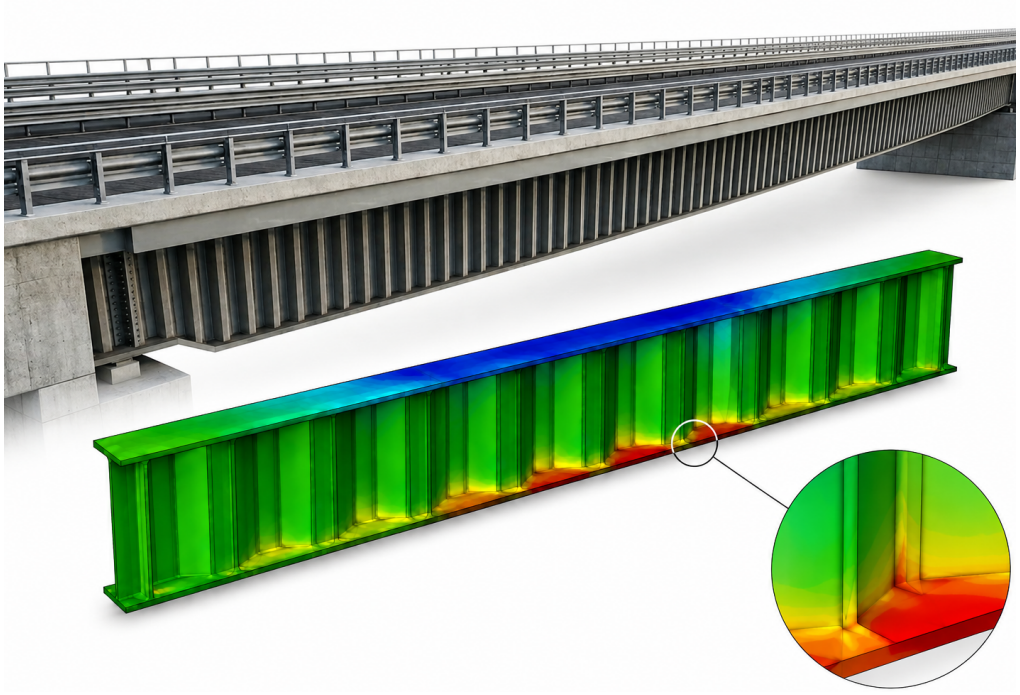
Fatigue Strength of Steel Beams with Corrugated Webs

Fatigue Assessment, Parametric Analysis and SCF Predictive Equation Development Using Effective Notch Stress Method



Ghaith Alharami

Mohammad Sami Raad

Master's Thesis in Structural Engineering and Building Technology
Department of Architecture and Civil Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

MASTER'S THESIS ACEX30

Fatigue Strength of Steel Beams with Corrugated Webs

Fatigue Assessment, Parametric Analysis and SCF Predictive Equation Development Using Effective Notch Stress Method



CHALMERS
UNIVERSITY OF TECHNOLOGY

Ghaith Alharami
Mohammad Sami Raad

Department of Architecture and Civil Engineering
Division of Structural Engineering–Light Weight Structures Group
Chalmers University of Technology, Gothenburg, Sweden 2026

Fatigue Strength of Steel Beams with Corrugated Web: Fatigue Assessment, Parametric Analysis, and SCF Predictive Equation Development Using Effective Notch Stress Method

© Ghaith Alharami, 2026.
© Mohammad Sami Raad, 2026.



Supervisor: PhD. Fatima Hlal, WSP
Examiner: Professor Mohammad Al-Emrani, Department of Architecture and Civil Engineering

Department of Architecture and Civil Engineering
Division of Structural Engineering
Light Weight Structures Group
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden

Abstract

Corrugated web girders (CWGs) have gained increasing attention in bridge engineering due to their high structural efficiency and favorable weight-to-strength ratio. However, fatigue assessment remains challenging because the corrugated geometry induces local stress concentrations that cannot be adequately captured using conventional nominal stress approaches. Moreover, no generalized formulation currently exists to relate nominal stresses to local fatigue-critical stresses while accounting for key corrugation parameters.

This thesis investigates the fatigue behavior of trapezoidal corrugated web girders (TCWGs) using the effective notch stress (ENS) approach. A detailed finite element (FE) model of a full corrugated web girder was developed and validated against four experimental tests. The validated model is subsequently simplified to a corrugated web beam representation (CWT: flange with corrugated web subjected to tension) while preserving an equivalent stress concentration factor (SCF). A comparison between the Hot Spot Stress and Effective Notch Stress methods was performed to identify the most suitable approach for further analyses. Based on this selection, an extensive parametric study covering a wide range of geometrical parameters was conducted, and the resulting numerical database was used to develop an equation for predicting SCF_{ENS}.

The results showed that the corrugation angle, θ, and the bend radius-to-web thickness ratio, R/t_w, are the governing parameters influencing SCF_{ENS}. Increasing R/t_w reduced stress concentrations, with a more pronounced effect at larger corrugation angles. Regression equations developed from the FE results showed good agreement with the numerical database, achieving R^2 values above 0.94. The proposed equations provide a practical tool for estimating effective notch stresses without refined FE analyses, supporting fatigue assessment and design optimization of trapezoidal corrugated web girders.

Keywords: Corrugated web girders, fatigue assessment, effective notch stress, structural hot-spot stress, stress concentration factor, finite element analysis, parametric study, regression model.

Sammanfattning

Korrugerade livbalkar (CWG) har fått ökad uppmärksamhet inom broteknik tack vare sin höga strukturella effektivitet och sitt fördelaktiga styrka-vikt-förhållande. Utmattningsbedömning är dock fortfarande en utmaning eftersom den korrugerade geometrin ger upphov till lokala spänningskoncentrationer som inte kan beskrivas tillräckligt väl med konventionella metoder baserade på nominella spänningar. Dessutom saknas en generell formulering som relaterar nominella spänningar till lokala utmattningskritiska spänningar med hänsyn till de viktigaste korrugeringsparametrarna.

Detta examensarbete undersöker utmattningsbeteendet hos trapetskorrugerade livbalkar (TCWG) med hjälp av Effective Notch Spänningar (ENS) metoden. En detaljerad finita elementmodell (FE-modell) av en fullständig korrugerad livbalk utvecklades och validerades mot fyra experimentella försök. Den validerade modellen förenklades därefter till en balkrepresentation med korrugerat liv (CWT) samtidigt som en ekvivalent spänningskoncentrationsfaktor (SCF) bibehölls. Därefter jämfördes metoderna Hot Spot Stress och Effective Notch Stress för att identifiera den mest lämpliga metoden för vidare analyser. En omfattande parametrisk studie genomfördes sedan, och den resulterande numeriska databasen användes för att utveckla en ekvation för att förutsäga SCF_{ENS}.

Resultaten visade att korrugeringsvinkeln, θ, och förhållandet mellan böjradie och livtjocklek, R/t_w, är de parametrar som har störst inverkan på $\mathrm{SCF}_{\mathrm{ENS}}$. En ökning av R/t_w minskade spänningskoncentrationerna, särskilt vid större korrugeringsvinklar. De regressionsmodeller som utvecklades från FE-resultaten visade god överensstämmelse med den numeriska databasen och uppnådde R^2-värden över 0.94. De föreslagna ekvationerna ger ett praktiskt verktyg för att uppskatta Effective Notch Spänningar utan detaljerade FE-analyser och kan därmed stödja utmattningsbedömning och dimensionering av trapetskorrugerade livbalkar.

Nyckelord: Korrugerade livbalkar, utmattningsbedömning, Effective Notch Spänning, hot-spot-spänning, spänningskoncentrationsfaktor, finita elementanalys, parametrisk studie, regressionsmodell.

Acknowledgements

This master's thesis was carried out at the Department of Architecture and Civil Engineering (ACE) at Chalmers University of Technology in collaboration with the Bridge Department at WSP in Gothenburg.

We would like to express our sincere gratitude to our supervisor, **Dr. Fatima Hlal**, for her continuous support, guidance, and encouragement throughout this project. Her expertise, availability, and valuable feedback have been essential to the successful completion of this thesis. We are especially grateful for the time she devoted to discussions, technical guidance, and constructive comments, which significantly improved the quality of this work. Working under her supervision has been both inspiring and highly educational.

We would also like to thank our examiner, **Professor Mohammad Al-Emrani**, for sharing his extensive knowledge and experience in structural engineering, and for his valuable comments and guidance throughout the thesis. His insightful feedback and academic perspective have contributed greatly to the development and quality of this work.

A special thank you is extended to our friends and fellow students at Chalmers University of Technology for their friendship, support, and encouragement during the past five years. The many shared experiences, challenges, and memorable moments have made our time at Chalmers both enjoyable and rewarding. We would also like to express our sincere appreciation to our opponents and friends, Mahmoud Alhamad and Ayel Osman, for their valuable feedback, constructive discussions, and insightful comments throughout the thesis process.

Finally, we would like to express our deepest gratitude to our families for their unwavering support, patience, and encouragement throughout our studies. Their belief in us has been invaluable and has made this achievement possible.

Ghaith Alharami Mohammad Sami Raad Gothenburg, June 2026

Contents

Abstract ii

Sammanfattning iii

Acknowledgements iv

Nomenclature xii

1 Introduction 1

1.1 Background and State of the Art . . . 1
1.2 Problem Statement . . . 2
1.3 Research Gap . . . 2
1.4 Aim and Objectives . . . 2
1.5 Methodology Overview . . . 3
1.6 Scope and Limitations . . . 4

2 Literature Review 5

2.1 Fundamentals of Fatigue in Steel Structures . . . 5
2.1.1 Introduction to Fatigue . . . 5
2.1.2 S–N Curves and Detail Categories . . . 6
2.2 Corrugated Web Girders in Structural Engineering . . . 7
2.2.1 Geometry and Governing Parameters . . . 7
2.2.2 Accordion Effect . . . 8
2.2.3 Transverse Bending Effect . . . 9
2.3 Fatigue Design Methods . . . 13
2.3.1 Nominal Stress Method . . . 13
2.3.2 Structural Hot-Spot Stress Method . . . 14
2.3.3 Effective Notch Stress Method . . . 17
2.3.4 Comparison Between Methods . . . 19
2.4 Stress Concentration Factor (SCF) . . . 20
2.4.1 Definition and Theoretical Background . . . 20
2.4.2 SCF Interpretation in SHSS . . . 20
2.4.3 SCF Interpretation in ENS . . . 21

2.5 Review of Previous Experimental Investigation . . . 21
2.5.1 Ibrahim et al. . . . 22
2.5.2 Kövesdi et al. . . . 22
2.5.3 Wang and Wang . . . 22
2.5.4 Tong et al. . . . 23
2.6 Review of Previous Numerical Studies . . . 23
2.6.1 Anami et al. . . . 23
2.6.2 Xu et al. . . . 25
2.6.3 Liu et al. . . . 26
2.6.4 Wang et al. . . . 27
2.6.5 Ibrahim et al. . . . 28
2.7 Summary of Literature Findings . . . 28
2.7.1 Influence of Governing Parameters on SCF . . . 30
2.7.2 Dominant vs Secondary Parameters . . . 31
2.7.3 Identified Research Gap . . . 31

3 Theoretical Framework **33**
3.1 Governing Geometric Parameters . . . 33
3.2 Selection of Weld Throat Thickness . . . 33
3.3 Rationale for Adopting the ENS Approach . . . 34

4 FE Modeling **35**
4.1 Global Finite Element Model and Validation . . . 36
4.1.1 Purpose of the Global Model . . . 36
4.1.2 Geometry and Reference Specimens . . . 37
4.1.3 Finite Element Modeling . . . 38
4.1.4 Mesh Sensitivity Analysis . . . 40
4.1.5 Global Model Validation . . . 41
4.2 Local Sub-model for SHSS Evaluation . . . 42
4.2.1 Sub-modeling Procedure . . . 42
4.2.2 Local mesh refinement . . . 43
4.2.3 Validation of Hot Spot Model . . . 44
4.3 Refined Sub-model for ENS Evaluation . . . 46
4.3.1 Sub-modeling Procedure . . . 46
4.3.2 The Result Derived from the ENS . . . 47
4.4 Discussion of the Multi-level FE Approach . . . 48
4.5 Chapter Summary . . . 50

5 Parametric Modeling strategy **54**
5.1 Transforming from CWG to CWT . . . 54

5.1.1 Selection of Nominal Stress Location . . . 55
5.1.2 Boundary Condition Investigation . . . 56
5.1.3 Selection of Web Height . . . 57
5.1.4 Validation against the Full Girder . . . 58
5.2 Parametric Study Configuration . . . 59
5.3 Automated Multi-level FE Workflow . . . 60
5.3.1 Database of Numerical Results . . . 61

6 Results of the Parametric Study 64
6.1 Effect of Corrugation Bend Radius-to-Web Thickness Ratio (R/t_w) . . . 64
6.2 Effect of Corrugation Angle (θ) . . . 65
6.3 Influence of Secondary Parameters . . . 67
6.3.1 Effect of the Depth Ratio ($\boldsymbol{a_3/b_f}$) and the Shape Ratio ($\boldsymbol{a_1/a_4}$) . 67
6.3.2 Effect of the Parallel-to-Inclined Fold Ratio ($\boldsymbol{a_1/a_2}$) . . . 67
6.4 Relative Importance and Parameter Interaction . . . 68
6.5 Development of a Predictive $\mathbf{SCF_{ENS}}$ Formula . . . 69

7 Discussion 72
7.1 Applicability of the Effective Notch Stress Method . . . 72
7.2 Influence of Geometric Parameters on Stress Concentration . . . 73
7.3 Evaluation of the Developed Prediction Models . . . 74

8 Conclusion 75
8.1 Recommendations for Future Research . . . 76

9 Statement on the Use of Artificial Intelligence 78

A Calculations for Validation of the FE Models i

B Additional S–N Curve Analyses iii
B.1 Nominal Stress Method . . . iii
B.2 Structural Hot-Spot Stress Method . . . v
B.3 Effective Notch Stress Method . . . vii

C Representative FE Models Used in the Parametric Study ix

D Python Script for Development of the $\mathrm{SCF_{ENS}}$ Prediction Equation. xii

E Results Obtained from Numerical Analysis cxvi
E.1 Parametric Study Results for $R/t_w = 2$ . . . cxvi
E.2 Parametric Study Results for $R/t_w = 6$ . . . cxxv
E.3 Parametric Study Results for $R/t_w = 10$ . . . cxxxii

List of Figures

1.1 Methodological framework adopted in the present thesis. 3

2.1 S–N curves corresponding to different fatigue detail categories (FAT classes). 6
2.2 Geometry and parameters of the corrugated web girder cross-section. . . 7
2.3 Geometric parameters of a trapezoidal corrugation profile and comparison with other corrugation shapes. 8
2.4 Transverse bending in flanges induced by shear flow in a corrugated web. 9
2.5 Transverse bending and stress decomposition in a corrugated web girder. 10
2.6 Refined analytical model for transverse bending in a corrugated web. . . 13
2.7 Nominal stress distribution under axial tensile loading. 14
2.8 Decomposition of total stress into nominal, bending, and notch stress components. 15
2.9 Fatigue crack initiation and hot-spot locations in a transverse attachment weld. 15
2.10 Extrapolation procedures for determining hot-spot stress at welded connections [3]. 16
2.11 Recommended S–N curves for the structural hot-spot stress method [2]. . 17
2.12 Application of the effective notch stress method with a reference notch radius of 1 mm. 18
2.13 Effective notch stress S–N curves for selected FAT classes [2]. 19
2.14 Influence of corrugation geometry on the hot-spot stress to nominal stress ratio [21]. 24
2.15 Influence of (left) corrugation angle and (right) bend radius on the stress concentration factor [23]. 26

4.1 Integrated finite element analysis workflow showing the global model analysis, hot-spot stress analysis, and effective notch stress evaluation procedure. 36
4.2 Geometry of the analysed corrugated web girders and corresponding geometric parameters [28]. 37
4.3 Overview of the experimental setup, loading system, and support conditions 38
4.4 Global FE–model and applied boundary conditions in Abaqus CAE. A value of “0” denotes a restrained degree of freedom, while “–” denotes a free degree of freedom. 39

4.5 Finite element model of the HSS member. 43
4.6 SHSS sub-model stress distribution for the two investigated meshes and corresponding mesh sensitivity results for the GA-1 girder. 44
4.7 Refined ENS finite element model and corresponding mesh sizes used in the analysis. 47
4.8 S–N curve for nominal stress method compared with the DC 90 design curve, mean regression curve, and characteristic curve. 51
4.9 S–N curve for the Hot-spot method compared with the DC 100 design curve, mean regression curve, and characteristic curve. 52
4.10 S–N curve for the effective notch stress method compared with the DC 225 design curve, mean regression curve, and characteristic curve. 53

5.1 Simplification of the global finite element model into the representative two-wave girder segment used in the parametric study. 55
5.2 Location of the nominal stress evaluation along the flange of the two-wave segment. The stress values (S_{11}) at several positions along the flange are shown to illustrate the non-uniform stress distribution caused by the corrugated web geometry. 56
5.3 Investigated boundary-condition configurations considered in the simplified tension specimen study based on the geometry of girder GA-1. 57
5.4 Investigated web heights considered in the simplified tension specimen study. 58
5.5 Overview of the automated workflow adopted in the parametric study. . . . 61
5.6 Overview of the numerical database generation process, including parameter selection, finite element analysis, geometry verification, quality-control filtering , and development of the final regression database for $\mathrm{SCF_{ENS}}$ prediction. 63

6.1 Effect of the bend radius-to-web thickness ratio (R/t_w) on SCF_{ENS}. . . . 65
6.2 Effect of the corrugation angle (θ) on $\mathrm{SCF_{ENS}}$. 65
6.3 Interaction effect between the corrugation angle θ and bend radius ratio R/t_w on $\mathrm{SCF_{ENS}}$. Increasing corrugation angle amplifies the sensitivity of the stress concentration factor to changes in bend radius. 66
6.4 Effect of the secondary geometric ratios a_3/b_f and a_1/a_4 on $\mathrm{SCF_{ENS}}$. The results indicate a comparatively smaller influence than that observed for the corrugation angle θ and bend radius ratio R/t_w. 67
6.5 Effect of the parallel-to-inclined fold ratio a_1/a_2 on $\mathrm{SCF_{ENS}}$ for different corrugation angles and bend radius ratios. 68
6.6 Relative influence of the investigated geometric parameters on $\mathrm{SCF_{ENS}}$, expressed as the mean parameter sensitivity ΔSCF. 69

6.7 Regression performance of the developed model for specimens with $R/t_w =$ 2, 6, and 10. The red line indicates perfect agreement between the finite element results and model predictions. 71

List of Tables

2.1 C-factor for different corrugation profiles (Abbas et al., 2007). 12
2.2 Summary of the influence of governing corrugation parameters on SCF based on previous studies. 29
2.3 Classification of governing parameters influencing stress concentration. . . 31

4.1 Fatigue test loading scheme for all specimen series [29]. 38
4.2 Mesh convergence study based on midspan deflection. 40
4.3 Comparison between analytical calculations and FE results for global validation. Applied load $P = 385\,\text{kN}$ for GA, GB, and GC, and $P = 415\,\text{kN}$ for GD. 41
4.4 Comparison of experimental and FE-derived hot-spot stresses using linear and quadratic extrapolation methods. Applied load $P = 385\,\text{kN}$ for GA, GB, and GC, and $P = 415\,\text{kN}$ for GD. 45
4.5 Hot-spot stress ranges- and stress concentration factors for the CWG specimens. N_f are obtained from the reference study [20]. 46
4.6 Effective notch stress ranges- and stress concentration factors for the CWG specimens. N_f are obtained from the reference study [20]. 48

5.1 Comparison between the simplified tension specimen and the full girder for structural hot-spot stress (SHSS) evaluation. 59
5.2 Comparison between the simplified tension specimen and the full girder for effective notch stress (ENS) evaluation. 59
5.3 Geometrical parameters and investigated ranges used in the parametric study. 60

6.1 Definitions of the model constant C and the λ_i terms used in Eq. (6.2). . 70

Nomenclature

Abbreviations

CBR	Constant Bending Region
CWG	Corrugated Web Girder
CWPG	Corrugated Web Plate Girder
CWT	Corrugated Web Tension specimen
ENS	Effective Notch Stress
FE	Finite Element
FEM	Finite Element Method
IIW	International Institute of Welding
NS	Nominal Stress
RMSE	Root Mean Square Error
SCF	Stress Concentration Factor
SCF_{ENS}	Effective Notch Stress Concentration Factor
SCF_{HS}	Hot-Spot Stress Concentration Factor
SHSS	Structural Hot-Spot Stress

Latin Symbols

a_1	Length of parallel fold [mm]
a_2	Length of inclined fold [mm]
a_3	Corrugation depth [mm]

a_4	Projection of inclined fold [mm]
b_f	Flange width [mm]
C	Regression constant [-]
E	Young's modulus [MPa]
h_w	Web height [mm]
N_f	Number of cycles to failure [-]
P	Applied load [kN]
R	Corrugation bend radius [mm]
R^2	Coefficient of determination [-]
t_f	Flange thickness [mm]
t_w	Web thickness [mm]

Dimensionless Geometric Ratios

a_1/a_2	Parallel-to-inclined fold ratio [-]
a_1/a_4	Shape ratio [-]
a_3/b_f	Depth ratio [-]
b_f/t_f	Flange slenderness ratio [-]
h_w/t_w	Web slenderness ratio [-]
R/t_w	Bend radius ratio [-]
t_f/t_w	Thickness ratio [-]

Stress Quantities

$\Delta\sigma_n$	Nominal stress range [MPa]
$\Delta\sigma_{HS}$	Hot-spot stress range [MPa]
ΔSCF	Parameter sensitivity measure [-]
σ_n	Nominal stress [MPa]

σ_{ENS}	Effective notch stress [MPa]
σ_{HS}	Structural hot-spot stress [MPa]

Greek Symbols

Δ	Difference or range operator
λ_i	Regression interaction term [-]
θ	Corrugation angle [°]

Chapter 1

Introduction

This chapter presents the background and motivation for the present study, together with an overview of the current state of research on fatigue assessment of corrugated web girders. The research gap addressed by the thesis is identified, followed by the aim and objectives of the work, a summary of the adopted methodology, and the scope and limitations of the investigation.

1.1 Background and State of the Art

Fatigue failure in steel welded constructions is primarily caused by stress peaks, and the stress peaks occur at the points where there is geometry change of the detail itself and change due to welding. The most common method used for fatigue design analysis in bridge constructions is the nominal stress method, as described in EN 1993-1-9 [1]. The method assumes clear and well-defined nominal stress, but the results are limited to specific details where the corrugated web girders (CWGs), flange-to-web welded detail is not covered.

For Complex details where it is difficult to define the nominal stress, engineers have proposed several local methods for calculating fatigue life in welded structures. One of these methods is the structural hot spot stress (SHSS) method, and another is the effective notch stress (ENS) method. Both methods are used for obtaining more accurate results in calculating fatigue life in structures where the stress state is more complex in its nature e.g. in corrugated web girders.

Corrugated web girders have unique mechanical properties; for example, there is the accordion effect and bending due to shear flow in the corrugated web. For all experiments and simulations, the S-point, where both inclined and flat folds meet, is identified as being critical for fatigue. Several studies have also indicated that parameters such as

the corrugation angle, θ, and bend radius influence the stress concentration behavior. However, most previous investigations considered only limited parameter ranges or varied one parameter at a time. The other parameters were found to be of secondary importance; however, this thesis investigate them in greater detail.

Although considerable research has been done, the research has usually focused on limited ranges of parameters, and usually only one parameter is investigated at a time. No single, unified relation has yet been developed that links nominal stress with stress indicators, including all relevant corrugation parameters.

1.2 Problem Statement

There is currently no generalized predictive formulation that relates nominal stress to local fatigue-critical stress measures in corrugated web girders while accounting for the governing geometric parameters.

1.3 Research Gap

Although considerable research has been conducted on the fatigue response of corrugated web girders, the current understanding remains largely based on case-specific investigations. These studies, both experimental and numerical, are limited in terms of the geometrical configurations considered. Although the influence of corrugation angle and bending radius in stress concentration has been established, no general formulation is available to relate nominal stresses with local stresses, considering the important geometrical parameters. Moreover, some geometrical parameters remain unstudied such as the web thickness, t_{w}. Furthermore, all results presented in previous studies are based on a limited number of experiments and therefore need to be expanded to ensure that the obtained results can be reliably applied in broader applications.

1.4 Aim and Objectives

The objective of this study is to investigate the influence of corrugation parameters on fatigue-induced stress concentrations in steel corrugated web girders. To establish a quantitative relationship between nominal stress and Effective Notch Stress, ENS, approaches using the stress concentration factor (SCF).

1.5 Methodology Overview

Figure 1.1 presents an overview of the methodology adopted in the present thesis. The research is conducted using a three-dimensional linear elastic finite element (FE) framework developed in Abaqus. A global model of the corrugated web girder is first established to determine nominal stress and overall stress distribution under bending-dominated loading. Sub-modeling techniques are then employed to generate a refined local model of the flange-to-web welded connection at the fatigue-critical S-point.

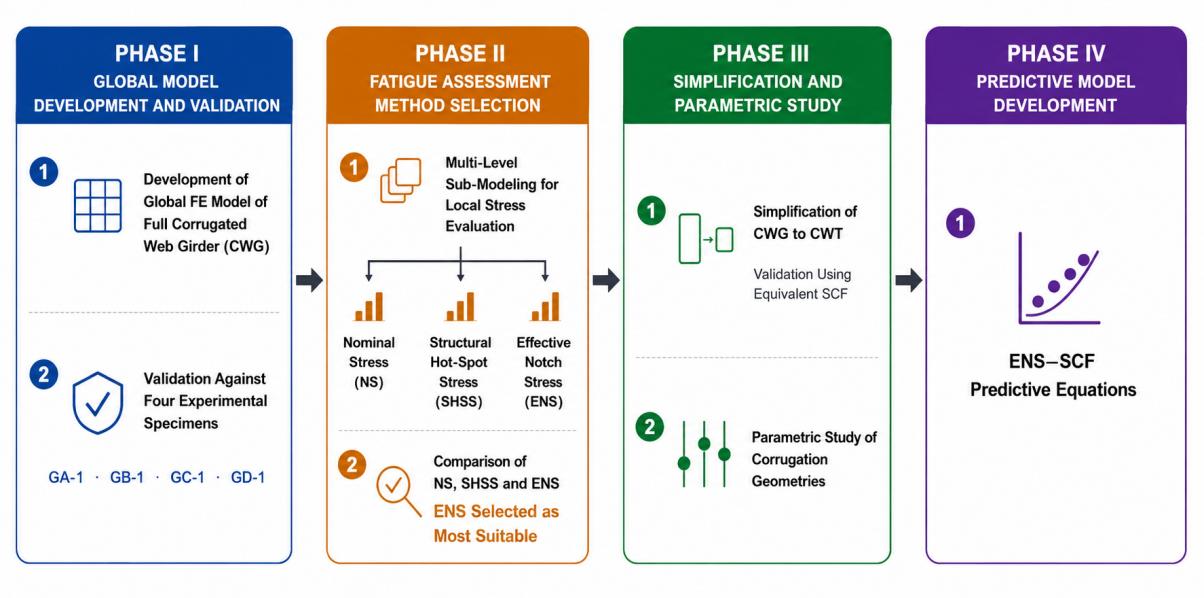


Figure 1.1: Methodological framework adopted in the present thesis.

Local stresses are determined in two methods: Structural Hot-Spot Stress (SHSS) and Effective Notch Stress (ENS). In the SHSS analysis, stresses are determined at specific offsets from the weld toe, after which the extrapolation of stresses is done according to IIW recommendations and the new EN1993-1-14 regulation [2], [3]. In ENS analysis, a virtual notch radius at the weld toe is created, depending on the plate thickness [2], after which the stress analysis is carried out using a very fine mesh to obtain accurate results of the stress gradient. Convergence studies are done to ensure the stability of the results.

In a systematic way, a parametric analysis is carried out by varying the main geometric parameters, such as the corrugation angle θ, the bend radius R, and a set of secondary parameters. For each set of parameters, numerical calculations are carried out for the determination of nominal stress, hot spot stress, and effective notch stress. The stresses are then used to obtain the stress concentration factors. The obtained results form the basis for establishing analytical relationships between nominal stresses and local stresses. A predictive formula is then developed based on the relationships derived. This allows

fatigue evaluation without resorting to detailed finite element modeling for each individual case.

1.6 Scope and Limitations

The limitations of the present thesis are mainly related to the applied geometric configurations. The results of this thesis work is limited to the corrugation dimensions included which cover wide range of beams and bridge girders of corrugated webs.

Moreover, the investigation is restricted to fatigue limit state considerations, assuming linear elastic material behavior at the global level. The study is further limited to fatigue failure in the form of weld toe cracking, while other potential failure modes are not considered.

Chapter 2

Literature Review

In this chapter, previous studies will be presented and evaluated based on the different parameters that could affect fatigue strength. Each study will be presented and summarized to be used later in the report. First, fatigue as a phenomenon will be presented, followed by a description of the nominal stress method, the structural hot-spot method, and the effective notch approach.

2.1 Fundamentals of Fatigue in Steel Structures

This chapter presents the fundamental principles of fatigue design assessment in steel structures. Emphasis is placed on an introduction to Fatigue, together with an introduction to the concept of SCF and application of S–N curves.

2.1.1 Introduction to Fatigue

It is a known fact that in any structure or machine, if the level of the maximum load acting on the structure towards failure exceeds the level of the constituted material yield limit, failure is assumed to have taken place in the structure. Nevertheless, it is noted that in a structure that experiences fluctuating stress levels, even when the stress is below the elastic material standard, failure could be noted in the structure as a result of Fatigue failure [4].

When designing steel girders in nowadays structures, static as well as dynamic forces have to be taken into consideration. Fatigue failures often occur without sufficient warning, potentially resulting in sever accidents and substantial economic consequences. Fatigue failure has been defined as a permanent local change in a material subjected to alternating stresses and strains that can cause cracks in the material. The fatigue cracks usually start in areas of structural details where stress concentration occurs. These higher stress levels, which usually cause fatigue failure, are normally brought about by stress concentrations

in the material that are caused by the disturbance of the stress flow because of the change in geometry, defects in welded details, and other stress raiser effects [4].

2.1.2 S–N Curves and Detail Categories

Fatigue properties for the components made of steel construction materials are presented in EN1993-1-9 [1] by the concept of fatigue resistance curves, also called Wöhler or S–N curves. Which describe the relationship between the stress range $\Delta\sigma$ and N_R, the number of cycles before failure.

These curves are derived from testing a representative sample that includes important real-world characteristics, such as geometric discontinuities, welding flaws, residual stresses, and fabrication tolerances. The test results are statistically analyzed to derive characteristic fatigue resistance values, which are defined at a 95% survival probability level. When plotted on a logarithmic scale, the stress range and cycle count form a linear relationship, which is the fatigue resistance curve. Each constructional detail is assigned to a Detail Category (DC), which represents the characteristic reference value of fatigue resistance, $\Delta\sigma_c$, defined at $N_c = 2 \cdot 10^6$ cycles as shown in Figure 2.1.

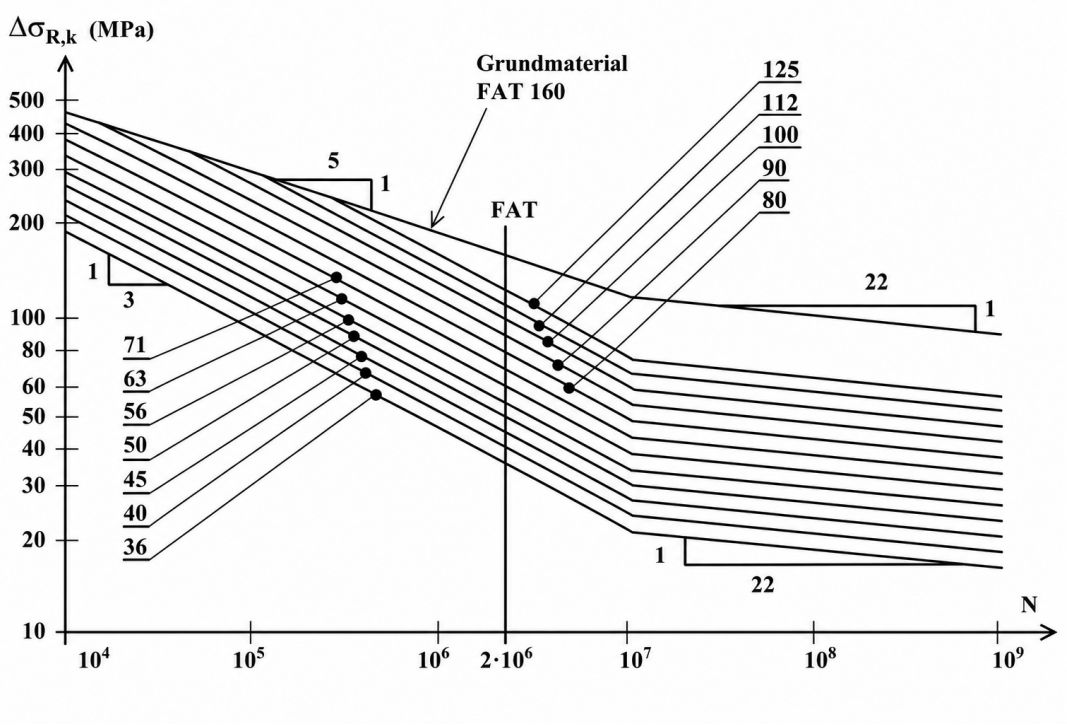


Figure 2.1: S–N curves corresponding to different fatigue detail categories (FAT classes).

The detail category therefore identifies a specific fatigue resistance curve. A lower DC value corresponds to lower fatigue resistance and results in a downward shift of the

curve in the S–N diagram. The fatigue resistance curve consists of several segments, and each segment corresponds to specific zones of the curve. Transitions are characterized by changes in the behavior of the material when subjected to fatigue, and they happen within specific cycles. In relation to the nominal stress technique being discussed, the choice of the fatigue resistance curve depends on the detail classification, taking into consideration the shape and weld design. It is very important that the chosen stress range is in line with the selected fatigue resistance curve.

2.2 Corrugated Web Girders in Structural Engineering

2.2.1 Geometry and Governing Parameters

The geometric parameters that define a corrugated web beam include the corrugation angle (θ), the bend radius (R), the lengths of the flat (a_1) and inclined folds (a_2), and the corrugation depth (a_3) as illustrated in Figure 2.2.

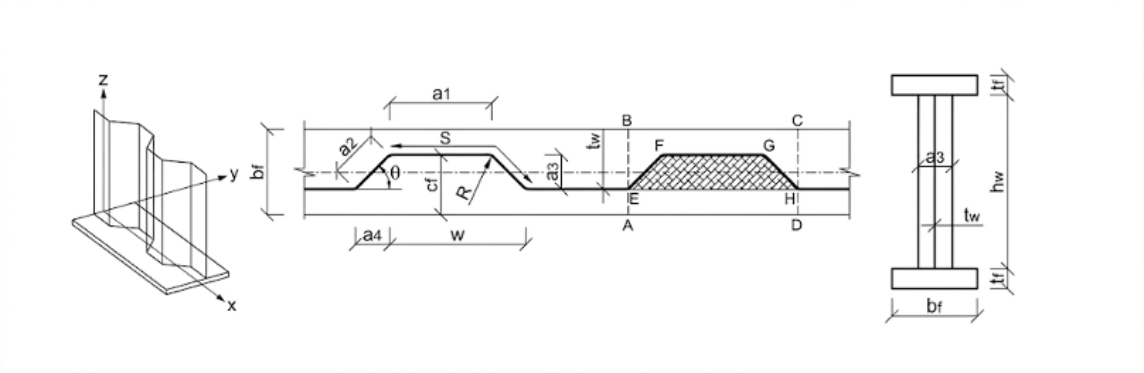


Figure 2.2: Geometry and parameters of the corrugated web girder cross-section.

The driving factor for the application of corrugated web girders is predominantly their increased shear resistance, along with the enhanced utilization of materials, compared to conventional flat web girders [5]. When deep girder formations are used, usually flat web girders demand stiffeners, either longitudinal or transversal to avoid shear buckling. Corrugated web girders possess inherent stiffness due to their geometric profile, thus avoiding the necessity for these additional stiffeners.

An experimental study conducted by Leblouba et al. [6] showed that a beam with a corrugated web can attain a shear strength 1.6 times higher than that of a beam with a flat web of equivalent dimensions. Furthermore, the geometry and profile of the corrugations contribute significantly to the structural efficiency and behavior of the structures under fatigue conditions. The geometrical difference between corrugations will alter the stress

distribution between the web and the flanges, thus influencing the stress concentration in the structure. There are various profiles that can be used in the creation of corrugations; examples include the rectangular profile, the trapezoidal profile, the sinusoidal profile, and the triangular profile as shown in Figure 2.3.

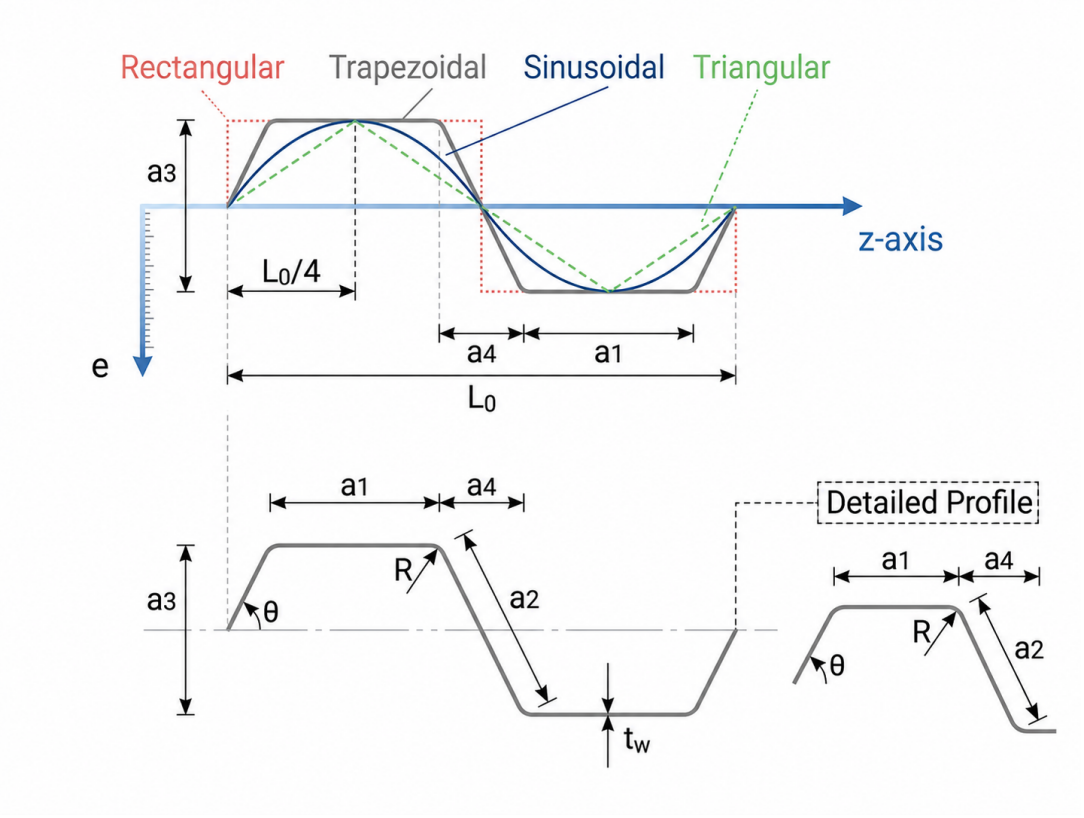


Figure 2.3: Geometric parameters of a trapezoidal corrugation profile and comparison with other corrugation shapes.

Trapezoidal corrugations are commonly used for bridges due to their superior shear stiffness and feasibility of manufacture. Sinusoidal corrugations have smoother variations in geometry and thus can reduce stress concentration at local points, whereas the rectangular and triangular profiles have sharper variations in geometry and therefore higher stress concentration.

2.2.2 Accordion Effect

The behavior of corrugated web girders has been observed to exhibit the phenomenon of the Accordion effect. Due to the lower rigidity of the corrugated web in the axis direction, allowing stretching and contracting behavior. Hence, the web cannot be considered as a contributing factor to bending behavior. Inaam et al. [7] devised a web participation factor, which has been used in highlighting the impact of corrugated webs with reference to flexural strength. The findings of that study revealed that it is not appropriate to fully negate the impact of the web while considering beams with reference to bending.

2.2.3 Transverse Bending Effect

In CW beams, the shear flow between the web and the flange follows the corrugation profile, inducing transverse bending within the plane of the flanges, see Figure 2.4. The torsional effects from this shear flow are therefore induced throughout the section. This is mainly due to the eccentricity relative to the longitudinal x-axis, parameter a_3 in Figure 2.2, leading to additional transverse bending moments around the flange's strong axis.

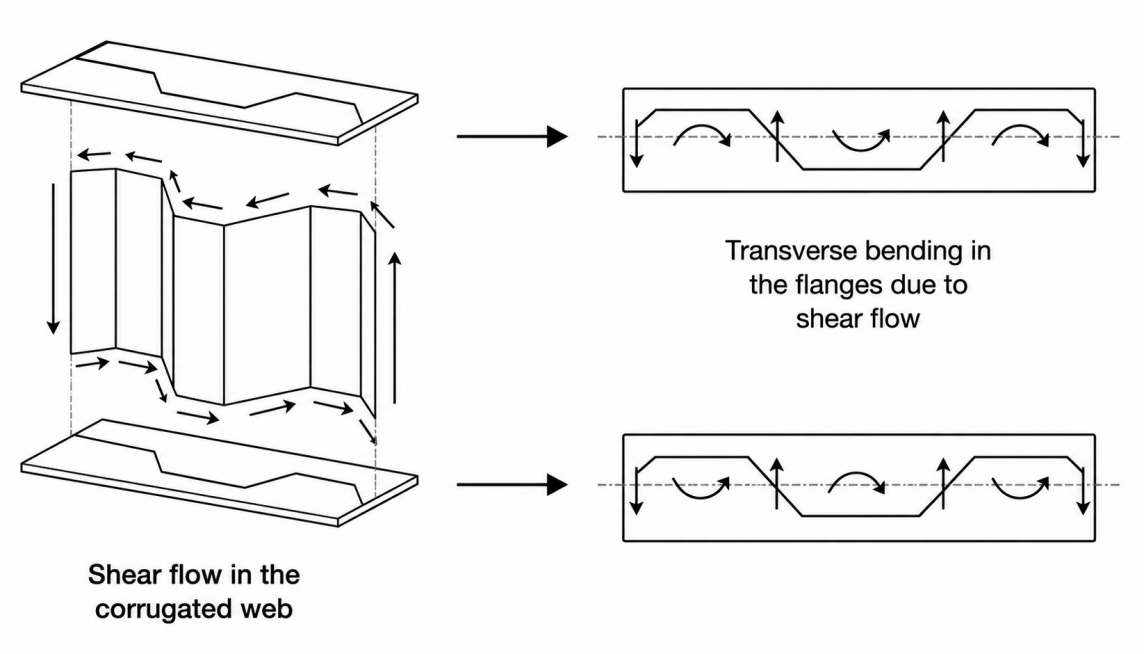


Figure 2.4: Transverse bending in flanges induced by shear flow in a corrugated web.

As for the corrugated web girders, various states of stress can occur, especially for the flange. In addition to nominal stresses resulting from general bending, stresses resulting from the transverse effects of the above-mentioned shear flow occurs. Which is shared by the corrugated web, due to the asymmetric position of the folds with respect to the axis of the flange.

On the other hand, results of experimental girders and calculations performed with the finite element method show that transverse bending can be extended into the range of nominal pure bending [8]. That is the case when loads are applied, or supports are introduced in the inclined folds, and a non-integer number of waves is contained within the shear span. The addition of transverse bending will lead to more congruent values of the concentration of stress. Furthermore, if transverse bending stresses can be conservatively estimated using analytical or simplified design models, they should be included in the fatigue assessment. In such cases, the nominal stress may be defined as the sum of the

global bending stress and the transverse bending contribution, as illustrated in Figure 2.5.

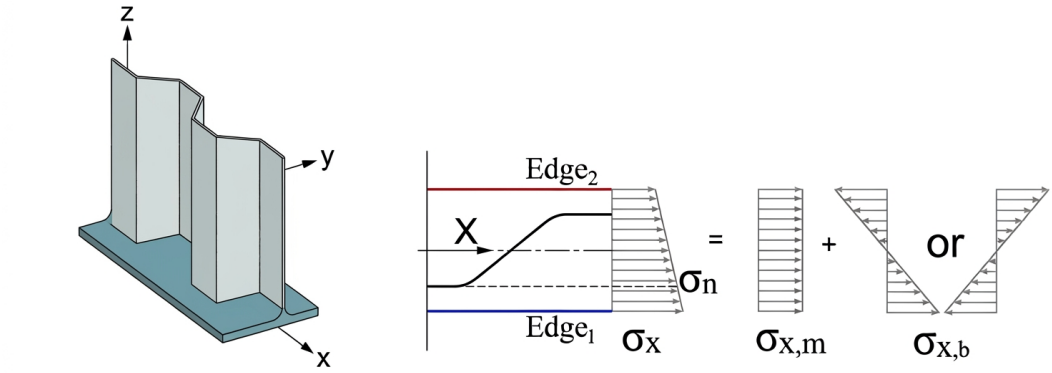


Figure 2.5: Transverse bending and stress decomposition in a corrugated web girder.

Fictitious Load Method & C-Factor Method

Abbas et al. [9] developed the fictitious load method to find the effects of transverse bending deformations in corrugated web girders. This method includes an equivalent transverse load on the flange. This transverse shear and transverse bending of the flange satisfy the effect of the shear flows in the corrugated web. This method was confirmed to fit well with experimental results in the evaluation of transverse bending effects on the flange.

By applying the method, the analysis of the behavior of both flanges is conducted using conventional beam theory. This theory helps in the global in-plane analysis of the beams, including the effect of twisting as transverse bending of the flange. To calculate the transverse bending moment in the case of CWG with a trapezoidal corrugated profile, the equivalent sinusoidal profile is considered. Equation 2.1 shows an analytical closed form solution for the case of a uniformly distributed load.

$$M_t^{\sin} = \frac{p_y \cdot L_0^2 \cdot a_3}{2 \cdot h_w} \left\{ \frac{1}{\Pi^2} \left[\left(1 - 2 \cdot \frac{z}{L_0} \right) \Pi \cdot c_\xi + 2 \cdot s_\xi + (\Pi \cdot c_\Pi - 2 \cdot s_\Pi + \Pi) \frac{z}{L_0} - \Pi \right] \right\} \tag{2.1}$$

where:

$$\Pi = 2 \cdot \pi \cdot n$$
$$c_\xi = \cos\left(2 \cdot \pi \cdot n \cdot \frac{z}{L_0}\right)$$
$$c_\Pi = \cos\left(2 \cdot \pi \cdot n\right)$$
$$s_\xi = \sin\left(2 \cdot \pi \cdot n \cdot \frac{z}{L_0}\right)$$
$$s_\Pi = \sin\left(2 \cdot \pi \cdot n\right)$$

p_y	Uniformly distributed in-plane load
L_0	Girder span
n	Number of corrugation waves
z	Position along the girder length

Abbas et al. [9] proposed a simplified analytical method, hereafter called the C-factor method, for an approximate evaluation of the transverse bending moment for corrugation shapes other than the reference case. This method couples the principle of virtual work with the fictitious load method. In this approach, the flange transverse bending moment is connected to the geometry of the corrugated web through a built-up area function. A non-dimensional coefficient, which characterizes the corrugation profile and is referred to as the C-factor. The C-factor is determined as the ratio between the area under one half-wave of the actual corrugation profile and that of an equivalent sinusoidal profile possessing the same wavelength and corrugation depth. The obtained C-factor enables the transformation of the transverse bending moment from the sinusoidal reference case to other corrugation geometries, see equation 2.2.

$$M_t^{\text{corr}} = C \cdot M_t^{\text{sin}} \tag{2.2}$$

The C-factor for different web corrugation profiles is given according to [9].

Table 2.1: C-factor for different corrugation profiles (Abbas et al., 2007).

Corrugation	Area under half wavelength $\int_0^{L_0} e\,dz$	C-factor
Sinusoidal	$\frac{a_3 L_0}{2\pi}$	1
Trapezoidal	$\frac{a_3}{2}\left(a_1 + \frac{a_4}{2}\right)$	$\pi\left(\frac{a_1 + a_4/2}{L_0}\right)$
Triangular	$\frac{a_3 L_0}{8}$	$\frac{\pi}{4}$
Rectangular	$\frac{a_3 L_0}{4}$	$\frac{\pi}{2}$

The transverse bending moment originates from the shear flow carried by the corrugated web and reaches its maximum in the inclined folds, where the eccentricity of the shear flow induces the highest transverse bending effects in the flange.

Refined Methods

In the numerical calculation provided by Kövesdi et.al. [10], a result comparison with the previous Eurocode design method EN 1993-1-5 [11] was conducted. The aim is to develop a more refined method for determining the transverse bending moment produced by shear flow in corrugated webs. In their method, the flange is considered similar to a beam that is subject to axial force, shear, and bending effects due to shear flow in the web. For satisfying global equilibrium conditions, it is assumed that the corrugated web has a starting and ending inclination, along with a whole number of corrugation wavelengths between these two sections.

Four different girder configurations were considered in the study, along with changes in their geometry, support, and loading conditions. In all cases, the highest response was recorded when both the support and applied load were placed at the center of an inclined fold, along with a full number of corrugation waves between these points. It has been noted that in the worst case, the transverse bending moment can be calculated as equation 2.3 [10].

$$M_{z,\max} = \frac{V_z\, a_3(2a_1 + a_4)}{2h_w} \tag{2.3}$$

The maximum transverse bending moment occurs at the location where the inclined web fold intersects the z-axis. The corresponding value, $M_{z,\max}$, is given by Equation 2.3 and is illustrated in Figure 2.6.

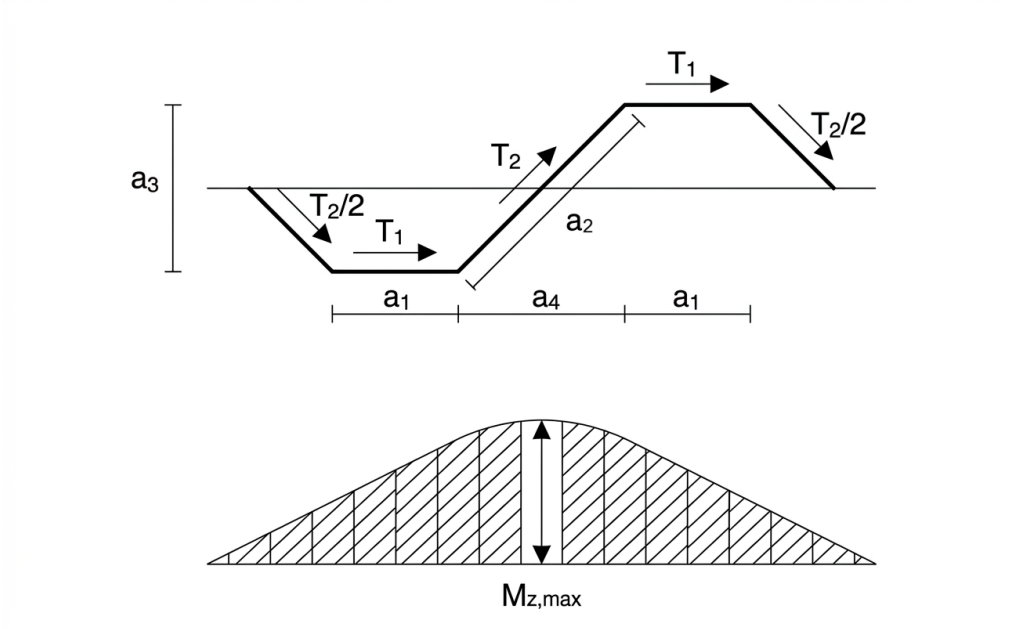


Figure 2.6: Refined analytical model for transverse bending in a corrugated web.

2.3 Fatigue Design Methods

The three methods for fatigue assessment of welded steel details are the nominal stress method and the two local methods, i.e., the structural hot-spot stress method (SHSS method) and the effective notch stress method (ENS method). The fundamental objective of all fatigue design methods is to find the stress ranges, and the fatigue life of a critical section by using an appropriate S-N curve. The major difference between the methods lies in their ability to capture the local effects.

2.3.1 Nominal Stress Method

The nominal stress method is considered the most widely used approach for fatigue assessment of welded steel structures and forms the basis of most design codes. In this method, the fatigue analysis is performed using the nominal stress derived from the global forces and moments acting on the structural member. According to [4], the nominal stress is defined as "the sum of the membrane stress and the bending stress in the section, assuming a uniform stress distribution."

In general, the calculation of the nominal stress is performed using conventional beam theory, neglecting the effects of the local geometry. Therefore, the effects of stress concentrations due to welds, notches, etc., are not considered. The assumption of a uniform stress distribution is shown in Figure 2.7, where a plate is subjected to a uniform tensile stress. A uniform stress distribution is achieved except near the weld. In the

context of the nominal stress method, the effects of stress concentrations are considered indirectly through the selection of an appropriate detail category. Each structural detail is associated with a specific S-N curve, which is derived from the test data. According to EN 1993-1-9 [1], the nominal stress range is evaluated at a location sufficiently remote from the welded joint to exclude the effects of local stress concentrations. The fatigue life is then determined using the appropriate S–N curve corresponding to the detail category.

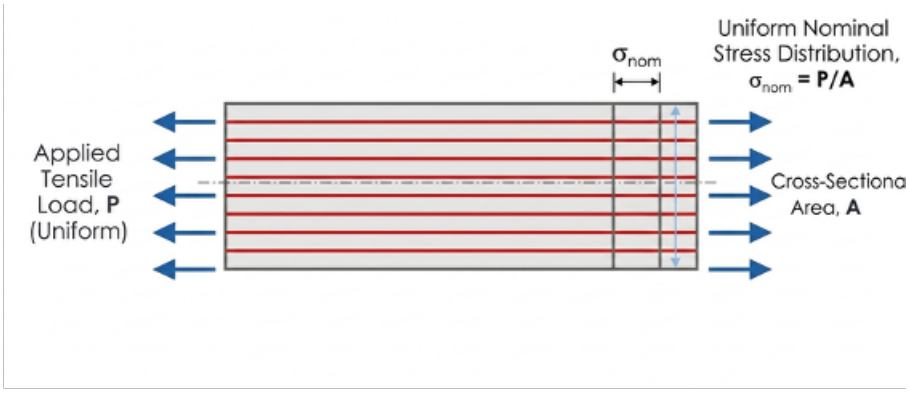


Figure 2.7: Nominal stress distribution under axial tensile loading.

The nominal stress method is effective and established for structural details that have a well-defined geometry. However, the applicability of the nominal stress method is restricted in the analysis of complex welded joints, as the concept of nominal stress is unclear.

2.3.2 Structural Hot-Spot Stress Method

The structural hot-spot stress method (SHSS) can be viewed as a representative local stress-based approach for fatigue strength assessment of welded details. This approach is often used for structural components with complex geometries for which the nominal stress cannot be defined properly [12].

In the vicinity of a welded joint, the stress distribution is non-linear with a high peak at the weld toe, as illustrated by Figure 2.8. It can be divided into three components: nominal membrane stresses (σ_{nom}), bending stresses due to structural geometry (σ_{bend}), and non-linear notch stresses associated with the geometry of the welded joint (σ_{notch}). While the nominal and bending stresses describe the structural contribution to the total stress distribution, the non-linear notch stress accounts for the effects of the geometry of the welded joint.

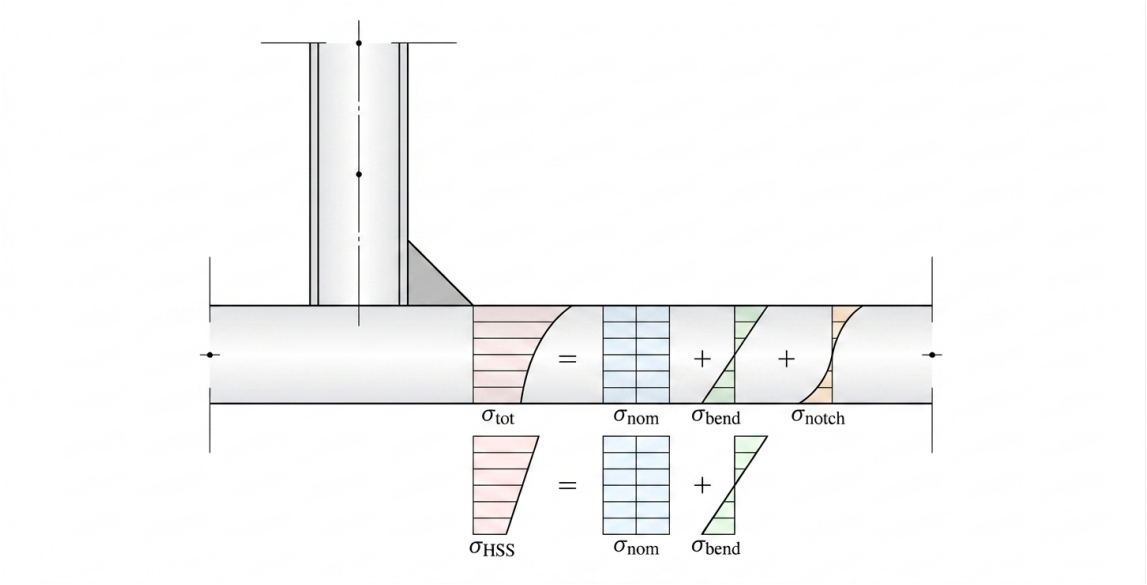


Figure 2.8: Decomposition of total stress into nominal, bending, and notch stress components.

Depending on the stress variation through the plate thickness, two different hot-spot configurations can be identified, termed Type (a) and Type (b) hot spots, as presented in Figure 2.9. In the case of Type (a) hot spot, there is a variation in stress through the plate thickness, with significant bending effects. In contrast, for Type (b) hot spots, the stress variation through the plate thickness is relatively uniform. This categorization is related to both the evaluation procedure and the selection of reference points used in hot-spot stress evaluation [2].

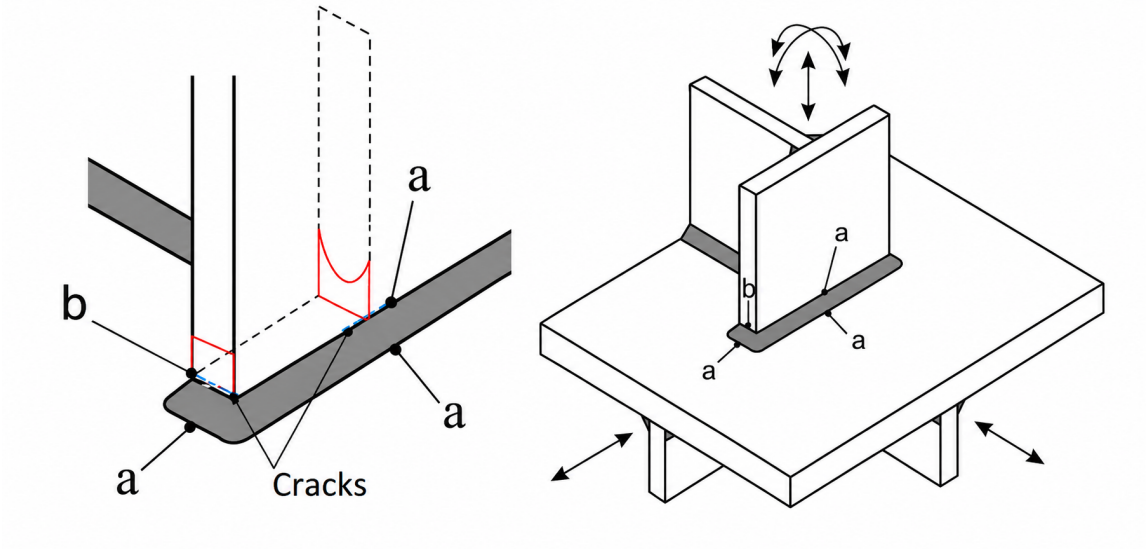


Figure 2.9: Fatigue crack initiation and hot-spot locations in a transverse attachment weld.

In the SHSS method, the non-linear notch stress term is removed from the fatigue assessment procedure. Instead, the structural hot-spot stress is determined by linear or

quadratic extrapolation from the stresses calculated at reference points located at a certain distance from the weld toe, as illustrated in Figure 2.10 [2]. This approach excludes the non-linear notch stress term due to the difficulty in accurately defining the geometry of the welded joint during the design process [13]. For Type (a) hot spot, the reference points recommended by EN 1993-1-14 [3] depend on the selected extrapolation procedure. In the case of fine-mesh finite element models, the structural hot-spot stress can be calculated using linear extrapolation according to Eq. 2.4.

$$\sigma_{HS} = 1.67\,\sigma_{0.4t} - 0.67\,\sigma_{1.0t} \tag{2.4}$$

where $\sigma_{0.4t}$ and $\sigma_{1.0t}$ are surface stresses evaluated at distances of $0.4t$ and $1.0t$ from the weld toe, respectively. For a more accurate representation of non-linear stress distributions, quadratic extrapolation may be adopted according to Eq. 2.5.

$$\sigma_{HS} = 2.52\,\sigma_{0.4t} - 2.24\,\sigma_{0.9t} + 0.72\,\sigma_{1.4t} \tag{2.5}$$

where surface stresses are extracted at distances of $0.4t$, $0.9t$, and $1.4t$ from the weld toe.

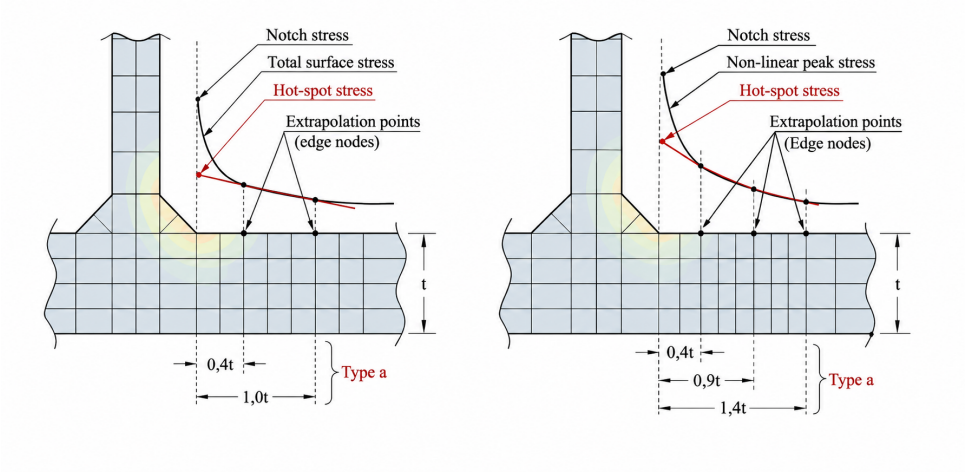


Figure 2.10: Extrapolation procedures for determining hot-spot stress at welded connections [3].

The structural hot-spot stress (SHSS) method is considered suitable for fatigue assessment using finite element analysis, as it accounts for the effects of structural geometry. Compared to the nominal stress method, the SHSS method offers improved applicability, as it requires fewer S–N curves, with the influence of macro-geometry incorporated into the evaluation. The corresponding S–N curves recommended for the structural hot-spot stress method are presented in Figure 2.11.

