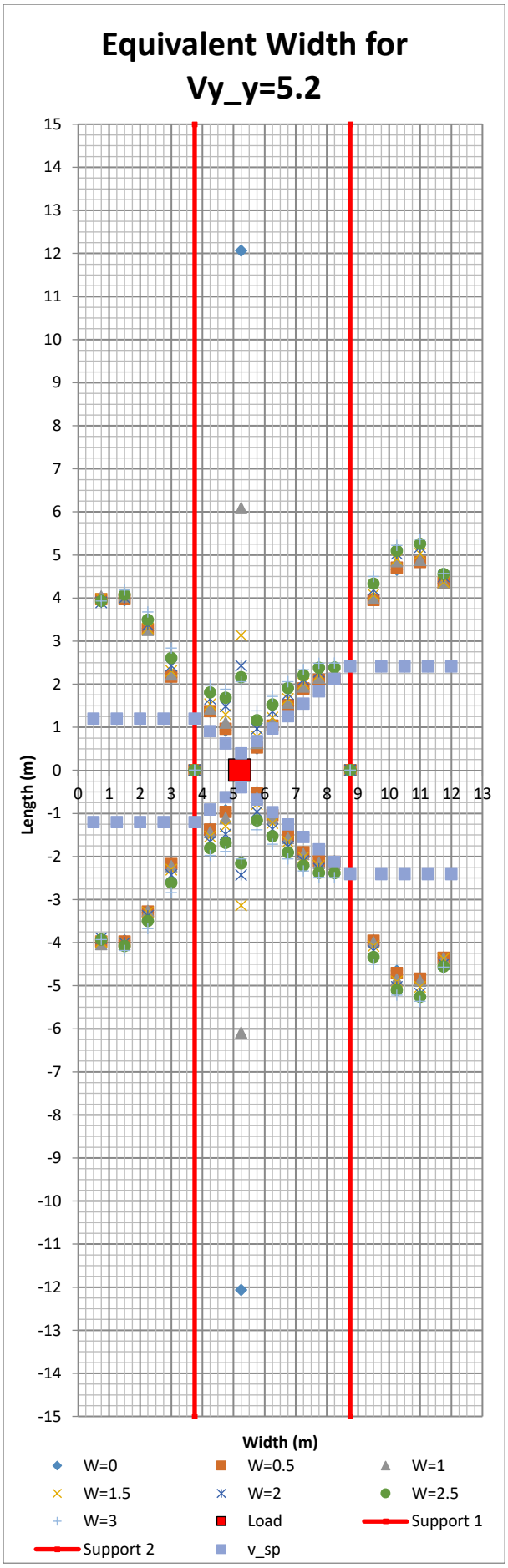

Equivalent Width for My_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

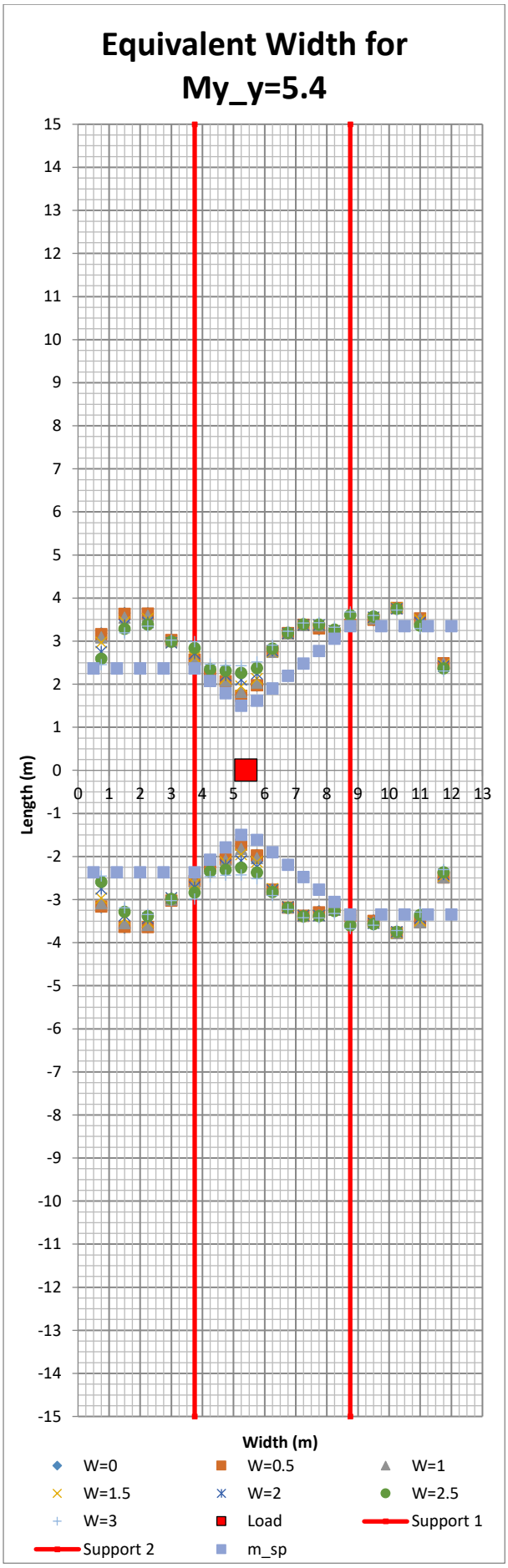

Equivalent Width for Vy_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

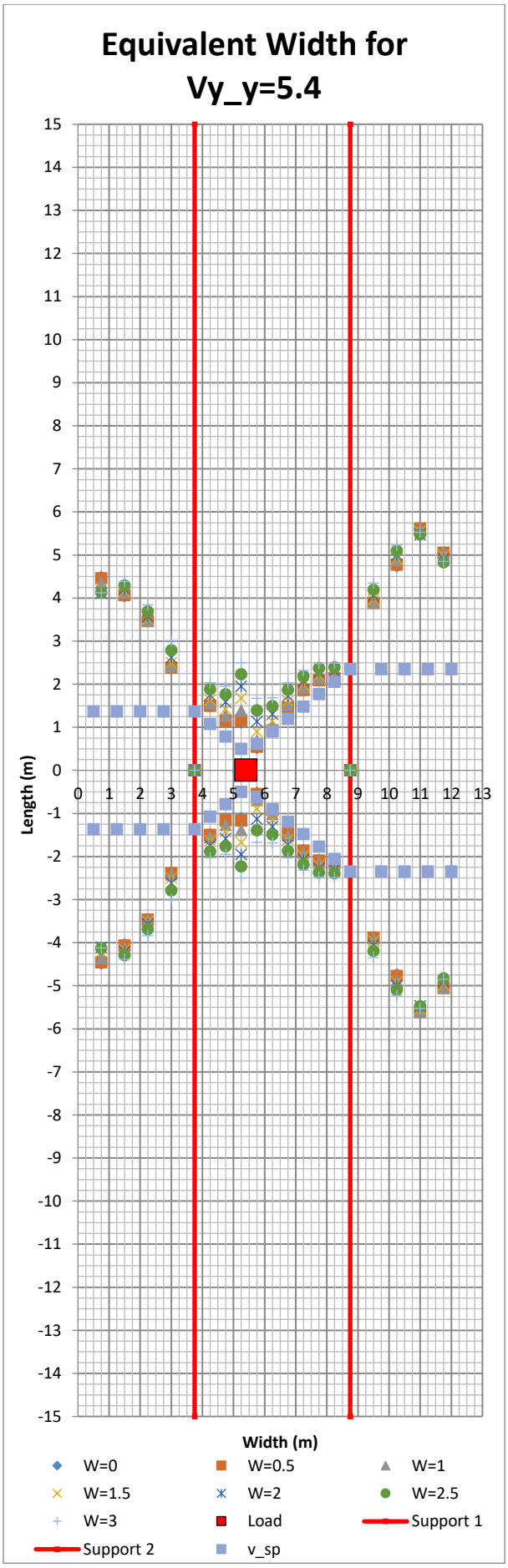

Equivalent Width for My_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

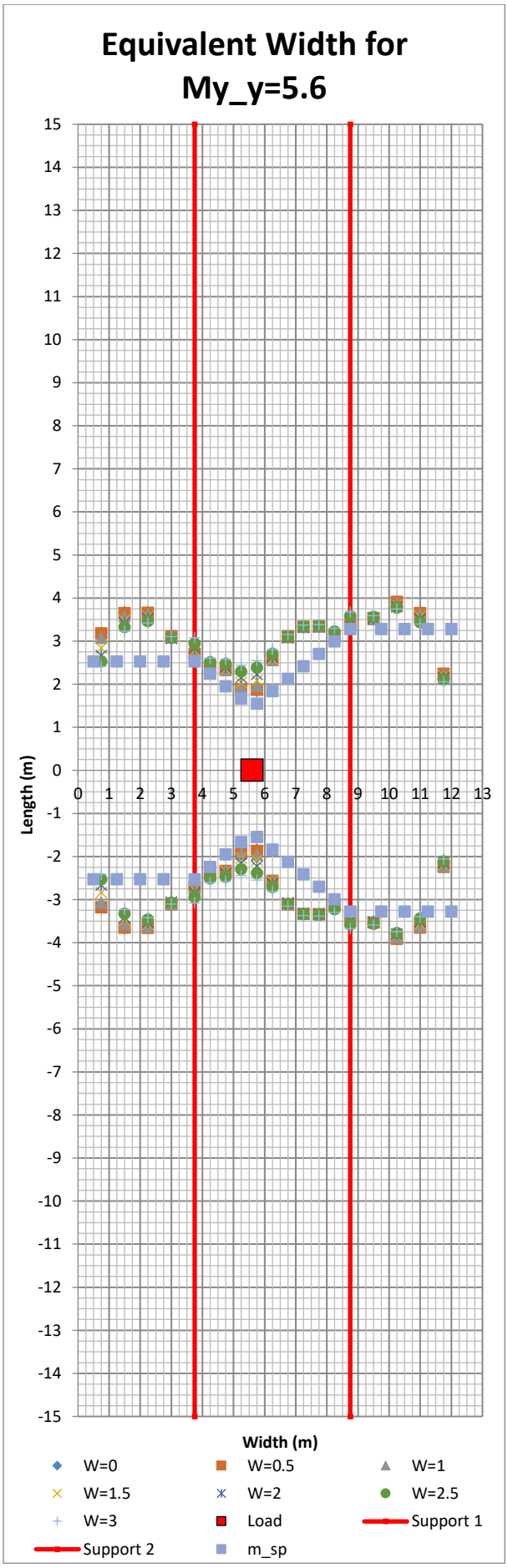

Equivalent Width for Vy_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

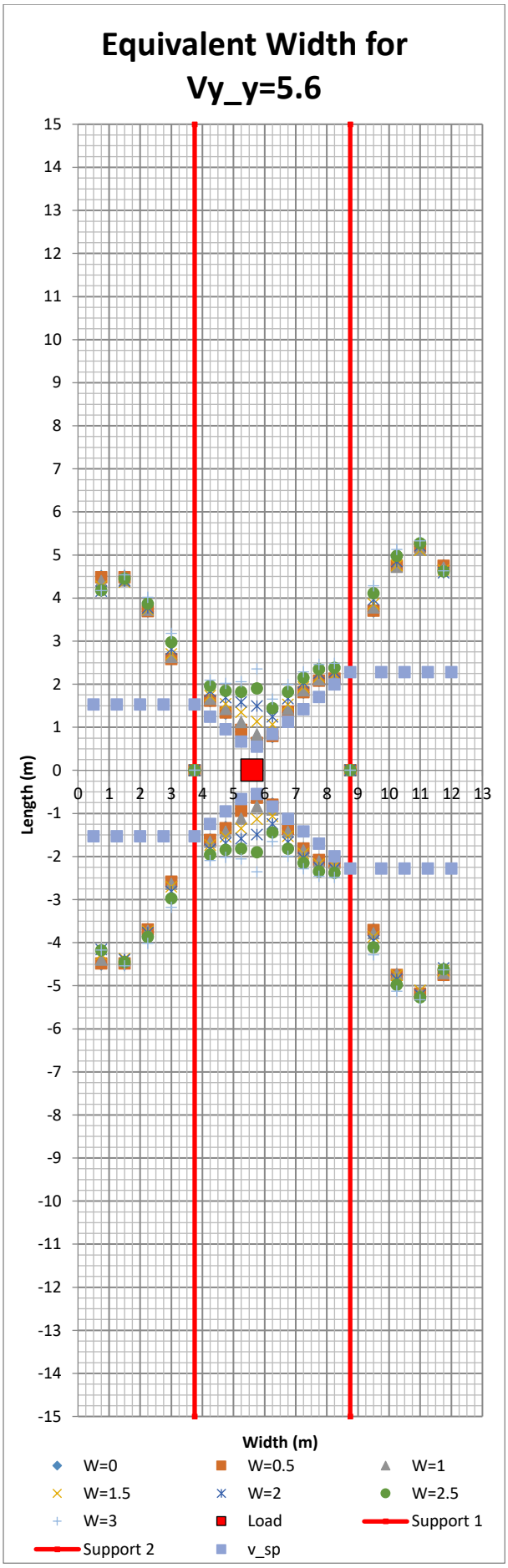

Equivalent Width for My_y=5.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

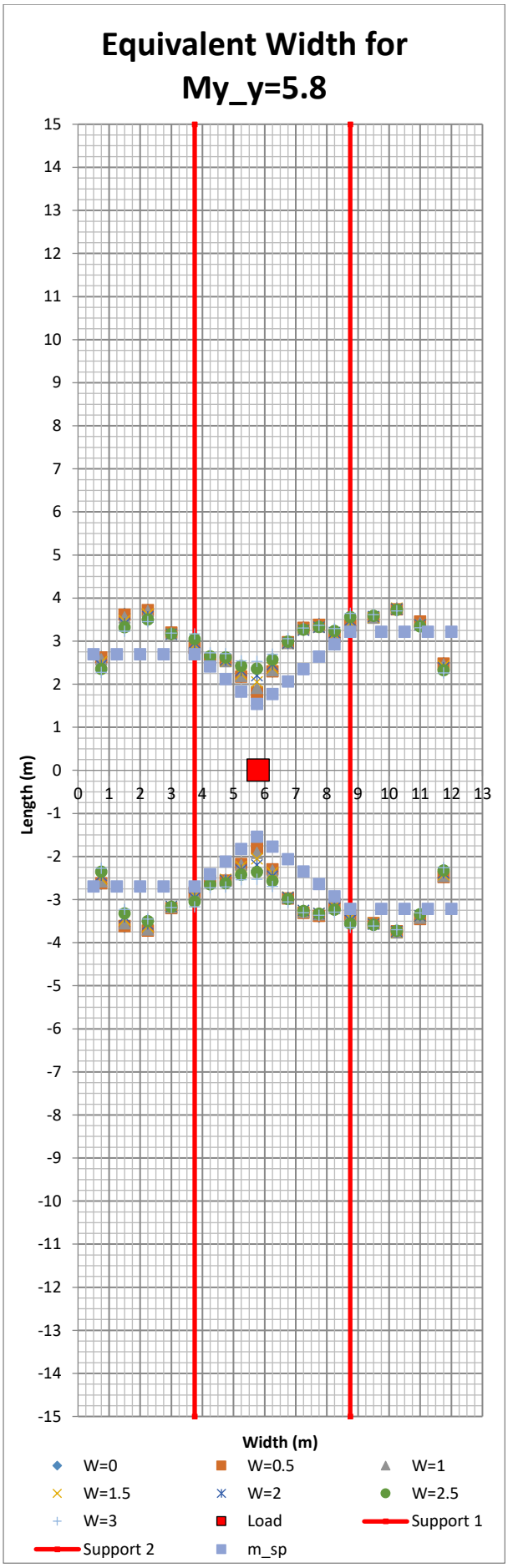

Equivalent Width for Vy_y=5.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

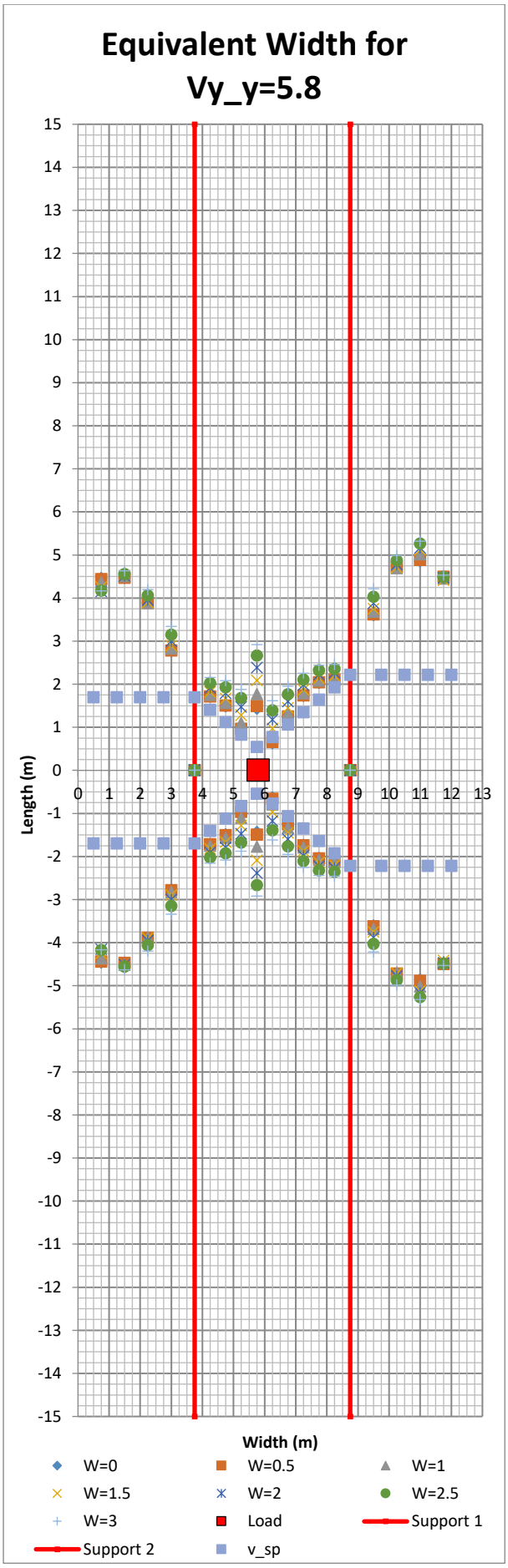

Equivalent Width for My_y=5.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

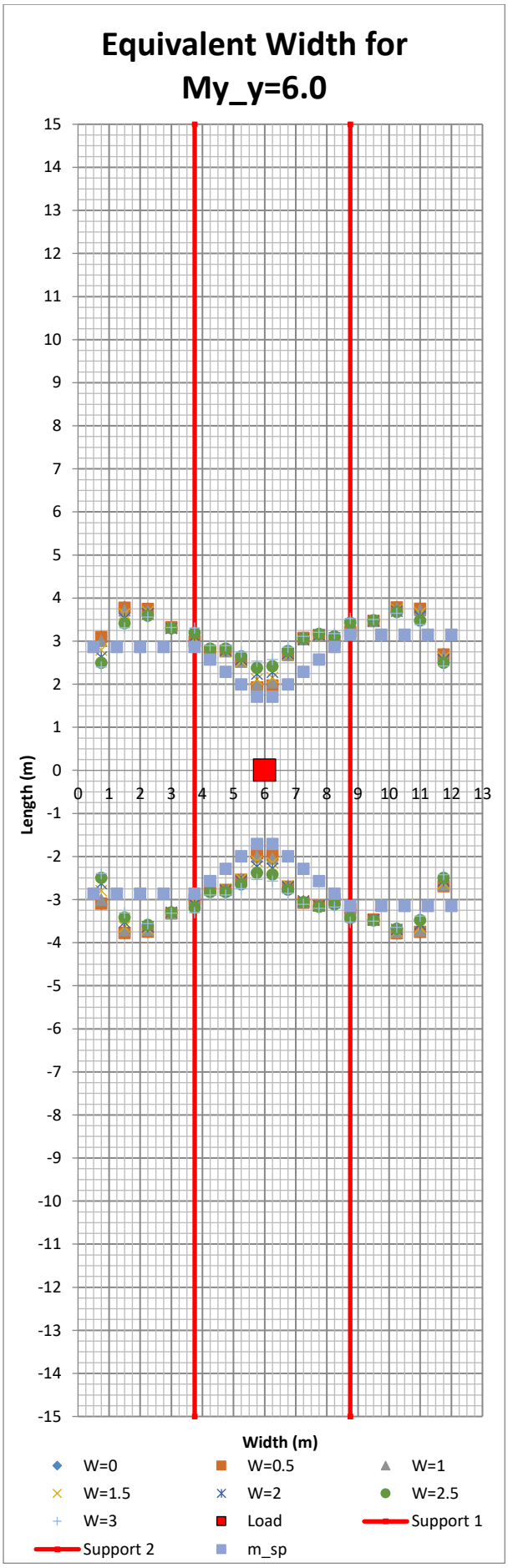

Equivalent Width for Vy_y=5.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

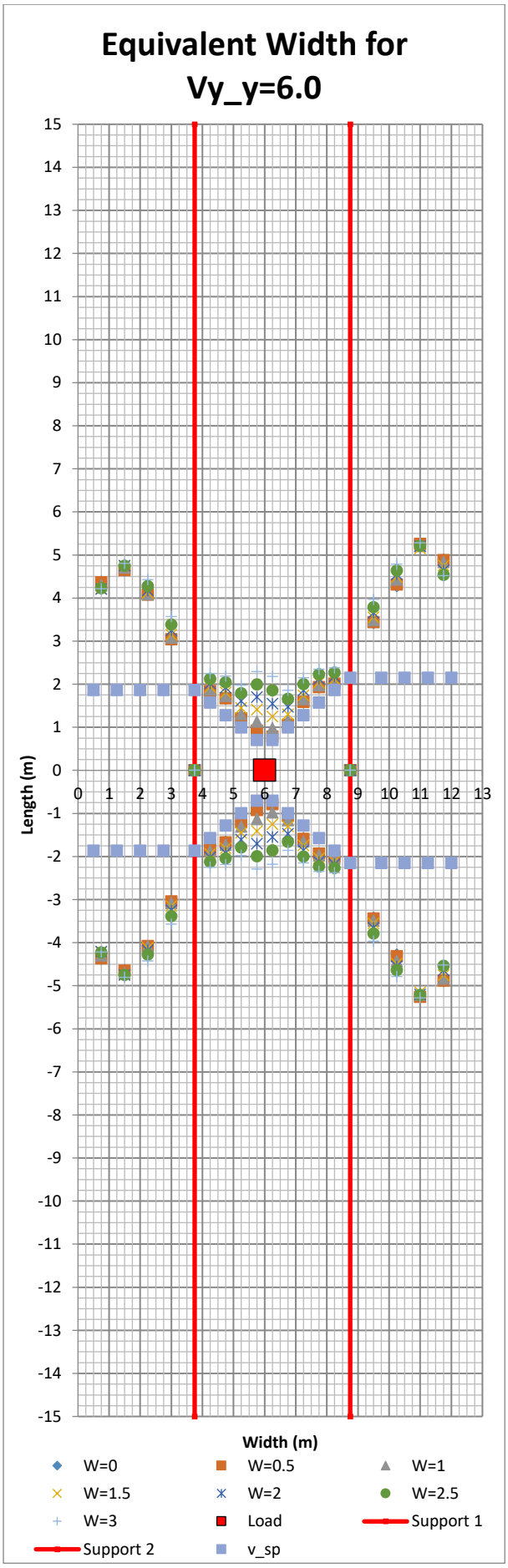

Equivalent Width for My_y=5.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

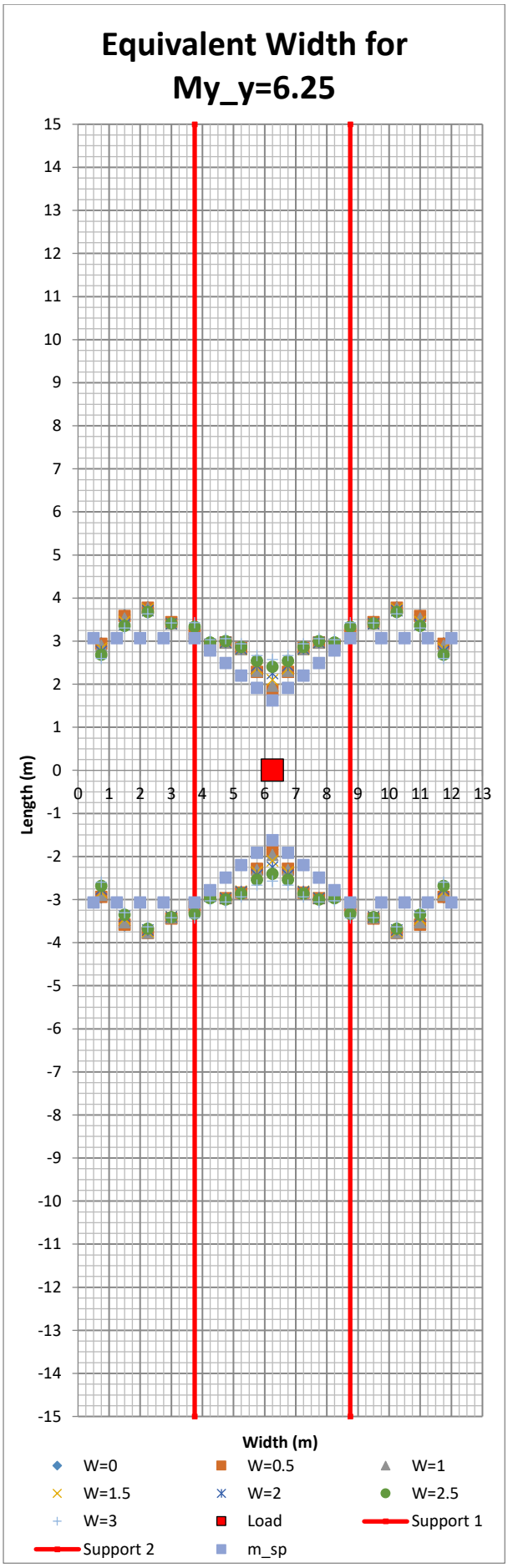

Equivalent Width for Vy_y=5.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

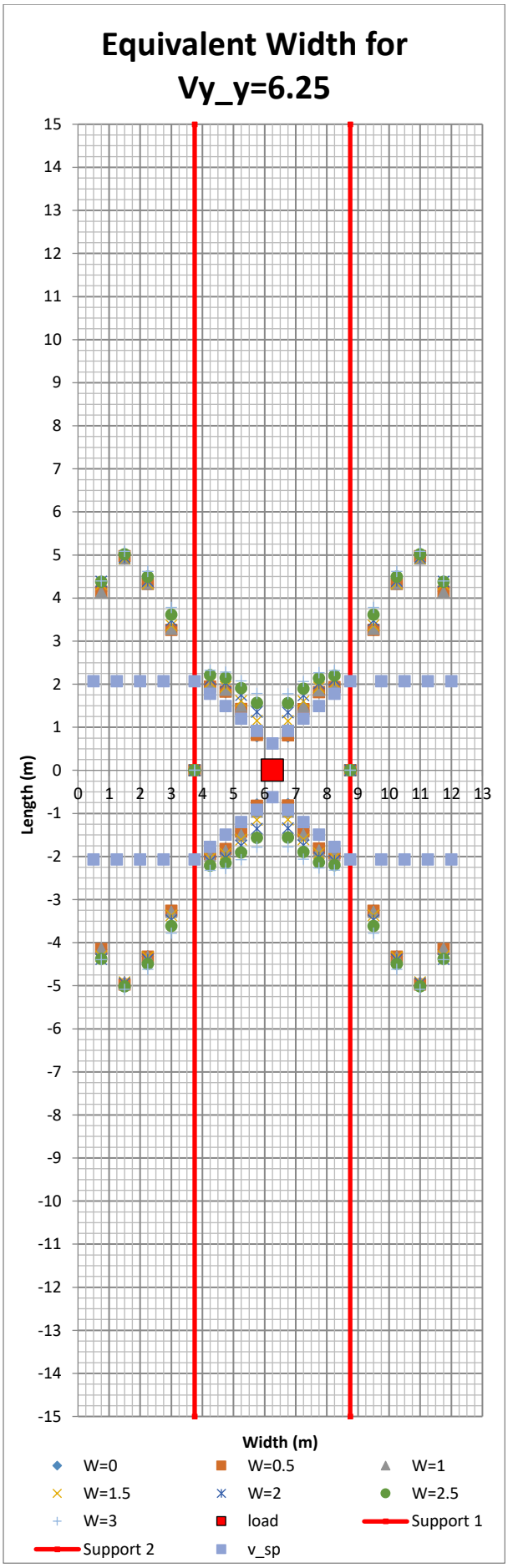

Equivalent Width for My_y=5.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=5.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

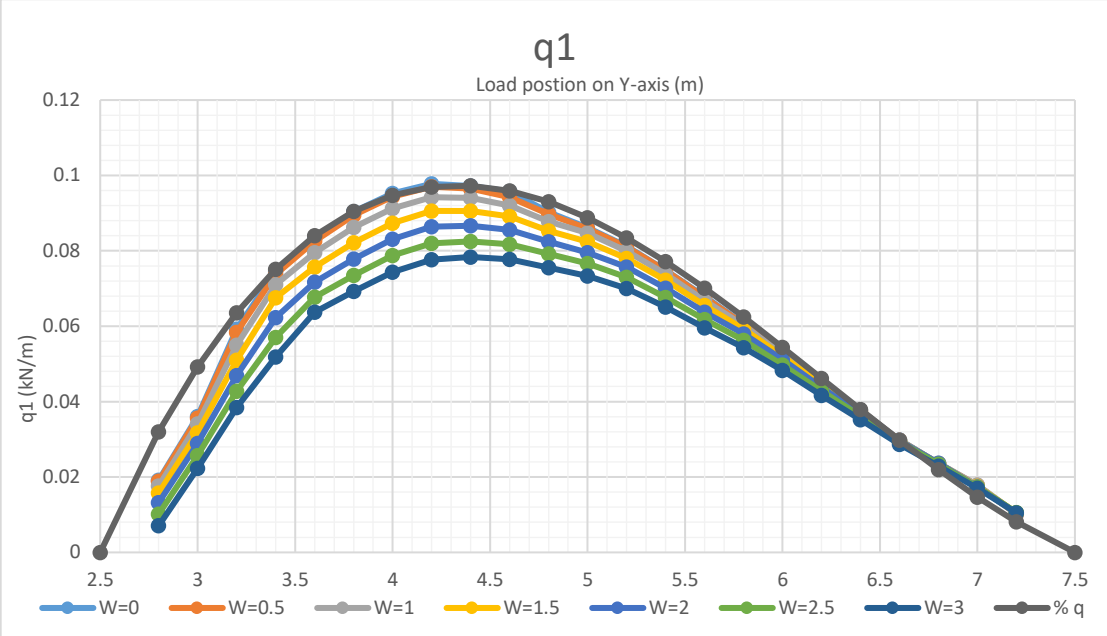

Equivalent Width for My_y=6.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

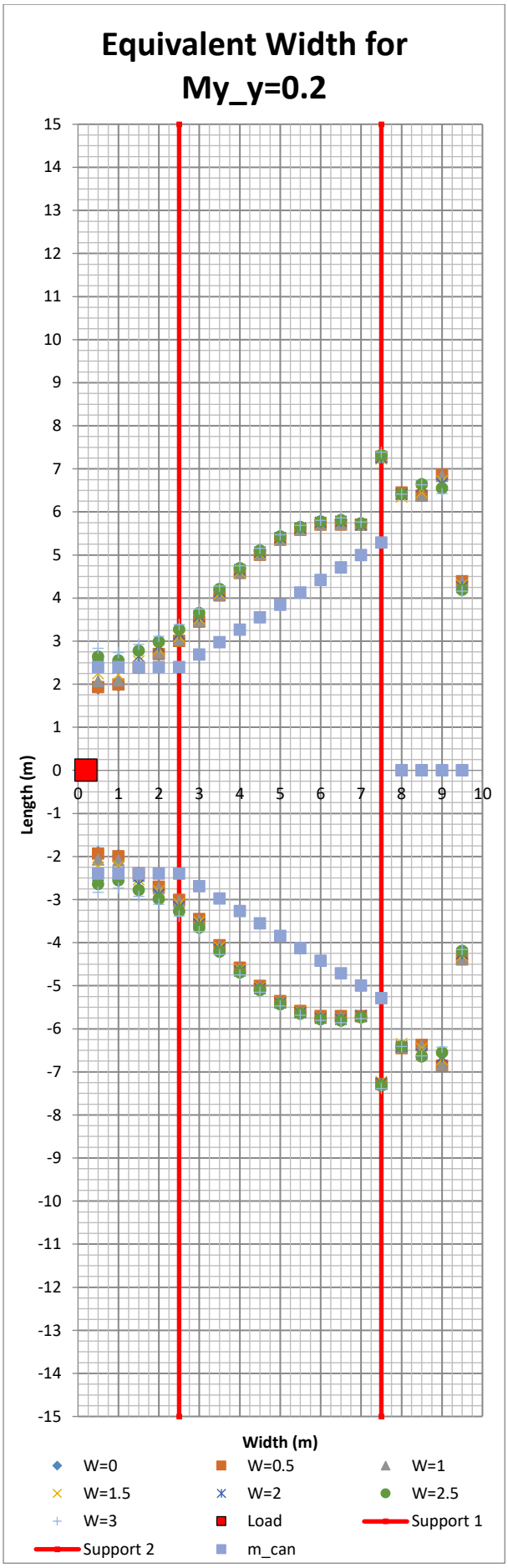

Equivalent Width for Vy_y=6.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

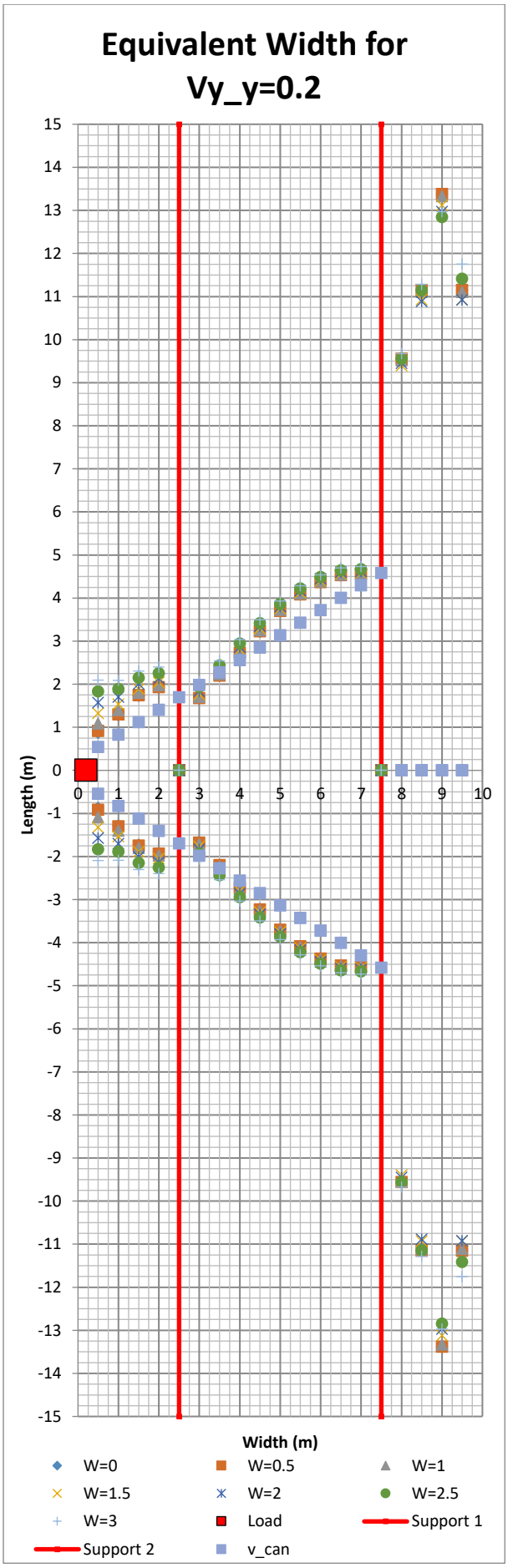

Equivalent Width for My_y=6.25
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

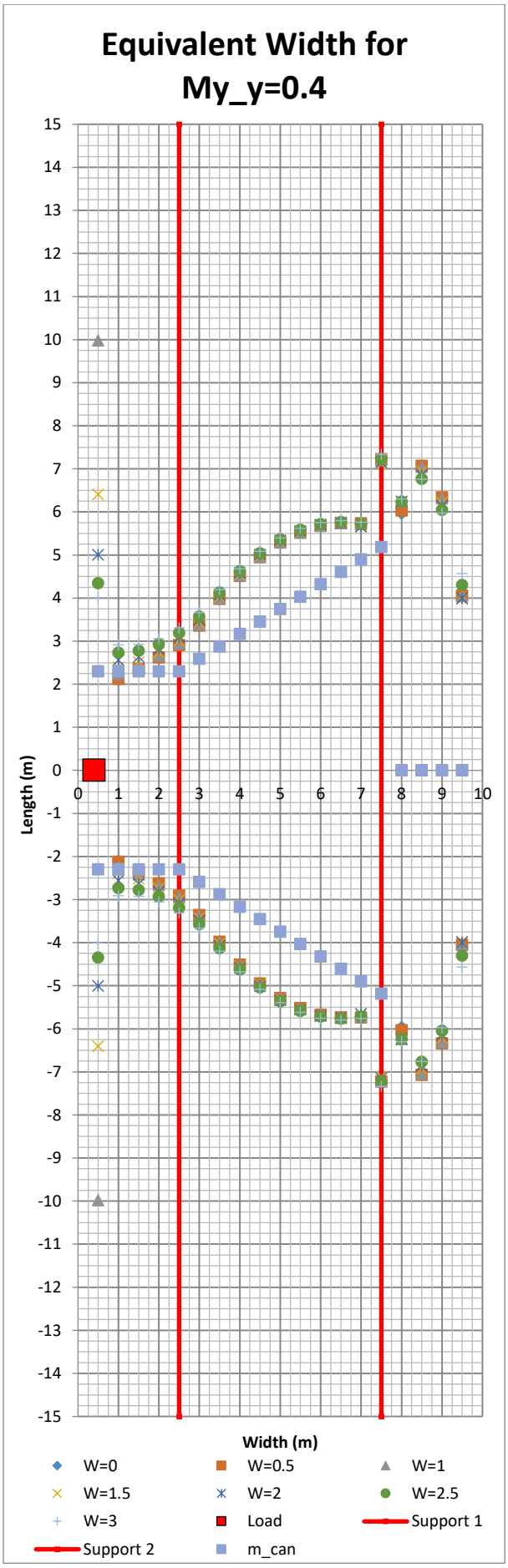

Equivalent Width for Vy_y=6.25
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
load
Support 1
Support 2
v_sp

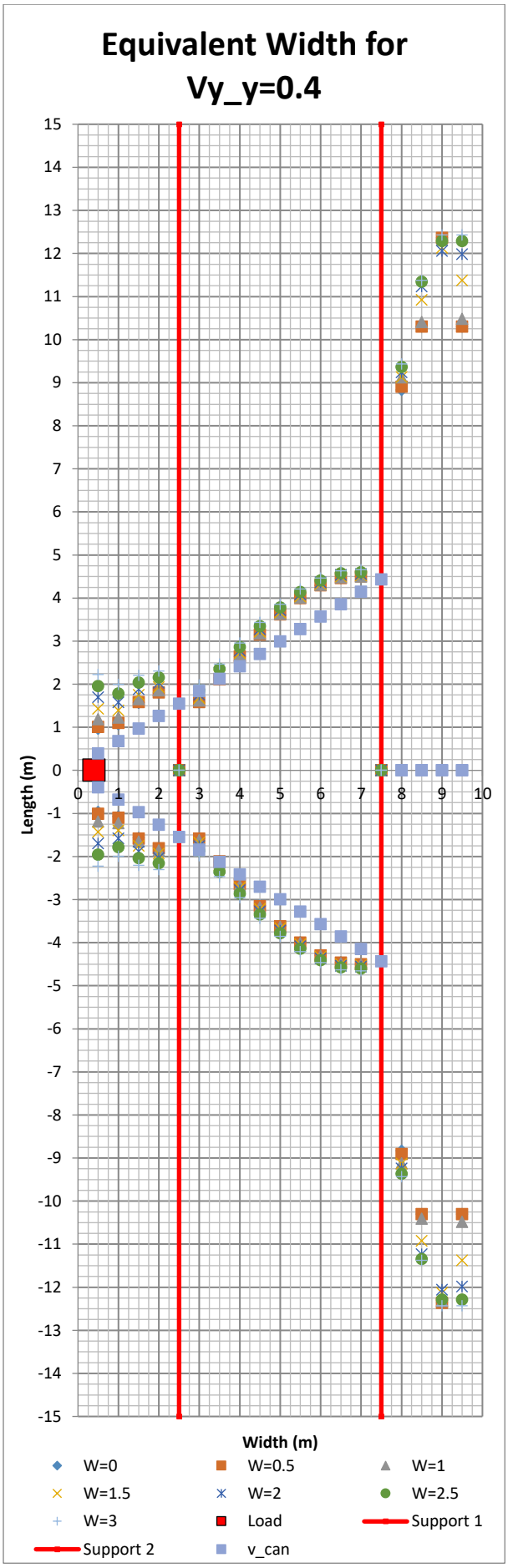
11.6 Sixth Variation $t = [0.15, 0.3, 0.3]m$