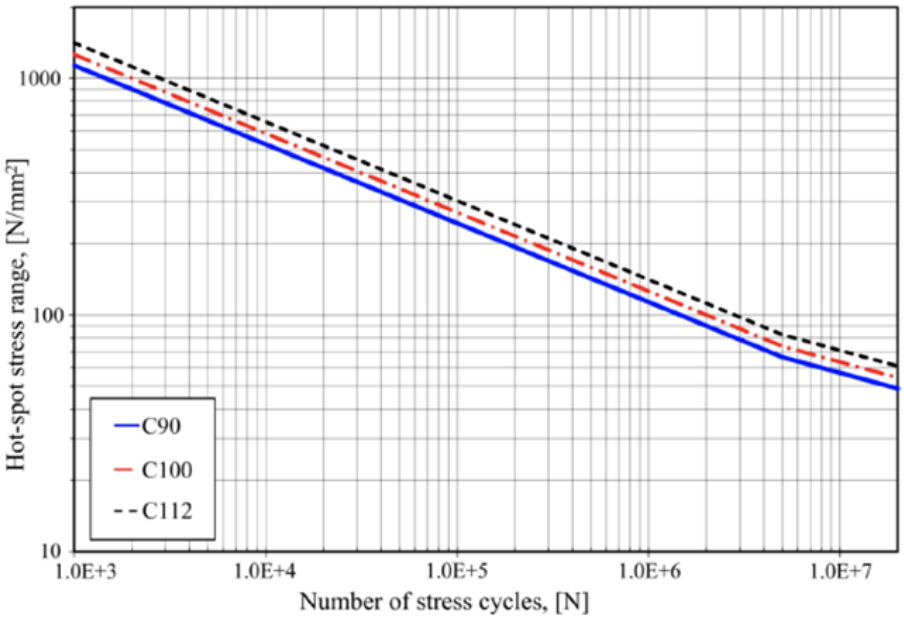


Figure 2.11: Recommended S–N curves for the structural hot-spot stress method [2].

However, it should be noted that stresses arising from weld geometry and possible weld imperfections are not considered, which can lead to deviations from actual fatigue behavior [12]. The effective notch stress method can also be used for a more refined stress assessment.

2.3.3 Effective Notch Stress Method

The Effective Notch Stress (ENS) method is a local fatigue approach that considers stress concentrations caused by global structural geometry and local weld geometry. In the ENS method, fatigue analysis is performed by considering the elastic stress value at the weld toe or root, which are the points most likely to initiate fatigue cracks. In comparison to the nominal stress method and the structural hot-spot stress method, the fatigue stress analysis performed by the ENS method is more detailed [2].

Local stress raisers are caused by geometrical discontinuities, which are changes in geometry, notches, and welds. The notch stress is very sensitive to changes in the sharpness of the weld. In fact, theoretically, an infinitely sharp notch can create an infinite stress value. To overcome this problem, the ENS method uses a standard notch radius that replaces the actual weld geometry and produces a specific stress value.

Consistent with the recommendations of the International Institute of Welding (IIW), a reference notch radius of 1 mm is normally adopted for welded steel structures where the thickness of the plates exceeds 5 mm, as shown in Figure 2.12 [14]. By adopting a

constant reference radius, the effect of changes in weld geometry and local imperfections is reduced, and the fatigue strength can be assessed for a range of welded details using one specific S–N curve.

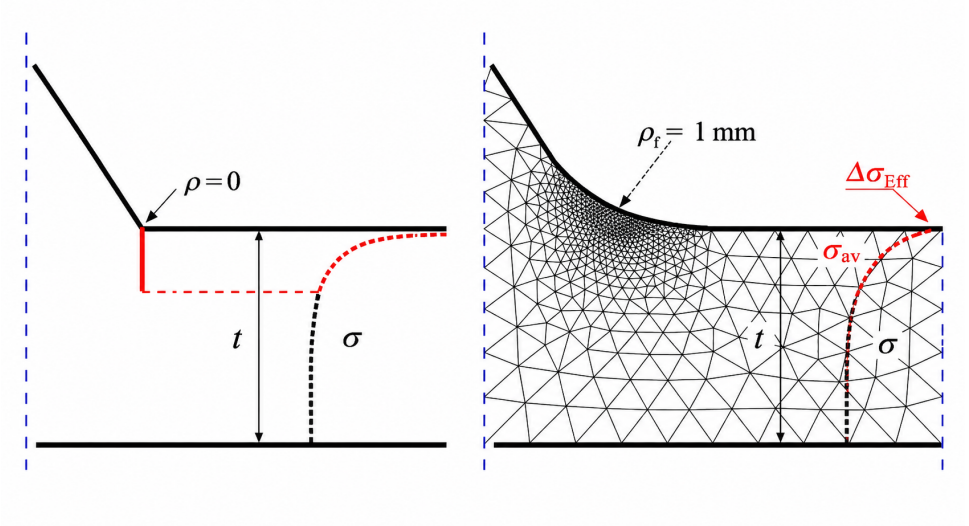


Figure 2.12: Application of the effective notch stress method with a reference notch radius of 1 mm.

The ENS is normally computed using a finite element analysis with a refined mesh around the weld toe or root. In accordance with the applied fatigue calculation approach and loading conditions, the maximum principal stress and von Mises stress may be employed. The IIW recommendations provide different S–N curves depending on the reference notch radius and the selected stress definition [2]. For a notch radius of 1 mm, the recommended FAT classes are FAT225 for principal stress and FAT200 for von Mises stress, whereas a notch radius of 0.05 mm corresponds to FAT630 and FAT560, respectively, as shown in Figure 2.13.

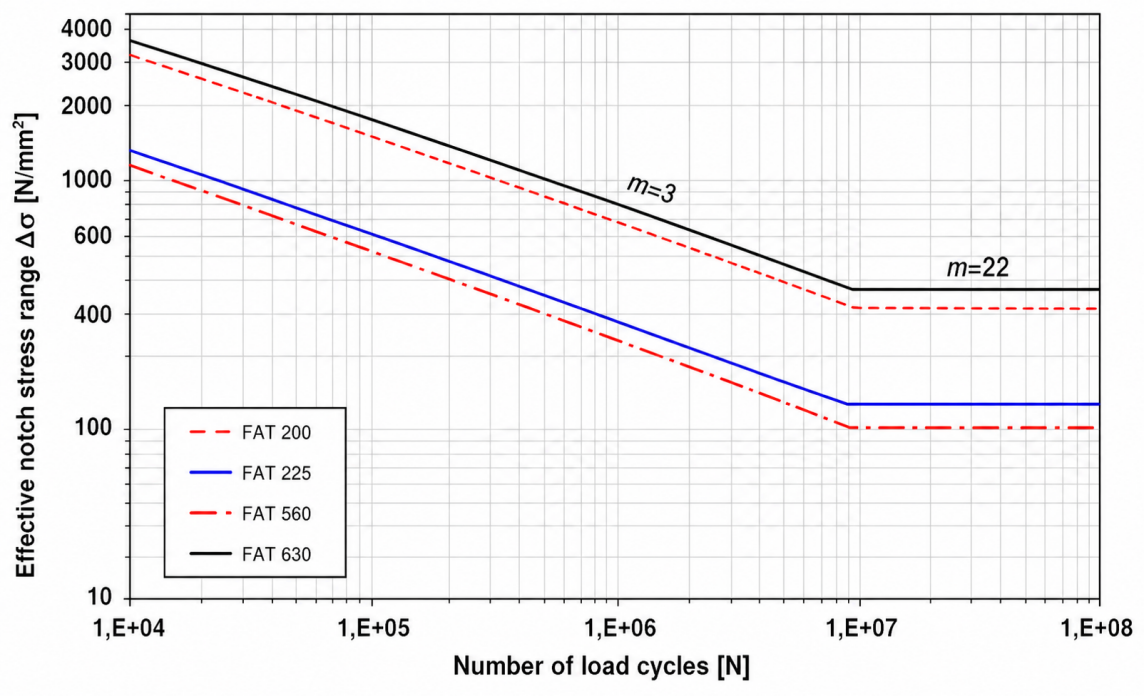


Figure 2.13: Effective notch stress S–N curves for selected FAT classes [2].

Although more computation is required, a more detailed picture of local stresses is provided, especially in cases of welded joints under fatigue. A comparison of the three methods is given in the following section.

2.3.4 Comparison Between Methods

These three fatigue assessment methods differ mainly in the treatment of the effects of stress concentrations and the refinement of the modelling used [2], [14].

In the nominal stress method, the evaluation of the stresses is global, and the effects of the stress concentrations are included indirectly in the form of detail categories. This method is computationally efficient but has limitations in complex geometries.

In the structural hot-spot stress method, the effects of the macro-geometrical features are included by extrapolating the stresses in the vicinity of the weld toe, reducing the dependence on the detail categories. However, the effects of the local weld notch are still not included.

In the effective notch stress method, the evaluation of the stresses is the most refined, as the weld geometry is included in the modeling in the form of a reference notch radius recommended by the IIW [14]. Although a high degree of accuracy and reduced scatter can be achieved, the modeling effort required in the form of a finite element analysis is high.

In the case of corrugated web girders, where a complex stress redistribution occurs in the region of the flange-to-web connection, local approaches are generally more suitable.

2.4 Stress Concentration Factor (SCF)

Stress concentrations are present at discontinuities in the structure's geometry, including welds, notches, and changes in the cross-sectional geometry of the structure. This affects the distribution of stress in the structural component, causing stress concentration. This concentration of stress is usually determined by the stress concentration factor, a key component in the fatigue analysis of welded steel structures.

2.4.1 Definition and Theoretical Background

In the elastic regime, the stress concentration factor can be defined as the ratio between the maximum stress at the discontinuity and the nominal stress within the member [15]. The SCF can be described as in Eq. 2.6.

$$\text{SCF} = \frac{\sigma_{\max}}{\sigma_{\text{nom}}} \tag{2.6}$$

Where $\sigma_{\max}$ represents the maximum stress at the discontinuity, and σ_{nom} represents the nominal stress calculated through global equilibrium and section properties using conventional beam theory. The stress concentration factor can be defined as a dimensionless quantity that provides a useful tool for evaluating the severity of discontinuities. In fatigue design, it can be used as a tool for evaluating the deviation between the stress states.

2.4.2 SCF Interpretation in SHSS

In the context of the structural hot-spot stress (SHSS) method, the SCF is defined as in defined in Eq. 2.7 [16].

$$\text{SCF} = \frac{\sigma_{hs}}{\sigma_n} \tag{2.7}$$

where σ_{hs} the structural hot-spot stress is found by extrapolation of stresses measured at a specified distance from the weld, and σ_n the nominal stress is determined from beam theory.

For corrugated web girders, the folded web structure leads to a complex redistribution of stresses in the flange-web connection, particularly in the region of the S-point, where

inclined and flat folds intersect. It has been shown by experimental and numerical investigations that, in corrugated web girders, SCF is heavily influenced by various geometric parameters, for example, corrugation angle, bend radius, web slenderness, and flange dimensions [16]. It follows that SCF acts as a useful tool for assessing the influence of these governing parameters.

2.4.3 SCF Interpretation in ENS

In ENS method, fatigue analysis is directly performed based on the effective notch stress calculated at the weld toe or root locations where crack initiation is most likely to occur. Although the stress concentration factor is not explicitly utilized within the fatigue verification procedure, the ratio of the effective notch stress and the nominal stress can be interpreted as a generalized stress concentration factor, see Eq. 2.8. This ratio offers useful insight into the influence of geometric parameters on the stress field and can be used as a tool for parametric studies.

$$\text{SCF} = \frac{\sigma_{ens}}{\sigma_n} \tag{2.8}$$

where σ_{ens} the Effective notch stress, and σ_n the nominal stress is determined from beam theory.

In terms of corrugated web girders, a relationship between the nominal stress and the effective notch stress must be developed. This relationship can reduce computational complexity. Thus, the conceptual role of the SCF within the ENS method must be understood as a prerequisite for the development of the predictive formulation proposed in this thesis.

2.5 Review of Previous Experimental Investigation

Regarding experimental studies on trapezoidal corrugated web plate girders (CWPGs), most research has focused on evaluating fatigue characteristics under cyclical loading, identifying crack initiation mechanisms, and studying stress concentration characteristics as affected by geometric parameters. The following subsections identify the most relevant experimental studies on CWPGs, as they relate to the validation and modelling approach adopted in this thesis.

2.5.1 Ibrahim et al.

Fatigue experiments were carried out by Ibrahim et al. [17] on six trapezoidal corrugated web plate girders subjected to four-point bending with cyclic loading. The fatigue experiments were carried out on hybrid plate girders consisting of thin corrugated webs welded with flat flanges. The four-point bending ensured a constant bending moment over a particular section, enabling fatigue behavior under different stress ranges.

The fatigue experiments showed that fatigue crack development always started at the weld toe at the intersection of the inclined web fold and the tension flange. The fatigue crack then proceeded across the width of the flange, resulting in fracture. The S-N curve was determined using regression analysis. The fatigue results showed that fatigue behavior is much better for trapezoidal CWPGs compared with stiffened flat web plate girders. The fatigue life was found to be 50–80% longer than that of flat web plate girders with transverse stiffeners welded onto the tension flange, and 30–50% longer than that of flat web plate girders with transverse stiffeners cut short from the flange. The fatigue behavior of trapezoidal CWPGs is found to be between AASHTO fatigue categories B and C.

2.5.2 Kövesdi et al.

Kövesdi et al. [18] carried out extensive experimental research on the structural response of corrugated web girders, with special emphasis on the interaction of bending and shear, as well as the effects of transverse bending. Even though it was not the primary focus, the experimental work provides significant insights into stress redistribution in the girder cross-section.

The results of this research clearly illustrate the so-called accordion effect, in which the corrugated web resists shear forces, whereas the flanges resist normal stresses resulting from bending. This results in minimal longitudinal stress in the web, except in areas close to the flange–web intersection. Another significant outcome was that the results depend on geometric parameters, i.e., corrugation angles, fold configuration, etc.

2.5.3 Wang and Wang

Wang and Wang [19] used an experimental and analytical approach to study the fatigue behavior of the welds between the corrugated steel web and the flange plate. The authors carried out fatigue testing on the specimens made from the web/flange interface, with different corrugation angles and different transition curvature radius.

According to the experimental observations, the fatigue cracks occurred at the weld

toe, where the specimen had the external transition curvature between the inclined and longitudinal folds. The authors demonstrated the significant role of the geometric parameters, where an increase in the corrugation angle led to an increase in the stress concentration because of the greater inclination of the corrugation relative to the principal stress direction. In addition, the authors found that an increase in the bending radius reduced the stress at the weld toe.

2.5.4 Tong et al.

Tong et al. [20] carried out fatigue experiments on trapezoidal corrugated web girders by utilizing simplified corrugated web T-section (CWT) specimens that mimic the stress distribution of the corrugated web girder. The experiments confirmed that the simplified specimen, due to the accordion effect, had a similar flange stress distribution as the corrugated web girder. The fatigue experiments were conducted on the simplified corrugated web T-section specimen with different corrugated web radii, 30 mm and 60 mm. The experiments measured the hot-spot stresses on the specimen. The hot-spot stresses were measured at the S-point, the intersection point between the inclined fold and the longitudinal weld. The measured values of the stress concentration factors (SCF) were 1.63 for the 30 mm radius specimen and 1.54 for the 60 mm radius specimen. The results indicated that the radius affects the stress concentration. The S-N curve based on the hot-spot stresses was derived, revealing that the local stress calculation minimizes the scatter in fatigue life prediction.

2.6 Review of Previous Numerical Studies

This chapter will introduce a review of the previously conducted numerical analysis work that is related to corrugated web girders. The main aim of this review is to assess the degree to which corrugated web parameters affect the fatigue strength and the stress concentration factor. The results from the review will be summarized and used as a basis for the discussion in this thesis.

2.6.1 Anami et al.

Chapter 5 of the report written by Anami et al. [21] discusses the results of a parametric finite element analysis of the stresses at the weld toes of corrugated web girders subjected to a constant bending moment. In the study, linear elastic three-dimensional finite element models are used, and the element formulation is based on the eight-node solid element.

In modeling, there are two steps. First, a global model of the entire girder is analyzed to find the displacement field. Then, a local model of the weld toe area of the girder,

especially in the area where the web and flange meet, is developed, and the boundary conditions are the global displacements. In the local model, a radius of 1 mm is used in the weld toe area.

For the assessment of fatigue, the Structural Hot Spot Stress (SHSS) method was utilized and modified to consider local effects caused by notches. Hot-spot stress was determined by extrapolating longitudinal stresses at $0.4T$ and $1.0T$ from the weld toe. This was based on IIW recommendations. Total stress at the weld toe was defined as the product of a structural stress concentration factor (K_{hot}) and a local toe profile factor (K_{toe}).

From the parametric analysis, it can be noted that the corrugation angle, represented by θ, plays a vital role in the modulation of stress concentration. Reduction in the value of θ results in a considerable reduction in hot-spot stress concentration and toe stress concentration. The bend radius of the fold corner, represented by R, has been identified as the second most important parameter that influences the modulation of stress concentration. An increase in the value of the bend radius results in a linear reduction in the peak of the stress concentration at point S, especially at higher values of the corrugation angle according to Figure 2.14.

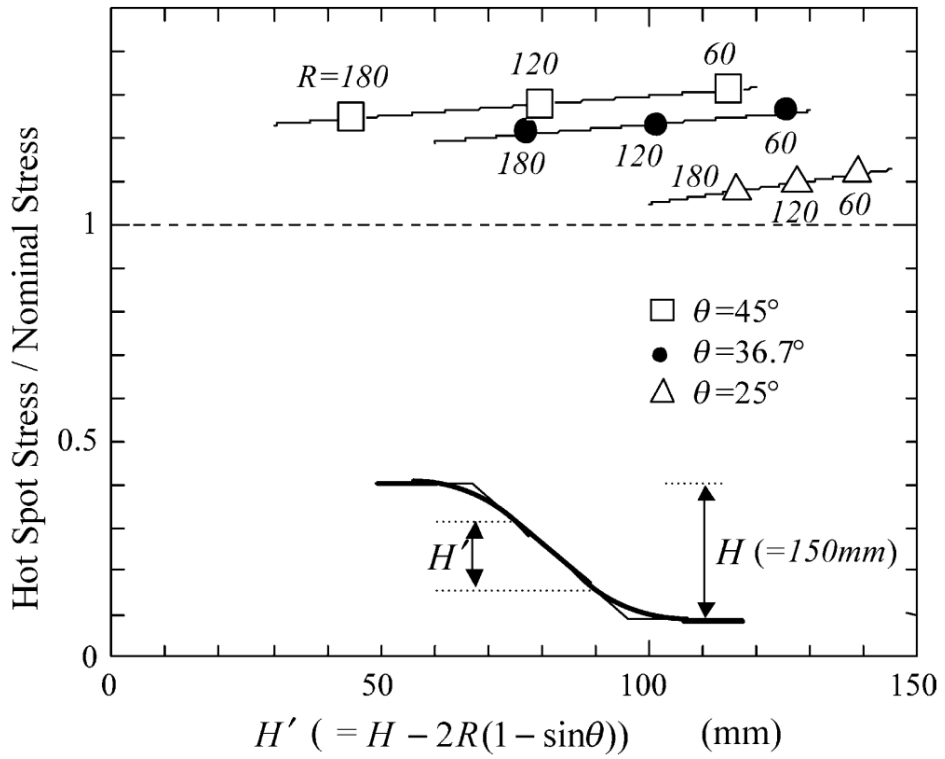


Figure 2.14: Influence of corrugation geometry on the hot-spot stress to nominal stress ratio [21].

Additional parameters that influence the stress concentration include the corrugation depth, represented by H, the longitudinal fold length, represented by L, the width of the flange, represented by W, and the flange thickness, represented by T. An increase in the value of the flange thickness reduces the stress concentration, whereas an increase in the value of the flange width increases the stress concentration, especially at lower values of the corrugation angle.

Overall, the study demonstrates that fatigue-critical stresses are governed primarily by corrugation angle and bend radius, and optimization of these parameters is more effective than modifying flange geometry.

2.6.2 Xu et al.

Xu et al. [22] combined a large-scale fatigue test and a numerical investigation of the oblique I-girder with a trapezoidal corrugated web, subjected to four-point bending. On the numerical side, the paper employs a two-step finite element method in ANSYS. First, a precise global three-dimensional solid element model (SOLID186) is used to investigate the stress field. Then, detailed sub-models are developed in the vicinity of the S-point, which is a critical area in the girder. For the calculation of the effective notch stress, a fictitious notch radius of $r_{\text{ref}} = 1\,\text{mm}$ is used at the weld toe, as recommended by the IIW. For the calculation of the structural hot-spot stresses, the stresses are determined by extrapolation at $0.4t$, $0.9t$, and $1.4t$.

For the determination of fatigue life, the authors use three different methods: the nominal stress method (FAT 90), the structural hot-spot stress method (FAT 120), and the effective notch stress method (FAT 225). In the notch area, the mesh size is refined to as small as 0.25 mm.

During the investigations into the influence of the corrugation parameters on the behavior, the parametric FE analysis results indicate that the most significant parameters influencing stress concentration at the S-point are the corrugation angle and the bending radius. Increasing the corrugation angle from 30° to 60° increases the fatigue notch factor by as much as 30%. On the other hand, decreasing the radius increases the fatigue notch factor significantly, and the effect is most pronounced when the radius drops below the specified value; keeping the radius at or above 150 mm, or $23t_w$, significantly reduces stress concentration. On the other hand, the weld size, k, appears to be a secondary parameter, and the flange inclination angle, α, appears to be negligible.

The major findings and takeaways for the future, therefore, are as follows: fatigue

cracks occur at the S-point, but in the presence of shear forces, the critical region where fatigue occurs shifts from a bending-dominated zone to a zone where both bending and shear act simultaneously, reducing fatigue life by as much as 11%. Among all the evaluation methods, the most conservative method is the effective notch stress method. These studies therefore conclude that the fatigue life of corrugated web girders is a function of stress concentration, which in turn depends on the corrugation angle and the radius.

2.6.3 Liu et al.

Liu et al. [23] investigated the fatigue performance of large-dimension corrugated steel web (CSW) girders with wavelengths ranging from 1000 to 2000 mm under four-point bending.

A three-step validated three-dimensional finite element analysis in ABAQUS was carried out. These steps are as follows: a global analysis to determine the stress field, Sub-model I to find the hot-spot stress at the fatigue-critical S-point, and Sub-model II, which used a refined mesh to conduct the fatigue life prediction, with the minimum element size being 0.25 mm. A fictitious notch was incorporated at the weld toe where the web and flange meet, and the radius of the notch was 1 mm. The hot-spot stress was determined using the IIW three-point extrapolation method, and the stress concentration factor was defined as:

$$K_s = \frac{\sigma_{hs}}{\sigma_n} \tag{2.9}$$

As far as fatigue is concerned, the stress concentration factor K_s at the S-point increases almost in a straight line with the corrugation angle θ_w, for all wavelengths, see Figure 2.15

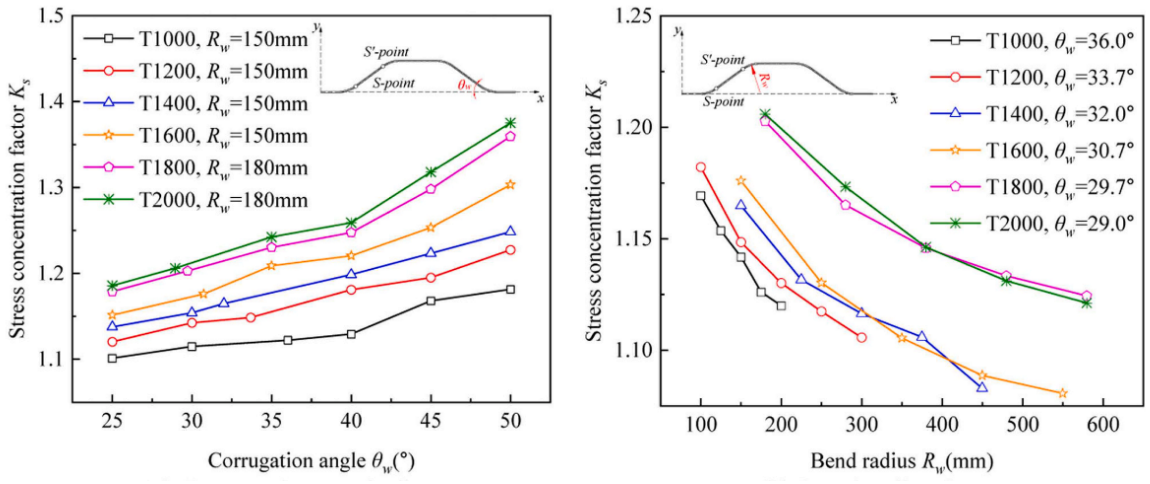


Figure 2.15: Influence of (left) corrugation angle and (right) bend radius on the stress concentration factor [23].

With the increase in wavelength, the sensitivity of the stress concentration factor to the corrugation angle increases, indicating a higher fatigue risk. On the other hand, increasing the bend radius R_w decreases the stress concentration factor K_s, but to a lesser extent for large bend radii. Moreover, longer wavelengths (1600–2000 mm) always cause a higher stress concentration factor compared to shorter wavelengths.

The key takeaway from the fatigue results is that as the wavelength increases, the fatigue detail categories decrease. In other words, for wavelengths between 1600 and 2000 mm, the fatigue classes fall in the FAT 50–71 range (IIW). In comparison, shorter wavelengths at or below 1400 mm correspond to higher fatigue classes, which is consistent with the results from earlier experimental work. The warning from the study is that the fatigue resistance of large-dimension CSWs could be overestimated by using design rules based on typical wavelengths at or below 1000 mm.

2.6.4 Wang et al.

Wang et al. [24] also attempted to investigate the thickness of the flange by developing a detailed three-dimensional finite element analysis to examine the stress concentration and fatigue performance at the weld area where the web meets the flange for corrugated steel web beams. The fillet weld was modeled explicitly by utilizing SOLID185 elements. The authors also utilized a finer mesh at the weld toe and weld root to simulate the stress gradient. The material was defined using a bilinear elastic–plastic model.

Fatigue analysis was performed using the Effective Notch Stress (ENS) approach. A fictitious notch radius of 1 mm was considered at the weld toe and root, following the IIW guidelines. Analysis of the notch stresses at various angular sections, i.e., 0° to 45°, was performed to identify the most critical point of intersection of the fold. The FE results indicate that the peak stress occurs at a location where the longitudinal and inclined folds intersect on the corrugated web, i.e., the S-point. Whether the peak stress occurs at the weld toe or the weld root depends on the reference angle.

Based on the parametric analysis conducted, the flange thickness t_f is seen as a minor parameter that influences the stress distribution in the flange and weld zone. An increase in the value of t_f would only bring about a small improvement in the fatigue life without contributing significantly to fatigue resistance after a particular point when compared to geometrical parameters. In contrast to the above, the angle of corrugation θ is a significant parameter in fatigue strength because of stress concentration.

2.6.5 Ibrahim et al.

The parametric finite element analysis carried out in [17] revealed that the fold-line radius of curvature (R) is a major geometric variable influencing the stress concentration at the corrugation fold and tension flange area. A larger radius of curvature between the parallel and inclined folds can effectively lower the stress concentration factor (SCF) in the corrugated web and tension flange.

Such a phenomenon has also been observed in previous research, where the shape of the corrugation was found to have a prominent effect on the stress concentration distribution along the corrugation shape, see section 2.2.3. Specifically, as the bending radius increases, the corrugation shape tends to become more sinusoidal. This results in fewer geometric sharp points of intersection, hence a smoother transition of the shear stress and reduced levels of stress concentrations.

2.7 Summary of Literature Findings

The results of the investigated experimental and numerical studies are presented in Table 2.2. Table 2.2 provides a summary of the cumulative effects of the governing corrugation parameters on the stress concentration factor (SCF), which guides the research direction of the present study.

Table 2.2: Summary of the influence of governing corrugation parameters on SCF based on previous studies.

Parameters	Previous research	Method	Impact on the SCF
Corrugation angle, θ	Anami et al.	SHSS using large-scale FEM testing modified for local notch effect	Increasing θ from 25° to 36.7° increased SCF from 1.4 to 1.85 (∼32%). Increasing θ from 30° to 60° resulted in ∼30% increase in stress concentration.
	Xu et al.	ENS and effective notch factor with parametric FEM	Increasing θ from 30° to 60° led to ∼25–30% increase in effective notch stress and SCF at the S-point.
	Liu et al.	SHSS using large-scale testing	Increasing θ from 25° to 50° results in SCF ranges 1.10–1.18 up to 1.18–1.37, corresponding to a 7–16% increase depending on corrugation wavelength.
Bending radius, R **[mm]**	Anami et al.	SHSS using large-scale FEM testing modified for local notch effect	Increasing R from 60 to 180 reduced SCF from approximately 1.26 to 1.22 (∼3–4% reduction).
	Xu et al.	ENS and effective notch factor with parametric FEM	Smaller R increases stress concentration at the weld toe, especially near the transition curvature region.
	Liu et al.	SHSS using large-scale testing	Increasing R from 150 mm to 600 mm reduced SCF from 1.20 to $\sim$ 1.12 (∼6–7% reduction). Sensitivity decreases for large radii ($>$ 400 mm).
Flange thickness, t_f **[mm]**	Wang et al.	FE model based on ENS	Increasing t_f from 5 to 10 mm increased fatigue life from 2.09 × 10^6 to 4.89 × 10^6 cycles (∼134% increase), inducing lower SCF.

Parameters	Previous research	Method	Impact on the SCF
	Anami et al.	SHSS using large-scale FEM testing modified for local notch effect	Increasing t_f from 20 to 40 mm reduced hot-spot stress at point S from 1.23 to 1.17 (∼5% reduction). Influence becomes negligible beyond 40 mm.
Flange width, b_f [mm]	Anami et al.	SHSS using large-scale FEM testing modified for local notch effect	Increasing b_f from 225 to 450 mm changed SCF from 1.86 to 1.84 (∼1% increase), indicating negligible influence.
Corrugation depth, a_3 [mm]	Anami et al.	SHSS using large-scale FEM testing modified for local notch effect	Increasing a_3 from 120 to 180 mm changed SCF from 1.23 to 1.22 (∼1% reduction), indicating negligible influence on SCF.
Length of longitudinal fold, a_1 [mm]	Anami et al.	SHSS using large-scale FEM testing modified for local notch effect	Varying a_1 from 109 to 439 mm changed SCF from 1.21 to 1.24 (∼2–3% increase). Influence on SCF is relatively small.

2.7.1 Influence of Governing Parameters on SCF

Fatigue failure, based on the current studies on the behavior of trapezoidal corrugated web girders, follows a specific pattern. Most experiments on the fatigue behavior of the girder indicate that fatigue crack initiation occurs at the S-point, where the inclined fold of the corrugation and the weld between the web and the flange intersect [16], [25].

The literature review on the behavior of the stress concentration factor at the weld toe suggests that the web's geometric layout significantly influences the stress concentration factor at the weld toe. The most significant factors are the corrugation angle and the bending radius. Increasing the corrugation angle was found to cause a significant increase in the stress concentration factor by 25–30%, depending on the specific configuration [21], [22], [23]. Increasing the bending radius, on the other hand, was found to cause a reduction in the stress concentration factor by smoothing the transition between the inclined and longitudinal folds, thus reducing the sharpness of the stress concentration.

Increasing the flange thickness (t_f), the flange width (b_f), the corrugation depth

(a_3), and the longitudinal fold length (a_1) has a relatively small effect on the stress concentration factor at the weld toe. The stress concentration factor was found to decrease with increasing flange thickness, while only minor variations are observed when the flange width, corrugation depth, and longitudinal fold length are increased.

2.7.2 Dominant vs Secondary Parameters

The key parameters that influence stress concentration can be broadly classified into three groups, dominant, secondary, and unstudied parameters, as shown in Table 2.3. The two dominant parameters influence the sharpness of the structure, and this, in turn, affects the redistribution of stresses around the S-point.

Table 2.3: Classification of governing parameters influencing stress concentration.

Dominant parameters	**Secondary parameters**	**Unstudied parameters**
Corrugation angle (θ)	Flange thickness (t_f)	web thickness (t_w)
Bending radius (R)	Weld size	
	Flange width (b_f)	
	Corrugation depth (a_3)	
	Longitudinal fold length (a_1)	

On the other hand, the secondary parameters influence local stiffness, and their influence on stress concentration is relatively moderate to small in comparison to θ and R. It has been established by several researchers that local methods, such as the hot-spot stress method and the effective notch stress method, provide a better prediction of fatigue-critical stresses compared to the nominal stress method, particularly for the S-point [16], [21].

2.7.3 Identified Research Gap

The literature reviewed in the context of the current research reveals that the fatigue behavior of corrugated web girders has been extensively studied through experimental testing and comprehensive finite element modeling. The dominant effect of the corrugation angle (θ) and bending radius (R) on stress concentration at the S-point has been corroborated in most studies.

However, it has also been observed that the vast majority of studies considered the effect of parameters in isolation through the one-factor-at-a-time (OFAT) method within narrow ranges of geometrical variables. Fatigue assessment has also been performed primarily using the structural hot-spot stress method and the effective notch stress method individually. Although these methods are more accurate than the nominal stress method, they require sophisticated modeling of the weld area and involve high computational costs.

Despite the extensive numerical and experimental work carried out on corrugated web girders, no unified formulation has been presented to establish a direct relationship between nominal and local stresses while systematically accounting for the governing geometric parameters. In addition, the influence of web thickness (t_w has not been explicitly investigated in previous studies. Accordingly, the effect of web thickness is investigated in the present study.

Chapter 3

Theoretical Framework

3.1 Governing Geometric Parameters

In addition to the findings presented in Section 2.7, the effects of the corrugation depth and the fold lengths are primarily associated with secondary bending in the flange. They also influence the load transfer mechanism between the web and the flange. The role of the corrugation depth in this mechanism has been highlighted as discussed in [16]. These parameters have a certain effect on the stress distribution, but it is relatively less significant than the corrugation angle and bend radius. In addition, the parameters related to the flange, i.e., the flange width and thickness, have a certain effect on the local stiffness of the flange plate and, subsequently, on the magnitude of the hot spot stress. In fact, it has been shown that once a certain stiffness is achieved, the effects of these parameters become moderate.

Investigations on the weld size and the geometric details of the weld region have shown that, considering the fabrication limitations, the effect of the changes in the weld size on the stress concentration is less significant compared to the geometric details of the corrugation profile [16], [22].

3.2 Selection of Weld Throat Thickness

The weld throat thickness of the flange-to-web connection in corrugated web girders has previously been determined based on the findings reported in [26]. Where the fatigue cracking risk at both the weld toe and the weld root is evaluated using the effective notch stress method. The results obtained in [26] indicate that the fatigue cracking risk at the weld root can be avoided if the following condition in Eq. 3.1 is satisfied.

$$\frac{2a_w}{t_w} \geq 1.15 \tag{3.1}$$

Thus, in this study, the weld throat thickness has been selected such that the condition in equation 3.1 is always valid for all cases. Therefore, the fatigue performance of the flange-to-web connection in corrugated web girders can be ensured with crack initiation more likely to occur at critical locations namely at the weld toe.

3.3 Rationale for Adopting the ENS Approach

The fatigue assessment of the flange-to-web welded detail in corrugated web girders is particularly challenging due to the complex stress flow introduced by the corrugated geometry. Unlike flat web girders, corrugated webs generate additional local stress concentrations at the intersection between inclined and flat folds (S-point), where fatigue cracks are most frequently observed. In such cases, the nominal stress approach can not capture the local notch effects associated with weld geometry and stress gradients.

The Effective Notch Stress (ENS) method provides a more refined local assessment by explicitly modeling the weld toe and weld root using an idealized notch radius, as recommended by the IIW. This allows the local stress concentration effects governing crack initiation to be directly evaluated. As demonstrated in [26], the ENS method significantly reduces scatter in fatigue evaluation compared to the nominal stress method and confirms the applicability of DC225 when the maximum principal effective notch stress is used.

Moreover, the ENS approach enables the evaluation of both weld toe and weld root cracking, which is essential for corrugated web girders where the force transfer mechanism is more complex than in conventional welded details. In addition, the ENS method captures the influence of plate thickness through the modeling of the stress gradient in the vicinity of the weld, an effect that is not explicitly accounted for in the structural hot-spot stress method. For these reasons, the ENS method is considered more appropriate for assessing the fatigue performance of the studied welded connection in this project.

Chapter 4

FE Modeling

This chapter presents a numerical investigation into the fatigue strength of trapezoidal corrugated girders. The analysis employs nominal stress, structural hot-spot stress, and effective notch stress methods, implemented using finite element software; Abaqus. The numerical models are verified through validation against a reference study by Zuo et al. [27]. Which serves as the basis for validating the modeled girder, in addition to convergence studies.

The modeling process has been carried out using a sequential methodology according to Figure 4.1. First, a global model was created to represent the beam's geometry, loading conditions, and boundary conditions. This global modeling was performed to check that the results obtained through numerical analysis are consistent with the results obtained through experiments, as described in the reference study by Zuo et al. [27], [28], [29]. Subsequently, sub models according to EN1993-1-14 [3] have been created to study the local stress conditions. This approach enables a trade-off between efficiency and modeling accuracy to be achieved.

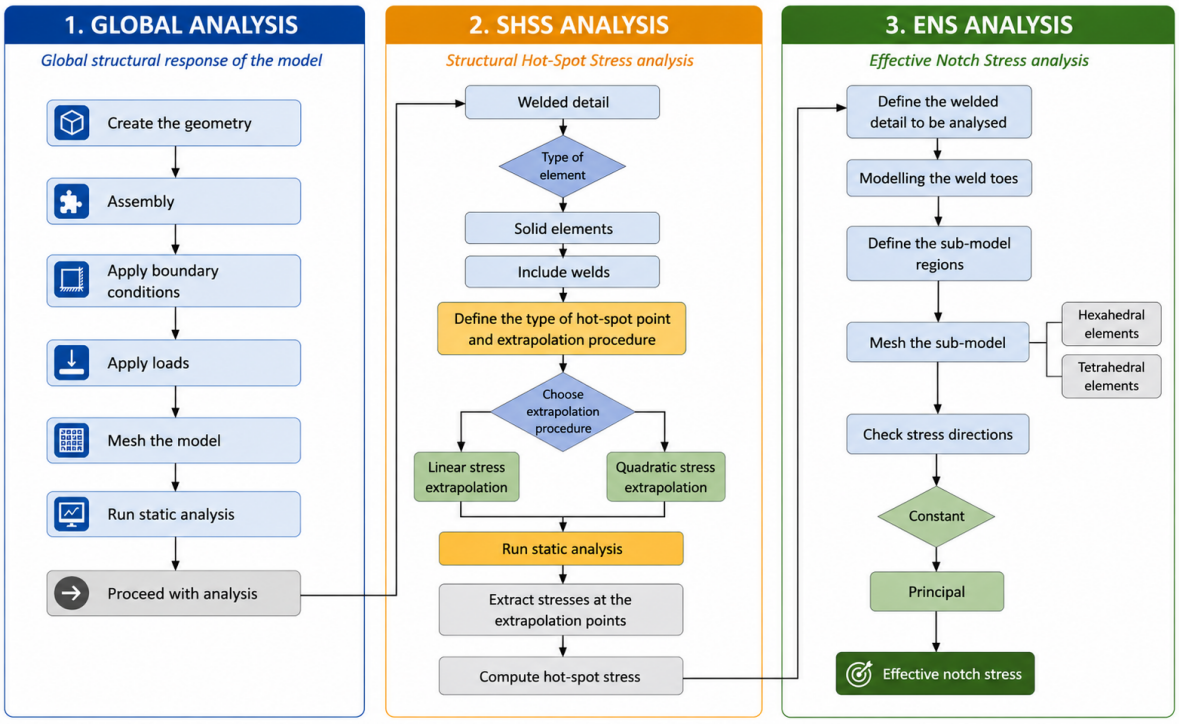


Figure 4.1: Integrated finite element analysis workflow showing the global model analysis, hot-spot stress analysis, and effective notch stress evaluation procedure.

Note that only the constant bending region, CBR, will be considered, and the S-point is of major interest. Therefore, any results reported in the reference study outside the CBR will be ignored, together with the results where fatigue a failure occurred at the B-point.

4.1 Global Finite Element Model and Validation

This section provides information on the development of the global finite element model and its validation by comparing it with analytical calculations used in the reference study [28]. The modeling approach, the mesh convergence study, and the result comparisons are explained in the following sections.

4.1.1 Purpose of the Global Model

The first step of the numerical analysis was the creation of a global finite element model of the corrugated steel girders. The objective of this step was to simulate the global behavior of the girders that were tested in the reference study. It also serves to validate that the finite element model accurately represents the experimentally tested configuration.

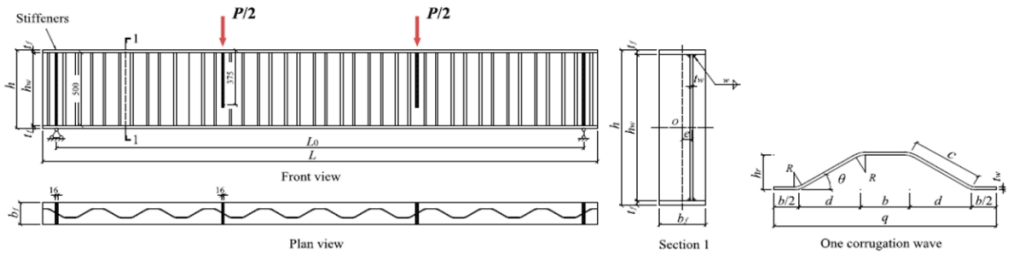
In this step of the analysis, the global structural response of the four reference specimen girders was validated with respect to the bending response, nominal stresses, and the

deflection using classical beam theory. However, the authors in the reference study [28] defined the stress concentration factor (SCF) based on the nominal stress of geometrically identical girders with a flat web. To enable a consistent comparison between the finite element results and the experimental data, the stresses reported in the experimental study were converted into equivalent stresses. These equivalent stresses were then applied to corrugated web girders prior to defining the loading conditions in Abaqus, see Appendix A. Similarly, the other girders were modeled through the same procedure and their nominal stresses were verified by beam theory.

The global finite element model must first be validated to ensure that it accurately represents the global response of the girder. Only then can the model be considered suitable as the basis for local finite element analysis, the structural hot spot stress analysis and the effective notch stress analysis.

4.1.2 Geometry and Reference Specimens

In order to validate the global model, four corrugated web girders from the experimental test of Zuo et al [28] were selected. The girders selected for validation purposes include GA-1, GB-1, GC-1, and GD-1. These girders have trapezoidal corrugated web with different geometric configurations, as illustrated in Figure 4.2. The use of several girder configurations allows for a more comprehensive assessment of the robustness of the proposed approach for FE modeling.



Group	Main geometry [mm]							Corrugation geometry [mm]							θ
	L	L_0	b_f	t_f	h	h_w	t_w	e	R	b	d	c	h_r	q	[°]
GA	3600	3420	150	16	532	500	5	30	30	80	100	116.6	60	360	30.9
GB	3600	3420	150	16	532	500	5	30	30	120	60	84.8	60	360	45.0
GC	3600	3420	150	16	532	500	5	30	60	120	60	84.8	60	360	45.0
GD	3500	3250	150	16	532	500	5	30	30	150	100	116.6	60	500	30.9

Figure 4.2: Geometry of the analysed corrugated web girders and corresponding geometric parameters [28].

The girders were subjected to four-point bending, resulting in a region of constant bending moment between the two loading points. To simulate this loading configuration accurately, the boundary conditions, load application, and proper implementation of the side baffles must be carefully defined, see test configuration below in Figure 4.3.

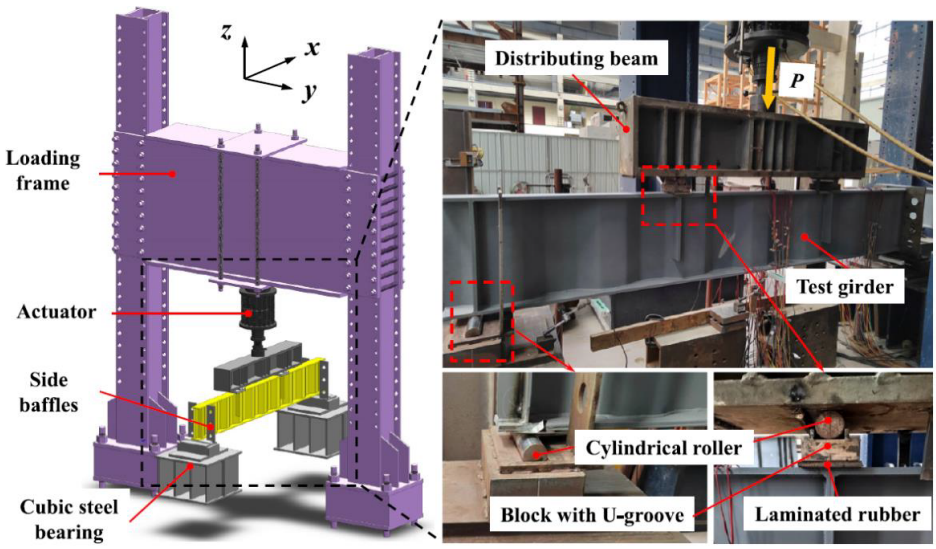


Figure 4.3: Overview of the experimental setup, loading system, and support conditions

A load of 385 kN, corresponding to the reference girder (GA-1), was applied in the finite element model to ensure direct comparison with the experimental study. This approach provides sufficient validation of the model for the reference case. The remaining girders are modeled in the same manner, with their nominal stresses verified using beam theory. In the fatigue analysis phase, a cyclic loading scheme, ΔP, is applied in accordance with Table 4.1.

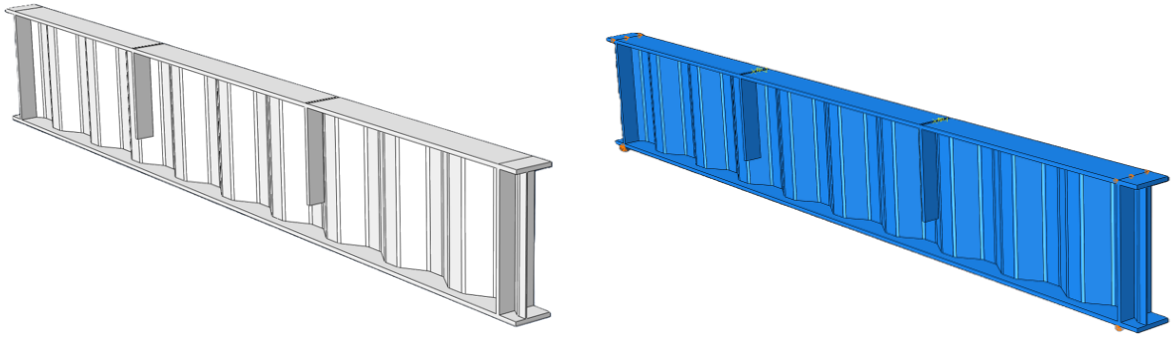
Table 4.1: Fatigue test loading scheme for all specimen series [29].

Parameter	GA				GB				GC				GD	
	GA-1	GA-2	GA-3	GA-4	GB-1	GB-3	GB-4	GB-5	GC-1	GC-2	GC-3	GC-4	GD-2	GD-4
P_{max} [kN]	385	400	418	436	385	418	436	455	385	400	418	436	432	472
P_{min} [kN]	100	100	100	100	100	100	100	100	100	100	100	100	100	100
ΔP [kN]	285	300	318	336	285	318	336	355	285	300	318	336	332	372

4.1.3 Finite Element Modeling

The girders were modeled using three-dimensional solid elements that included the top flange, bottom flange, corrugated web, and transverse stiffeners. All the structural components were merged into one part to ensure compatibility between the components without having to define tie constraints.

Boundary conditions are applied to ensure that the model behaves simply supported. To achieve this, a partition is created at the center of the end-stiffener location, which is then used to impose the boundary conditions on both sides. In the case of the experiment, the stability of the beam in the lateral direction can be ensured by providing baffles at the sides of the beam near the supports. To simulate this condition in the FE–model, the nodes at the top of the outer stiffeners were restrained in the lateral direction, i.e. U2 = 0. Ensuring that the beam does not move in the lateral direction. The applied boundary conditions are listed in Table 4.4.



(a) Undeformed model. **(b)** Applied boundary conditions.

Boundary region	U_x	U_y	U_z	R_x	R_y	R_z
Left support	–	0	0	–	–	–
Right support	0	0	0	–	–	–
Side baffles	–	0	–	–	–	–

Figure 4.4: Global FE–model and applied boundary conditions in Abaqus CAE. A value of "0" denotes a restrained degree of freedom, while "–" denotes a free degree of freedom.

The material used in this analysis is linear-elastic steel, characterized by a Young's modulus of $E = 210$ GPa and a Poisson's ratio of $\nu = 0.3$. The numerical model is discretized using 10-node quadratic tetrahedron (C3D10) brick elements with reduced integration. These elements are employed to accurately capture the bending behavior of three-dimensional structures while maintaining sufficient numerical efficiency.

The plates that were used in the experiment to apply loads and to provide support to the girder were not explicitly modeled. Their impact on the girder is simulated using reference points and kinematic coupling constraints (master points). The edges of the stiffeners were constrained to the reference points and were allowed to move as if they were constrained by the plates. The load is spread across the thickness of the stiffener.

In the reference study by Zou et al. [28], the nominal stress for specimen GA-1 was determined as 139 MPa using classical beam theory. In this approach, the nominal stress (σ_n) is defined as the normal stress at the upper surface of the tensile flange. The stress was calculated based on the applied bending moment and the section properties of an equivalent flat-web girder. To enable comparison with the reference study, a conversion of the stress values was performed using Eq. 4.1. This was achieved by first determining the nominal stress based on an equivalent flat-web girder $\sigma_{n,\mathrm{FWG}}$. Subsequently, $\sigma_{n,\mathrm{FWG}}$ was used to evaluate the nominal stress $\sigma_{n,\mathrm{CWG}}$ at the same location in the CWGs. Ensuring a consistent basis for comparison between the two approaches by their stiffness relation.

$$\sigma_{n,\mathrm{CWG}} = \frac{I_{\mathrm{FWG}}}{I_{\mathrm{CWG}}} \sigma_{n,\mathrm{FWG}} \text{ [MPa]} \tag{4.1}$$

4.1.4 Mesh Sensitivity Analysis

A mesh convergence study was conducted to determine an appropriate mesh size for the global finite element model. The study was performed on the GA girder, which was considered representative of the overall structural behavior. A range of mesh sizes, varying from 100 mm to 10 mm, was investigated in accordance to Table 4.2. For each mesh size, the maximum midspan deflection was recorded as a key global response parameter sensitive to mesh refinement.

Table 4.2: Mesh convergence study based on midspan deflection.

Case	Mesh size [mm]	Deflection [mm]	Deviation [%]
Reference	100	4.909	
1	50	4.964	1.0
2	30	4.985	0.4
3	25	5.000	0.3
4	20	5.030	1.0
5	10	5.010	−0.4

As the mesh size decreased, the deflection values varied smoothly and approached a nearly constant value, indicating convergence. All considered mesh sizes produced very close deflection results, with deviations remaining within an acceptable range. Therefore, a mesh size of 20 mm was selected for the global analysis, as it provides a suitable balance

between computational efficiency and numerical accuracy. Larger mesh sizes also yielded reasonably accurate results and can be adopted for girder analysis to reduce computational time.

4.1.5 Global Model Validation

To verify the accuracy of the global finite element model, the numerical results were compared with analytical solutions in terms of nominal stress and maximum deflection. The nominal stress was obtained from the finite element model by defining a path along the top surface of the bottom flange within the constant moment region. Along this path, the longitudinal stress component (S11) was extracted, and its average value was used in comparison with the analytical stress.

The maximum deflection at midspan has been evaluated using the finite element results. This has been compared to the analytical results for deflection. The comparison between the results obtained using the finite element model and those obtained using the analytical approach is shown in Table 4.3.

Table 4.3: Comparison between analytical calculations and FE results for global validation. Applied load $P = 385\,\text{kN}$ for GA, GB, and GC, and $P = 415\,\text{kN}$ for GD.

Girder	**Nominal Stress**			**Deflection**		
	Analytical σ_n **(MPa)**	**FE** σ_n **(MPa)**	**Deviation (%)**	**Analytical** δ **(mm)**	**FE** δ **(mm)**	**Deviation (%)**
GA-1	162.6	158.0	2.5	4.96	5.03	1.4
GB-1	162.6	160.0	1.6	4.96	5.03	1.4
GC-1	162.6	159.0	2.4	4.96	5.08	2.6
GD-1	162.3	159.0	2.3	4.59	4.63	0.8

The results show that there is a good degree of agreement between the results obtained using the finite element approach and those obtained using the analytical approach. The difference between the results for nominal stresses and deflection is within 0.8–2.6%. Overall, this agreement is regarded as acceptable for the purpose of global validation. The discrepancies could be explained as resulting from the simplified assumptions used in the analytical calculations.

As may be concluded from the above results, it is possible to state that the FE model developed is capable of simulating the global behavior of corrugated web girders with acceptable accuracy. Therefore, this FE model could be accepted as a reliable tool for upcoming local analysis.

4.2 Local Sub-model for SHSS Evaluation

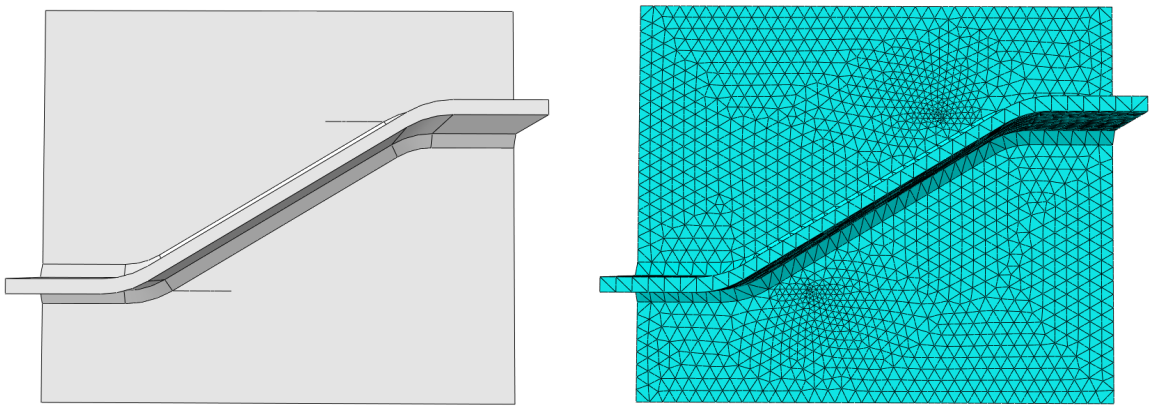
The SHSS analysis involves a finer mesh compared to the global model in order to accurately capture the local effects near the weld toe. Based on the global analysis, a mesh size of 20 mm is sufficient for estimating the overall structural behavior. However, this level of discretizations is too coarse for evaluating stress concentrations.

A sub-model approach was therefore adopted to overcome this limitation by considering a reduced portion of the structure with a finer mesh. The same element type used in the global model was retained in the sub-model. The sub-model consists of half wavelength of the girder structure located in the constant bending region and includes the weld geometry, which was not represented in the global model.

4.2.1 Sub-modeling Procedure

The SHSS model is developed as a sub-model of the global model, focusing on the constant bending region, which represents the critical area of interest in this study. Since all boundary conditions and applied loads have already been incorporated and solved in the global model, it is not necessary to redefine them for the sub-model. Instead, the sub-model obtains its loading conditions directly from the global analysis through the transfer of displacement fields.

To ensure proper coupling between the global model and the sub-model, new boundary conditions must be applied on the cut faces of the sub-model. Specifically, these boundary conditions are defined using nodal displacement data extracted from the corresponding region of the global model. This approach enables the sub-model to accurately reproduce the displacement field of the parent structure, while allowing for the inclusion of finer geometric details and the use of a significantly refined mesh. The developed sub-model is presented in Figure 4.5.



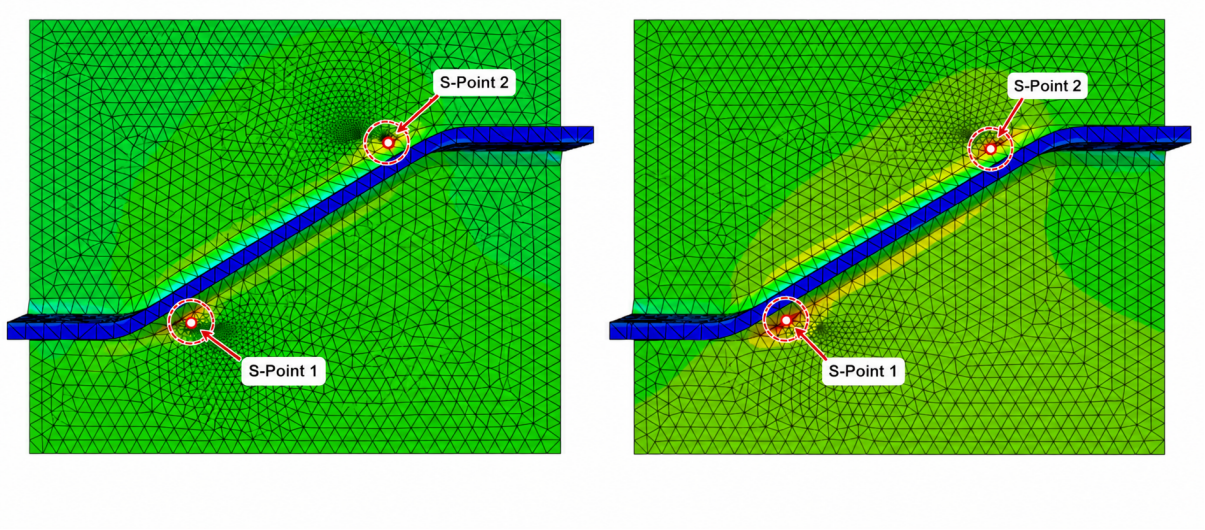
(a) Geometry and cross-section of the HSS specimen. **(b)** Finite element mesh of the HSS specimen.

Figure 4.5: Finite element model of the HSS member.

4.2.2 Local mesh refinement

The structural hot-spot stress is calculated using a stress extrapolation procedure, in which stresses are evaluated at specified distances from the weld toe. The selection of these distances depends on the extrapolation technique adopted, as described in section 2.3.2. In this study, two extrapolation methods were employed, namely linear extrapolation and quadratic extrapolation. Their effects on the results were compared in order to assess the influence of the chosen extrapolation technique on the accuracy of the stress evaluation.

Since the results of stress extrapolation are highly sensitive to mesh density, a fine mesh was adopted throughout the sub-model used for SHSS calculation. In addition, a local mesh sensitivity analysis was performed, with particular focus on the region surrounding the stress extraction points. Ensuring that the mesh is capable of accurately capturing the stress gradients near the weld toe. The mesh sensitivity analysis presented in Figure 4.6 shows that using three elements per path segment for stress extraction yields results comparable to those obtained with an extremely fine mesh. It can therefore be concluded that the adopted mesh resolution is sufficient to provide accurate stress values.



Mesh type	S-point	SHSS (Experimental) [MPa]	SHSS (FE) [MPa]
1 mm (Left)	1	208.4	209
	2	208.4	195
3 elements per distance (Right)	1	208.4	208
	2	208.4	194

Figure 4.6: SHSS sub-model stress distribution for the two investigated meshes and corresponding mesh sensitivity results for the GA-1 girder.

In the reference test case, strain gauges were installed at predefined measurement locations, with S-point 1 representing the primary measurement position. It should be noted, however, that the stress values obtained at the two S-points are not identical. This discrepancy arises from the influence of transverse bending on the local stress distribution within the cross-section.

4.2.3 Validation of Hot Spot Model

To verify the hot spot stress values calculated using the sub-model analysis, mesh sensitivity analysis, extrapolation methods comparison, and experimental data validation are considered. Furthermore, linear and quadratic extrapolation techniques were used to analyze the impact of each approach on HSS values. As a result, while both approaches show a similar trend, quadratic extrapolation provides slightly better estimates that match the experimental values. It is because the quadratic approach is superior in approximating non-linear stress distribution at the weld toe region.

Table 4.4: Comparison of experimental and FE-derived hot-spot stresses using linear and quadratic extrapolation methods. Applied load $P = 385\,\text{kN}$ for GA, GB, and GC, and $P = 415\,\text{kN}$ for GD.

Specimen	**S-point**	**Exp.** σ_{HS} **[MPa]**	**Quad.** σ_{HS} **[MPa]**	**Lin.** σ_{HS} **[MPa]**	**Dev. quad.** [%]	**Dev. lin.** [%]
GA-1	1	208.35	208.71	202.38	0.2	-2.9
GB-1	1	246.10	221.00	213.39	-10.2	-13.3
GC-1	1	206.93	210.96	204.05	1.9	-1.4
GD-1	1	234.94	224.28	216.86	-4.5	-7.7

Comparison between experimental hot spot stresses and numerical predictions using the extrapolation technique for specimens at S-point 1 is shown in Table 4.4. On the whole, the correlation is quite good, as the deviations vary between -10.2% and $+1.9\%$. Results for specimens GA-1 and GC-1 coincide closely with the experimental data, whereas results for specimen GB-1 demonstrate a higher deviation value. This error can be justified according to the findings of the reference study [28], where the location of the strain gauges relative to the weld toe may have affected the recorded strain values. The high sensitivity of SHSS to the location of the stress measurement means that slight errors in the measurement location could result in the observed deviation of stress values. The deviation recorded in the stress value of GB-1 can thus be explained by experimental measurement errors.

Now that the FE model has been validated against the experimental setup, the loading conditions applied in Abaqus should be updated to correspond to the loads presented in Table 4.1. The results obtained from this analysis will yield the SCF_{HS}, which will subsequently be used to evaluate the actual SHSS considered in the fatigue assessment conducted in the reference study [29]. Table 4.5 presents the nominal stress ranges obtained from the experimental testing and FE–analysis. Moreover, it shows the calculated hot-spot stresses and stress concentration factors SCF_{HS} for the investigated CWG specimens.

Table 4.5: Hot-spot stress ranges- and stress concentration factors for the CWG specimens. N_f are obtained from the reference study [20].

Specimen	$\Delta\sigma_{\text{nom,test}}$ **[MPa]**	N_f **(cycles)**	$\Delta\sigma_{\text{nom,FE}}$ **[MPa]**	$\Delta\sigma_{\text{HS,FE}}$ **[MPa]**	$\mathbf{SCF_{HS,FE}}$ **[-]**
GA-1	127	1 373 000	109	155	1.42
GA-2	130	1 434 000	115	164	1.43
GA-3	133	1 450 000	121	173	1.43
GB-1	114	1 758 000	108	163	1.51
GB-3	122	1 104 000	121	182	1.51
GB-4	133	688 000	127	193	1.51
GB-5	150	527 000	135	203	1.51
GC-1	104	2 350 000	109	156	1.43
GC-2	109	1 970 000	115	164	1.43
GC-3	114	1 807 000	121	173	1.43
GC-4	123	1 206 000	128	184	1.43
GC-5	129	782 000	136	194	1.43
GD-2	113	2 523 000	118	180	1.52
GD-5	129	869 000	140	211	1.51

It should be emphasized that SHSS and SCF_{HS} under fatigue loading were not correctly evaluated in the reference study. Since the nominal stress was calculated based on a flat web girder with identical dimensions. Consequently, this assumption results in higher hot-spot stresses and SCF_{HS} values compared to the actual structural behavior.

4.3 Refined Sub-model for ENS Evaluation

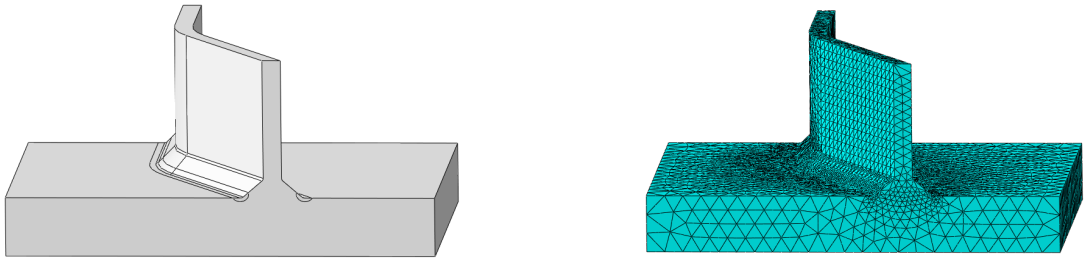
As discussed in the literature study, the ENS methodology offers a more precise description of the geometry of welded joints regarding geometrical discontinuity. Therefore, the finite element model generated for the ENS calculation includes all geometrical effects to account for the stress concentration on the local geometry.

4.3.1 Sub-modeling Procedure

To ensure that computations remain efficient, a sub-modeling technique is employed once again. Specifically, the localized sub-model is defined based on the region surrounding

the S-point 1 from which the effective notch stresses are extracted. In this sub-model, a fine mesh density is assumed near the weld, whereas a coarser mesh density is utilized at distances farther from the weld. The mesh to be applied here follows the recommendations in EN1993-1-14 [3] as well as the IIW recommendation [2].

Proper boundary conditions are set up to establish connections between the SHSS- and the ENS-model and are based on the displacement field. At the same time, it is also possible to use sectional forces or stresses to perform such work. The methodology of evaluating effective stress in notches using the sub-modeling approach is shown schematically in Figure 4.1. Furthermore, the weld geometry satisfies the condition $\frac{2a_w}{h_w} \geq 1.15$, indicating that additional mesh refinement at the weld root is not required. Consequently, the adopted mesh density is considered sufficient to accurately capture the stress distribution without introducing unnecessary computational complexity, the mesh applied is illustrated in Figure 4.7.



Part	**Mesh**
Weld toe	0.15 mm
Local region around the weld toe	1 mm
Entire model	5 mm

Figure 4.7: Refined ENS finite element model and corresponding mesh sizes used in the analysis.

4.3.2 The Result Derived from the ENS

Now that the model is built up, the load should be updated to correspond to the loads presented in Table 4.1. The results obtained from this analysis will yield the ENS stresses

and the corresponding SCF_{ENS}. Subsequently the ENS stresses will be used to evaluate the fatigue assessment conducted in the reference study.

Table 4.6: Effective notch stress ranges- and stress concentration factors for the CWG specimens. N_f are obtained from the reference study [20].

Specimen	$\Delta\sigma_{\text{nom,test}}$ **[MPa]**	N_f **(cycles)**	$\Delta\sigma_{\text{nom,FE}}$ **[MPa]**	$\Delta\sigma_{\text{ENS,FE}}$ **[MPa]**	$\mathbf{SCF_{ENS,FE}}$ **[-]**
GA-1	127	1 373 000	109	261	2.40
GA-2	130	1 434 000	115	273	2.38
GA-3	133	1 450 000	121	291	2.40
GB-1	114	1 758 000	108	320	2.96
GB-3	122	1 104 000	121	357	2.96
GB-4	133	688 000	127	377	2.96
GB-5	150	527 000	135	398	2.96
GC-1	104	2 350 000	109	268	2.46
GC-2	109	1 970 000	115	282	2.46
GC-3	114	1 807 000	121	299	2.46
GC-4	123	1 206 000	128	316	2.46
GC-5	129	782 000	136	333	2.46
GD-2	113	2 523 000	118	281	2.37
GD-5	129	869 000	140	331	2.37

It is evident from Table 4.6 that the highest values of SCF_{ENS}, approximately 2.96, are obtained for the GB girders among the tested specimen groups. This indicates that the GB geometry gives rise to larger stress concentrations compared to the other girders. Such behavior is expected, since the GB group is characterized by the largest corrugation angle and the smallest bending radius, which is consistent with the findings reported in the literature review. In contrast, the lowest SCF_{ENS} values are observed for the GD specimens.

4.4 Discussion of the Multi-level FE Approach

The proposed multi-level finite element method appeared to be a feasible and dependable tool for the estimation of fatigue damage of corrugated web girders. This approach allows to considerably reduce computation time and at the same time retain accuracy in

important regions.

The global model reproduces mainly the structural behavior of the girders, including their rigidity, nominal stress distribution, and deflection pattern. In order to estimate the precision of the model, the comparison with beam theory was performed. Deviations between the results of these two approaches are less than 3%, which proves that the model boundary and loading conditions used, as well as element formulation, can reflect the experimental data sufficiently accurately. It is crucially important because all inaccuracies will be automatically passed further on to local models.

The employment of sub-modeling for the analysis of SHSS results in obtaining a much more accurate stress field around the weld toe in comparison with the results that could be achieved using only the global model. Indeed, due to their high dependence on stress gradients and extrapolation distances, hot-spot stresses require finer local meshes for the successful extraction of results. The carried out mesh sensitivity analysis indicates that the chosen mesh density leads to converged and computationally inexpensive results. In addition, the comparison between linear and quadratic extrapolation methods shows that the latter performs better due to its inherent nonlinearity near the weld toe.

The SHSS calculation method also provides another example of the limitation of structural stress analysis techniques. While the structural stress concentration due to joint geometry is indeed captured using the technique, the resulting stresses are highly sensitive to the specific stress extraction location and the corresponding extrapolation process used. It becomes apparent when studying the GB-1 specimen where greater differences have been noticed between experimental and analytical stresses. As stated in the reference study, even small shifts in strain gauge locations from the weld toe could produce an impact on the final value of stress obtained.

It is worth noting that ENS analysis corresponds to yet another level of refinement, since the local notch geometry of the weld toe is considered directly within the numerical simulation. In particular, the SHSS analysis estimates the structural stresses without considering the notch zone at all; however, the ENS analysis takes into account the local stress magnification due to the radius of the weld toe. Therefore, the results of the ENS analysis exhibit substantially larger values of the stress concentration factor compared to those obtained from the SHSS analysis. Such findings are rather predictable, since the ENS analysis incorporates both the stress concentration factor and the effect of the local notch.

One notable strength associated with the modeling approach employed lies in the

appropriate scaling for each of the modeling levels used. For instance, the global modeling level will address the structural mechanics of the entire girder system, while the SHSS level will examine the stress redistribution due to the structural system near the weld joint, with the ENS addressing the notch stresses responsible for causing cracks. This allows for a realistic approach to fatigue analysis without the computational complexities involved when dealing with a single modeling level.

However, some constraints exist with respect to the used modeling technique. The analysis was carried out using linear-elastic material characteristics as well as simplified weld geometries, while the actual joints have some geometric irregularities, residual stresses, and manufacturing defects, which might affect fatigue strength.

4.5 Chapter Summary

The analysis of the fatigue behavior of corrugated web girders, based on the application of nominal stress, structural hot-spot stress, and effective notch stress techniques, shows significant differences in the ability of these approaches to describe the fatigue behavior of such structures.

The obtained S–N curves show that the nominal stress technique yields the highest scatter in the data. Corresponding to a relatively poor correlation with a coefficient of determination of $R^2 = 0.352$ as shown in Figure 4.8. This indicates that the nominal stress method is insufficient to fully account for the complexities associated with local stress redistribution caused by the corrugated web geometry and welded connections. Since local stress concentrations are not explicitly considered, considerable scatter is observed between the different specimen types.

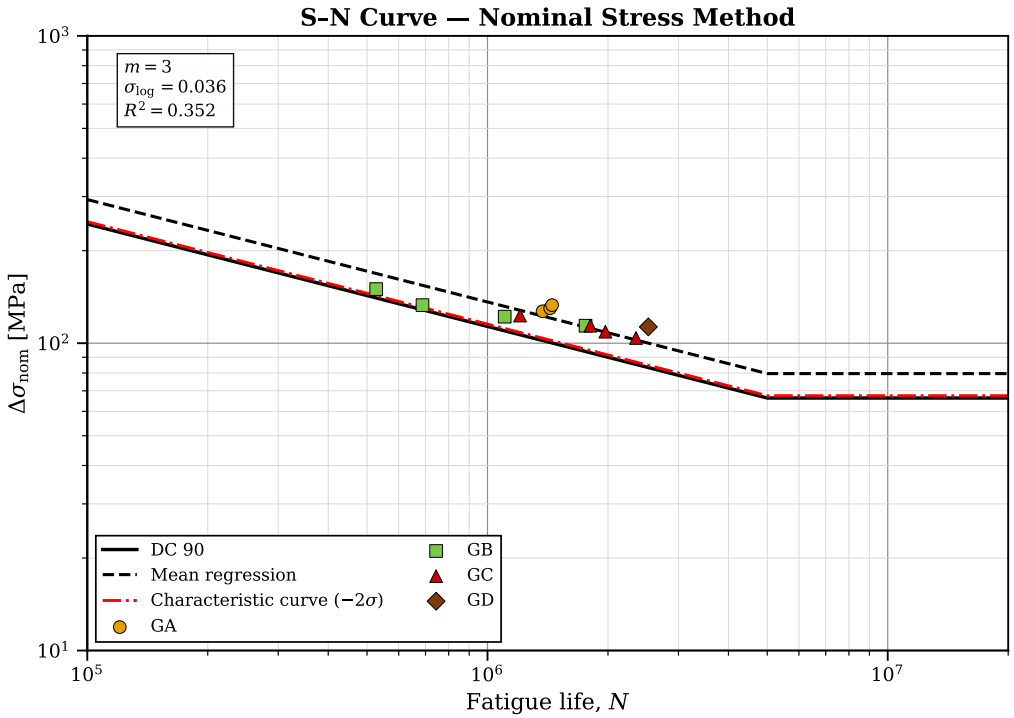


Figure 4.8: S–N curve for nominal stress method compared with the DC 90 design curve, mean regression curve, and characteristic curve.

In contrast, the implementation of the structural hot-spot stress method significantly improves the fatigue correlation, resulting in a coefficient of determination of $R^2 = 0.559$ as illustrated in Figure 4.9. By accounting for the structural stress concentration at the weld toe, the SHSS approach produces substantially less scatter compared to the nominal stress technique. Nevertheless, some degree of scatter remains due to the sensitivity of the extrapolated stresses to the selected stress extrapolation procedure, as well as the inability of the SHSS method to account for the plate thickness effect, t_f.

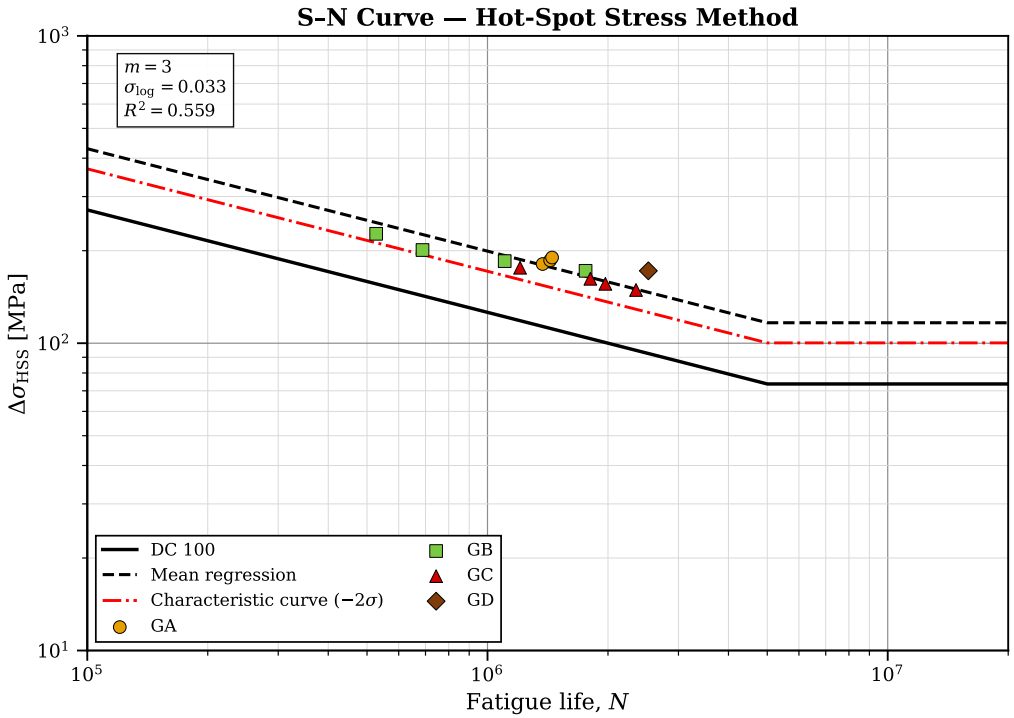


Figure 4.9: S–N curve for the Hot-spot method compared with the DC 100 design curve, mean regression curve, and characteristic curve.

Among the considered methods, the effective notch stress approach provides the most accurate correlation of the specimens. A coefficient of determination of $R^2 = 0.867$ is achieved, as illustrated in Figure 4.10. The significant reduction in scatter demonstrates the superior capability of the ENS approach to estimate the stresses responsible for fatigue crack initiation. As previously discussed, this high level of accuracy is achieved by accounting for the weld notch geometry and the associated local stress concentration effects. Consequently, the ENS model provides a physically more representative description of the fatigue-critical region.

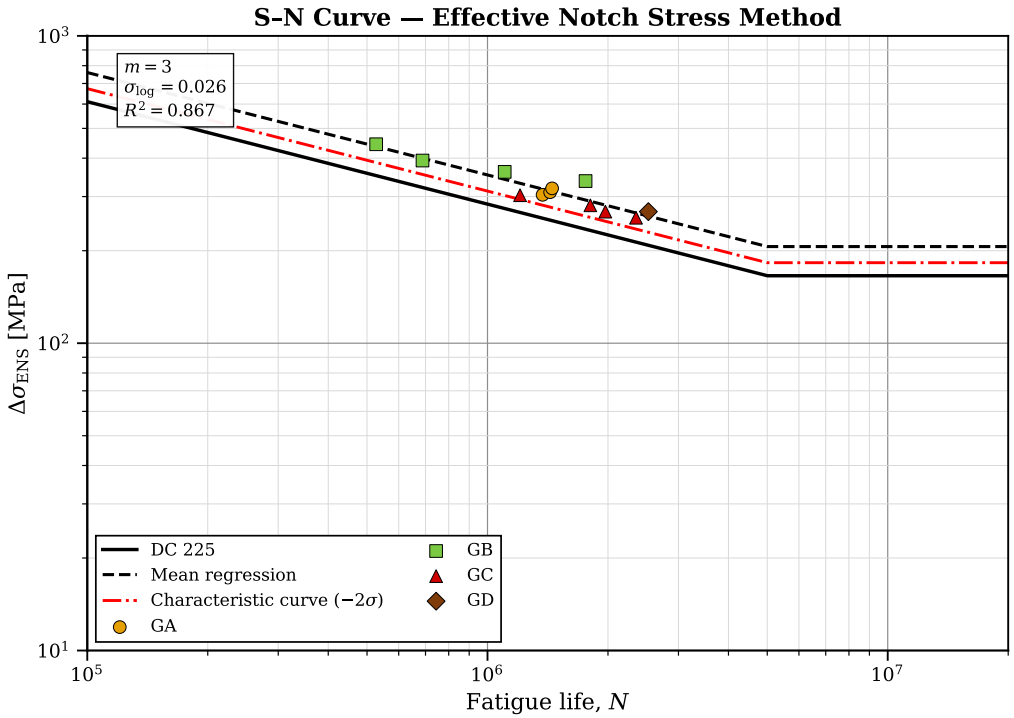


Figure 4.10: S–N curve for the effective notch stress method compared with the DC 225 design curve, mean regression curve, and characteristic curve.

Chapter 5

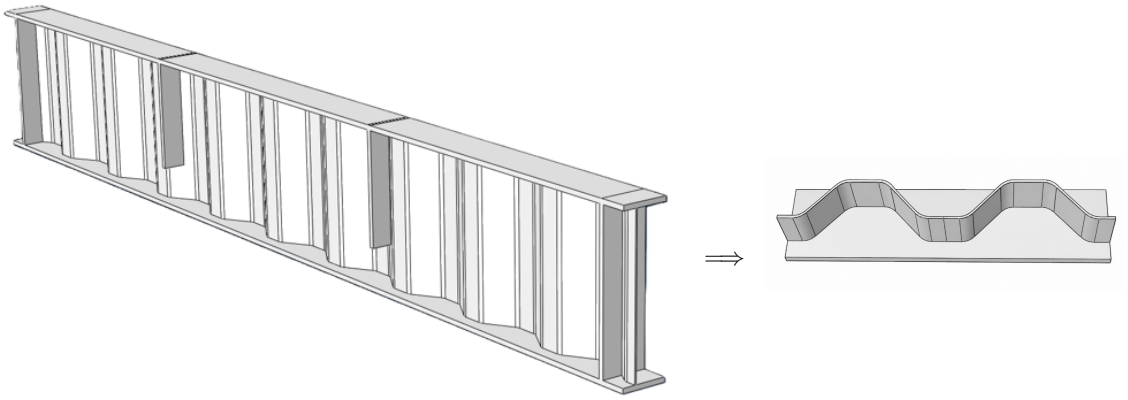
Parametric Modeling strategy

Since the ENS method was found to be the most accurate fatigue assessment method irrespective of the number of modeled or tested girders, it will be implemented in an automated Python–Abaqus script to perform a comprehensive parametric study. Where an extended range of corrugated design parameters is investigated.

5.1 Transforming from CWG to CWT

The parametric study required a large number of finite element analyses. Using fully corrugated web girders for all cases would lead to very high computational costs and long analysis times. To overcome this problem, a simplified tension specimen was developed. The specimen represents the fatigue-critical flange-to-web connection while reducing the required analysis time. It was designed to reproduce the local stress behavior of the full girder, especially near the S-point, where fatigue cracks are most likely to start.

The simplified specimen consisted of two corrugation waves representing the constant-moment region of the girder. This configuration was chosen to reduce the influence of the boundary conditions while maintaining the stress redistribution caused by the corrugated web geometry. The specimen also incorporates the same flange-to-web welded detail and corrugation profile as the full-girder models by initially adopting the same geometric configurations as the GA, GB, and GC girders according to Table 4.2. The two-wave CWT model shown in Figure 5.1 will first be verified and validated against these configurations before being implemented in the automated Python script.



(a) Global model. **(b)** Two-wave segment.

Figure 5.1: Simplification of the global finite element model into the representative two-wave girder segment used in the parametric study.

The applied tensile force was adjusted so that the nominal longitudinal stress in the flange matched the nominal stress from the full girder in the constant-moment region. The nominal stress was evaluated at the upper surface of the tension flange along the middle of the parallel fold. This location showed a relatively stable stress distribution and was less affected by local stress concentrations and boundary condition effects.

5.1.1 Selection of Nominal Stress Location

In corrugated web girders, the nominal stress is not distributed uniformly along the flange. This is mainly due to the transverse bending induced by the corrugated web geometry. As a result, the stress level varies at different locations along the corrugation profile, particularly near the inclined folds and the S-point regions. Since the stress concentration factor (SCF) is defined as the ratio between the local stress and the nominal stress, the selected location for evaluating the nominal stress can influence the calculated SCF values.

To study this effect, the longitudinal stress distribution along the upper surface of the tension flange was evaluated at several locations as illustrated in Figure 5.2. The results showed higher stress values near the inclined folds and the S-point regions. This behavior is caused by local transverse bending effects generated by the shear flow in the corrugated web.

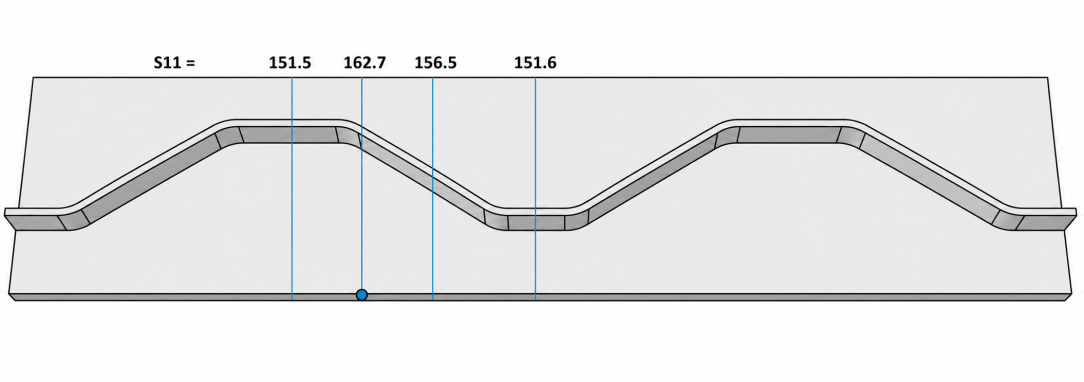


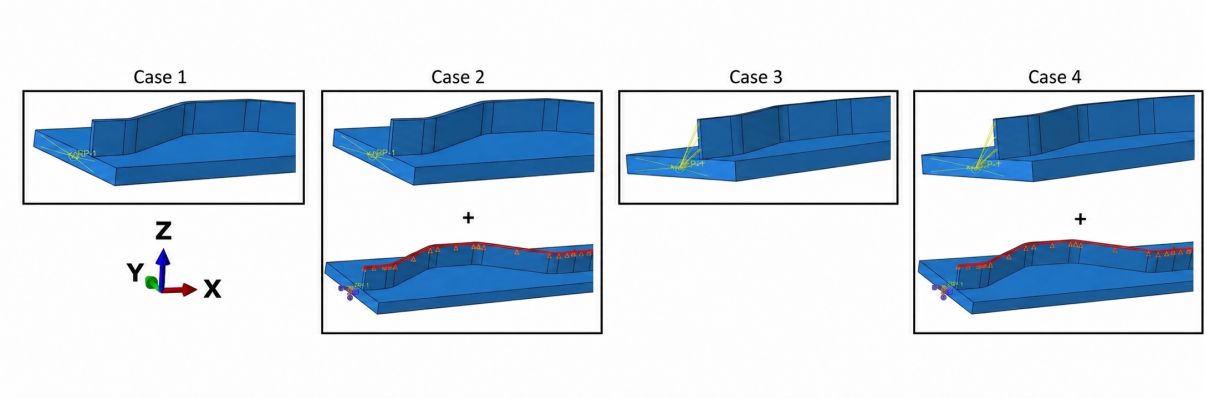
Figure 5.2: Location of the nominal stress evaluation along the flange of the two-wave segment. The stress values (S_{11}) at several positions along the flange are shown to illustrate the non-uniform stress distribution caused by the corrugated web geometry.

Based on these results, the nominal stress location used in this study was selected in the middle of the parallel fold in the central region of the specimen. This location showed a relatively stable stress distribution and was less affected by local stress concentrations and boundary condition effects. It was also located sufficiently far from the specimen ends, making it suitable for obtaining a representative nominal stress for the SCF evaluation.

5.1.2 Boundary Condition Investigation

The boundary conditions applied to the simplified tension specimen can influence the stress distribution in the flange-to-web connection region. Therefore, a boundary condition study was conducted to identify the configuration that best reproduces the behavior of the full corrugated web girder.

Four different boundary condition configurations were investigated. In the first configuration, only the flange was restrained. In the second configuration, the flange and the upper edge of the corrugated web were restrained in the out-of-plane direction (Z-direction). The third configuration included restraint of the flange and the side surface of the web. In the fourth configuration, the flange, the side surface of the web, and the upper edge of the web were all restrained. The investigated boundary condition configurations are presented in Figure 5.3.



Case	Boundary condition description	SCF
1	Flange restraint only	2.35
2	Flange restraint and top web restrained in the out-of-plane (Z) direction	2.37
3	Flange restraint and web side-surface restraint	2.35
4	Combined flange restraint, web side-surface restraint, and top-web restraint	2.35

Figure 5.3: Investigated boundary-condition configurations considered in the simplified tension specimen study based on the geometry of girder GA-1.

The resulting stress distributions and $\mathrm{SCF}_{\mathrm{ENS}}$ values were compared with those obtained from the full girder model. Among the investigated configurations, the second boundary condition configuration exhibited the closest agreement with the full girder results. It provided a realistic representation of the stress distribution while avoiding excessive local restraint effects. Therefore, this configuration was selected and adopted in all subsequent parametric analyses.

5.1.3 Selection of Web Height

The height of the corrugated web in the simplified tension specimen may also influence the local stress distribution in the flange-to-web connection region. Due to the accordion effect, the corrugated web is typically assumed to make a negligible contribution to the global bending stiffness of the girder when classical beam theory is employed. However, in the finite element model, a portion of the corrugated web may still contribute to the overall structural stiffness. Consequently, a dedicated sensitivity study was conducted to evaluate the effect of web height and to identify an appropriate value for the subsequent parametric study. Three different web heights were considered as shown in Figure 5.4.

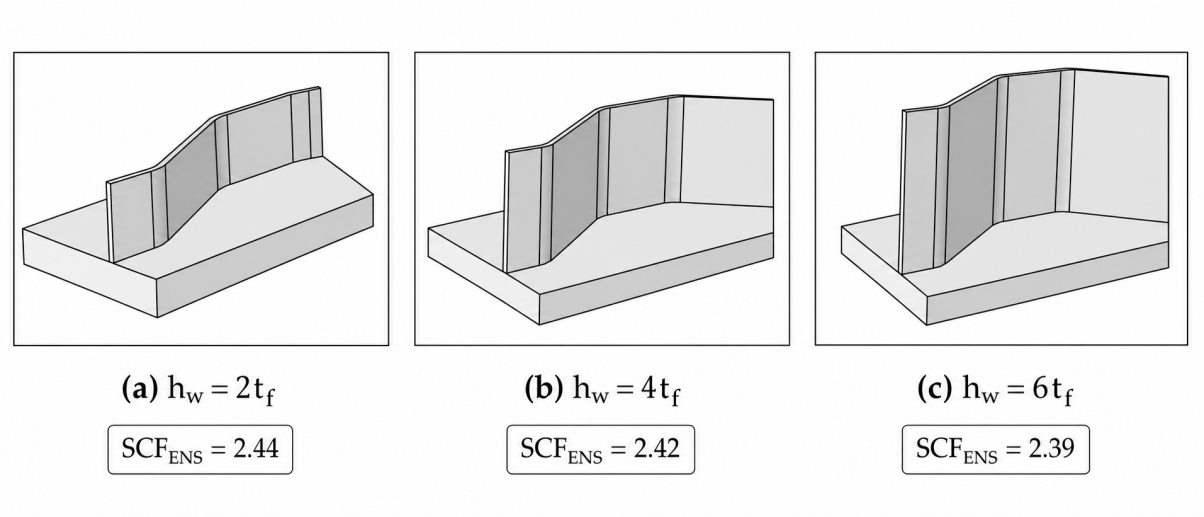


Figure 5.4: Investigated web heights considered in the simplified tension specimen study.

The results showed that increasing the web height reduced the influence of the specimen boundaries and improved the stress distribution near the S-point, resulting in a closer agreement with the full girder model. Among the investigated cases, the specimen with a web height of $h_w = 6t_f$ exhibited the best agreement with the full girder results. Based on these findings, a web height of $h_w = 6t_f$ was adopted for all subsequent numerical analyses in the parametric study.

5.1.4 Validation against the Full Girder

The accuracy of the simplified tension specimen was assessed by comparing its stress results with those obtained from the corresponding full girder models. The comparison included both the structural hot-spot stress (SHSS) and the effective notch stress (ENS) at the fatigue-critical S-point. Furthermore, the corresponding stress concentration factors, SCF_{SHSS} and SCF_{ENS}, were evaluated for both modeling approaches.

The results demonstrated excellent agreement between the simplified tension specimen and the full girder model. For the SHSS evaluation, the difference in SCF_{SHSS} was less than 5% for all investigated specimens. Similarly, the ENS results showed only minor deviations, resulting in closely matching values of SCF_{ENS}. These findings indicate that the simplified tension specimen can accurately reproduce the local stress behavior at the flange-to-web connection region while significantly reducing the computational effort compared to the full girder model. A comparison of the results obtained from the simplified tension specimen and the full girder model is presented in Tables 5.1 and 5.2.

Table 5.1: Comparison between the simplified tension specimen and the full girder for structural hot-spot stress (SHSS) evaluation.

Specimen	Nominal Stress		SHSS		SCF_{HSS}		Difference
	CWT	Full Girder	CWT	Full Girder	CWT	Full Girder	(%)
	σ_n (MPa)	σ_n (MPa)	σ_{HS} (MPa)	σ_{HS} (MPa)	SCF_{HSS}	SCF_{HSS}	
GA-1	162.40	162.62	220	230.6	1.35	1.42	4.82
GB-1	161.90	162.62	235	244.5	1.45	1.51	4.03
GC-1	163.20	162.62	226	233.4	1.38	1.43	3.26

Table 5.2: Comparison between the simplified tension specimen and the full girder for effective notch stress (ENS) evaluation.

Specimen	Nominal Stress		ENS		SCF_{ENS}		Difference
	CWT	Full Girder	CWT	Full Girder	CWT	Full Girder	(%)
	σ_n (MPa)	σ_n (MPa)	σ_{ENS} (MPa)	σ_{ENS} (MPa)	SCF_{ENS}	SCF_{ENS}	
GA-1	162.40	162.62	384	390	2.36	2.40	1.50
GB-1	161.90	162.62	475	481	2.93	2.96	0.89
GC-1	163.20	162.62	420	400	2.57	2.46	4.41

5.2 Parametric Study Configuration

After the simplified tension specimen had been developed and validated, a parametric study was conducted to investigate the influence of the main geometric parameters on the stress concentration factor (SCF). The investigated parameters included the corrugation angle (θ), corrugation depth (a_3), flange width (b_f), flange thickness (t_f), web thickness (t_w), bend radius-to-web thickness ratio (R/t_w), and parallel fold length (a_1).

The investigated parameter ranges are summarized in Table 5.3. The selected values cover a wide range of practical corrugated web girder geometries, the full modeled specimens dimensions are presented in Appendix E. The bend radius was defined through the ratio R/t_w, for which three values were considered: 2, 6, and 10. These values were selected based on the recommendations provided in EN 1993-1-8 [30] for welding in cold-formed zones.

Table 5.3: Geometrical parameters and investigated ranges used in the parametric study.

Parameter	Symbol	Range
Corrugation geometry		
Corrugation angle	θ	30°, 45°, 60°, 70°, 75°
Corrugation depth	a_3	50, 100, 200 mm
Cross-sectional dimensions		
Flange width	b_f	100, 200, 400, 600 mm for $a_3 = 50$ mm 200, 400, 800, 1200 mm for $a_3 = 100$ mm 400, 800, 1200 mm for $a_3 = 200$ mm
Flange thickness	t_f	5, 10, 20 mm 10, 20, 40 mm 20, 40, 60 mm 40, 60, 80 mm
Web thickness	t_w	4, 8, 10 mm for $a_3 = 50$ mm 4, 8, 12, 16 mm for $a_3 = 100$ mm and $a_3 = 200$ mm
Local geometry parameters		
Bend radius-to-web thickness ratio	R/t_w	2, 6, 10
Parallel fold length	a_1	200, 600 mm

5.3 Automated Multi-level FE Workflow

To handle the large number of simulations included in the parametric study, an automated finite element workflow was developed using Python scripting in Abaqus. The workflow automatically generated the finite element models, created the meshes, applied the boundary conditions, submitted the analysis jobs, extracted the results, and stored the data for all investigated cases.

The numerical analyses were performed using a multi-level sub-modeling approach consisting of three finite element models. First, a global model was used to determine the nominal stress distribution and the overall structural response. Subsequently, a local sub-model was generated from the global model for the structural hot-spot stress (SHSS)

evaluation. Finally, a refined sub-sub-model was created from the SHSS model for the effective notch stress (ENS) evaluation.

The displacement field obtained from each parent model was automatically transferred to the corresponding sub-model through sub-modeling boundary conditions. After completion of the analyses, the nominal stress, SHSS, and ENS results were extracted automatically. The corresponding stress concentration factors (SCFs) were then calculated for both the SHSS and ENS approaches.

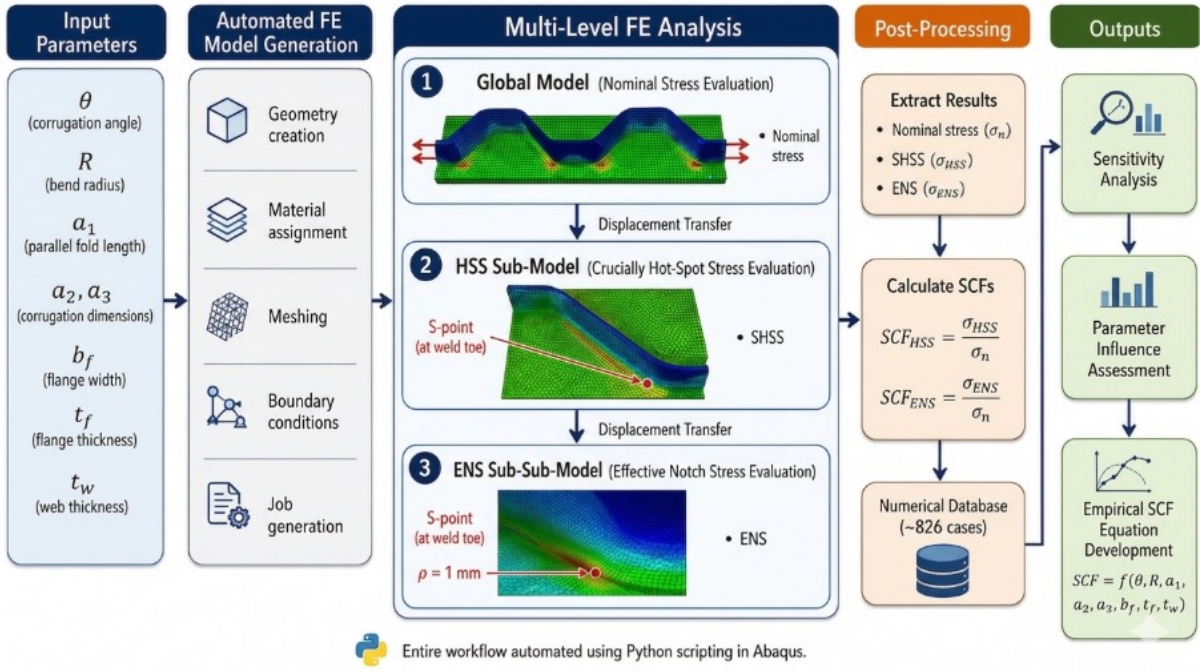


Figure 5.5: Overview of the automated workflow adopted in the parametric study.

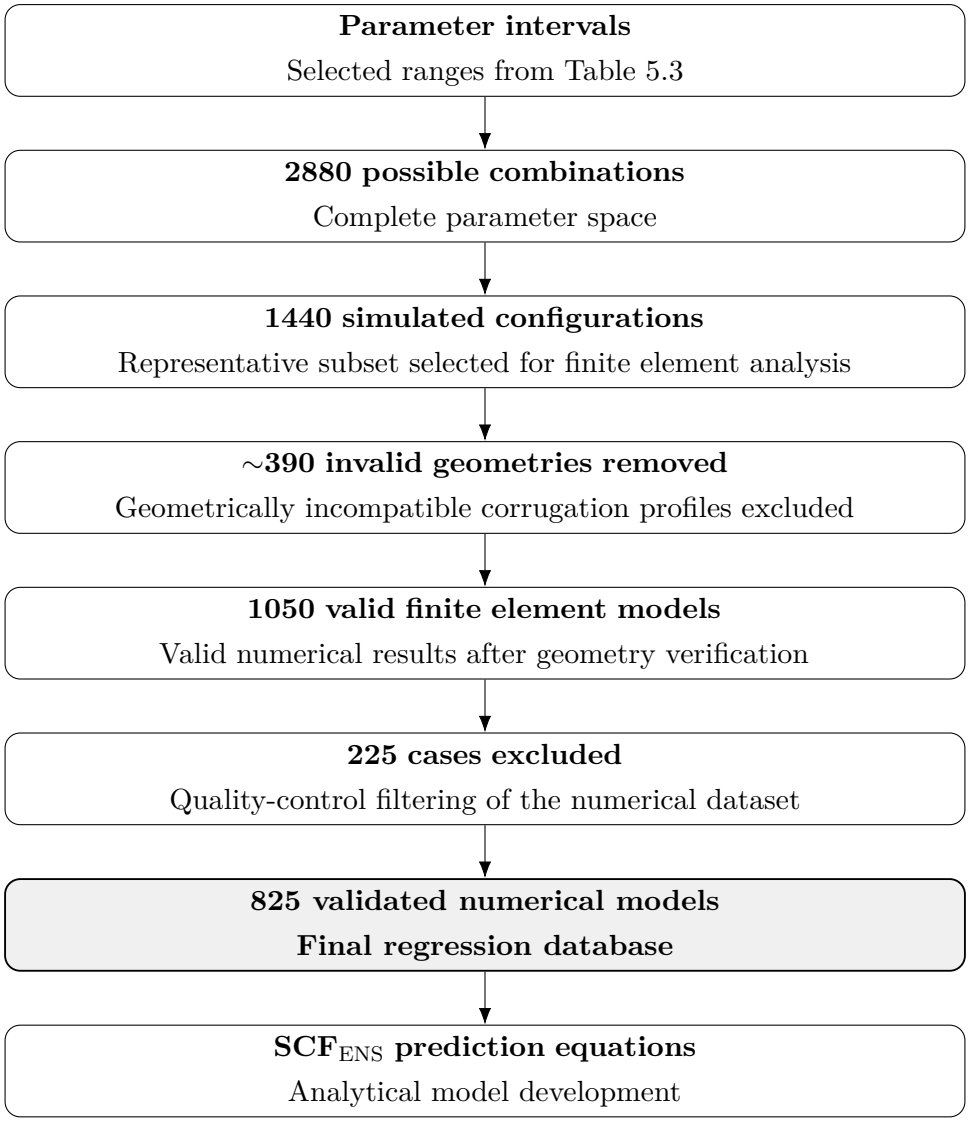
A schematic illustration of the automated workflow employed in this study is presented in Figure 5.5. The developed workflow enabled the efficient evaluation of a large number of numerical cases while maintaining a consistent modeling and post-processing procedure throughout the entire parametric study.

5.3.1 Database of Numerical Results

Based on the selected parameter combinations in Table 5.3, a total of 2880 numerical cases were generated automatically using an Abaqus scripting framework. Despite the fact that the complete parameter space consisted of 2880 possibilities, only 1440 possibilities were studied. This was necessary in order to reduce the computational expenses and time while covering a representative subset of the investigated range of parameters. Since the objective of the study was to identify the influence of the governing geometric parameters

and establish general predictive relationships, a reduced but uniformly distributed sample set was considered sufficient.

Approximately 350 cases were excluded as certain parameter combinations resulted in geometrically incompatible corrugation profiles. Consequently, the final database consisted of approximately 1050 valid numerical models covering a wide range of corrugated web geometries and structural configurations. For each valid numerical case, the nominal stress, and effective notch stress (ENS) were extracted from the finite element models. Based on these results, the corresponding stress concentration factors SCF_{ENS} were obtained. It should be noted that, at this stage, only results associated with incomplete geometric corrugation profiles have been excluded. Furthermore, during the subsequent statistical analysis and regression fitting process, additional results are identified and excluded as outliers as described in Figure 5.6.



Stage	Number of cases
Possible parameter combinations	2880
Simulated configurations	1440
Invalid geometries removed	∼390
Valid finite element models	1050
Quality-control filtering	225
Final regression database	**825**

Figure 5.6: Overview of the numerical database generation process, including parameter selection, finite element analysis, geometry verification, quality-control filtering , and development of the final regression database for SCF_{ENS} prediction.

Chapter 6

Results of the Parametric Study

To investigate the influence of the main geometric parameters on the effective notch stress concentration factor (SCF_{ENS}), a representative set of finite element models was analysed. The selected cases covered the minimum and maximum values of the investigated parameters while maintaining a manageable number of models. The parameters considered were the weld-toe radius ratio (R/t_w), corrugation angle (θ), corrugation depth ratio (a_3/b_f), and shape ratio (a_1/a_4).

6.1 Effect of Corrugation Bend Radius-to-Web Thickness Ratio (R/t_w)

The effect of the ratio between the bend radius and the web thickness (R/t_w) on the effective notch stress concentration factor (SCF_{ENS}) for varying corrugation angles and depth ratios is illustrated in Figure 6.1. In all analyzed cases, the value of SCF_{ENS} is found to decrease when increasing the R/t_w ratio from 2 to 10. This demonstrates that increasing the bend radius reduces the stress concentration in the corrugation fold region. It can be concluded that with the increase of the bend radius, the notch stresses are reduced due to a smoother geometric transition that allows more gradually stress distribution.

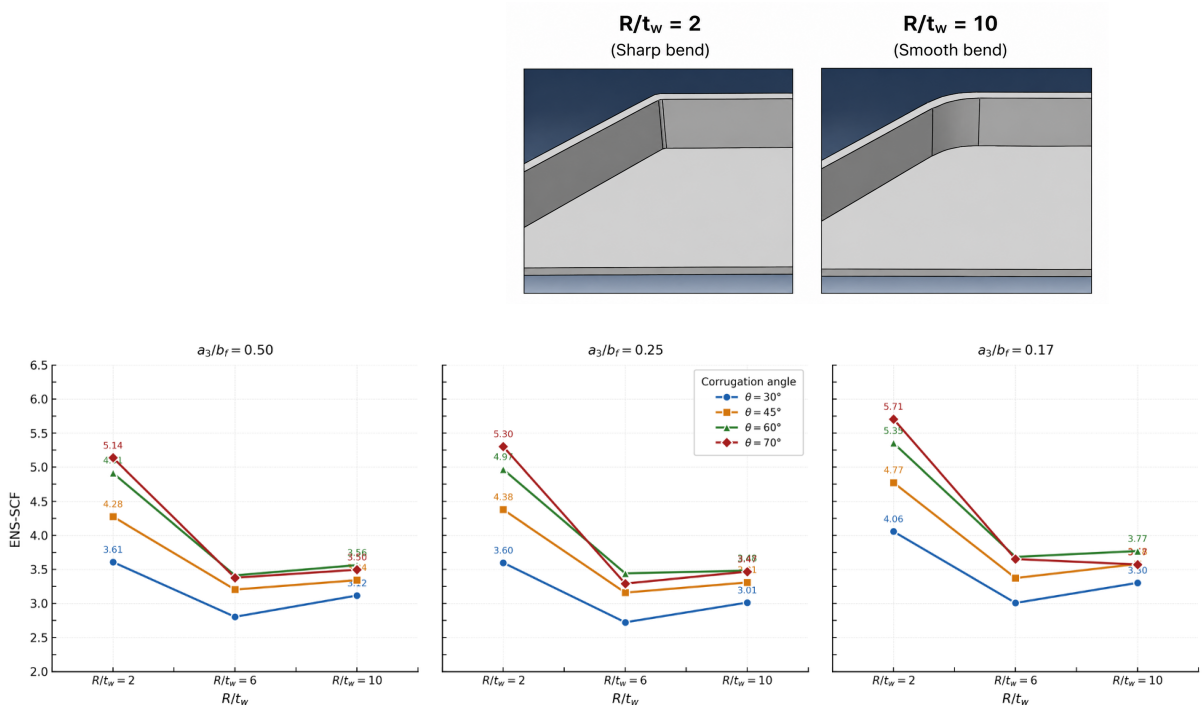


Figure 6.1: Effect of the bend radius-to-web thickness ratio (R/t_w) on SCF_{ENS}.

The same general trend is observed for all investigated corrugation angles and depth ratios. The influence of R/t_w becomes more noticeable at larger corrugation angles and smaller depth ratios, where the difference between sharp and smooth bends is greater. Overall, the results show that the bend radius-to-web thickness ratio is one of the most important geometric parameters affecting the (SCF_{ENS}).

6.2 Effect of Corrugation Angle (θ)

Figure 6.2 shows the effect of the corrugation angle (θ) on the effective notch stress concentration factor (SCF_{ENS}) for different values of R/t_w and a_3/b_f. In all investigated cases, SCF_{ENS} increases as the corrugation angle increases from 30° to 70°.

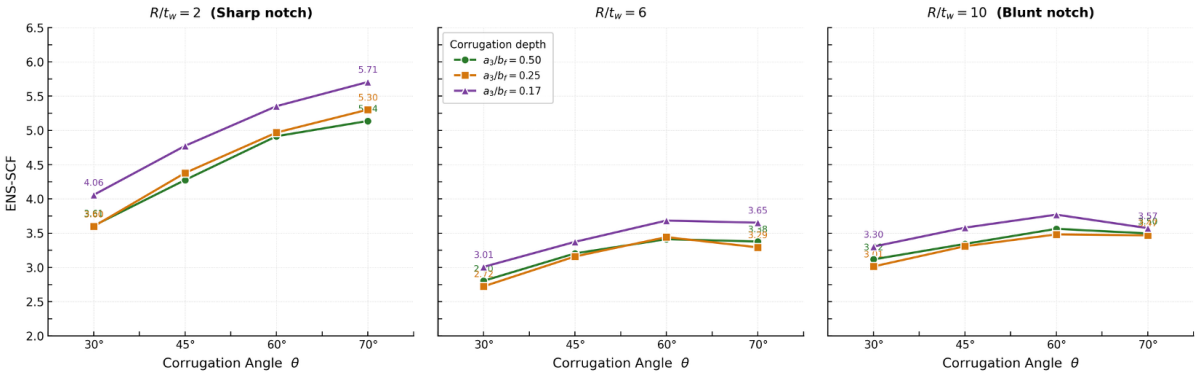


Figure 6.2: Effect of the corrugation angle (θ) on $\mathrm{SCF}_{\mathrm{ENS}}$.

The effect of the corrugation angle is more pronounced for small values of R/t_w, where the highest SCF_{ENS} values are obtained. As R/t_w increases, the influence of the corrugation angle becomes smaller, although the same increasing trend can still be observed. This suggests that the corrugation angle and bend radius ratio jointly affect the local stress concentration. This trend can be explained by the geometry of the corrugated web. Larger corrugation angles produce steeper inclined folds, which lead to a more concentrated transfer of stresses through the web. As a result, the local bending effects near the web-to-flange connection increase, leading to higher notch stresses. The results indicate that the corrugation angle is once again one of the main parameters influencing SCF_{ENS}.

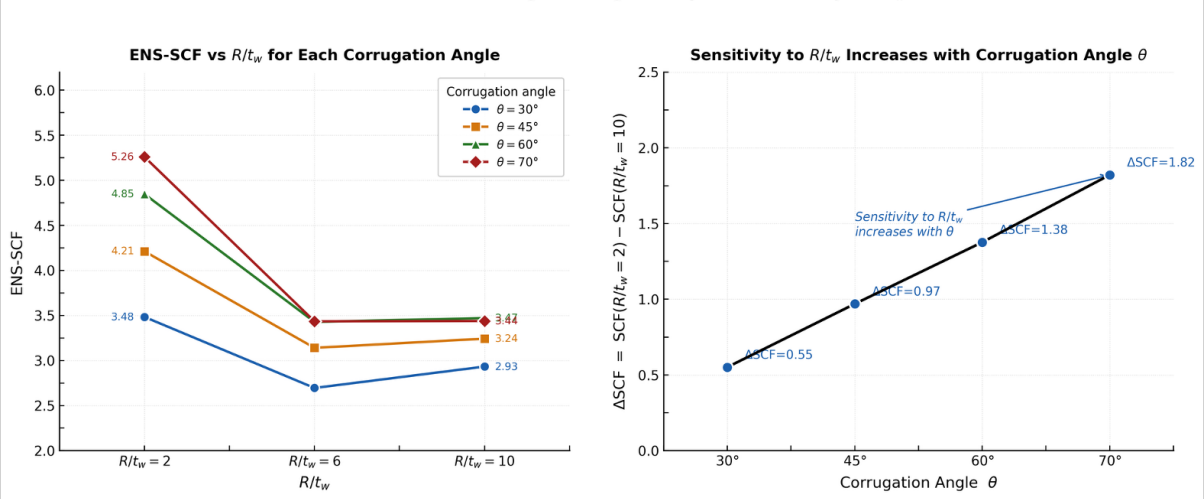


Figure 6.3: Interaction effect between the corrugation angle θ and bend radius ratio R/t_w on $\mathrm{SCF}_{\mathrm{ENS}}$. Increasing corrugation angle amplifies the sensitivity of the stress concentration factor to changes in bend radius.

From the obtained findings, one can see the obvious presence of the interaction between the corrugation angle θ and the bend radius ratio R/t_w. Despite the general trend for reducing the effective notch stress concentration factor with rising R/t_w value for all geometrical configurations. The efficiency of such changes is highly dependent on the corrugation angle. Namely, the sensitivity parameter $\Delta\mathrm{SCF} = \mathrm{SCF}(R/t_w = 2) - \mathrm{SCF}(R/t_w = 10)$ varies from 0.51 to 1.82 for the considered range of values of the corrugation angle as seen in Figure 6.3. which means that the sensitivity increases approximately by three times. Thus, one may conclude that the positive effect of rising the bend radius becomes more pronounced for steep corrugation geometries.

6.3 Influence of Secondary Parameters

While θ and R/t_w were identified as the dominant parameters governing SCF_{ENS}, additional geometric ratios may also influence the stress concentration behavior. Their effects are examined in the following sections.

6.3.1 Effect of the Depth Ratio (a_3/b_f) and the Shape Ratio (a_1/a_4)

Figure 6.4 shows the influence of the depth ratio (a_3/b_f) and shape ratio (a_1/a_4) on the effective notch stress concentration factor (SCF_{ENS}). For the depth ratio, SCF_{ENS} increases slightly as a_3/b_f decreases. For the shape ratio, SCF_{ENS} generally increases as a_1/a_4 increases. Similar trends are observed for all investigated cases, indicating that both parameters influence the local stress concentration behavior.

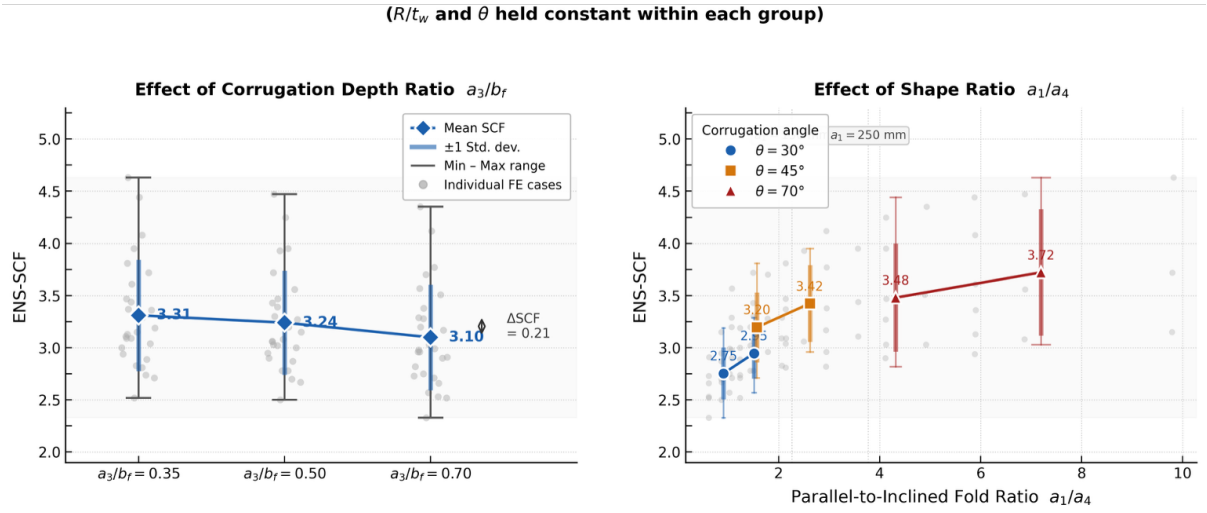


Figure 6.4: Effect of the secondary geometric ratios a_3/b_f and a_1/a_4 on $\mathrm{SCF}_{\mathrm{ENS}}$. The results indicate a comparatively smaller influence than that observed for the corrugation angle θ and bend radius ratio R/t_w.

Although these parameters affect SCF_{ENS}, their influence is considerably smaller than that of R/t_w and θ. Therefore, a_3/b_f and a_1/a_4 can be regarded as secondary parameters. Their primary role is to provide minor adjustments in the development of the predictive SCF_{ENS} equations.

6.3.2 Effect of the Parallel-to-Inclined Fold Ratio (a_1/a_2)

Figure 6.5 shows the effect of the parallel-to-inclined fold ratio (a_1/a_2) on SCF_{ENS} for different combinations of R/t_w and θ. Within each group of constant θ, the results indicate a decrease in SCF_{ENS} as a_1/a_2 increases. This behavior is related to the coupling between a_1/a_2 and the corrugation depth in the script.

$$\frac{a_1}{a_2} = \frac{a_1 \tan\theta}{a_3} \tag{6.1}$$

As shown in Eq. (6.1), higher values of a_1/a_2 in the present database are generally associated with smaller values of a_3. Because the corrugation depth ratio (a_3/b_f) also influences SCF_{ENS}, the observed trend reflects the combined effect of both parameters rather than the isolated influence of a_1/a_2.

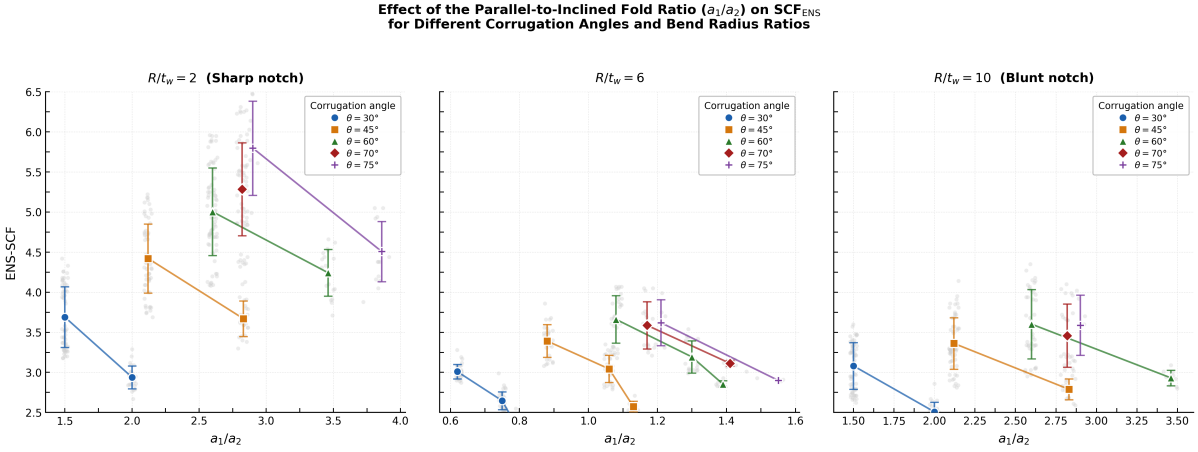


Figure 6.5: Effect of the parallel-to-inclined fold ratio a_1/a_2 on $\mathrm{SCF}_{\mathrm{ENS}}$ for different corrugation angles and bend radius ratios.

6.4 Relative Importance and Parameter Interaction

Figure 6.6 summarizes the relative influence of the investigated geometric parameters on the (SCF_{ENS}) based on the complete database of 825 finite element models. The results show that the bend radius-to-web thickness ratio (R/t_w) and the corrugation angle (θ) are the most important parameters, producing the largest changes in SCF_{ENS}. The mean ΔSCF values for these parameters are 1.35 and 1.06, respectively. Among the secondary parameters, the shape ratio (a_1/a_4) shows a moderate influence with a mean ΔSCF of 0.80, while the corrugation depth ratio (a_3/b_f) has the smallest effect, with a mean ΔSCF of 0.68. These results indicate that the overall stress concentration behavior is mainly controlled by the bend geometry and corrugation angle, while the remaining geometric ratios have a smaller effect on the SCF_{ENS}.

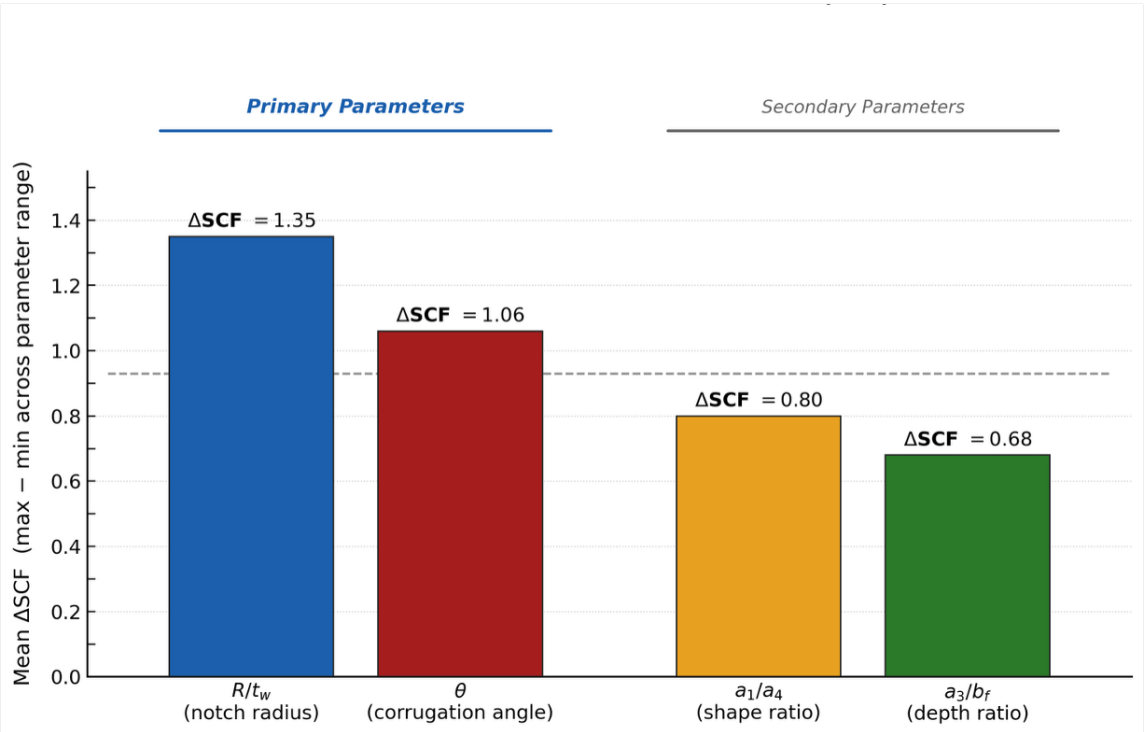


Figure 6.6: Relative influence of the investigated geometric parameters on $\mathrm{SCF_{ENS}}$, expressed as the mean parameter sensitivity ΔSCF.

6.5 Development of a Predictive $\mathrm{SCF_{ENS}}$ Formula

The results of the parametric study conducted in the above sections have confirmed that the fatigue response of corrugated web girders is strongly related to the interaction between the corrugation angle θ and the ratio between the bend radius and web thickness R/t_w. Although some other geometrical parameters were recognized as influencing the effective notch stress concentration factor, their effects were of minor importance compared to the two identified factors. This lays the foundation for building an empirical equations for predicting the effective notch stress concentration factor based on the geometrical parameters of the corrugated web girders. The goal of building such a predictive model is to derive the relationship between the nominal stress and effective notch stress that does not require finite element analysis.

The regression modeling was conducted on the above-mentioned numerical data provided by the database of Section 5.3.1. Following the elimination of invalid geometry and statistical outliers, 825 FE model results were retained for modeling purposes. It has been observed from the numerical data that the effect of the governing variables varies significantly among the bend radius ratios studied. Hence, a single generic equation could not be formed to cover the entire parameter range. Separate regression models were thus created for each bend radius ratio examined in the study; specifically, $R/t_w = 2$, R/t_w

= 6, and $R/t_w = 10$. The generic model is defined by Eq. 6.2 where C is a regression constant and λ_i represents parameter interaction terms accounting for the influence of the governing geometric variables. The complete definitions of the regression coefficients are provided in Table 6.1.

$$\mathrm{SCF}_{\mathrm{ENS}} = \exp\left(C + \sum_{i=1}^{n} \lambda_i\right) \tag{6.2}$$

Table 6.1: Definitions of the model constant C and the λ_i terms used in Eq. (6.2).

R/t_w	C	n	Terms λ_1–λ_3	Terms λ_4–λ_6
2	0.902	5	$\lambda_1 = 0.441\ \ln(R)\sin\theta$ $\lambda_2 = 0.028\ \ln(R)\ln(a_3/b_f)$ $\lambda_3 = -0.097\ \ln(R)\ln(a_1/a_2)$	$\lambda_4 = 0.009\ \ln^2(t_w/t_f)$ $\lambda_5 = 0.021\ \ln^3(a_1/a_2)$
6	0.549	5	$\lambda_1 = 0.290\ \ln(R)\sin\theta$ $\lambda_2 = -0.810\ \ln^3(a_1/a_2)$ $\lambda_3 = 0.060\ \ln(a_3/b_f)$	$\lambda_4 = -0.012\ \ln(R)\tan\theta$ $\lambda_5 = 0.057\ \ln(a_1/a_2)\ln(t_w/t_f)$
10	-0.709	6	$\lambda_1 = 0.328\ \ln(R)\sin\theta$ $\lambda_2 = 0.056\ \ln(a_1/a_2)\ln(a_3/b_f)$ $\lambda_3 = 0.069\ \ln(a_1/a_2)\tan\theta$	$\lambda_4 = -0.033\ \ln(t_w/t_f)\ln(b_f/t_f)$ $\lambda_5 = 1.330\ \cos\theta$ $\lambda_6 = 0.127\ \ln(a_3/b_f)\cos\theta$

The structure of the developed equation reflects the behavior observed throughout the parametric study. Terms involving θ and R appear consistently in all three formulations, confirming their dominant influence on stress concentration. In contrast, the geometric ratios a_1/a_2, a_3/b_f, t_w/t_f, and b_f/t_f primarily act as correction parameters that refine the prediction for specific geometric configurations.

The coefficient of determination for the proposed formulation ranges between $R^2 = 0.941$ and $R^2 = 0.951$ as can be seen in Figure 6.7. Demonstrating that more than 94% of the variation in the finite element results is captured by the proposed equations. Furthermore, the obtained RMSE values remain low, indicating a limited prediction error throughout the investigated parameter range.

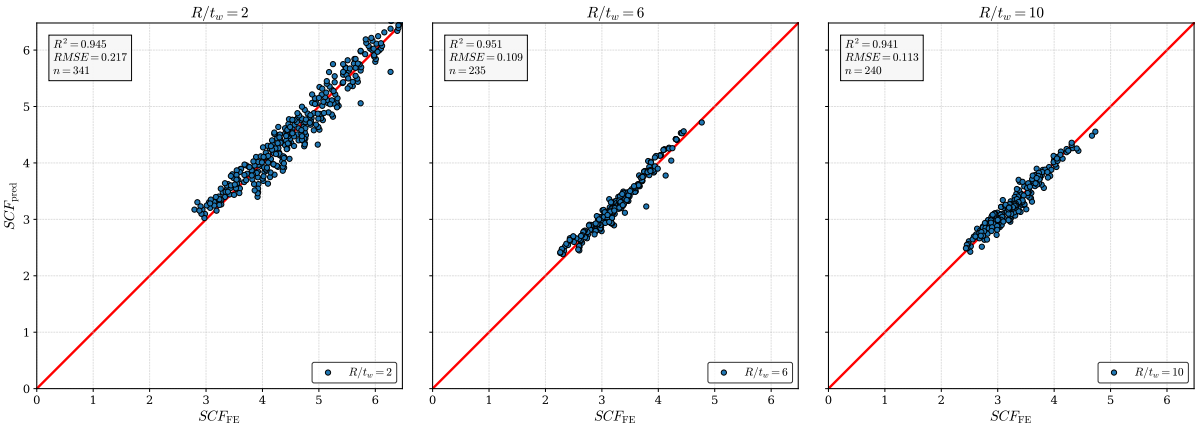


Figure 6.7: Regression performance of the developed model for specimens with $R/t_w = 2$, 6, and 10. The red line indicates perfect agreement between the finite element results and model predictions.

It is evident that the findings confirm the effectiveness of the formulated model in predicting the $\mathrm{SCF_{ENS}}$ values obtained using the advanced finite element analysis. Therefore, the new set of formulations will be useful for fatigue assessment in corrugated web girders. Through integrating the computed $\mathrm{SCF_{ENS}}$ and the nominal stress, the effective notch stress can then be determined without conducting multi-level FEM analysis.

Chapter 7

Discussion

The major aim of this research is to conduct a fatigue analysis on trapezoidal corrugated web girders using the Effective Notch Stress (ENS) technique, and come up with simplified equations of estimation for predicting stress concentrations associated with fatigue. By validation of existing experimental results, carrying out parametric studies, and formulating regression prediction models, a better understanding of the influence of the corrugation geometry on the fatigue performance has been achieved.

7.1 Applicability of the Effective Notch Stress Method

Validation showed that the ENS approach is able to correctly estimate the local stress concentrations in corrugated web girders at the regions where the folds transition occurs. The numerical predictions correlated well with existing experimental data on fatigue performance of corrugated web girders and correctly indicated the locations of the critical fatigue zones found earlier in various studies. Thus, the ENS approach may be safely used for fatigue assessments of corrugated web structures. In contrast to the traditional stress approaches that only consider the global stress distribution in the cross-section without taking into account any geometrical discontinuity or redistribution, the ENS approach has the advantage of considering the real behavior of the material in such types of structures where fatigue failures depend not on the global stress state but rather on the local stress concentration.

Successful validation carried out in this study confirms the suitability of the ENS approach for fatigue analysis of corrugated web girders and, in particular, for generating numerical databases for developing prediction models for simplified design purposes.

7.2 Influence of Geometric Parameters on Stress Concentration

The results obtained from the parametric study showed that all parameters studied affect the effective notch stress concentration factor; however, their degree of effect varies substantially. Out of the parameters considered, the corrugation angle, θ, and the bend radius ratio, R/t_w, are two of the most significant variables that determine the value of stress concentration factor.