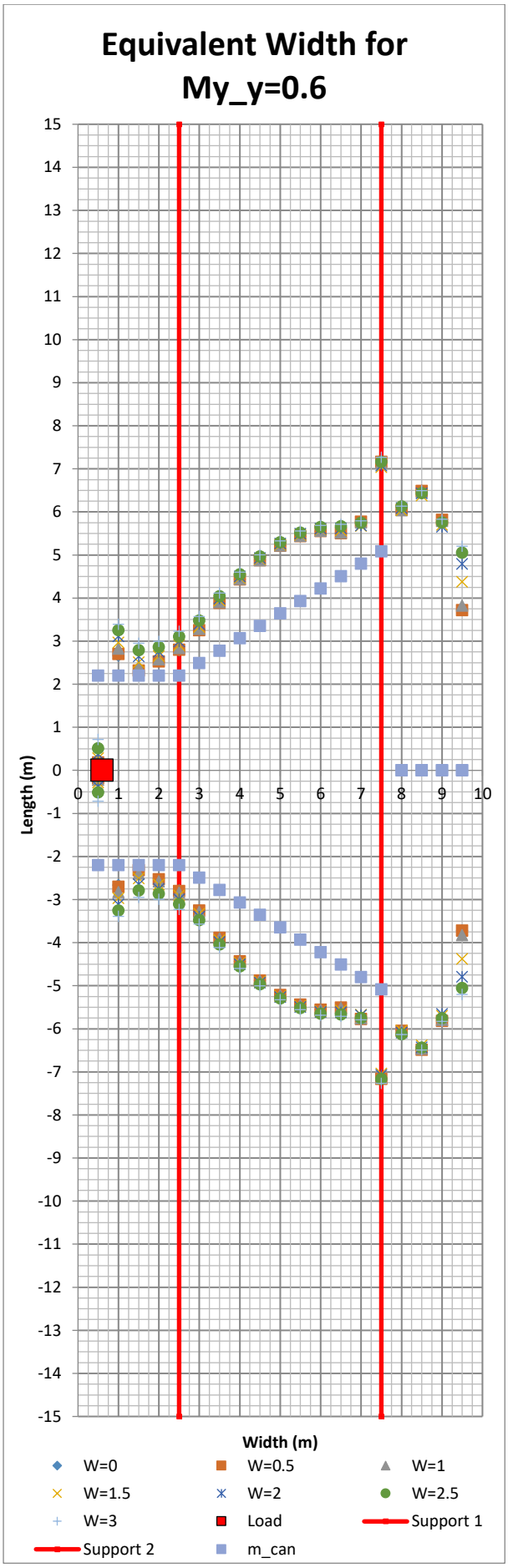


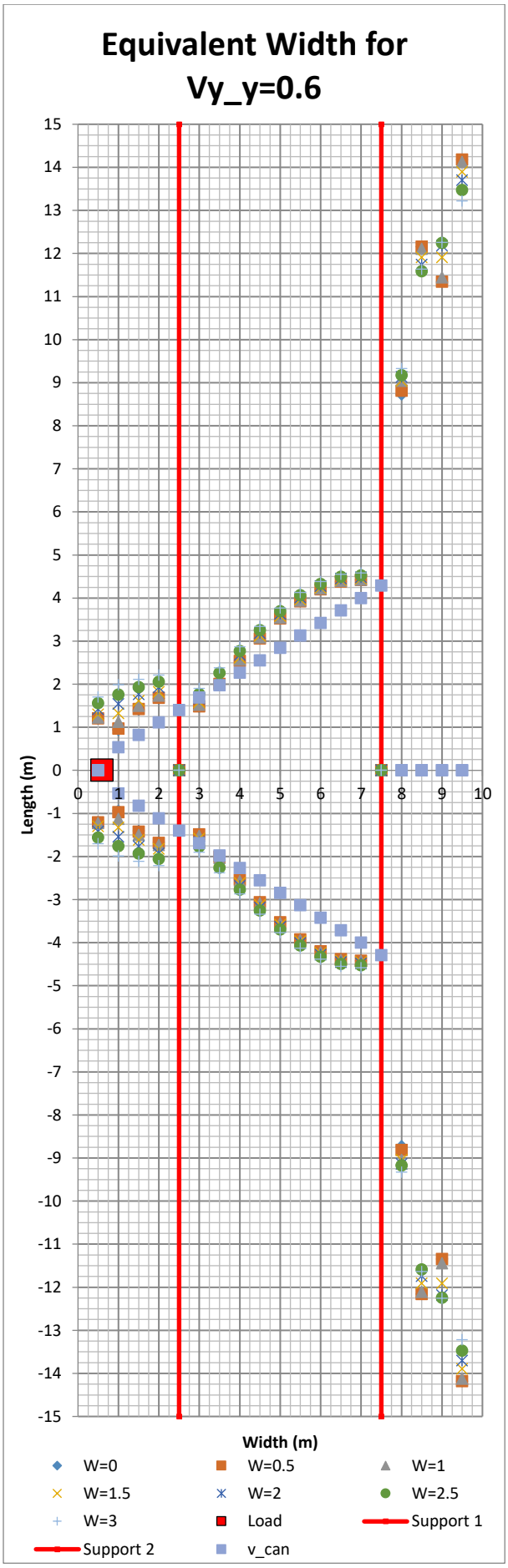

Equivalent Width for My_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

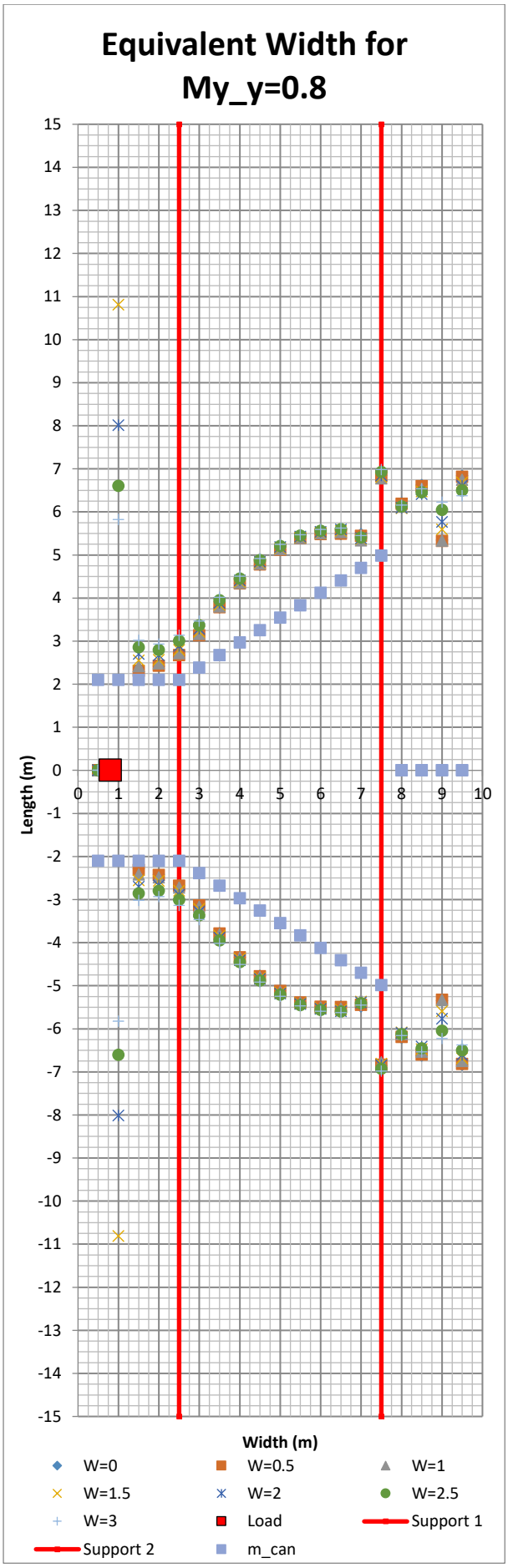

Equivalent Width for Vy_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

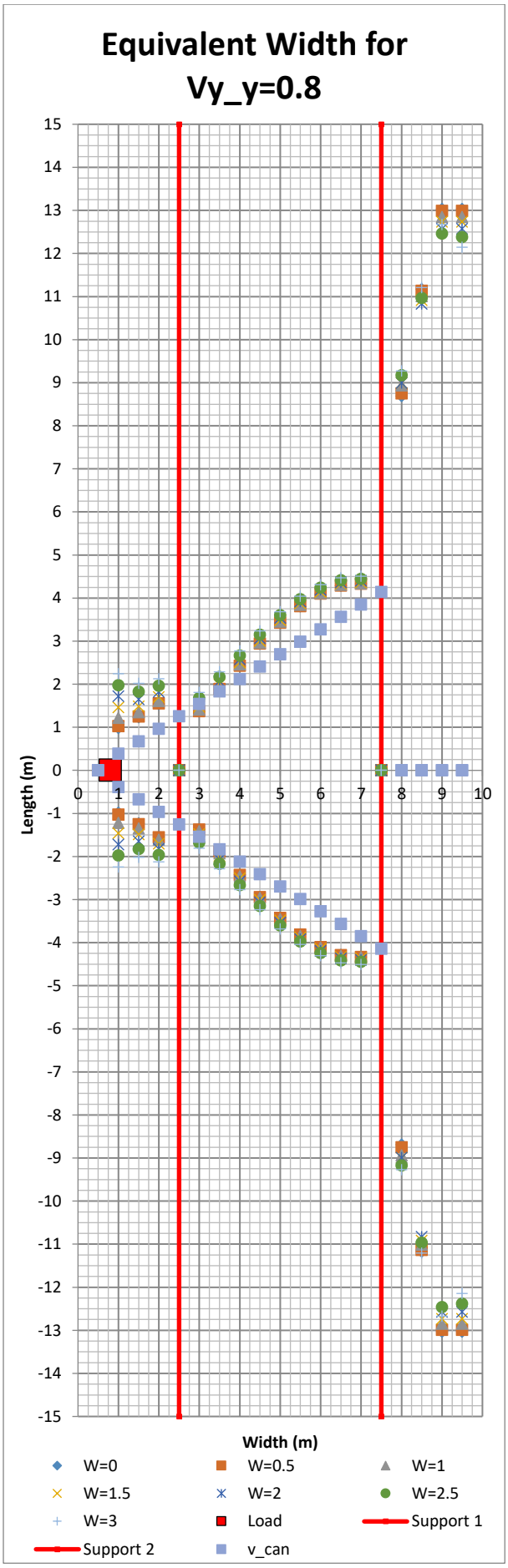

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

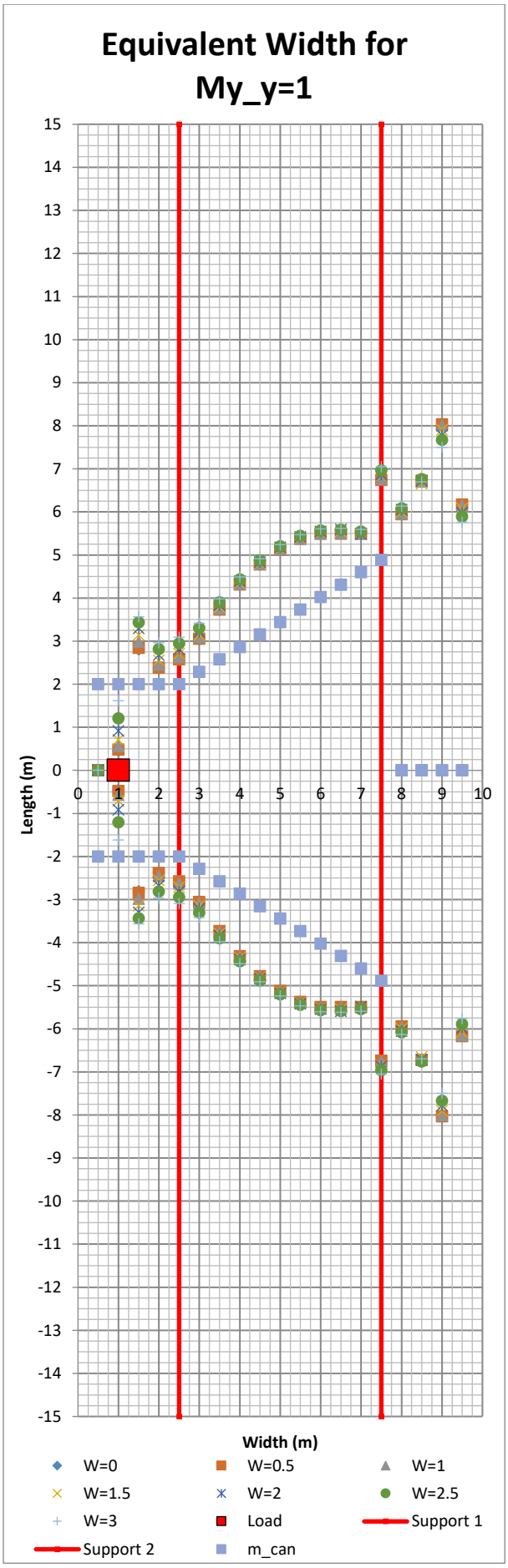

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

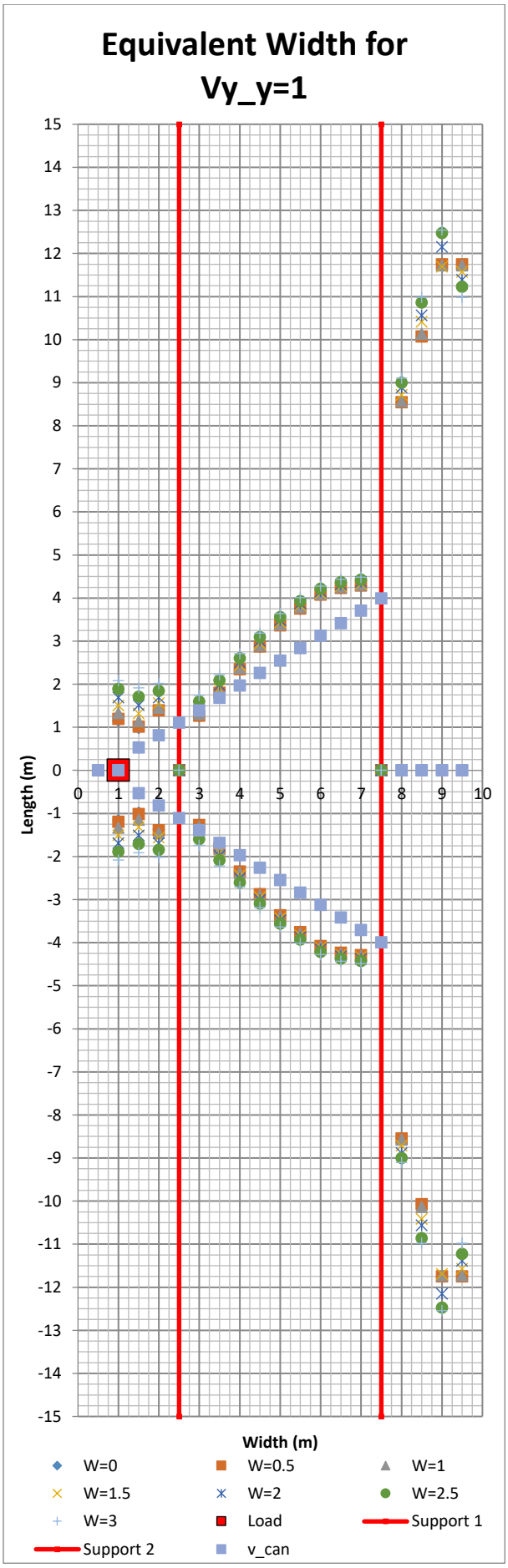

Equivalent Width for My_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

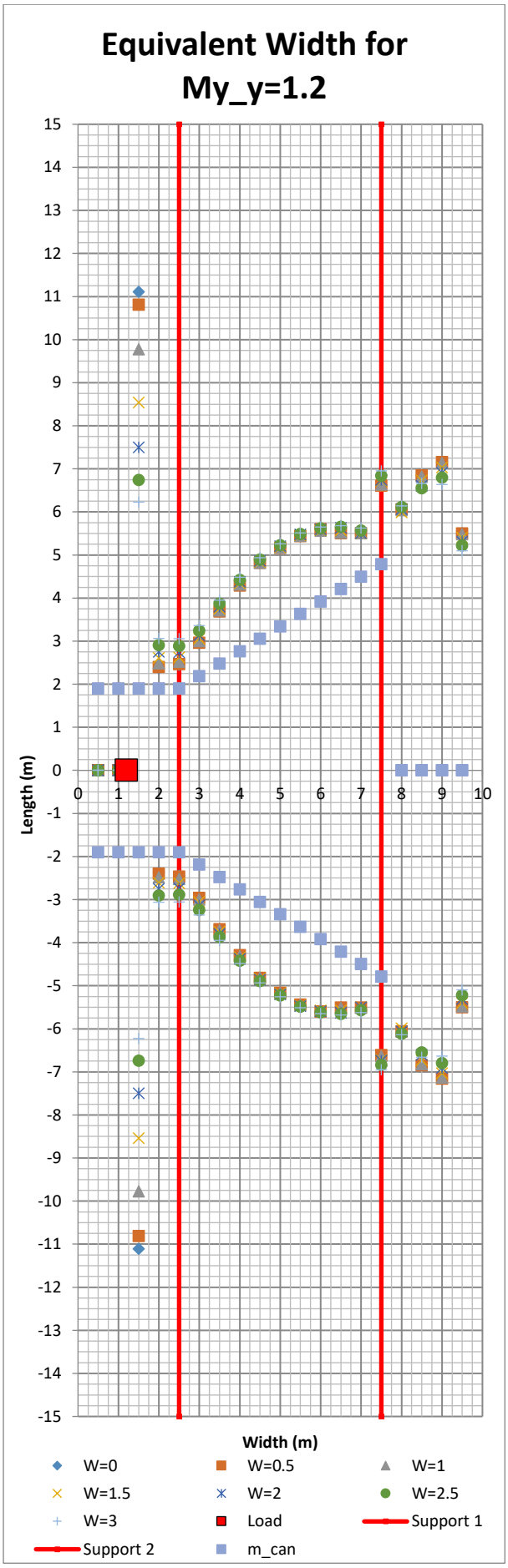

Equivalent Width for Vy_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

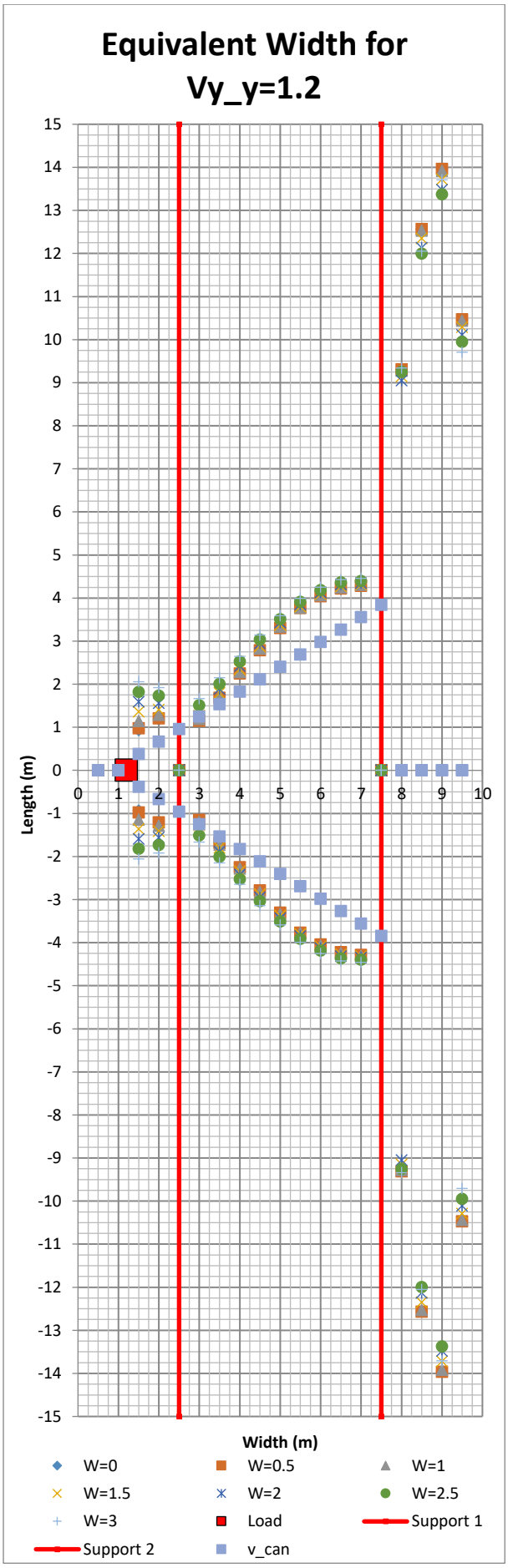

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

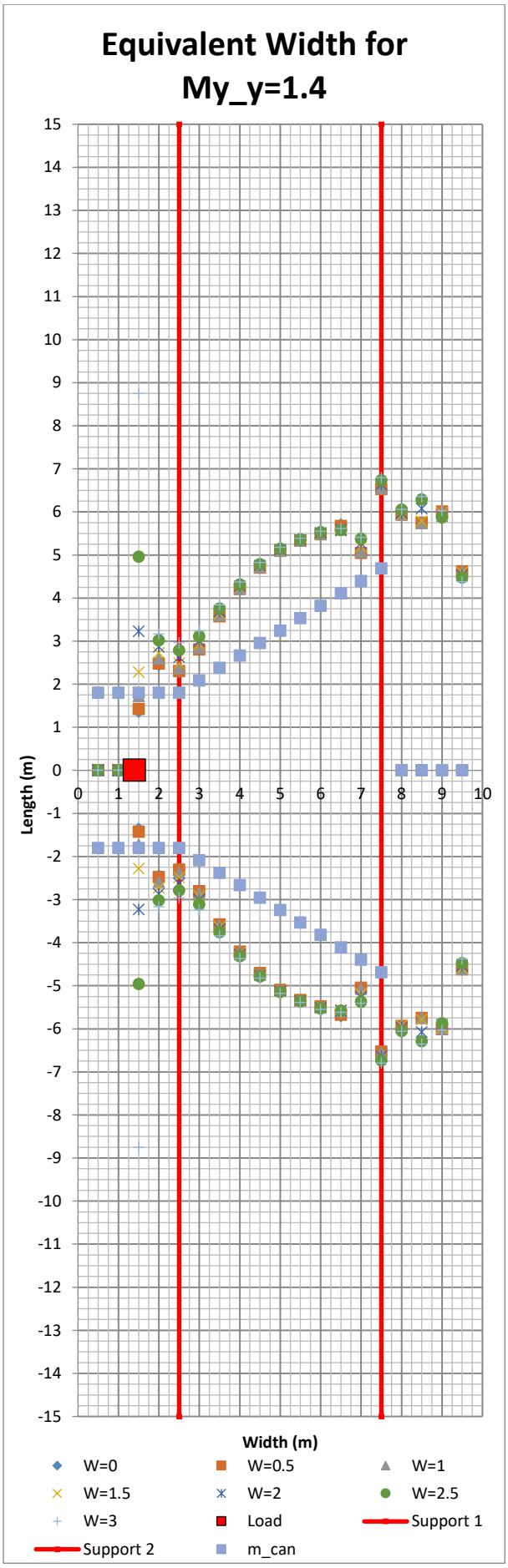

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

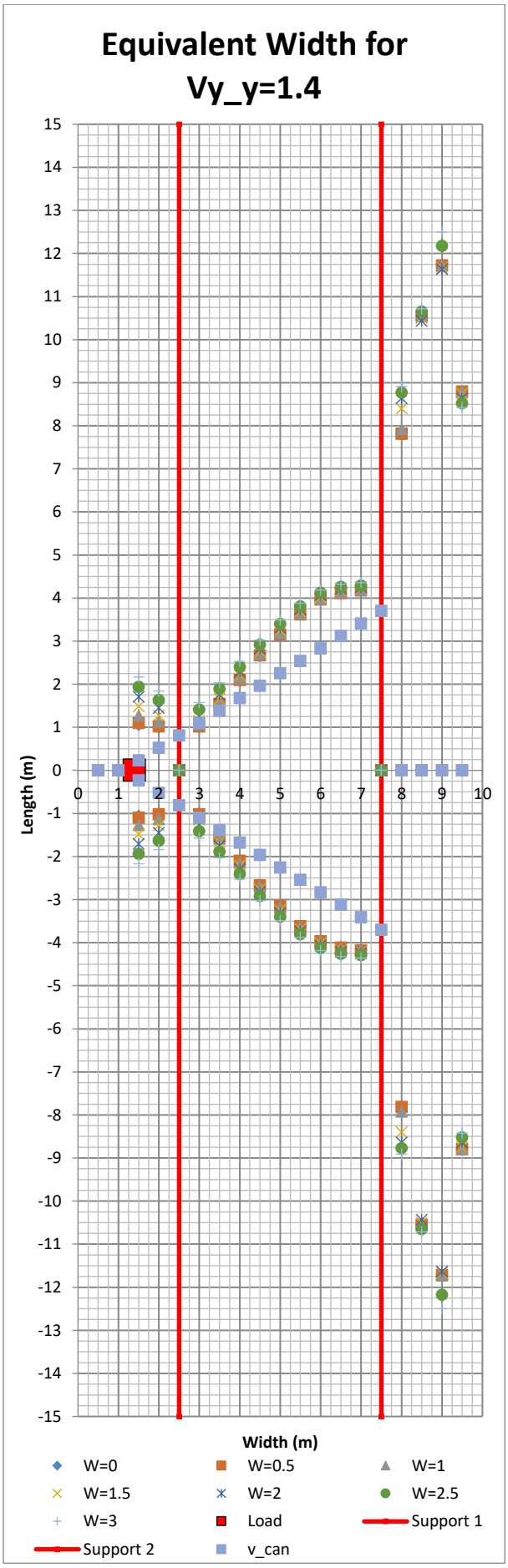

Equivalent Width for My_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

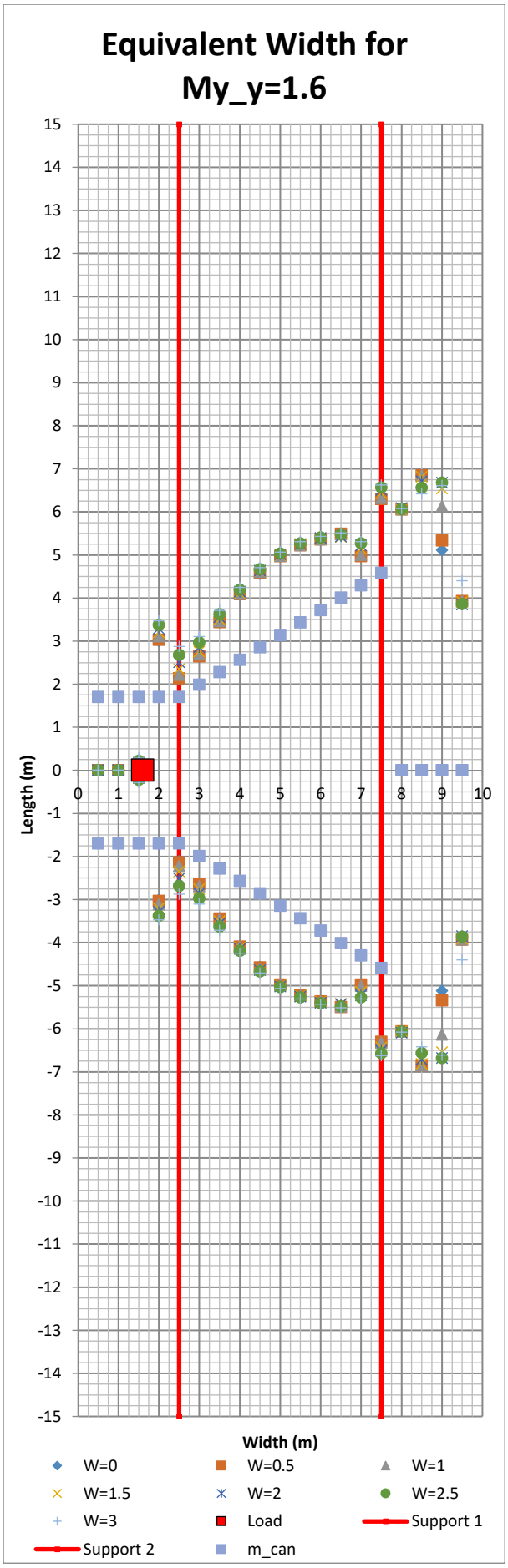

Equivalent Width for Vy_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

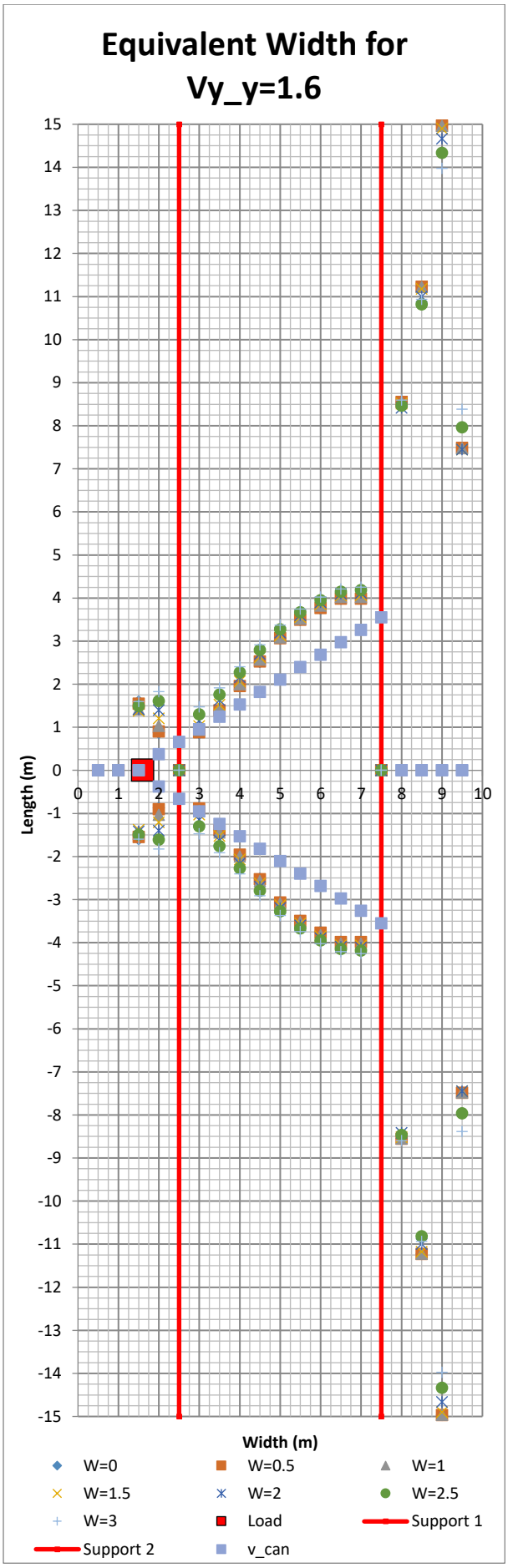

Equivalent Width for My_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

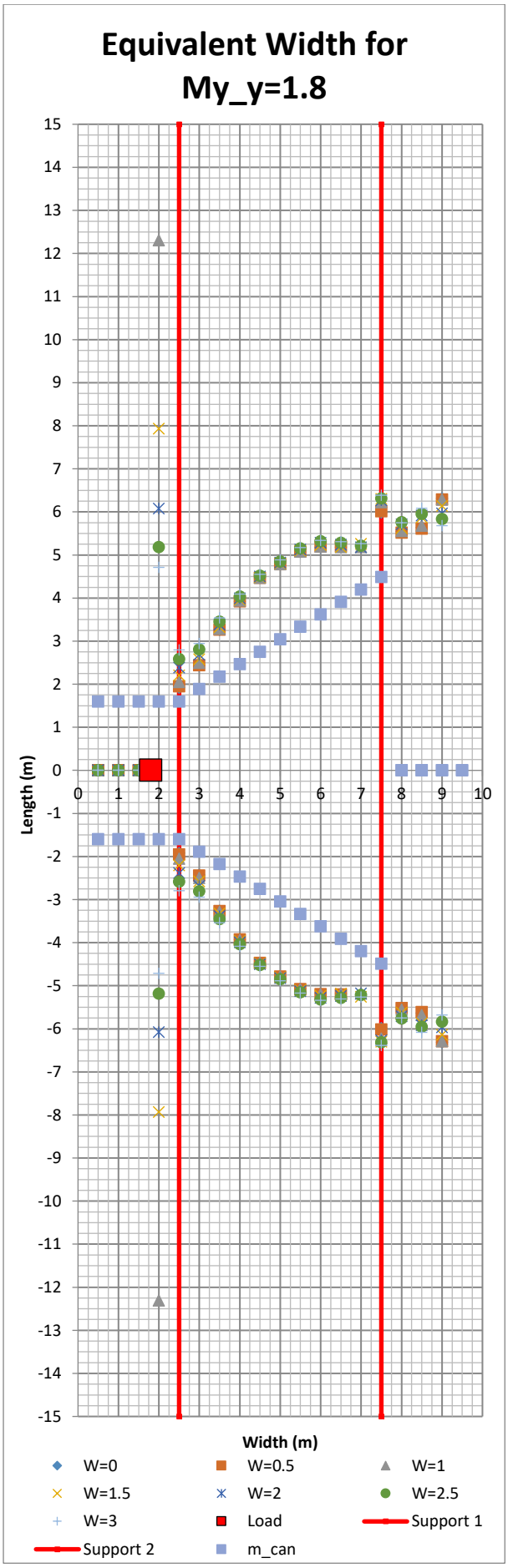

Equivalent Width for Vy_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

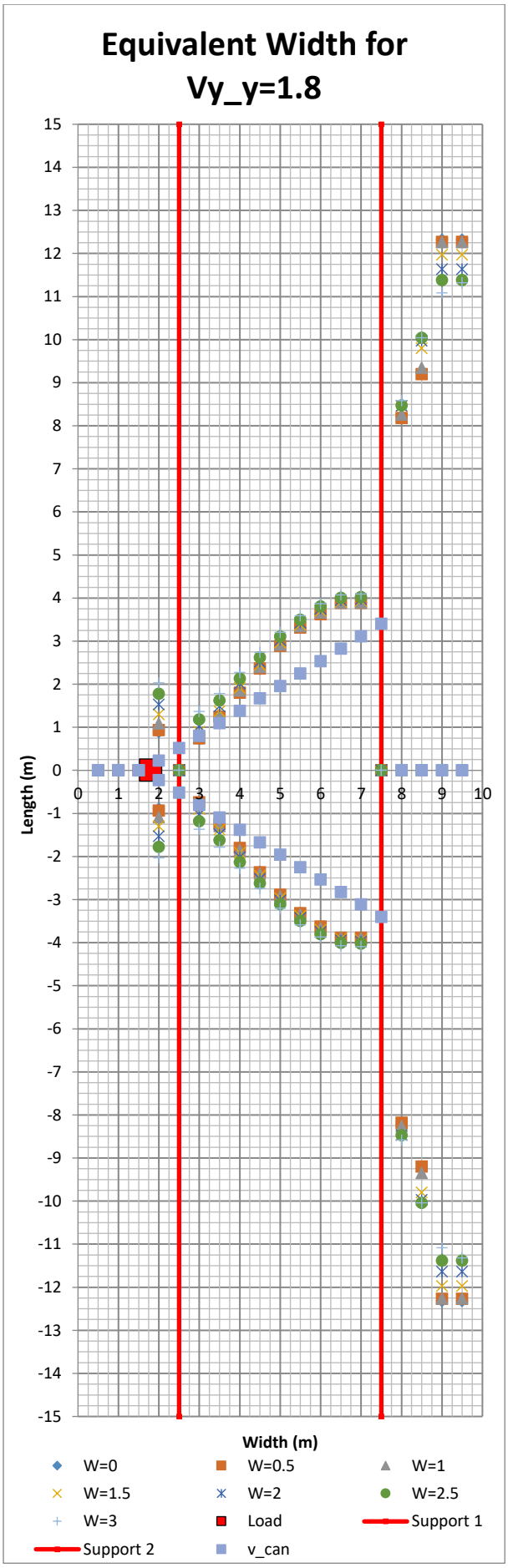

Equivalent Width for My_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

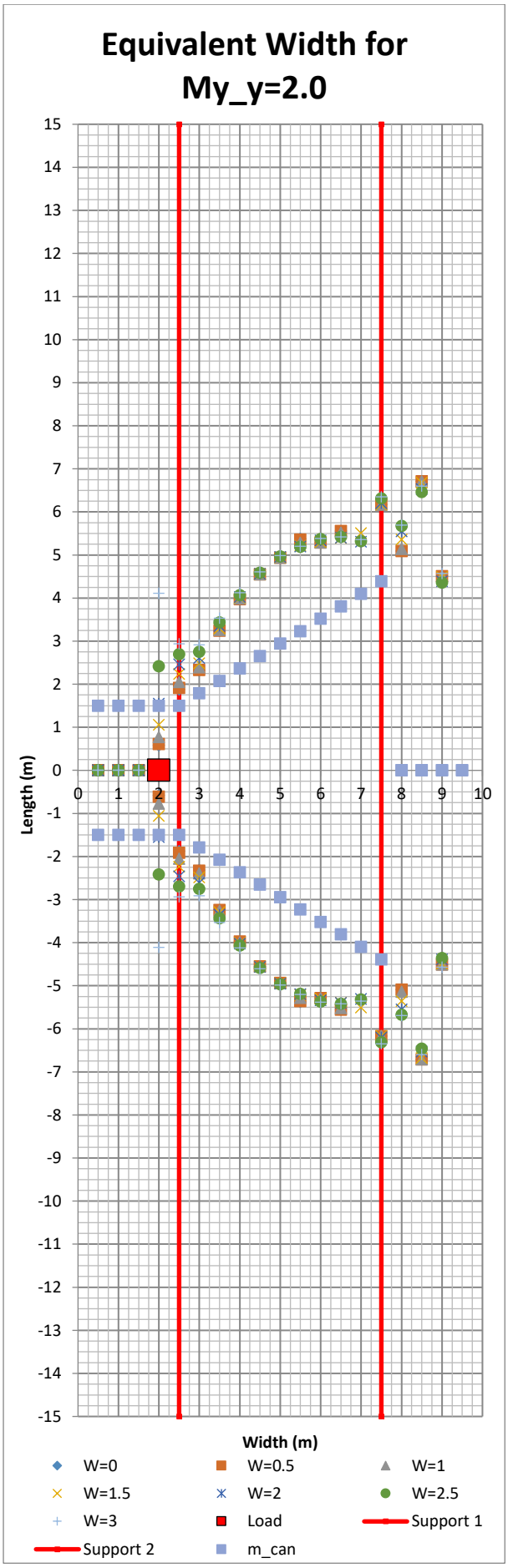

Equivalent Width for Vy_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

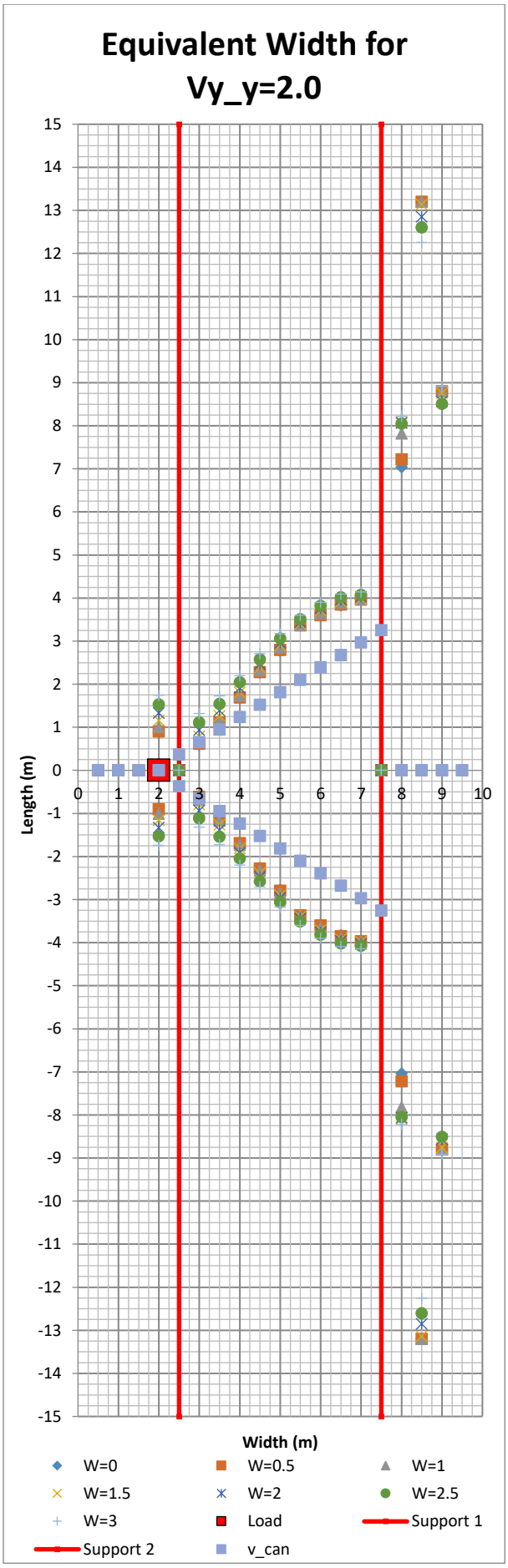

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

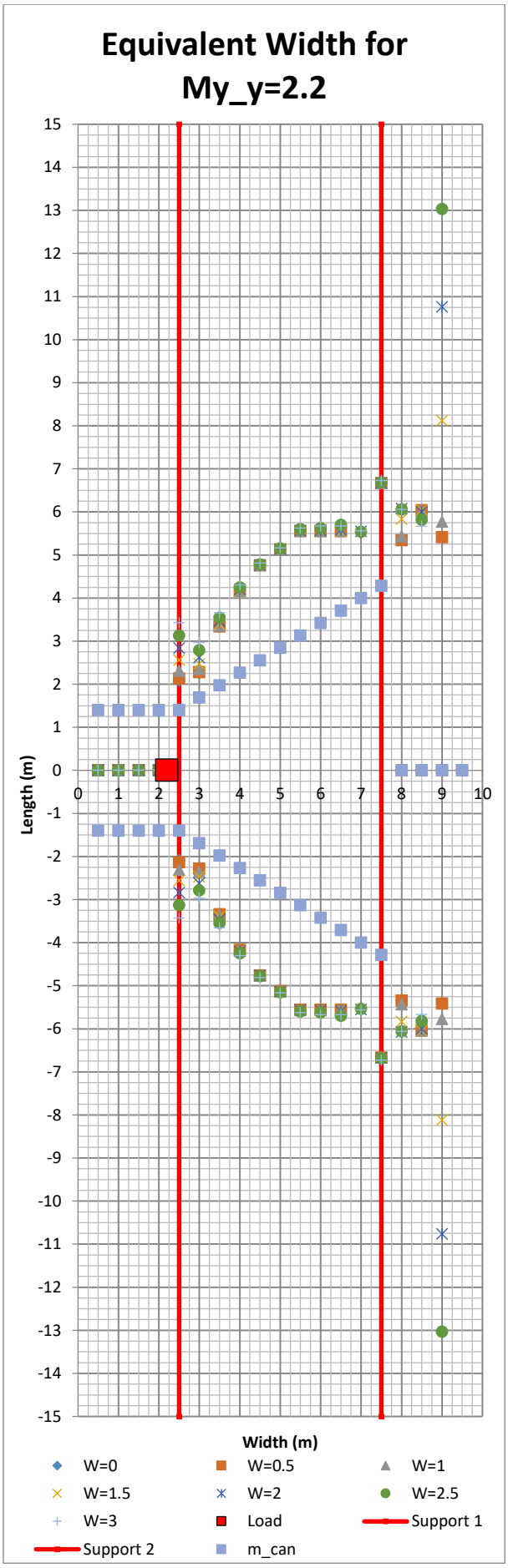

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

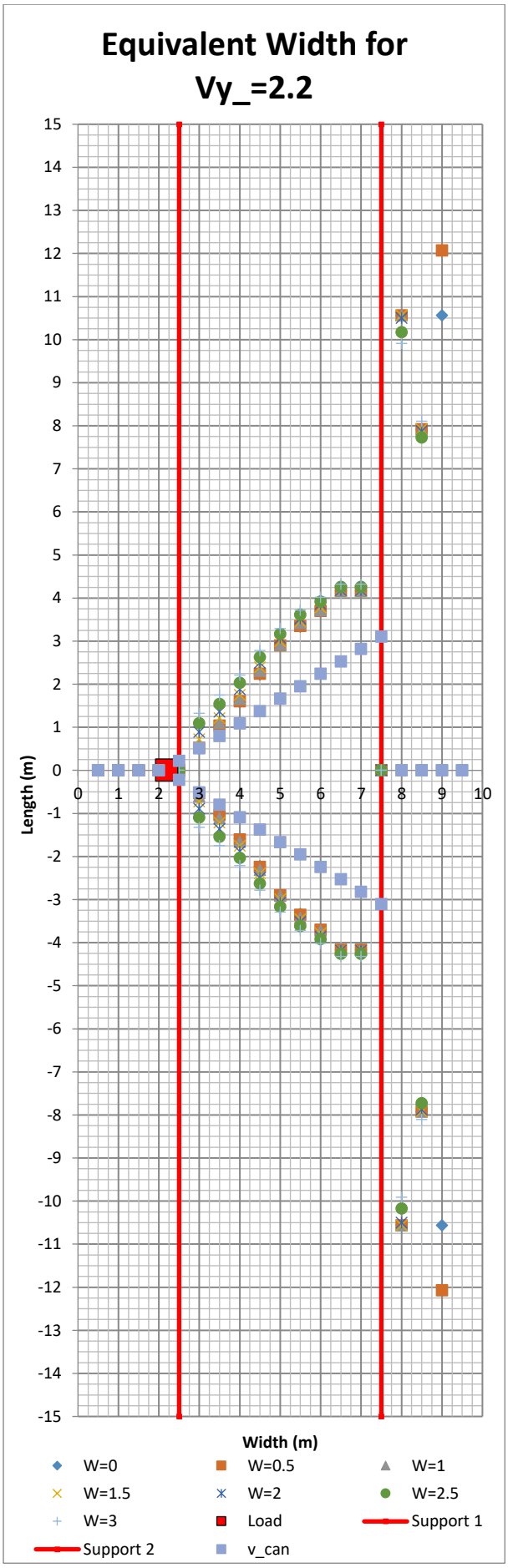

Equivalent Width for My_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

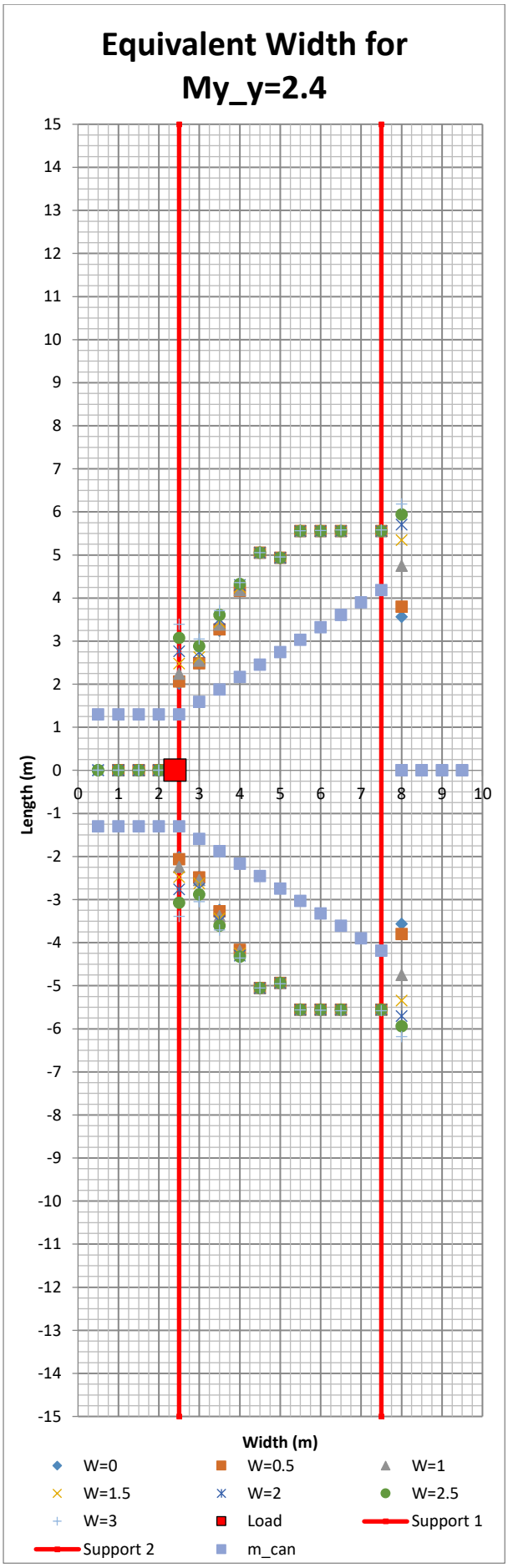

Equivalent Width for Vy_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

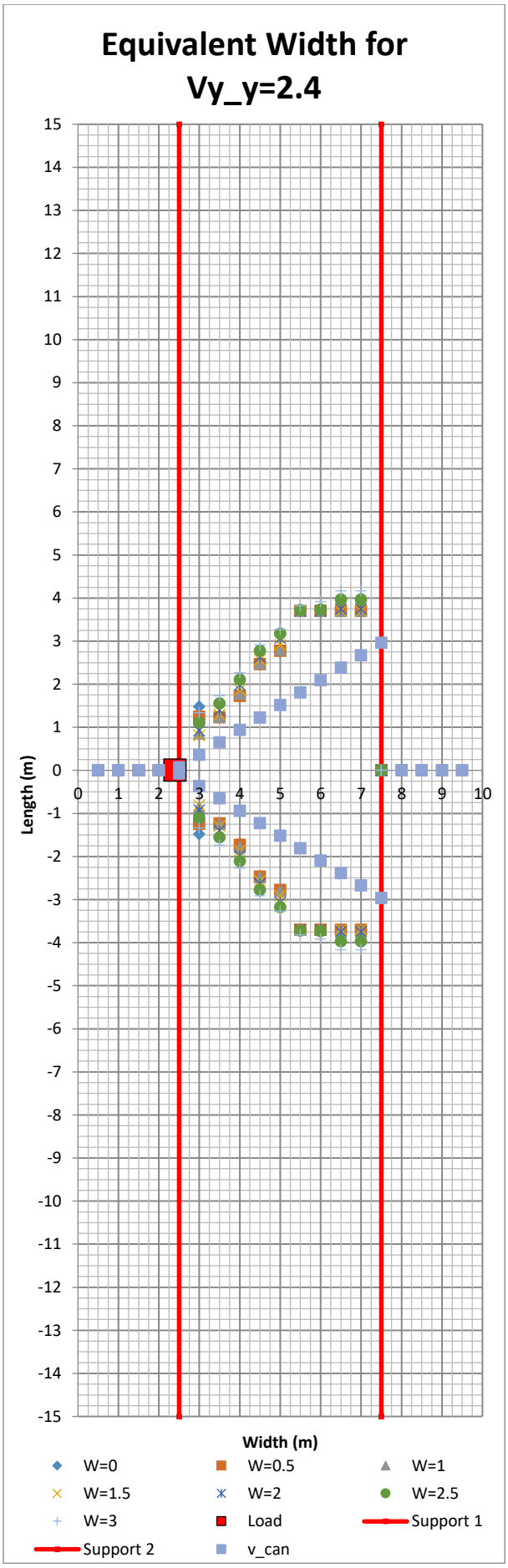

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

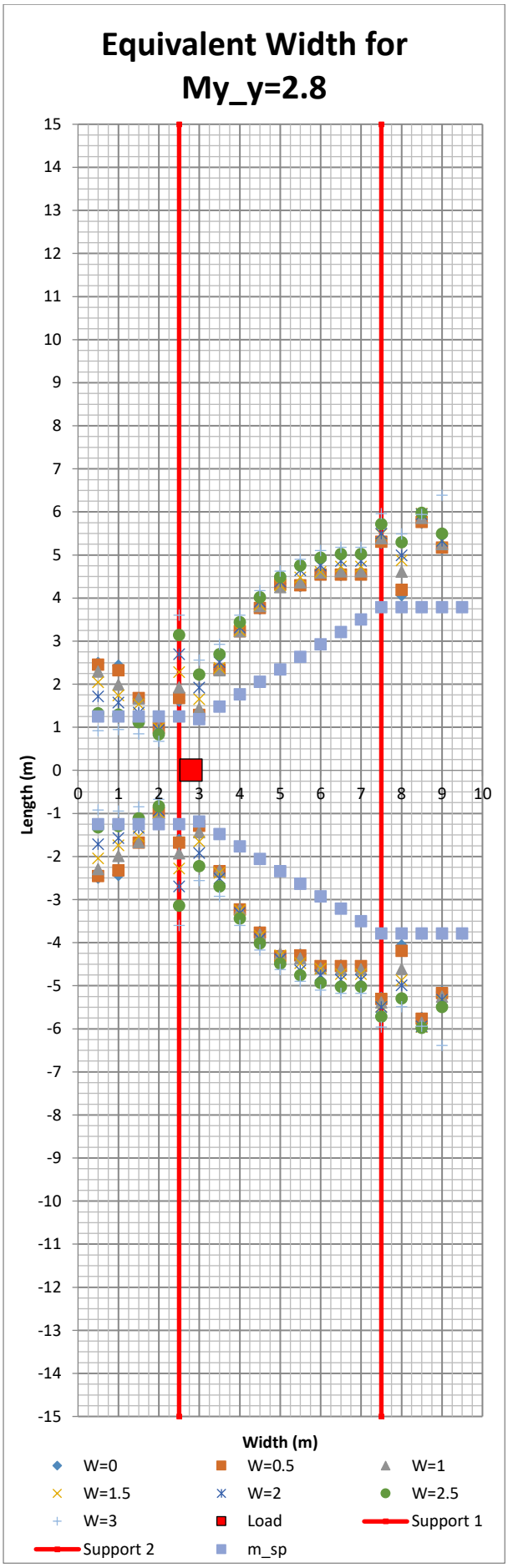

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

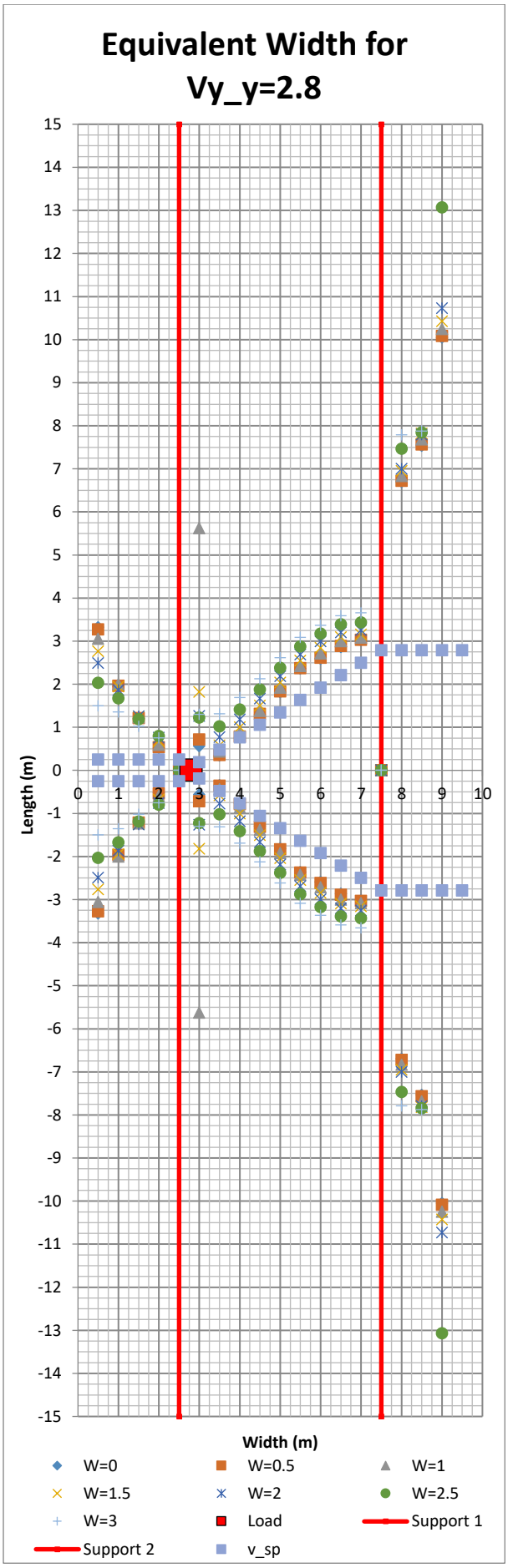

Equivalent Width for My_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

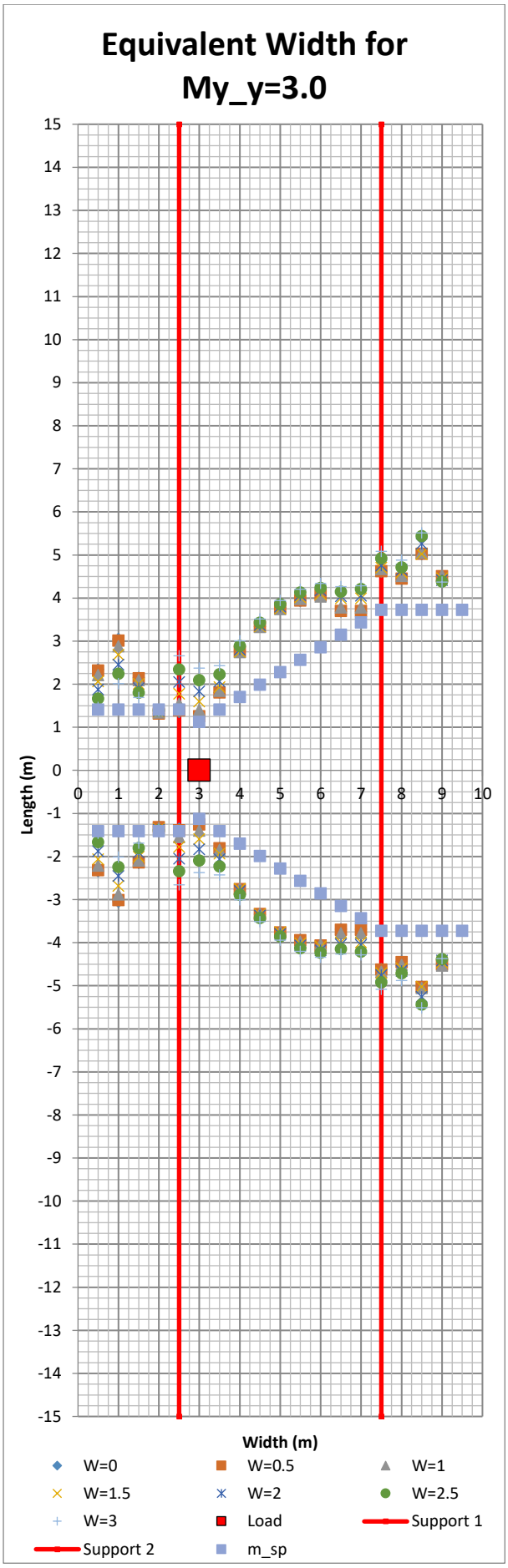

Equivalent Width for Vy_=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

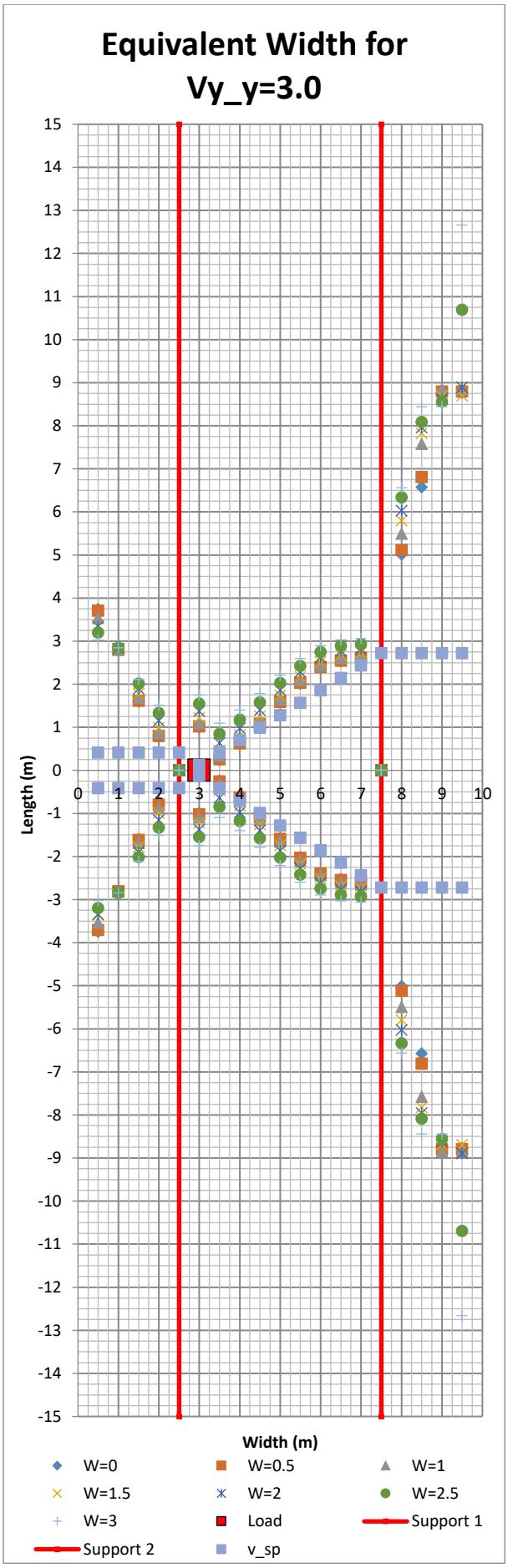

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

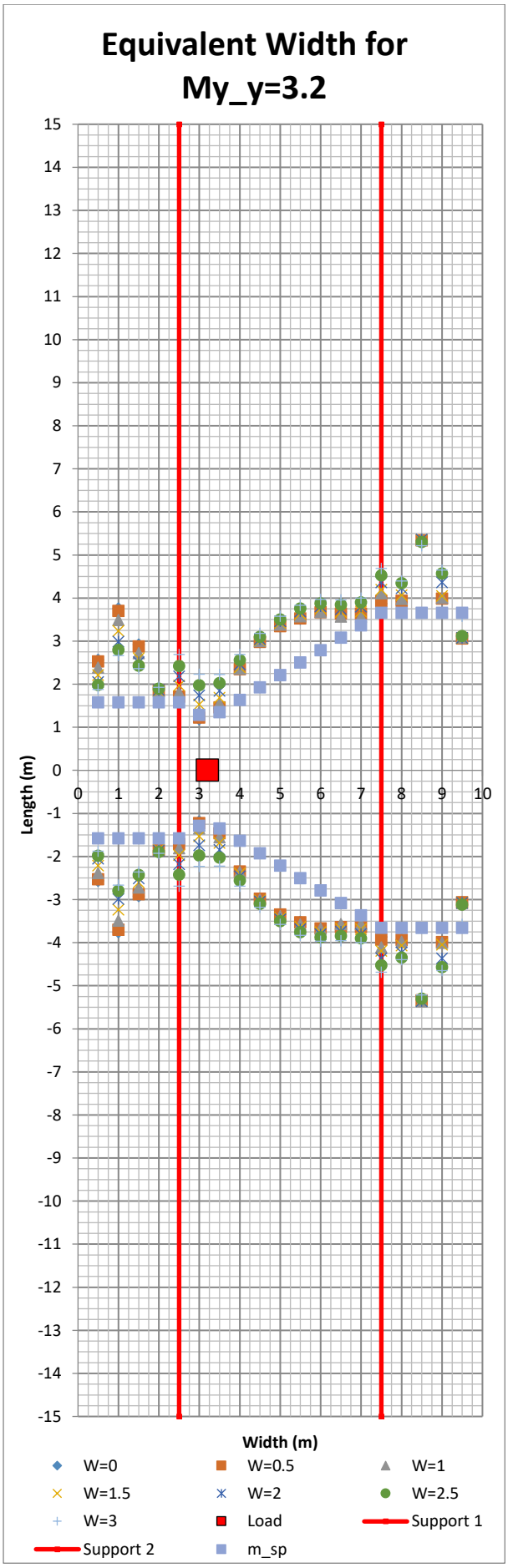

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

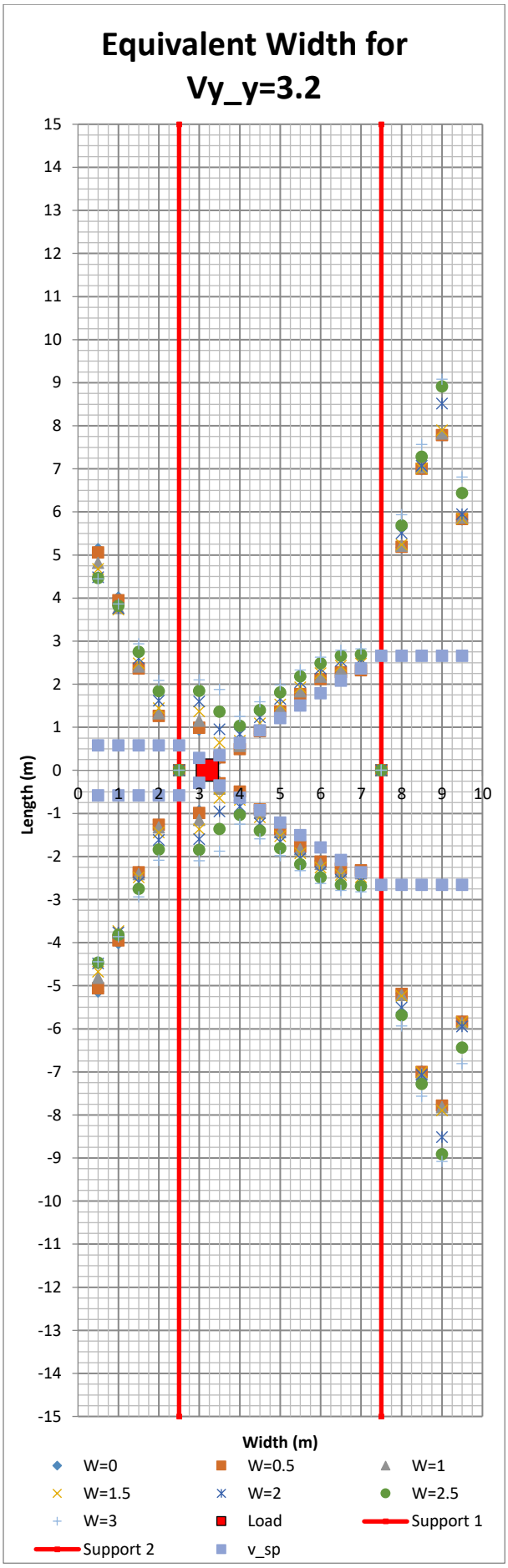