The considerable effect of the corrugation angle on the SCF can be attributed to the changes in geometry as a result of an increasing angle. Since the corrugation fold angle increases, the transition between webs is rather abrupt, hence resulting into increased stress gradients. Therefore, as the value of θ increases, the magnitude of $\mathrm{SCF}_{\mathrm{ENS}}$ becomes larger. On the contrary, when the value of R/t_w increases, a smooth geometry transition between webs occurs. Since the bend radius increases, the geometry transition is relatively smooth. This translates into gradual distribution of stresses throughout the region of the fold angle. Hence, with an increase in R/t_w, the value of the effective notch SCF decreases.

Perhaps one of the most interesting results of the parametric study is the relation between the corrugation angle and the ratio of the bend radius to the wall thickness. While the increase in R/t_w leads to lower stress concentration in all the cases studied, the importance of this increase becomes more pronounced for greater values of the corrugation angle. The sensitivity factor, which is equal to the difference between the values of stress concentration coefficients corresponding to ratios of $R/t_w = 2$ and $R/t_w = 10$, shows a marked tendency to increase with increasing corrugation angle. This means that the effect of increased bend radius becomes increasingly significant for highly angled corrugations. Thus, from the standpoint of design considerations, increasing the bend radius proves to be highly beneficial in applications involving large corrugation angles.

Furthermore, the other geometric parameters like the corrugation depth, web thickness, flange width, and flange thickness affect the stress concentration factor. Though the influences of these geometric parameters are usually lower compared to θ and R/t_w, they still help to form the total stress pattern. Therefore, they remained in the regression equations used for predicting the stress concentration factor.

7.3 Evaluation of the Developed Prediction Models

One of the important outputs from the current study is the development of equations for predicting the SCF_{ENS} based on regression analysis using certain geometrical parameters. These models were developed using the numerical simulation data set comprising 825 cases of finite element analysis involving various practical geometries. The results of the prediction equations showed very good correlation with the results from finite element analyses. The value of the coefficient of determination was greater than 0.94 for all the cases analyzed, meaning that the models developed successfully explain the major trends of the stress concentration factor. Also, low root mean square errors indicate a good estimation of the results for the entire range of the considered parameters.

Thus, it can be stated that the selected geometric parameters sufficiently characterize the stress concentration factor behavior which is relevant to fatigue. Therefore, the prediction equations are a viable tool for evaluating the notch stresses in the corrugated web girders. Another benefit that comes from the developed formulas is their practical applicability. Engineers do not need to conduct complex finite element analysis with the creation of a geometrical model, meshing, and stress calculation procedures; it is enough to calculate the effective notch stress via the provided formulae.

Chapter 8

Conclusion

The present thesis investigates the fatigue response of trapezoidal corrugated web girders using the effective notch stress (ENS) approach. The study begins with the development of a detailed finite element (FE) model of a full corrugated web girder, which is subsequently validated against four experimental tests to ensure accuracy and reliability. Following validation, the model is simplified to a corrugated web beam representation while maintaining an equivalent stress concentration factor (SCF). A comparative evaluation of the Hot Spot Stress and Effective Notch Stress methods is then conducted to identify the most suitable approach for further analysis. Based on this selection, a comprehensive parametric study is performed over a wide range of geometrical parameters, and the results are used to formulate simplified equations for estimating fatigue-related stress concentrations, leading to the proposal of an $\mathrm{SCF}_{\mathrm{ENS}}$ equation.

It was shown that the Effective Notch Stress (ENS) method can accurately predict the fatigue failure location (S-point) in corrugated webs. It can therefore be concluded that the approach used in the present thesis allows a physically sound description of fatigue problems in corrugated web girders in comparison with the conventional nominal stress approach.

It was demonstrated from the conducted parametric study that the parameters θ (the corrugation angle) and R/t_w (the bend radius-to-web thickness ratio) influence the most the effective notch stress concentration factor, $\mathrm{SCF}_{\mathrm{ENS}}$. It was observed that increasing θ led to higher stress concentration, while increasing the ratio R/t_w resulted in lower stress concentrations due to smooth geometry transitions around the folds.

One key conclusion from this study relates to the interaction effect between θ and R/t_w. In addition to the general trend that increasing the ratio R/t_w lowers the stress concentration factor irrespective of geometry, it is noteworthy that the reduction effect becomes significant with higher angles of corrugation. Using the numerical data obtained

from the parametric study and based on the 825 valid finite element models, regression equations were derived for estimating the value of SCF_{ENS} as a function of geometric parameters. For each case, namely, $R/t_w = 2$, 6 and 10, separate predictive models were created due to the dependence of geometric parameters on bend radius ratio. These regression equations showed good agreement with finite element simulation results, with the values of coefficient of determination being higher than 0.94, and very small error in predictions for all considered cases. These equations allow the estimation of effective notch stresses directly from nominal stresses, thereby reducing the reliance on detailed finite element modeling. Consequently, the computational effort associated with fatigue assessment may be significantly decreased, while also supporting a more efficient evaluation process and facilitating design optimization.

8.1 Recommendations for Future Research

Further work should focus on expanding the use of this method into other types of geometries and different bend radius ratios. Experimental studies should be carried out using various configurations to ensure more accuracy of the equations presented. In addition, further studies need to consider the impact that manufacturing factors as imperfect welding joints, stress conditions, and tolerance conditions have on the efficiency of this method.

However, several aspects that limit the applicability and accuracy of the equations should be addressed. The first limitation concerns the geometry. The present work considered only trapezoidal corrugated webs with three fixed values of the ratio of the bend radius and wall thickness: $R/t_w = 2$, 6, and 10. Future investigations could explore realistic values of the bend radius and formulate a generalized model for all the cases.

The second issue is associated with the experimental validation. The current study relied on numerical analysis only, while previous experimental results confirm the accuracy of the formulations. However, further experiments may be needed to confirm the results obtained via simulation techniques.

The third limitation relates to neglecting certain factors affecting the fatigue properties. Specifically, no attention was paid to residual stresses, welding quality, manufacturing accuracy, and other similar considerations that can affect the material properties significantly in real-life conditions.

Finally, the current equations apply only to certain ranges of the parameters. As such,

caution should be exercised while applying the equations outside the investigated ranges.

Chapter 9

Statement on the Use of Artificial Intelligence

Tools of artificial intelligence (AI) were used while writing this thesis for the purpose of language polishing, text editing, advice on formatting, and code refinement. AI outputs were carefully reviewed, tested, and improved by the authors before their implementation in this thesis.

The scientific part of the thesis, including the formulation of research problems, methodology, FEM modeling, analyses, results interpretation, discussions, and conclusions, was produced and analyzed by the authors alone. The authors take complete responsibility for the reliability and originality of the scientific information contained in this thesis.

Bibliography

[1] *Eurocode 3: Design of steel structures – part 1-9: Fatigue*, CEN, 2023.

[2] M. Al-Emrani and M. Aygül, "Fatigue design of steel and composite bridges," Chalmers University of Technology, Gothenburg, Sweden, Tech. Rep. Report 2014:10, 2014.

[3] *Eurocode 3: Design of steel structures – part 1-14: Design assisted by finite element analysis*, CEN, 2021.

[4] M. Al-Emrani, "Steel structures," Chalmers University of Technology, Department of Architecture and Civil Engineering, Division of Structural Engineering, Gothenburg, Sweden, Tech. Rep. Report ACE 2023:1, 2023.

[5] M. F. Hassanein, Y. Shao, and M. Zhou, *Behavior and Design of Trapezoidally Corrugated Web Girders for Bridge Construction: Recent Advances*. Woodhead Publishing, 2022, ISBN: 9780323884372.

[6] M. Leblouba, A. S. Karzad, S. W. Tabsh, and S. Barakat, "Plated versus corrugated web steel girders in shear: Behavior, parametric analysis, and reliability-based design optimization," *Buildings*, vol. 12, no. 12, p. 2046, 2022. DOI: 10.3390/buildings12122046.

[7] Q. Inaam and A. Upadhyay, "Accordion effect in bridge girders with corrugated webs," *Journal of Constructional Steel Research*, vol. 188, p. 107 040, 2022. DOI: 10.1016/j.jcsr.2021.107040.

[8] F. Hlal, "Stainless steel corrugated web i-girders for composite road bridges," Ph.D. dissertation, Chalmers University of Technology, Gothenburg, Sweden, 2025.

[9] H. H. Abbas, R. Sause, and R. G. Driver, "Simplified analysis of flange transverse bending of corrugated web i-girders under in-plane moment and shear," *Engineering Structures*, vol. 29, pp. 2816–2824, 2007. DOI: 10.1016/j.engstruct.2007.01.006.

[10] B. Kövesdi, B. Jáger, and L. Dunai, "Stress distribution in the flanges of girders with corrugated webs," *Journal of Constructional Steel Research*, vol. 79, pp. 204–215, 2012. DOI: 10.1016/j.jcsr.2012.07.023.

[11] European Committee for Standardization (CEN), *Eurocode 3: Design of steel structures – part 1-5: Plated structural elements*, EN 1993-1-5:2005, Brussels, Belgium, 2005.

[12] A. Lindqvist and H. Nilsson, "Effective notch stress analysis of transverse attachments in steel bridges: A parametric fatigue life assessment," M.S. thesis, Chalmers University of Technology, Gothenburg, Sweden, 2016.

[13] F. Z. Akhlaghi, "Fatigue life assessment of welded bridge details using structural hot spot stress method: A numerical and experimental case study," M.S. thesis, Chalmers University of Technology, Gothenburg, Sweden, 2009.

[14] A. F. Hobbacher and J. Baumgartner, *Recommendations for Fatigue Design of Welded Joints and Components* (IIW Collection), 3rd ed. Cham, Switzerland: Springer, 2024, ISBN: 978-3-031-57666-9. DOI: 10.1007/978-3-031-57667-6.

[15] W. D. Pilkey, *Peterson's Stress Concentration Factors*, 2nd ed. New York: John Wiley & Sons, Inc., 1997, ISBN: 0-471-53849-3.

[16] Z. Wang, L. Tan, and Q. Wang, "Fatigue strength evaluation of welded structural details in corrugated steel web girders," *International Journal of Steel Structures*, vol. 13, no. 4, pp. 707–721, 2013. DOI: 10.1007/s13296-013-4012-z.

[17] S. A. Ibrahim, W. W. El-Dakhakhni, and M. Elgaaly, "Behavior of bridge girders with corrugated webs under monotonic and cyclic loading," *Engineering Structures*, vol. 28, pp. 1941–1955, 2006. DOI: 10.1016/j.engstruct.2006.03.026.

[18] B. Kövesdi and L. Dunai, "Fatigue life of girders with trapezoidally corrugated webs: An experimental study," *International Journal of Fatigue*, vol. 64, pp. 22–32, 2014. DOI: 10.1016/j.ijfatigue.2014.02.017.

[19] Z. Wang and Q. Wang, "Fatigue assessment of welds joining corrugated steel webs to flange plates," *Engineering Structures*, vol. 73, pp. 1–12, 2014. DOI: 10.1016/j.engstruct.2014.04.041.

[20] L. Tong, Z. Zhao, G. Zuo, H. Wang, and C. Pan, "Experimental study on fatigue behavior of trapezoidal corrugated-web girders based on t-section members," *Engineering Structures*, vol. 298, p. 117 078, 2024. DOI: 10.1016/j.engstruct.2023.117078.

[21] K. Anami, R. Sause, and H. H. Abbas, "Fatigue of web-flange weld of corrugated web girders: Influence of web corrugation geometry and flange geometry on web-flange weld toe stresses," *International Journal of Fatigue*, vol. 27, pp. 373–381, 2005. DOI: 10.1016/j.ijfatigue.2004.08.006.

[22] J. Xu, H. Sun, S. Cai, W. Sun, and B. Zhang, "Fatigue testing and analysis of i-girders with trapezoidal corrugated webs," *Engineering Structures*, 2019.

[23] X. Liu, Z. Xiong, Y. Liu, J. Li, and C. Liang, "Optimization and fatigue assessment of girders with large-dimension corrugated steel webs," *Structures*, vol. 64, p. 106 619, 2024. DOI: 10.1016/j.istruc.2024.106619.

[24] Z. Y. Wang, Q. Y. Wang, and R. J. Jiang, "Finite element based fatigue assessment of corrugated steel web beams in highway bridges," *International Journal of Civil Engineering*, vol. 13, no. 4A, pp. 419–431, 2015.

[25] S. A.-B. Ibrahim, "Fatigue analysis and instability problems of plate girders with corrugated webs," Ph.D. dissertation, Drexel University, 2001.

[26] F. Hlal and M. Al-Emrani, "Detail categories for the flange-to-web weld detail in corrugated web girders," *Engineering Structures*, vol. 324, p. 119 342, 2025. DOI: 10.1016/j.engstruct.2024.119342.

[27] G. Zuo, L. Tong, Z. Zhao, H. Wang, W. Shi, and Y. Yan, "Experimental study on fatigue strength of trapezoidal corrugated-web steel girders based on hot spot stress method," *Thin-Walled Structures*, vol. 208, p. 112 814, 2025. DOI: 10.1016/j.tws.2024.112814.

[28] G. Zuo, L. Tong, Z. Zhao, H. Wang, W. Shi, and Y. Yan, "Scf formulae of corrugated-web steel girders: Experiments and numerical analysis," *Journal of Constructional Steel Research*, vol. 228, p. 109 445, 2025. DOI: 10.1016/j.jcsr.2025.109445.

[29] G. Zuo, L. Tong, Z. Zhao, H. Wang, and Y. Yan, "A unified design method for fatigue strength of corrugated-web steel girders," *Journal of Constructional Steel Research*, vol. 241, p. 110 322, 2026. DOI: 10.1016/j.jcsr.2026.110322.

[30] *Eurocode 3: Design of steel structures – part 1-8: Design of joints*, Brussels, Belgium: European Committee for Standardization (CEN), 2005.

[31] F. Hlal and M. Al-Emrani, "Load effects in beams with corrugated webs: Numerical study," in *Nordic Steel Construction Conference*, Luleå, Sweden, 2024.

[32] S. Saad-Eldeen, K. Tran Nguyen, Y. Garbatov, and C. Guedes Soares, "Fatigue stress assessment accounting for contact effect," in *Tenth International Conference on Marine Sciences and Technologies*, 2010.

[33] E. Save and K. Åkermo, "Fatigue performance of welded steel girders with corrugated webs," M.S. thesis, Chalmers University of Technology, Gothenburg, Sweden, 2020.

[34] A. Emme and E. Svensson, "Fatigue strength of bridge girders with corrugated web," M.S. thesis, Chalmers University of Technology, Gothenburg, Sweden, 2024.

[35] I. C. III.2, *Fatigue and Fracture*. Seoul, Korea: International Ship and Offshore Structures Congress, 2009.

[36] A. Göransson, "Fatigue life analysis of weld ends: Comparison between testing and fem calculations," M.S. thesis, Linköping University, 2014.

[37] F. Hlal, M. Al-Emrani, and M. Amani, "Preliminary study on plate girders with corrugated webs," Chalmers University of Technology, Gothenburg, Sweden, Tech. Rep. Report ACE 2022:3, 2022.

[38] M. S. Cheung and W. C. Li, "Probabilistic fatigue and fracture analyses of steel bridges," *Structural Safety*, vol. 23, pp. 245–262, 2003. DOI: 10.1016/S0167-4730(02)00067-X.

[39] C. M. Sonsino, W. Fricke, F. de Bruyne, A. Hoppe, A. Ahmadi, and G. Zhang, "Notch stress concepts for the fatigue assessment of welded joints," *International Journal of Fatigue*, vol. 34, pp. 2–16, 2012. DOI: 10.1016/j.ijfatigue.2010.04.011.

[40] W. Yao, K. Xia, and Y. Gu, "On the fatigue notch factor kf," *International Journal of Fatigue*, vol. 17, pp. 245–251, 1995.

[41] V. A. Järvenpää and K. Mäntyjärvi, "Fatigue performance of welded materials," in *Welding of Metallic Materials*, Elsevier, 2023, pp. 537–560. DOI: 10.1016/B978-0-323-90552-7.00015-8.

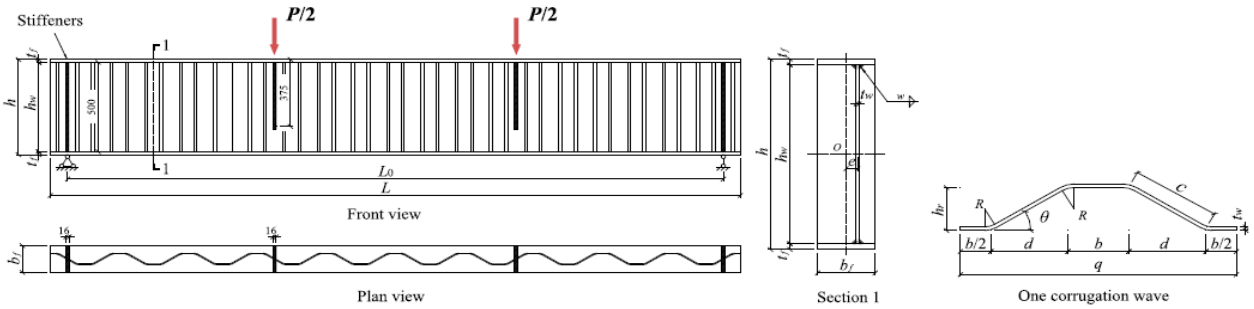
Appendix A

Calculations for Validation of the FE Models

This appendix presents the analytical calculations used to validate the finite element models against the reference experimental study by [28]. The calculations include the determination of nominal stresses and deflections for the investigated girders.

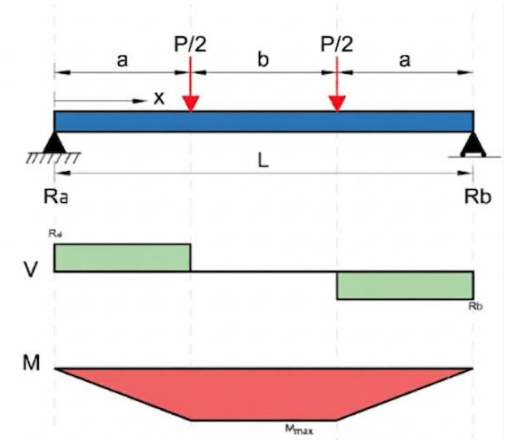
Detailed dimensions of CWGs (unit: mm).

Category	Group	L	L0	w	bf	tf	h	hw	tw	e	R	b	d	c	hr	q	θ
TCWGs	GA	3600	3420	5	150	16	532	500	5	30	30	80	100	116.6	60	360	30.9
TCWGs	GB	3600	3420	5	150	16	532	500	5	30	30	120	60	84.8	60	360	45
TCWGs	GC	3600	3420	5	150	16	532	500	5	30	60	120	60	84.8	60	360	45
TCWGs	GD	3500	3250	5	150	16	532	500	5	30	30	150	100	116.6	60	500	30.9



Load, stresses and deflection calculation

	GA-1 GB-1 GC-1	GD-1	
Total load P	385	415.0	kN
Load at each point P/2	***192.5***	***207.5***	kN
Beam length L	3600	3500	mm
Distance btw supports L0	3420	3250	mm
a (distance btw support and point load	1080	1000	mm
b (distance btw point loads)	1260	1250	mm
M = P * a / 2			
Max moment	207.9	207.5	kN.m
V = P / 2			
Max shear force	192.5	207.5	kN
E	210	210	Gpa
v (poisson ratio)	0.3	0.3	[-]
G (Shear modulus)	80.77	80.77	Gpa



G = Shear modulus of elasticity (kN/mm²) = E / [2(1 + v)]

$$I_{\text{flanges}} = 2\left[\frac{b_f t_f^3}{12} + b_f t_f \left(\frac{h_w + t_f}{2}\right)^2\right]$$

I (flanges only)	3E+08	3E+08	mm^4

CWG: web contribution neglected in normal stress definition

$$I_{\text{FWG}} = I_{\text{flanges}} + \frac{t_w h_w^3}{12}$$

I FWG	4E+08	4E+08	mm^4

FWG: flat-web girder with web contribution included

Normal stress at the upper surface of the bottom flange in the constant-moment region

	GA-1, GB-1, GC-1		GD-1	
Top surface	CWG	FWG	CWG	FWG
y [mm]	**σ [Mpa]**	**σ [Mpa]**	**σ [Mpa]**	**σ [Mpa]**
250	162.62	139.83	162.30	139.56

$$\sigma = \frac{My}{I}$$

Midspan deflection

GA-1, GB-1, GC-1		GD-1	
CWG	FWG	CWG	FWG
δ [mm]	**δ [mm]**	**δ [mm]**	**δ [mm]**
4.96	4.41	4.59	4.09

$$\delta = \delta_b + \delta_s$$

$$\delta = \frac{Pa_s}{8EI_R}\left[\frac{4}{3}a_s^2 + 2a_s b + \frac{b^2}{2}\right] + \frac{(P/2)(a_s)}{GA_w}$$

Appendix B

Additional S–N Curve Analyses

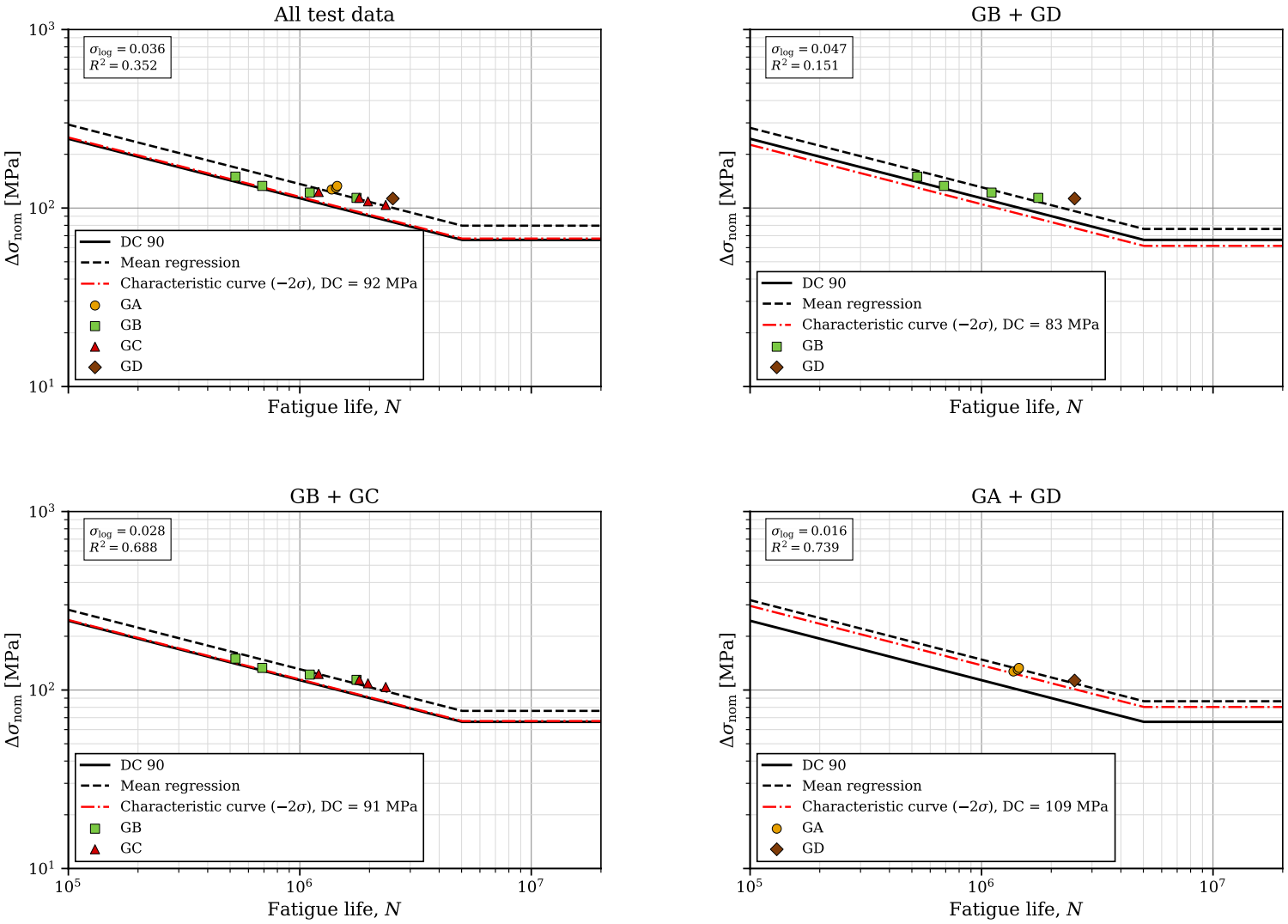
Here, some further S-N curves analysis will be performed in the framework of the following three approaches: Nominal Stress (NS), Structural Hot-Spot Stress (HSS), and Effective Notch Stress (ENS). Presented plots give insights about fatigue properties of the samples and are complementary to those analyzed in Chapter 4.

In addition to all test data conducted from the reference study [28], there is a need to analyze some of the results for a few pairs in order to find out how particular geometric parameters affect fatigue behavior. In the first case, GB and GD are compared in order to understand the effect of the corrugation angle θ under almost constant values of other geometric parameters. When considering GA and GC, one should pay attention to the effect of the bend radius R. This time, there is no difference in the geometry of the samples except for the factor mentioned above. Analysis of the characteristics of GA and GD samples is needed in order to investigate the effect of the parallel fold length (a_1).

S-N curves found together with mean regression lines and other fatigue parameters are presented below.

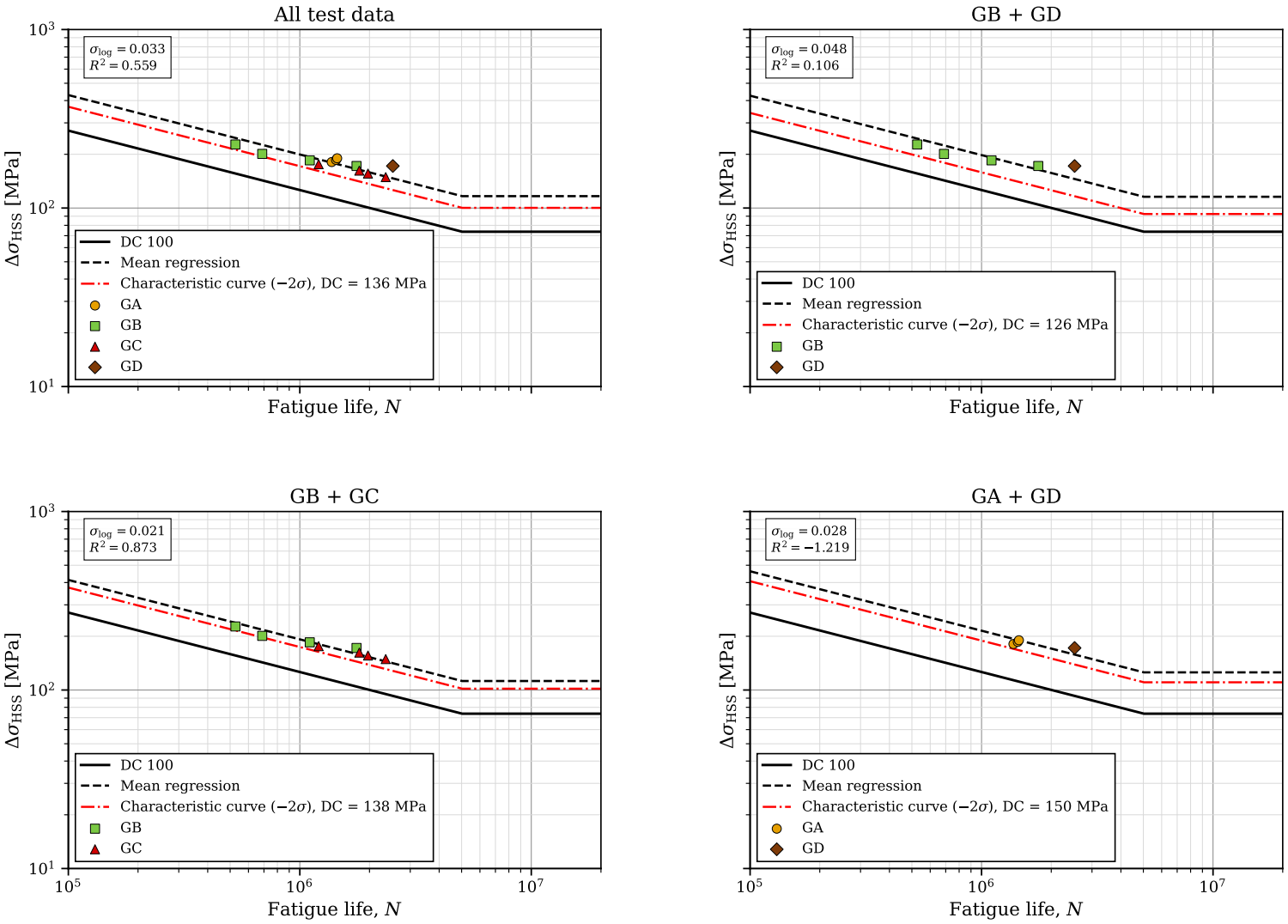
B.1 Nominal Stress Method

S-N Curves — Nominal Stress Method



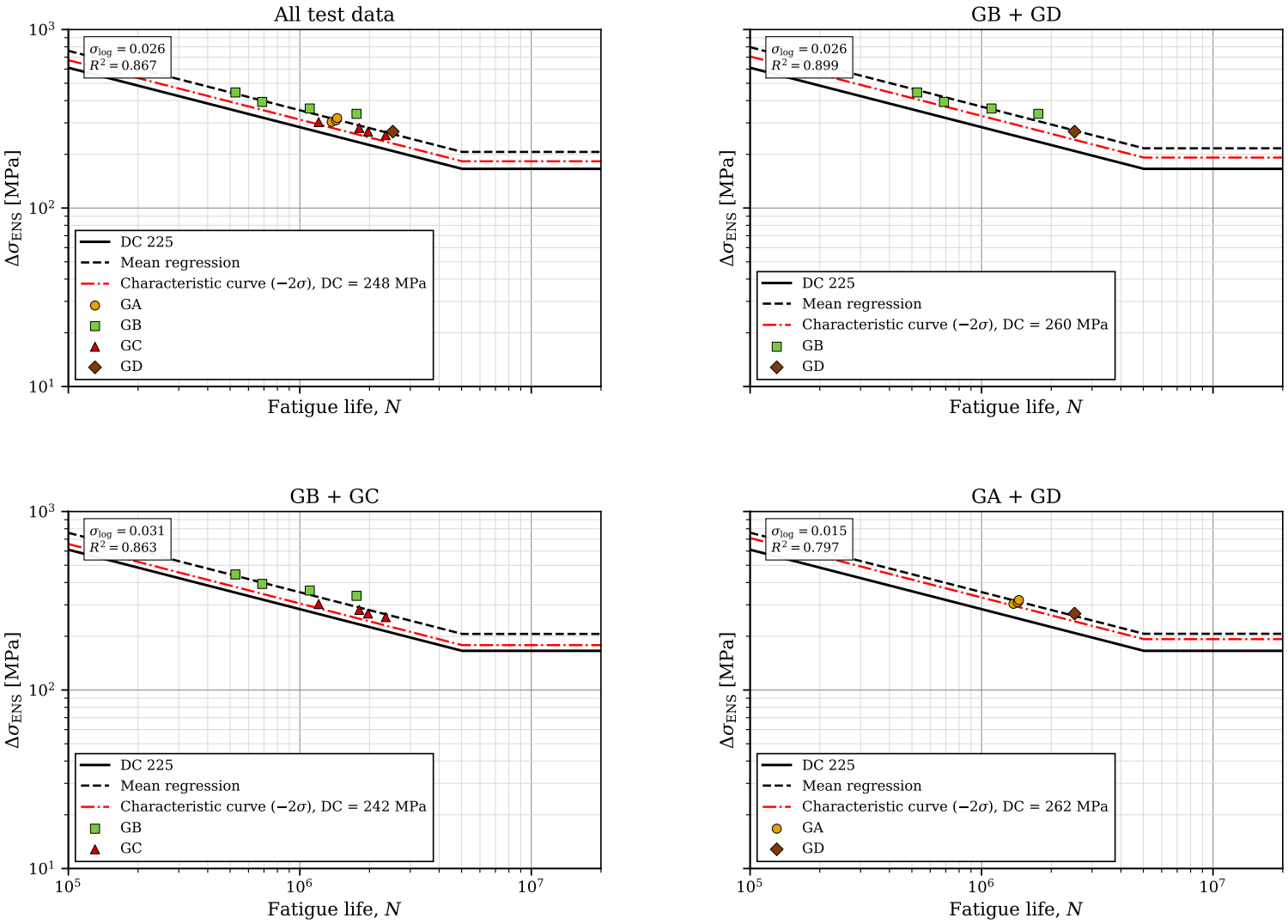
B.2 Structural Hot-Spot Stress Method

S-N Curves — Hot-Spot Stress Method



B.3 Effective Notch Stress Method

S-N Curves — Effective Notch Stress Method



Appendix C

Representative FE Models Used in the Parametric Study

The results presented in Figures 6.1 and 6.2 are based on representative finite element (FE) models selected from the complete parametric database. For each parameter combination, the representative model corresponds to the FE case whose SCF_{ENS} is closest to the mean value of the corresponding group. The following pages provide the geometric dimensions, case identifiers, and SCF_{ENS} values of the selected representative FE models.

Geometric Dimensions and Case IDs of the Representative FE Models Used in Figure 6.1

Figure 6.1 shows SCF_ENS versus R/t_w (panels: a_3/b_f = 0.50, 0.25, 0.17 — lines: θ = 30°, 45°, 60°, 70°). Each data point is the mean of all FE cases sharing the same (a_3/b_f, θ, R/t_w) combination. The representative case selected is the individual FE model whose SCF_ENS is closest to the group mean.

Case ID	R/tw	θ [°]	R [mm]	bf [mm]	tf[mm]	tw [mm]	a_1 [mm]	a_2 [mm]	a_3 [mm]	a_3/b_f	SCF_ENS (rep. case)	SCF_ENS (group mean)
658	2	30	16	1200	40	8	600	400	200	0.17	4.09	4.058
656	6	30	48	1200	40	8	250	400	200	0.17	3.02	3.008
660	10	30	80	1200	40	8	600	400	200	0.17	3.3	3.303
1402	2	45	16	1200	60	8	600	282.8	200	0.17	4.74	4.774
1400	6	45	48	1200	60	8	250	282.8	200	0.17	3.36	3.373
1380	10	45	80	1200	40	8	600	282.8	200	0.17	3.53	3.58
2098	2	60	16	1200	40	8	600	230.9	200	0.17	5.3	5.353
2096	6	60	48	1200	40	8	250	230.9	200	0.17	3.67	3.684
2100	10	60	80	1200	40	8	600	230.9	200	0.17	3.68	3.77
2842	2	70	16	1200	60	8	600	212.8	200	0.17	5.58	5.706
2864	6	70	48	1200	80	8	250	212.8	200	0.17	3.69	3.652
2820	10	70	80	1200	40	8	600	212.8	200	0.17	3.65	3.572
328	2	30	24	400	40	12	300	200	100	0.25	3.54	3.598
296	6	30	48	400	20	8	150	200	100	0.25	2.74	2.723
300	10	30	80	400	20	8	300	200	100	0.25	2.98	3.014
1348	2	45	8	800	80	4	600	282.8	200	0.25	4.38	4.378
1016	6	45	48	400	20	8	150	141.4	100	0.25	3.17	3.159
1302	10	45	40	800	40	4	600	282.8	200	0.25	3.3	3.309
1792	2	60	24	400	60	12	300	115.5	100	0.25	4.94	4.968
1736	6	60	48	400	20	8	150	115.5	100	0.25	3.44	3.442
1740	10	60	80	400	20	8	300	115.5	100	0.25	3.44	3.482
2512	2	70	24	400	60	12	300	106.4	100	0.25	5.26	5.303
2786	6	70	24	800	80	4	250	212.8	200	0.25	3.25	3.292
2796	10	70	80	800	80	8	600	212.8	200	0.25	3.67	3.468
514	2	30	16	400	20	8	600	400	200	0.50	3.58	3.608
278	6	30	72	200	40	12	150	200	100	0.50	2.8	2.804
282	10	30	120	200	40	12	300	200	100	0.50	3.15	3.115
1252	2	45	8	400	40	4	600	282.8	200	0.50	4.35	4.275
1226	6	45	24	400	20	4	250	282.8	200	0.50	3.21	3.204
1254	10	45	40	400	40	4	600	282.8	200	0.50	3.3	3.316
1954	2	60	16	400	20	8	600	230.9	200	0.50	4.88	4.914
1712	6	60	48	200	40	8	150	115.5	100	0.50	3.45	3.412
1980	10	60	80	400	40	8	600	230.9	200	0.50	3.72	3.613
2674	2	70	16	400	20	8	600	212.8	200	0.50	5.1	5.138
2666	6	70	24	400	20	4	250	212.8	200	0.50	3.33	3.378
2700	10	70	80	400	40	8	600	212.8	200	0.50	3.69	3.579

Notes: (1) $a_2 = a_3 / \sin\theta$ is the slant length of the inclined fold [mm]. (2) SCF_ENS (rep. case) = ENS-SCF of the single selected case. (3) SCF_ENS (group mean) = mean over all cases in that (a_3/b_f, θ, R/t_w) group. (4) Case IDs correspond to the identifiers in the full parametric database.

Geometric Dimensions and Case IDs of the Representative FE Models Used in Figure 6.2

Figure 6.2 shows SCF_ENS versus corrugation angle θ (panels: R/t_w = 2, 6, 10 — lines: a_3/b_f = 0.50, 0.25, 0.17).

Case ID	R/tw	θ [°]	R [mm]	bf [mm]	tf[mm]	tw [mm]	a_1 [mm]	a_2 [mm]	a_3 [mm]	a_3/b_f	SCF_ENS (rep. case)	SCF_ENS (group mean)
658	2	30	16	1200	40	8	600	400	200	0.17	4.09	4.058
1402	2	45	16	1200	60	8	600	282.8	200	0.17	4.74	4.774
2098	2	60	16	1200	40	8	600	230.9	200	0.17	5.3	5.353
2842	2	70	16	1200	60	8	600	212.8	200	0.17	5.58	5.706
328	2	30	24	400	40	12	300	200	100	0.25	3.54	3.598
1348	2	45	8	800	80	4	600	282.8	200	0.25	4.38	4.378
1792	2	60	24	400	60	12	300	115.5	100	0.25	4.94	4.968
2512	2	70	24	400	60	12	300	106.4	100	0.25	5.26	5.303
514	2	30	16	400	20	8	600	400	200	0.50	3.58	3.608
1252	2	45	8	400	40	4	600	282.8	200	0.50	4.35	4.275
1954	2	60	16	400	20	8	600	230.9	200	0.50	4.88	4.914
2674	2	70	16	400	20	8	600	212.8	200	0.50	5.1	5.138
656	6	30	48	1200	40	8	250	400	200	0.17	3.02	3.008
1400	6	45	48	1200	60	8	250	282.8	200	0.17	3.36	3.373
2096	6	60	48	1200	40	8	250	230.9	200	0.17	3.67	3.684
2864	6	70	48	1200	80	8	250	212.8	200	0.17	3.69	3.652
296	6	30	48	400	20	8	150	200	100	0.25	2.74	2.723
1016	6	45	48	400	20	8	150	141.4	100	0.25	3.17	3.159
1736	6	60	48	400	20	8	150	115.5	100	0.25	3.44	3.442
2786	6	70	24	800	80	4	250	212.8	200	0.25	3.25	3.292
278	6	30	72	200	40	12	150	200	100	0.50	2.8	2.804
1226	6	45	24	400	20	4	250	282.8	200	0.50	3.21	3.204
1712	6	60	48	200	40	8	150	115.5	100	0.50	3.45	3.412
2666	6	70	24	400	20	4	250	212.8	200	0.50	3.33	3.378
660	10	30	80	1200	40	8	600	400	200	0.17	3.3	3.303
1380	10	45	80	1200	40	8	600	282.8	200	0.17	3.53	3.58
2100	10	60	80	1200	40	8	600	230.9	200	0.17	3.68	3.77
2820	10	70	80	1200	40	8	600	212.8	200	0.17	3.65	3.572
300	10	30	80	400	20	8	300	200	100	0.25	2.98	3.014
1302	10	45	40	800	40	4	600	282.8	200	0.25	3.3	3.309
1740	10	60	80	400	20	8	300	115.5	100	0.25	3.44	3.482
2796	10	70	80	800	80	8	600	212.8	200	0.25	3.67	3.468
282	10	30	120	200	40	12	300	200	100	0.50	3.15	3.115
1254	10	45	40	400	40	4	600	282.8	200	0.50	3.3	3.316
1980	10	60	80	400	40	8	600	230.9	200	0.50	3.72	3.613
2700	10	70	80	400	40	8	600	212.8	200	0.50	3.69	3.579

Notes: (1) $a_2 = a_3 / \sin\theta$ is the slant length of the inclined fold [mm]. (2) SCF_ENS (rep. case) = ENS-SCF of the single selected case. (3) SCF_ENS (group mean) = mean over all cases in that (a_3/b_f, θ, R/t_w) group. (4) All cases are drawn from the verified 816-case FE database.

Appendix D

Python Script for Development of the $\mathrm{SCF_{ENS}}$ Prediction Equation.

The complete Python script used for the development of the proposed $\mathrm{SCF_{ENS}}$ prediction equation is presented below.

```
# ============================================================
# MASTER THESIS SCRIPT
# Fatigue Assessment of Corrugated Web Girders
# ============================================================
#
# This script is developed as part of a Master's thesis at
# Chalmers University of Technology in collaboration with WSP.
#
# The aim of the project is to investigate fatigue behaviour
# in steel girders with corrugated webs, with a focus on the
# weld toe at the inclined fold between the web and flange.
#
# The script builds a multi-scale finite element workflow in Abaqus:
#
# 1) Global Model
#    - Full girder model
#    - Used to obtain nominal stress distribution
#
# 2) HSS Submodel (Hot Spot Stress)
#    - Local cut model around S-points (SP4 and SP8)
#    - Used to evaluate structural hot-spot stresses
#
# 3) ENS Submodel (Effective Notch Stress)
#    - High-resolution notch model with 1 mm radius
#    - Used to capture local stress concentration at the weld toe
#
# The script is fully parametric and supports batch analysis
# using CSV input files. It automatically:
#    - Generates geometry
#    - Applies boundary conditions
#    - Creates meshes: coarse, transition, and fine
#    - Submits jobs locally or on a cluster
#    - Extracts stresses: Nominal, HSS, and ENS
#    - Computes Stress Concentration Factors (SCF)
#
```

```
# Key features:
#    - Robust meshing around the weld toe using centerline logic
#    - Adaptive trimming to reduce model size and runtime
#    - Consistent extraction of stresses at SP4 and SP8
#    - Designed for large parametric studies
#
# The final objective is to establish a relationship between:
#    Nominal stress, Hot-Spot stress, and Notch stress
#
# This can reduce the need for detailed FE modelling
# in practical bridge design.
#
# Authors: Ghaith Alharami and Mohammad Sami Raad
# Date: May 2026
# ============================================================

# -*- coding: mbcs -*-
from part import *
from sketch import *
from assembly import *
from interaction import *
from material import *
from section import *
from step import *
from mesh import *
from abaqusConstants import *
from visualization import *
from odbAccess import openOdb
import regionToolset
import math
import csv
import os
import shutil
import time

# ============================================================
# 0. RUN / CLUSTER SETTINGS
# ============================================================
# Edit these values to match your cluster / workstation setup.
JOB_NUM_CPUS = int(os.environ.get('SLURM_CPUS_PER_TASK', 6))
JOB_NUM_GPUS = 0
JOB_MEMORY_PERCENT = 90
JOB_QUEUE = None
JOB_SCRATCH = os.environ.get('ABAQUS_SCRATCH', '')


# output folders
RESULTS_ROOT_DIR = 'results'
ODB_ROOT_DIR = 'odb'
SUMMARY_ROOT_DIR = 'summary'
CURRENT_CASE_TAG = 'BATCH'

LOG_LEVEL = 1
# 0 = silent
# 1 = important only
# 2 = debug details

def log_print(*args, **kwargs):
    global CURRENT_CASE_TAG

```

```
    level = kwargs.get('level', 1)

    if level > LOG_LEVEL:
        return

    if len(args) == 0:
        text_msg = ''

    elif len(args) == 1:
        text_msg = str(args[0])

    elif len(args) == 2:
        a0, a1 = args
        s0 = str(a0)
        s1 = str(a1)

        if s0.startswith('case_') or s0 == 'BATCH':
            CURRENT_CASE_TAG = s0
            text_msg = s1
        else:
            text_msg = s0 + ' ' + s1

    else:
        text_msg = ' '.join([str(a) for a in args])

    print('[CASE %s] %s' % (CURRENT_CASE_TAG, text_msg))


# ============================================================
# 1. PARAMETERS
# ============================================================
def get_params(overrides=None):
    P = {}

    # -------------------------
    # Independent parameters
    # -------------------------
    P['bf'] = 150.0            # flange width
    P['tf'] = 16.0             # flange thickness

    P['tw'] = 5.0              # web thickness
    P['b'] = 80.0              # flat length of corrugation
    P['hr'] = 60              # corrugation height
    P['R'] = 30               # bend radius
    P['theta'] = 30.9          # corrugation angle in degrees

    # Weld throat from relation: 2a = 1.15 * tw
    P['weld_throat'] = 0.575 * P['tw']
    P['F_load'] = 402900.0
    P['global_n_waves'] = 2

    # global BC study options
    P['couple_web_to_rp'] = False
    P['restrain_web_top_z'] = False

     # mesh
    P['mesh_size_global'] = 10.0
    P['mesh_size_hss'] = 5.0
    P['hss_n_elem_per_segment'] = 3

```

```
    P['mesh_size_ens_global'] = 5
    P['mesh_size_ens_transition'] = 1.0
    P['mesh_size_ens_toe'] = 0.15

    # material
    P['E'] = 210000.0
    P['nu'] = 0.30

    # -------------------------
    # Derived parameters
    # -------------------------
    P['hw'] = 2.0 * P['tf']

    P['theta_rad'] = math.radians(P['theta'])
    P['tan_theta'] = math.tan(P['theta_rad'])

    P['d'] = P['hr'] / P['tan_theta']
    P['L'] = P['b'] + P['d']
    P['q'] = 2.0 * P['L']
    P['x_global_left'] = -P['L']
    P['x_global_right'] = P['x_global_left'] + P['global_n_waves'] * P['q']

    P['b_half'] = 0.5 * P['b']

    P['weld_leg'] = P['weld_throat'] / math.sin(math.radians(45.0))

    # ENS
    P['notch_radius'] = 1.0
    P['sqrt2'] = math.sqrt(2.0)

    if overrides is not None:
        for key, value in overrides.items():
            if key in P and value is not None:
                P[key] = value

        # re-compute dependent parameters after overrides
        if 'hw' not in overrides or overrides['hw'] is None:
            P['hw'] = 2.0 * P['tf']
        P['theta_rad'] = math.radians(P['theta'])
        P['tan_theta'] = math.tan(P['theta_rad'])
        P['d'] = P['hr'] / P['tan_theta']
        P['L'] = P['b'] + P['d']
        P['q'] = 2.0 * P['L']
        P['x_global_left'] = -P['L']
        P['x_global_right'] = P['x_global_left'] + P['global_n_waves'] * P['q']
        P['b_half'] = 0.5 * P['b']
        P['weld_throat'] = 0.575 * P['tw']
        P['weld_leg'] = P['weld_throat'] / math.sin(math.radians(45.0))

    return P

GLOBAL_MODEL_NAME = 'Global-Model'
GLOBAL_LOCAL_SOURCE_MODEL_NAME = 'Global-Local-Source'
HSS_SOURCE_MODEL_NAME = 'HSS-Source-Model'
HSS_MODEL_NAME = 'HSS-Model'
ENS_SP4_MODEL_NAME = 'ENS-SP4-Model'
ENS_SP8_MODEL_NAME = 'ENS-SP8-Model'


# ============================================================
```

```
# SAFE CONSTRAINT HELPERS (cluster-compatible)
# ============================================================
def safe_delete_constraint(model_obj, constraint_name):
    """
    Delete a model constraint only if the current Abaqus environment
    exposes model.constraints. Some cluster/noGUI environments may not.
    """
    try:
        if hasattr(model_obj, 'constraints'):
            if constraint_name in model_obj.constraints.keys():
                del model_obj.constraints[constraint_name]
    except:
        pass


def safe_suppress_all_constraints(model_obj):
    """
    Suppress all model constraints only if model.constraints exists.
    """
    try:
        if hasattr(model_obj, 'constraints'):
            for name in list(model_obj.constraints.keys()):
                try:
                    model_obj.constraints[name].suppress()
                except:
                    pass
    except:
        pass

# ============================================================
# 2. HELPER FUNCTIONS
# ============================================================

def recreate_model(model_name):
    if model_name in mdb.models.keys():
        del mdb.models[model_name]
    return mdb.Model(name=model_name)


def point_on_line(pa, pb, t):
    x = pa[0] + t * (pb[0] - pa[0])
    y = pa[1] + t * (pb[1] - pa[1])
    return (x, y)

def point_on_line_by_distance(pa, pb, dist):
    dx = pb[0] - pa[0]
    dy = pb[1] - pa[1]
    L1 = math.sqrt(dx * dx + dy * dy)

    x = pa[0] + dist * dx / L1
    y = pa[1] + dist * dy / L1
    return (x, y)

def point_from_end(pa, pb, dist):
    dx = pa[0] - pb[0]
    dy = pa[1] - pb[1]
    L1 = math.sqrt(dx * dx + dy * dy)

    x = pb[0] + dist * dx / L1
    y = pb[1] + dist * dy / L1
```

```
    return (x, y)

def create_datum_plane_x_constant(part_obj, x_value):
    dp = part_obj.DatumPlaneByPrincipalPlane(
        principalPlane=YZPLANE,
        offset=x_value
    )
    return part_obj.datums[dp.id]


def get_or_create_datum_axis_z(part_obj):
    # check if already exists
    if 'DATUM_AXIS_Z' in part_obj.features.keys():
        datum_id = part_obj.features['DATUM_AXIS_Z'].id
        return part_obj.datums[datum_id]

    # create once
    da = part_obj.DatumAxisByPrincipalAxis(principalAxis=ZAXIS)

    # rename feature for reuse
    part_obj.features.changeKey(fromName=da.name, toName='DATUM_AXIS_Z')

    datum_id = part_obj.features['DATUM_AXIS_Z'].id
    return part_obj.datums[datum_id]


def create_edge_set_from_list(part_obj, edge_list, set_name):

    if len(edge_list) == 0:
        log_print('Warning: no edges found for %s' % set_name, level=2)
        return None

    if set_name in part_obj.sets.keys():
        del part_obj.sets[set_name]

    unique_indices = []
    seen = {}

    for e in edge_list:
        try:
            idx = e.index
        except:
            continue

        if idx not in seen:
            seen[idx] = True
            unique_indices.append(idx)

    if len(unique_indices) == 0:
        log_print('Warning: no valid edges for %s' % set_name, level=2)
        return None

    edge_seq = part_obj.edges[unique_indices[0]:unique_indices[0] + 1]

    for idx in unique_indices[1:]:
        edge_seq = edge_seq + part_obj.edges[idx:idx + 1]

    part_obj.Set(
        name=set_name,
        edges=edge_seq
```

```
    )

    log_print('%s edges = %d' % (set_name, len(unique_indices)))

    return part_obj.sets[set_name]

def split_final_transition_to_fine_transition(model_name, P, Y):

    p = mdb.models[model_name].parts['Specimen']

    input_set = 'SET_ENS_FINAL_CELL_TRANSITION_EDGES'

    if input_set not in p.sets.keys():
        log_print('%s not found in %s' % (input_set, model_name), level=2)
        return {'fine': 0, 'transition': 0}

    edges_in = p.sets[input_set].edges

    fine_edges = []
    transition_edges = []

    z_fine_min = P['tf'] - 0.02
    z_fine_max = P['tf'] + 0.30
    z_tf_tol = 0.02

    tol_y_cut = 0.75
    tol_x_start = 0.75

    # SP4 walks mainly +x, SP8 walks mainly -x
    if model_name == ENS_SP8_MODEL_NAME:
        walk_direction = -1.0
    else:
        walk_direction = 1.0

    z_tf_edges = []
    z_tf_edge_indices = {}

    def get_edge_points(e):
        try:
            v_ids = e.getVertices()
            if len(v_ids) == 2:
                p1 = p.vertices[v_ids[0]].pointOn[0]
                p2 = p.vertices[v_ids[1]].pointOn[0]

                mid = (
                    0.5 * (p1[0] + p2[0]),
                    0.5 * (p1[1] + p2[1]),
                    0.5 * (p1[2] + p2[2])
                )

                return v_ids[0], v_ids[1], p1, p2, mid
        except:
            pass

        try:
            mid = e.pointOn[0]
            return None, None, mid, mid, mid
        except:
            return None, None, None, None, None
```

```
    # ----------------------------
    # 1) z-based classification
    # ----------------------------
    for e in edges_in:

        v1, v2, p1, p2, mid = get_edge_points(e)

        if mid is None:
            continue

        x_mid, y_mid, z_mid = mid
        z1 = p1[2]
        z2 = p2[2]

        if not (z_fine_min <= z1 <= z_fine_max and
                z_fine_min <= z2 <= z_fine_max):
            transition_edges.append(e)
            continue

        if abs(z_mid - P['tf']) <= z_tf_tol:
            z_tf_edges.append(e)
            z_tf_edge_indices[e.index] = True
            continue

        on_y_cut = (
            abs(y_mid - Y['y_min']) <= tol_y_cut or
            abs(y_mid - Y['y_max']) <= tol_y_cut
        )

        if on_y_cut:
            transition_edges.append(e)
        else:
            fine_edges.append(e)

    # ----------------------------
    # 2) connectivity for z=tf edges
    # ----------------------------
    vertex_to_edges = {}
    edge_to_vertices = {}
    x_values = []

    for e in z_tf_edges:

        v1, v2, p1, p2, mid = get_edge_points(e)

        if v1 is None or v2 is None:
            continue

        edge_to_vertices[e.index] = (v1, v2)

        if v1 not in vertex_to_edges:
            vertex_to_edges[v1] = []
        if v2 not in vertex_to_edges:
            vertex_to_edges[v2] = []

        vertex_to_edges[v1].append(e)
        vertex_to_edges[v2].append(e)

        x_values.append(p1[0])
        x_values.append(p2[0])
```

```
456
457     if len(x_values) == 0:
458         create_edge_set_from_list(p, transition_edges, 'SET_ENS_CENTERLINE_TRANSITION')
459         create_edge_set_from_list(p, fine_edges, 'SET_ENS_CENTERLINE_FINE')
460         return {'fine': len(fine_edges), 'transition': len(transition_edges)}
461
462     x_left_local = min(x_values)
463     x_right_local = max(x_values)
464
465     # choose start side depending on model
466     if walk_direction > 0:
467         x_start = x_left_local
468         x_end = x_right_local
469     else:
470         x_start = x_right_local
471         x_end = x_left_local
472
473     # ----------------------------
474     # 3) start vertices: max y and min y are outer tracks
475     # ----------------------------
476     start_vertices = []
477
478     for vid in vertex_to_edges.keys():
479         x, y, z = p.vertices[vid].pointOn[0]
480
481         if abs(z - P['tf']) <= z_tf_tol and abs(x - x_start) <= tol_x_start:
482             start_vertices.append((vid, x, y, z))
483
484     transition_z_tf_edge_indices = {}
485
486     if len(start_vertices) < 2:
487         log_print('%s warning: could not find enough start vertices.' % model_name, level=2)
488
489         for e in z_tf_edges:
490             fine_edges.append(e)
491
492     else:
493         start_min_y = min(start_vertices, key=lambda item: item[2])[0]
494         start_max_y = max(start_vertices, key=lambda item: item[2])[0]
495
496         outer_start_vertices = [start_min_y, start_max_y]
497
498         # ----------------------------
499         # 4) walk connected outer tracks
500         # ----------------------------
501         for start_vid in outer_start_vertices:
502
503             current_vid = start_vid
504             previous_edge_index = None
505
506             safety_counter = 0
507             max_steps = len(z_tf_edges) + 10
508
509             while safety_counter < max_steps:
510                 safety_counter += 1
511
512                 connected_edges = vertex_to_edges.get(current_vid, [])
513                 candidate_edges = []
514
515                 x_current = p.vertices[current_vid].pointOn[0][0]
```

```
516
517                 for e in connected_edges:
518                     if e.index == previous_edge_index:
519                         continue
520
521                     if e.index not in z_tf_edge_indices:
522                         continue
523
524                     v_a, v_b = edge_to_vertices[e.index]
525                     other_vid = v_b if v_a == current_vid else v_a
526                     x_other = p.vertices[other_vid].pointOn[0][0]
527
528                     # SP4: move +x
529                     # SP8: move -x
530                     if walk_direction > 0:
531                         ok_move = x_other >= x_current - 1.0e-6
532                     else:
533                         ok_move = x_other <= x_current + 1.0e-6
534
535                     if ok_move:
536                         candidate_edges.append((e, other_vid, x_other))
537
538                 if len(candidate_edges) == 0:
539                     break
540
541                 if walk_direction > 0:
542                     candidate_edges.sort(key=lambda item: item[2], reverse=True)
543                 else:
544                     candidate_edges.sort(key=lambda item: item[2])
545
546                 next_edge = candidate_edges[0][0]
547                 next_vid = candidate_edges[0][1]
548
549                 transition_z_tf_edge_indices[next_edge.index] = True
550
551                 previous_edge_index = next_edge.index
552                 current_vid = next_vid
553
554                 x_new = p.vertices[current_vid].pointOn[0][0]
555
556                 if walk_direction > 0:
557                     if x_new >= x_end - tol_x_start:
558                         break
559                 else:
560                     if x_new <= x_end + tol_x_start:
561                         break
562
563         # ----------------------------
564         # 5) classify z=tf edges
565         # ----------------------------
566         for e in z_tf_edges:
567             if e.index in transition_z_tf_edge_indices:
568                 transition_edges.append(e)
569             else:
570                 fine_edges.append(e)
571
572     create_edge_set_from_list(
573         p,
574         transition_edges,
575         'SET_ENS_CENTERLINE_TRANSITION'
```

```
    )

    create_edge_set_from_list(
        p,
        fine_edges,
        'SET_ENS_CENTERLINE_FINE'
    )

    log_print('%s final fine edges = %d' % (model_name, len(fine_edges)), level=2)
    log_print('%s final transition edges = %d' % (model_name, len(transition_edges)), level=2)

    return {
        'fine': len(fine_edges),
        'transition': len(transition_edges)
    }

def debug_ens_cells(model_name):

    p = mdb.models[model_name].parts['Specimen']

    log_print('-----------------------------------', level=2)
    log_print('%s CELL DEBUG' % model_name, level=2)

    for c in p.cells:
        try:
            x, y, z = c.pointOn[0]
            edge_ids = c.getEdges()
            face_ids = c.getFaces()

            log_print(
                'cell %d: pointOn=(%.3f, %.3f, %.3f), edges=%d, faces=%d'
                % (c.index, x, y, z, len(edge_ids), len(face_ids)), level=2
            )
        except:
            pass

    log_print('-----------------------------------', level=2)


def create_ens_transition_edges_from_final_cells(model_name, P):

    p = mdb.models[model_name].parts['Specimen']

    transition_cell_indices = [1, 2]

    edge_list = []
    seen = {}

    z_min = P['tf'] - 0.5
    z_max = P['tf'] + 3.0

    for cid in transition_cell_indices:

        if cid >= len(p.cells):
            log_print('%s warning: cell %d not found' % (model_name, cid), level=2)
            continue

        c = p.cells[cid]

        try:
```

```
            for eid in c.getEdges():

                if eid in seen:
                    continue

                e = p.edges[eid]

                keep_edge = False

                try:
                    pts = [e.pointOn[0]]

                    v_ids = e.getVertices()
                    for vid in v_ids:
                        pts.append(p.vertices[vid].pointOn[0])

                    for pt in pts:
                        z = pt[2]
                        if z_min <= z <= z_max:
                            keep_edge = True
                            break

                except:
                    pass

                if keep_edge:
                    seen[eid] = True
                    edge_list.append(e)

        except:
            pass

    create_edge_set_from_list(
        p,
        edge_list,
        'SET_ENS_FINAL_CELL_TRANSITION_EDGES'
    )

    log_print('%s final-cell transition edges near flange = %d'
              % (model_name, len(edge_list)), level=2)


# ========================================================
# 3. GEOMETRY KEY POINTS
# ========================================================

def get_geometry_points(P):
    G = {}

    bf = P['bf']
    tw = P['tw']
    b = P['b']
    hr = P['hr']
    d = P['d']
    L = P['L']

    # ------------------------------------------------------
    # Flange rectangle in XY plane, extruded in +Z
    # Global specimen now spans two corrugation waves:
    #   x from -L to +3L when global_n_waves = 2
```

```
    # ----------------------------------------------------
    G['flange_p1'] = (P['x_global_left'], -bf / 2.0)
    G['flange_p2'] = (P['x_global_right'],  bf / 2.0)

    # ----------------------------------------------------
    # Reference one-wave corrugated web outer profile in XY plane
    # This local reference wave stays unchanged and is still used
    # later for S-points / HSS / ENS around x ~= 0.
    # ----------------------------------------------------
    delta = 0.5 * tw * math.tan(0.5 * P['theta_rad'])

    G['p1']  = (-b - d,             -hr/2.0 + tw/2.0)
    G['p2']  = (-b/2.0 - d - delta, -hr/2.0 + tw/2.0)
    G['p3']  = (-b/2.0 - delta,      hr/2.0 + tw/2.0)
    G['p4']  = ( b/2.0 + delta,      hr/2.0 + tw/2.0)
    G['p5']  = ( b/2.0 + d + delta, -hr/2.0 + tw/2.0)
    G['p6']  = ( b + d,             -hr/2.0 + tw/2.0)

    G['p7']  = ( b + d,             -hr/2.0 - tw/2.0)
    G['p8']  = ( b/2.0 + d - delta, -hr/2.0 - tw/2.0)
    G['p9']  = ( b/2.0 - delta,      hr/2.0 - tw/2.0)
    G['p10'] = (-b/2.0 + delta,      hr/2.0 - tw/2.0)
    G['p11'] = (-b/2.0 - d + delta, -hr/2.0 - tw/2.0)
    G['p12'] = (-b - d,             -hr/2.0 - tw/2.0)

    return G

# ========================================================
# 4. BUILD GLOBAL GEOMETRY
# ========================================================
def build_flange_part(model, P, G):
    sketch_name = '__flange_profile__'
    part_name = 'Flange'

    if sketch_name in model.sketches.keys():
        del model.sketches[sketch_name]

    x1 = P['x_global_left']
    x2 = P['x_global_right']

    s = model.ConstrainedSketch(
        name=sketch_name,
        sheetSize=2.0 * (x2 - x1 + P['q'])
    )
    s.rectangle(
        point1=(x1, -P['bf'] / 2.0),
        point2=(x2,  P['bf'] / 2.0)
    )

    part_obj = model.Part(
        name=part_name,
        dimensionality=THREE_D,
        type=DEFORMABLE_BODY
    )

    part_obj.BaseSolidExtrude(
        sketch=s,
        depth=P['tf']
    )

```

```
    del model.sketches[sketch_name]
    return part_obj


def build_flange_part_one_wave_local(model, P):
    sketch_name = '__flange_profile_local__'
    part_name = 'Flange'

    if sketch_name in model.sketches.keys():
        del model.sketches[sketch_name]

    s = model.ConstrainedSketch(name=sketch_name, sheetSize=4.0 * P['L'])
    s.rectangle(point1=(-P['L'], -P['bf'] / 2.0), point2=(P['L'], P['bf'] / 2.0))

    part_obj = model.Part(
        name=part_name,
        dimensionality=THREE_D,
        type=DEFORMABLE_BODY
    )

    part_obj.BaseSolidExtrude(
        sketch=s,
        depth=P['tf']
    )

    del model.sketches[sketch_name]
    return part_obj


def assemble_and_merge_local_source(model, flange_part, web_part, P):
    a = model.rootAssembly
    a.DatumCsysByDefault(CARTESIAN)

    a.Instance(name='Flange-1', part=flange_part, dependent=ON)
    a.Instance(name='Web-1', part=web_part, dependent=ON)
    a.translate(instanceList=('Web-1',), vector=(0.0, 0.0, P['tf']))

    a.InstanceFromBooleanMerge(
        name='Specimen',
        instances=(
            a.instances['Flange-1'],
            a.instances['Web-1']
        ),
        keepIntersections=ON,
        originalInstances=SUPPRESS,
        domain=GEOMETRY
    )

    return model.parts['Specimen']


def build_local_source_model(param_overrides=None):
    P = get_params(param_overrides)
    G = get_geometry_points(P)

    model = recreate_model(GLOBAL_LOCAL_SOURCE_MODEL_NAME)

    flange_part = build_flange_part_one_wave_local(model, P)
    web_part = build_web_part(model, P, G)
    specimen_part = assemble_and_merge_local_source(model, flange_part, web_part, P)
```

```
816
817     assign_global_material_and_section(model, specimen_part, P)
818     create_global_step(model)
819
820     S = get_global_s_points(P, G)
821
822     return model, specimen_part, P, G, S
823
824
825 def build_web_part(model, P, G):
826     sketch_name = '__web_profile__'
827     part_name = 'Web'
828
829     if sketch_name in model.sketches.keys():
830         del model.sketches[sketch_name]
831
832     s = model.ConstrainedSketch(name=sketch_name, sheetSize=4.0 * P['q'])
833
834     # ------------------------------------------------------
835     # Draw web contour lines
836     # ------------------------------------------------------
837     s.Line(point1=G['p1'],  point2=G['p2'])
838     s.Line(point1=G['p2'],  point2=G['p3'])
839     s.Line(point1=G['p3'],  point2=G['p4'])
840     s.Line(point1=G['p4'],  point2=G['p5'])
841     s.Line(point1=G['p5'],  point2=G['p6'])
842     s.Line(point1=G['p6'],  point2=G['p7'])
843     s.Line(point1=G['p7'],  point2=G['p8'])
844     s.Line(point1=G['p8'],  point2=G['p9'])
845     s.Line(point1=G['p9'],  point2=G['p10'])
846     s.Line(point1=G['p10'], point2=G['p11'])
847     s.Line(point1=G['p11'], point2=G['p12'])
848     s.Line(point1=G['p12'], point2=G['p1'])
849
850     # ------------------------------------------------------
851     # Simple sketch constraints
852     # ------------------------------------------------------
853     s.HorizontalConstraint(entity=s.geometry[2], addUndoState=False)
854     s.HorizontalConstraint(entity=s.geometry[4], addUndoState=False)
855     s.HorizontalConstraint(entity=s.geometry[6], addUndoState=False)
856     s.VerticalConstraint(entity=s.geometry[7], addUndoState=False)
857     s.HorizontalConstraint(entity=s.geometry[8], addUndoState=False)
858     s.HorizontalConstraint(entity=s.geometry[10], addUndoState=False)
859     s.HorizontalConstraint(entity=s.geometry[12], addUndoState=False)
860     s.VerticalConstraint(entity=s.geometry[13], addUndoState=False)
861
862     # ------------------------------------------------------
863     # Fillet helper points
864     # ------------------------------------------------------
865     near_2_a  = point_on_line(G['p1'],  G['p2'],  0.8)
866     near_2_b  = point_on_line(G['p2'],  G['p3'],  0.2)
867
868     near_3_a  = point_on_line(G['p2'],  G['p3'],  0.8)
869     near_3_b  = point_on_line(G['p3'],  G['p4'],  0.2)
870
871     near_4_a  = point_on_line(G['p3'],  G['p4'],  0.8)
872     near_4_b  = point_on_line(G['p4'],  G['p5'],  0.2)
873
874     near_5_a  = point_on_line(G['p4'],  G['p5'],  0.8)
875     near_5_b  = point_on_line(G['p5'],  G['p6'],  0.2)
```

```
876
877     near_8_a  = point_on_line(G['p7'],  G['p8'],  0.8)
878     near_8_b  = point_on_line(G['p8'],  G['p9'],  0.2)
879
880     near_9_a  = point_on_line(G['p8'],  G['p9'],  0.8)
881     near_9_b  = point_on_line(G['p9'],  G['p10'], 0.2)
882
883     near_10_a = point_on_line(G['p9'],  G['p10'], 0.8)
884     near_10_b = point_on_line(G['p10'], G['p11'], 0.2)
885
886     near_11_a = point_on_line(G['p10'], G['p11'], 0.8)
887     near_11_b = point_on_line(G['p11'], G['p12'], 0.2)
888
889     # ----------------------------------------------------
890     # Apply fillets
891     # ----------------------------------------------------
892     s.FilletByRadius(curve1=s.geometry[2],  curve2=s.geometry[3],
893                      nearPoint1=near_2_a,  nearPoint2=near_2_b,  radius=P['R'])
894
895     s.FilletByRadius(curve1=s.geometry[3],  curve2=s.geometry[4],
896                      nearPoint1=near_3_a,  nearPoint2=near_3_b,  radius=P['R'])
897
898     s.FilletByRadius(curve1=s.geometry[4],  curve2=s.geometry[5],
899                      nearPoint1=near_4_a,  nearPoint2=near_4_b,  radius=P['R'])
900
901     s.FilletByRadius(curve1=s.geometry[5],  curve2=s.geometry[6],
902                      nearPoint1=near_5_a,  nearPoint2=near_5_b,  radius=P['R'])
903
904     s.FilletByRadius(curve1=s.geometry[8],  curve2=s.geometry[9],
905                      nearPoint1=near_8_a,  nearPoint2=near_8_b,  radius=P['R'])
906
907     s.FilletByRadius(curve1=s.geometry[9],  curve2=s.geometry[10],
908                      nearPoint1=near_9_a,  nearPoint2=near_9_b,  radius=P['R'])
909
910     s.FilletByRadius(curve1=s.geometry[10], curve2=s.geometry[11],
911                      nearPoint1=near_10_a, nearPoint2=near_10_b, radius=P['R'])
912
913     s.FilletByRadius(curve1=s.geometry[11], curve2=s.geometry[12],
914                      nearPoint1=near_11_a, nearPoint2=near_11_b, radius=P['R'])
915
916     part_obj = model.Part(
917         name=part_name,
918         dimensionality=THREE_D,
919         type=DEFORMABLE_BODY
920     )
921
922     part_obj.BaseSolidExtrude(
923         sketch=s,
924         depth=P['hw']
925     )
926
927     del model.sketches[sketch_name]
928     return part_obj
929
930
931 def assemble_and_merge_global(model, flange_part, web_part, P):
932     a = model.rootAssembly
933     a.DatumCsysByDefault(CARTESIAN)
934
935     a.Instance(name='Flange-1', part=flange_part, dependent=ON)
```

```
936
937         # first wave (original position)
938         a.Instance(name='Web-1', part=web_part, dependent=ON)
939         a.translate(instanceList=('Web-1',), vector=(0.0, 0.0, P['tf']))
940
941         # second wave shifted by +q = +2L
942         a.Instance(name='Web-2', part=web_part, dependent=ON)
943         a.translate(instanceList=('Web-2',), vector=(P['q'], 0.0, P['tf']))
944
945         a.InstanceFromBooleanMerge(
946             name='Specimen',
947             instances=(
948                 a.instances['Flange-1'],
949                 a.instances['Web-1'],
950                 a.instances['Web-2']
951             ),
952             keepIntersections=ON,
953             originalInstances=SUPPRESS,
954             domain=GEOMETRY
955         )
956
957         return model.parts['Specimen']
958
959     def build_global_geometry(param_overrides=None):
960         P = get_params(param_overrides)
961         G = get_geometry_points(P)
962
963         model = recreate_model(GLOBAL_MODEL_NAME)
964
965         flange_part = build_flange_part(model, P, G)
966         web_part = build_web_part(model, P, G)
967         specimen_part = assemble_and_merge_global(model, flange_part, web_part, P)
968
969         return model, specimen_part, P, G
970
971     # ============================================================
972     # 5. GLOBAL S-POINTS
973     # ============================================================
974
975     def get_global_s_points(P, G):
976         S = {}
977
978         trim = P['R'] * math.tan(P['theta_rad'] / 2.0)
979
980         # S-points = tangent points between fillet and inclined line
981         S['SP3']  = point_from_end(G['p2'],  G['p3'],  trim)
982         S['SP4']  = point_on_line_by_distance(G['p4'], G['p5'], trim)
983         S['SP8']  = point_on_line_by_distance(G['p8'], G['p9'], trim)
984         S['SP11'] = point_from_end(G['p10'], G['p11'], trim)
985
986         return S
987
988     # ============================================================
989     # 6. GLOBAL MATERIAL / SECTION / STEP
990     # ============================================================
991     def assign_global_material_and_section(model, specimen_part, P):
992         material_name = 'Steel'
993         section_name = 'SteelSection'
994         set_name = 'SET_ALL_CELLS'
995
```

```
    # ----------------------------------------------------
    # Material
    # ----------------------------------------------------
    if material_name not in model.materials.keys():
        model.Material(name=material_name)
        model.materials[material_name].Elastic(table=((P['E'], P['nu']), ))

    # ----------------------------------------------------
    # Section
    # ----------------------------------------------------
    if section_name not in model.sections.keys():
        model.HomogeneousSolidSection(
            name=section_name,
            material=material_name,
            thickness=None
        )

    # ----------------------------------------------------
    # Region = all cells of specimen
    # ----------------------------------------------------
    specimen_cells = specimen_part.cells[:]

    if set_name in specimen_part.sets.keys():
        del specimen_part.sets[set_name]

    specimen_part.Set(
        name=set_name,
        cells=specimen_cells
    )

    region = specimen_part.sets[set_name]

    specimen_part.SectionAssignment(
        region=region,
        sectionName=section_name,
        offset=0.0,
        offsetType=MIDDLE_SURFACE,
        offsetField='',
        thicknessAssignment=FROM_SECTION
    )

def create_global_step(model):
    model.StaticStep(
        name='Step-1',
        previous='Initial'
    )

def build_global_model(param_overrides=None):
    model, specimen_part, P, G = build_global_geometry(param_overrides)

    assign_global_material_and_section(model, specimen_part, P)
    create_global_step(model)

    S = get_global_s_points(P, G)

    return model, specimen_part, P, G, S


# ======================================================
# 7. BUILD HSS MODEL - COPY + LOCAL CUTS
```

```python
# ======================================================
def build_hss_model_from_global(P):
    # IMPORTANT:
    # HSS is copied from a stable local one-wave source model,
    # not from the two-wave global model.
    build_local_source_model(P)

    if HSS_MODEL_NAME in mdb.models.keys():
        del mdb.models[HSS_MODEL_NAME]

    mdb.Model(
        name=HSS_MODEL_NAME,
        objectToCopy=mdb.models[GLOBAL_LOCAL_SOURCE_MODEL_NAME]
    )

    model_hss = mdb.models[HSS_MODEL_NAME]
    specimen_hss = model_hss.parts['Specimen']

    return model_hss, specimen_hss


def get_hss_cut_parameters(P):
    C = {}

    # keep same logic as your previous model
    C['cut1_offset'] = P['b'] / 4.0
    C['cut2_offset'] = P['L']

    # very large cut box so it always covers full section
    C['cut_box_half'] = 5.0 * max(P['bf'], P['tf'] + P['hw'], P['hr'], P['L'])
    C['sheet_size_cut'] = 2.5 * C['cut_box_half']

    C['z_mid'] = 0.5 * (P['tf'] + P['hw'])

    return C

def get_hss_run_cut_window(P, S):
    """
    Smaller HSS window around SP4 and SP8.
    The margin must be larger than the HSS extraction distances
    and larger than the ENS trim margin.
    """

    margin = max(
        2.0 * P['tf'],
        1.0 * P['R'],
        40.0
    )

    x_sp_min = min(S['SP4'][0], S['SP8'][0])
    x_sp_max = max(S['SP4'][0], S['SP8'][0])

    x_left = x_sp_min - margin
    x_right = x_sp_max + margin

    # keep inside one-wave local source
    x_left = max(x_left, -P['L'])
    x_right = min(x_right, P['L'])

    log_print('-----------------------------------')
```

```
    log_print('NEW HSS RUN CUT WINDOW')
    log_print('SP min x = %.3f' % x_sp_min)
    log_print('SP max x = %.3f' % x_sp_max)
    log_print('margin   = %.3f' % margin)
    log_print('x_left   = %.3f' % x_left)
    log_print('x_right  = %.3f' % x_right)
    log_print('width    = %.3f' % (x_right - x_left))
    log_print('-----------------------------------')

    return {
        'x_left': x_left,
        'x_right': x_right,
        'margin': margin
    }

def get_hss_run_cut_window_from_ens(P, W4, W8):
    """
    HSS run window based on ENS trim windows (safer).
    """

    extra_margin = max(
        1.0 * P['tf'],
        0.5 * P['R'],
        30.0
    )

    x_left = min(W4['x_left'], W8['x_left']) - extra_margin
    x_right = max(W4['x_right'], W8['x_right']) + extra_margin

    x_left = max(x_left, -P['L'])
    x_right = min(x_right, P['L'])

    log_print('-----------------------------------', level=2)
    log_print('NEW HSS RUN CUT WINDOW FROM ENS', level=2)
    log_print('ENS min x = %.3f' % min(W4['x_left'], W8['x_left']), level=2)
    log_print('ENS max x = %.3f' % max(W4['x_right'], W8['x_right']), level=2)
    log_print('extra margin = %.3f' % extra_margin, level=2)
    log_print('x_left       = %.3f' % x_left, level=2)
    log_print('x_right      = %.3f' % x_right, level=2)
    log_print('width        = %.3f' % (x_right - x_left), level=2)
    log_print('-----------------------------------', level=2)

    return {
        'x_left': x_left,
        'x_right': x_right
    }

def cut_hss_with_plane(model_hss, specimen_hss, datum_plane, up_edge,
                       sketch_origin, cut_box_half,
                       sheet_size_cut, plane_side, flip_extrude, sketch_name):

    if sketch_name in model_hss.sketches.keys():
        del model_hss.sketches[sketch_name]

    model_hss.ConstrainedSketch(
        name=sketch_name,
        sheetSize=sheet_size_cut,
        transform=specimen_hss.MakeSketchTransform(
            sketchPlane=datum_plane,
            sketchPlaneSide=plane_side,
```

```
            sketchUpEdge=up_edge,
            sketchOrientation=RIGHT,
            origin=sketch_origin
        )
    )

    s = model_hss.sketches[sketch_name]

    s.rectangle(
        point1=(-cut_box_half, -cut_box_half),
        point2=( cut_box_half,  cut_box_half)
    )

    specimen_hss.CutExtrude(
        sketch=s,
        sketchPlane=datum_plane,
        sketchPlaneSide=plane_side,
        sketchUpEdge=up_edge,
        sketchOrientation=RIGHT,
        flipExtrudeDirection=flip_extrude
    )

    del model_hss.sketches[sketch_name]


def build_hss_local_cut_model(P):
    model_hss, specimen_hss = build_hss_model_from_global(P)
    C = get_hss_cut_parameters(P)

    # -----------------------------------------------------
    # Robust datum axis for sketch orientation
    # -----------------------------------------------------
    z_axis = get_or_create_datum_axis_z(specimen_hss)

    # -----------------------------------------------------
    # Datum planes at x = constant
    # -----------------------------------------------------
    x_cut_1 = P['L'] - C['cut1_offset']
    x_cut_2 = P['L'] - C['cut2_offset']

    datum1 = create_datum_plane_x_constant(
        part_obj=specimen_hss,
        x_value=x_cut_1
    )

    datum2 = create_datum_plane_x_constant(
        part_obj=specimen_hss,
        x_value=x_cut_2
    )

    # -----------------------------------------------------
    # Cut 1 : remove right side
    # -----------------------------------------------------
    cut_hss_with_plane(
        model_hss=model_hss,
        specimen_hss=specimen_hss,
        datum_plane=datum1,
        up_edge=z_axis,
        sketch_origin=(x_cut_1, 0.0, C['z_mid']),
        cut_box_half=C['cut_box_half'],
```

```
        sheet_size_cut=C['sheet_size_cut'],
        plane_side=SIDE1,
        flip_extrude=ON,
        sketch_name='__cut_profile_1__'
    )

    # ----------------------------------------------------
    # Cut 2 : remove left side
    # ----------------------------------------------------
    cut_hss_with_plane(
        model_hss=model_hss,
        specimen_hss=specimen_hss,
        datum_plane=datum2,
        up_edge=z_axis,
        sketch_origin=(x_cut_2, 0.0, C['z_mid']),
        cut_box_half=C['cut_box_half'],
        sheet_size_cut=C['sheet_size_cut'],
        plane_side=SIDE2,
        flip_extrude=ON,
        sketch_name='__cut_profile_2__'
    )

    return model_hss, specimen_hss, C

def save_hss_source_model_for_ens():
    """
    Save the current wide HSS model as source for ENS,
    before trimming the HSS run model.
    """
    if HSS_SOURCE_MODEL_NAME in mdb.models.keys():
        del mdb.models[HSS_SOURCE_MODEL_NAME]

    mdb.Model(
        name=HSS_SOURCE_MODEL_NAME,
        objectToCopy=mdb.models[HSS_MODEL_NAME]
    )

    log_print('HSS source model saved for ENS.')


def trim_hss_run_model(P, HSSW):
    """
    Trim HSS-Model to smaller run window.
    """
    model_hss = mdb.models[HSS_MODEL_NAME]
    p = model_hss.parts['Specimen']

    z_mid = 0.5 * (P['tf'] + P['hw'])

    x_left = HSSW['x_left']
    x_right = HSSW['x_right']

    cut_part_with_x_plane_keep_right(
        model_obj=model_hss,
        part_obj=p,
        P=P,
        x_cut=x_left,
        z_mid=z_mid,
        sketch_name='__hss_run_cut_left__'
    )
```

```
    p.regenerate()

    cut_part_with_x_plane_keep_left(
        model_obj=model_hss,
        part_obj=p,
        P=P,
        x_cut=x_right,
        z_mid=z_mid,
        sketch_name='__hss_run_cut_right__'
    )
    p.regenerate()

    P['hss_run_x_left'] = x_left
    P['hss_run_x_right'] = x_right

    C = get_hss_cut_parameters(P)
    C['x_left'] = x_left
    C['x_right'] = x_right

    log_print('-----------------------------------')
    log_print('HSS run model trimmed')
    log_print('x_left  = %.3f' % x_left)
    log_print('x_right = %.3f' % x_right)
    log_print('width   = %.3f' % (x_right - x_left))
    log_print('-----------------------------------')

    return model_hss, p, C


# ============================================================
# 8. HSS DOUBLE FILLET WELDS
# ============================================================
def get_hss_weld_reference_points(P):
    W = {}

    # side 1 root
    # based on successful manual macro:
    # origin y = (hr - tw)/2
    W['side_1_root'] = (
        0.0,
        0.5 * (P['hr'] - P['tw']),
        P['tf']
    )

    # side 2 root
    # based on successful manual macro:
    # origin y = (hr + tw)/2
    W['side_2_root'] = (
        0.0,
        0.5 * (P['hr'] + P['tw']),
        P['tf']
    )

    return W


def get_hss_fillet_profile_points(P, side):
    leg = P['weld_leg']
    sheet_size = 30.0 * leg

```

```
    if side == 'side_1':
        # from successful manual macro:
        # (0,0) -> (-leg,0) -> (0,+leg)
        return {
            'sheet_size': sheet_size,
            'p1': (0.0, 0.0),
            'p2': (-leg, 0.0),
            'p3': (0.0, leg)
        }

    elif side == 'side_2':
        # from successful manual macro:
        # (0,0) -> (-leg,0) -> (0,-leg)
        return {
            'sheet_size': sheet_size,
            'p1': (0.0, 0.0),
            'p2': (-leg, 0.0),
            'p3': (0.0, -leg)
        }

    else:
        raise ValueError('Unknown weld side: %s' % side)

def create_one_hss_fillet_weld(model_hss, specimen_hss,
                               weld_name,
                               root_point,
                               sketch_plane,
                               sketch_up_edge,
                               sweep_path,
                               profile_data):
    sketch_name = '__%s_profile__' % weld_name

    if sketch_name in model_hss.sketches.keys():
        del model_hss.sketches[sketch_name]

    model_hss.ConstrainedSketch(
        name=sketch_name,
        sheetSize=profile_data['sheet_size'],
        transform=specimen_hss.MakeSketchTransform(
            sketchPlane=sketch_plane,
            sketchPlaneSide=SIDE1,
            sketchUpEdge=sketch_up_edge,
            sketchOrientation=RIGHT,
            origin=root_point
        )
    )

    s = model_hss.sketches[sketch_name]

    # optional construction lines, similar to manual macro
    half = 0.5 * profile_data['sheet_size']
    s.ConstructionLine(point1=(-half, 0.0), point2=(half, 0.0))
    s.ConstructionLine(point1=(0.0, -half), point2=(0.0, half))

    specimen_hss.projectReferencesOntoSketch(
        filter=COPLANAR_EDGES,
        sketch=s
    )

    s.Line(point1=profile_data['p1'], point2=profile_data['p2'])
```

```
    s.Line(point1=profile_data['p2'], point2=profile_data['p3'])
    s.Line(point1=profile_data['p3'], point2=profile_data['p1'])

    specimen_hss.SolidSweep(
        path=sweep_path,
        profile=s,
        sketchOrientation=RIGHT,
        sketchPlane=sketch_plane,
        sketchUpEdge=sketch_up_edge
    )

    del model_hss.sketches[sketch_name]

def add_hss_side_1_weld(P):
    model_hss = mdb.models[HSS_MODEL_NAME]
    specimen_hss = model_hss.parts['Specimen']
    W = get_hss_weld_reference_points(P)

    profile_data = get_hss_fillet_profile_points(P, 'side_1')

    create_one_hss_fillet_weld(
        model_hss=model_hss,
        specimen_hss=specimen_hss,
        weld_name='hss_weld_side_1',
        root_point=W['side_1_root'],
        sketch_plane=specimen_hss.faces[0],
        sketch_up_edge=specimen_hss.edges[4],
        sweep_path=specimen_hss.edges.getSequenceFromMask(('[#f8000000 ]', ), ),
        profile_data=profile_data
    )


def add_hss_side_2_weld(P):
    model_hss = mdb.models[HSS_MODEL_NAME]
    specimen_hss = model_hss.parts['Specimen']
    W = get_hss_weld_reference_points(P)

    profile_data = get_hss_fillet_profile_points(P, 'side_2')

    create_one_hss_fillet_weld(
        model_hss=model_hss,
        specimen_hss=specimen_hss,
        weld_name='hss_weld_side_2',
        root_point=W['side_2_root'],
        sketch_plane=specimen_hss.faces[10],
        sketch_up_edge=specimen_hss.edges[31],
        sweep_path=specimen_hss.edges.getSequenceFromMask(('[#0 #1f00 ]', ), ),
        profile_data=profile_data
    )


def add_hss_double_fillet_welds(P):
    add_hss_side_1_weld(P)
    add_hss_side_2_weld(P)

# ============================================================
# 8A. HSS SIDE-FACE PARTITION FOR WELD EXCLUSION
#     (macro-based, parametric origins)
# ============================================================
def partition_all_hss_side_faces_for_weld(P, C):
```

```
    model_hss = mdb.models[HSS_MODEL_NAME]
    p = model_hss.parts['Specimen']

    leg = P['weld_leg']
    tf = P['tf']

    y_low  = 0.5 * (P['hr'] - P['tw'])
    y_high = 0.5 * (P['hr'] + P['tw'])
    x_right = P['L'] - C['cut1_offset']

    # ------------------------------------------------
    # 1) left lower
    # ------------------------------------------------
    model_hss.ConstrainedSketch(
        gridSpacing=1.5*leg,
        name='__profile__',
        sheetSize=60.0*leg,
        transform=p.MakeSketchTransform(
            sketchPlane=p.faces[20],
            sketchPlaneSide=SIDE1,
            sketchUpEdge=p.edges[57],
            sketchOrientation=RIGHT,
            origin=(0.0, y_low, tf)
        )
    )
    p.projectReferencesOntoSketch(
        filter=COPLANAR_EDGES,
        sketch=model_hss.sketches['__profile__']
    )
    s = model_hss.sketches['__profile__']
    s.Line(point1=(0.0, leg), point2=(0.0, 0.0))
    s.Line(point1=(0.0, 0.0), point2=(-leg, 0.0))
    p.PartitionFaceBySketch(
        faces=p.faces.getSequenceFromMask(('[#100000 ]',),),),
        sketch=s,
        sketchUpEdge=p.edges[57]
    )
    del model_hss.sketches['__profile__']

    # ------------------------------------------------
    # 2) left upper
    # ------------------------------------------------
    model_hss.ConstrainedSketch(
        gridSpacing=1.5*leg,
        name='__profile__',
        sheetSize=60.0*leg,
        transform=p.MakeSketchTransform(
            sketchPlane=p.faces[0],
            sketchPlaneSide=SIDE1,
            sketchUpEdge=p.edges[9],
            sketchOrientation=RIGHT,
            origin=(0.0, y_high, tf)
        )
    )
    p.projectReferencesOntoSketch(
        filter=COPLANAR_EDGES,
        sketch=model_hss.sketches['__profile__']
    )
    s = model_hss.sketches['__profile__']
    s.Line(point1=(-leg, 0.0), point2=(0.0, 0.0))
```

```
    s.Line(point1=(0.0, 0.0), point2=(0.0, -leg))
    p.PartitionFaceBySketch(
        faces=p.faces.getSequenceFromMask(('[#1 ]',),),
        sketch=s,
        sketchUpEdge=p.edges[9]
    )
    del model_hss.sketches['__profile__']

    # ------------------------------------------------
    # 3) right upper
    # ------------------------------------------------
    model_hss.ConstrainedSketch(
        gridSpacing=1.5*leg,
        name='__profile__',
        sheetSize=80.0*leg,
        transform=p.MakeSketchTransform(
            sketchPlane=p.faces[27],
            sketchPlaneSide=SIDE1,
            sketchUpEdge=p.edges[66],
            sketchOrientation=RIGHT,
            origin=(x_right, -y_high, tf)
        )
    )
    p.projectReferencesOntoSketch(
        filter=COPLANAR_EDGES,
        sketch=model_hss.sketches['__profile__']
    )
    s = model_hss.sketches['__profile__']
    s.Line(point1=(-leg, 0.0), point2=(0.0, 0.0))
    s.Line(point1=(0.0, 0.0), point2=(0.0, -leg))
    p.PartitionFaceBySketch(
        faces=p.faces.getSequenceFromMask(('[#8000000 ]',),),
        sketch=s,
        sketchUpEdge=p.edges[66]
    )
    del model_hss.sketches['__profile__']

    # ------------------------------------------------
    # 4) right lower
    # ------------------------------------------------
    model_hss.ConstrainedSketch(
        gridSpacing=1.5*leg,
        name='__profile__',
        sheetSize=60.0*leg,
        transform=p.MakeSketchTransform(
            sketchPlane=p.faces[0],
            sketchPlaneSide=SIDE1,
            sketchUpEdge=p.edges[5],
            sketchOrientation=RIGHT,
            origin=(x_right, -y_low, tf)
        )
    )
    p.projectReferencesOntoSketch(
        filter=COPLANAR_EDGES,
        sketch=model_hss.sketches['__profile__']
    )
    s = model_hss.sketches['__profile__']
    s.Line(point1=(0.0, leg), point2=(0.0, 0.0))
    s.Line(point1=(0.0, 0.0), point2=(-leg, 0.0))
    p.PartitionFaceBySketch(
```

```
        faces=p.faces.getSequenceFromMask(('[#1 ]',),),
        sketch=s,
        sketchUpEdge=p.edges[5]
    )
    del model_hss.sketches['__profile__']

    log_print('All HSS weld side-face partitions created successfully.', level=2)

# ============================================================
# 9. HSS TOE POINTS FROM GLOBAL S-POINTS
# ============================================================
def get_hss_toe_points(P, S):
    H = {}

    s = math.sin(P['theta_rad'])
    c = math.cos(P['theta_rad'])
    leg = P['weld_leg']

    # ------------------------------------------------------
    # Side near SP4
    # ------------------------------------------------------
    H['toe_sp4'] = (
        S['SP4'][0] + s * leg,
        S['SP4'][1] + c * leg,
        P['tf']
    )

    # ------------------------------------------------------
    # Side near SP8
    # ------------------------------------------------------
    H['toe_sp8'] = (
        S['SP8'][0] - s * leg,
        S['SP8'][1] - c * leg,
        P['tf']
    )

    return H

# ============================================================
# 10. HSS SAMPLING POINTS FROM TOE POINTS
# ============================================================
def get_hss_sampling_points(P, H):
    HS = {}

    d04 = 0.4 * P['tf']
    d09 = 0.9 * P['tf']
    d14 = 1.4 * P['tf']

    # ------------------------------------------------------
    # Side near SP4  -> move in +x
    # ------------------------------------------------------
    toe_sp4 = H['toe_sp4']

    HS['sp4_04'] = (toe_sp4[0] + d04, toe_sp4[1], toe_sp4[2])
    HS['sp4_09'] = (toe_sp4[0] + d09, toe_sp4[1], toe_sp4[2])
    HS['sp4_14'] = (toe_sp4[0] + d14, toe_sp4[1], toe_sp4[2])

    # ------------------------------------------------------
    # Side near SP8  -> move in -x
    # ------------------------------------------------------
```

```
    toe_sp8 = H['toe_sp8']

    HS['sp8_04'] = (toe_sp8[0] - d04, toe_sp8[1], toe_sp8[2])
    HS['sp8_09'] = (toe_sp8[0] - d09, toe_sp8[1], toe_sp8[2])
    HS['sp8_14'] = (toe_sp8[0] - d14, toe_sp8[1], toe_sp8[2])

    return HS

# ========================================================
# 10A. HSS LOCAL SAMPLING CHAINS
# ========================================================
def get_hss_chain_points_for_side(H, HS, side_name):
    """
    Return local chain points for one HSS side:
        p0 -> p1 -> p2 -> p3

    p0 = toe
    p1 = 0.4t
    p2 = 0.9t
    p3 = 1.4t
    """
    if side_name == 'SP4':
        return (
            H['toe_sp4'],
            HS['sp4_04'],
            HS['sp4_09'],
            HS['sp4_14']
        )

    elif side_name == 'SP8':
        return (
            H['toe_sp8'],
            HS['sp8_04'],
            HS['sp8_09'],
            HS['sp8_14']
        )

    else:
        raise ValueError('Unknown side_name: %s' % side_name)


def get_point_between(pa, pb, t):
    return (
        pa[0] + t * (pb[0] - pa[0]),
        pa[1] + t * (pb[1] - pa[1]),
        pa[2] + t * (pb[2] - pa[2])
    )


def get_vertex_xyz(part_obj, vertex_index):
    return part_obj.vertices[vertex_index].pointOn[0]


def get_edge_direction_and_length(part_obj, edge_obj):
    v_ids = edge_obj.getVertices()

    if len(v_ids) < 2:
        return (0.0, 0.0, 0.0), 0.0

    p1 = get_vertex_xyz(part_obj, v_ids[0])
```

```
    p2 = get_vertex_xyz(part_obj, v_ids[1])

    dx = p2[0] - p1[0]
    dy = p2[1] - p1[1]
    dz = p2[2] - p1[2]

    L = math.sqrt(dx * dx + dy * dy + dz * dz)

    if L < 1.0e-12:
        return (0.0, 0.0, 0.0), 0.0

    return (dx / L, dy / L, dz / L), L


def find_face_from_chain_interior(part_obj, p0, p3):
    """
    Pick a point inside the local face near the chain.
    """
    probe = get_point_between(p0, p3, 0.5)
    face_seq = part_obj.faces.findAt((probe,))
    return face_seq[0]

def create_hss_sampling_partition_on_face(model_name, P, p0, p1, p2, p3, label):
    model_hss = mdb.models[model_name]
    part_obj = model_hss.parts['Specimen']

    try:
        part_obj.regenerate()
    except:
        pass

    # choose target face from chain interior
    target_face = find_face_from_chain_interior(part_obj, p0, p3)

    # create datum points at the four chain points
    dp0 = part_obj.DatumPointByCoordinate(coords=p0)
    dp1 = part_obj.DatumPointByCoordinate(coords=p1)
    dp2 = part_obj.DatumPointByCoordinate(coords=p2)
    dp3 = part_obj.DatumPointByCoordinate(coords=p3)

    d0 = part_obj.datums[dp0.id]
    d1 = part_obj.datums[dp1.id]
    d2 = part_obj.datums[dp2.id]
    d3 = part_obj.datums[dp3.id]

    # partition 0-1
    part_obj.PartitionFaceByShortestPath(
        faces=(target_face,),
        point1=d0,
        point2=d1
    )

    try:
        part_obj.regenerate()
    except:
        pass

    # find face again after topology change
    target_face = find_face_from_chain_interior(part_obj, p1, p3)