Equivalent Width for My_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

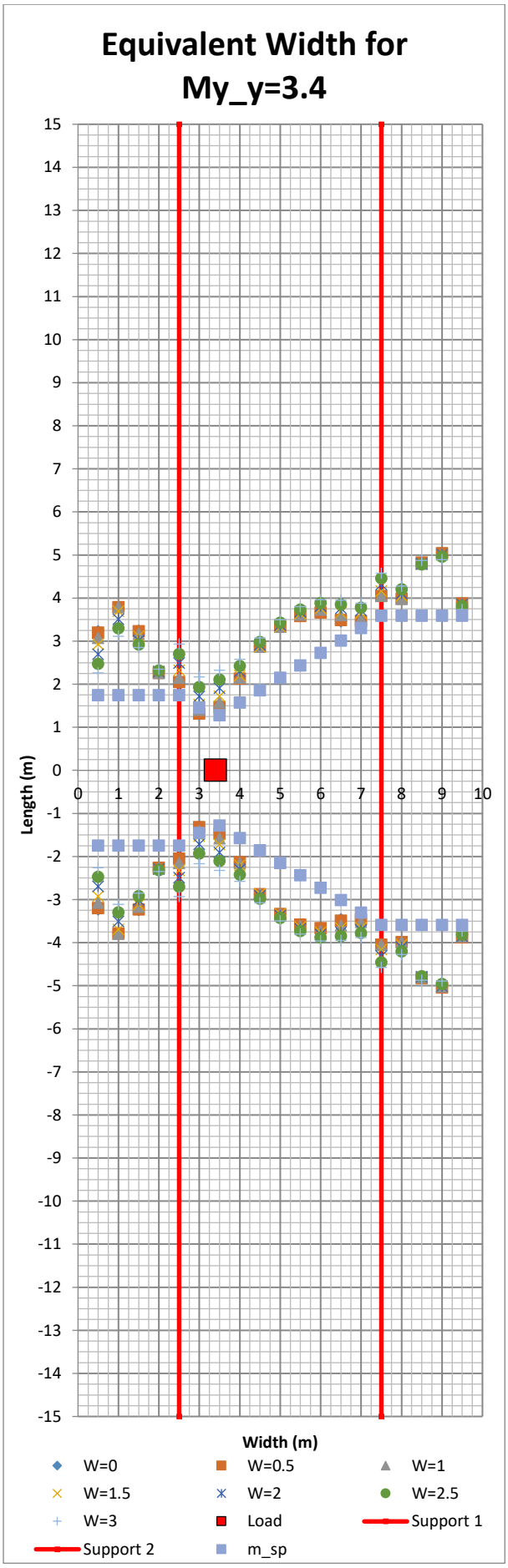

Equivalent Width for Vy_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

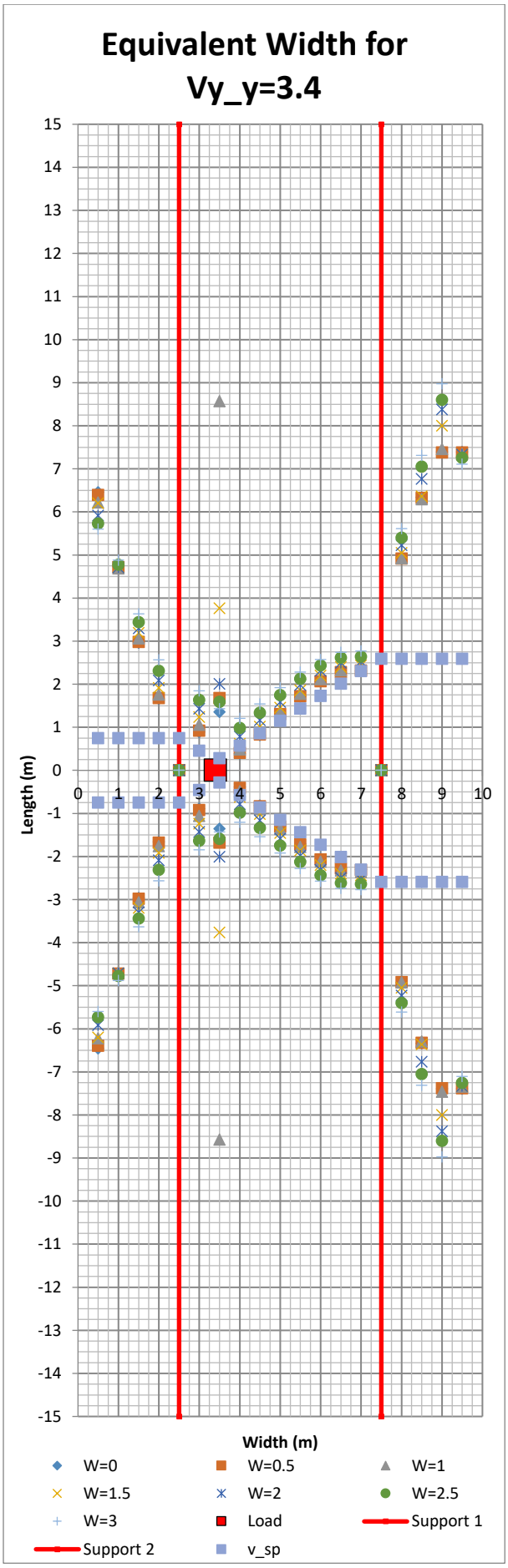

Equivalent Width for My_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

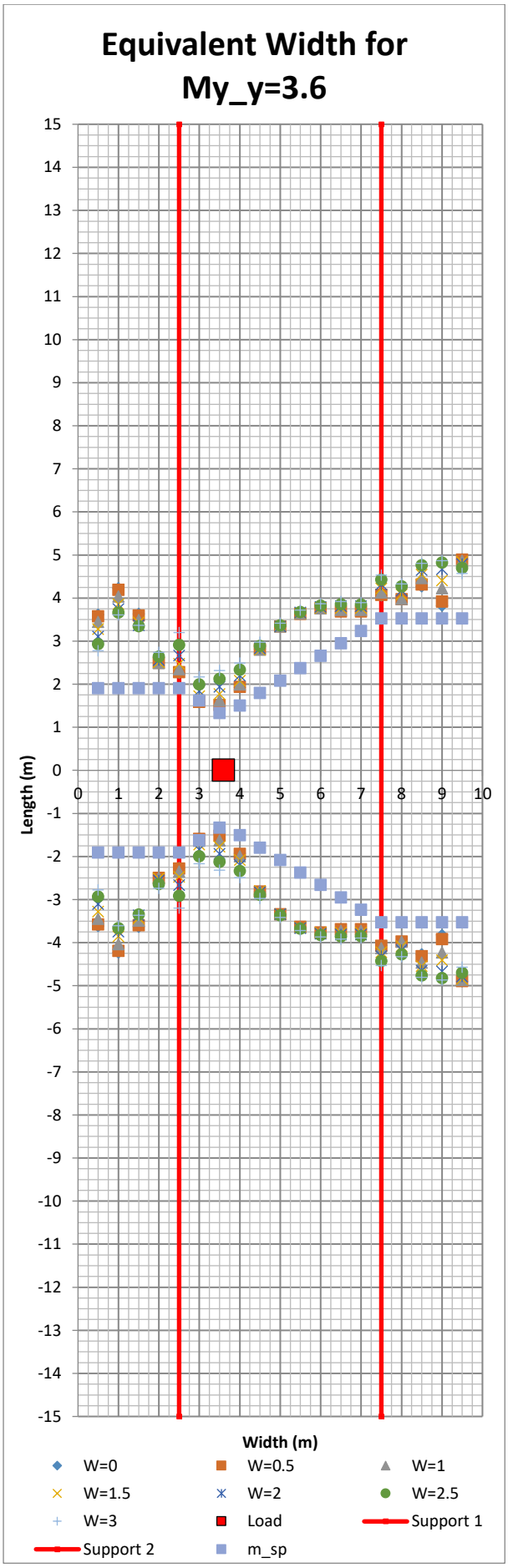

Equivalent Width for Vy_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

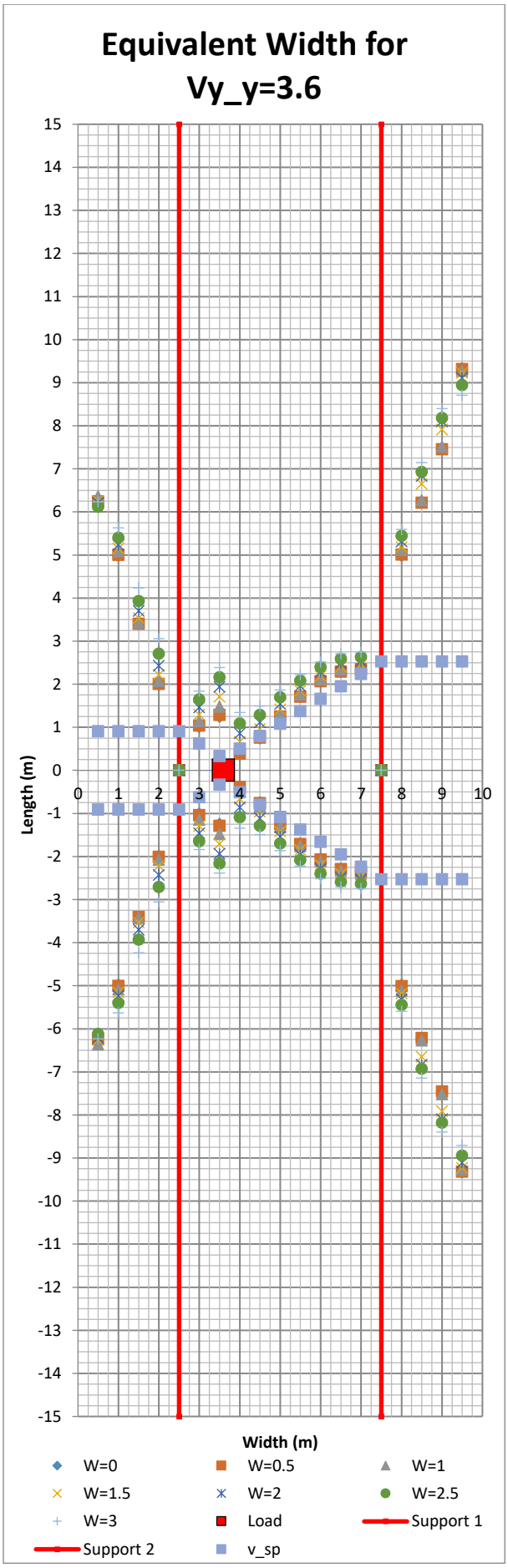

Equivalent Width for My_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

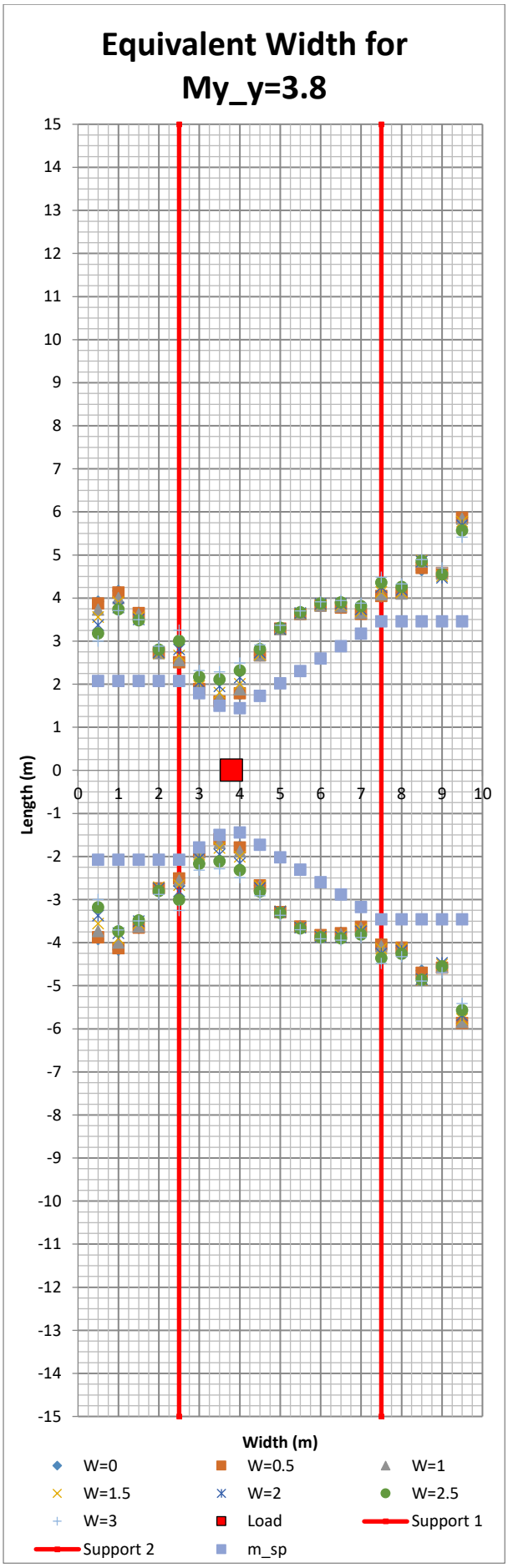

Equivalent Width for Vy_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

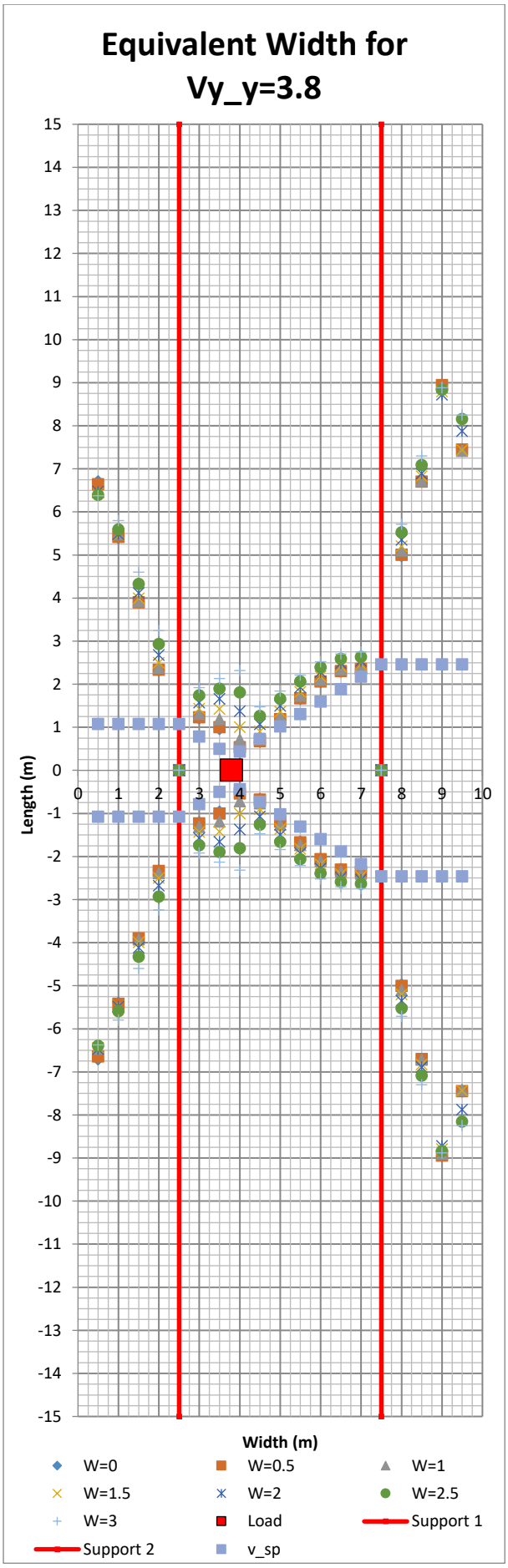

Equivalent Width for My_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

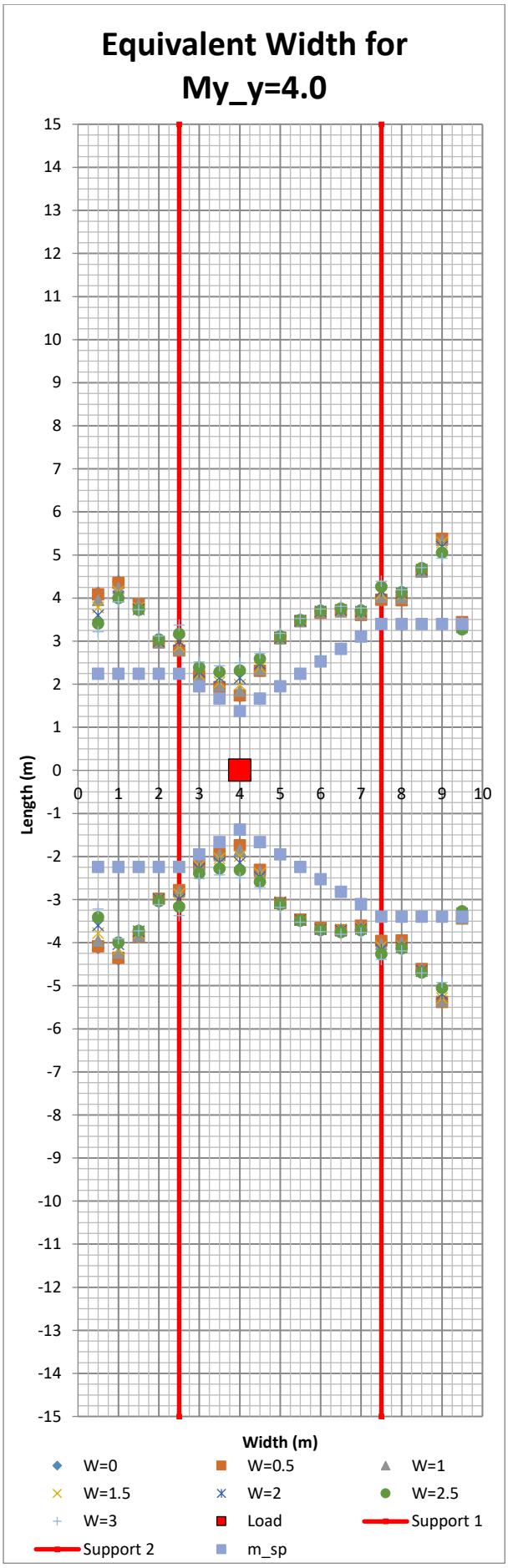

Equivalent Width for Vy_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

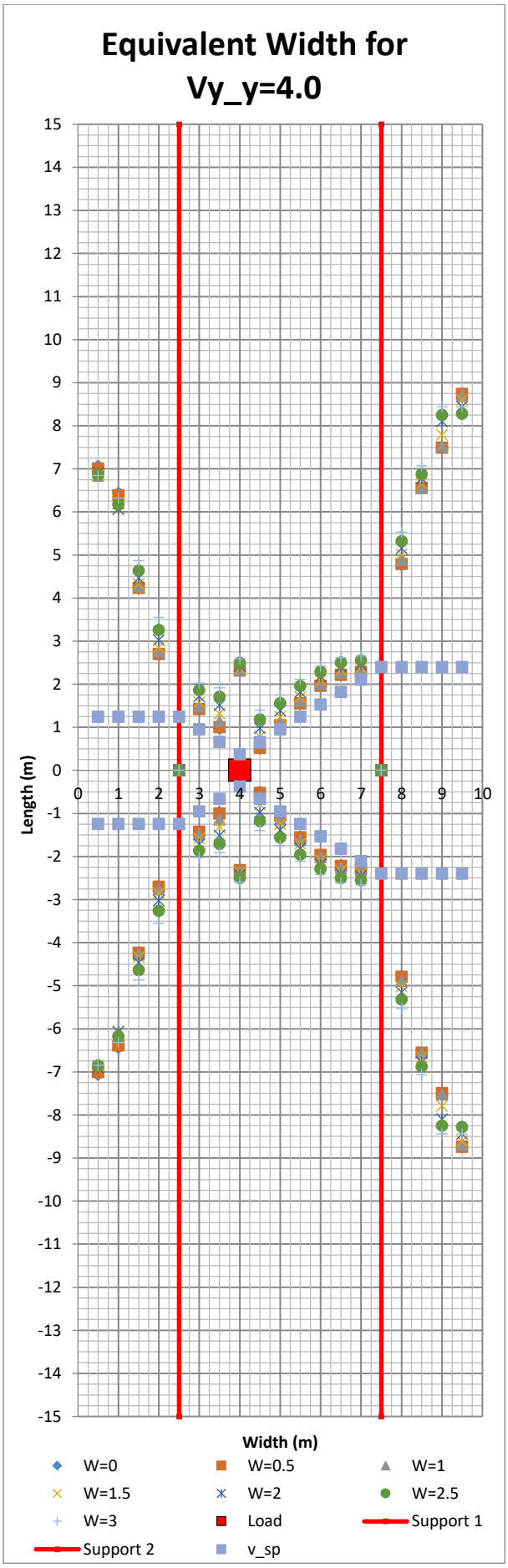

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

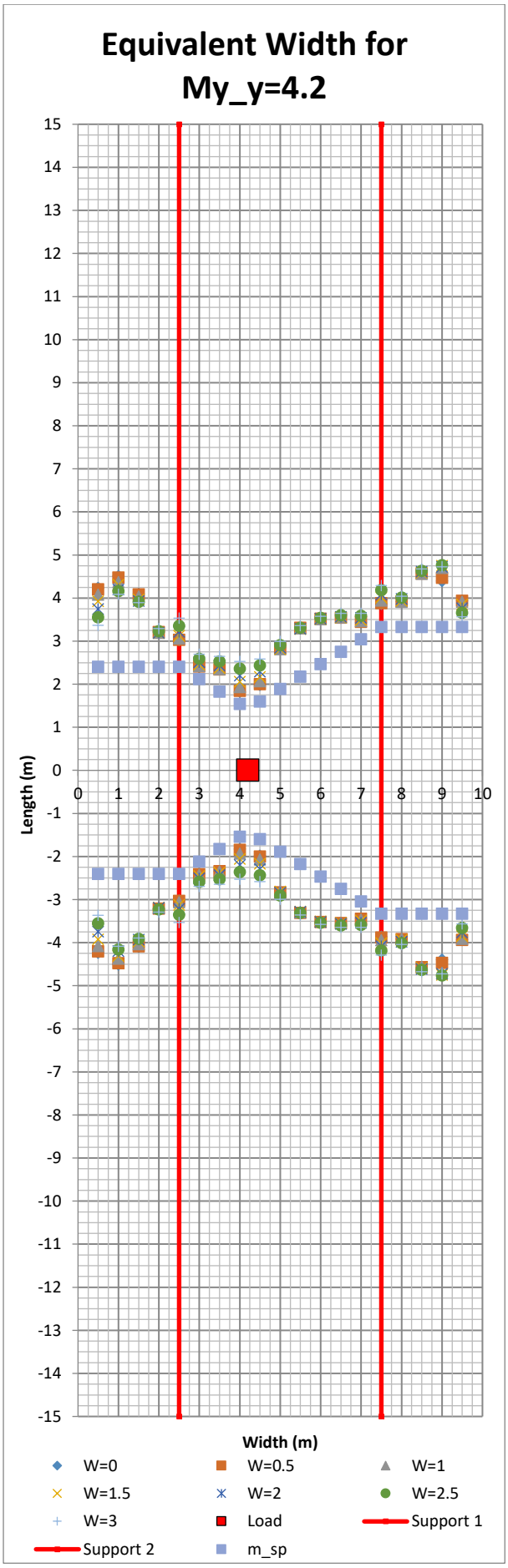

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

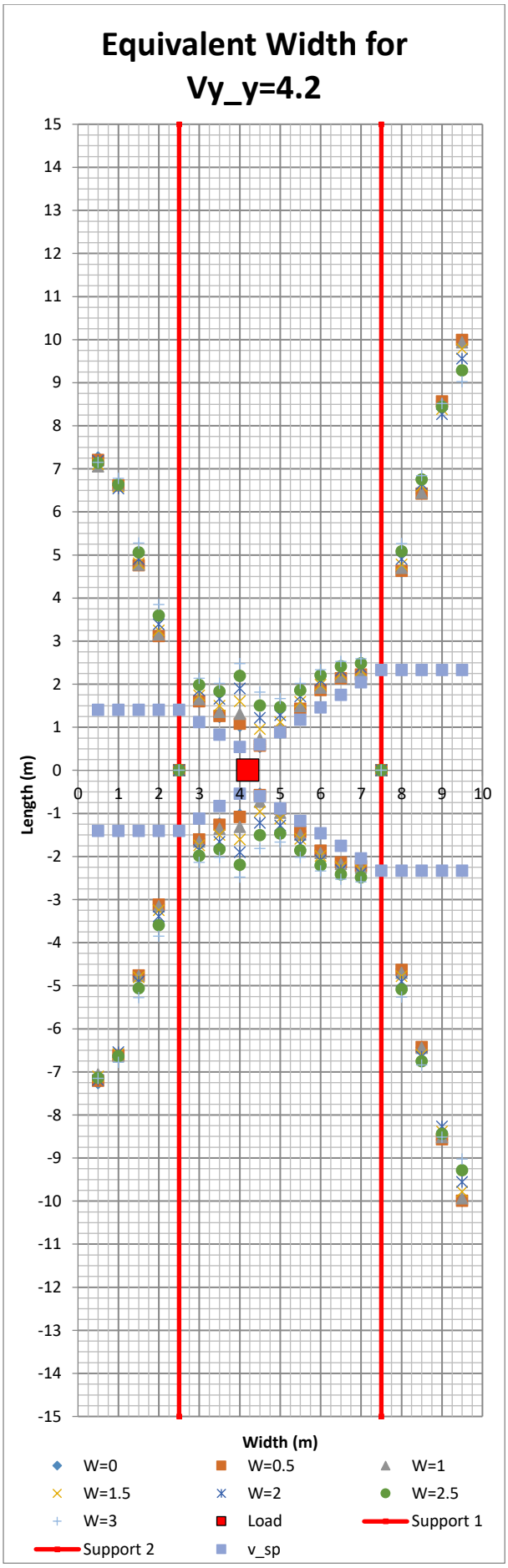

Equivalent Width for My_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

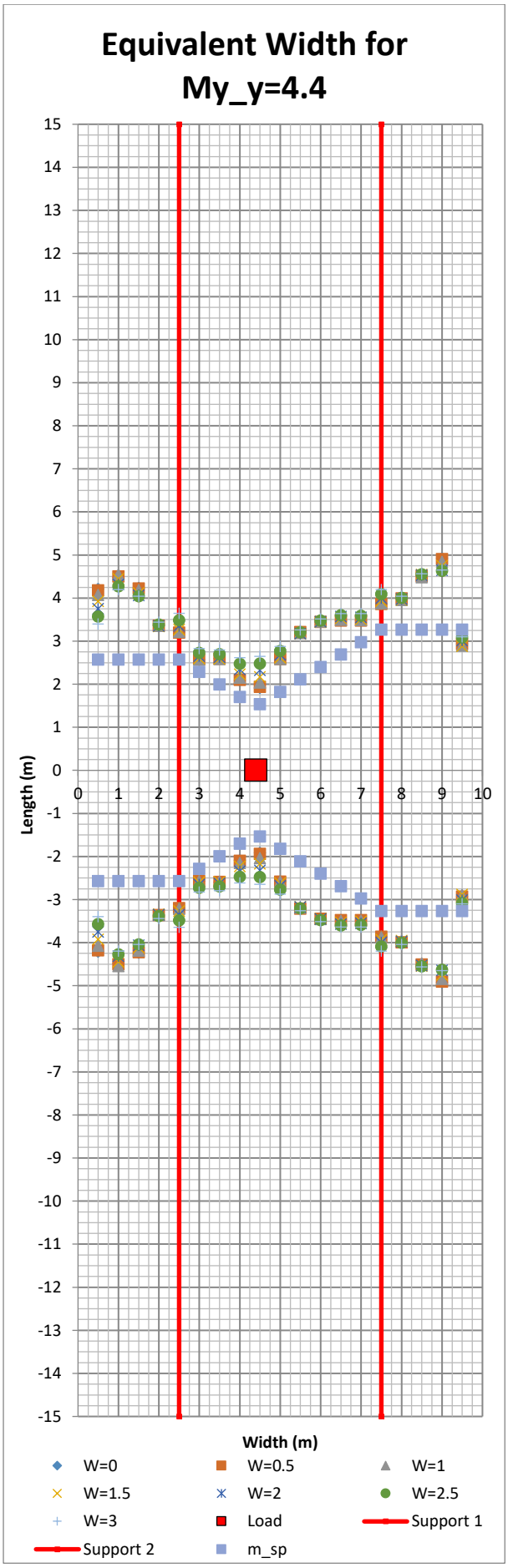

Equivalent Width for Vy_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

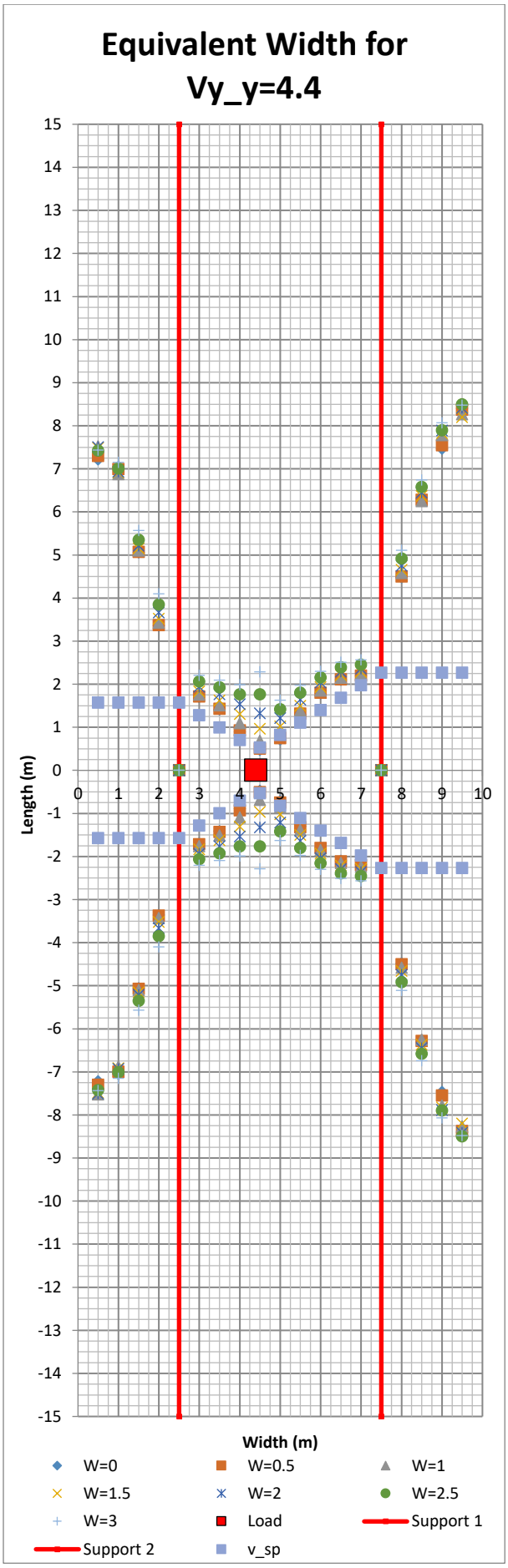

Equivalent Width for My_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

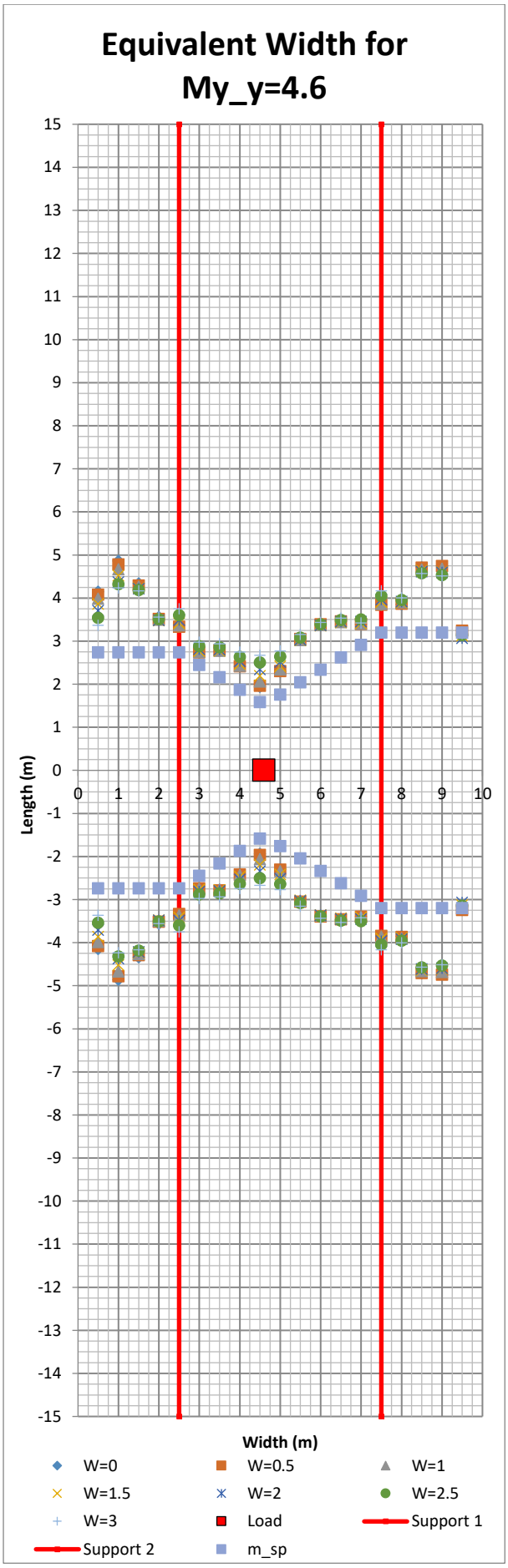

Equivalent Width for Vy_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

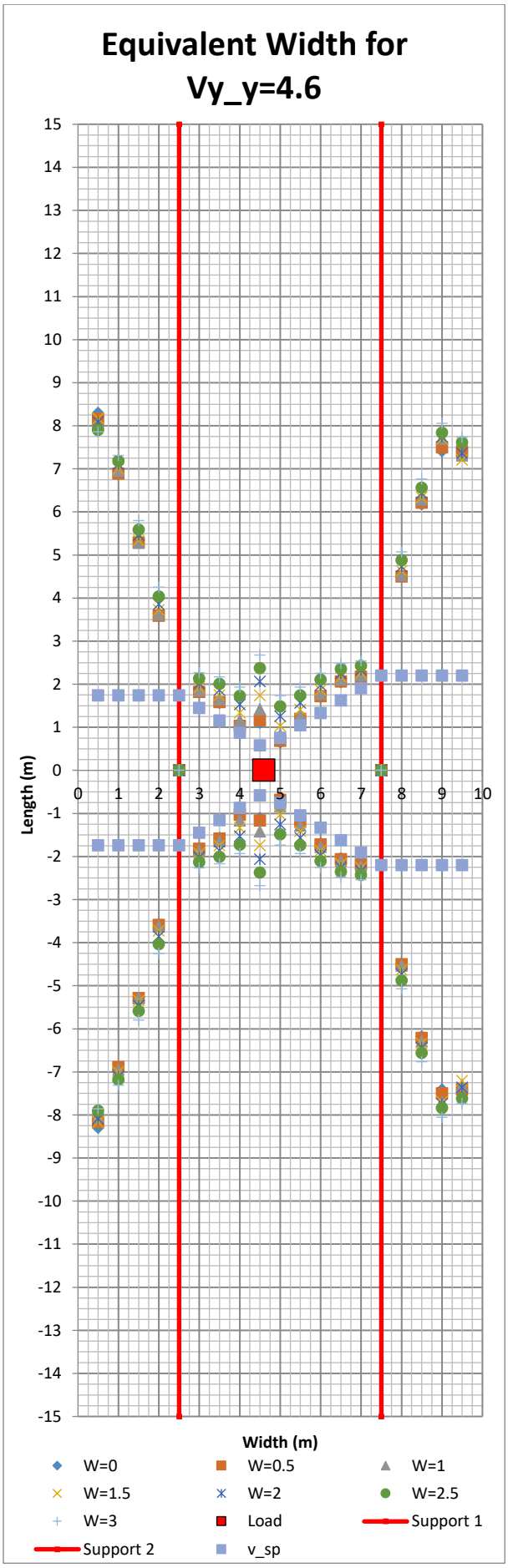

Equivalent Width for My_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

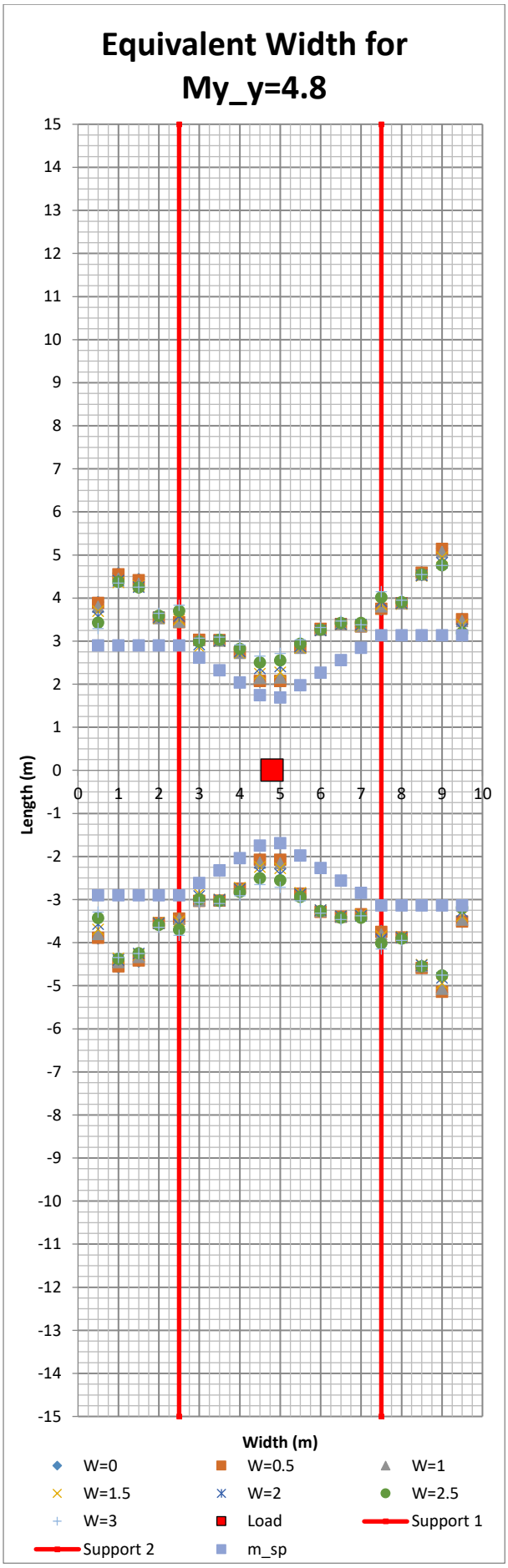

Equivalent Width for Vy_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

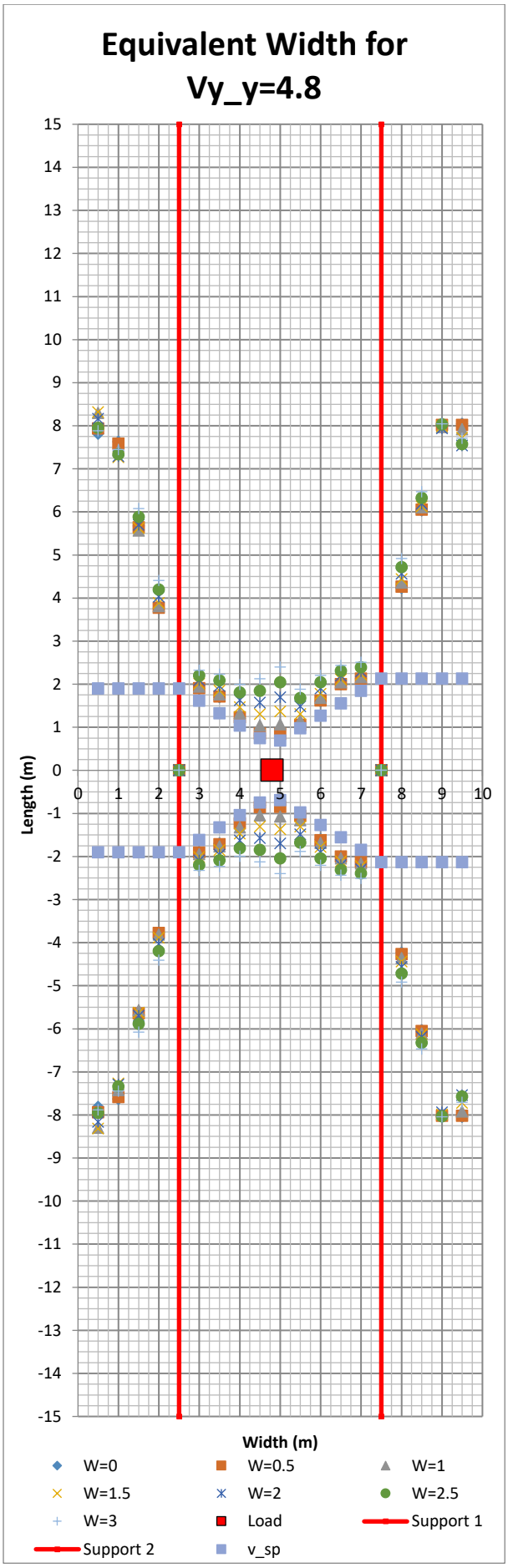

Equivalent Width for My_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

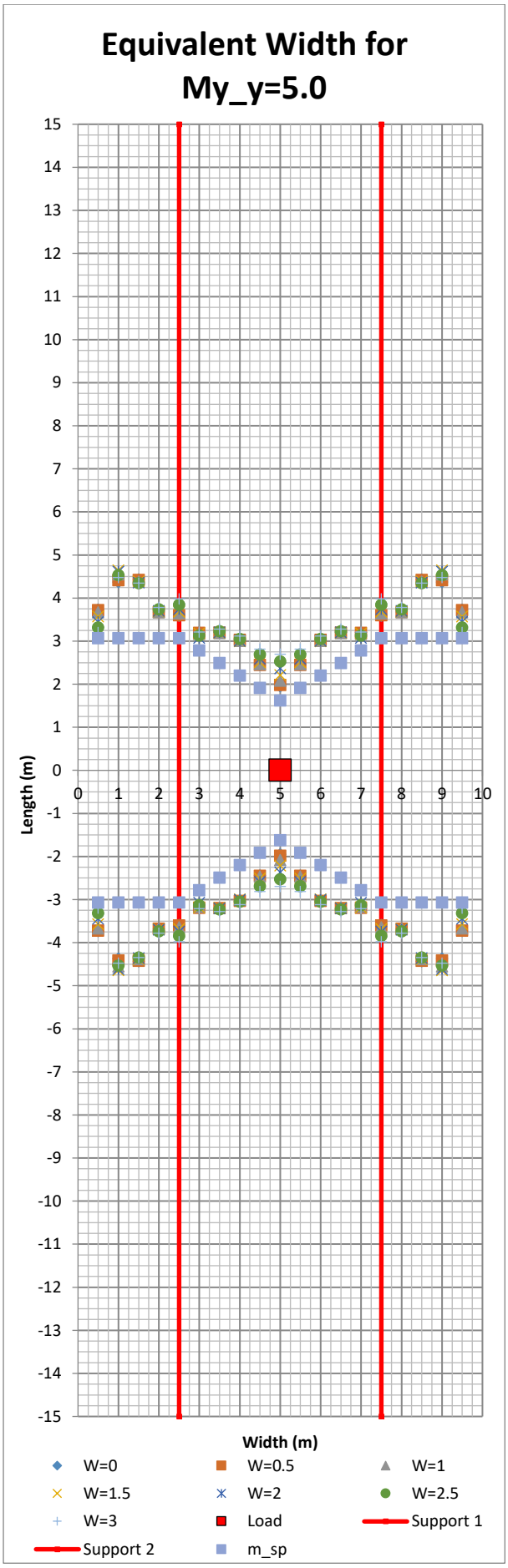

Equivalent Width for Vy_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

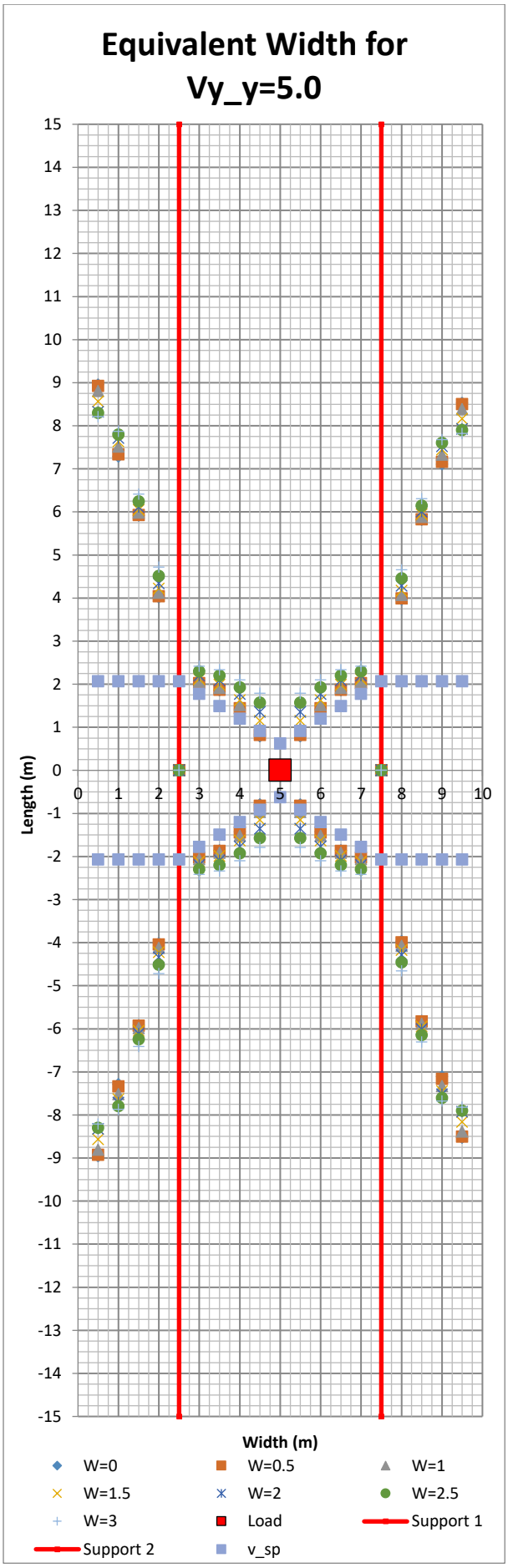

Equivalent Width for My_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

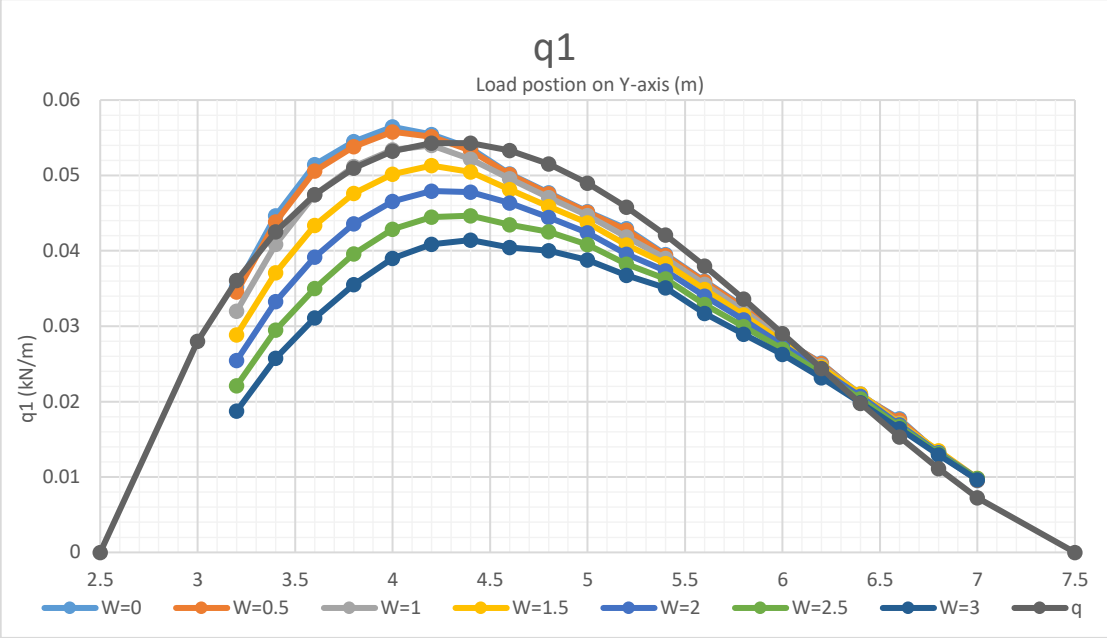

Equivalent Width for My_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

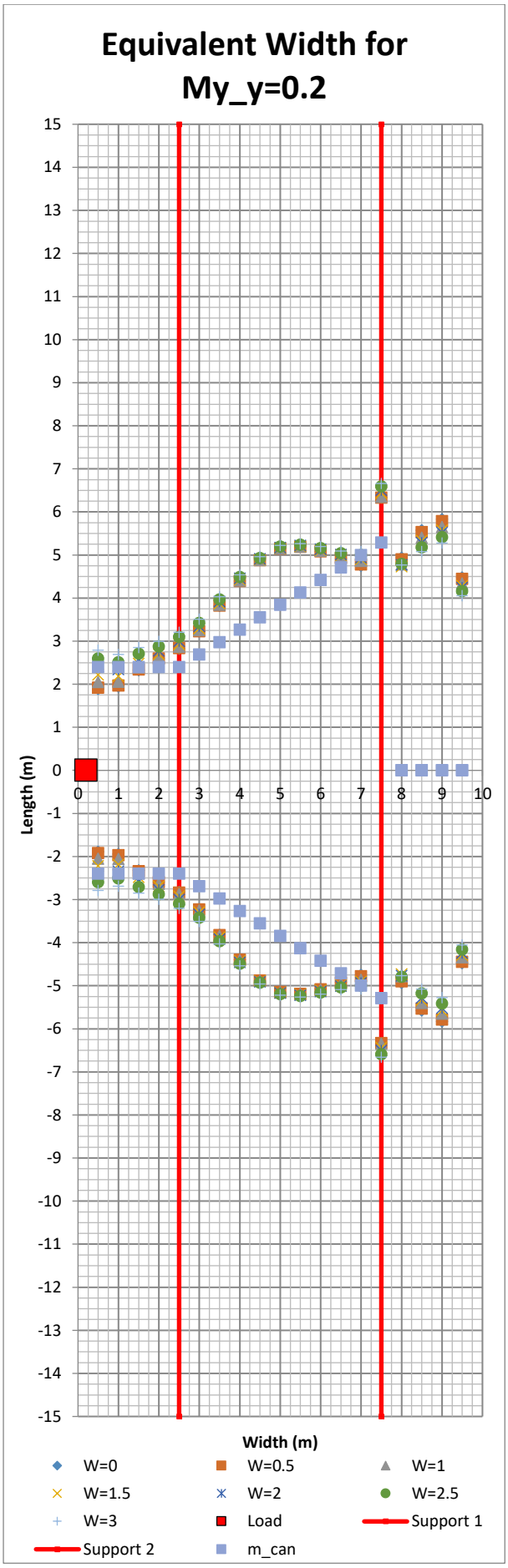

Equivalent Width for Vy_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

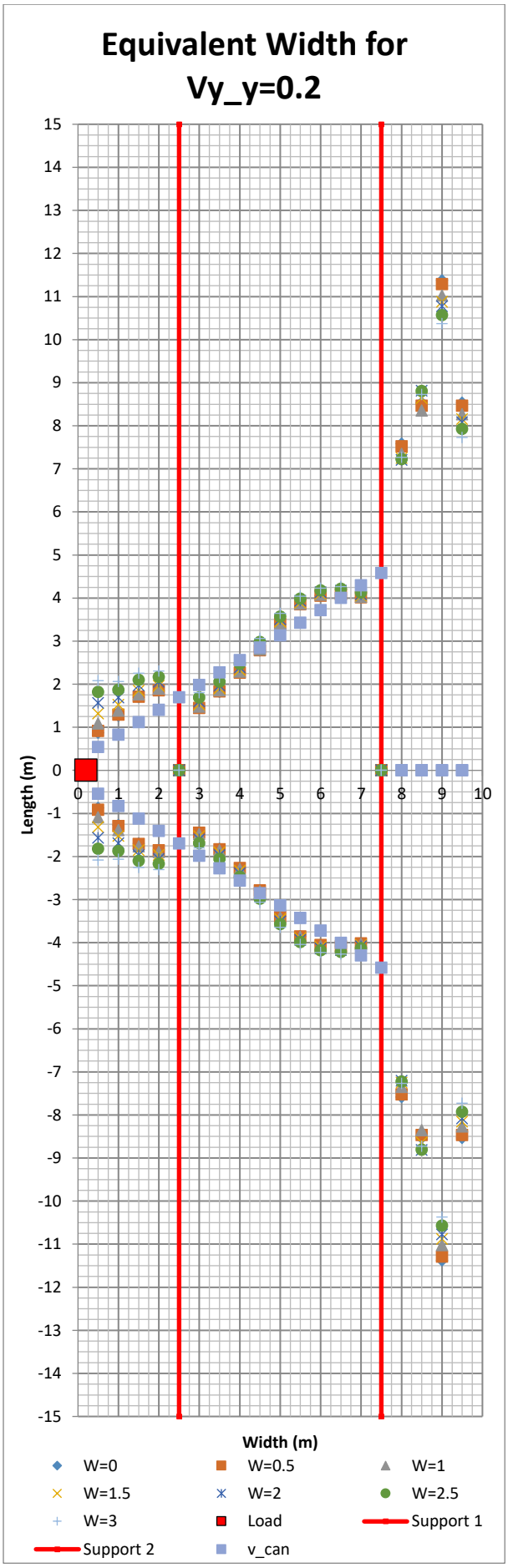

Equivalent Width for My_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

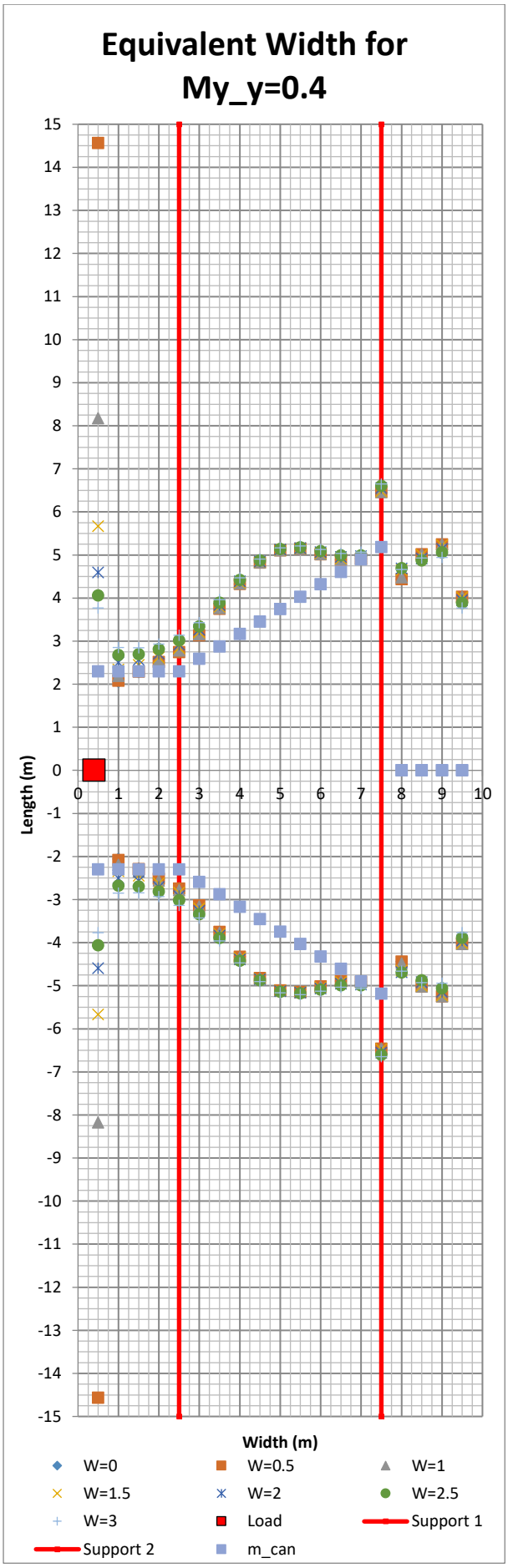

Equivalent Width for Vy_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

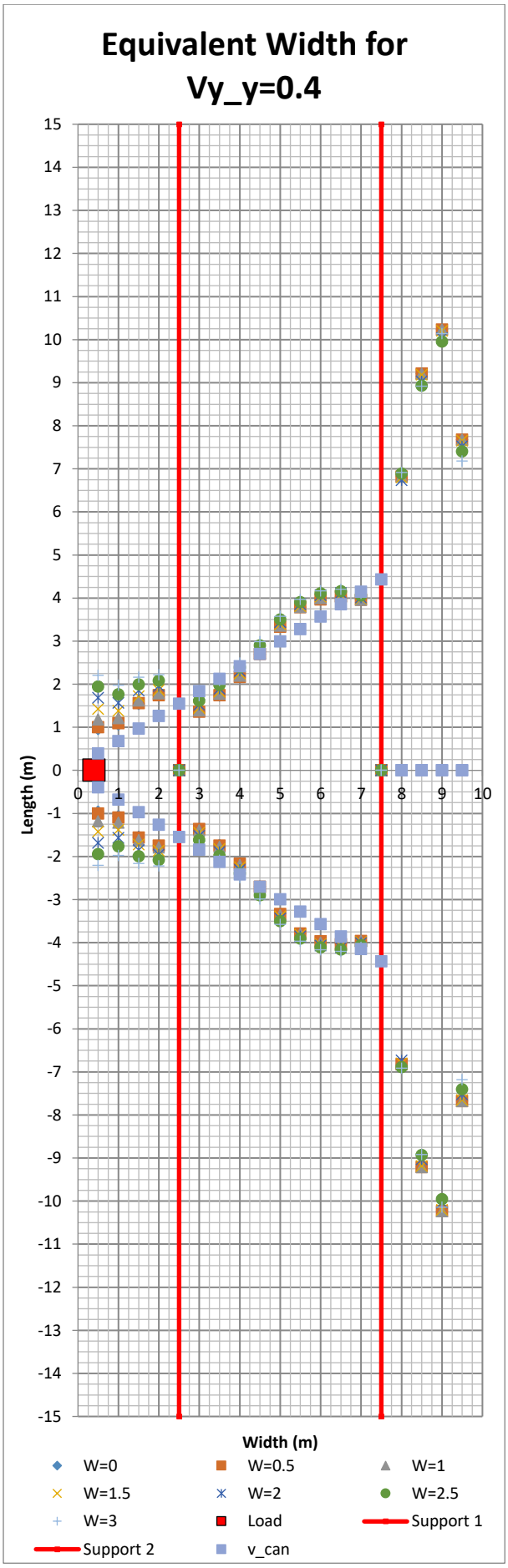
11.7 Seventh Variation $t = [0.15, 0.3, 0.45]m$

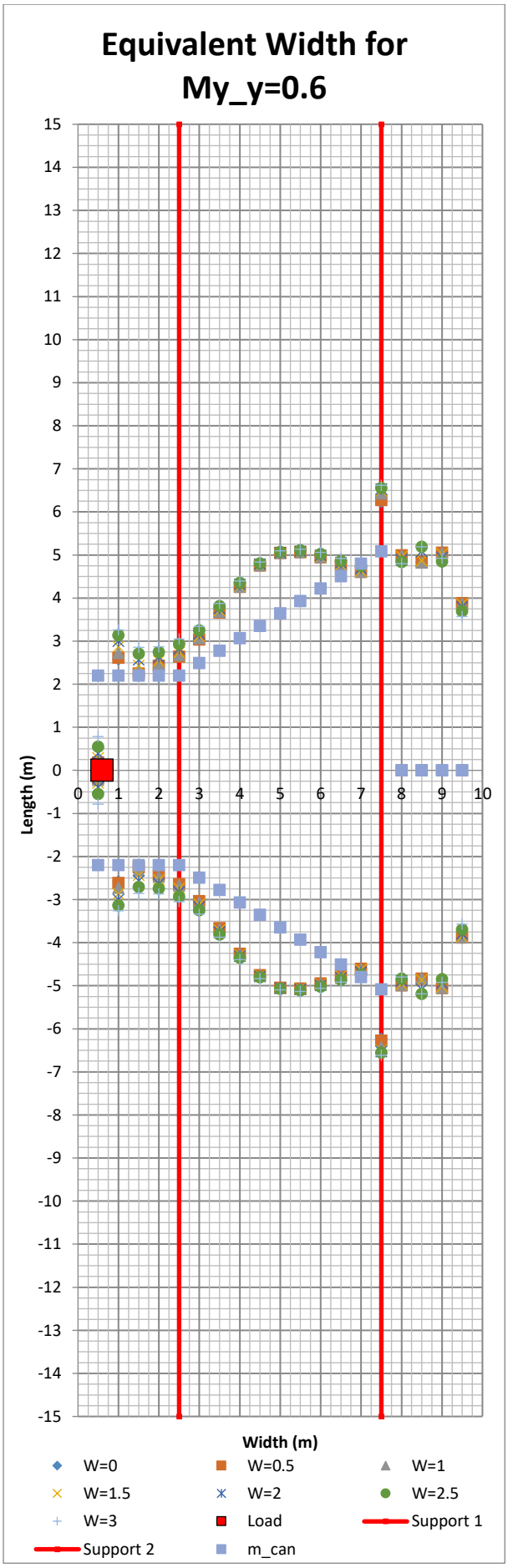


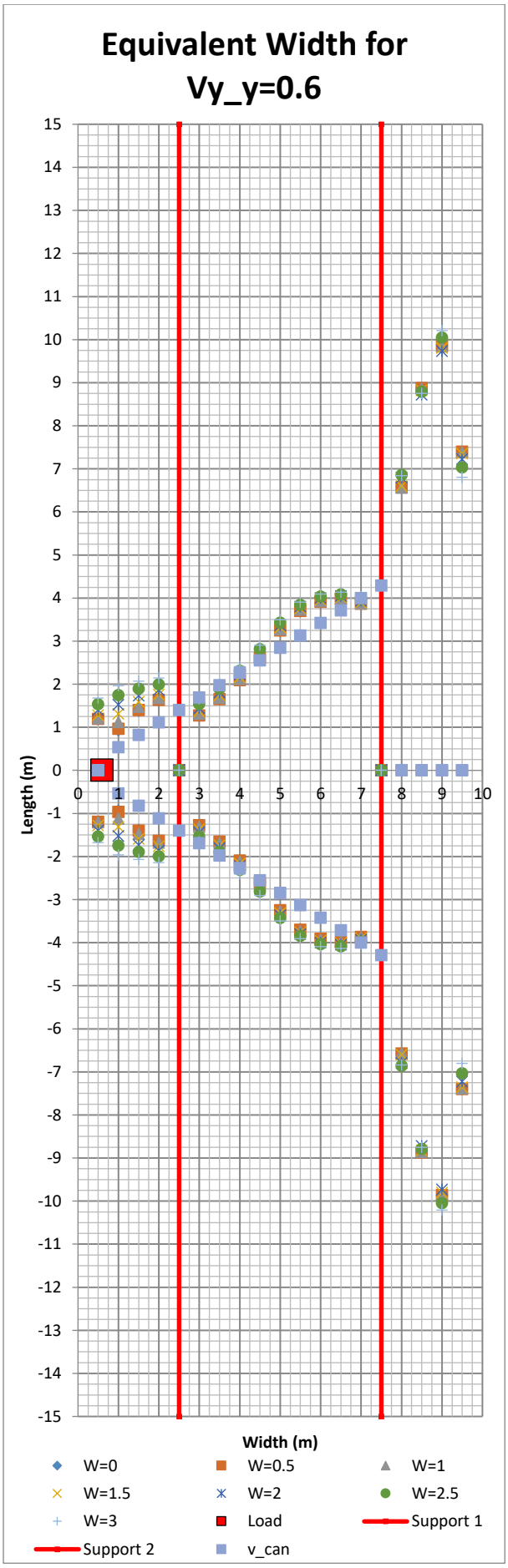

Equivalent Width for My_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

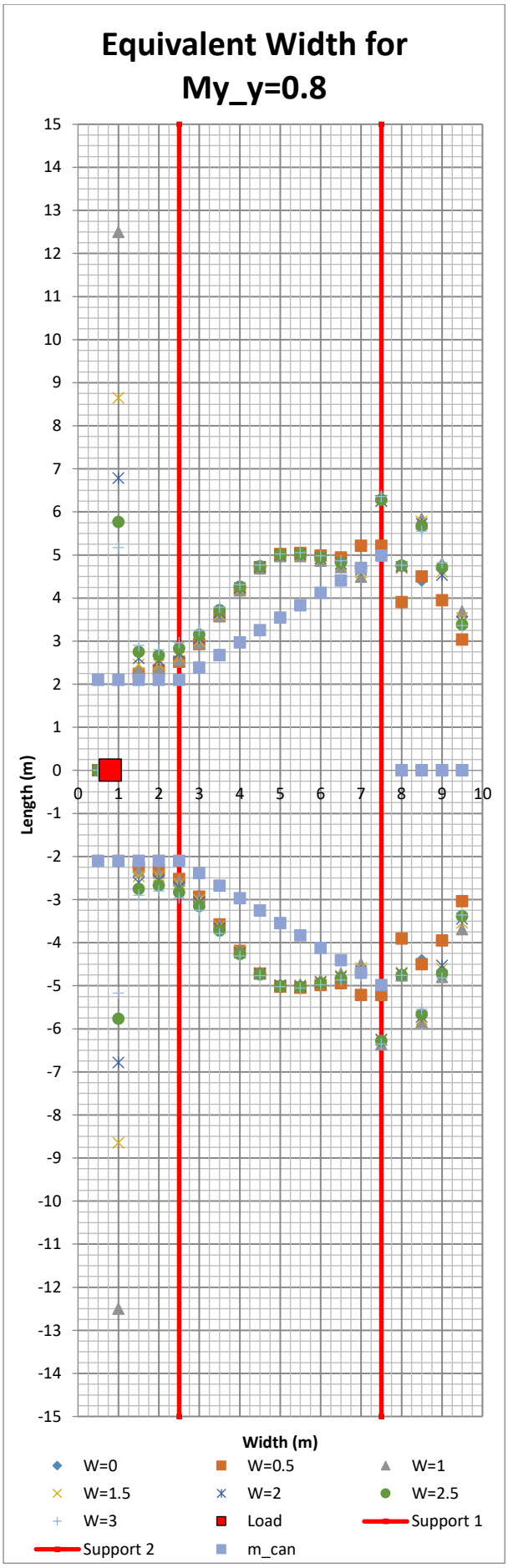

Equivalent Width for Vy_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

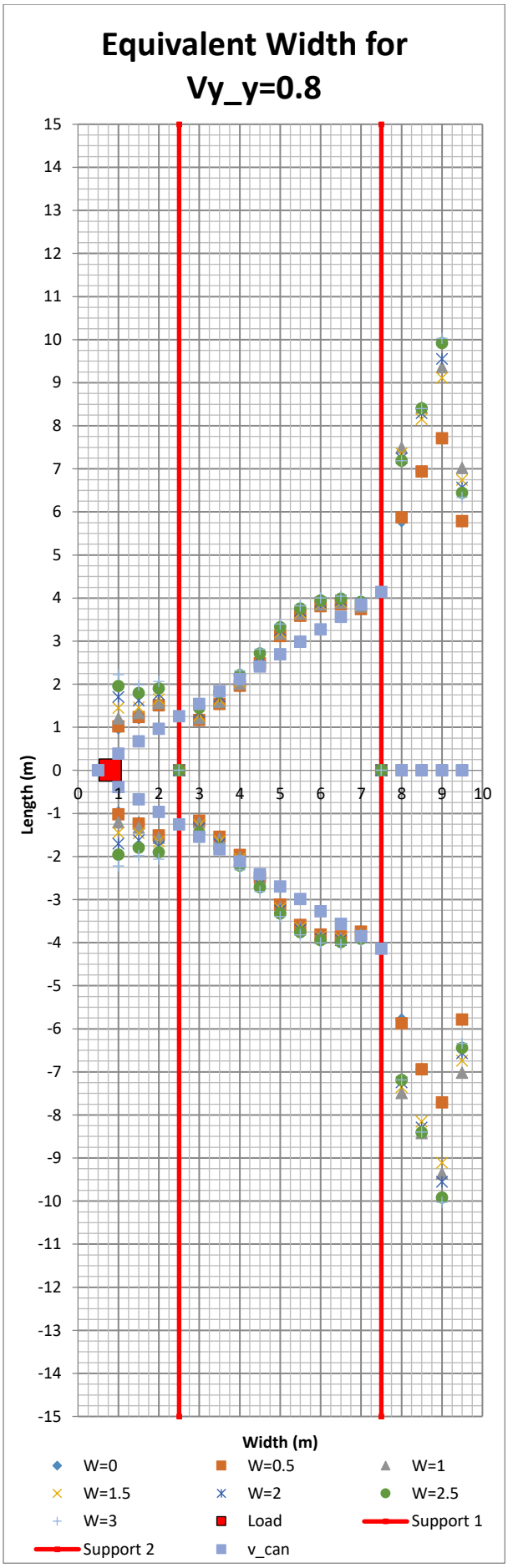

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

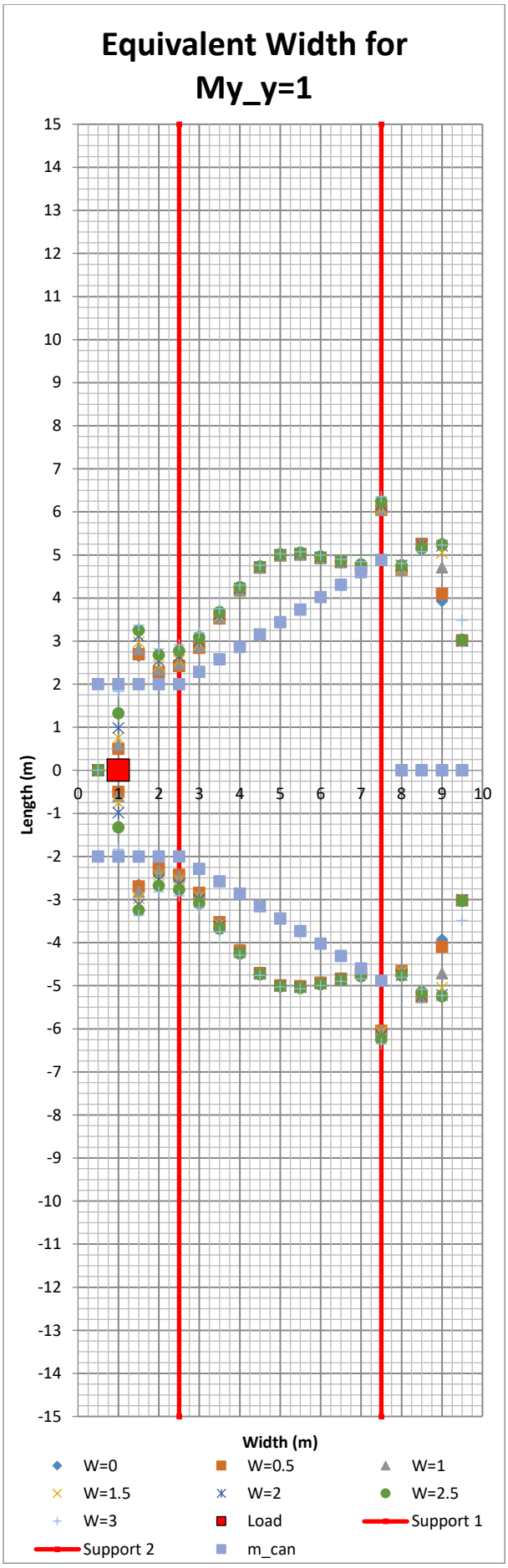

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

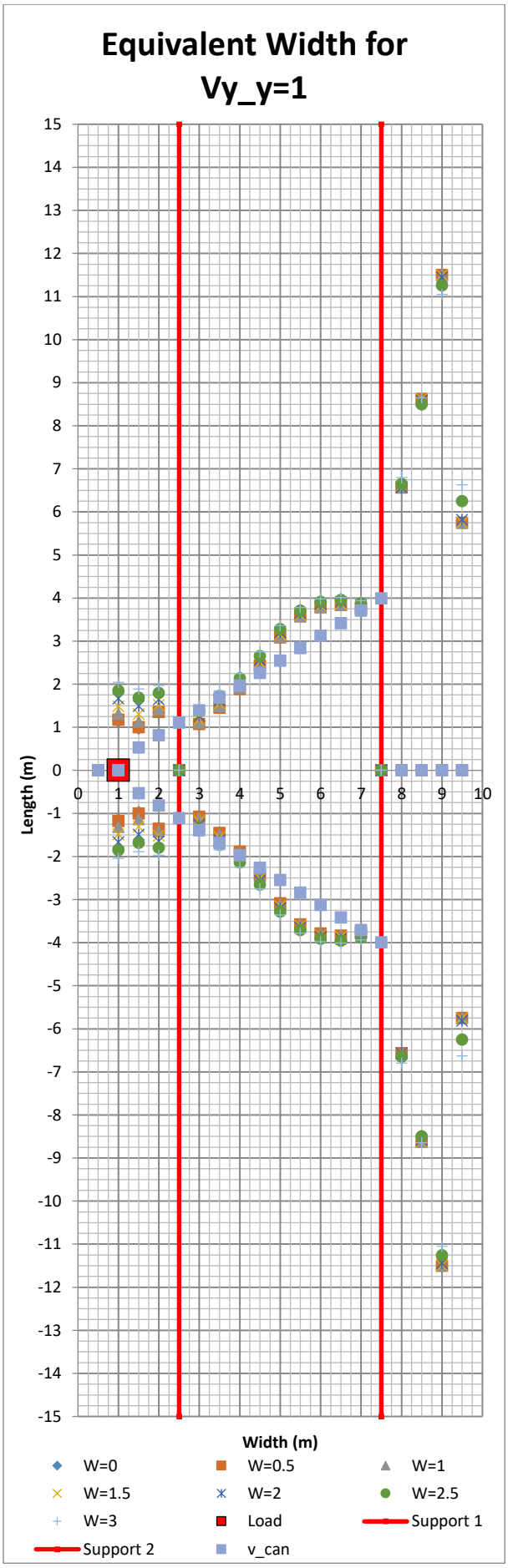

Equivalent Width for My_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

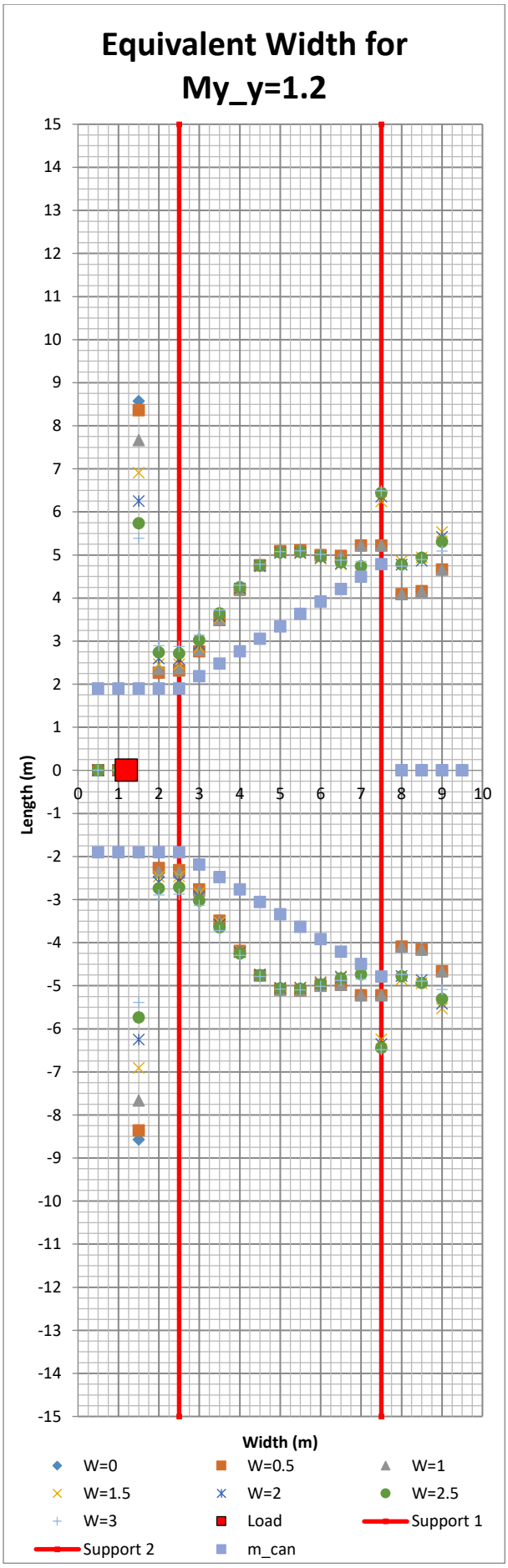

Equivalent Width for Vy_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

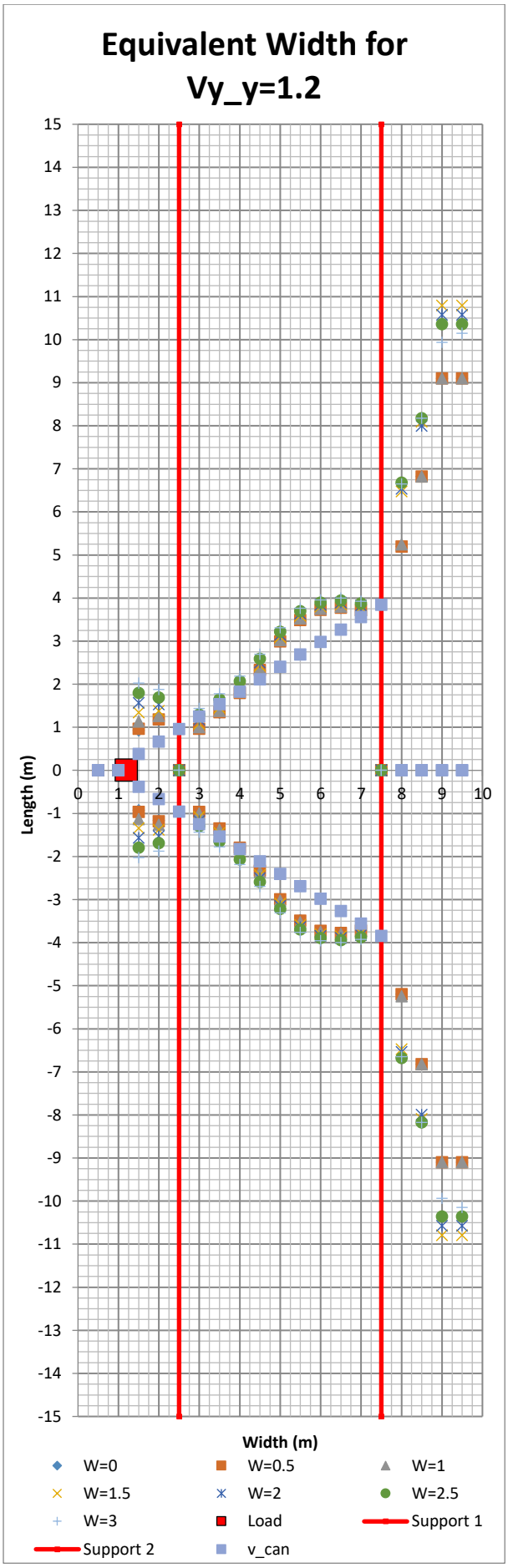

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

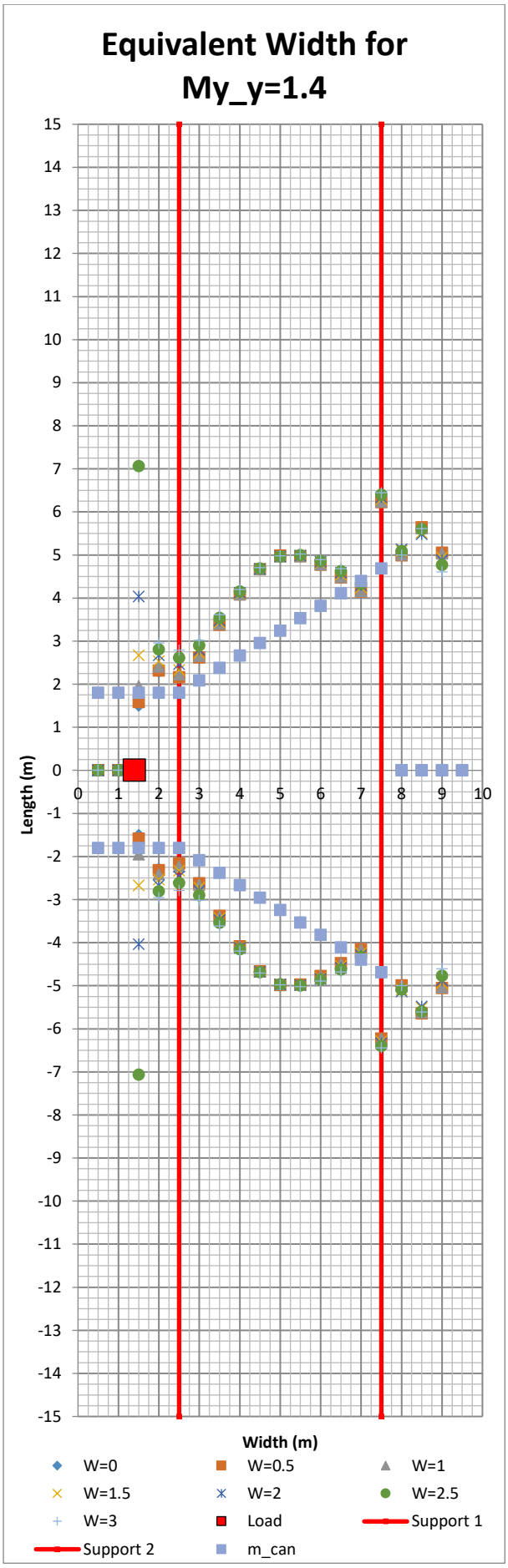

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

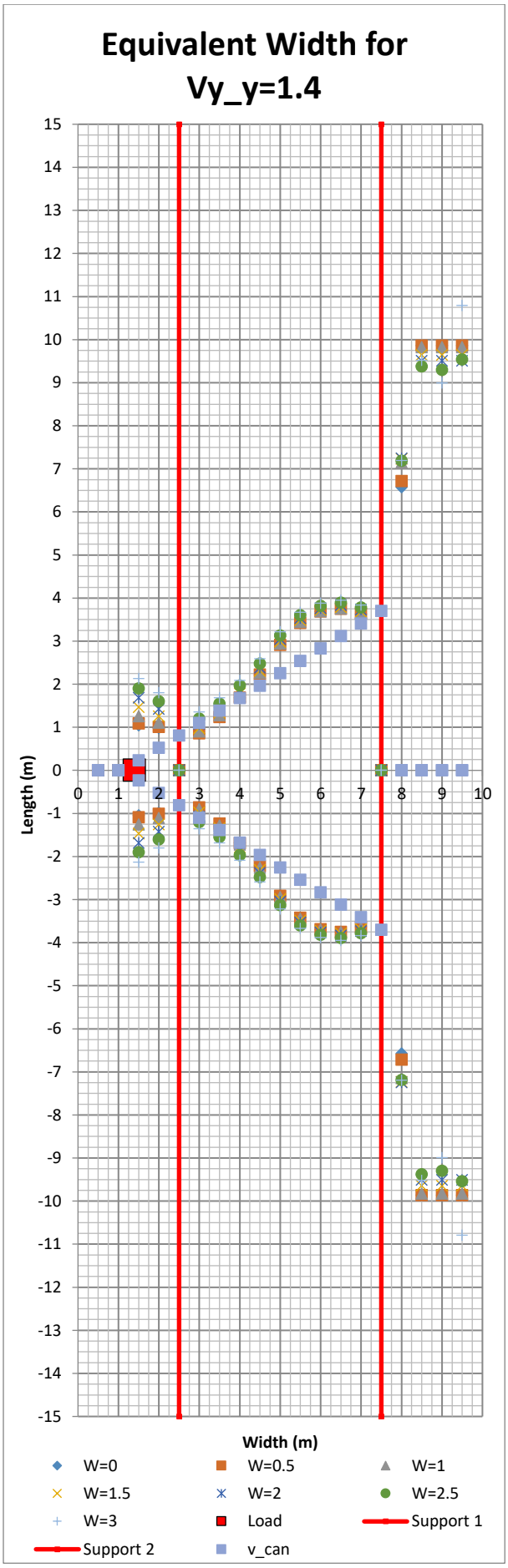

Equivalent Width for My_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

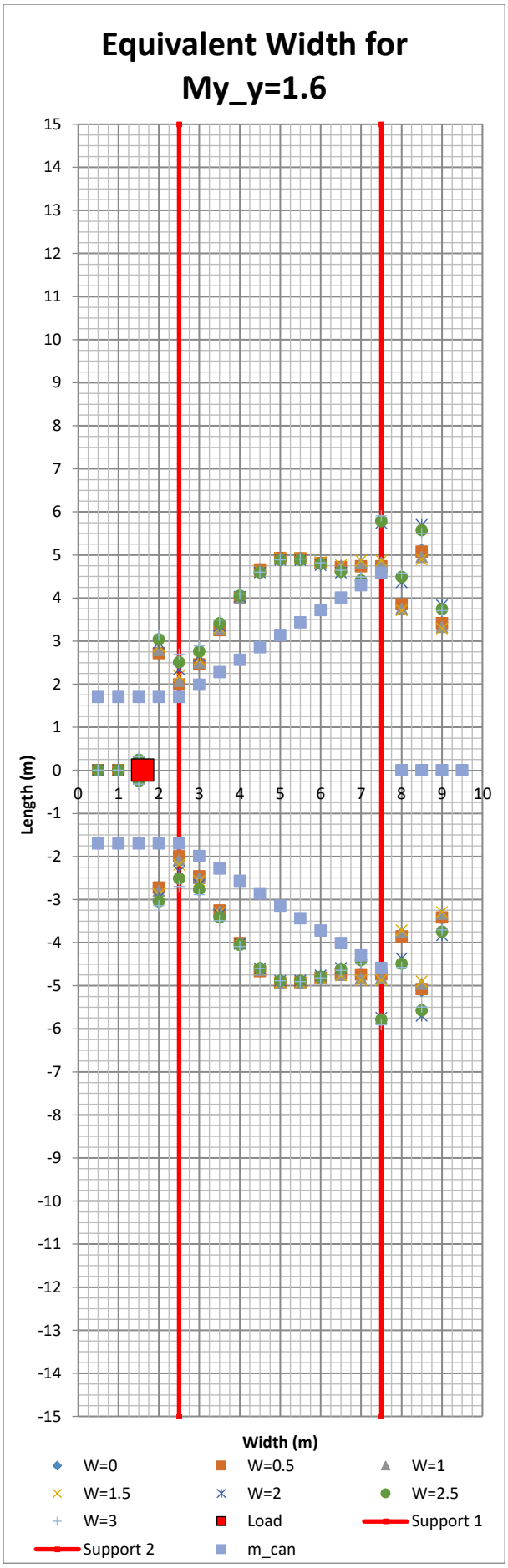

Equivalent Width for Vy_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

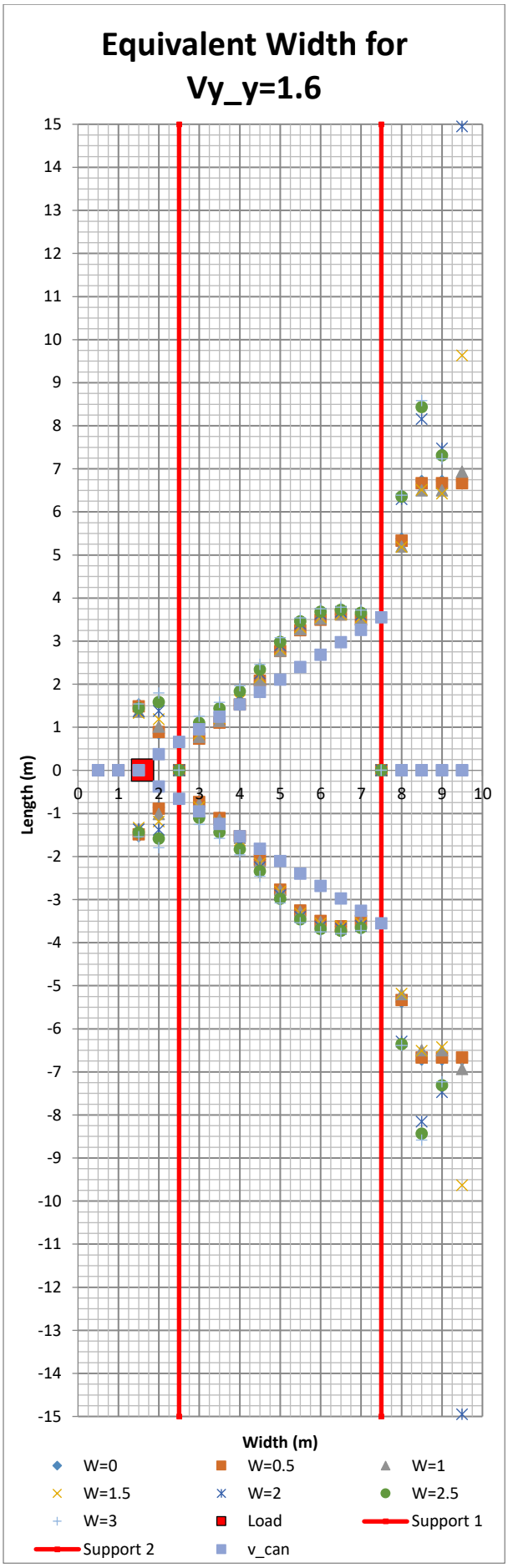

Equivalent Width for My_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

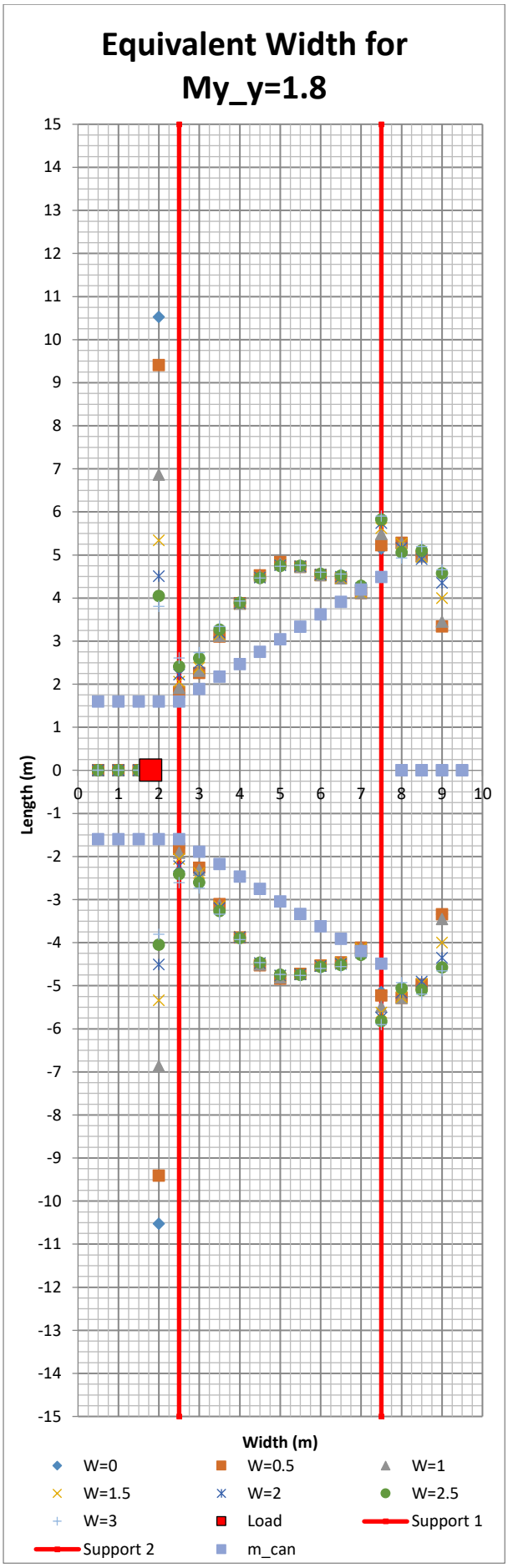

Equivalent Width for Vy_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

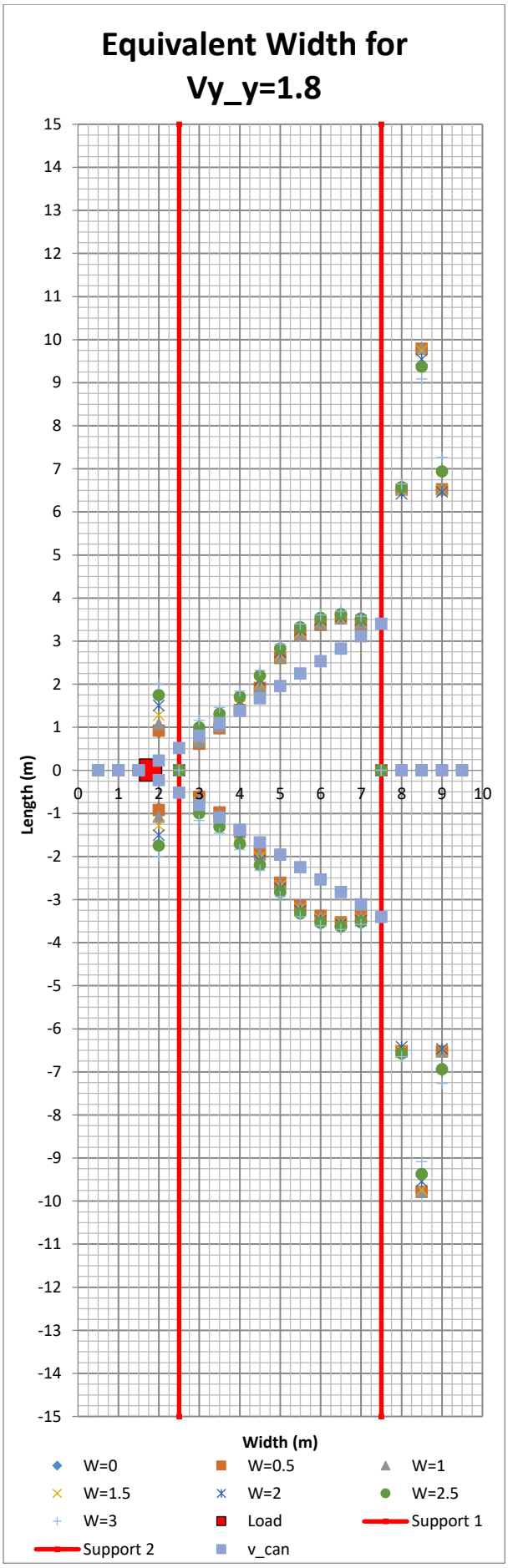

Equivalent Width for My_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

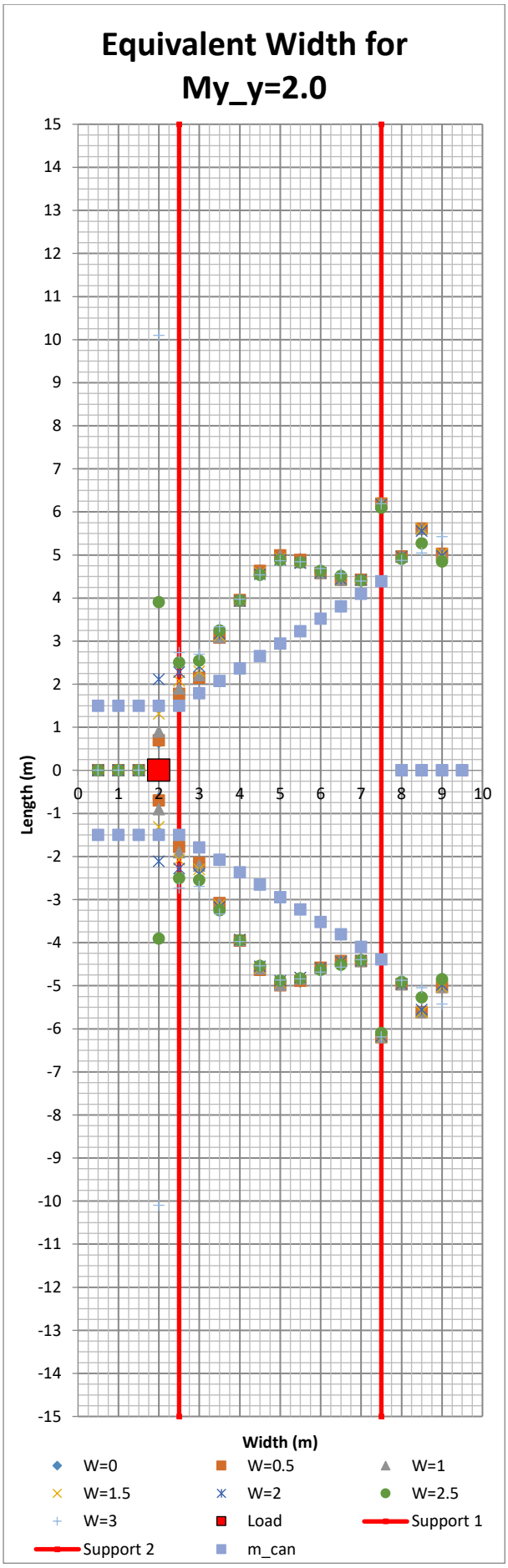

Equivalent Width for Vy_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

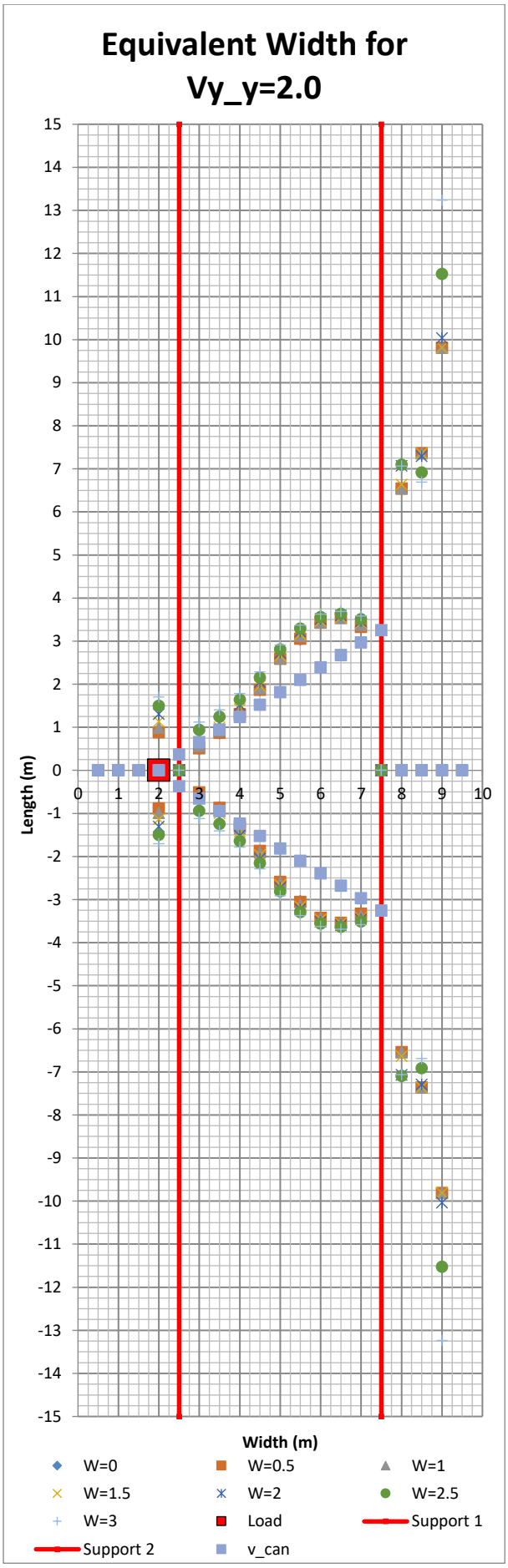

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

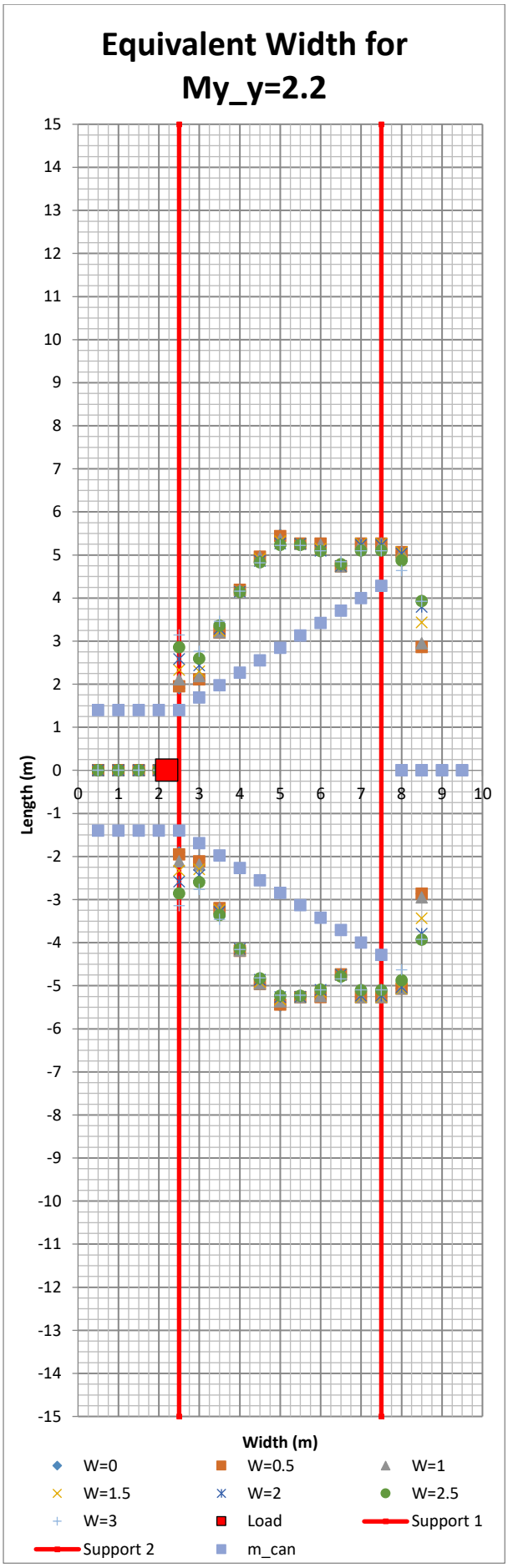

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

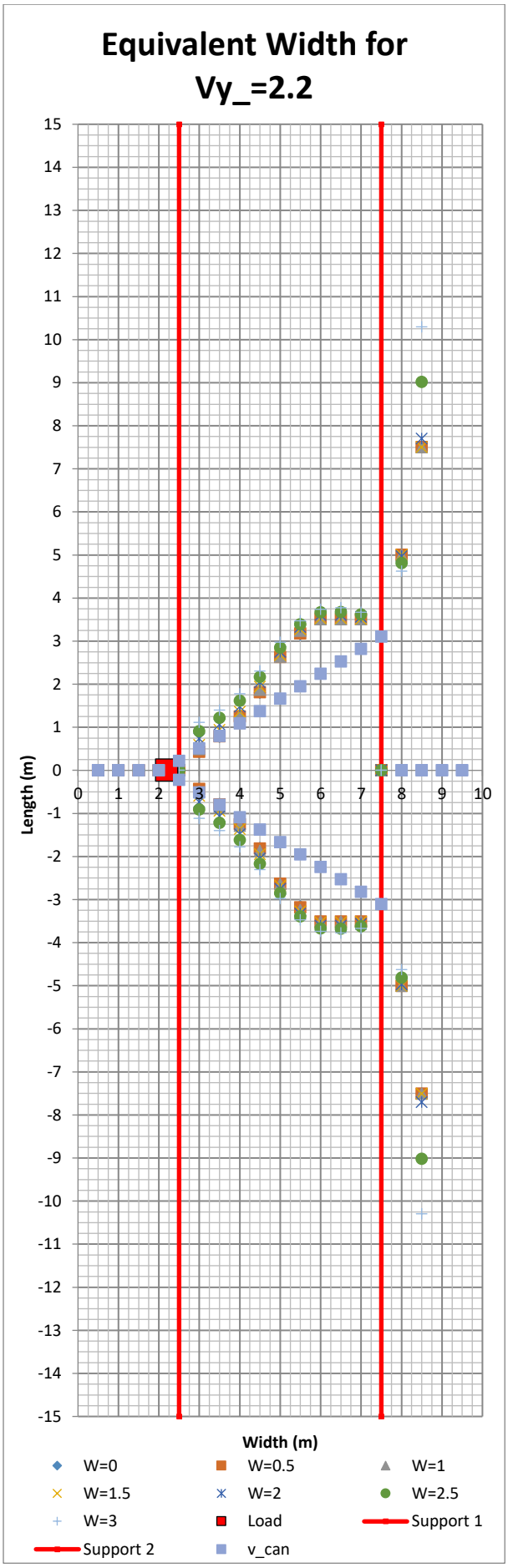

Equivalent Width for My_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

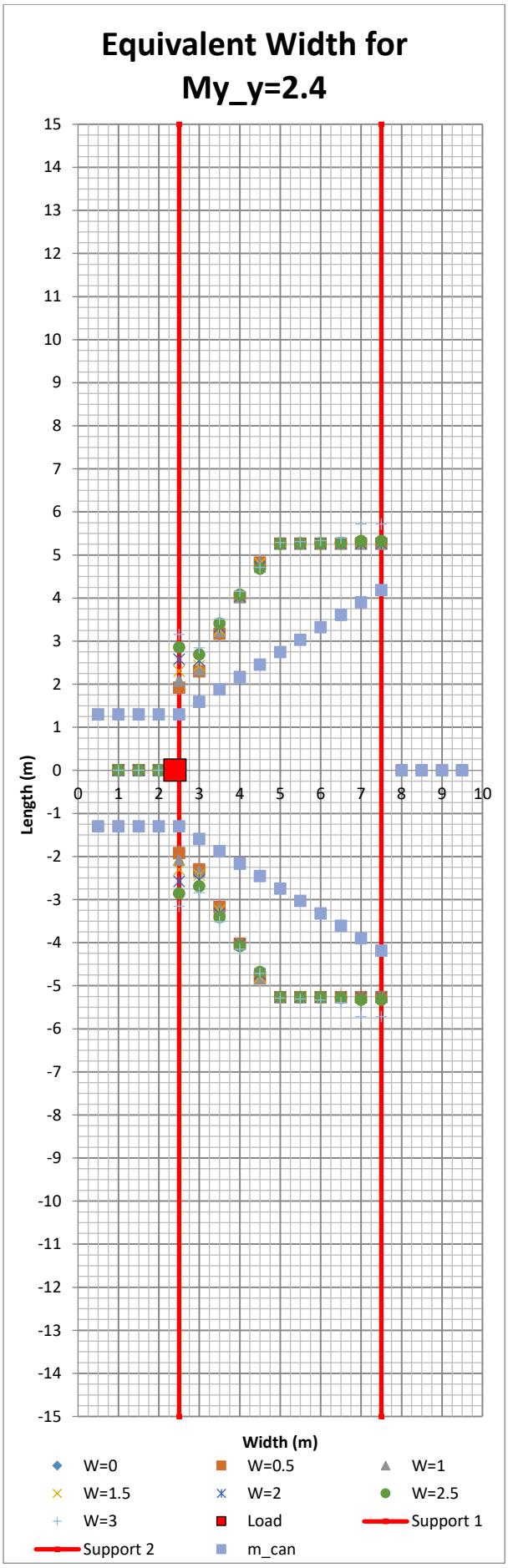

Equivalent Width for Vy_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

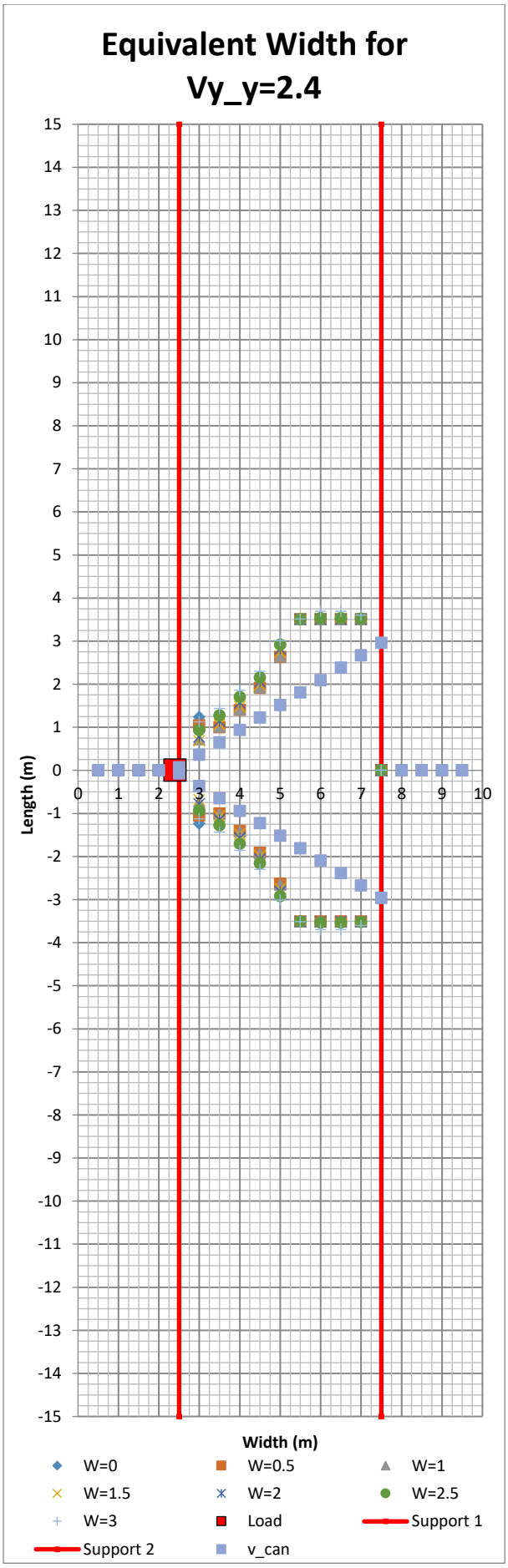

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

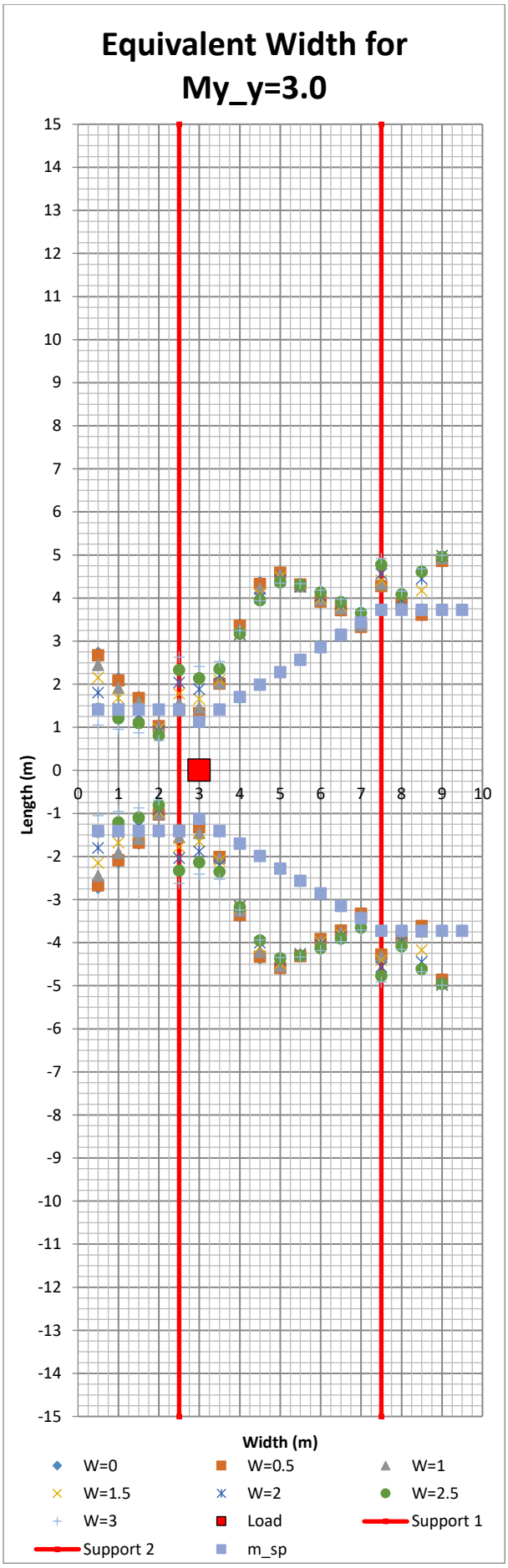

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

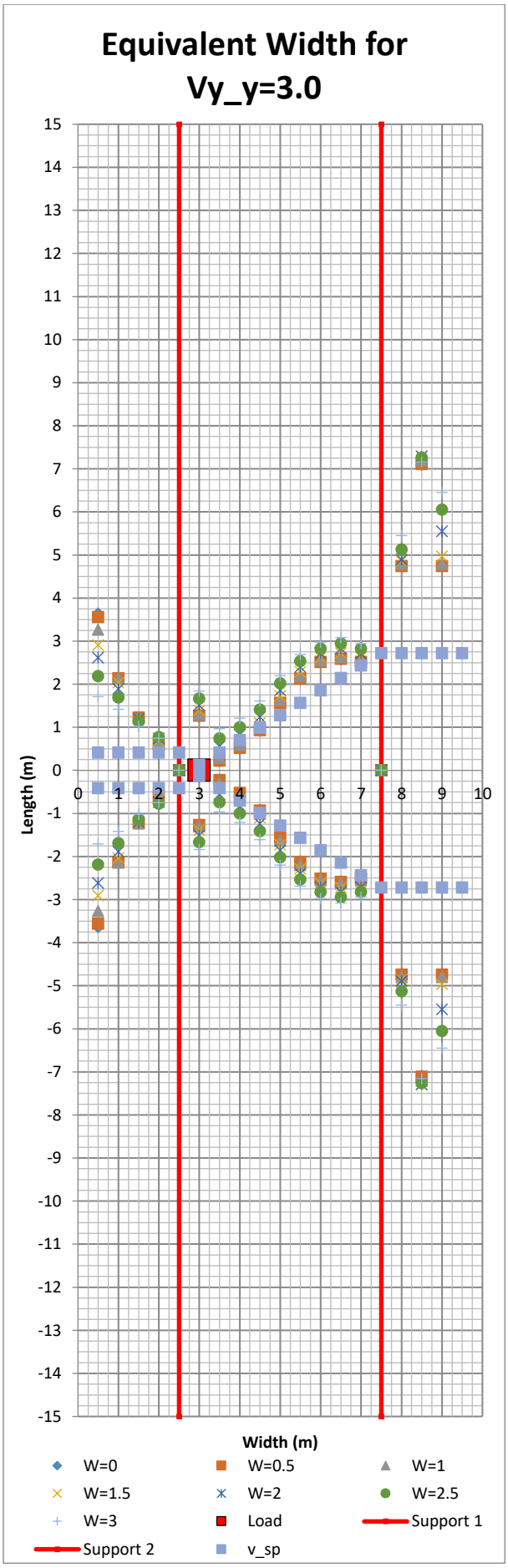

Equivalent Width for My_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

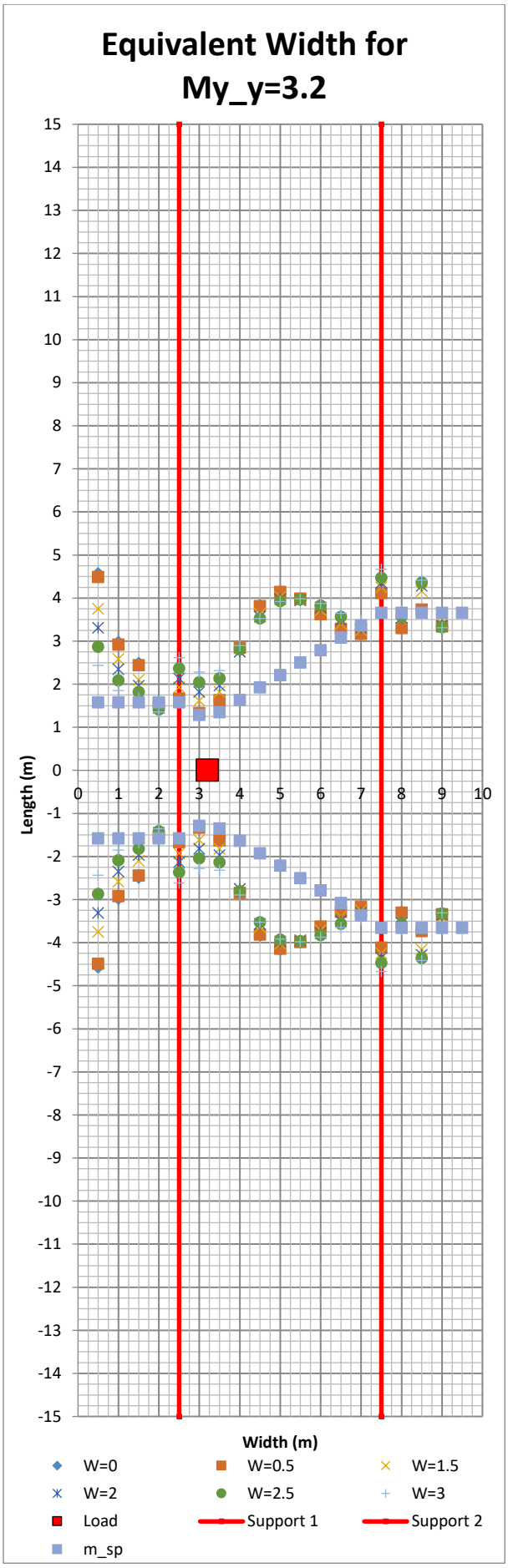

Equivalent Width for Vy_=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

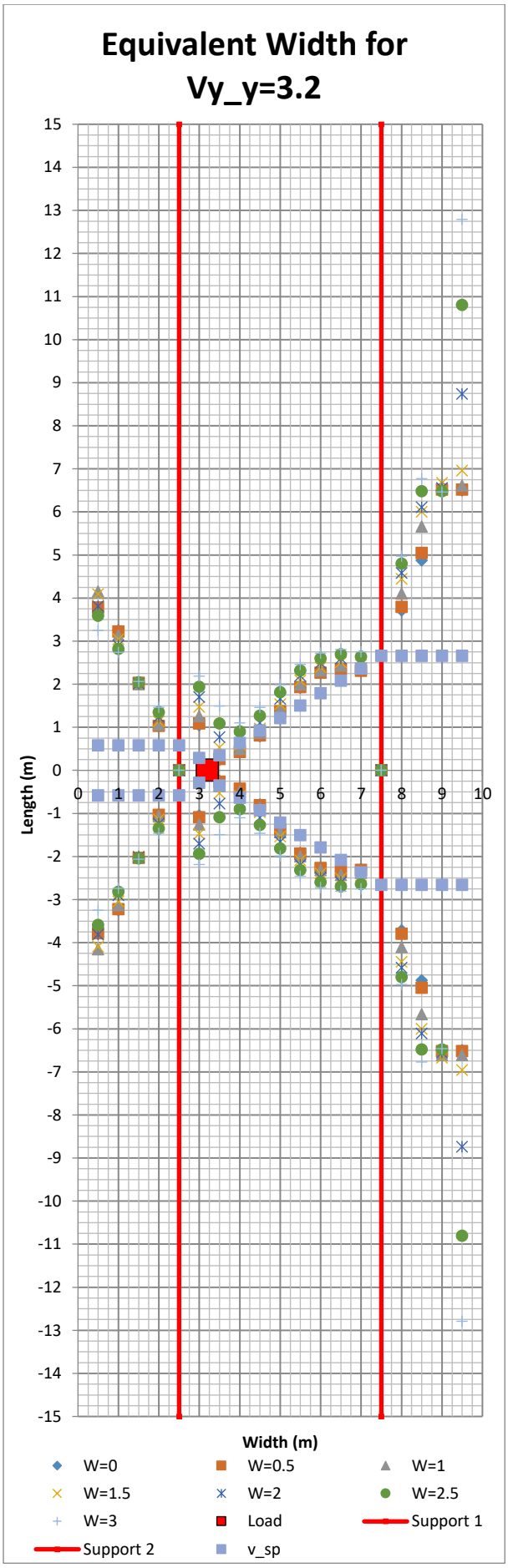

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

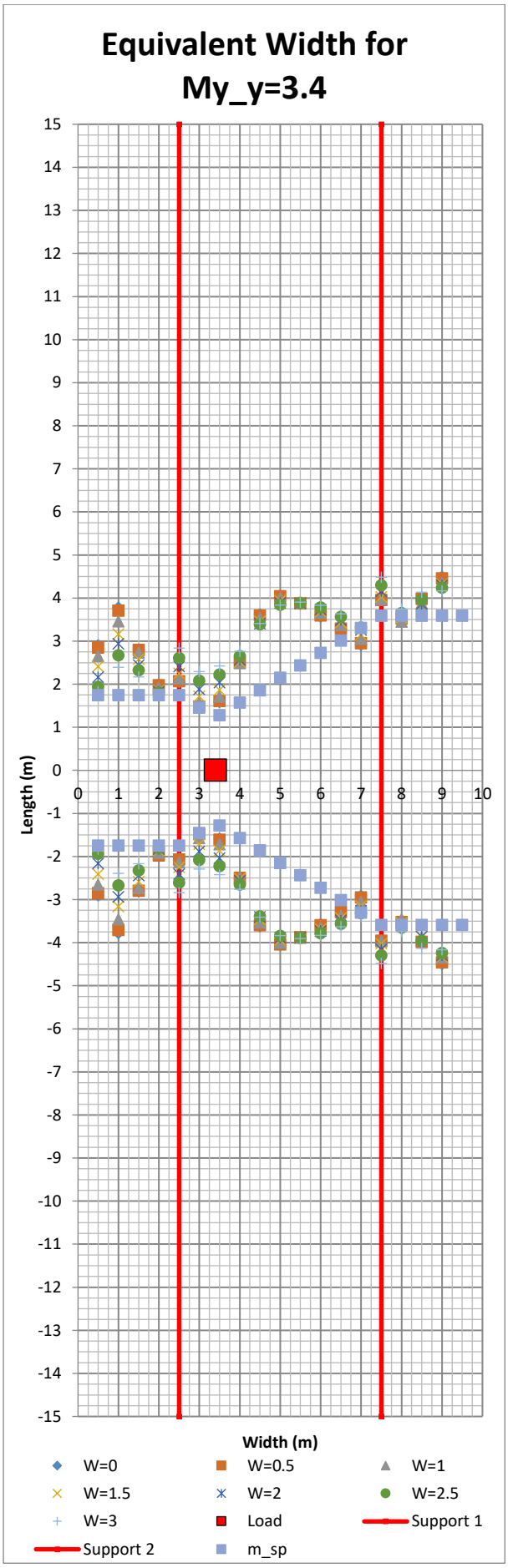

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

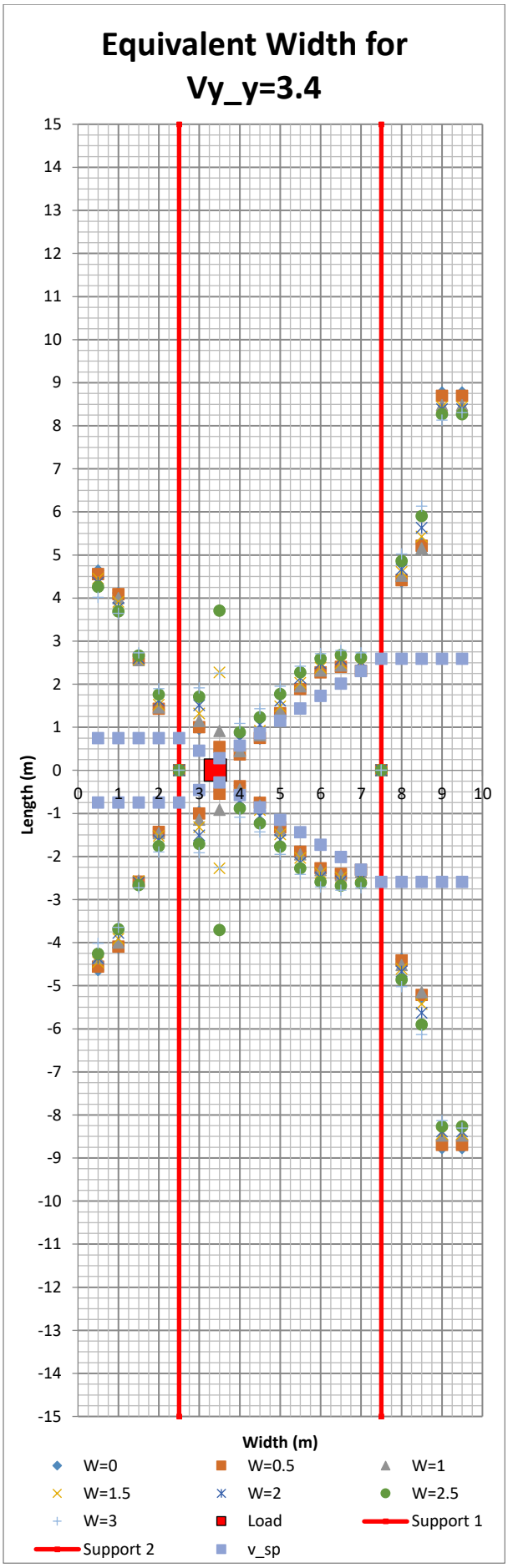

Equivalent Width for My_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

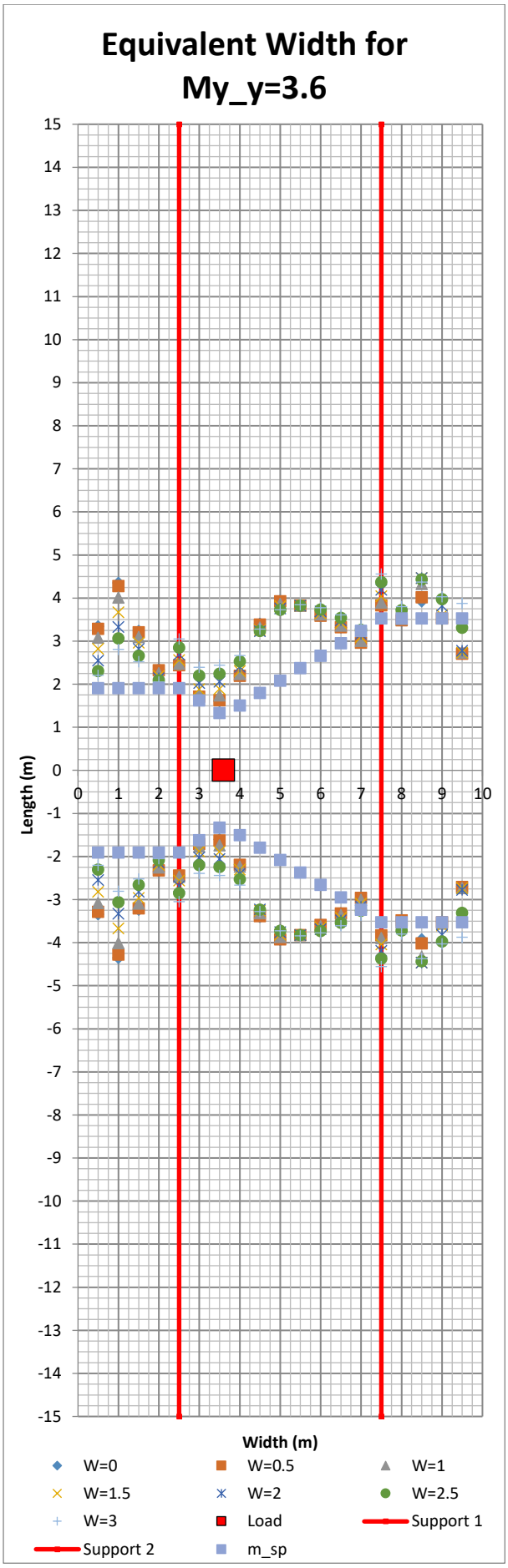

Equivalent Width for Vy_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

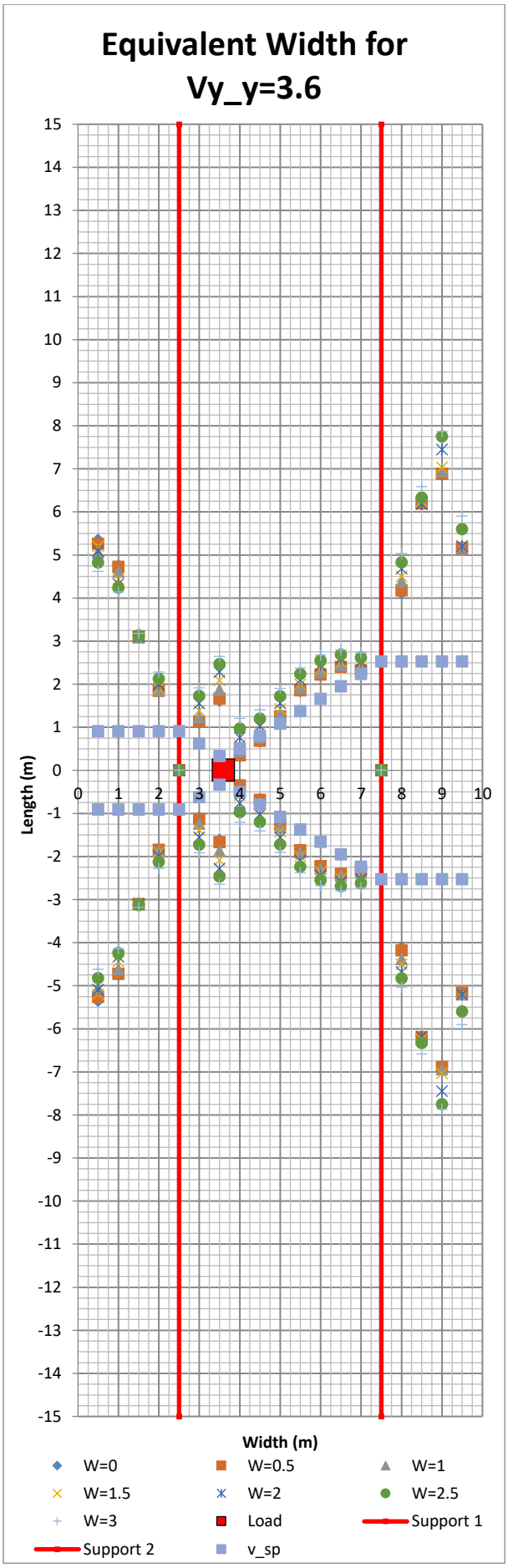

Equivalent Width for My_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

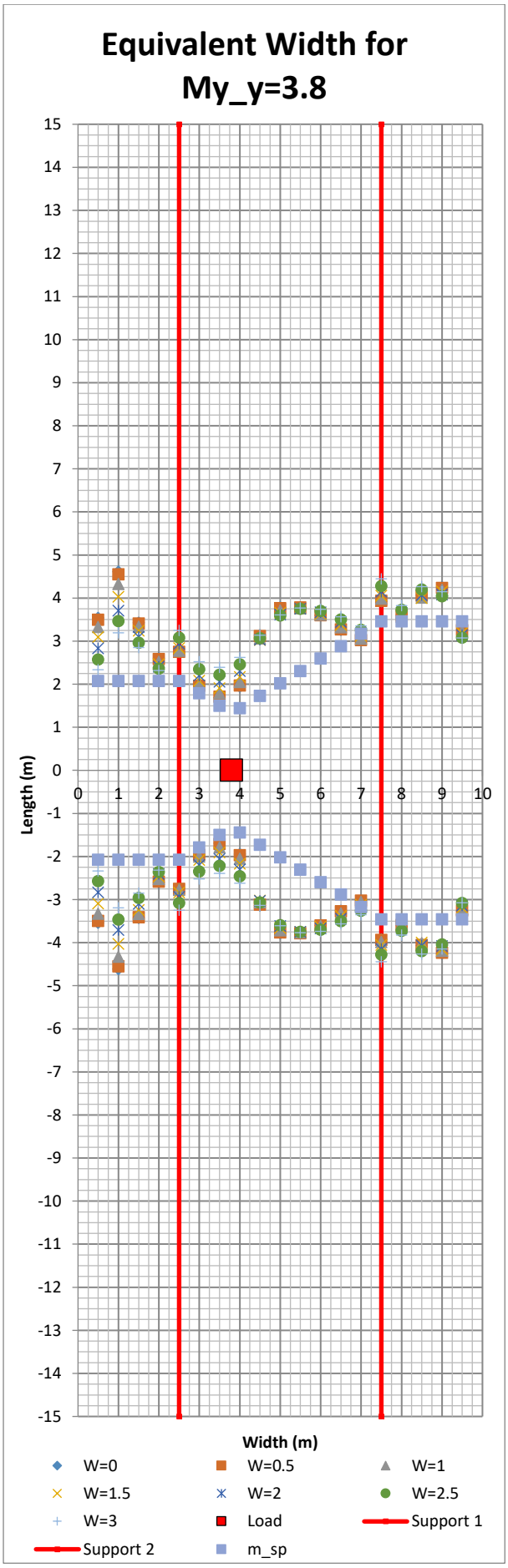

Equivalent Width for Vy_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

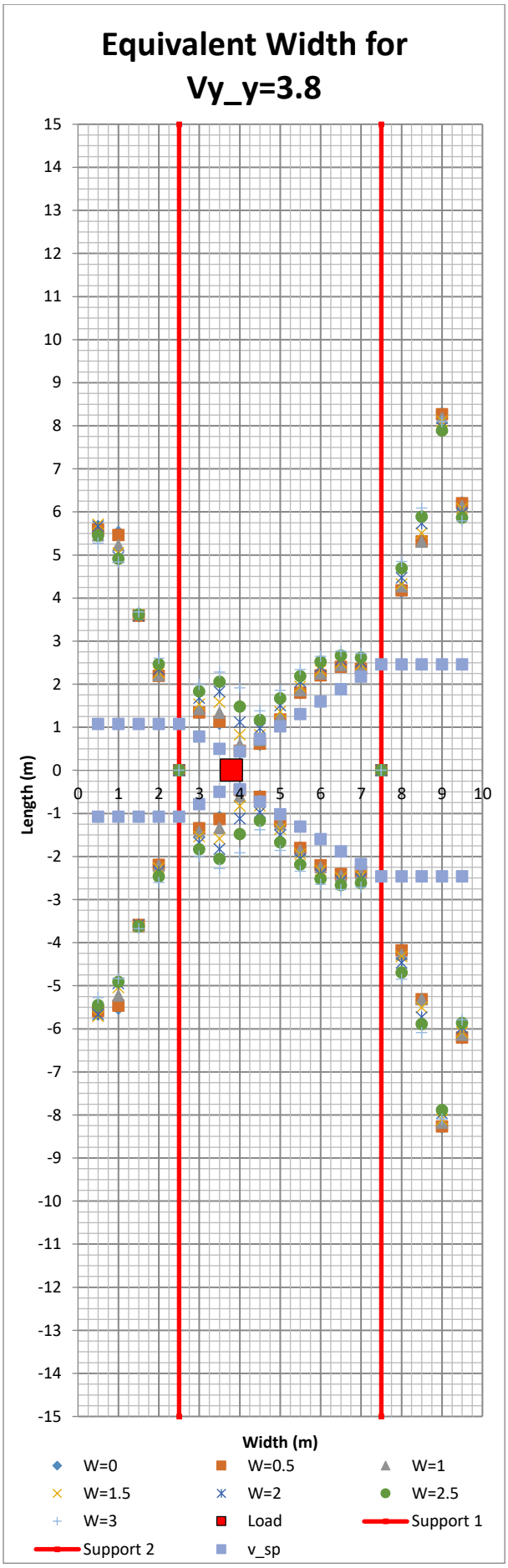

Equivalent Width for My_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

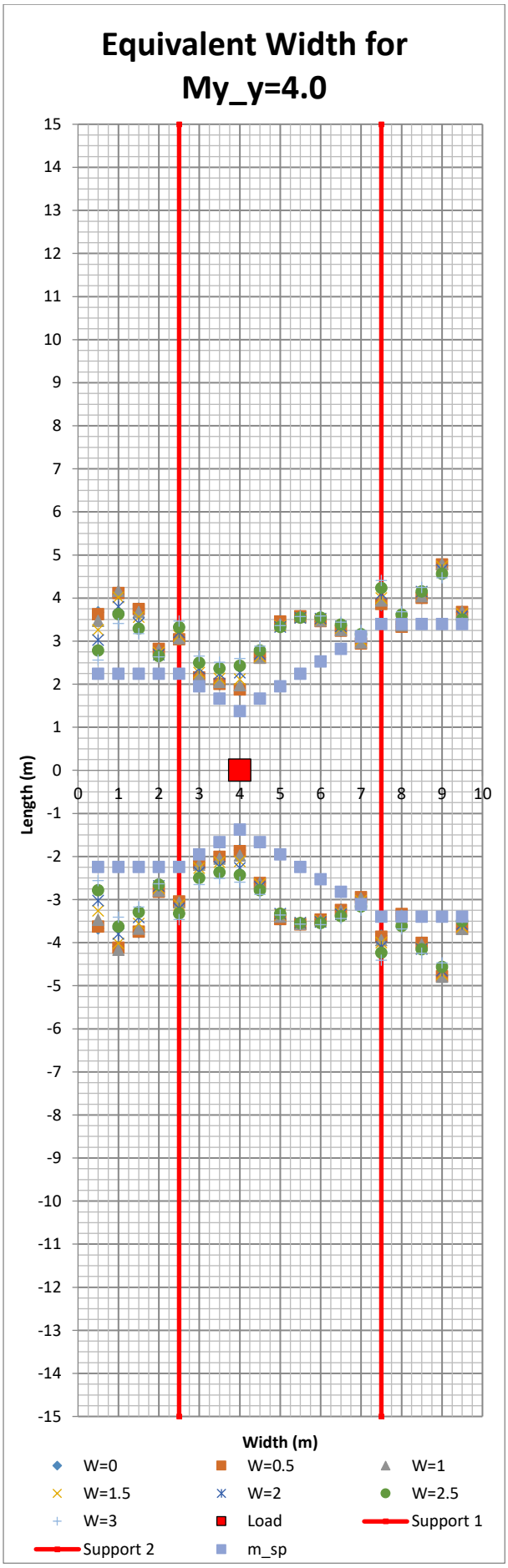

Equivalent Width for Vy_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

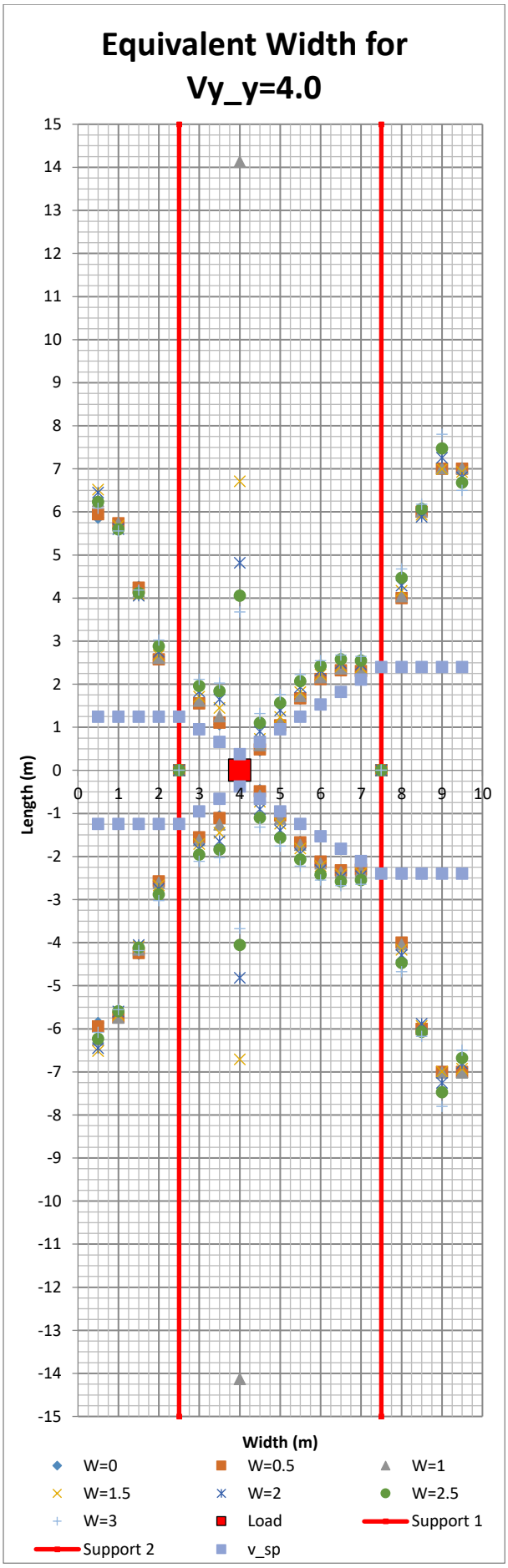

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

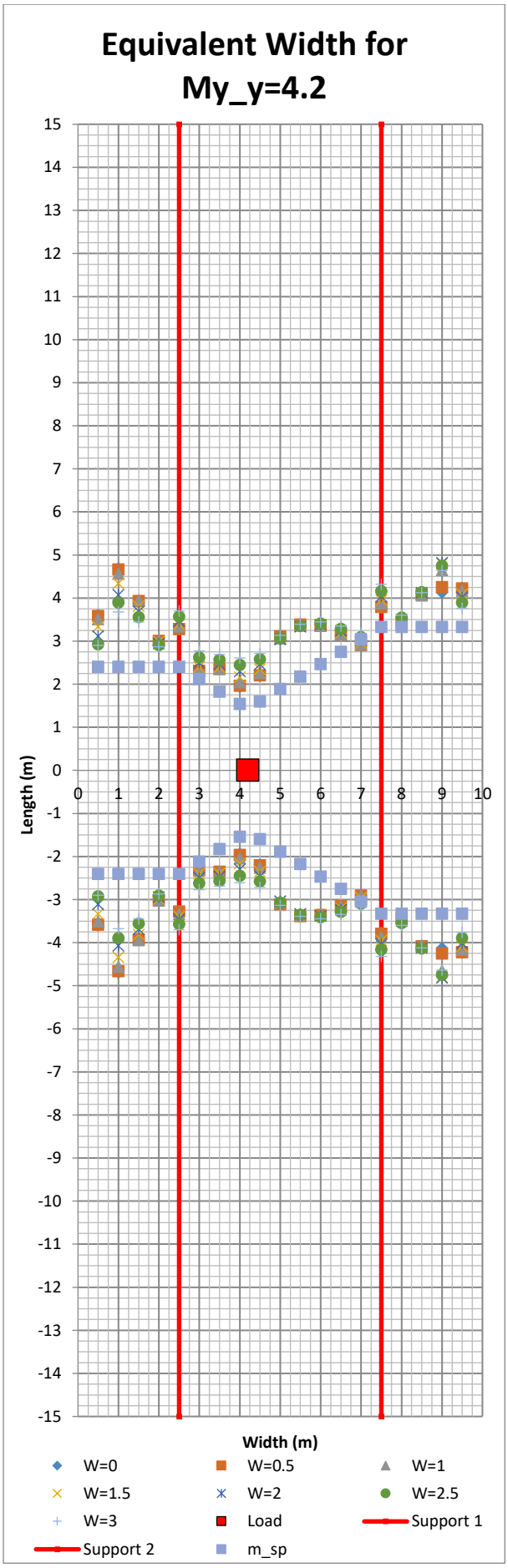

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

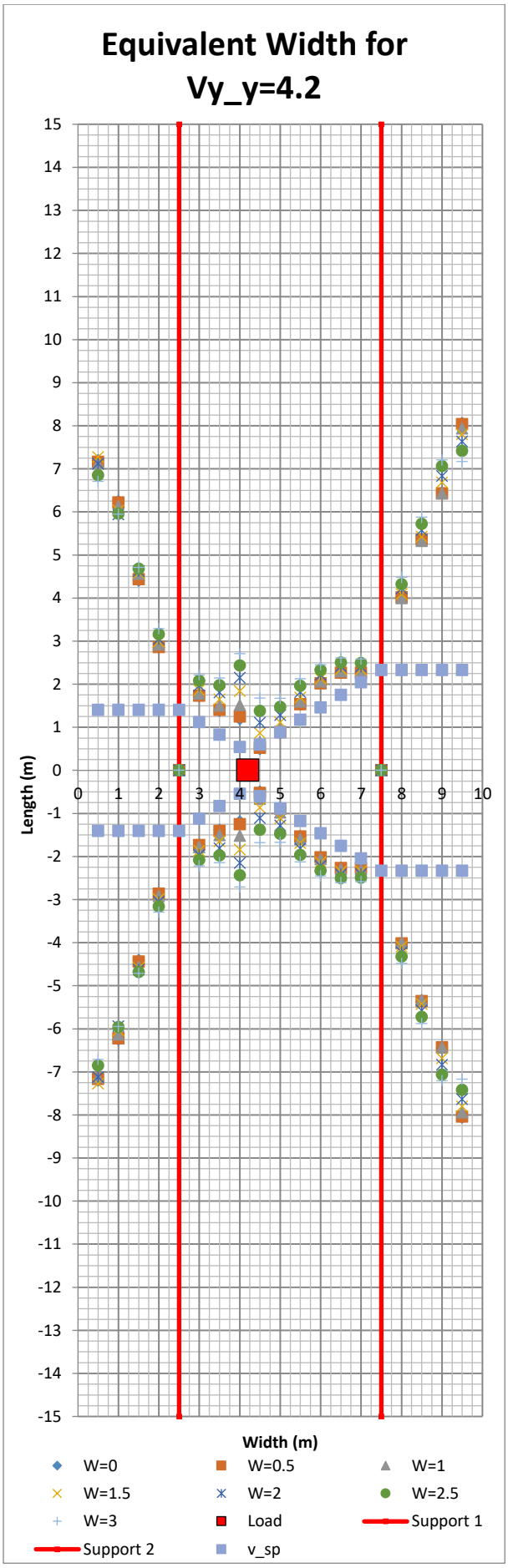

Equivalent Width for My_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

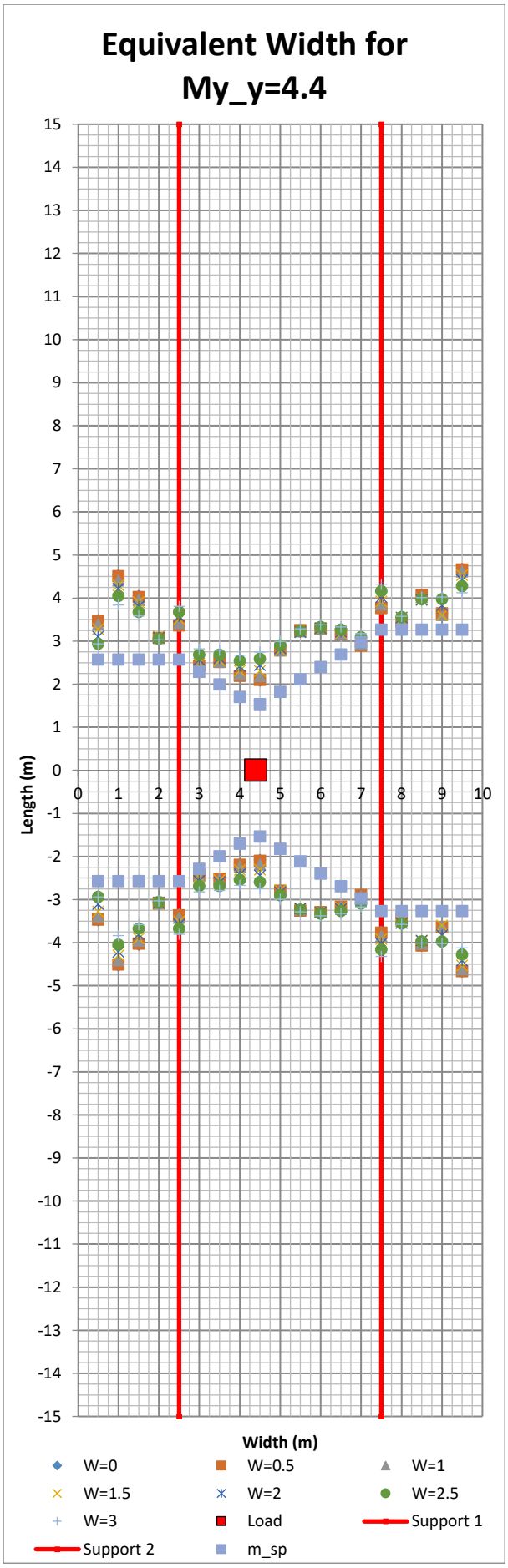

Equivalent Width for Vy_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

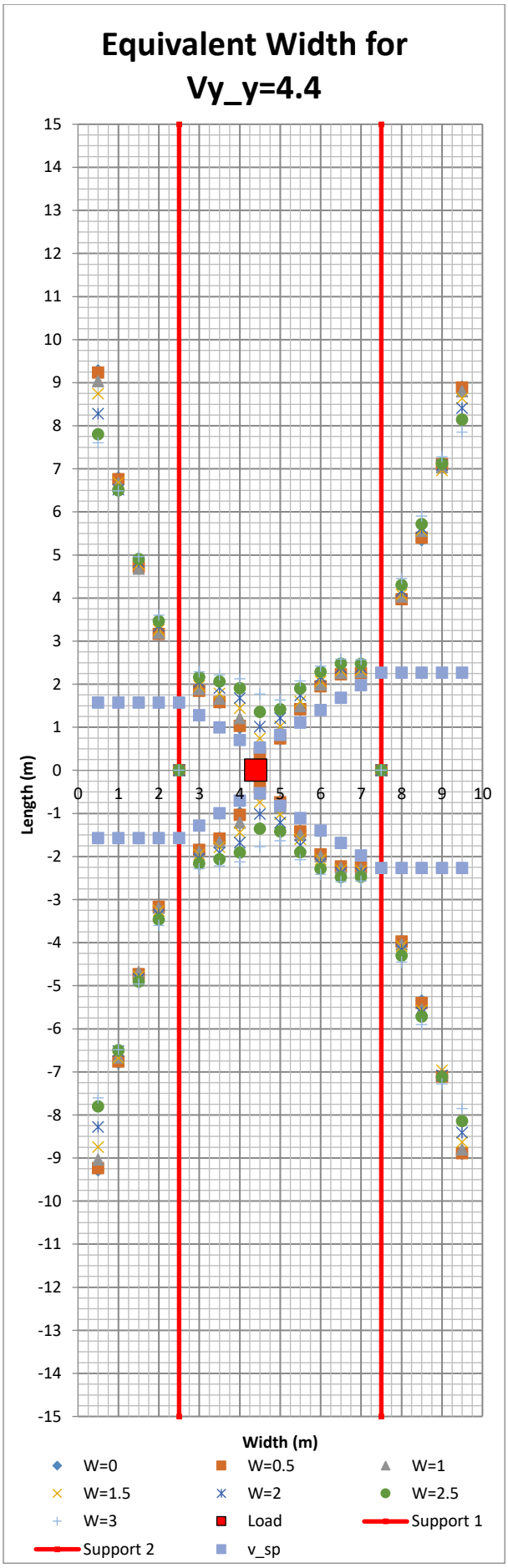