```

```
    # partition 1-2
    part_obj.PartitionFaceByShortestPath(
        faces=(target_face,),
        point1=d1,
        point2=d2
    )

    try:
        part_obj.regenerate()
    except:
        pass

    # find face again after topology change
    target_face = find_face_from_chain_interior(part_obj, p2, p3)

    # partition 2-3
    part_obj.PartitionFaceByShortestPath(
        faces=(target_face,),
        point1=d2,
        point2=d3
    )

    try:
        part_obj.regenerate()
    except:
        pass

    log_print('HSS sampling partition created for %s' % label)


def create_hss_sampling_edge_sets(part_obj, p0, p1, p2, p3, prefix):
    mids = [
        get_point_between(p0, p1, 0.5),
        get_point_between(p1, p2, 0.5),
        get_point_between(p2, p3, 0.5),
    ]

    set_names = [
        prefix + '_SEG_01',
        prefix + '_SEG_12',
        prefix + '_SEG_23',
    ]

    for mid, set_name in zip(mids, set_names):
        edge_seq = part_obj.edges.findAt((mid,))

        if set_name in part_obj.sets.keys():
            del part_obj.sets[set_name]

        part_obj.Set(
            name=set_name,
            edges=edge_seq
        )


def apply_hss_sampling_chains(P, H, HS):
    model_hss = mdb.models[HSS_MODEL_NAME]
    part_obj = model_hss.parts['Specimen']

    # ----------------------------
```

```
    # SP4 side
    # ----------------------------
    p0, p1, p2, p3 = get_hss_chain_points_for_side(H, HS, 'SP4')

    create_hss_sampling_partition_on_face(
        model_name=HSS_MODEL_NAME,
        P=P,
        p0=p0, p1=p1, p2=p2, p3=p3,
        label='SP4'
    )

    try:
        part_obj.regenerate()
    except:
        pass

    create_hss_sampling_edge_sets(
        part_obj=part_obj,
        p0=p0, p1=p1, p2=p2, p3=p3,
        prefix='SET_HSS_SP4'
    )

    # ----------------------------
    # SP8 side
    # ----------------------------
    p0, p1, p2, p3 = get_hss_chain_points_for_side(H, HS, 'SP8')

    create_hss_sampling_partition_on_face(
        model_name=HSS_MODEL_NAME,
        P=P,
        p0=p0, p1=p1, p2=p2, p3=p3,
        label='SP8'
    )

    try:
        part_obj.regenerate()
    except:
        pass

    create_hss_sampling_edge_sets(
        part_obj=part_obj,
        p0=p0, p1=p1, p2=p2, p3=p3,
        prefix='SET_HSS_SP8'
    )

    try:
        part_obj.regenerate()
    except:
        pass


# ============================================================
# 11. BUILD ENS MODELS
# ============================================================
def copy_hss_to_ens_models():
    # delete old ENS models if they exist
    if ENS_SP4_MODEL_NAME in mdb.models.keys():
        del mdb.models[ENS_SP4_MODEL_NAME]

    # copy HSS -> ENS-SP4
```

```
    mdb.Model(
        name=ENS_SP4_MODEL_NAME,
        objectToCopy=mdb.models[HSS_SOURCE_MODEL_NAME]
    )

    model_ens_sp4 = mdb.models[ENS_SP4_MODEL_NAME]
    specimen_sp4 = model_ens_sp4.parts['Specimen']

    return model_ens_sp4, specimen_sp4


def suppress_hss_weld_features_in_part(part_obj):
    # try common feature names first
    candidate_names = ['Solid sweep-1', 'Solid sweep-2']

    for feat_name in candidate_names:
        if feat_name in part_obj.features.keys():
            try:
                part_obj.features[feat_name].suppress()
            except:
                pass


def build_initial_ens_models():
    model_ens_sp4, specimen_sp4 = copy_hss_to_ens_models()

    # suppress HSS fillet welds in ENS-SP4 model
    suppress_hss_weld_features_in_part(specimen_sp4)

    # regenerate part
    specimen_sp4.regenerate()

    return model_ens_sp4, specimen_sp4

# ============================================================
# 12. ENS NOTCH WELDS
# ============================================================
def get_ens_notch_reference_points(P):
    E = {}

    # side 1 root
    E['side_1_root'] = (
        0.0,
        0.5 * (P['hr'] - P['tw']),
        P['tf']
    )

    # side 2 root
    E['side_2_root'] = (
        0.0,
        0.5 * (P['hr'] + P['tw']),
        P['tf']
    )

    return E


def get_ens_notch_profile_points(P, side):
    leg = P['weld_leg']
    r = P['notch_radius']
```

```
    sqrt2 = P['sqrt2']

    dx = r * (1.0 - 1.0 / sqrt2)
    dy = r * (sqrt2 - 1.0)

    sheet_size = 30.0 * max(leg, r)

    if side == 'side_1':
        # based on successful manual macro for side 1
        return {
            'sheet_size': sheet_size,
            'p1': (0.0, 0.0),
            'p2': (-leg, 0.0),
            'p3': (-dx, leg - dx),
            'p4': (0.0, leg + dy)
        }

    elif side == 'side_2':
        # based on successful manual macro for side 2
        return {
            'sheet_size': sheet_size,
            'p1': (0.0, 0.0),
            'p2': (-leg, 0.0),
            'p3': (-dx, -leg + dx),
            'p4': (0.0, -(leg + dy))
        }

    else:
        raise ValueError('Unknown ENS notch side: %s' % side)


def create_one_ens_notch_weld(model_ens, specimen_ens,
                              notch_name,
                              root_point,
                              sketch_plane,
                              sketch_up_edge,
                              sweep_path,
                              profile_data):
    sketch_name = '__%s_profile__' % notch_name

    if sketch_name in model_ens.sketches.keys():
        del model_ens.sketches[sketch_name]

    model_ens.ConstrainedSketch(
        name=sketch_name,
        sheetSize=profile_data['sheet_size'],
        transform=specimen_ens.MakeSketchTransform(
            sketchPlane=sketch_plane,
            sketchPlaneSide=SIDE1,
            sketchUpEdge=sketch_up_edge,
            sketchOrientation=RIGHT,
            origin=root_point
        )
    )

    s = model_ens.sketches[sketch_name]

    half = 0.5 * profile_data['sheet_size']
    s.ConstructionLine(point1=(-half, 0.0), point2=(half, 0.0))
    s.ConstructionLine(point1=(0.0, -half), point2=(0.0, half))
```

```
2016
2017        specimen_ens.projectReferencesOntoSketch(
2018            filter=COPLANAR_EDGES,
2019            sketch=s
2020        )
2021
2022        # first line
2023        s.Line(point1=profile_data['p1'], point2=profile_data['p2'])
2024
2025        # second line
2026        line_2 = s.Line(point1=profile_data['p2'], point2=profile_data['p3'])
2027
2028        # tangent arc
2029        s.ArcByStartEndTangent(
2030            entity=line_2,
2031            point1=profile_data['p3'],
2032            point2=profile_data['p4']
2033        )
2034
2035        # closing line
2036        s.Line(point1=profile_data['p1'], point2=profile_data['p4'])
2037
2038        specimen_ens.SolidSweep(
2039            path=sweep_path,
2040            profile=s,
2041            sketchOrientation=RIGHT,
2042            sketchPlane=sketch_plane,
2043            sketchUpEdge=sketch_up_edge
2044        )
2045
2046        del model_ens.sketches[sketch_name]
2047
2048
2049    def add_ens_side_1_notch_to_model(model_name, P):
2050        model_ens = mdb.models[model_name]
2051        specimen = model_ens.parts['Specimen']
2052        E = get_ens_notch_reference_points(P)
2053
2054        profile_data = get_ens_notch_profile_points(P, 'side_1')
2055
2056        create_one_ens_notch_weld(
2057            model_ens=model_ens,
2058            specimen_ens=specimen,
2059            notch_name='ens_side_1_notch',
2060            root_point=E['side_1_root'],
2061            sketch_plane=specimen.faces[0],
2062            sketch_up_edge=specimen.edges[4],
2063            sweep_path=specimen.edges.getSequenceFromMask(('[#f8000000 ]', ), ),
2064            profile_data=profile_data
2065        )
2066
2067
2068    def add_ens_side_2_notch_to_model(model_name, P):
2069        model_ens = mdb.models[model_name]
2070        specimen = model_ens.parts['Specimen']
2071        E = get_ens_notch_reference_points(P)
2072
2073        profile_data = get_ens_notch_profile_points(P, 'side_2')
2074
2075        create_one_ens_notch_weld(
```

```
        model_ens=model_ens,
        specimen_ens=specimen,
        notch_name='ens_side_2_notch',
        root_point=E['side_2_root'],
        sketch_plane=specimen.faces[14],
        sketch_up_edge=specimen.edges[40],
        sweep_path=specimen.edges.getSequenceFromMask(('[#0 #3e0000 ]', ), ),
        profile_data=profile_data
    )


def add_ens_double_notch_welds_to_model(model_name, P):
    add_ens_side_1_notch_to_model(model_name, P)
    add_ens_side_2_notch_to_model(model_name, P)

# ============================================================
# 12A-12B. ENS HELPER ARC + LOCAL SWEEP PARTITIONS
# ============================================================
def get_ens_helper_arc_points(P, side):
    """
    2 mm helper arc with SAME center as the 1 mm notch arc.
    """
    r_notch = P['notch_radius']
    r_helper = 2.0
    sqrt2 = P['sqrt2']
    a = P['weld_leg']

    if side == 'side_1':
        arc_center = (-r_notch, a + (sqrt2 - 1.0))
    elif side == 'side_2':
        arc_center = (-r_notch, -(a + (sqrt2 - 1.0)))
    else:
        raise ValueError('Unknown ENS helper arc side: %s' % side)

    arc_point = (arc_center[0] + r_helper, arc_center[1])
    trim_point = (arc_center[0] - r_helper, arc_center[1])

    return {
        'arc_center': arc_center,
        'arc_point': arc_point,
        'trim_point': trim_point
    }


def create_one_ens_helper_arc_partition(model_name,
                                        sketch_name,
                                        sketch_plane,
                                        sketch_up_edge,
                                        sketch_origin,
                                        faces_for_partition,
                                        side,
                                        P,
                                        sheet_size,
                                        grid_spacing,
                                        label):
    model_obj = mdb.models[model_name]
    part_obj = model_obj.parts['Specimen']

    if sketch_name in model_obj.sketches.keys():
        del model_obj.sketches[sketch_name]
```

```
2136
2137        data = get_ens_helper_arc_points(P, side)
2138
2139        model_obj.ConstrainedSketch(
2140            name=sketch_name,
2141            sheetSize=sheet_size,
2142            gridSpacing=grid_spacing,
2143            transform=part_obj.MakeSketchTransform(
2144                sketchPlane=sketch_plane,
2145                sketchPlaneSide=SIDE1,
2146                sketchUpEdge=sketch_up_edge,
2147                sketchOrientation=RIGHT,
2148                origin=sketch_origin
2149            )
2150        )
2151
2152        s = model_obj.sketches[sketch_name]
2153
2154        part_obj.projectReferencesOntoSketch(
2155            filter=COPLANAR_EDGES,
2156            sketch=s
2157        )
2158
2159        circle_obj = s.CircleByCenterPerimeter(
2160            center=data['arc_center'],
2161            point1=data['arc_point']
2162        )
2163
2164        s.autoTrimCurve(
2165            curve1=circle_obj,
2166            point1=data['trim_point']
2167        )
2168
2169        part_obj.PartitionFaceBySketch(
2170            faces=faces_for_partition,
2171            sketch=s,
2172            sketchUpEdge=sketch_up_edge
2173        )
2174
2175        del model_obj.sketches[sketch_name]
2176
2177        try:
2178            part_obj.regenerate()
2179        except:
2180            pass
2181
2182        log_print(label + ' created successfully.', level=2)
2183
2184
2185    def apply_one_ens_sweep_partition(model_name, cell_mask, edges_tuple, sweep_edge_id, label):
2186
2187        model_obj = mdb.models[model_name]
2188        part_obj = model_obj.parts['Specimen']
2189
2190        try:
2191            part_obj.regenerate()
2192        except:
2193            pass
2194
2195        try:
```

```
        cells_region = part_obj.cells[:]

        part_obj.PartitionCellBySweepEdge(
            cells=cells_region,
            edges=edges_tuple,
            sweepPath=part_obj.edges[sweep_edge_id]
        )

        try:
            part_obj.regenerate()
        except:
            pass

        log_print(label + ' created successfully.', level=2)
        log_print('%s cells after %s = %d'
                  % (model_name, label, len(part_obj.cells)), level=2)

    except Exception as e:
        log_print(label + ' skipped. Reason: %s' % str(e), level=2)


def add_all_local_partitions_to_ens_sp4_model(P):
    model_name = ENS_SP4_MODEL_NAME
    part_obj = mdb.models[model_name].parts['Specimen']

    # ----------------------------------
    # SIDE 1 : helper arc
    # ----------------------------------
    create_one_ens_helper_arc_partition(
        model_name=model_name,
        sketch_name='__ens_sp4_helper_arc_side_1__',
        sketch_plane=part_obj.faces[29],
        sketch_up_edge=part_obj.edges[77],
        sketch_origin=(0.0, 0.5 * (P['hr'] - P['tw']), P['tf']),
        faces_for_partition=part_obj.faces.getSequenceFromMask(('[#20000000 ]',), ),
        side='side_1',
        P=P,
        sheet_size=314.98,
        grid_spacing=7.87,
        label='ENS-SP4 helper arc partition side 1'
    )

    # ----------------------------------
    # SIDE 1 : sweeps
    # ----------------------------------
    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0], part_obj.edges[1]),
        sweep_edge_id=73,
        label='ENS-SP4 sweep partition side 1 - 1'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=76,
        label='ENS-SP4 sweep partition side 1 - 2'
    )
```

```
2256
2257    apply_one_ens_sweep_partition(
2258        model_name=model_name,
2259        cell_mask='[#1 ]',
2260        edges_tuple=(part_obj.edges[0],),
2261        sweep_edge_id=75,
2262        label='ENS-SP4 sweep partition side 1 - 3'
2263    )
2264
2265    apply_one_ens_sweep_partition(
2266        model_name=model_name,
2267        cell_mask='[#1 ]',
2268        edges_tuple=(part_obj.edges[0],),
2269        sweep_edge_id=76,
2270        label='ENS-SP4 sweep partition side 1 - 4'
2271    )
2272
2273    apply_one_ens_sweep_partition(
2274        model_name=model_name,
2275        cell_mask='[#1 ]',
2276        edges_tuple=(part_obj.edges[0],),
2277        sweep_edge_id=76,
2278        label='ENS-SP4 sweep partition side 1 - 5'
2279    )
2280
2281    # ----------------------------------
2282    # SIDE 2 : helper arc
2283    # ----------------------------------
2284    create_one_ens_helper_arc_partition(
2285        model_name=model_name,
2286        sketch_name='__ens_sp4_helper_arc_side_2__',
2287        sketch_plane=part_obj.faces[12],
2288        sketch_up_edge=part_obj.edges[54],
2289        sketch_origin=(0.0, 0.5 * (P['hr'] + P['tw']), P['tf']),
2290        faces_for_partition=part_obj.faces.getSequenceFromMask(('[#1000 ]',), ),
2291        side='side_2',
2292        P=P,
2293        sheet_size=314.98,
2294        grid_spacing=7.87,
2295        label='ENS-SP4 helper arc partition side 2'
2296    )
2297
2298    # ----------------------------------
2299    # SIDE 2 : sweeps
2300    # ----------------------------------
2301    apply_one_ens_sweep_partition(
2302        model_name=model_name,
2303        cell_mask='[#1 ]',
2304        edges_tuple=(part_obj.edges[1], part_obj.edges[2]),
2305        sweep_edge_id=83,
2306        label='ENS-SP4 sweep partition side 2 - 1'
2307    )
2308
2309    apply_one_ens_sweep_partition(
2310        model_name=model_name,
2311        cell_mask='[#1 ]',
2312        edges_tuple=(part_obj.edges[0],),
2313        sweep_edge_id=84,
2314        label='ENS-SP4 sweep partition side 2 - 2'
2315    )
```

```
    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=82,
        label='ENS-SP4 sweep partition side 2 - 3'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=86,
        label='ENS-SP4 sweep partition side 2 - 4'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=87,
        label='ENS-SP4 sweep partition side 2 - 5'
    )


def add_all_local_partitions_to_ens_sp8_model(P):
    model_name = ENS_SP8_MODEL_NAME
    part_obj = mdb.models[model_name].parts['Specimen']

    # ------------------------------------
    # SIDE 1 : helper arc
    # ------------------------------------
    create_one_ens_helper_arc_partition(
        model_name=model_name,
        sketch_name='__ens_sp8_helper_arc_side_1__',
        sketch_plane=part_obj.faces[29],
        sketch_up_edge=part_obj.edges[77],
        sketch_origin=(0.0, 0.5 * (P['hr'] - P['tw']), P['tf']),
        faces_for_partition=part_obj.faces.getSequenceFromMask(('[#20000000 ]',), ),
        side='side_1',
        P=P,
        sheet_size=314.98,
        grid_spacing=7.87,
        label='ENS-SP8 helper arc partition side 1'
    )

    # ------------------------------------
    # SIDE 1 : sweeps
    # ------------------------------------
    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0], part_obj.edges[1]),
        sweep_edge_id=73,
        label='ENS-SP8 sweep partition side 1 - 1'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
```

```
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=76,
        label='ENS-SP8 sweep partition side 1 - 2'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=75,
        label='ENS-SP8 sweep partition side 1 - 3'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=76,
        label='ENS-SP8 sweep partition side 1 - 4'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=76,
        label='ENS-SP8 sweep partition side 1 - 5'
    )

    # ----------------------------------
    # SIDE 2 : helper arc
    # ----------------------------------
    create_one_ens_helper_arc_partition(
        model_name=model_name,
        sketch_name='__ens_sp8_helper_arc_side_2__',
        sketch_plane=part_obj.faces[12],
        sketch_up_edge=part_obj.edges[54],
        sketch_origin=(0.0, 0.5 * (P['hr'] + P['tw']), P['tf']),
        faces_for_partition=part_obj.faces.getSequenceFromMask(('[#1000 ]',), ),
        side='side_2',
        P=P,
        sheet_size=314.98,
        grid_spacing=7.87,
        label='ENS-SP8 helper arc partition side 2'
    )

    # ----------------------------------
    # SIDE 2 : sweeps
    # ----------------------------------
    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[1], part_obj.edges[2]),
        sweep_edge_id=83,
        label='ENS-SP8 sweep partition side 2 - 1'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
```

```
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=84,
        label='ENS-SP8 sweep partition side 2 - 2'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=82,
        label='ENS-SP8 sweep partition side 2 - 3'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=86,
        label='ENS-SP8 sweep partition side 2 - 4'
    )

    apply_one_ens_sweep_partition(
        model_name=model_name,
        cell_mask='[#1 ]',
        edges_tuple=(part_obj.edges[0],),
        sweep_edge_id=87,
        label='ENS-SP8 sweep partition side 2 - 5'
    )


# ============================================================
# 13. ENS LOCAL CUTS AROUND SP4 / SP8
# ============================================================
def clamp_value(x, xmin, xmax):
    return max(xmin, min(x, xmax))


def get_ens_cut_window(P, S, target_sp_name):
    """
    Return a robust x-window around the requested S-point.

    target_sp_name:
        'SP4' or 'SP8'
    """
    W = {}

    if target_sp_name not in ('SP4', 'SP8'):
        raise ValueError('target_sp_name must be SP4 or SP8')

    # ------------------------------------------------------
    # target x-location
    # ------------------------------------------------------
    x0 = S[target_sp_name][0]

    # ------------------------------------------------------
    # base half-window
    # ------------------------------------------------------
    half_window = max(
        4.0 * P['tf'],
```

```
        2.5 * P['R'],
        0.22 * P['L']
    )

    # ----------------------------------------------------
    # available x-range of current HSS specimen
    # HSS keeps approximately:
    #   x_left_available  = 0
    #   x_right_available = L - a1/4
    # ----------------------------------------------------
    x_left_available = 0.0
    x_right_available = P['L'] - P['b'] / 4.0

    x_left = x0 - half_window
    x_right = x0 + half_window

    # ----------------------------------------------------
    # midpoint guard between SP4 and SP8
    # prevents ENS-SP4 from extending too far toward SP8
    # and vice versa
    # ----------------------------------------------------
    x_mid = 0.5 * (S['SP4'][0] + S['SP8'][0])

    if target_sp_name == 'SP4':
        x_right = min(x_right, x_mid)

    elif target_sp_name == 'SP8':
        x_left = max(x_left, x_mid)

    # ----------------------------------------------------
    # clamp to available HSS range
    # ----------------------------------------------------
    x_left = clamp_value(x_left, x_left_available, x_right_available)
    x_right = clamp_value(x_right, x_left_available, x_right_available)

    # ----------------------------------------------------
    # safety: ensure minimum width
    # ----------------------------------------------------
    min_width = max(2.0 * P['tf'], 2.0 * P['weld_leg'], 5.0)

    if (x_right - x_left) < min_width:
        xc = 0.5 * (x_left + x_right)
        x_left = clamp_value(
            xc - 0.5 * min_width,
            x_left_available,
            x_right_available
        )
        x_right = clamp_value(
            xc + 0.5 * min_width,
            x_left_available,
            x_right_available
        )

    W['target_sp_name'] = target_sp_name
    W['x0'] = x0
    W['half_window'] = half_window
    W['x_left'] = x_left
    W['x_right'] = x_right
    W['x_left_available'] = x_left_available
    W['x_right_available'] = x_right_available
```

```
    W['x_mid'] = x_mid

    return W


def cut_part_with_x_plane_keep_right(model_obj, part_obj, P, x_cut, z_mid, sketch_name):
    """
    Cut with YZ plane at x = x_cut, removing the LEFT side.
    Remaining part keeps the RIGHT side.
    """
    z_axis = get_or_create_datum_axis_z(part_obj)
    datum_plane = create_datum_plane_x_constant(part_obj, x_cut)

    cut_box_half = 5.0 * max(P['bf'], P['tf'] + P['hw'], P['hr'], P['L'])
    sheet_size_cut = 2.5 * cut_box_half

    cut_hss_with_plane(
        model_hss=model_obj,
        specimen_hss=part_obj,
        datum_plane=datum_plane,
        up_edge=z_axis,
        sketch_origin=(x_cut, 0.0, z_mid),
        cut_box_half=cut_box_half,
        sheet_size_cut=sheet_size_cut,
        plane_side=SIDE1,
        flip_extrude=OFF,
        sketch_name=sketch_name
    )


def cut_part_with_x_plane_keep_left(model_obj, part_obj, P, x_cut, z_mid, sketch_name):
    """
    Cut with YZ plane at x = x_cut, removing the RIGHT side.
    Remaining part keeps the LEFT side.
    """
    z_axis = get_or_create_datum_axis_z(part_obj)
    datum_plane = create_datum_plane_x_constant(part_obj, x_cut)

    cut_box_half = 5.0 * max(P['bf'], P['tf'] + P['hw'], P['hr'], P['L'])
    sheet_size_cut = 2.5 * cut_box_half

    cut_hss_with_plane(
        model_hss=model_obj,
        specimen_hss=part_obj,
        datum_plane=datum_plane,
        up_edge=z_axis,
        sketch_origin=(x_cut, 0.0, z_mid),
        cut_box_half=cut_box_half,
        sheet_size_cut=sheet_size_cut,
        plane_side=SIDE1,
        flip_extrude=ON,
        sketch_name=sketch_name
    )

def create_datum_plane_y_constant(part_obj, y_value):
    dp = part_obj.DatumPlaneByPrincipalPlane(
        principalPlane=XZPLANE,
        offset=y_value
    )
    return part_obj.datums[dp.id]
```

```
2616
2617
2618  def cut_part_with_y_plane_keep_positive(model_obj, part_obj, P, y_cut, z_mid, sketch_name):
2619      """
2620      Cut with XZ plane at y = y_cut.
2621      Keep y > y_cut, remove lower/negative-y side.
2622      """
2623      z_axis = get_or_create_datum_axis_z(part_obj)
2624      datum_plane = create_datum_plane_y_constant(part_obj, y_cut)
2625
2626      cut_box_half = 5.0 * max(P['bf'], P['tf'] + P['hw'], P['hr'], P['L'])
2627      sheet_size_cut = 2.5 * cut_box_half
2628
2629      cut_hss_with_plane(
2630          model_hss=model_obj,
2631          specimen_hss=part_obj,
2632          datum_plane=datum_plane,
2633          up_edge=z_axis,
2634          sketch_origin=(0.0, y_cut, z_mid),
2635          cut_box_half=cut_box_half,
2636          sheet_size_cut=sheet_size_cut,
2637          plane_side=SIDE1,
2638          flip_extrude=OFF,
2639          sketch_name=sketch_name
2640      )
2641
2642
2643  def cut_part_with_y_plane_keep_negative(model_obj, part_obj, P, y_cut, z_mid, sketch_name):
2644      """
2645      Cut with XZ plane at y = y_cut.
2646      Keep y < y_cut, remove upper/positive-y side.
2647      """
2648      z_axis = get_or_create_datum_axis_z(part_obj)
2649      datum_plane = create_datum_plane_y_constant(part_obj, y_cut)
2650
2651      cut_box_half = 5.0 * max(P['bf'], P['tf'] + P['hw'], P['hr'], P['L'])
2652      sheet_size_cut = 2.5 * cut_box_half
2653
2654      cut_hss_with_plane(
2655          model_hss=model_obj,
2656          specimen_hss=part_obj,
2657          datum_plane=datum_plane,
2658          up_edge=z_axis,
2659          sketch_origin=(0.0, y_cut, z_mid),
2660          cut_box_half=cut_box_half,
2661          sheet_size_cut=sheet_size_cut,
2662          plane_side=SIDE1,
2663          flip_extrude=ON,
2664          sketch_name=sketch_name
2665      )
2666
2667  def get_or_create_datum_axis_y(part_obj):
2668      if 'DATUM_AXIS_Y' in part_obj.features.keys():
2669          datum_id = part_obj.features['DATUM_AXIS_Y'].id
2670          return part_obj.datums[datum_id]
2671
2672      da = part_obj.DatumAxisByPrincipalAxis(principalAxis=YAXIS)
2673      part_obj.features.changeKey(fromName=da.name, toName='DATUM_AXIS_Y')
2674
2675      datum_id = part_obj.features['DATUM_AXIS_Y'].id
```

```
    return part_obj.datums[datum_id]


def create_datum_plane_z_constant(part_obj, z_value):
    dp = part_obj.DatumPlaneByPrincipalPlane(
        principalPlane=XYPLANE,
        offset=z_value
    )
    return part_obj.datums[dp.id]


def cut_part_with_z_plane_keep_below(model_obj, part_obj, P, z_cut, sketch_name):
    """
    Cut with XY plane at z = z_cut.
    Keep z < z_cut, remove upper web part.
    """
    y_axis = get_or_create_datum_axis_y(part_obj)
    datum_plane = create_datum_plane_z_constant(part_obj, z_cut)

    cut_box_half = 5.0 * max(P['bf'], P['tf'] + P['hw'], P['hr'], P['L'])
    sheet_size_cut = 2.5 * cut_box_half

    cut_hss_with_plane(
        model_hss=model_obj,
        specimen_hss=part_obj,
        datum_plane=datum_plane,
        up_edge=y_axis,
        sketch_origin=(0.0, 0.0, z_cut),
        cut_box_half=cut_box_half,
        sheet_size_cut=sheet_size_cut,
        plane_side=SIDE1,
        flip_extrude=ON,
        sketch_name=sketch_name
    )


def apply_extra_hss_run_cuts(P, H):
    """
    Extra HSS run cuts:
    - z = 3*tf
    - y = toe_sp4[1] + 1.8*tf
    - y = toe_sp8[1] - 1.8*tf
    """
    model_hss = mdb.models[HSS_MODEL_NAME]
    p = model_hss.parts['Specimen']

    z_cut = 3.0 * P['tf']
    y_cut_top = H['toe_sp4'][1] + 1.8 * P['tf']
    y_cut_bot = H['toe_sp8'][1] - 1.8 * P['tf']

    z_mid = 0.5 * (P['tf'] + z_cut)

    cut_part_with_z_plane_keep_below(
        model_obj=model_hss,
        part_obj=p,
        P=P,
        z_cut=z_cut,
        sketch_name='__hss_extra_cut_z_top__'
    )
    p.regenerate()
```

```
    cut_part_with_y_plane_keep_negative(
        model_obj=model_hss,
        part_obj=p,
        P=P,
        y_cut=y_cut_top,
        z_mid=z_mid,
        sketch_name='__hss_extra_cut_y_top__'
    )
    p.regenerate()

    cut_part_with_y_plane_keep_positive(
        model_obj=model_hss,
        part_obj=p,
        P=P,
        y_cut=y_cut_bot,
        z_mid=z_mid,
        sketch_name='__hss_extra_cut_y_bottom__'
    )
    p.regenerate()

    P['hss_z_top_cut'] = z_cut
    P['hss_y_top_cut'] = y_cut_top
    P['hss_y_bottom_cut'] = y_cut_bot

    log_print('-----------------------------------')
    log_print('Extra HSS run cuts applied')
    log_print('z_top_cut    = %.3f' % z_cut)
    log_print('y_top_cut    = %.3f' % y_cut_top)
    log_print('y_bottom_cut = %.3f' % y_cut_bot)
    log_print('-----------------------------------')

def build_ens_local_cut_model(model_name, P, S, target_sp_name):
    """
    Keep only a local x-window around SP4 or SP8.

    Rules:
    - skip unnecessary cuts if the cut plane is too close to an existing boundary
    - if two cuts are detected:
        * SP4 model keeps RIGHT cut only
        * SP8 model keeps LEFT cut only
    """
    model_obj = mdb.models[model_name]
    part_obj = model_obj.parts['Specimen']

    W = get_ens_cut_window(P, S, target_sp_name)
    z_mid = 0.5 * (P['tf'] + P['hw'])

    tol_x = max(1.0, 0.5 * P['tf'])

    do_left_cut = (W['x_left'] - W['x_left_available']) > tol_x
    do_right_cut = (W['x_right_available'] - W['x_right']) > tol_x

    # ----------------------------------------------------
    # if two cuts are detected, use fixed strategy by model
    # ----------------------------------------------------
    if do_left_cut and do_right_cut:
        log_print('Warning: two ENS cuts detected in %s' % model_name)

        if target_sp_name == 'SP4':
```

```
            # ENS-SP4 should keep only the RIGHT cut
            do_left_cut = False
            do_right_cut = True
            log_print('SP4 strategy: keeping RIGHT cut only.')

        elif target_sp_name == 'SP8':
            # ENS-SP8 should keep only the LEFT cut
            do_left_cut = True
            do_right_cut = False
            log_print('SP8 strategy: keeping LEFT cut only.')

        else:
            raise ValueError('Unknown target_sp_name: %s' % target_sp_name)

    # ----------------------------------------------------
    # store final decision inside W for Block 14
    # ----------------------------------------------------
    W['do_left_cut_final'] = do_left_cut
    W['do_right_cut_final'] = do_right_cut

    log_print('-----------------------------------')
    log_print('%s local ENS cut check:' % model_name)
    log_print('target   = %s' % target_sp_name)
    log_print('x0       = %.3f' % W['x0'])
    log_print('x_left   = %.3f' % W['x_left'])
    log_print('x_right  = %.3f' % W['x_right'])
    log_print('x_mid    = %.3f' % W['x_mid'])
    log_print('do_left_cut  = %s' % str(do_left_cut))
    log_print('do_right_cut = %s' % str(do_right_cut))
    log_print('-----------------------------------')

    # ----------------------------------------------------
    # cut 1: remove left side only if needed
    # ----------------------------------------------------
    if do_left_cut:
        cut_part_with_x_plane_keep_right(
            model_obj=model_obj,
            part_obj=part_obj,
            P=P,
            x_cut=W['x_left'],
            z_mid=z_mid,
            sketch_name='__%s_cut_left__' % target_sp_name
        )
        part_obj.regenerate()

    # ----------------------------------------------------
    # cut 2: remove right side only if needed
    # ----------------------------------------------------
    if do_right_cut:
        cut_part_with_x_plane_keep_left(
            model_obj=model_obj,
            part_obj=part_obj,
            P=P,
            x_cut=W['x_right'],
            z_mid=z_mid,
            sketch_name='__%s_cut_right__' % target_sp_name
        )
        part_obj.regenerate()

    log_print('%s local ENS cut done.' % model_name)
```

```
2856
2857     return W
2858
2859 # ============================================================
2860 # 13.B ENS FINAL TRIM WINDOW PREVIEW
2861 # ============================================================
2862 def get_final_ens_x_margin(P):
2863     """
2864     Safe X margin for final ENS trimming.
2865     Overlap between ENS-SP4 and ENS-SP8 is allowed.
2866     """
2867     return max(
2868         1.5 * P['tf'],
2869         4.0 * P['weld_leg'],
2870         0.5 * P['R'],
2871         40.0
2872     )
2873
2874
2875 def preview_final_ens_trim_windows(P, H):
2876     """
2877     Preview final ENS trimming windows around the weld toe region.
2878     Overlap between ENS-SP4 and ENS-SP8 is allowed.
2879     No cutting is performed here.
2880     """
2881
2882     margin = get_final_ens_x_margin(P)
2883
2884     W4 = {}
2885     W8 = {}
2886
2887     x4 = H['toe_sp4'][0]
2888     x8 = H['toe_sp8'][0]
2889
2890     # Available x-range of current ENS/HSS part
2891     x_min_available = 0.0
2892     x_max_available = P['L'] - P['b'] / 4.0
2893     tol_x = max(1.0, 0.1 * P['tf'])
2894
2895     # ENS-SP4
2896     W4['target_sp_name'] = 'SP4'
2897     W4['target_x'] = x4
2898     W4['margin'] = margin
2899     W4['x_left_raw'] = x4 - margin
2900     W4['x_right_raw'] = x4 + margin
2901     W4['x_left'] = clamp_value(W4['x_left_raw'], x_min_available, x_max_available)
2902     W4['x_right'] = clamp_value(W4['x_right_raw'], x_min_available, x_max_available)
2903
2904     # ENS-SP8
2905     W8['target_sp_name'] = 'SP8'
2906     W8['target_x'] = x8
2907     W8['margin'] = margin
2908     W8['x_left_raw'] = x8 - margin
2909     W8['x_right_raw'] = x8 + margin
2910     W8['x_left'] = clamp_value(W8['x_left_raw'], x_min_available, x_max_available)
2911     W8['x_right'] = clamp_value(W8['x_right_raw'], x_min_available, x_max_available)
2912
2913     # Decide which cuts are actually needed
2914     W4['do_left_cut_final'] = (W4['x_left'] - x_min_available) > tol_x
2915     W4['do_right_cut_final'] = (x_max_available - W4['x_right']) > tol_x
```

```
    W8['do_left_cut_final'] = (W8['x_left'] - x_min_available) > tol_x
    W8['do_right_cut_final'] = (x_max_available - W8['x_right']) > tol_x

    overlap = W4['x_right'] - W8['x_left']

    log_print('-----------------------------------', level=2)
    log_print('FINAL ENS TRIM WINDOW PREVIEW', level=2)
    log_print('available x_min = %.3f' % x_min_available, level=2)
    log_print('available x_max = %.3f' % x_max_available, level=2)
    log_print('margin = %.3f mm' % margin, level=2)

    log_print('ENS-SP4 target toe x = %.3f' % W4['target_x'], level=2)
    log_print('ENS-SP4 x_left raw   = %.3f' % W4['x_left_raw'], level=2)
    log_print('ENS-SP4 x_right raw  = %.3f' % W4['x_right_raw'], level=2)
    log_print('ENS-SP4 x_left       = %.3f' % W4['x_left'], level=2)
    log_print('ENS-SP4 x_right      = %.3f' % W4['x_right'], level=2)
    log_print('ENS-SP4 do_left_cut  = %s' % str(W4['do_left_cut_final']), level=2)
    log_print('ENS-SP4 do_right_cut = %s' % str(W4['do_right_cut_final']), level=2)

    log_print('ENS-SP8 target toe x = %.3f' % W8['target_x'], level=2)
    log_print('ENS-SP8 x_left raw   = %.3f' % W8['x_left_raw'], level=2)
    log_print('ENS-SP8 x_right raw  = %.3f' % W8['x_right_raw'], level=2)
    log_print('ENS-SP8 x_left       = %.3f' % W8['x_left'], level=2)
    log_print('ENS-SP8 x_right      = %.3f' % W8['x_right'], level=2)
    log_print('ENS-SP8 do_left_cut  = %s' % str(W8['do_left_cut_final']), level=2)
    log_print('ENS-SP8 do_right_cut = %s' % str(W8['do_right_cut_final']), level=2)

    if overlap > 0.0:
        log_print('Warning: ENS-SP4 and ENS-SP8 windows overlap by %.3f mm.' % overlap, level=2)
        log_print('This is allowed because the two ENS models are separate.', level=2)

    log_print('-----------------------------------', level=2)

    return W4, W8

# ============================================================
# 13.C FINAL ENS TRIMMING (APPLY CUTS AFTER PARTITIONS)
# ============================================================
def trim_final_ens_models_after_partitions(P, W4, W8):

    log_print('-----------------------------------', level=2)
    log_print('APPLYING FINAL ENS TRIMMING', level=2)
    log_print('-----------------------------------', level=2)

    z_mid = 0.5 * (P['tf'] + P['hw'])
    tol_x = max(1.0, 0.1 * P['tf'])

    # ----------------------------
    # ENS-SP4
    # ----------------------------
    model4 = mdb.models[ENS_SP4_MODEL_NAME]
    p4 = model4.parts['Specimen']

    if W4['do_left_cut_final']:
        cut_part_with_x_plane_keep_right(
            model_obj=model4,
            part_obj=p4,
            P=P,
            x_cut=W4['x_left'],
```

```
            z_mid=z_mid,
            sketch_name='__final_trim_sp4_left__'
        )
        p4.regenerate()

    if W4['do_right_cut_final']:
        cut_part_with_x_plane_keep_left(
            model_obj=model4,
            part_obj=p4,
            P=P,
            x_cut=W4['x_right'],
            z_mid=z_mid,
            sketch_name='__final_trim_sp4_right__'
        )
        p4.regenerate()

    log_print(
        'ENS-SP4 trimmed: x_left = %.3f, x_right = %.3f'
        % (W4['x_left'], W4['x_right']), level=2
    )

    # ------------------------------
    # ENS-SP8
    # ------------------------------
    model8 = mdb.models[ENS_SP8_MODEL_NAME]
    p8 = model8.parts['Specimen']

    if W8['do_left_cut_final']:
        cut_part_with_x_plane_keep_right(
            model_obj=model8,
            part_obj=p8,
            P=P,
            x_cut=W8['x_left'],
            z_mid=z_mid,
            sketch_name='__final_trim_sp8_left__'
        )
        p8.regenerate()

    if W8['do_right_cut_final']:
        cut_part_with_x_plane_keep_left(
            model_obj=model8,
            part_obj=p8,
            P=P,
            x_cut=W8['x_right'],
            z_mid=z_mid,
            sketch_name='__final_trim_sp8_right__'
        )
        p8.regenerate()

    log_print(
        'ENS-SP8 trimmed: x_left = %.3f, x_right = %.3f'
        % (W8['x_left'], W8['x_right']), level=2
    )

    log_print('FINAL ENS TRIMMING DONE.', level=2)

def trim_final_ens_models_in_y(P, Y4, Y8):

    log_print('-----------------------------------', level=2)
    log_print('APPLYING FINAL ENS Y-TRIMMING', level=2)
```

```
    log_print('-----------------------------------', level=2)

    z_mid = 0.5 * (P['tf'] + P['hw'])

    # ENS-SP4
    model4 = mdb.models[ENS_SP4_MODEL_NAME]
    p4 = model4.parts['Specimen']

    cut_part_with_y_plane_keep_positive(
        model_obj=model4,
        part_obj=p4,
        P=P,
        y_cut=Y4['y_min'],
        z_mid=z_mid,
        sketch_name='__final_trim_sp4_y_min__'
    )
    p4.regenerate()

    cut_part_with_y_plane_keep_negative(
        model_obj=model4,
        part_obj=p4,
        P=P,
        y_cut=Y4['y_max'],
        z_mid=z_mid,
        sketch_name='__final_trim_sp4_y_max__'
    )
    p4.regenerate()

    log_print(
        'ENS-SP4 Y-trimmed: y_min = %.3f, y_max = %.3f'
        % (Y4['y_min'], Y4['y_max']), level=2
    )

    # ENS-SP8
    model8 = mdb.models[ENS_SP8_MODEL_NAME]
    p8 = model8.parts['Specimen']

    cut_part_with_y_plane_keep_positive(
        model_obj=model8,
        part_obj=p8,
        P=P,
        y_cut=Y8['y_min'],
        z_mid=z_mid,
        sketch_name='__final_trim_sp8_y_min__'
    )
    p8.regenerate()

    cut_part_with_y_plane_keep_negative(
        model_obj=model8,
        part_obj=p8,
        P=P,
        y_cut=Y8['y_max'],
        z_mid=z_mid,
        sketch_name='__final_trim_sp8_y_max__'
    )
    p8.regenerate()

    log_print(
        'ENS-SP8 Y-trimmed: y_min = %.3f, y_max = %.3f'
        % (Y8['y_min'], Y8['y_max']), level=2
```

```
    )

    log_print('FINAL ENS Y-TRIMMING DONE.', level=2)


# ========================================================
# 13D. FINAL ENS Y-TRIM WINDOW PREVIEW
# ========================================================
def preview_final_ens_y_trim_windows(P, H):
    """
    Preview final ENS trimming windows in Y direction.
    No cutting is performed here.
    """

    if 'hss_y_bottom_cut' in P.keys() and 'hss_y_top_cut' in P.keys():
        y_min_available = P['hss_y_bottom_cut']
        y_max_available = P['hss_y_top_cut']
    else:
        y_min_available = -0.5 * P['bf']
        y_max_available =  0.5 * P['bf']

    y4 = H['toe_sp4'][1]
    y8 = H['toe_sp8'][1]

    Y4 = {}
    Y8 = {}
    y_mid = 0.5 * (y4 + y8)

    # ENS-SP4
    Y4['target_sp_name'] = 'SP4'
    Y4['y_min_raw'] = y_mid - 10.0
    Y4['y_max_raw'] = y4 + 0.5 * (y_max_available - y4)
    Y4['y_min'] = clamp_value(Y4['y_min_raw'], y_min_available, y_max_available)
    Y4['y_max'] = clamp_value(Y4['y_max_raw'], y_min_available, y_max_available)

    # ENS-SP8
    Y8['target_sp_name'] = 'SP8'
    Y8['y_min_raw'] = y8 + 0.5 * (y_min_available - y8)
    Y8['y_max_raw'] = y_mid + 10.0
    Y8['y_min'] = clamp_value(Y8['y_min_raw'], y_min_available, y_max_available)
    Y8['y_max'] = clamp_value(Y8['y_max_raw'], y_min_available, y_max_available)

    log_print('-----------------------------------', level=2)
    log_print('FINAL ENS Y-TRIM WINDOW PREVIEW', level=2)
    log_print('available y_min = %.3f' % y_min_available, level=2)
    log_print('available y_max = %.3f' % y_max_available, level=2)

    log_print('toe_sp4 y = %.3f' % y4, level=2)
    log_print('toe_sp8 y = %.3f' % y8, level=2)

    log_print('ENS-SP4 y_min raw = %.3f' % Y4['y_min_raw'], level=2)
    log_print('ENS-SP4 y_max raw = %.3f' % Y4['y_max_raw'], level=2)
    log_print('ENS-SP4 y_min     = %.3f' % Y4['y_min'], level=2)
    log_print('ENS-SP4 y_max     = %.3f' % Y4['y_max'], level=2)

    log_print('ENS-SP8 y_min raw = %.3f' % Y8['y_min_raw'], level=2)
    log_print('ENS-SP8 y_max raw = %.3f' % Y8['y_max_raw'], level=2)
    log_print('ENS-SP8 y_min     = %.3f' % Y8['y_min'], level=2)
    log_print('ENS-SP8 y_max     = %.3f' % Y8['y_max'], level=2)
    log_print('-----------------------------------', level=2)
```

```
3156
3157        return Y4, Y8
3158
3159    # ============================================================
3160    # 13E. ENS CENTERLINE POINTS PREVIEW
3161    # ============================================================
3162    def compute_ens_centerline_points(P, S, W4, W8):
3163        """
3164        Compute preview points 1-4 for ENS centerline paths.
3165        No partition, no mesh, no seeding.
3166        """
3167
3168        sqrt2_minus_1 = math.sqrt(2.0) - 1.0
3169        theta = P['theta_rad']
3170
3171        offset = P['weld_leg'] + sqrt2_minus_1
3172        n = 0.5 * P['tw'] + offset
3173
3174        zc = P['tf'] + P['notch_radius']
3175
3176        C4 = {}
3177        C8 = {}
3178
3179        # ----------------------------
3180        # ENS-SP4
3181        # ----------------------------
3182        x_left_4 = W4['x_left']
3183        x_right_4 = W4['x_right']
3184
3185        x_sp4 = S['SP4'][0]
3186        y_sp4 = S['SP4'][1]
3187
3188        C4['p1'] = (x_left_4,  0.5 * P['hr'] + n, zc)
3189        C4['p2'] = (x_left_4,  0.5 * P['hr'] - n, zc)
3190
3191        y3 = y_sp4 - (x_right_4 - x_sp4) * math.tan(theta) + offset / math.cos(theta)
3192        y4 = y3 - (2.0 * offset + P['tw']) / math.cos(theta)
3193
3194        C4['p3'] = (x_right_4, y3, zc)
3195        C4['p4'] = (x_right_4, y4, zc)
3196
3197        # ----------------------------
3198        # ENS-SP8
3199        # ----------------------------
3200        x_left_8 = W8['x_left']
3201        x_right_8 = W8['x_right']
3202
3203        x_sp8 = S['SP8'][0]
3204        y_sp8 = S['SP8'][1]
3205
3206        C8['p1'] = (x_right_8, -0.5 * P['hr'] - n, zc)
3207        C8['p2'] = (x_right_8, -0.5 * P['hr'] + n, zc)
3208
3209        y3 = y_sp8 + (x_sp8 - x_left_8) * math.tan(theta) - offset / math.cos(theta)
3210        y4 = y3 + (2.0 * offset + P['tw']) / math.cos(theta)
3211
3212        C8['p3'] = (x_left_8, y3, zc)
3213        C8['p4'] = (x_left_8, y4, zc)
3214
3215        return C4, C8
```

```
3216
3217
3218 def create_datum_points_for_centerline_preview(model_name, C, label):
3219     """
3220     Create datum points only for visual checking.
3221     """
3222
3223     p = mdb.models[model_name].parts['Specimen']
3224
3225     for key in ['p1', 'p2', 'p3', 'p4']:
3226         p.DatumPointByCoordinate(coords=C[key])
3227
3228     try:
3229         p.regenerate()
3230     except:
3231         pass
3232
3233     log_print('-----------------------------------', Level =2)
3234     log_print('%s ENS centerline preview points:' % label)
3235     for key in ['p1', 'p2', 'p3', 'p4']:
3236         log_print('%s = %s' % (key, str(C[key])))
3237     log_print('Datum points created only for preview.')
3238     log_print('-----------------------------------')
3239
3240
3241 def preview_ens_centerline_points(P, S, W4, W8):
3242     """
3243     Preview centerline points for ENS-SP4 and ENS-SP8.
3244     """
3245
3246     C4, C8 = compute_ens_centerline_points(P, S, W4, W8)
3247
3248     create_datum_points_for_centerline_preview(
3249         model_name=ENS_SP4_MODEL_NAME,
3250         C=C4,
3251         label='ENS-SP4'
3252     )
3253
3254     create_datum_points_for_centerline_preview(
3255         model_name=ENS_SP8_MODEL_NAME,
3256         C=C8,
3257         label='ENS-SP8'
3258     )
3259
3260     return C4, C8
3261
3262
3263
3264 def densify_path_points(path_points, n_sub=2):
3265
3266     new_pts = []
3267
3268     for i in range(len(path_points) - 1):
3269         A = path_points[i]
3270         B = path_points[i + 1]
3271
3272         new_pts.append(A)
3273
3274         for j in range(1, n_sub + 1):
3275             t = float(j) / (n_sub + 1)
```

```
3276
3277                new_pts.append((
3278                    A[0] + t * (B[0] - A[0]),
3279                    A[1] + t * (B[1] - A[1]),
3280                    A[2] + t * (B[2] - A[2])
3281                ))
3282
3283        new_pts.append(path_points[-1])
3284
3285        return new_pts
3286
3287
3288    # ============================================================
3289    # 13F. ENS CENTERLINE PATH PREVIEW WITH FILLET
3290    # ============================================================
3291    def _unit_2d(v):
3292        L = math.sqrt(v[0]*v[0] + v[1]*v[1])
3293        if L < 1.0e-12:
3294            raise ValueError('Zero length vector.')
3295        return (v[0]/L, v[1]/L)
3296
3297
3298    def _line_intersection_2d(p1, u1, p2, u2):
3299        x1, y1 = p1
3300        x2, y2 = p2
3301        ux1, uy1 = u1
3302        ux2, uy2 = u2
3303
3304        den = ux1 * uy2 - uy1 * ux2
3305        if abs(den) < 1.0e-12:
3306            raise ValueError('Lines are nearly parallel.')
3307
3308        dx = x2 - x1
3309        dy = y2 - y1
3310
3311        t = (dx * uy2 - dy * ux2) / den
3312
3313        return (x1 + t * ux1, y1 + t * uy1)
3314
3315
3316    def _build_fillet_path_2d(p_left, p_right, theta, radius, mode, n_arc=12):
3317        """
3318        Build centerline path points:
3319        - mode='SP4': left line horizontal, right line inclined downward
3320        - mode='SP8': left line inclined upward, right line horizontal
3321        """
3322
3323        c = math.cos(theta)
3324        s = math.sin(theta)
3325
3326        if mode == 'SP4':
3327            u_left_line = (1.0, 0.0)
3328            u_right_line = (c, -s)
3329
3330        elif mode == 'SP8':
3331            u_left_line = (c, -s)
3332            u_right_line = (1.0, 0.0)
3333
3334        else:
3335            raise ValueError('mode must be SP4 or SP8')
```

```
    corner = _line_intersection_2d(
        (p_left[0], p_left[1]), u_left_line,
        (p_right[0], p_right[1]), u_right_line
    )

    # directions from corner toward the real end points
    v1 = _unit_2d((p_left[0] - corner[0], p_left[1] - corner[1]))
    v2 = _unit_2d((p_right[0] - corner[0], p_right[1] - corner[1]))

    dot12 = v1[0]*v2[0] + v1[1]*v2[1]
    dot12 = max(-1.0, min(1.0, dot12))

    alpha = math.acos(dot12)

    if alpha < 1.0e-6:
        raise ValueError('Fillet angle is too small.')

    tangent_dist = radius / math.tan(0.5 * alpha)

    t1 = (
        corner[0] + v1[0] * tangent_dist,
        corner[1] + v1[1] * tangent_dist
    )

    t2 = (
        corner[0] + v2[0] * tangent_dist,
        corner[1] + v2[1] * tangent_dist
    )

    bis = _unit_2d((v1[0] + v2[0], v1[1] + v2[1]))
    center_dist = radius / math.sin(0.5 * alpha)

    center = (
        corner[0] + bis[0] * center_dist,
        corner[1] + bis[1] * center_dist
    )

    a1 = math.atan2(t1[1] - center[1], t1[0] - center[0])
    a2 = math.atan2(t2[1] - center[1], t2[0] - center[0])

    da = a2 - a1
    while da > math.pi:
        da -= 2.0 * math.pi
    while da < -math.pi:
        da += 2.0 * math.pi

    pts = []
    pts.append(p_left)

    for i in range(1, n_arc + 1):
        a = a1 + da * float(i) / float(n_arc + 1)
        pts.append((
            center[0] + radius * math.cos(a),
            center[1] + radius * math.sin(a),
            p_left[2]
        ))

    pts.append(p_right)
```

```
    pts_dense = densify_path_points(pts, n_sub=2)

    return {
        'corner': (corner[0], corner[1], p_left[2]),
        't1': (t1[0], t1[1], p_left[2]),
        't2': (t2[0], t2[1], p_left[2]),
        'center': (center[0], center[1], p_left[2]),
        'path_points': pts_dense
    }


def create_datum_points_for_centerline_path(model_name, path_data, label):
    """
    Create datum points along the calculated centerline path.
    Preview only: no mesh, no partition, no seeding.
    """

    p = mdb.models[model_name].parts['Specimen']

    for pt in path_data['path_points']:
        p.DatumPointByCoordinate(coords=pt)

    p.DatumPointByCoordinate(coords=path_data['corner'])
    p.DatumPointByCoordinate(coords=path_data['center'])
    p.DatumPointByCoordinate(coords=path_data['t1'])
    p.DatumPointByCoordinate(coords=path_data['t2'])

    try:
        p.regenerate()
    except:
        pass

    log_print('%s centerline path preview datum points created.' % label, level=2)

# ============================================================
# 13G. ENS EDGE SETS FROM LOCAL PARTITION CELLS
# ============================================================
def get_edges_from_cells(part_obj, cell_seq):
    edge_indices = {}
    edge_list = []

    for c in cell_seq:
        try:
            e_ids = c.getEdges()
            for eid in e_ids:
                if eid not in edge_indices:
                    edge_indices[eid] = True
                    edge_list.append(part_obj.edges[eid])
        except:
            pass

    return edge_list


def collect_cells_near_toe_region(model_name, P, W, target_side):

    p = mdb.models[model_name].parts['Specimen']

    x_min = W['x_left'] - 40.0
    x_max = W['x_right'] + 40.0
```

```
3456
3457      z_min = P['tf'] - 5.0
3458      z_max = P['tf'] + P['hw'] + 5.0
3459
3460      if target_side == 'SP4':
3461          y_center = 0.5 * P['hr']
3462      elif target_side == 'SP8':
3463          y_center = -0.5 * P['hr']
3464      else:
3465          raise ValueError('target_side must be SP4 or SP8')
3466
3467      y_margin_fine = P['tw'] + 2.0 * P['weld_leg'] + 5.0
3468      y_margin_transition = P['tw'] + 2.0 * P['weld_leg'] + 15.0
3469
3470      cells_fine = []
3471      cells_transition = []
3472
3473      for c in p.cells:
3474          try:
3475              x, y, z = c.pointOn[0]
3476          except:
3477              continue
3478
3479          if not (x_min <= x <= x_max):
3480              continue
3481
3482          if not (z_min <= z <= z_max):
3483              continue
3484
3485          dy = abs(y - y_center)
3486
3487          if dy <= y_margin_transition:
3488              cells_transition.append(c)
3489
3490          if dy <= y_margin_fine:
3491              cells_fine.append(c)
3492
3493      log_print('%s ALL cells = %d' % (model_name, len(p.cells)), level=2)
3494      log_print('%s %s cells_fine = %d' % (model_name, target_side, len(cells_fine)), level=2)
3495      log_print('%s %s cells_transition = %d' % (model_name, target_side, len(cells_transition)), level=2)
3496
3497      return cells_fine, cells_transition
3498
3499
3500  def create_ens_cell_based_edge_sets_for_model(model_name, P, W, target_side):
3501      p = mdb.models[model_name].parts['Specimen']
3502
3503      cells_fine, cells_transition_all = collect_cells_near_toe_region(
3504          model_name,
3505          P,
3506          W,
3507          target_side
3508      )
3509
3510      edges_fine = get_edges_from_cells(p, cells_fine)
3511      edges_transition_all = get_edges_from_cells(p, cells_transition_all)
3512
3513      fine_indices = {}
3514      for e in edges_fine:
3515          fine_indices[e.index] = True
```

```
3516
3517      edges_transition = edges_transition_all
3518
3519      create_edge_set_from_list(
3520          p,
3521          edges_transition,
3522          'SET_ENS_CENTERLINE_TRANSITION'
3523      )
3524
3525      create_edge_set_from_list(
3526          p,
3527          edges_fine,
3528          'SET_ENS_CENTERLINE_FINE'
3529      )
3530
3531      log_print('%s cell-based edge sets created.' % model_name, level=2)
3532      log_print('%s fine edges = %d' % (model_name, len(edges_fine)), level=2)
3533      log_print('%s transition edges = %d' % (model_name, len(edges_transition)), level=2)
3534
3535      return {
3536          'fine': len(edges_fine),
3537          'transition': len(edges_transition)
3538      }
3539
3540  # ============================================================
3541  # 14. ENS BC CUT FACE / SURFACE
3542  # ============================================================
3543
3544  def get_faces_near_x(part_obj, x_target, P, tol):
3545      """
3546      Return FaceArray of faces near x = x_target using bounding box.
3547      """
3548      y_margin = 2.0 * P['bf']
3549      z_margin = 2.0 * (P['tf'] + P['hw'])
3550
3551      faces_found = part_obj.faces.getByBoundingBox(
3552          xMin=x_target - tol,
3553          yMin=-y_margin,
3554          zMin=-z_margin,
3555          xMax=x_target + tol,
3556          yMax= y_margin,
3557          zMax= z_margin
3558      )
3559
3560      return faces_found
3561
3562
3563  def create_bc_surface_from_actual_cut(model_name, W, P):
3564      """
3565      Create BC surface ONLY on the face generated by the actual ENS cut.
3566      Uses the final cut decision stored in W by Block 13.
3567      """
3568      model_obj = mdb.models[model_name]
3569      part_obj  = model_obj.parts['Specimen']
3570
3571      # -------------------------------------
3572      # tolerance for locating cut face
3573      # -------------------------------------
3574      tol_x = max(0.2, 0.1 * P['tf'])
3575
```

```
    # -------------------------------------
    # use FINAL cut decision from Block 13
    # -------------------------------------
    do_left_cut = W.get('do_left_cut_final', False)
    do_right_cut = W.get('do_right_cut_final', False)

    # -------------------------------------
    # choose BC location from final decision
    # -------------------------------------
    if do_right_cut and not do_left_cut:
        x_bc = W['x_right']

    elif do_left_cut and not do_right_cut:
        x_bc = W['x_left']

    elif do_left_cut and do_right_cut:
        raise ValueError(
            'Both final left and right ENS cuts are active in %s. '
            'This should not happen after Block 13 filtering.' % model_name
        )

    else:
        raise ValueError(
            'No final ENS cut detected for BC in model %s' % model_name
        )

    # -------------------------------------
    # find faces near the chosen cut plane
    # -------------------------------------
    faces_bc = get_faces_near_x(part_obj, x_bc, P, tol_x)

    if len(faces_bc) == 0:
        raise ValueError(
            'No face found near x = %.3f in model %s' % (x_bc, model_name)
        )

    # -------------------------------------
    # remove old set/surface if already exist
    # -------------------------------------
    if 'SET_ENS_CUT' in part_obj.sets.keys():
        del part_obj.sets['SET_ENS_CUT']

    if 'SURF_ENS_CUT' in part_obj.surfaces.keys():
        del part_obj.surfaces['SURF_ENS_CUT']

    # -------------------------------------
    # create set
    # -------------------------------------
    part_obj.Set(
        name='SET_ENS_CUT',
        faces=faces_bc
    )

    # -------------------------------------
    # create surface
    # -------------------------------------
    part_obj.Surface(
        name='SURF_ENS_CUT',
        side1Faces=faces_bc
    )
```

```
3636
3637        # ------------------------------------
3638        # debug print
3639        # ------------------------------------
3640        log_print('------------------------------------')
3641        log_print('%s BC CUT SURFACE:' % model_name)
3642        log_print('x_bc = %.3f' % x_bc)
3643        log_print('faces found = %d' % len(faces_bc))
3644        log_print('SET  -> SET_ENS_CUT')
3645        log_print('SURF -> SURF_ENS_CUT')
3646        log_print('------------------------------------')
3647
3648        return {
3649            'x_bc': x_bc,
3650            'n_faces': len(faces_bc)
3651        }
3652
3653    # ============================================================
3654    # 15A. GLOBAL ASSEMBLY + RP + COUPLING + BC + LOAD
3655    # ============================================================
3656
3657    def clear_assembly_instances_and_features(assembly_obj):
3658        """
3659        Suppress old assembly features if they exist.
3660        Useful when rerunning the script many times.
3661        """
3662        for feat_name in list(assembly_obj.features.keys()):
3663            try:
3664                assembly_obj.features[feat_name].suppress()
3665            except:
3666                pass
3667
3668
3669    def create_global_assembly_instance(model):
3670        """
3671        Create fresh assembly instance for the global specimen.
3672        """
3673        a = model.rootAssembly
3674        a.DatumCsysByDefault(CARTESIAN)
3675
3676        clear_assembly_instances_and_features(a)
3677
3678        part_obj = model.parts['Specimen']
3679        a.Instance(
3680            name='Specimen-1',
3681            part=part_obj,
3682            dependent=ON
3683        )
3684
3685        a.regenerate()
3686        return a, a.instances['Specimen-1']
3687
3688    def create_global_reference_points(model, a, P):
3689
3690        rp_fixed_feature = a.ReferencePoint(
3691            point=(P['x_global_left'], 0.0, 0.5 * P['tf'])
3692        )
3693
3694        rp_load_feature  = a.ReferencePoint(
3695            point=(P['x_global_right'], 0.0, 0.5 * P['tf'])
```

```
    )

    if 'RP_Fixed' in a.sets.keys():
        del a.sets['RP_Fixed']
    if 'RP_Load' in a.sets.keys():
        del a.sets['RP_Load']

    a.Set(
        name='RP_Fixed',
        referencePoints=(a.referencePoints[rp_fixed_feature.id], )
    )

    a.Set(
        name='RP_Load',
        referencePoints=(a.referencePoints[rp_load_feature.id], )
    )


def get_global_end_faces_flange_only(inst, P, tol=None):
    """
    Return ONLY flange faces at x = -L and x = +L
    (exclude web by restricting z-range)
    """
    if tol is None:
        tol = max(0.2, 0.1 * P['tf'])

    y_margin = 2.0 * P['bf']

    # restrict to flange thickness only
    z_min = -tol
    z_max = P['tf'] + tol

    # LEFT side (fixed)
    faces_fixed = inst.faces.getByBoundingBox(
        xMin=P['x_global_left'] - tol,
        xMax=P['x_global_left'] + tol,
        yMin=-y_margin,
        yMax= y_margin,
        zMin=z_min,
        zMax=z_max
    )

    # RIGHT side (load)
    faces_load = inst.faces.getByBoundingBox(
        xMin=P['x_global_right'] - tol,
        xMax=P['x_global_right'] + tol,
        yMin=-y_margin,
        yMax= y_margin,
        zMin=z_min,
        zMax=z_max
    )

    return faces_fixed, faces_load


def get_global_end_faces_flange_and_web(inst, P, tol=None):
    """
    Return all specimen end faces at x = -L and x = +L
    including both flange and web faces.
    """
```

```
    if tol is None:
        tol = max(0.2, 0.1 * P['tf'])

    y_margin = 2.0 * P['bf']
    z_margin = 2.0 * (P['tf'] + P['hw'])

    faces_fixed = inst.faces.getByBoundingBox(
        xMin=P['x_global_left'] - tol,
        xMax=P['x_global_left'] + tol,
        yMin=-y_margin,
        yMax= y_margin,
        zMin=-tol,
        zMax=z_margin
    )

    faces_load = inst.faces.getByBoundingBox(
        xMin=P['x_global_right'] - tol,
        xMax=P['x_global_right'] + tol,
        yMin=-y_margin,
        yMax= y_margin,
        zMin=-tol,
        zMax=z_margin
    )

    return faces_fixed, faces_load


def create_global_top_web_surface(a, inst, P, tol=None):
    """
    Create assembly-level surface on the TOP web face(s) at z = tf + hw.
    This is used only for optional Z-restraint study.
    """
    if tol is None:
        tol = max(0.2, 0.1 * P['tf'])

    x_margin = max(abs(P['x_global_left']), abs(P['x_global_right'])) + 2.0 * P['L']
    y_margin = 2.0 * P['bf']
    z_top = P['tf'] + P['hw']

    faces_web_top = inst.faces.getByBoundingBox(
        xMin=-x_margin,
        xMax= x_margin,
        yMin=-y_margin,
        yMax= y_margin,
        zMin=z_top - tol,
        zMax=z_top + tol
    )

    if len(faces_web_top) == 0:
        raise ValueError('No top web faces found near z = tf + hw')

    if 'WebTop_surface' in a.surfaces.keys():
        del a.surfaces['WebTop_surface']

    a.Surface(
        name='WebTop_surface',
        side1Faces=faces_web_top
    )

    log_print('WebTop_surface faces = %d' % len(faces_web_top))
```

```
3816
3817
3818 def create_global_end_surfaces(model, a, inst, P):
3819     """
3820     Create assembly-level surfaces on x = -L and x = +L end faces.
3821     """
3822     if P.get('couple_web_to_rp', False):
3823         faces_fixed, faces_load = get_global_end_faces_flange_and_web(inst, P)
3824     else:
3825         faces_fixed, faces_load = get_global_end_faces_flange_only(inst, P)
3826
3827     if len(faces_fixed) == 0:
3828         raise ValueError('No global fixed-end faces found near x = -L')
3829
3830     if len(faces_load) == 0:
3831         raise ValueError('No global load-end faces found near x = +L')
3832
3833     if 'Fixed_surface' in a.surfaces.keys():
3834         del a.surfaces['Fixed_surface']
3835     if 'Load_surface' in a.surfaces.keys():
3836         del a.surfaces['Load_surface']
3837
3838     a.Surface(
3839         name='Fixed_surface',
3840         side1Faces=faces_fixed
3841     )
3842
3843     a.Surface(
3844         name='Load_surface',
3845         side1Faces=faces_load
3846     )
3847
3848     log_print('-----------------------------------')
3849     log_print('Global end surfaces created:')
3850     log_print('Fixed_surface faces = %d' % len(faces_fixed))
3851     log_print('Load_surface  faces = %d' % len(faces_load))
3852     log_print('-----------------------------------')
3853
3854
3855 def create_global_couplings(model, P):
3856     """
3857     Couple only the end surfaces to their reference points.
3858     Top web restraint, when requested, is applied directly on WebTop_surface.
3859     """
3860     safe_delete_constraint(model, 'Constraint-Fixed')
3861     safe_delete_constraint(model, 'Constraint-Load')
3862     safe_delete_constraint(model, 'Constraint-WebTopZ')
3863
3864     a = model.rootAssembly
3865
3866     model.Coupling(
3867         name='Constraint-Fixed',
3868         controlPoint=a.sets['RP_Fixed'],
3869         surface=a.surfaces['Fixed_surface'],
3870         influenceRadius=WHOLE_SURFACE,
3871         couplingType=KINEMATIC,
3872         localCsys=None,
3873         u1=ON, u2=ON, u3=ON,
3874         ur1=ON, ur2=ON, ur3=ON,
3875
```

```
    )

    model.Coupling(
        name='Constraint-Load',
        controlPoint=a.sets['RP_Load'],
        surface=a.surfaces['Load_surface'],
        influenceRadius=WHOLE_SURFACE,
        couplingType=KINEMATIC,
        localCsys=None,
        u1=ON, u2=ON, u3=ON,
        ur1=ON, ur2=ON, ur3=ON,

    )


def apply_global_boundary_conditions(model, P):
    """
    Apply global BCs:
    - fixed RP fully fixed
    - load RP guided only in x direction
    - optional top-web surface restrained directly only in Z
    """
    a = model.rootAssembly

    if 'BC-Fixed' in model.boundaryConditions.keys():
        del model.boundaryConditions['BC-Fixed']
    if 'BC-LoadGuide' in model.boundaryConditions.keys():
        del model.boundaryConditions['BC-LoadGuide']
    if 'BC-WebTopZ' in model.boundaryConditions.keys():
        del model.boundaryConditions['BC-WebTopZ']

    if 'SET_WebTopBC' in a.sets.keys():
        del a.sets['SET_WebTopBC']

    model.DisplacementBC(
        name='BC-Fixed',
        createStepName='Initial',
        region=a.sets['RP_Fixed'],
        u1=SET, u2=SET, u3=SET,
        ur1=SET, ur2=SET, ur3=SET,
        amplitude=UNSET,
        distributionType=UNIFORM,
        fieldName='',
        localCsys=None
    )

    model.DisplacementBC(
        name='BC-LoadGuide',
        createStepName='Initial',
        region=a.sets['RP_Load'],
        u1=UNSET, u2=SET, u3=SET,
        ur1=SET, ur2=SET, ur3=SET,
        amplitude=UNSET,
        distributionType=UNIFORM,
        fieldName='',
        localCsys=None
    )

    if P.get('restrain_web_top_z', False):
        inst = a.instances['Specimen-1']
```

```
        tol = max(0.2, 0.1 * P['tf'])

        faces_web_top = inst.faces.getByBoundingBox(
            xMin=(P['x_global_left'] - 2.0 * P['L']),
            xMax=(P['x_global_right'] + 2.0 * P['L']),
            yMin=-2.0 * P['bf'],
            yMax= 2.0 * P['bf'],
            zMin=P['tf'] + P['hw'] - tol,
            zMax=P['tf'] + P['hw'] + tol
        )

        if len(faces_web_top) == 0:
            raise ValueError('No top web faces found for BC-WebTopZ')

        a.Set(
            name='SET_WebTopBC',
            faces=faces_web_top
        )

        model.DisplacementBC(
            name='BC-WebTopZ',
            createStepName='Initial',
            region=a.sets['SET_WebTopBC'],
            u1=UNSET, u2=UNSET, u3=SET,
            ur1=UNSET, ur2=UNSET, ur3=UNSET,
            amplitude=UNSET,
            distributionType=UNIFORM,
            fieldName='',
            localCsys=None
        )


def apply_global_load(model, P):
    """
    Apply tensile concentrated force at the load RP.
    """
    a = model.rootAssembly

    if 'Tension-Load' in model.loads.keys():
        del model.loads['Tension-Load']

    model.ConcentratedForce(
        name='Tension-Load',
        createStepName='Step-1',
        region=a.sets['RP_Load'],
        cf1=P['F_load'],
        distributionType=UNIFORM,
        field='',
        localCsys=None
    )


def build_global_bc_and_load(model, P):
    """
    Full global assembly + RP + surfaces + couplings + BC + load.
    """
    a, inst = create_global_assembly_instance(model)
    create_global_reference_points(model, a, P)
    create_global_end_surfaces(model, a, inst, P)

```

```
    if P.get('restrain_web_top_z', False):
        create_global_top_web_surface(a, inst, P)

    create_global_couplings(model, P)
    apply_global_boundary_conditions(model, P)
    apply_global_load(model, P)

    a.regenerate()

    log_print('==================================')
    log_print('Global BC + Load setup done.')
    log_print('RP_Fixed, RP_Load created')
    if P.get('restrain_web_top_z', False):
        log_print('WebTop_surface created')
        log_print('BC-WebTopZ applied directly on WebTop_surface')
    log_print('Fixed_surface, Load_surface created')
    log_print('Constraint-Fixed, Constraint-Load created')
    log_print('BC-Fixed, BC-LoadGuide, Tension-Load created')
    log_print('==================================')


# ========================================================
# 15B. HSS SUBMODEL BC FROM GLOBAL (CUT SURFACES WITHOUT WELD)
# ========================================================

def clear_old_hss_analysis_objects(model_hss):
    """
    Suppress old loads, BCs, and constraints in HSS model
    so only the new HSS SubmodelBC remains.
    """
    for name in list(model_hss.loads.keys()):
        try:
            model_hss.loads[name].suppress()
        except:
            pass

    for name in list(model_hss.boundaryConditions.keys()):
        try:
            model_hss.boundaryConditions[name].suppress()
        except:
            pass

    safe_suppress_all_constraints(model_hss)

    log_print('Old HSS loads / BCs / constraints suppressed.')


def _safe_face_area(face_obj):
    try:
        return face_obj.getSize(printResults=False)
    except:
        try:
            return face_obj.getSize()
        except:
            return None


def _filter_out_small_weld_faces(face_seq, P):
    """
    Remove small weld faces from HSS cut plane.
```

```
    """
    kept = []
    removed = []

    area_limit = 2.0 * (P['weld_leg'] ** 2)

    for f in face_seq:
        area = _safe_face_area(f)

        if area is None:
            kept.append(f)
            continue

        if area > area_limit:
            kept.append(f)
        else:
            removed.append(f)

    return kept, removed


def _build_face_sequence_from_kept(faces_all, faces_keep):
    """
    Rebuild valid Abaqus face sequence.
    """
    keep_indices = []

    for i, f in enumerate(faces_all):
        for g in faces_keep:
            if f.index == g.index:
                keep_indices.append(i)
                break

    if len(keep_indices) == 0:
        raise ValueError('No kept faces after weld filtering.')

    seq = faces_all[keep_indices[0]:keep_indices[0] + 1]

    for idx in keep_indices[1:]:
        seq = seq + faces_all[idx:idx + 1]

    return seq


def get_hss_cut_face_arrays(inst, P):
    """
    Get HSS cut faces at:
    - actual x-left boundary
    - actual x-right boundary
    - actual y-bottom boundary
    - actual y-top boundary
    - actual z-top boundary
    """

    tol = max(0.2, 0.1 * P['tf'])

    xs = []
    ys = []
    zs = []

```

```
    for v in inst.vertices:
        try:
            x, y, z = v.pointOn[0]
            xs.append(x)
            ys.append(y)
            zs.append(z)
        except:
            pass

    if len(xs) == 0:
        raise ValueError('Could not read HSS instance vertices to detect cut faces.')

    x_left = min(xs)
    x_right = max(xs)
    y_bottom = min(ys)
    y_top = max(ys)
    z_top = max(zs)

    x_margin = 2.0 * P['L']
    y_margin = 2.0 * P['bf']
    z_margin = 2.0 * (P['tf'] + P['hw'])

    log_print('-----------------------------------')
    log_print('HSS actual cut face search')
    log_print('x_left actual   = %.3f' % x_left)
    log_print('x_right actual  = %.3f' % x_right)
    log_print('y_bottom actual = %.3f' % y_bottom)
    log_print('y_top actual    = %.3f' % y_top)
    log_print('z_top actual    = %.3f' % z_top)
    log_print('tol             = %.3f' % tol)
    log_print('-----------------------------------')

    faces_x_left = inst.faces.getByBoundingBox(
        xMin=x_left - tol,
        xMax=x_left + tol,
        yMin=-y_margin,
        yMax= y_margin,
        zMin=-tol,
        zMax= z_margin
    )

    faces_x_right = inst.faces.getByBoundingBox(
        xMin=x_right - tol,
        xMax=x_right + tol,
        yMin=-y_margin,
        yMax= y_margin,
        zMin=-tol,
        zMax= z_margin
    )

    faces_y_bottom = inst.faces.getByBoundingBox(
        xMin=-x_margin,
        xMax= x_margin,
        yMin=y_bottom - tol,
        yMax=y_bottom + tol,
        zMin=-tol,
        zMax= z_margin
    )

    faces_y_top = inst.faces.getByBoundingBox(
```

```
4176            xMin=-x_margin,
4177            xMax= x_margin,
4178            yMin=y_top - tol,
4179            yMax=y_top + tol,
4180            zMin=-tol,
4181            zMax= z_margin
4182        )
4183
4184        faces_z_top = inst.faces.getByBoundingBox(
4185            xMin=-x_margin,
4186            xMax= x_margin,
4187            yMin=-y_margin,
4188            yMax= y_margin,
4189            zMin=z_top - tol,
4190            zMax=z_top + tol
4191        )
4192
4193        return faces_x_left, faces_x_right, faces_y_bottom, faces_y_top, faces_z_top
4194
4195
4196
4197    def create_hss_cut_sets_and_surface_no_weld(model_hss, a, inst, P):
4198        """
4199        Create HSS cut sets/surface excluding very small weld faces.
4200        Includes all HSS run cut boundaries:
4201        x-left, x-right, y-bottom, y-top, z-top.
4202        """
4203
4204        faces_x_left_all, faces_x_right_all, faces_y_bottom_all, faces_y_top_all, faces_z_top_all =
        ↪  get_hss_cut_face_arrays(inst, P)
4205
4206        if len(faces_x_left_all) == 0:
4207            raise ValueError('No faces at HSS x-left cut')
4208        if len(faces_x_right_all) == 0:
4209            raise ValueError('No faces at HSS x-right cut')
4210        if len(faces_y_bottom_all) == 0:
4211            raise ValueError('No faces at HSS y-bottom cut')
4212        if len(faces_y_top_all) == 0:
4213            raise ValueError('No faces at HSS y-top cut')
4214        if len(faces_z_top_all) == 0:
4215            raise ValueError('No faces at HSS z-top cut')
4216
4217        faces_x_left_keep, faces_x_left_removed = _filter_out_small_weld_faces(
4218            faces_x_left_all, P
4219        )
4220        faces_x_right_keep, faces_x_right_removed = _filter_out_small_weld_faces(
4221            faces_x_right_all, P
4222        )
4223        faces_y_bottom_keep, faces_y_bottom_removed = _filter_out_small_weld_faces(
4224            faces_y_bottom_all, P
4225        )
4226        faces_y_top_keep, faces_y_top_removed = _filter_out_small_weld_faces(
4227            faces_y_top_all, P
4228        )
4229        faces_z_top_keep, faces_z_top_removed = _filter_out_small_weld_faces(
4230            faces_z_top_all, P
4231        )
4232
4233        seq_x_left = _build_face_sequence_from_kept(faces_x_left_all, faces_x_left_keep)
4234        seq_x_right = _build_face_sequence_from_kept(faces_x_right_all, faces_x_right_keep)
```

```
    seq_y_bottom = _build_face_sequence_from_kept(faces_y_bottom_all, faces_y_bottom_keep)
    seq_y_top = _build_face_sequence_from_kept(faces_y_top_all, faces_y_top_keep)
    seq_z_top = _build_face_sequence_from_kept(faces_z_top_all, faces_z_top_keep)

    for name in [
        'SET_HSS_CUT_LEFT',
        'SET_HSS_CUT_RIGHT',
        'SET_HSS_CUT_Y_BOTTOM',
        'SET_HSS_CUT_Y_TOP',
        'SET_HSS_CUT_Z_TOP',
        'SET_HSS_CUT_ALL'
    ]:
        if name in a.sets.keys():
            del a.sets[name]

    if 'SURF_HSS_CUT_ALL' in a.surfaces.keys():
        del a.surfaces['SURF_HSS_CUT_ALL']

    a.Set(name='SET_HSS_CUT_LEFT', faces=seq_x_left)
    a.Set(name='SET_HSS_CUT_RIGHT', faces=seq_x_right)
    a.Set(name='SET_HSS_CUT_Y_BOTTOM', faces=seq_y_bottom)
    a.Set(name='SET_HSS_CUT_Y_TOP', faces=seq_y_top)
    a.Set(name='SET_HSS_CUT_Z_TOP', faces=seq_z_top)

    a.SetByBoolean(
        name='SET_HSS_CUT_ALL',
        sets=(
            a.sets['SET_HSS_CUT_LEFT'],
            a.sets['SET_HSS_CUT_RIGHT'],
            a.sets['SET_HSS_CUT_Y_BOTTOM'],
            a.sets['SET_HSS_CUT_Y_TOP'],
            a.sets['SET_HSS_CUT_Z_TOP']
        ),
        operation=UNION
    )

    all_cut_faces = (
        seq_x_left +
        seq_x_right +
        seq_y_bottom +
        seq_y_top +
        seq_z_top
    )

    a.Surface(
        name='SURF_HSS_CUT_ALL',
        side1Faces=all_cut_faces
    )

    log_print('-----------------------------------')
    log_print('HSS cut surface with extra cuts:')
    log_print('x-left   kept = %d, removed = %d'
              % (len(faces_x_left_keep), len(faces_x_left_removed)))
    log_print('x-right  kept = %d, removed = %d'
              % (len(faces_x_right_keep), len(faces_x_right_removed)))
    log_print('y-bottom kept = %d, removed = %d'
              % (len(faces_y_bottom_keep), len(faces_y_bottom_removed)))
    log_print('y-top    kept = %d, removed = %d'
              % (len(faces_y_top_keep), len(faces_y_top_removed)))
    log_print('z-top    kept = %d, removed = %d'
```

```
4295                % (len(faces_z_top_keep), len(faces_z_top_removed)))
4296      log_print('SET  -> SET_HSS_CUT_ALL')
4297      log_print('SURF -> SURF_HSS_CUT_ALL')
4298      log_print('-----------------------------------')
4299
4300
4301  def create_hss_assembly_instance(model_hss):
4302      """
4303      Fresh assembly instance for HSS specimen.
4304      """
4305      a = model_hss.rootAssembly
4306      a.DatumCsysByDefault(CARTESIAN)
4307
4308      # suppress old assembly features
4309      for feat in list(a.features.keys()):
4310          try:
4311              a.features[feat].suppress()
4312          except:
4313              pass
4314
4315      part_obj = model_hss.parts['Specimen']
4316
4317      a.Instance(
4318          name='Specimen-1',
4319          part=part_obj,
4320          dependent=ON
4321      )
4322
4323      a.regenerate()
4324      return a, a.instances['Specimen-1']
4325
4326
4327  def link_hss_to_global_job(model_hss, global_job_name='Job-Global'):
4328      """
4329      Link HSS submodel to Global job.
4330      """
4331      model_hss.setValues(globalJob=global_job_name)
4332
4333
4334  def apply_hss_submodel_bc(model_hss, a):
4335      """
4336      Apply HSS SubmodelBC on assembly cut set.
4337      """
4338      if 'BC-HSS' in model_hss.boundaryConditions.keys():
4339          try:
4340              del model_hss.boundaryConditions['BC-HSS']
4341          except:
4342              pass
4343
4344      model_hss.SubmodelBC(
4345          name='BC-HSS',
4346          createStepName='Step-1',
4347          region=a.sets['SET_HSS_CUT_ALL'],
4348          dof=(1, 2, 3),
4349          exteriorTolerance=0.6,
4350          absoluteExteriorTolerance=None,
4351          globalDrivingRegion='',
4352          globalIncrement=0,
4353          globalStep='1',
4354
```

```
    )


def build_hss_submodel_bc_no_weld(P, global_job_name='Job-Global'):
    """
    Full HSS submodel setup (no weld faces in BC).
    """
    model_hss = mdb.models[HSS_MODEL_NAME]

    # 1) suppress old analysis objects
    clear_old_hss_analysis_objects(model_hss)

    # 2) fresh assembly
    a, inst = create_hss_assembly_instance(model_hss)

    # 3) create cut sets / surface without weld
    create_hss_cut_sets_and_surface_no_weld(model_hss, a, inst, P)

    # 4) link to Global
    link_hss_to_global_job(model_hss, global_job_name)

    # 5) apply HSS SubmodelBC
    apply_hss_submodel_bc(model_hss, a)

    log_print('====================================')
    log_print('HSS SubmodelBC ready (no weld)')
    log_print('Driven from:', global_job_name)
    log_print('Only BC-HSS should remain active')
    log_print('====================================')

# ============================================================
# 15C. ENS ASSEMBLY + SUBMODEL BC FROM HSS
#      (BC CUT SURFACE WITHOUT WELD REGION)
# ============================================================

def delete_old_ens_submodel_bcs(model_ens):
    """
    Delete old ENS/HSS submodel BCs completely before recreating assembly regions.
    """
    for bc_name in ['BC-ENS', 'BC-HSS']:
        if bc_name in model_ens.boundaryConditions.keys():
            try:
                del model_ens.boundaryConditions[bc_name]
                log_print('Old %s deleted.' % bc_name)
            except:
                pass


def clear_old_ens_analysis_objects(model_ens):
    """
    Suppress old loads, BCs, and constraints in ENS model,
    but skip BC-ENS and BC-HSS because they are deleted separately first.
    """
    for name in list(model_ens.loads.keys()):
        try:
            model_ens.loads[name].suppress()
        except:
            pass
```

```
4415      for name in list(model_ens.boundaryConditions.keys()):
4416          if name in ('BC-ENS', 'BC-HSS'):
4417              continue
4418          try:
4419              model_ens.boundaryConditions[name].suppress()
4420          except:
4421              pass
4422
4423      safe_suppress_all_constraints(model_ens)
4424
4425      log_print('Old ENS loads / BCs / constraints suppressed.')
4426
4427
4428  def get_ens_instance_name(model_name):
4429      """
4430      Return a short, stable assembly instance name for ENS models.
4431      """
4432      if model_name == ENS_SP4_MODEL_NAME:
4433          return 'Specimen-SP4-1'
4434      elif model_name == ENS_SP8_MODEL_NAME:
4435          return 'Specimen-SP8-1'
4436      else:
4437          return 'Specimen-ENS-1'
4438
4439
4440  def create_ens_assembly_instance(model_name):
4441      """
4442      Create fresh assembly instance for ENS specimen.
4443      """
4444      model_ens = mdb.models[model_name]
4445      a = model_ens.rootAssembly
4446      a.DatumCsysByDefault(CARTESIAN)
4447
4448      # suppress old assembly features
4449      for feat_name in list(a.features.keys()):
4450          try:
4451              a.features[feat_name].suppress()
4452          except:
4453              pass
4454
4455      part_obj = model_ens.parts['Specimen']
4456      instance_name = get_ens_instance_name(model_name)
4457
4458      if instance_name in a.instances.keys():
4459          try:
4460              del a.instances[instance_name]
4461          except:
4462              pass
4463
4464      a.Instance(
4465          name=instance_name,
4466          part=part_obj,
4467          dependent=ON
4468      )
4469
4470      a.regenerate()
4471      return model_ens, a, a.instances[instance_name]
4472
4473
4474  def _safe_ens_face_area(face_obj):
```

```
    try:
        return face_obj.getSize(printResults=False)
    except:
        try:
            return face_obj.getSize()
        except:
            return None

def _concat_face_sequences(face_seq_list):
    """
    Concatenate Abaqus face sequences safely.
    """
    out = None

    for seq in face_seq_list:
        if len(seq) == 0:
            continue

        if out is None:
            out = seq[0:1]
            for i in range(1, len(seq)):
                out = out + seq[i:i+1]
        else:
            for i in range(len(seq)):
                out = out + seq[i:i+1]

    return out


def get_ens_cut_faces_near_x(inst, x_value, P):
    tol = max(0.2, 0.1 * P['tf'])

    return inst.faces.getByBoundingBox(
        xMin=x_value - tol,
        xMax=x_value + tol,
        yMin=-2.0 * P['bf'],
        yMax= 2.0 * P['bf'],
        zMin=-tol,
        zMax=2.0 * (P['tf'] + P['hw'])
    )


def get_ens_cut_faces_near_y(inst, y_value, P):
    tol = max(0.2, 0.1 * P['tf'])
    x_margin = 2.0 * P['L']

    return inst.faces.getByBoundingBox(
        xMin=-x_margin,
        xMax= x_margin,
        yMin=y_value - tol,
        yMax=y_value + tol,
        zMin=-tol,
        zMax=2.0 * (P['tf'] + P['hw'])
    )


def create_ens_bc_cut_set_on_assembly(model_name, P, W, Y):
    """
    Create assembly-level ENS SubmodelBC set/surface from all final trim faces:
    - X boundary faces: x_left and x_right
```

```
4535        even if one side is inherited from HSS and was not cut now.
4536    - Y boundary faces: y_min and y_max
4537    """
4538    model_ens = mdb.models[model_name]
4539    a = model_ens.rootAssembly
4540
4541    inst_name = get_ens_instance_name(model_name)
4542    inst = a.instances[inst_name]
4543
4544    face_groups = []
4545
4546    # Always include both X boundary faces
4547    faces_x_left = get_ens_cut_faces_near_x(inst, W['x_left'], P)
4548    faces_x_right = get_ens_cut_faces_near_x(inst, W['x_right'], P)
4549
4550    face_groups.append(faces_x_left)
4551    face_groups.append(faces_x_right)
4552
4553    # Always include both Y boundary faces
4554    faces_y_min = get_ens_cut_faces_near_y(inst, Y['y_min'], P)
4555    faces_y_max = get_ens_cut_faces_near_y(inst, Y['y_max'], P)
4556
4557    face_groups.append(faces_y_min)
4558    face_groups.append(faces_y_max)
4559
4560    all_faces = _concat_face_sequences(face_groups)
4561
4562    if all_faces is None or len(all_faces) == 0:
4563        raise ValueError('No ENS BC cut faces found for model %s' % model_name)
4564
4565    if 'SET_ENS_BC_CUT' in a.sets.keys():
4566        del a.sets['SET_ENS_BC_CUT']
4567
4568    if 'SURF_ENS_BC_CUT' in a.surfaces.keys():
4569        del a.surfaces['SURF_ENS_BC_CUT']
4570
4571    a.Set(
4572        name='SET_ENS_BC_CUT',
4573        faces=all_faces
4574    )
4575
4576    a.Surface(
4577        name='SURF_ENS_BC_CUT',
4578        side1Faces=all_faces
4579    )
4580
4581    log_print('-----------------------------------', level=2)
4582    log_print('%s ENS BC cut surface from all boundary faces:' % model_name, level=2)
4583    log_print('x_left  = %.3f, faces = %d' % (W['x_left'], len(faces_x_left)), level=2)
4584    log_print('x_right = %.3f, faces = %d' % (W['x_right'], len(faces_x_right)), level=2)
4585    log_print('y_min   = %.3f, faces = %d' % (Y['y_min'], len(faces_y_min)), level=2)
4586    log_print('y_max   = %.3f, faces = %d' % (Y['y_max'], len(faces_y_max)), level=2)
4587    log_print('total BC faces = %d' % len(all_faces), level=2)
4588    log_print('SET  -> SET_ENS_BC_CUT', level=2)
4589    log_print('SURF -> SURF_ENS_BC_CUT', level=2)
4590    log_print('-----------------------------------', level=2)
4591
4592
4593 def link_ens_to_hss_job(model_name, hss_job_name='Job-HSS'):
4594     """
```

```
    Link ENS submodel to HSS job.
    """
    model_ens = mdb.models[model_name]
    model_ens.setValues(globalJob=hss_job_name)


def apply_ens_submodel_bc(model_name):
    """
    Apply ENS SubmodelBC on assembly-level cut set.
    """
    model_ens = mdb.models[model_name]
    a = model_ens.rootAssembly

    if 'BC-ENS' in model_ens.boundaryConditions.keys():
        try:
            del model_ens.boundaryConditions['BC-ENS']
        except:
            pass

    model_ens.SubmodelBC(
        name='BC-ENS',
        createStepName='Step-1',
        region=a.sets['SET_ENS_BC_CUT'],
        dof=(1, 2, 3),
        exteriorTolerance=0.6,
        absoluteExteriorTolerance=None,
        globalDrivingRegion='',
        globalIncrement=0,
        globalStep='1',

    )


def build_one_ens_submodel_bc(model_name, P, W, Y, hss_job_name='Job-HSS'):
    """
    Full ENS submodel boundary setup for one ENS model:
    - delete old BC-ENS / BC-HSS first
    - suppress old loads / BCs / constraints
    - fresh assembly instance
    - create assembly BC cut set/surface without weld region
    - link to HSS job
    - apply ENS SubmodelBC
    """
    model_ens = mdb.models[model_name]

    # 1) delete old ENS/HSS submodel BCs first
    delete_old_ens_submodel_bcs(model_ens)

    # 2) suppress other old analysis objects
    clear_old_ens_analysis_objects(model_ens)

    # 3) fresh assembly
    model_ens, a, inst = create_ens_assembly_instance(model_name)

    # 4) create BC cut set/surface without weld
    create_ens_bc_cut_set_on_assembly(model_name, P, W, Y)

    # 5) link to HSS
    link_ens_to_hss_job(model_name, hss_job_name=hss_job_name)
```

```
    # 6) apply ENS SubmodelBC
    apply_ens_submodel_bc(model_name)

    a.regenerate()

    log_print('====================================')
    log_print('%s ENS SubmodelBC ready (no weld)' % model_name)
    log_print('Driven from:', hss_job_name)
    log_print('Only BC-ENS should remain active')
    log_print('====================================')


def build_all_ens_submodel_bc(P, W4, W8, Y4, Y8, hss_job_name='Job-HSS'):
    build_one_ens_submodel_bc(ENS_SP4_MODEL_NAME, P, W4, Y4, hss_job_name=hss_job_name)
    build_one_ens_submodel_bc(ENS_SP8_MODEL_NAME, P, W8, Y8, hss_job_name=hss_job_name)

# ============================================================
# 16A. GLOBAL MESH
# ============================================================
def mesh_global_model(P):
    model = mdb.models[GLOBAL_MODEL_NAME]
    p = model.parts['Specimen']

    # delete old mesh if exists
    try:
        p.deleteMesh()
    except:
        pass

    try:
        p.regenerate()
    except:
        pass

    # element type
    p.setElementType(
        elemTypes=(
            ElemType(elemCode=C3D20R, elemLibrary=STANDARD),
            ElemType(elemCode=C3D15, elemLibrary=STANDARD),
            ElemType(elemCode=C3D10, elemLibrary=STANDARD)
        ),
        regions=(p.cells[:],)
    )

    # free tetra mesh
    p.setMeshControls(
        regions=p.cells[:],
        elemShape=TET,
        technique=FREE
    )

    # uniform global seed
    p.seedPart(
        size=P['mesh_size_global'],
        deviationFactor=0.1,
        minSizeFactor=0.1
    )

    p.generateMesh()
```

```
4715
4716  # ============================================================
4717  # 16B. HSS MESH
4718  # ============================================================
4719  def seed_hss_sampling_segments_by_number(part_obj, P):
4720      seg_set_names = [
4721          'SET_HSS_SP4_SEG_01',
4722          'SET_HSS_SP4_SEG_12',
4723          'SET_HSS_SP4_SEG_23',
4724          'SET_HSS_SP8_SEG_01',
4725          'SET_HSS_SP8_SEG_12',
4726          'SET_HSS_SP8_SEG_23',
4727      ]
4728
4729      for set_name in seg_set_names:
4730          if set_name not in part_obj.sets.keys():
4731              raise ValueError('%s not found in HSS part sets' % set_name)
4732
4733          part_obj.seedEdgeByNumber(
4734              edges=part_obj.sets[set_name].edges,
4735              number=P['hss_n_elem_per_segment'],
4736              constraint=FINER
4737          )
4738
4739      log_print('HSS sampling segments seeded with %d elements each.'
4740            % P['hss_n_elem_per_segment'])
4741
4742  def mesh_hss_model(P):
4743      model = mdb.models[HSS_MODEL_NAME]
4744      p = model.parts['Specimen']
4745
4746      try:
4747          p.deleteMesh()
4748      except:
4749          pass
4750
4751      try:
4752          p.regenerate()
4753      except:
4754          pass
4755
4756      p.setElementType(
4757          elemTypes=(
4758              ElemType(elemCode=C3D20R, elemLibrary=STANDARD),
4759              ElemType(elemCode=C3D15, elemLibrary=STANDARD),
4760              ElemType(
4761                  elemCode=C3D10,
4762                  elemLibrary=STANDARD,
4763                  secondOrderAccuracy=OFF,
4764                  distortionControl=DEFAULT
4765              )
4766          ),
4767          regions=(p.cells[:],)
4768      )
4769
4770      p.setMeshControls(
4771          regions=p.cells[:],
4772          elemShape=TET,
4773          technique=FREE
4774      )
```

```
4775
4776      p.seedPart(
4777          size=P['mesh_size_hss'],
4778          deviationFactor=0.1,
4779          minSizeFactor=0.1
4780      )
4781
4782      p.seedEdgeBySize(
4783          edges=p.edges,
4784          size=P['mesh_size_hss'],
4785          deviationFactor=0.1,
4786          minSizeFactor=0.1,
4787          constraint=FINER
4788      )
4789
4790      seed_hss_sampling_segments_by_number(p, P)
4791
4792      p.generateMesh()
4793
4794      log_print('HSS mesh generated successfully.')
4795      log_print('HSS element count = %d' % len(p.elements))
4796
4797
4798
4799  def create_cell_debug_sets(model_name):
4800      p = mdb.models[model_name].parts['Specimen']
4801
4802      for cid in [0, 1, 2]:
4803          if cid < len(p.cells):
4804              set_name = 'DEBUG_CELL_%d' % cid
4805
4806              if set_name in p.sets.keys():
4807                  del p.sets[set_name]
4808
4809              p.Set(
4810                  name=set_name,
4811                  cells=p.cells[cid:cid+1]
4812              )
4813
4814              log_print('%s created for cell %d' % (set_name, cid), level=2)
4815          else:
4816              log_print('cell %d does not exist in %s' % (cid, model_name), level=2)
4817
4818
4819  # ============================================================
4820  # 16C. ENS MESH (CLEAN VERSION)
4821  # ============================================================
4822
4823  def mesh_one_ens_model(model_name, P):
4824
4825      model = mdb.models[model_name]
4826      p = model.parts['Specimen']
4827
4828      # ----------------------------
4829      # Reset mesh
4830      # ----------------------------
4831      try:
4832          p.deleteMesh()
4833      except:
4834          pass
```

```
4835
4836        try:
4837            p.regenerate()
4838        except:
4839            pass
4840
4841        if len(p.cells) == 0:
4842            raise ValueError('%s has zero cells before mesh setup.' % model_name)
4843
4844        # -----------------------------
4845        # Element type
4846        # -----------------------------
4847        p.setElementType(
4848            elemTypes=(
4849                ElemType(elemCode=C3D20R, elemLibrary=STANDARD),
4850                ElemType(elemCode=C3D15, elemLibrary=STANDARD),
4851                ElemType(
4852                    elemCode=C3D10,
4853                    elemLibrary=STANDARD,
4854                    secondOrderAccuracy=OFF,
4855                    distortionControl=DEFAULT
4856                )
4857            ),
4858            regions=(p.cells[:],)
4859        )
4860
4861        # -----------------------------
4862        # Mesh control
4863        # -----------------------------
4864        p.setMeshControls(
4865            regions=p.cells[:],
4866            elemShape=TET,
4867            technique=FREE
4868        )
4869
4870        # -----------------------------
4871        # Delete old seeds
4872        # -----------------------------
4873        try:
4874            p.deleteSeeds(regions=p.edges[:])
4875        except:
4876            pass
4877
4878        # -----------------------------
4879        # 1) Global coarse mesh
4880        # -----------------------------
4881        p.seedEdgeBySize(
4882            edges=p.edges[:],
4883            size=P['mesh_size_ens_global'],
4884            deviationFactor=0.1,
4885            minSizeFactor=0.1,
4886            constraint=FINER
4887        )
4888
4889        # -----------------------------
4890        # 2) Transition mesh
4891        # -----------------------------
4892        if 'SET_ENS_FINAL_CELL_TRANSITION_EDGES' in p.sets.keys():
4893            p.seedEdgeBySize(
4894                edges=p.sets['SET_ENS_FINAL_CELL_TRANSITION_EDGES'].edges,
```

```
            size=P['mesh_size_ens_transition'],
            deviationFactor=0.1,
            minSizeFactor=0.1,
            constraint=FINER
        )

    # ----------------------------
    # 3) Fine mesh (toe) with adaptive size
    # ----------------------------
    toe_size = P['mesh_size_ens_toe']

    if P['theta'] > 65.0:
        toe_size = 0.18

    if 'SET_ENS_CENTERLINE_FINE' in p.sets.keys():
        p.seedEdgeBySize(
            edges=p.sets['SET_ENS_CENTERLINE_FINE'].edges,
            size=toe_size,
            deviationFactor=0.1,
            minSizeFactor=0.1,
            constraint=FINER
        )

    # ----------------------------
    # Generate mesh
    # ----------------------------
    p.generateMesh()

    if len(p.elements) == 0:
        raise ValueError('%s mesh generated zero elements.' % model_name)


# ----------------------------
# Wrappers
# ----------------------------
def mesh_one_ens_model_sp4(P):
    mesh_one_ens_model(ENS_SP4_MODEL_NAME, P)

def mesh_one_ens_model_sp8(P):
    mesh_one_ens_model(ENS_SP8_MODEL_NAME, P)

# ============================================================
# 17. JOB HELPERS
# ============================================================
def delete_job_if_exists(job_name):
    if job_name in mdb.jobs.keys():
        try:
            del mdb.jobs[job_name]
        except:
            pass


def create_standard_job(job_name, model_name,
                        num_cpus=JOB_NUM_CPUS,
                        num_gpus=JOB_NUM_GPUS,
                        memory_percent=JOB_MEMORY_PERCENT,
                        queue_name=JOB_QUEUE,
                        scratch_dir=JOB_SCRATCH):

    delete_job_if_exists(job_name)
```

```
4955
4956        mdb.Job(
4957            name=job_name,
4958            model=model_name,
4959            description='',
4960            type=ANALYSIS,
4961            atTime=None,
4962            waitMinutes=0,
4963            waitHours=0,
4964            queue=queue_name,
4965            memory=memory_percent,
4966            memoryUnits=PERCENTAGE,
4967            getMemoryFromAnalysis=True,
4968            explicitPrecision=SINGLE,
4969            nodalOutputPrecision=SINGLE,
4970            echoPrint=OFF,
4971            modelPrint=OFF,
4972            contactPrint=OFF,
4973            historyPrint=OFF,
4974            scratch=scratch_dir,
4975            resultsFormat=ODB,
4976            userSubroutine='',
4977            numCpus=num_cpus,
4978            numGPUs=num_gpus,
4979            numDomains=num_cpus,
4980            multiprocessingMode=DEFAULT
4981        )
4982
4983
4984    def submit_and_wait(job_name):
4985        mdb.jobs[job_name].submit(consistencyChecking=OFF)
4986        mdb.jobs[job_name].waitForCompletion()
4987        log_print('%s finished.' % job_name)
4988
4989    # ============================================================
4990    # 18. CREATE ALL JOBS
4991    # ============================================================
4992    def create_all_analysis_jobs():
4993        create_standard_job('Job-Global', GLOBAL_MODEL_NAME)
4994        create_standard_job('Job-HSS', HSS_MODEL_NAME)
4995        create_standard_job('Job-ENS-SP4', ENS_SP4_MODEL_NAME)
4996        create_standard_job('Job-ENS-SP8', ENS_SP8_MODEL_NAME)
4997
4998        log_print('All jobs created successfully.')
4999
5000    # ============================================================
5001    # 20. RESULTS EXTRACTION
5002    # ============================================================
5003
5004    # ============================================================
5005    # 20.A GLOBAL EXTRACTION - S11 ACROSS FLANGE WIDTH
5006    # ============================================================
5007    def extract_global_nominal_s11_from_width_path(P, job_name='Job-Global',
5008                                                   x_nom=0.0,
5009                                                   output_file='Global_Nominal_S11.csv'):
5010        odb_path = job_name + '.odb'
5011        odb = openOdb(path=odb_path)
5012
5013        vp = session.viewports['Viewport: 1']
5014        vp.setValues(displayedObject=odb)
```

```
    vp.makeCurrent()
    vp.maximize()

    vp.odbDisplay.setFrame(step=0, frame=-1)
    vp.odbDisplay.display.setValues(plotState=(CONTOURS_ON_UNDEF,))
    vp.odbDisplay.setPrimaryVariable(
        variableLabel='S',
        outputPosition=INTEGRATION_POINT,
        refinement=(COMPONENT, 'S11')
    )

    # ----------------------------------------
    # Path across flange width (Y direction)
    # at top flange surface z = tf
    # and fixed x = x_nom
    # ----------------------------------------
    p1 = (x_nom, -P['bf'] / 2.0, P['tf'])
    p2 = (x_nom,  P['bf'] / 2.0, P['tf'])

    path = session.Path(
        name='Path-Global-Flange-Width',
        type=POINT_LIST,
        expression=(p1, p2)
    )

    xy = session.XYDataFromPath(
        name='Global_Nominal_S11',
        path=path,
        includeIntersections=True,
        projectOntoMesh=True,
        pathStyle=PATH_POINTS,
        numIntervals=50,
        projectionTolerance=0.0,
        shape=UNDEFORMED,
        labelType=TRUE_DISTANCE,
        variable=(('S', INTEGRATION_POINT, ((COMPONENT, 'S11'),)),)
    )

    s_values = [y for x, y in xy.data]

    if len(s_values) == 0:
        odb.close()
        raise ValueError('No S11 values extracted from global flange-width path.')

    avg_s11 = sum(s_values) / float(len(s_values))

    log_print('====================================')
    log_print('Global nominal stress extraction done.')
    log_print('x_nom = %.3f' % x_nom)
    log_print('Average S11 = %.6f MPa' % avg_s11)
    log_print('====================================')

    with open(output_file, 'w') as f:
        f.write('Distance,S11\n')
        for x, y in xy.data:
            f.write('%.12g,%.12g\n' % (x, y))
        f.write('\nAverage_S11,%.12g\n' % avg_s11)

    odb.close()
```

```
5075        return avg_s11
5076
5077    # ============================================================
5078    # 20.B HSS EXTRACTION - MORE ACCURATE POINT-BY-POINT
5079    # ============================================================
5080    def _prepare_viewport_for_s11_path(odb):
5081        vp = session.viewports['Viewport: 1']
5082        vp.setValues(displayedObject=odb)
5083        vp.makeCurrent()
5084        vp.maximize()
5085
5086        step_names = list(odb.steps.keys())
5087
5088        if len(step_names) == 0:
5089            raise ValueError('No analysis steps found in ODB.')
5090
5091        last_step_index = len(step_names) - 1
5092        last_step = odb.steps[step_names[-1]]
5093
5094        if len(last_step.frames) == 0:
5095            raise ValueError('No frames found in ODB step: %s' % step_names[-1])
5096
5097        vp.odbDisplay.setFrame(
5098            step=last_step_index,
5099            frame=len(last_step.frames) - 1
5100        )
5101
5102        vp.odbDisplay.display.setValues(plotState=(CONTOURS_ON_UNDEF,))
5103        vp.odbDisplay.setPrimaryVariable(
5104            variableLabel='S',
5105            outputPosition=INTEGRATION_POINT,
5106            refinement=(COMPONENT, 'S11')
5107        )
5108
5109        return vp
5110
5111    def _make_tiny_point_path(point_xyz, half_len=0.05):
5112        """
5113        Build a very short path centered at the requested point.
5114        This is more probe-like than using one long path across all points.
5115        """
5116        x, y, z = point_xyz
5117
5118        p1 = (x - half_len, y, z)
5119        p2 = (x + half_len, y, z)
5120
5121        return (p1, p2)
5122
5123    def _extract_s11_at_single_point(job_name, point_xyz, xy_name, path_half_len=0.05):
5124        """
5125        Extract S11 near one single target point using a very short local path.
5126        Returns one scalar value, intended to be closer to manual probe behavior.
5127        """
5128        odb_path = job_name + '.odb'
5129        odb = openOdb(path=odb_path)
5130
5131        _prepare_viewport_for_s11_path(odb)
5132
5133        p1, p2 = _make_tiny_point_path(point_xyz, half_len=path_half_len)
5134
```

```
    path = session.Path(
        name='Path-' + xy_name,
        type=POINT_LIST,
        expression=(p1, p2)
    )

    xy = session.XYDataFromPath(
        name=xy_name,
        path=path,
        includeIntersections=True,
        projectOntoMesh=True,
        pathStyle=PATH_POINTS,
        numIntervals=1,
        projectionTolerance=0.0,
        shape=UNDEFORMED,
        labelType=TRUE_DISTANCE,
        variable=(('S', INTEGRATION_POINT, ((COMPONENT, 'S11'),)),)
    )

    data = list(xy.data)
    odb.close()

    if len(data) == 0:
        raise ValueError('No S11 value extracted for %s' % xy_name)

    # If Abaqus returns more than one point, take the middle/last representative value.
    if len(data) == 1:
        return data[0][1], data

    mid_index = int(len(data) / 2)
    return data[mid_index][1], data

def _extract_s11_at_single_point_from_odb(odb, point_xyz, xy_name, path_half_len=0.05):
    """
    Same extraction logic as _extract_s11_at_single_point, but reuses an
    already opened ODB to avoid repeated open/close overhead.
    """
    _prepare_viewport_for_s11_path(odb)

    p1, p2 = _make_tiny_point_path(point_xyz, half_len=path_half_len)

    path = session.Path(
        name='Path-' + xy_name,
        type=POINT_LIST,
        expression=(p1, p2)
    )

    xy = session.XYDataFromPath(
        name=xy_name,
        path=path,
        includeIntersections=True,
        projectOntoMesh=True,
        pathStyle=PATH_POINTS,
        numIntervals=1,
        projectionTolerance=0.0,
        shape=UNDEFORMED,
        labelType=TRUE_DISTANCE,
        variable=(('S', INTEGRATION_POINT, ((COMPONENT, 'S11'),)),)
    )

```

```
    data = list(xy.data)
    if len(data) == 0:
        raise ValueError('No S11 value extracted for %s' % xy_name)

    if len(data) == 1:
        return data[0][1], data

    mid_index = int(len(data) / 2)
    return data[mid_index][1], data

def calculate_hss_surface_stress_from_three_points(s04, s09, s14):
    """
    Quadratic extrapolation to weld toe / hot-spot location
    using values at:
        0.4t, 0.9t, 1.4t
    """
    return 2.52 * s04 - 2.24 * s09 + 0.72 * s14

def extract_one_hss_surface_stress(P, HS, side_name,
                                   job_name='Job-HSS',
                                   output_file=None):
    """
    Extract one HSS result using three separate local point-based reads:
        0.4t, 0.9t, 1.4t
    """
    path_half_len = 0.001
    odb = openOdb(path=job_name + '.odb')
    try:
        if side_name == 'SP4':
            p04 = HS['sp4_04']
            p09 = HS['sp4_09']
            p14 = HS['sp4_14']

            s04, data04 = _extract_s11_at_single_point_from_odb(
                odb, p04, 'HSS_SP4_S11_04', path_half_len
            )
            s09, data09 = _extract_s11_at_single_point_from_odb(
                odb, p09, 'HSS_SP4_S11_09', path_half_len
            )
            s14, data14 = _extract_s11_at_single_point_from_odb(
                odb, p14, 'HSS_SP4_S11_14', path_half_len
            )

        elif side_name == 'SP8':
            p04 = HS['sp8_04']
            p09 = HS['sp8_09']
            p14 = HS['sp8_14']

            s04, data04 = _extract_s11_at_single_point_from_odb(
                odb, p04, 'HSS_SP8_S11_04', path_half_len
            )
            s09, data09 = _extract_s11_at_single_point_from_odb(
                odb, p09, 'HSS_SP8_S11_09', path_half_len
            )
            s14, data14 = _extract_s11_at_single_point_from_odb(
                odb, p14, 'HSS_SP8_S11_14', path_half_len
            )

        else:
            raise ValueError('side_name must be SP4 or SP8')
```

```
    finally:
        odb.close()

    hss_s11 = calculate_hss_surface_stress_from_three_points(
        s04=s04,
        s09=s09,
        s14=s14
    )

    log_print('==================================')
    log_print('HSS extraction done for %s' % side_name)
    log_print('S11(0.4t) = %.6f MPa' % s04)
    log_print('S11(0.9t) = %.6f MPa' % s09)
    log_print('S11(1.4t) = %.6f MPa' % s14)
    log_print('HSS S11   = %.6f MPa' % hss_s11)
    log_print('==================================')

    if output_file is not None:
        with open(output_file, 'w') as f:
            f.write('PointLabel,S11\n')
            f.write('0.4t,%.12g\n' % s04)
            f.write('0.9t,%.12g\n' % s09)
            f.write('1.4t,%.12g\n' % s14)
            f.write('HSS_S11,%.12g\n' % hss_s11)

    return {
        'side': side_name,
        's04': s04,
        's09': s09,
        's14': s14,
        'hss_s11': hss_s11
    }

def extract_all_hss_surface_stresses(P, HS, job_name='Job-HSS', case_id=None):

    if case_id is None:
        file_sp4 = 'HSS_SP4_S11.csv'
        file_sp8 = 'HSS_SP8_S11.csv'
    else:
        file_sp4 = 'HSS_SP4_S11_case_%d.csv' % case_id
        file_sp8 = 'HSS_SP8_S11_case_%d.csv' % case_id

    r_sp4 = extract_one_hss_surface_stress(
        P=P,
        HS=HS,
        side_name='SP4',
        job_name=job_name,
        output_file=file_sp4
    )

    r_sp8 = extract_one_hss_surface_stress(
        P=P,
        HS=HS,
        side_name='SP8',
        job_name=job_name,
        output_file=file_sp8
    )

    return {
        'SP4': r_sp4,
```

```
        'SP8': r_sp8
    }

# ========================================================
# 20.C ENS MAX PRINCIPAL EXTRACTION
#      (NO SETS / NO EXTRA PARTITIONS)
# ========================================================

def _get_odb_instance_name_case_insensitive(odb, instance_name):
    for k in odb.rootAssembly.instances.keys():
        if k.upper() == instance_name.upper():
            return k
    raise ValueError('ODB instance %s not found.' % instance_name)


def _get_target_sp_name_from_model(model_name):
    if model_name == ENS_SP4_MODEL_NAME:
        return 'SP4'
    elif model_name == ENS_SP8_MODEL_NAME:
        return 'SP8'
    else:
        raise ValueError('Unknown ENS model name: %s' % model_name)


def _get_cut_x_values_for_ens_model(model_name, P, W):
    """
    Return x-locations of relevant cut planes:
    - actual ENS local cut plane(s) from Block 13
    - inherited HSS cut plane
    """
    x_list = []

    do_left_cut = W.get('do_left_cut_final', False)
    do_right_cut = W.get('do_right_cut_final', False)

    if do_left_cut:
        x_list.append(W['x_left'])

    if do_right_cut:
        x_list.append(W['x_right'])

    # inherited HSS cut plane
    if model_name == ENS_SP4_MODEL_NAME:
        x_list.append(0.0)
    elif model_name == ENS_SP8_MODEL_NAME:
        x_list.append(P['L'] - P['b'] / 4.0)
    else:
        raise ValueError('Unknown ENS model name: %s' % model_name)

    return x_list

# ========================================================
# 20.C.2 ENS NODE-AVERAGED MAX PRINCIPAL EXTRACTION
# ========================================================

def _get_target_sp_name_from_model(model_name):
    if model_name == ENS_SP4_MODEL_NAME:
        return 'SP4'
    elif model_name == ENS_SP8_MODEL_NAME:
        return 'SP8'
```

```
5375        else:
5376            raise ValueError('Unknown ENS model name: %s' % model_name)
5377
5378
5379    def _get_cut_x_values_for_ens_model(model_name, P, W):
5380        """
5381        Return x-locations of relevant cut planes:
5382        - actual ENS local cut plane(s) from Block 13
5383        - inherited HSS cut plane
5384        """
5385        x_list = []
5386
5387        do_left_cut = W.get('do_left_cut_final', False)
5388        do_right_cut = W.get('do_right_cut_final', False)
5389
5390        if do_left_cut:
5391            x_list.append(W['x_left'])
5392
5393        if do_right_cut:
5394            x_list.append(W['x_right'])
5395
5396        if model_name == ENS_SP4_MODEL_NAME:
5397            x_list.append(0.0)
5398        elif model_name == ENS_SP8_MODEL_NAME:
5399            x_list.append(P['L'] - P['b'] / 4.0)
5400        else:
5401            raise ValueError('Unknown ENS model name: %s' % model_name)
5402
5403        return x_list
5404
5405
5406    def _get_target_point_for_ens_probe(P, S, H, model_name):
5407        target_sp = _get_target_sp_name_from_model(model_name)
5408
5409        if target_sp == 'SP4':
5410            toe = H['toe_sp4']
5411        elif target_sp == 'SP8':
5412            toe = H['toe_sp8']
5413        else:
5414            raise ValueError('Unknown target SP: %s' % target_sp)
5415
5416        return (
5417            toe[0],
5418            toe[1],
5419            P['tf'] + 0.5 * P['notch_radius']
5420        )
5421
5422
5423    def extract_max_principal_from_ens_node_avg(model_name,
5424                                                job_name,
5425                                                P,
5426                                                S,
5427                                                H,
5428                                                W,
5429                                                output_file=None):
5430        """
5431        Extract value close to Abaqus Probe when contour is shown as:
5432            S, Max. Principal (Avg: 100%)
5433
5434        Strategy:
```

```
    - use field S at ELEMENT_NODAL
    - filter local nodes around target S-point
    - exclude only nodes very near cut planes
    - average repeated contributions for each node
    - then take the maximum averaged nodal value
    """
    odb = openOdb(path=job_name + '.odb')

    inst_name_model = get_ens_instance_name(model_name)
    inst_name_odb = _get_odb_instance_name_case_insensitive(odb, inst_name_model)
    inst = odb.rootAssembly.instances[inst_name_odb]
    node_coords = {}
    for n in inst.nodes:
        node_coords[n.label] = n.coordinates

    step_names = list(odb.steps.keys())
    if len(step_names) == 0:
        odb.close()
        raise ValueError('No analysis steps found in %s' % job_name)

    last_step = odb.steps[step_names[-1]]
    if len(last_step.frames) == 0:
        odb.close()
        raise ValueError('No frames found in last step of %s' % job_name)

    last_frame = last_step.frames[-1]
    stress_field = last_frame.fieldOutputs['S']
    stress_nodal = stress_field.getSubset(position=ELEMENT_NODAL)

    target_pt = _get_target_point_for_ens_probe(P, S, H, model_name)
    x_cut_list = _get_cut_x_values_for_ens_model(model_name, P, W)

    # same local region that worked well for hotspot capture
    x_half = 4.0
    y_half = 4.0
    z_half = 0.5
    cut_tol = 1.5

    # ------------------------------------------------
    # collect all nodal contributions
    # because ELEMENT_NODAL returns repeated values
    # from different attached elements
    # ------------------------------------------------
    node_sum = {}
    node_count = {}
    node_coord = {}
    node_elem_labels = {}

    for v in stress_nodal.values:
        try:
            node_label = v.nodeLabel
        except:
            continue

        if node_label is None:
            continue

        if node_label not in node_coords:
            continue
```

```
        x, y, z = node_coords[node_label]

        # exclude nodes too close to cut planes
        reject = False
        for x_cut in x_cut_list:
            if abs(x - x_cut) < cut_tol:
                reject = True
                break
        if reject:
            continue

        # keep only local region around target
        if abs(x - target_pt[0]) > x_half:
            continue
        if abs(y - target_pt[1]) > y_half:
            continue
        if abs(z - target_pt[2]) > z_half:
            continue

        mp = v.maxPrincipal

        if node_label not in node_sum:
            node_sum[node_label] = 0.0
            node_count[node_label] = 0
            node_coord[node_label] = (x, y, z)
            node_elem_labels[node_label] = []

        node_sum[node_label] += mp
        node_count[node_label] += 1

        try:
            node_elem_labels[node_label].append(v.elementLabel)
        except:
            pass

    if len(node_sum) == 0:
        odb.close()
        raise ValueError('No accepted nodal values found in %s' % model_name)

    # ------------------------------------------------
    # average per node, then find max averaged node
    # ------------------------------------------------
    max_node = None
    max_stress = None
    max_xyz = None
    max_attached_elems = None
    max_raw_count = None

    for node_label in node_sum.keys():
        avg_mp = node_sum[node_label] / float(node_count[node_label])

        if (max_stress is None) or (avg_mp > max_stress):
            max_stress = avg_mp
            max_node = node_label
            max_xyz = node_coord[node_label]
            max_attached_elems = sorted(set(node_elem_labels[node_label]))
            max_raw_count = node_count[node_label]

    odb.close()
```

```
5555      log_print('==================================')
5556      log_print('ENS node-avg max principal extraction done')
5557      log_print('model         = %s' % model_name)
5558      log_print('job           = %s' % job_name, level=2)
5559      log_print('target pt     = %s' % str(target_pt))
5560      log_print('x_half        = %.6f' % x_half, level=2)
5561      log_print('y_half        = %.6f' % y_half, level=2)
5562      log_print('z_half        = %.6f' % z_half, level=2)
5563      log_print('cut tol       = %.6f' % cut_tol, level=2)
5564      log_print('accepted nodes= %d' % len(node_sum), level=2)
5565      log_print('max princ     = %.6f MPa' % max_stress)
5566      log_print('node label    = %s' % str(max_node))
5567      log_print('node coords   = %s' % str(max_xyz))
5568      log_print('n contribs    = %s' % str(max_raw_count), level=2)
5569      log_print('elements      = %s' % str(max_attached_elems), level=2)
5570      log_print('==================================')
5571
5572      if output_file is not None:
5573          with open(output_file, 'w') as f:
5574              f.write('ModelName,JobName,MaxPrincipalNodeAvg,NodeLabel,X,Y,Z,NContribs,AttachedElements\n')
5575              f.write('%s,%s,%.12g,%s,%.12g,%.12g,%.12g,%s,"%s"\n' % (
5576                  model_name,
5577                  job_name,
5578                  max_stress,
5579                  str(max_node),
5580                  max_xyz[0], max_xyz[1], max_xyz[2],
5581                  str(max_raw_count),
5582                  ','.join([str(e) for e in max_attached_elems])
5583              ))
5584
5585      return {
5586          'model_name': model_name,
5587          'job_name': job_name,
5588          'max_principal_node_avg': max_stress,
5589          'node_label': max_node,
5590          'node_xyz': max_xyz,
5591          'accepted_nodes': len(node_sum),
5592          'n_contribs': max_raw_count,
5593          'attached_elements': max_attached_elems
5594      }
5595
5596  def extract_all_ens_max_principal_node_avg(P, S, H, W4, W8, job_name_ens4, job_name_ens8, case_id=None):
5597
5598      if case_id is None:
5599          file_sp4 = 'ENS_SP4_NodeAvg.csv'
5600          file_sp8 = 'ENS_SP8_NodeAvg.csv'
5601      else:
5602          file_sp4 = 'ENS_SP4_NodeAvg_case_%d.csv' % case_id
5603          file_sp8 = 'ENS_SP8_NodeAvg_case_%d.csv' % case_id
5604
5605      r4 = extract_max_principal_from_ens_node_avg(
5606          model_name=ENS_SP4_MODEL_NAME,
5607          job_name=job_name_ens4,
5608          P=P,
5609          S=S,
5610          H=H,
5611          W=W4,
5612          output_file=file_sp4
5613      )
5614
```

```
    r8 = extract_max_principal_from_ens_node_avg(
        model_name=ENS_SP8_MODEL_NAME,
        job_name=job_name_ens8,
        P=P,
        S=S,
        H=H,
        W=W8,
        output_file=file_sp8
    )

    return {
        'SP4': r4,
        'SP8': r8
    }

# ============================================================
# 21. BATCH CSV INPUT / SUMMARY CSV OUTPUT
# ============================================================
INPUT_CSV_FILE = 'input_cases.csv'
RESULTS_SUMMARY_FILE = os.environ.get(
    'RESULTS_SUMMARY_FILE',
    'Batch_Results.csv'
)

INPUT_COLUMNS = [
    'case_id',
    'theta', 'R', 'bf', 'tf', 'tw', 'hw', 'b', 'hr',
    'couple_web_to_rp', 'restrain_web_top_z'
]

RESULT_COLUMNS = [
    'case_id',
    'theta', 'R', 'bf', 'tf', 'tw', 'hw', 'b', 'hr',
    'global_s11_L',
    'hss_sp4_s11', 'hss_sp8_s11',
    'ens_sp4_max_principal', 'ens_sp8_max_principal',
    'scf_hss_sp4_L', 'scf_hss_sp8_L',
    'scf_ens_sp4_L', 'scf_ens_sp8_L',
    'status', 'error_message'
]

def ensure_dir(path_name):
    if not os.path.isdir(path_name):
        os.makedirs(path_name)

def parse_float_from_csv(key, value):
    if value is None:
        raise ValueError('Missing value for %s' % key)
    txt = str(value).strip()
    if txt == '':
        raise ValueError('Empty value for %s' % key)
    try:
        return float(txt)
    except:
        raise ValueError('Invalid numeric value for %s: %s' % (key, txt))

def parse_bc_01_from_csv(key, value):
    if value is None:
        raise ValueError('Missing BC value for %s' % key)
    txt = str(value).strip()
```

```
    if txt not in ('0', '1'):
        raise ValueError('%s must be 0 or 1, got: %s' % (key, txt))
    return bool(int(txt))

def read_cases_from_csv(csv_file):
    """
    Required input columns:
    case_id,theta,R,bf,tf,tw,hw,b,hr,couple_web_to_rp,restrain_web_top_z

    Material, mesh, weld, and load values stay fixed inside get_params().
    BC options must be 0 or 1.
    """
    if not os.path.isfile(csv_file):
        raise ValueError('Input CSV file not found: %s' % csv_file)

    cases = []

    with open(csv_file, 'r') as f:
        reader = csv.DictReader(f)

        if reader.fieldnames is None:
            raise ValueError('Input CSV has no header row.')

        fieldnames = [str(k).strip() for k in reader.fieldnames]
        missing = []
        for col in INPUT_COLUMNS:
            if col not in fieldnames:
                missing.append(col)
        if len(missing) > 0:
            raise ValueError('Missing required input columns: %s' % ', '.join(missing))

        for row_index, row in enumerate(reader):
            overrides = {}

            for key in ['theta', 'R', 'bf', 'tf', 'tw', 'hw', 'b', 'hr']:
                overrides[key] = parse_float_from_csv(key, row.get(key, None))

            overrides['couple_web_to_rp'] = parse_bc_01_from_csv(
                'couple_web_to_rp', row.get('couple_web_to_rp', None)
            )
            overrides['restrain_web_top_z'] = parse_bc_01_from_csv(
                'restrain_web_top_z', row.get('restrain_web_top_z', None)
            )

            P = get_params(overrides)
            P['case_id'] = int(parse_float_from_csv('case_id', row.get('case_id', None)))
            P['global_n_waves'] = int(P['global_n_waves'])
            P['hss_n_elem_per_segment'] = int(P['hss_n_elem_per_segment'])
            cases.append(P)

    if len(cases) == 0:
        raise ValueError('No cases found in input CSV file: %s' % csv_file)

    return cases

def write_input_template(csv_file):
    if os.path.isfile(csv_file):
        return

    P0 = get_params()
```

```
    with open(csv_file, 'w') as f:
        writer = csv.DictWriter(f, fieldnames=INPUT_COLUMNS)
        writer.writeheader()
        writer.writerow({
            'case_id': 1,
            'theta': P0['theta'],
            'R': P0['R'],
            'bf': P0['bf'],
            'tf': P0['tf'],
            'tw': P0['tw'],
            'hw': P0['hw'],
            'b': P0['b'],
            'hr': P0['hr'],
            'couple_web_to_rp': 1 if P0['couple_web_to_rp'] else 0,
            'restrain_web_top_z': 1 if P0['restrain_web_top_z'] else 0
        })

def init_summary_csv(summary_file):
    with open(summary_file, 'w') as f:
        writer = csv.DictWriter(f, fieldnames=RESULT_COLUMNS)
        writer.writeheader()

def safe_round(value, ndigits=2):
    if value is None:
        return ''
    try:
        return round(float(value), ndigits)
    except:
        return value

def safe_scf(local_stress, global_stress):
    try:
        g = float(global_stress)
        if abs(g) < 1.0e-12:
            return ''
        return float(local_stress) / g
    except:
        return ''

def make_base_result_row(case_id, P, status='RUNNING', error_message=''):
    row = {}
    for col in RESULT_COLUMNS:
        row[col] = ''

    row['case_id'] = case_id
    row['theta'] = safe_round(P['theta'], 2)
    row['R'] = safe_round(P['R'], 2)
    row['bf'] = safe_round(P['bf'], 2)
    row['tf'] = safe_round(P['tf'], 2)
    row['tw'] = safe_round(P['tw'], 2)
    row['hw'] = safe_round(P['hw'], 2)
    row['b'] = safe_round(P['b'], 2)
    row['hr'] = safe_round(P['hr'], 2)
    row['status'] = status
    row['error_message'] = error_message

    return row

def append_summary_row(summary_file, row):
    with open(summary_file, 'a') as f:
```

```
5795            writer = csv.DictWriter(f, fieldnames=RESULT_COLUMNS)
5796            writer.writerow(row)
5797
5798
5799    def apply_ens_z_cut_only(P):
5800        """
5801        Apply only top Z cut to both ENS models.
5802        Cut location: z = 3*tf.
5803        """
5804
5805        for model_name in [ENS_SP4_MODEL_NAME, ENS_SP8_MODEL_NAME]:
5806
5807            model_obj = mdb.models[model_name]
5808            p = model_obj.parts['Specimen']
5809
5810            z_cut = 3.0 * P['tf']
5811
5812            cut_part_with_z_plane_keep_below(
5813                model_obj=model_obj,
5814                part_obj=p,
5815                P=P,
5816                z_cut=z_cut,
5817                sketch_name='__ens_z_cut_top__'
5818            )
5819
5820            p.regenerate()
5821
5822            log_print('%s Z cut applied at z = %.3f' % (model_name, z_cut))
5823
5824
5825    def apply_ens_z_cut_only_sp4(P):
5826        model_obj = mdb.models[ENS_SP4_MODEL_NAME]
5827        p = model_obj.parts['Specimen']
5828
5829        z_cut = 3.0 * P['tf']
5830
5831        cut_part_with_z_plane_keep_below(
5832            model_obj=model_obj,
5833            part_obj=p,
5834            P=P,
5835            z_cut=z_cut,
5836            sketch_name='__ens_sp4_z_cut_top__'
5837        )
5838
5839        p.regenerate()
5840
5841        log_print('ENS-SP4 Z cut applied at z = %.3f' % z_cut)
5842
5843
5844    def trim_final_ens_sp4_after_partitions(P, W4):
5845        log_print('APPLYING FINAL ENS-SP4 X TRIMMING')
5846
5847        z_mid = 0.5 * (P['tf'] + P['hw'])
5848        model4 = mdb.models[ENS_SP4_MODEL_NAME]
5849        p4 = model4.parts['Specimen']
5850
5851        if W4['do_left_cut_final']:
5852            cut_part_with_x_plane_keep_right(
5853                model_obj=model4,
5854                part_obj=p4,
```

```
            P=P,
            x_cut=W4['x_left'],
            z_mid=z_mid,
            sketch_name='__final_trim_sp4_left__'
        )
        p4.regenerate()

    if W4['do_right_cut_final']:
        cut_part_with_x_plane_keep_left(
            model_obj=model4,
            part_obj=p4,
            P=P,
            x_cut=W4['x_right'],
            z_mid=z_mid,
            sketch_name='__final_trim_sp4_right__'
        )
        p4.regenerate()

    log_print('ENS-SP4 X trimmed: x_left = %.3f, x_right = %.3f'
              % (W4['x_left'], W4['x_right']))


def trim_final_ens_sp4_in_y(P, Y4):
    log_print('APPLYING FINAL ENS-SP4 Y TRIMMING')

    z_mid = 0.5 * (P['tf'] + P['hw'])
    model4 = mdb.models[ENS_SP4_MODEL_NAME]
    p4 = model4.parts['Specimen']

    cut_part_with_y_plane_keep_positive(
        model_obj=model4,
        part_obj=p4,
        P=P,
        y_cut=Y4['y_min'],
        z_mid=z_mid,
        sketch_name='__final_trim_sp4_y_min__'
    )
    p4.regenerate()

    cut_part_with_y_plane_keep_negative(
        model_obj=model4,
        part_obj=p4,
        P=P,
        y_cut=Y4['y_max'],
        z_mid=z_mid,
        sketch_name='__final_trim_sp4_y_max__'
    )
    p4.regenerate()

    log_print('ENS-SP4 Y trimmed: y_min = %.3f, y_max = %.3f'
              % (Y4['y_min'], Y4['y_max']))


def run_one_case(case_id, P, summary_file):
    global CURRENT_CASE_TAG
    CURRENT_CASE_TAG = 'case_%d' % case_id

    case_dir = os.path.join(RESULTS_ROOT_DIR, 'case_%d' % case_id)
    ensure_dir(case_dir)

```

```
    row = make_base_result_row(case_id, P)
    t_case = time.time()

    def print_time(label):
        log_print('%s time = %.2f min' % (label, (time.time() - t_case) / 60.0))

    job_name_global = 'Job-Global-%d' % case_id
    job_name_hss = 'Job-HSS-%d' % case_id
    job_name_ens4 = 'Job-ENS-SP4-%d' % case_id
    job_name_ens8 = 'Job-ENS-SP8-%d' % case_id

    log_print('==============================')
    log_print('Running case %d' % case_id)
    log_print('==============================')

    try:
        # ----------------------------
        # GLOBAL
        # ----------------------------
        model, specimen_part, P, G, S = build_global_model(P)
        build_global_bc_and_load(model, P)
        mesh_global_model(P)
        log_print('Global elements = %d' % len(mdb.models[GLOBAL_MODEL_NAME].parts['Specimen'].elements))
        print_time('After Global mesh')
        log_print('START creating Global job')


        create_standard_job(job_name_global, GLOBAL_MODEL_NAME)
        log_print('START submitting Global job')


        submit_and_wait(job_name_global)
        log_print('END Global job')
        print_time('After Global job')

        # ----------------------------
        # HSS
        # ----------------------------
        H = get_hss_toe_points(P, S)
        HS = get_hss_sampling_points(P, H)

        W4_preview, W8_preview = preview_final_ens_trim_windows(P, H)
        Y4_preview, Y8_preview = preview_final_ens_y_trim_windows(P, H)
        HSSW = get_hss_run_cut_window_from_ens(P, W4_preview, W8_preview)

        model_hss, specimen_hss, C = build_hss_local_cut_model(P)
        add_hss_double_fillet_welds(P)
        partition_all_hss_side_faces_for_weld(P, C)

        # save wide HSS for ENS before trimming
        save_hss_source_model_for_ens()

        # trim only HSS run model
        model_hss, specimen_hss, C = trim_hss_run_model(P, HSSW)
        apply_extra_hss_run_cuts(P, H)
        Y4_preview, Y8_preview = preview_final_ens_y_trim_windows(P, H)

        apply_hss_sampling_chains(P, H, HS)

        build_hss_submodel_bc_no_weld(P, global_job_name=job_name_global)
```

```
5975        mesh_hss_model(P)
5976
5977        log_print('HSS elements = %d' % len(mdb.models[HSS_MODEL_NAME].parts['Specimen'].elements))
5978        print_time('After HSS mesh')
5979
5980        create_standard_job(job_name_hss, HSS_MODEL_NAME)
5981        submit_and_wait(job_name_hss)
5982        print_time('After HSS job')
5983
5984        # ----------------------------
5985        # ENS
5986        # ----------------------------
5987        model_ens_sp4, specimen_sp4 = build_initial_ens_models()
5988
5989        add_ens_double_notch_welds_to_model(ENS_SP4_MODEL_NAME, P)
5990        #add_ens_double_notch_welds_to_model(ENS_SP8_MODEL_NAME, P)
5991
5992        add_all_local_partitions_to_ens_sp4_model(P)
5993        #add_all_local_partitions_to_ens_sp8_model(P)
5994
5995        #apply_ens_z_cut_only(P)
5996        apply_ens_z_cut_only_sp4(P)
5997
5998        trim_final_ens_sp4_after_partitions(P, W4_preview)
5999        trim_final_ens_sp4_in_y(P, Y4_preview)
6000
6001        create_ens_transition_edges_from_final_cells(ENS_SP4_MODEL_NAME, P)
6002        split_final_transition_to_fine_transition(ENS_SP4_MODEL_NAME, P, Y4_preview)
6003
6004        build_one_ens_submodel_bc(
6005            ENS_SP4_MODEL_NAME,
6006            P,
6007            W4_preview,
6008            Y4_preview,
6009            hss_job_name=job_name_hss
6010        )
6011
6012        create_cell_debug_sets(ENS_SP4_MODEL_NAME)
6013
6014        mesh_one_ens_model(ENS_SP4_MODEL_NAME, P)
6015
6016        print_time('After ENS mesh')
6017        #return row
6018
6019        create_standard_job(job_name_ens4, ENS_SP4_MODEL_NAME)
6020
6021        submit_and_wait(job_name_ens4)
6022        print_time('After ENS-SP4 job')
6023
6024        if not os.path.isfile(job_name_ens4 + '.odb'):
6025            raise ValueError('%s.odb was not created. Check %s.msg and %s.dat'
6026                    % (job_name_ens4, job_name_ens4, job_name_ens4))
6027
6028        # ----------------------------
6029        # GLOBAL EXTRACTION - ONLY x = L
6030        # ----------------------------
6031        t_ext = time.time()
6032        global_s11_L = extract_global_nominal_s11_from_width_path(
6033            P=P,
6034            job_name=job_name_global,
```

```
            x_nom=P['L'],
            output_file=os.path.join(
                case_dir,
                'Global_Nominal_S11_case_%d_L.csv' % case_id
            )
        )
        log_print('Global extraction time = %.2f min' % ((time.time() - t_ext) / 60.0))

        row['global_s11_L'] = safe_round(global_s11_L, 2)

        # ------------------------------
        # HSS EXTRACTION
        # ------------------------------
        t_ext = time.time()
        hss_results = extract_all_hss_surface_stresses(
            P=P,
            HS=HS,
            job_name=job_name_hss,
            case_id=case_id
        )

        for fname in [
            'HSS_SP4_S11_case_%d.csv' % case_id,
            'HSS_SP8_S11_case_%d.csv' % case_id
        ]:
            if os.path.isfile(fname):
                try:
                    shutil.move(fname, os.path.join(case_dir, fname))
                except:
                    pass

        hss_sp4 = hss_results['SP4']['hss_s11']
        hss_sp8 = hss_results['SP8']['hss_s11']
        log_print('HSS extraction time = %.2f min' % ((time.time() - t_ext)/60.0))
        row['hss_sp4_s11'] = safe_round(hss_sp4, 2)
        row['hss_sp8_s11'] = safe_round(hss_sp8, 2)

        # ------------------------------
        # ENS EXTRACTION - SP4 ONLY
        # ------------------------------
        t_ext = time.time()

        ens_sp4_result = extract_max_principal_from_ens_node_avg(
            model_name=ENS_SP4_MODEL_NAME,
            job_name=job_name_ens4,
            P=P,
            S=S,
            H=H,
            W=W4_preview,
            output_file='ENS_SP4_NodeAvg_case_%d.csv' % case_id
        )

        fname = 'ENS_SP4_NodeAvg_case_%d.csv' % case_id
        if os.path.isfile(fname):
            try:
                shutil.move(fname, os.path.join(case_dir, fname))
            except:
                pass

        ens_sp4 = ens_sp4_result['max_principal_node_avg']
```

```
6095            ens_sp8 = None
6096
6097            log_print('ENS-SP4 extraction time = %.2f min'
6098                      % ((time.time() - t_ext) / 60.0))
6099
6100            row['ens_sp4_max_principal'] = safe_round(ens_sp4, 2)
6101            row['ens_sp8_max_principal'] = ''
6102
6103            # ----------------------------
6104            # SCF - BASED ONLY ON global_s11_L
6105            # ----------------------------
6106            row['scf_hss_sp4_L'] = safe_round(safe_scf(hss_sp4, global_s11_L), 2)
6107            row['scf_hss_sp8_L'] = safe_round(safe_scf(hss_sp8, global_s11_L), 2)
6108            row['scf_ens_sp4_L'] = safe_round(safe_scf(ens_sp4, global_s11_L), 2)
6109            row['scf_ens_sp8_L'] = ''
6110
6111            row['status'] = 'OK'
6112            row['error_message'] = ''
6113
6114            total_case_time = (time.time() - t_case) / 60.0
6115            log_print('TOTAL case %d time = %.2f min' % (case_id, total_case_time))
6116
6117            log_print('Finished case %d successfully.' % case_id)
6118
6119        except Exception as e:
6120            row['status'] = 'FAILED'
6121            row['error_message'] = str(e)
6122            log_print('ERROR in case %d: %s' % (case_id, str(e)))
6123
6124        append_summary_row(summary_file, row)
6125        return row
6126
6127    def run_batch_from_csv(input_csv=INPUT_CSV_FILE, summary_csv=RESULTS_SUMMARY_FILE):
6128        ensure_dir(RESULTS_ROOT_DIR)
6129        ensure_dir(ODB_ROOT_DIR)
6130        ensure_dir(SUMMARY_ROOT_DIR)
6131
6132        write_input_template(input_csv)
6133
6134        cases = read_cases_from_csv(input_csv)
6135        slurm_id = os.environ.get('SLURM_ARRAY_TASK_ID', None)
6136
6137        if slurm_id is not None:
6138            target_index = int(slurm_id) - 1
6139
6140            if target_index < 0 or target_index >= len(cases):
6141                raise ValueError(
6142                    'SLURM_ARRAY_TASK_ID = %d is outside CSV range. Number of cases = %d'
6143                    % (int(slurm_id), len(cases))
6144                )
6145
6146            cases = [cases[target_index]]
6147
6148        init_summary_csv(summary_csv)
6149
6150        log_print('====================================')
6151        log_print('Batch input CSV  = %s' % input_csv)
6152        log_print('Summary output   = %s' % summary_csv)
6153        log_print('Number of cases  = %d' % len(cases))
6154        log_print('Case numbering starts from 1')
```

```
    log_print('SCF reference    = global_s11_L only')
    log_print('====================================')

    rows = []

    for i, P in enumerate(cases, start=1):
        case_id = int(P['case_id'])
        log_print('Running input case_id %d (index %d/%d)'
                  % (case_id, i, len(cases)))

        rows.append(run_one_case(case_id, P, summary_csv))

    log_print('====================================')
    log_print('Batch finished.')
    log_print('Summary CSV written to: %s' % summary_csv)
    log_print('====================================')

    return rows

# ============================================================
# 22. MAIN RUN
# ============================================================
run_batch_from_csv(INPUT_CSV_FILE, RESULTS_SUMMARY_FILE)
```

Appendix E

Results Obtained from Numerical Analysis

This appendix presents the complete numerical database generated from the parametric finite element analyses used for the development of the proposed SCF_{ENS} equations.

E.1 Parametric Study Results for $R/t_w = 2$

Parametric study results for the development of the SCF-ENS equation for R/tw = 2

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
508	30	8	400	20	4	120	600	400	200	47.96	47.51	168.44	3.51	3.55
532	30	8	400	40	4	240	600	400	200	23.71	23.76	93.32	3.94	3.93
556	30	8	400	80	4	480	600	400	200	11.96	11.88	46.55	3.89	3.92
580	30	8	800	40	4	240	600	400	200	12.1	12.22	47.95	3.96	3.92
604	30	8	800	60	4	360	600	400	200	8.08	8.15	31.96	3.96	3.92
652	30	8	1200	40	4	240	600	400	200	8.19	8.23	32.2	3.93	3.91
676	30	8	1200	60	4	360	600	400	200	5.46	5.49	21.48	3.93	3.92
700	30	8	1200	80	4	480	600	400	200	4.12	4.11	15.89	3.86	3.86
244	30	8	200	20	4	120	300	200	100	91.96	89.93	303.26	3.3	3.37
268	30	8	200	40	4	240	300	200	100	46.45	44.97	152.54	3.28	3.39
292	30	8	400	20	4	120	300	200	100	47.37	47.51	158.41	3.34	3.33
316	30	8	400	40	4	240	300	200	100	23.96	23.76	78.3	3.27	3.30
340	30	8	400	60	4	360	300	200	100	16.24	15.84	51.65	3.18	3.26
364	30	8	800	40	4	240	300	200	100	12.3	12.22	39.97	3.25	3.27
388	30	8	800	60	4	360	300	200	100	8.27	8.15	26.29	3.18	3.23
436	30	8	1200	40	4	240	300	200	100	8.26	8.23	26.91	3.26	3.27
460	30	8	1200	60	4	360	300	200	100	5.54	5.49	17.75	3.2	3.24
514	30	16	400	20	8	120	600	400	200	45.38	44.97	162.41	3.58	3.61
562	30	16	400	80	8	480	600	400	200	11.59	11.24	46.46	4.01	4.13
586	30	16	800	40	8	240	600	400	200	11.85	11.88	49.11	4.14	4.13
634	30	16	800	80	8	480	600	400	200	5.97	5.94	23.96	4.01	4.03
658	30	16	1200	40	8	240	600	400	200	8.08	8.07	33.07	4.09	4.10
682	30	16	1200	60	8	360	600	400	200	5.39	5.38	21.6	4.01	4.01
706	30	16	1200	80	8	480	600	400	200	4.06	4.04	16.15	3.98	4.00
544	30	24	400	40	12	240	600	400	200	22.12	21.34	91.71	4.15	4.30
568	30	24	400	80	12	480	600	400	200	11.33	10.67	45.7	4.03	4.28
592	30	24	800	40	12	240	600	400	200	11.6	11.55	49.39	4.26	4.28
616	30	24	800	60	12	360	600	400	200	7.77	7.70	32.36	4.16	4.20
640	30	24	800	80	12	480	600	400	200	5.87	5.78	23.86	4.06	4.13
664	30	24	1200	40	12	240	600	400	200	7.98	7.92	33.37	4.18	4.21
688	30	24	1200	60	12	360	600	400	200	5.33	5.28	21.87	4.1	4.14
712	30	24	1200	80	12	480	600	400	200	4.02	3.96	16.13	4.01	4.07
22	30	8	100	10	4	60	200	100	50	345.79	324.92	924.84	2.67	2.85
40	30	8	100	20	4	120	200	100	50	176.74	162.46	523.75	2.96	3.22
58	30	8	200	10	4	60	200	100	50	182.11	179.87	502.69	2.76	2.79
76	30	8	200	20	4	120	200	100	50	91.73	89.93	276.22	3.01	3.07
94	30	8	200	40	4	240	200	100	50	47.45	44.97	137.31	2.89	3.05
112	30	8	400	20	4	120	200	100	50	47.89	47.51	141.66	2.96	2.98
130	30	8	400	40	4	240	200	100	50	24.45	23.76	69.75	2.85	2.94
148	30	8	400	60	4	360	200	100	50	16.54	15.84	45.97	2.78	2.90
166	30	8	600	20	4	120	200	100	50	32.52	32.28	95.91	2.95	2.97
184	30	8	600	40	4	240	200	100	50	16.49	16.14	47.11	2.86	2.92
202	30	8	600	60	4	360	200	100	50	11.1	10.76	31.02	2.79	2.88
250	30	16	200	20	8	120	300	200	100	85.57	81.23	294.25	3.44	3.62

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
274	30	16	200	40	8	240	300	200	100	44.21	40.61	150.8	3.41	3.71
298	30	16	400	20	8	120	300	200	100	45.42	44.97	160.05	3.52	3.56
322	30	16	400	40	8	240	300	200	100	23.22	22.48	79.56	3.43	3.54
346	30	16	400	60	8	360	300	200	100	15.84	14.99	51.95	3.28	3.47
370	30	16	800	40	8	240	300	200	100	12.16	11.88	40.79	3.35	3.43
394	30	16	800	60	8	360	300	200	100	8.19	7.92	26.68	3.26	3.37
418	30	16	800	80	8	480	300	200	100	6.2	5.94	19.81	3.2	3.34
442	30	16	1200	40	8	240	300	200	100	8.19	8.07	27.6	3.37	3.42
466	30	16	1200	60	8	360	300	200	100	5.5	5.38	18.01	3.27	3.35
550	30	32	400	40	16	240	600	400	200	21.59	20.31	91.6	4.24	4.51
574	30	32	400	80	16	480	600	400	200	11.06	10.15	45.58	4.12	4.49
598	30	32	800	40	16	240	600	400	200	11.38	11.24	50.35	4.42	4.48
622	30	32	800	60	16	360	600	400	200	7.66	7.49	32.58	4.26	4.35
646	30	32	800	80	16	480	600	400	200	5.79	5.62	24.19	4.18	4.30
670	30	32	1200	40	16	240	600	400	200	7.87	7.77	34.07	4.33	4.38
694	30	32	1200	60	16	360	600	400	200	5.27	5.18	22.05	4.18	4.26
718	30	32	1200	80	16	480	600	400	200	3.99	3.89	16.3	4.09	4.19
280	30	24	200	40	12	240	300	200	100	42.69	37.03	150.35	3.52	4.06
328	30	24	400	40	12	240	300	200	100	22.67	21.34	80.22	3.54	3.76
352	30	24	400	60	12	360	300	200	100	15.55	14.23	52.38	3.37	3.68
376	30	24	800	40	12	240	300	200	100	12.01	11.55	41.27	3.44	3.57
400	30	24	800	60	12	360	300	200	100	8.11	7.70	26.84	3.31	3.49
424	30	24	800	80	12	480	300	200	100	6.15	5.78	19.87	3.23	3.44
448	30	24	1200	40	12	240	300	200	100	8.14	7.92	28.03	3.44	3.54
472	30	24	1200	60	12	360	300	200	100	5.47	5.28	18.16	3.32	3.44
46	30	16	100	20	8	120	200	100	50	163.77	136.11	524.2	3.2	3.85
82	30	16	200	20	8	120	200	100	50	86.36	81.23	284.09	3.29	3.50
100	30	16	200	40	8	240	200	100	50	45.83	40.61	137.58	3	3.39
118	30	16	400	20	8	120	200	100	50	46.55	44.97	145.47	3.12	3.24
136	30	16	400	40	8	240	200	100	50	23.93	22.48	70.71	2.95	3.15
154	30	16	400	60	8	360	200	100	50	16.31	14.99	46.23	2.83	3.08
172	30	16	600	20	8	120	200	100	50	32.01	31.09	99.05	3.09	3.19
190	30	16	600	40	8	240	200	100	50	16.28	15.54	47.76	2.93	3.07
208	30	16	600	60	8	360	200	100	50	11.01	10.36	31.22	2.84	3.01
286	30	32	200	40	16	240	300	200	100	40.65	34.03	150.39	3.7	4.42
334	30	32	400	40	16	240	300	200	100	21.93	20.31	81.34	3.71	4.01
358	30	32	400	60	16	360	300	200	100	15.17	13.54	52.62	3.47	3.89
382	30	32	800	40	16	240	300	200	100	11.85	11.24	41.91	3.54	3.73
406	30	32	800	60	16	360	300	200	100	8.02	7.49	26.97	3.36	3.60
430	30	32	800	80	16	480	300	200	100	6.09	5.62	19.92	3.27	3.54
454	30	32	1200	40	16	240	300	200	100	8.08	7.77	28.56	3.53	3.67
478	30	32	1200	60	16	360	300	200	100	5.44	5.18	18.35	3.38	3.54
106	30	20	200	40	10	240	200	100	50	44.75	38.74	138.36	3.09	3.57
142	30	20	400	40	10	240	200	100	50	23.73	21.90	70.86	2.99	3.24
160	30	20	400	60	10	360	200	100	50	16.22	14.60	46.33	2.86	3.17
196	30	20	600	40	10	240	200	100	50	16.18	15.26	48.01	2.97	3.15

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
214	30	20	600	60	10	360	200	100	50	10.96	10.17	31.26	2.85	3.07
742	45	8	100	10	4	60	200	70.71	50	349.01	324.92	1151.21	3.3	3.54
760	45	8	100	20	4	120	200	70.71	50	176.32	162.46	637.37	3.61	3.92
766	45	16	100	20	8	120	200	70.71	50	164.24	136.11	678.5	4.13	4.98
796	45	8	200	20	4	120	200	70.71	50	92.06	89.93	333.34	3.62	3.71
802	45	16	200	20	8	120	200	70.71	50	86.99	81.23	356.3	4.1	4.39
814	45	8	200	40	4	240	200	70.71	50	47.75	44.97	165.44	3.46	3.68
820	45	16	200	40	8	240	200	70.71	50	45.66	40.61	174.66	3.83	4.30
826	45	20	200	40	10	240	200	70.71	50	44.79	38.74	176.58	3.94	4.56
832	45	8	400	20	4	120	200	70.71	50	48.05	47.51	171.18	3.56	3.60
838	45	16	400	20	8	120	200	70.71	50	46.78	44.97	182.69	3.91	4.06
850	45	8	400	40	4	240	200	70.71	50	24.53	23.76	84.28	3.44	3.55
856	45	16	400	40	8	240	200	70.71	50	24.02	22.48	89.42	3.72	3.98
862	45	20	400	40	10	240	200	70.71	50	23.83	21.90	90.61	3.8	4.14
868	45	8	400	60	4	360	200	70.71	50	16.56	15.84	56.01	3.38	3.54
874	45	16	400	60	8	360	200	70.71	50	16.35	14.99	58.75	3.59	3.92
880	45	20	400	60	10	360	200	70.71	50	16.26	14.60	59.08	3.63	4.05
886	45	8	600	20	4	120	200	70.71	50	32.53	32.28	115.87	3.56	3.59
892	45	16	600	20	8	120	200	70.71	50	32.05	31.09	124.87	3.9	4.02
904	45	8	600	40	4	240	200	70.71	50	16.51	16.14	56.9	3.45	3.53
910	45	16	600	40	8	240	200	70.71	50	16.29	15.54	60.64	3.72	3.90
916	45	20	600	40	10	240	200	70.71	50	16.22	15.26	61.44	3.79	4.03
922	45	8	600	60	4	360	200	70.71	50	11.11	10.76	37.79	3.4	3.51
928	45	16	600	60	8	360	200	70.71	50	11.02	10.36	39.55	3.59	3.82
934	45	20	600	60	10	360	200	70.71	50	10.97	10.17	39.95	3.64	3.93
964	45	8	200	20	4	120	300	141.42	100	92.65	89.93	352.71	3.81	3.92
970	45	16	200	20	8	120	300	141.42	100	88.45	81.23	367.67	4.16	4.53
988	45	8	200	40	4	240	300	141.42	100	46.52	44.97	178.37	3.83	3.97
994	45	16	200	40	8	240	300	141.42	100	44.76	40.61	185.51	4.14	4.57
1000	45	24	200	40	12	240	300	141.42	100	43.21	37.03	191.04	4.42	5.16
1018	45	16	400	20	8	120	300	141.42	100	46.05	44.97	194.74	4.23	4.33
1036	45	8	400	40	4	240	300	141.42	100	23.98	23.76	91.59	3.82	3.86
1042	45	16	400	40	8	240	300	141.42	100	23.28	22.48	96.82	4.16	4.31
1048	45	24	400	40	12	240	300	141.42	100	22.66	21.34	99.83	4.41	4.68
1054	45	32	400	40	16	240	300	141.42	100	22.11	20.31	102.78	4.65	5.06
1060	45	8	400	60	4	360	300	141.42	100	16.26	15.84	60.82	3.74	3.84
1066	45	16	400	60	8	360	300	141.42	100	15.87	14.99	63.85	4.02	4.26
1072	45	24	400	60	12	360	300	141.42	100	15.55	14.23	65.81	4.23	4.63
1078	45	32	400	60	16	360	300	141.42	100	15.27	13.54	67.18	4.4	4.96
1084	45	8	800	40	4	240	300	141.42	100	12.3	12.22	46.84	3.81	3.83
1090	45	16	800	40	8	240	300	141.42	100	12.14	11.88	49.62	4.09	4.18
1096	45	24	800	40	12	240	300	141.42	100	12.03	11.55	51.55	4.29	4.46
1102	45	32	800	40	16	240	300	141.42	100	11.89	11.24	53.13	4.47	4.73
1108	45	8	800	60	4	360	300	141.42	100	8.28	8.15	31.02	3.75	3.81
1114	45	16	800	60	8	360	300	141.42	100	8.19	7.92	32.72	4	4.13
1120	45	24	800	60	12	360	300	141.42	100	8.12	7.70	33.85	4.17	4.40

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
1126	45	32	800	60	16	360	300	141.42	100	8.04	7.49	34.58	4.3	4.61
1132	45	8	800	80	4	480	300	141.42	100	6.25	6.11	23.03	3.69	3.77
1138	45	16	800	80	8	480	300	141.42	100	6.2	5.94	24.32	3.92	4.09
1150	45	32	800	80	16	480	300	141.42	100	6.11	5.62	25.7	4.21	4.57
1156	45	8	1200	40	4	240	300	141.42	100	8.25	8.23	31.44	3.81	3.82
1162	45	16	1200	40	8	240	300	141.42	100	8.19	8.07	33.49	4.09	4.15
1168	45	24	1200	40	12	240	300	141.42	100	8.12	7.92	34.85	4.29	4.40
1174	45	32	1200	40	16	240	300	141.42	100	8.09	7.77	35.92	4.44	4.62
1180	45	8	1200	60	4	360	300	141.42	100	5.54	5.49	20.9	3.77	3.81
1186	45	16	1200	60	8	360	300	141.42	100	5.5	5.38	22.04	4	4.10
1192	45	24	1200	60	12	360	300	141.42	100	5.46	5.28	22.83	4.18	4.32
1198	45	32	1200	60	16	360	300	141.42	100	5.44	5.18	23.46	4.31	4.53
1234	45	16	400	20	8	120	600	282.84	200	45.66	44.97	199.41	4.37	4.43
1252	45	8	400	40	4	240	600	282.84	200	23.86	23.76	103.87	4.35	4.37
1258	45	16	400	40	8	240	600	282.84	200	23.14	22.48	109.03	4.71	4.85
1264	45	24	400	40	12	240	600	282.84	200	22.42	21.34	111.68	4.98	5.23
1270	45	32	400	40	16	240	600	282.84	200	21.92	20.31	113.63	5.18	5.60
1276	45	8	400	80	4	480	600	282.84	200	11.95	11.88	52.69	4.41	4.44
1282	45	16	400	80	8	480	600	282.84	200	11.56	11.24	54.73	4.73	4.87
1306	45	16	800	40	8	240	600	282.84	200	11.9	11.88	56.52	4.75	4.76
1312	45	24	800	40	12	240	600	282.84	200	11.69	11.55	58.56	5.01	5.07
1318	45	32	800	40	16	240	600	282.84	200	11.51	11.24	60.14	5.22	5.35
1324	45	8	800	60	4	360	600	282.84	200	8.08	8.15	35.46	4.39	4.35
1330	45	16	800	60	8	360	600	282.84	200	7.92	7.92	37.75	4.77	4.77
1336	45	24	800	60	12	360	600	282.84	200	7.79	7.70	38.8	4.98	5.04
1342	45	32	800	60	16	360	600	282.84	200	7.7	7.49	39.9	5.18	5.32
1348	45	8	800	80	4	480	600	282.84	200	6.11	6.11	26.74	4.38	4.38
1354	45	16	800	80	8	480	600	282.84	200	5.99	5.94	28.08	4.69	4.73
1366	45	32	800	80	16	480	600	282.84	200	5.82	5.62	29.66	5.1	5.28
1372	45	8	1200	40	4	240	600	282.84	200	8.19	8.23	35.85	4.38	4.36
1378	45	16	1200	40	8	240	600	282.84	200	8.1	8.07	38.23	4.72	4.74
1384	45	24	1200	40	12	240	600	282.84	200	8.02	7.92	39.7	4.95	5.01
1390	45	32	1200	40	16	240	600	282.84	200	7.93	7.77	40.8	5.15	5.25
1396	45	8	1200	60	4	360	600	282.84	200	5.46	5.49	23.91	4.38	4.36
1402	45	16	1200	60	8	360	600	282.84	200	5.39	5.38	25.54	4.74	4.75
1408	45	24	1200	60	12	360	600	282.84	200	5.34	5.28	26.33	4.93	4.99
1414	45	32	1200	60	16	360	600	282.84	200	5.29	5.18	27.11	5.12	5.23
1420	45	8	1200	80	4	480	600	282.84	200	4.12	4.11	18.01	4.37	4.38
1426	45	16	1200	80	8	480	600	282.84	200	4.07	4.04	18.96	4.66	4.70
1432	45	24	1200	80	12	480	600	282.84	200	4.03	3.96	19.58	4.86	4.95
1438	45	32	1200	80	16	480	600	282.84	200	3.99	3.89	20.1	5.03	5.17
1462	60	8	100	10	4	60	200	57.74	50	351.81	324.92	1304.4	3.71	4.01
1480	60	8	100	20	4	120	200	57.74	50	177.28	162.46	726.82	4.1	4.47
1486	60	16	100	20	8	120	200	57.74	50	169.26	136.11	781.17	4.62	5.74
1516	60	8	200	20	4	120	200	57.74	50	92.5	89.93	377.46	4.08	4.20
1522	60	16	200	20	8	120	200	57.74	50	87.93	81.23	409.64	4.66	5.04

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
1534	60	8	200	40	4	240	200	57.74	50	47.87	44.97	186.94	3.91	4.16
1540	60	16	200	40	8	240	200	57.74	50	46.13	40.61	204.06	4.42	5.02
1546	60	20	200	40	10	240	200	57.74	50	45.36	38.74	205.79	4.54	5.31
1552	60	8	400	20	4	120	200	57.74	50	48.28	47.51	194.64	4.03	4.10
1558	60	16	400	20	8	120	200	57.74	50	47.2	44.97	213.39	4.52	4.75
1570	60	8	400	40	4	240	200	57.74	50	24.64	23.76	95.41	3.87	4.02
1576	60	16	400	40	8	240	200	57.74	50	24.2	22.48	104.39	4.31	4.64
1582	60	20	400	40	10	240	200	57.74	50	23.98	21.90	105.82	4.41	4.83
1606	60	8	600	20	4	120	200	57.74	50	32.59	32.28	131.65	4.04	4.08
1612	60	16	600	20	8	120	200	57.74	50	32.15	31.09	144.39	4.49	4.64
1624	60	8	600	40	4	240	200	57.74	50	16.53	16.14	64.45	3.9	3.99
1630	60	16	600	40	8	240	200	57.74	50	16.35	15.54	70.86	4.33	4.56
1636	60	20	600	40	10	240	200	57.74	50	16.25	15.26	72.23	4.44	4.73
1684	60	8	200	20	4	120	300	115.47	100	94.03	89.93	387.47	4.12	4.31
1708	60	8	200	40	4	240	300	115.47	100	46.74	44.97	197.69	4.23	4.40
1714	60	16	200	40	8	240	300	115.47	100	44.79	40.61	214.48	4.79	5.28
1720	60	24	200	40	12	240	300	115.47	100	43.62	37.03	224.7	5.15	6.07
1726	60	32	200	40	16	240	300	115.47	100	42.58	34.03	230.85	5.42	6.78
1738	60	16	400	20	8	120	300	115.47	100	46.38	44.97	219.36	4.73	4.88
1756	60	8	400	40	4	240	300	115.47	100	24.07	23.76	101.14	4.2	4.26
1762	60	16	400	40	8	240	300	115.47	100	23.4	22.48	110.79	4.73	4.93
1768	60	24	400	40	12	240	300	115.47	100	23.01	21.34	116.91	5.08	5.48
1774	60	32	400	40	16	240	300	115.47	100	22.47	20.31	119.94	5.34	5.91
1780	60	8	400	60	4	360	300	115.47	100	16.34	15.84	66.75	4.09	4.21
1786	60	16	400	60	8	360	300	115.47	100	15.96	14.99	73.2	4.59	4.88
1792	60	24	400	60	12	360	300	115.47	100	15.73	14.23	77.77	4.94	5.47
1798	60	32	400	60	16	360	300	115.47	100	15.44	13.54	79.65	5.16	5.88
1804	60	8	800	40	4	240	300	115.47	100	12.31	12.22	51.75	4.2	4.23
1810	60	16	800	40	8	240	300	115.47	100	12.17	11.88	57.05	4.69	4.80
1816	60	24	800	40	12	240	300	115.47	100	12.05	11.55	60.43	5.02	5.23
1822	60	32	800	40	16	240	300	115.47	100	11.95	11.24	62.2	5.21	5.53
1828	60	8	800	60	4	360	300	115.47	100	8.28	8.15	34.07	4.11	4.18
1834	60	16	800	60	8	360	300	115.47	100	8.21	7.92	37.63	4.59	4.75
1840	60	24	800	60	12	360	300	115.47	100	8.13	7.70	40	4.92	5.19
1846	60	32	800	60	16	360	300	115.47	100	8.07	7.49	41.08	5.09	5.48
1852	60	8	800	80	4	480	300	115.47	100	6.25	6.11	25.48	4.08	4.17
1858	60	16	800	80	8	480	300	115.47	100	6.21	5.94	27.89	4.49	4.70
1864	60	24	800	80	12	480	300	115.47	100	6.17	5.78	29.69	4.82	5.14
1876	60	8	1200	40	4	240	300	115.47	100	8.25	8.23	34.87	4.23	4.24
1882	60	16	1200	40	8	240	300	115.47	100	8.19	8.07	38.41	4.69	4.76
1888	60	24	1200	40	12	240	300	115.47	100	8.14	7.92	40.78	5.01	5.15
1894	60	32	1200	40	16	240	300	115.47	100	8.09	7.77	42.09	5.2	5.42
1900	60	8	1200	60	4	360	300	115.47	100	5.54	5.49	22.88	4.13	4.17
1906	60	16	1200	60	8	360	300	115.47	100	5.5	5.38	25.31	4.6	4.70
1912	60	24	1200	60	12	360	300	115.47	100	5.47	5.28	26.95	4.93	5.11
1924	60	8	1200	80	4	480	300	115.47	100	4.17	4.11	17.07	4.09	4.15

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
1930	60	16	1200	80	8	480	300	115.47	100	4.15	4.04	18.75	4.52	4.65
1936	60	24	1200	80	12	480	300	115.47	100	4.13	3.96	19.99	4.84	5.05
1948	60	8	400	20	4	120	600	230.94	200	48.02	47.51	210.91	4.39	4.44
1954	60	16	400	20	8	120	600	230.94	200	46.12	44.97	225.21	4.88	5.01
1972	60	8	400	40	4	240	600	230.94	200	23.86	23.76	113.01	4.74	4.76
1978	60	16	400	40	8	240	600	230.94	200	23.27	22.48	122.14	5.25	5.43
1984	60	24	400	40	12	240	600	230.94	200	22.61	21.34	126.88	5.61	5.95
1990	60	32	400	40	16	240	600	230.94	200	22.07	20.31	130.32	5.9	6.42
1996	60	8	400	80	4	480	600	230.94	200	11.97	11.88	56.63	4.73	4.77
2002	60	16	400	80	8	480	600	230.94	200	11.65	11.24	61.03	5.24	5.43
2008	60	24	400	80	12	480	600	230.94	200	11.41	10.67	64.21	5.63	6.02
2014	60	32	400	80	16	480	600	230.94	200	11.13	10.15	66.23	5.95	6.52
2020	60	8	800	40	4	240	600	230.94	200	12.13	12.22	57.9	4.77	4.74
2026	60	16	800	40	8	240	600	230.94	200	11.93	11.88	63.34	5.31	5.33
2032	60	24	800	40	12	240	600	230.94	200	11.73	11.55	66.29	5.65	5.74
2038	60	32	800	40	16	240	600	230.94	200	11.53	11.24	68.61	5.95	6.10
2044	60	8	800	60	4	360	600	230.94	200	8.09	8.15	38.22	4.72	4.69
2050	60	16	800	60	8	360	600	230.94	200	7.93	7.92	41.87	5.28	5.29
2056	60	24	800	60	12	360	600	230.94	200	7.8	7.70	44.3	5.68	5.75
2062	60	32	800	60	16	360	600	230.94	200	7.7	7.49	45.87	5.96	6.12
2068	60	8	800	80	4	480	600	230.94	200	6.12	6.11	28.69	4.69	4.69
2074	60	16	800	80	8	480	600	230.94	200	6	5.94	31.25	5.21	5.26
2080	60	24	800	80	12	480	600	230.94	200	5.91	5.78	33.03	5.59	5.72
2086	60	32	800	80	16	480	600	230.94	200	5.82	5.62	34.26	5.89	6.10
2092	60	8	1200	40	4	240	600	230.94	200	8.19	8.23	39.12	4.78	4.75
2098	60	16	1200	40	8	240	600	230.94	200	8.1	8.07	42.92	5.3	5.32
2104	60	24	1200	40	12	240	600	230.94	200	8.01	7.92	45.02	5.62	5.69
2110	60	32	1200	40	16	240	600	230.94	200	7.94	7.77	46.66	5.88	6.00
2116	60	8	1200	60	4	360	600	230.94	200	5.47	5.49	25.77	4.71	4.70
2122	60	16	1200	60	8	360	600	230.94	200	5.4	5.38	28.32	5.25	5.26
2128	60	24	1200	60	12	360	600	230.94	200	5.36	5.28	30.01	5.6	5.68
2134	60	32	1200	60	16	360	600	230.94	200	5.3	5.18	31.14	5.88	6.01
2140	60	8	1200	80	4	480	600	230.94	200	4.12	4.11	19.33	4.69	4.70
2146	60	16	1200	80	8	480	600	230.94	200	4.07	4.04	21.11	5.18	5.23
2152	60	24	1200	80	12	480	600	230.94	200	4.04	3.96	22.36	5.54	5.65
2158	60	32	1200	80	16	480	600	230.94	200	4	3.89	23.22	5.81	5.98
2182	70	8	100	10	4	60	200	53.21	50	352.88	324.92	1368.25	3.88	4.21
2404	70	8	200	20	4	120	300	106.42	100	93.68	89.93	407.24	4.35	4.53
2410	70	16	200	20	8	120	300	106.42	100	90.31	81.23	440.72	4.88	5.43
2428	70	8	200	40	4	240	300	106.42	100	46.85	44.97	207.84	4.44	4.62
2434	70	16	200	40	8	240	300	106.42	100	45.36	40.61	228.47	5.04	5.63
2440	70	24	200	40	12	240	300	106.42	100	43.62	37.03	239.75	5.5	6.47
2446	70	32	200	40	16	240	300	106.42	100	42.65	34.03	247.44	5.8	7.27
2458	70	16	400	20	8	120	300	106.42	100	46.67	44.97	230.96	4.95	5.14
2476	70	8	400	40	4	240	300	106.42	100	24.18	23.76	105.93	4.38	4.46
2482	70	16	400	40	8	240	300	106.42	100	23.54	22.48	118.79	5.05	5.28

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
2488	70	24	400	40	12	240	300	106.42	100	23.07	21.34	124.61	5.4	5.84
2494	70	32	400	40	16	240	300	106.42	100	22.63	20.31	128.9	5.7	6.35
2500	70	8	400	60	4	360	300	106.42	100	16.39	15.84	70.83	4.32	4.47
2506	70	16	400	60	8	360	300	106.42	100	16.04	14.99	77.69	4.84	5.18
2512	70	24	400	60	12	360	300	106.42	100	15.76	14.23	82.9	5.26	5.83
2518	70	32	400	60	16	360	300	106.42	100	15.49	13.54	86.16	5.56	6.36
2524	70	8	800	40	4	240	300	106.42	100	12.33	12.22	54.21	4.4	4.43
2530	70	16	800	40	8	240	300	106.42	100	12.18	11.88	60.95	5	5.13
2536	70	24	800	40	12	240	300	106.42	100	12.07	11.55	64.92	5.38	5.62
2542	70	32	800	40	16	240	300	106.42	100	11.97	11.24	66.92	5.59	5.95
2548	70	8	800	60	4	360	300	106.42	100	8.3	8.15	36.17	4.36	4.44
2554	70	16	800	60	8	360	300	106.42	100	8.22	7.92	40.03	4.87	5.06
2560	70	24	800	60	12	360	300	106.42	100	8.15	7.70	42.44	5.21	5.51
2566	70	32	800	60	16	360	300	106.42	100	8.09	7.49	44.44	5.49	5.93
2596	70	8	1200	40	4	240	300	106.42	100	8.27	8.23	36.39	4.4	4.42
2602	70	16	1200	40	8	240	300	106.42	100	8.19	8.07	41.02	5.01	5.08
2608	70	24	1200	40	12	240	300	106.42	100	8.14	7.92	43.44	5.33	5.49
2614	70	32	1200	40	16	240	300	106.42	100	8.1	7.77	45.22	5.58	5.82
2620	70	8	1200	60	4	360	300	106.42	100	5.55	5.49	24.27	4.37	4.42
2626	70	16	1200	60	8	360	300	106.42	100	5.5	5.38	26.9	4.89	5.00
2632	70	24	1200	60	12	360	300	106.42	100	5.47	5.28	28.69	5.24	5.43
2638	70	32	1200	60	16	360	300	106.42	100	5.44	5.18	29.75	5.46	5.74
2668	70	8	400	20	4	120	600	212.84	200	48.23	47.51	215.44	4.47	4.53
2674	70	16	400	20	8	120	600	212.84	200	46.34	44.97	236.27	5.1	5.25
2692	70	8	400	40	4	240	600	212.84	200	23.99	23.76	116.64	4.86	4.91
2698	70	16	400	40	8	240	600	212.84	200	23.39	22.48	126.98	5.43	5.65
2704	70	24	400	40	12	240	600	212.84	200	22.83	21.34	133.97	5.87	6.28
2710	70	32	400	40	16	240	600	212.84	200	22.16	20.31	138.18	6.24	6.80
2716	70	8	400	80	4	480	600	212.84	200	12.07	11.88	58.61	4.86	4.93
2722	70	16	400	80	8	480	600	212.84	200	11.71	11.24	64.54	5.51	5.74
2728	70	24	400	80	12	480	600	212.84	200	11.51	10.67	68.72	5.97	6.44
2782	70	32	800	60	16	360	600	212.84	200	7.74	7.49	48.83	6.31	6.52
2788	70	8	800	80	4	480	600	212.84	200	6.14	6.11	29.66	4.83	4.85
2794	70	16	800	80	8	480	600	212.84	200	6.02	5.94	32.94	5.47	5.55
2800	70	24	800	80	12	480	600	212.84	200	5.93	5.78	34.95	5.9	6.05
2806	70	32	800	80	16	480	600	212.84	200	5.85	5.62	36.69	6.27	6.53
2812	70	8	1200	40	4	240	600	212.84	200	8.19	8.23	40.43	4.93	4.91
2818	70	16	1200	40	8	240	600	212.84	200	8.12	8.07	44.76	5.52	5.55
2824	70	24	1200	40	12	240	600	212.84	200	8.04	7.92	47.32	5.89	5.98
2830	70	32	1200	40	16	240	600	212.84	200	7.95	7.77	49.65	6.25	6.39
2842	70	16	1200	60	8	360	600	212.84	200	5.4	5.38	30.15	5.58	5.60
2848	70	24	1200	60	12	360	600	212.84	200	5.35	5.28	31.88	5.95	6.04
2854	70	32	1200	60	16	360	600	212.84	200	5.31	5.18	33.22	6.26	6.41
2860	70	8	1200	80	4	480	600	212.84	200	4.13	4.11	20	4.84	4.86
2866	70	16	1200	80	8	480	600	212.84	200	4.08	4.04	22.26	5.46	5.52
2872	70	24	1200	80	12	480	600	212.84	200	4.04	3.96	23.65	5.86	5.97

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
2878	70	32	1200	80	16	480	600	212.84	200	4	3.89	24.89	6.23	6.41
2182	75	8	100	10	4	60	200	51.76	50	343.59	324.92	1390.16	4.05	4.28
2200	75	8	100	20	4	120	200	51.76	50	177.15	162.46	786.29	4.44	4.84
2206	75	16	100	20	8	120	200	51.76	50	168.84	136.11	853.01	5.05	6.27
2236	75	8	200	20	4	120	200	51.76	50	92.88	89.93	407.03	4.38	4.53
2242	75	16	200	20	8	120	200	51.76	50	88.74	81.23	447.98	5.05	5.51
2254	75	8	200	40	4	240	200	51.76	50	48.1	44.97	201.97	4.2	4.49
2272	75	8	400	20	4	120	200	51.76	50	48.3	47.51	211.6	4.38	4.45
2278	75	16	400	20	8	120	200	51.76	50	47.35	44.97	233.73	4.94	5.20
2290	75	8	400	40	4	240	200	51.76	50	24.67	23.76	102.17	4.14	4.30
2326	75	8	600	20	4	120	200	51.76	50	32.67	32.28	143.17	4.38	4.43
2332	75	16	600	20	8	120	200	51.76	50	32.18	31.09	158.3	4.92	5.09
2344	75	8	600	40	4	240	200	51.76	50	16.57	16.14	69	4.16	4.27
2734	75	32	400	80	16	480	600	207.06	200	11.29	10.15	73.01	6.47	7.19
2740	75	8	800	40	4	240	600	207.06	200	12.17	12.22	61.02	5.01	4.99
2746	75	16	800	40	8	240	600	207.06	200	11.97	11.88	67.11	5.61	5.65
2752	75	24	800	40	12	240	600	207.06	200	11.8	11.55	71.21	6.03	6.16
2758	75	32	800	40	16	240	600	207.06	200	11.64	11.24	75.41	6.48	6.71
2764	75	8	800	60	4	360	600	207.06	200	8.13	8.15	40.55	4.99	4.98
2770	75	16	800	60	8	360	600	207.06	200	7.97	7.92	45.06	5.65	5.69
2776	75	24	800	60	12	360	600	207.06	200	7.86	7.70	48.32	6.14	6.27

E.2 Parametric Study Results for $R/t_w = 6$

Parametric study results for the development of the SCF-ENS equation for R/tw = 6

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
506	30	24	400	20	4	120	250	400	200	47.15	47.51	140.67	2.98	2.96
530	30	24	400	40	4	240	250	400	200	24.25	23.76	71.46	2.95	3.01
554	30	24	400	80	4	480	250	400	200	12.46	11.88	35.72	2.87	3.01
578	30	24	800	40	4	240	250	400	200	12.39	12.22	36.51	2.95	2.99
602	30	24	800	60	4	360	250	400	200	8.33	8.15	24.12	2.90	2.96
626	30	24	800	80	4	480	250	400	200	6.27	6.11	18.11	2.89	2.96
650	30	24	1200	40	4	240	250	400	200	8.31	8.23	24.65	2.96	3.00
674	30	24	1200	60	4	360	250	400	200	5.57	5.49	16.27	2.92	2.97
698	30	24	1200	80	4	480	250	400	200	4.19	4.11	12.22	2.92	2.97
218	30	24	200	10	4	60	150	200	100	179.50	179.87	437.59	2.44	2.43
242	30	24	200	20	4	120	150	200	100	92.71	89.93	243.30	2.62	2.71
266	30	24	200	40	4	240	150	200	100	48.65	44.97	122.70	2.52	2.73
290	30	24	400	20	4	120	150	200	100	48.34	47.51	125.88	2.60	2.65
314	30	24	400	40	4	240	150	200	100	24.77	23.76	62.38	2.52	2.63
338	30	24	400	60	4	360	150	200	100	16.66	15.84	41.51	2.49	2.62
362	30	24	800	40	4	240	150	200	100	12.50	12.22	31.83	2.55	2.60
386	30	24	800	60	4	360	150	200	100	8.37	8.15	21.16	2.53	2.60
434	30	24	1200	40	4	240	150	200	100	8.33	8.23	21.40	2.57	2.60
458	30	24	1200	60	4	360	150	200	100	5.57	5.49	14.16	2.54	2.58
512	30	48	400	20	8	120	250	400	200	44.76	44.97	135.40	3.03	3.01
536	30	48	400	40	8	240	250	400	200	23.50	22.48	71.33	3.04	3.17
584	30	48	800	40	8	240	250	400	200	12.19	11.88	36.85	3.02	3.10
608	30	48	800	60	8	360	250	400	200	8.25	7.92	24.33	2.95	3.07
632	30	48	800	80	8	480	250	400	200	6.23	5.94	18.12	2.91	3.05
656	30	48	1200	40	8	240	250	400	200	8.24	8.07	24.92	3.02	3.09
680	30	48	1200	60	8	360	250	400	200	5.54	5.38	16.44	2.97	3.06
704	30	48	1200	80	8	480	250	400	200	4.17	4.04	12.24	2.93	3.03
542	30	72	400	40	12	240	250	400	200	22.76	21.34	70.99	3.12	3.33
566	30	72	400	80	12	480	250	400	200	12.15	10.67	36.06	2.97	3.38
590	30	72	800	40	12	240	250	400	200	12.03	11.55	37.10	3.08	3.21
614	30	72	800	60	12	360	250	400	200	8.18	7.70	24.57	3.00	3.19
638	30	72	800	80	12	480	250	400	200	6.20	5.78	18.36	2.96	3.18
662	30	72	1200	40	12	240	250	400	200	8.18	7.92	25.18	3.08	3.18
686	30	72	1200	60	12	360	250	400	200	5.52	5.28	16.65	3.02	3.15
710	30	72	1200	80	12	480	250	400	200	4.16	3.96	12.43	2.99	3.14
20	30	24	100	10	4	60	80	100	50	346.83	324.92	807.50	2.33	2.49
38	30	24	100	20	4	120	80	100	50	187.07	162.46	421.57	2.25	2.59
56	30	24	200	10	4	60	80	100	50	186.04	179.87	422.17	2.27	2.35
74	30	24	200	20	4	120	80	100	50	97.13	89.93	215.83	2.22	2.40
92	30	24	200	40	4	240	80	100	50	49.82	44.97	106.56	2.14	2.37
110	30	24	400	20	4	120	80	100	50	49.57	47.51	110.71	2.23	2.33
128	30	24	400	40	4	240	80	100	50	25.11	23.76	54.40	2.17	2.29
146	30	24	400	60	4	360	80	100	50	16.76	15.84	36.28	2.16	2.29
164	30	24	600	20	4	120	80	100	50	33.20	32.28	74.51	2.24	2.31

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
182	30	24	600	40	4	240	80	100	50	16.75	16.14	36.52	2.18	2.26
200	30	24	600	60	4	360	80	100	50	11.17	10.76	24.35	2.18	2.26
248	30	48	200	20	8	120	150	200	100	86.12	81.23	244.26	2.84	3.01
272	30	48	200	40	8	240	150	200	100	46.92	40.61	124.35	2.65	3.06
296	30	48	400	20	8	120	150	200	100	46.67	44.97	128.00	2.74	2.85
320	30	48	400	40	8	240	150	200	100	24.34	22.48	63.78	2.62	2.84
344	30	48	400	60	8	360	150	200	100	16.49	14.99	42.16	2.56	2.81
368	30	48	800	40	8	240	150	200	100	12.42	11.88	32.70	2.63	2.75
392	30	48	800	60	8	360	150	200	100	8.34	7.92	21.55	2.58	2.72
416	30	48	800	80	8	480	150	200	100	6.28	5.94	16.01	2.55	2.70
440	30	48	1200	40	8	240	150	200	100	8.31	8.07	22.08	2.66	2.74
464	30	48	1200	60	8	360	150	200	100	5.56	5.38	14.54	2.62	2.70
548	30	96	400	40	16	240	250	400	200	21.85	20.31	71.72	3.28	3.53
572	30	96	400	80	16	480	250	400	200	11.91	10.15	36.19	3.04	3.56
596	30	96	800	40	16	240	250	400	200	11.81	11.24	37.74	3.19	3.36
620	30	96	800	60	16	360	250	400	200	8.12	7.49	24.96	3.08	3.33
644	30	96	800	80	16	480	250	400	200	6.15	5.62	18.50	3.01	3.29
668	30	96	1200	40	16	240	250	400	200	8.11	7.77	25.68	3.17	3.30
692	30	96	1200	60	16	360	250	400	200	5.49	5.18	16.94	3.09	3.27
716	30	96	1200	80	16	480	250	400	200	4.14	3.89	12.53	3.02	3.22
278	30	72	200	40	12	240	150	200	100	45.10	37.03	126.21	2.80	3.41
326	30	72	400	40	12	240	150	200	100	23.96	21.34	64.59	2.70	3.03
350	30	72	400	60	12	360	150	200	100	16.35	14.23	42.59	2.61	2.99
374	30	72	800	40	12	240	150	200	100	12.35	11.55	33.19	2.69	2.87
398	30	72	800	60	12	360	150	200	100	8.32	7.70	21.79	2.62	2.83
422	30	72	800	80	12	480	150	200	100	6.27	5.78	16.27	2.60	2.82
446	30	72	1200	40	12	240	150	200	100	8.28	7.92	22.53	2.72	2.85
470	30	72	1200	60	12	360	150	200	100	5.56	5.28	14.75	2.65	2.79
44	30	48	100	20	8	120	80	100	50	171.97	136.11	437.78	2.55	3.22
80	30	48	200	20	8	120	80	100	50	93.17	81.23	223.01	2.39	2.75
98	30	48	200	40	8	240	80	100	50	48.89	40.61	109.56	2.24	2.70
116	30	48	400	20	8	120	80	100	50	48.80	44.97	114.59	2.35	2.55
134	30	48	400	40	8	240	80	100	50	24.92	22.48	55.89	2.24	2.49
152	30	48	400	60	8	360	80	100	50	16.69	14.99	37.37	2.24	2.49
170	30	48	600	20	8	120	80	100	50	32.91	31.09	77.87	2.37	2.50
188	30	48	600	40	8	240	80	100	50	16.68	15.54	37.87	2.27	2.44
206	30	48	600	60	8	360	80	100	50	11.14	10.36	25.17	2.26	2.43
284	30	96	200	40	16	240	150	200	100	43.07	34.03	128.85	2.99	3.79
332	30	96	400	40	16	240	150	200	100	23.42	20.31	65.53	2.80	3.23
356	30	96	400	60	16	360	150	200	100	16.09	13.54	43.08	2.68	3.18
380	30	96	800	40	16	240	150	200	100	12.26	11.24	33.74	2.75	3.00
404	30	96	800	60	16	360	150	200	100	8.27	7.49	22.06	2.67	2.94
428	30	96	800	80	16	480	150	200	100	6.24	5.62	16.48	2.64	2.93
452	30	96	1200	40	16	240	150	200	100	8.24	7.77	23.12	2.81	2.97
476	30	96	1200	60	16	360	150	200	100	5.54	5.18	15.02	2.71	2.90
740	45	24	100	10	4	60	80	70.71	50	354.73	324.92	961.30	2.71	2.96

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
758	45	24	100	20	4	120	80	70.71	50	187.45	162.46	488.89	2.61	3.01
794	45	24	200	20	4	120	80	70.71	50	96.92	89.93	249.67	2.58	2.78
812	45	24	200	40	4	240	80	70.71	50	49.73	44.97	124.46	2.50	2.77
830	45	24	400	20	4	120	80	70.71	50	49.55	47.51	127.56	2.57	2.68
848	45	24	400	40	4	240	80	70.71	50	25.08	23.76	63.37	2.53	2.67
884	45	24	600	20	4	120	80	70.71	50	33.27	32.28	85.47	2.57	2.65
902	45	24	600	40	4	240	80	70.71	50	16.76	16.14	42.45	2.53	2.63
938	45	24	200	10	4	60	150	141.4	100	181.86	179.87	523.81	2.88	2.91
962	45	24	200	20	4	120	150	141.4	100	92.73	89.93	275.50	2.97	3.06
968	45	48	200	20	8	120	150	141.4	100	89.29	81.23	292.47	3.28	3.60
986	45	24	200	40	4	240	150	141.4	100	48.12	44.97	136.84	2.84	3.04
992	45	48	200	40	8	240	150	141.4	100	47.16	40.61	145.99	3.10	3.59
998	45	72	200	40	12	240	150	141.4	100	45.63	37.03	152.91	3.35	4.13
1016	45	48	400	20	8	120	150	141.4	100	47.21	44.97	149.73	3.17	3.33
1034	45	24	400	40	4	240	150	141.4	100	24.68	23.76	69.81	2.83	2.94
1040	45	48	400	40	8	240	150	141.4	100	24.33	22.48	74.94	3.08	3.33
1046	45	72	400	40	12	240	150	141.4	100	24.04	21.34	77.62	3.23	3.64
1058	45	24	400	60	4	360	150	141.4	100	16.63	15.84	46.32	2.79	2.92
1064	45	48	400	60	8	360	150	141.4	100	16.48	14.99	49.63	3.01	3.31
1070	45	72	400	60	12	360	150	141.4	100	16.35	14.23	51.78	3.17	3.64
1082	45	24	800	40	4	240	150	141.4	100	12.47	12.22	35.49	2.85	2.90
1088	45	48	800	40	8	240	150	141.4	100	12.42	11.88	38.10	3.07	3.21
1094	45	72	800	40	12	240	150	141.4	100	12.37	11.55	39.87	3.22	3.45
1106	45	24	800	60	4	360	150	141.4	100	8.36	8.15	23.52	2.81	2.89
1112	45	48	800	60	8	360	150	141.4	100	8.35	7.92	25.20	3.02	3.18
1118	45	72	800	60	12	360	150	141.4	100	8.32	7.70	26.50	3.18	3.44
1130	45	24	800	80	4	480	150	141.4	100	6.28	6.11	17.65	2.81	2.89
1136	45	48	800	80	8	480	150	141.4	100	6.27	5.94	18.96	3.02	3.19
1154	45	24	1200	40	4	240	150	141.4	100	8.31	8.23	23.64	2.85	2.87
1160	45	48	1200	40	8	240	150	141.4	100	8.31	8.07	25.59	3.08	3.17
1166	45	72	1200	40	12	240	150	141.4	100	8.27	7.92	26.88	3.25	3.39
1184	45	48	1200	60	8	360	150	141.4	100	5.57	5.38	16.91	3.04	3.14
1190	45	72	1200	60	12	360	150	141.4	100	5.55	5.28	17.85	3.22	3.38
1226	45	24	400	20	4	120	250	282.8	200	47.28	47.51	151.54	3.21	3.19
1232	45	48	400	20	8	120	250	282.8	200	45.42	44.97	158.54	3.49	3.53
1250	45	24	400	40	4	240	250	282.8	200	24.08	23.76	76.01	3.16	3.20
1256	45	48	400	40	8	240	250	282.8	200	23.37	22.48	80.87	3.46	3.60
1262	45	72	400	40	12	240	250	282.8	200	22.84	21.34	83.57	3.66	3.92
1268	45	96	400	40	16	240	250	282.8	200	22.26	20.31	85.94	3.86	4.23
1274	45	24	400	80	4	480	250	282.8	200	12.35	11.88	38.04	3.08	3.20
1280	45	48	400	80	8	480	250	282.8	200	12.18	11.24	40.30	3.31	3.58
1286	45	72	400	80	12	480	250	282.8	200	11.98	10.67	41.97	3.50	3.93
1304	45	48	800	40	8	240	250	282.8	200	12.19	11.88	41.49	3.40	3.49
1310	45	72	800	40	12	240	250	282.8	200	12.03	11.55	42.89	3.56	3.71
1316	45	96	800	40	16	240	250	282.8	200	11.95	11.24	44.01	3.68	3.91
1322	45	24	800	60	4	360	250	282.8	200	8.29	8.15	25.78	3.11	3.16

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
1328	45	48	800	60	8	360	250	282.8	200	8.21	7.92	27.50	3.35	3.47
1334	45	72	800	60	12	360	250	282.8	200	8.14	7.70	28.61	3.51	3.72
1346	45	24	800	80	4	480	250	282.8	200	6.25	6.11	19.37	3.10	3.17
1352	45	48	800	80	8	480	250	282.8	200	6.21	5.94	20.62	3.32	3.47
1358	45	72	800	80	12	480	250	282.8	200	6.16	5.78	21.44	3.48	3.71
1370	45	24	1200	40	4	240	250	282.8	200	8.28	8.23	26.14	3.16	3.18
1376	45	48	1200	40	8	240	250	282.8	200	8.22	8.07	27.99	3.40	3.47
1382	45	72	1200	40	12	240	250	282.8	200	8.19	7.92	29.03	3.55	3.67
1388	45	96	1200	40	16	240	250	282.8	200	8.15	7.77	29.87	3.66	3.84
1394	45	24	1200	60	4	360	250	282.8	200	5.55	5.49	17.35	3.13	3.16
1400	45	48	1200	60	8	360	250	282.8	200	5.52	5.38	18.56	3.36	3.45
1406	45	72	1200	60	12	360	250	282.8	200	5.50	5.28	19.34	3.52	3.66
1418	45	24	1200	80	4	480	250	282.8	200	4.18	4.11	13.04	3.12	3.17
1424	45	48	1200	80	8	480	250	282.8	200	4.16	4.04	13.90	3.34	3.44
1430	45	72	1200	80	12	480	250	282.8	200	4.15	3.96	14.47	3.49	3.65
1460	60	24	100	10	4	60	80	57.74	50	359.75	324.92	1053.65	2.93	3.24
1478	60	24	100	20	4	120	80	57.74	50	187.71	162.46	539.44	2.87	3.32
1496	60	24	200	10	4	60	80	57.74	50	191.04	179.87	545.37	2.85	3.03
1514	60	24	200	20	4	120	80	57.74	50	97.44	89.93	274.36	2.82	3.05
1550	60	24	400	20	4	120	80	57.74	50	49.54	47.51	140.02	2.83	2.95
1604	60	24	600	20	4	120	80	57.74	50	33.16	32.28	93.34	2.82	2.89
1658	60	24	200	10	4	60	150	115.5	100	186.07	179.87	563.86	3.03	3.13
1682	60	24	200	20	4	120	150	115.5	100	94.07	89.93	292.44	3.11	3.25
1688	60	48	200	20	8	120	150	115.5	100	90.52	81.23	320.11	3.54	3.94
1706	60	24	200	40	4	240	150	115.5	100	48.33	44.97	146.50	3.03	3.26
1712	60	48	200	40	8	240	150	115.5	100	47.08	40.61	162.21	3.45	3.99
1730	60	24	400	20	4	120	150	115.5	100	48.57	47.51	150.78	3.10	3.17
1736	60	48	400	20	8	120	150	115.5	100	47.81	44.97	164.46	3.44	3.66
1754	60	24	400	40	4	240	150	115.5	100	24.74	23.76	74.90	3.03	3.15
1760	60	48	400	40	8	240	150	115.5	100	24.49	22.48	82.88	3.38	3.69
1778	60	24	400	60	4	360	150	115.5	100	16.66	15.84	49.81	2.99	3.15
1802	60	24	800	40	4	240	150	115.5	100	12.45	12.22	38.04	3.06	3.11
1808	60	48	800	40	8	240	150	115.5	100	12.42	11.88	42.41	3.42	3.57
1826	60	24	800	60	4	360	150	115.5	100	8.35	8.15	25.27	3.03	3.10
1874	60	24	1200	40	4	240	150	115.5	100	8.31	8.23	25.33	3.05	3.08
1880	60	48	1200	40	8	240	150	115.5	100	8.30	8.07	28.25	3.40	3.50
1898	60	24	1200	60	4	360	150	115.5	100	5.56	5.49	16.81	3.02	3.06
1946	60	24	400	20	4	120	250	230.9	200	47.86	47.51	158.90	3.32	3.34
1952	60	48	400	20	8	120	250	230.9	200	46.43	44.97	171.98	3.70	3.82
1970	60	24	400	40	4	240	250	230.9	200	24.18	23.76	79.66	3.29	3.35
1976	60	48	400	40	8	240	250	230.9	200	23.59	22.48	87.08	3.69	3.87
1982	60	72	400	40	12	240	250	230.9	200	23.06	21.34	91.87	3.98	4.31
1994	60	24	400	80	4	480	250	230.9	200	12.39	11.88	39.44	3.18	3.32
2000	60	48	400	80	8	480	250	230.9	200	12.22	11.24	43.47	3.56	3.87
2006	60	72	400	80	12	480	250	230.9	200	12.04	10.67	46.22	3.84	4.33
2012	60	96	400	80	16	480	250	230.9	200	11.90	10.15	48.45	4.07	4.77

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
2018	60	24	800	40	4	240	250	230.9	200	12.34	12.22	40.80	3.31	3.34
2024	60	48	800	40	8	240	250	230.9	200	12.20	11.88	44.76	3.67	3.77
2030	60	72	800	40	12	240	250	230.9	200	12.11	11.55	47.47	3.92	4.11
2042	60	24	800	60	4	360	250	230.9	200	8.30	8.15	27.00	3.25	3.31
2048	60	48	800	60	8	360	250	230.9	200	8.22	7.92	29.68	3.61	3.75
2054	60	72	800	60	12	360	250	230.9	200	8.17	7.70	31.56	3.86	4.10
2060	60	96	800	60	16	360	250	230.9	200	8.12	7.49	33.01	4.07	4.40
2066	60	24	800	80	4	480	250	230.9	200	6.25	6.11	20.12	3.22	3.29
2072	60	48	800	80	8	480	250	230.9	200	6.21	5.94	22.28	3.58	3.75
2078	60	72	800	80	12	480	250	230.9	200	6.19	5.78	23.65	3.82	4.09
2084	60	96	800	80	16	480	250	230.9	200	6.15	5.62	24.83	4.04	4.42
2090	60	24	1200	40	4	240	250	230.9	200	8.28	8.23	27.45	3.32	3.34
2096	60	48	1200	40	8	240	250	230.9	200	8.22	8.07	30.19	3.67	3.74
2102	60	72	1200	40	12	240	250	230.9	200	8.19	7.92	32.09	3.92	4.05
2114	60	24	1200	60	4	360	250	230.9	200	5.54	5.49	18.15	3.28	3.31
2120	60	48	1200	60	8	360	250	230.9	200	5.51	5.38	19.99	3.63	3.72
2126	60	72	1200	60	12	360	250	230.9	200	5.49	5.28	21.30	3.88	4.03
2132	60	96	1200	60	16	360	250	230.9	200	5.49	5.18	22.32	4.07	4.31
2138	60	24	1200	80	4	480	250	230.9	200	4.17	4.11	13.52	3.24	3.29
2144	60	48	1200	80	8	480	250	230.9	200	4.16	4.04	15.00	3.61	3.72
2150	60	72	1200	80	12	480	250	230.9	200	4.14	3.96	15.95	3.85	4.03
2156	60	96	1200	80	16	480	250	230.9	200	4.14	3.89	16.78	4.05	4.32
2180	70	24	100	10	4	60	80	53.21	50	367.35	324.92	1075.65	2.93	3.31
2402	70	24	200	20	4	120	150	106.4	100	94.44	89.93	297.55	3.15	3.31
2426	70	24	200	40	4	240	150	106.4	100	48.68	44.97	149.44	3.07	3.32
2450	70	24	400	20	4	120	150	106.4	100	48.66	47.51	153.70	3.16	3.23
2474	70	24	400	40	4	240	150	106.4	100	24.78	23.76	76.47	3.09	3.22
2522	70	24	800	40	4	240	150	106.4	100	12.45	12.22	38.85	3.12	3.18
2594	70	24	1200	40	4	240	150	106.4	100	8.34	8.23	25.77	3.09	3.13
2666	70	24	400	20	4	120	250	212.8	200	48.20	47.51	160.37	3.33	3.38
2690	70	24	400	40	4	240	250	212.8	200	24.31	23.76	80.37	3.31	3.38
2696	70	48	400	40	8	240	250	212.8	200	23.80	22.48	89.08	3.74	3.96
2702	70	72	400	40	12	240	250	212.8	200	23.43	21.34	94.94	4.05	4.45
2714	70	24	400	80	4	480	250	212.8	200	12.42	11.88	39.89	3.21	3.36
2720	70	48	400	80	8	480	250	212.8	200	12.27	11.24	44.28	3.61	3.94
2786	70	24	800	80	4	480	250	212.8	200	6.26	6.11	20.38	3.25	3.33
2792	70	48	800	80	8	480	250	212.8	200	6.23	5.94	22.82	3.67	3.84
2810	70	24	1200	40	4	240	250	212.8	200	8.27	8.23	27.73	3.35	3.37
2816	70	48	1200	40	8	240	250	212.8	200	8.22	8.07	30.95	3.76	3.83
2822	70	72	1200	40	12	240	250	212.8	200	8.20	7.92	33.17	4.05	4.19
2834	70	24	1200	60	4	360	250	212.8	200	5.54	5.49	18.49	3.34	3.37
2840	70	48	1200	60	8	360	250	212.8	200	5.51	5.38	20.52	3.72	3.81
2846	70	72	1200	60	12	360	250	212.8	200	5.50	5.28	22.13	4.02	4.19
2858	70	24	1200	80	4	480	250	212.8	200	4.17	4.11	13.71	3.29	3.33
2864	70	48	1200	80	8	480	250	212.8	200	4.16	4.04	15.37	3.69	3.81
2180	75	24	100	10	4	60	80	51.76	50	369.11	324.92	1074.16	2.91	3.31

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
2216	75	24	200	10	4	60	80	51.76	50	193.25	179.87	558.22	2.89	3.10
2378	75	24	200	10	4	60	150	103.5	100	188.96	179.87	568.77	3.01	3.16
2738	75	24	800	40	4	240	250	207.1	200	12.34	12.22	41.41	3.35	3.39
2744	75	48	800	40	8	240	250	207.1	200	12.24	11.88	45.89	3.75	3.86
2762	75	24	800	60	4	360	250	207.1	200	8.30	8.15	27.43	3.31	3.37
2768	75	48	800	60	8	360	250	207.1	200	8.24	7.92	30.53	3.70	3.86
2774	75	72	800	60	12	360	250	207.1	200	8.21	7.70	32.76	3.99	4.25

E.3 Parametric Study Results for $R/t_w = 10$

Parametric study results for the development of the SCF-ENS equation for R/tw = 10

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
24	30	40	100	10	4	60	200	100	50	338.32	324.92	856.01	2.53	2.63
42	30	40	100	20	4	120	200	100	50	174.26	162.46	453.74	2.60	2.79
60	30	40	200	10	4	60	200	100	50	182.45	179.87	447.59	2.45	2.49
78	30	40	200	20	4	120	200	100	50	91.8	89.93	234.53	2.55	2.61
84	30	80	200	20	8	120	200	100	50	86.45	81.23	246.88	2.86	3.04
96	30	40	200	40	4	240	200	100	50	47.52	44.97	115.88	2.44	2.58
102	30	80	200	40	8	240	200	100	50	45.89	40.61	118.23	2.58	2.91
108	30	100	200	40	10	240	200	100	50	45.16	38.74	119.77	2.65	3.09
114	30	40	400	20	4	120	200	100	50	47.9	47.51	119.85	2.50	2.52
120	30	80	400	20	8	120	200	100	50	46.58	44.97	124.28	2.67	2.76
132	30	40	400	40	4	240	200	100	50	24.45	23.76	58.83	2.41	2.48
138	30	80	400	40	8	240	200	100	50	24	22.48	60.04	2.50	2.67
144	30	100	400	40	10	240	200	100	50	23.82	21.90	60.37	2.53	2.76
150	30	40	400	60	4	360	200	100	50	16.55	15.84	38.85	2.35	2.45
156	30	80	400	60	8	360	200	100	50	16.35	14.99	39.24	2.40	2.62
162	30	100	400	60	10	360	200	100	50	16.26	14.60	39.21	2.41	2.69
168	30	40	600	20	4	120	200	100	50	32.53	32.28	81.15	2.49	2.51
174	30	80	600	20	8	120	200	100	50	31.99	31.09	84.51	2.64	2.72
186	30	40	600	40	4	240	200	100	50	16.49	16.14	39.74	2.41	2.46
198	30	100	600	40	10	240	200	100	50	16.22	15.26	40.84	2.52	2.68
204	30	40	600	60	4	360	200	100	50	11.11	10.76	26.22	2.36	2.44
210	30	80	600	60	8	360	200	100	50	11.02	10.36	26.48	2.40	2.56
216	30	100	600	60	10	360	200	100	50	10.99	10.17	26.50	2.41	2.60
246	30	40	200	20	4	120	300	200	100	91.72	89.93	255.22	2.78	2.84
252	30	80	200	20	8	120	300	200	100	85.66	81.23	260.20	3.04	3.20
270	30	40	200	40	4	240	300	200	100	46.32	44.97	127.21	2.75	2.83
282	30	120	200	40	12	240	300	200	100	42.19	37.03	132.94	3.15	3.59
294	30	40	400	20	4	120	300	200	100	47.45	47.51	132.21	2.79	2.78
300	30	80	400	20	8	120	300	200	100	45.51	44.97	135.49	2.98	3.01
318	30	40	400	40	4	240	300	200	100	23.97	23.76	64.98	2.71	2.74
330	30	120	400	40	12	240	300	200	100	22.7	21.34	67.39	2.97	3.16
336	30	160	400	40	16	240	300	200	100	22.09	20.31	69.11	3.13	3.40
342	30	40	400	60	4	360	300	200	100	16.25	15.84	43.33	2.67	2.74
348	30	80	400	60	8	360	300	200	100	15.88	14.99	43.45	2.74	2.90
354	30	120	400	60	12	360	300	200	100	15.59	14.23	44.47	2.85	3.13
360	30	160	400	60	16	360	300	200	100	15.3	13.54	44.96	2.94	3.32
366	30	40	800	40	4	240	300	200	100	12.31	12.22	33.14	2.69	2.71
378	30	120	800	40	12	240	300	200	100	12.03	11.55	34.34	2.86	2.97
384	30	160	800	40	16	240	300	200	100	11.86	11.24	34.92	2.95	3.11
390	30	40	800	60	4	360	300	200	100	8.27	8.15	22.05	2.66	2.71
396	30	80	800	60	8	360	300	200	100	8.19	7.92	22.18	2.71	2.80
402	30	120	800	60	12	360	300	200	100	8.12	7.70	22.67	2.79	2.94
408	30	160	800	60	16	360	300	200	100	8.05	7.49	22.71	2.82	3.03
414	30	40	800	80	4	480	300	200	100	6.25	6.11	16.35	2.62	2.68

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
420	30	80	800	80	8	480	300	200	100	6.2	5.94	16.47	2.66	2.77
426	30	120	800	80	12	480	300	200	100	6.15	5.78	16.68	2.71	2.89
432	30	160	800	80	16	480	300	200	100	6.11	5.62	16.83	2.75	2.99
438	30	40	1200	40	4	240	300	200	100	8.26	8.23	22.36	2.71	2.72
450	30	120	1200	40	12	240	300	200	100	8.14	7.92	23.31	2.86	2.94
456	30	160	1200	40	16	240	300	200	100	8.09	7.77	23.86	2.95	3.07
462	30	40	1200	60	4	360	300	200	100	5.54	5.49	14.87	2.68	2.71
468	30	80	1200	60	8	360	300	200	100	5.51	5.38	14.98	2.72	2.78
474	30	120	1200	60	12	360	300	200	100	5.47	5.28	15.35	2.80	2.91
510	30	40	400	20	4	120	600	400	200	48	47.51	140.10	2.92	2.95
534	30	40	400	40	4	240	600	400	200	23.74	23.76	77.12	3.25	3.25
540	30	80	400	40	8	240	600	400	200	22.98	22.48	76.49	3.33	3.40
546	30	120	400	40	12	240	600	400	200	22.12	21.34	76.74	3.47	3.60
552	30	160	400	40	16	240	600	400	200	21.53	20.31	77.53	3.60	3.82
558	30	40	400	80	4	480	600	400	200	11.97	11.88	38.47	3.21	3.24
564	30	80	400	80	8	480	600	400	200	11.59	11.24	37.63	3.25	3.35
570	30	120	400	80	12	480	600	400	200	11.27	10.67	38.00	3.37	3.56
576	30	160	400	80	16	480	600	400	200	11.07	10.15	38.37	3.47	3.78
582	30	40	800	40	4	240	600	400	200	12.09	12.22	39.58	3.27	3.24
588	30	80	800	40	8	240	600	400	200	11.85	11.88	39.72	3.35	3.34
594	30	120	800	40	12	240	600	400	200	11.62	11.55	40.10	3.45	3.47
600	30	160	800	40	16	240	600	400	200	11.36	11.24	40.55	3.57	3.61
606	30	40	800	60	4	360	600	400	200	8.07	8.15	26.20	3.25	3.22
612	30	80	800	60	8	360	600	400	200	7.91	7.92	26.30	3.32	3.32
618	30	120	800	60	12	360	600	400	200	7.79	7.70	26.36	3.39	3.42
624	30	160	800	60	16	360	600	400	200	7.67	7.49	26.72	3.48	3.57
636	30	80	800	80	8	480	600	400	200	5.97	5.94	19.35	3.24	3.26
642	30	120	800	80	12	480	600	400	200	5.88	5.78	19.53	3.32	3.38
648	30	160	800	80	16	480	600	400	200	5.81	5.62	19.83	3.41	3.53
654	30	40	1200	40	4	240	600	400	200	8.18	8.23	26.53	3.24	3.22
660	30	80	1200	40	8	240	600	400	200	8.09	8.07	26.71	3.30	3.31
666	30	120	1200	40	12	240	600	400	200	7.98	7.92	26.96	3.38	3.40
672	30	160	1200	40	16	240	600	400	200	7.88	7.77	27.25	3.46	3.51
678	30	40	1200	60	4	360	600	400	200	5.46	5.49	17.60	3.22	3.21
684	30	80	1200	60	8	360	600	400	200	5.39	5.38	17.71	3.29	3.29
690	30	120	1200	60	12	360	600	400	200	5.34	5.28	17.77	3.33	3.37
696	30	160	1200	60	16	360	600	400	200	5.29	5.18	18.00	3.40	3.47
702	30	40	1200	80	4	480	600	400	200	4.11	4.11	13.13	3.19	3.19
708	30	80	1200	80	8	480	600	400	200	4.06	4.04	13.03	3.21	3.23
714	30	120	1200	80	12	480	600	400	200	4.02	3.96	13.17	3.27	3.33
720	30	160	1200	80	16	480	600	400	200	4	3.89	13.38	3.35	3.44
744	45	40	100	10	4	60	200	70.71	50	353.61	324.92	1000.71	2.83	3.08
762	45	40	100	20	4	120	200	70.71	50	178.12	162.46	512.58	2.88	3.16
780	45	40	200	10	4	60	200	70.71	50	183.56	179.87	511.03	2.78	2.84
798	45	40	200	20	4	120	200	70.71	50	92.26	89.93	261.12	2.83	2.90
804	45	80	200	20	8	120	200	70.71	50	87.2	81.23	271.99	3.12	3.35

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
816	45	40	200	40	4	240	200	70.71	50	47.82	44.97	128.67	2.69	2.86
822	45	80	200	40	8	240	200	70.71	50	46.25	40.61	131.19	2.84	3.23
834	45	40	400	20	4	120	200	70.71	50	48.09	47.51	133.81	2.78	2.82
840	45	80	400	20	8	120	200	70.71	50	46.95	44.97	136.90	2.92	3.04
852	45	40	400	40	4	240	200	70.71	50	24.55	23.76	65.47	2.67	2.76
858	45	80	400	40	8	240	200	70.71	50	24.16	22.48	66.43	2.75	2.95
870	45	40	400	60	4	360	200	70.71	50	16.57	15.84	42.98	2.59	2.71
888	45	40	600	20	4	120	200	70.71	50	32.58	32.28	90.66	2.78	2.81
894	45	80	600	20	8	120	200	70.71	50	32.15	31.09	93.48	2.91	3.01
906	45	40	600	40	4	240	200	70.71	50	16.52	16.14	44.27	2.68	2.74
912	45	80	600	40	8	240	200	70.71	50	16.37	15.54	45.12	2.76	2.90
924	45	40	600	60	4	360	200	70.71	50	11.12	10.76	29.04	2.61	2.70
966	45	40	200	20	4	120	300	141.4	100	92.83	89.93	275.28	2.97	3.06
972	45	80	200	20	8	120	300	141.4	100	87.69	81.23	297.63	3.39	3.66
990	45	40	200	40	4	240	300	141.4	100	46.51	44.97	137.51	2.96	3.06
996	45	80	200	40	8	240	300	141.4	100	44.77	40.61	147.08	3.29	3.62
1014	45	40	400	20	4	120	300	141.4	100	47.68	47.51	141.73	2.97	2.98
1020	45	80	400	20	8	120	300	141.4	100	45.84	44.97	150.79	3.29	3.35
1038	45	40	400	40	4	240	300	141.4	100	23.98	23.76	69.88	2.91	2.94
1044	45	80	400	40	8	240	300	141.4	100	23.3	22.48	74.95	3.22	3.33
1050	45	120	400	40	12	240	300	141.4	100	22.92	21.34	77.21	3.37	3.62
1062	45	40	400	60	4	360	300	141.4	100	16.27	15.84	46.13	2.83	2.91
1074	45	120	400	60	12	360	300	141.4	100	15.68	14.23	51.29	3.27	3.61
1080	45	160	400	60	16	360	300	141.4	100	15.45	13.54	49.91	3.23	3.69
1086	45	40	800	40	4	240	300	141.4	100	12.3	12.22	35.71	2.90	2.92
1092	45	80	800	40	8	240	300	141.4	100	12.15	11.88	38.28	3.15	3.22
1098	45	120	800	40	12	240	300	141.4	100	12.04	11.55	39.51	3.28	3.42
1104	45	160	800	40	16	240	300	141.4	100	11.97	11.24	39.03	3.26	3.47
1110	45	40	800	60	4	360	300	141.4	100	8.28	8.15	23.52	2.84	2.89
1122	45	120	800	60	12	360	300	141.4	100	8.14	7.70	26.04	3.20	3.38
1128	45	160	800	60	16	360	300	141.4	100	8.1	7.49	25.41	3.14	3.39
1134	45	40	800	80	4	480	300	141.4	100	6.25	6.11	17.58	2.81	2.88
1146	45	120	800	80	12	480	300	141.4	100	6.17	5.78	19.29	3.13	3.34
1152	45	160	800	80	16	480	300	141.4	100	6.15	5.62	18.94	3.08	3.37
1158	45	40	1200	40	4	240	300	141.4	100	8.25	8.23	24.08	2.92	2.93
1164	45	80	1200	40	8	240	300	141.4	100	8.2	8.07	25.89	3.16	3.21
1170	45	120	1200	40	12	240	300	141.4	100	8.16	7.92	26.81	3.29	3.39
1176	45	160	1200	40	16	240	300	141.4	100	8.13	7.77	26.61	3.27	3.42
1182	45	40	1200	60	4	360	300	141.4	100	5.54	5.49	15.82	2.86	2.88
1194	45	120	1200	60	12	360	300	141.4	100	5.48	5.28	17.76	3.24	3.36
1230	45	40	400	20	4	120	600	282.8	200	47.83	47.51	147.01	3.07	3.09
1236	45	80	400	20	8	120	600	282.8	200	46.05	44.97	154.66	3.36	3.44
1254	45	40	400	40	4	240	600	282.8	200	23.72	23.76	78.38	3.30	3.30
1260	45	80	400	40	8	240	600	282.8	200	23.15	22.48	82.67	3.57	3.68
1266	45	120	400	40	12	240	600	282.8	200	22.54	21.34	85.84	3.81	4.02
1272	45	160	400	40	16	240	600	282.8	200	21.7	20.31	89.91	4.14	4.43

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
1278	45	40	400	80	4	480	600	282.8	200	11.93	11.88	39.21	3.29	3.30
1284	45	80	400	80	8	480	600	282.8	200	11.55	11.24	41.19	3.57	3.66
1302	45	40	800	40	4	240	600	282.8	200	12.13	12.22	39.99	3.30	3.27
1308	45	80	800	40	8	240	600	282.8	200	11.9	11.88	42.28	3.55	3.56
1314	45	120	800	40	12	240	600	282.8	200	11.71	11.55	43.72	3.73	3.78
1320	45	160	800	40	16	240	600	282.8	200	11.5	11.24	45.41	3.95	4.04
1326	45	40	800	60	4	360	600	282.8	200	8.08	8.15	26.62	3.30	3.27
1332	45	80	800	60	8	360	600	282.8	200	7.93	7.92	28.14	3.55	3.55
1338	45	120	800	60	12	360	600	282.8	200	7.81	7.70	29.25	3.75	3.80
1344	45	160	800	60	16	360	600	282.8	200	7.7	7.49	30.14	3.91	4.02
1350	45	40	800	80	4	480	600	282.8	200	6.11	6.11	19.87	3.25	3.25
1356	45	80	800	80	8	480	600	282.8	200	5.99	5.94	21.03	3.51	3.54
1362	45	120	800	80	12	480	600	282.8	200	5.9	5.78	21.95	3.72	3.80
1368	45	160	800	80	16	480	600	282.8	200	5.83	5.62	22.50	3.86	4.00
1374	45	40	1200	40	4	240	600	282.8	200	8.19	8.23	26.97	3.29	3.28
1380	45	80	1200	40	8	240	600	282.8	200	8.09	8.07	28.59	3.53	3.54
1386	45	120	1200	40	12	240	600	282.8	200	8.02	7.92	29.52	3.68	3.73
1392	45	160	1200	40	16	240	600	282.8	200	7.92	7.77	30.58	3.86	3.93
1398	45	40	1200	60	4	360	600	282.8	200	5.46	5.49	17.94	3.28	3.27
1404	45	80	1200	60	8	360	600	282.8	200	5.39	5.38	19.03	3.53	3.54
1410	45	120	1200	60	12	360	600	282.8	200	5.34	5.28	19.78	3.71	3.75
1416	45	160	1200	60	16	360	600	282.8	200	5.29	5.18	20.36	3.85	3.93
1422	45	40	1200	80	4	480	600	282.8	200	4.12	4.11	13.38	3.25	3.25
1428	45	80	1200	80	8	480	600	282.8	200	4.06	4.04	14.19	3.49	3.52
1434	45	120	1200	80	12	480	600	282.8	200	4.03	3.96	14.83	3.68	3.75
1440	45	160	1200	80	16	480	600	282.8	200	3.99	3.89	15.19	3.81	3.91
1464	60	40	100	10	4	60	200	57.74	50	350	324.92	1054.05	3.01	3.24
1482	60	40	100	20	4	120	200	57.74	50	177.33	162.46	547.88	3.09	3.37
1500	60	40	200	10	4	60	200	57.74	50	183.99	179.87	537.46	2.92	2.99
1518	60	40	200	20	4	120	200	57.74	50	92.49	89.93	277.33	3.00	3.08
1536	60	40	200	40	4	240	200	57.74	50	47.87	44.97	135.87	2.84	3.02
1554	60	40	400	20	4	120	200	57.74	50	48.3	47.51	142.06	2.94	2.99
1572	60	40	400	40	4	240	200	57.74	50	24.66	23.76	68.93	2.80	2.90
1608	60	40	600	20	4	120	200	57.74	50	32.61	32.28	96.36	2.96	2.98
1626	60	40	600	40	4	240	200	57.74	50	16.55	16.14	46.66	2.82	2.89
1686	60	40	200	20	4	120	300	115.5	100	93.49	89.93	285.65	3.06	3.18
1716	60	80	200	40	8	240	300	115.5	100	45.33	40.61	157.80	3.48	3.89
1734	60	40	400	20	4	120	300	115.5	100	47.88	47.51	146.04	3.05	3.07
1740	60	80	400	20	8	120	300	115.5	100	46.18	44.97	158.77	3.44	3.53
1758	60	40	400	40	4	240	300	115.5	100	24.08	23.76	73.02	3.03	3.07
1764	60	80	400	40	8	240	300	115.5	100	23.49	22.48	80.18	3.41	3.57
1782	60	40	400	60	4	360	300	115.5	100	16.36	15.84	48.25	2.95	3.05
1806	60	40	800	40	4	240	300	115.5	100	12.32	12.22	37.35	3.03	3.06
1812	60	80	800	40	8	240	300	115.5	100	12.22	11.88	41.06	3.36	3.46
1830	60	40	800	60	4	360	300	115.5	100	8.29	8.15	24.63	2.97	3.02
1878	60	40	1200	40	4	240	300	115.5	100	8.25	8.23	25.14	3.05	3.05

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
1884	60	80	1200	40	8	240	300	115.5	100	8.21	8.07	27.78	3.38	3.44
1902	60	40	1200	60	4	360	300	115.5	100	5.53	5.49	16.60	3.00	3.03
1974	60	40	400	40	4	240	600	230.9	200	23.8	23.76	79.38	3.34	3.34
1980	60	80	400	40	8	240	600	230.9	200	23.21	22.48	86.30	3.72	3.84
1986	60	120	400	40	12	240	600	230.9	200	22.48	21.34	90.66	4.03	4.25
1992	60	160	400	40	16	240	600	230.9	200	21.81	20.31	94.90	4.35	4.67
1998	60	40	400	80	4	480	600	230.9	200	11.97	11.88	39.97	3.34	3.37
2010	60	120	400	80	12	480	600	230.9	200	11.4	10.67	46.19	4.05	4.33
2016	60	160	400	80	16	480	600	230.9	200	11.25	10.15	48.01	4.27	4.73
2022	60	40	800	40	4	240	600	230.9	200	12.14	12.22	40.48	3.33	3.31
2028	60	80	800	40	8	240	600	230.9	200	11.92	11.88	44.20	3.71	3.72
2034	60	120	800	40	12	240	600	230.9	200	11.74	11.55	46.24	3.94	4.00
2040	60	160	800	40	16	240	600	230.9	200	11.52	11.24	47.83	4.15	4.25
2046	60	40	800	60	4	360	600	230.9	200	8.09	8.15	27.02	3.34	3.32
2058	60	120	800	60	12	360	600	230.9	200	7.85	7.70	31.41	4.00	4.08
2064	60	160	800	60	16	360	600	230.9	200	7.73	7.49	32.31	4.18	4.31
2070	60	40	800	80	4	480	600	230.9	200	6.12	6.11	20.18	3.30	3.30
2082	60	120	800	80	12	480	600	230.9	200	5.94	5.78	23.40	3.94	4.05
2088	60	160	800	80	16	480	600	230.9	200	5.85	5.62	24.25	4.14	4.31
2094	60	40	1200	40	4	240	600	230.9	200	8.19	8.23	27.33	3.34	3.32
2100	60	80	1200	40	8	240	600	230.9	200	8.11	8.07	29.88	3.68	3.70
2106	60	120	1200	40	12	240	600	230.9	200	8.03	7.92	31.30	3.90	3.95
2112	60	160	1200	40	16	240	600	230.9	200	7.94	7.77	32.28	4.07	4.15
2118	60	40	1200	60	4	360	600	230.9	200	5.47	5.49	18.21	3.33	3.32
2130	60	120	1200	60	12	360	600	230.9	200	5.35	5.28	21.24	3.97	4.02
2136	60	160	1200	60	16	360	600	230.9	200	5.31	5.18	21.84	4.12	4.22
2142	60	40	1200	80	4	480	600	230.9	200	4.13	4.11	13.59	3.29	3.30
2154	60	120	1200	80	12	480	600	230.9	200	4.04	3.96	15.80	3.91	3.99
2160	60	160	1200	80	16	480	600	230.9	200	4.01	3.89	16.38	4.09	4.22
2430	70	40	200	40	4	240	300	106.4	100	47.03	44.97	143.58	3.05	3.19
2478	70	40	400	40	4	240	300	106.4	100	24.22	23.76	73.10	3.02	3.08
2526	70	40	800	40	4	240	300	106.4	100	12.33	12.22	37.43	3.03	3.06
2550	70	40	800	60	4	360	300	106.4	100	8.3	8.15	24.54	2.96	3.01
2598	70	40	1200	40	4	240	300	106.4	100	8.25	8.23	25.47	3.09	3.10
2622	70	40	1200	60	4	360	300	106.4	100	5.54	5.49	16.47	2.97	3.00
2694	70	40	400	40	4	240	600	212.8	200	23.82	23.76	76.16	3.20	3.21
2700	70	80	400	40	8	240	600	212.8	200	23.13	22.48	85.29	3.69	3.79
2706	70	120	400	40	12	240	600	212.8	200	22.45	21.34	90.64	4.04	4.25
2718	70	40	400	80	4	480	600	212.8	200	12.03	11.88	38.81	3.23	3.27
2724	70	80	400	80	8	480	600	212.8	200	11.67	11.24	43.65	3.74	3.88
2730	70	120	400	80	12	480	600	212.8	200	11.44	10.67	46.95	4.10	4.40
2742	75	40	800	40	4	240	600	207.1	200	12.17	12.22	38.20	3.14	3.13
2748	75	80	800	40	8	240	600	207.1	200	11.96	11.88	43.05	3.60	3.62
2754	75	120	800	40	12	240	600	207.1	200	11.76	11.55	46.08	3.92	3.99
2766	75	40	800	60	4	360	600	207.1	200	8.12	8.15	25.57	3.15	3.14
2772	75	80	800	60	8	360	600	207.1	200	7.97	7.92	29.52	3.70	3.73

case_id	θ	R	bf	tf	tw	hw	a1	a2	a3	$\sigma_{nom.FE}$	$\sigma_{nom,Hand}$	σ_{ENS}	SCF_{FE}	SCF_{Hand}
2778	75	120	800	60	12	360	600	207.1	200	7.88	7.70	31.72	4.02	4.12
2790	70	40	800	80	4	480	600	212.8	200	6.13	6.11	19.55	3.19	3.20
2796	70	80	800	80	8	480	600	212.8	200	6.02	5.94	22.07	3.67	3.72
2802	70	120	800	80	12	480	600	212.8	200	5.96	5.78	23.76	3.99	4.11
2814	70	40	1200	40	4	240	600	212.8	200	8.2	8.23	26.23	3.20	3.19
2820	70	80	1200	40	8	240	600	212.8	200	8.1	8.07	29.62	3.65	3.67
2838	70	40	1200	60	4	360	600	212.8	200	5.47	5.49	17.61	3.22	3.21
2844	70	80	1200	60	8	360	600	212.8	200	5.4	5.38	20.13	3.72	3.74
2850	70	120	1200	60	12	360	600	212.8	200	5.37	5.28	21.44	3.99	4.06
2862	70	40	1200	80	4	480	600	212.8	200	4.13	4.11	13.17	3.19	3.20
2868	70	80	1200	80	8	480	600	212.8	200	4.08	4.04	14.89	3.65	3.69
2874	70	120	1200	80	12	480	600	212.8	200	4.06	3.96	16.05	3.96	4.05