Equivalent Width for My_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

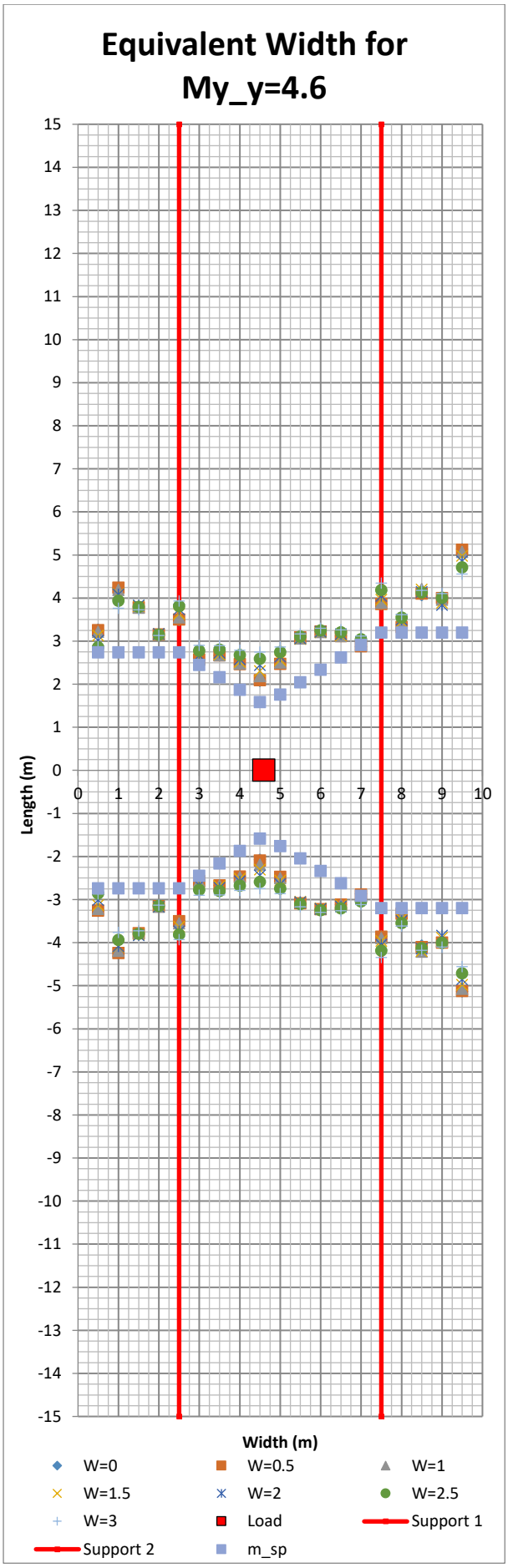

Equivalent Width for Vy_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

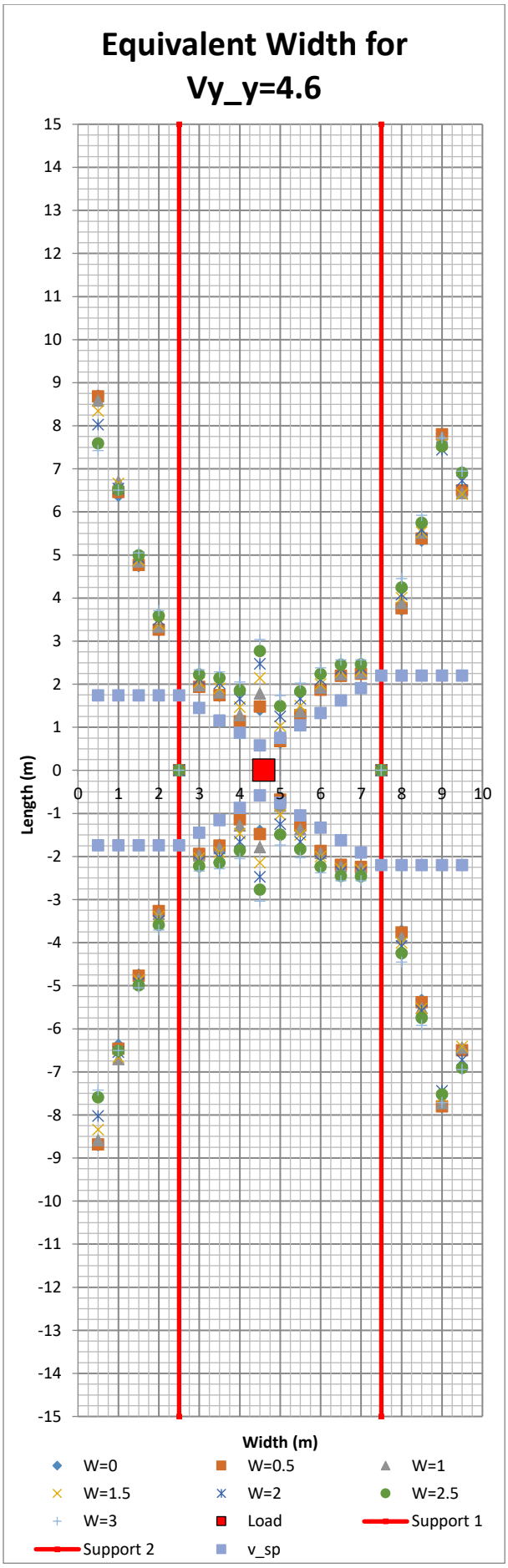

Equivalent Width for My_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

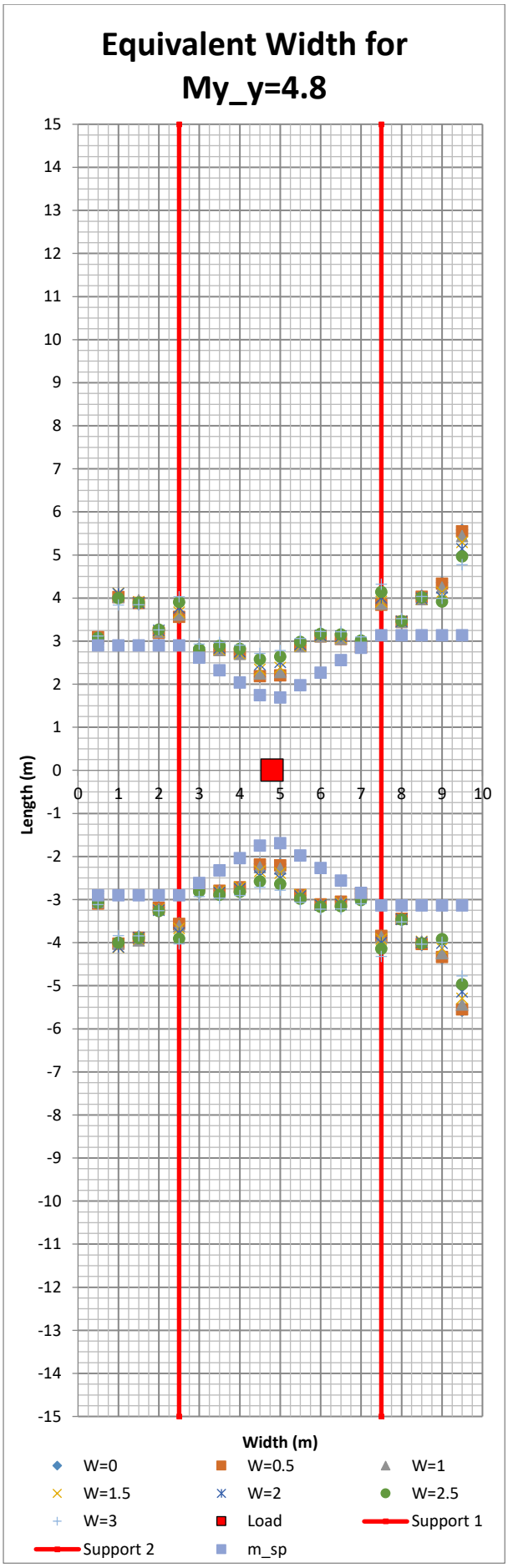

Equivalent Width for Vy_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

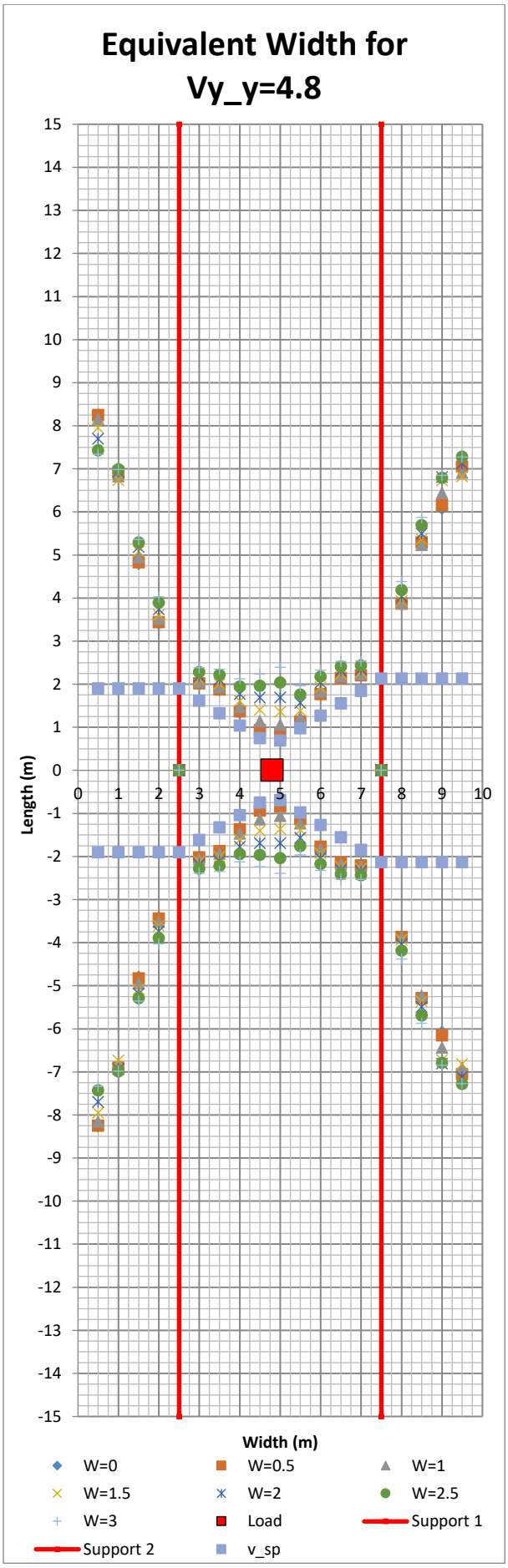

Equivalent Width for My_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

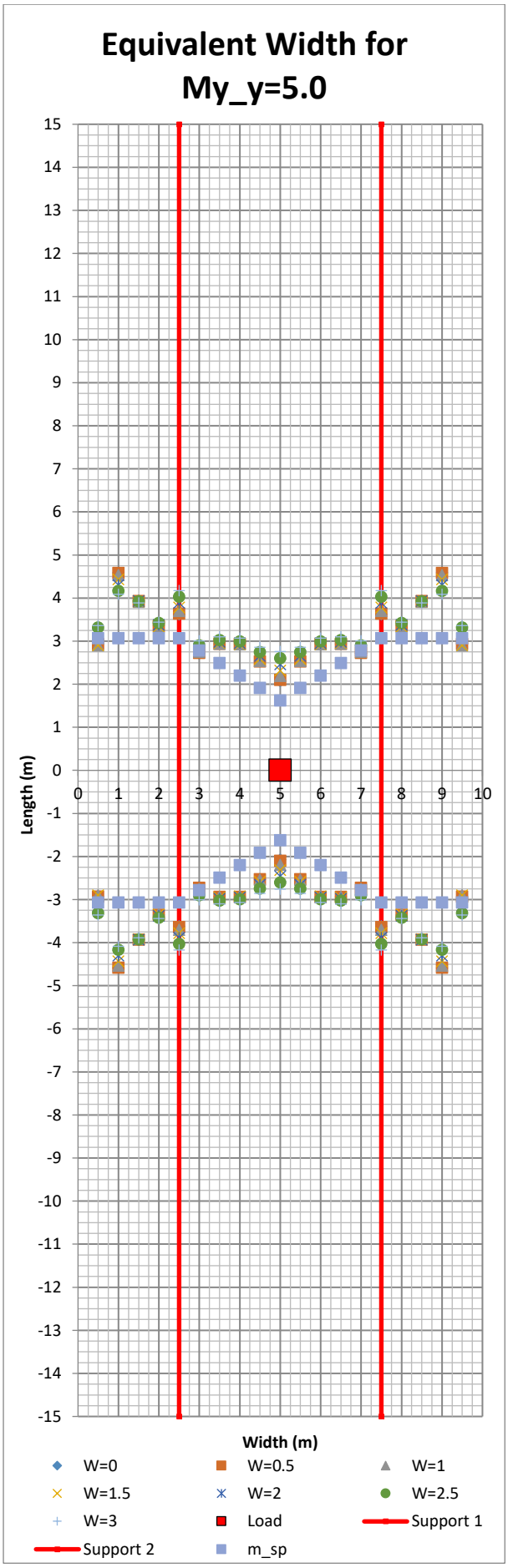

Equivalent Width for Vy_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

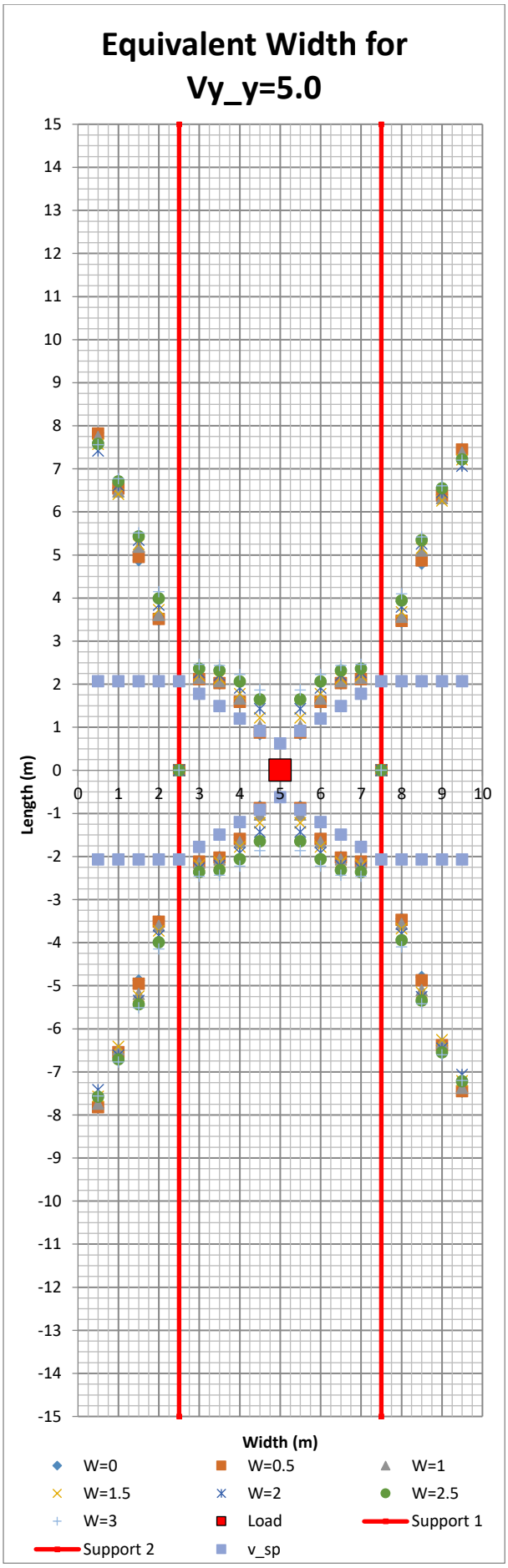

Equivalent Width for My_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

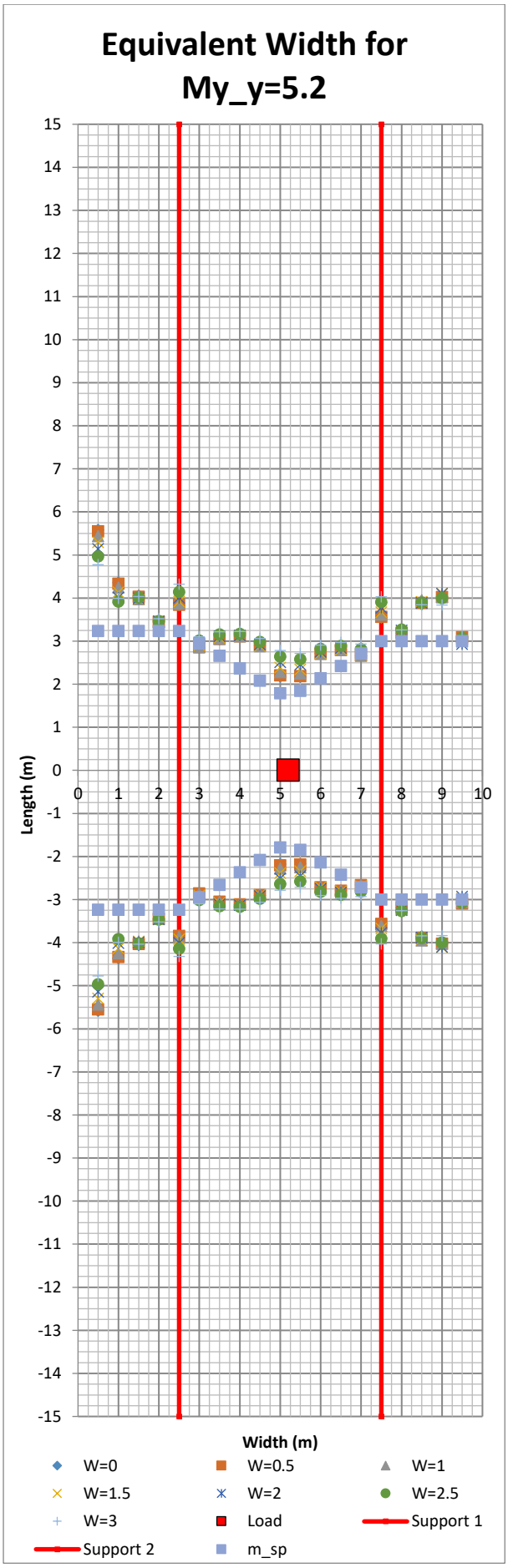

Equivalent Width for Vy_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

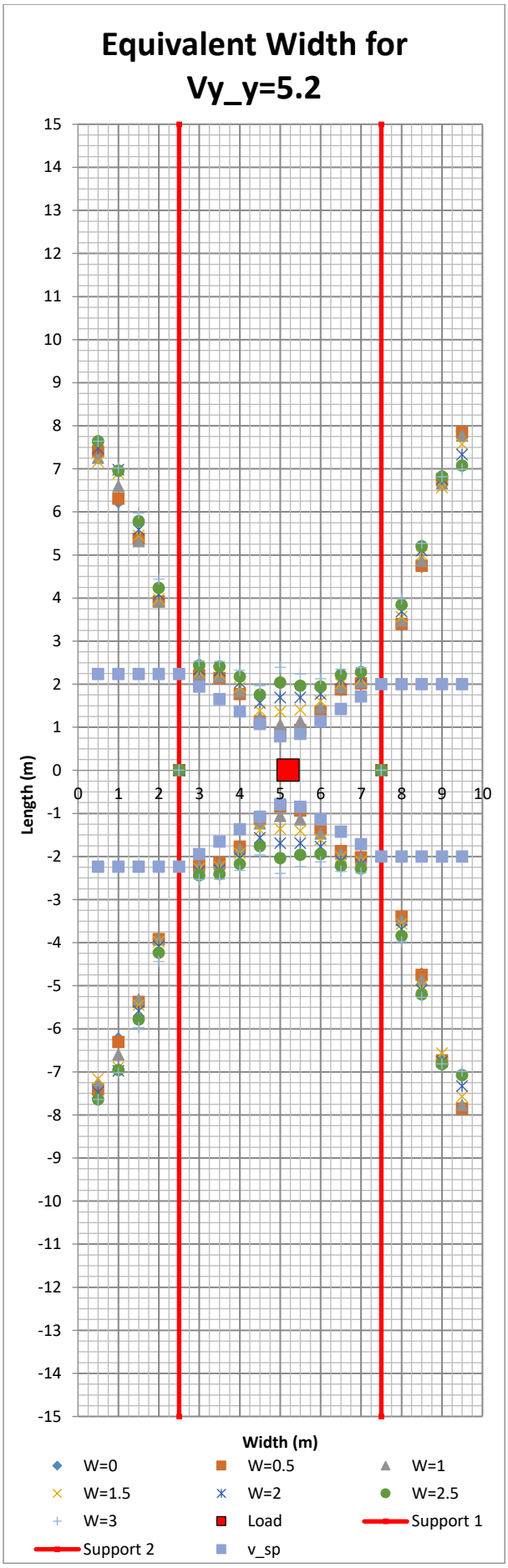

Equivalent Width for My_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

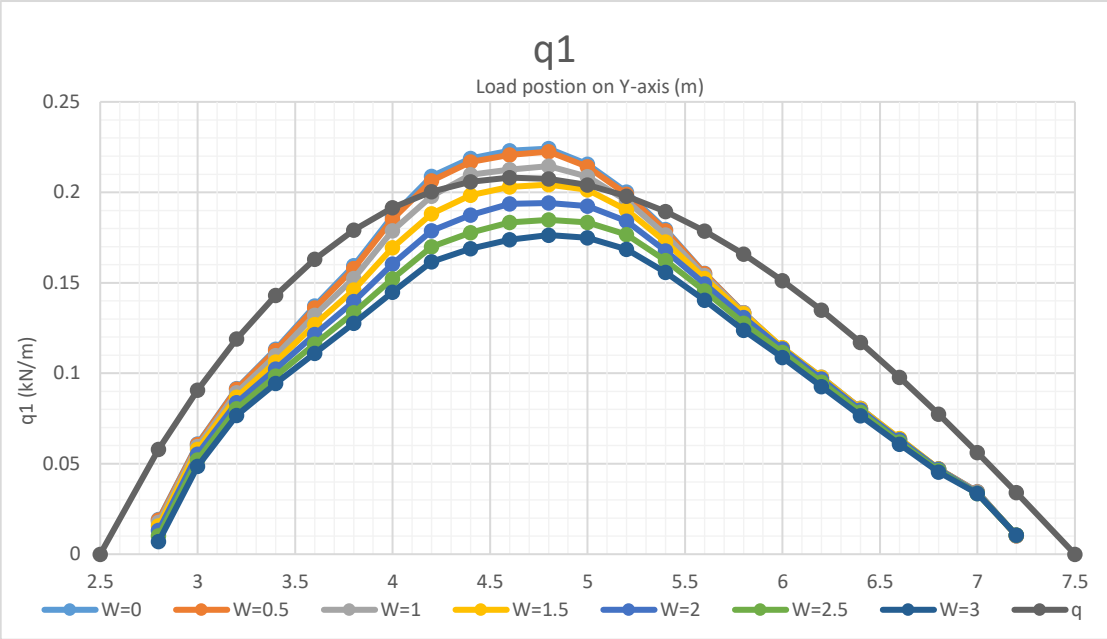

Equivalent Width for My_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

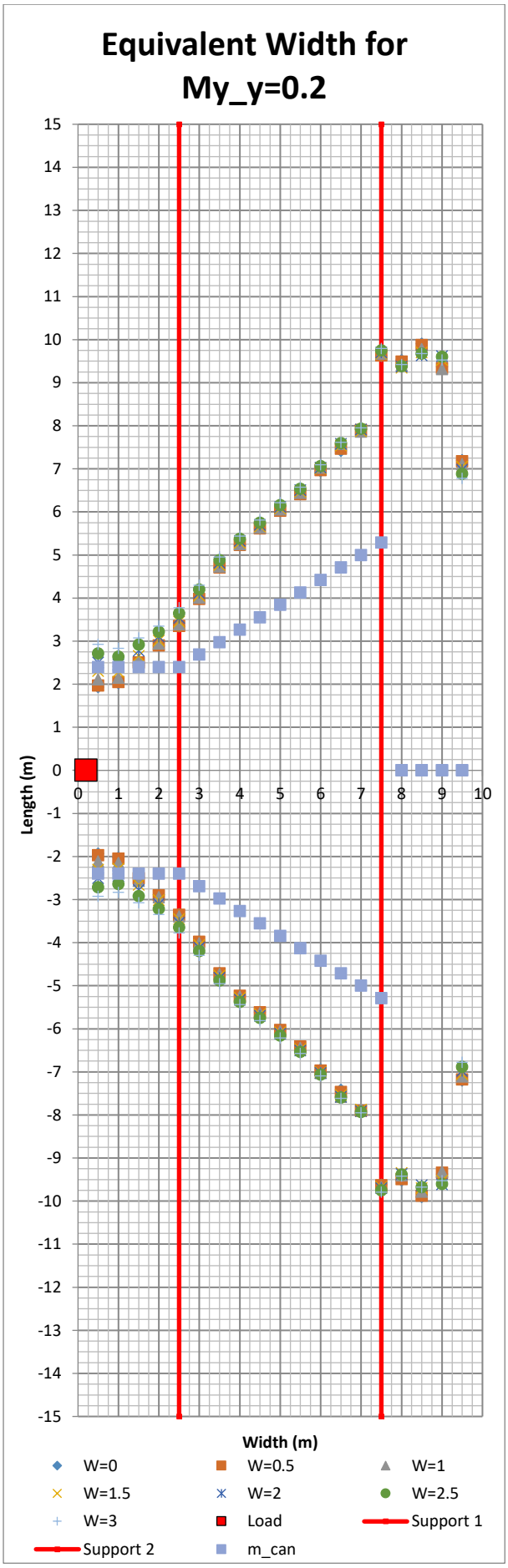

Equivalent Width for Vy_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

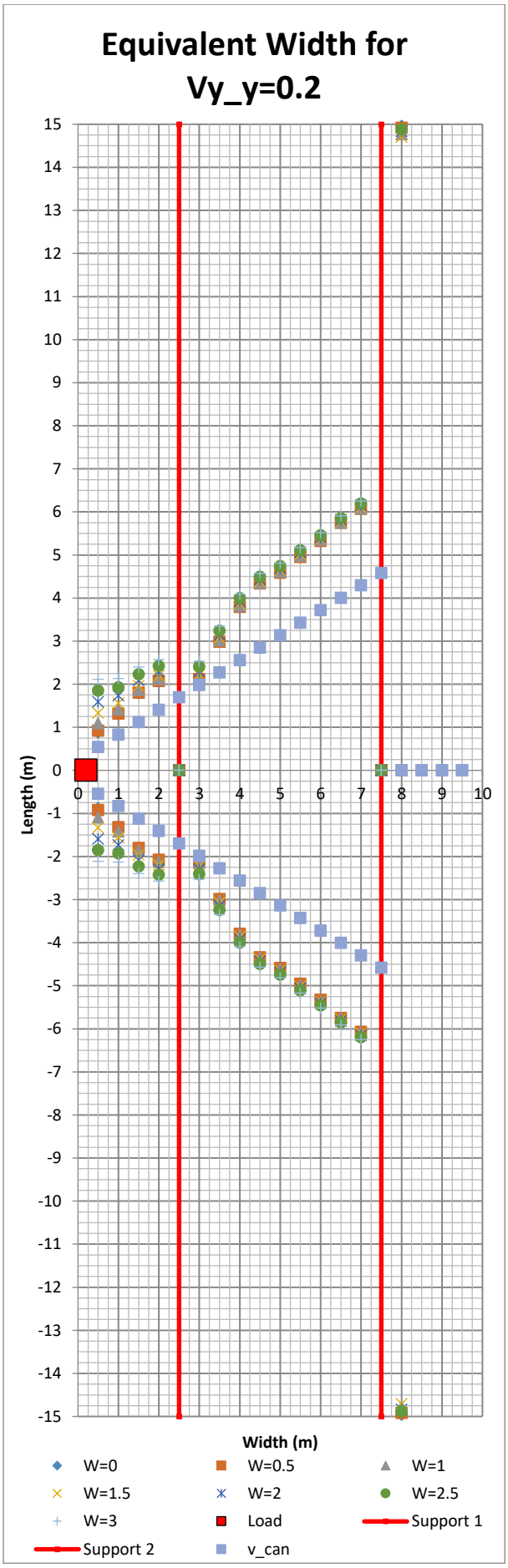

Equivalent Width for My_y=5.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

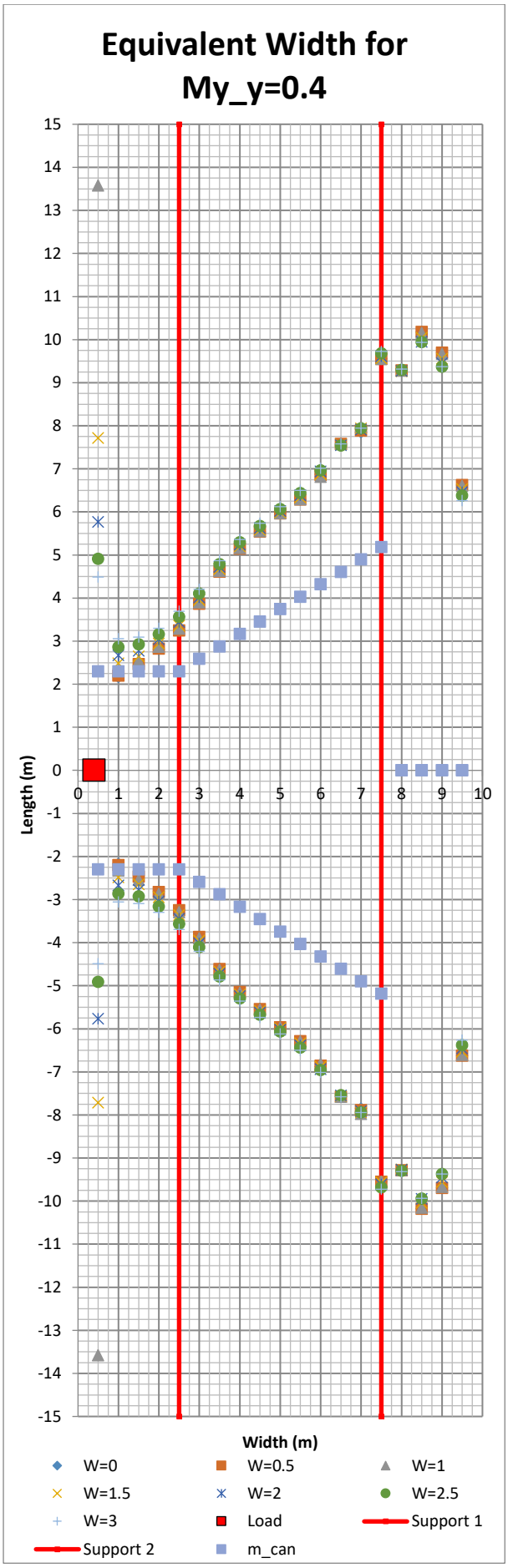

Equivalent Width for Vy_y=5.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

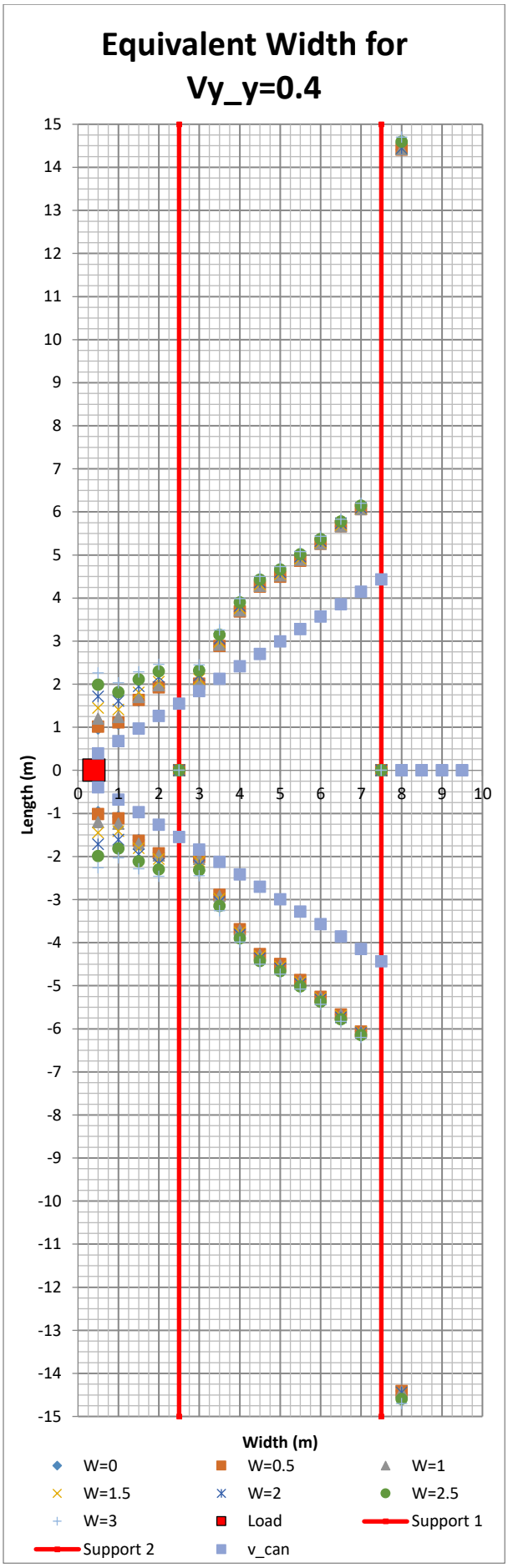
11.8 Eighth Variation $t = [0.15, 0.3, 0.15]m$

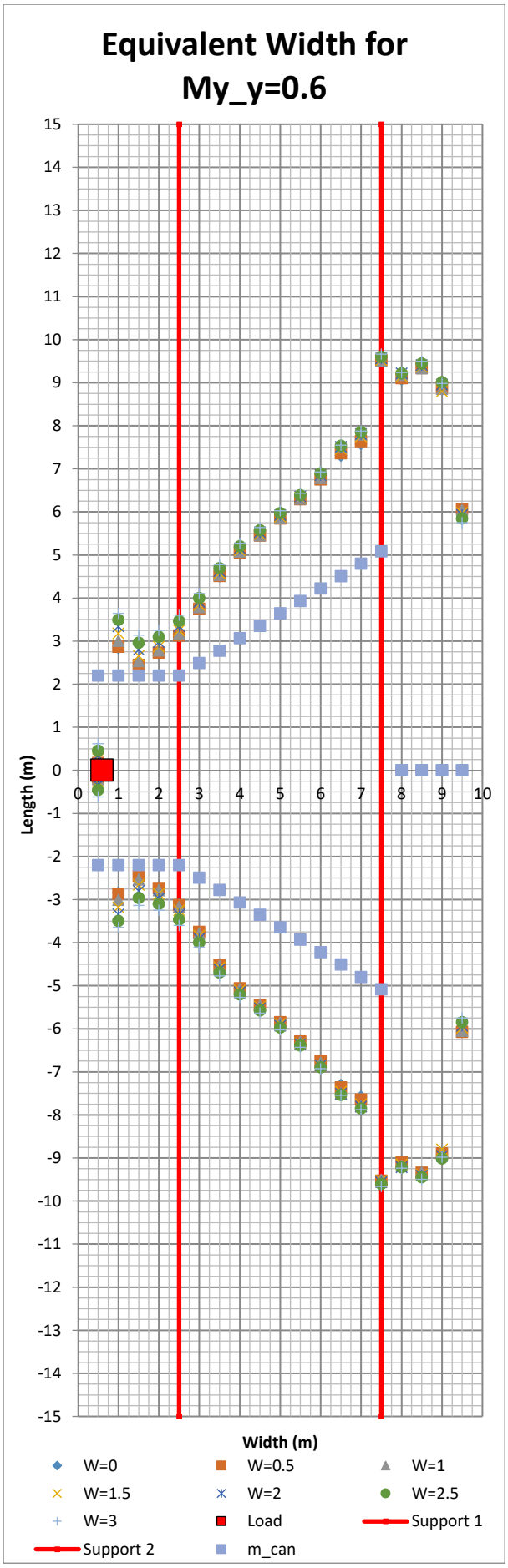


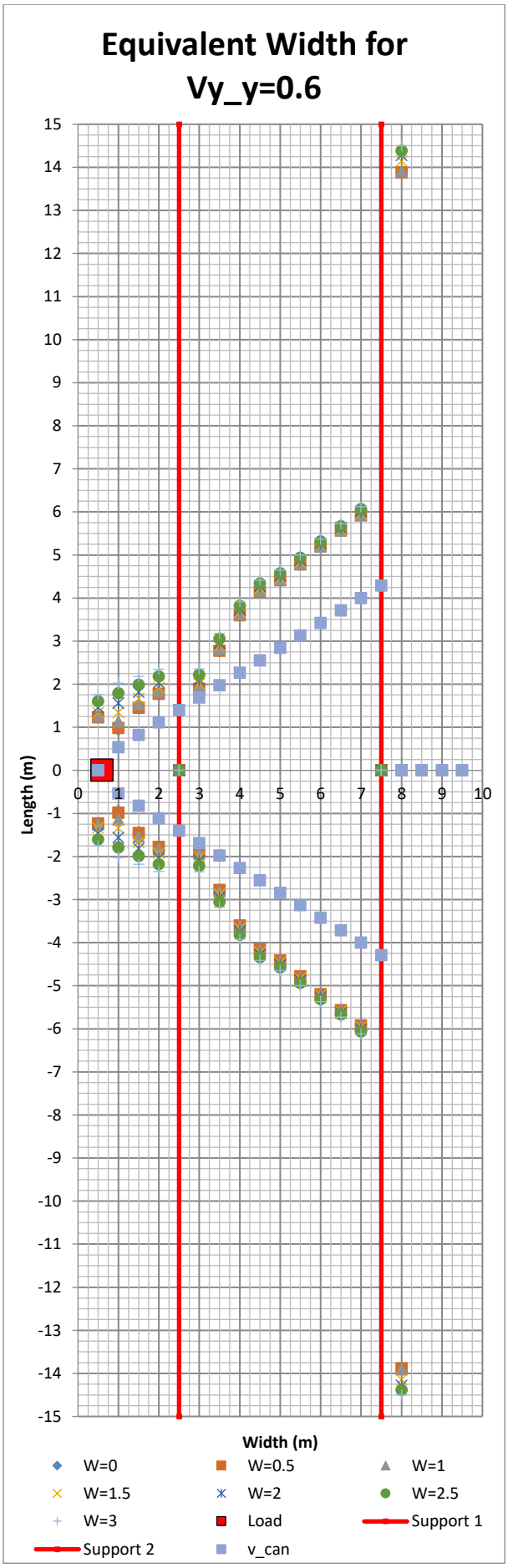

Equivalent Width for My_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

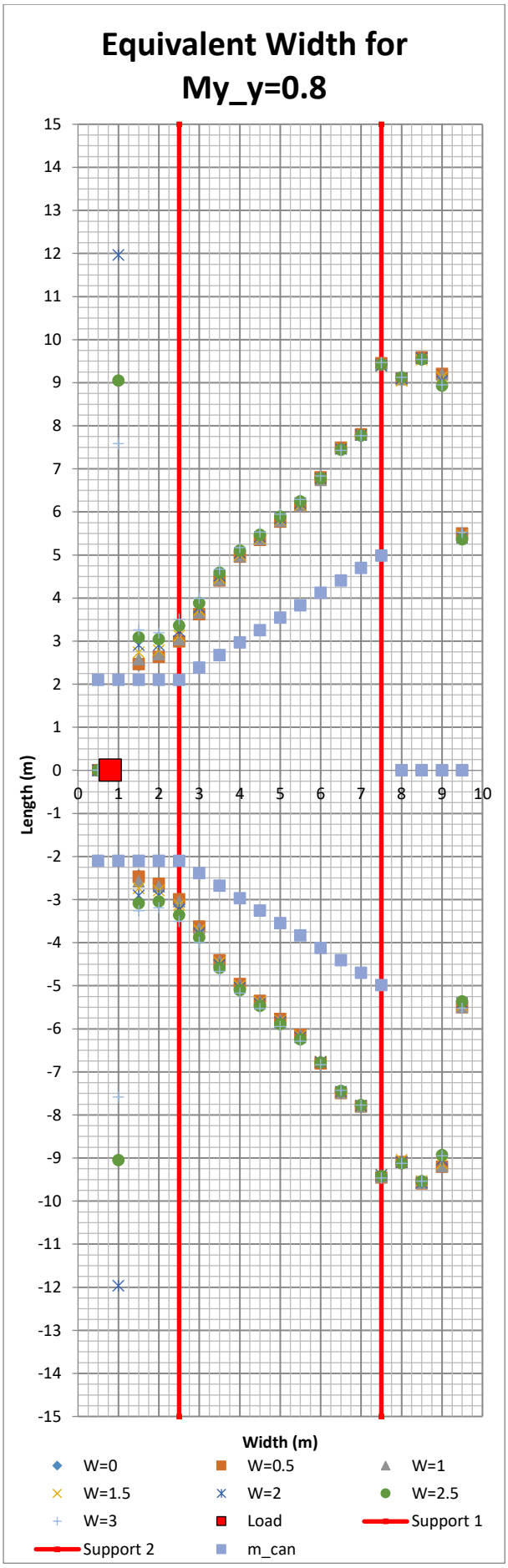

Equivalent Width for Vy_y=0.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

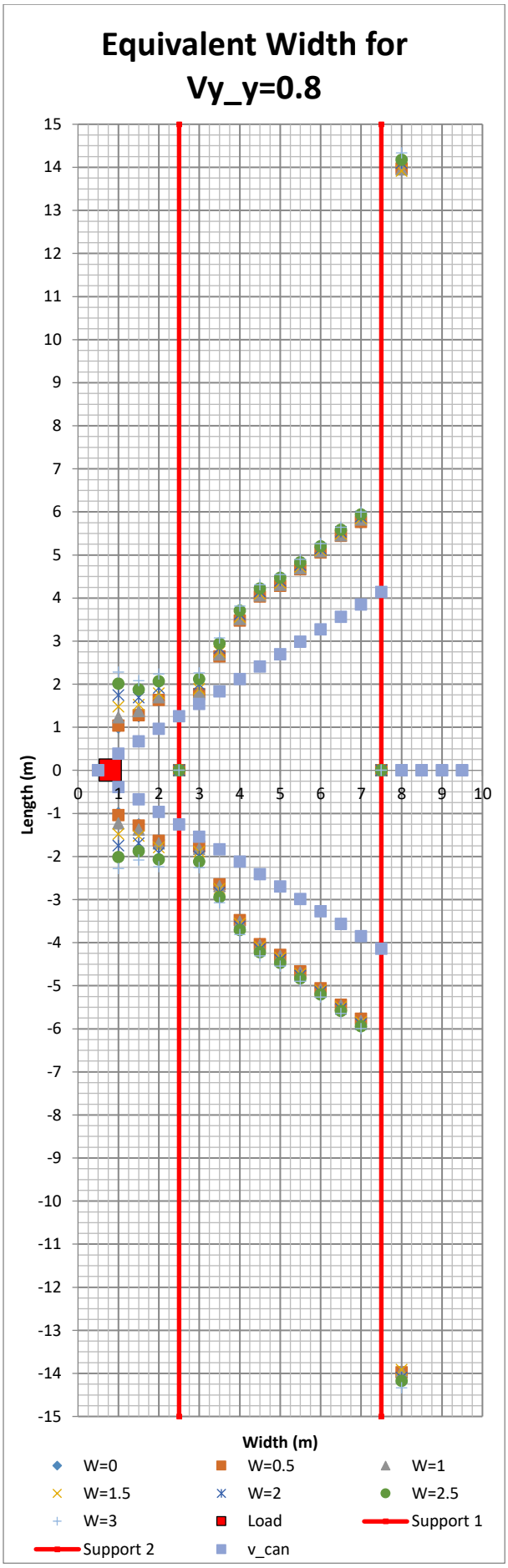

Equivalent Width for My_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

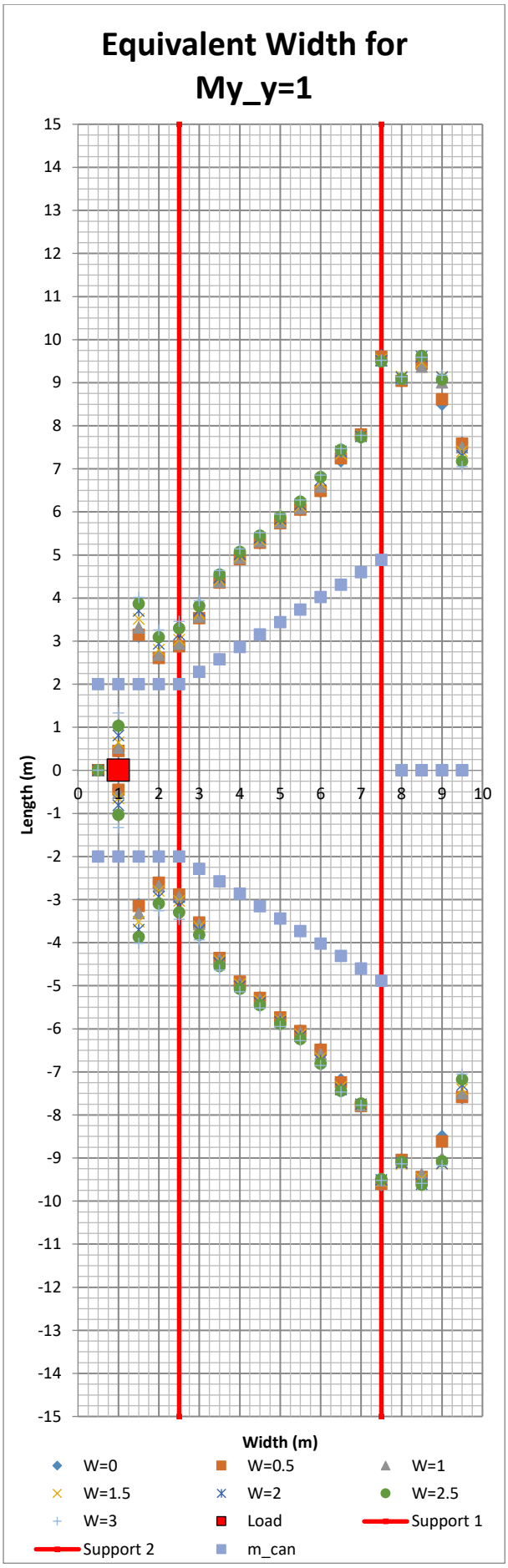

Equivalent Width for Vy_y=0.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

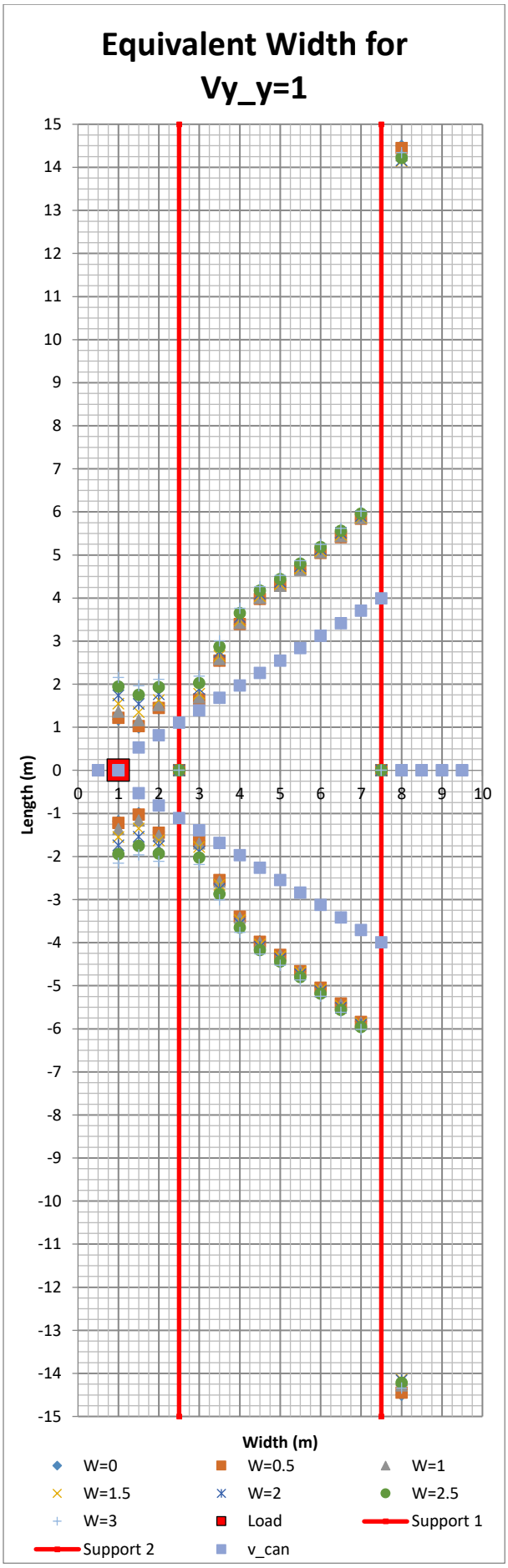

Equivalent Width for My_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

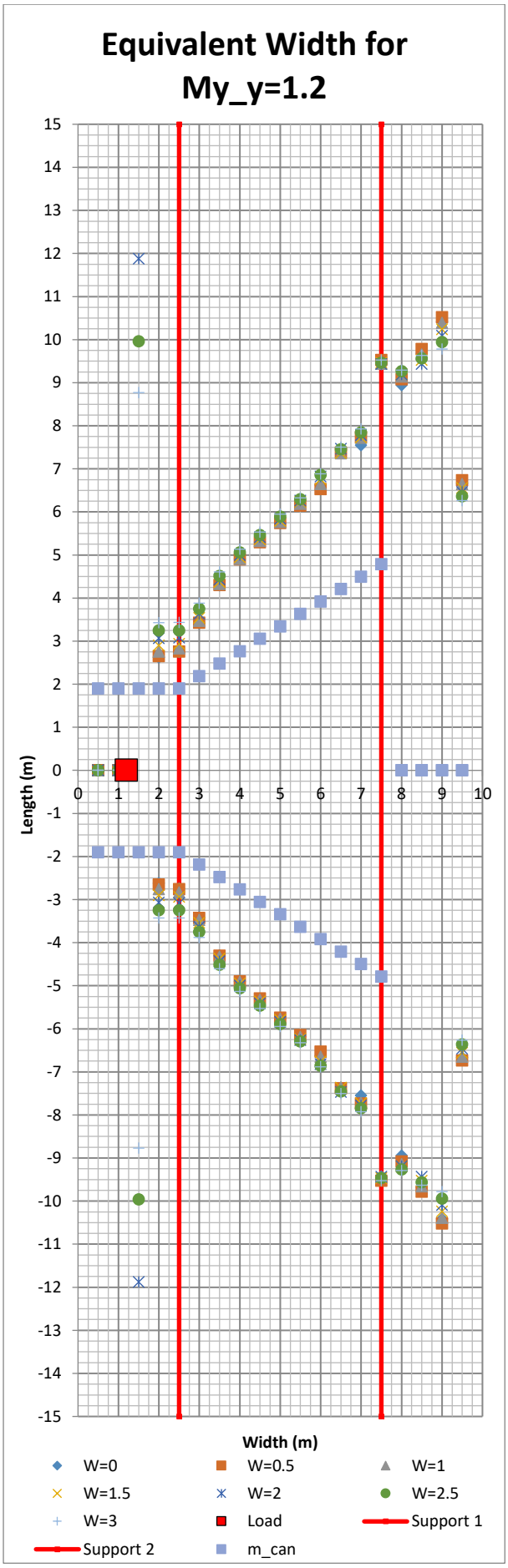

Equivalent Width for Vy_y=0.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

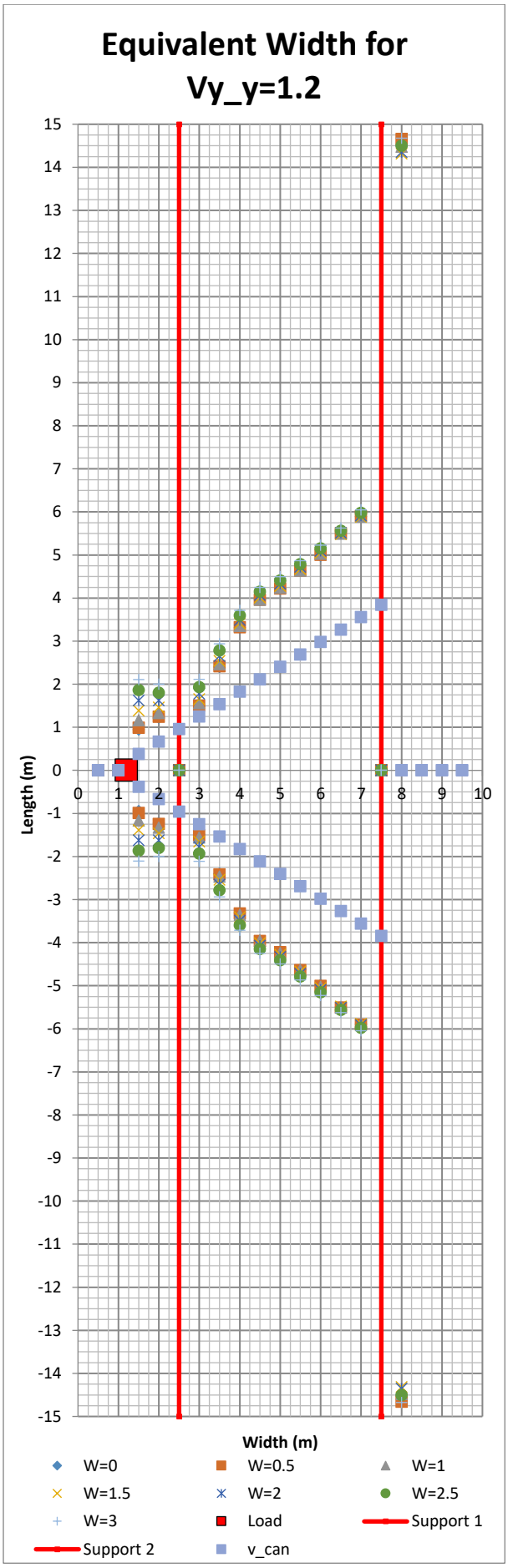

Equivalent Width for My_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

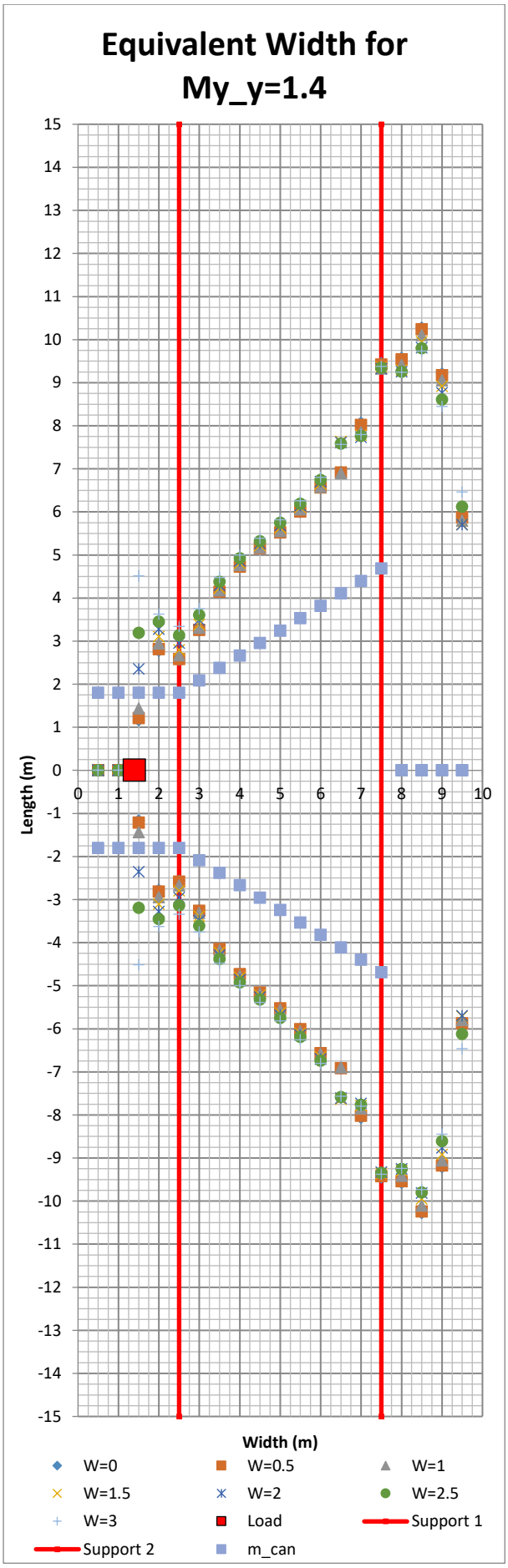

Equivalent Width for Vy_y=0.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

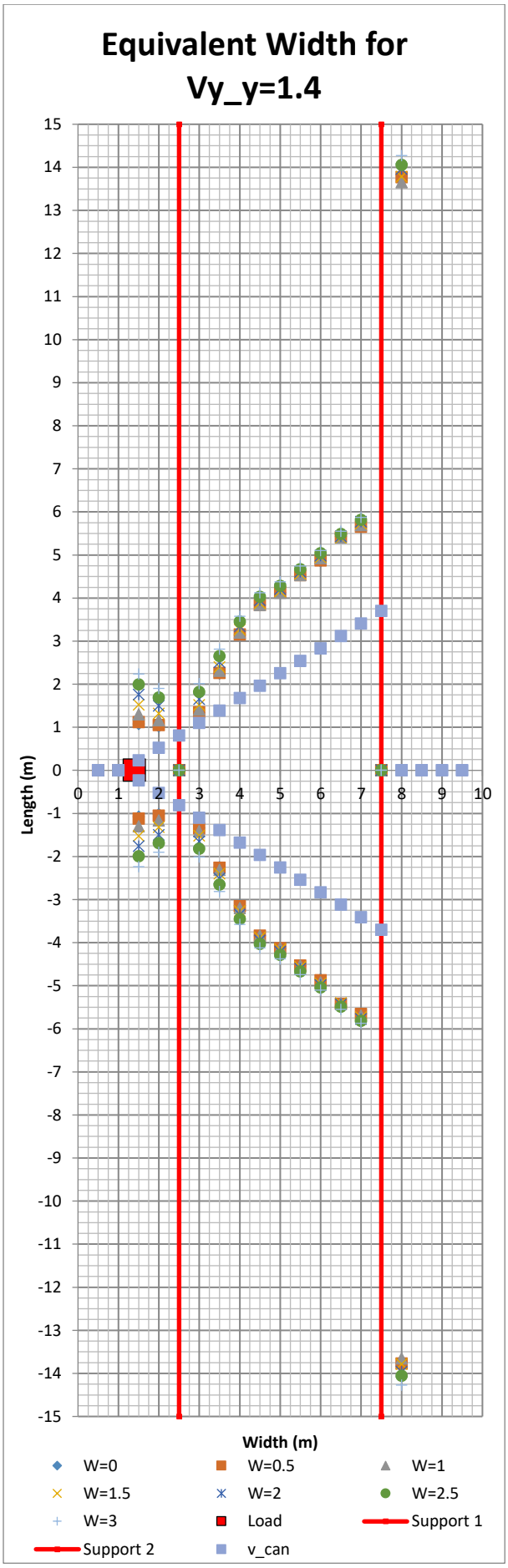

Equivalent Width for My_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

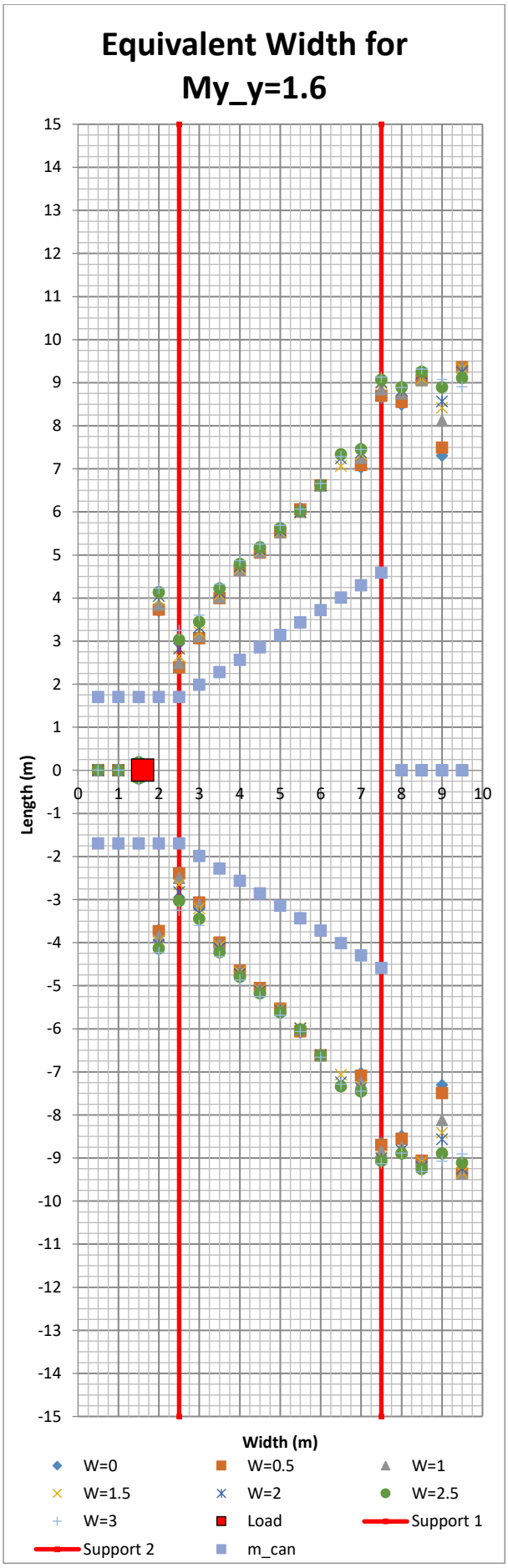

Equivalent Width for Vy_y=1
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

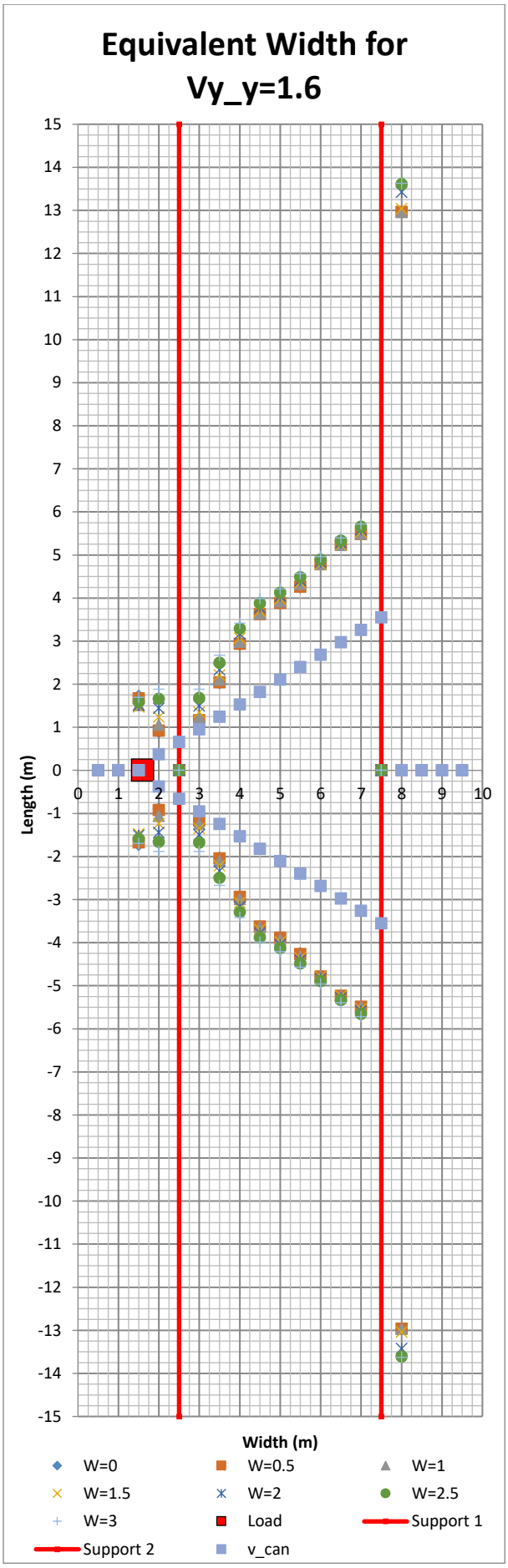

Equivalent Width for My_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

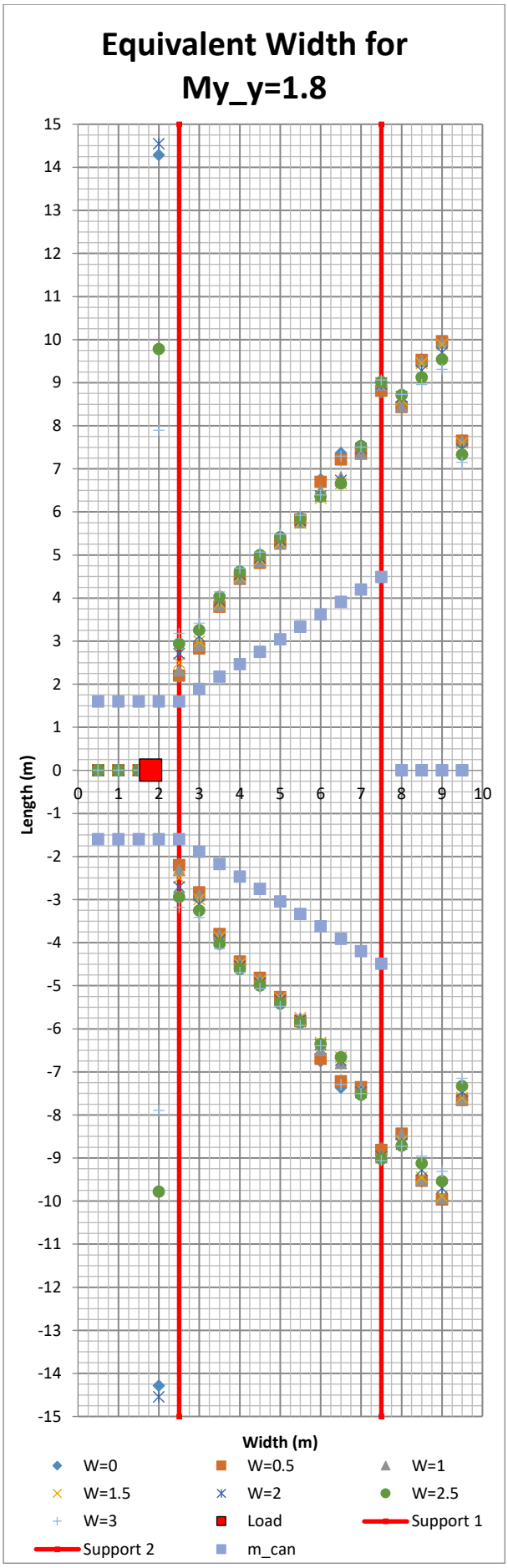

Equivalent Width for Vy_y=1.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

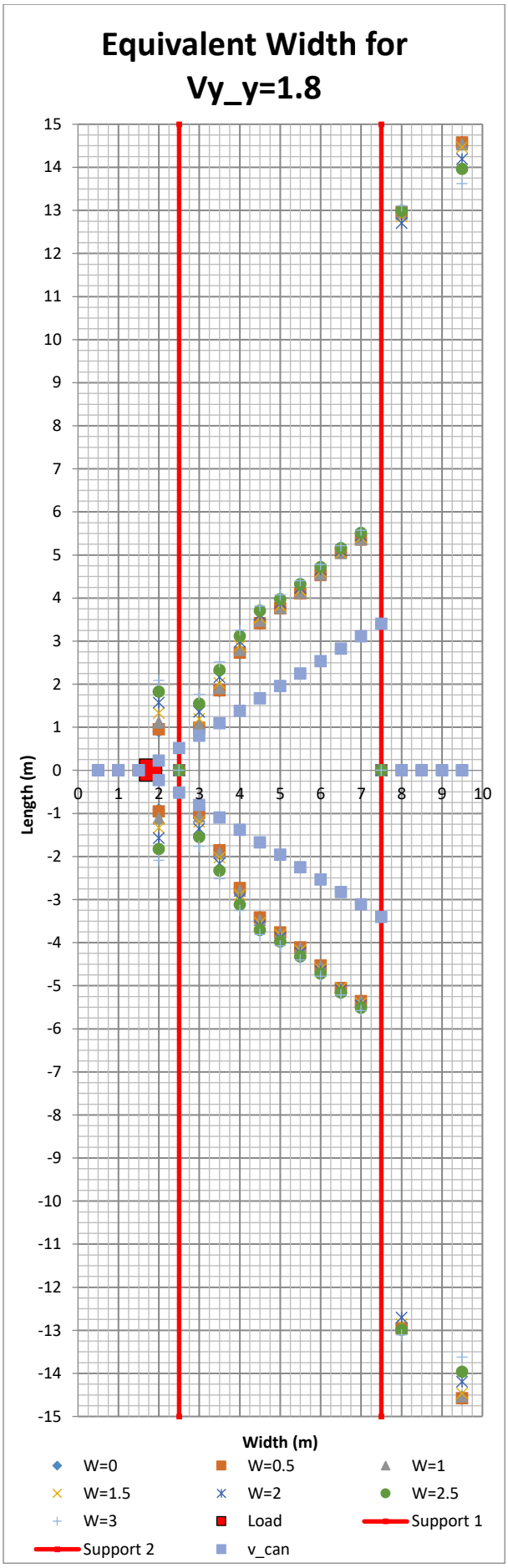

Equivalent Width for My_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

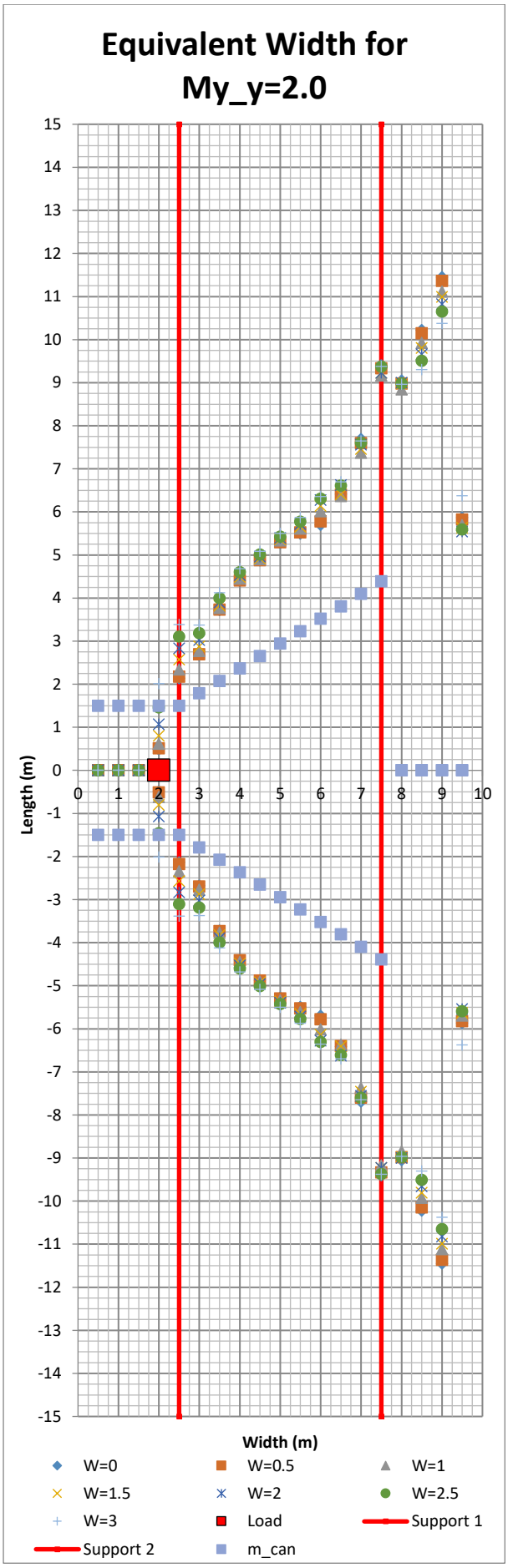

Equivalent Width for Vy_y=1.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

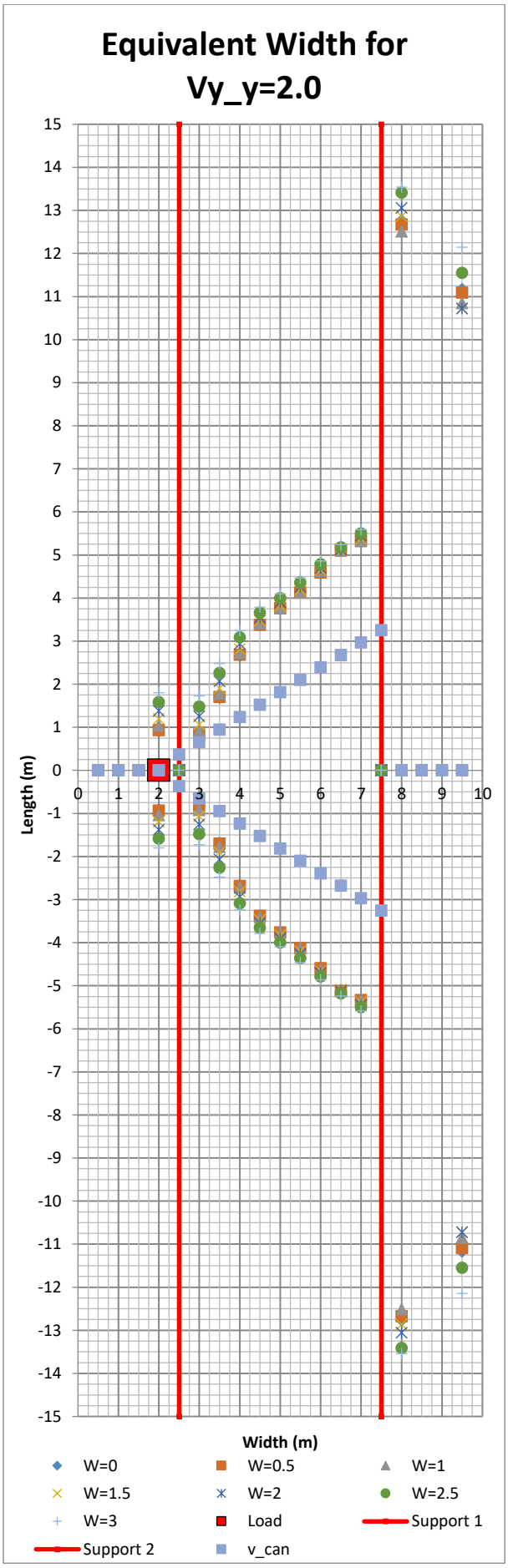

Equivalent Width for My_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

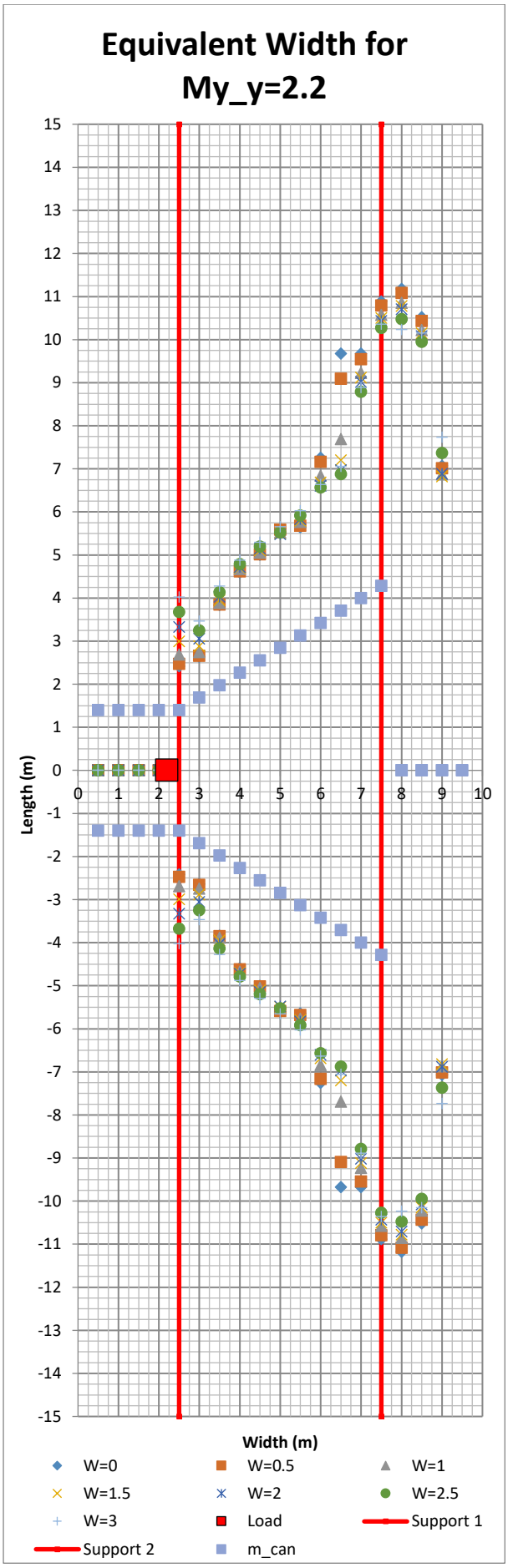

Equivalent Width for Vy_y=1.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

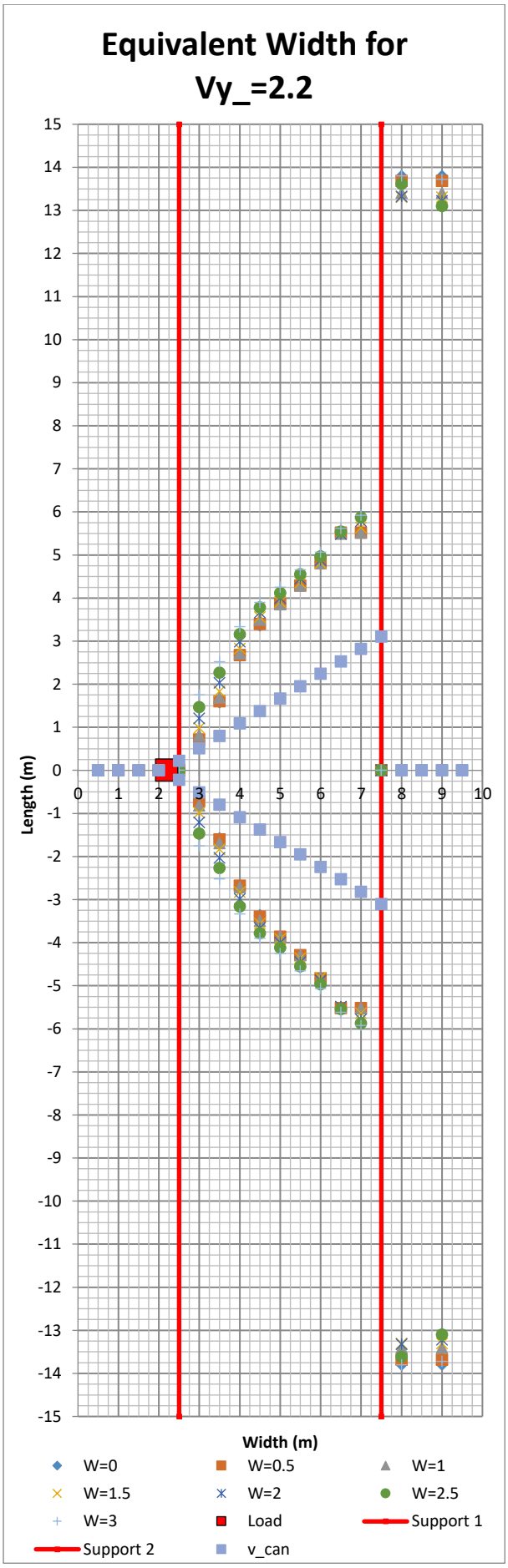

Equivalent Width for My_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

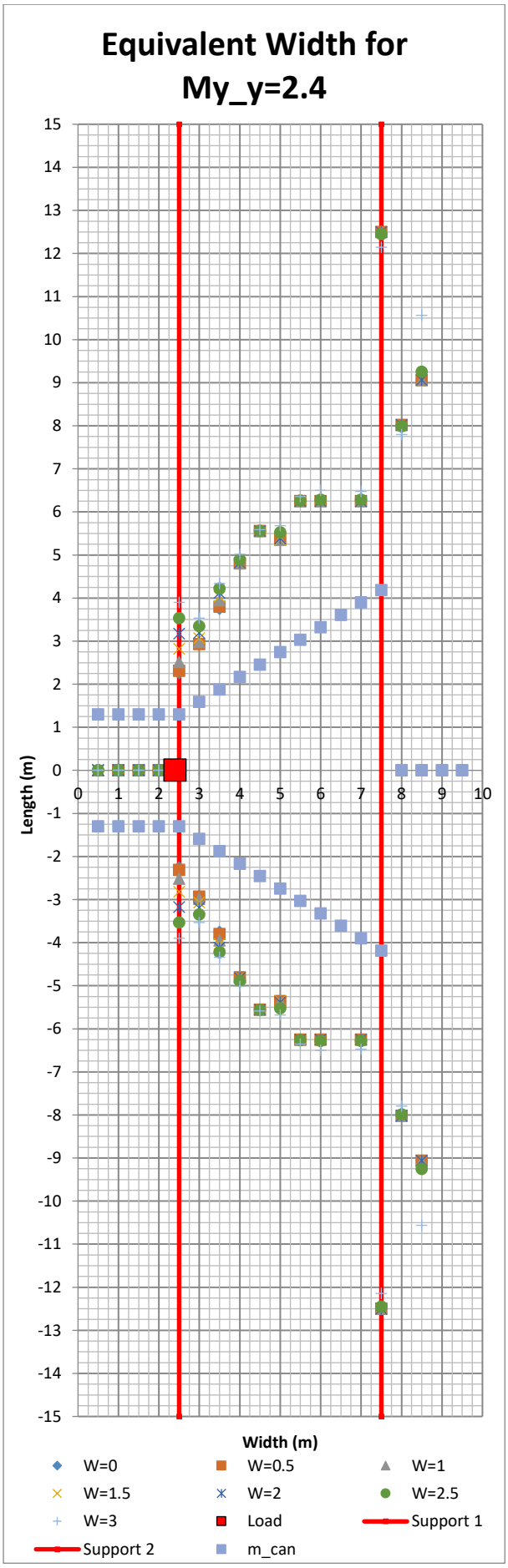

Equivalent Width for Vy_y=1.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

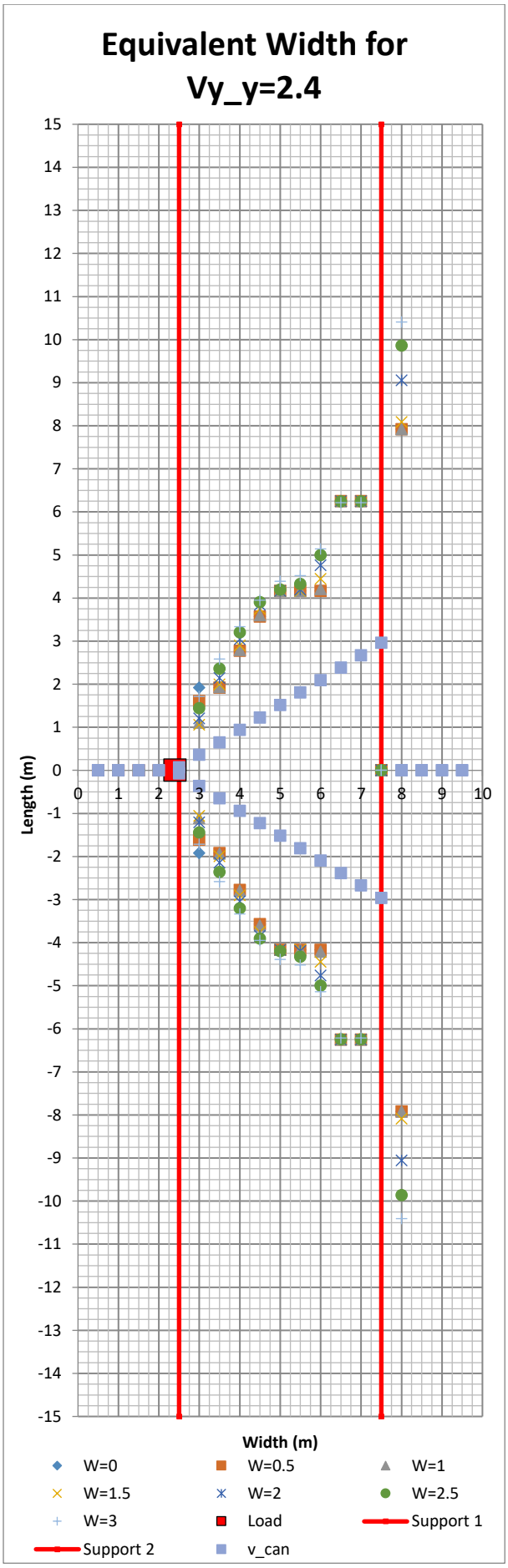

Equivalent Width for My_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

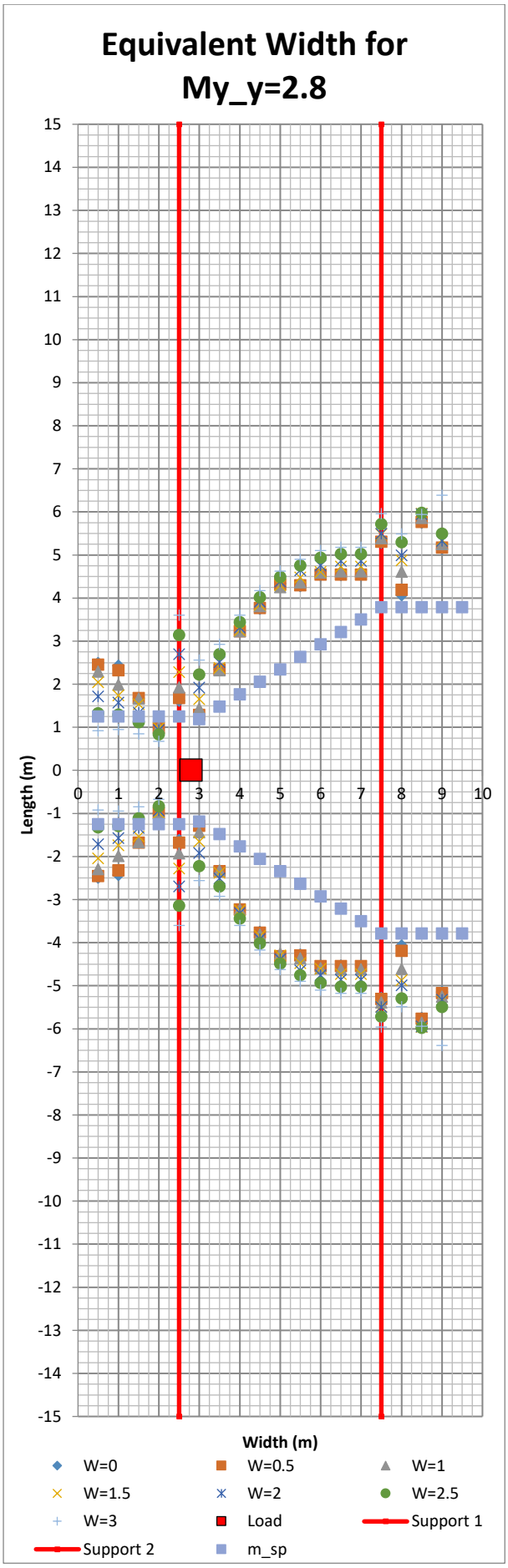

Equivalent Width for Vy_y=2.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

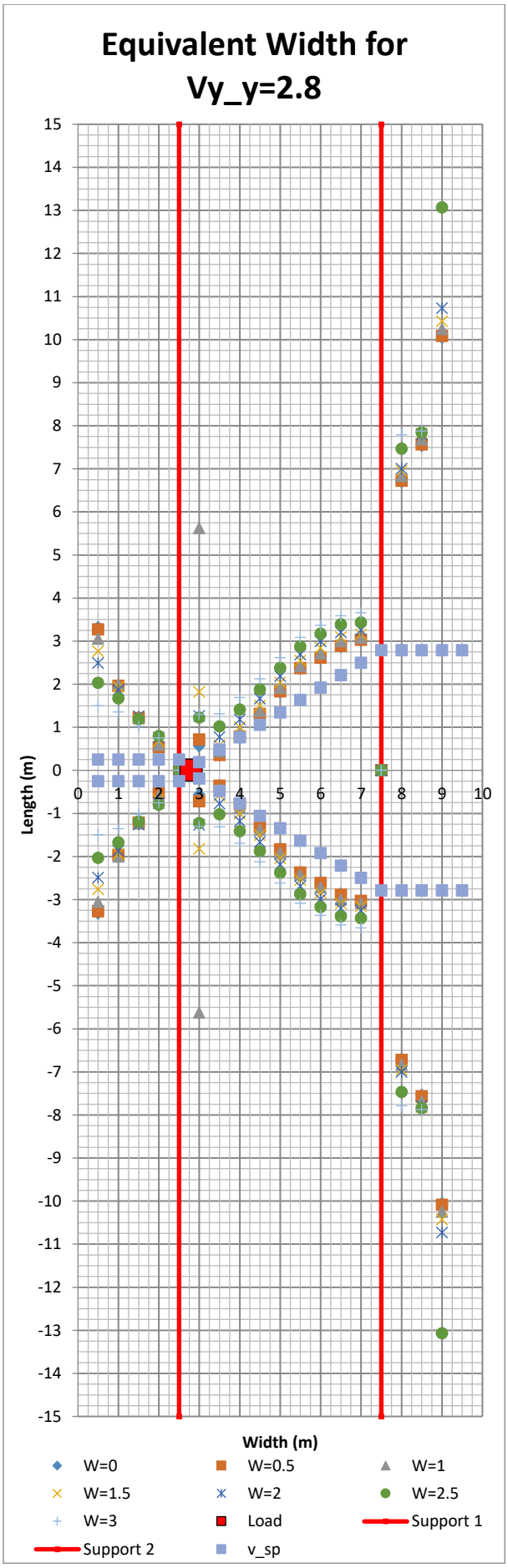

Equivalent Width for My_y=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

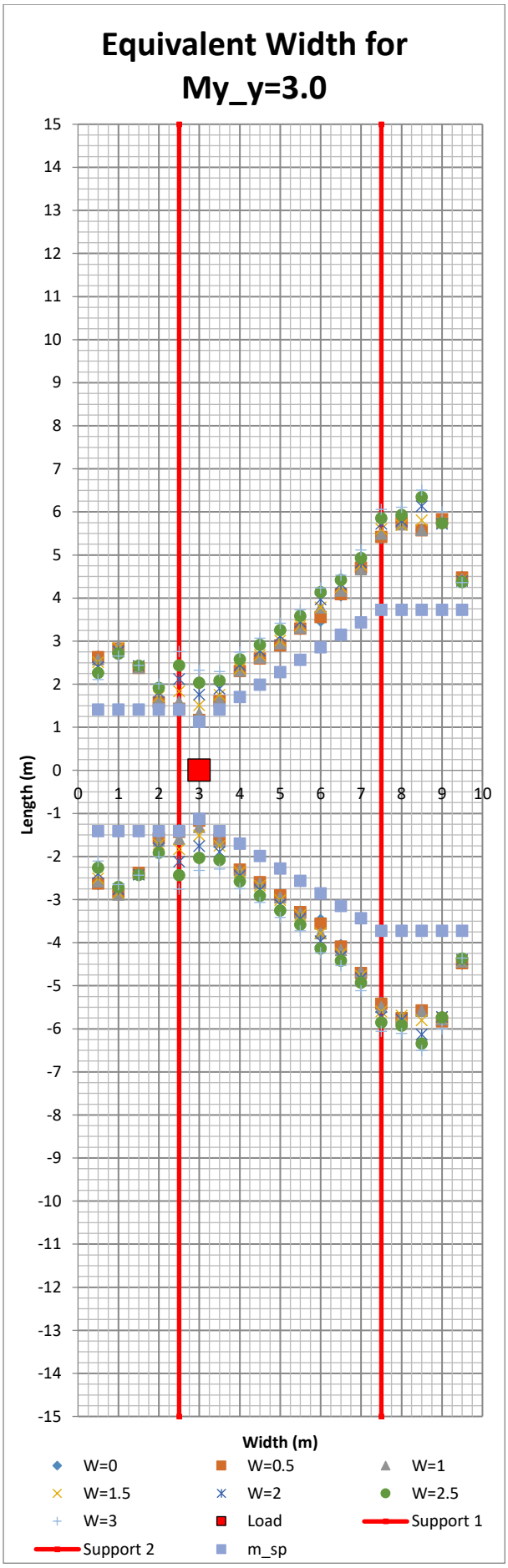

Equivalent Width for Vy_=2.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

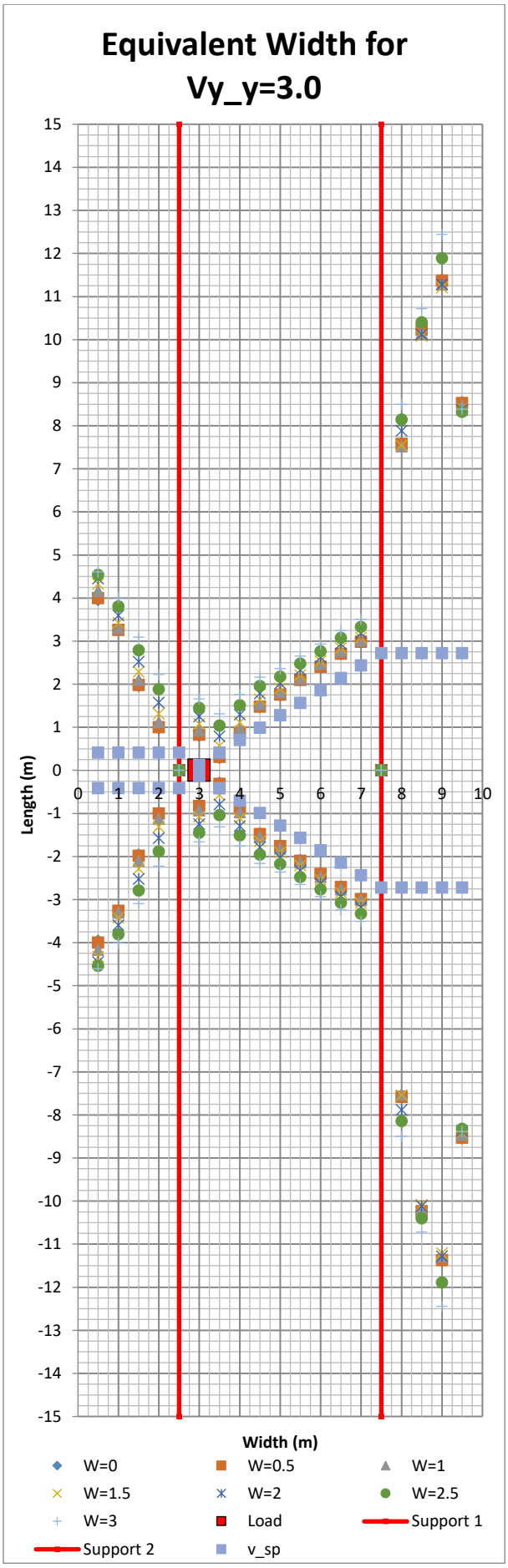

Equivalent Width for My_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_can

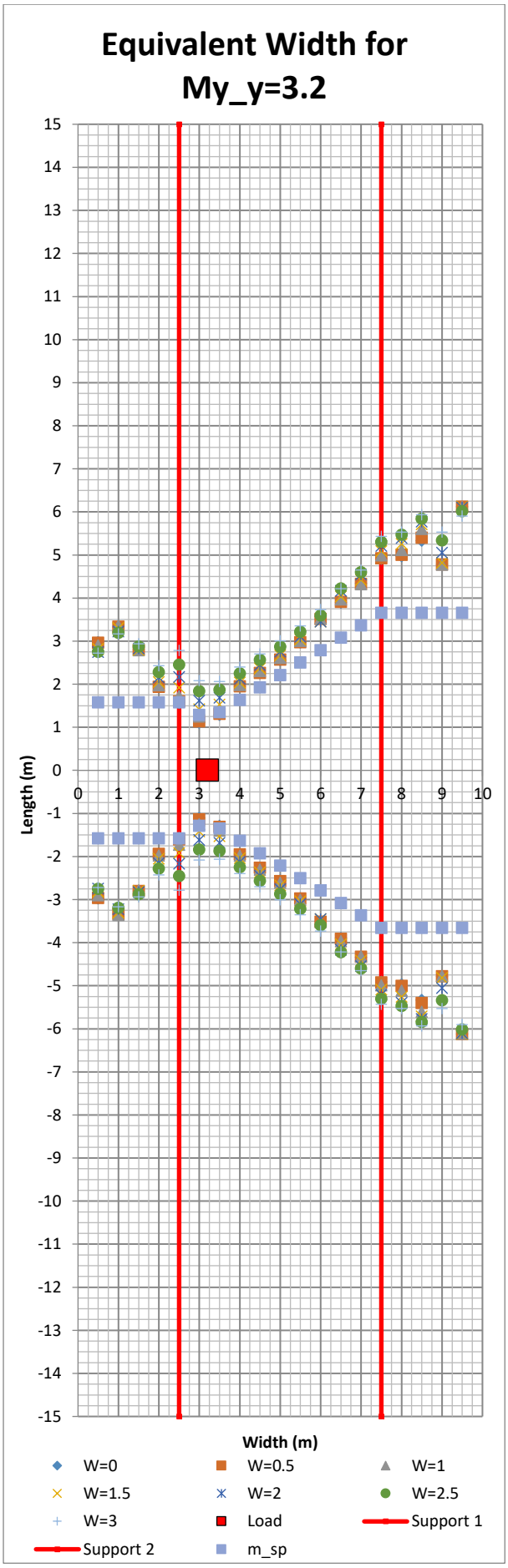

Equivalent Width for Vy_y=2.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_can

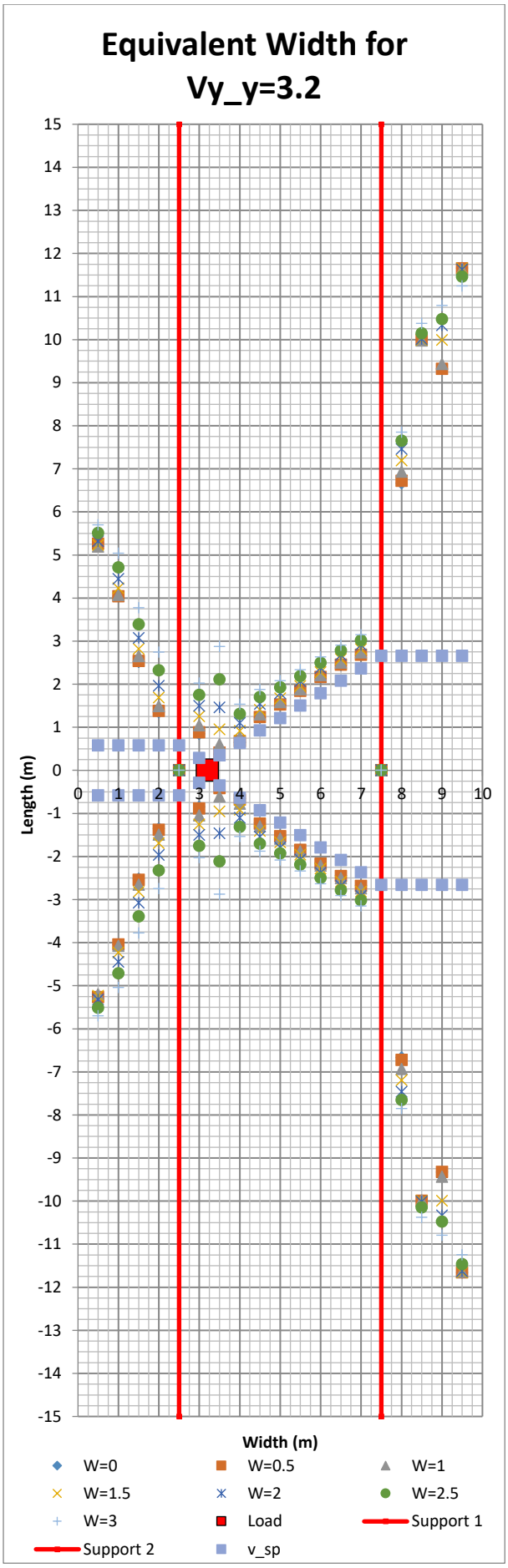

Equivalent Width for My_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

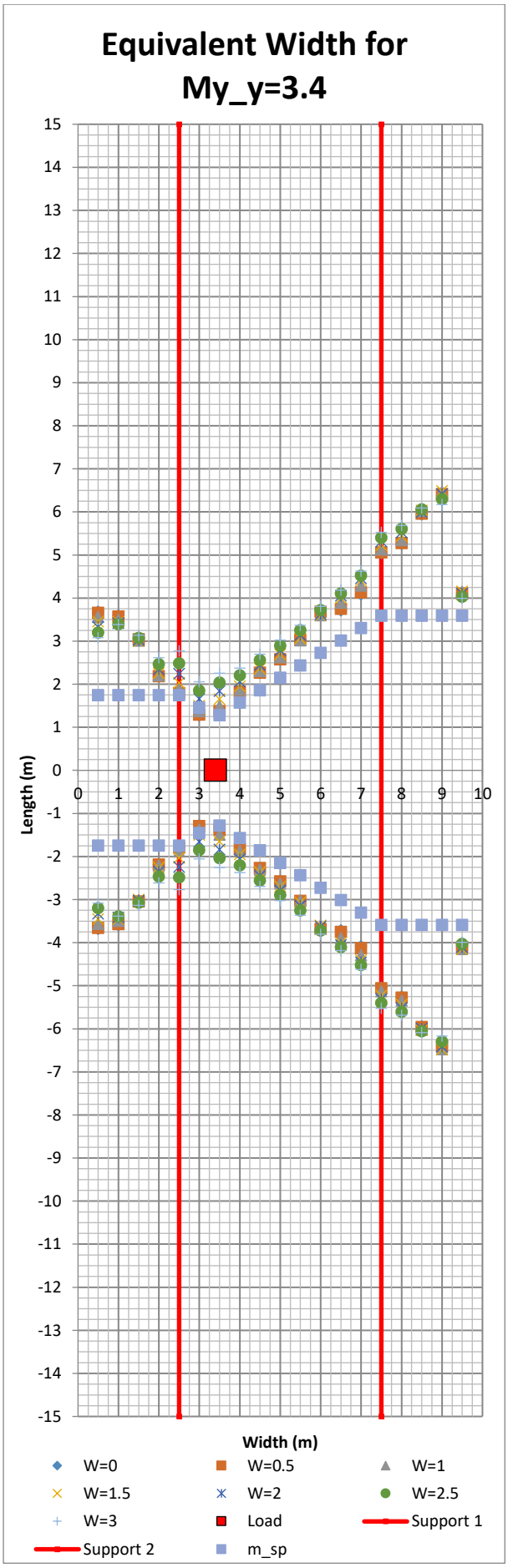

Equivalent Width for Vy_y=2.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

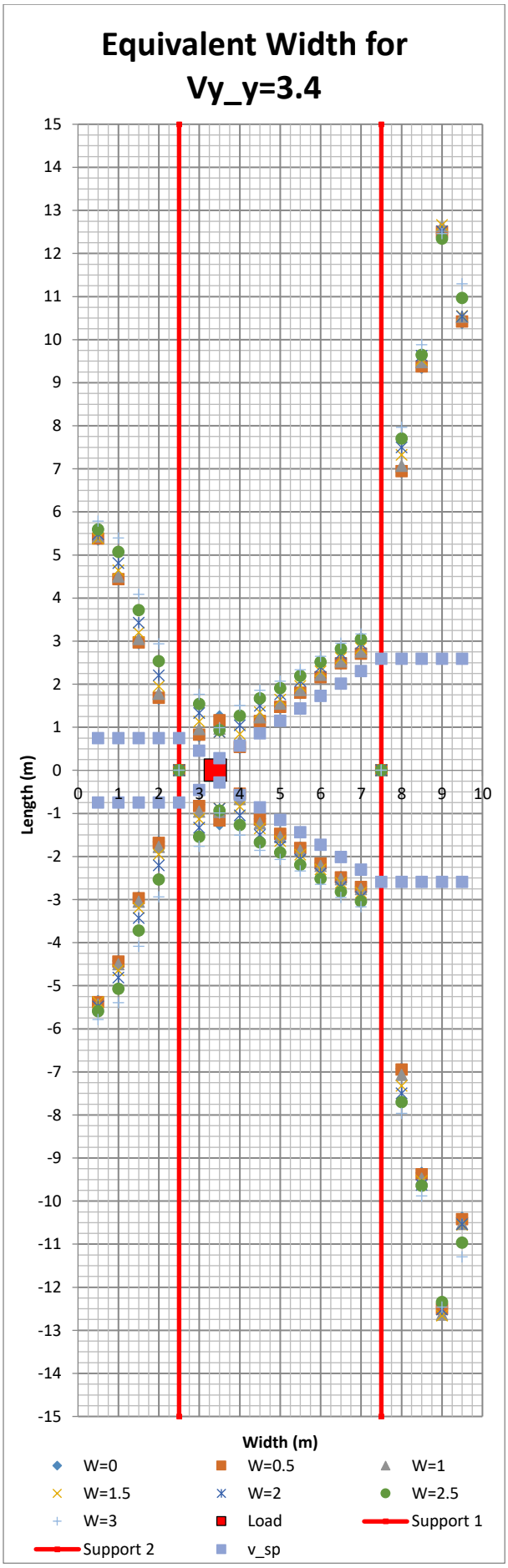

Equivalent Width for My_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

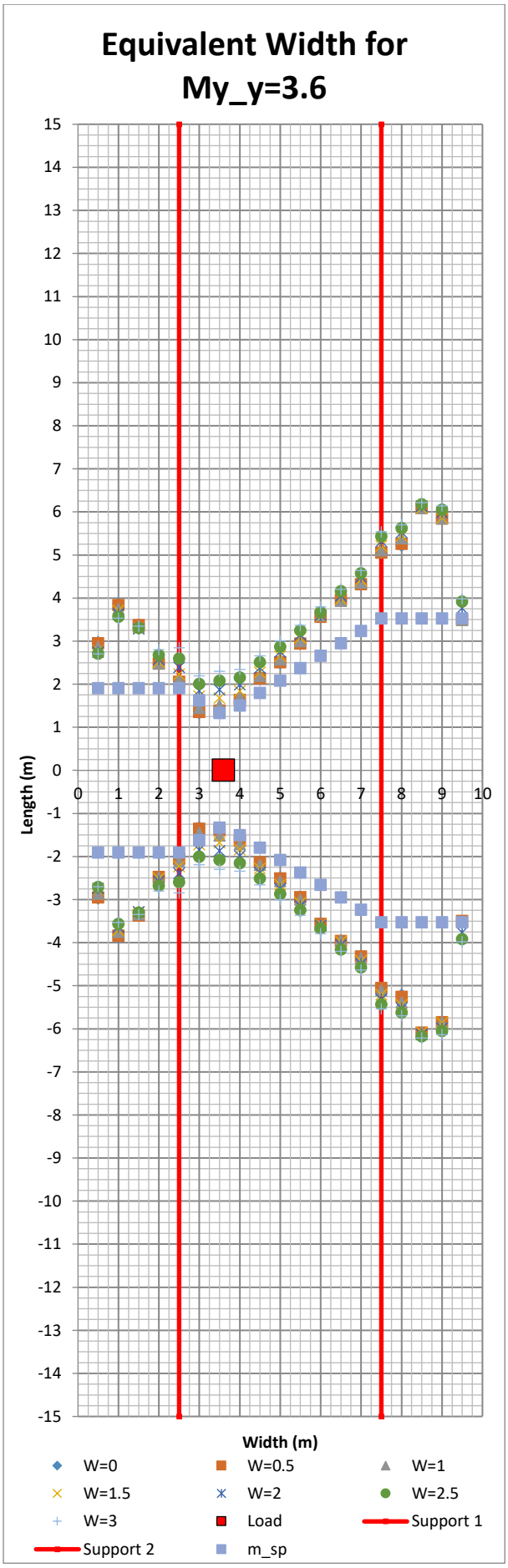

Equivalent Width for Vy_y=3.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

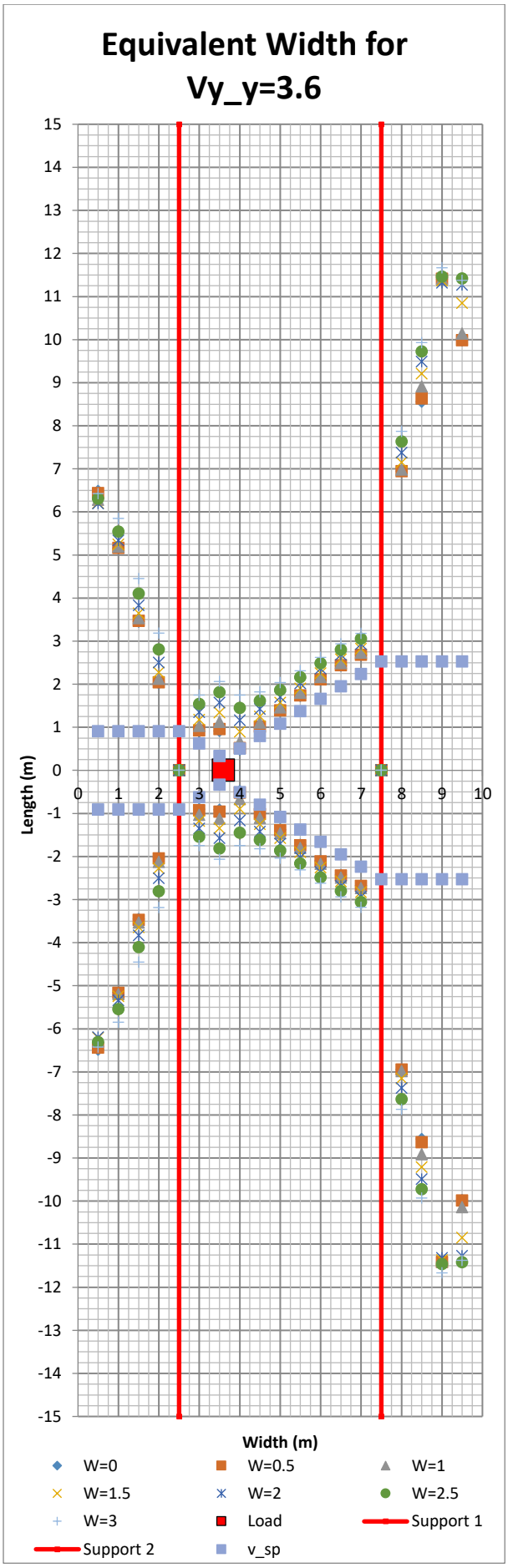

Equivalent Width for My_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

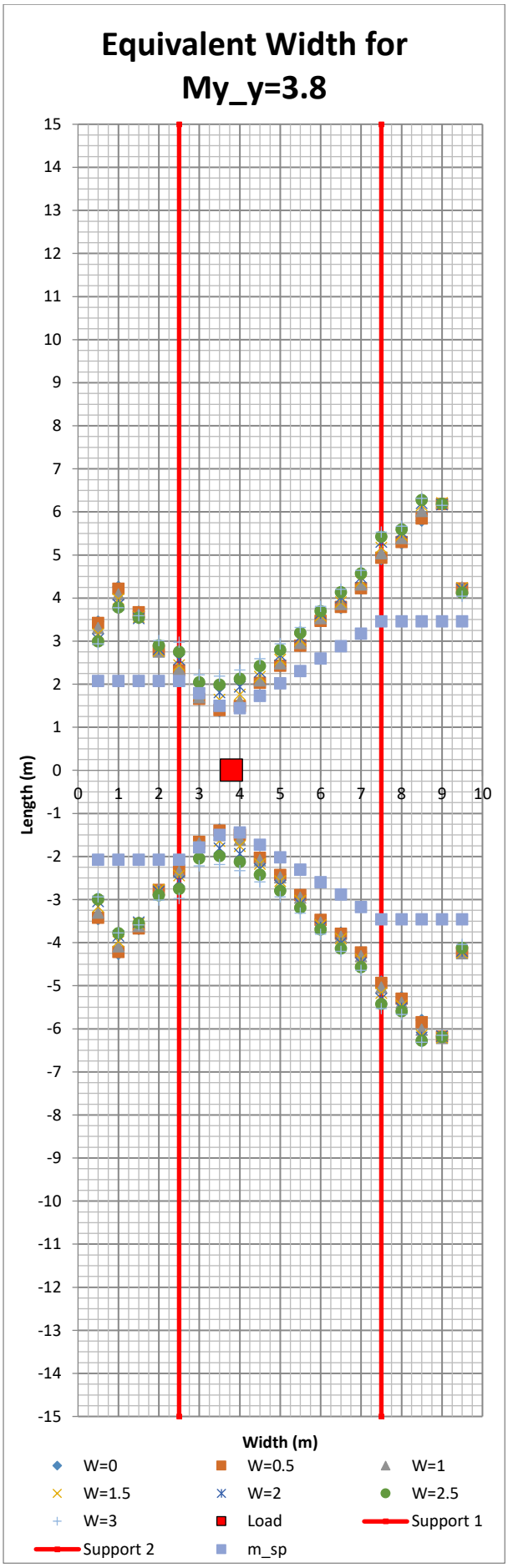

Equivalent Width for Vy_y=3.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

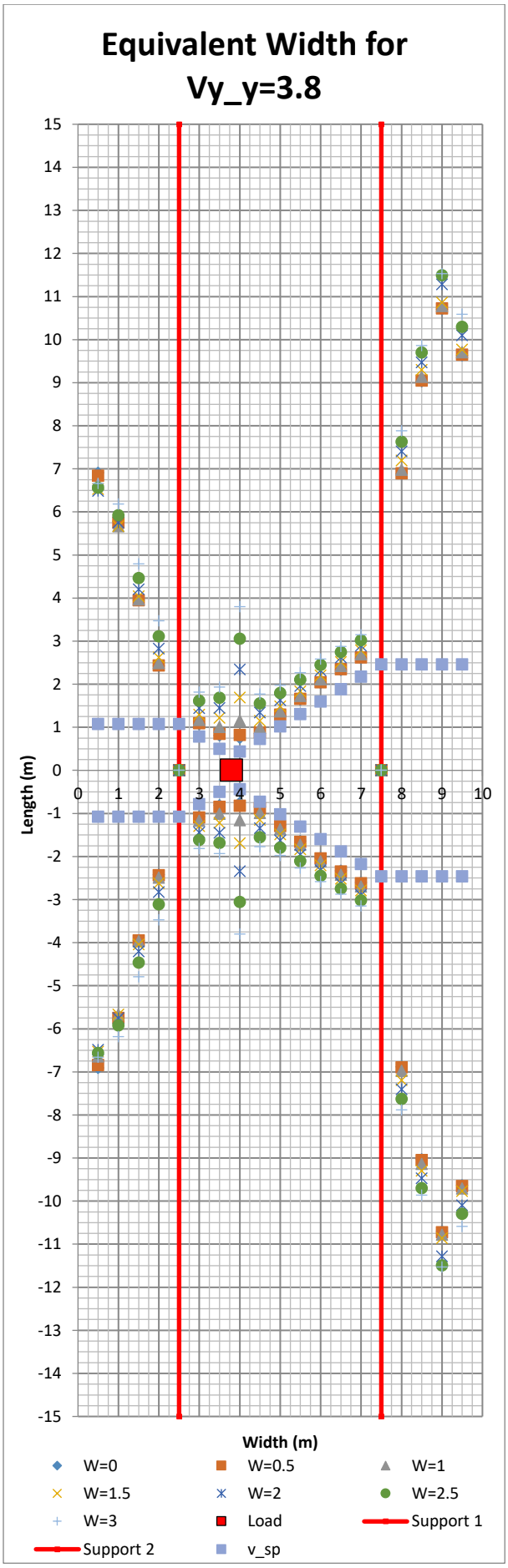

Equivalent Width for My_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

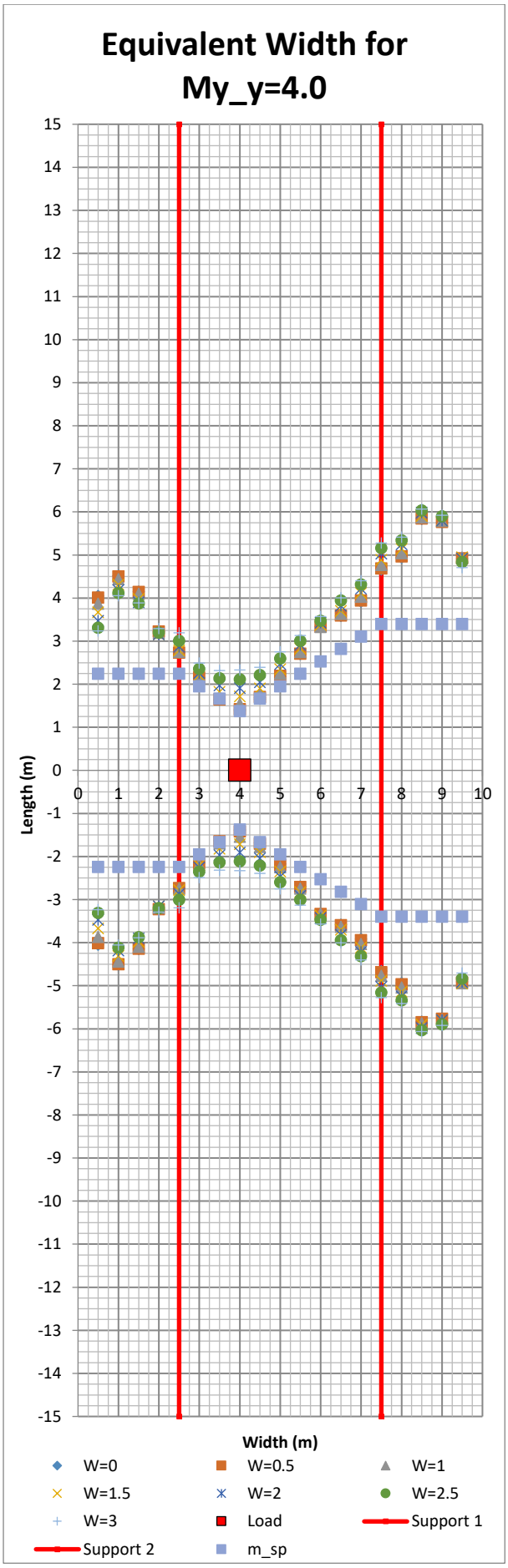

Equivalent Width for Vy_y=3.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

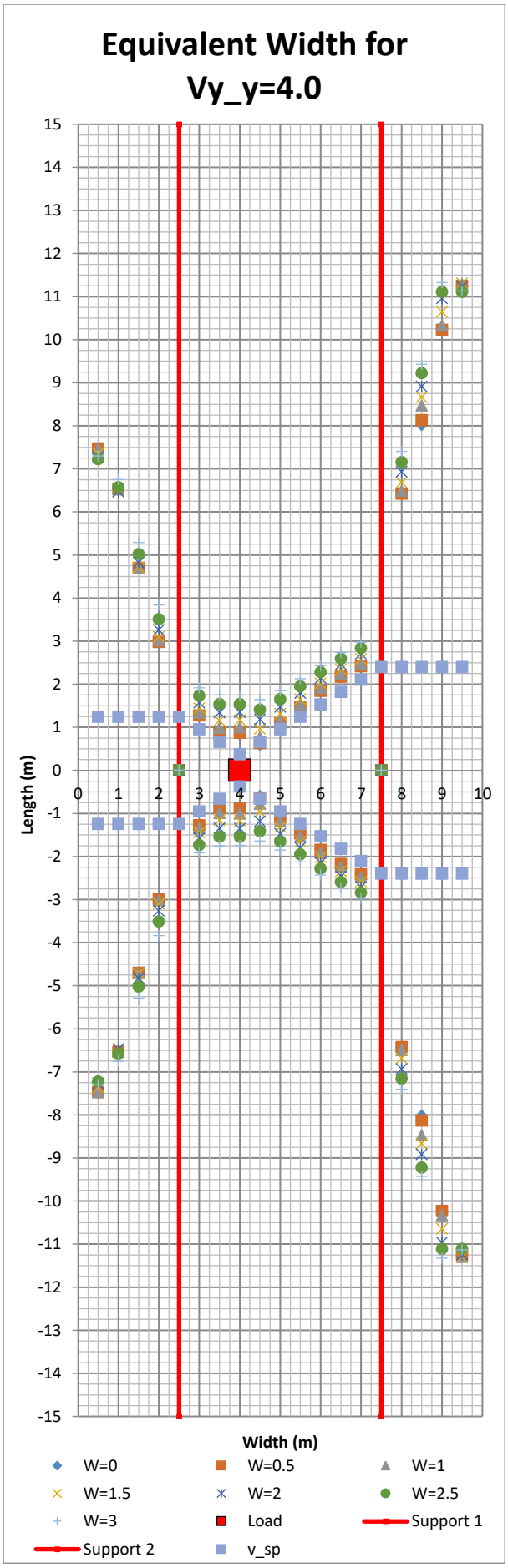

Equivalent Width for My_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

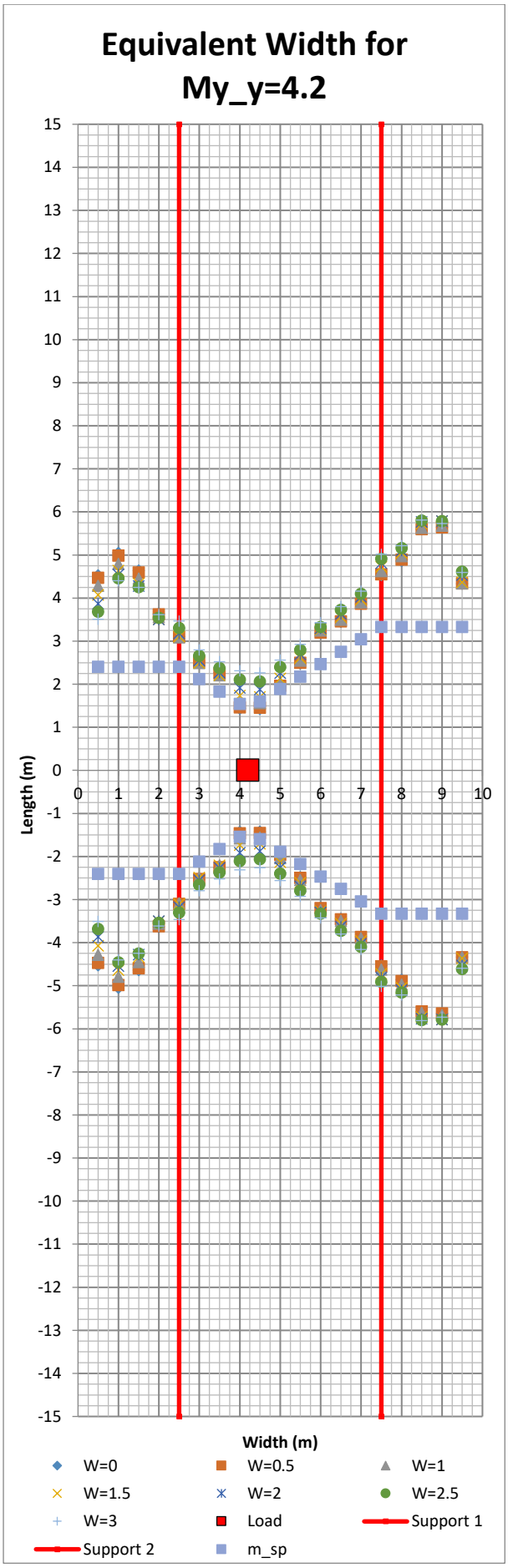

Equivalent Width for Vy_y=3.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

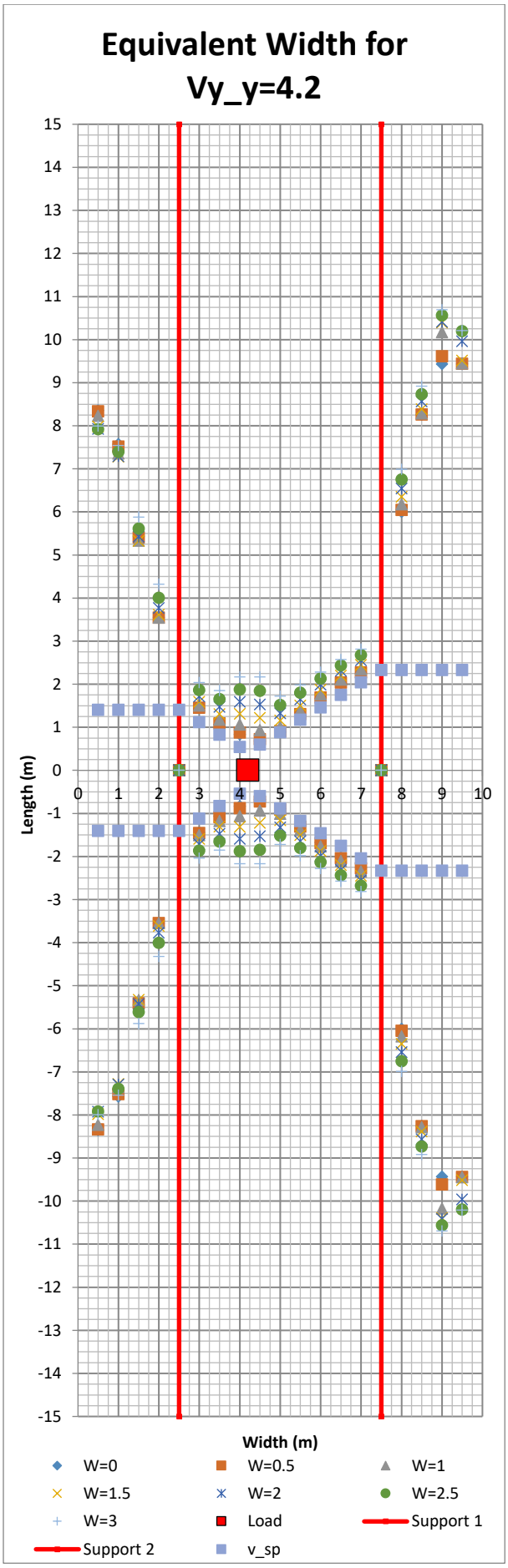

Equivalent Width for My_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

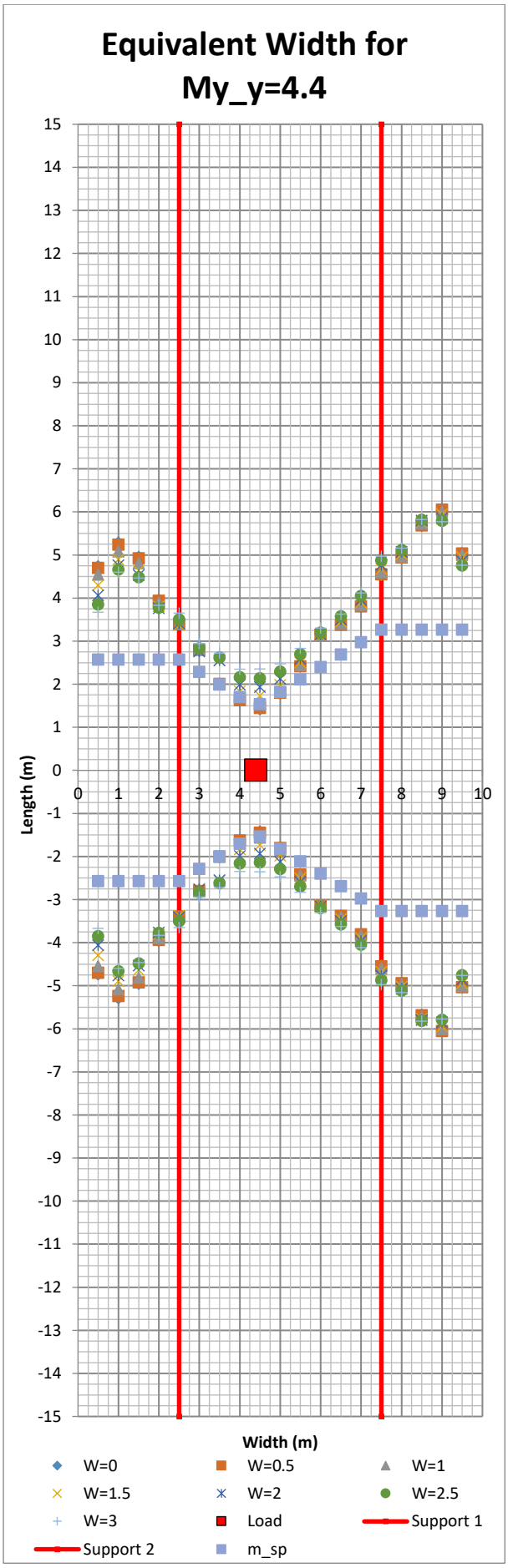

Equivalent Width for Vy_y=3.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

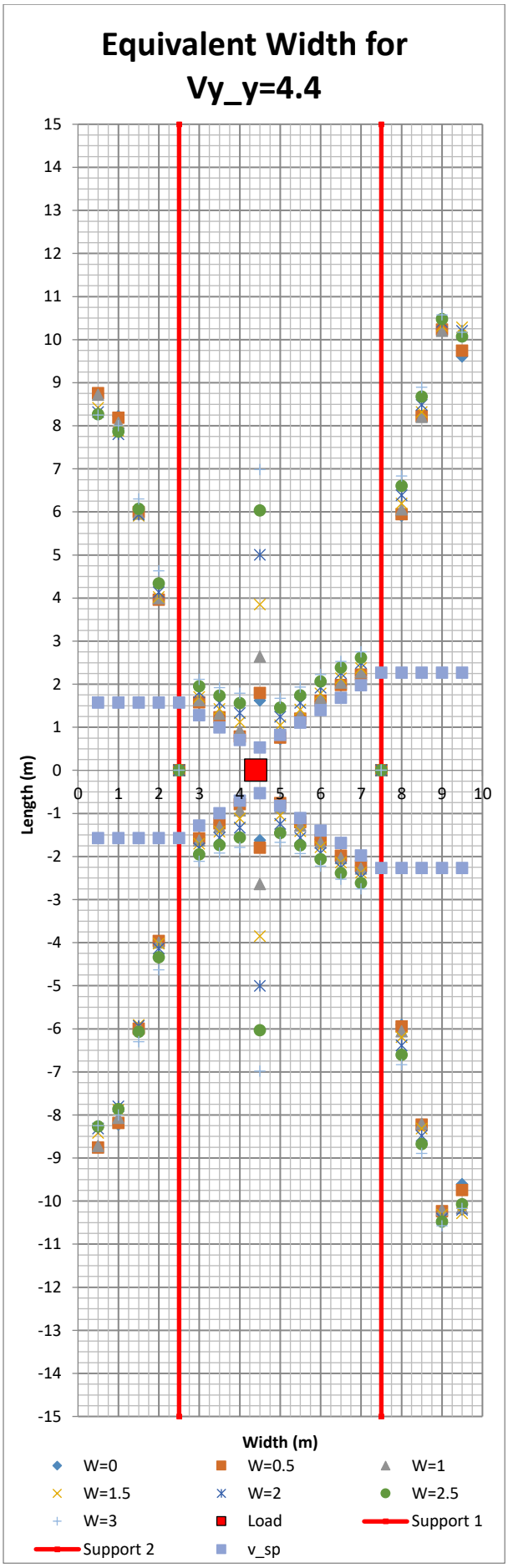

Equivalent Width for My_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

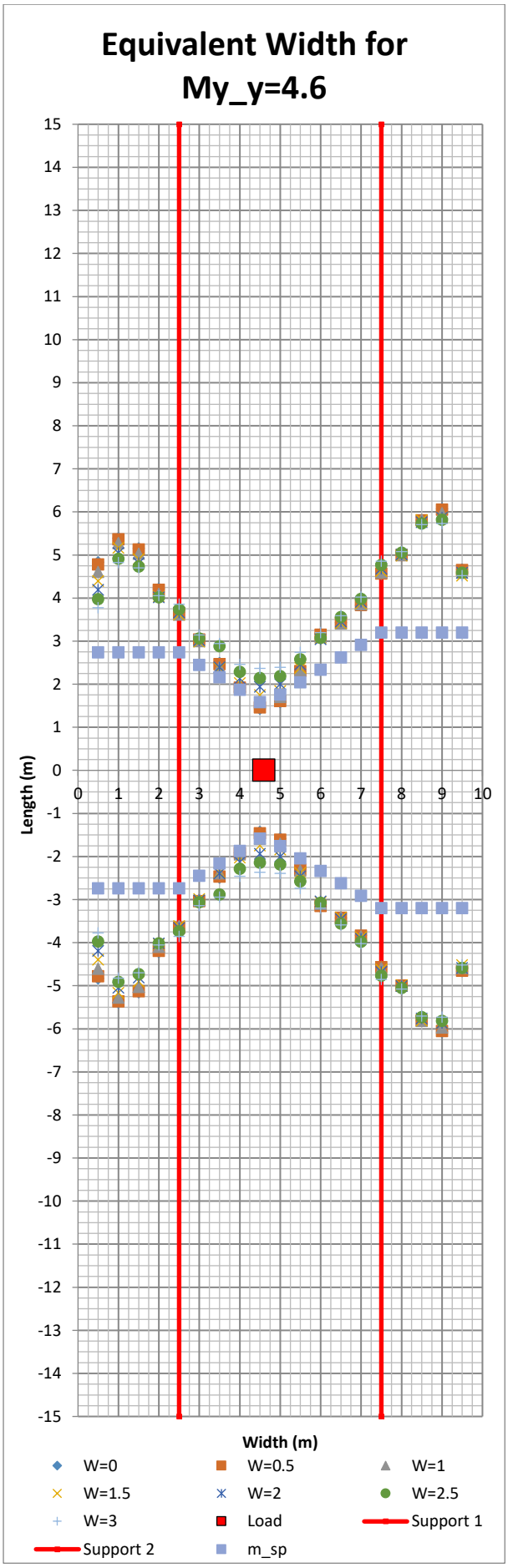

Equivalent Width for Vy_y=4.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

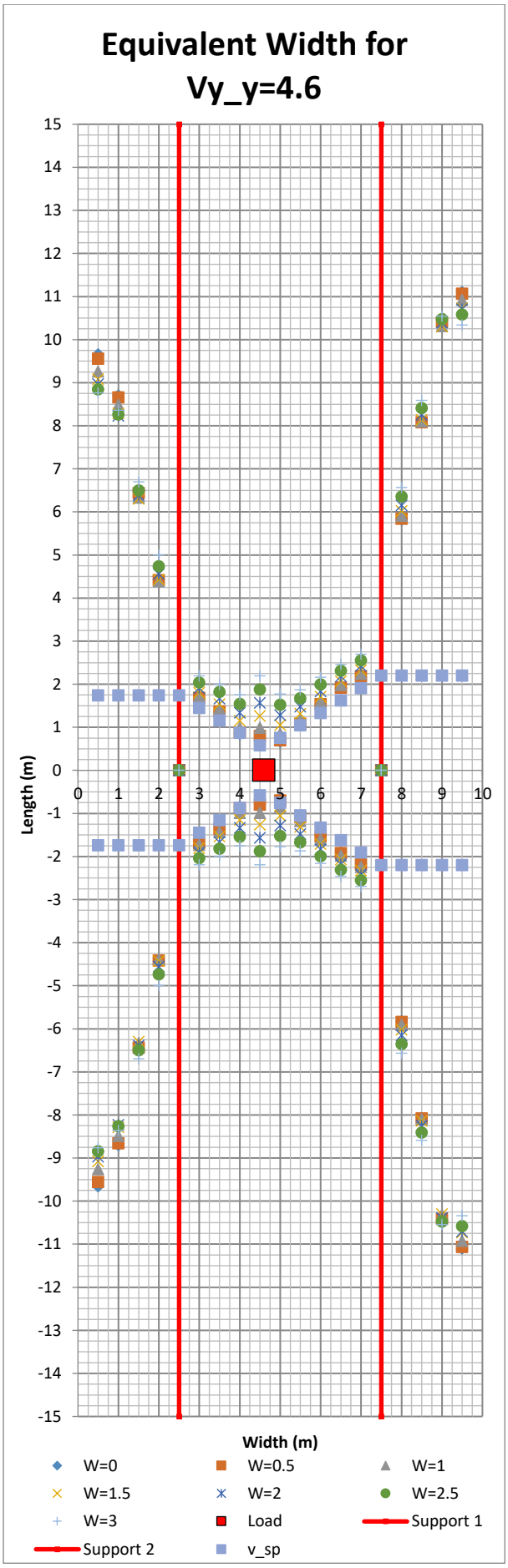

Equivalent Width for My_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

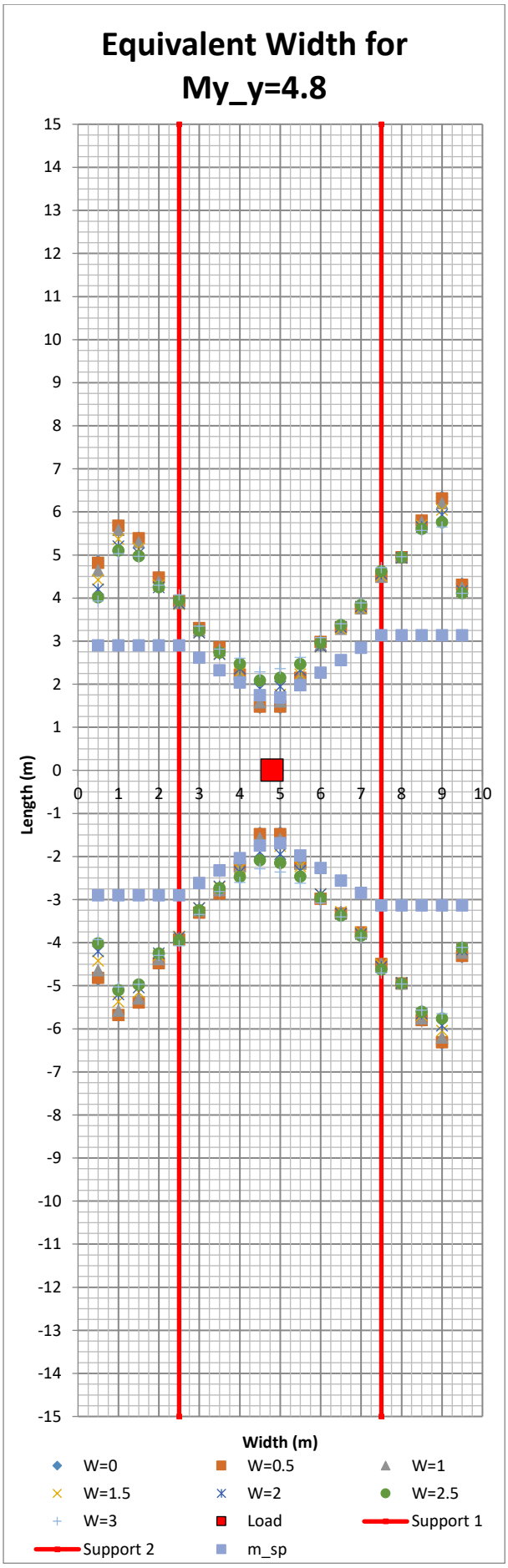

Equivalent Width for Vy_y=4.2
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

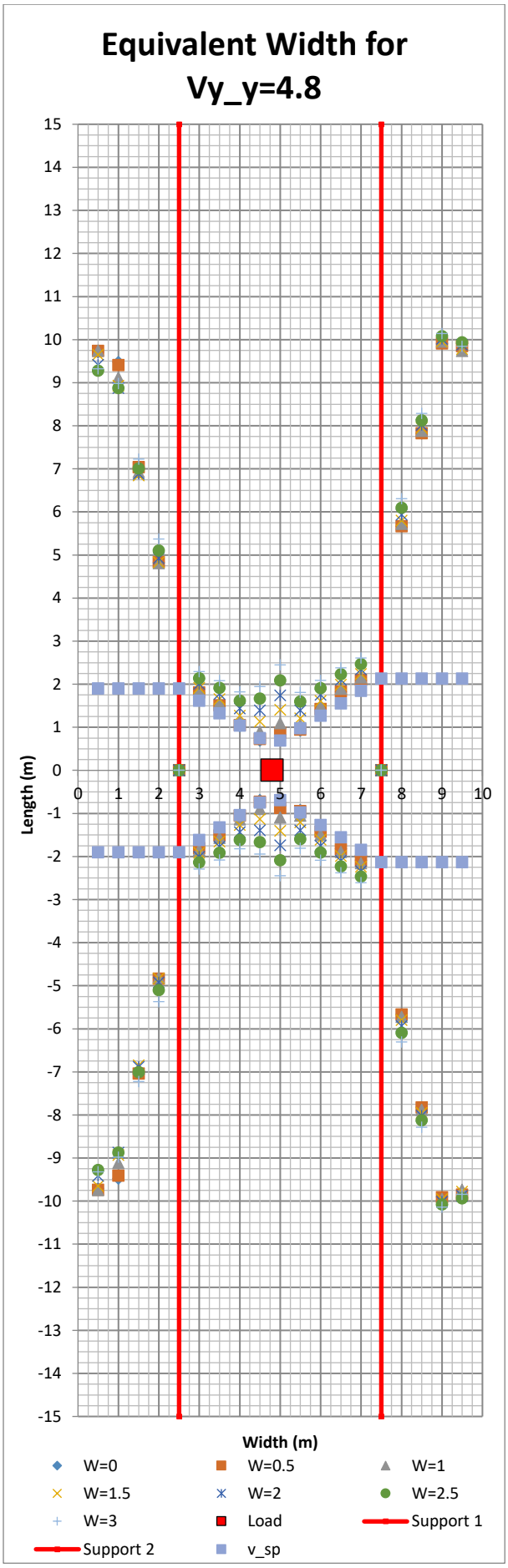

Equivalent Width for My_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp

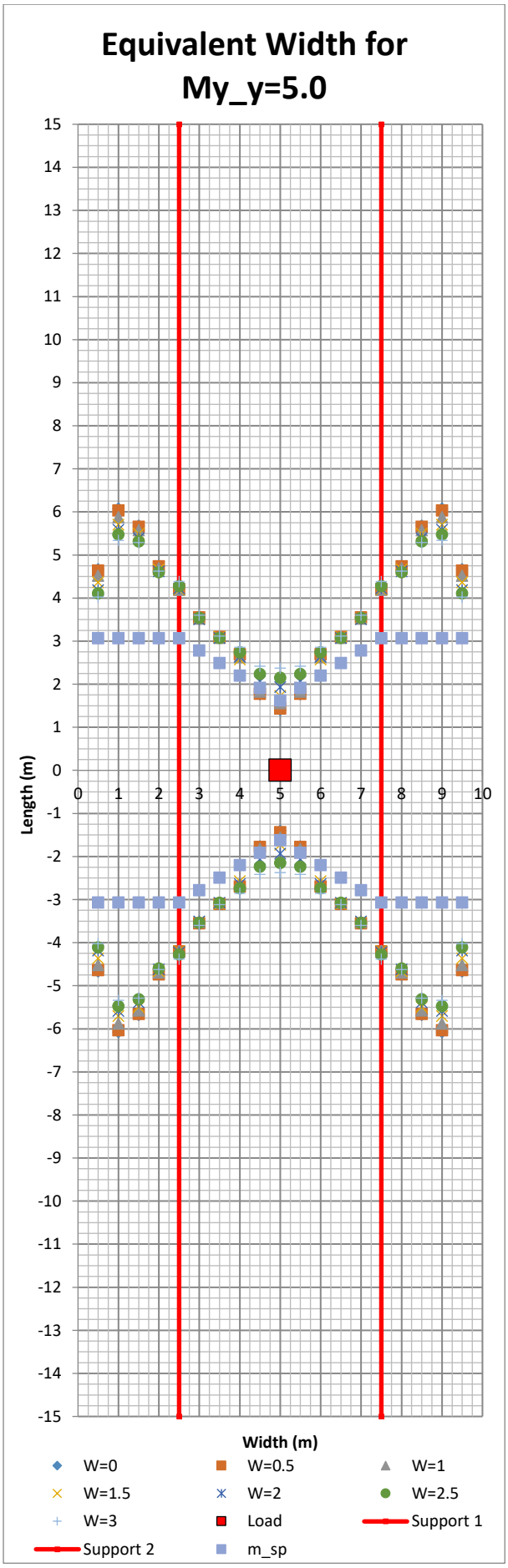

Equivalent Width for Vy_y=4.4
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

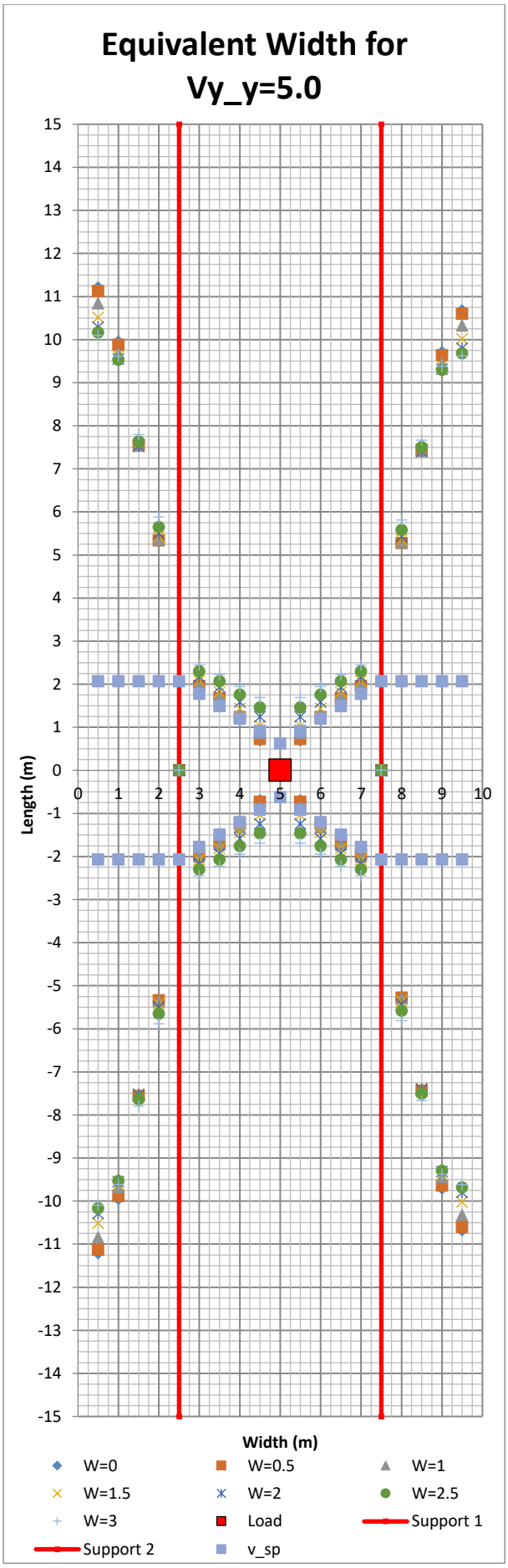

Equivalent Width for My_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=4.6
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp


Equivalent Width for My_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=4.8
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp


Equivalent Width for My_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
m_sp


Equivalent Width for Vy_y=5.0
Length (m)
Width (m)
W=0
W=0.5
W=1
W=1.5
W=2
W=2.5
W=3
Load
Support 1
Support 2
v_sp

12 APPENDIX D: Python script

12.1 Directory structure:

```
./BSC/_coor.bsc
./BSC/_M2.bsc
./BSC/_Mxy.bsc
./BSC/_My.bsc
./BSC/_Vy.bsc
./temp/Diagrams
./xlsx
./fdAPI_wrapper_mod.py
./Main.py
./plate.py
./Parser.py
./x_section_w.py
./y_intergration_w.py
./handcal.py
./exp_env.py
./XLSX.py
```

12.2_coor.bsc


```
1. <?xml version="1.0" encoding="UTF-8"?><fdscript
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="fdscript.xsd"><fdscriptheader><title>
   FEM-Design Batch
   template</title><version>1900</version><module>sframe</module><logfi
   le>batchtable.log</logfile></fdscriptheader><cmddoctable
   command=""><doctable><listdll></listdll><listproc>femNode_ListProc</
   listproc><font><name>Tahoma</name><type>DEFAULT_CHARSET</type><size>
   0.003</size><width>1</width><slant>0</slant></font><version>1900</ve
   rsion><index>0</index><suffix></suffix><coldata><num>0</num><format>
   %ld</format><width>15</width><flags>0</flags></coldata><coldata><num
   >1</num><format>%.3f</format><width>15</width><flags>0</flags></cold
   ata><coldata><num>2</num><format>%.3f</format><width>15</width><flag
   s>0</flags></coldata><coldata><num>3</num><format>%.3f</format><widt
   h>15</width><flags>0</flags></coldata><units><num>0</num><unit>0</un
   it></units><units><num>1</num><unit>1</unit></units><units><num>2</n
   um><unit>3</unit></units><units><num>3</num><unit>2</unit></units><u
   nits><num>4</num><unit>0</unit></units><units><num>5</num><unit>0</u
   nit></units><units><num>6</num><unit>0</unit></units><units><num>7</
   num><unit>4</unit></units><units><num>8</num><unit>0</unit></units><
   units><num>9</num><unit>0</unit></units><units><num>10</num><unit>0<
   /unit></units><units><num>11</num><unit>0</unit></units><units><num>
   12</num><unit>0</unit></units><units><num>13</num><unit>0</unit></un
   its><units><num>14</num><unit>0</unit></units><units><num>15</num><u
   nit>0</unit></units><units><num>16</num><unit>0</unit></units><units
   ><num>17</num><unit>0</unit></units><units><num>18</num><unit>0</uni
   t></units><units><num>19</num><unit>0</unit></units><units><num>20</
   num><unit>0</unit></units><units><num>21</num><unit>0</unit></units>
   <units><num>22</num><unit>0</unit></units><units><num>23</num><unit>
   0</unit></units><units><num>24</num><unit>0</unit></units><units><nu
   m>25</num><unit>0</unit></units><units><num>26</num><unit>0</unit></
   units><units><num>27</num><unit>0</unit></units><units><num>28</num>
   <unit>0</unit></units><units><num>29</num><unit>0</unit></units><uni
   ts><num>30</num><unit>0</unit></units><units><num>31</num><unit>0</u
   nit></units><units><num>32</num><unit>0</unit></units><units><num>33
   </num><unit>0</unit></units><units><num>34</num><unit>0</unit></unit
   s><units><num>35</num><unit>0</unit></units><units><num>36</num><uni
   t>0</unit></units><units><num>37</num><unit>0</unit></units><units><
   num>38</num><unit>0</unit></units><units><num>39</num><unit>0</unit>
   </units><units><num>40</num><unit>0</unit></units><units><num>41</nu
   m><unit>0</unit></units><units><num>42</num><unit>0</unit></units><u
   nits><num>43</num><unit>0</unit></units><units><num>44</num><unit>0<
   /unit></units><units><num>45</num><unit>0</unit></units><units><num>
   46</num><unit>0</unit></units><units><num>47</num><unit>0</unit></un
   its><units><num>48</num><unit>0</unit></units><units><num>49</num><u
   nit>0</unit></units><units><num>50</num><unit>0</unit></units><units
   ><num>51</num><unit>0</unit></units><units><num>52</num><unit>0</uni
   t></units><units><num>53</num><unit>0</unit></units><units><num>54</
   num><unit>0</unit></units><units><num>55</num><unit>0</unit></units>
   <units><num>56</num><unit>0</unit></units><units><num>57</num><unit>
   0</unit></units><units><num>58</num><unit>0</unit></units><units><nu
   m>59</num><unit>0</unit></units><units><num>60</num><unit>0</unit></
   units><units><num>61</num><unit>0</unit></units><units><num>62</num>
   <unit>0</unit></units><units><num>63</num><unit>0</unit></units><res
   type>0</restype></doctable></cmddoctable><cmdendsession/></fdscript>
```

12.3 _M2.bsc


```
1. <?xml version="1.0" encoding="UTF-8"?><fdscript
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="fdscript.xsd"><fdscriptheader><title>
   FEM-Design Batch
   template</title><version>1900</version><module>sframe</module><logfi
   le>batchtable.log</logfile></fdscriptheader><cmddoctable
   command=""><doctable><listdll></listdll><listproc>frCaseIntfDerShell
   _ListProc</listproc><font><name>Tahoma</name><type>DEFAULT_CHARSET</
   type><size>0.003</size><width>1</width><slant>0</slant></font><versi
   on>1900</version><index>65536</index><suffix>Ultimate - Load case:
   LL</suffix><coldata><num>0</num><format>%s</format><width>12</width>
   <flags>0</flags></coldata><coldata><num>1</num><format>%s</format><w
   idth>15</width><flags>0</flags></coldata><coldata><num>2</num><forma
   t>%ld</format><width>15</width><flags>0</flags></coldata><coldata><n
   um>3</num><format>%.3f</format><width>15</width><flags>0</flags></co
   ldata><coldata><num>4</num><format>%.3f</format><width>15</width><fl
   ags>0</flags></coldata><coldata><num>5</num><format>%.3f</format><wi
   dth>15</width><flags>0</flags></coldata><coldata><num>6</num><format
   >%.3f</format><width>15</width><flags>0</flags></coldata><coldata><n
   um>7</num><format>%.3f</format><width>15</width><flags>0</flags></co
   ldata><coldata><num>8</num><format>%.3f</format><width>15</width><fl
   ags>0</flags></coldata><coldata><num>9</num><format>%s</format><widt
   h>15</width><flags>1</flags></coldata><units><num>0</num><unit>0</un
   it></units><units><num>1</num><unit>1</unit></units><units><num>2</n
   um><unit>3</unit></units><units><num>3</num><unit>2</unit></units><u
   nits><num>4</num><unit>0</unit></units><units><num>5</num><unit>0</u
   nit></units><units><num>6</num><unit>0</unit></units><units><num>7</
   num><unit>4</unit></units><units><num>8</num><unit>0</unit></units><
   units><num>9</num><unit>0</unit></units><units><num>10</num><unit>0<
   /unit></units><units><num>11</num><unit>0</unit></units><units><num>
   12</num><unit>0</unit></units><units><num>13</num><unit>0</unit></un
   its><units><num>14</num><unit>0</unit></units><units><num>15</num><u
   nit>0</unit></units><units><num>16</num><unit>0</unit></units><units
   ><num>17</num><unit>0</unit></units><units><num>18</num><unit>0</uni
   t></units><units><num>19</num><unit>0</unit></units><units><num>20</
   num><unit>0</unit></units><units><num>21</num><unit>0</unit></units>
   <units><num>22</num><unit>0</unit></units><units><num>23</num><unit>
   0</unit></units><units><num>24</num><unit>0</unit></units><units><nu
   m>25</num><unit>0</unit></units><units><num>26</num><unit>0</unit></
   units><units><num>27</num><unit>0</unit></units><units><num>28</num>
   <unit>0</unit></units><units><num>29</num><unit>0</unit></units><uni
   ts><num>30</num><unit>0</unit></units><units><num>31</num><unit>0</u
   nit></units><units><num>32</num><unit>0</unit></units><units><num>33
   </num><unit>0</unit></units><units><num>34</num><unit>0</unit></unit
   s><units><num>35</num><unit>0</unit></units><units><num>36</num><uni
   t>0</unit></units><units><num>37</num><unit>0</unit></units><units><
   num>38</num><unit>0</unit></units><units><num>39</num><unit>0</unit>
   </units><units><num>40</num><unit>0</unit></units><units><num>41</nu
   m><unit>0</unit></units><units><num>42</num><unit>0</unit></units><u
   nits><num>43</num><unit>0</unit></units><units><num>44</num><unit>0<
   /unit></units><units><num>45</num><unit>0</unit></units><units><num>
   46</num><unit>0</unit></units><units><num>47</num><unit>0</unit></un
   its><units><num>48</num><unit>0</unit></units><units><num>49</num><u
   nit>0</unit></units><units><num>50</num><unit>0</unit></units><units
   ><num>51</num><unit>0</unit></units><units><num>52</num><unit>0</uni
```

```
t></units><units><num>53</num><unit>0</unit></units><units><num>54</
num><unit>0</unit></units><units><num>55</num><unit>0</unit></units>
<units><num>56</num><unit>0</unit></units><units><num>57</num><unit>
0</unit></units><units><num>58</num><unit>0</unit></units><units><nu
m>59</num><unit>0</unit></units><units><num>60</num><unit>0</unit></
units><units><num>61</num><unit>0</unit></units><units><num>62</num>
<unit>0</unit></units><units><num>63</num><unit>0</unit></units><opt
ions><surface>0</surface></options><restype>1</restype></doctable></
cmddoctable><cmdendsession/></fdscript>
```

12.4 _My.bsc , _Mxy.bsc, and _Vy.bsc


```
1. <?xml version="1.0" encoding="UTF-8"?><fdscript
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="fdscript.xsd"><fdscriptheader><title>
   FEM-Design Batch
   template</title><version>1900</version><module>sframe</module><logfi
   le>batchtable.log</logfile></fdscriptheader><cmddoctable
   command=""><doctable><listdll></listdll><listproc>frCaseIntfShell_Li
   stProc</listproc><font><name>Tahoma</name><type>DEFAULT_CHARSET</typ
   e><size>0.003</size><width>1</width><slant>0</slant></font><version>
   1900</version><index>65536</index><suffix>Ultimate - Load case:
   LL</suffix><coldata><num>0</num><format>%s</format><width>12</width>
   <flags>0</flags></coldata><coldata><num>1</num><format>%s</format><w
   idth>15</width><flags>0</flags></coldata><coldata><num>2</num><forma
   t>%ld</format><width>15</width><flags>0</flags></coldata><coldata><n
   um>3</num><format>%.3f</format><width>15</width><flags>0</flags></co
   ldata><coldata><num>4</num><format>%.3f</format><width>15</width><fl
   ags>0</flags></coldata><coldata><num>5</num><format>%.3f</format><wi
   dth>15</width><flags>0</flags></coldata><coldata><num>6</num><format
   >%.3f</format><width>15</width><flags>0</flags></coldata><coldata><n
   um>7</num><format>%.3f</format><width>15</width><flags>0</flags></co
   ldata><coldata><num>8</num><format>%.3f</format><width>15</width><fl
   ags>0</flags></coldata><coldata><num>9</num><format>%.3f</format><wi
   dth>15</width><flags>0</flags></coldata><coldata><num>10</num><forma
   t>%.3f</format><width>15</width><flags>0</flags></coldata><coldata><
   num>11</num><format>%s</format><width>15</width><flags>1</flags></co
   ldata><units><num>0</num><unit>0</unit></units><units><num>1</num><u
   nit>1</unit></units><units><num>2</num><unit>3</unit></units><units>
   <num>3</num><unit>2</unit></units><units><num>4</num><unit>0</unit><
   /units><units><num>5</num><unit>0</unit></units><units><num>6</num><
   unit>0</unit></units><units><num>7</num><unit>4</unit></units><units
   ><num>8</num><unit>0</unit></units><units><num>9</num><unit>0</unit>
   </units><units><num>10</num><unit>0</unit></units><units><num>11</nu
   m><unit>0</unit></units><units><num>12</num><unit>0</unit></units><u
   nits><num>13</num><unit>0</unit></units><units><num>14</num><unit>0<
   /unit></units><units><num>15</num><unit>0</unit></units><units><num>
   16</num><unit>0</unit></units><units><num>17</num><unit>0</unit></un
   its><units><num>18</num><unit>0</unit></units><units><num>19</num><u
   nit>0</unit></units><units><num>20</num><unit>0</unit></units><units
   ><num>21</num><unit>0</unit></units><units><num>22</num><unit>0</uni
   t></units><units><num>23</num><unit>0</unit></units><units><num>24</
   num><unit>0</unit></units><units><num>25</num><unit>0</unit></units>
   <units><num>26</num><unit>0</unit></units><units><num>27</num><unit>
   0</unit></units><units><num>28</num><unit>0</unit></units><units><nu
   m>29</num><unit>0</unit></units><units><num>30</num><unit>0</unit></
   units><units><num>31</num><unit>0</unit></units><units><num>32</num>
   <unit>0</unit></units><units><num>33</num><unit>0</unit></units><uni
   ts><num>34</num><unit>0</unit></units><units><num>35</num><unit>0</u
   nit></units><units><num>36</num><unit>0</unit></units><units><num>37
   </num><unit>0</unit></units><units><num>38</num><unit>0</unit></unit
   s><units><num>39</num><unit>0</unit></units><units><num>40</num><uni
   t>0</unit></units><units><num>41</num><unit>0</unit></units><units><
   num>42</num><unit>0</unit></units><units><num>43</num><unit>0</unit>
   </units><units><num>44</num><unit>0</unit></units><units><num>45</nu
   m><unit>0</unit></units><units><num>46</num><unit>0</unit></units><u
   nits><num>47</num><unit>0</unit></units><units><num>48</num><unit>0<
   /unit></units><units><num>49</num><unit>0</unit></units><units><num>
```

```
50</num><unit>0</unit></units><units><num>51</num><unit>0</unit></un
its><units><num>52</num><unit>0</unit></units><units><num>53</num><u
nit>0</unit></units><units><num>54</num><unit>0</unit></units><units
><num>55</num><unit>0</unit></units><units><num>56</num><unit>0</uni
t></units><units><num>57</num><unit>0</unit></units><units><num>58</
num><unit>0</unit></units><units><num>59</num><unit>0</unit></units>
<units><num>60</num><unit>0</unit></units><units><num>61</num><unit>
0</unit></units><units><num>62</num><unit>0</unit></units><units><nu
m>63</num><unit>0</unit></units><options><surface>0</surface></optio
ns><restype>1</restype></doctable></cmddoctable><cmdendsession/></fd
script>
```

12.5 Main.py

```
import os
import Parser as parser
import XLSX
import plate as plate
import pandas as pd
from pprint import pprint
import numpy as np
import xlsxwriter


alpha = 0.5
lm = 5
lk = alpha * lm
y = [0, lk, lk + lm/2, lk + lm, lm + 2*lk]
x = 30
SecStep_lm = lm/10
SecStep_lk = 2* lk/10
Left_Sec_lk = np.arange(SecStep_lk, lk + SecStep_lk, SecStep_lk)

Sec_lm = np.arange(SecStep_lm + lk, lm + lk + SecStep_lm, SecStep_lm)

Right_Sec_lk = np.arange(SecStep_lk + lm + lk, lm + 2*lk, SecStep_lk)


FEMIn = {         'dia' : 0.2,
                 'alpha' : alpha,
                    'lm' : lm,
                    'lk' : lk,
                     'x' : x,
                     'y' : y,
                 'xload' : x/2,
                 'yload' : 0,
                     't' : [0.3, 0.3, 0.3, 0.3, 0.3],
             'Ext_list' : ['_My', '_M2', '_Vy', '_Mxy', '_coor'],
         'materialName' : "C40/50",  #C40/50 = 35GPa
                                     #C12/15 = 27GPa

                 'xsupp' : x,
                 'ysupp' : [lk ,  lk + lm],
        'loadIntensity' : 6.25,
              'poisson' : 0                    }


ParserIn= {'Direction' : 'x',
                    'y' : [1,2],
                    'x' : [2,1],
           'Parameter' : ['My','Vy'],
                   'Vy' : int(10),
                   'M2' : int(4),
                   'My' : int(4),
                  'Mxy' : int(5),
          'Section_My' : list(Left_Sec_lk) + list(Sec_lm) + list(Right_Sec_lk), # [1.25, 1.3, 2, 2.5, 3, 4, 5, 6, 7.5, 8], #

        'Section_Mxy' : list(Left_Sec_lk) + list(Sec_lm) + list(Right_Sec_lk),  # [1.25, 1.3, 2, 2.5, 3, 4, 5, 6, 7.5, 8], #
```

```
55.
56.         'Section_Vy' : list(Left_Sec_lk) + list(Sec_lm) + list(Right_Sec_lk),  # [ 2, 2.15, 2.3, 2.4, 2.45, 2.5, 2.6, 2.7, 3, 4, 5, 7.5],
57.               'Mesh' : float(0.5),
58.          'plot_sort' : 3,
59.                  'W' : [30, 0, 0.5, 1, 1.5, 2, 2.5, 3],
60.       'aspect_ratio' : float(x/y[len(y)-1])   }
61.
62.
63. print(ParserIn['Section_My'])
64.
65. w_col = len(ParserIn['W'])
66.
67. TableForm= {   'y_head' : 0,
68.             'Para_avg' : 1,        #First Row in Excel sheet
69.            'Hand_calc' : w_col + 4,
70.                 'q1q2' : w_col*2 + 6,
71.            'Eq_header' : w_col*3 + 8,          #Header values for eq width (Hidden)
72.           '-ve_header' : w_col*4 + 8,
73.           '+ve_header' : w_col*5 + 8,
74.          'Main Header' : w_col*6 + 11        #Vertical Table header
75.               }
76.
77. Pathdic= {'fil' : 'Alpha' + str(alpha) + '_Y' + str(y[len(y)-1]) + '_T=' + str(FEMIn['t']),
78.
79.          'dir' : 'Alpha' + str(alpha) + '_Y' + str(y[len(y)-1]) + '_X' + str(x)}
80.
81. XLSX._chkfile(Pathdic, ParserIn)
82. workbook = xlsxwriter.Workbook('xlsx/' + Pathdic['dir'] + '/'+ Pathdic['fil'] + ParserIn.get('Direction') + '.xlsx', {'strings_to_numbers': True})
83. workbook_diag = xlsxwriter.Workbook('./temp/Diagrams/diag_Vy.xlsx', {'strings_to_numbers': True})
84.
85. for yload in list(range(2, 4, 2)) : #list(range(2, 62, 2)) + [62.5] :#
86.     FEMIn['yload'] = yload/10
87.     try:
88.
89.         if __name__ == '__main__':
90.                 plate.Deck(Pathdic, ParserIn, FEMIn, True)
91.                 print('Trying Load at y= ' + str(FEMIn['yload']))
92.                 Para_dict = parser.Analyze(Pathdic, workbook_diag, ParserIn, FEMIn)
93.                 #with open('log.txt', 'w') as out:
94.                 #pprint(Para_dict, stream = out)
95.                 #pprint(Para_dict)
96.                 XLSX.xlsExcel(Pathdic, workbook, Para_dict, ParserIn, FEMIn, TableForm)
97.
98.     except:
99.         print('No result can be extracted for load at y= ' + str(FEMIn['yload']))
100.            pass
101.
102.    workbook_diag.close()
```

```
    workbook.close()
```

12.6 Plate.py

```
1. import fdAPI_wrapper_mod as fd
2. import os
3.
4.
5. def Deck(Pathdic, ParserIn, FEMIn, close):
6.   # init model
7.   fd.initiateModel("S")
8.   direction = fd.coord(0,0,-1)
9.
10.
11.   # add material
12.   material = fd.addMaterial(fd.material(FEMIn['materialName'], "0", "0",
   FEMIn['poisson']))
13.
14.   # add structural parts
15.   x = FEMIn['x']
16.   y = FEMIn['y']
17.   for i in range(0, len(y)-1):
18.       p0 = fd.coord(0,y[i],0)
19.       p1 = fd.coord(x,y[i+1],0)
20.       fd.addPlate(material, FEMIn['t'][i], FEMIn['t'][i+1], " ", p0, p1,
   "top", str(ParserIn['Mesh']))
21.
22.
23.   # add load cases
24.   dl = fd.addLoadCase("DL", False)
25.   ll = fd.addLoadCase("LL", False)
26.
27.
28.   # create load
29.   p0 = fd.coord(FEMIn['xload'] - FEMIn['dia'], FEMIn['yload'] -
   FEMIn['dia'] ,0)
30.   p1 = fd.coord(FEMIn['xload'] + FEMIn['dia'], FEMIn['yload'] +
   FEMIn['dia'] ,0)
31.   fd.addSurfaceLoad(FEMIn['loadIntensity'], direction,ll, p0, p1 )
32.
33.
34.   # add load combinations
35.   fd.addLoadComb("LC1", "U", [dl, ll], [1, 1])
36.
37.   # create points
38.   p0 = fd.coord(0,FEMIn['ysupp'][0],0)
39.   p1 = fd.coord(FEMIn['xsupp'],FEMIn['ysupp'][0],0)
40.   p2 = fd.coord(0,FEMIn['ysupp'][1],0)
41.   p3 = fd.coord(FEMIn['xsupp'],FEMIn['ysupp'][1],0)
42.
43.
44.   # create supports
45.   fd.addLineSupport(p0, p1, "hinged")
46.   fd.addLineSupport(p2, p3, "zpinned")
47.
48.
49.   #Verifying and creating working directory
50.   if not os.path.exists('temp/'+ Pathdic['dir']):
51.     os.mkdir('temp/'+ Pathdic['dir'])
52.
```

```
53.
54.   # create struxml
55.   filePath = 'temp/'+ Pathdic['dir'] +'/'+ Pathdic['fil'] + '_P_pos'+
   str(FEMIn['yload']) + '_model.struxml'
56.   fd.finish(filePath)
57.
58.
59.   #create bsc
60.   batchfile = ['BSC/' + str(x) + '.bsc' for x in FEMIn['Ext_list']]
61.   exportfile = ['temp/'+ Pathdic['dir'] + '/' + Pathdic['fil'] +
   '_P_pos'+ str(FEMIn['yload']) + str(x) + '.txt' for x in
   FEMIn['Ext_list']]
62.
63.   #fd.runFD('LIN', False, close, 'no', filePath, batchfile, exportfile)
64.
65.   #open fd
66.   fd.openFD(filePath)
67.
68.   return
```

12.7 Parser.py

```
import pandas as pd
import os
import x_section_w
import y_intergration_w
import handcal
import numpy as np


def _Sort(df, by):
    df.sort_values(by=[by], inplace = True)
    df.reset_index(drop=True, inplace=True)
    return df

def _Coordata(coor_df, ParserIn, Sec):
    #Reading the FEM design 19 coordinates batchfile intothe node-coordinates dataframe (coor_df)
    Index= ParserIn.get(ParserIn.get('Direction'))
    meshsize = ParserIn.get('Mesh')

    node_df = coor_df.loc[abs(coor_df[Index[0]]-Sec) <= (meshsize)] #change the column number 1:X 2:Y      (meshsize/2)0.5= torellence
    #print(node_df)
    return (node_df)


def _Secdata(node_df, load_df, Index1, Index2):
    extract_load_df = pd.DataFrame()   #where the nodes corrdinates are stored
    filtered_nodes_df = pd.DataFrame()
    #Extracting the related nodes data
    node_df_list = [x for x in node_df[0].tolist() if x != 'nan']
    extract_load_df = extract_load_df.append(load_df[load_df[2].isin(node_df_list)] , ignore_index = True)

    #Filtering the node-coordinates dataframe according to the extracted node-load dataframe
    filtered_nodes_df = filtered_nodes_df.append(node_df[node_df[0].isin(extract_load_df[2].tolist())], ignore_index = True)
    filtered_nodes_df = _Sort(filtered_nodes_df, 0)
    extract_load_df = _Sort(extract_load_df, 2)

    result_df = pd.concat([extract_load_df[2], extract_load_df[Index2],filtered_nodes_df[0],
                            filtered_nodes_df[Index1[1]], filtered_nodes_df[Index1[0]]], axis=1)
    result_df.columns = range(result_df.shape[1])
    result_df = _Sort(result_df, 3)
    #print(result_df)
    return(result_df)

def q1q2(worksheet, FEMIn, ParserIn, coor_df, load_df, Index, Para):
    Q={}
    col=0
```

```
    for w in ParserIn.get('W'):
        q = {}
        xlist = []
        plist = []
        for Sec in list(np.arange(ParserIn['Mesh'],2*FEMIn['lk']+FEMIn['lm']-ParserIn['Mesh'], ParserIn['Mesh'])):
            para_df1 = _Secdata(_Coordata(coor_df, ParserIn, Sec), load_df, Index, ParserIn.get(Para))
            (No, X, Y, P) = x_section_w.section_interp(Sec, para_df1, w)
            xlist.append(list(Y)[0])
            plist.append(y_intergration_w.average (0, Para, Sec, X, P, w, False))
        if Para == 'My':
            worksheet.write(0, col, "w=" + str(w))
            worksheet.write_column(1, col, xlist)
            worksheet.write_column(1, col+1, plist)
            col+=2
        #result= pd.DataFrame(list(zip(xlist, plist)))
        #result.to_csv('./temp/Diagrams/' + str(FEMIn['yload']) + '_' + str(w) + '.csv', index = False)

        (q['q' + str(FEMIn.get('ysupp')[0])], q['q' + str(FEMIn.get('ysupp')[1])]) = handcal.q_moment(FEMIn, xlist, plist, w)
        Q[w] = q

    return Q
    # Q={}
    # for Sec in FEMIn['ysupp']:
    #     para_df1 = _Secdata(_Coordata(coor_df, ParserIn, Sec), load_df, Index, ParserIn.get(Para))
    #     (No, X, Y, P) = x_section_w.section_interp(Sec, para_df1, W)
    #     Q['q' + str(Sec)] =  y_intergration_w.average (FEMIn.get('lk'), Para, Sec, X, P, W, True)



def Analyze(Pathdic, workbook_diag, ParserIn, FEMIn):
    #initilizing the dataframes
    Index = ParserIn.get(ParserIn.get('Direction'))
    w = ParserIn.get('W')
    #nod_dic =_Coordata(Path, ParserIn)
    Para_dict={}
    coor_df=pd.read_csv('./temp/'+ Pathdic['dir'] +'/'+ Pathdic['fil'] + '_P_pos'+ str(FEMIn['yload']) + '_coor.txt',sep='      ', header=None, engine='python')
    worksheet = workbook_diag.add_worksheet(str(FEMIn['yload']))
    for Para in ParserIn.get('Parameter'):
        load_df=pd.read_csv('./temp/'+ Pathdic['dir'] +'/'+ Pathdic['fil'] + '_P_pos'+ str(FEMIn['yload']) + '_' + Para + '.txt',sep=' ', header=None, engine='python')
        data_dict={}
        q = q1q2(worksheet, FEMIn, ParserIn, coor_df, load_df, Index, ParserIn.get('Parameter')[0])   #
        for Sec in (ParserIn.get('Section_'+ Para)):
            para_df1 = _Secdata(_Coordata(coor_df, ParserIn, Sec), load_df, Index, ParserIn.get(Para))
            w_dict= {}
            for W in w:
```

```
                sec_dict={}
                q_w = q.get(W)  #{'q2.5' : 0 , 'q7.5' : 0}  #
                #para_df1.to_csv('./temp/File Name.csv', index = False)
                (No, X, Y, P) = x_section_w.section_interp(Sec, para_df1, W)
                #result= pd.DataFrame(list(zip(X, P)))
                #result.to_csv('./temp/File Name.csv', index = False)
                #if Para == 'Vy': sec_dict[Para] = list(abs(np.array(P)))
                sec_dict[Para] = list(P)     #else :
                sec_dict['X'] = list(X)
                     sec_dict['Y'] = list(Y)
                     sec_dict[Para + '_avg'] = y_intergration_w.average(0, Para, Sec, X, P, W, False)
                     sec_dict[Para + '_h'] = handcal.hand_calculation(FEMIn, q_w, Para, Sec)
                     if Sec in FEMIn['ysupp']:
                         sec_dict['q' + str(Sec)] = q_w.get('q' + str(Sec))
                     w_dict[W] = sec_dict
                 data_dict[Sec] = w_dict
            Para_dict[Para] = data_dict
        return(Para_dict)
```

12.8x_section_w.py

```
import numpy as np

def section_interp (x_need, df, w):

    # part 1 ==================================================================
    # define arrays
    m =  df[1].tolist()       #Moment/ Shear values
    y =  df[3].tolist()       #Y coordinates
    node = df[0].tolist()      #Nodes
    x =  df[4].tolist()       #X coordinates

    node_a = np.array(node)
    x_a = np.array(x)
    y_a = np.array(y)
    m_a = np.array(m)

    # find left and right section
    x_left = min(x_a)
    x_right = max(x_a)

    # indexing that separate the array in left and right
    ind_lfet = x_a == x_left
    ind_right = x_a == x_right

    # separate the arrays in lfet and right
    node_a_left = node_a[ind_lfet]
    node_a_right = node_a[ind_right]
    x_a_left = x_a[ind_lfet]
    x_a_right = x_a[ind_right]
    x_a_need = np.full(np.shape(x_a_left ),x_need)
    y_a_left = y_a[ind_lfet]
    y_a_right = y_a[ind_right]

    # force at left and right section
    m_a_left = m_a[ind_lfet]
    m_a_right = m_a[ind_right]

    if x_right == x_left and x_left == x_need:
        # print("existing section")
        m_a_interp = m_a_left
        node_a_int = node_a_left

    elif x_right > x_left and x_left <= x_need <= x_right:
        # print("new interpolated section")
        # m_a_left and m_a_right as matrix
        m_m =np.transpose(np.vstack((m_a_left, m_a_right)))

        # mark the nodes with "+0.5" if selected x is not exist
        if x_need == x_left :
            # print("existing section")
            node_a_int = node_a_left

        elif  x_need == x_right :
            # print("existing section")
            node_a_int = node_a_right
```

```
56.
57.         else :
58.             # print("new interpolated section")
59.             node_a_int = np.array([node1 + 0.5 for node1 in
   node_a_left])
60.
61.         # interpolation
62.         # predefine the interpolation vector
63.         m_interp = np.zeros((np.size(m_a_left),1))
64.
65.         # interpolating as a for loop
66.         for i in range (0, np.size(m_a_left)):
67.             m_interp[i] = np.interp(x_need, [x_left, x_right], m_m[i])
68.
69.         m_a_interp = m_interp.reshape(np.size(m_interp))
70.
71.
72.     else :
73.         print ("input error")
74.
75.     # part 2 in y direction
   ========================================================
76.     # slab length (biggest distribution width can be chosen)
77.     w_lim = max(y) - min(y)
78.
79.     if w_lim >= w > 0:
80.         # define the points
81.         y_mid = (max(y)-min(y))/2 + min(y)
82.         y_up = y_mid + w/2
83.         y_down = y_mid - w/2
84.
85.         # find out if the end points exist at the chosen path
86.         case = [y1 == y_up or y1 == y_down for y1 in y_a_left]
87.         ca = sum (np.ones(np.shape(case))[case])
88.
89.
90.         # logic structure, return as a narrowed y array
91.         if ca == 2:
92.             # print ("existing nodes")
93.             ind_na = [y2 <= y_up and y2 >= y_down for y2 in y_a_left]
94.             y_a_na = y_a_left [ind_na]
95.
96.             node_a_na = node_a_int[ind_na]
97.             x_a_na = x_a_need [ind_na]
98.             y_a_na = y_a_left [ind_na]
99.             m_a_na = m_a_interp[ind_na]
100.
101.
102.             elif ca == 0:
103.                 # print ("created new nodes")
104.                 # indexing for nearest nodes
105.                 ind_left_upup =  y_a_left == min(y_a_left[np.array([y3-
   y_up  for y3 in y_a_left]) > 0])
106.                 ind_left_updown = y_a_left ==
   max(y_a_left[np.array([y3-y_up  for y3 in y_a_left]) < 0])
107.                 ind_left_downup = y_a_left ==
   min(y_a_left[np.array([y3-y_down  for y3 in y_a_left]) > 0])
108.                 ind_left_downdown = y_a_left ==
   max(y_a_left[np.array([y3-y_down  for y3 in y_a_left]) < 0])
```

```
109.
110.                # indexing array y (nodes between upper and lower
   nodes)
111.                ind_bt = [y2 < y_up and y2 > y_down for y2 in y_a_left]
112.
113.                # interpolation for upper and lower nodes
114.                m_inter_up = np.interp(y_up,
   np.hstack((y_a_left[ind_left_updown], y_a_left[ind_left_upup])),
   np.hstack((m_a_interp[ind_left_updown],m_a_interp[ind_left_upup])))
115.                m_inter_down = np.interp(y_down,
   np.hstack((y_a_left[ind_left_downdown], y_a_left[ind_left_downup])),
   np.hstack((m_a_interp[ind_left_downdown],m_a_interp[ind_left_downup])))
116.
117.                # output
118.                node_a_na =
   np.hstack((np.array(node_a_int[ind_left_downdown]+0.25),node_a_int[ind_b
   t],np.array(node_a_int[ind_left_upup]+0.25)))
119.                x_a_na = np.hstack((x_a_need[ind_left_downdown],
   x_a_need[ind_bt], x_a_need[ind_left_upup]))
120.                y_a_na = np.hstack(([y_down], y_a_left[ind_bt],
   [y_up]))
121.                m_a_na = np.hstack(([m_inter_down], m_a_interp[ind_bt],
   [m_inter_up]))
122.
123.            else:
124.                print ("missing point in y-direction")
125.
126.        elif w == 0 :
127.            # check the numbers of nodes is odd or even.
128.            # case even
129.            if np.size (y_a_left) % 2 == 0 :
130.
131.                ind_na = int(np.size(y_a_left)/2-1)
132.
133.                # out put
134.                node_a_na = node_a_int [ind_na]
135.                x_a_na = [x_a_need [ind_na].tolist()]
136.                y_a_na = [y_a_left [ind_na].tolist()]
137.                m_a_na = [m_a_interp[ind_na].tolist()]
138.
139.            # case odd
140.            elif np.size (y_a_left) % 2 == 1 :
141.
142.                ind_na = int((np.size(y_a_left)+1)/2-1)
143.
144.                print (ind_na)
145.
146.                # out put
147.                node_a_na = np.array( node_a_int[ind_na] + 0.25 )
148.                x_a_na = [x_a_need [ind_na].tolist()]
149.                y_a_na = [y_a_left [ind_na].tolist()]
150.                m_a_na = [m_a_interp[ind_na].tolist()]
151.
152.        else:
153.            #print ("chosen width out of range, using full slab
   length")
154.            # redefine w
155.            w = w_lim
156.            # define the points
```

```
            y_mid = (max(y)-min(y))/2 + min(y)
            y_up = y_mid + w/2
            y_down = y_mid - w/2

            # find out if the end points exist at the chosen path
            case = [y1 == y_up or y1 == y_down for y1 in y_a_left]
            ca = sum (np.ones(np.shape(case))[case])

            # logic return as a narrowed y array
            if ca == 2:
                # print ("existing nodes")
                # print (y_up, y_down)
                ind_na = [y2 <= y_up and y2 >= y_down for y2 in y_a_left]
                y_a_na = y_a_left [ind_na]

                # out put
                node_a_na = node_a_int [ind_na]
                x_a_na = x_a_need [ind_na]
                y_a_na = y_a_left [ind_na]
                m_a_na = m_a_interp[ind_na]

            else:
                print ("missing point at y-direction")


        # final output
        node = node_a_na
        y = x_a_na
        x = y_a_na
        m = m_a_na

        return (node, x, y, m)


    # define the variables

    # node = [1, 6, 5, 10, 2, 3, 7, 8, 4, 9]
    # x = [10, 20, 10, 20, 10, 10, 20, 20, 10, 20]
    # y = [0, 0, 40, 40, 10, 20, 10, 20, 30, 30]
    # m = [-100, 60, 50, 150, 0, 200, 120, 300, 100, 400]
    # x_need = 15

    # w = 45


    # section_interp (x_need, node, x, y, m, w)
```

12.9 y_intergration_w.py

```
# assumption: all the arrays have to be sorted in such sequence that y from smallest to biggest!
# return q1 and q2 if selected section at support.
# q1 and q2 base on averaging width "w"

def average (Lcan, Para, x_sec, y, m, w, arg):
    import numpy as np
    # look up the variables
    q = 0

    # define arrays
    y_a = np.array(y)
    m_a = np.array(m)

    if w !=0 :
        # extract and modify the needed arrays
        m_a_mod = m_a[0:-1] + m_a[1:] # top + bottom
        y_a_mod = (y_a[1:] - y_a[0:-1])/2 # height / 2

        # intergration
        m_tot = sum((m_a_mod * y_a_mod))

        # average force by given width

        m_avg = m_tot / w

    elif w == 0 :
        m_tot = float (m_a)
        m_avg = m_tot

    # return result
    if arg == True and Para == 'My':
        q = (-m_tot*2) / (Lcan**2)
        return (q)
    elif arg == True and Para == 'Vy':
        q = (-m_tot) / (Lcan)
        return (q)
    else:
        return (m_avg)



# testing in put
# node = [1.75,     2.5, 3.5,  4.5, 5.75]
# x =    2.5
# y =    [2.5,      10,  20,   30,  30.5]
# m =    [-187.5,   60,  250,  250, 137.5]

# Sec = 7.5

# lk = 2.5
# lm = 5
# alpha = 0.5
# w = 20

```

```
# FEMIn = {         'dia' : 0.2,
#                 'alpha' : alpha,
#                    'lm' : lm,
#                    'lk' : lk,
#                     'x' : x,
#                     'y' : y,
#                 'xload' : x/2,
#                 'yload' : 5,
#                     't' : [0.3, 0.3, 0.3, 0.3, 0.3],
#              'Ext_list' : ['_My', '_M2', '_Vy', '_Mxy', '_coor'],
#          'materialName' : "C40/50",
#                 'xsupp' : x,
#                 'ysupp' : [lk ,  lk + lm],
#         'loadIntensity' : 10,
#               'poisson' : 0}


# (m_avg, q1, q2) = average (FEMIn, Sec, y, m, w)
# print(m_avg, q1, q2)
```

12.10 handcal.py

```
import numpy as np

# moment influence line
def inf_M (x_sec, x_p, alfa, Lsp):
    # span ratio alfa = Lcan/Lsp
    Lcan = alfa*Lsp
    Ltot = Lcan*2+Lsp

    if x_p < Lcan :
        if x_p < x_sec <= Lcan :
            return -(x_sec-x_p)
        elif Lcan < x_sec <= (Lcan + Lsp):
            return -(Lcan-x_p)*(1-(x_sec-Lcan)/Lsp)
        else:
            return 0
    elif Lcan < x_p < Lcan + Lsp and Lcan <= x_sec <= Lcan + Lsp :
        if x_p < x_sec:
            return ((x_p - Lcan)*(Lcan + Lsp - x_p) / Lsp)*(1 - (x_sec - x_p) / (Lcan + Lsp - x_p))
        else:
            return ((x_p - Lcan)*(Lcan + Lsp - x_p) / Lsp)*((x_sec - Lcan) / (x_p - Lcan))
    elif x_p == Lcan or x_p == Lcan + Lsp :
        return 0
    elif  Lcan + Lsp < x_p :
        if Lcan + Lsp < x_sec <= x_p :
            return -(x_p - x_sec)
        elif Lcan < x_sec <= Lcan + Lsp :
            return -(x_p - Lcan - Lsp)*((x_sec-Lcan) / Lsp)
        else:
            return 0
    else:
        return 0
    return




# Moment for distributed load:
def M_q (alfa, Lsp, x_sec, x_p, q, a, q1, q2):
    # span ratio alfa = Lcan/Lsp
    Lcan = alfa*Lsp
    Ltot = Lcan*2+Lsp

    # part1 for acting load

    # a: distribution length of load
    # n_q = 100: number of finite grains of n_q to be divide
    ai = np.arange (-a/2, a/2 + a/100, a/100)
    xi = ai + np.full_like(ai, x_p)
    qi = np.full_like(ai, q*a/(100+1))
    inf_a_M = np.array ([inf_M (x_sec, x, alfa, Lsp) for x in xi])

    # part2 for dummy load:

    # q1:
```

```
    # a=Lsp: distribution length of load
    # n_q = 100: number of finite grains of n_q to be divide
    ai_1 = np.arange (-Lcan/2, Lcan/2 + Lcan/100, Lcan/100)
    xi_1 = ai_1 + np.full_like(ai_1, Lcan/2)
    qi_1 = np.full_like(ai_1, q1*Lcan/(100+1))
    inf_a_M_1 = np.array ([inf_M (x_sec, x, alfa, Lsp) for x in xi_1])

    # q2:
    # a=Lcan: distribution length of load
    # n_q = 100: number of finite grains of n_q to be divide
    ai_2 = np.arange (-Lcan/2, Lcan/2 + Lcan/100, Lcan/100)
    xi_2 = ai_2 + np.full_like(ai_2, (Lcan/2 + Lsp + Lcan))
    qi_2 = np.full_like(ai_2, q2*Lcan/(100+1))
    inf_a_M_2 = np.array ([inf_M (x_sec, x, alfa, Lsp) for x in xi_2])

    return (float (sum(qi*inf_a_M)) + float (sum(qi_1*inf_a_M_1)) +
float (sum(qi_2*inf_a_M_2)))


# shear force influence line
def inf_V (x_sec, x_p, alfa, Lsp):
    # span ratio alfa = Lcan/Lsp
    Lcan = alfa*Lsp
    Ltot = Lcan*2+Lsp

    if 0 <= x_p < Lcan :
        if x_p >= x_sec:
            return 0
        if x_p < x_sec < Lcan :
            return -1
        if x_sec == Lcan :
            return 0
        if Lcan < x_sec < Lcan + Lsp :
            return (Lcan - x_p) / Lsp
        if Lcan + Lsp <= x_sec <= Ltot :
            return 0

    elif Lcan <= x_p <= Lcan + Lsp :
        if 0 <= x_sec <= Lcan :
            return 0
        if Lcan < x_sec < x_p :
            return 1 - ((x_p - Lcan) / Lsp)
        if x_sec == x_p :
            return 0
        if x_p < x_sec < Lcan + Lsp :
            return - ((x_p - Lcan) / Lsp)
            if Lcan + Lsp <= x_sec <= Ltot :
                return 0

        elif Lcan + Lsp < x_p <= Ltot :
            if 0 <= x_sec <= Lcan :
                return 0
            if Lcan < x_sec < Lcan + Lsp :
                return 1 - ((x_p - Lcan) / Lsp)
            if x_sec == Lcan + Lsp :
                return 0
            if Lcan + Lsp < x_sec < x_p :
                return 1
            if x_p <= x_sec <= Ltot :
```

```
                return 0

        print('Out of the influence line at value' + str(x_p))
        return 0



    # shear force for distributed load:
    def V_q (alfa, Lsp, x_sec, x_p, q, a, q1, q2):

        # span ratio alfa = Lcan/Lsp
        Lcan = alfa*Lsp

        # part1 acting load
        # a: distribution length of load
        # n_q = 100: number of finite grains of n_q to be divide

        ai = np.arange (-a/2, a/2+a/100, a/100)
        xi = ai + np.full_like(ai, x_p)
        qi = np.full_like(ai, q*a/(100+1))
        inf_a_V = np.array ([inf_V (x_sec, x, alfa, Lsp) for x in xi])

        # part2 dummy load
        # q1
        # a=Lcan: distribution length of load
        # n_q = 100: number of finite grains of n_q to be divide

        ai_1 = np.arange (-Lcan/2, Lcan/2 + Lcan/100, Lcan/100)
        xi_1 = ai_1 + np.full_like(ai_1, Lcan/2)
        qi_1 = np.full_like(ai_1, q1*Lcan/(100+1))
        inf_a_V_1 = np.array ([inf_V (x_sec, x, alfa, Lsp) for x in xi_1])

        # q2
        # a=Lcan: distribution length of load
        # n_q = 100: number of finite grains of n_q to be divide

        ai_2 = np.arange (-Lcan/2, Lcan/2 + Lcan/100, Lcan/100)
        xi_2 = ai_2 + np.full_like(ai_2, (Lcan/2 + Lsp + Lcan))
        qi_2 = np.full_like(ai_2, q2*Lcan/(100+1))
        inf_a_V_2 = np.array ([inf_V (x_sec, x, alfa, Lsp) for x in xi_2])



        # need an "-" before result for main code to match FEM design
        return ((float (-sum(qi*inf_a_V)) + float (-sum(qi_1*inf_a_V_1)) + float (-sum(qi_2*inf_a_V_2))))



    def hand_calculation (FEMIn, q1q2, Para, Sec):
        alfa = FEMIn.get('alpha')
        Lsp = FEMIn.get('lm')
        x_sec = Sec
        x_p = FEMIn.get('yload')
        q = FEMIn.get('loadIntensity') * FEMIn.get('dia') * 2
        a = FEMIn.get('dia')*2
        q1 = q1q2.get('q' + str(FEMIn.get('ysupp')[0]))
```

```
169.         q2 = q1q2.get('q' + str(FEMIn.get('ysupp')[1]))
170.
171.
172.         if Para == 'My': return M_q (alfa, Lsp, x_sec, x_p, q, a, q1,
    q2)
173.         elif  Para == 'Vy': return V_q (alfa, Lsp, x_sec, x_p, q, a,
    q1, q2)
174.         else :
175.             return print ('FEM output error')
176.         return
177.
178.     def _slice(y0, y1, y2, y3, z0, z1, z2, z3 ):
179.         x0 = y0 [z0]
180.         x1 = y1 [z1]
181.         x2 = y2 [z2]
182.         x3 = y3 [z3]
183.         return (x0, x1, x2, x3)
184.
185.     def _find(m_a):
186.         up = [m1 < 0 for m1 in m_a]
187.         down = [m2 > 0 for m2 in m_a]
188.         return(up, down)
189.
190.     def _ind(up, down):
191.         ind_11 = [m1 == max(up) for m1 in up]
192.         ind_12 = [m2 == min(down) for m2 in down]
193.         return(ind_11, ind_12)
194.
195.
196.     def q_moment (FEMIn, x, m, w) :
197.         q1 = 0
198.         q2 = 0
199.         alfa = FEMIn.get('alpha')
200.         Lsp = FEMIn.get('lm')
201.         x_p = FEMIn.get('yload')
202.         q = FEMIn.get('loadIntensity') * FEMIn.get('dia') * 2
203.         a = FEMIn.get('dia')*2
204.         Lcan = alfa*Lsp
205.         Ltot = Lcan * 2 + Lsp
206.
207.         # span ratio alfa = Lcan/Lsp
208.         if w < FEMIn.get('x'):
209.             # define the arrays
210.             # node_a = np.array(node)
211.             x_a = np.array(x)
212.             m_a = np.array(m)
213.
214.             if Lcan < x_p < Lcan + Lsp :
215.                 # slice the arrays to left and right
216.                 ind_left = [Lcan <= x1 <= x_p for x1 in x_a]
217.                 ind_right = [x_p <= x2 <= (Lcan + Lsp) for x2 in x_a]
218.                 (x_a_left, x_a_right, m_a_left, m_a_right) =
    _slice(x_a, x_a, m_a, m_a, ind_left, ind_right, ind_left, ind_right)
219.
220.                 # slice the arrays to up and down
221.                 (ind_leftup, ind_leftdown) = _find(m_a_left)
222.                 (ind_rightup, ind_rightdown) = _find(m_a_right)
223.
```

```
224.                (x_a_leftup, x_a_leftdown, x_a_rightup, x_a_rightdown)
   = _slice(x_a_left, x_a_left, x_a_right,
225.
   x_a_right, ind_leftup, ind_leftdown, ind_rightup, ind_rightdown)
226.                [m_a_leftup, m_a_leftdown, m_a_rightup, m_a_rightdown]
   = _slice(m_a_left, m_a_left, m_a_right,
227.
   m_a_right, ind_leftup, ind_leftdown, ind_rightup, ind_rightdown)
228.                # locate 4 points
229.                (ind_11, ind_12) = _ind(m_a_leftup, m_a_leftdown)
230.                (ind_21, ind_22) = _ind(m_a_rightup, m_a_rightdown)
231.
232.                (x_11, x_12, x_21, x_22) = _slice(x_a_leftup,
   x_a_leftdown, x_a_rightup,
233.
   x_a_rightdown, ind_11, ind_12, ind_21, ind_22)
234.                (m_11, m_12, m_21, m_22) = _slice(m_a_leftup,
   m_a_leftdown, m_a_rightup,
235.
   m_a_rightdown, ind_11, ind_12, ind_21, ind_22)
236.
237.                # interpolation to find the 0 moment position
238.                x_1 =float (x_11 + (abs (m_11) / (abs (m_11) + abs
   (m_12))) * (x_12 - x_11))
239.                x_2 =float (x_22 + (abs (m_22) / (abs (m_21) + abs
   (m_22))) * (x_21 - x_22))
240.
241.                # find the moment for x_1 and x_2 at hand calculation
   model
242.                m_1 = - M_q (alfa, Lsp, x_1, x_p, q, a, 0, 0)
243.                m_2 = - M_q (alfa, Lsp, x_2, x_p, q, a, 0, 0)
244.
245.                incl = (m_2 - m_1) / (x_2 - x_1)
246.                # use inclination to extand the line to suport
247.                # moment at suport 1 and 2
248.                m_s1 = m_1 - (x_1 - Lcan)*incl
249.                m_s2 = m_2 + (Lcan + Lsp - x_2)*incl
250.
251.                # calculate q1 and q2
252.                q1 = -(m_s1*2) / (Lcan**2)
253.                q2 = -(m_s2*2) / (Lcan**2)
254.
255.            elif x_p < Lcan :
256.                # slice the arrays at span
257.                x_a_span = x_a [[Lcan <= x_a1 <= Lcan + Lsp for x_a1 in
   x_a]]
258.                m_a_span = m_a [[Lcan <= x_a1 <= Lcan + Lsp for x_a1 in
   x_a]]
259.
260.                # slice the arrays to up and down
261.                (ind_spanup, ind_spandown) = _find(m_a_span)
262.
263.                # locate 2 points
264.                (ind_1, ind_2) = _ind(m_a_span[ind_spanup],
   m_a_span[ind_spandown])
265.                (x_1, x_2, m_1, m_2) = _slice(x_a_span[ind_spanup],
   x_a_span[ind_spandown], m_a_span[ind_spanup], m_a_span[ind_spandown],
266.                                          ind_1, ind_2, ind_1, ind_2)
267.
```

```
                # interpolation to find the 0 moment position
                x_0 = float (x_1 + (abs(m_1) /(abs(m_1) + abs(m_2))) * (x_2 - x_1) )

                # find the moment for x_0 at hand calculation model
                m_0 = - M_q (alfa, Lsp, x_0, x_p, q, a, 0, 0)
                incl = m_0 / (x_0 - Lcan)

                # use inclination to extand the line to suport
                m_s2 = m_0 + (Lcan + Lsp - x_0)*incl

                # calculate q1 and q2
                q2 = -(m_s2*2) / (Lcan**2)

            elif x_p > Lcan + Lsp :
                # slice the arrays at span
                x_a_span = x_a [[Lcan <= x_a1 <= Lcan + Lsp for x_a1 in x_a]]
                m_a_span = m_a [[Lcan <= x_a1 <= Lcan + Lsp for x_a1 in x_a]]

                # slice the arrays to up and down
                (ind_spanup, ind_spandown) = _find(m_a_span)

                # locate 2 points
                (ind_1, ind_2) = _ind(m_a_span[ind_spanup], m_a_span[ind_spandown])
                (x_1, x_2, m_1, m_2) = _slice(x_a_span[ind_spanup], x_a_span[ind_spandown], m_a_span[ind_spanup], m_a_span[ind_spandown],
                                              ind_1, ind_2, ind_1, ind_2)

                # interpolation to find the 0 moment position
                x_0 = float (x_2 + (abs (m_2) / (abs (m_1) + abs (m_2))) * (x_1 - x_2))

                # find the moment for x_0 at hand calculation model
                m_0 = - M_q (alfa, Lsp, x_0, x_p, q, a, 0, 0)
                incl = m_0 / (Lcan + Lsp - x_0)

                # use inclination to extand the line to suport
                m_s1 = m_0 + (x_0 - Lcan)*incl

                # calculate q1 and q2
                q1 = -(m_s1*2) / (Lcan**2)


        # print (x_0, q1, q2)
        return (q1, q2)
```

12.11 exp_env.py

```
import numpy as np
import numpy.matlib
import pandas as pd
import math as ma
import handcal as hc


# expression q for hand calculation model
def q1_hand (con1_q, con2_q, q, a, Lcan, Lsp, x_p, d_sup, d_mid):
    q1_h = 0


    if Lcan + Lsp >= x_p >= Lcan:
        q1_h = con1_q * (2*q*a*(x_p-Lcan)*(Lsp+Lcan-x_p)*(Lsp+Lcan-x_p+con2_q)) / (Lcan**2 * Lsp**2)


    return (q1_h)

def q2_hand (con1_q, con2_q, q, a, Lcan, Lsp, x_p, d_sup, d_mid):
    q2_h = 0

    if Lcan + Lsp >= x_p >= Lcan:
        q2_h = con1_q * (2*q*a*(x_p-Lcan)*(x_p-Lcan+con2_q)*(Lsp+Lcan-x_p)) / (Lcan**2 * Lsp**2)

    return (q2_h)



# expression equivalent width moment
def w_M (alfa, Lcan, Lsp, d_edge, d_mid, d_sup, x_p, x_sec):
    w_m = 0.4
    #load at cantilever
    if x_p <= Lcan:
        if Lcan >= x_sec:
            w_m = Lsp*alfa + 2*(Lcan-x_p)*(d_edge/d_sup)

        elif Lcan < x_sec <= Lcan+Lsp:
            w_m = Lsp*alfa + 2*(Lcan-x_p)*(d_edge/d_sup) + 2*(x_sec-Lcan)*ma.tan(ma.radians(30))

    #load at span
    elif Lcan < x_p <= Lcan+Lsp/2:
        if x_sec < Lcan:
            w_m = 2 + 0.1*Lsp*(x_p-Lcan) + 2*(x_p-Lcan)*ma.tan(ma.radians(30))

        elif x_sec > Lcan + Lsp:
            w_m = 2 + 0.1*Lsp*(x_p-Lcan) + 2*(Lcan + Lsp - x_p)*ma.tan(ma.radians(30))


        elif  Lcan <= x_sec <= Lcan + Lsp:
            w_m = 2 + 0.1*Lsp*(x_p-Lcan) + 2*abs(x_sec - x_p)*ma.tan(ma.radians(30))
```

```
51.
52.     return (w_m)
53.
54.
55. # expression equivalent width shear
56. def w_V (alfa, Lcan, Lsp, d_edge, d_mid, d_sup, x_p, x_sec):
57.     w_v = 0.4
58.     #load at cantilever
59.     if x_p <= Lcan:
60.         if Lcan + Lsp >= x_sec >= x_p:
61.             w_v = (2/3)*(Lcan-x_p)*(d_edge/d_sup) + 2*(x_sec-
    x_p)*ma.tan(ma.radians(30))
62.
63.
64.     #load at span
65.     elif Lcan < x_p <= Lcan+Lsp/2:
66.         if x_sec < Lcan:
67.             w_v = (x_p - Lcan)*0.5 + 2 * (x_p -
    Lcan)*ma.tan(ma.radians(30))
68.
69.         elif  x_sec > Lcan + Lsp:
70.             w_v = (x_p - Lcan)*0.5 + 2 * abs(Lcan + Lsp -
    x_p)*ma.tan(ma.radians(30))
71.
72.         elif  Lcan <= x_sec <= Lcan + Lsp:
73.             w_v = (x_p - Lcan)*0.5 + 2 * abs(x_sec -
    x_p)*ma.tan(ma.radians(30))
74.
75.     if w_v <0.4:
76.         w_v=0.4
77.
78.     return (w_v)
79.
80.
81.
82. def envelope (q, a, x_sec, x_p, alfa, Lsp, d_edge, d_sup, d_mid):
83.
84.     Lcan = alfa * Lsp
85.
86.     # define the arrays
87.     x_p_a = np.array(x_p)
88.     x_sec_a = np.array(x_sec)
89.
90.     q1_h_a = np.zeros_like (x_p_a)
91.     q2_h_a = np.zeros_like (x_p_a)
92.
93.     m_max_a = np.zeros_like (x_sec_a)
94.     m_min_a = np.zeros_like (x_sec_a)
95.
96.     v_max_a = np.zeros_like (x_sec_a)
97.     v_min_a = np.zeros_like (x_sec_a)
98.
99.     row_a = np.arange (0,np.size (x_p_a),1)
100.         column_a = np.arange (0,np.size (x_sec_a),1)
101.
102.         # define the matrixes
103.         curves_m =  np.matlib.zeros ((np.size(x_p_a),np.size(x_sec_a)))
104.
105.         curves_v =  np.matlib.zeros ((np.size(x_p_a),np.size(x_sec_a)))
```

```
106.
107.        w_m = np.matlib.zeros ((np.size(x_p_a),np.size(x_sec_a)))
108.        w_v = np.matlib.zeros ((np.size(x_p_a),np.size(x_sec_a)))
109.
110.        # overwriting curves_m
111.        for r in row_a:
112.            # calculate q1_h and q2_h
113.            q1_h_a[r] = q1_hand (0.35, 1.3, q, a, Lcan, Lsp, x_p_a[r],
    d_sup, d_mid)
114.            q2_h_a[r] = q2_hand (0.35, 1.3, q, a, Lcan, Lsp, x_p_a[r],
    d_sup, d_mid)
115.            #print (q1_h_a)
116.            # calculate moment and shear (full matrix)
117.            curves_m [r,] =[ hc.M_q (alfa, Lsp, x, x_p_a[r], q, a,
    q1_h_a[r], q2_h_a[r]) for x in x_sec_a]
118.            curves_v [r,] =[ hc.V_q (alfa, Lsp, x, x_p_a[r], q, a,
    q1_h_a[r], q2_h_a[r]) for x in x_sec_a]
119.
120.            # calculate eq width for moment and shear (full matrix)
121.            w_m [r,] =[ w_M (alfa, Lcan, Lsp, d_edge, d_mid, d_sup,
    x_p_a[r], x) for x in x_sec_a]
122.            w_v [r,] =[ w_V (alfa, Lcan, Lsp, d_edge, d_mid, d_sup,
    x_p_a[r], x) for x in x_sec_a]
123.
124.        #print(w_v)
125.        # overwriting results
126.        for c in column_a:
127.            m_max_a [c] = max((curves_m/w_m) [:,c])
128.            m_min_a [c] = min((curves_m/w_m) [:,c])
129.            v_max_a [c] = max((curves_v/w_v) [:,c])
130.            v_min_a [c] = min((curves_v/w_v) [:,c])
131.
132.        matrix_m = pd.DataFrame((w_m))
133.        matrix_v = pd.DataFrame((w_v))
134.        #print (matrix_m)
135.        #print(curves_v/w_v)
136.        #print (m_max_a)
137.        #print (m_min_a)
138.        #print (v_max_a)
139.        #print (v_min_a)
140.        result = pd.DataFrame(list(zip(m_max_a,
    m_min_a,v_max_a,v_min_a)))
141.        result.to_csv('result.csv', mode='a', header=False)
142.
143.        matrix_m.to_csv('matrix_m1.csv', mode='a', header=False)
144.        matrix_v.to_csv('matrix_v1.csv', mode='a', header=False)
145.        return
146.
147.
148.
149.    x_sec = [0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9,9.5]
150.    x_p =
    [0.2,0.4,0.6,0.8,1,1.2,1.4,1.6,1.8,2,2.2,2.4,2.8,3,3.2,3.4,3.6,3.8,4,4.2
    ,4.4,4.6,4.8,5]
151.
152.    envelope (2.5, 0.4, x_sec, x_p, 0.5, 5, 0.3, 0.3, 0.3)
```

12.12 XLSX.py

```
1. import pandas as pd
2. import os
3. import xlsxwriter
4. import shutil
5.
6.
7.
8. def _chkfile(Pathdic, ParserIn):
9.     Direction = ParserIn.get('Direction')
10.    if not os.path.exists('xlsx/' + Pathdic['dir'] ):
11.        os.mkdir('xlsx/' + Pathdic['dir'] )
12.
13.    if os.path.isfile('xlsx/' + Pathdic['dir'] + '/'+ Pathdic['fil']  +
   Direction + '.xlsx'):
14.        overwirte = input("File exists, overwirte?(Y,N) (Default =
   overwirte)")
15.        if overwirte == 'N':
16.            print('Please rename and close the file and retry again')
17.        else:
18.            while True:
19.                try:
20.                    os.remove('xlsx/' + Pathdic['dir']  + '/'+
   Pathdic['fil']  + Direction + '.xlsx')
21.                except IOError:
22.                    input('Please close Excelfile to continue')
23.                    continue
24.                break
25.
26.def _no2lt(col):
27.    col = xlsxwriter.utility.xl_col_to_name(col)
28.    return (col)
29.
30.def _color(workbook, No, bold ):
31.    No_dict = { '01' :'cell_format01','02' : 'cell_format02', '03' :
   'cell_format03'}
32.    format_dict = {'01': 'E0F2F6' ,'02' : 'yellow' ,'03' : 'E1E1E1'}
33.    No_dict[No] = workbook.add_format()
34.    No_dict[No].set_pattern(1)
35.    No_dict[No].set_bg_color(format_dict[No])
36.    if bold == True: No_dict[No].set_bold()
37.    return No_dict[No]
38.
39.
40.def _fill_data(ParserIn, FEMIn, Para_dict, worksheet, format_dict, Para,
   TableForm, col):
41.    q1q2 = 1
42.    col1 = col + 1
43.    w_col = len(ParserIn['W'])
44.    sec_dict = Para_dict.get(Para)
45.    for Sec in sec_dict.keys():
46.        w_dict = sec_dict.get(Sec)
47.        worksheet.set_column(_no2lt(col1+1) + ':' + _no2lt(col1+w_col*2-
   1), None, None, {'hidden': True})       #Hiding col
```

```
48.         worksheet.write(TableForm['Para_avg'], col1, Para +
    '_avg',format_dict['head'])     #Writting header for avg
49.         worksheet.write(TableForm['Hand_calc'], col1, Para +
    '_h',format_dict['head'])     #Writting header for h
50.         if Sec in FEMIn['ysupp']:
51.             worksheet.write(TableForm['q1q2'], col1, 'q' + str(q1q2)
    ,format_dict['head'])     #Writting header for q1q2
52.             q1q2 + 1
53.         row = 1
54.         for W in w_dict.keys():
55.             data_dict = w_dict.get(W)
56.             worksheet.write_column(TableForm['Para_avg']+1, 0, ('W=' +
    str(x) for x in ParserIn['W']),format_dict['head'])    #header for w col
    for avg
57.             worksheet.write_column(TableForm['Eq_header']+1, 0, ('W=' +
    str(x) for x in ParserIn['W']), format_dict['head'])   #header for w col
    for beq
58.             worksheet.write_column(TableForm['Hand_calc']+1, 0, ('W=' +
    str(x) for x in ParserIn['W']), format_dict['head'])
59.             worksheet.write_column(TableForm['q1q2']+1, 0, ('W=' +
    str(x) for x in ParserIn['W']), format_dict['head'])
60.
61.             worksheet.write(TableForm['Main Header'], col, 'Y_w=' +
    str(W),format_dict['head'])          #header for x col
62.             worksheet.write_column(TableForm['Main Header']+1, col,
    data_dict['X'],format_dict['head'])        #X values col
63.
64.             worksheet.write(TableForm['Main Header'], col+1, Para +'_Y='
    + str(Sec),format_dict['head'])     #header Para col
65.             worksheet.write_column(TableForm['Main Header']+1, col+1,
    data_dict[Para],format_dict[Para])       #Para values col
66.
67.             worksheet.write(row+TableForm['Para_avg'], col1,
    data_dict[Para + '_avg'],format_dict[Para])              #average Para
    value
68.
69.             worksheet.write(row+TableForm['Hand_calc'], col1,
    data_dict[Para + '_h'],format_dict[Para])         #Writting hand calc
70.             if Sec in FEMIn['ysupp']:
71.                 worksheet.write(row+TableForm['q1q2'], col1,
    data_dict['q' + str(Sec)],format_dict[Para])
72.
73.             if data_dict[Para + '_avg'] != 0:
74.                 worksheet.write(row+TableForm['Eq_header'], col1,
    data_dict[Para + '_h']/data_dict[Para + '_avg'],format_dict[Para])
    # Beq values
75.                 worksheet.write(row+TableForm['-ve_header'], col1,
    data_dict[Para + '_h']/2/data_dict[Para + '_avg'],format_dict[Para])
    # 1/2 Beq values
76.                 worksheet.write(row+TableForm['+ve_header'], col1,
    data_dict[Para + '_h']/(-2)/data_dict[Para + '_avg'],format_dict[Para])
    # -1/2 Beq values
77.             row+=1
78.             col+=2
79.         worksheet.write(TableForm['Eq_header'], col1, Sec,
    format_dict['head'])    #Section header
80.         worksheet.write(TableForm['y_head'], col1, Sec,
    format_dict['head'])
81.         col1 = col+1
```

```
        worksheet.write('A'+ str(TableForm['-ve_header']+4), '=MAX(B'+ str(TableForm['-ve_header']+3) +':'+_no2lt(col) + str(TableForm['+ve_header']+1) + ')')  #Max value of Beq for plot
        worksheet.write('A'+ str(TableForm['-ve_header']+5), '=-A'+ str(TableForm['-ve_header']+4))     #- Max value of Beq for plot

    return(col1)


def _chartformat(chart, chart_dict, ParserIn):
    chart.show_hidden_data()
    #chart.set_style(chart_dict['style'])
    axis = {       'name': chart_dict['x_label'],
             'minor_unit': chart_dict['minor_unit'],
             'major_unit': chart_dict['major_unit'],
        'major_gridlines': {    'visible': True,
                                  'line': {'width': 1.0, 'dash_type': 'solid'}},
        'minor_gridlines': {    'visible': True,
                                  'line': {'width': 1.0, 'dash_type': 'solid'}},
        'major_unit_type': chart_dict['x_unit']}

        chart.set_x_axis(axis)
        axis['name'] = chart_dict['y_label']
        axis['major_unit_type'] = chart_dict['y_unit']
        chart.set_y_axis(axis)

        chart.set_size({'x_scale': chart_dict['size'] , 'y_scale': chart_dict['aspect_ratio']*chart_dict['size']})
        chart.set_title({'name': chart_dict['Tiltle']})
        return(chart)


    def _Graph(chart, sheetname, FEMIn, ParserIn, TableForm):
        w_col = len(ParserIn['W'])
        chart.add_series({'categories': '{' + str(sheetname) +'}',
                              'values': '{0}',
                                'name': 'Load',
                              'marker': {
                                'type': 'square',
                                'size': 20,
                              'border': {'color': 'black'},
                                'fill':   {'color': 'red'}}})


        chart.add_series({'categories': '{' + str(FEMIn['ysupp'][0])  + ',' + str(FEMIn['ysupp'][0]) +'}',
                              'values': '=' + sheetname + '!A' + str(TableForm['-ve_header']+4) + ':' + 'A'+ str(TableForm['-ve_header']+5),
                                'name': 'Support 1',
                              'marker': {
                                'type': 'square',
                                'size': 2,
                              'border': {'color': 'red'},
                                'fill':   {'color': 'red'}} ,
                                'line': {'color': 'red','width': 2}})

```

```
132.
133.        chart.add_series({'categories': '{' + str(FEMIn['ysupp'][1])  +
    ',' + str(FEMIn['ysupp'][1]) +'}',
134.                              'values': '=' + sheetname + '!A' +
    str(TableForm['-ve_header']+4) + ':' + 'A'+ str(TableForm['-
    ve_header']+5),
135.                                'name': 'Support 2',
136.                              'marker': {
137.                                'type': 'square',
138.                                'size': 2,
139.                              'border': {'color': 'red'},
140.                                'fill':   {'color': 'red'}} ,
141.                                'line': {'color': 'red', 'width': 2}})
142.
143.                                #chart1.add_series({'categories':'{' +
    str(FEMIn['yload'] + FEMIn['dia']) + ',' + str(FEMIn['yload'] +
    FEMIn['dia']) + ',' +
144.       #                                        str(FEMIn['yload'] -
    FEMIn['dia']) + ',' + str(FEMIn['yload'] - FEMIn['dia']) +'}',
145.        #                       'values':'{' + str(- FEMIn['dia']) +
    ',' + str(  FEMIn['dia']) + ',' +
146.        #                                        str(  FEMIn['dia']) +
    ',' + str(- FEMIn['dia']) +'}',
147.         #                        'name': 'Load',
148.
149.        #                         'line': {'color': 'red'}})
150.        return
151.
152.    def _eqchart(ParserIn, sheetname, Para, workbook, col1, col2,
    TableForm):
153.        #chart5 = workbook.add_chart({'type': 'scatter', 'subtype':
    'smooth'})
154.        w_col = len(ParserIn['W'])
155.        chart_dict = {'Tiltle' : 'Equivalent Width for '+ Para,
156.                        'size' : 1.1,
157.                       'style' : 14,
158.                      'x_unit' : 'm',
159.                     'x_label' : 'Width (m)',
160.                      'y_unit' : 'm',
161.                     'y_label' : 'Length (m)',
162.                   'minor_unit': ParserIn['Mesh'],
163.                   'major_unit': 1 ,
164.                 'aspect_ratio': ParserIn['aspect_ratio']}
165.
166.        chart = _chartformat(workbook.add_chart({'type': 'scatter'}),
    chart_dict, ParserIn)
167.
168.        scol1 = _no2lt(col1)
169.        scol2 = _no2lt(col2-1)
170.        for row in range(TableForm['-
    ve_header']+3,TableForm['+ve_header']+2):
171.            chart.add_series({'categories': '=(' + sheetname + '!$' +
    scol1 + '$'+ str(TableForm['Eq_header']+1) +':$'+  scol2 +'$'+
    str(TableForm['Eq_header']+1) +','
172.                                          + '' + sheetname + '!$' +
    scol1 + '$'+ str(TableForm['Eq_header']+1) +':$'+  scol2 +'$'+
    str(TableForm['Eq_header']+1) +')',
173.                                 'values': '=(' + sheetname + '!$' +
    scol1 + '$' + str(row)  + ':$'+  scol2 +'$'+ str(row) + ','
```

```
174.                                                + '' + sheetname + '!$' +
   scol1 + '$' + str(row+w_col)  + ':$'+  scol2 +'$'+ str(row+w_col) + ')'
   ,
175.                                        'name': '=' + sheetname + '!$A$'+
   str(row - w_col),
176.                                        'marker': {'type': 'automatic'},})
177.         return(chart)
178.
179.     def _secchart(ParserIn, sheetname, workbook, col1, col2, dlen,
   refcell):
180.         #chart5 = workbook.add_chart({'type': 'scatter', 'subtype':
   'smooth'})
181.         chart_dict = {'Tiltle' : '=' + sheetname + '!$' +
   refcell['Title'],
182.                          'size' : 1.2,
183.                         'style' : 14,
184.                        'x_unit' : 'm',
185.                      'x_label' : 'Width (m)',
186.                        'y_unit' : 'm',
187.                      'y_label' : 'Length (m)',
188.                    'minor_unit': ParserIn['Mesh'],
189.                    'major_unit': 1,
190.                  'aspect_ratio': 1.7 }
191.
192.         major_gridlines = {'visible': True,
193.                            'line': {'width': 1.0, 'dash_type':
   'solid'}}
194.         minor_gridlines = {'visible': True,
195.                            'line': {'width': 1.0, 'dash_type':
   'solid'}}
196.
197.         refcell['x']['major_gridlines'] = major_gridlines
198.         refcell['x']['minor_gridlines'] = minor_gridlines
199.         refcell['y']['major_gridlines'] = major_gridlines
200.         refcell['y']['minor_gridlines'] = minor_gridlines
201.
202.         chart = _chartformat(workbook.add_chart({'type': 'scatter',
   'subtype': 'smooth'}), chart_dict, ParserIn)
203.         chart.set_x_axis(refcell['x'])
204.         chart.set_y_axis(refcell['y'])
205.
206.         i=0
207.         for col in range(col1+3 ,col2+3, 2):
208.             chart.add_series({'categories': '=(' + sheetname + '!$' +
   _no2lt(col) + '$1:$'+  _no2lt(col) +'$'+ str(dlen) + ')',
209.                                 'values': '=(' + sheetname + '!$' +
   _no2lt(col+1) + '$1:$'+  _no2lt(col+1) +'$'+ str(dlen) + ')',
210.                                   'name': 'W =' +
   str(ParserIn['W'][i]),
211.                                 'marker': {
212.                                   'type': 'automatic',
213.                                   'size': 2,} ,
214.                                   'line': {'width': 1.5}})
215.             i+=1
216.         return(chart)
217.
218.     def _multisecchart(ParserIn, Para_dict, sheetname, workbook, col1,
   col2, Para, refcell, TableForm):
```

```
219.         #chart5 = workbook.add_chart({'type': 'scatter', 'subtype':
   'smooth'})
220.         len_itr =
   len(Para_dict['My'][ParserIn['Section_My'][0]][max(ParserIn['W'])]['X'])
221.         w_col = len(ParserIn['W'])
222.
223.         if Para == 'My':
224.             col2 = col1
225.             col1 = 0
226.
227.         chart_dict = {'Tiltle' : refcell['Title'],
228.                         'size' : 1.2,
229.                       'style' : 14,
230.                     'x_unit' : 'm',
231.                   'x_label' : 'Width (m)',
232.                    'y_unit' : 'm',
233.                   'y_label' : 'Length (m)',
234.                 'minor_unit': ParserIn['Mesh'],
235.                 'major_unit': 1,
236.               'aspect_ratio': 1.7 }
237.
238.         major_gridlines = {'visible': True,
239.                            'line': {'width': 1.0, 'dash_type':
   'solid'}}
240.         minor_gridlines = {'visible': True,
241.                            'line': {'width': 1.0, 'dash_type':
   'solid'}}
242.
243.         refcell['x']['major_gridlines'] = major_gridlines
244.         refcell['x']['minor_gridlines'] = minor_gridlines
245.         refcell['y']['major_gridlines'] = major_gridlines
246.         refcell['y']['minor_gridlines'] = minor_gridlines
247.
248.         chart = _chartformat(workbook.add_chart({'type': 'scatter',
   'subtype': 'smooth'}), chart_dict, ParserIn)
249.         chart.set_x_axis(refcell['x'])
250.         chart.set_y_axis(refcell['y'])
251.
252.         i=0
253.         for col in range(col1 ,col2, w_col*2):
254.             chart.add_series({'categories': '=(' + sheetname + '!$' +
   _no2lt(col) + '$'+ str(TableForm['Main Header']+2) +':$'+  _no2lt(col)
   +'$'+ str(TableForm['Main Header']+1 + len_itr) + ')',
255.                                 'values': '=(' + sheetname + '!$' +
   _no2lt(col+1) + '$'+ str(TableForm['Main Header']+2) +':$'+
   _no2lt(col+1) +'$'+ str(TableForm['Main Header']+1 + len_itr) + ')',
256.                                   'name': '=' + sheetname + '!$' +
   _no2lt(col+1) + '$'+ str(TableForm['Main Header']+1),
257.                                 'marker': {
258.                                   'type': 'automatic',
259.                                   'size': 2,} ,
260.                                   'line': {'width': 1.5}})
261.             i+=1
262.         return(chart)
263.
264.     def _fill_formula(ParserIn, Para_dict, worksheet, col, refcell,
   overwrite, TableForm):
265.         w_col = len(ParserIn['W'])
```

```
266.         len_itr =
   len(Para_dict['My'][ParserIn['Section_My'][0]][max(ParserIn['W'])]['X'])
   #fixed to My
267.
268.         worksheet.write_column( TableForm['Main Header']+1 , col + 2,
   list(range(1,len_itr+1)))
269.
270.         for j in range (TableForm['Main Header']+1, len_itr +
   TableForm['Main Header']+1):
271.             for i in range(-2, w_col*2-2):
272.                 row = ('OFFSET($A$'+ str(TableForm['Main Header']+1)
   +',$'+ _no2lt(col+2)
273.                     +str(j+1) +',MATCH($'+ refcell + ',$'+
   str(TableForm['Main Header']+1) +
274.                     ':$'+ str(TableForm['Main Header']+1) + ',0)+' +
   str(i) +')')
275.                 worksheet.write( j , col + 5 + i, '=IF(' + row + '=0,'
   + _no2lt(col + 5 + i) + str(j) + ',' + row + ')')
276.             k = 0
277.             if overwrite == True:
278.                 for i in range(-1, w_col*2-3,2):
279.                     row = ('=OFFSET($A$'+ str(TableForm['Para_avg'] +
   3) +',' + str(k) + ',MATCH($'+ refcell + ',$'
280.                                         + str(TableForm['Main
   Header']+1) + ':$'+ str(TableForm['Main Header']+1) + ',0)-1)')
281.                     worksheet.write( j , col + 7 + i, row)
282.                     k+=1
283.         worksheet.set_column(_no2lt(col+2) + ':' + _no2lt(col + w_col*2
   +3), None, None, {'hidden': True})
284.         return(col + w_col*2, w_col*4 + 8 + len_itr)
285.
286.
287.     def xlsExcel(Pathdic, workbook, Para_dict, ParserIn, FEMIn,
   TableForm):
288.         w_col = len(ParserIn['W'])
289.         Direction = ParserIn.get('Direction')
290.
291.         sheetname = str(FEMIn['yload'])
292.
293.         worksheet = workbook.add_worksheet(sheetname)
294.
295.         format_dict = {'M2' : _color(workbook, '01', False)
296.                       ,'Vy' : _color(workbook, '02', False)
297.                     ,'head' : _color(workbook, '03', True)
298.                       ,'My' : _color(workbook, '01', False)
299.                      ,'Mxy' : _color(workbook, '02', False)
300.                                                            }
301.
302.         Para1 = ParserIn['Parameter'][0]
303.         Para2 = ParserIn['Parameter'][1]
304.
305.         col1 = 1
306.         col2 = _fill_data(ParserIn, FEMIn, Para_dict,  worksheet,
   format_dict, Para1, TableForm, 0)
307.         chart1 = _eqchart(ParserIn, sheetname, Para1, workbook, col1,
   col2, TableForm)
308.         _Graph(chart1, sheetname, FEMIn, ParserIn, TableForm)
309.
310.         col1 = col2
```

```
311.         col2 = _fill_data(ParserIn, FEMIn, Para_dict,  worksheet,
   format_dict, Para2, TableForm, col2-1)
312.         chart2 = _eqchart(ParserIn, sheetname, Para2, workbook, col1,
   col2, TableForm)
313.         _Graph(chart2, sheetname, FEMIn, ParserIn, TableForm)
314.
315.
316.         refcell = { '1' : {'Title' : _no2lt(col2+1)+ '$' +
   str(w_col*4+42),
317.                                 'x':  {'name' : 'Width (m)',
318.                                        'min':
   min(Para_dict['My'][ParserIn['Section_My'][0]][ParserIn['W'][w_col-
   1]]['X'])*0.75,
319.                                        'max':
   max(Para_dict['My'][ParserIn['Section_My'][0]][ParserIn['W'][w_col-
   1]]['X'])*1.25},
320.                                 'y':  {'name' : Para1 + ' (kN.m/m)',
321.                                        'minor_unit': '0.01',
   #min(Para_dict['M2'][ParserIn['Section_M2'][0]][max(ParserIn['W'])]['M2'
   ]),
322.                                        'major_unit': '0.05'}},
   #max(Para_dict['M2'][ParserIn['Section_M2'][0]][max(ParserIn['W'])]['M2'
   ])}},
323.                     '2' : {'Title' : _no2lt(col2+1)+ '$' +
   str(w_col*4+43),
324.                                 'x':  {'name' : 'Width (m)',
325.                                        'min':
   min(Para_dict['Vy'][ParserIn['Section_Vy'][0]][ParserIn['W'][w_col-
   1]]['X'])*0.75,
326.                                        'max':
   max(Para_dict['Vy'][ParserIn['Section_Vy'][0]][ParserIn['W'][w_col-
   1]]['X'])*1.25},
327.                                 'y':  {'name' : Para2 + ' (kN/m)',
328.                                        'minor_unit': '0.01',
   #min(Para_dict['M2'][ParserIn['Section_M2'][0]][max(ParserIn['W'])]['M2'
   ]),
329.                                        'major_unit': '0.05'}},
330.                     '3' : {'Title' : Para1 + ' all sections',
331.                                 'x':  {'name' : 'Width (m)',
332.                                        'min':
   max(Para_dict['My'][ParserIn['Section_My'][0]][max(ParserIn['W'])]['X'])
   /3,
333.                                        'max':
   max(Para_dict['My'][ParserIn['Section_My'][0]][max(ParserIn['W'])]['X'])
   *2/3},
334.                                 'y':  {'name' : Para1 + ' (kN.m/m)',
335.                                        'minor_unit': '0.01',
   #min(Para_dict['M2'][ParserIn['Section_M2'][0]][max(ParserIn['W'])]['M2'
   ]),
336.                                        'major_unit': '0.05'}},
   #max(Para_dict['M2'][ParserIn['Section_M2'][0]][max(ParserIn['W'])]['M2'
   ])}},
337.                     '4' : {'Title' : Para2 + ' all sections',
338.                                 'x':  {'name' : 'Width (m)',
339.                                        'min':
   max(Para_dict['Vy'][ParserIn['Section_Vy'][0]][max(ParserIn['W'])]['X'])
   /3,
```

```
                                        'max': max(Para_dict['Vy'][ParserIn['Section_Vy'][0]][max(ParserIn['W'])]['X'])*2/3},
                                 'y':  {'name' : Para2 + ' (kN/m)',
                                        'minor_unit': '0.01', #min(Para_dict['M2'][ParserIn['Section_M2'][0]][max(ParserIn['W'])]['M2']),
                                        'major_unit': '0.05'}}}

        Tamplate_dict = {'A'+ str(TableForm['y_head']+1): 'Y=',
                         'A'+ str(TableForm['Para_avg']+1): 'w',
                         'A'+ str(TableForm['Hand_calc']) : 'Hand Calculation',
                         'A'+ str(TableForm['Hand_calc']+1) : 'My/Vy',
                         'A'+ str(TableForm['q1q2']+1) : 'q_Dummy',
                         'A'+ str(TableForm['Eq_header']+1) : 'Eq.width (m)',
                       'A'+ str(TableForm['-ve_header']+3) : 'Eq.width (m)',
            _no2lt(col2+1)+ str(w_col*4+41) : 'Plot 1',
                      refcell['1']['Title'] : Para1 +'_Y=' + str(ParserIn['Section_' + Para1][0]),
                      refcell['2']['Title'] : Para2 +'_Y=' + str(ParserIn['Section_' + Para2][0])}

        for key in Tamplate_dict.keys():
            worksheet.write(key, Tamplate_dict[key], format_dict['head'])


        hidden_rows = [TableForm['Para_avg']+1 , TableForm['Eq_header']+1, TableForm['q1q2']+1, TableForm['Hand_calc']+1]

        for row in hidden_rows + list(range((TableForm['-ve_header']+1), (w_col+ TableForm['+ve_header']+1))):
            worksheet.set_row(row, None, None, {'hidden': True})

        chart5 = _multisecchart(ParserIn, Para_dict, sheetname, workbook, col1-1, col2, Para1, refcell['3'], TableForm)

        chart6 = _multisecchart(ParserIn, Para_dict, sheetname, workbook, col1-1, col2-1, Para2, refcell['4'], TableForm)

        col1 = col2
        (col2, dlen) = _fill_formula(ParserIn, Para_dict, worksheet, col2, refcell['1']['Title'], True, TableForm)
        chart3 = _secchart(ParserIn, sheetname, workbook, col1, col2, dlen, refcell['1'])

        col1 = col2+1
        (col2, dlen) = _fill_formula(ParserIn, Para_dict, worksheet, col2+1, refcell['2']['Title'], True, TableForm)
        chart4 = _secchart(ParserIn, sheetname, workbook, col1, col2, dlen, refcell['2'])


        worksheet.insert_chart( _no2lt(col2 ) + '2', chart1)
        worksheet.insert_chart( _no2lt(col2 +13) + '2', chart2)
        worksheet.insert_chart( _no2lt(col2 ) + '66', chart3)
```

```
        worksheet.insert_chart( _no2lt(col2 +13) + '66', chart4)
        worksheet.insert_chart( _no2lt(col2 ) + '96', chart5)
        worksheet.insert_chart( _no2lt(col2 +13) + '96', chart6)
```

12.13 fdAPI_wrapper_mod.py

```
1. #!/usr/bin/env python3
2.
3. # --- Import and registrations ---
4. import subprocess
5. import xml.etree.ElementTree as ET
6. import xml.dom.minidom as minidom
7. import os
8. import uuid
9. import math
10. from datetime import datetime
11.
12. # --- Databases dir ---
13. tree_mat = ET.parse('data\\materials.struxml')
14. tree_sec = ET.parse('data\\sections.struxml')
15.
16. # --- Installation dir ---
17. FD_root = 'c:\\program files\\strusoft\\fem-design 19\\fd3dstruct'
18.
19. # ----------------------------------------------------------------------
20. # ----------------------------------------------------------------------
21. # ----------------------------------------------------------------------
22.
23. # --- Fixed variables (DO NOT CHANGE) ---
24. i = 0
25. nrcomb = 0
26. u_beam = 0
27. u_column = 0
28. u_truss = 0
29. u_plate = 0
30. u_wall = 0
31. u_edge = 0
32. u_cover = 0
33. u_support = 0
34. u_storey = 0
35. u_axis = 0
36. mat_handler = []
37. xml_annex = ''
38. xml_bars = []
39. xml_shells = []
40. xml_pointsupports = []
41. xml_linesupports = []
42. xml_surfacesupports = []
43. xml_loadcases = []
44. xml_loadcombs = []
45. xml_pointloads = []
46. xml_lineloads = []
47. xml_surfaceloads = []
48. xml_complexsection = []
49. orgUTC = '1970-01-01T00:00:00.000'
50.
51. root_mat = tree_mat.getroot()
52. root_sec = tree_sec.getroot()
53. ET.register_namespace('', 'urn:strusoft')
54. dir_path = os.path.dirname(os.path.realpath(__file__))
55.
56.
```

```
57. class coord:
58.     def __init__(self, x=0, y=0, z=0):
59.         self.x = x
60.         self.y = y
61.         self.z = z
62.
63.
64. class XML:
65.     def __init__(self, filename='', batch='', export=''):
66.         self.filename = filename
67.         self.batch = batch
68.         self.export = export
69.
70.
71. class material:
72.     def __init__(self, type='', creep='', shrinkage='' , poisson=''):
    #Modified
73.         self.type = type
74.         self.creep = creep
75.         self.shrinkage = shrinkage
76.         self.poisson = poisson
77.
78.
79. def ID(type):
80.     global u_beam, u_column, u_truss, u_cover, u_plate, u_wall, u_edge,
    u_support, u_axis
81.     if type == 'beam':
82.         u_beam = u_beam + 1
83.         return 'B.'+str(u_beam)
84.     if type =='column':
85.         u_column = u_column + 1
86.         return 'C.'+str(u_column)
87.     if type == 'truss':
88.         u_truss = u_truss + 1
89.         return 'T.'+str(u_truss)
90.     if type == 'cover':
91.         u_cover = u_cover + 1
92.         return 'CO.'+str(u_cover)
93.     if type == 'plate':
94.         u_plate = u_plate + 1
95.         return 'P.' +str(u_plate)
96.     if type == 'wall':
97.         u_wall = u_wall + 1
98.         return 'W.' +str(u_wall)
99.     if type == 'edge':
100.            u_edge = u_edge + 1
101.            return 'CE.' +str(u_edge)
102.        if type == 'support':
103.            u_support = u_support + 1
104.            return 'S.'+str(u_support)
105.        if type == 'axis':
106.            u_axis = u_axis + 1
107.            return str(u_axis)
108.
109.
110.     def genGUID():
111.         return str(uuid.uuid4())
112.
113.
```

```
114.     def genUTC():
115.         return datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%S.000")
116.
117.
118.     def nmz(v): # This function will normalize a vector
119.         normalized = [v[0]/math.sqrt(v[0]**2 + v[1]**2 + v[2]**2),
    v[1]/math.sqrt(v[0]**2 + v[1]**2 + v[2]**2), v[2]/math.sqrt(v[0]**2 +
    v[1]**2 + v[2]**2)]
120.         return normalized
121.
122.
123.     def normal(point1, point2, rot): # This function will find the
    normal to a plane defined by three points, two input, one defined as z+1
124.         point3 = coord(point1.x, point1.y, point1.z + 1) # Create a new
    point on the plane
125.         # Define two vectors which defines the plane:
126.         v1 = [point2.x - point1.x, point2.y - point1.y, point2.z -
    point1.z]
127.         v2 = [point3.x - point1.x, point3.y - point1.y, point3.z -
    point1.z]
128.         # Find crossproduct of the two vectors = normal to the plane:
129.         n = [v1[1] * v2[2] - v1[2] * v2[1], -(v1[0] * v2[2] - v1[2] *
    v2[0]), v1[0] * v2[1] - v1[1] * v2[0]]
130.         if math.sqrt(n[0]**2 + n[1]**2 + n[2]**2) == 0:
131.             n = [0, -1, 0]
132.         # Calculate the heading:
133.         heading = [point2.x - point1.x, point2.y - point1.y, point2.z -
    point1.z]
134.         # Adjust normal with regard to rotation:
135.         y = math.cos(rot * math.pi/180)
136.         z = math.sin(rot * math.pi/180)
137.         local_z = [heading[1] * n[2] - heading[2] * n[1], heading[2] *
    n[0] - heading[0] * n[2], heading[0] * n[1] - heading[1] * n[0]]
138.         local_z_norm = nmz(local_z)
139.         n_norm = nmz(n)
140.         n_rot = []
141.         for l in range(len(local_z_norm)):
142.             n_rot.append(n_norm[l] * y + local_z_norm[l] * z)
143.         # Normalize the final normal vector:
144.         n_final = [-n_rot[0]/math.sqrt(n_rot[0]**2 + n_rot[1]**2 +
    n_rot[2]**2), -n_rot[1]/math.sqrt(n_rot[0]**2 + n_rot[1]**2 +
    n_rot[2]**2), -n_rot[2]/math.sqrt(n_rot[0]**2 + n_rot[1]**2 +
    n_rot[2]**2)]
145.         return n_final
146.
147.
148.     def initiateModel(annex):
149.         global xml_annex, xml_root, xml_entities, xml_loads,
    xml_supports, xml_advanced, xml_sections
150.         xml_annex = annex
151.         xml_root = ET.Element('database', struxml_version='01.00.000',
    source_software='FEM-Design 18.00.002', start_time=orgUTC,
    end_time=genUTC(), guid=genGUID(), convertid='00000000-0000-0000-0000-
    000000000000', standard='EC', country=annex, xmlns='urn:strusoft')
152.         xml_entities = ET.SubElement(xml_root, 'entities')
153.         xml_sections = ET.SubElement(xml_root, 'sections')
154.         xml_materials = ET.SubElement(xml_root, 'materials')
155.         xml_loads = ET.SubElement(xml_entities, 'loads')
156.         xml_supports = ET.SubElement(xml_entities, 'supports')
```

```
157.         xml_advanced = ET.SubElement(xml_entities, 'advanced-fem')
158.         ET.SubElement(xml_root, 'end') # endtag
159.
160.     def finish(filename):
161.         for x in xml_complexsection:
162.             xml_sections.remove(x)
163.             xml_sections.append(x)
164.         for li in [xml_bars, xml_shells]: # sorting loop
165.             for x in li:
166.                 xml_entities.remove(x)
167.                 xml_entities.append(x)
168.         xml_entities.remove(xml_loads)
169.         xml_entities.remove(xml_supports)
170.         xml_entities.remove(xml_advanced)
171.         xml_entities.append(xml_loads)
172.         xml_entities.append(xml_supports)
173.         xml_entities.append(xml_advanced)
174.         if u_storey > 0:
175.             xml_entities.remove(xml_storeys)
176.             xml_entities.append(xml_storeys)
177.         if u_axis > 0:
178.             xml_entities.remove(xml_axes)
179.             xml_entities.append(xml_axes)
180.         for li in [xml_pointsupports, xml_linesupports,
    xml_surfacesupports]: # Sorting loop
181.             for x in li:
182.                 xml_supports.remove(x)
183.                 xml_supports.append(x)
184.         for li in [xml_pointloads, xml_lineloads, xml_surfaceloads,
    xml_loadcases, xml_loadcombs]: # sorting loop
185.             for x in li:
186.                 xml_loads.remove(x)
187.                 xml_loads.append(x)
188.         xml_reparsed = minidom.parseString(ET.tostring(xml_root))
189.         xml_root_pretty =
    minidom.parseString(xml_reparsed.toprettyxml())
190.         xml_file_handler = open(filename, "w+")
191.         xml_root_pretty.writexml(xml_file_handler)
192.         xml_file_handler.close()
193.
194.         # clear global
195.         xml_complexsection.clear()
196.         xml_bars.clear()
197.         xml_shells.clear()
198.         xml_pointsupports.clear()
199.         xml_linesupports.clear()
200.         xml_surfacesupports.clear()
201.         xml_pointloads.clear()
202.         xml_lineloads.clear()
203.         xml_surfaceloads.clear()
204.         xml_loadcases.clear()
205.         xml_loadcombs.clear()
206.
207.
208.     def addMaterial(i_material):
209.         matGUID = genGUID()
210.         if i_material.type in ('S235', 'S275', 'S355', 'S420', 'S450',
    'S460'):
211.             nameMat = i_material.type[0] + " " + i_material.type[1:4]
```

```
212.             add_material = ET.SubElement(xml_root.find('materials'),
    'material', guid=matGUID, last_change=orgUTC, action='added',
    standard='EC', country=xml_annex, name=nameMat)
213.             mat_add = root_mat.find(i_material.type)
214.         if i_material.type in ('C12/15', 'C16/20', 'C20/25', 'C25/30',
    'C28/35', 'C30/37', 'C32/40', 'C35/45', 'C40/50', 'C45/55', 'C50/60',
    'C54/65', 'C55/67', 'C58/70', 'C60/75', 'C70/85', 'C80/95', 'C90/105'):
215.             add_material = ET.SubElement(xml_root.find('materials'),
    'material', guid=matGUID, last_change=orgUTC, action='added',
    standard='EC', country=xml_annex, name=i_material.type)
216.             mat_add = root_mat.find(i_material.type.replace('/', ''))
217.             mat_add[0].attrib['creep'] = str(i_material.creep)
218.             mat_add[0].attrib['creep_sls'] = str(i_material.creep)
219.             mat_add[0].attrib['shrinkage'] = str(i_material.shrinkage)
220.             mat_add[0].attrib['nu'] = str(i_material.poisson)
221.         if i_material.type in ('C14', 'C16', 'C18', 'C20', 'C22',
    'C24', 'C27', 'C30', 'C35', 'C40', 'C45', 'C50', 'D18', 'D24', 'D30',
    'D35', 'D40', 'D50', 'D60', 'D70', 'L40s', 'L40c'):
222.             add_material = ET.SubElement(xml_root.find('materials'),
    'material', guid=matGUID, last_change=orgUTC, action='added',
    standard='EC', country=xml_annex, name=i_material.type)
223.             mat_add = root_mat.find(i_material.type)
224.         if i_material.type in ('GL 20h', 'GL 22h', 'GL 24h', 'GL 26h',
    'GL 28h', 'GL 30h', 'GL 32h', 'GL 36h'):
225.             add_material = ET.SubElement(xml_root.find('materials'),
    'material', guid=matGUID, last_change=orgUTC, action='added',
    standard='EC', country=xml_annex, name=i_material.type.replace(' ', '
    '))
226.             mat_add = root_mat.find(i_material.type.replace(' ', ''))
227.         if i_material.type in ('GL 20c', 'GL 22c', 'GL 24c', 'GL 26c',
    'GL 28c', 'GL 30c', 'GL 32c', 'GL 36c'):
228.             add_material = ET.SubElement(xml_root.find('materials'),
    'material', guid=matGUID, last_change=orgUTC, action='added',
    standard='EC', country=xml_annex, name=i_material.type)
229.             mat_add = root_mat.find(i_material.type.replace(' ', ''))
230.         add_material.append(mat_add[0])
231.         return matGUID
232.
233.
234.     def addSection(section, eccentricity):
235.         global i
236.         complexGUID = genGUID()
237.         i = i+1
238.         sec_add = root_sec.find(section)
239.         sec_name_find = sec_add.attrib['name']
240.         simpGUID = sec_add.attrib['guid']
241.         mat_handler.append([i, section, simpGUID])
242.         sec_name = sec_name_find.rstrip(', ').split(', ')
243.         add_section = ET.SubElement(xml_root.find('sections'),
    'section', guid=simpGUID, last_change=orgUTC, action='added',
    name=(sec_name[0] + ', ' + sec_name[1] + ', ' + sec_name[2]),
    type='custom', fd_name_code=sec_name[0], fd_name_type=sec_name[1],
    fd_name_size=sec_name[2])
244.         add_section.append(sec_add[0])
245.         ET.SubElement(add_section, 'end')
246.         add_section_complex = ET.SubElement(xml_root.find('sections'),
    'complex_section', guid=complexGUID, last_change=orgUTC, action='added')
247.         add_section_complex_child_start =
    ET.SubElement(add_section_complex, 'section', pos='0', guid=simpGUID)
```

```
248.         ET.SubElement(add_section_complex_child_start, 'ecc', x=str(0),
   y=str(eccentricity.y), z=str(eccentricity.z))
249.         ET.SubElement(add_section_complex_child_start, 'end')
250.         add_section_complex_child_end =
   ET.SubElement(add_section_complex, 'section', pos='1', guid=simpGUID)
251.         ET.SubElement(add_section_complex_child_end, 'ecc', x=str(0),
   y=str(eccentricity.y), z=str(eccentricity.z))
252.         ET.SubElement(add_section_complex_child_end, 'end')
253.         secFabric = 'False' # Set default value
254.         if add_section.attrib['fd_name_type'] in ('IPE', 'HE-A', 'HE-
   B', 'HE-M', 'I', 'LE', 'LU', 'T', 'TPS', 'U', 'UAP', 'UKB', 'UKC',
   'UPE', 'UPE-Swedish', 'VKR', 'Z'):
255.             secFabric = 'rolled'
256.             add_section.attrib['fd-mat'] = '0'
257.         if add_section.attrib['fd_name_type'] in ('CHS', 'KCKR', 'KKR',
   'VCKR'):
258.             secFabric = 'cold_worked'
259.             add_section.attrib['fd-mat'] = '1'
260.         if add_section.attrib['fd_name_type'] in ('D', 'DR'):
261.             secFabric = 'welded'
262.             add_section.attrib['fd-mat'] = '2'
263.         if sec_name[0] == 'Concrete sections':
264.             add_section.attrib['fd-mat'] = '3'
265.         if sec_name[0] == 'Timber sections':
266.             add_section.attrib['fd-mat'] = '4'
267.         xml_complexsection.append(add_section_complex)
268.         return [simpGUID, complexGUID, secFabric, eccentricity]
269.
270.
271.     def openFD(filename):
272.         subprocess.Popen([FD_root, dir_path + '\\' + filename])
273.
274.
275.     def runFD(analysis, save, close, design, filename, batchfile,
   exportfile):            #Batchfile and filename must be string lists
276.         run_root = ET.Element('fdscript')
277.         run_root.attrib['xmlns:xsi'] =
   'http://www.w3.org/2001/XMLSchema-instance'
278.         run_root.attrib['xsi:noNamespaceSchemaLocation'] =
   'fdscript.xsd'
279.         run_header = ET.SubElement(run_root, 'fdscriptheader')
280.         run_title = ET.SubElement(run_header, 'title')
281.         run_title.text = 'Generated script for FEM-Design'
282.         run_version = ET.SubElement(run_header, 'version')
283.         run_version.text = '1.0'
284.         run_module = ET.SubElement(run_header, 'module')
285.         run_module.text = 'SFRAME'
286.         run_log = ET.SubElement(run_header, 'logfile')
287.         run_log.text = dir_path
288.         run_cmdopen = ET.SubElement(run_root, 'cmdopen', command='; CXL
   CS2SHELL OPEN')
289.         run_filename = ET.SubElement(run_cmdopen, 'filename')
290.         run_filename.text = dir_path + '\\' + filename
291.         if design == 'steel':
292.             ET.SubElement(run_root, 'cmduser', command='; CXL $MODULE
   STEELDESIGN')
293.         elif design == 'timber':
294.             ET.SubElement(run_root, 'cmduser', command='; CXL $MODULE
   TIMBERDESIGN')
```

```
295.         elif design == 'RC':
296.             ET.SubElement(run_root, 'cmduser', command='; CXL $MODULE
    RCDESIGN')
297.         else:
298.             ET.SubElement(run_root, 'cmduser', command='; CXL $MODULE
    RESMODE')
299.         run_cmdcalc = ET.SubElement(run_root, 'cmdcalculation',
    command='; CXL $MODULE CALC')
300.         run_analysis = ET.SubElement(run_cmdcalc, 'analysis')
301.         if analysis == 'STAB':
302.             run_analysis.attrib['calcStab'] = '1'
303.         elif analysis == 'IMP':
304.             run_analysis.attrib['calcImpf'] = '1'
305.         else:
306.             run_analysis.attrib['calcCase'] = '1'
307.             run_analysis.attrib['calcComb'] = '1'
308.         run_analysis.attrib['elemfine'] = '1'
309.         run_analysis.attrib['peaksmoothing'] = '1'
310.         run_comb = ET.SubElement(run_analysis, 'comb', NLEmaxiter='30',
    PLdefloadstep='20', PLminloadstep='2', PLmaxeqiter='30')
311.         if design == 'steel' or design == 'timber' or design == 'RC':
312.             run_analysis.attrib['calcDesign'] = '1'
313.             run_design = ET.SubElement(run_cmdcalc, "design")
314.             cmax = ET.SubElement(run_design, "cmax")
315.             autodesign = ET.SubElement(run_design, "autodesign")
316.             autodesign.text= "false"
317.             check = ET.SubElement(run_design, "check")
318.             check.text = "true"
319.         for i_comb in range(nrcomb):
320.             run_combitem = ET.SubElement(run_comb, 'combitem')
321.             if analysis == 'NLE':
322.                 run_combitem.attrib['NLE'] = '1'
323.             if analysis == 'NLE+PL':
324.                 run_combitem.attrib['NLE'] = '1'
325.                 run_combitem.attrib['PL'] = '1'
326.             if analysis == 'CR':
327.                 run_combitem.attrib['NLE'] = '1'
328.                 run_combitem.attrib['Cr'] = '1'
329.             if analysis == '2ND':
330.                 run_combitem.attrib['NLE'] = '1'
331.                 run_combitem.attrib['f2nd'] = '1'
332.         for i in range (0, len(exportfile)):
333.             ET.SubElement(run_root, 'cmdlistgen', command='$ MODULECOM
    LISTGEN', bscfile=dir_path + '\\' + batchfile[i], outfile=dir_path +
    '\\' + exportfile[i])
334.         if save == True:
335.             run_cmdsave = ET.SubElement(run_root, 'cmdsave', command='; 
    CXL CS2SHELL SAVE')
336.             run_save = ET.SubElement(run_cmdsave, 'filename')
337.             run_save.text = dir_path + '\\' + filename + '.str'
338.         if close != False:
339.             ET.SubElement(run_root, 'cmdendsession')
340.         run_reparsed = minidom.parseString(ET.tostring(run_root))
341.         run_root_pretty =
    minidom.parseString(run_reparsed.toprettyxml())
342.         run_file_handle = open("data\\run.fdscript", "w+")
343.         run_root_pretty.writexml(run_file_handle)
344.         run_file_handle.close()
```

```
345.         subprocess.run([FD_root, '/s', dir_path +
   "\\data\\run.fdscript"])
346.
347.
348.     def addAxis(point1, point2):
349.         global xml_axes
350.         axisID = ID('axis')
351.         if u_axis == 1:
352.             xml_axes = ET.SubElement(xml_entities, 'axes')
353.         add_axis = ET.SubElement(xml_axes, 'axis', guid=genGUID(),
   last_change=genUTC(), action='added', id=axisID, id_is_letter='false',
   prefix='')
354.         ET.SubElement(add_axis, 'start_point', x=str(point1.x),
   y=str(point1.y))
355.         ET.SubElement(add_axis, 'end_point', x=str(point2.x),
   y=str(point2.y))
356.
357.
358.     def addStorey(dimensionX, dimensionY, level):
359.         global xml_storeys, u_storey
360.         u_storey = u_storey + 1
361.         if u_storey == 1:
362.             xml_storeys = ET.SubElement(xml_entities, 'storeys')
363.         add_storey = ET.SubElement(xml_storeys, 'storey',
   guid=genGUID(), last_change=genUTC(), action='added', name=('Storey at '
   + str(level) + ' m'), dimension_x=str(dimensionX),
   dimension_y=str(dimensionY))
364.         ET.SubElement(add_storey, 'origo', x='0', y='0', z=str(level))
365.         ET.SubElement(add_storey, 'direction', x='1', y='0')
366.
367.
368.     def addBeam(section_INFO, material_ID, point1, point2, rotation,
   release):
369.         barName = ID('beam')
370.         local_y = normal(point1, point2, rotation)
371.         add_beam = ET.SubElement(xml_root.find('entities'), 'bar',
   name=barName, type='beam', guid=genGUID(), last_change=genUTC(),
   action='added')
372.         add_beam_child = ET.SubElement(add_beam, 'bar_part',
   guid=genGUID(), last_change=genUTC(), action='added', name=(barName +
   '.1'), complex_material=material_ID, complex_section=section_INFO[1],
   ecc_calc='true')
373.         if section_INFO[2] != 'False':
374.             add_beam_child.attrib['made'] = section_INFO[2]
375.         add_beam_grandchild = ET.SubElement(add_beam_child, 'curve',
   type='line')
376.         ET.SubElement(add_beam_grandchild, 'point', x=str(point1.x),
   y=str(point1.y), z=str(point1.z))
377.         ET.SubElement(add_beam_grandchild, 'point', x=str(point2.x),
   y=str(point2.y), z=str(point2.z))
378.         ET.SubElement(add_beam_child, 'local-y', x=str(local_y[0]),
   y=str(local_y[1]), z=str(local_y[2]))
379.         if release == 'hinged':
380.             ET.SubElement(add_beam_child, 'connectivity', m_x='true',
   m_y='true', m_z='true', r_x='true', r_y='false', r_z='false')
381.             ET.SubElement(add_beam_child, 'connectivity', m_x='true',
   m_y='true', m_z='true', r_x='true', r_y='false', r_z='false')
382.         else:
```

```
383.             ET.SubElement(add_beam_child, 'connectivity', m_x='true',
    m_y='true', m_z='true', r_x='true', r_y='true', r_z='true')
384.             ET.SubElement(add_beam_child, 'connectivity', m_x='true',
    m_y='true', m_z='true', r_x='true', r_y='true', r_z='true')
385.         add_beam_ecc = ET.SubElement(add_beam_child, 'eccentricity',
    use_default_physical_alignment='false')
386.         ET.SubElement(add_beam_ecc, 'analytical', x=str(0),
    y=str(section_INFO[3].y), z=str(section_INFO[3].z))
387.         ET.SubElement(add_beam_ecc, 'analytical', x=str(0),
    y=str(section_INFO[3].y), z=str(section_INFO[3].z))
388.         ET.SubElement(add_beam_ecc, 'physical', x=str(0),
    y=str(section_INFO[3].y), z=str(section_INFO[3].z))
389.         ET.SubElement(add_beam_ecc, 'physical', x=str(0),
    y=str(section_INFO[3].y), z=str(section_INFO[3].z))
390.         ET.SubElement(add_beam_child, 'end')
391.         ET.SubElement(add_beam, 'end')
392.         xml_bars.append(add_beam)
393.
394.
395.     def addColumn(section_INFO, material_ID, point1, height, rotation,
    release):
396.         barName = ID('column')
397.         local_y = normal(point1, coord(point1.x, point1.y, point1.z +
    height), rotation)
398.         add_column = ET.SubElement(xml_root.find('entities'), 'bar',
    name=barName, type='column', guid=genGUID(), last_change=genUTC(),
    action='added')
399.         add_column_child = ET.SubElement(add_column, 'bar_part',
    guid=genGUID(), last_change=genUTC(), action='added', name=(barName +
    '.1'), complex_material=material_ID, complex_section=section_INFO[1],
    ecc_calc='true')
400.         if section_INFO[2] != 'False':
401.             add_column_child.attrib['made'] = section_INFO[2]
402.         add_column_grandchild = ET.SubElement(add_column_child,
    'curve', type='line')
403.         ET.SubElement(add_column_grandchild, 'point', x=str(point1.x),
    y=str(point1.y), z=str(point1.z))
404.         ET.SubElement(add_column_grandchild, 'point', x=str(point1.x),
    y=str(point1.y), z=str(point1.z + height))
405.         ET.SubElement(add_column_child, 'local-y', x=str(local_y[0]),
    y=str(local_y[1]), z=str(local_y[2]))
406.         if release == 'hinged':
407.             ET.SubElement(add_column_child, 'connectivity', m_x='true',
    m_y='true', m_z='true', r_x='true', r_y='false', r_z='false')
408.             ET.SubElement(add_column_child, 'connectivity', m_x='true',
    m_y='true', m_z='true', r_x='true', r_y='false', r_z='false')
409.         else:
410.             ET.SubElement(add_column_child, 'connectivity', m_x='true',
    m_y='true', m_z='true', r_x='true', r_y='true', r_z='true')
411.             ET.SubElement(add_column_child, 'connectivity', m_x='true',
    m_y='true', m_z='true', r_x='true', r_y='true', r_z='true')
412.         add_column_ecc = ET.SubElement(add_column_child,
    'eccentricity', use_default_physical_alignment='false')
413.         ET.SubElement(add_column_ecc, 'analytical', x=str(0),
    y=str(section_INFO[3].y), z=str(section_INFO[3].z))
414.         ET.SubElement(add_column_ecc, 'analytical', x=str(0),
    y=str(section_INFO[3].y), z=str(section_INFO[3].z))
415.         ET.SubElement(add_column_ecc, 'physical', x=str(0),
    y=str(section_INFO[3].y), z=str(section_INFO[3].z))
```

```
        ET.SubElement(add_column_ecc, 'physical', x=str(0), y=str(section_INFO[3].y), z=str(section_INFO[3].z))
        ET.SubElement(add_column_child, 'end')
        ET.SubElement(add_column, 'end')
        xml_bars.append(add_column)


    def addTruss(section_INFO, material_ID, point1, point2):
        barName = ID('truss')
        local_y = normal(point1, point2, 0)
        add_truss = ET.SubElement(xml_root.find('entities'), 'bar', name=barName, type='truss', guid=genGUID(), last_change=genUTC(), action='added')
        add_truss_child = ET.SubElement(add_truss, 'bar_part', guid=genGUID(), last_change=genUTC(), action='added', name=(barName + '.1'), complex_material=material_ID, complex_section=section_INFO[0])
        if section_INFO[2] != 'False':
            add_truss_child.attrib['made'] = section_INFO[2]
        add_truss_grandchild = ET.SubElement(add_truss_child, 'curve', type='line')
        ET.SubElement(add_truss_grandchild, 'point', x=str(point1.x), y=str(point1.y), z=str(point1.z))
        ET.SubElement(add_truss_grandchild, 'point', x=str(point2.x), y=str(point2.y), z=str(point2.z))
        ET.SubElement(add_truss_child, 'local-y', x=str(local_y[0]), y=str(local_y[1]), z=str(local_y[2]))
        ET.SubElement(add_truss_child, 'end')
        ET.SubElement(add_truss, 'end')
        xml_bars.append(add_truss)


    def addPlate(material_ID, thickness1, thickness2, release, point1, point2, alignment, meshSize):     #Modified
        shellName = ID('plate')
        add_plate = ET.SubElement(xml_root.find('entities'), 'slab', name=shellName, guid=genGUID(), last_change=genUTC(), action='added', type='plate')
        add_plate_child = ET.SubElement(add_plate, 'slab_part', guid=genGUID(), last_change=genUTC(), action='added', name=(shellName + '.1'),
        complex_material=material_ID, alignment=alignment, align_offset='0', ortho_alfa='0', ortho_ratio='1', mesh_size=meshSize)
        add_plate_grandchild = ET.SubElement(add_plate_child, 'contour')
        add_plate_greatgrandchild1 = ET.SubElement(add_plate_grandchild, 'edge', type='line')
        ET.SubElement(add_plate_greatgrandchild1, 'point', x=str(point1.x), y=str(point1.y), z=str(point1.z))
        ET.SubElement(add_plate_greatgrandchild1, 'point', x=str(point2.x), y=str(point1.y), z=str(point2.z))
        if release == 'hinged':
            edgeName = ID('edge')
            add_hinge = ET.SubElement(add_plate_greatgrandchild1, 'edge_connection', guid=genGUID(), last_change=genUTC(), action='added', name=edgeName, moving_local='true', joined_start_point='true', joined_end_point='true')
            add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
```

```
451.             ET.SubElement(add_hinge_child, 'motions', x_neg='10000000',
    x_pos='10000000', y_neg='10000000', y_pos='10000000', z_neg='10000000',
    z_pos='10000000')
452.             ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
    x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
453.         add_plate_greatgrandchild2 =
    ET.SubElement(add_plate_grandchild, 'edge', type='line')
454.         ET.SubElement(add_plate_greatgrandchild2, 'point',
    x=str(point2.x), y=str(point1.y), z=str(point2.z))
455.         ET.SubElement(add_plate_greatgrandchild2, 'point',
    x=str(point2.x), y=str(point2.y), z=str(point2.z))
456.         if release == 'hinged':
457.             edgeName = ID('edge')
458.             add_hinge = ET.SubElement(add_plate_greatgrandchild2,
    'edge_connection', guid=genGUID(), last_change=genUTC(), action='added',
    name=edgeName, moving_local='true', joined_start_point='true',
    joined_end_point='true')
459.             add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
460.             ET.SubElement(add_hinge_child, 'motions', x_neg='10000000',
    x_pos='10000000', y_neg='10000000', y_pos='10000000', z_neg='10000000',
    z_pos='10000000')
461.             ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
    x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
462.         add_plate_greatgrandchild3 =
    ET.SubElement(add_plate_grandchild, 'edge', type='line')
463.         ET.SubElement(add_plate_greatgrandchild3, 'point',
    x=str(point2.x), y=str(point2.y), z=str(point2.z))
464.         ET.SubElement(add_plate_greatgrandchild3, 'point',
    x=str(point1.x), y=str(point2.y), z=str(point1.z))
465.         if release == 'hinged':
466.             edgeName = ID('edge')
467.             add_hinge = ET.SubElement(add_plate_greatgrandchild3,
    'edge_connection', guid=genGUID(), last_change=genUTC(), action='added',
    name=edgeName, moving_local='true', joined_start_point='true',
    joined_end_point='true')
468.             add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
469.             ET.SubElement(add_hinge_child, 'motions', x_neg='10000000',
    x_pos='10000000', y_neg='10000000', y_pos='10000000', z_neg='10000000',
    z_pos='10000000')
470.             ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
    x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
471.         add_plate_greatgrandchild4 =
    ET.SubElement(add_plate_grandchild, 'edge', type='line')
472.         ET.SubElement(add_plate_greatgrandchild4, 'point',
    x=str(point1.x), y=str(point2.y), z=str(point1.z))
473.         ET.SubElement(add_plate_greatgrandchild4, 'point',
    x=str(point1.x), y=str(point1.y), z=str(point1.z))
474.         if release == 'hinged':
475.             edgeName = ID('edge')
476.             add_hinge = ET.SubElement(add_plate_greatgrandchild4,
    'edge_connection', guid=genGUID(), last_change=genUTC(), action='added',
    name=edgeName, moving_local='true', joined_start_point='true',
    joined_end_point='true')
477.             add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
478.             ET.SubElement(add_hinge_child, 'motions', x_neg='10000000',
    x_pos='10000000', y_neg='10000000', y_pos='10000000', z_neg='10000000',
    z_pos='10000000')
479.             ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
    x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
```

```
480.         ET.SubElement(add_plate_child, 'thickness', x='0',
   y=str(point1.y), z='0', val=str(thickness1))             #Modified
481.         ET.SubElement(add_plate_child, 'thickness', x='0',
   y=str(point2.y), z='0', val=str(thickness2))             #Modified
482.         ET.SubElement(add_plate_child, 'thickness', x=str(point2.x),
   y=str(point2.y), z=str(point2.z), val=str(thickness2))
483.         ET.SubElement(add_plate_child, 'local_pos', x=str(point1.x),
   y=str(point1.y), z=str(point1.z)) # The position of the local coordinate
   system should be any point inside the plate, in this case in the first
   point
484.         ET.SubElement(add_plate_child, 'local_x', x='1', y='0', z='0')
   # The direction of the local system is hard coded to be the same as the
   global coordinate system
485.         ET.SubElement(add_plate_child, 'local_y', x='0', y='1', z='0')
   # The direction of the local system is hard coded to be the same as the
   global coordinate system
486.         ET.SubElement(add_plate_child, 'end')
487.         ET.SubElement(add_plate, 'end')
488.         xml_shells.append(add_plate)
489.
490.
491.     def addPlateComplex(material_ID, thickness, release, *points,
   alignment ):
492.         shellName = ID('plate')
493.         add_plate = ET.SubElement(xml_root.find('entities'), 'slab',
   name=shellName, guid=genGUID(), last_change=genUTC(), action='added',
   type='plate')
494.         add_plate_child = ET.SubElement(add_plate, 'slab_part',
   guid=genGUID(), last_change=genUTC(), action='added', name=(shellName +
   '.1'), complex_material=material_ID, alignment=alignment,
   align_offset='0', ortho_alfa='0', ortho_ratio='1')
495.         add_plate_grandchild = ET.SubElement(add_plate_child,
   'contour')
496.         for j in range(len(points)):
497.             add_plate_greatgrandchild =
   ET.SubElement(add_plate_grandchild, 'edge', type='line')
498.             ET.SubElement(add_plate_greatgrandchild, 'point',
   x=str(points[j].x), y=str(points[j].y), z=str(points[j].z))
499.             if j == len(points)-1:
500.                 ET.SubElement(add_plate_greatgrandchild, 'point',
   x=str(points[0].x), y=str(points[0].y), z=str(points[0].z))
501.             elif j != len(points)-1:
502.                 ET.SubElement(add_plate_greatgrandchild, 'point',
   x=str(points[j+1].x), y=str(points[j+1].y), z=str(points[j+1].z))
503.             if release == 'hinged':
504.                 edgeName = ID('edge')
505.                 add_hinge = ET.SubElement(add_plate_greatgrandchild,
   'edge_connection', guid=genGUID(), last_change=genUTC(), action='added',
   name=edgeName, moving_local='true', joined_start_point='true',
   joined_end_point='true')
506.                 add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
507.                 ET.SubElement(add_hinge_child, 'motions',
   x_neg='10000000', x_pos='10000000', y_neg='10000000', y_pos='10000000',
   z_neg='10000000', z_pos='10000000')
508.                 ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
   x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
509.         ET.SubElement(add_plate_child, 'thickness', x='0', y='0',
   z='0', val=str(thickness))
```

```
510.         ET.SubElement(add_plate_child, 'local_pos', x=str(points[0].x),
   y=str(points[0].y), z=str(points[0].z)) # The position of the local
   coordinate system should be any point inside the plate, in this case in
   the first point
511.         ET.SubElement(add_plate_child, 'local_x', x='1', y='0', z='0')
   # The direction of the local system is hard coded to be the same as the
   global coordinate system
512.         ET.SubElement(add_plate_child, 'local_y', x='0', y='1', z='0')
   # The direction of the local system is hard coded to be the same as the
   global coordinate system
513.         ET.SubElement(add_plate_child, 'end')
514.         ET.SubElement(add_plate, 'end')
515.         xml_shells.append(add_plate)
516.
517.
518.     def addWall(material_ID, thickness, release, point1, point2):
519.         shellName = ID('wall')
520.         add_wall = ET.SubElement(xml_root.find('entities'), 'slab',
   name=shellName, guid=genGUID(), last_change=genUTC(), action='added',
   type='wall')
521.         add_wall_child = ET.SubElement(add_wall, 'slab_part',
   guid=genGUID(), last_change=genUTC(), action='added', name=(shellName +
   '.1'), complex_material=material_ID, alignment='center',
   align_offset='0', ortho_alfa='0', ortho_ratio='1')
522.         add_wall_grandchild = ET.SubElement(add_wall_child, 'contour')
523.         add_wall_greatgrandchild1 = ET.SubElement(add_wall_grandchild,
   'edge', type='line')
524.         ET.SubElement(add_wall_greatgrandchild1, 'point',
   x=str(point1.x), y=str(point1.y), z=str(point1.z))
525.         ET.SubElement(add_wall_greatgrandchild1, 'point',
   x=str(point2.x), y=str(point2.y), z=str(point1.z))
526.         if release == 'hinged':
527.             edgeName = ID('edge')
528.             add_hinge = ET.SubElement(add_wall_greatgrandchild1,
   'edge_connection', guid=genGUID(), last_change=genUTC(), action='added',
   name=edgeName, moving_local='true', joined_start_point='true',
   joined_end_point='true')
529.             add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
530.             ET.SubElement(add_hinge_child, 'motions', x_neg='10000000',
   x_pos='10000000', y_neg='10000000', y_pos='10000000', z_neg='10000000',
   z_pos='10000000')
531.             ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
   x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
532.         add_wall_greatgrandchild2 = ET.SubElement(add_wall_grandchild,
   'edge', type='line')
533.         ET.SubElement(add_wall_greatgrandchild2, 'point',
   x=str(point2.x), y=str(point2.y), z=str(point1.z))
534.         ET.SubElement(add_wall_greatgrandchild2, 'point',
   x=str(point2.x), y=str(point2.y), z=str(point2.z))
535.         if release == 'hinged':
536.             edgeName = ID('edge')
537.             add_hinge = ET.SubElement(add_wall_greatgrandchild2,
   'edge_connection', guid=genGUID(), last_change=genUTC(), action='added',
   name=edgeName, moving_local='true', joined_start_point='true',
   joined_end_point='true')
538.             add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
539.             ET.SubElement(add_hinge_child, 'motions', x_neg='10000000',
   x_pos='10000000', y_neg='10000000', y_pos='10000000', z_neg='10000000',
   z_pos='10000000')
```

```
540.             ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
   x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
541.         add_wall_greatgrandchild3 = ET.SubElement(add_wall_grandchild,
   'edge', type='line')
542.         ET.SubElement(add_wall_greatgrandchild3, 'point',
   x=str(point2.x), y=str(point2.y), z=str(point2.z))
543.         ET.SubElement(add_wall_greatgrandchild3, 'point',
   x=str(point1.x), y=str(point1.y), z=str(point2.z))
544.         if release == 'hinged':
545.             edgeName = ID('edge')
546.             add_hinge = ET.SubElement(add_wall_greatgrandchild3,
   'edge_connection', guid=genGUID(), last_change=genUTC(), action='added',
   name=edgeName, moving_local='true', joined_start_point='true',
   joined_end_point='true')
547.             add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
548.             ET.SubElement(add_hinge_child, 'motions', x_neg='10000000',
   x_pos='10000000', y_neg='10000000', y_pos='10000000', z_neg='10000000',
   z_pos='10000000')
549.             ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
   x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
550.         add_wall_greatgrandchild4 = ET.SubElement(add_wall_grandchild,
   'edge', type='line')
551.         ET.SubElement(add_wall_greatgrandchild4, 'point',
   x=str(point1.x), y=str(point1.y), z=str(point2.z))
552.         ET.SubElement(add_wall_greatgrandchild4, 'point',
   x=str(point1.x), y=str(point1.y), z=str(point1.z))
553.         if release == 'hinged':
554.             edgeName = ID('edge')
555.             add_hinge = ET.SubElement(add_wall_greatgrandchild4,
   'edge_connection', guid=genGUID(), last_change=genUTC(), action='added',
   name=edgeName, moving_local='true', joined_start_point='true',
   joined_end_point='true')
556.             add_hinge_child = ET.SubElement(add_hinge, 'rigidity')
557.             ET.SubElement(add_hinge_child, 'motions', x_neg='10000000',
   x_pos='10000000', y_neg='10000000', y_pos='10000000', z_neg='10000000',
   z_pos='10000000')
558.             ET.SubElement(add_hinge_child, 'rotations', x_neg='0',
   x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
559.         ET.SubElement(add_wall_child, 'thickness', x='0', y='0', z='0',
   val=str(thickness))
560.         ET.SubElement(add_wall_child, 'local_pos', x=str(point1.x),
   y=str(point1.y), z=str(point1.z)) # The position of the local coordinate
   system should be any point inside the plate, in this case in the first
   point
561.         local_y = normal(point2, point1, 0)
562.         ET.SubElement(add_wall_child, 'local_x', x=str(local_y[0]),
   y=str(local_y[1]), z=str(local_y[2]))
563.         ET.SubElement(add_wall_child, 'local_y', x=str(0), y=str(0),
   z=str(1))
564.         ET.SubElement(add_wall_child, 'end')
565.         ET.SubElement(add_wall, 'end')
566.         xml_shells.append(add_wall)
567.
568.
569.     def addPointSupport(point, type):
570.         supportName = ID('support')
571.         add_pointsupport_root = xml_root.find('entities')
```

```
572.         add_pointsupport = ET.SubElement(add_pointsupport_root[1],
    'point_support', guid=genGUID(), last_change=genUTC(), action='added',
    name=supportName)
573.         add_pointsupport_child = ET.SubElement(add_pointsupport,
    'group')
574.         ET.SubElement(add_pointsupport_child, 'local_x', x='1', y='0',
    z='0')
575.         ET.SubElement(add_pointsupport_child, 'local_y', x='0', y='1',
    z='0')
576.         add_pointsupport_grandchild =
    ET.SubElement(add_pointsupport_child, 'rigidity')
577.         if type == 'fixed':
578.             ET.SubElement(add_pointsupport_grandchild, 'motions',
    x_neg='10000000000', x_pos='10000000000', y_neg='10000000000',
    y_pos='10000000000', z_neg='10000000000', z_pos='10000000000')
579.             ET.SubElement(add_pointsupport_grandchild, 'rotations',
    x_neg='10000000000', x_pos='10000000000', y_neg='10000000000',
    y_pos='10000000000', z_neg='10000000000', z_pos='10000000000')
580.         elif type == 'xyhinged':
581.             ET.SubElement(add_pointsupport_grandchild, 'motions',
    x_neg='10000000000', x_pos='10000000000', y_neg='10000000000',
    y_pos='10000000000', z_neg='0', z_pos='0')
582.             ET.SubElement(add_pointsupport_grandchild, 'rotations',
    x_neg='0', x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
583.         elif type == 'zpinned':
584.             ET.SubElement(add_pointsupport_grandchild, 'motions',
    x_neg='0', x_pos='0', y_neg='0', y_pos='0', z_neg='10000000000',
    z_pos='10000000000')
585.             ET.SubElement(add_pointsupport_grandchild, 'rotations',
    x_neg='0', x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
586.         else:
587.             ET.SubElement(add_pointsupport_grandchild, 'motions',
    x_neg='10000000000', x_pos='10000000000', y_neg='10000000000',
    y_pos='10000000000', z_neg='10000000000', z_pos='10000000000')
588.             ET.SubElement(add_pointsupport_grandchild, 'rotations',
    x_neg='0', x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
589.         ET.SubElement(add_pointsupport, 'position', x=str(point.x),
    y=str(point.y), z=str(point.z))
590.         xml_pointsupports.append(add_pointsupport)
591.
592.
593.     def addLineSupport(point1, point2, type):
594.         supportName = ID('support')
595.         add_linesupport_root = xml_root.find('entities')
596.         add_linesupport = ET.SubElement(add_linesupport_root[1],
    'line_support', guid=genGUID(), last_change=genUTC(), action='added',
    name=supportName)
597.         add_linesupport_child = ET.SubElement(add_linesupport, 'group')
598.         local_y = normal(point1, point2, 0)
599.         ET.SubElement(add_linesupport_child, 'local_x', x='0', y='0',
    z='1')
600.         ET.SubElement(add_linesupport_child, 'local_y',
    x=str(local_y[0]), y=str(local_y[1]), z=str(local_y[2]))
601.         add_linesupport_grandchild =
    ET.SubElement(add_linesupport_child, 'rigidity')
602.         if type == 'fixed':
603.             ET.SubElement(add_linesupport_grandchild, 'motions',
    x_neg='10000000000', x_pos='10000000000', y_neg='10000000000',
    y_pos='10000000000', z_neg='10000000000', z_pos='10000000000')
```

```
604.             ET.SubElement(add_linesupport_grandchild, 'rotations',
    x_neg='10000000000', x_pos='10000000000', y_neg='10000000000',
    y_pos='10000000000', z_neg='10000000000', z_pos='10000000000')
605.         else:
606.             ET.SubElement(add_linesupport_grandchild, 'motions',
    x_neg='10000000000', x_pos='10000000000', y_neg='10000000000',
    y_pos='10000000000', z_neg='10000000000', z_pos='10000000000')
607.             ET.SubElement(add_linesupport_grandchild, 'rotations',
    x_neg='0', x_pos='0', y_neg='0', y_pos='0', z_neg='0', z_pos='0')
608.         add_linesupport_edge = ET.SubElement(add_linesupport, 'edge',
    type='line')
609.         ET.SubElement(add_linesupport_edge, 'point', x=str(point1.x),
    y=str(point1.y), z=str(point1.z))
610.         ET.SubElement(add_linesupport_edge, 'point', x=str(point2.x),
    y=str(point2.y), z=str(point2.z))
611.         ET.SubElement(add_linesupport_edge, 'normal',
    x=str(local_y[0]), y=str(local_y[1]), z=str(local_y[2]))
612.         ET.SubElement(add_linesupport, 'normal', x='0', y='0', z='1')
613.         xml_linesupports.append(add_linesupport)
614.
615.
616.     def addSurfaceSupport(point1, point2):
617.         supportName = ID('support')
618.         add_surfacesupport_root = xml_root.find('entities')
619.         add_surfacesupport = ET.SubElement(add_surfacesupport_root[1],
    'surface_support', guid=genGUID(), last_change=genUTC(), action='added',
    name=supportName)
620.         add_surfacesupport_child = ET.SubElement(add_surfacesupport,
    'region')
621.         add_surfacesupport_grandchild =
    ET.SubElement(add_surfacesupport_child, 'contour')
622.         add_surfacesupport_greatgrandchild1 =
    ET.SubElement(add_surfacesupport_grandchild, 'edge', type='line')
623.         ET.SubElement(add_surfacesupport_greatgrandchild1, 'point',
    x=str(point1.x), y=str(point1.y), z=str(point1.z))
624.         ET.SubElement(add_surfacesupport_greatgrandchild1, 'point',
    x=str(point2.x), y=str(point1.y), z=str(point2.z))
625.         add_surfacesupport_greatgrandchild2 =
    ET.SubElement(add_surfacesupport_grandchild, 'edge', type='line')
626.         ET.SubElement(add_surfacesupport_greatgrandchild2, 'point',
    x=str(point2.x), y=str(point1.y), z=str(point2.z))
627.         ET.SubElement(add_surfacesupport_greatgrandchild2, 'point',
    x=str(point2.x), y=str(point2.y), z=str(point2.z))
628.         add_surfacesupport_greatgrandchild3 =
    ET.SubElement(add_surfacesupport_grandchild, 'edge', type='line')
629.         ET.SubElement(add_surfacesupport_greatgrandchild3, 'point',
    x=str(point2.x), y=str(point2.y), z=str(point2.z))
630.         ET.SubElement(add_surfacesupport_greatgrandchild3, 'point',
    x=str(point1.x), y=str(point2.y), z=str(point1.z))
631.         add_surfacesupport_greatgrandchild4 =
    ET.SubElement(add_surfacesupport_grandchild, 'edge', type='line')
632.         ET.SubElement(add_surfacesupport_greatgrandchild4, 'point',
    x=str(point1.x), y=str(point2.y), z=str(point1.z))
633.         ET.SubElement(add_surfacesupport_greatgrandchild4, 'point',
    x=str(point1.x), y=str(point1.y), z=str(point1.z))
634.         add_surfacesupport_motion = ET.SubElement(add_surfacesupport,
    'rigidity')
```

```
635.         ET.SubElement(add_surfacesupport_motion, 'motions',
   x_neg='10000000000', x_pos='10000000000', y_neg='10000000000',
   y_pos='10000000000', z_neg='10000000000', z_pos='10000000000')
636.         xml_surfacesupports.append(add_surfacesupport)
637.
638.
639.     def addCover(point1, point2):
640.         coverName = ID('cover')
641.         add_cover_root = xml_root.find('entities')
642.         add_cover = ET.SubElement(add_cover_root[2], 'cover',
   guid=genGUID(), last_change=genUTC(), action='added', name=coverName)
643.         add_cover_child = ET.SubElement(add_cover, 'region')
644.         add_cover_grandchild = ET.SubElement(add_cover_child,
   'contour')
645.         add_cover_greatgrandchild1 =
   ET.SubElement(add_cover_grandchild, 'edge', type='line')
646.         ET.SubElement(add_cover_greatgrandchild1, 'point',
   x=str(point1.x), y=str(point1.y), z=str(point1.z))
647.         ET.SubElement(add_cover_greatgrandchild1, 'point',
   x=str(point2.x), y=str(point1.y), z=str(point2.z))
648.         add_cover_greatgrandchild2 =
   ET.SubElement(add_cover_grandchild, 'edge', type='line')
649.         ET.SubElement(add_cover_greatgrandchild2, 'point',
   x=str(point2.x), y=str(point1.y), z=str(point2.z))
650.         ET.SubElement(add_cover_greatgrandchild2, 'point',
   x=str(point2.x), y=str(point2.y), z=str(point2.z))
651.         add_cover_greatgrandchild3 =
   ET.SubElement(add_cover_grandchild, 'edge', type='line')
652.         ET.SubElement(add_cover_greatgrandchild3, 'point',
   x=str(point2.x), y=str(point2.y), z=str(point2.z))
653.         ET.SubElement(add_cover_greatgrandchild3, 'point',
   x=str(point1.x), y=str(point2.y), z=str(point1.z))
654.         add_cover_greatgrandchild4 =
   ET.SubElement(add_cover_grandchild, 'edge', type='line')
655.         ET.SubElement(add_cover_greatgrandchild4, 'point',
   x=str(point1.x), y=str(point2.y), z=str(point1.z))
656.         ET.SubElement(add_cover_greatgrandchild4, 'point',
   x=str(point1.x), y=str(point1.y), z=str(point1.z))
657.
658.
659.     def addPointLoad(force, direction, loadCase, point):
660.         add_pointload_root = xml_root.find('entities')
661.         add_pointload = ET.SubElement(add_pointload_root[0],
   'point_load', load_case=loadCase, guid=genGUID(), last_change=genUTC(),
   action='added', load_type='force')
662.         ET.SubElement(add_pointload, 'direction', x=str(direction.x),
   y=str(direction.y), z=str(direction.z))
663.         ET.SubElement(add_pointload, 'load', x=str(point.x),
   y=str(point.y), z=str(point.z), val=str(force))
664.         xml_pointloads.append(add_pointload)
665.
666.
667.     def addLineLoad(force, direction, loadCase, point1, point2):
668.         add_lineload_root = xml_root.find('entities')
669.         add_lineload = ET.SubElement(add_lineload_root[0], 'line_load',
   load_case=loadCase, guid=genGUID(), last_change=genUTC(),
   action='added', load_dir='constant', load_projection='false',
   load_type='force')
```

```
670.         add_lineload_child = ET.SubElement(add_lineload, 'edge',
   type='line')
671.         ET.SubElement(add_lineload_child, 'point', x=str(point1.x),
   y=str(point1.y), z=str(point1.z))
672.         ET.SubElement(add_lineload_child, 'point', x=str(point2.x),
   y=str(point2.y), z=str(point2.z))
673.         normalx = normal(point1, point2, 0)
674.         ET.SubElement(add_lineload_child, 'normal', x=str(normalx[0]),
   y=str(normalx[1]), z=str(normalx[2]))
675.         ET.SubElement(add_lineload, 'direction', x=str(direction.x),
   y=str(direction.y), z=str(direction.z))
676.         ET.SubElement(add_lineload, 'normal', x=str(normalx[0]),
   y=str(normalx[1]), z=str(normalx[2]))
677.         ET.SubElement(add_lineload, 'load', x=str(point1.x),
   y=str(point1.y), z=str(point1.z), val=str(force))
678.         ET.SubElement(add_lineload, 'load', x=str(point2.x),
   y=str(point2.y), z=str(point2.z), val=str(force))
679.         xml_lineloads.append(add_lineload)
680.
681.
682.     def addSurfaceLoad(force, direction, loadCase, point1, point2):
683.         add_surfaceload_root = xml_root.find('entities')
684.         add_surfaceload = ET.SubElement(add_surfaceload_root[0],
   'surface_load', load_case=loadCase, guid=genGUID(),
   last_change=genUTC(), action='added', load_projection='false',
   load_type='force')
685.         add_surfaceload_child = ET.SubElement(add_surfaceload,
   'region')
686.         add_surfaceload_grandchild =
   ET.SubElement(add_surfaceload_child, 'contour')
687.         add_surfaceload_greatgrandchild1 =
   ET.SubElement(add_surfaceload_grandchild, 'edge', type='line')
688.         ET.SubElement(add_surfaceload_greatgrandchild1, 'point',
   x=str(point1.x), y=str(point1.y), z=str(point1.z))
689.         ET.SubElement(add_surfaceload_greatgrandchild1, 'point',
   x=str(point2.x), y=str(point1.y), z=str(point2.z))
690.         add_surfaceload_greatgrandchild2 =
   ET.SubElement(add_surfaceload_grandchild, 'edge', type='line')
691.         ET.SubElement(add_surfaceload_greatgrandchild2, 'point',
   x=str(point2.x), y=str(point1.y), z=str(point2.z))
692.         ET.SubElement(add_surfaceload_greatgrandchild2, 'point',
   x=str(point2.x), y=str(point2.y), z=str(point2.z))
693.         add_surfaceload_greatgrandchild3 =
   ET.SubElement(add_surfaceload_grandchild, 'edge', type='line')
694.         ET.SubElement(add_surfaceload_greatgrandchild3, 'point',
   x=str(point2.x), y=str(point2.y), z=str(point2.z))
695.         ET.SubElement(add_surfaceload_greatgrandchild3, 'point',
   x=str(point1.x), y=str(point2.y), z=str(point1.z))
696.         add_surfaceload_greatgrandchild4 =
   ET.SubElement(add_surfaceload_grandchild, 'edge', type='line')
697.         ET.SubElement(add_surfaceload_greatgrandchild4, 'point',
   x=str(point1.x), y=str(point2.y), z=str(point1.z))
698.         ET.SubElement(add_surfaceload_greatgrandchild4, 'point',
   x=str(point1.x), y=str(point1.y), z=str(point1.z))
699.         ET.SubElement(add_surfaceload, 'direction', x=str(direction.x),
   y=str(direction.y), z=str(direction.z))
700.         ET.SubElement(add_surfaceload, 'load', x=str(point1.x),
   y=str(point1.y), z=str(point1.z), val=str(force))
701.         xml_surfaceloads.append(add_surfaceload)
```

```
702.
703.
704.     def addSurfaceLoadComplex(force, direction, loadCase, *points):
705.         add_surfaceload_root = xml_root.find('entities')
706.         add_surfaceload = ET.SubElement(add_surfaceload_root[0],
    'surface_load', load_case=loadCase, guid=genGUID(),
    last_change=genUTC(), action='added', load_projection='false',
    load_type='force')
707.         add_surfaceload_child = ET.SubElement(add_surfaceload,
    'region')
708.         add_surfaceload_grandchild =
    ET.SubElement(add_surfaceload_child, 'contour')
709.         for j in range(len(points)):
710.             add_surfaceload_greatgrandchild =
    ET.SubElement(add_surfaceload_grandchild, 'edge', type='line')
711.             ET.SubElement(add_surfaceload_greatgrandchild, 'point',
    x=str(points[j].x), y=str(points[j].y), z=str(points[j].z))
712.             if j == len(points)-1:
713.                 ET.SubElement(add_surfaceload_greatgrandchild, 'point',
    x=str(points[0].x), y=str(points[0].y), z=str(points[0].z))
714.             elif j != len(points)-1:
715.                 ET.SubElement(add_surfaceload_greatgrandchild, 'point',
    x=str(points[j+1].x), y=str(points[j+1].y), z=str(points[j+1].z))
716.         ET.SubElement(add_surfaceload, 'direction', x=str(direction.x),
    y=str(direction.y), z=str(direction.z))
717.         ET.SubElement(add_surfaceload, 'load', x=str(points[0].x),
    y=str(points[0].y), z=str(points[0].z), val=str(force))
718.         xml_surfaceloads.append(add_surfaceload)
719.
720.
721.     def addLoadCase(name, selfWeight):
722.         global xml_loadcases
723.         loadCaseID = genGUID()
724.         add_loadcase_root = xml_root.find('entities')
725.         add_loadcase = ET.SubElement(add_loadcase_root[0], 'load_case',
    duration_class='permanent', guid=loadCaseID, last_change=genUTC(),
    action='added', name=name)
726.         if selfWeight == True:
727.             add_loadcase.attrib['type'] = 'dead_load'
728.         else:
729.             add_loadcase.attrib['type'] = 'static'
730.         xml_loadcases.append(add_loadcase)
731.         return loadCaseID
732.
733.
734.     def addLoadComb(nameLoadComb, type, loadCaseList, gammaLoadCases):
735.         global xml_loadcombs, nrcomb
736.         nrcomb = nrcomb + 1
737.         add_loadcase_root = xml_root.find('entities')
738.         add_loadcomb = ET.SubElement(add_loadcase_root[0],
    'load_combination', guid=genGUID(), last_change=genUTC(),
    action='added', name=nameLoadComb)
739.         if type == 'U':
740.             add_loadcomb.attrib['type'] = 'ultimate_ordinary'
741.         if type == 'Ua':
742.             add_loadcomb.attrib['type'] = 'ultimate_accidental'
743.         if type == 'Us':
744.             add_loadcomb.attrib['type'] = 'ultimate_seismic'
745.         if type == 'Sq':
```

```
            add_loadcomb.attrib['type'] = 'serviceability_quasi_permanent'
        if type == 'Sf':
            add_loadcomb.attrib['type'] = 'serviceability_frequent'
        if type == 'Sc':
            add_loadcomb.attrib['type'] = 'serviceability_characteristic'
        for i_LC in range(len(loadCaseList)):
            ET.SubElement(add_loadcomb, 'load_case', guid=loadCaseList[i_LC], gamma=str(gammaLoadCases[i_LC]))
        xml_loadcombs.append(add_loadcomb)

```

DEPARTMENT OF XXXX XXXXXXXXXXX
XXXXXXXX XXXXXXXXXXX XX
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY