FRAMEWORK FOR THE GENERATION AND DESIGN OF NATURALLY FUNCTIONALLY GRADED LATTICE STRUCTURES

A Dissertation
Presented to
The Academic Faculty

by

Mahmoud Ali Alzahrani

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Mechanical Engineering

Georgia Institute of Technology
August 2020

FRAMEWORK FOR THE GENERATION AND DESIGN OF NATURALLY FUNCTIONALLY GRADED LATTICE STRUCTURES

I dedicate this dissertation to my family for their support and love.

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Dr. Seung-Kyum Choi, for his support and guidance to me with his knowledge throughout my graduate studies and school experience. I would also like to thank the reading committee, Dr. David Rosen, Dr. Jarek Rossignac, Dr. Steven Liang, and Dr. Graeme Kennedy, for their time to read and provide their knowledge and expertise in the completion of my Ph.D. dissertation.

I would also like to thank my family, especially my grandfather and mother, for their support and sacrificing their time to come here to America so that I can complete my graduate studies. Special thanks to my professors at King Abdulaziz University, for the things they taught me during my undergraduate studies and for their support for me to pursue my graduate degrees.

Finally, I would like to thank my lab-mates and friends at Georgia Tech, Recep Gorguluarslan, Sung-Kun Hwang, Ali Alsaibie, and Andrew Carlile for their help along the way and their support.

TABLE OF CONTENTS

ACKNOWLEDGMENTS IV

LIST OF TABLES VII

LIST OF FIGURES IX

LIST OF SYMBOLS AND ABBREVIATIONS XXI

SUMMARY XXIV

CHAPTER 1. INTRODUCTION 1

1.1 Cellular Solids 1

1.2 Topology Optimization 4

1.3 Functionally Graded Lattice (FGL) 11

1.3.1 Naturally Functionally Graded Lattice (NFGL) Structures 19

1.4 Research Objectives 28

1.5 Research Questions 28

1.6 Dissertation Organization 35

CHAPTER 2. CURRENT STATE OF THE ART 37

2.1 Error Diffusion 37

2.2 Stochastic Nodal Generation method 42

2.3 Local Volume Constraint Optimization 45

2.4 Adaptive Quadtree Optimization 51

2.5 Drawbacks of Existing Methods 57

CHAPTER 3. NATURALLY FUNCTIONALLY GRADED LATTICE (NFGL) FRAMEWORK 61

3.1 NFGL Nodes Generation Algorithm (Simplified Sphere Packing) 63

3.2 NFGL Structure Generation Algorithm 70

3.3 Similarity Analysis using MSSIM Index Algorithm 77

3.4 Updating SR Value Algorithm 83

CHAPTER 4. APPLICATION EXAMPLES 86

4.1 Comparison of NFGL Framework with other Algorithms in the Literature 86

4.1.1 Computational Cost of Algorithms 87

4.1.2 Structural Performance 118

4.1.3 Robustness Testing 134

4.1.4 Conformity to Design Domain 145

4.1.4.1 Circular Design Domain 145

4.1.4.2 Curved Path Design Domain 153

4.2 Automotive Control Arm Optimization Under Multiple Loading Conditions 164

4.3 Injection Mold Lattice Cooling Channel Design 171

CHAPTER 5. CONCLUSIONS AND FUTURE WORK 184

5.1 Contributions 184

5.1.1 Addressing the Research Questions 184

5.1.2 List of Contributions 189

5.2 Future work 194

REFERENCES 197

LIST OF TABLES

Table 1 Organization of the Dissertation 35

Table 2 Advantages and drawbacks of the existing methods 59

Table 3 Cantilever beam mechanical properties 88

Table 4 Breakdown of the tasks that each method will perform to generate the NFGL structures 90

Table 5 Computational time to generate the NFGL structure of the cantilever beam using NFGL Framework, Stochastic Nodal Generation, Error Diffusion, Local Volume Constraint, and Adaptive Quadtree methods 93

Table 6 Breakdown of the computational cost of the cantilever beam for the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods 94

Table 7 Number of iterations for each topology optimization method used in the cantilever beam 95

Table 8 Computational time to generate the NFGL structure of the simply supported beam using NFGL Framework, Stochastic Nodal Generation, Error Diffusion, Local Volume Constraint, and Adaptive Quadtree methods 107

Table 9 Breakdown of the computational cost of the simply supported beam for the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods....................108

Table 10 Number of iterations for each topology optimization method used in the simply supported beam 109

Table 11 Compliance of different NFLG structure for the cantilever beam using the NFGL Framework, Stochastic Nodal Generation, Error Diffusion, Adaptive Quadtree, and Local Volume Constraint methods....................120

Table 12 Determining $\boldsymbol{SR}$ for the cantilever beam at different mesh sizes using Algorithm 3 and Algorithm 4 126

Table 13 Compliance of different NFLG structure for the simply supported beam using the NFGL Framework, Stochastic Nodal Generation, Error Diffusion, Adaptive Quadtree, and Local Volume Constraint methods..128

Table 14 Determining $\boldsymbol{SR}$ for the simply supported beam at different mesh sizes using Algorithm 3 and Algorithm 4 133

Table 15 Computational cost for the generation of NFGL structures at different $\boldsymbol{n}$ values 145

Table 16 Computational cost of the NFGL Framework, Error Diffusion, and Stochastic Nodal Generation methods for the circular design domain 152

Table 17 Computational cost for generating the NFGL structure for both curved paths using NFGL Framework, Error Diffusion, and Stochastic Nodal Generation methods..164

Table 18 Automotive control arm boundary and loading condition for both load cases.166

Table 19 Automotive control arm material properties..166

Table 20 Determining $\boldsymbol{SR}$ value for the control arm using Algorithm 3 and Algorithm 4 171

Table 21 Determining $\boldsymbol{SR}$ value injection mold cooling channel NFGL structure using Algorithm 3 and Algorithm 4 183

LIST OF FIGURES

Figure 1 Classification of cellular solids into a) two-dimensional polygons and three dimensional b) open cells c) closed cells polyhedra 1

Figure 2 Further classification of cellular solids based on cell configuration [8] 3

Figure 3 Example of a uniform and conformal lattice structures [18] 4

Figure 4 Topology optimization a) Discrete optimization b) Continuum optimization [25] 5

Figure 5 An example of a Michell Truss [19] 6

Figure 6 Discrete optimization of a truss structure with Ground Structure optimization a) Ground structure b) optimized truss design [26] 6

Figure 7 Homogenization topology optimization for continuum structures [27] 8

Figure 8 Topology optimization problem of a generalized shape 8

Figure 9 Structure optimization with SIMP method a) Design domain b) Optimization result c) Solid model obtained from optimization showing a large overhang structure d) Section view showing enclosed void [35] 11

Figure 10 Example of functionally graded cellular structures in nature [37, 38, 40] 12

Figure 11 Scaffold design with varying radial porosity [41] 13

Figure 12 a) Topology optimization result under bending loading condition b) Generated FGL structure from optimization result mapping [43] 14

Figure 13 Size Matching and Scaling unit-cell library a) MAV loading conditions b) Stress field results c) Generated FGL structure [44] 15

Figure 14 Generated FGL structure from mapping SIMP optimization results to unit-cell porosity [45] 15

Figure 15 Application of RDM method a) SIMP optimization results b) Generated FGL structure [46] 16

Figure 16 Different strategies to generate FGL structures a) Intersected b) Graded c) Scaled [47] 17

Figure 17 Experimental compression test on uniform lattice structures and graded lattice structures [60] 19

Figure 18 Generated NFGL structures using Function Representation a) Unit-cell type grading [63] b) Unit-cell size grading [62] 20

Figure 19 Application of the Error Diffusion method to generate NFGL structures a) Input density field b) Generated NFGL structure [64] 21

Figure 20 Generated structure using Error Diffusion a) Stress field b) Generated Voronoi sites c-d) Two steps of the optimization process e) Optimized graded foam structure [66] 22

Figure 21 Generation of graded foam structure using Quadtrees and Error Diffusion a) Deisgn domain b) Input density field c) Generated Quadtree structure d) Quadtree structure after Error Diffusion e) Generated space-filling structure f) Generated graded foam structure [67] 22

Figure 22 A model of a finger generated using randomized Voronoi foam [68] 23

Figure 23 Generation of NFGL structure by random nodal displacement a) Uniform nodes b) Randomly displaced nodes c) Generated NFGL structure [x] 24

Figure 24 Structure optimization by using rhombic unit-cells a) Initial structure b) Optimized structure [70] 24

Figure 25 Generated hollowed foam structure a) Voronoi diagram with inserted ellipse b) Insertion of four more ellipses c) Voronoi Diagram with 100 ellipses d) Fabricated part [71] 25

Figure 26 Skin temperature before and after wearing an orthopedic cast design) optimized cast b) uniform unit-cell cast [72]..26

Figure 27 Generation of bone-like porous structure using local volume constraint optimization [73]..27

Figure 28 Generation of NFGL nodes using Error Diffusion a) Input density field b) Generated NFGL nodes [64]...38

Figure 29 The two-dimensional filter proposed by Floyd and Steinberg 39

Figure 30 An example of the dithering process using the Error Diffusion method for a 5×5 grid of pixels [64] 40

Figure 31 Generated NFGL structure from Error Diffusion method a) NFGL nodes b) Generated NFGL structure using Delaunay triangulation [64]..41

Figure 32 Change in mean unit-cell edge length as the input density field values are increased [64]..42

Figure 33 Generation of NFGL nodes using Stochastic Nodal Generation method a) Input density field b) Cell subdivision c) generated NFGL nodes...44

Figure 34 Different NFGL nodes generated with each iteration of the Stochastic Nodal Generation method...44

Figure 35 Generated structure using Local Volume Constraint at different local volume ratios with a radius of 6 a) $\boldsymbol{\alpha = 0.6}$ b) $\boldsymbol{\alpha = 0.5}$ c) $\boldsymbol{\alpha = 0.4}$ [73]...................................49

Figure 36 Generated structure using Local Volume Constraint at different local volume ratios with a radius of 12 a) $\boldsymbol{\alpha = 0.6}$ b) $\boldsymbol{\alpha = 0.5}$ c) $\boldsymbol{\alpha = 0.4}$ [73].................................49

Figure 37 Generated structure using Local Volume Constraint at different local volume ratios with a radius of 6 and $\boldsymbol{\alpha = 0.6}$ a) No constraint b) $\boldsymbol{\alpha total = 0.50}$ c) $\boldsymbol{\alpha total = 0.4}$ [73]..50

Figure 38 Quadtree grid of a rectangular design domain a) Cell subdivision b) FEM mesh of the domain [68]...53

Figure 39 Generated cantilever beam NFGL structure without the refinement of $\chi i, jk$ values [68]..54

Figure 40 Generated cantilever NFGL beam after the refinement filter is applied [68] ...55

Figure 41 Generated cantilever NFGL beam after the balanced Quadtree refinement filter is applied [68] 55

Figure 42 Different structures generated using Adaptive Quadtree optimization method [68]..56

Figure 43 Framework to generate NFGL structures 62

Figure 44 Influence sphere radius of three nodes, colored in black, in a density field......65

Figure 45 Removal of nodes inside the influence sphere of a colored node if their relative density is less than the colored node relative density 66

Figure 46 NFGL nodes generated by the algorithm 66

Figure 47 Illustration of the simplified sphere packing...67

Figure 48 Algorithm 1 inputs a) Density field ρf b) input mesh nodes $\mathbf{N}c*$ 67

Figure 49 Generated NFGL nodes $\mathbf{N}c$ with a) $SR = 1$ b) $SR = 1.5$ c) $SR = 2$ d) $SR = 3$ 68

Figure 50 Section of a tapered beam with a circular hole a) Beam initial mesh b) Density field 69

Figure 51 Generated NFGL nodes at different SR values a) $SR = 2$ b) $SR = 3$ c) $SR = 5$ 69

Figure 52 Interior nodes generation in a tetrahedral unit-cell a) $NLayer = 1$ b) $NLayer = 3$ c) $NLayer = 9$ 73

Figure 53 NFGL nodal diameters based on $\boldsymbol{dlattice}$ values a) using the minimum value b) using the maximum value 75

Figure 54 NFGL strut diameters based on $\boldsymbol{dlattice}$ values a) using the maximum value b) using the minimum value 75

Figure 55 Generated NFGL structures using nodal diameter a) $\boldsymbol{SR = 2}$ b) $\boldsymbol{SR = 3}$ c) $\boldsymbol{SR = 5}$ 76

Figure 56 Generated NFGL structures using strut diameter a) $\boldsymbol{SR = 2}$ b) $\boldsymbol{SR = 3}$ c) $\boldsymbol{SR = 5}$ 76

Figure 57 A close-up view of the generated NFGL struts at $\boldsymbol{SR = 5}$ for a) Nodal diameter b) Strut diameter 77

Figure 58 Relative density Array for a generalized shape showing assigned values 78

Figure 59 Relative density Array for a generalized shape showing assigned values 82

Figure 60 Change in MSSIM index as a function $\boldsymbol{SR}$ showing the fitted power curve.....83

Figure 61 Loading conditions for the a) cantilever beam b) simply supported beam 88

Figure 62 Unpenalized SIMP optimization results a) Cantilever beam b) Simply supported beam 89

Figure 63 Computational cost of the cantilever beam for the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods against the number of FE elements used 96

Figure 64 Generated NFGL structures using the Stochastic Nodal Generation method for the cantilever beam at different sizes 98

Figure 65 Generated NFGL structures using the Error Diffusion method for the cantilever beam at different sizes 99

Figure 66 Generated NFGL structures using the Adaptive Quadtree method for the cantilever beam at different sizes..100

Figure 67 Generated NFGL structures using the Local Volume Constraint method for the cantilever beam at different sizes..101

Figure 68 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 64×32 at different $\boldsymbol{SR}$ values..102

Figure 69 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 128×64 at different $\boldsymbol{SR}$ values..102

Figure 70 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 256×128 at different $\boldsymbol{SR}$ values..103

Figure 71 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 512×256 at different $\boldsymbol{SR}$ values..103

Figure 72 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 640×320 at different $\boldsymbol{SR}$ values..104

Figure 73 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 768×384 at different $\boldsymbol{SR}$ values..104

Figure 74 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 1024×512 at different $\boldsymbol{SR}$ values..105

Figure 75 Generated NFGL structures using the Stochastic Nodal Generation method for the simply supported beam at different sizes...110

Figure 76 Generated NFGL structures using the Error Diffusion method for the simply supported beam at different sizes..111

Figure 77 Generated NFGL structures using the Adaptive Quadtree method for the simply supported beam at different sizes...112

Figure 78 Generated NFGL structures using the Local Volume Constraint method for the simply supported beam at different sizes...113

Figure 79 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 64×32 at different $\boldsymbol{SR}$ values 114

Figure 80 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 128×64 at different $\boldsymbol{SR}$ values 114

Figure 81 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 256×128 at different $\boldsymbol{SR}$ values 115

Figure 82 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 512×256 at different $\boldsymbol{SR}$ values 115

Figure 83 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 640×320 at different $\boldsymbol{SR}$ values 116

Figure 84 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 768×384 at different $\boldsymbol{SR}$ values 116

Figure 85 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 1024×512 at different $\boldsymbol{SR}$ values 117

Figure 86 Computational cost of the simply supported beam for the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods against the number of FE elements used..117

Figure 87 Change in compliance of the generated NFGL structures for the cantilever beam using the NFGL Framework as a function $\boldsymbol{RL}$...121

Figure 88 Change in compliance of the generated NFGL structures for the cantilever beam using the NFGL Framework as a function $\boldsymbol{SR}$...121

Figure 89 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 256×128 at $\boldsymbol{RL}$ values..122

Figure 90 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 640×320 at $\boldsymbol{RL}$ values..123

Figure 91 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 768×384 at $\boldsymbol{RL}$ values..124

Figure 92 Change in the MSSIM index as a function of $\boldsymbol{SR}$ for the cantilever beam 126

Figure 93 Change in compliance of the generated NFGL structures for the simply supported beam using the NFGL Framework as a function $\boldsymbol{RL}$ 129

Figure 94 Change in compliance of the generated NFGL structures for the simply supported beam using the NFGL Framework as a function $\boldsymbol{SR}$ 129

Figure 95 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 256×128 at $\boldsymbol{RL}$ values 130

Figure 96 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 640×320 at $\boldsymbol{RL}$ values 131

Figure 97 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 768×384 at $\boldsymbol{RL}$ values 132

Figure 98 Change in the MSSIM index as a function of $\boldsymbol{SR}$ for the simply supported beam...133

Figure 99 Sinusoidal input density field functions at different frequency values a) $\boldsymbol{n = 0.5}$ b) $\boldsymbol{n = 1}$ c) $\boldsymbol{n = 1.8}$ d) $\boldsymbol{n = 2}$ 135

Figure 100 Generated NFGL nodes using the Stochastic Nodal Generation method with a) $\boldsymbol{n = 0.5}$ b) $\boldsymbol{n = 1}$ c) $\boldsymbol{n = 1.8}$ d)$\boldsymbol{n = 2}$ 136

Figure 101 Generation of NFGL nodes at different steps for the Stochastic Nodal Generation using $\boldsymbol{n = 2}$ 137

Figure 102 Cell boundaries for the upper right corner at step 20 138

Figure 103 Generated NFGL nodes and structures using the Error Diffusion method for different values of $\boldsymbol{n}$..........139

Figure 104 Generated NFGL nodes and structures using the NFGL Framework for $\boldsymbol{n} = \mathbf{0.5}$ at different $\boldsymbol{SR}$ values..........141

Figure 105 Generated NFGL nodes and structures using the NFGL Framework for $\boldsymbol{n} = \mathbf{1}$ at different $\boldsymbol{SR}$ values..........142

Figure 106 Generated NFGL nodes and structures using the NFGL Framework for $\boldsymbol{n} = \mathbf{1.8}$ at different $\boldsymbol{SR}$ values..........143

Figure 107 Generated NFGL nodes and structures using the NFGL Framework for $\boldsymbol{n} = \mathbf{2}$ at different $\boldsymbol{SR}$ values..........144

Figure 108 Density gradient of the circular design domain..........146

Figure 109 Transformation of the input density image into NFGL nodes using the Error Diffusion method for the circular design domain..........147

Figure 110 Generated NFGL structure using the Error Diffusion method for the circular design domain..........147

Figure 111 Close up of the generated NFGL unit-cells using the Error Diffusion method for the circular design domain..........148

Figure 112 Generated NFGL nodes using Stochastic Nodal Generation for the circular design domain a) Generated cells b) Generated NFGL nodes..........149

Figure 113 Generated NFGL structure using the Stochastic Nodal Generation method for the circular design domain..........149

Figure 114 Circular Design domain FEM mesh nodes..........151

Figure 115 Close up of the generated NFGL unit-cells using the NFGL Framework for the circular design domain with $\boldsymbol{SR} = \mathbf{5}$..........151

Figure 116 Generated NFGL structures using the NFGL Framework for the circular design domain a) $\boldsymbol{SR = 2}$ b) $\boldsymbol{SR = 3}$ c) $\boldsymbol{SR = 5}$ d) $\boldsymbol{SR = 10}$ 152

Figure 117 Curved path design domain a) $\boldsymbol{n = 0.25}$ b) $\boldsymbol{n = 4}$.....................................153

Figure 118 Generated NFGL structure for the curved path with $\boldsymbol{n = 0.25}$ using the Error Diffusion method a) Generated NFGL nodes b) Corresponding NFGL structure 154

Figure 119 Generated NFGL structure for the curved path with $\boldsymbol{n = 4}$ using the Error Diffusion method a) Generated NFGL nodes b) Corresponding NFGL structure 155

Figure 120 Generated NFGL nodes for the curved path for the Stochastic Nodal Generation method with a) $\boldsymbol{n = 0.25}$ b) $\boldsymbol{n = 4}$..156

Figure 121 Input FEM mesh for the design domains of the curved paths.......................157

Figure 122 Generated NFGL structure for the curved path with $\boldsymbol{n = 0.25}$ using the NFGL Framework method with $\boldsymbol{SR = 2}$ a) Generated NFGL nodes b) Corresponding NFGL structure 158

Figure 123 Generated NFGL structure for the curved path with $\boldsymbol{n = 0.25}$ using the NFGL Framework method with $\boldsymbol{SR = 3}$ a) Generated NFGL nodes b) Corresponding NFGL structure 159

Figure 124 Generated NFGL structure for the curved path with $\boldsymbol{n = 0.25}$ using the NFGL Framework method with $\boldsymbol{SR = 4}$ a) Generated NFGL nodes b) Corresponding NFGL structure 160

Figure 125 Generated NFGL structure for the curved path with $\boldsymbol{n = 4}$ using the NFGL Framework method with $\boldsymbol{SR = 2}$ a) Generated NFGL nodes b) Corresponding NFGL structure..161

Figure 126 Generated NFGL structure for the curved path with $\boldsymbol{n = 4}$ using the NFGL Framework method with $\boldsymbol{SR = 3}$ a) Generated NFGL nodes b) Corresponding NFGL structure..162

Figure 127 Generated NFGL structure for the curved path with $\boldsymbol{n} = \mathbf{4}$ using the NFGL Framework method with $\boldsymbol{SR} = \mathbf{4}$ a) Generated NFGL nodes b) Corresponding NFGL structure..163

Figure 128 Automotive control arm that will be optimized using lattice structures 165

Figure 129 SIMP optimization results for the automotive control arm 167

Figure 130 Generated FGL structure for the automotive control arm 167

Figure 131 Generated NFGL structure for the control arm with $\boldsymbol{SR} = \mathbf{3}$ 169

Figure 132 Automotive control arm combined compliance before and after size optimization at different $\boldsymbol{SR}$ values 170

Figure 133 Computational cost of conducting size optimization for the control arm lattice struts at different $\boldsymbol{SR}$ values..170

Figure 134 MSSIM index for the automotive control arm as a function of $\boldsymbol{SR}$ 171

Figure 135 Loading and boundary conditions for one-fourth of the injection mold cooling channel [55] 172

Figure 136 Optimized injection mold cooling channel using FGL structures via AH [55] 174

Figure 137 Optistruct optimization results for the injection mold cooling channel using a size of 10×10...175

Figure 138 Generated NFGL structures for the injection mold cooling channel using a size of 10×10 with different $\boldsymbol{SR}$ values..176

Figure 139 Relative mechanical compliance of the NFGL structure for the injection mold cooling channel with a size of 10×10 176

Figure 140 Relative thermal compliance of the NFGL structure for the injection mold cooling channel with a size of 10×10 177

Figure 141 Comparison of the total surface area subjected to convection heat transfer for two structures with different unit-cell sizes..177

Figure 142 Optistruct optimization results for the injection mold cooling channel using a size of 100×100...179

Figure 143 Generated NFGL structures for the injection mold cooling channel using a size of 100×100 with different $\boldsymbol{SR}$ values...180

Figure 144 Relative mechanical compliance of the NFGL structure for the injection mold cooling channel with a size of 100×100 181

Figure 145 Relative thermal compliance of the NFGL structure for the injection mold cooling channel with a size of 100×100 181

Figure 146 MSSIM index for the NFGL structure using a size of 10×10 at different $\boldsymbol{SR}$ values 182

Figure 147 MSSIM index for the NFGL structure using a size of 100×100 at different $\boldsymbol{SR}$ values 182

LIST OF SYMBOLS AND ABBREVIATIONS

α	Local volume constraint
A	Area
$A_{element}$	Solid element area
$A_{lattice}$	Lattice unit-cell area
C	Compliance
C_m	Mechanical compliance
C_T	Thermal compliance
CGAL	Computational Geometry Algorithms Library
d	NFGL lattice/node diameter
$\mathbf{d}$	NFGL lattice/node diameter vector
$\mathbf{d}_{lattice}$	Unit-cell struts' diameter
d_T	Diameter type
$d_{lattice}$	Unit-cell lattice diameter
E	Elastic modulus
n	Frequency
FEA	Finite Element Analysis
FEM	Finite Element Method
$\mathbf{F}$	Force vector
FGL	Functionally Graded Lattice
h	Convection coefficient
k	Conductivity
$\mathbf{K}$	Stiffness matrix

$\mathbf{K_m}$	Mechanical stiffness matrix
$\mathbf{K_T}$	Conduction stiffness matrix
$\mathbf{K}_{\boldsymbol{h}}$	Convection stiffness matrix
$\mathbf{L}_c$	NFGL structure struts connectivity vector
$\mathbf{m}$	Exterior or interior mesh nodes array
m_e	Number of exterior nodes in the mesh
m_i	Number of interior nodes in the mesh
$MSSIM$	Mean Structural Similarity index
$\mathbf{N}_c$	NFGL node coordinates array
$\mathbf{N}_c^*$	Base mesh node coordinates array
N_e	Number of FEM elements
N_{layer}	Number of layers for T_n
NFGL	Naturally Functionally Graded Lattice
Ω	Design domain
p	SIMP penalty
P	Force
ρ	FEM relative density
$\bar{\rho}$	Cellular solids relative density
ρ^*	Interior node relative density
$\boldsymbol{\rho}$	Relative density vector
ρ_{eff}	Unit-cell effective relative density
ρ_f	Density field function
R	Influence sphere radius
R_L	Largest influence sphere radius

R_S	Smallest influence sphere radius
S_R	Unit-cell size ratio
SSIM	Structural Similarity index
T_n	Tetrahedral number
θ	Temperature
$\mathbf{U}$	Displacement vector
V	Volume
$V_{element}$	Solid element volume
$V_{lattice}$	Lattice unit-cell volume
$\mathbf{x}$	Relative density local matrix
$\mathbf{X}$	Relative density tensor
$\mathbf{y}$	NFGL local approximate relative density matrix
$\mathbf{Y}$	NFGL approximate relative density tensor

SUMMARY

Functionally Graded Lattice (FGL) Structures have shown improved performance over uniform lattice structures in different fields. These structures contain unit-cells of varying porosity based on different functional requirements, which alters the properties of the structures. Another form of functional grading can be seen in materials in nature, where the cellular structure can vary in both cell porosity and size. Therefore, to distinguish between lattice structures that vary in porosity only and lattice structures that vary in both, we will refer to the latter in this research as Naturally Functionally Graded Lattice (NFGL) structures. However, research into NFGL structures' performance against FGL structures in the literature is lacking. Furthermore, the current methods in the literature to generate these structures are severely limited and suffer from multiple drawbacks, such as being computationally expensive, generate non-conformal lattice structure, stochastic in structure, limited in their ability to vary the unit-cell size ratios, and other drawbacks.

To address these issues, this research aims to develop a framework, namely the NFGL Framework, to generate NFGL structures without the drawbacks that exist in current methods and to improve the performance of the generated structures using the NFGL Framework against existing FGL structures. The NFGL Framework uses a novel method to generate nodes for NFGL structures from the nodes of a finite element mesh that conforms to the design domain and a density field input of the domain using a developed simplified sphere packing algorithm, which are then connected using Delaunay Triangulations. Furthermore, the NFGL Framework can perform a similarity analysis using a modified Mean Structural Similarity (MSSIM) index to improve the performance of the generated NFGL structure. The generated structures using the NFGL Framework were tested against the existing methods and showed to overcome the drawbacks of these methods with improved performance and computational time. Furthermore, the generated NFGL structures were tested against FGL structures and the results showed a performance gain from the use of NFGL structures over FGL structures with a reduced computational cost.

CHAPTER 1. INTRODUCTION

1.1 Cellular Solids

Cellular solids are structures made of an interconnected network of plates or struts. The plates and struts form the faces and the edges of what is called a cell. Cellular solids are common in nature and can be seen in materials such as wood, bones, and coral [1]. Cellular solids have been recently used to design structures that can provide multi-functional materials that can fulfill a variety of requirements such as high specific strength, thermal insulation, heat transfer, energy absorption, and energy harvesting [2-4]. They can be classified into two types; a two-dimensional array of polygons, and three-dimensional polyhedra cells as shown in Figure 1-a. Three-dimensional cells are further classified into open-cells (Figure 1-b) and closed-cells (Figure 1-c).

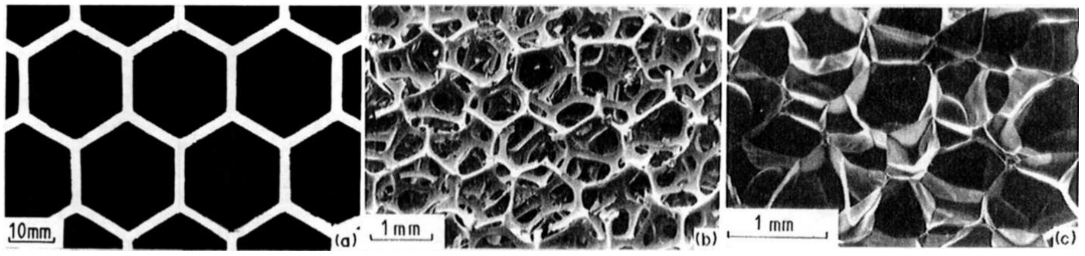


Figure 1 Classification of cellular solids into a) two-dimensional polygons and three dimensional b) open cells c) closed cells polyhedra

The fabrication of these structures varies depending on their type. Two-dimensional polygons, such as honeycomb structures, can be formed by sheet metal pressing and extrusion. Three-dimensional structures can be fabricated by foaming and solid-state processing such as electro-deposition and vapor-deposition [5, 6]. However, the produced cells are random in their arrangement. Varying the manufacturing parameters of these processes can alter the shape and the size of the cells but the generated structures still inherit a stochastic nature in the structure. Moreover, manufacturing processes, such as deformation forming and investment casting, have been used to fabricate three-dimensional, non-stochastic, arrangement of cellular structures [7]. However, these manufacturing processes require precise control, complicated apparatus, and further steps to assemble the structures.

Cellular solids can further be classified based on their cells' configuration into stochastic and non-stochastic structures as shown in Figure 2 [8]. Stochastic structures are commonly known as foams and they can be classified into open-cells or closed-cells foams. Non-stochastic structures are known as lattice structures and the cell is called a unit-cell. A unit-cell can be defined as a geometric set of points defined by a function, $f(x, y, z)$, inside a bounding domain, $g(x, y, z)$, such that $\{f(x, y, z) \geq 0 \ \forall \ x, y, z \ \in \mathbb{R} \mid g(x, y, z) \geq 0\}$. The unit-cell can be replicated, scaled, and oriented across the design domain Ω. An important property of lattice structures is the relative density, $\bar{\rho}$, which is a ratio between the volume of the cellular solid in the unit-cell and the volume of the bounding domain. The inverse of the relative density is known as the porosity of the unit-cell. Therefore, properties that increase with relative density, reduces as the porosity increases.

With the rise of additive manufacturing (AM) methods [9], the fabrication of complex lattice structures has gained considerable attention over foams due to their ability to provide lightweight and stronger structures compared to foams. The deformation of lattice structures is governed by the stretching of unit-cells, unlike most foams where their deformation is governed by the bending of the cell faces and edges. This deformation behavior affects the strength of the cellular structure [10-13]. Furthermore, the strength of cellular structures is related to their relative density. The strength of foams scales as $\bar{\rho}^{1.5}$ while lattice structures strength scales as $\bar{\rho}$. Therefore, a lattice structure with a relative density of $\bar{\rho} = 0.1$ is three times stronger than a foam counterpart with the same relative density. As a result, a lot of commercial software have included options for lattice generation and design [14, 15] or created their own lattice design platform [16, 17].

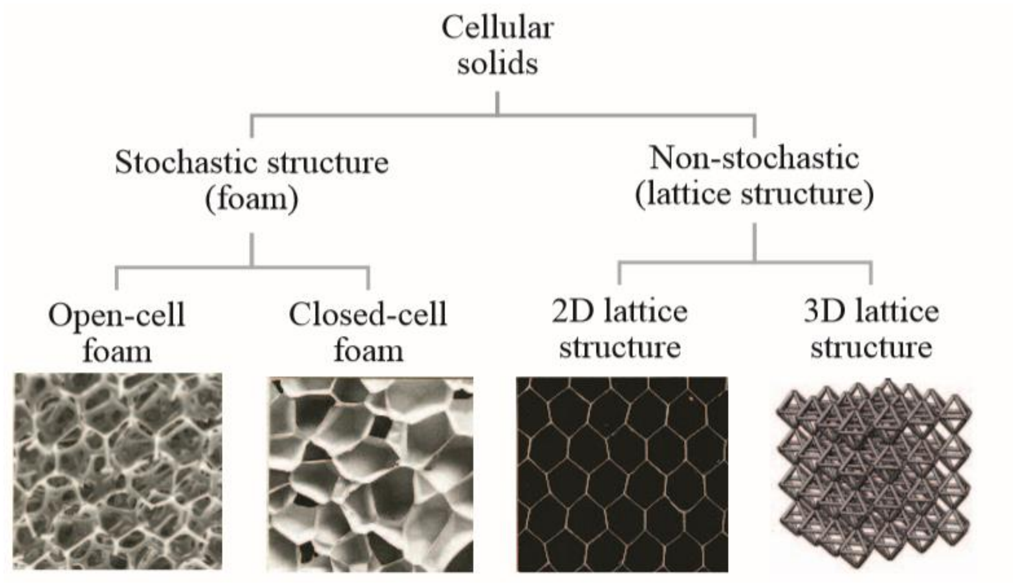


Figure 2 Further classification of cellular solids based on cell configuration [8]

Another important aspect of cellular structures is their ability to generate structures that conform to the design domain. Such structures are called conformal lattice structures, which is a term coined in [18] and they provide better stiffness to the structure than uniform structures. Figure 3 shows an example of a uniform and a conformal lattice structure. The term conformal will be used in this research in the same manner to denote cellular structures that conform to the design domain surface.

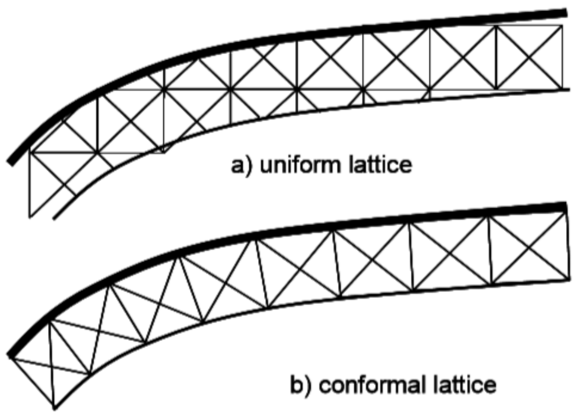


Figure 3 Example of a uniform and conformal lattice structures [18]

1.2 Topology Optimization

Topology optimization aims to optimize the material distribution in a given design domain under specified loading conditions to satisfy design constraints. Topology optimization can be divided based on the type of structure being optimized into two types: Discrete and Continuum as shown in Figure 4.

Discrete topology optimization has been employed mainly to truss and frame structures. The first study in discrete topology optimization was conducted by Michell in 1904 [19], which showed that the weight of the structure reaches a minimum when all the members follow the path of maximum strain magnitude. A structure that follows this optimality criterion is called a “Mitchell Truss”. An example of such a structure is shown in Figure 5. Following his work, no significant work was done for half a century until 1964 when methods for discrete structures optimization gained traction [20]. These methods can be categorized into three categories: Geometric, Hybrid, and Ground Structure [21]. In the geometric method, the design variables are the joint coordinates and the cross-sectional properties of the members. The number of design variables remains fixed during the optimization process and the joint coordinates and cross-sectional properties are optimized at the same time. In the hybrid method, the optimization of the members is carried out first, then the location of the joints is optimized. As for the ground structure method, a dense network of members with all potential connections to joints is generated in the design space, as shown in Figure 6, and the size of the members is then optimized while keeping the location of the joints fixed. Extensions have been made to the ground structure method to include the change in joint location and the growth of new members [22-24]. However, with the increase in the number of design variables, discrete topology optimization methods become computationally expensive. The complexity of optimizing these structures raises exponentially with the number of design variables.

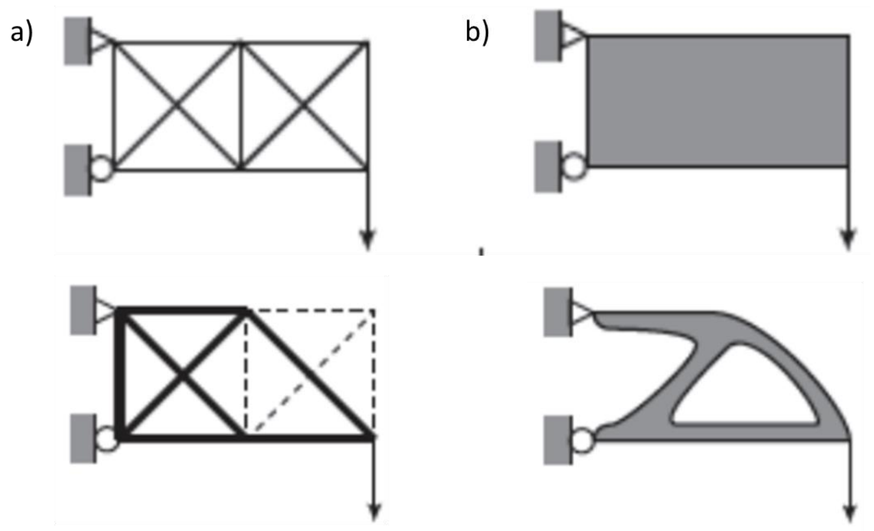


Figure 4 Topology optimization a) Discrete optimization b) Continuum optimization [25]

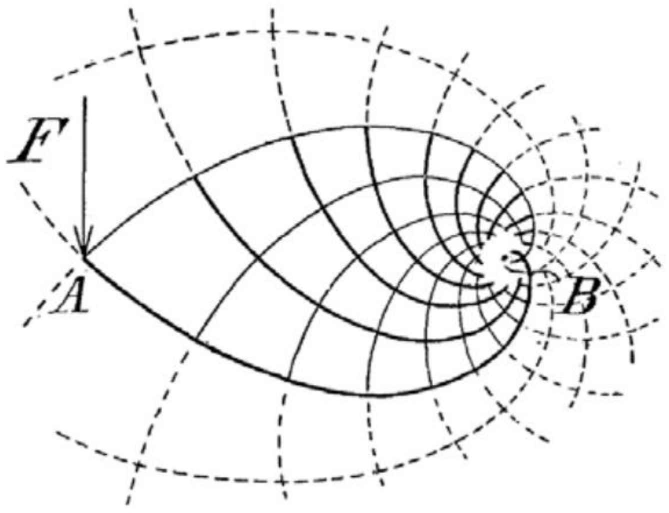


Figure 5 An example of a Michell Truss [19]

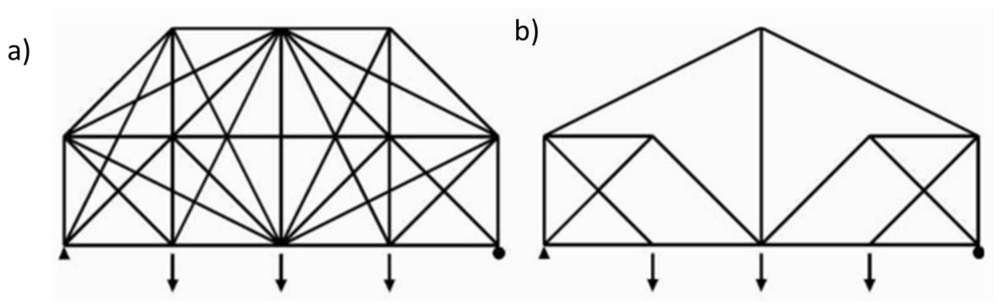


Figure 6 Discrete optimization of a truss structure with Ground Structure optimization a) Ground structure b) optimized truss design [26]

The early use of topology optimization in continuum structures traces back to the homogenization approach by Bendsoe and Kikuchi in 1988 [27]. The homogenization approach is a multi-scaling optimization, where the design space is partitioned into small patterned microstructures of a lower scale compared to the actual structure's scale as shown in Figure 7. In this optimization method, the design variables are the parameters of the lower scale composites. This, however, increases the computationally expensive immensely, which makes it cumbersome if not impossible to evaluate (for optimization cases other than compliance) the optimal parameters of the microstructure. Furthermore, there is no definite length-scale associated with the microstructures, making it difficult to fabricate the generated designs [28, 29]. Due to these drawbacks with the homogenization approach, Bendsoe proposed a simplified density approach [30], which was named the Solid Isotropic Material with Penalty (SIMP) approach [31]. The SIMP approach uses the Finite Element Method (FEM), as shown in Figure 8, to represent the design domain Ω under specified loading conditions and assumes a relative density value ρ (not to be confused with the cellular solids relative density $\bar{\rho}$) that is assigned to each element in the FEM model. The design variables in the SIMP optimization are the relative density values ρ_i of each element i in which a value of $\rho_i = 1$ denotes a solid region and a value of $\rho_i = 0$ denotes a void region in the design domain. However, using a discrete value for ρ_i would require the use of discrete optimization methods. So, in order use gradient based optimization methods, the values of ρ are allowed to take any value between 0 and 1. To avoid any ambiguity on the interpretation of intermediate values of ρ, a penalty is imposed on these values to push them to either 0 or 1.

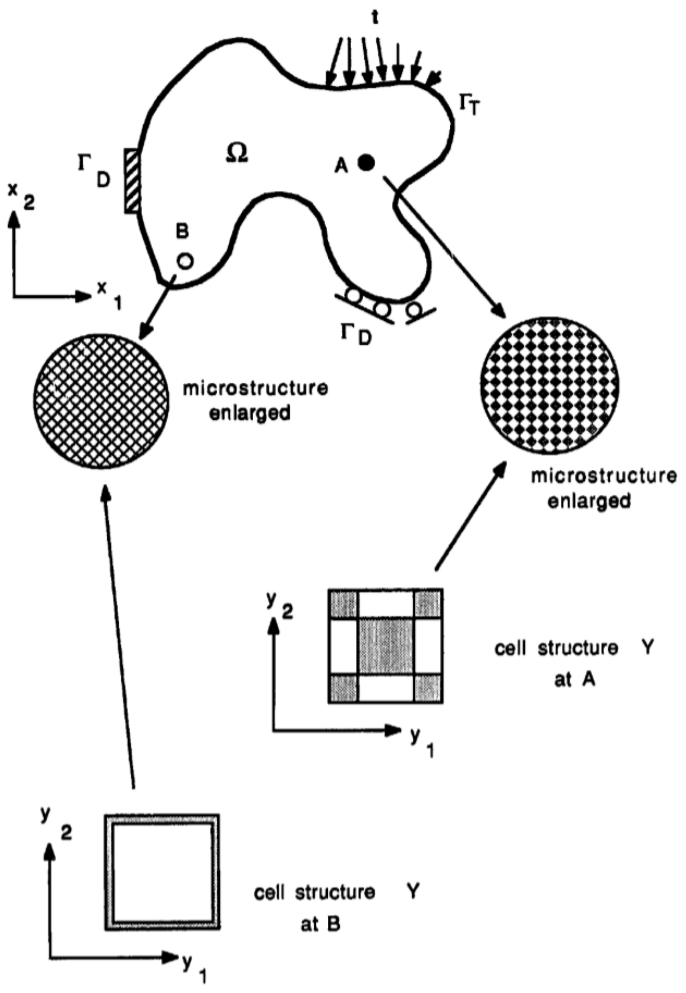


Figure 7 Homogenization topology optimization for continuum structures [27]

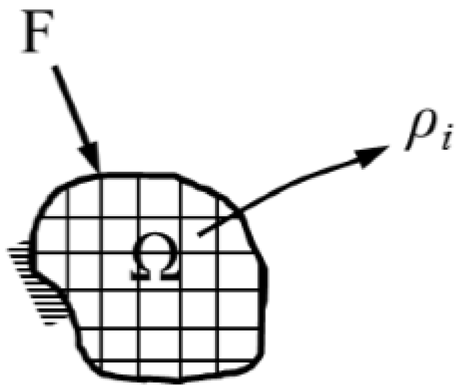


Figure 8 Topology optimization problem of a generalized shape

A typical SIMP optimization formulation for minimizing compliance has the following form:

$$\min C(\boldsymbol{\rho}) = \frac{1}{2}\sum_{i=1}^{N_e} \rho_i^p \mathbf{U}_i{}^T \mathbf{K}_i \mathbf{U}_i \tag{1}$$

Subject to

$$\rho_{min} \leq \rho \leq 1 \tag{2}$$

$$\mathbf{U}(\boldsymbol{\rho}) = \mathbf{K}(\boldsymbol{\rho})^{-\mathbf{1}}\mathbf{F} \tag{3}$$

$$\sum_{i=1}^{N_e} V_i \rho_i \leq V_{target} \tag{4}$$

Where C is the compliance of the structure, $\boldsymbol{\rho}$ is the vector of containing the relative densities ρ of the FE model, N_e is the number of FEM elements in the model, p is the penalty exponent of the SIMP formulation, $\mathbf{U}$ is the vector of nodal displacements, $\mathbf{K}$ is global stiffness matrix, ρ_{min} is the minimum relative density value to avoid any numerical instability in the FEM model, V_i is the volume of the FEM element, V_{target} is the target volume. The penalty exponent is typically chosen as $p = 3$ [32] to ensure that a design with distinct solid and void regions is obtained from the optimization process. However, the generated structures from topology optimization often resemble organic shapes that are often difficult if not impossible to manufacture with traditional manufacturing methods, which might also require shape optimization and manually interpreting the generated designs [33]. This has moved the attention towards AM to fabricate the generated structures. However, intermediate densities that still exist are converted into either a solid or void, which leads to differences between the optimization results and the fabricated part via AM as can be seen in Figure 9-b. This difference will cause changes in stress distribution in the part and would also violate the volume constraint, due to the conversion of elements into solids. Furthermore, even with AM technology, the organic shape of the generated structure can still pose a problem during fabrication. The overhangs present in the generated structure require support structures, which increase the build time, cost, and difficulty in post-machining (Figure 9-c) [34]. Moreover, three-dimensional parts can have enclosed voids that can trap support material or build materials, which are impossible to remove without damaging the fabricated part (Figure 9-d) [35]. Another issue with generated structures from topology optimization is that the obtained design is not optimal mathematically compared to using lower penalty values. This has directed attention to the use of lattice structures. Since lattice structures are open cells, they can be fabricated with AM without excessive use of support materials or creating enclosed voids that can trap build material inside. Furthermore, lattice structures allow the use of different penalty values in the optimization process, allowing for a more optimal solution compared to the solid counterpart [14]. This combination of topology optimization and lattice structures facilitated the design of structures known as Functionally Graded Lattice structures.

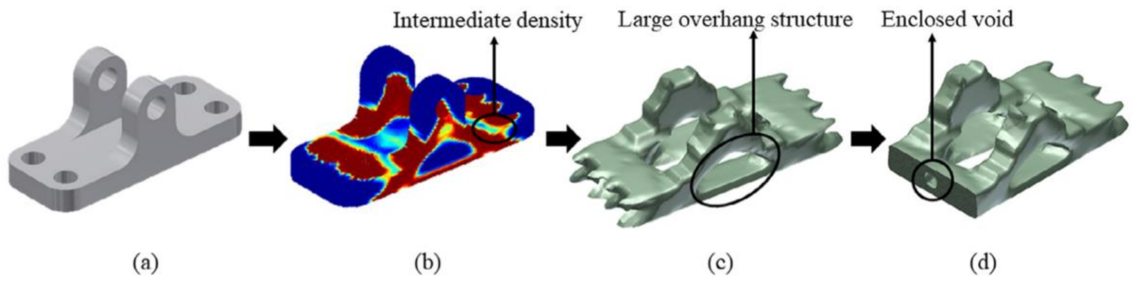


Figure 9 Structure optimization with SIMP method a) Design domain b) Optimization result c) Solid model obtained from optimization showing a large overhang structure d) Section view showing enclosed void [35]

1.3 Functionally Graded Lattice (FGL)

Functionally Graded Lattice (FGL) structures are structures where the density gradient of the solid material changes over the volume, which leads to changes in the mechanical properties of the lattice structure [36]. The change in density gradient can be due to changes in porosity, unit-cell size, or orientation. A similar gradient change can be seen in cellular structures in nature such as bones and flower stems as shown in Figure 10 [37-39], where the relative density changes based on function and location. The earliest use of FGL structures can be traced back to the homogenization optimization in [27], where the patterned composites that constitute the structure have varying relative density. But, as mentioned in section 1.2, the process was computationally expensive and the fabrication of such structures had its complications. But, with advances in AM technologies, it facilitated the fabrication of FGL structures with reasonable time and cost. Thus, directing researchers' attention into investigating the performance of FGL structures and their design and fabrication methods.

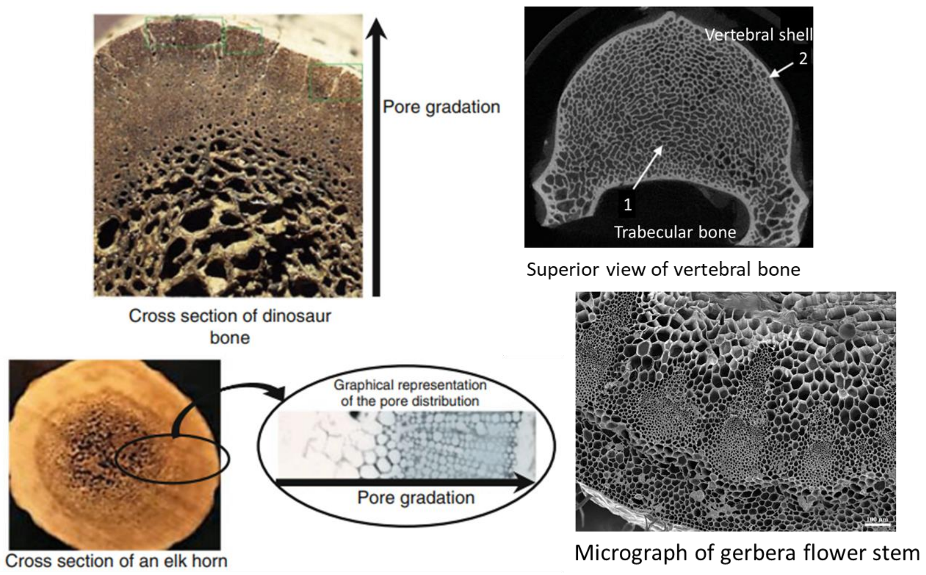

Pore gradation
Cross section of dinosaur bone
Vertebral shell
2
1
Trabecular bone
Superior view of vertebral bone
Graphical representation of the pore distribution
Pore gradation
Cross section of an elk horn
Micrograph of gerbera flower stem

Figure 10 Example of functionally graded cellular structures in nature [37, 38, 40]

Kalita *et al.* [41] used fused deposition modeling (FDM) to fabricate scaffolds with segments of varying relative density in the radial direction as shown in Figure 11. However, the relative density in each segment was determined manually by having the innermost segment of low density and the outermost of high density. This was due to the lack of models that relate the lattice grading to the mechanical requirements in the CAD systems that time to produce a continuous grading in the lattice [42].

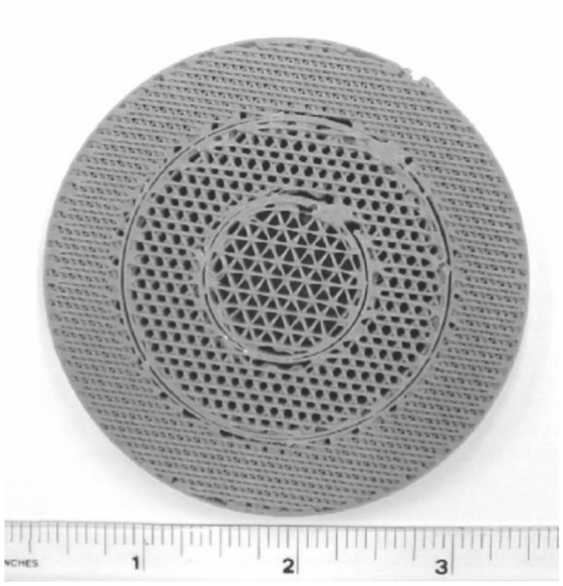


Figure 11 Scaffold design with varying radial porosity [41]

Burblies *et al.* [43] proposed the use of topology optimization and Selective Laser Sintering (SLS) to design and fabricate 2D FGL structures to mimic bone tissue porosity. The FE mesh elements were used as basic unit-cells and the porosity of each unit-cell was determined by mapping the relative density from the optimization results in Figure 12-a to each unit-cell in Figure 12-b. However, while the optimization method was similar to the SIMP optimization, it used discrete porosities that were correlated to the optimization relative density values.

Nguyen *et al.* [44] used FEM stress results in their Size Matching and Scaling method to generate FGL structures from a library of unit-cells where the diameters were pre-optimized based on the local stress states for each unit-cell. The method relies on the observation that the stress distribution in the lattice unit-cell will be similar to that of a solid FEM element of similar shape. This allowed the method to reduce the massive number of design variables into two variables, which are the minimum and maximum diameter values. Thus it can generate FGL structures with a reduced computational cost compared to using discrete topology optimization algorithms. Figure 13 shows the unit-cell library and its application on a Micro Air Vehicle (MAV).

The use of the SIMP method to create NFGL structures was proposed by Brackett *et al.* [45] in 2011, where the unit-cell porosity is treated as a continuous variable that is correlated to optimization results. This allowed for the relaxation of the penalty value during the optimization process, which allows for the generation of more optimal designs compared to using a penalty value of $p = 3$. Although a value of $p = 1$ is possible to use, most of the work in the literature uses a value around 2, which better correlates the stiffness of the unit-cell with the optimization results and agrees with Gibson and Ashby's work [1]. Figure 14 shows an example of mapping the SIMP results to an FGL structure for a cantilever beam subjected to a point load at the bottom corner end.

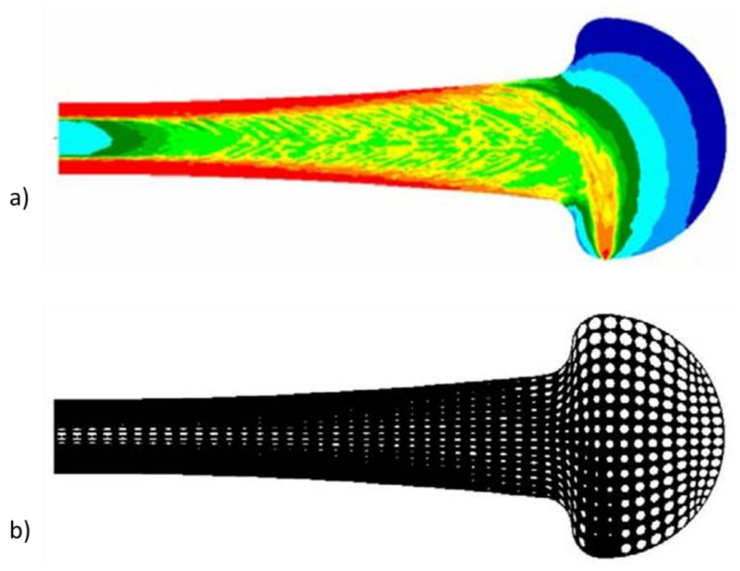


Figure 12 a) Topology optimization result under bending loading condition b) Generated FGL structure from optimization result mapping [43]

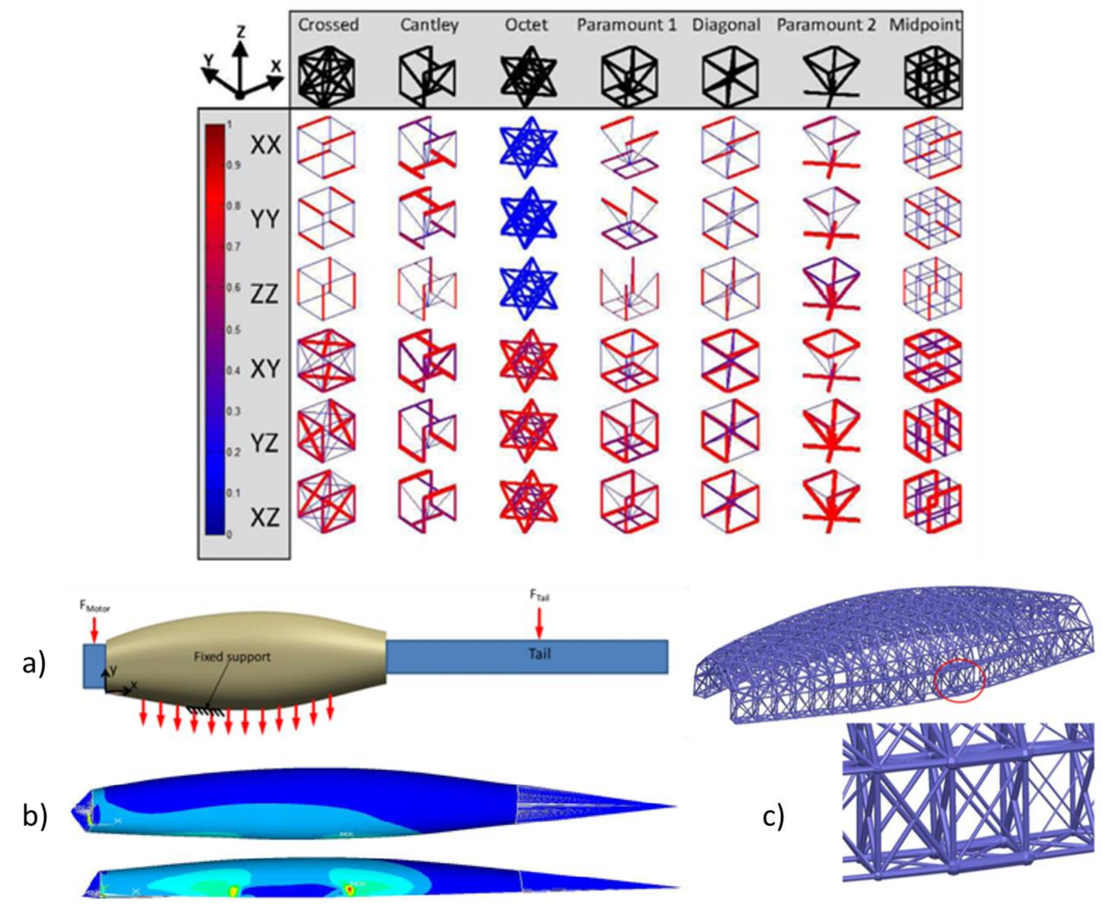


Figure 13 Size Matching and Scaling unit-cell library a) MAV loading conditions b) Stress field results c) Generated FGL structure [44]

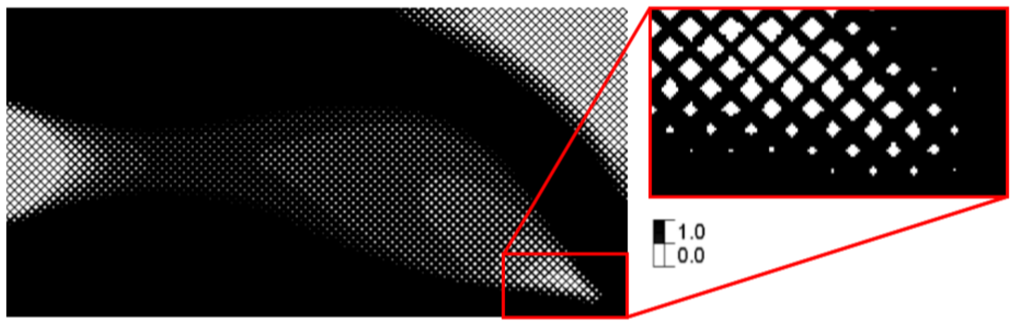


Figure 14 Generated FGL structure from mapping SIMP optimization results to unit-cell porosity [45]

In our lab [46], we developed the Relative Density Mapping (RDM) method, which used multi-objective SIMP optimization to map the relative density values to a lattice structure that is larger in scale to the FEM mesh used in the optimization. Unlike the work done in by Brackett *et al.*, the RDM method was not mapping the relative density values by an "element to unit-cell" basis, but rather from a collection of elements surrounding the structure's struts. The relative densities were also weighted based on their distance to the strut, thus reducing the influence of elements that are further away from the FGL strut on the strut's size. The FGL structure was shown to handle multiple loading conditions with a reduced computational cost in the design process. Figure 15 shows an application example of the RDM method.

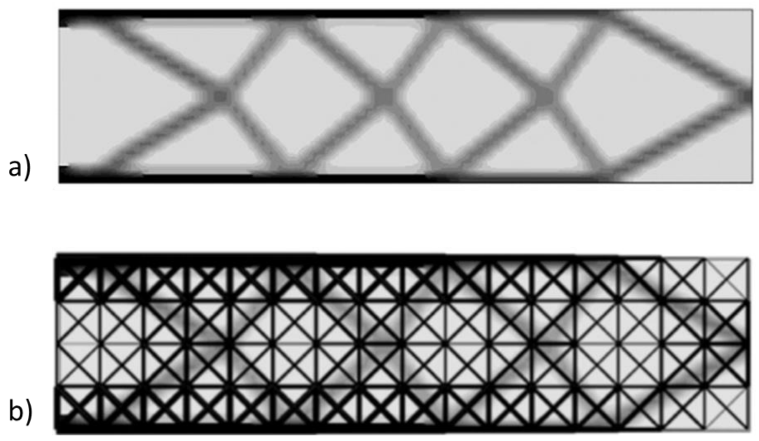


Figure 15 Application of RDM method a) SIMP optimization results b) Generated FGL structure [46]

The approach of using the SIMP method to generate FGL structures was then implemented by [14] in their software Optistruct. The FE mesh edges are replaced by beam elements with their diameters determined based on the optimization relative density values.

Panesar *et al.* [47] presented in their work different strategies for mapping SIMP optimization results to design lattice structures. In their work, they used the optimization results to generate a structure that is mixed between solid and uniform lattice, which they called intersected; a graded lattice structure; and scaled lattice that uses scaled porosity values compared to the graded structure as shown in Figure 16. However, the optimization was unpenalized, which caused discrepancies between the lattice stiffness and the optimization result. The solid structure in their work had the highest stiffness followed then by the intersected and graded structure but it required more supporting structures. The uniform lattice had the lowest stiffness of all lattice structures, which is similar to the findings in [48].

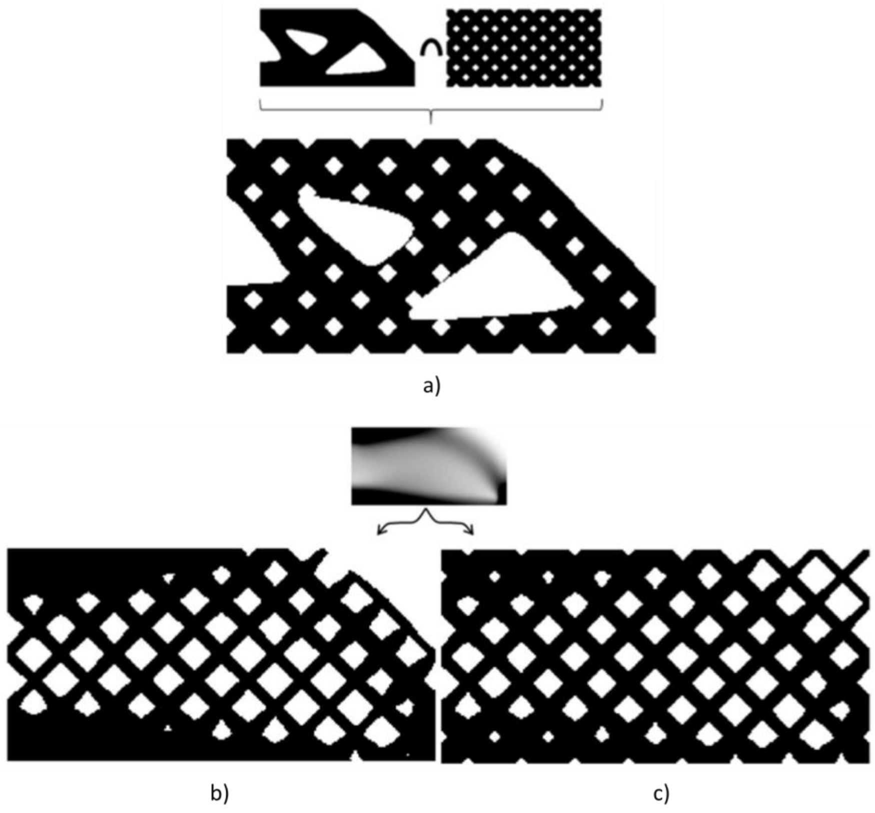
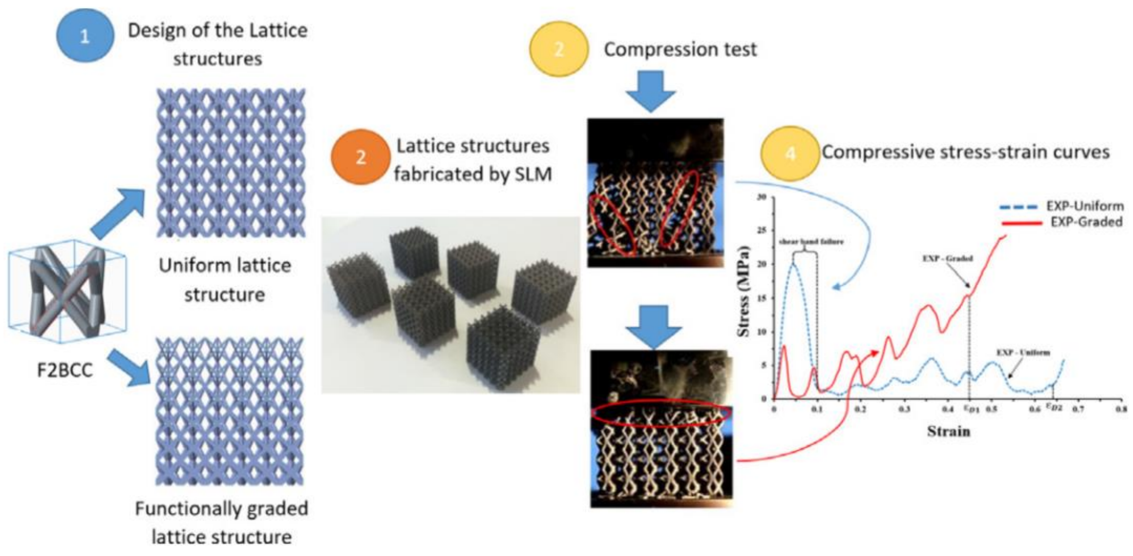


Figure 16 Different strategies to generate FGL structures a) Intersected b) Graded c) Scaled [47]

Other work in the literature used Asymptotic Homogenization (AH) to derive the material stiffness matrix of the FEM elements as a function of the unit-cell porosity and then utilizing it in the SIMP optimization to generate FGL structures [49-52]. This allowed for the optimization to be carried out without having discrepancies between the lattice properties and the optimization result, but it is limited to the unit-cell configuration that was homogenized and it increases the computational cost of the optimization process compared to the SIMP optimization [51, 53]. Further application of AH to design FGL structures can be seen in [54, 55], where FGL structures were used to optimize the cooling channels of injection molds. AH was used to derive the thermal and mechanical properties of stiffness and conductivity matrices for a cubic unit-cell. Similarly, the AH approach was deployed in [56-58] to design and fabricate different FGL unit-cell types. The approach was applied to design Face-Centered Cubic (FCC), Body-Centered Cubic (BCC), and Octet unit-cells in [56, 57] and to design Gyrod FGL structures in [58] and.

On the experimental work side, Maskery *et al.* [59] conducted an experimental investigation on the mechanical behavior of FGL structures compared to non-graded lattice structures. The experiments have shown that FGL structures are more favorable for energy absorption due to their predictable deformation behavior. The same findings were observed in [3, 36, 60, 61] when comparing FGL structures using different unit-cell configurations. The plateau stress was also notably higher in FGL structures when compared to their non-graded counterparts and showed better energy absorption as can be seen in Figure 17.



1
Design of the Lattice structures
Uniform lattice structure
F2BCC
Functionally graded lattice structure
2
Lattice structures fabricated by SLM
2
Compression test
4
Compressive stress-strain curves
EXP-Uniform
EXP-Graded
shear band failure
EXP - Graded
EXP - Uniform
Stress (MPa)
Strain

Figure 17 Experimental compression test on uniform lattice structures and graded lattice structures [60]

All the work mentioned above focused on FGL structures with a porosity gradient only while keeping the unit-cell size fixed. However, as seen in Figure 10, structures in nature tend to not only vary in porosity but also the size of the cells. Therefore, to differentiate between the research done on FGL structures with only porosity gradient and FGL structures with both porosity and unit-cell size gradient, we will use the name Naturally Functionally Gradient Lattice (NFGL) structures in this research to for FGL structures with both porosity and unit-cell size variation, since these structures are closer to functionally graded materials in nature.

1.3.1 *Naturally Functionally Graded Lattice (NFGL) Structures*

As mentioned in the previous section, the term NFGL will be used to denote FGL structures with both porosity and unit-cell size variation to differentiate between them and FGL structures with a porosity gradient only. In this section, a review of the research done on NFGL structures in the literature will be provided. Detailed on some of the work regarding this work will be provided in CHAPTER 2 to fully explain

A parametric approach using Function Representation (Frep) to create lattice structures with varying unit-cell sizes was proposed by Pasko *et al.* [62] in 2010 and was improved upon by Frayazinov *et al.* [63] in 2013. The unit-cell would be defined using a continuous real function inside the design domain and replicated using a replication function. This would allow for the change in porosity, unit-cell size, and even type as shown in Figure 18. However, this requires parametrizing the unit-cells and determining an appropriate replicating function for different design domains accordingly. This becomes an issue when dealing with multiple complex design domains especially when conformal unit-cells are required and can distort the unit-cells severely.

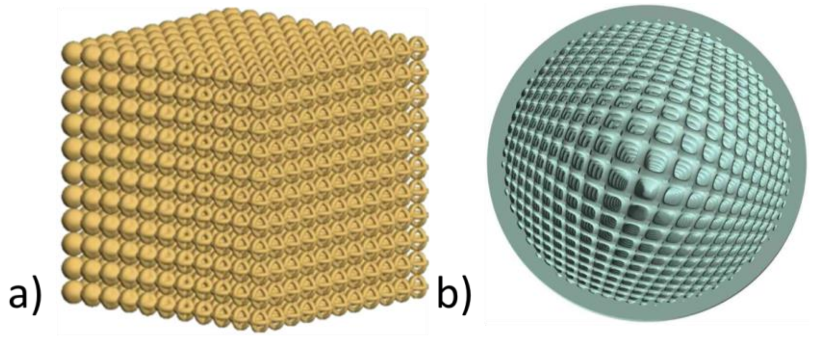


Figure 18 Generated NFGL structures using Function Representation a) Unit-cell type grading [63] b) Unit-cell size grading [62]

Brackett *et al.* [64] in 2014 used Error Diffusion methods [65] to dither the pixels of a density field input. These pixels then form the joints or nodes of the NFGL structure, which then generate the struts of the structure using Delaunay triangulation or Voronoi tessellation based on a density field input that represents some desired functional gradient as shown in Figure 19. But since the Error Diffusion method relies on the use of a filter of a fixed size, it limits the ability to freely set the maximum unit-cell size that can be generated within the structure. Furthermore, it requires a rectangular domain to dither the NFGL structure nodes.

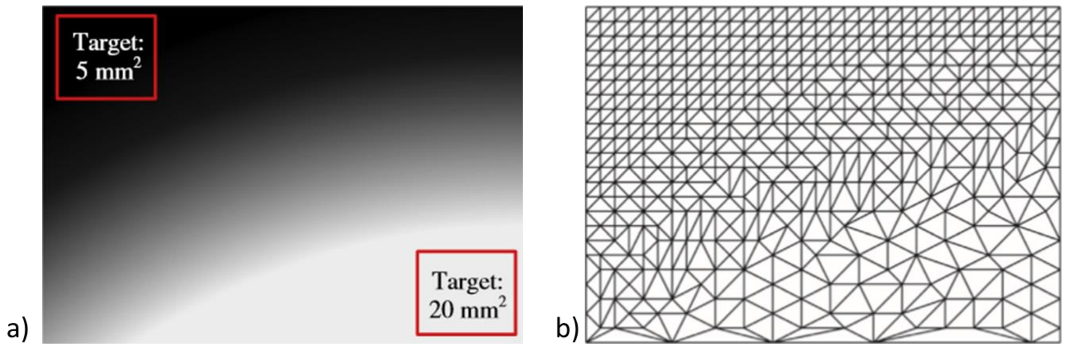


Figure 19 Application of the Error Diffusion method to generate NFGL structures a) Input density field b) Generated NFGL structure [64]

The Error Diffusion approach was also adapted in the work by Lu *et al.* [66] and Kuipers *et al.* [67]. In [66] the Error Diffusion method was used to create nodes that will act as sites for Voronoi foam cells based on a stress field input. The number of sites and the porosity of each site are then optimized to minimize the total weight of the structure under stress constraints. Figure 20 shows the steps used in generating and optimizing a foam structure. As for the work in [67], the design domain is first subdivided into cells of various sizes based on the required density distribution, then Error Diffusion is further utilized to reduce the discrepancy between the subdivided cells and the density distribution. The generated cells were then used to create space-filling surfaces accordingly which are then trimmed to create the foam structure of the design domain as shown in Figure 21. Both methods rely on creating foam structures that are trimmed in order to fit inside the design domain as seen in the figures.

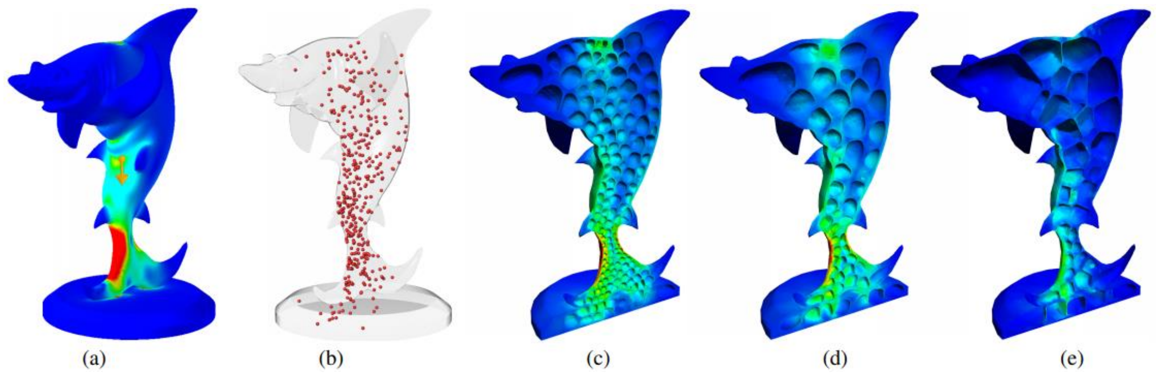


Figure 20 Generated structure using Error Diffusion a) Stress field b) Generated Voronoi sites c-d) Two steps of the optimization process e) Optimized graded foam structure [66]

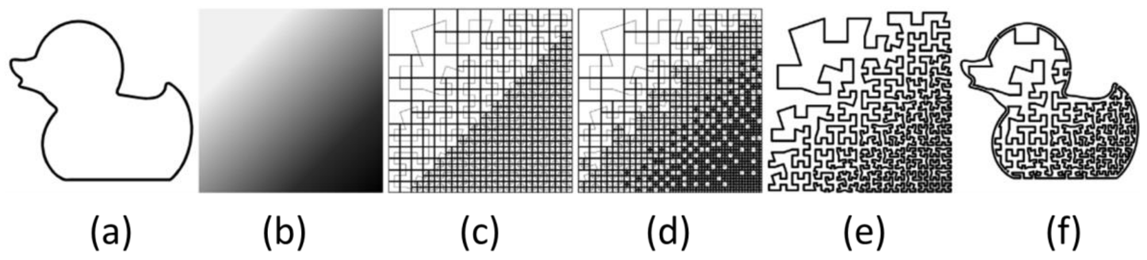


Figure 21 Generation of graded foam structure using Quadtrees and Error Diffusion a) Design domain b) Input density field c) Generated Quadtree structure d) Quadtree structure after Error Diffusion e) Generated space-filling structure f) Generated graded foam structure [67]

Martinez *et al.* [68] developed a method to use randomly distribute nodes inside cells based on a desired density field input to generate Voronoi foams as seen in Figure 22. Although the structure generated is considered a foam structure, the method can still be used to generate NFGL structures. The generated nodes are then used to generate the Voronoi cells, which edges will become the structure's struts. Unlike the Error Diffusion method, the generated structures with this method are not restricted in the unit-cell size generated. However, due to the randomness in the algorithm, each realization will have a deviation in the elastic modulus that was measured to be around 3.3%. This approach was then used in [68] to generate orthotropic foams by utilizing the stress field to control the orientation of the struts, and the density field from SIMP optimization to control the porosity and stretch of the generated cells. The struts are generated by connecting a node to its *k*-nearest neighbors in an asymmetrical manner controlled by the orientation and stretch desired at the node.

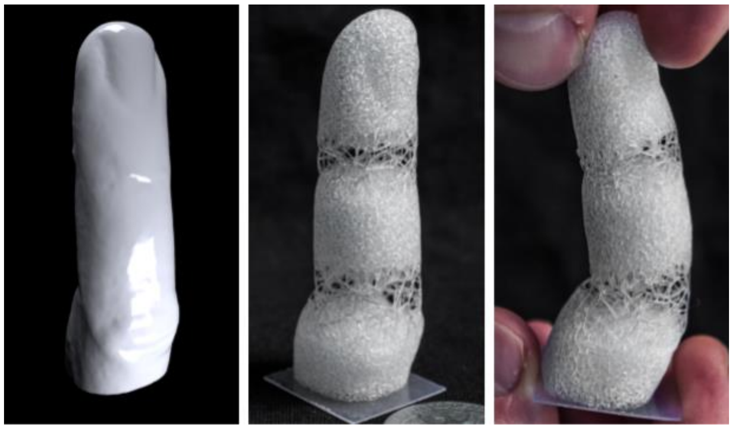


Figure 22 A model of a finger generated using randomized Voronoi foam [68]

Another approach, to generate Voronoi foams with varying unit-cell size, was proposed by Wang *et al.* [69]. But it also relies on randomness when distributing nodes inside the design domain. The nodes are generated in a uniform manner as shown in Figure 23, and then randomly displaced inside a spherical region, which radius is based on the required spatial variation. The nodes are then used to generate the Voronoi cells that would become the foam structure. However, the generation of the uniform nodes requires the distribution function to be known in order to generate the nodes. And this is not always available, especially when using distributions that are generated from FEM or topology optimization results, which limits the use of this method.

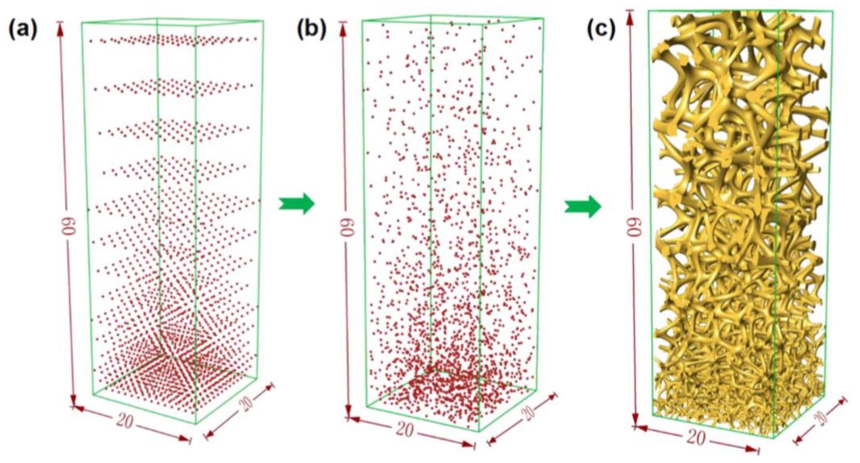


Figure 23 Generation of NFGL structure by random nodal displacement a) Uniform nodes b) Randomly displaced nodes c) Generated NFGL structure [69]

Wu *et al.* [70] used 2d extruded graded rhombic unit-cells as self-supporting infills. The generation of the structure was carried out by subdividing each rhombic unit-cell as needed. This allowed the method to generate size variation in each unit-cell but it required an optimization process using penalized SIMP optimization with additional constraints, which affects the computational cost. Figure 24 shows an example of an optimized structure using this method. The structure in Figure 24-a is the initial shape of the design domain. The unit-cells are then subdivided through the optimization process until the objective and constraints are minimized as shown in Figure 24-b.

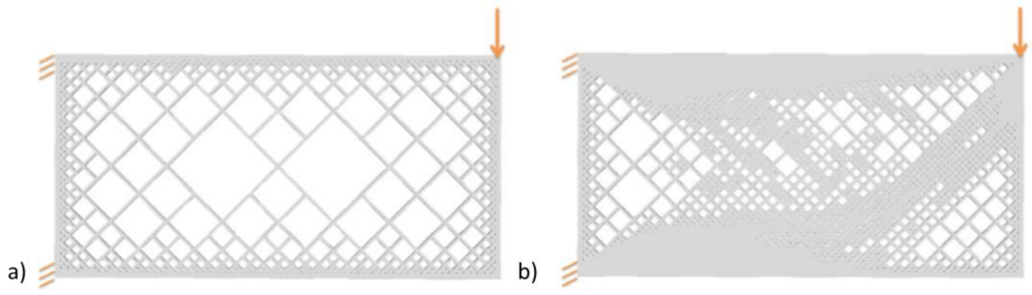


Figure 24 Structure optimization by using rhombic unit-cells a) Initial structure b) Optimized structure [70]

Another approach to create similar graded structures was proposed by Lee *et al.* [71] where ellipses were used instead of rhombic cells. The ellipses are created by first producing a Voronoi diagram of the cross-section with the highest area that is parallel to the build direction, and then placing a circle approximation of the ellipse at the vertex with the largest empty circle called the clearance probe. The Voronoi diagram is then regenerated locally with the new ellipse included and the process is then repeated until a required amount of ellipses is generated as shown in Figure 25. The ellipses are then extruded to fill the part that is being fabricated.

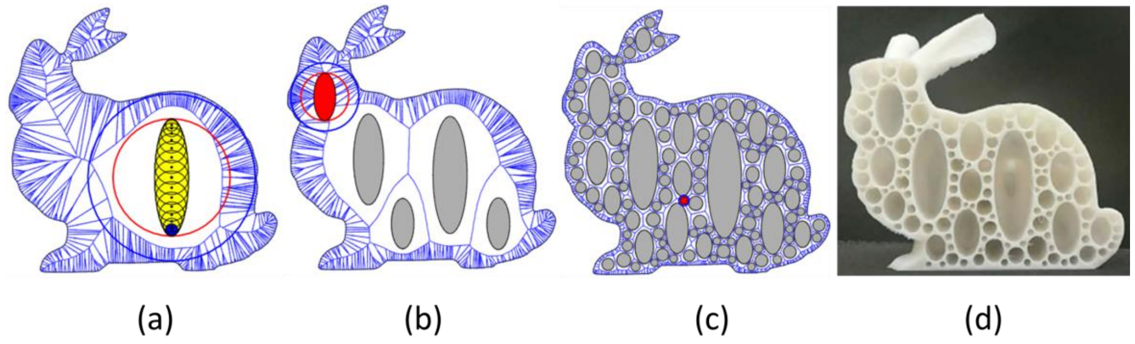


Figure 25 Generated hollowed foam structure a) Voronoi diagram with inserted ellipse b) Insertion of four more ellipses c) Voronoi Diagram with 100 ellipses d) Fabricated part [71]

Zhang *et al.* [72] designed orthopedic casts to be used for thermal comfort, by distributing Voronoi unit-cells across the surface. The distribution was also carried out through an optimization process to optimize the location of the cell centroid based on a thermal distribution map over the design surface. Figure 26 shows a comparison for the temperature difference between the cast with optimized unit-cell size and a uniform unit-cell cast. As shown in the figure, the cast with varying unit-cell size showed lower temperatures compared to the uniform cast.

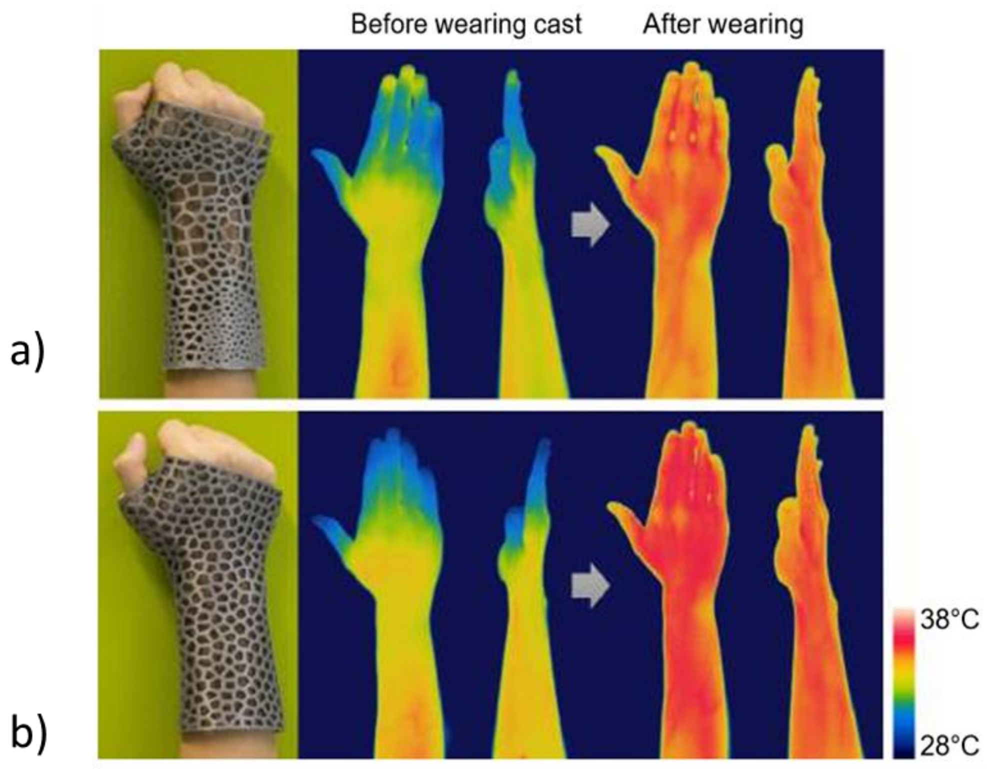


Figure 26 Skin temperature before and after wearing an orthopedic cast design) optimized cast b) uniform unit-cell cast [72]

Wu *et al.* [73] proposed a method to generate NFGL structures by adding a local volume constraint to the SIMP optimization algorithm that controls the percentage of solid material in a given region of elements. This allowed the method to create bone-like porous structures as shown in Figure 27. However, the largest unit-cell size is restricted by the size of the region of elements used to determine the percentage of solid material. Furthermore, the additional constraints increase the computational time for the optimization to converge.

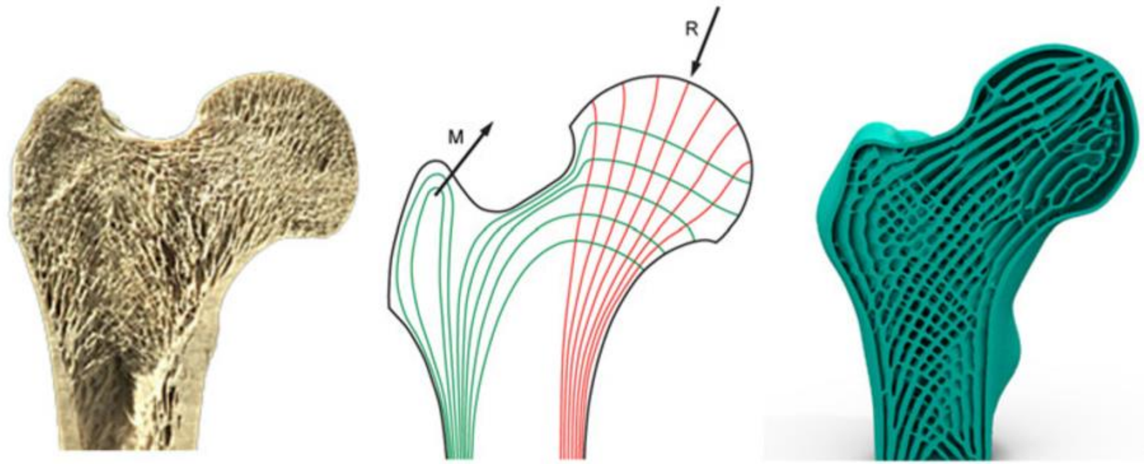


Figure 27 Generation of bone-like porous structure using local volume constraint optimization [73]

Wu has also proposed the Adaptive Quadtree optimization approach in [74], which focused on subdividing cells similar to the work in [70]. But it also relied on conducting a penalized SIMP optimization process with additional constraints. and the generated structure does not conform to the design domain surface.

As can be seen from the above literature, little work has been done in researching the performance of NFGL structures, and the work shown in the literature to generate NFGL structures suffers from multiple limitations. It either has a limit on the maximum unit-cell size, relies on randomness in the distribution of nodes, or require an optimization process to be carried out in order to generate structures, which increases the computational cost with the increase in design variables due to the need of conducting FEA in each iteration. However, the work in the literature that does require the use of optimization methods tend to generate NFGL structures with varying unit-cell sizes. This shows that NFGL structures generate structures with improved performance compared to FGL structures, which is apparent in natural materials also. However, the research done into generating these NFGL structures for use in practical applications and their performance compared to FGL structures is still limited and needs further exploration.

1.4 Research Objectives

The limitations of the existing methods in the literature in generating NFGL structures and the limited work on investigating the performance of NFGL structures against FGL structures limit the utilization of these structures in practical applications. The work in this research aims to overcome these limitations. Thus, the objectives of this research are:

- The development of a framework to design NFGL structures in a deterministic and computationally efficient manner.
- To provide the ability to create NFGL structures of varying unit-cell size ratios without strict limitations or restrictions.
- To provide better structural performances of NFGL structures against existing FGL structures.

1.5 Research Questions

In section 1.3.1, it was mentioned that some of the algorithms in the literature rely heavily on conducting an optimization algorithm to optimize the location of nodes in order to generate NFGL structures. This requires the use of FEA in each iteration of the optimization process, which increases the computational cost to generate the structure. Moreover, some of the algorithms rely on pseudo-random algorithms to generate seeds that are used to generate the structure nodes. This use of randomly generated nodes to generate the structure will produce a different structure with each execution of the algorithm, causing a difference in the properties of the generated structures. This difference will add to the computational cost when uncertainty quantification is needed for reliability analysis [75-77]. Furthermore, since the nodal placement is random, it will not generate conformal NFGL structures. To address these issues, the following research question is formulated and answered in this research.

Research Question 1:

How can we reduce the computational cost while controlling the randomness in nodal placements to generate conformal naturally functionally graded lattices?

Hypothesis 1:

If the NFGL nodal placement is determined based on a predetermined uniform grid of nodes that conforms to the design domain surface in a non-iterative manner, then the computational cost would be reduced and the NFGL structure would be generated in a deterministic way and conforming to the design domain.

Based on hypothesis 1, the algorithm that determines the placement of nodes in the design domain that will be used to generate the NFGL structure plays a significant role. To reduce the computational cost, the algorithm should not require the use of an optimization algorithm to adjust the location of nodes in each iteration. But it can utilize the information from a density field that can be generated from an unpenalized SIMP optimization or the solution of an FEA problem, such as temperature or stress distribution.

Using the solution of an unpenalized SIMP optimization would not cause a significant increase in the computational cost since the topology optimization problem would be reduced into a convex optimization [78], which is not as computationally expensive as penalized SIMP optimization. Furthermore, the same density field can be used to determine different nodal placements to generate different NFGL structures with varying unit-cell size ratios without having to regenerate a new density field for each structure, thus saving more computational time. Moreover, the number of elements used in the optimization would be would lower. Unlike optimizing the NFGL directly in each iteration, where the optimization process requires additional constraints, finer mesh, and penalization to ensure distinct solid/void regions in the other approaches that were proposed in the literature as discussed earlier. Thus increasing the number of design variables and computational cost immensely compared to just using the results from unpenalized SIMP optimization.

To control the randomness in the generated NFGL structure, the algorithm should use the information from the density field to determine the placement of nodes without relying on stochastic approaches. This can be achieved if the nodes were generated from a uniform grid of predetermined nodes in the structure. As stated in hypothesis 1, if the placement of these predetermined nodes conforms to the design domain, then the generated NFGL structure will also conform to the design domain. To generate such a gird, the use of FE mesh nodes that conform to the design domain is proposed. A similar concept, although doesn't work for non-rectangular design domains, can be seen in the work by Brackett *et al.* [64] where the density field image pixels were used to generate a uniform grid of nodes that are transformed into a grid of varying distances between the nodes using

Error Diffusion. However, the use of error diffusion has its limitations on the maximum unit-cell size ratio that can be generated. The filters used in the process [79] restricts the size of the maximum unit-cell size based on the filter size [80]. Using a larger filter will affect the smallest unit-cell size achievable. Furthermore, controlling the unit-cell size ratio requires modifying the input density field in an iterative process to reach the desired ratio. From these issues, we formulate the second research question as follows:

Research Question 2:

How can we remove the restriction on the unit-cell size ratio of NFGL structures in a computationally efficient way?

Hypothesis 2:

If a node can control the presence of other nodes adjacent to it from a normalized density field input rather than a shared effect of multiple nodes on the existence of a node, then the unit-cell size ratio can be adjusted.

From hypothesis 2, the proposed algorithm should allow the nodes in the FE mesh to control if other nodes should be adjacent to it or not, unlike when using a filter where the nodes alter the value of other adjacent nodes according to how the filter is set up. The adjacency to a node should also be controlled by the density field value at the node. A similar concept can be seen in adaptive meshing [81], where FE elements of varying sizes are generated based on the concept of ellipse packing [82]. Ellipses in 2D or ellipsoids in 3D of specific size and orientation are generated based on the size and anisotropy requirements in the elements. Then, these ellipses/ellipsoids are packed in an iterative process where the ellipses are moved to ensure no overlapping between them. The center of each ellipse/ellipsoid is then used to generate the nodes to create the FE mesh. In this manner, adaptive meshing controls the presence of nodes around each other, allowing for the generation of elements with different size ratios as needed that can be used to generate different unit-cell size ratios. However, the use of an iterative process to pack the ellipses/ellipsoids will incur additional computational cost each time the domain is remeshed and used to generate NFGL structures at different unit-cell size ratios. This led to the consideration of using sphere packing instead of ellipsoids in this research to reduce the computational cost associated with the orientation of ellipsoids. But this approach would also require the packing of spheres to be computed for each NFGL size requirements.

A simpler approach is proposed to be used in this research to eliminate the need to compute the spheres' packing process and further reduce the associated computational cost. The approach relies on utilizing the predetermined nodes that conform to the design domain, as explained in the requirements for the first research question. These nodes are then either removed or kept from the mesh based on their value in the normalized density field and the unit-cell size ratio needed. Each node will be treated as a sphere with a radius that is inversely proportional to the density value of the node. This will convert nodes of higher density values into spheres of smaller sizes compared to nodes of lower density values. If a node of lower density value is inside the sphere of a node of higher density value, then the node with the lower density value will be removed from the mesh. Once the process removes all the nodes that need to be removed, the resulting grid would be used to generate the NFGL structure struts. By varying the radii of the spheres, the unit-cell size ratio can be controlled without the need to regenerate a new mesh or through an iterative process that requires modifying the density field input for each desired unit-cell size ratio. CHAPTER 3 will provide a detailed explanation of the algorithms based on hypotheses 1 and 2.

With the proposed algorithm being able to generate NFGL structures at different unit-cell size ratios, the performance of the generated structure will vary. Algorithms that use optimization methods in each iteration tend to generate structures of varying unit-cell sizes. This shows that the variation in unit-cell size can indeed improve the performance of structures, which is also similar to materials in nature. But as the variation in unit-cell size increases, it can reach a point where it does not accurately represent the density field input. In this case, the performance is most likely to drop. From this issue, the third research question is formulated as follows:

Research Question 3:

How can we determine an appropriate unit-cell size ratio to improve the performance of NFGL structures to satisfy multifunctional requirements?

Hypothesis 3:

If we can quantitatively measure the similarity between the NFGL structure and the density field input that generated it, then we can correlate the variation in unit-cell size to the structural performance of the NFGL structure.

Based on hypothesis 3, there needs to be a way to correlate the generated NFGL structure to the input density field. Since the NFGL structure and density input are different in terms of the data that each represents, it is necessary to convert the NFGL into an approximate density field based on its unit-cell size ratio. The approximate density field should then be compared to the input density field to determine how similar the NFGL structure is to the input density field. To compare the two density fields, it is proposed in this research to utilize image quality assessment (IQA) methods [83]. Three methods were investigated, Root Mean Square (RMS), Peak Signal to Noise Ratio (PSNR) and Mean Structural Similarity (MSSIM) index [84]. Both RMS and PSNR fail to capture certain distortions in images compared to the MSSIM index. Images of different distortions could have the same RMS and PSNR error values even if some of these distortions do not cause a significant change in similarity to the original image. Furthermore, RMS error values are calculated such that the higher the error is, the higher the RMS value becomes. Similarly, PSNR error values are calculated such that the lower the error is, the higher the PSNR value becomes. So there is no direct relation between the error values and how similar two images are, since the error values are unbounded. This makes it difficult to measure the similarity between the density fields and investigate the effects of change in unit-cell size ratio on the performance of NFGL structures. As for the MSSIM index, the similarity is calculated as a value between 0 and 1, where 0 means that the images being compared are totally uncorrelated while 1 means that the images are exactly the same. Furthermore, since the values are bounded between 0 and 1, this facilitates the investigation of the effects of change in unit-cell size ratio for different geometries on the performance of NFGL structures.

Since the MSSIM index was developed to be used on pixels, the method will be improved upon in this research to deal with voxels that represent the density fields in 3D. Furthermore, the MSSIM index will be modified to deal with the density fields of different geometries. A detailed explanation of the improvement to the MSSIM index will be provided in CHAPTER 3.

1.6 Dissertation Organization

The organization of the chapters of this dissertation is shown in Table 1. Chapter 1 provides an introduction to cellular structures and topology optimization methods. A literature review of FGL and NFGL structures is also provided in the chapter. The chapter also outlines the objectives of this research and the research questions and hypothesis.

Table 1 Organization of the Dissertation

Chapter 1	Introduction
Chapter 2	Current State of the Art
Chapter 3	Naturally Functionally Graded Lattice (NFGL) Framework
Chapter 4	Application Examples
Chapter 5	Conclusions and Future work

Chapter 2 outlines the currently existing methods that can generate NFGL structures and provides a detailed explanation of each method and how they generate NFGL structures along with the advantages and drawbacks of each method. The chapter then gives a summarized discussion on the drawbacks of these methods and what are the expected advantages of this research in generating NFGL structures.

Chapter 3 will provide a detailed explanation of the framework that will be developed to generate NFGL structures in this research. Each algorithm that is involved in the development of the framework will be discussed in detail along with an example showing how each algorithm work.

Chapter 4 will demonstrate the application of the developed framework in three examples. The first example will compare the developed framework with the methods in chapter 2 to evaluate the performance of the framework. The second example will evaluate the performance of the developed framework against an FGL structure on the design and optimization of an automotive control arm. The third example will also evaluate the performance of the developed framework against an FGL structure but on a thermomechanical problem.

In chapter 5, concluding remakers will be provided by addressing the research questions and determining how the work, done in the research, answers them, and a list of the contributions this research is providing. Then, a list of the areas on which further future work can be explored will be presented.

CHAPTER 2. CURRENT STATE OF THE ART

In this chapter, the common methods that are used to generated NFGL structures will be discussed in detail and their advantages and drawbacks will be outlined. These methods are the Error Diffusion method by Bracket *et al.* [64], Stochastic Nodal Generation by Martinez *et al.* [68], Local Volume Constraint Optimization by Wu *et al.* [73], and Adaptive Quadtree Optimization by Wu *et al.* [74]. These methods were chosen since the other methods in the literature build upon them or are similar in concept to them and can be used to generate NFGL structures and not just foams. The methods were named based on the procedure used in order to generate NFGL structures.

2.1 Error Diffusion

The basic idea of this method is to use dithering to generate NFGL structure nodes as shown in Figure 28. The figure shows an input density field (Figure 28-a) and the generated NFGL nodes (Figure 28-b) using Error Diffusion to dither the pixels of the input density field.

To generate the NFGL nodes, an array representing the values of the input density field from 0 to 255 is generated. The values in the array, p, are then compared to a predefined value, t, as follows

$$b_{i,j} = \begin{cases} 255 & if\ p_{i,j} > t \\ 0 & otherwise \end{cases} \tag{5}$$

where b represents the values in a separate array that determines if pixel i, j should be placed or not based on b values (255 = white pixel and 0 = black pixel). Once the value of b for a pixel is determined, an error term, e, is calculated as

$$e_{i,j} = p_{i,j} - b_{i,j} \tag{6}$$

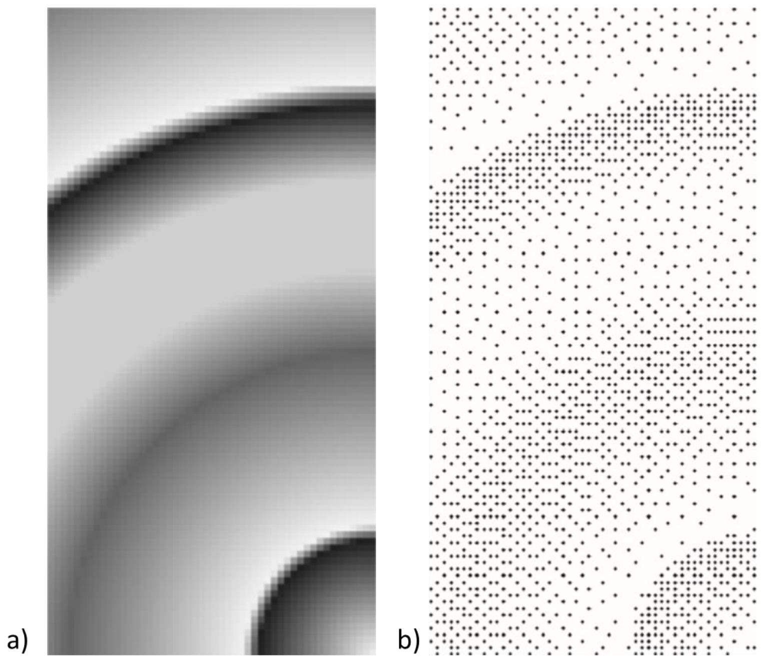


Figure 28 Generation of NFGL nodes using Error Diffusion a) Input density field b) Generated NFGL nodes [64]

The pixels adjacent to p are then modified based on the value of e by diffusing the error to the adjacent pixels, hence the name of the method. The diffuse process is done by applying a filter on the adjacent pixels as follows

$$\begin{bmatrix} p_{i+1,j} \\ p_{i+1,j+1} \\ p_{i,j+1} \\ p_{i-1,j} \end{bmatrix} = \begin{bmatrix} p_{i+1,j} \\ p_{i+1,j+1} \\ p_{i,j+1} \\ p_{i-1,j} \end{bmatrix} + e_{i,j} \begin{bmatrix} f_{i+1,j} \\ f_{i+1,j+1} \\ f_{i,j+1} \\ f_{i-1,j} \end{bmatrix} \tag{7}$$

Where f is a fraction determined by the filter used. The filter used was the one proposed by Floyd and Steinberg [79] which is shown in Figure 29. The error diffusion process continues until all the pixels in p are used and all the values in b are calculated. An example of this process is shown in Figure 30 for a 5×5 grid.

	$p_{i,j}$	7/16
3/16	5/16	1/16

Figure 29 The two-dimensional filter proposed by Floyd and Steinberg

As for the boundary pixels, a one-dimensional Error Diffusion is applied. This is done by using the same filter but without diffusing the error values to the pixels that are not on the boundary. Once all NFGL nodes are generated, the NFGL structure is created using either Delaunay triangulation or Voronoi tessellation. Figure 31 shows the generated NFGL structure using Delaunay triangulations for the NFGL nodes shown in Figure 28.

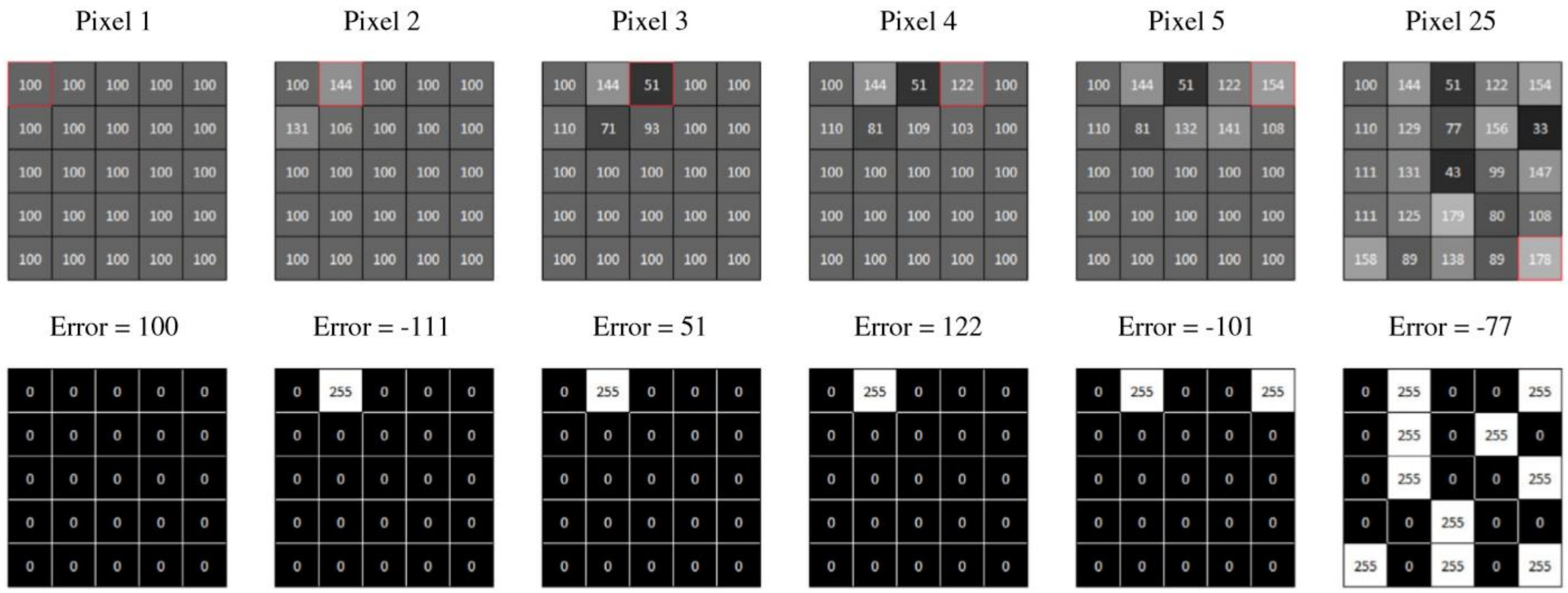


Figure 30 An example of the dithering process using the Error Diffusion method for a 5×5 grid of pixels [64]

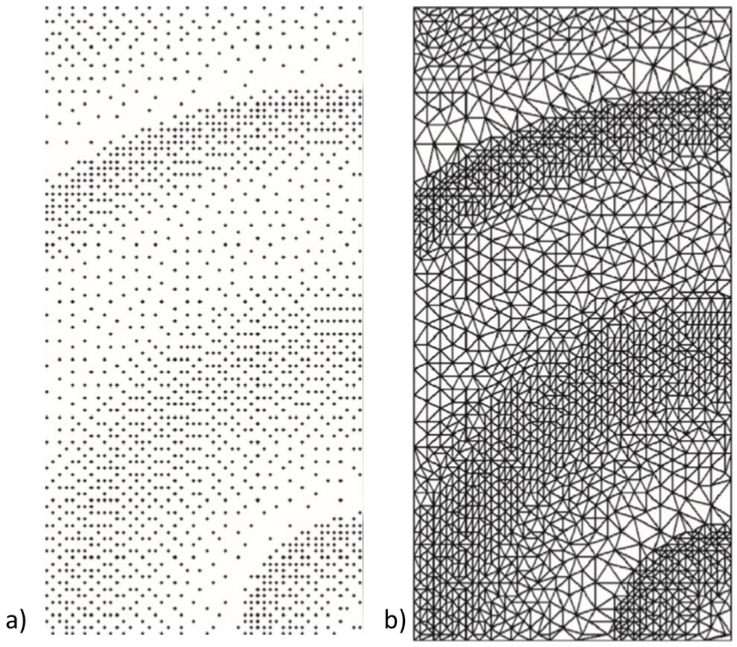


Figure 31 Generated NFGL structure from Error Diffusion method a) NFGL nodes b) Generated NFGL structure using Delaunay triangulation [64]

Based on the steps that the Error Diffusion method conducts, it can be apparent that there are drawbacks to the method. The first drawback is that it requires an image of the input density field to use its pixel values, which means that the design domain has to be rectangular. If a non-rectangular design domain was used, it has to be placed into one so that it can be used. The second drawback is that the generated NFGL structures will not be conformal if the design domain is not rectangular, which will be shown later in section 4.1.4. The third drawback is that there is still a restriction on the maximum unit-cell size ratio that can be achieved by this method. The size of the unit-cells is affected by pixel density used to represent the input density field and on the values of the density field. Increasing the pixel density would create smaller unit-cells, but would also reduce the size of the large unit-cells. Adjusting the input density values would increase the large unit-cells size, but the increase is limited due to the size of the filter. Figure 32 shows an example of how the unit-cell edge length is affected as the input density field values are increased. The values were changed from 0 to 240 as the method produces empty spaces when the value is close to 255.

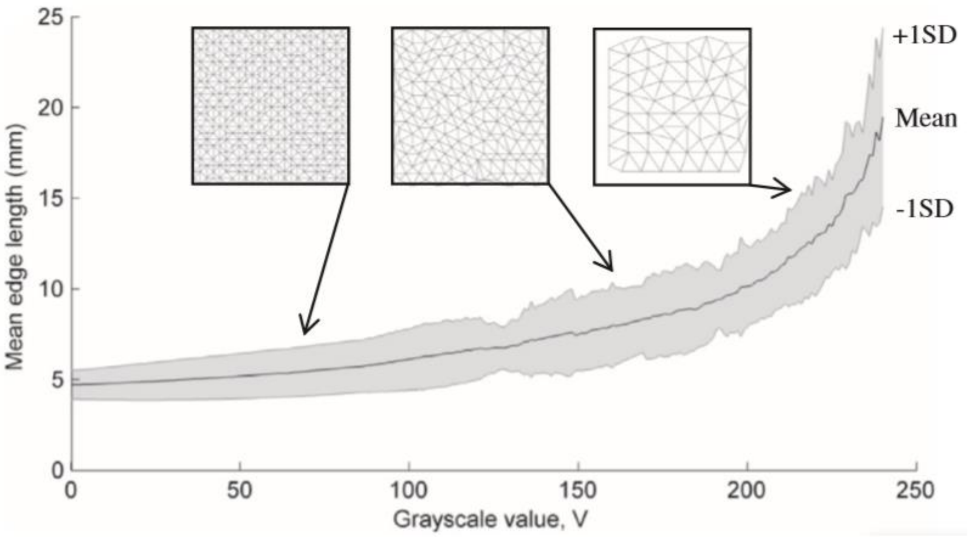


Figure 32 Change in mean unit-cell edge length as the input density field values are increased [64]

The advantages of this method can be seen in its low computational cost, as it doesn’t conduct an iterative optimization algorithm when generating NFGL nodes. The second advantage is that it can deal with any type of density field input as long as it is in a rectangular domain. The method will place the nodes through the design domain in a manner that captures the input density distribution requirement.

2.2 Stochastic Nodal Generation method

This method was used to generate Voronoi foams in the literature. However, it could be adjusted to generate NFGL structures by triangulating the generated node using Delaunay triangulation just like in the Error Diffusion method. The key idea of this method is to subdivide cells based on an input density field. Unlike the Error Diffusion method, the Stochastic Nodal Generation method generates its own nodes rather than using pre-placed nodes and turning them on or off. However, the nodes are generated in a stochastic manner. The node generation process starts by first determining an input cell size, l, and center, s, that will be used in an input density function, ρ_f. If the value of $l^2 \times \rho_f(S)$ is greater than 2^2 in a two-dimensional case, or $l^3 \times \rho_f(S)$ is greater than 2^3 for a three-dimensional case, the cell is subdivided and the values of l and S are changed to reflect the new cell size and center. This process carries on until the condition is met. Once the condition is met, nodes are placed randomly inside random subdivisions of the currently selected cell. The higher the density value, the more subdivisions are needed to meet the required condition.

An example of this is shown in Figure 33. The input density field is shown in Figure 33-a, while Figure 33-b shows the cell subdivisions based on the input density field. It is clear in the figure that the cell has more subdivisions in regions of higher density compared to regions of lower density. Figure 33-c shows the generated nodes that were placed in random locations inside of each subdivision while Figure 33-d shows the generated NFGL structure from the nodes.

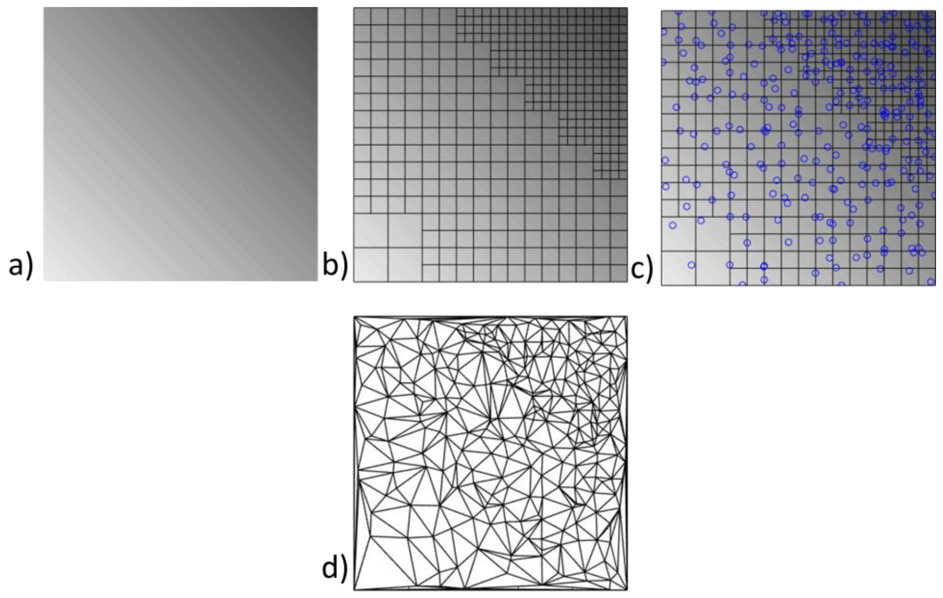


Figure 33 Generation of NFGL nodes using Stochastic Nodal Generation method a) Input density field b) Cell subdivision c) generated NFGL nodes

The stochastic nature of the method adds uncertainty to the generated NFGL nodes location as can be seen in Figure 34. Although this behavior could be modified in this method, it will still be included in order to address the issues associated with random nodal placement, since there are other methods in the literature that rely on stochastic means to generate NFGL nodes. This would help in addressing the other issues that this method exhibits.

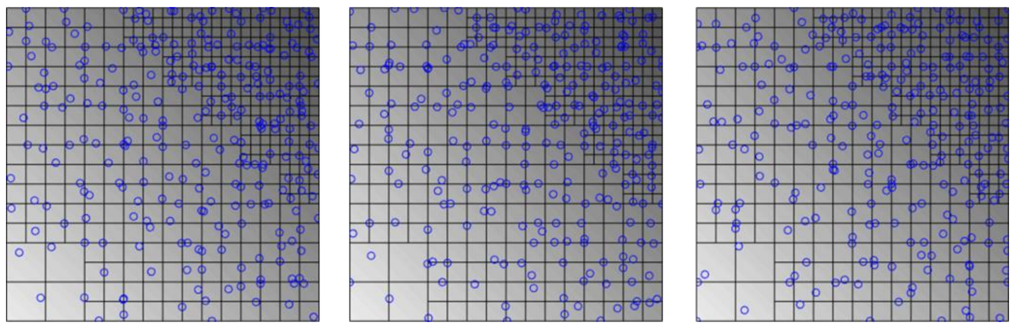


Figure 34 Different NFGL nodes generated with each iteration of the Stochastic Nodal Generation method

Based on the description of the method, a couple of drawbacks arise when generating NFGL structures. The first, as stated earlier, is that each iteration of the method can generate structures with different NFGL nodes. This difference causes deviation in the mechanical properties of the generated structure which can reach 3.3%, which when added to uncertainties from manufacturing can add more. The second issue is that the generated structures will be conformal when creating NFGL structures with Delaunay triangulations. Using Voronoi tessellations will create foam structures that are bending dominant, which are weaker as explained in section 1.1 regardless of the node placement, whether it is random or not. The third and major issue is the dependence of the method on the scale of the design domain. If condition to place nodes was met without the method adequately subdividing the cell based on the input density field, the method would generate nodes that do not represent the input density field at all. This can happen in small-scaled problems where small values of l can create values that are less than 2^2 or 2^3. Further discussion on this issue will be discussed in details in section 4.1.3

As for the advantages of the method, the first is its low computational cost. This related to the fact that the method does not require expensive optimization processes when placing the nodes inside the design domain. The second advantage is that it generates its own nodes without the need to have an initial grid of nodes.

2.3 Local Volume Constraint Optimization

This method aims to generate bone-like structures that can be used as infills for 3D printed parts by introducing additional constraints to the SIMP optimization method. The additional constraints aim to restrict the local material accumulation in a region surrounding an element. The optimization formulation is as follows

$$\min C(\boldsymbol{\Phi}) = \frac{1}{2}\mathbf{U}^T\mathbf{K}\mathbf{U} \tag{8}$$

Subject to

$$\mathbf{U}(\boldsymbol{\Phi}) = \mathbf{K}(\boldsymbol{\Phi})^{-\mathbf{1}}\mathbf{F} \tag{9}$$

$$\Phi \in [0,1] \tag{10}$$

$$g_1(\boldsymbol{\Phi}) = \frac{\left(\frac{1}{n}\sum_{i=1}^{M_e}\rho_i^{p_n}\right)^{\frac{1}{p_n}}}{\alpha} - 1 \leq 0 \tag{11}$$

$$g_2 = \rho_{avg} - \alpha_{total} \leq 0 \tag{12}$$

where Φ is a continuous design variable that is projected to the relative density ρ of the elements, $\boldsymbol{\Phi}$ is the vector containing Φ values, M_e is the group of elements surrounding element i based on a certain distance, p_n is p-norm exponent, α is the total volume ratio limit for the elements in M_e, ρ_{avg} is the average relative density of the design domain, α_{total} is the limit on the volume ratio of the design domain.

The projection of Φ unto ρ is done in order to ensure a 0-1 value for ρ by first applying a filter to Φ to prevent checkerboard patterns.

$$\widetilde{\Phi}_i = \frac{\sum_{\mathrm{j}=1}^{M_e} \omega_j \Phi_j}{\sum_{\mathrm{j}=1}^{M_e} \omega_j} \tag{13}$$

where $\widetilde{\Phi}_i$ is the filtered value of Φ for element i, ω is the weight of the effect of element j on element i.

After the filtering of Φ is done, the value of $\widetilde{\Phi}$ is projected unto ρ as follows

$$\rho_i(\widetilde{\Phi}_i) = \frac{\tanh\left(\frac{\beta}{2}\right) + \tanh\left(\beta\left(\widetilde{\Phi}_i - \frac{1}{2}\right)\right)}{2\tanh\left(\frac{\beta}{2}\right)} \tag{14}$$

where β is a parameter that controls how sharp ρ value changes from 0 to 1 (chosen as 16). Once the value of ρ is determined, it is penalized in a similar manner to Eq. 1 to ensure that intermediate values are pushed further towards 0 and 1.

The constraint in Eq. 11 controls the local volume in the elements in M_e to ensure that it doesn't go over the local volume ratio limit α while the constraint in Eq. 12 ensures that the total volume of the design domain doesn't exceed the total volume ratio limit a_{total}. This is done by summing up the volume ratio of all elements as

$$\rho_{avg} = \frac{\sum_{i=1}^{N_e} \rho_i v_i}{\sum_{i=1}^{N_e} v_i} \tag{15}$$

Where v_i is the element's volume. It should be noted that the summation is done over all of the elements, N_e, in the design domain. If the constraint in Eq. 12 was ignored, the generated structure will strictly impose the local volume ratio constraint, which will set an upper limit value of α.

After setting up the optimization problem and solving it, the generated design would have a bone-like shape that resembles NFGL structures. Figure 35 shows three generated structures after the optimization with different α values without imposing the constraint in Eq. 12. The structures were generated using a radius value of $R = 6$. The size of the cells in the structure does show a variation across the design domain. As the value of α reduces, the porosity in the structure increases as expected. However, the restriction on the local volume ratio forces the structure to create porosities across even when not needed. This can reduce the performance of the structure since the material cannot accumulate in locations where it can strengthen the structure to form thicker struts. Increasing the radius to include more surrounding elements helps in increasing the size of the generated cells and creating thicker struts as shown in Figure 36. However, it will cause more material to be forced into locations where it's not needed to satisfy the constraint on α.

As for the total volume constraint of the generated structure, imposing the constraint in Eq. 12 allows the generated structure to satisfy the total volume constraint as shown in Figure 37-b and c while trying to maintain the local volume constraint.

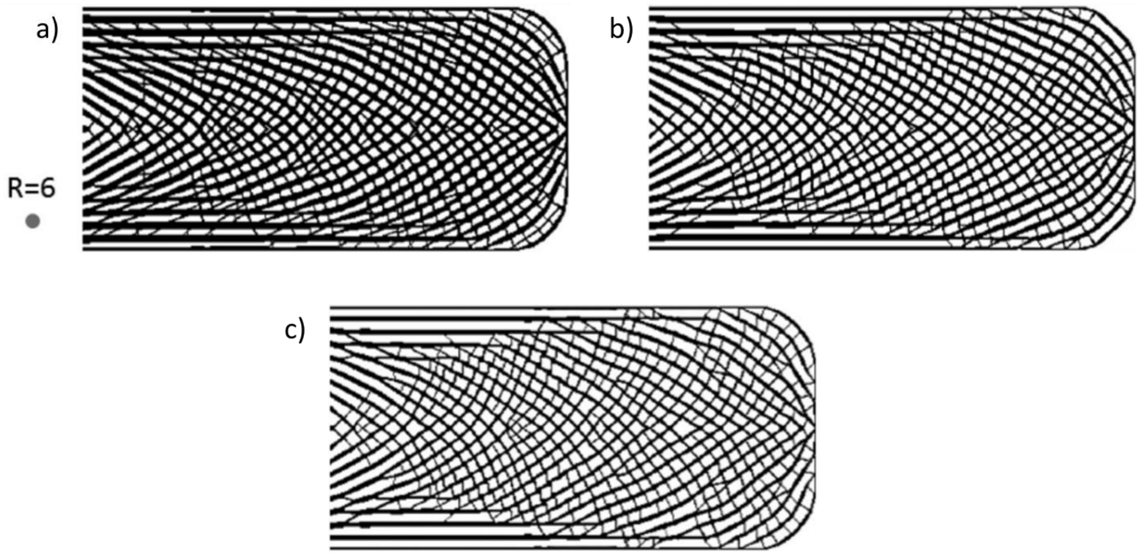


Figure 35 Generated structure using Local Volume Constraint at different local volume ratios with a radius of 6 a) $\alpha = 0.6$ b) $\alpha = 0.5$ c) $\alpha = 0.4$ [73]

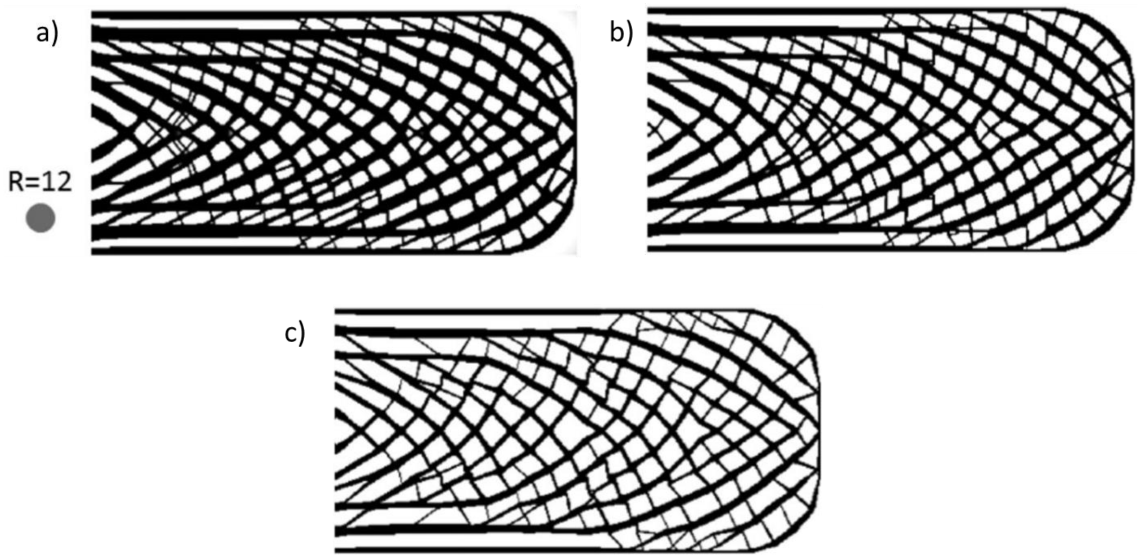


Figure 36 Generated structure using Local Volume Constraint at different local volume ratios with a radius of 12 a) $\alpha = 0.6$ b) $\alpha = 0.5$ c) $\alpha = 0.4$ [73]

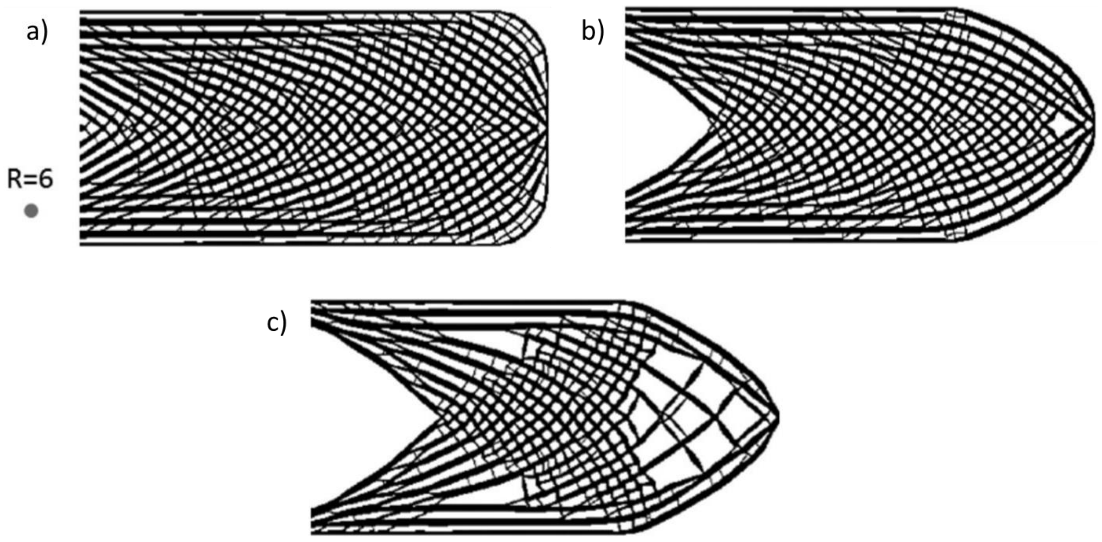


Figure 37 Generated structure using Local Volume Constraint at different local volume ratios with a radius of 6 and $\alpha = 0.6$ a) No constraint b) $\alpha_{total} = 0.50$ c) $\alpha_{total} = 0.4$ [73]

Based on the above description of the method, some drawbacks can be pointed out. The first is that the method requires topology optimization in each iteration when generating the NFGL structure. With the additional constraints and increased nonlinearity of the problem, the computational cost of this approach increases significantly. Furthermore, when creating different designs at different sizes, the topology optimization needs to be rerun again, thus increasing the computational cost further. The second drawback that was pointed out earlier is that the local volume ratio, while it helps in generating NFGL structures, limits the accumulation of elements in areas where thicker struts are needed. This can reduce the performance of the generated structure as thinner struts would be generated in areas where it would require more material. This also causes a restriction on the unit-cell size ratio across the structure.

There are a couple of advantages to this method for creating NFGL structures. The first is the ability of the method to generate structures are conformal to the design domain.

The second advantage is the ability to control the orientation of the struts. The second advantage is that the generated structures are not limited to a certain cell type. The generated cells and connectivity between them are generated through the optimization process.

2.4 Adaptive Quadtree Optimization

This method shares some similarities with the Stochastic Nodal Generation method, where the design domain is subdivided to generate the NFGL structure. The key difference is that the edges of the subdivided cells are what constitutes the NFGL struts and nodes, rather than placing the nodes inside the cells. Another difference is that the subdivisions are caused by a modified SIMP topology optimization process rather than relying on an input density field.

The modification of the SIMP optimization is done through introducing a design variable that controls the subdivision of cells in the design domain as follows

$$\min C(\boldsymbol{\rho}) = \frac{1}{2}\mathbf{U}^T\mathbf{KU} \tag{16}$$

Subject to

$$\mathbf{U}(\boldsymbol{\rho}) = \mathbf{K^{-1}F} \tag{17}$$

$$\chi_{i,j}^{k} \in [0,1], \qquad k = 1,2 \ldots \bar{k} \tag{18}$$

$$\sum_{i=1}^{N_e} V_i \rho_i \leq V_{target} \tag{19}$$

Where $\chi_{i,j}^{k}$ is the continuous variable that controls the subdivision of cells i and j at subdivision level k, $\bar{k}$ is the maximum allowable subdivision is related to the size of a $2^m \times 2^m$ square FEM mesh used in the design domain as

$$\bar{k} = m - 2 \tag{20}$$

Figure 38 shows an illustration of a rectangular design domain showing the cells and subdivisions with a different color for each level and the underlying FEM mesh generated. It is apparent that this method imposes a restriction on the number of FEM elements that will be used in the design domain to allow for appropriate subdivisions to be included. Based on Eq. 20, the minimum size of the FEM mesh to be used with only one level would be 8×8.

To perform the optimization process, the design variable $\chi_{i,j}^{k}$ has to be mapped to the relative density of the FEM elements ρ. The mapping is doing by using a sparse transformation matrix $\boldsymbol{T}^k$ of size $2^{2m} n_i^0 n_j^0 \times n_i^k n_j^k$, where n_i^0 and n_j^0 are the number of cells before subdivisions in the i and j direction as already shown in Figure 38, which in this case would be $n_i^0 = 4$ and $n_j^0 = 2$; and n_i^k and n_j^k are the number of cells at the k_{th} level in the i and j direction (for $k = 2, n_i^2 = 8$ and $n_j^2 = 4$). The values of ρ are then calculated as

$$\boldsymbol{\rho} = \sum_{k=0}^{\bar{k}} \boldsymbol{T}^k \boldsymbol{\chi}^k \tag{21}$$

As apparent from the equation, there's no influence on a cell from its parent cell. So if the optimization was to be carried out as is, the generated structure would end up with many suspended struts as shown in Figure 39. To refine the structure, a filtering process can be applied as follows

$$\tilde{\chi}_{i,j}^{k} \approx \left(\frac{1}{k}\sum_{l=0}^{k-1}\left(\chi_{i-1,j-1}^{k-l}\right)^{p_n}\right)^{\frac{1}{p_n}} \tag{22}$$

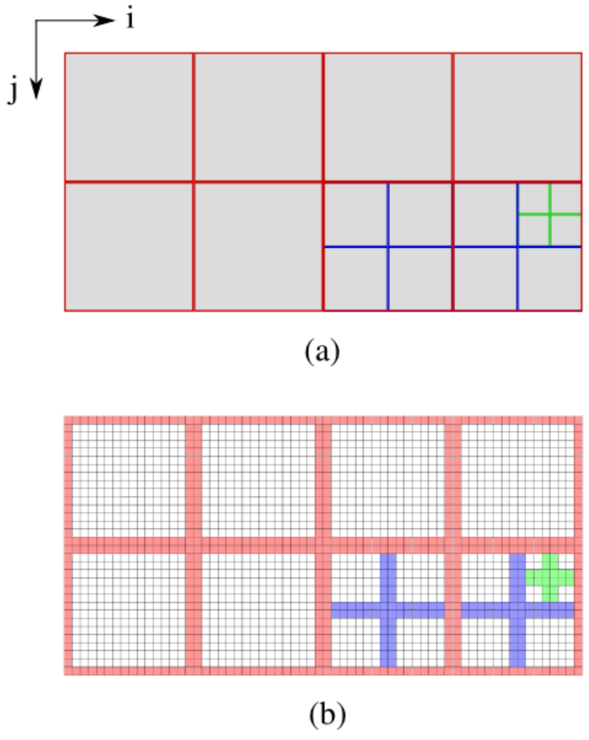


Figure 38 Quadtree grid of a rectangular design domain a) Cell subdivision b) FEM mesh of the domain [68]

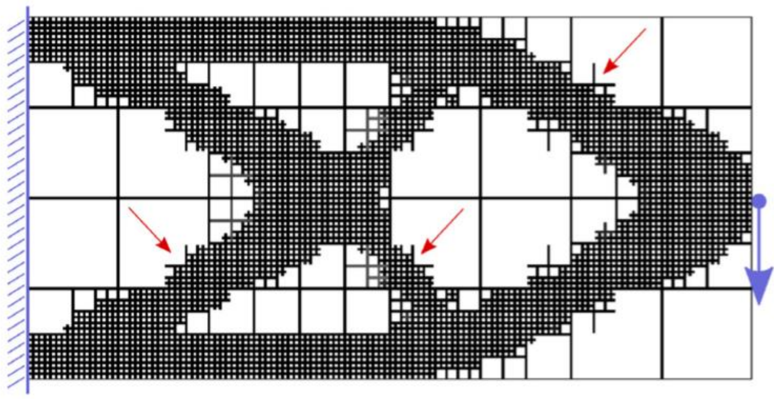


Figure 39 Generated cantilever beam NFGL structure without the refinement of $\chi_{i,j}^{k}$ values [68]

By applying the refinement filter on the same cantilever beam in Figure 39, the suspended struts no longer exist as shown in Figure 40. However, the generated structure doesn’t show a gradual change in the subdivisions of the cells. This is because the cell is only affected by its parent cell in the filtering process. By including the influence of cells that are neighboring the parent cells, the change in subdivisions can become more gradual. This is done by introducing different filtering on the values of $\chi_{i,j}^{k}$ that can produce balanced Quadtree subdivisions.

$$\bar{\chi}_{i,j}^{k} = \frac{\tanh\left(\beta\frac{1}{2}\right) + \tanh\left(\beta\left(\chi_{i,j}^{k} - \frac{1}{2}\right)\right)}{\tanh\left(\beta\frac{1}{2}\right) + \tanh\left(\beta\left(1 - \frac{1}{2}\right)\right)} \tag{23}$$

which is similar to Eq. 14 with β controlling how sharp $\bar{\chi}_{i,j}^{k}$ goes from 0 to 1 in a continuous manner. The relative density is then updated to

$$\boldsymbol{\rho} = \sum_{k=0}^{\bar{k}} \boldsymbol{T}^k \, \bar{\boldsymbol{\chi}}^k \qquad (24)$$

where $\bar{\boldsymbol{\chi}}^k$ is the vector of all $\bar{\chi}^k$ values.

The generated balanced structure using the filter in Eq. 23 is shown in Figure 41. Allowing the cells to gradually subdivide increases the robustness of the structure under uncertain load, but with a sacrifice in the compliance of the structure compared to the unfiltered structure.

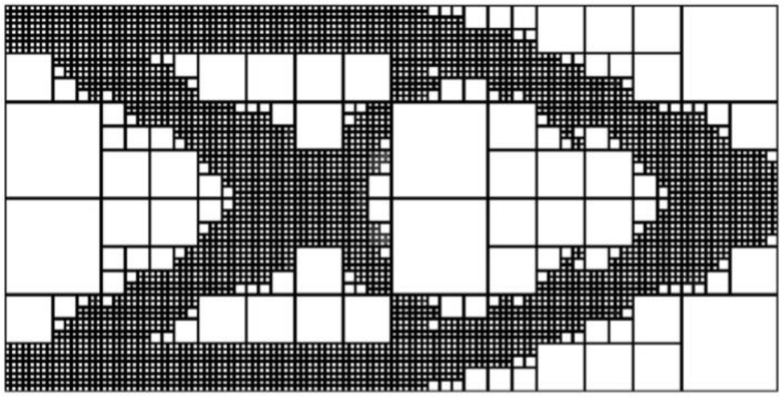


Figure 40 Generated cantilever NFGL beam after the refinement filter [68]

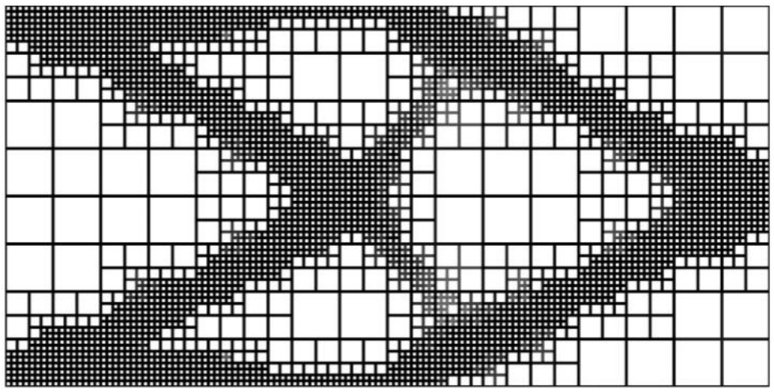


Figure 41 Generated cantilever NFGL beam after the balanced Quadtree refinement filter is applied [68]

A couple of drawbacks arise when using this method to create NFGL structures. The first is that the generated NFGL structures are not conformal as can be seen in the previous figures and Figure 42. Another issue is the increase in the computational cost associated with performing the topology optimization process in each iteration. Also, the method requires that the number of elements has to be a multiple of 2 to the power of 3 or more and that the struts be of a fixed cross-section of two elements.

The advantage of the method is its ability to generate NFGL structures with different cell size ratios by adjusting the number of levels that it can produce. However, this comes at the cost of increasing the computational time. It should be noted that the size ratio is an integer since it can only be generated by the subdivision of the parent cells.

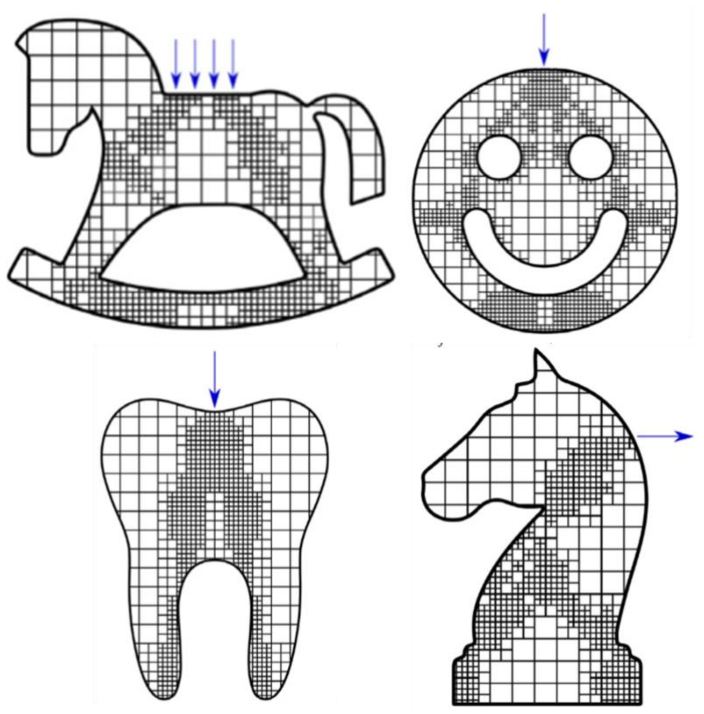


Figure 42 Different structures generated using Adaptive Quadtree optimization method [68]

2.5 Drawbacks of Existing Methods

This section will summarizer the drawbacks and advantages of all the existing methods discussed in this chapter highlight the expected advantages of the proposed method in this research. The proposed method will be named the Naturally Functionally Graded Lattice (NFGL) Framework, which will be the name that will be used to refer to the method throughout this research. Table 2 shows the advantages and drawbacks of the existing methods discussed earlier in this chapter.

From the table, it is apparent that the Error Diffusion and Stocastich Nodal Generation methods share the same advantage of being low in computational cost. The Error Diffusion method has the advantage of being able to handle any type of input density field (assuming it's rectangular) compared to the Stochastic Nodal Generation method. But it requires an input grid of nodes to be generated for it to be used. Also, the filter limits the possible unit-cell size ratios that the method can attain. Unlike the Stochastic Nodal Generation method, where it generates its own nodes without requiring an input grid of nodes. However, the generated nodes are random and introduce uncertainty in the generated structure. Furthermore, its dependence on the actual scale of the design domain to generate the nodes can cause issues with its ability to handle different input density fields. Both methods also share similar drawbacks, such as the need to alter the input density field if a different size for the unit-cells is required and their inability to generate conformal NFGL structures. Moreover, if a unit-cell size ratio is desired, it cannot be achieved intuitively. Different iterations have to be conducted from both methods until the desired ratio is achieved. But compared to the Adaptive Quadtree and Local Volume Constraint methods, the Error Diffusion and Stochastic Nodal Generation have the advantage of not requiring the topology optimization process (if used) to be rerun to generate different NFGL structures.

As for the Local Volume Constraint method, it suffers from having a significantly high computational time compared to the other existing methods due to the added constraints and high nonlinearity of the objective function being optimized. It also constrains the size ratio of the cells in the generated structure, so they cannot be controlled freely and limits the accumulation of materials that can form thicker sturts where needed. But it has the advantage of being able to generate conformal NFGL structures compared to the other methods. Furthermore, it doesn’t require a unit-cell type to be assigned to it, since it generates the cells based on the optimization process. On the other hand, the Adaptive Quadtree method is able to generate NFGL structures of different unit-cell size ratios by just increasing the number of subdivision levels. However, this adds up the computational cost quickly, since it requires adding more elements for the optimization problem. And the size of the FEM mesh has to be constrained to a power of 2 being 3 or higher. Moreover, the generated NFGL structures are not conformal to the design domain. Both the Local Volume Constraint and Adaptive Quad methods share the same drawback of requiring the topology optimization process to be rerun whenever the design parameters are changed, further incurring more computational cost.

Table 2 Advantages and drawbacks of the existing methods

Adaptive Quadtree	**Local Volume Constraint**	**Error Diffusion**	**Stochastic Nodal Generation**
Advantages: • Can generate NFGL structures with varying unit-cell size ratios • Deterministic **Drawbacks:** • High computational cost • Structures are not conformal to design domain • Constrains used FEM mesh size • Requires rerunning the optimization process when parameters are changed • Fixed strut cross-section	**Advantages:** • Can generate conformal NFGL structures • Controlled strut orientation • Does not require a unit-cell type to be assigned • Deterministic **Drawbacks:** • Significantly high computational cost. • The cell size ratio is constrained • Requires rerunning the optimization process when parameters are changed	**Advantages:** • Low computational cost • Can handle any type of rectangular density fields • Deterministic **Drawbacks:** • Structures are not conformal to the design domain • Requires a rectangular density field • The unit-cell size ratio is restricted • Requires altering the input density field to adjust the size of unit-cells	**Advantages:** • Low computational cost • Generates NFGL nodes without an initial grid of nodes **Drawbacks:** • Generated nodes and structure are random • Structures are not conformal to the design domain • Dependence on the scale of the design domain • Requires altering the input density field to adjust the size of unit-cells

The proposed NFGL Framework in this research aims to overcome most of the drawbacks in generating NFGL structures that exist in the methods discussed in this chapter. From the research questions and hypotheses in section 1.5, the following advantages are expected from the NFGL framework:

- Low computational cost
- Can handle any type of density fields regardless of shape
- Can generate Conformal NFGL structures
- Can generate NFGL structures with varying unit-cell size ratios without restrictions
- The generated unit-cell size ratio is intuitive and adjustable
- Does not require altering the input density field or rerunning a topology optimization process
- Deterministic

To assess the ability of the NFGL Framework to provide these advantages, the NFGL Framework will be applied to different application examples in CHAPTER 4. The next chapter will discuss the NFGL Framework in detail and the underlying algorithms that will be used to generate NFGL structures.

CHAPTER 3. NATURALLY FUNCTIONALLY GRADED LATTICE (NFGL) FRAMEWORK

This chapter will outline the NFGL Framework and discuss the algorithms that are involved in developing the framework that will be used to generate NFGL structures. Figure 3 shows a process flowchart of the NFGL Framework and highlights where each algorithm will be involved. The process starts by the user providing four inputs: the density field, the base mesh that will be used, the required unit-cell size ratio S_R and the lattice diameter type d_T (further details will be provided in section 3.2). The density field can be obtained from processes such as topology optimization, FEA, or any user-defined density field input. Once the density field is obtained, a density field function, ρ_f, that calculates the density value based on the coordinates given is created (the function could also be provided directly to the framework). If the density field is based on scattered points in the design domain, as in the case from FEA or topology optimization, then ρ_f would perform a linear interpolation between the scattered points to determine the density value. As for the base mesh, it can either be generated by the user using common FE meshing algorithms or by utilizing the mesh that was used in topology optimization or FEA to provide the base mesh node coordinates $\mathbf{N}_c^*$. The three inputs are then used to generate the NFGL nodes $\mathbf{N}_c$, and the algorithm to do so will be outlined in detail in section3.1. Once the nodes are obtained, the corresponding NFGL structure struts, $\mathbf{L}_c$, and diameters, $\mathbf{d}$, are generated using the algorithm outlined in section 3.2. The user can then choose whether to conduct a similarity analysis (section 3.3) or accept the generated NFGL structure data. If the user decided to conduct the similarity analysis, the MSSIM index is updated (section 3.4) and if the value is acceptable, the process ends by providing the three NFGL structure output data $\mathbf{N}_c, \mathbf{L}_c$ and $\mathbf{d}$. If not, then the value of S_R is updated using the algorithm in section until an acceptable MSSIM value is reached to produce the outputs.

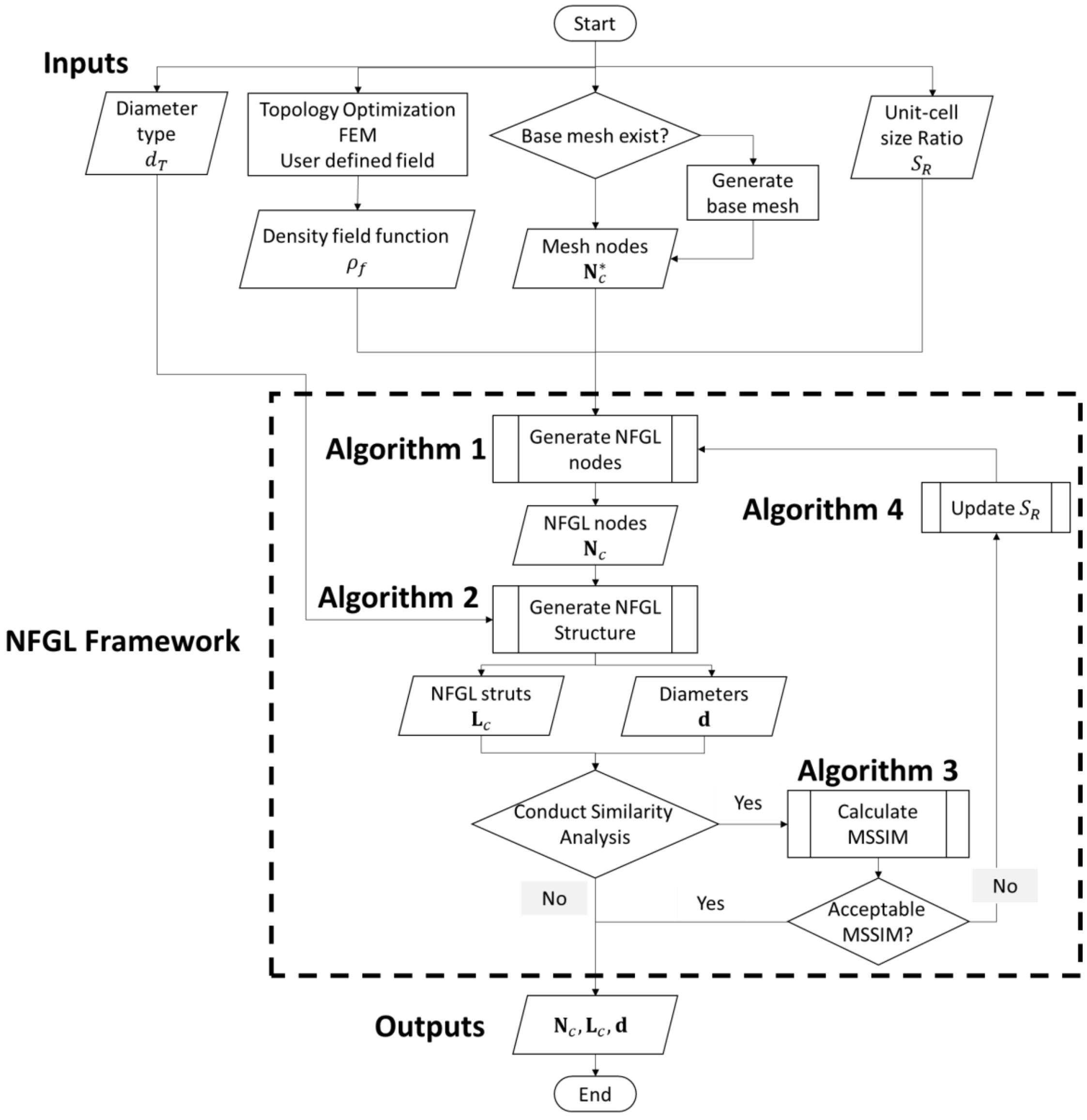


Figure 43 Framework to generate NFGL structures

3.1 NFGL Nodes Generation Algorithm (Simplified Sphere Packing)

This Algorithm is the first component of the NFGL Framework. The focus of this algorithm is to create the nodes that will be used to generate the NFGL structure by using the inputs $\mathbf{N}_c^*$, ρ_f and S_R that were provided by the user. The key idea of this algorithm is to treat the nodes in $\mathbf{N}_c^*$ as spheres of varying radii values that can affect the presence of other nodes around it through a very simplified sphere packing process. Algorithm 1 shows the steps that will generate the NFGL nodes. The algorithm starts by first calculating the relative density of all the nodes in $\rho_f(\mathbf{N}_c^*)$ and storing them in $\boldsymbol{\rho}_n$. Then the values in $\boldsymbol{\rho}_n$ are arranged based on their relative density values in descending order. This arrangement helps in reducing the computational cost by preventing any calculations to be done on nodes that will be potentially removed. Once the nodes are organized, the exterior nodes are assigned to vector **m** of size $m_e \times 1$, where m_e is the number of exterior nodes in the $\mathbf{N}_c^*$, while preserving the ordering of the nodes based on their relative density values. By starting with the exterior nodes first, the generated NFGL nodes will have the geometrical boundary of the design domain preserved from being tampered with by an interior node. The next step starts from the first node in **m,** where an influence sphere radius is calculated based on the relative density value of the node and the unit-cell size ratio that the user inputs. Figure 44 shows an illustration of the influence sphere of a node in 2D. The influence sphere radius is calculated as

$$R(i) = R_L(1 - \boldsymbol{\rho}_n(i)\,) + R_S\boldsymbol{\rho}_n(i) \tag{25}$$

where $R(i)$ is the radius of the influence sphere for node i, R_L is the largest permissible influence sphere radius and R_s is the smallest influence sphere radius. The value of R_s is chosen to be the same as the size of the smallest element in the base mesh, unless the user wants to change it. The value of R_L can be calculated from the unit-cell size ratio as

$$S_R = \frac{R_L}{R_S} \tag{26}$$

where S_R is the unit-cell size ratio requested by the user.

Algorithm 1. NFGL nodes generation

Procedure: NFGL_Nodes$\left(\mathbf{N}_c^*, \rho_f, S_R\right)$
Input: Base mesh node coordinates $\mathbf{N}_c^*$, density field function ρ_f, Unit-cell size ratio S_R

Output: NFGL node Coordinates $\mathbf{N}_c$

```
1:  $\boldsymbol{\rho}_n$ ← Determine node relative density from $\rho_f(\mathbf{N}_c^*)$
2:  Arrange $\boldsymbol{\rho}_n$ in descending order
3:  m ← Extract exterior nodes from $\mathbf{N}_c^*$
4:  For $i$ ← m do
5:      j ← Find all nodes inside the influence radius of $\mathbf{N}_c^*(i)$
6:      j ← Find all nodes $\boldsymbol{\rho}_n(\mathbf{j}) < \boldsymbol{\rho}_n(i)$
7:      Remove $\mathbf{N}_c^*(\mathbf{j})$
8:  End For
9:  m ← Extract interior nodes from $\mathbf{N}_c^*$
10: Repeat Steps 4-8
11: $\mathbf{N}_c$ ← $\mathbf{N}_c^*$
12: Return $\mathbf{N}_c$
```

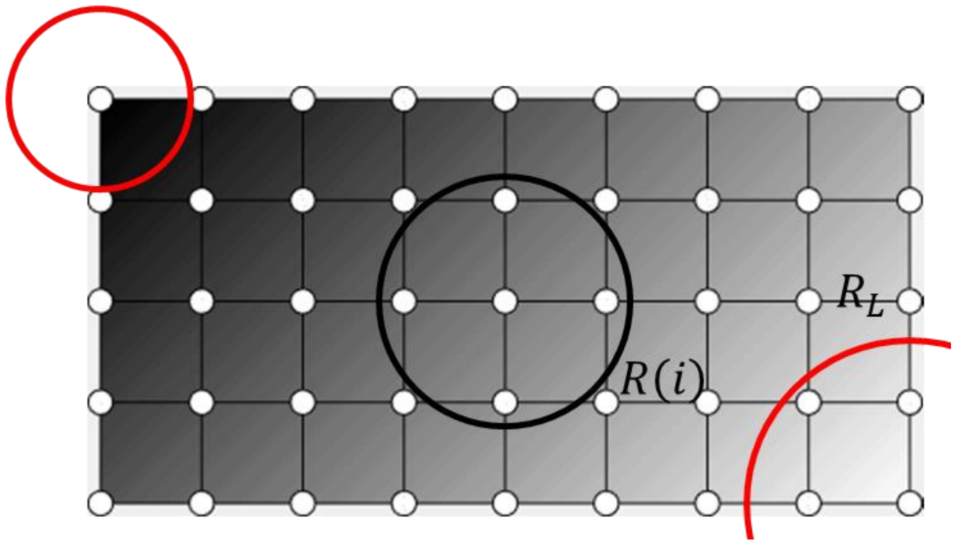


Figure 44 Influence sphere radius of three nodes, colored in black, in a density field

The algorithm then loops across all nodes in $\mathbf{m}$ finds all the nodes that are inside the influence sphere of an exterior node. If a node j is inside the influence sphere of node i with $\boldsymbol{\rho}_n(j) < \boldsymbol{\rho}_n(i)$ then that node is flagged for removal. This flagging eliminates the need to calculate any influence of a node since it shouldn't exist due to it being flagged for removal by another node. Hence why the algorithm starts from the nodes of high ρ values. Once all the nodes in $\mathbf{m}$ have been cycled through, the algorithm assigns the interior nodes to $\mathbf{m}$ of size $m_i \times 1$, where m_i is the number of interior nodes. The process is then repeated for the interior nodes. Once both exterior and interior nodes are processed, the algorithm finishes and produces the NFGL nodes, $\mathbf{N}_c$, that will be utilized in the next section. The size of $\mathbf{N}_c$ is $N_n \times N_D$ where N_n is the number of NFGL nodes and N_D is the dimension (2 or 3). Figure 45 shows an illustration of this process. The two marked nodes are considered for removal since their relative density values are less than that of the colored node. Figure 46 shows the generated NFGL nodes after the algorithm finishes. This process can be described as a simplified sphere packing of spheres/circles with a radius that is half of the influence sphere radius for each node. Figure 47 shows the circles generated using half the radius of the influence sphere and how it looks similar to sphere packing but without the iterative process of moving the nodes, hence why it can be called a simplified sphere packing process.

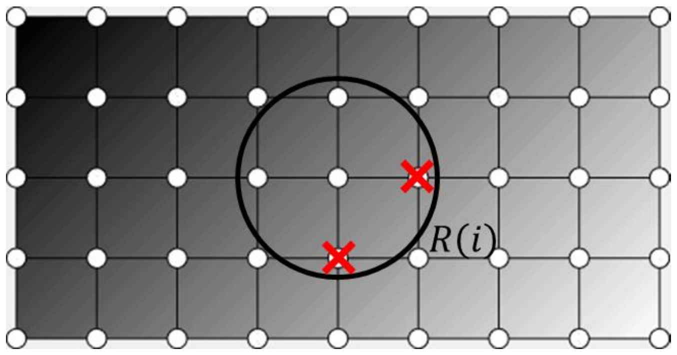


Figure 45 Removal of nodes inside the influence sphere of a colored node if their relative density is less than the colored node relative density

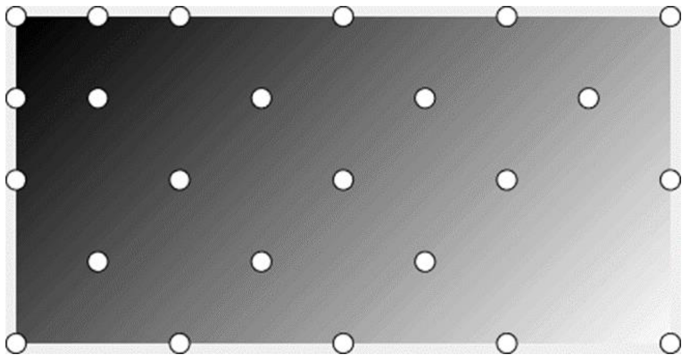


Figure 46 NFGL nodes generated by the algorithm

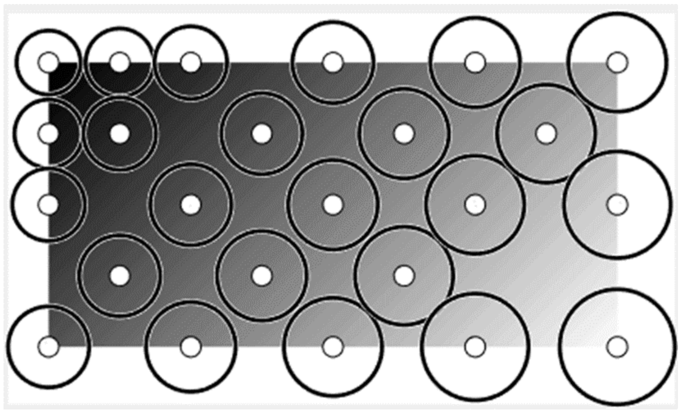
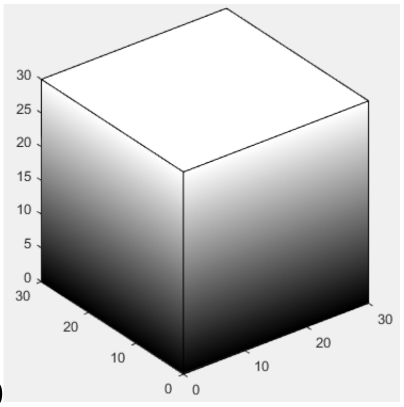


Figure 47 Illustration of the simplified sphere packing

An initial test of the algorithm was performed by supplying a test density input field ρ_f and mesh nodes $\mathbf{N}_c^*$ of a cube as shown in Figure 48. The density field ramps from high density to low density linearly from bottom to top. Four NFGL nodes were generated with different values of $S_R = 1, 1.5, 2$ and 3 as shown in Figure 49. The generated results show consistency to the input density field where the bottom nodes remain unaffected while the spacing for the top nodes changes as S_R value changes.

a) 

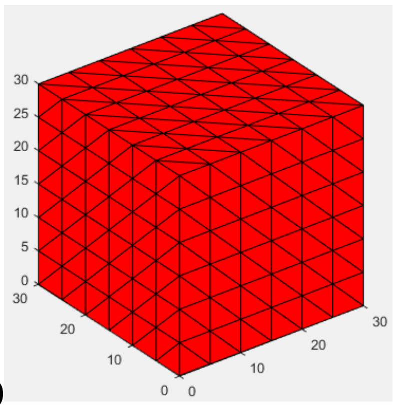
b) 

Figure 48 Algorithm 1 inputs a) Density field ρ_f b) input mesh nodes $\mathbf{N}_c^*$

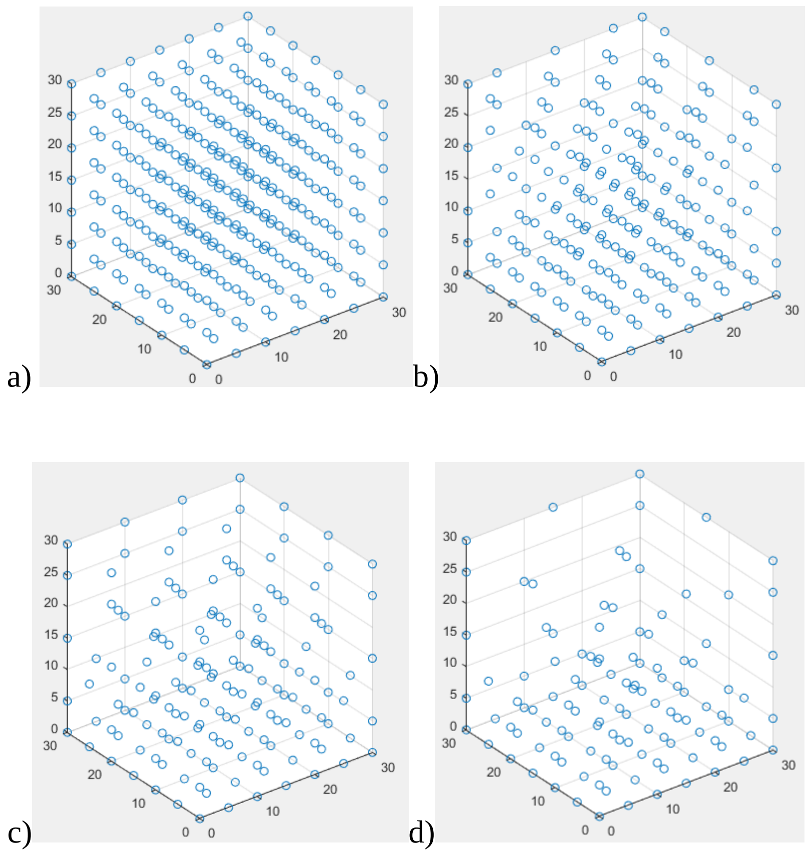


Figure 49 Generated NFGL nodes $\mathbf{N}_c$ with a) $S_R = 1$ b) $S_R = 1.5$ c) $S_R = 2$ d) $S_R = 3$

To further test Algorithm 1, a three-dimensional section of a rectangular tapered beam with a circular through-hole as shown in Figure 50 will be tested. The density field of the tapered beam changes as

$$\rho_f(x, y) = 2.8539\left(\sqrt{x^2 + y^2}\right)^{-0.7565} \tag{27}$$

The resulting NFGL nodes for the beam for different S_R values are shown in Figure 51. The generated nodes show good agreement with the density field as expected.

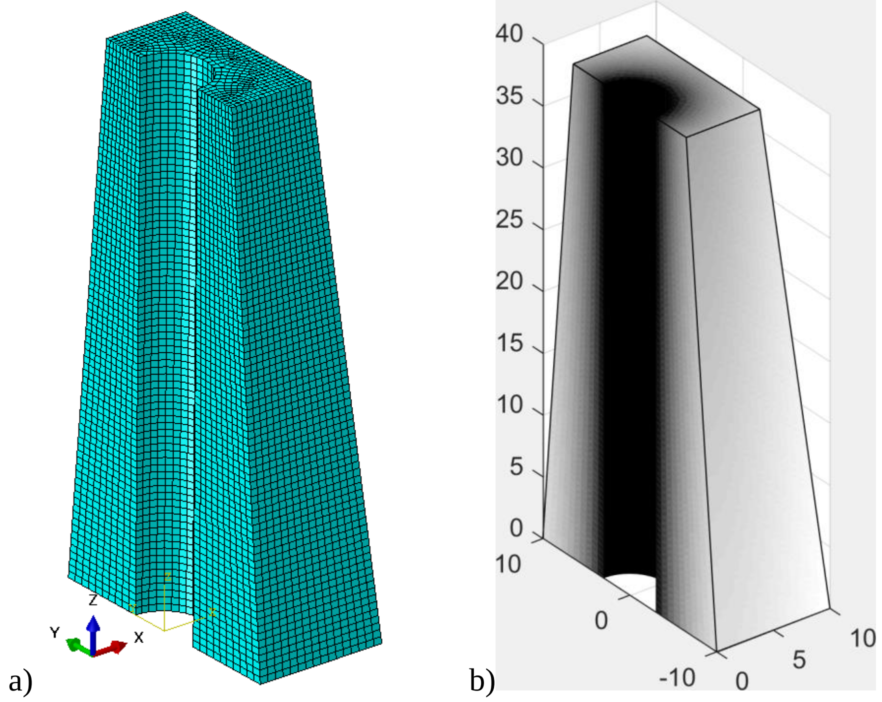


Figure 50 Section of a tapered beam with a circular hole a) Beam initial mesh b) Density field

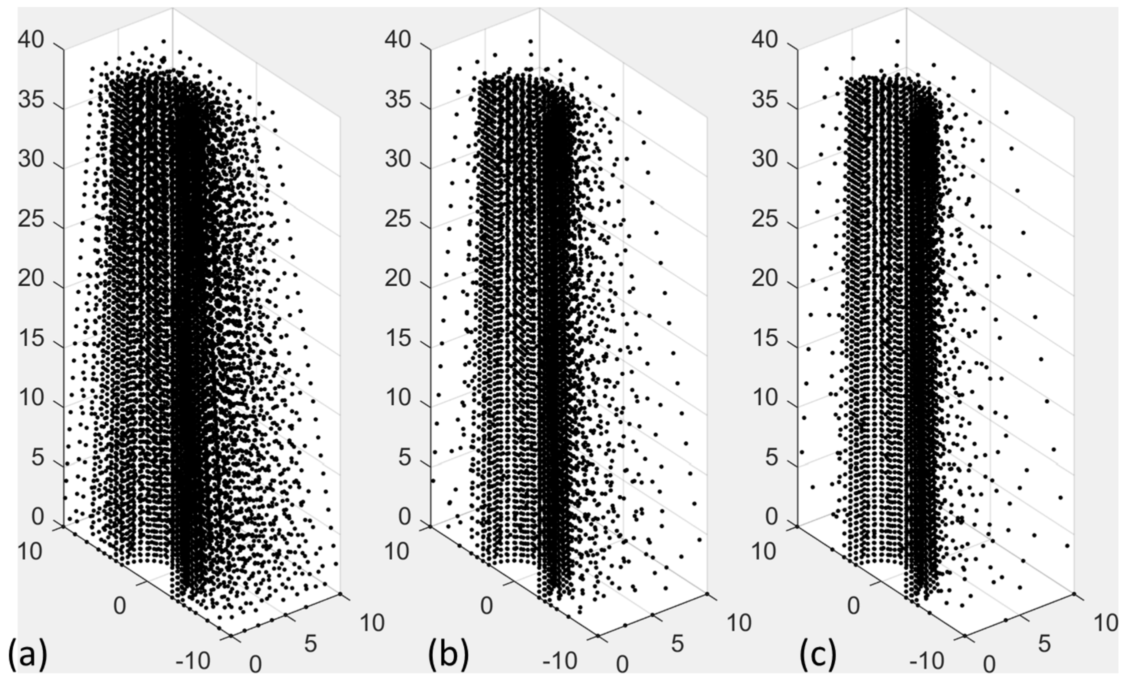


Figure 51 Generated NFGL nodes at different S_R values a) $S_R = 2$ b) $S_R = 3$ c) $S_R = 5$

3.2 NFGL Structure Generation Algorithm

This algorithm, shown in Algorithm 2, is the second component of the NFGL Framework and is used to generate the NFGL structure from the nodes generated from the algorithm 1. The NFGL structure will be generated from tetrahedral/triangular unit-cells, since they provide stretch dominated structures and can be generated from any NFGL nodes. Algorithm 2 will use $\mathbf{N}_c$, d_T, and ρ_f as inputs and provide the lattice connectivity $\mathbf{L}_c$ and diameter values $\mathbf{d}$ based on d_T. The algorithm first starts by creating Delaunay Triangulations from the NFGL nodes $\mathbf{N}_c$. The reasons for choosing Delaunay Triangulations is because it is available in almost all commercial software, the available implementation in Matlab that is based on the Computational Geometry Algorithms Library (CGAL) implementation can create unique triangulations even with degenerate cases [85], and most importantly it creates equiangular triangulations that can prevent the formation of long and thin struts from badly shaped elements. Once the triangulation process is done, the connectivity list for the triangulations is stored in $\mathbf{U}_c$ which will contain N_U elements that correspond to the number of unit-cells that will be created using these elements. Once the unit-cells are created, the connectivity list is used to create lattice struts between each pair of nodes and is stored in $\mathbf{L}_c$ that will contain N_L struts. Each strut in $\mathbf{L}_C$ will be treated as a circular rod in this research.

Algorithm 2. NFGL struts generation

```
Procedure: NFGL_Struts($\mathbf{N}_c$, $d_T$,$\rho_f$)
Input: NFGL node Coordinates $\mathbf{N}_c$ with $N_n$ nodes, diameter type $d_T$, density field
function $\rho_f$
Output: Lattice struts connectivity $\mathbf{L}_c$, struts/nodal diameter $\mathbf{d}$
1:  $\mathbf{U}_c$ with $N_U$ unit-cells ← Unit-cell connectivity list from Delaunay
    Triangulation of $\mathbf{N}_c$
2:  $\mathbf{L}_c$ with $N_L$ struts ← Create strut connectivity list from $\mathbf{U}_c$
3:  For $i \leftarrow 1$ $to$ $N_U$ do
4:      If three-dimensional design
5:          $\mathbf{d}_{lattice}(i)$ via Eq. (35)
6:      Else
7:          $\mathbf{d}_{lattice}(i)$ via Eq. (36)
8:      End if
9:  End For
10: If $d_T = 1$
11:     For $i \leftarrow 1$ $to$ $N_n$ do
12:         $\mathbf{j}$ ← Find struts sharing node $\mathbf{N}_c(i)$
13:         $\mathbf{d}(i) = \min(\mathbf{d}_{lattice}(\mathbf{j}))$
14:     End For
15: Else
16:     For $i \leftarrow 1$ $to$ $N_L$ do
17:         $\mathbf{j}$ ←Find duplicates of $\mathbf{L}_c(i)$
18:         $\mathbf{d}(i) = \max(\mathbf{d}_{lattice}(\mathbf{j}))$
19:     End For
20: End
21: Remove duplicate struts in $\mathbf{L}_c$
22: Return $\mathbf{L}_c$ and $\mathbf{d}$
```

The next step is to determine the strut diameter values in each unit-cell, $\mathbf{d}_{lattice}$. This requires the calculation of the effective relative density, ρ_{eff}, for each lattice unit-cell. The use of Gauss-Legendre quadratures was explored compared to sampling points inside the unit-cell. But the need to map arbitrary elements to evaluate ρ_{eff} increases the computational cost slightly compared to sampling the points inside the unit-cell with a very small increase in the accuracy of ρ_{eff} value. So it was decided to sample the points instead. To do so, interior nodes will be generated in each unit-cell based on the element's dimensionality. In the case of tetrahedral elements, layers to facilitate the generation of these nodes in a uniform manner is created. The number of layers is determined according to the volume of the unit-cell in the following manner

$$T_n = \left\lceil \frac{V_{element}}{R_S^3} \right\rceil \tag{28}$$

Where T_n is the number of interior points, which is also known as the tetrahedral number in 3D or Triangular number in 2D [86], $V_{element}$ is the volume of the solid element counterpart of the unit-cell. The number of interior points in the unit-cell is then determined as

$$T_n = \sum_{i=1}^{N_{layer}} \frac{i(i+1)}{2} \tag{29}$$

where N_{Layer} is the number of layers. Figure 52 shows an example of three unit-cells and the interior nodes generated based on T_n values. In the case of a triangular element, the area of a solid element counterpart is used to determine T_n.

$$T_n = \left\lceil \frac{A_{element}}{R_S^2} \right\rceil \tag{30}$$

$$T_n = \frac{N_{Layer}\left(N_{Layer} + 1\right)}{2} \tag{31}$$

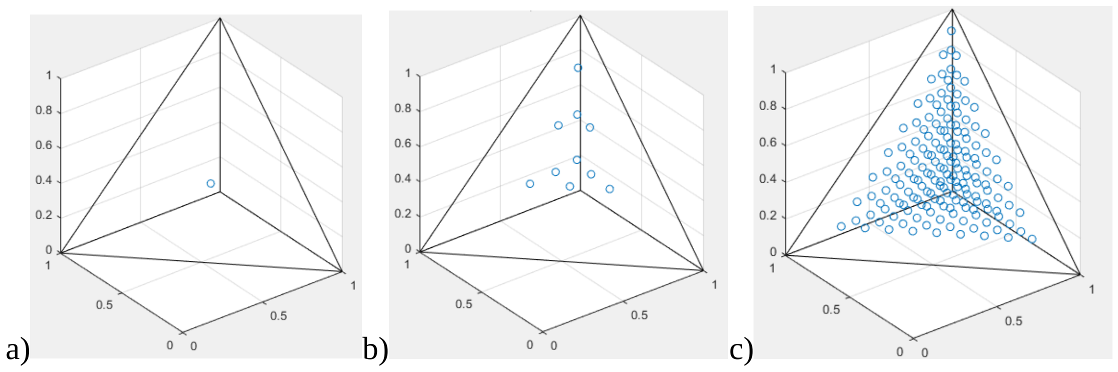


Figure 52 Interior nodes generation in a tetrahedral unit-cell a) $\boldsymbol{N_{Layer}} = \mathbf{1}$ b) $\boldsymbol{N_{Layer}} = \mathbf{3}$ c) $\boldsymbol{N_{Layer}} = \mathbf{9}$

Once the interior points are generated, the value of, ρ_{eff} is calculated as

$$\rho_{eff} = \frac{1}{T_n}\sum_{i=1}^{T_n}\rho_i^* \tag{32}$$

Where ρ_i^* is the relative density value of the interior nodes for the unit-cell. The value of ρ_{eff} is then used to calculate the required lattice volume $V_{lattice}$ or area $A_{lattice}$.

$$V_{lattice} = \rho_{eff} \cdot V_{element} \tag{33}$$

$$A_{lattice} = \rho_{eff} \cdot A_{element} \tag{34}$$

The cross-sectional area of the struts is assumed to be the same in a unit-cell, so the lattice diameter is determined as follows for 3D and 2D respectively

$$\mathbf{d}_{lattice}(i) = 4\sqrt{\frac{V_{lattice}(i)}{\pi\sum_{j=1}^{6}L_j}} \tag{35}$$

$$\mathbf{d}_{lattice}(i) = 2\frac{A_{lattice(i)}}{\sum_{j=1}^{3} L_j} \tag{36}$$

Where $\mathbf{d}_{lattice}(i)$ is the cross-sectional area of the lattice struts in unit-cell i and L_j is the length of the strut j in the unit-cell. These diameter values are not unique since there are still duplicate struts between unit-cells. So, a unique value for the diameter must be determined. The diameter values are either determined as nodal diameters or strut diameters based on d_T value. If a nodal diameter is desired $(d_T = 1)$, the minimum value is chosen from the duplicate diameters of struts **j**. Figure 53-a shows an illustration of the case when the minimum value is used and Figure 53-b shows when the maximum diameter is used. It is clear from the figure, that using the maximum value would produce struts of large diameters in areas of low relative density value. Hence, the minimum value is preferred.

$$\mathbf{d}(i) = \min\left(\mathbf{d}_{lattice}(\mathbf{j})\right) \tag{37}$$

where **d** contains the unique diameter values. Alternatively, if a strut diameter is desired $(d_T \neq 1)$, the maximum value is chosen from the duplicate diameters of struts **j**. Figure 54a shows an illustration of the case when the maximum value is used and Figure 54b shows when the minimum diameter is used. In this case, using the minimum value would produce struts of small diameters in areas of high relative density, which is why the maximum value is preferred.

$$\mathbf{d}(i) = \max\left(\mathbf{d}_{lattice}(\mathbf{j})\right) \tag{38}$$

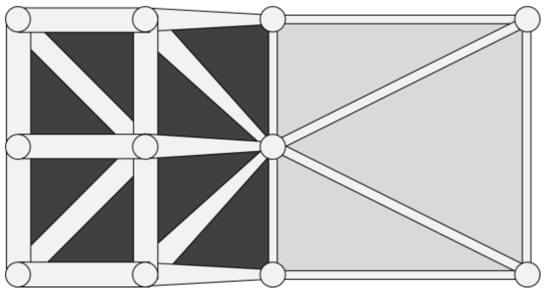
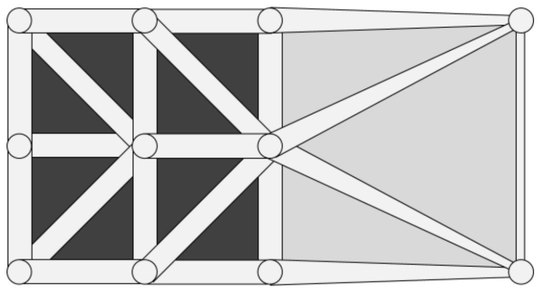
a)  b) 

Figure 53 NFGL nodal diameters based on $d_{lattice}$ values a) using the minimum value b) using the maximum value

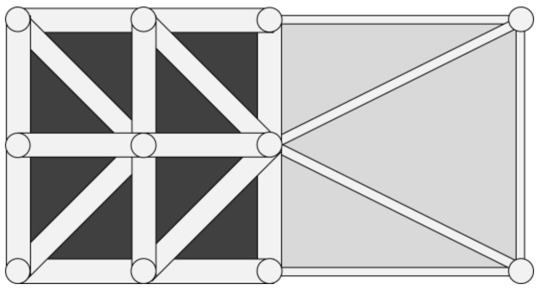
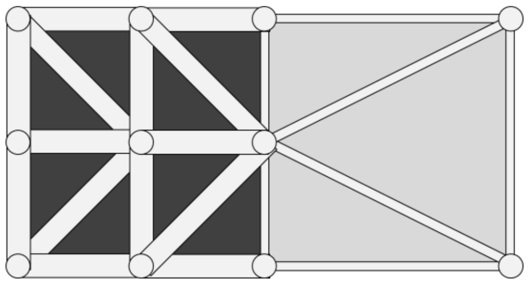
a)  b) 

Figure 54 NFGL strut diameters based on $d_{lattice}$ values a) using the maximum value b) using the minimum value

To test Algorithm 2, the generated NFGL nodes for the tapered beam in Figure 51 were used to generate the NFGL structure. Figure 55 shows the generated structure using nodal diameter values $(d_T = 1)$ at different S_R values and Figure 56 shows the generated NFGL structures using strut diameter $(d_T \neq 1)$ at different S_R values. Figure 57 shows a close up of two structures from both figures at $S_R = 5$ to show how the diameters vary in the strucures.

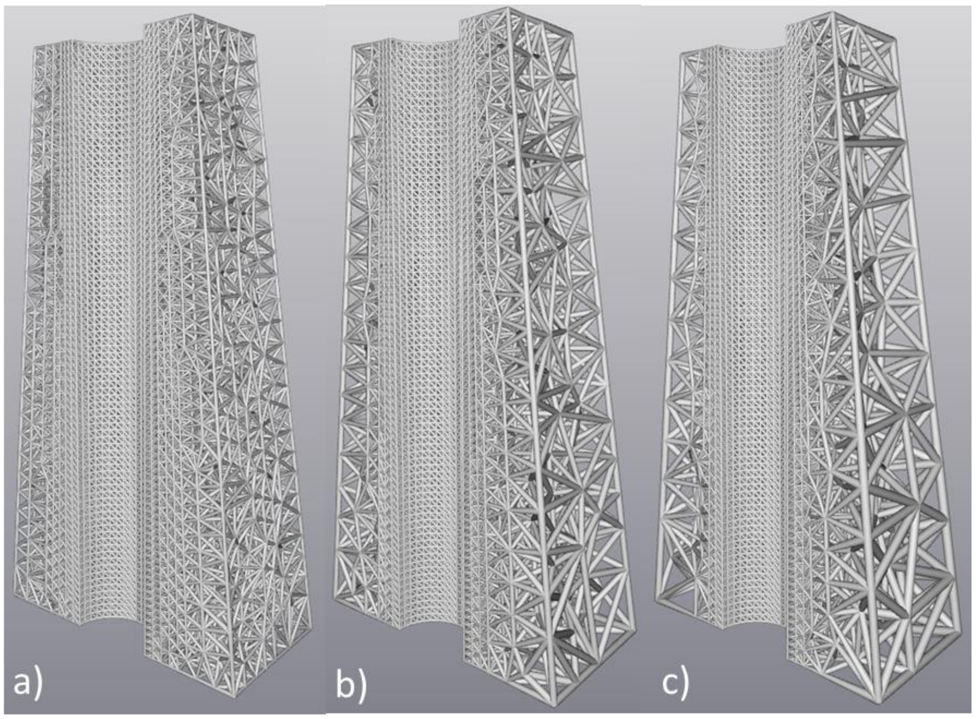


Figure 55 Generated NFGL structures using nodal diameter a) $S_R = 2$ b) $S_R = 3$ c) $S_R = 5$

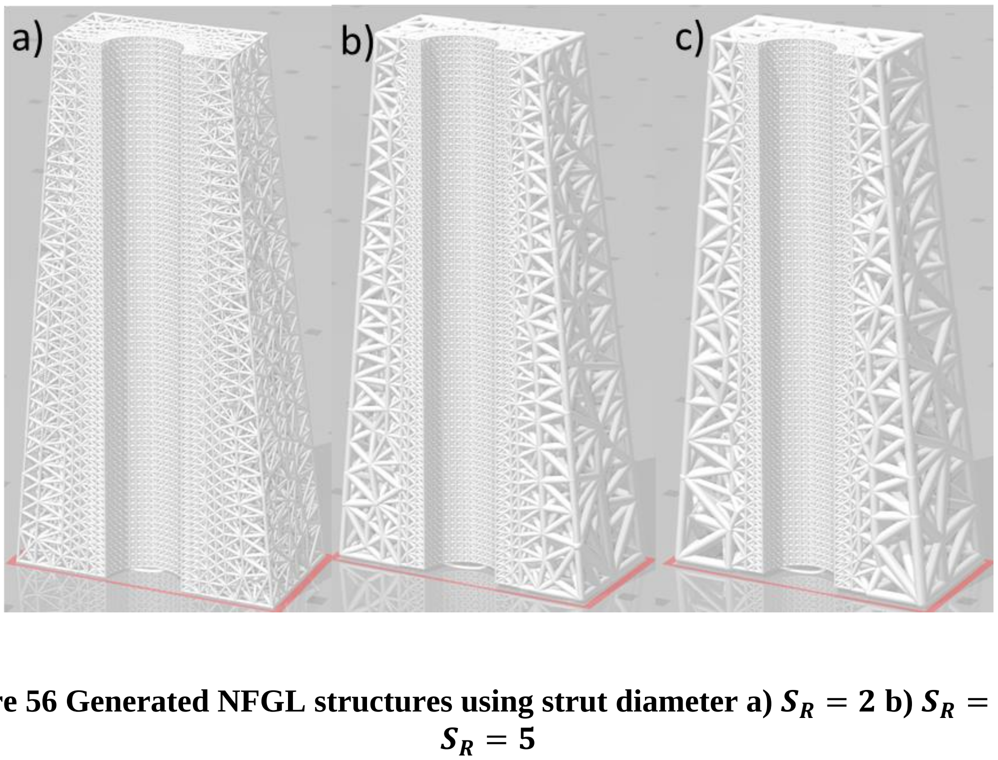
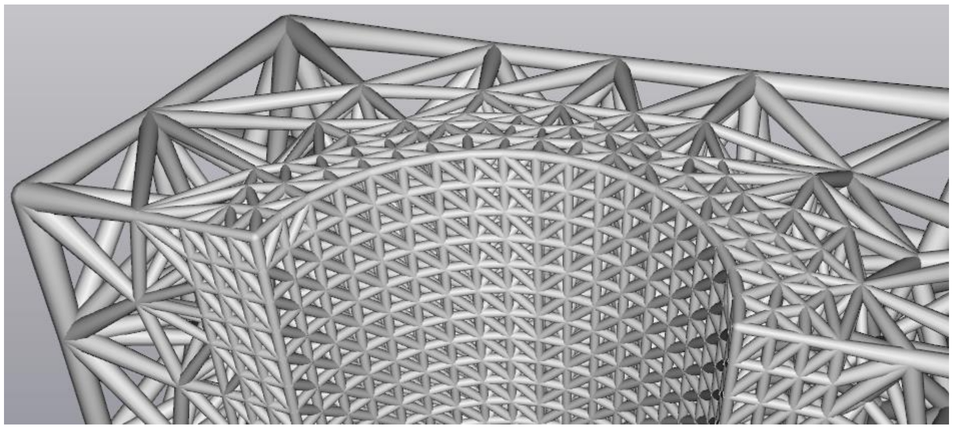


Figure 56 Generated NFGL structures using strut diameter a) $S_R = 2$ b) $S_R = 3$ c) $S_R = 5$

a) 

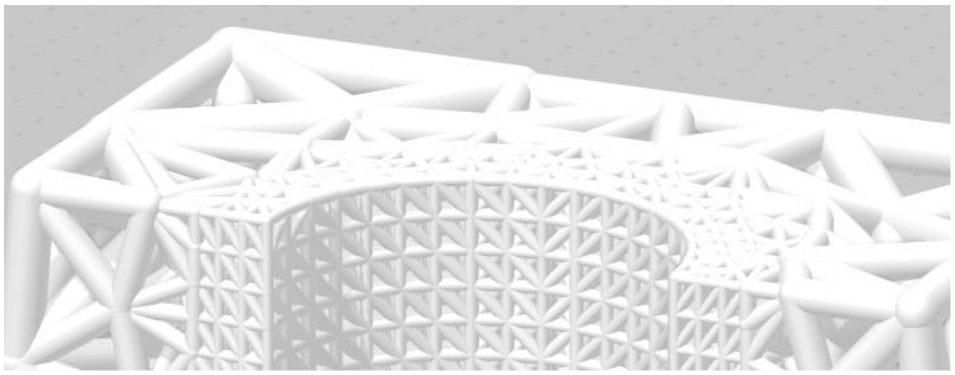
b) 

Figure 57 A close-up view of the generated NFGL struts at $S_R = 5$ for a) Nodal diameter b) Strut diameter

3.3 Similarity Analysis using MSSIM Index Algorithm

This algorithm is optional in the NFGL Framework if the user requires to improve the performance of the generated NFGL. If the user only requires a certain unit-cell size ratio, then they can proceed to generate tne NFGL structure from the outputs of algorithm 1 and 2 without utilizing this algorithm or algorithm 4.

As explained in research question 3, this algorithm will utilize the MSSIM index to determine the similarity between the generated NFGL structure and density field input. The algorithm to calculate the MSSIM index will be extended to include three-dimensional voxels in this research. In the beginning, two tensors **X** and **Y** of size $n \times m \times l$ that contains the relative density values for both the density field input and the NFGL structure respectively are created. The values in **X** and **Y** are at uniformly distributed points that cover the design domain. Since the arrays can contain points that are outside of the design domain, as shown in Figure 58, these points will be assigned a value of -1. The figure shows a matrix of a two-dimensional generalized shape, where elements inside the domain are assigned a value, and the elements outside are assigned a value of -1. This will help in reducing the computational cost of having to calculate the MSSIM index for the entire design domain and improves the accuracy of the similarity analysis by only focusing on the points that are in the design domain. Then, **X** and **Y** are provided to Algorithm 3 as inputs. The algorithm starts by choosing the first element in the array and checking if its value is equal to -1. If the value is -1, then the algorithm cycles to the next element in the array. Otherwise, the calculation of a local structural similarity (SSIM) index is carried out.

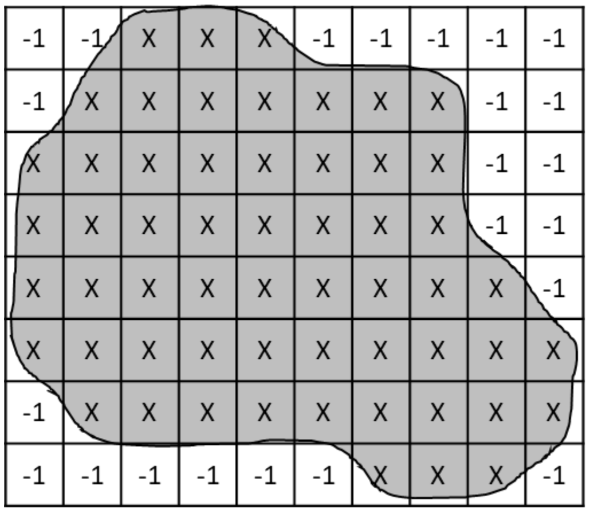


Figure 58 Relative density Array for a generalized shape showing assigned values

Algorithm 3. MSSIM Calculation

```
Procedure: MSSIM(X, Y)
Input: Input relative density tensors X, NFGL relative density tensor Y
Output: Mean Structural Similarity MSSIM
1:  $M = 0$
2:  For $i \leftarrow 1$ to $n$
3:      For $j \leftarrow 1$ to $m$
4:          For $k \leftarrow 1$ to $l$
5:              If $\mathbf{X}(i,j,k) \neq -1$
6:                  $M = M + 1$
7:                  x ← Select $\mathbf{X}(i,j,k)$ and appropriate neighboring points
8:                  y ← Select $\mathbf{Y}(i,j,k)$ and appropriate neighboring points
9:                  Apply gaussian filter to x and y
10:                 SSIM($M$) ← Determine using x and y via Eq. (46)
11:             End If
12:         End For
13:     End For
14: End For
15: MSSIM via Eq. (47)
16: Return MSSIM
```

SSIM index is calculated around a local set of points **x** and **y,** as shown in Figure 59**,** in **X** and **Y** that are weighted to a circular, symmetric Gaussian filter [87]. To extend the calculation of SSIM to voxels, a 3D Gaussian filter will be used. The SSIM is defined as a function of three components: luminance $l(\mathbf{X},\mathbf{Y})$, contrast $c(\mathbf{X},\mathbf{Y})$ and structure $s(\mathbf{X},\mathbf{Y})$. The luminance is defined as

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{39}$$

Where μ_x and μ_y are the mean intensity values, which corresponds to the mean relative density value of the points in **x** and **y** respectively, C_1 is constant for stability when both μ_x and μ_y are zero. The value of μ_x, and similarly μ_y, is determined as follows

$$\mu_x = \sum_{i=1}^{m} \omega_i x_i \tag{40}$$

Where m number of points in **x** and **y** respectively, ω_i is the gaussian weight of point i and x_i is the relative density value for that point in **x**. As for C_1, it is defined as

$$C_1 = (0.001L)^\wedge 2 \tag{41}$$

Where L is the dynamic range of the elements in the arrays, which in the case of relative density is 1. Therefore, the value of C_1 would be 1×10^{-6}.

The contrast is defined as

$$c(\mathbf{x}, \boldsymbol{y}) = \frac{2\sigma_x\sigma_y + C_1}{\sigma_x^2 + \sigma_y^2 + C_1} \tag{42}$$

Where σ_x and σ_y are the standard deviations for the points in **x** and **y** respectively and is defined for σ_x and similarity σ_y as

$$\sigma_x = \sqrt{\sum_{i=1}^{m} \omega_i (x_i - \mu_x)^2} \tag{43}$$

The structure is defined as

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_1}{\sigma_x\sigma_y + C_1} \tag{44}$$

Where σ_{xy} is the covariance and is defined as

$$\sigma_{xy} = \sum_{i=1}^{m} \omega_i (x_i - \mu_x)(y_i - \mu_y) \tag{45}$$

After all three functions are evaluated, the SSIM is calculated by multiplying all three.

$$SSIM = l(\mathbf{x}, \mathbf{y}) \cdot c(\mathbf{x}, \mathbf{y}) \cdot s(\mathbf{x}, \mathbf{y}) \tag{46}$$

If **y** was exactly similar to **x**, all three components would be equal to 1, thus the SSIM index would be 1 as well. Once the values of all local SSIM indices for all the elements in the arrays are calculated, the MSSIM index is then obtained as the mean value for all local SSIM values calculated.

$$MSSIM(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^{M} SSIM(\mathbf{x}_j, \mathbf{y_j}) \tag{47}$$

The value of MSSIM will determine how similar the generated NFGL is to the density field input. With the increase in the unit-cell size ratio, the value of MSSIM will decrease. This will help in providing a proper range to the unit-cell size ratio for the NFGL structure.

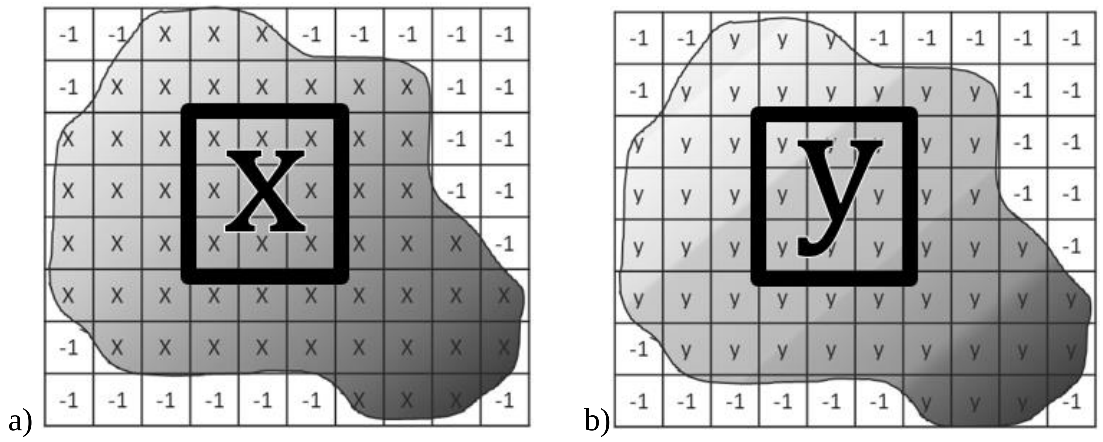


Figure 59 Relative density Array for a generalized shape showing assigned values

a) Input density field matrix b) NFGL structure density matrix

To test Algorithm 3, the tapered beam will be used to determine the change in the MSSIM index values as S_R value changes. Figure 60 shows the change in the MSSIM index as S_R value increases. The figure shows that the MSSIM index value starts to decrease similar to a power function of the form

$$MSSIM = a\, S_R^{-b} \tag{48}$$

where a and b are constants. Furthermore, the figure shows the fitted power curve in Eq. 48 to the data points of the MSSIM index. The fitted curve shows good agreement to the MSSIM index curve. However, the beginning of the curve appears to have a small plateau region. So using a higher value of S_R when fitting the curve would be recommended to avoid increasing the number of iterations due to the beginning of the curve having a very small slope. The figure also shows two fitted curves, one with the inclusion of all data points and one by ignoring the points near the plateau region.

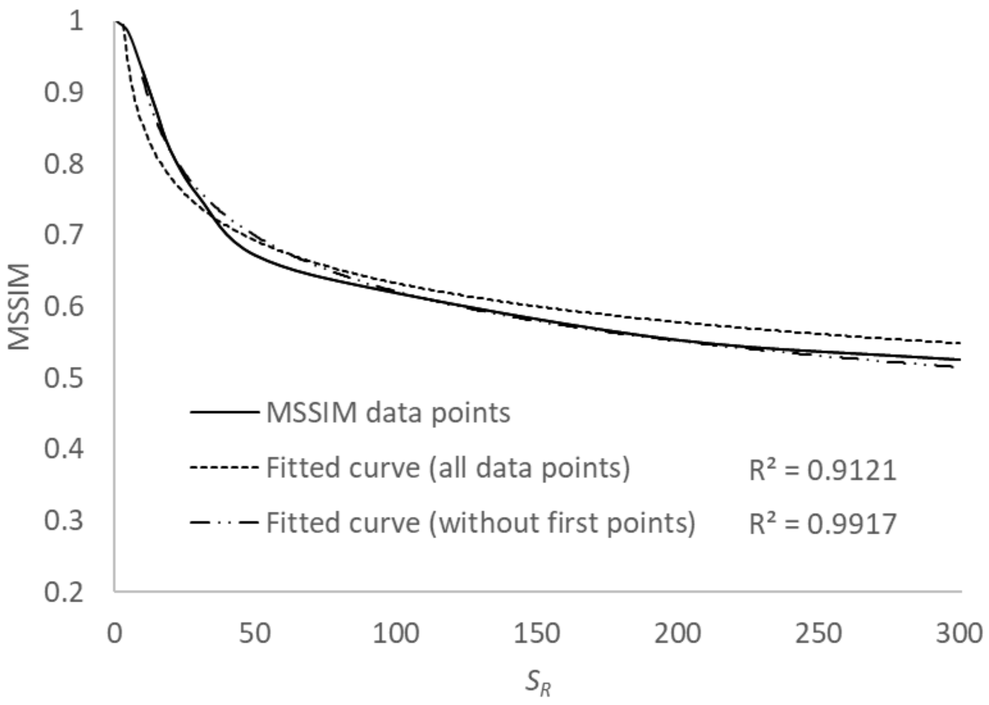


Figure 60 Change in MSSIM index as a function S_R showing the fitted power curve

Further examples that will be provided in CHAPTER 4 will show similar behavior. By utilizing this relation, the value of S_R could be updated with very few iterations to obtain a certain MSSIM index threshold.

3.4 Updating S_R Value Algorithm

After calculating the MSSIM index value, the NFGL Framework would then have to update the unit-cell size ratio accordingly. Therefore, if there's a required similarity threshold that the user desires, the NFGL Framework should be able to achieve it accordingly and with the lest computational cost possible. Based on the observations in section 3.3, the MSSIM index shows a power function relation with the change in S_R value. Algorithm 4 utilizies this relation in order to update S_R value until the MSSIM index falls into the required threshold. The data points obtained from calculating the MSSIM index are stored into two vectors, $\mathbf{S}_R$ and **MSSIM** that holds the values of S_R and the calculated MSSIM indices respectively. The algorithm uses the values in $\mathbf{S}_R$ and **MSSIM** in a linearized form of Eq. 48 in order to perform a linear regression to obtain the values of a and b.

$$\log(MSSIM) = \log(a) - b\log(S_R) \tag{49}$$

$$b = \frac{\sum\left(\log(S_R) - \overline{\log(S_R)}\right)\left(\log(MSSIM) - \overline{\log(MSSIM)}\right)}{\sum\left(\log(S_R) - \overline{\log(S_R)}\right)^2} \tag{50}$$

$$a = \exp\left(\overline{\log(MSSIM)} - b\,\overline{\log(S_R)}\right) \tag{51}$$

where $\overline{\log(S_R)}$ and $\overline{\log(MSSIM)}$ are the mean values of $\log(S_R)$ and $\log(MSSIM)$ respectively. To avoid any singularities, arrays will be padded by 1 at the end. These added values represent the fact that when $S_R = 1$, then the MSSIM index would also be 1. Once the values are obtained from Eqs. 49-51, Eq. 48 is used to determine the new value of S_R that falls inside the required threshold.

$$S_R = \exp\left(\frac{\log(MSSIM) - \log(a)}{-b}\right) \tag{52}$$

The obtained S_R value is then fed back to Algorithm 1 to regenerate the NFGL structure accordingly. The process is repeated until the generated NFGL structure MSSIM index value is within the desired threshold. Once that happens, the NFGL Framework outputs the NFGL structure design parameters and terminates.

Algorithm 4. Updating S_R

```
Procedure: Update_$S_R$($\mathbf{S}_R$, $\mathbf{MSSIM}$)
Input: Vector of previous $S_R$ values $\mathbf{S}_R$, Vector of previous $MSSIM$ values $\mathbf{MSSIM}$
Output: New $S_R$
17: Pad $\mathbf{S}_R$ and $\mathbf{MSSIM}$ with 1 at the end.
18: Calculate $b$ using Eq. 50
19: Calculate $a$ using Eq. 51
20: Calculate $S_R$ using Eq. 52
21: Return $S_R$
```

CHAPTER 4. APPLICATION EXAMPLES

This chapter will demonstrate the application of the NFGL Framework on different examples to compare and demonstrate the ability of the NFGL Framework to design and Generate NFGL structures. The first example will provide a comparison of the NFGL Framework against the methods discussed in CHAPTER 2. The second example will investigate the performance of the NFGL structures generated by the NFGL Framework against FGL structures in the design and optimization of an automotive control arm under multiple loading conditions. The third example will investigate the thermomechanical performance of generated NFGL structures against FGL structures on the design of an injection mold lattice cooling channel.

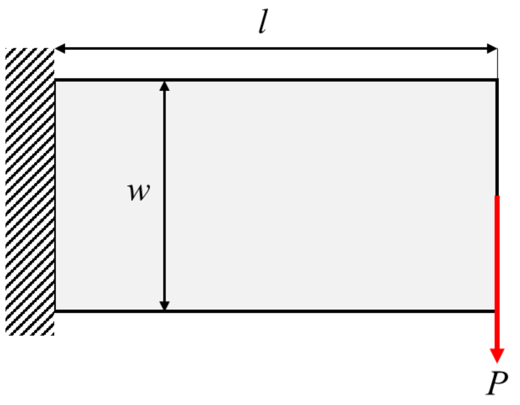
4.1 Comparison of NFGL Framework with other Algorithms in the Literature

In this section, the NFGL Framework will be compared with the Error Diffusion, Adaptive Quadtree, Local Volume Constraint, and the Stochastic Nodal Generation methods in generating NFGL structures using four examples. The first and second examples will evaluate the NFGL Framework's computational cost and structural performance respectively against the aforementioned methods in the literature. The third example will evaluate the robustness of the NFGL Framework against the Stochastic Nodal Generation and Error Diffusion methods by applying them on a sinusoidal input density field. The fourth example will evaluate the NFGL Framework's ability to generate conformal NFGL structures using a circular design domain and a curved path against the same methods in the third example.

These methods were chosen to be compared with the NFGL Framework because they encompass all of the existing methods in the literature that were used to generate NFGL structures. However, the reason why the Adaptive Quadtree and Local Volume Constraint methods were omitted in the third and fourth example, was because they cannot generate NFGL structures from a given density field. They require running a topology optimization of a design domain under loading conditions to generate the lattice structure from the FE mesh.

4.1.1 Computational Cost of Algorithms

To evaluate the computational cost of the methods, a 2D cantilever and a simply supported beam will be used to generate NFGL structures using the NFGL Framework and the other methods. Figure 61 shows the dimensions, the boundary conditions, and the loading conditions that the beams are subjected to for the cantilever beam (Figure 61-a) and the right half of the simply supported beam (Figure 61-b). The cantilever beam is subjected to a unit load on the right side of the beam and is completely fixed on the other end. While half of the simply supported beam is used, due to symmetry, where the symmetry line is prevented from movement in the horizontal direction, the upper left corner is subjected to a point load, and the lower right corner is fixed from movement in the vertical direction as shown in the figure. The cantilever beam geometry will be generated with different FE mesh sizes using four noded 2D quadrilateral elements to determine the computational time needed to generate the lattice structure. The mechanical properties of the beam are shown in Table 3.

a) 

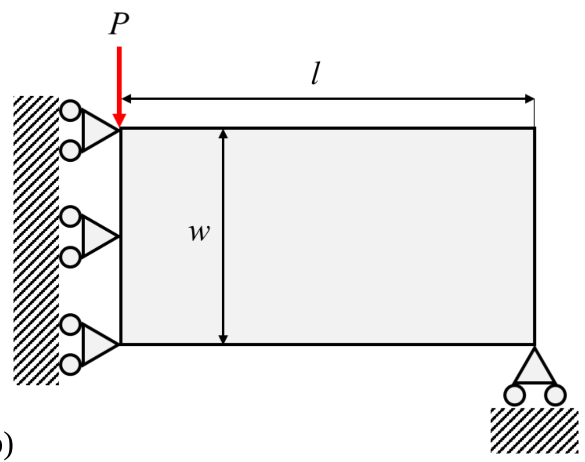
b) 

Figure 61 Loading conditions for the a) cantilever beam b) simply supported beam

Table 3 Cantilever beam mechanical properties

Property	**Value**
l (mm)	200
w (mm)	100
t (mm)	1
E (MPa)	1
P (N)	1

where *l*, *w*, and *t* are the beam's length, width, and thickness respectively; *E* is the elastic modulus of the beam; and *P* is the load on the beam. Furthermore, to generate the lattice structure for the beam, a topology optimization process will be carried out. The Local Volume Constraint and Adaptive Quadtree methods will conduct a modified form of the SIMP optimization, as explained in sections 2.3 and 2.4 respectively, while the other methods will use unpenalized SIMP optimization to generate the input density field. The objective of the topology optimization is to minimize the structure's compliance while constraining the lattice volume to 52% of the solid beam volume. Figure 62 shows the results of the unpenalized SIMP optimization for both beams.

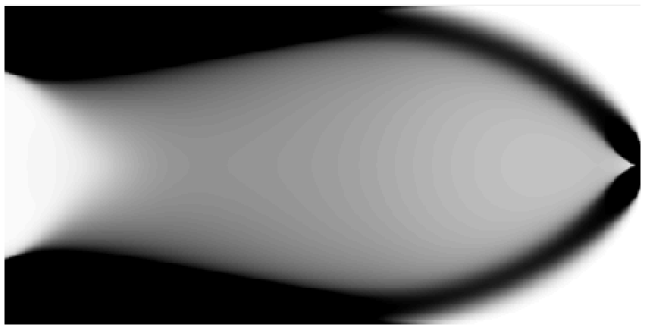
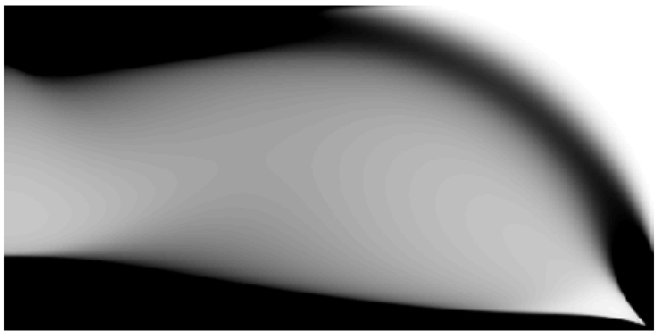
a)

b)


Figure 62 Unpenalized SIMP optimization results a) Cantilever beam b) Simply supported beam

To calculate the computational cost of each method, the time required to run the method and generate the NFGL structure will be recorded. Table 4 shows the tasks that each method will perform in order to generate the NFGL structure. From the table, it can be noticed that all methods will require the generation of the FE mesh of the design domain in order to conduct the topology optimization process. The Adaptive Quadtree and Local Volume Constraint methods rely on their modified topology optimization process to generate the NFGL structures. As for the other methods, they require a density field input to generate the NFGL nodes, so the unpenalized topology optimization results in Figure 62 was used. After the NFGL nodes are generated, the NFGL structure would be generated using Algorithm 2 in section 3.2.

Table 4 Breakdown of the tasks that each method will perform to generate the NFGL structures

Algorithm	**Task**			
Adaptive Quadtree	Generate design domain mesh	Conduct Quadtree Optimization to generate NFGL structure		
Local Volume Constraint	Generate design domain mesh	Conduct Local Volume Constraint Optimization to generate NFGL structure		
Stochastic Nodal Generation	Generate design domain mesh	Generate density field input (unpenalized topology Opt.)	Create NFGL nodes using Stochastic Nodal Generation	Create NFGL structure using Algorithm 2
Error Diffusion	Generate design domain mesh	Generate density field input (unpenalized topology Opt.)	Create NFGL nodes using Error Diffusion	Create NFGL structure using Algorithm 2
NFGL Framework	Generate design domain mesh	Generate density field input (unpenalized topology Opt.)	Create NFGL nodes using Algorithm 1	Create NFGL structure using Algorithm 2

To apply the adaptive quadtree method, the number of elements along the beam's length and width has to be a multiple of 2 to the power of 3 or more to generate the smallest unit-cell size in the structure possible through refining larger unit-cells. Therefore, all the different FE mesh sizes that will be used for the other algorithms will be the same as the ones used in the Adaptive Quadtree mesh. The sizes that will be used in this example are 64×32, 128×64, 256×128, 512×256, 640×320, 768×384, and 1024×512. Furthermore, NFGL structures using the NFGL framework will be generated at different S_R values to evaluate the effect of changes in S_R values on the computational cost of the NFGL Framework. The values of S_R that will be used are 1, 2, 3, and 30. These values were chosen to act as an upper and lower bound for the computational cost of the NFGL framework. The results of the total computational time of each method for the cantilever beam are shown in Table 5 while Table 6 shows a breakdown of the computational time for the NFGL Framework, Stochastic Nodal Generation, and Error Diffusion methods.

From Table 5, it is clear that the Local Volume Constraint method is the most computationally expensive method followed by the Adaptive Quadtree method. The reason for that is because these two methods rely on conducting penalized topology optimization with additional constraints in each iteration during the generation of the NFGL structure. This agrees with hypothesis 1 that in order to reduce the computational cost, the developed algorithm should not rely on topology optimization to determine the location of nodes in the NFGL structure in an iterative manner. Furthermore, since the topology optimization process in both the Adaptive Quadtree and Local Volume Constraint methods is penalized and has additional constraints, the complexity of the optimization process increases. This also increases the number of iterations needed for convergence during optimization as shown in Table 7. The table shows the number of iterations used in the topology optimization process for the unpenalized SIMP method against Adaptive Quadtree and Local Volume Constraint methods.

Looking back to Table 6, it is clear that the increase in S_R values causes a reduction in the computational cost for the NFGL Framework. This reduction in computational cost is due to the NFGL Framework removing more nodes that are unnecessary for the NFGL structure, which allows the NFGL Framework to deal with fewer nodes to generate the NFGL structure. Furthermore, the computational cost of the NFGL Framework is significantly close to the Error Diffusion method when $S_R = 2$. While the computational cost when $S_R = 1$ is the highest, since it doesn't remove any node during the removal process. Furthermore, the generated structure with $S_R = 1$ would just be a regular FGL structure without any changes in unit-cell size. Therefore, utilizing the NFGL Framework for such a case would be pointless. But when using $S_R = 2$, the computational cost almost dropped by half. Therefore, a high value of S_R is a good recommendation when using the NFGL framework to significantly reduce the computational cost.

Table 5 Computational time to generate the NFGL structure of the cantilever beam using NFGL Framework, Stochastic Nodal Generation, Error Diffusion, Local Volume Constraint, and Adaptive Quadtree methods

	Time							
	NFGL Framework				Stochastic Nodal	Error	Local Volume	Adaptive
No. elements	S_R = 1	S_R = 2	S_R = 3	S_R = 30	Generation	Diffusion	Constraint	Quadtree
64×32	1.85	1.78	1.61	1.55	4.74	1.78	41.00	50.60
128×64	7.28	6.69	6.41	7.15	9.09	6.65	123.55	146.84
256×128	32.36	29.20	28.17	27.80	28.89	29.19	959.16	360.11
512×256	130.85	119.15	116.63	113.04	108.87	119.52	12815	957.72
640×320	272.09	252.23	247.62	243.34	234.48	252.34	16125	1280.85
768×384	389.28	356.12	349.61	342.09	326.97	356.60	95607	2826.77
1024×512	695.58	643.36	633.16	619.78	593.53	645.62	N/A	4524.41

Table 6 Breakdown of the computational cost of the cantilever beam for the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods

	Time						
	Unpenalized	NFGL Framework				Stochastic Nodal	Error
No. elements	SIMP	$S_R = 1$	$S_R = 2$	$S_R = 3$	$S_R = 30$	Generation	Diffusion
64×32	1.41	0.44	0.37	0.20	0.14	3.33	0.37
128×64	5.61	1.67	1.08	0.79	1.54	3.47	1.04
256×128	25.57	6.79	3.63	2.60	2.23	3.32	3.62
512×256	105.64	25.21	13.51	10.99	7.40	3.23	13.87
640×320	230.95	41.14	21.29	16.67	12.39	3.53	21.39
768×384	323.60	65.67	32.52	26.00	18.49	3.36	32.99
1024×512	590.11	105.48	53.26	43.06	29.68	3.42	55.51

Table 7 Number of iterations for each topology optimization method used in the cantilever beam

No. elements	Unpenalized SIMP	Local Volume Constraint	Adaptive Quadtree
64×32	32	542	146
128×64	42	505	271
256×128	53	483	296
512×256	54	520	223
640×320	64	519	215
768×384	57	502	269
1024×512	65	N/A	266

To further investigate the computational cost of the NFGL framework and how it performs against the Stochastic Nodal Generation and Error Diffusion methods, the computational cost of each algorithm, excluding the time for performing the SIMP optimization, is plotted against the number of elements in the beam’s mesh. As for the Adaptive Quadtree and Local Volume Constraint methods, they were omitted due to their significantly large computational cost compared to the other algorithms. Figure 63 shows the computational cost plot against the number of elements for the NFGL Framework, Stochastic Nodal Generation, and Error Diffusion methods.

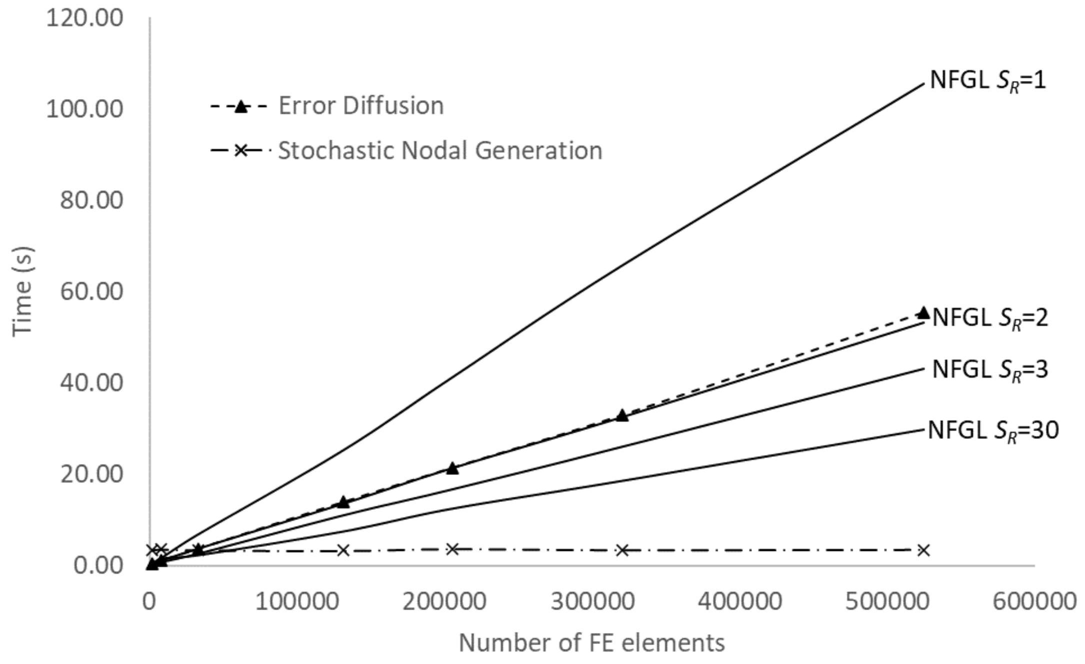


Figure 63 Computational cost of the cantilever beam for the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods against the number of FE elements used

From Figure 63, the computational cost of the Error Diffusion method and NFGL Framework is almost the same when $S_R = 2$ and becomes better as the value of S_R increases, which is a good indication of how well the NFGL Framework would perform against the Error Diffusion method with the increase in the number of input elements. However, the Stochastic Nodal Generation method shows no change in cost as expected. This is due to the fact that it does not require input nodes to generate the NFGL structure, so the structure is generated based on the actual scale of the design domain from randomly placed nodes as explained in section 2.2. But, although this seems to be an advantage to use the Stochastic Nodal Generation method over the other two, it has significant drawbacks when generating NFGL structure that will be discussed in-depth in the next sections.

The reduction in computational cost, as the values S_R of increases, is not linear. The difference in computational cost between $S_R = 2$ and $S_R = 3$ is almost as much as the difference between $S_R = 3$ and $S_R = 30$, which is expected since the algorithm would reach a point where the number of removed nodes will not increase as much to the point where the computational cost would go down in a significant manner.

The generated NFGL structures using all five algorithms are shown in figures 64-74. As expected, the structures generated using the Stochastic Nodal Generation method in Figure 64 look similar and the size of the largest unit-cell size is unaffected by the number of elements since it doesn't rely on the FE mesh to generate the NFGL structure. However, the unit-cell size ratio is not controlled by the method, thus it doesn't change in all the generated NFGL structures by this method. As for the Error Diffusion structures in Figure 65, they are heavily affected by the number of elements used in the generation process. The size of the largest unit-cell reduces as the number of elements increases. Similarly, the Local Volume Constraint method in Figure 67 shows the same behavior. The size of the largest unit-cell reduces as the number of elements increases in the structure. This is because of using a fixed region of elements for the local volume to be constrained in the algorithm. But even with a variable radius, the unit-cell size ratio would still not be controllable since the algorithm would enforce a porosity equal to α locally. As for the Adaptive Quadtree method in Figure 66, it does show a change in the unit-cell size ratio as the number of elements increases in the design domain. However, it is clear that it cannot create conformal designs since it only subdivides the unit-cells. So the generated design only contains lattice struts that are either horizontal or vertical.

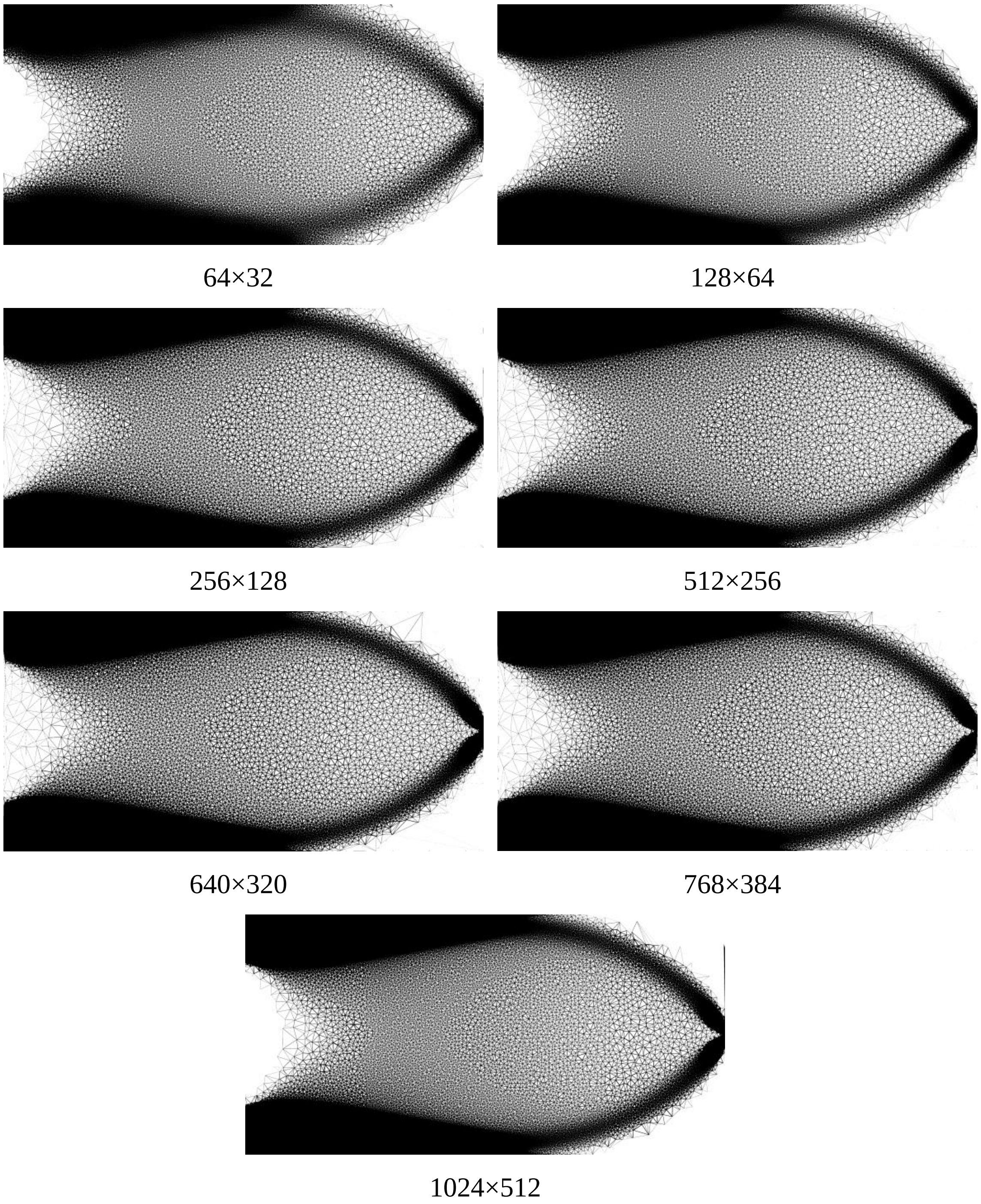
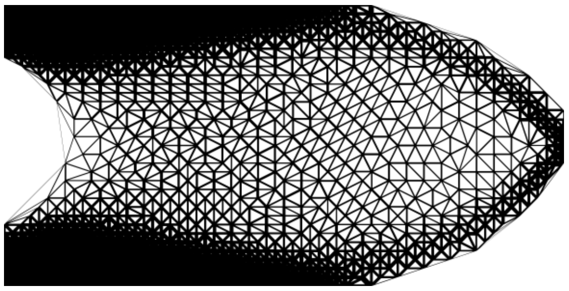
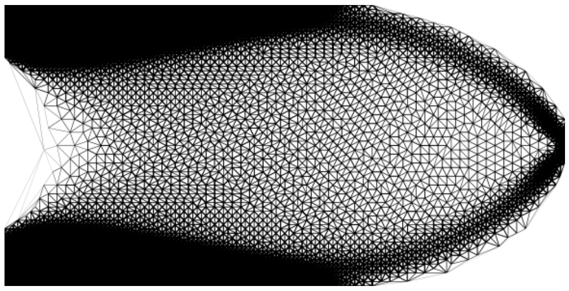


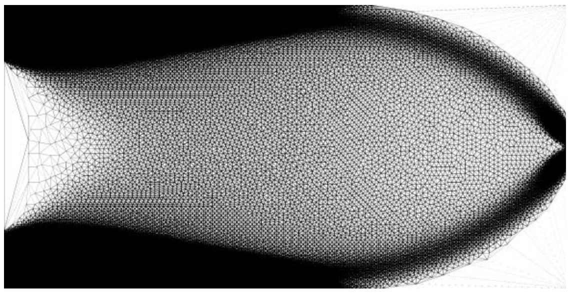
Figure 64 Generated NFGL structures using the Stochastic Nodal Generation method for the cantilever beam at different sizes



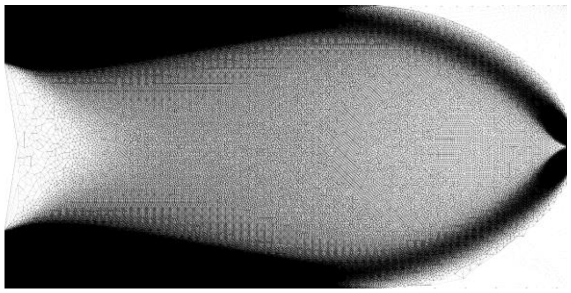
64×32



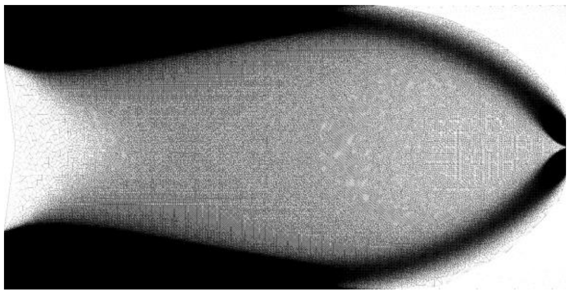
128×64



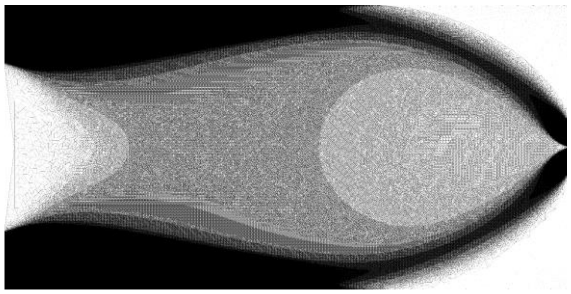
256×128



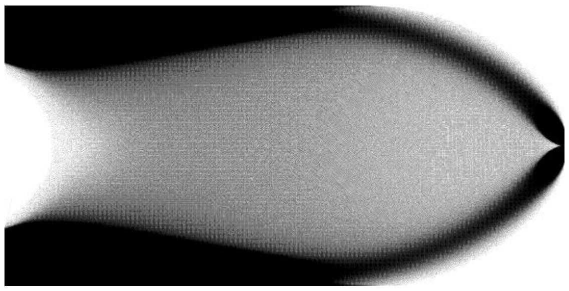
512×256



640×320



768×384



1024×512

Figure 65 Generated NFGL structures using the Error Diffusion method for the cantilever beam at different sizes

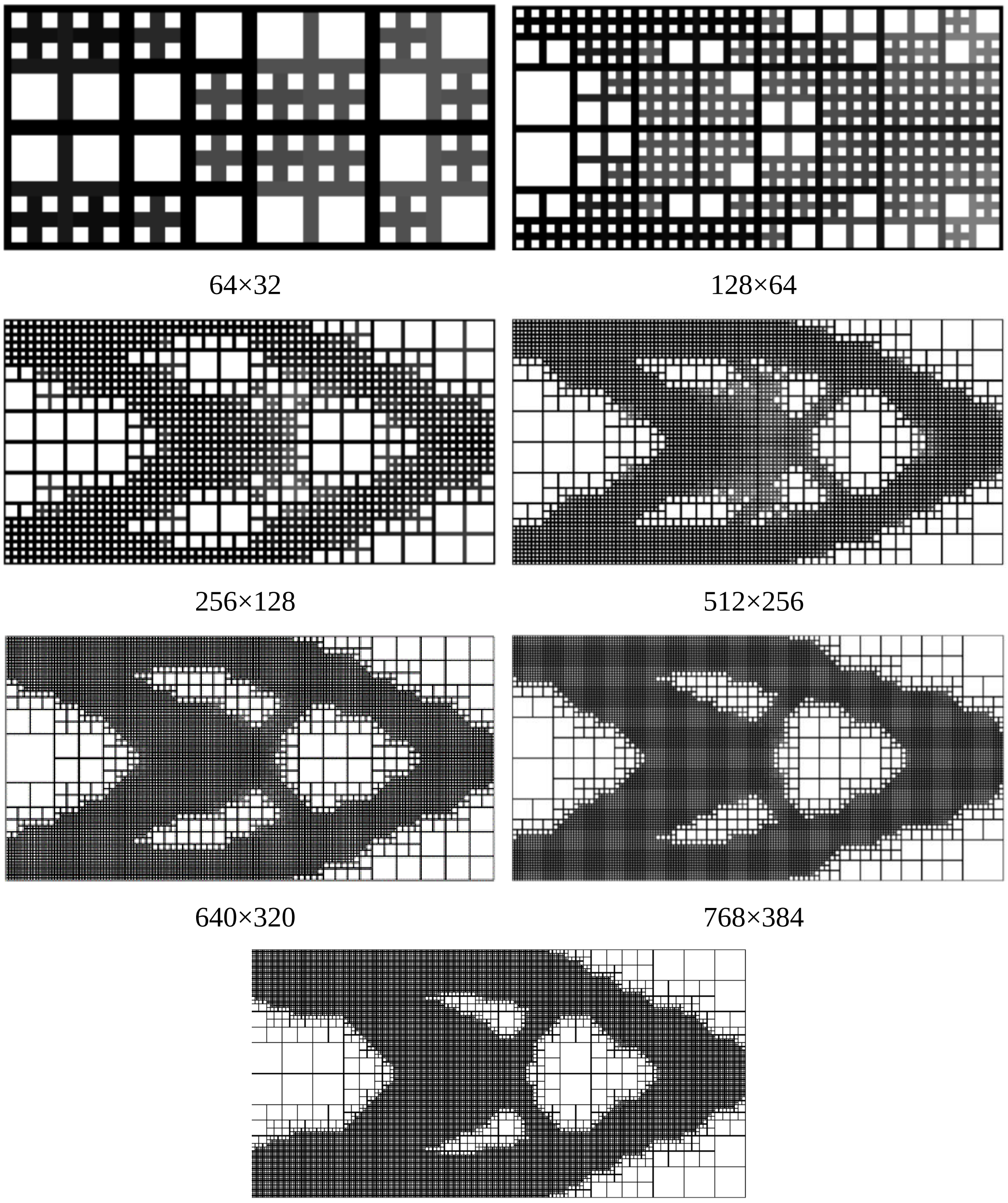


Figure 66 Generated NFGL structures using the Adaptive Quadtree method for the cantilever beam at different sizes

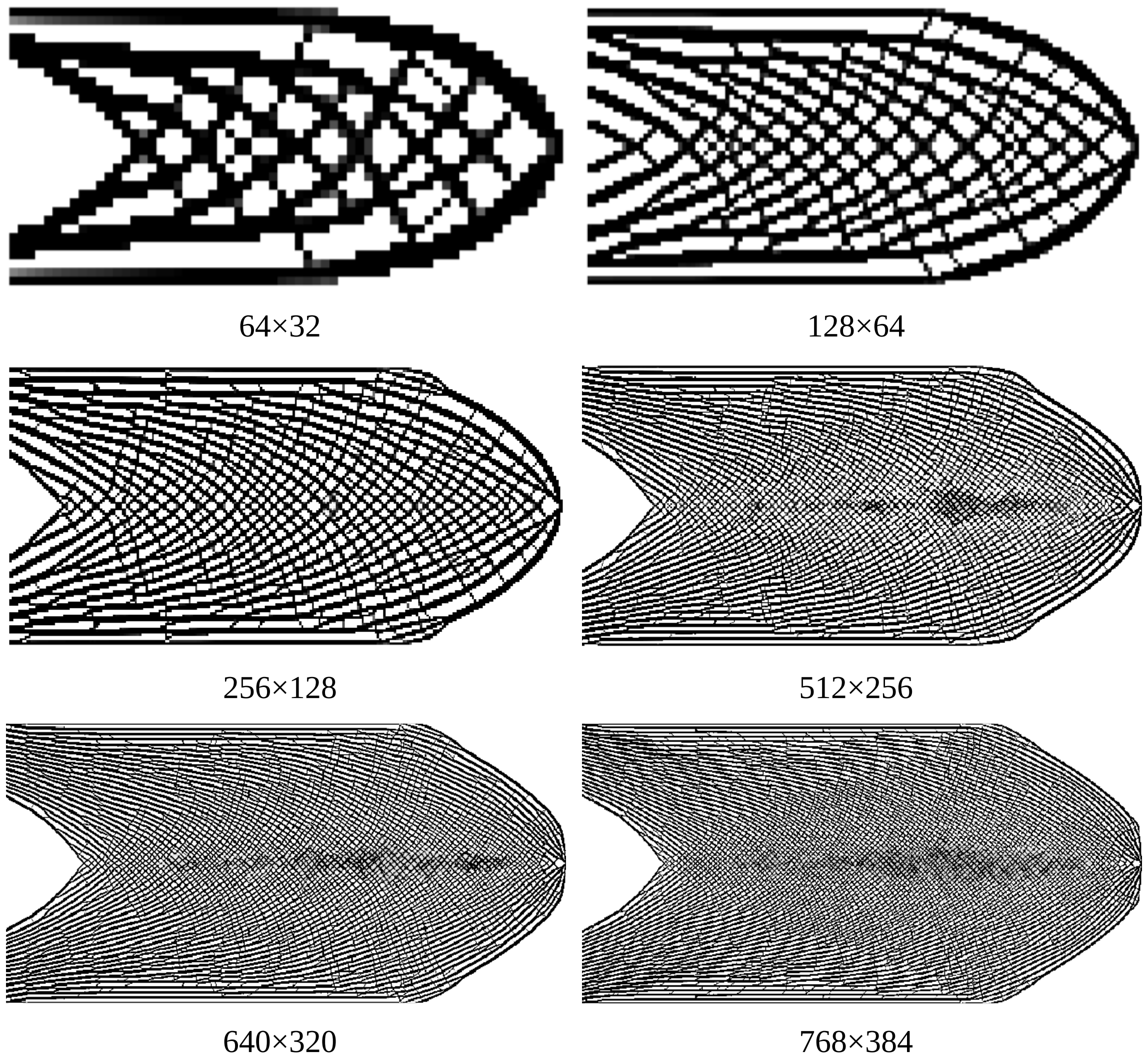


Figure 67 Generated NFGL structures using the Local Volume Constraint method for the cantilever beam at different sizes

Unlike the other algorithms, the NFGL Framework (figures 68-74) is able to generate NFGL structures with different unit-cell size ratios easily. The increase in the number of elements allows for smaller unit-cells to be generated. And since a fixed S_R value was used in all the designs, the largest unit-cells become smaller as the number of elements in the design domain increases. However, larger sizes can still be generated by higher values of S_R easily.

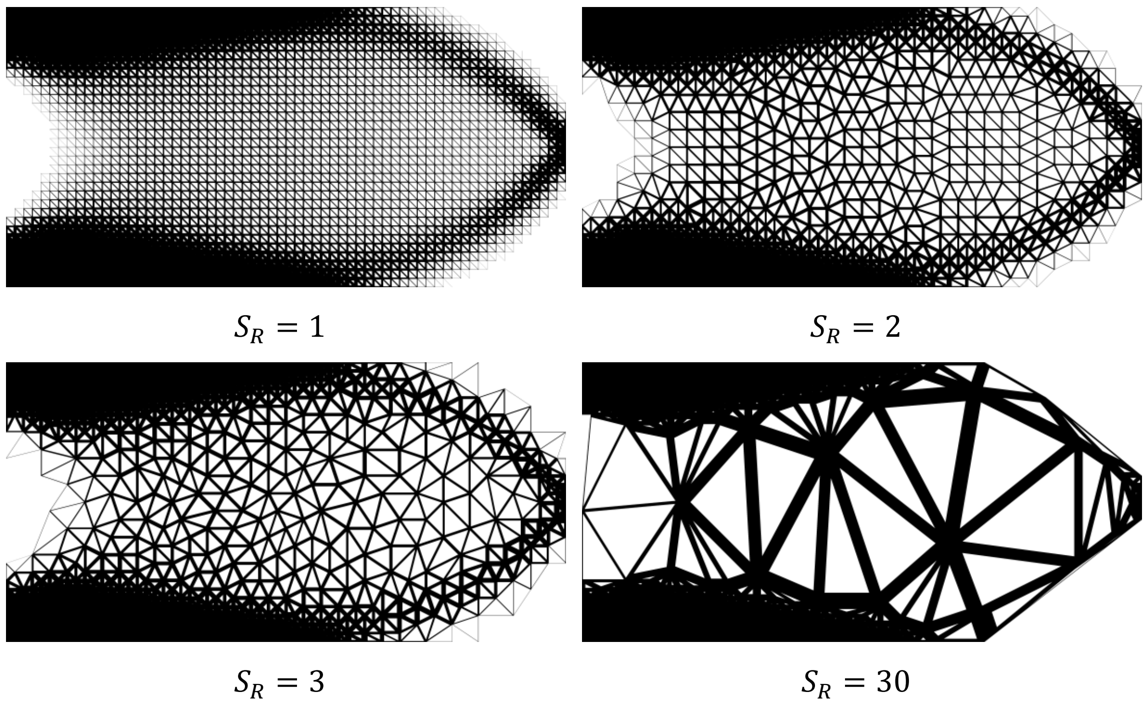


Figure 68 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 64×32 at different S_R values

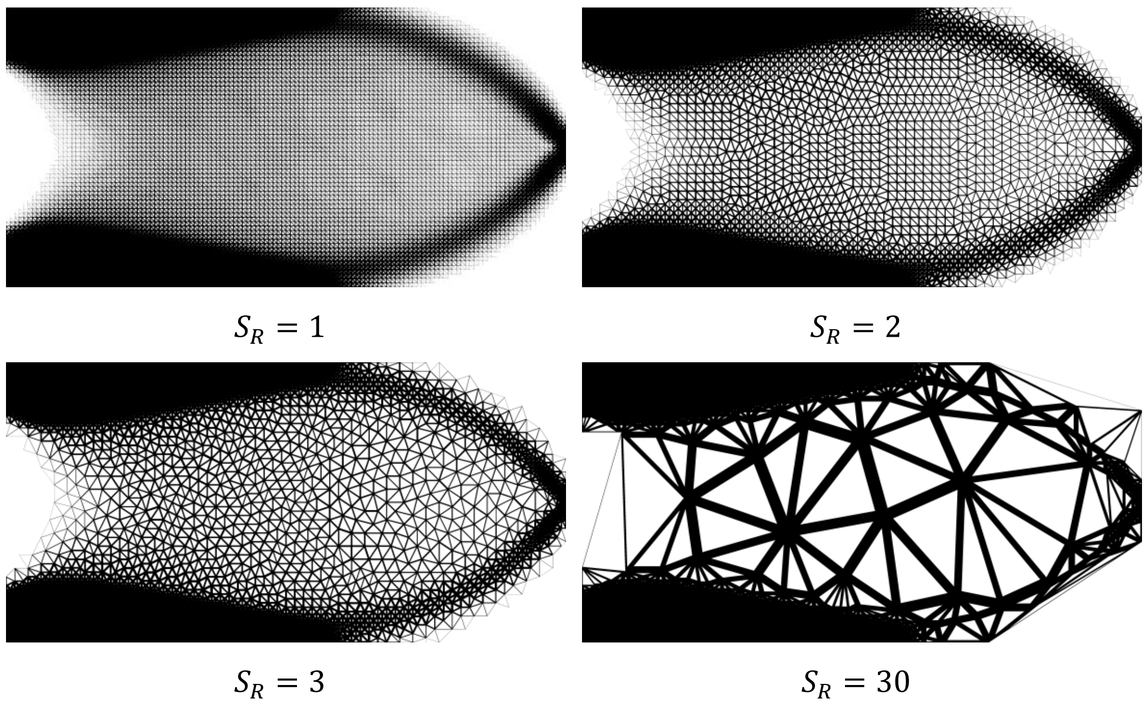


Figure 69 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 128×64 at different S_R values

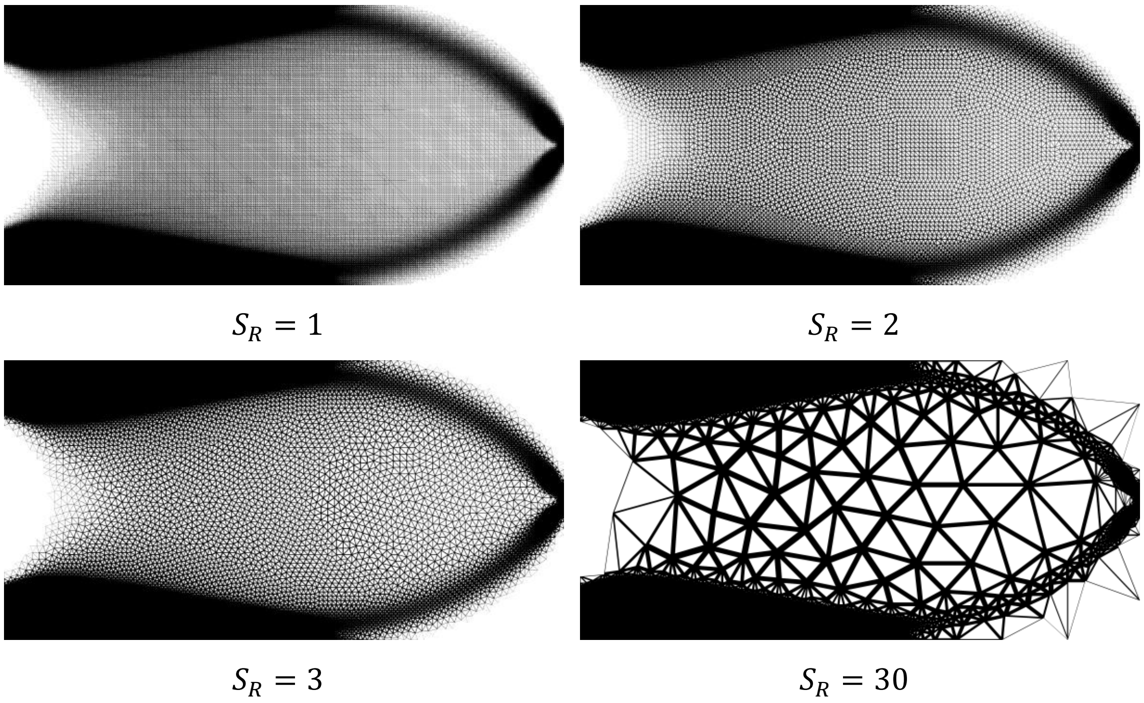


Figure 70 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 256×128 at different S_R values

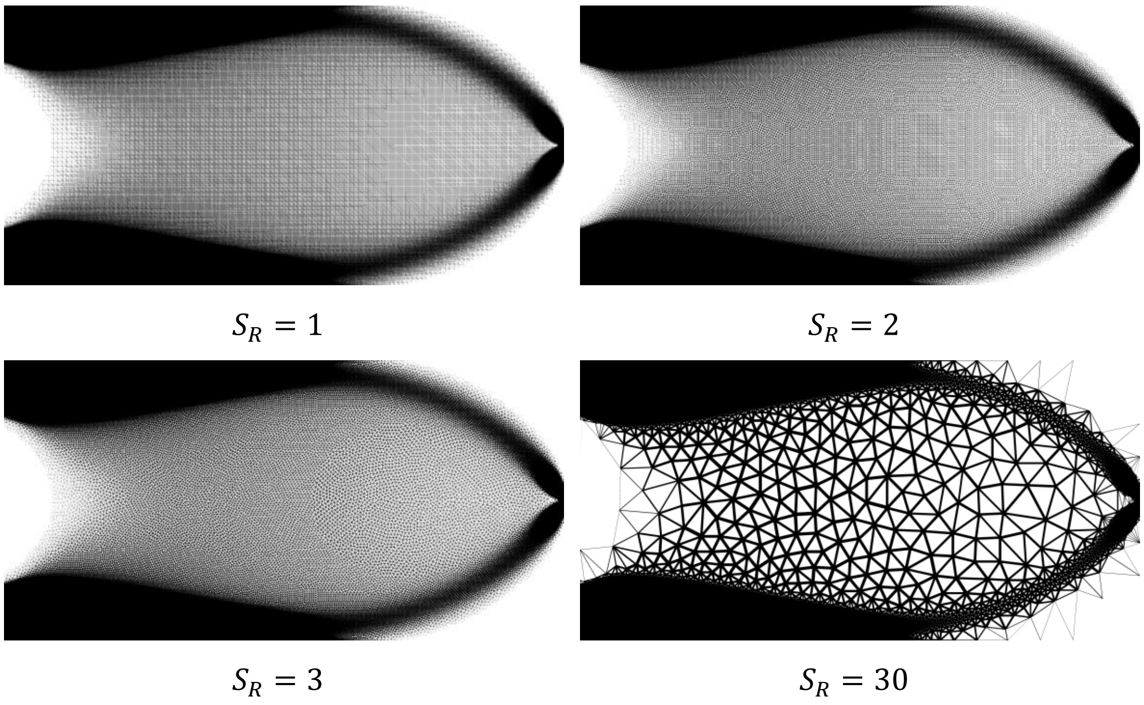
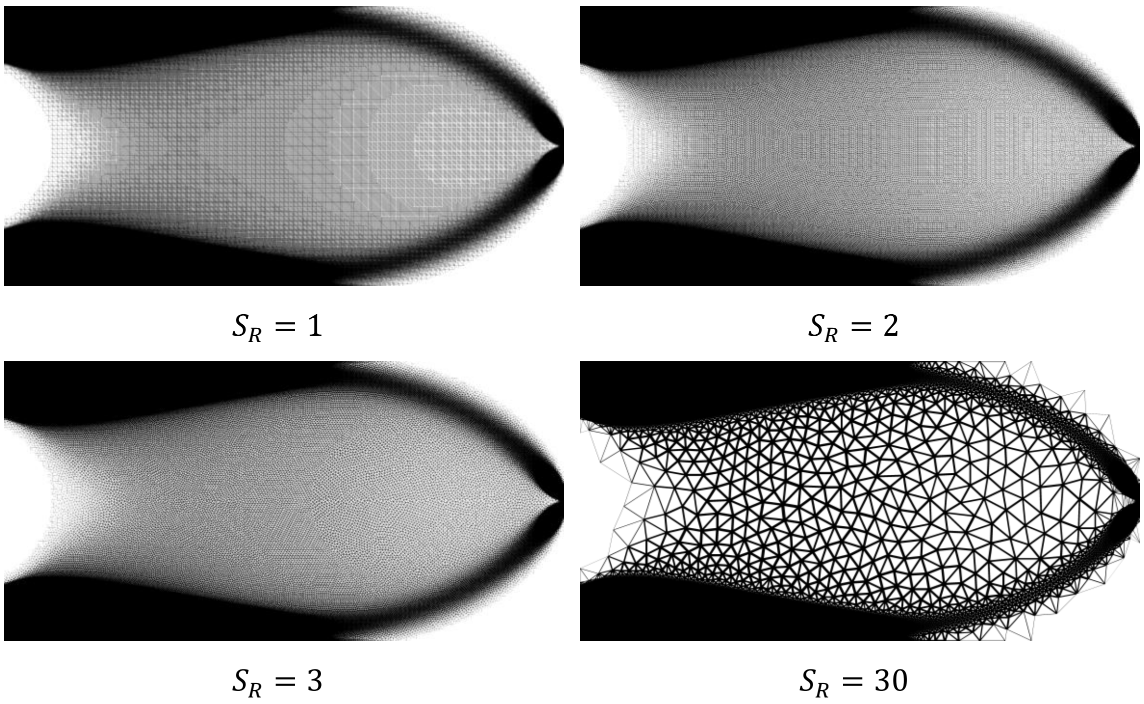


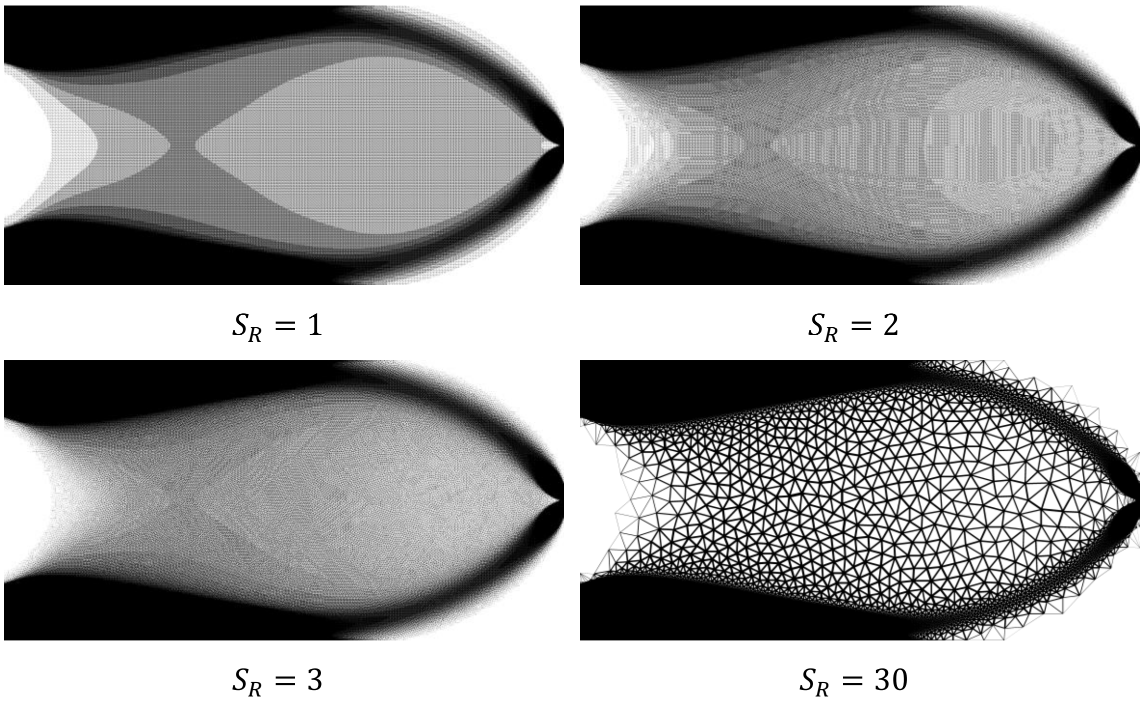
Figure 71 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 512×256 at different S_R values



$S_R = 1$ $S_R = 2$

$S_R = 3$ $S_R = 30$

Figure 72 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 640×320 at different S_R values



$S_R = 1$ $S_R = 2$

$S_R = 3$ $S_R = 30$

Figure 73 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 768×384 at different S_R values

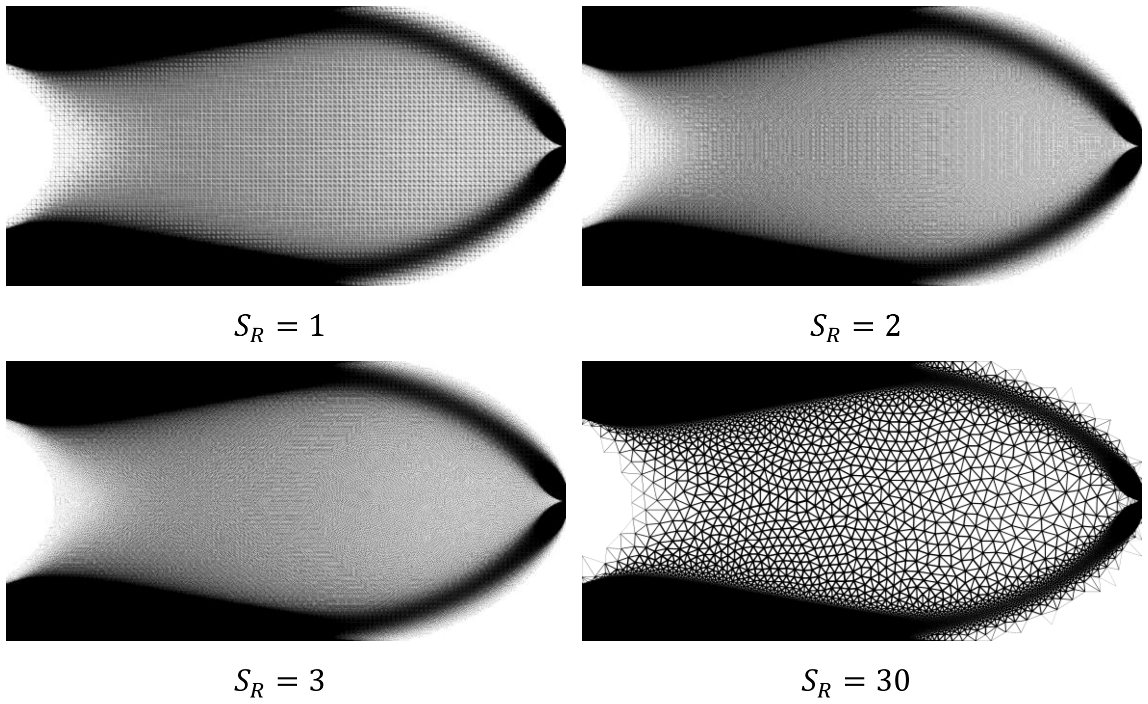


Figure 74 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 1024×512 at different S_R values

As for the simply supported beam case, similar behavior is also seen from all methods. Table 8 shows the computational cost of each method when generating the NFGL structure. Similar to Table 5, the Local Volume Constraint method was the most computationally expensive followed by the Adaptive Quadtree method. As explained previously, this is due to the usage of topology optimization in each iteration during the process of generating the NFGL structure, unlike the other methods which only used the unpenalized SIMP optimization that required significantly less time to finish. Table 9 shows the breakdown of the computational cost of the NFGL Framework, Stochastic Nodal Generation, and Error Diffusion methods. The same trend in Table 6 is also seen here where the computational cost of the NFGL Framework drops drastically when $S_R = 2$ and becomes slightly better than the Error Diffusion computational cost, which further strengthens the recommendation that a high value of S_R is a good starting point for the

NFGL Framework. As for the number of iterations needed for the topology optimization processes, the results are shown in Table 10. The results show a similar trend to the results seen in Table 7. The generated NFGL structures from the methods are also shown in figures 75-85. The structures also show a similar trend to that seen in figures 68-74. The Stochastic Nodal Generation method is unaffected by the mesh size as expected. The Error Diffusion method shows a reduction in the unit-cell sizes as the mesh size increases, which increases the number of nodes used. The Adaptive Quadtree method show unit-cell size variation but with non-conformal unit-cells. The Local Volume Constraint method shows reduced sizes as the size increases due to the fixed region of elements used. The NFGL Framework shows different generated structures at different S_R values and as the value of S_R increases, the structure deviates further from the expected shape in the case where the large unit-cells become too large.

The computational cost for generating the NFGL structure for the simply supported beam is shown in Figure 86. Similar to Figure 63, the NFGL Framework and Error Diffusion showed a similar cost when $S_R = 2$ as the number of elements increases, with the NFGL Framework having lower computational cost as S_R values increases. The Stochastic Nodal Generation method was not affected as expected since it doesn’t generate the NFGL nodes from the input elements but rather based on the scale of the design domain.

Table 8 Computational time to generate the NFGL structure of the simply supported beam using NFGL Framework, Stochastic Nodal Generation, Error Diffusion, Local Volume Constraint, and Adaptive Quadtree methods

	Time							
	NFGL Framework				Stochastic Nodal Generation	Error Diffusion	Local Volume Constraint	Adaptive Quadtree
No. elements	S_R = 1	S_R = 2	S_R = 3	S_R = 30				
64×32	2.24	2.08	2.02	1.95	5.05	2.06	95.54	46.77
128×64	7.39	6.60	6.45	6.03	8.90	6.57	167.18	99.93
256×128	29.34	26.07	25.55	24.72	25.99	26.13	1563.09	375.30
512×256	144.82	132.24	130.12	126.33	122.89	133.35	4020.55	1136.27
640×320	256.90	240.12	236.18	230.46	222.33	240.57	11292	2014.41
768×384	393.55	363.71	358.81	350.48	335.73	365.75	29578	2612.05
1024×512	638.98	589.29	580.23	566.30	540.47	593.33	N/A	4800.69

Table 9 Breakdown of the computational cost of the simply supported beam for the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods

	Time						
	Unpenalized	NFGL Framework				Stochastic Nodal	Error
No. elements	SIMP	S_R = 1	S_R = 2	S_R = 3	S_R = 30	Generation	Diffusion
64×32	1.68	0.55	0.40	0.34	0.27	3.37	0.38
128×64	5.55	1.84	1.05	0.90	0.48	3.35	1.02
256×128	22.72	6.63	3.35	2.83	2.01	3.28	3.41
512×256	119.57	25.26	12.67	10.55	6.76	3.32	13.79
640×320	219.00	37.90	21.12	17.18	11.46	3.33	21.57
768×384	332.49	61.06	31.22	26.33	17.99	3.25	33.27
1024×512	537.14	101.84	52.15	43.09	29.16	3.33	56.19

Table 10 Number of iterations for each topology optimization method used in the simply supported beam

No. elements	Unpenalized SIMP	Local Volume Constraint	Adaptive Quadtree
64×32	35	303	178
128×64	42	331	271
256×128	46	545	300
512×256	54	482	279
640×320	61	504	332
768×384	56	542	273
1024×512	52	N/A	270

From the results presented in this section, it is clear that the NFGL Framework is computationally efficient when compared to other methods and can easily adjust the unit-cell size ratio without additional changes to the input density field or optimization process. Furthermore, it can out perform the Error Diffusion method in terms of computational cost as the value of S_R increases, which further strengthens the recommendation of using higher values of S_R when generating NFGL structures. This also makes it a favorable choice to use when investigating the effects of the unit-cell size ratio on the structures.

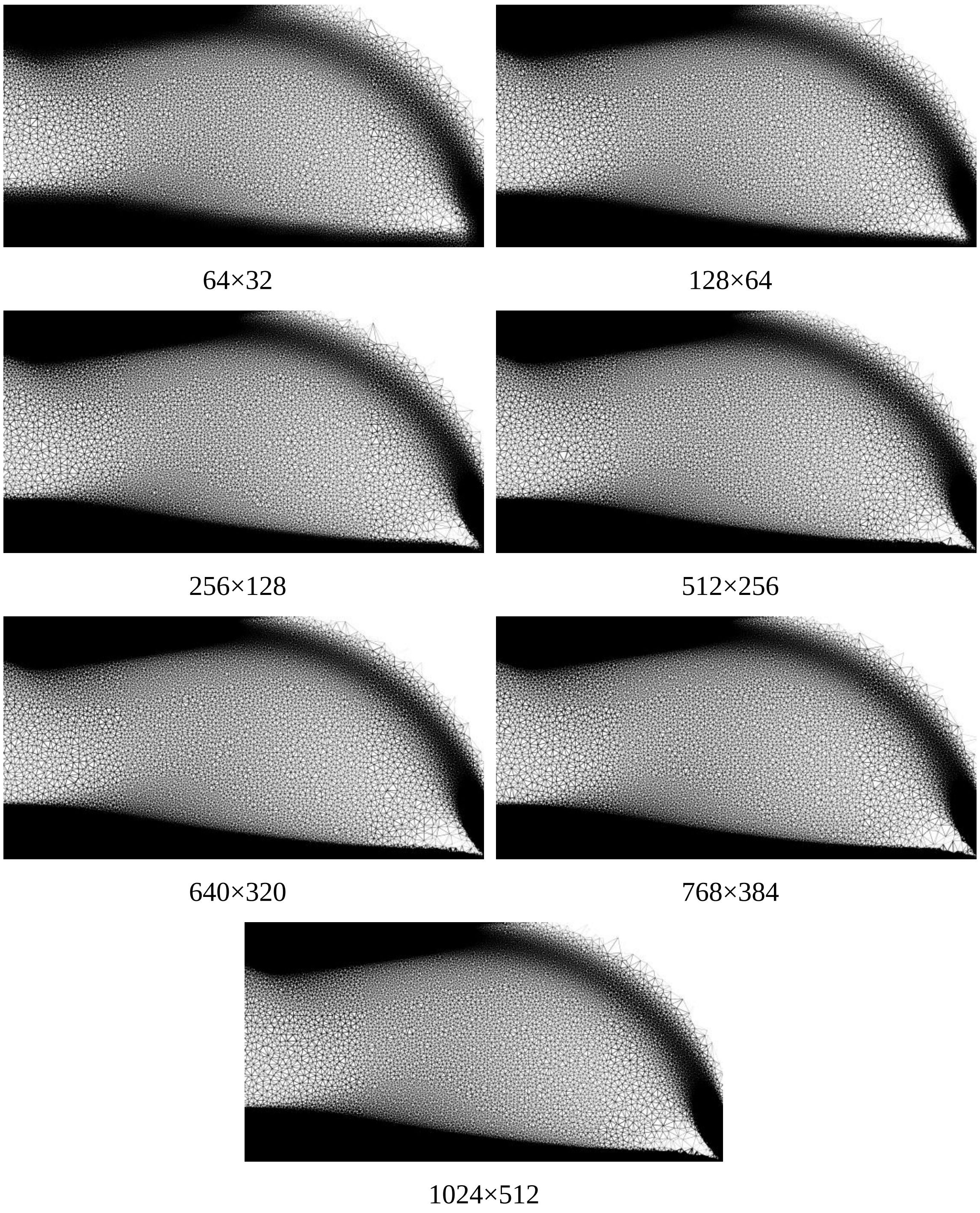


Figure 75 Generated NFGL structures using the Stochastic Nodal Generation method for the simply supported beam at different sizes

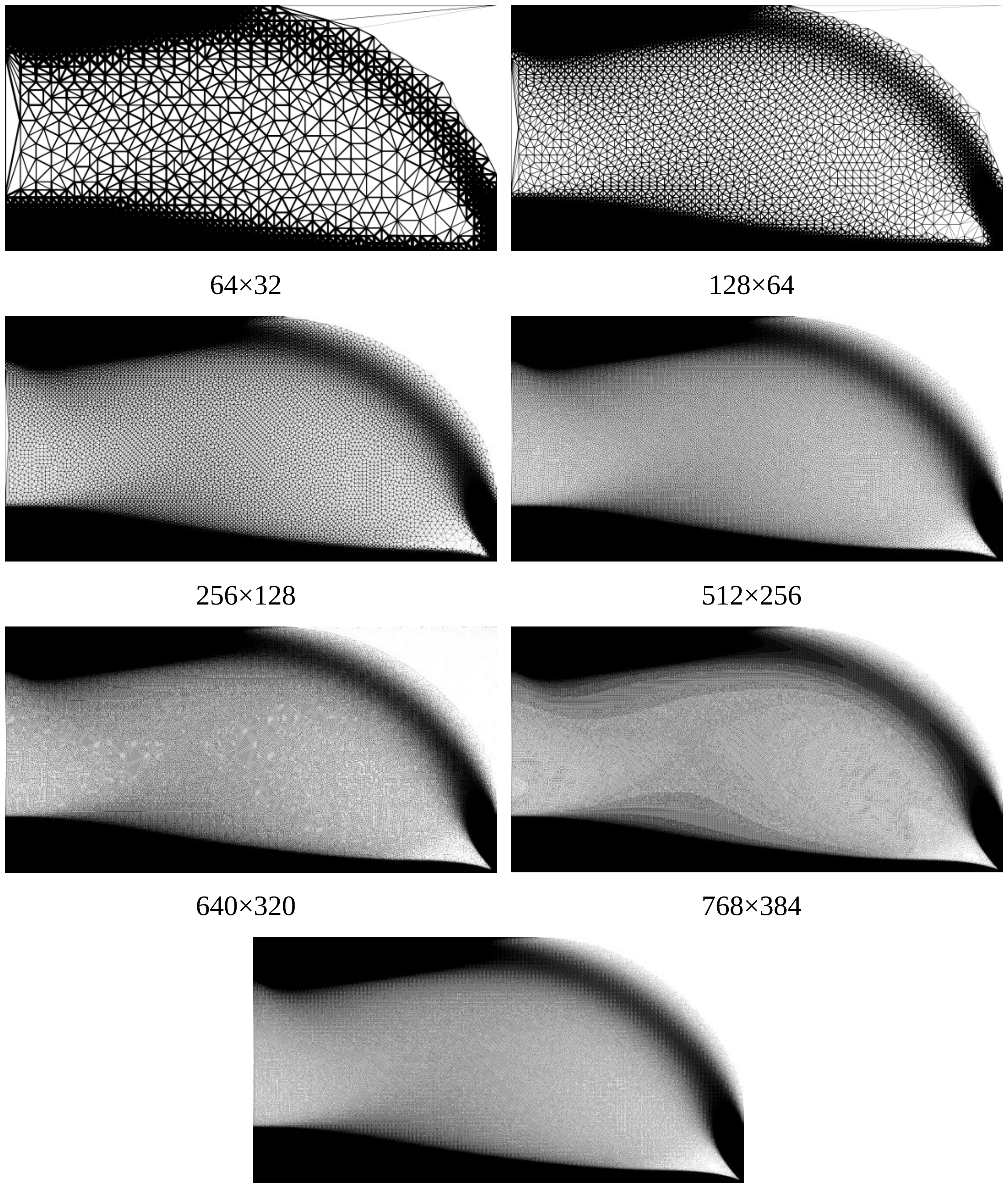


Figure 76 Generated NFGL structures using the Error Diffusion method for the simply supported beam at different sizes

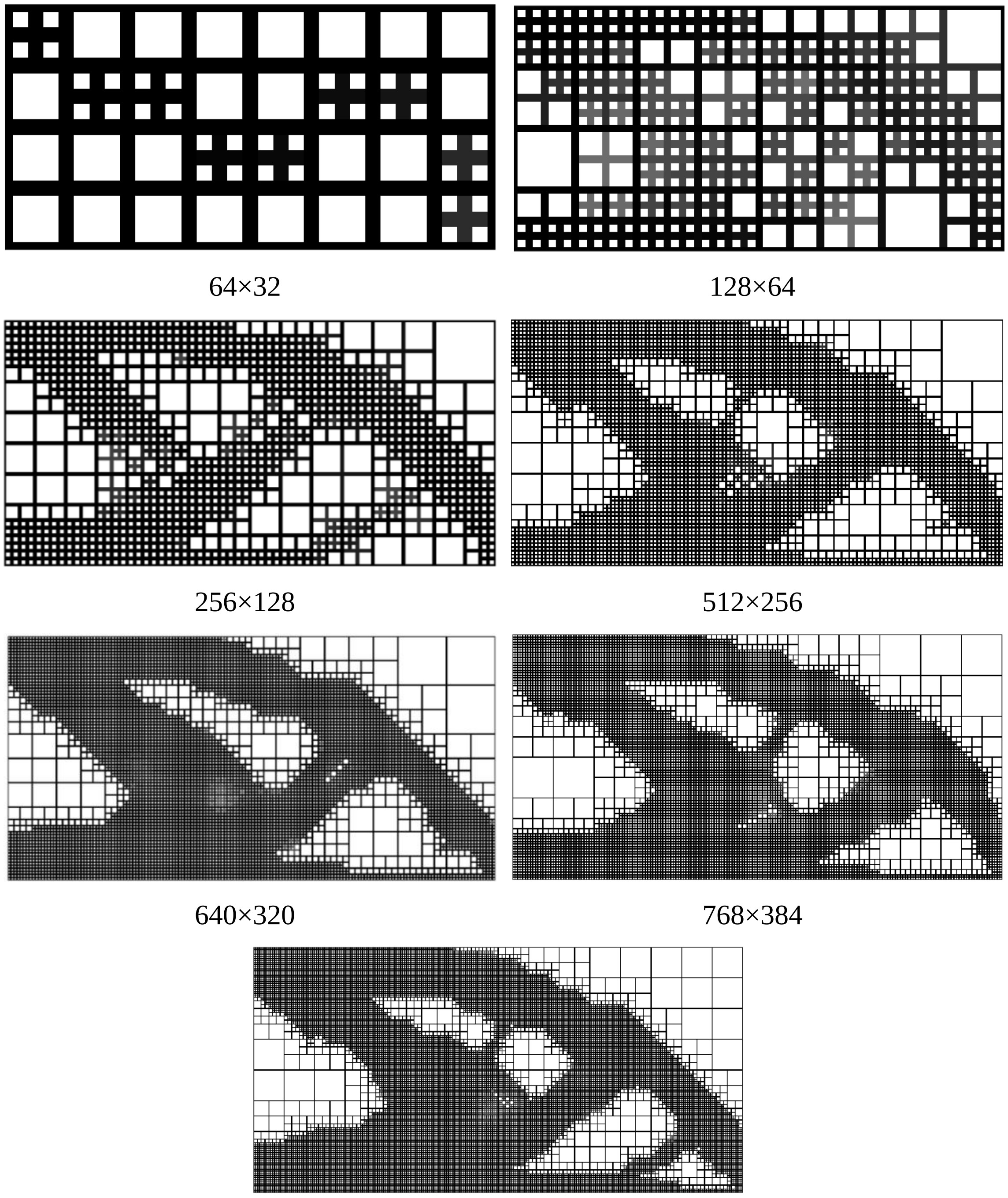
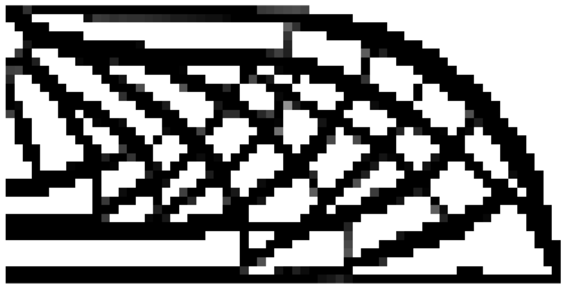
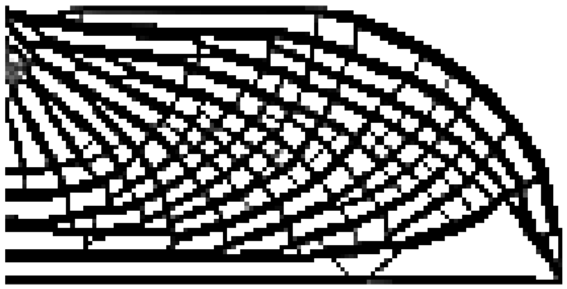


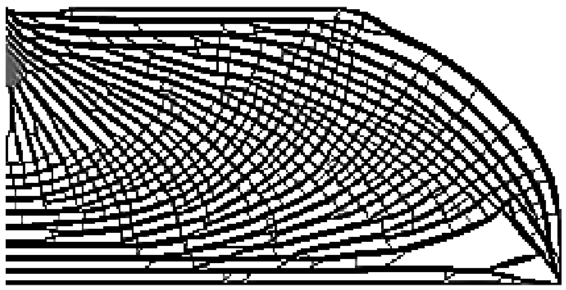
Figure 77 Generated NFGL structures using the Adaptive Quadtree method for the simply supported beam at different sizes



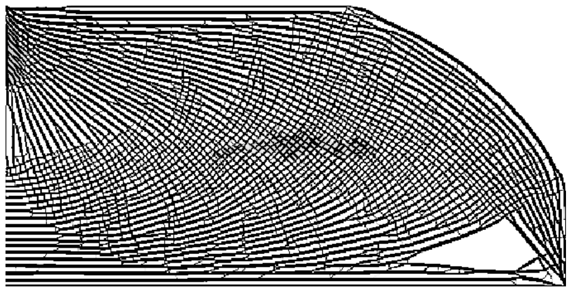
64×32



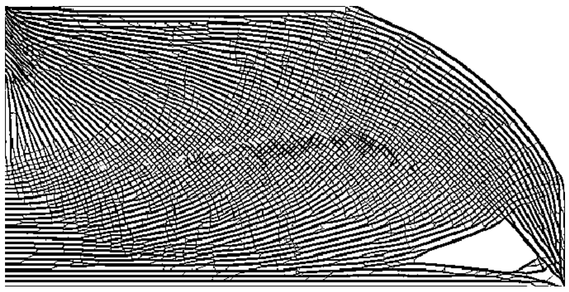
128×64



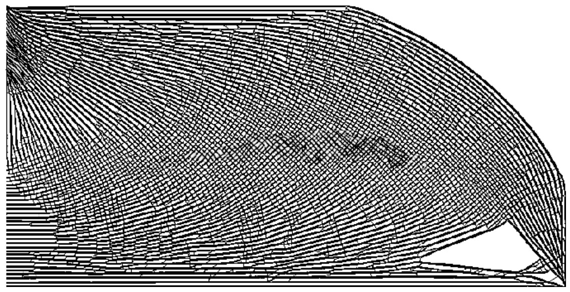
256×128



512×256



640×320



768×384

Figure 78 Generated NFGL structures using the Local Volume Constraint method for the simply supported beam at different sizes

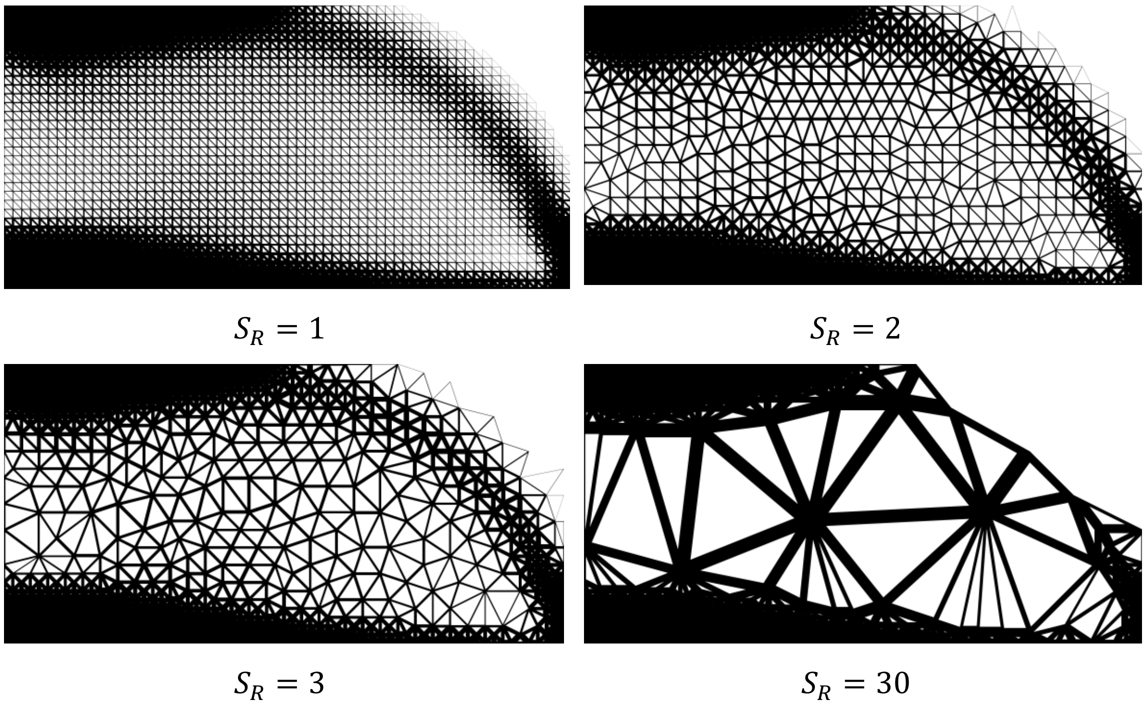


Figure 79 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 64×32 at different S_R values

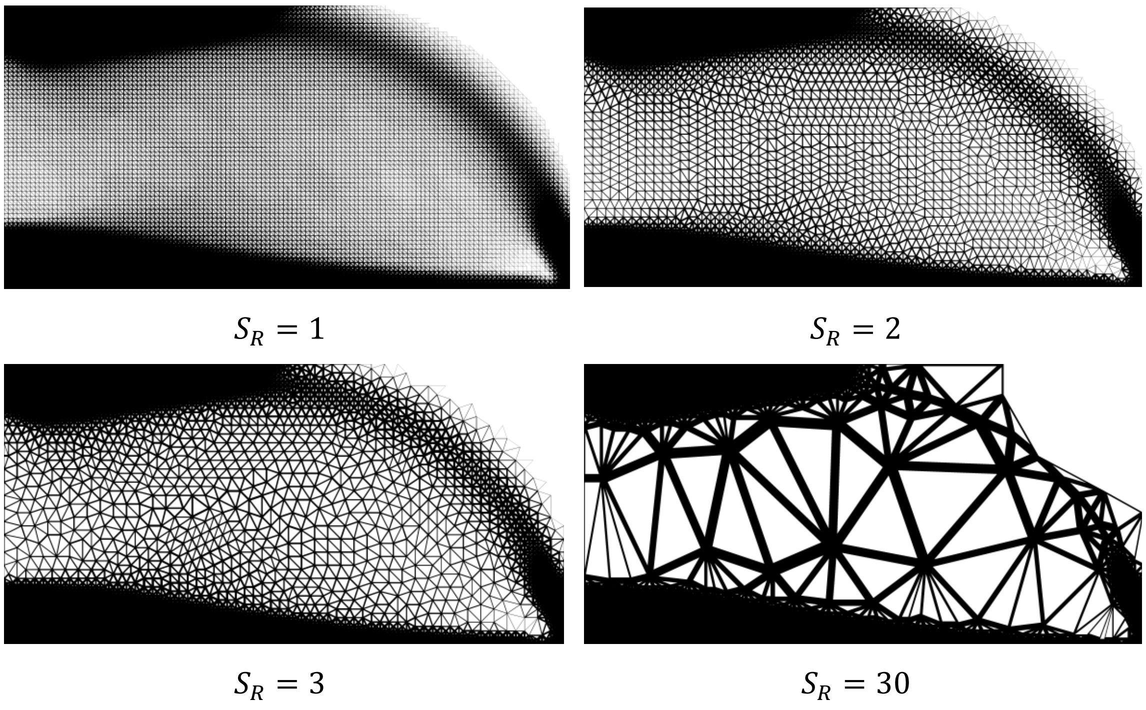


Figure 80 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 128×64 at different S_R values

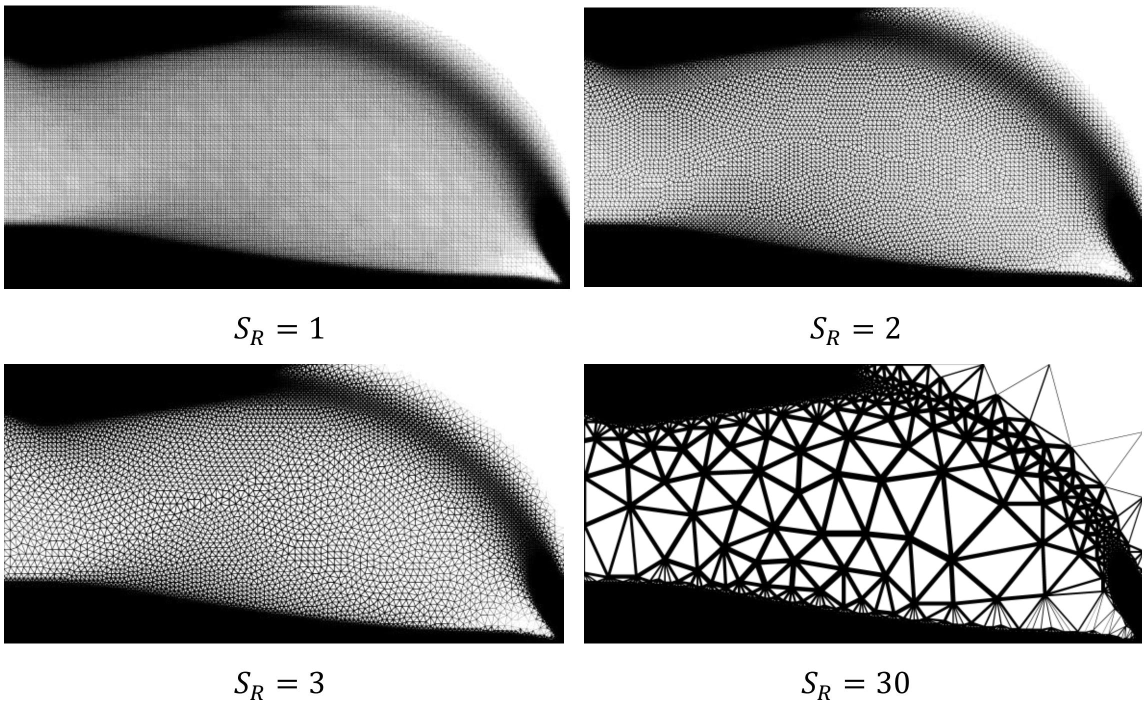


Figure 81 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 256×128 at different S_R values

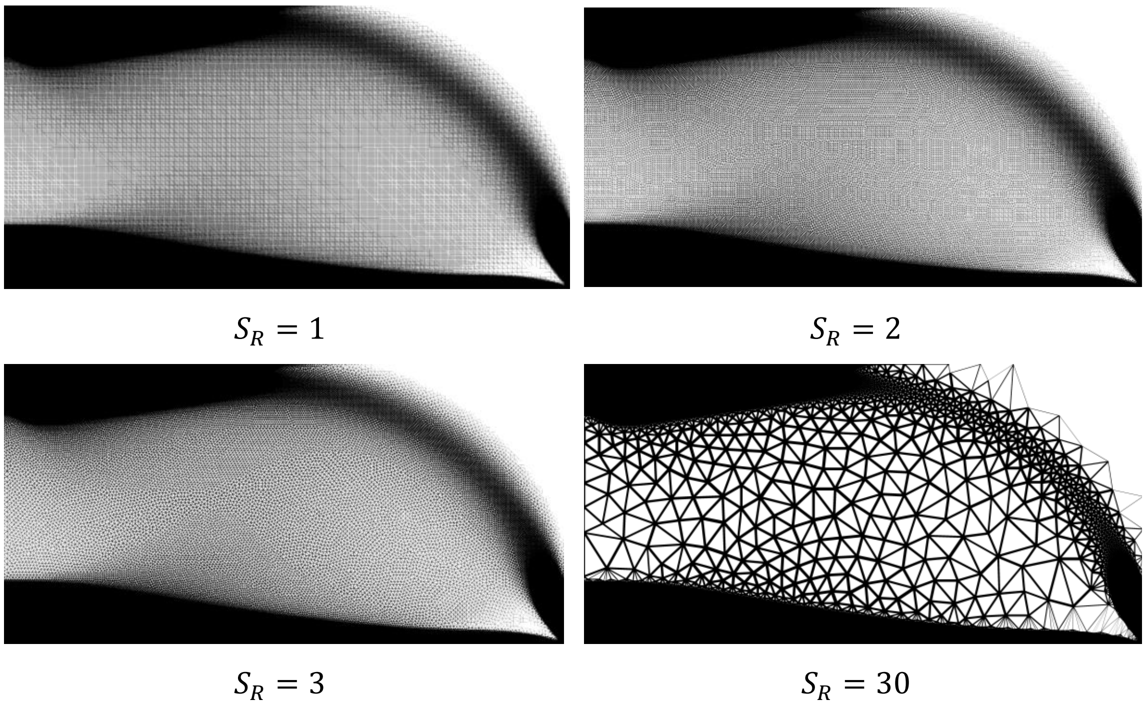


Figure 82 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 512×256 at different S_R values

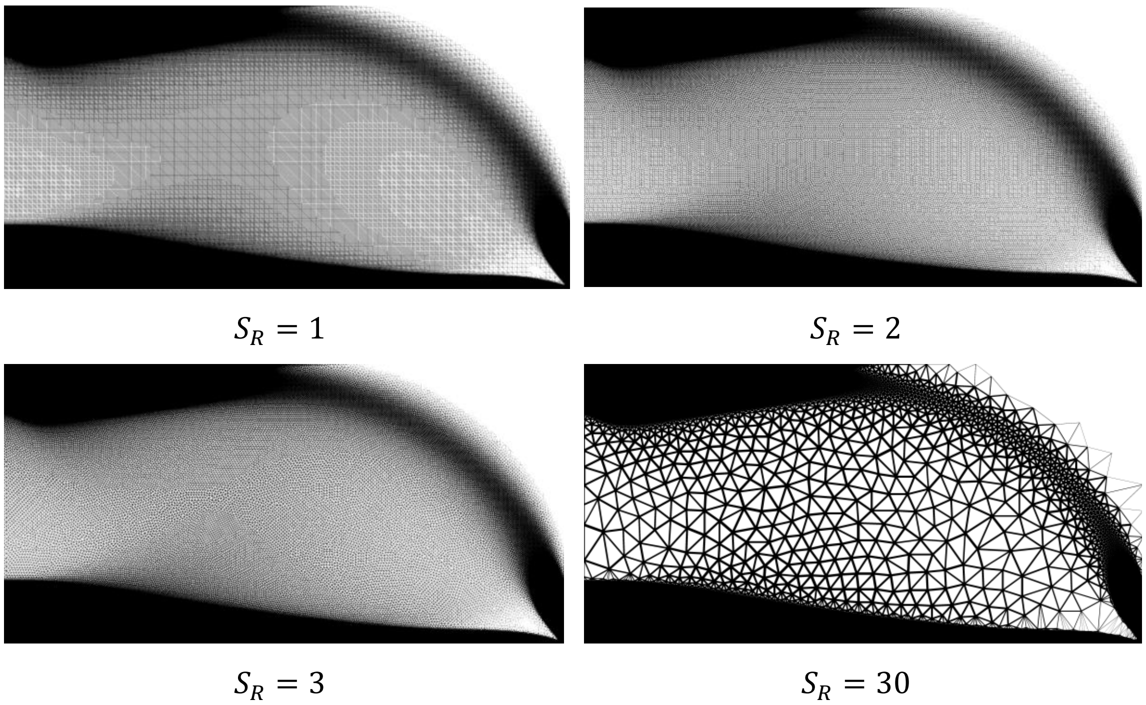


Figure 83 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 640×320 at different S_R values

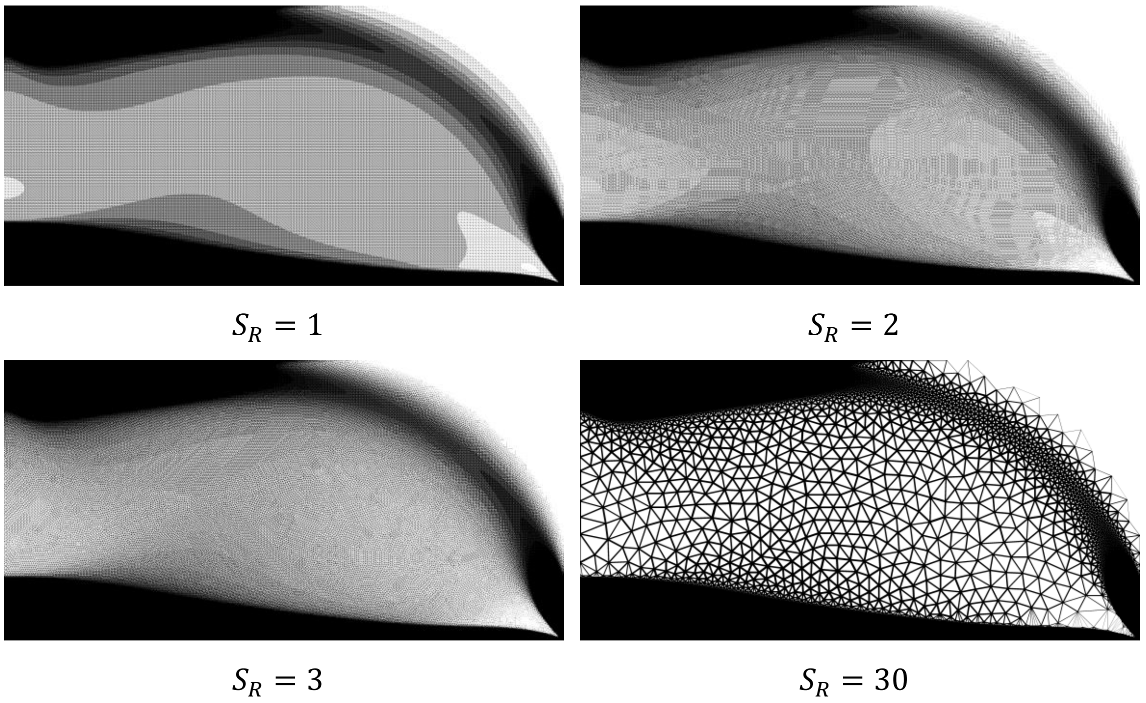


Figure 84 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 768×384 at different S_R values

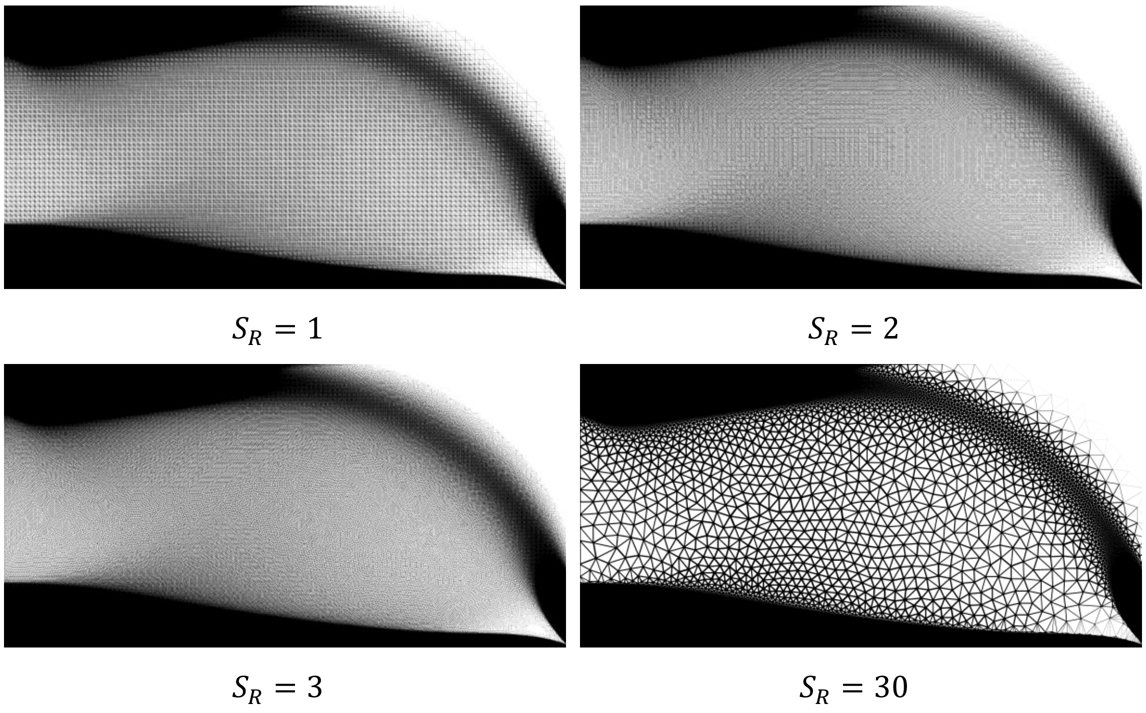


Figure 85 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 1024×512 at different S_R values

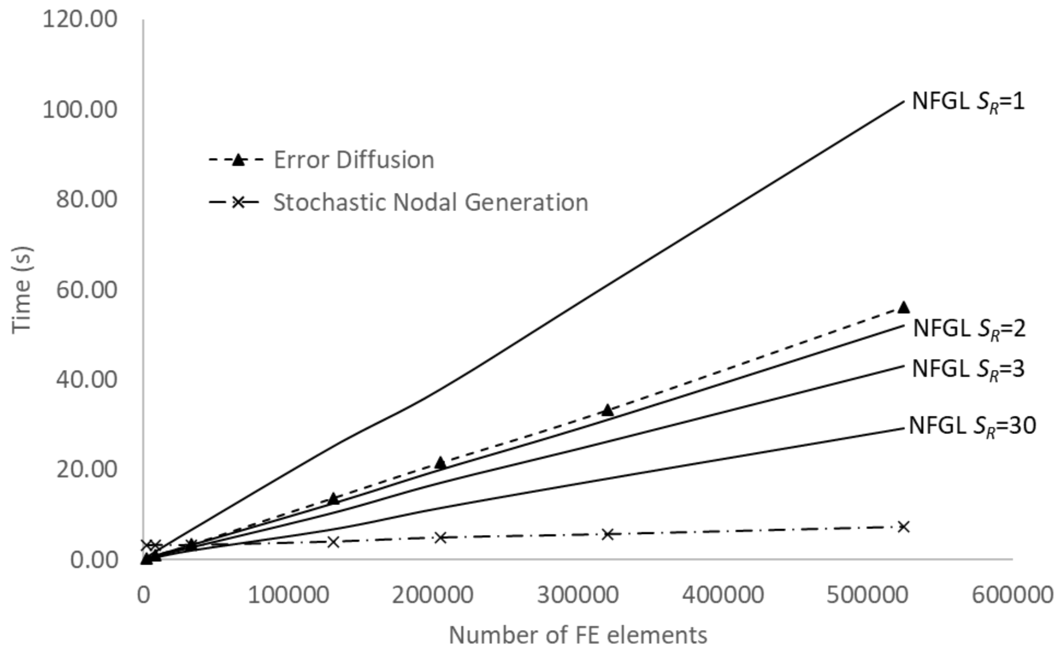


Figure 86 Computational cost of the simply supported beam for the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods against the number of FE elements used

4.1.2 Structural Performance

To evaluate the structural performance of the NFGL Framework with the other algorithms used in section 4.1.1, the compliance of the generated NFGL structures for the cantilever beam and simply supported beam, shown in Figure 61, will be calculated. Furthermore, the effects of changes in the size of the unit-cells on the generated NFGL structure's compliance will be investigated. Therefore, different NFGL structures using the NFGL framework will be generated with different large influence sphere radii, R_L, to generate a fixed large unit-cell sizes. The initial value of R_L will the same as R_S, while the other values that will be used are 2, 3, 5, 15, 30, 50, and 90mm. Furthermore, the MSSIM index will be calculated for these structures in order to investigate the effects of changes in the unit-cell size ratio on the compliance of the generated NFGL structures. As for the number of elements that will be used as in input for each algorithm, due to the high computational cost of the Adaptive Quadtree and the Local Volume Constraint methods, only the size 256×128 will be used. This size is similar to the size used in the literature for these methods. As for the other three methods, only three sizes will be used, which are 256×128, 640×320, and 768×384. The smallest size was chosen such that it uses the same size as the Adaptive Quadtree and Local Volume Constraint methods, while the largest size was chosen because it had results produced for the Local Volume Constraint method, which would aid in comparing the structure performance of all other methods with it.

Table 11 shows the compliance of the generated NFGL structures at the sizes mentioned for the cantilever beam. From the table, the Adaptive Quadtree approach has the worst compliance compared to the other designs. And with its high computational cost, the Adaptive Quadtree method becomes unfavorable to use. The Local Volume Constraint method shows better results than the Adaptive Quadtree method, but the improvement in the compliance compared to the other three methods does not justify the extremely high computational cost. Thus, the Local Volume Constraint method is also unfavorable to use. As for the Stochastic Nodal Generation method, the results show that it has worse compliance compared to the Error Diffusion method. And since the generated structure by the Stochastic Nodal Generation method is unaffected by the size of the input elements, the compliance value only fluctuates due to the stochastic nature of the generated NFGL structure. However, the Error Diffusion compliance seems to get worse as the number of elements increases. This is due to the size reduction in the size of the largest unit-cells in the NFGL structure when the number of elements increases. As for the NFGL Framework results, the compliance when no variation in unit-cell size is introduced ($S_R = 1$) was high and gets worse as the number of elements increases. However, when the unit-cell size variation is introduced into the generated NFGL structure, the compliance results improve and become better in most cases than the other designs. After a certain increase in R_L values, the compliance begins to worsen again. This is because the generated NFGL structure starts to deviate from the input density field distribution. Figure 87 shows how the compliance changes for the NFGL structure at different R_L values while Figure 88 shows the compliance change at different S_R values. From the figure, the compliance value reduces and plateaus at a certain range of R_L values. Then as explained earlier, the compliance increases as the structure deviate more from the input density field distribution. Figures 89-91 how the generated NFGL structures from the NFGL Framework.

Table 11 Compliance of different NFLG structure for the cantilever beam using the NFGL Framework, Stochastic Nodal Generation, Error Diffusion, Adaptive Quadtree, and Local Volume Constraint methods

No. elements	Compliance (N.m)											
	NFGL Framework								Stochastic Nodal Generation	Error Diffusion	Adaptive Quadtree	Local Volume Constraint
	S_R=1	R_L (mm)										
		2	3	5	15	30	50	90				
256×128	90.38	84.44	81.97	80.07	79.97	80.00	81.61	101.61	87.61	81.84	121.21	86.02
640×320	86.33	83.07	79.94	79.92	79.13	79.80	82.66	92.02	87.05	88.42	107.21	89.23
768×384	84.85	80.78	80.05	79.77	78.42	78.92	79.17	89.76	90.44	136.11	107.22	91.26

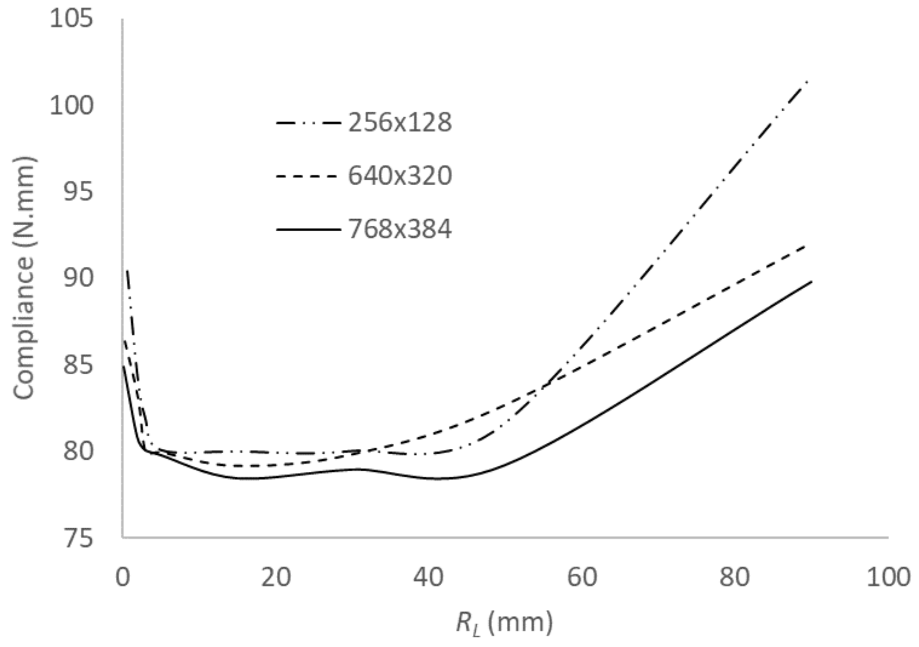


Figure 87 Change in compliance of the generated NFGL structures for the cantilever beam using the NFGL Framework as a function R_L

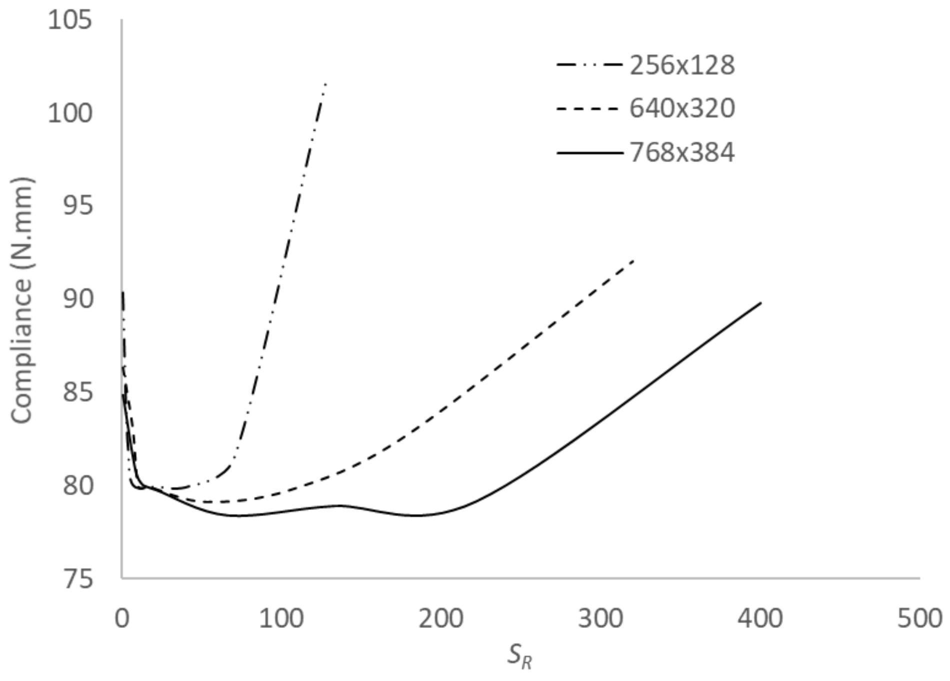


Figure 88 Change in compliance of the generated NFGL structures for the cantilever beam using the NFGL Framework as a function S_R

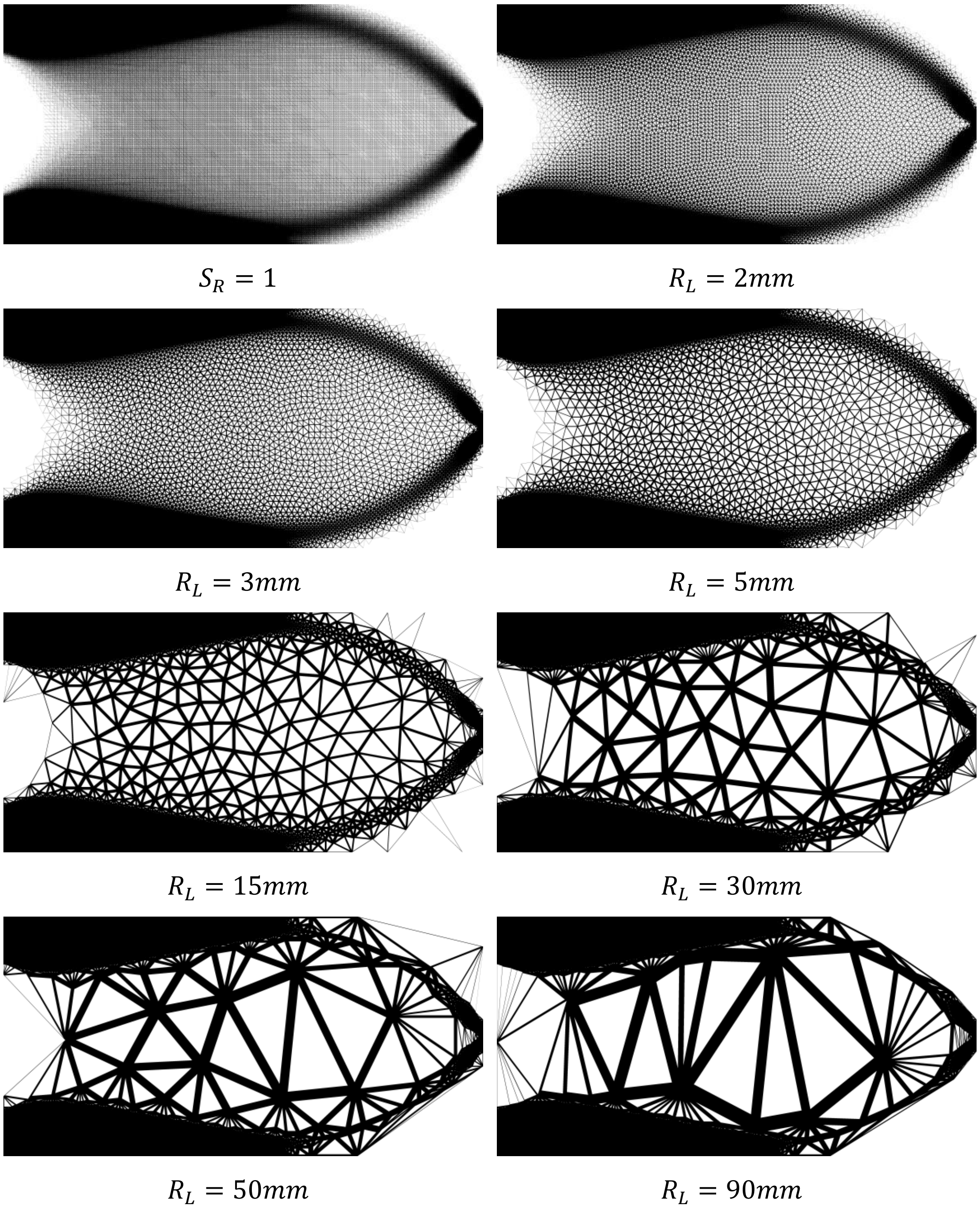


Figure 89 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 256×128 at R_L values

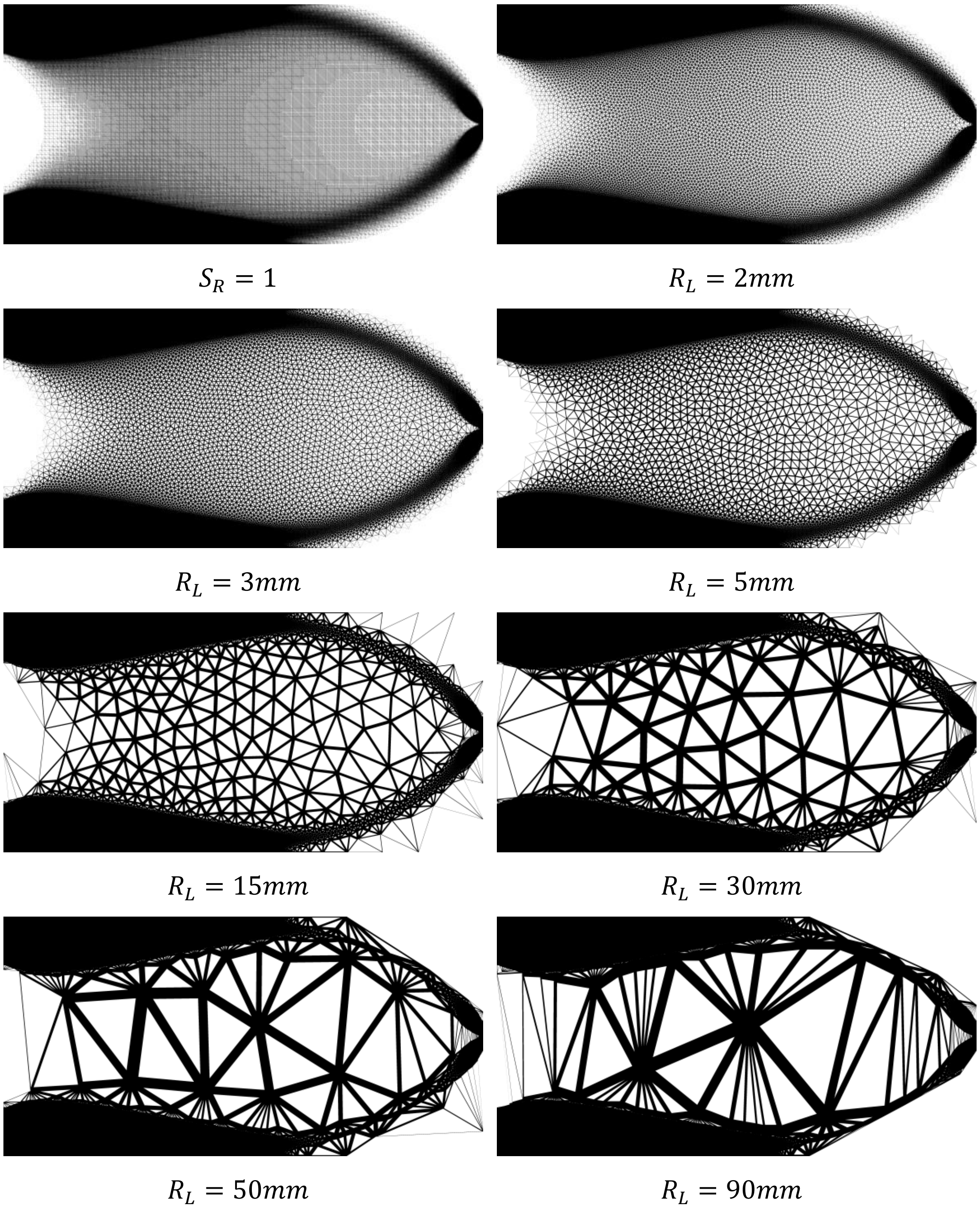


Figure 90 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 640×320 at R_L values

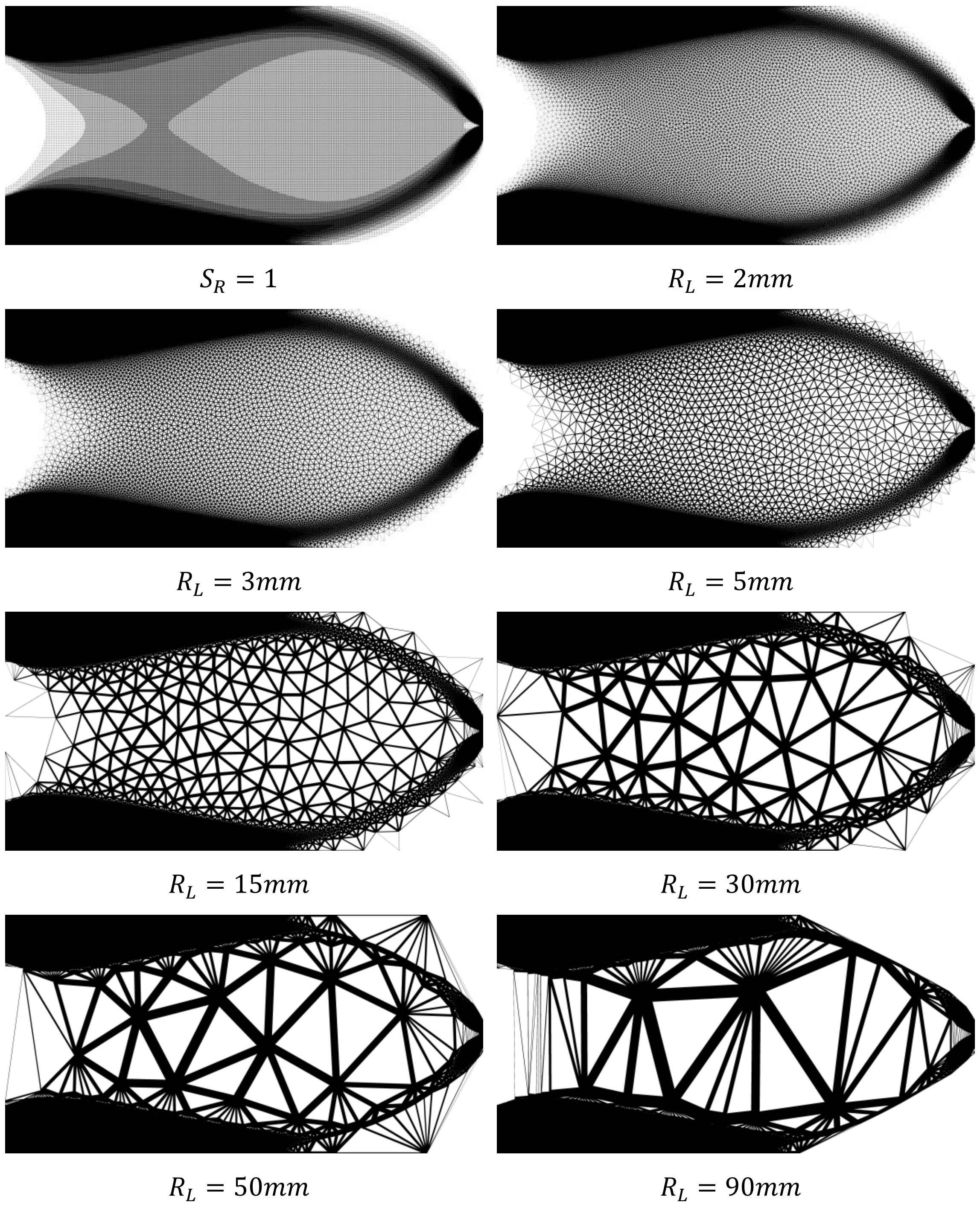


Figure 91 Generated NFGL structures using the NFGL Framework for the cantilever beam of size 768×384 at R_L values

In order to better understand how the structure begins to deviate from the input density field, the MSSIM index is calculated for all of the generated NFGL structures from the NFGL Framework. Figure 92 shows the change in the MSSIM index as S_R values changes. The change in the MSSIM index shows that it behaves as a power function relation with S_R similarly to what was observed in Figure 60 for all three sizes. Based on these results, an MSSIM index threshold of 70-75% shows to be a good range where the NFGL structure would show improvements over a regular FGL structure. So this threshold will be tested on the next examples to observe if it does indeed provide an improvement to the NFGL structure. Algorithm 3 and Algorithm 4 were applied to investigate how the NFGL Framework would determine an appropriate S_R value that falles within the required threshold. Table 12 shows the results for all three sizes. The time it took to run the algorithms was 10.9, 107.3, and112.8 seconds for the 256×128, 640×320, and 786×384 sizes respectively. The reason why the 640×320 and 786×384 times were similar, was because the 786×384 reached the threshold in 3 iterations as shown in the table. The determined S_R values in the table do fall within the required, thus showing that a power function fitting is sufficient to arrive at an appropriate S_R value.

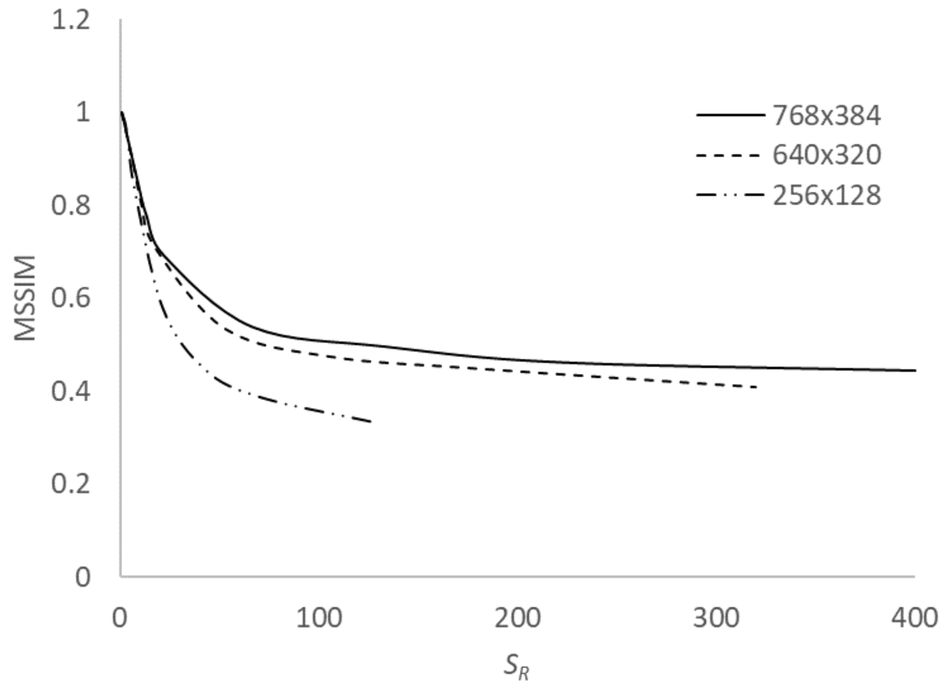


Figure 92 Change in the MSSIM index as a function of S_R for the cantilever beam

Table 12 Determining S_R for the cantilever beam at different mesh sizes using Algorithm 3 and Algorithm 4

No. elements	MSSIM	S_R
256×128	0.95	5
	0.29	486.5
	0.76	14.0
	0.73	14.3
640×320	0.97	5
	0.35	524.4
	0.49	52.9
	0.79	14.3
	0.74	16.5
768×384	0.95	5
	0.31	876.5
	0.71	21.5

Similarly, the same behavior in the cantilever beam case is seen in the simply supported beam case. Table 13 shows the compliance values for the different methods while Figure 93 and Figure 94 show the compliance change for the NFGL framework structures for different R_L and S_R values respectively. The generated NFGL structures for the simply supported beam are shown in figures 95-97. As for the MSSIM index, the results also show similar behavior to Figure 92. The suggested threshold of 70-75% does show improvement in the NFGL structure. Thus Algorithm 3 and Algorithm 4 were used to see if the NFGL Framework can determine the appropriate S_R values for the three sizes accordingly. The computational time for using the algorthms was 10.6, 93.1, and 184.6 seconds for the 256×128, 640×320, and 786×384 sizes respectively. Table 14 shows the results of running the algorithms. The results from the table show that the algorithms did indeed reach an S_R value that falls within the threshold of 70-75% without a significant increase in the computational cost compared to using the Adaptive Quadtree and Local Volume Constraint Methods and with improved results compared to the Stochastic Nodal Generation and Error Diffusion methods.

Table 13 Compliance of different NFLG structure for the simply supported beam using the NFGL Framework, Stochastic Nodal Generation, Error Diffusion, Adaptive Quadtree, and Local Volume Constraint methods

	Compliance (N.m)											
No. elements	NFGL Framework								Stochastic Nodal Generation	Error Diffusion	Adaptive Quadtree	Local Volume Constraint
	S_R=1	R_L (mm)										
		2	3	5	15	30	50	90				
256×128	118.25	109.62	107.76	106.34	107.26	106.83	106.23	111.19	116.77	107.63	139.02	108.29
640×320	115.44	106.82	106.27	105.46	105.89	105.63	111.21	124.39	116.07	122.99	136.70	113.30
768×384	122.52	107.13	106.50	105.85	105.84	105.32	107.41	120.15	115.81	193.96	138.09	114.92

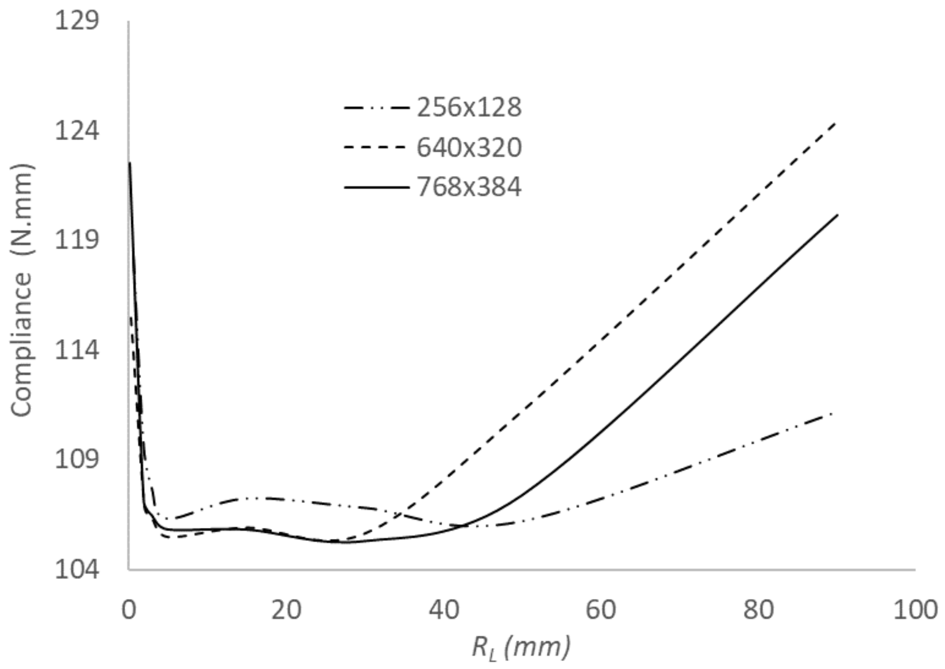


Figure 93 Change in compliance of the generated NFGL structures for the simply supported beam using the NFGL Framework as a function R_L

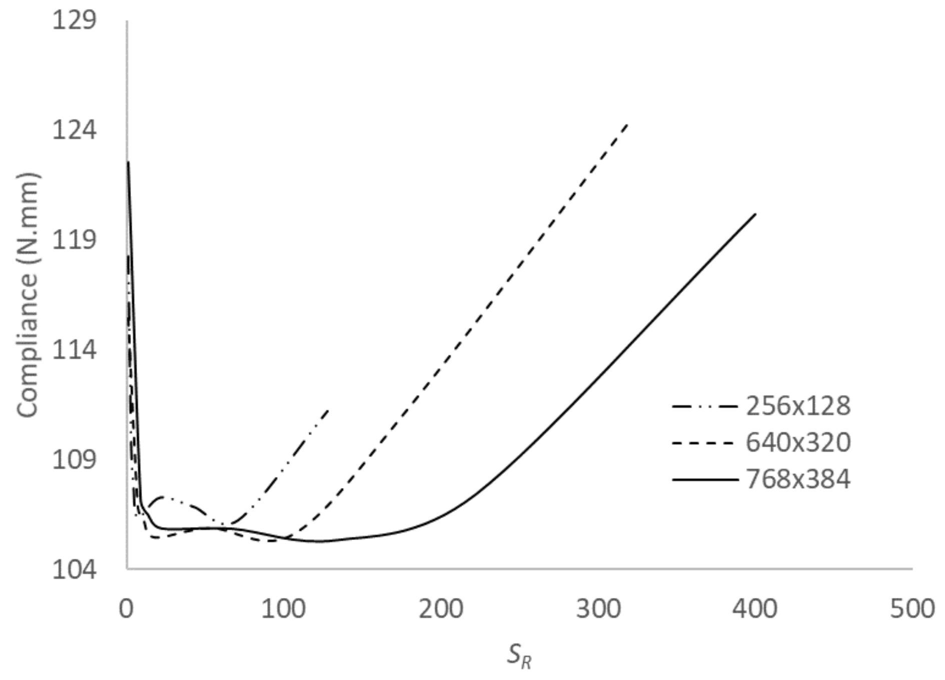


Figure 94 Change in compliance of the generated NFGL structures for the simply supported beam using the NFGL Framework as a function S_R

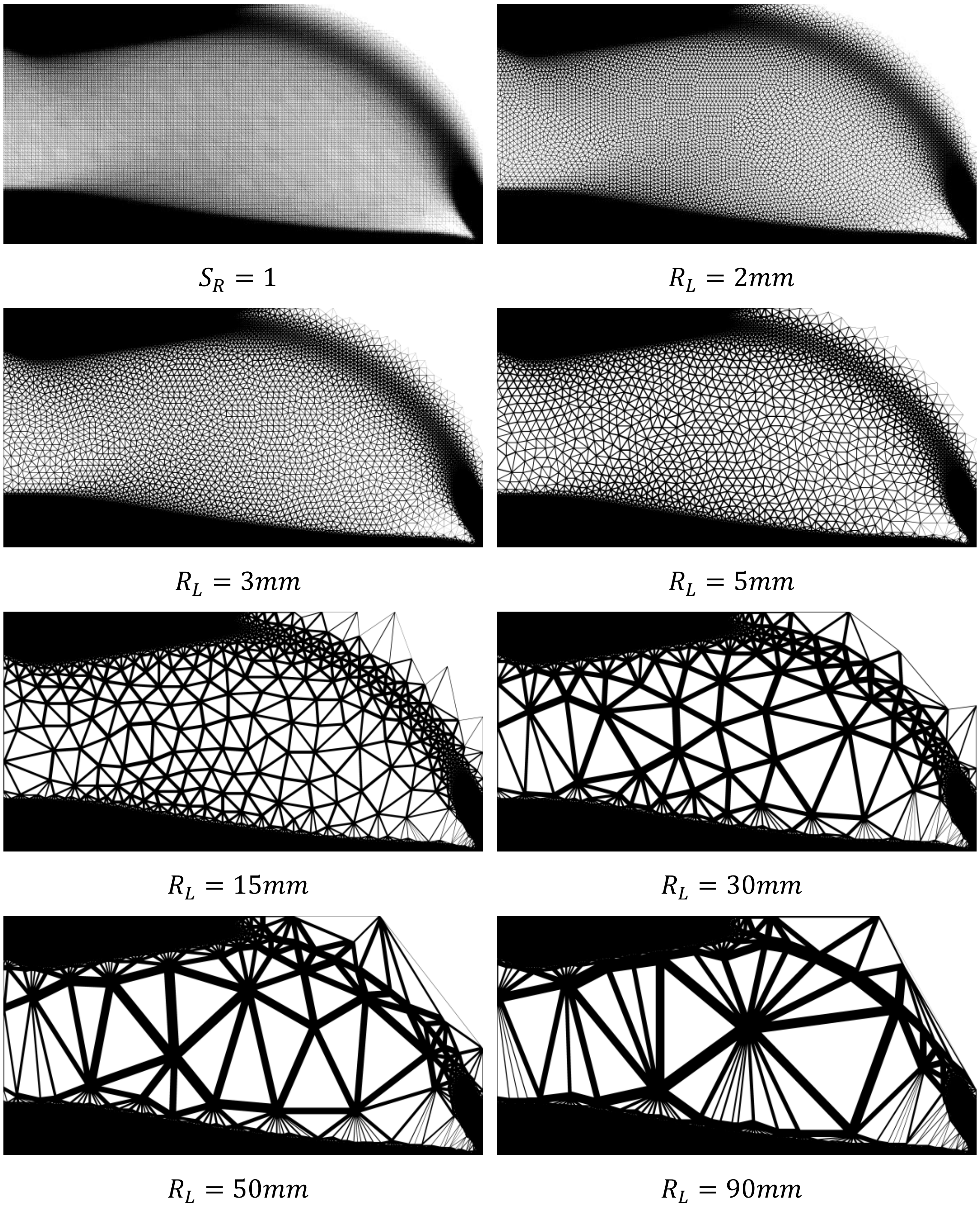


Figure 95 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 256×128 at R_L values

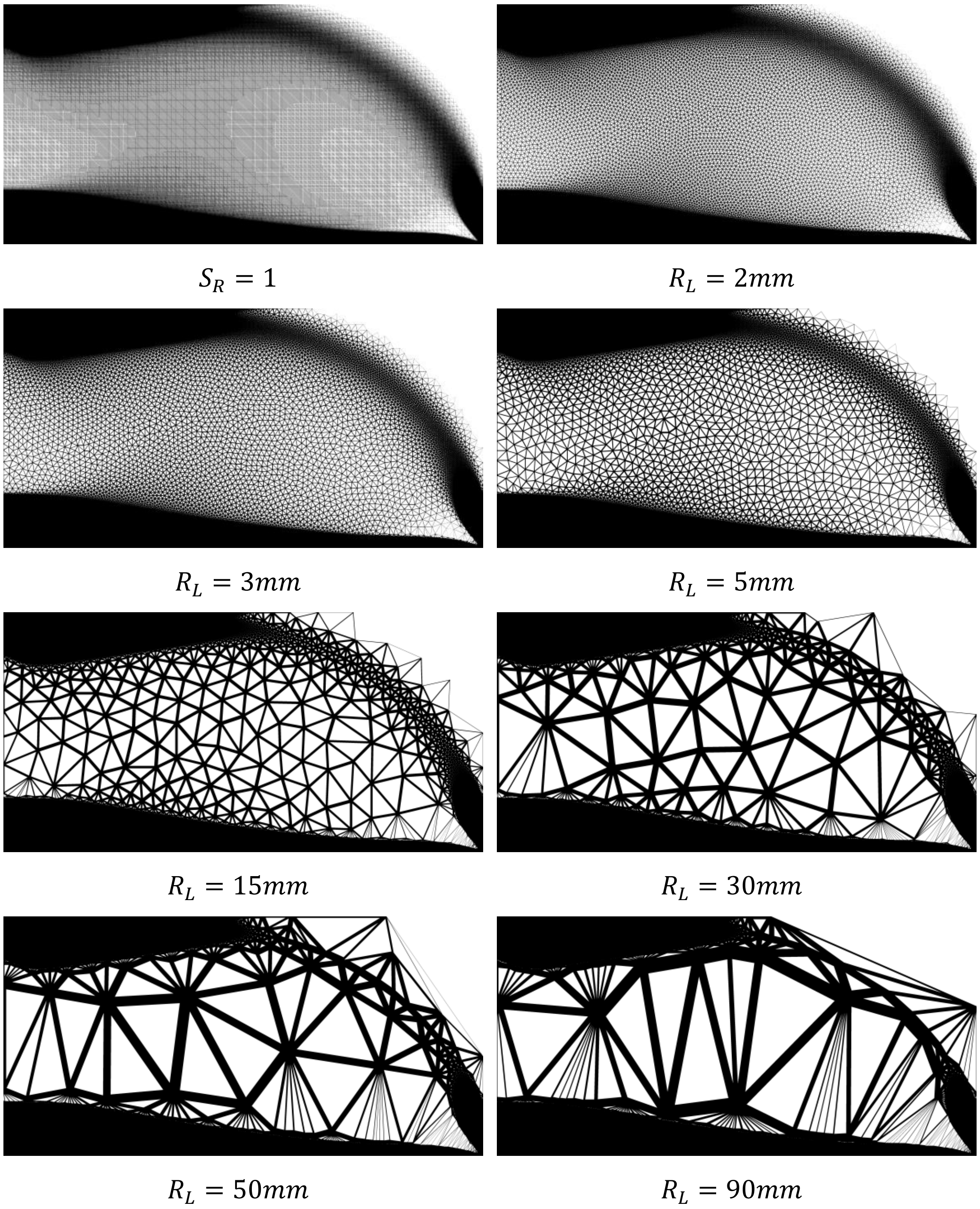


Figure 96 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 640×320 at R_L values

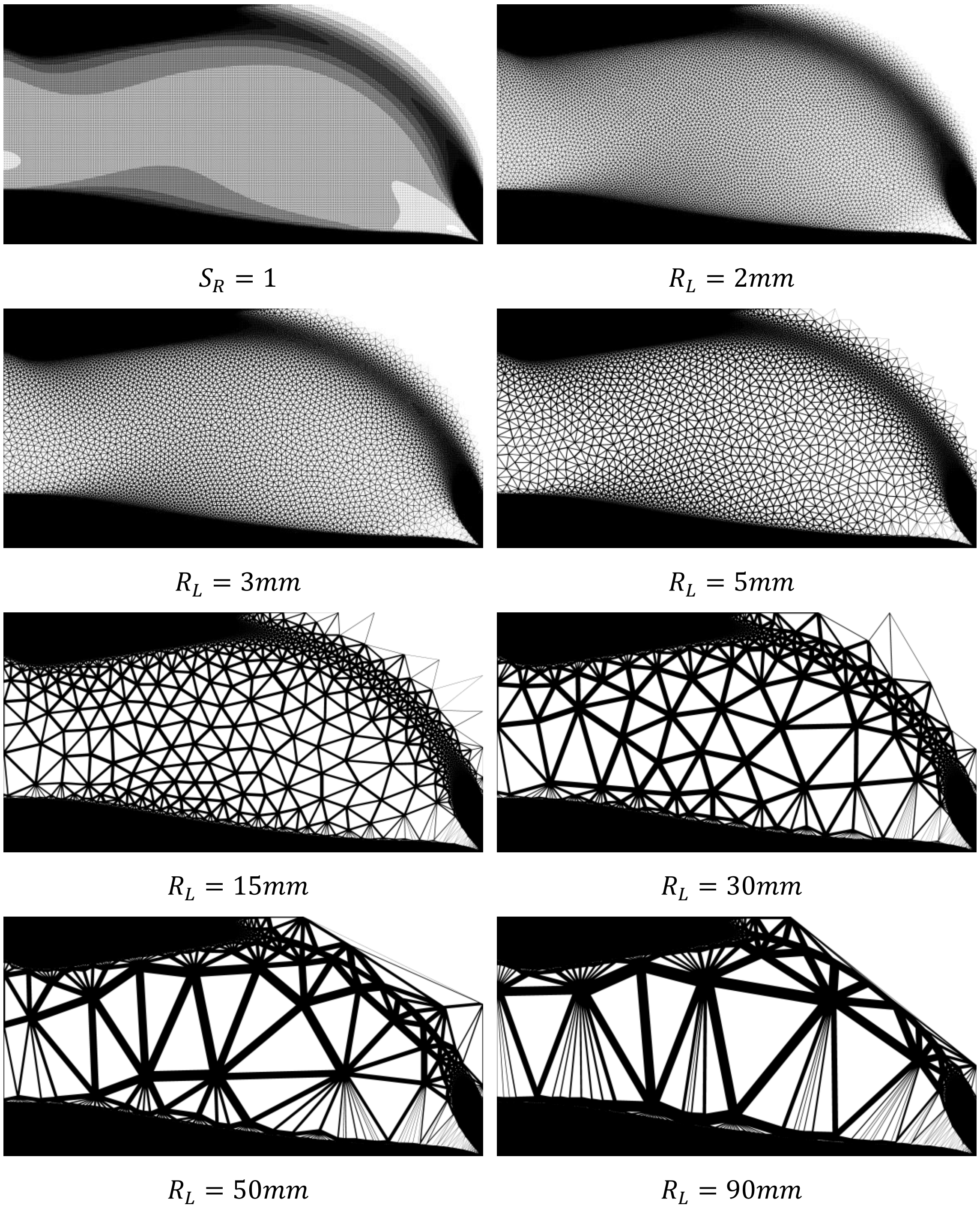


Figure 97 Generated NFGL structures using the NFGL Framework for the simply supported beam of size 768×384 at R_L values

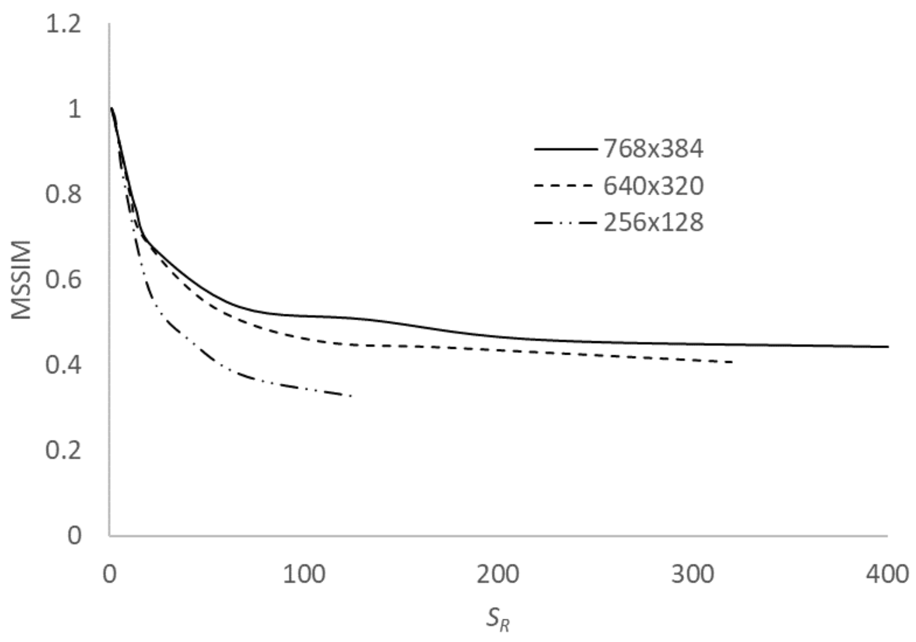


Figure 98 Change in the MSSIM index as a function of S_R for the simply supported beam

Table 14 Determining S_R for the simply supported beam at different mesh sizes using Algorithm 3 and Algorithm 4

No. elements	MSSIM	S_R
256×128	0.95	5
	0.23	456.2
	0.80	11.6
	0.75	13.4
640×320	0.95	5
	0.28	685.4
	0.79	13.7
	0.73	15.1
768×384	0.95	5
	0.29	984.2
	0.78	15.4
	0.75	16.6

4.1.3 Robustness Testing

Robustness is the ability of an algorithm to handle a wide range of data, and solve the problem as requested [88]. To test the robustness of the NFGL Framework, Stochastic Nodal Generation and Error Diffusion methods, different sinusoidal density fields input will be provided to the algorithms. The purpose of the test is to assess the ability of each algorithm to generate NFGL nodes based on the given inputs of varying density distributions. Eq. 53 describes the input density field function that will be used to generate the density distributions shown in Figure 99.

$$f(x,y) = \frac{cos\left(n\sqrt{(x-7.5)^2+(y-7.5)^2}\right)+1}{2} \tag{53}$$

where n is a factor to adjust the frequency of the function and is of the values 0.5, 1, 1.8, and 2 respectively.

The first method that will be used is the Stochastic Nodal Generation method. After applying the method, the drawbacks of the method becomes apparent. Figure 100 shows the generated NFGL nodes for n = 0.5, 1, 1.8 and 2 respectively. The generated NFGL nodes do not represent the density field distribution accurately as shown in figures. To visualize what’s causing the problem with the Stochastic Nodal Generation method, the steps of generating the subdivisions of the NFGL nodes were tracked as shown in Figure 101 for the case of $n = 2$. In step 8, the method does not further divide the cell because its center lies in a low-density region. So it is deemed sufficient to generate a node there based on the value of $l^2 \times \rho_f(S)$. This is also repeated in the remaining four corners as shown in step 20. Figure 102 also shows the boundaries of the cells of the upper right corner in step

20. Once all subdivisions are completed, the process is completed in step 67 and the generated NFGL nodes can be seen why they don't accurately represent the input density field. Therefore, the Stochastic Nodal Generation method is not robust enough to handle input density fields with rapidly changing densities.

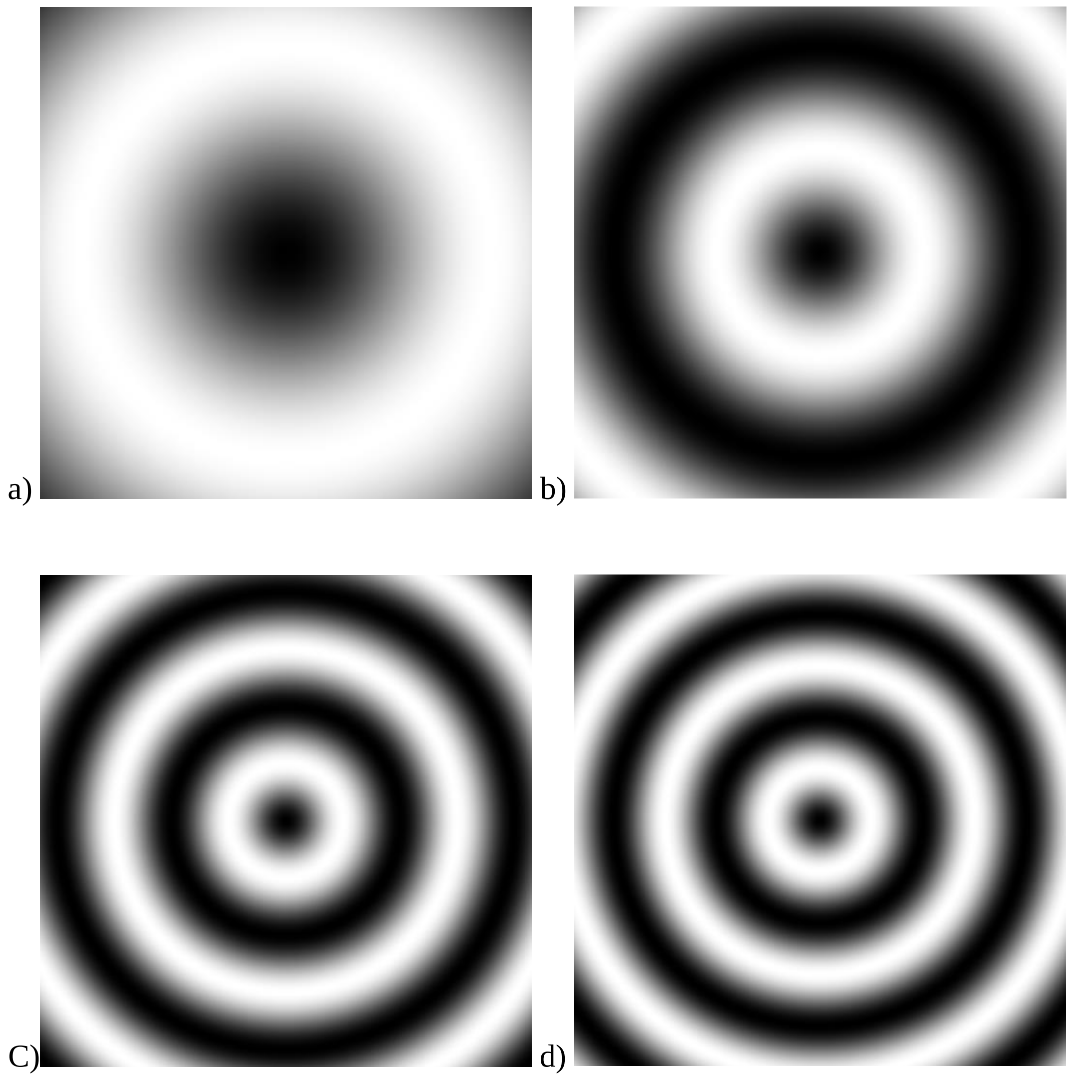
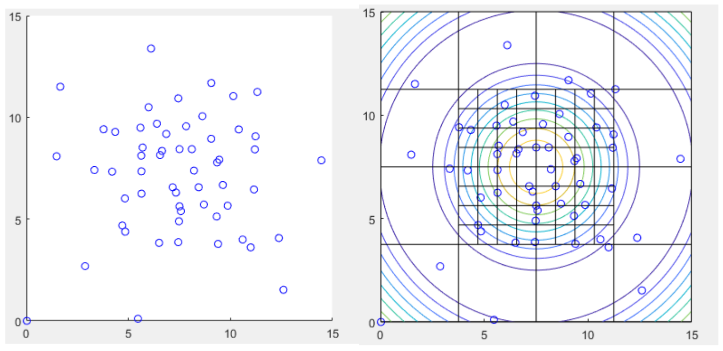
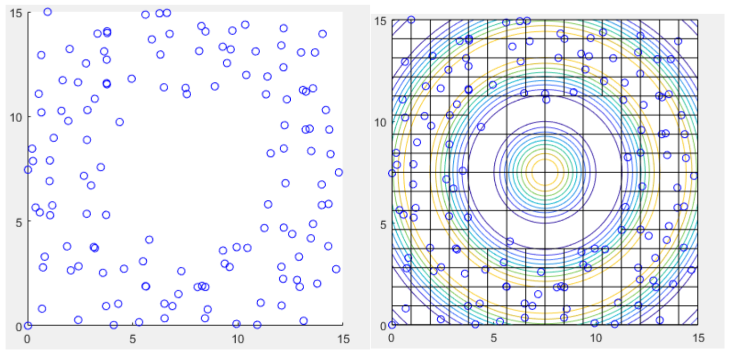
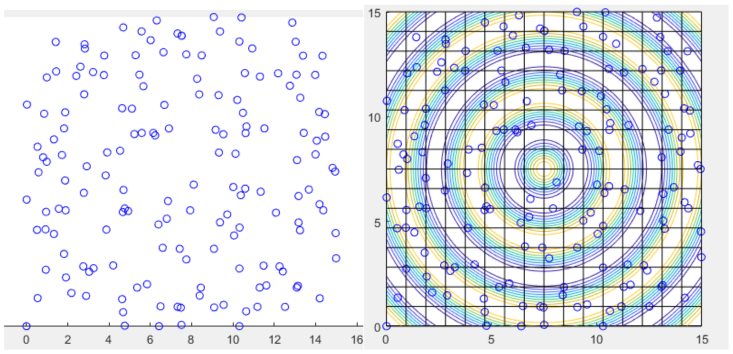


Figure 99 Sinusoidal input density field functions at different frequency values a) $n = 0.5$ b) $n = 1$ c) $n = 1.8$ d) $n = 2$


a)


b)


c)

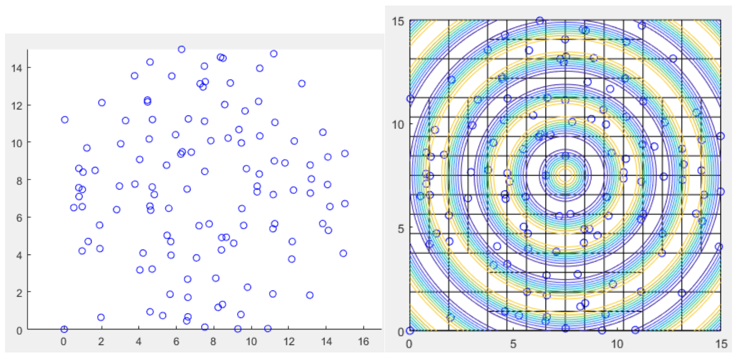

d)

Figure 100 Generated NFGL nodes using the Stochastic Nodal Generation method with a) $n = 0.5$ b) $n = 1$ c) $n = 1.8$ d) $n = 2$

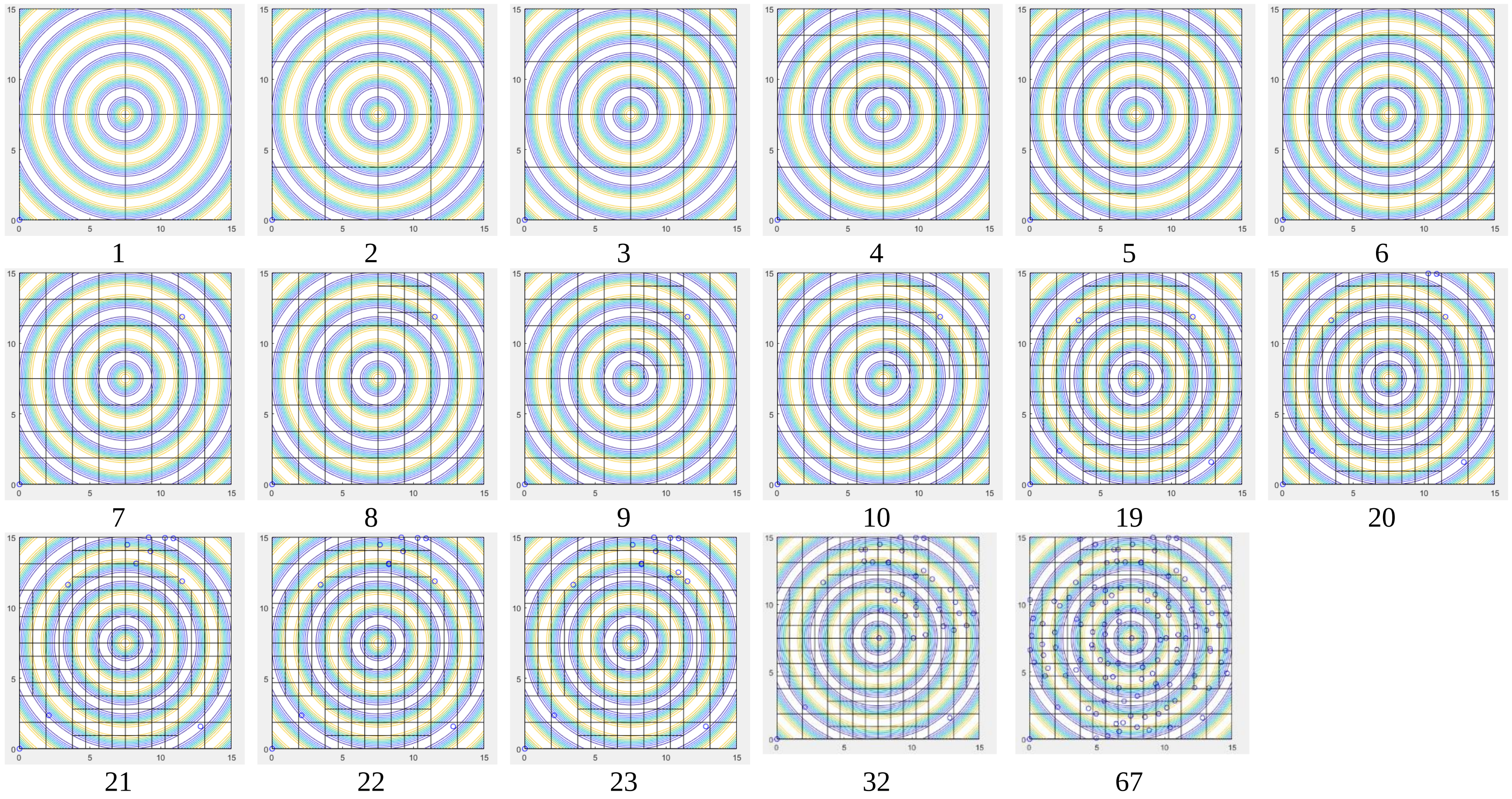


Figure 101 Generation of NFGL nodes at different steps for the Stochastic Nodal Generation using $n = 2$

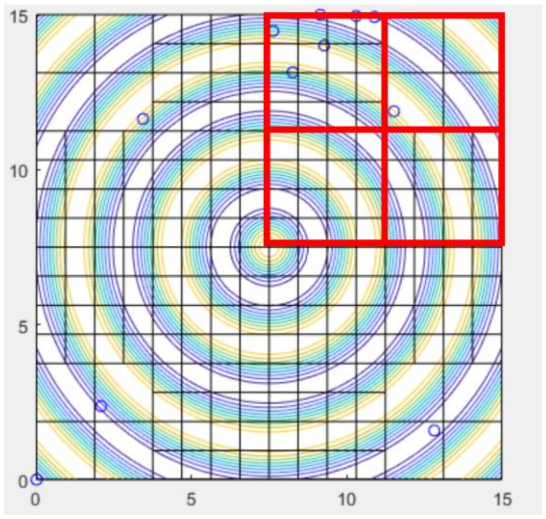


Figure 102 Cell boundaries for the upper right corner at step 20

The results of the Error Diffusion method using a grid size of 100×100 is shown in Figure 103. The figure shows the NFGL nodes and the corresponding NFGL structure generated from the nodes. Even though the NFGL nodes manage to capture a pattern similar to the input density fields, the generated NFGL structure from these nodes ends up with undesirable unit-cell shapes. These unit-cells would end up with long struts and very small cross-sectional sizes, which would create long and thin weak struts in the structure. This is due to the inability of the Error Diffusion method to freely control the unit-cell size ratio of the unit-cells. So in regions where the density is very low, the method doesn't generate any NFGL nodes at all because of how the error is being diffused in those regions. This would cause a jump in the unit-cell size ratio in these regions and the neighboring regions of higher density as seen in the figures. Furthermore, the same can happen when the boundaries have very low densities as in the case of n = 1. This test shows the drawbacks of the Error Diffusion method and the effects of its limitations on the unit-cells' size ratio.

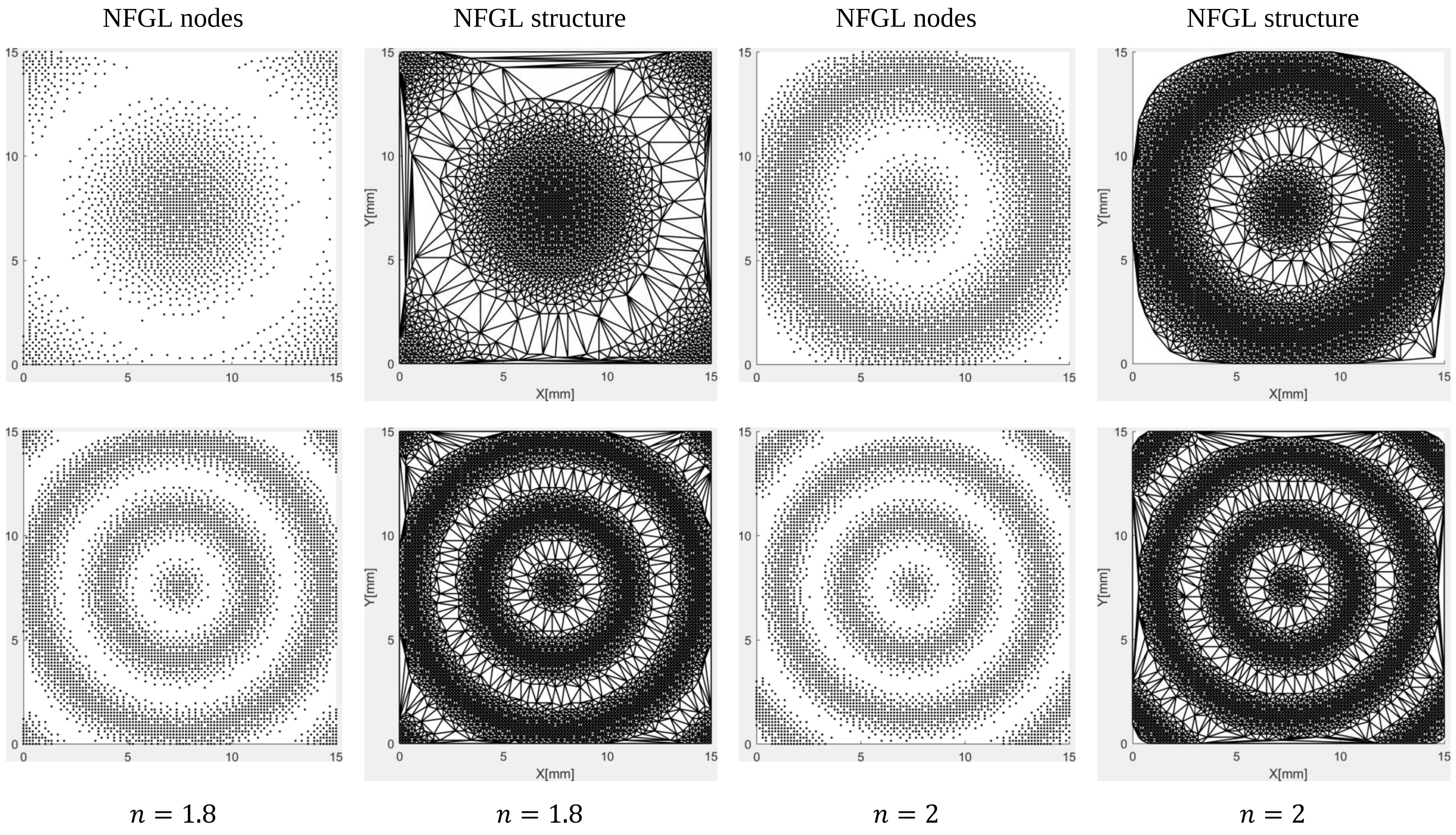


Figure 103 Generated NFGL nodes and structures using the Error Diffusion method for different values of n

The NFGL Framework was used to generate the NFGL structure at the different density inputs using S_R values of 2, 3, 4, and 5 using the same grid of 100×100 that as used in with the Error Diffusion method. The results for the different values of S_R for the different *n* values are shown in figures 104-107. Unlike the Error Diffusion results, the NFGL structures generated using the NFGL Framework do not exhibit the same drawback of having badly shaped unit-cells. Furthermore, the NFGL Framework is able to generate different unit-cell size ratios easily for different *n* values in a computational time similar or even better than the Error Diffusion method as shown in Table 15. This is because the NFGL Framework does not rely on having multiple nodes determine whether a node should exist or not in the grid but on the node itself controlling the existence of adjacent nodes around it, unlike the Error Diffusion method. Thus, further supporting hypothesis 2 on the ability of the Framework to create NFGL structures without restrictions on the unit-cell size ratio in a computationally efficient manner. Furthermore, the NFGL Framework showed to be more robust than the other methods when dealing with rapidly changing density fields. Thus showing the advantages of the NFGL Framework in generating NFGL structures compared to the other methods.

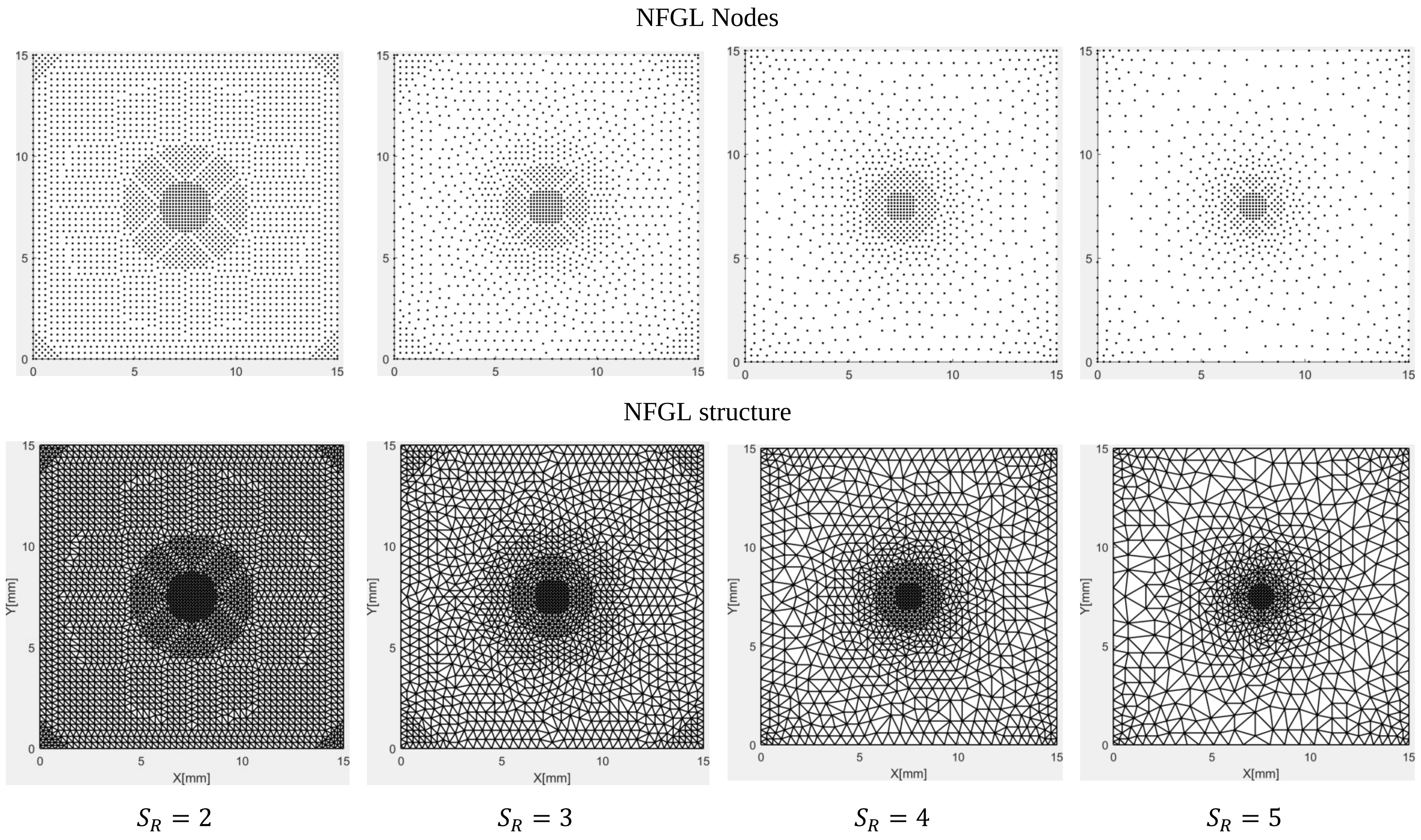


Figure 104 Generated NFGL nodes and structures using the NFGL Framework for $n = 0.5$ at different S_R values

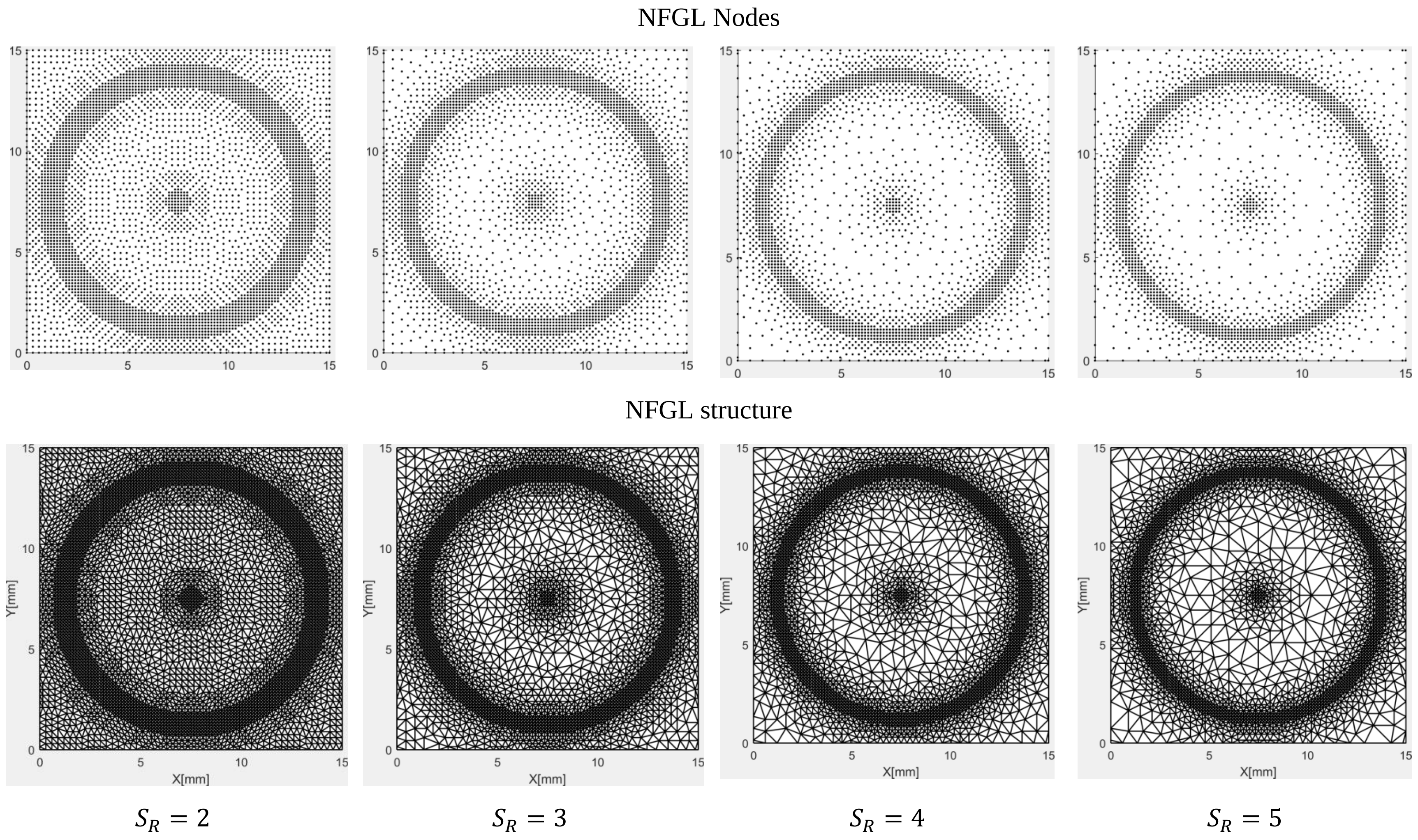


Figure 105 Generated NFGL nodes and structures using the NFGL Framework for $n = 1$ at different S_R values

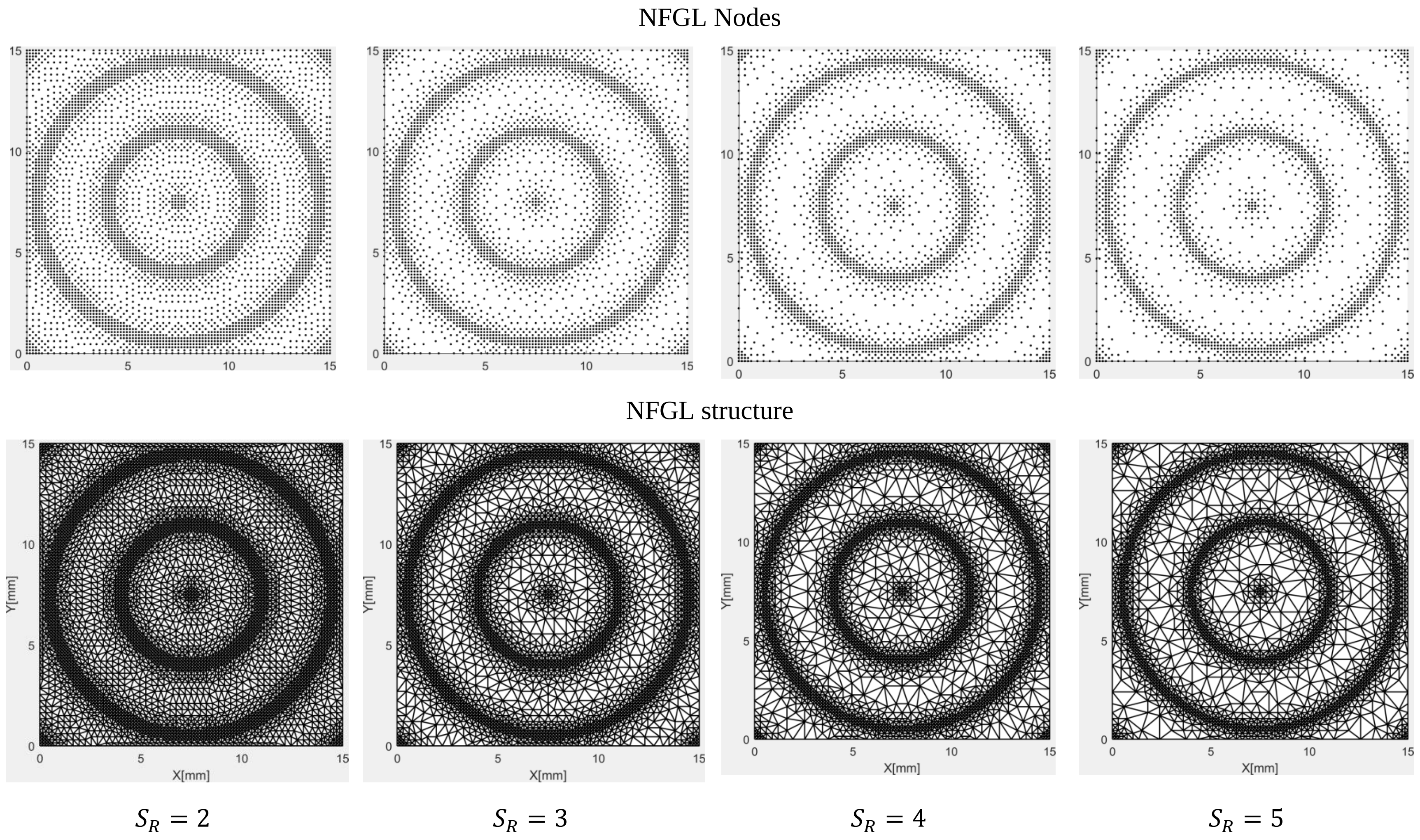
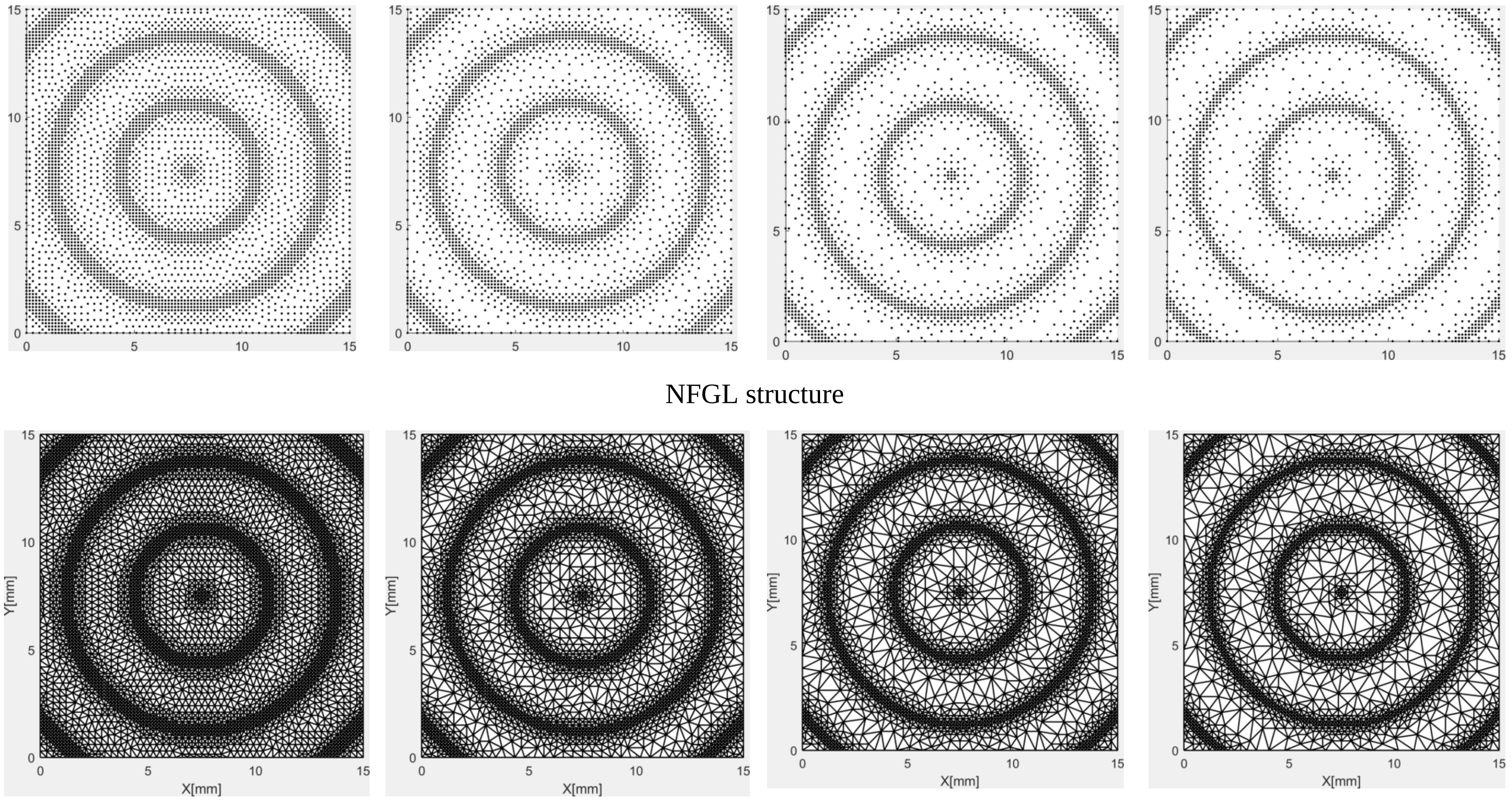


Figure 106 Generated NFGL nodes and structures using the NFGL Framework for $n = 1.8$ at different S_R values

NFGL Nodes



$S_R = 2$ $S_R = 3$ $S_R = 4$ $S_R = 5$

Figure 107 Generated NFGL nodes and structures using the NFGL Framework for $n = 2$ at different S_R values

Table 15 Computational cost for the generation of NFGL structures at different n values

n	NFGL Framework S_R				Error Diffusion
	2	3	4	5	
0.5	0.465	0.304	0.227	0.171	0.392
1	0.689	0.561	0.463	0.416	0.697
1.8	0.701	0.645	0.543	0.494	0.700
2	0.693	0.501	0.512	0.484	0.665

4.1.4 Conformity to Design Domain

In this section, the NFGL framework will be compared with the Stochastic Nodal Generation and Error Diffusion methods in generating a circular NFGL structure that conforms to the circular design domain and an NFGL structure that follows a curved path.

4.1.4.1 Circular Design Domain

Figure 108 shows the density input of the circular design domain that will be used to generate the NFGL structure and Eq. 54 describes how the density varies across the design domain. The center of the circle is at (15, 15), with a radius of 15, and has a maximum density of 1 at the edges of the circle and 0.091 at the center.

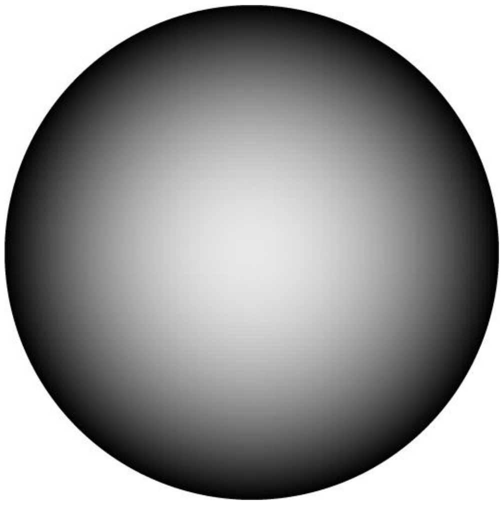


Figure 108 Density gradient of the circular design domain

$$f(x,y) = \begin{cases} 0 & if\ \sqrt{\left(\frac{x}{15}-1\right)^2+\left(\frac{y}{15}-1\right)^2} > 1 \\ \dfrac{\left(\frac{x}{15}-1\right)^2+\left(\frac{y}{15}-1\right)^2+0.1}{1.1} & otherwise \end{cases} \tag{54}$$

To apply the Error Diffusion method, an image that describes the density field was generated with a size of 80×80. Thus, the number of pixels is 6,400 with 4,872 of them inside of the circular design domain. Figure 109 shows the generated image of the density field and the generated NFGL nodes using the Error Diffusion method. Since the density field has to be represented in an 80×80 image, the edges of the circular domain become jagged and also form large straight lines. Figure 110 shows the NFGL structure generated from the Error Diffusion NFGL nodes. From the figure, it is clear that the edges of the NFGL structure form large straight lines around the design domain. Furthermore, this is more noticeable on the top, bottom, and sides of the structure, where the straight edges were the longest. Moreover, since the NFGL nodes were the pixels of the 80×80 image, they were not conforming to the design domain, thus the generated structure did not conform to the design domain as well, which confirms hypothesis 1 that the conformity of the nodes affects the conformity of the generated NFGL structure. Figure 111 shows a close up of the NFGL structure, showing how the edges form straight lines and also how the unit-cells of the NFGL structure do not conform to the circular design domain. This highlights the drawback of using the Error Diffusion method when used on non-rectangular design domains.

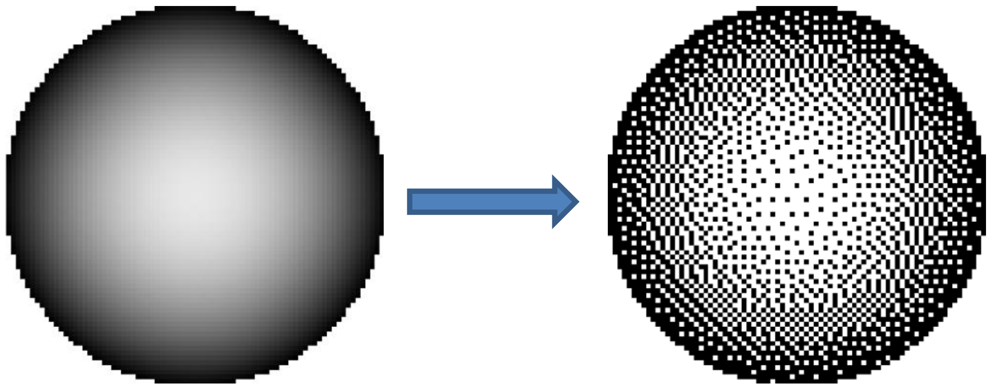


Figure 109 Transformation of the input density image into NFGL nodes using the Error Diffusion method for the circular design domain

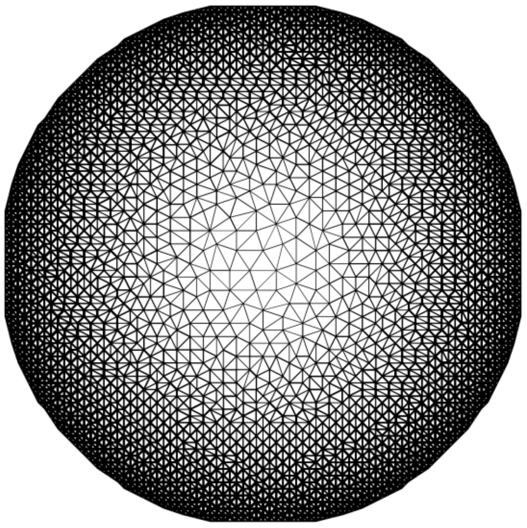


Figure 110 Generated NFGL structure using the Error Diffusion method for the circular design domain

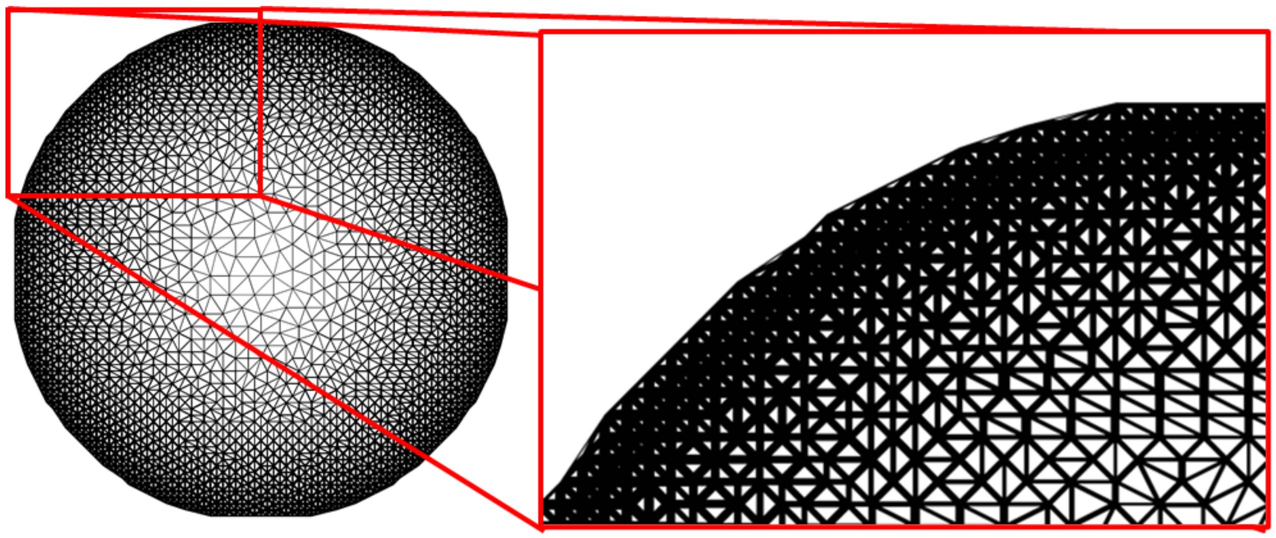


Figure 111 Close up of the generated NFGL unit-cells using the Error Diffusion method for the circular design domain

As for the Stochastic Nodal Generation method, the generated NFGL nodes are shown in Figure 112. From the figure, it is clear that, due to the stochastic nature of the method, the nodes will not be conformal to the design domain. Furthermore, there are nodes that are laying outside of the design domain region. Figure 113 shows the generated NFGL structure from the nodes in Figure 112. The structure is not conformal to the design domain as expected and the generated unit-cells look undesirable. Thus, the Stochastic Nodal Generation method is unfavorable when generating NFGL structures that require to be conformal to the design domain.

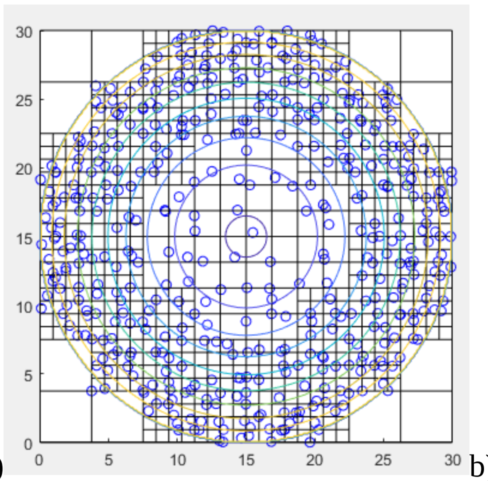
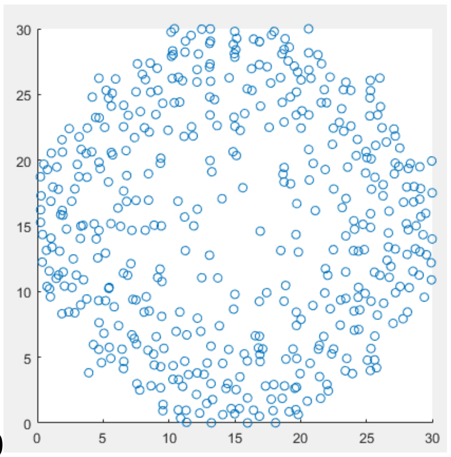
a)  b) 

Figure 112 Generated NFGL nodes using Stochastic Nodal Generation for the circular design domain a) Generated cells b) Generated NFGL nodes

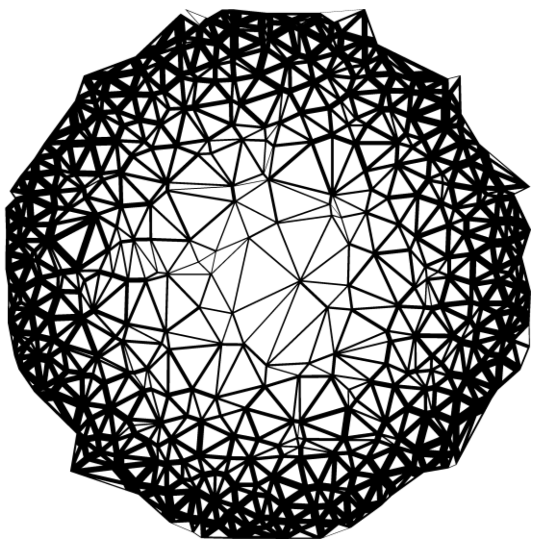


Figure 113 Generated NFGL structure using the Stochastic Nodal Generation method for the circular design domain

To apply the NFGL framework, a conformal mesh was generated. Figure 114-a shows the generated mesh, which took only 0.42 seconds and has 4824 nodes to be as close as possible to the number of nodes used in the Error Diffusion method. The generated NFGL structure is shown in Figure 114-b. The NFGL structure in the figure was generated using an S_R value of 5. The generated NFGL structure does not suffer from the drawbacks present in the Error Diffusion and Stochastic Nodal Generation methods. A closeup of the generated unit-cells is shown in Figure 115. From the figure, it is clear that the uni-cells of the structure are conformal to the design domain. Furthermore, Figure 116 shows different NFGL structures that were generated using S_R values of 2, 3, 5 and 10. The structures were generated without the need to remesh the design domain or adjusting the density field input.

Table 16 shows the computational cost for each of the methods when generating the NFGL structure. The Stochastic Nodal Generation had the lowest time since it had only 464 nodes, which the least number of nodes used to generate the structure among all methods. But the generated structure is not conformal and the uni-cells' shape is highly unfavorable. As for the NFGL Framework, the computational cost in the table does not include the 0.42 seconds needed to generate the initial mesh, since it is not required to regenerate the mesh for different S_R values. The results show that the NFGL framework can generate different NFGL structures with different size ratios in a computationally efficient manner when compared with the Error Diffusion method, which can only generate one NFGL structure with the given size of 80×80 and without altering the density field input. Thus, further confirming hypothesis 2 in showing that the NFGL Framework can generate NFGL structures with different unit-cell sizes ratios in a computationally efficient manner.

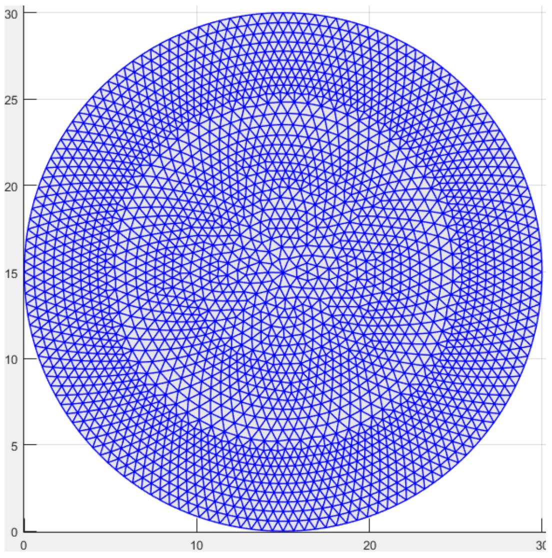
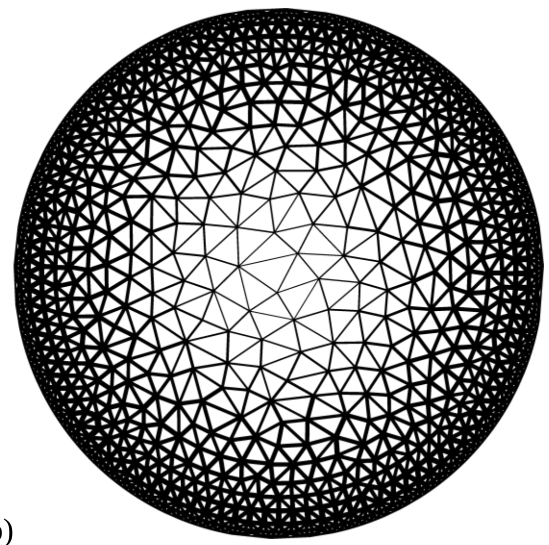
a)  b) 

Figure 114 Circular Design domain FEM mesh nodes

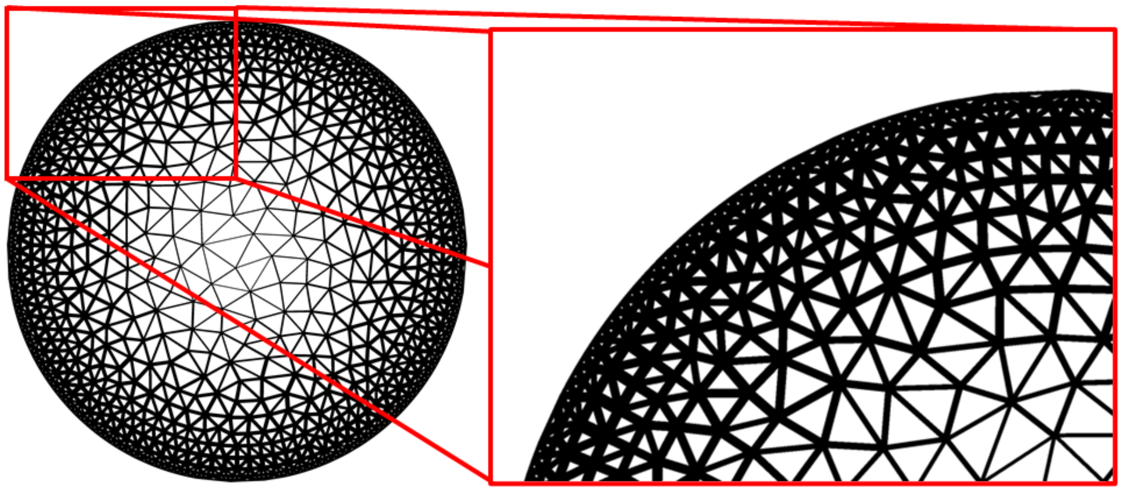


Figure 115 Close up of the generated NFGL unit-cells using the NFGL Framework for the circular design domain with $S_R = 5$

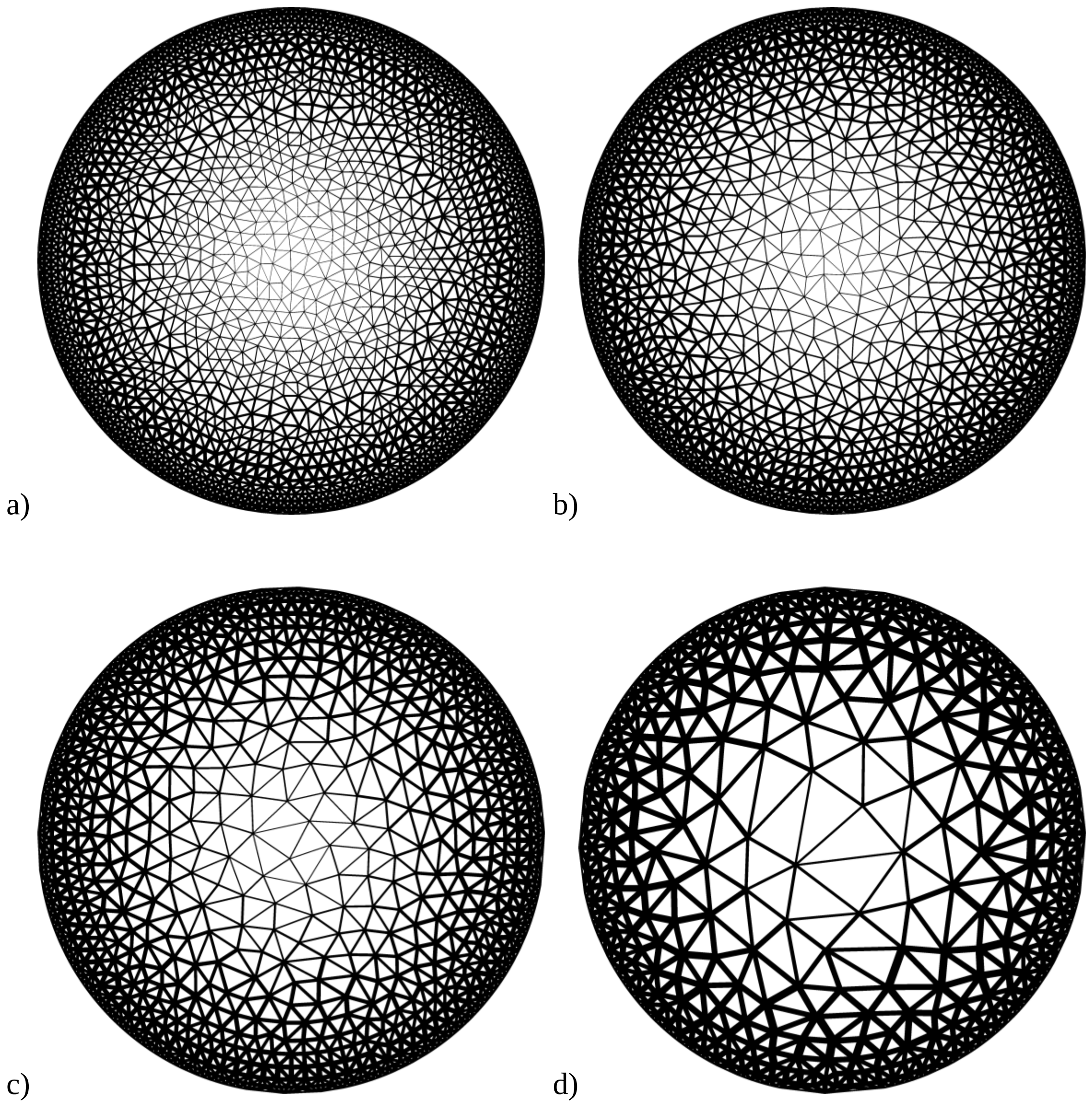


Figure 116 Generated NFGL structures using the NFGL Framework for the circular design domain a) $S_R = 2$ b) $S_R = 3$ c) $S_R = 5$ d) $S_R = 10$

Table 16 Computational cost of the NFGL Framework, Error Diffusion, and Stochastic Nodal Generation methods for the circular design domain

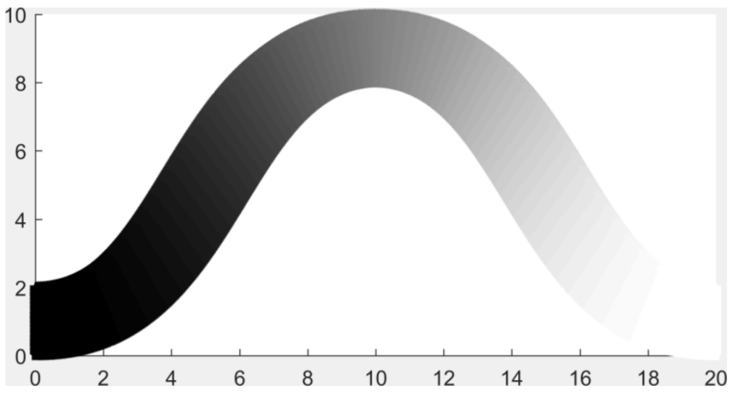
	NFGL Framework				Error Diffusion	Stochastic Nodal Generation
	S_R					
	2	3	5	10		
Time (s)	0.3795	0.2415	0.1920	0.1241	0.3918	0.0965
No. Nodes	4824				4872	464

4.1.4.2 Curved Path Design Domain

Two curved design domains will be used to further evaluate the performance of the methods in section 4.1.4.1 on the generation of conformal NFGL structures. The curves were generated using Bezier curves with the following density distributions along the curve

$$f(u) = \frac{\cos(2\pi nu) + 1}{2} \tag{55}$$

where u is the parameter of the Bezier curve, n is the frequency of the values 0.25 and 4 as shown in Figure 117

a)


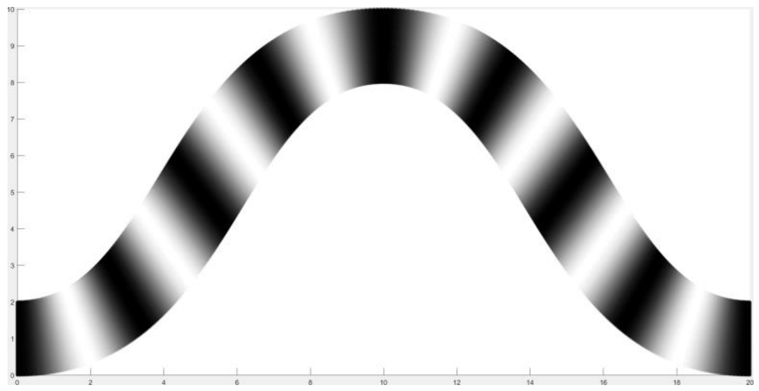
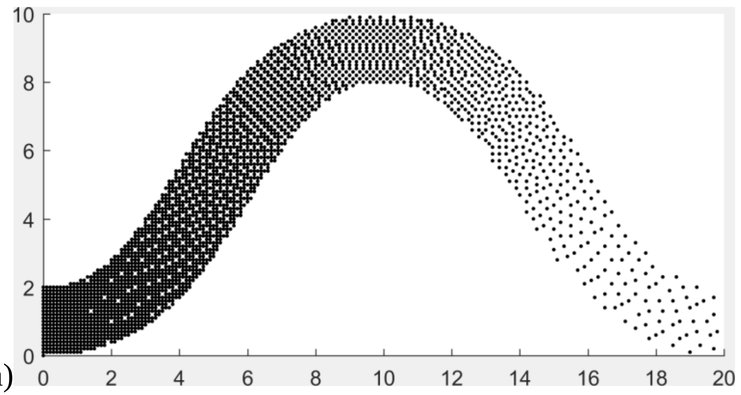
b)


Figure 117 Curved path design domain a) $n = 0.25$ b) $n = 4$

The Error Diffusion method was used on the design domains to generate the NFGL nodes and corresponding NFGL structure. Figure 118 shows the generated NFGL nodes and NFGL structure for $n = 0.25$ while Figure 119shows the same but for $n = 4$. From the figures, it is also apparent that the Error Diffusion method is not able to generate conformal NFGL structures. This becomes more apparent in regions where the density is low. But even in higher density regions, the NFGL unit-cells still do not conform to the design domain. Furthermore, when $n = 0.25$ the end of the design domain have no nodes generated to form unit-cells. So it ends up being truncated. This further shows the drawback of using the Error Diffusion method when creating conformal NFGL structures.

a) 

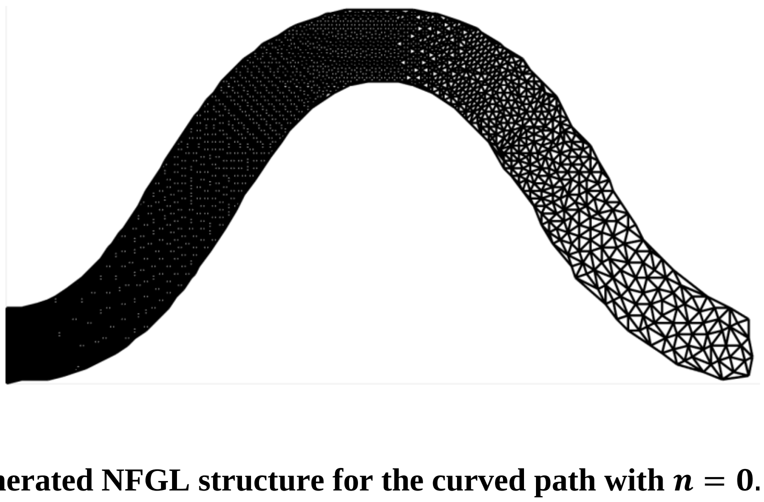
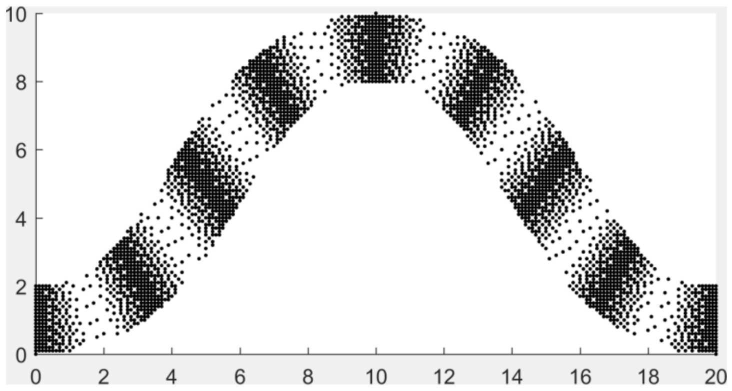
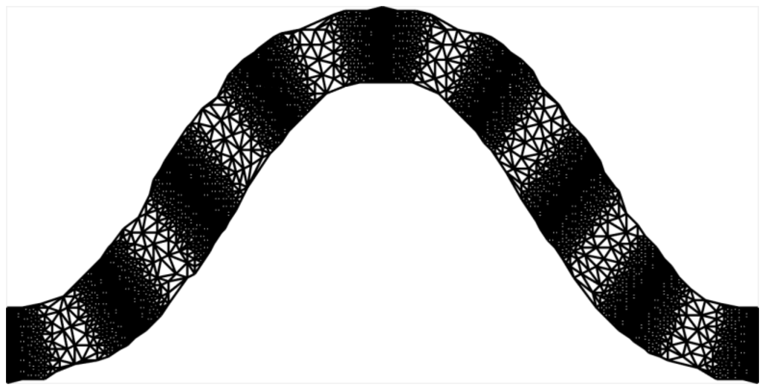
b) 

Figure 118 Generated NFGL structure for the curved path with $n = 0.25$ using the Error Diffusion method a) Generated NFGL nodes b) Corresponding NFGL structure



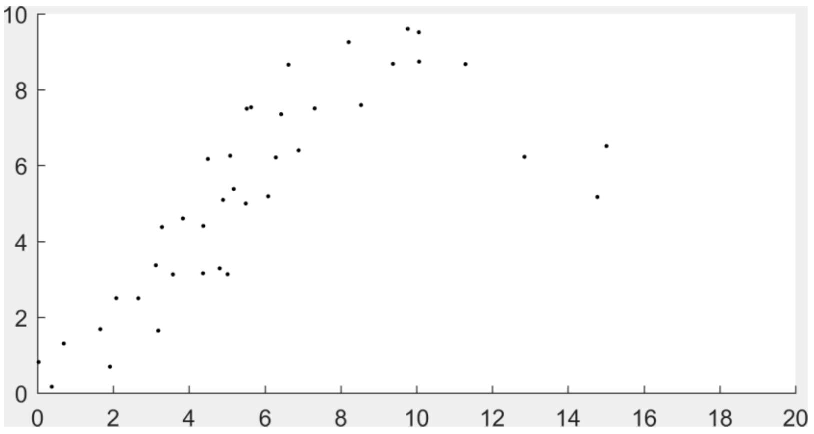
a)



b)

Figure 119 Generated NFGL structure for the curved path with $n = 4$ using the Error Diffusion method a) Generated NFGL nodes b) Corresponding NFGL structure

As for the Stochastic Nodal Generation method, it suffered from the same issue explained in section 4.1.3 and was not able to generate NFGL nodes that can represent the design domain as shown in Figure 120. Thus, further showing the drawbacks of this method, which makes it highly unfavorable to use.

a) 

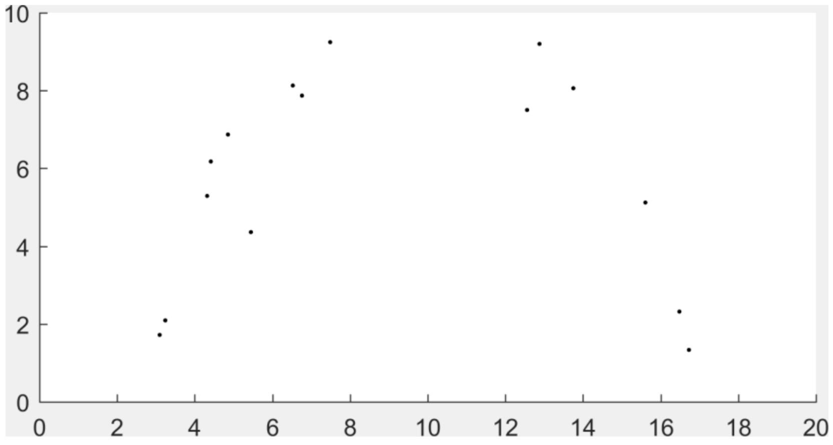
b) 

Figure 120 Generated NFGL nodes for the curved path for the Stochastic Nodal Generation method with a) $n = 0.25$ b) $n = 4$

As for the NFGL Framework, both design domains were generated using different S_R values ($S_R = 2$, 3, and 4). Figure 121 shows the FEM mesh that was used to generate the NFGL nodes and structures. The meshing process took 0.32 seconds and generated 12492 elements and 6537 nodes. The generated NFGL nodes and structures are shown in figures 122-127. The figures show how well the NFGL Framework can produce the NFGL nodes that are conformal to both curved design domains and how the corresponding NFGL structures also conform to the design domains. Furthermore, the unit-cell size ratio was easily adjustable without the need to alter the input density field distributions.

Table 17 shows the computational time for all three methods for both design domains. The NFGL Framework had lower computational cost compared to the Error Diffusion method for both cases.

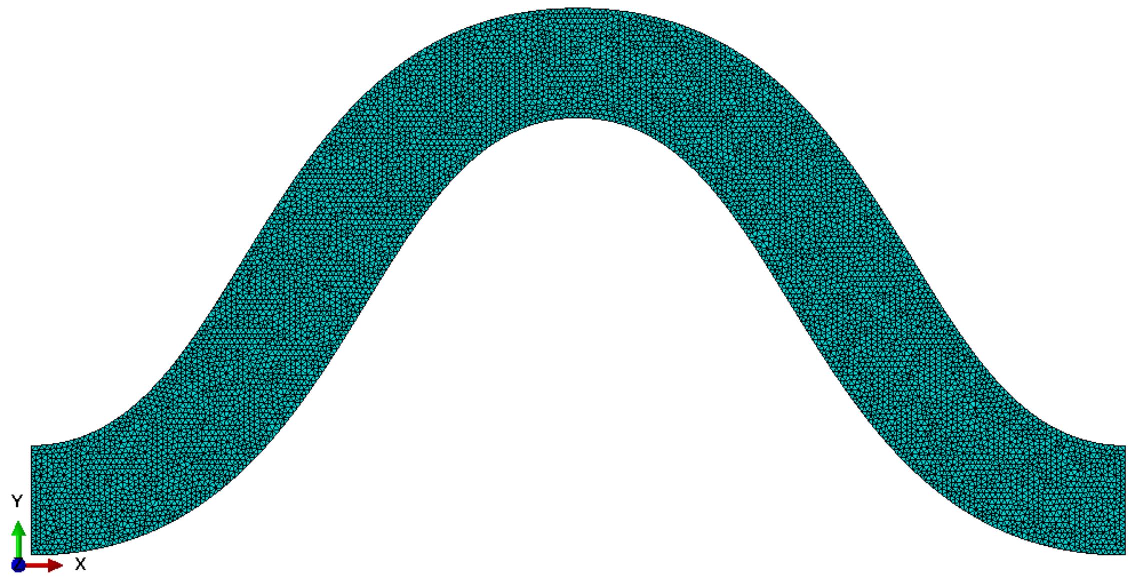
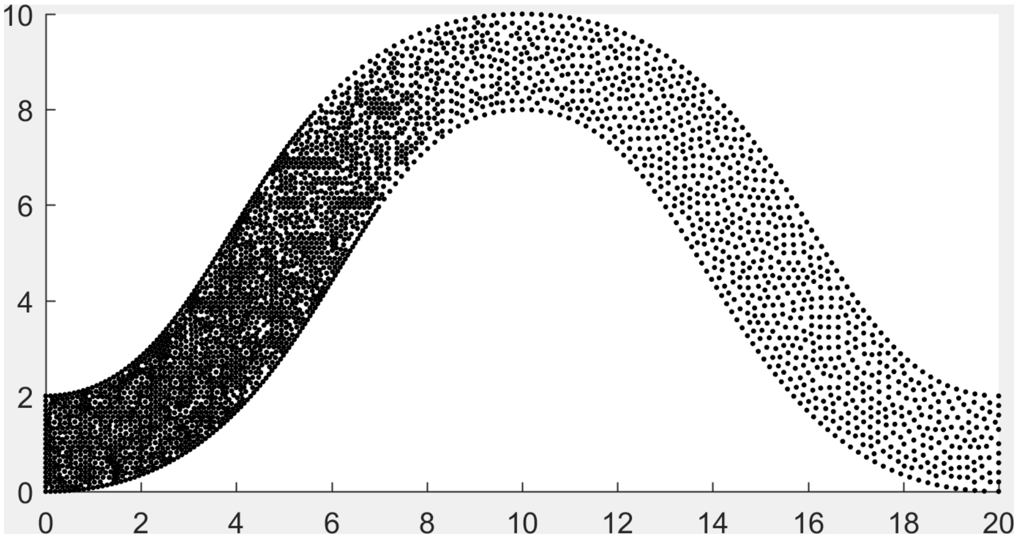
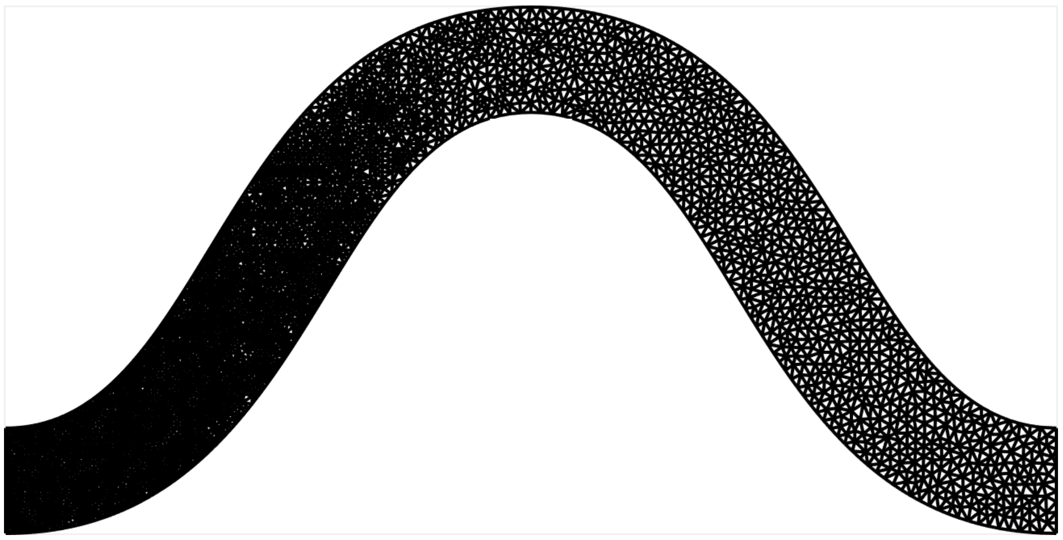


Figure 121 Input FEM mesh for the design domains of the curved paths

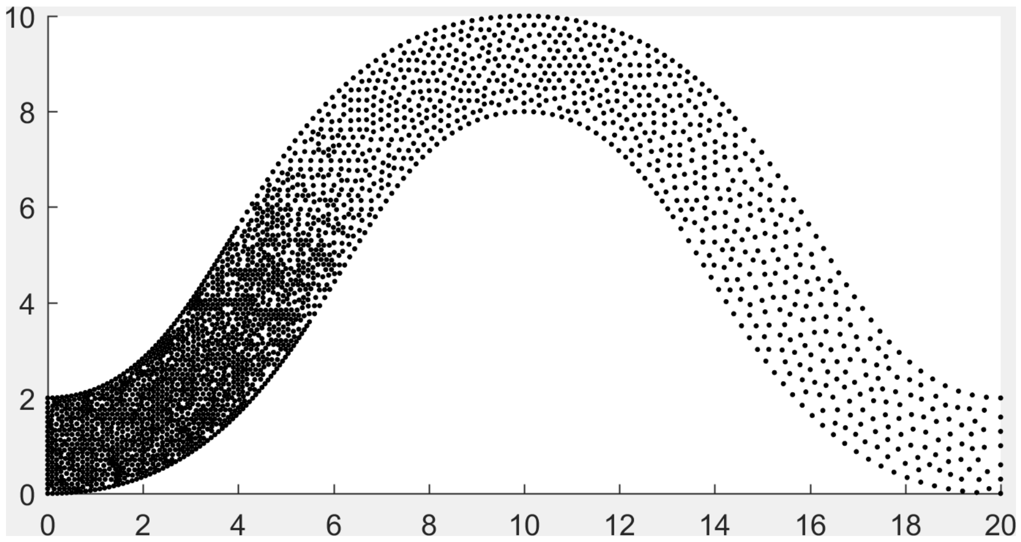


a)



b)

Figure 122 Generated NFGL structure for the curved path with $n = 0.25$ using the NFGL Framework method with $S_R = 2$ a) Generated NFGL nodes b) Corresponding NFGL structure

a) 

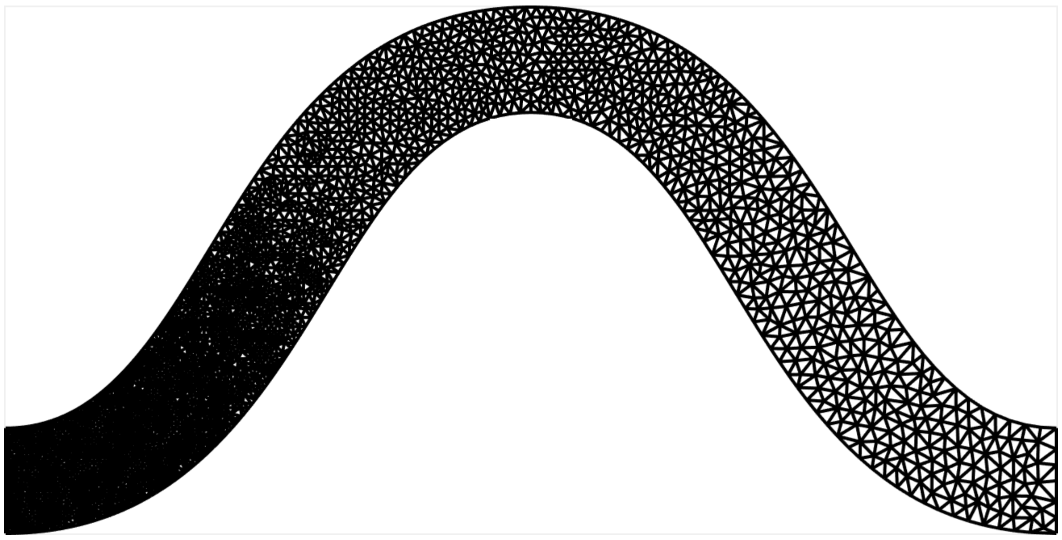
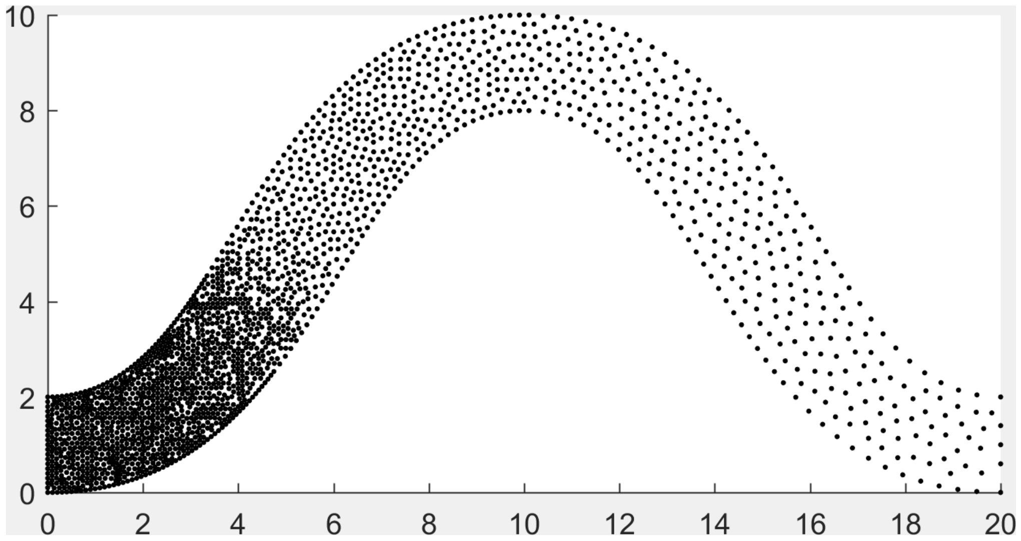
b) 

Figure 123 Generated NFGL structure for the curved path with $n = 0.25$ using the NFGL Framework method with $S_R = 3$ a) Generated NFGL nodes b) Corresponding NFGL structure

a)


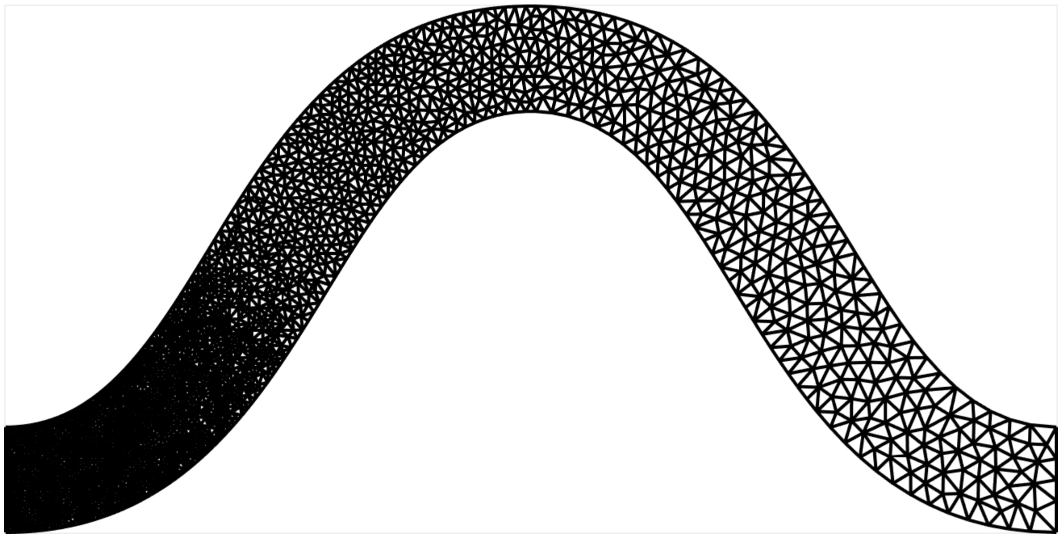
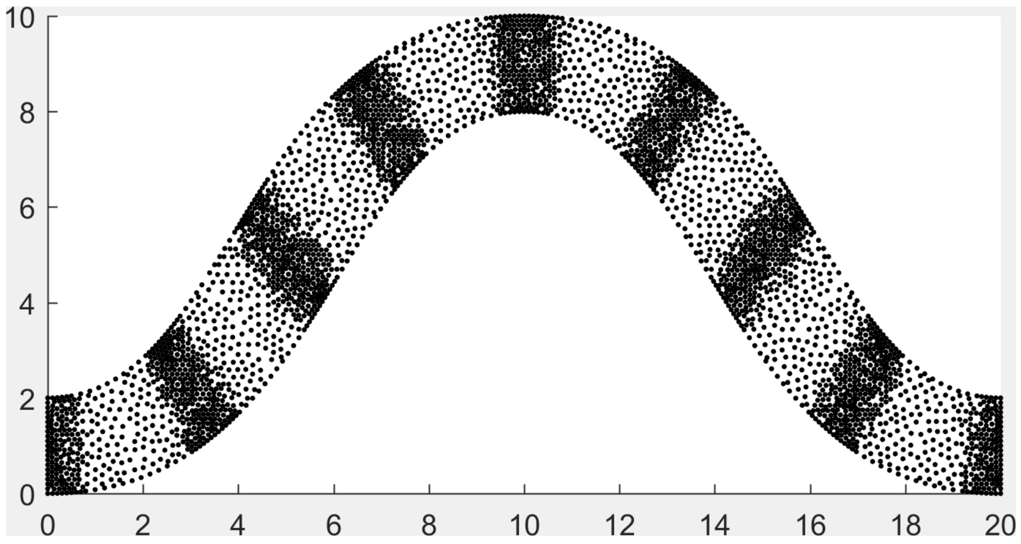
b)


Figure 124 Generated NFGL structure for the curved path with $n = 0.25$ using the NFGL Framework method with $S_R = 4$ a) Generated NFGL nodes b) Corresponding NFGL structure

a) 

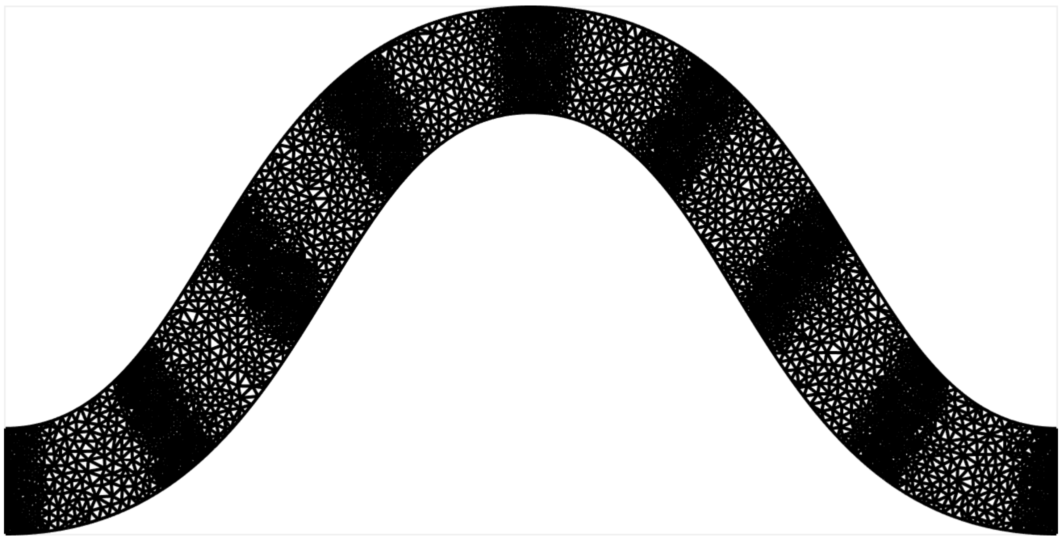
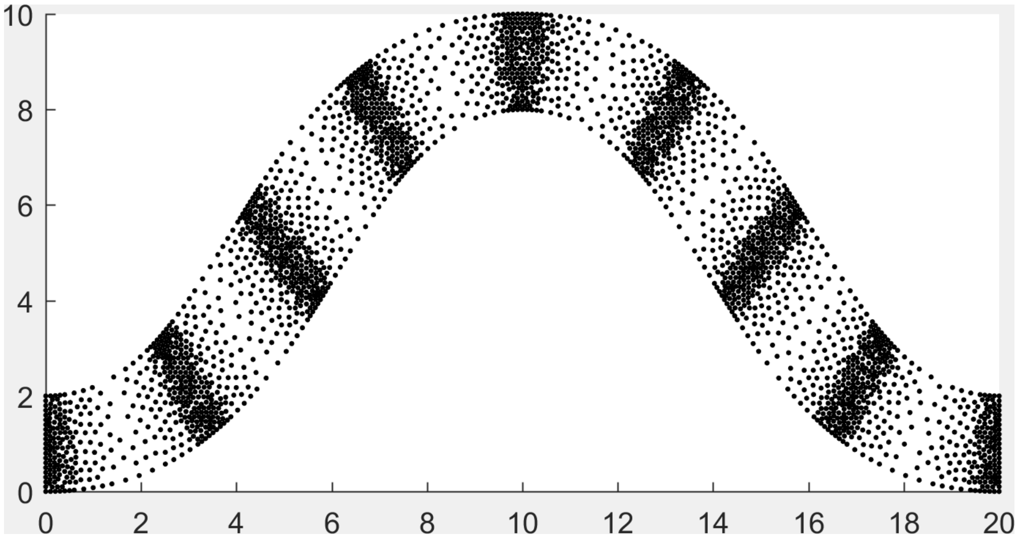
b) 

Figure 125 Generated NFGL structure for the curved path with $n = 4$ using the NFGL Framework method with $S_R = 2$ a) Generated NFGL nodes b) Corresponding NFGL structure

a)



b)

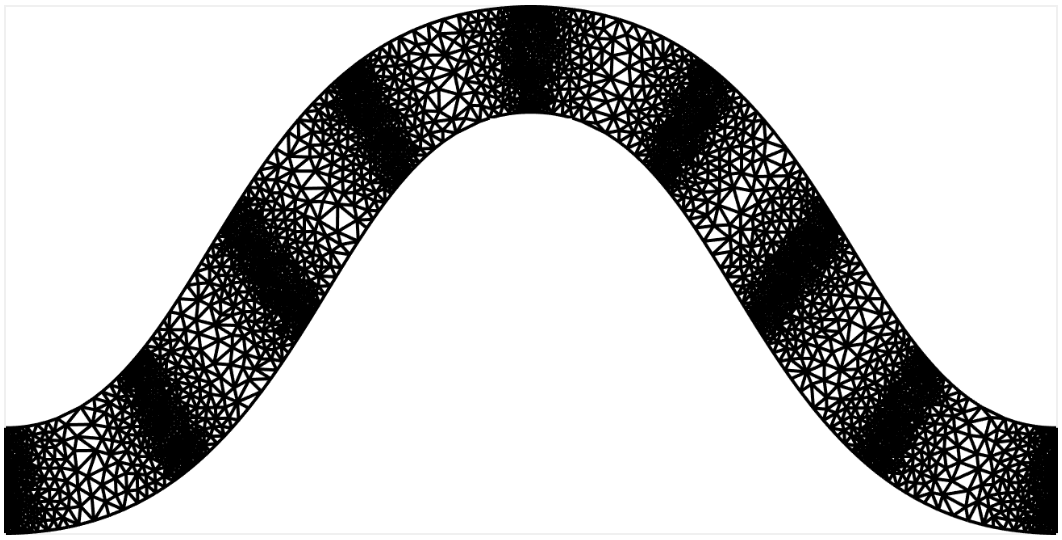
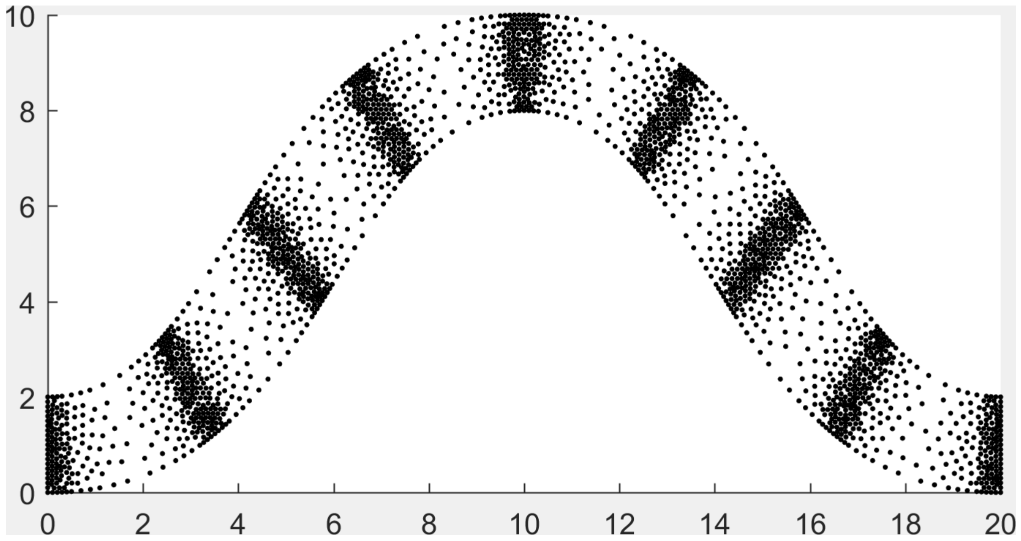


Figure 126 Generated NFGL structure for the curved path with $n = 4$ using the NFGL Framework method with $S_R = 3$ a) Generated NFGL nodes b) Corresponding NFGL structure

a)


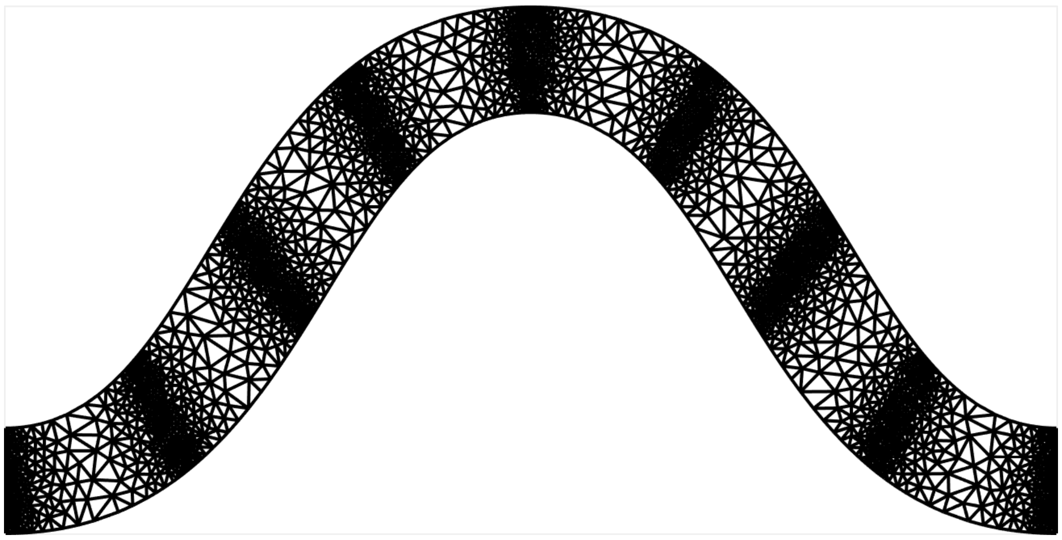
b)


Figure 127 Generated NFGL structure for the curved path with $n = 4$ using the NFGL Framework method with $S_R = 4$ a) Generated NFGL nodes b) Corresponding NFGL structure

Table 17 Computational cost for generating the NFGL structure for both curved paths using NFGL Framework, Error Diffusion, and Stochastic Nodal Generation methods

		NFGL Framework			Error Diffusion
		S_R			
	n	2	3	4	
Time (s)	0.25	0.777	0.689	0.536	0.994
	4	0.770	0.573	0.505	1.277

The results presented in this section show the advantage of the NFGL Framework over all of the other existing methods. It confirmed the expected advantages of the NFGL Framework stated in section 2.5. The generated NFGL structures are conformal, deterministic, robust, with easily adjustable unit-cell size ratios and at a low computational cost.

4.2 Automotive Control Arm Optimization Under Multiple Loading Conditions

This section will demonstrate the application of the NFGL Framework in optimizing an automotive control arm from an example provided by Optistruct for lattice optimization [89], shown in Figure 128, and highlighting the benefits of using NFGL structures over the FGL structure design. The control arm will be optimized and enhanced with lattice structures to reduce the weight and the generated lattice structures would be subjected to a sizing optimization after to further enhance the performance of the control arm. The use of the NFGL structure in the sizing optimization will also be investigated against regular FGL structures.

The control arm is subjected to two loading conditions on the control points P_A, P_B, P_C, P_D, and P_E that lie in the center of the holes in the geometry and are connected with rigid beam elements to the surface of the holes. Table 18 shows the loading condition for both cases and Table 19 shows the material properties for the control arm. The control arm was optimized using the SIMP optimization in Optistruct with a penalty of 1.8 to minimize the combined compliances of both load cases under a mass constraint of 11kgs. The optimization result is shown in Figure 129. Elements with relative densities between 0.05 and 0.8 were replaced by lattice structures and their diameters were calculated based on their relative density accordingly. Figure 130 shows the generated FGL structure generated based on the optimization result from [89].

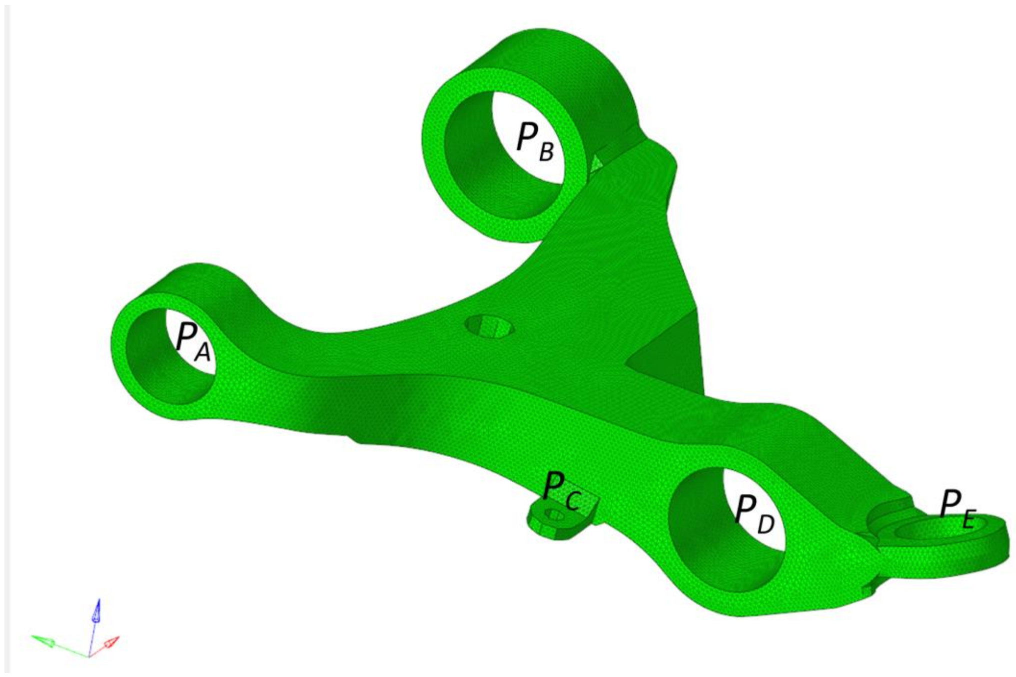


Figure 128 Automotive control arm that will be optimized using lattice structures

Table 18 Automotive control arm boundary and loading condition for both load cases

		Case 1			Case 2		
		Load		Constraint	Load		Constraint
		N	N.mm	mm	N	N.mm	mm
P_A	*x*	-5.21	2.07		932.90	0.86	
	y	21.21	0.70		-815.49	-0.18	
	z	4.25	-2.65	0	438.10	1.49	0
P_B	*x*	-1908.82	2.07		881.47	0.86	
	y	-2357.51	0.70	0	2248.27	-0.18	0
	z	10.00	-2.65	0	255.80	1.49	0
P_C	*x*	0.59			-0.51		
	y	0.34			0.24		
	z	-1.82			-1.64		
P_D	*x*			0			0
	y			0			0
	z			0			0
P_E	*x*	1301.31			-923.45		
	y	-313.71			-272.00		
	z	1019.46			814.81		

Table 19 Automotive control arm material properties

Property	**Value**
E (MPa)	160
υ	0.25
ρ_{solid} (kg/m^3)	7100

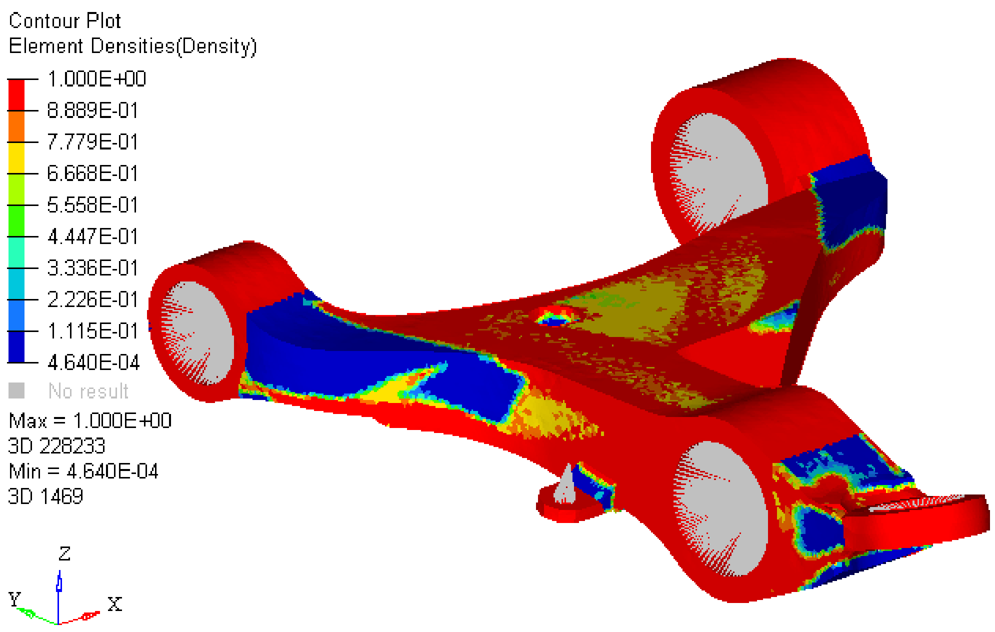


Figure 129 SIMP optimization results for the automotive control arm

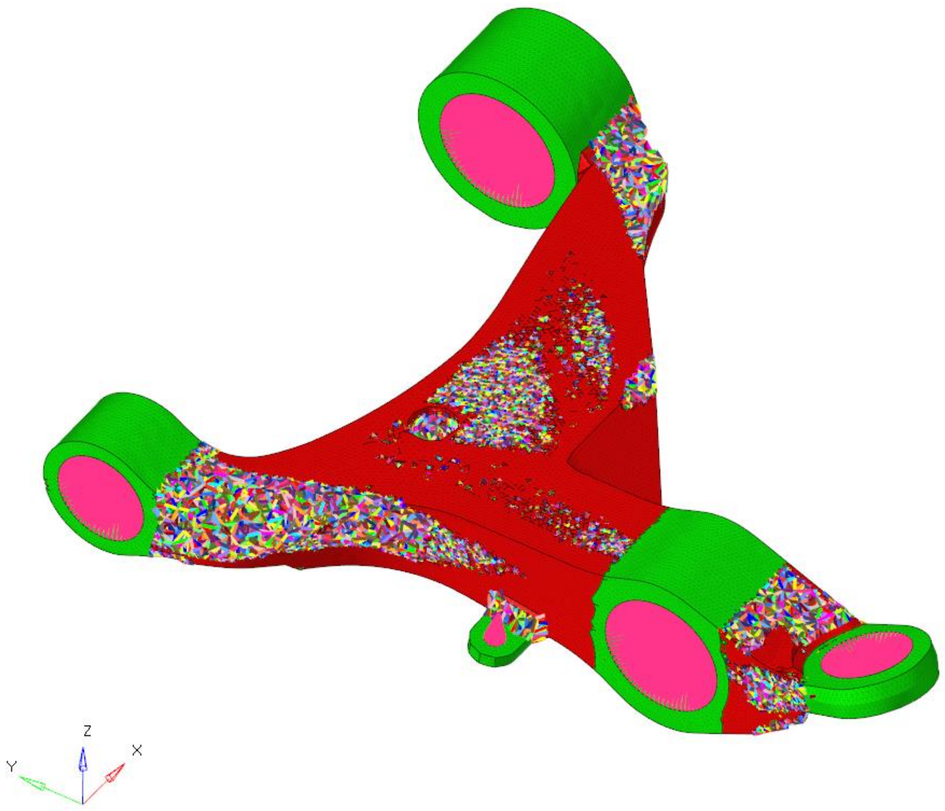


Figure 130 Generated FGL structure for the automotive control arm

To compare the performance of NFGL structures with the generated FGL structure, several NFGL structures were generated using $S_R = 2$, 3, 4, 5, 8 and 15 (Figure 131 shows the generated NFGL structure with $S_R = 3$). The combined compliances of both structures were evaluated before and after the sizing optimization was performed. Figure 132 shows the combined compliance of the control arm before and after size optimization was performed on the generated lattice structures at different S_R values. The value of $S_R = 1$ corresponds to the regular FGL structure. As shown in the figure, the compliance improves s the value of S_R increases and then starts to deteriorate after a certain threshold. This shows that using NFGL structures indeed provides improvement to the structures. Furthermore, even after optimization, there is a slight improvement from NFGL structures over FGL structures. But, the biggest improvement for NFGL structures of FGL structures is the computational time required to do the size optimization process. Figure 133 shows the computational time required to run the sizing optimization with and without the computational time needed to generate the NFGL structure. From the figure, it is clear that the use of NFGL structures reduces the computational time for size optimization significantly without incurring any significant increase in the computational cost. This is due to the reduced number of design variables to be considered as the value of S_R increases.

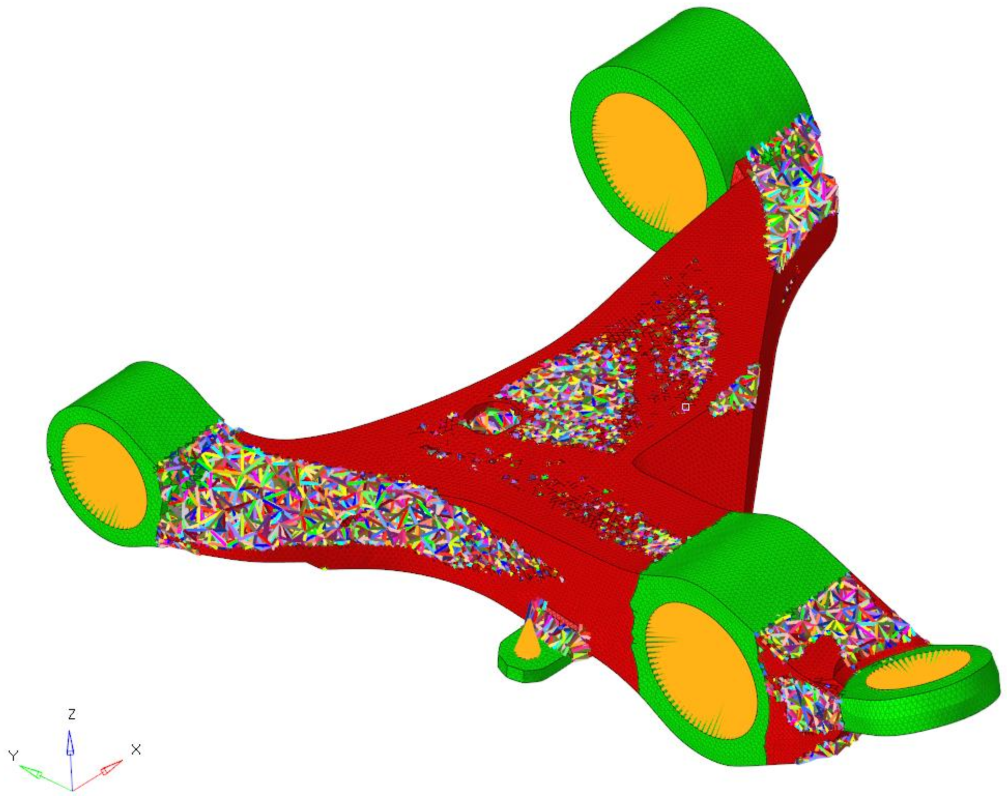


Figure 131 Generated NFGL structure for the control arm with $S_R = 3$

The MSSIM index value is shown in Figure 134 as the value of S_R increases. From the figure, it can be seen that the threshold of 70-75% does indeed fall into the region where the NFGL structure shows improvement over the FGL structure. By utilizing Algorithm 3 and Algorithm 4, a suitable S_R value can be determined. Table 20 shows the adjustments of S_R value using Algorithm 4. From the table, a value of S_R around 3 shows to fall into the required threshold, which agrees with Figure 134. The time required to conduct this search was 206.1 seconds, which is not a significant increase.

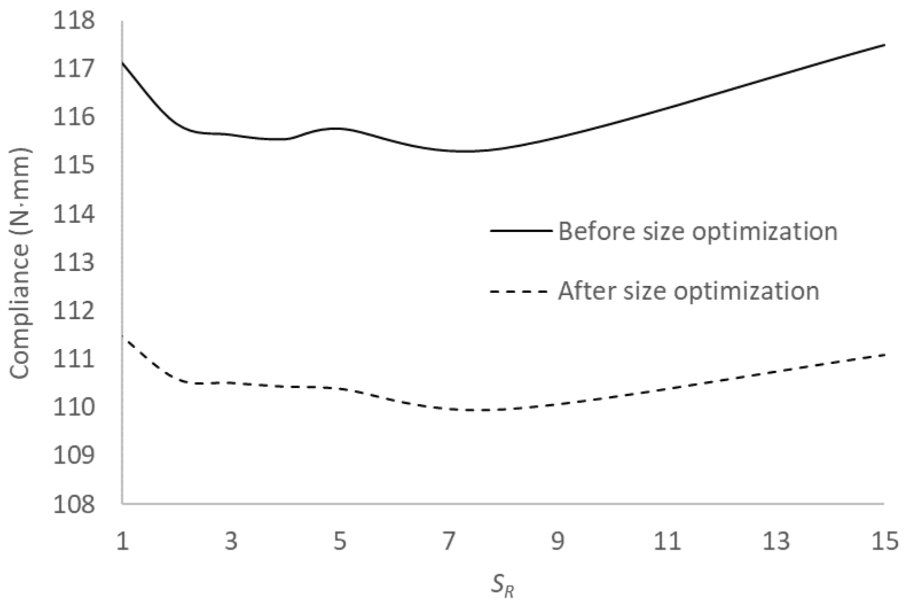


Figure 132 Automotive control arm combined compliance before and after size optimization at different S_R values

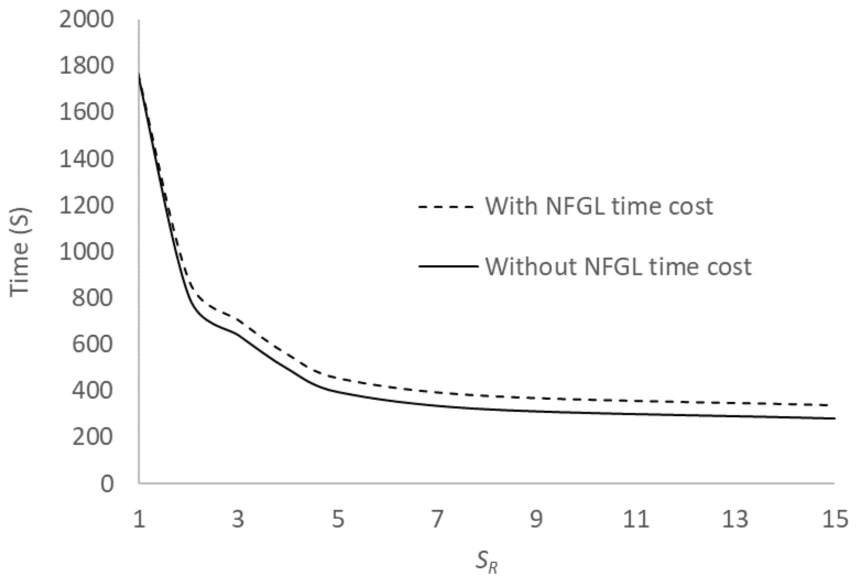


Figure 133 Computational cost of conducting size optimization for the control arm lattice struts at different S_R values

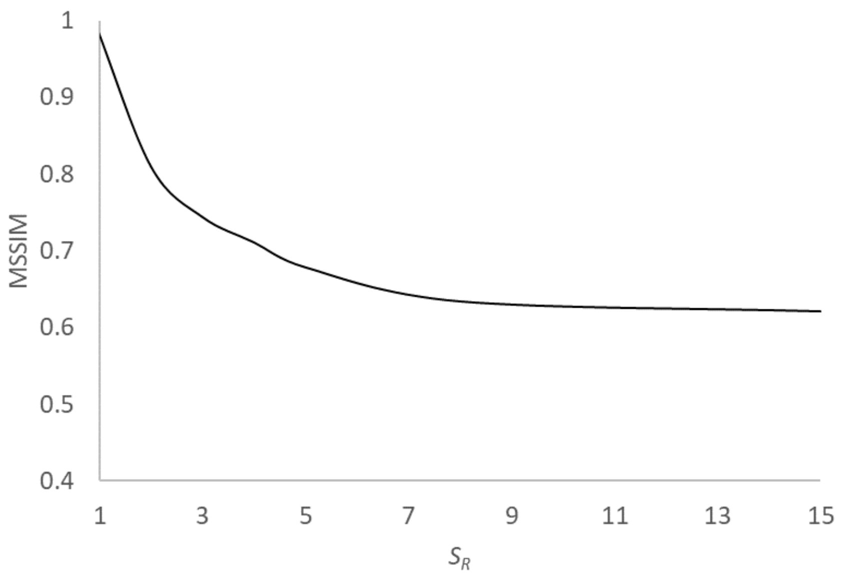


Figure 134 MSSIM index for the automotive control arm as a function of S_R

Table 20 Determining S_R value for the control arm using Algorithm 3 and Algorithm 4

MSSIM	S_R
0.8533	2
0.6968	3.5134
0.7401	3.1824

4.3 Injection Mold Lattice Cooling Channel Design

The performance of NFGL structures will be demonstrated in this section in handling a thermomechanical problem of designing the cooling channel section of an injection mold and comparing the results of the NFGL structure with the FGL structure generated in [55]. Figure 135 shows the boundary and loading conditions of the cooling channel. One-fourth of the design domain was modeled due to its symmetry and appropriate symmetry boundary conditions were added. The dimensions of the cooling channel are 100×100mm and the outside surfaces are subjected to a uniform injection pressure of $f_1 = 2\text{MPa}$, a side clamping pressure of $f_2 = 1\text{MPa}$, and a surface heat flux of $\Gamma_q = 1\text{mW/mm}^2$. The center of the cooling channel is constrained at a temperature of $P_T = 0$. A convection coefficient of $h = 0.01\text{mW/mm}^2$ was applied normal to the surface to represent natural convection. The material used had an elastic modulus of E = 1MPa and thermal conductivity of k = 1W/(m·k).

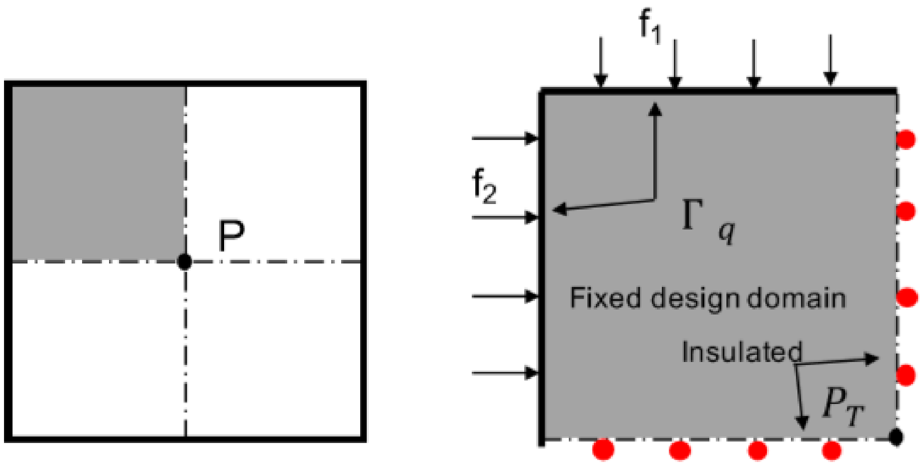


Figure 135 Loading and boundary conditions for one-fourth of the injection mold cooling channel [55]

The cooling channel in [55] was optimized using the SIMP optimization method by deriving the elastic and thermal properties matrices in terms of the porosity of a square lattice unit-cell using Asymptotic Homogenization. However, this method is computationally expensive due to the increased nonlinearity of the design variables to describe the stiffness matrices in terms of the unit-cell topology as explained in section 1.3. Thus, the design domain was meshed using 10×10 quadrilateral elements to reduce the computational cost. The optimization formulation was as follows:

$$\min(C) = w_1 C_M(\boldsymbol{\rho}) + w_2 C_T(\boldsymbol{\rho}) \tag{56}$$

Subject to

$$C_M = \mathbf{U}^T(\boldsymbol{\rho})\mathbf{K}_m(\boldsymbol{\rho})\mathbf{U}(\boldsymbol{\rho}) \tag{57}$$

$$C_T = \mathbf{T}(\boldsymbol{\rho})^T \left(\mathbf{K}_t(\boldsymbol{\rho}) + \mathbf{K}_t^h(\boldsymbol{\rho})\right)\mathbf{T}(\boldsymbol{\rho}) \tag{58}$$

$$\mathbf{K}_{\boldsymbol{m}}\mathbf{U} = \mathbf{F} \tag{59}$$

$$\left(\mathbf{K}_t(\boldsymbol{\rho}) + \mathbf{K}_t^h(\boldsymbol{\rho})\right)\mathbf{T}(\boldsymbol{\rho}) = \mathbf{q} \tag{60}$$

$$\rho_{min} \leq \rho \leq 1 \tag{61}$$

$$\sum_{i=1}^{N_e} V_i \rho_i \leq V_{target} \tag{62}$$

where C_m and C_T are the mechanical and thermal compliances of the structure; w_1 and w_2 are the weights for the mechanical and thermal compliance (3 and 1 in this example); K_m, K_t and K_t^h are the elastic, conduction, and convection stiffness matrices respectively; **T** is the temperature vector and **q** is the thermal load vector. The target volume is 41% of the initial volume. The results of the optimization in [55] are shown in Figure 136.

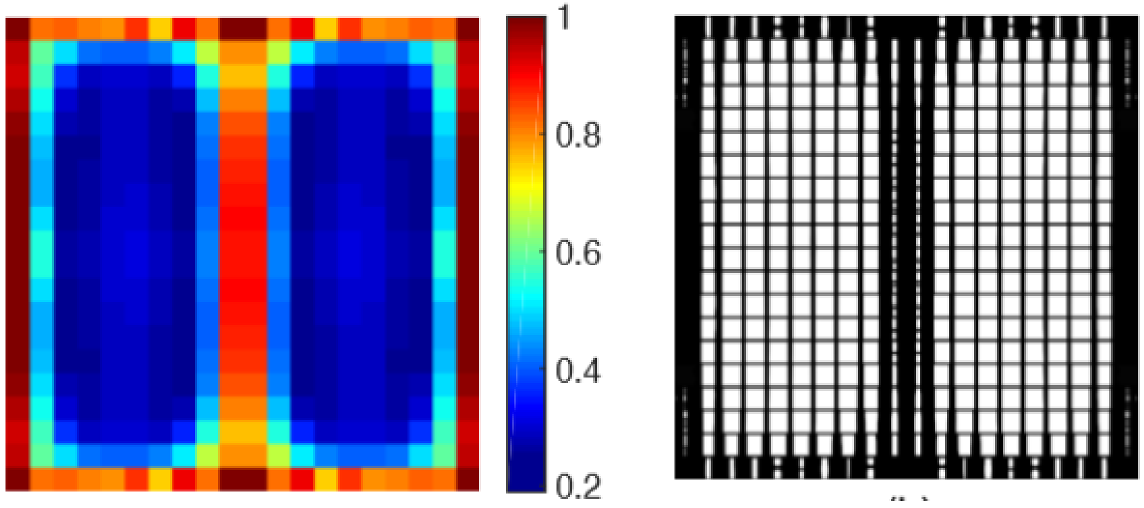


Figure 136 Optimized injection mold cooling channel using FGL structures via AH [55]

To apply the NFGL framework, the same optimization problem in Eq. 56 was solved using the unpenalized SIMP optimization in Optistruct since using the result in Figure 136 would not be suitable for different types of unit-cells. The same number of unit-cells was also used to compare the performance of the NFGL structure with the FGL structure in Figure 136. The optimization process took 4 seconds to finish and the results are shown in Figure 137. To check how the thermomechanical performance of the generated NFGL structure varies, S_R values of 2, 3, 4, 5 and 6 will be used. Figure 138 shows the generated NFGL structures while Figure 139 and Figure 140 show the results of the mechanical and thermal compliances relative to the FGL structure respectively. The mechanical compliance shows the expected behavior of improvement until the NFGL structure begins to deviate from the density input. Then, the compliance starts to worsen as S_R values begins to increase. Compared to the optimized FGL structure, the NFGL structure can reach almost similar compliance relatively without the need to conduct computationally expensive topology optimization using AH. As for the thermal compliance of the NFGL structure, a different behavior is noticed. The thermal compliance starts better in the case of $S_R = 1$ comapred to the optimized FGL design, but then starts to increase until it becomes higher. This behavior is due to the reduction in surface area for convection heat transfer as S_R values increases as shown in the illustration in Figure 141. The figure shows two square structures, with side length l, and thickness t and a porosity of 64%. The structure in Figure 141-a has four unit-cells while the structure in Figure 141-b has only one unit-cell. Both have the same top surface area due to having the same porosity. But the side surface area is higher in the case of Figure 141-a, which increases the area where convection heat transfer can occur. But despite that, the NFGL structure with S_R values between 2-4 and can perform relatively well with the optimized FGL structure and at a low computational cost, which further shows the benefits of varying the unit-cells' size ratio.

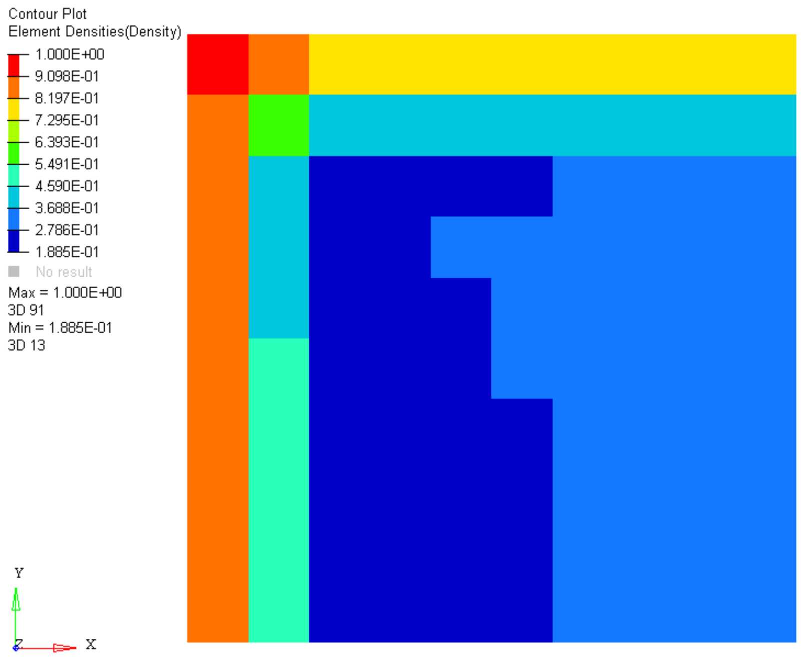


Figure 137 Optistruct optimization results for the injection mold cooling channel using a size of 10×10

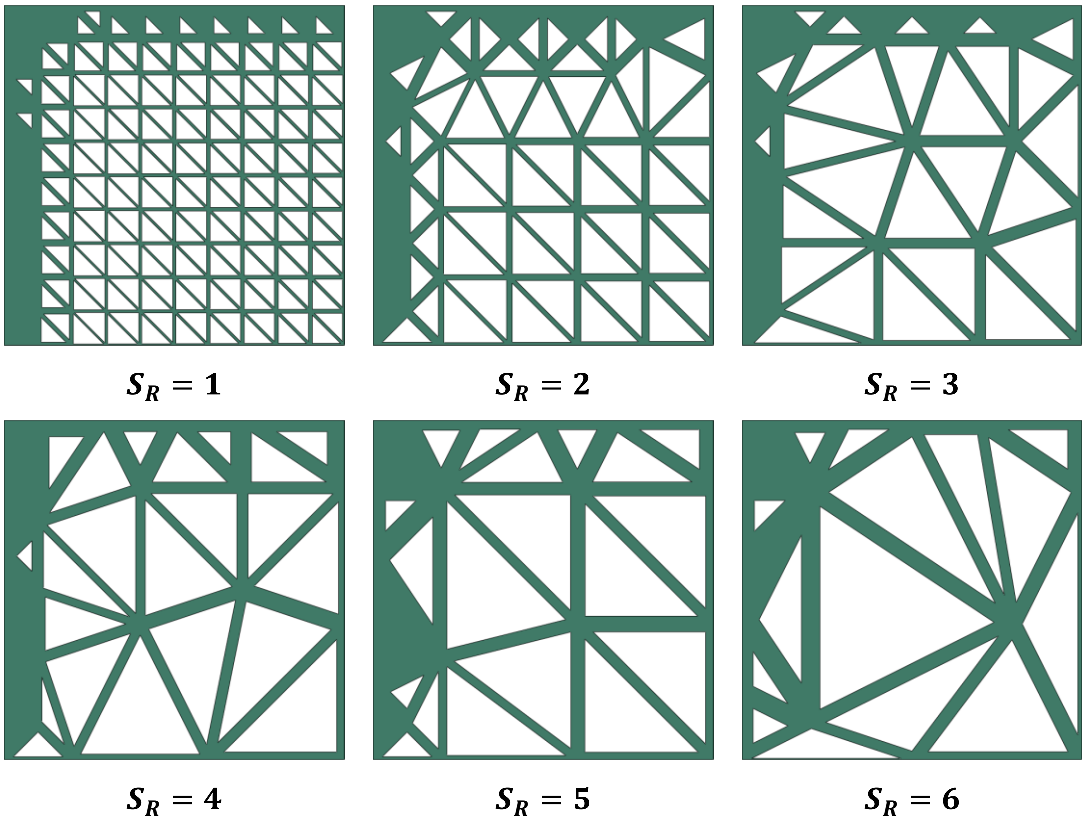


Figure 138 Generated NFGL structures for the injection mold cooling channel using a size of 10×10 with different S_R values

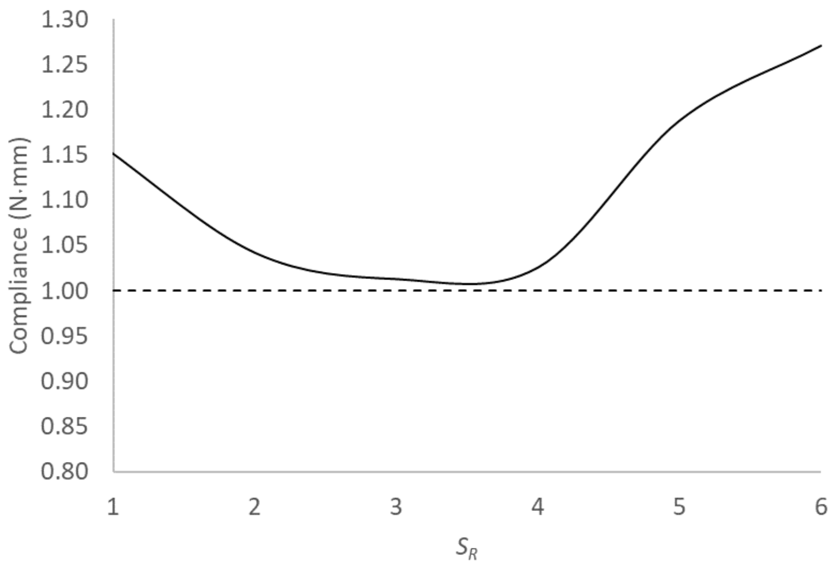


Figure 139 Relative mechanical compliance of the NFGL structure for the injection mold cooling channel with a size of 10×10

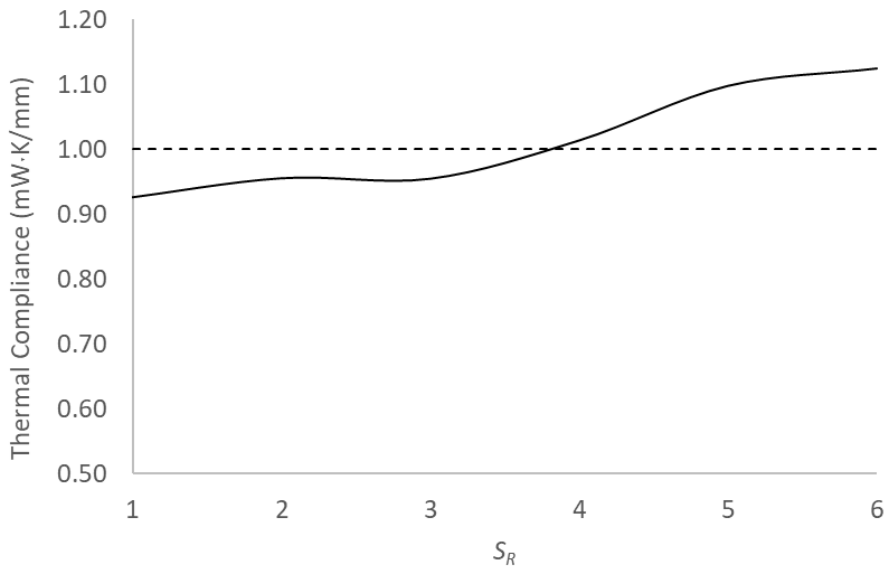
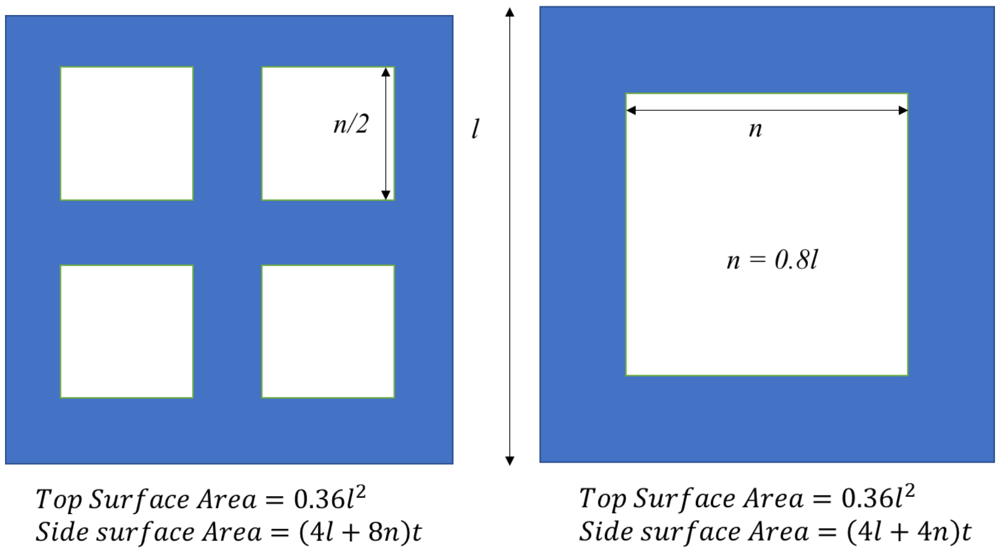


Figure 140 Relative thermal compliance of the NFGL structure for the injection mold cooling channel with a size of 10×10



$n/2$
l
n
$n = 0.8l$
$Top\ Surface\ Area = 0.36l^2$
$Side\ surface\ Area = (4l + 8n)t$
$Top\ Surface\ Area = 0.36l^2$
$Side\ surface\ Area = (4l + 4n)t$

Figure 141 Comparison of the total surface area subjected to convection heat transfer for two structures with different unit-cell sizes

Since only a grid of size 10×10 was used to generate the NFGL structure, the variation in unit-cell size ratio was very limited. To further investigate how the NFGL structures can behave in this example, the number of elements was increased to 100×100 elements. Figure 142 shows the topology optimization result in Optistruct using 100×100 elements, which took 90 seconds to solve. From the optimization results, the NFGL structure was generated using S_R values of 2, 4, 5, 10, 12, 15, 20, 25, 30, 40 and 50 as shown in Figure 143. Unlike the 10×10 NFGL structures, the 100×100 NFGL structures outperform the optimized FGL structure in both mechanical and thermal compliances as shown in Figure 144 and Figure 145. Even though the thermal compliance shows the same behavior of increasing as S_R value increases, it is still lower than the optimized FGL structure and can be obtained at a very efficient computational cost, further showing the benefits of NFGL structures over FGL structures.

Figure 146 and Figure 147 show the MSSIM index as the value of S_R increases for both the 10×10 and 100×100 cases respectively. The 10×10 case shows jumps in the data points due to the small number of unit-cells used in the calculation of the MSSIM index. However, the trend is similar to what is expected. The 100×100 case shows the expected behavior. Furthermore, the threshold of 70-75% is a viable range where the NFGL structure shows improvement over the FGL structure, which is similar to previous examples and further supports the recommendation of using that range. Table 21 shows the results of applying Algorithm 3 and Algorithm 4 to determine an appropriate S_R value for the NFGL structure. The algorithms took 0.2 seconds for the case of 10×10 and 3.7seconds for the case of 100×100. Based on the results, the value of $S_R = 3$ and $S_R = 10$ is suitable for the 10×10 and 100×100 cases respectively, which agrees with the results of the compliances.

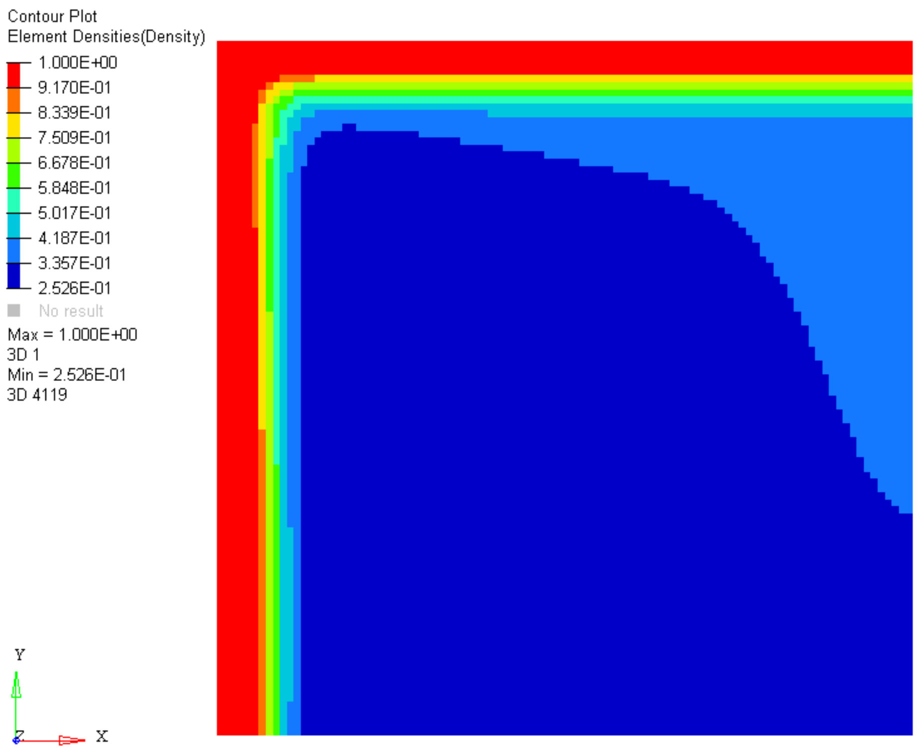


Figure 142 Optistruct optimization results for the injection mold cooling channel using a size of 100×100

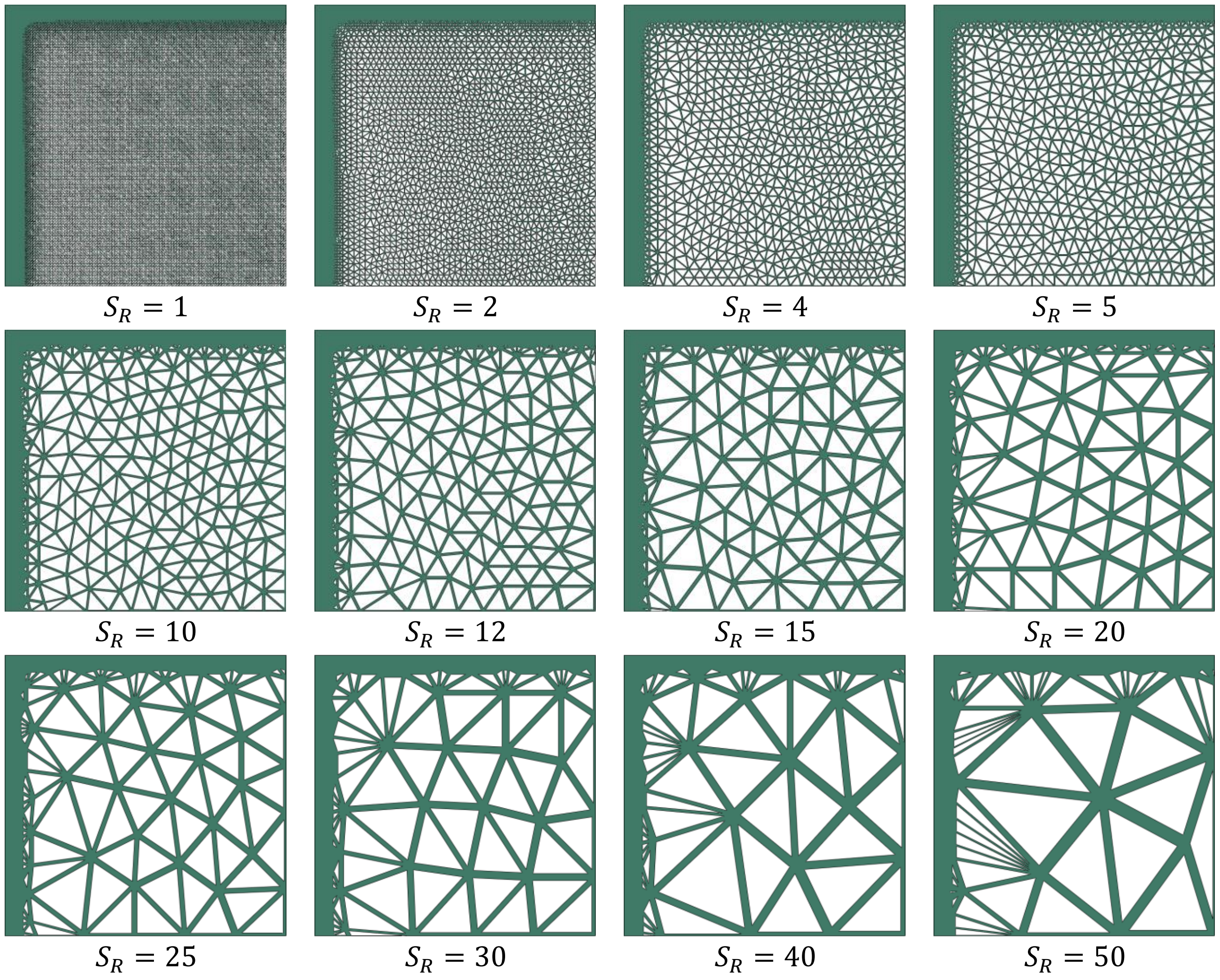


Figure 143 Generated NFGL structures for the injection mold cooling channel using a size of 100×100 with different S_R values

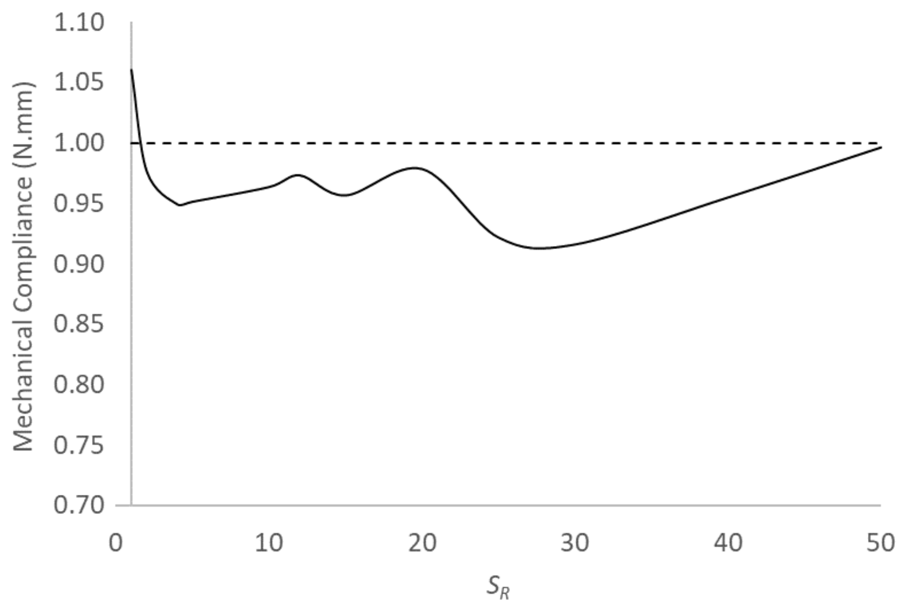


Figure 144 Relative mechanical compliance of the NFGL structure for the injection mold cooling channel with a size of 100×100

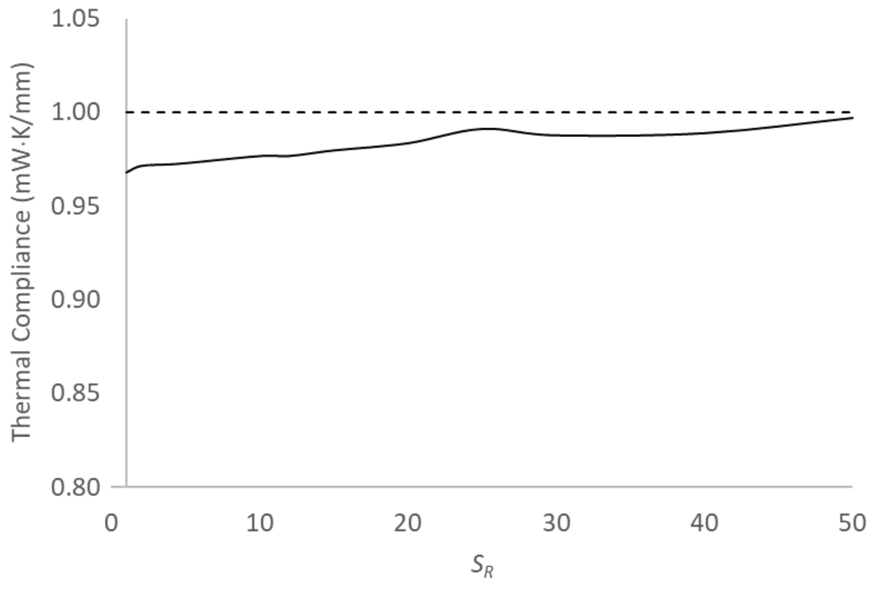


Figure 145 Relative thermal compliance of the NFGL structure for the injection mold cooling channel with a size of 100×100

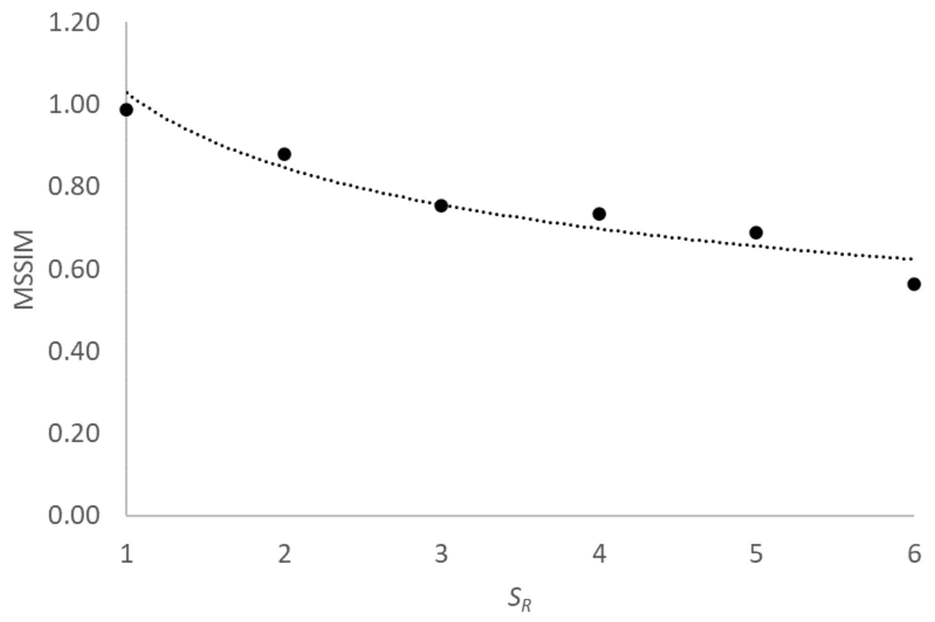


Figure 146 MSSIM index for the NFGL structure using a size of 10×10 at different S_R values

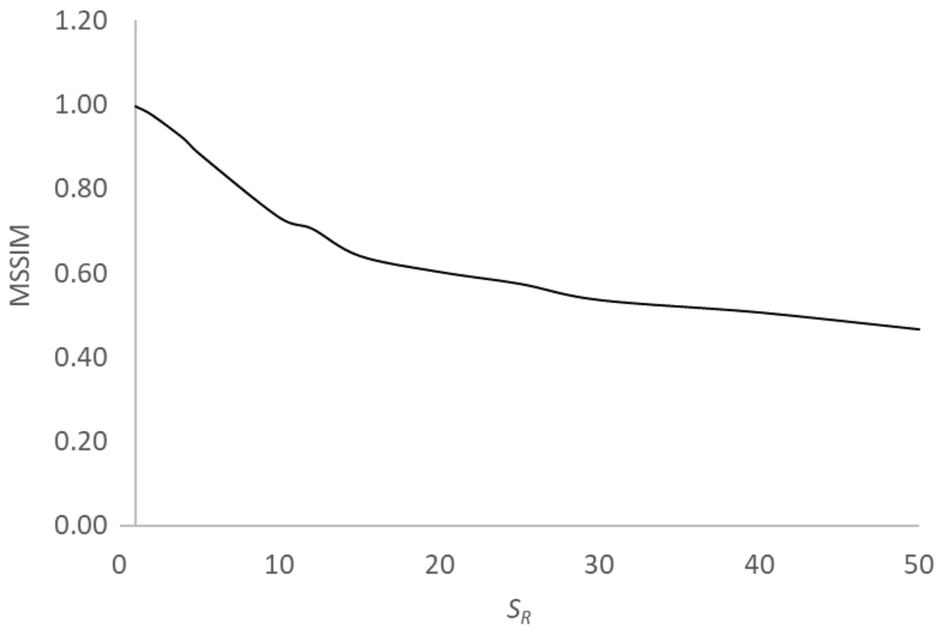


Figure 147 MSSIM index for the NFGL structure using a size of 100×100 at different S_R values

Table 21 Determining S_R value injection mold cooling channel NFGL structure using Algorithm 3 and Algorithm 4

size	MSSIM	S_R
10×10	0.8493	2
	0.7196	4.18
100×100	0.8823	5
	0.4405	87.7
	0.7554	9.04
	0.7323	9.91

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

This chapter concludes the dissertation and how the work in this dissertation addresses the research questions and provides the contributions made in section 5.1, the current limitations of this work in section 5.2, and potential areas for future work to further improve the NFGL Framework in section 5.3.

5.1 Contributions

This section will revisit the research questions that were posed in section 1.5 and discuss how the current research addresses them, based on the results and how it aligns with the hypotheses. Moreover, a list of the contributions that the current research provides will be given.

5.1.1 Addressing the Research Questions

One of the goals of this research was to develop a framework for designing conformal NFGL structures in a computationally efficient and deterministic manner. The major issues that were present in existing work in the literature were computational cost, uncertainty, and non-conformity to the design domain. Existing methods that can overcome one of these issues suffer from another. So these existing drawbacks led to the first research question.

Research Question 1:

How can we reduce the computational cost while controlling the randomness in nodal placements to generate conformal naturally functionally graded lattices?

Based on hypothesis 1, using an FEM mesh grid that is conformal to the design domain would allow the generation of NFGL nodes in a deterministic and conformal manner. This was based on the fact that the Error Diffusion method can generate NFGL structures at a low computational cost in a deterministic way, but does not generate conformal structures. So if the nodes were conformal to the design domain, the generated structure would also be conformal. The results of the generated structures using the NFGL Framework in CHAPTER 4 agrees with the hypothesis, especially in section 4.1.4. This was possible due to the fact that the NFGL Framework does not treat the grid nodes as pixels, but rather as points in space where the distance of each point is taken into consideration when generating NFGL nodes, which is something that the Error Diffusion method does not consider. And since the location of the nodes was predetermined in the grid, and that process of removing nodes is deterministic, the generated NFGL structure itself would be deterministic.

Hypothesis 1 also stated that the removal of nodes should not be done in an iterative manner where a computationally expensive process like topology optimization would be needed, like in the Local Volume Constraint and Adaptive Quadtree approach. Since the NFGL Framework does not rely on topology optimization in each iteration when generating NFGL nodes, the nodes are generated in a computationally low cost that is similar to the Error Diffusion and even better in some cases based on what unit-cell size ratio was used as can be seen in the results of section 4.1.1. Thus, it is apparent that the NFGL Framework addresses the concerns in research question 1.

Another goal of the research was to provide the ability to freely and easily generate NFGL structures with different unit-cell size ratios without strict restrictions or limitations.

As seen in existing work, all of the methods, except for the Adaptive Quadtree and similar methods, impose a restriction on the size ratio that can be generated. This restriction is placed either explicitly like the Local Volume Constraint or implicitly like the Error Diffusion and Stochastic Nodal Generation. The Local Volume Constraint method is restricted by the radius of neighboring elements that account for the local volume constraint, which as explained earlier prevents the accumulation of elements that can help in the generation of smaller cells. The Error Diffusion method restricts the largest unit-cell size because of the filter used and pixel density in the input density field. While the Stochastic Nodal Generation method restricts it based on the scale of the design domain. Both methods require adjustments to the density field input in order to change the unit-cell size ratio, but the attainable size ratios are still restricted, and correlating the intensity of the input density field is not intuitive to the user. As for methods like the Adaptive Quadtree that can create different unit-cell size ratios, increasing the number of levels requires increasing the FEM mesh size, which significantly increases the computational cost associated with conducting topology optimization. Furthermore, the methods do not generate conformal NFGL structures. These issues mentioned led to the second research question.

Research Question 2:

How can we remove the restriction on the unit-cell size ratio of NFGL structures in a computationally efficient way?

Hypothesis 2 stated that nodes should have the ability to control if a node should be adjacent to it or not from the grid proposed in hypothesis 1. This frees the method from relying on a fixed-sized range to affect the unit-cell size ratio as in the case of the Local Volume Constraint and Error Diffusion. But rather gives the node the ability to create its own range of influence. It further frees the node from being restricted to the design domain scale and makes it easier for the user to adjust the unit-cell size ratio without having to alter the input density field. This has led to the idea of using a simplified sphere packing process in the NFGL Framework, which would allow nodes to remove other nodes in their vacancy if they were inside their influence sphere and of lower density value. The results in CHAPTER 4 show how the NFGL Framework is able to adjust the unit-cell size ratio of the generated NFGL structure at a competitive computational cost. Furthermore, as the size ratio increases, the computational cost reduces. The highest computational cost is when the S_R value is 1, which just generates a regular FGL structure that can be generated without the NFGL Framework. This makes the NFGL Framework more favorable when exploring the effects of the change in the unit-cells size ratio in a computationally efficient manner and thus showing that the simplified sphere packing process in the NFGL Framework is sufficient to answer research question 2 and addresses the concerns posed in the question.

The third goal of this research was to provide better performance of generated NFGL structures against existing FGL structures since the work in the literature regarding that is lacking. The work in the literature focused mainly on the generation process of these NFGL structures without comparing it against FGL structures in terms of their performance. However, methods like the Adaptive Quadtree and Local Volume Constraint methods show that variations in unit-cell size do improve the performance of the structure in their results. The results of these methods are driven by the topology optimization process in each iteration. So a method like the NFGL Framework that can freely control the unit-cell size ratio should experience improvement in performance with the increase in the size ratio. But there would be a point where the performance would start to drop due to the generated structure no longer representing the input density field. This issue has led to the third research question.

Research Question 3:

How can we determine an appropriate unit-cell size ratio to improve the performance of NFGL structures to satisfy multifunctional requirements?

The third hypothesis states that the performance of the generated NFGL structure is correlated with how the generated structure relates to the input density field. But since the input density field and generated NFLG structure are not the same type of data, it was proposed in this research to convert the NFGL structure into an approximate density field and utilize the MSSIM index to determine the similarity between the input density field and the approximate field from the NFGL structure. By observing the value of the MSSIM index as the value of S_R increases in multiple examples, it was possible to determine that the MSSIM index had a power function relation with the value of S_R. This allowed the integration of the MSSIM index into the NFGL Framework by introducing an algorithm that updates the value of S_R. The updating process relies on the observation that an MSSIM index value between 70%-75% shows an improvement in the performance of the NFGL structure compared to the regular FGL structure. To attain an MSSIM index that lies within that threshold, a linear regression on a linearized power function is conducted. Even though this process required regeneration of the NFGL structure for each iteration, the low computational cost of the NFGL Framework and the low number of iterations needed did not cause much increase in the overall computational cost as exhibited in the application examples in CHAPTER 4. This allowed the NFGL Framework to address the third research question and showed that NFGL structures do provide improved performance compared to FGL structures.

5.1.2 List of Contributions

In this section, the contributions of this research will be highlighted.

A novel method for the generation of NFGL nodes in a computationally efficient way: We presented in this work an algorithm to generate NFGL nodes using a newly developed simplified sphere packing algorithm that can generate conformal NFGL structures for any design domain in a deterministic and computationally efficient manner, which addresses the concerns in research question 1. Furthermore, The developed algorithm can help generate NFGL structures with varying unit-cell size ratios in a computationally efficient manner without altering the input density field, which addresses the issues in research question 2 unlike the existing methods in the literature. Moreover, the presented examples in section 1.1.14.1 demonstrated the capabilities of the newly developed algorithm in generating NFGL nodes by evaluating its computational cost, robustness, and conformity of the generated NFGL structures to the design domain against other work in the literature.

A framework for the generation of conformal NFGL structures with low computational cost in a deterministic way: As seen from existing work in the literature, they all suffer from multiple drawbacks that limit their capability of truly generating and investigating the performance of NFGL structures. So the main contribution of this research is the development of the NFGL Framework to create NFGL structures without suffering from the limitations that exist in the literature. The NFGL Framework is able to overcome the limitations of existing work by utilizing a grid of conformal FEM mesh nodes and a simplified sphere packing algorithm. The examples in the dissertation provided a comparison of the NFGL Framework with the other methods in the literature and have shown that the NFGL Framework can create conformal NFGL structures in a deterministic way and with a computational time that is as efficient as the other methods and even better when the unit-cell size ratio is increased. Thus the NFGL Framework is the only method able to achieve that.

Investigation of the performance of NFGL structures: Because of the limitations of the other methods in the literature, it hinders the ability of these methods to fully explore the potential of NFGL structures. The high computational cost of some methods prevents them from exploring the effects of change in size ratio for different applications because of the need to rerun the process for each generated NFGL structure. Methods that place restrictions on the possible size ratios also do not show how NFGL structures behave as the size ratio changes. Furthermore, the change in the unit-cell size ratio requires adjusting the input density field pixel intensity values, which is not intuitive to users. As for methods that rely on stochastic algorithms, the uncertainty in the generated structures requires multiple runs to quantify the appropriate properties of the structure. And given that the methods in the literature suffer from more than one of these drawbacks, their ability to improve the generated structures becomes limited.

Since the NFGL Framework does not suffer from these drawbacks, it can easily generate multiple NFGL structures with the required size ratios without incurring an expensive computational cost or altering the input density field as the examples show. In the examples, the performance of the generated structures using the NFGL Framework was investigated, and based on the results of the examples, a threshold was determined that would produce an improved structural performance. Furthermore, a deterioration in the performance of the NFGL structure was addressed when the size ratio increases and was tied to the deviation of the NFGL structure from the input density field.

Correlation of NFGL structures to input density fields: One of the challenges in this research was correlating the change in the unit-cell size ratio of the NFGL structures being generated by the NFGL Framework. This was because the NFGL structure data represent nodal coordinates, connectivity, and strut geometry; while the input density field represents relative density values inside the design domain. For this reason, the use of the MSSIM index was proposed in this research, since the index value ranges from 0 to 1 based on how similar two images are. However, the implementation of the MSSIM index required representing the NFGL structure as an approximate density field and modifying the MSSIM index to be used on different geometrical shapes. The modified MSSIM algorithm was then implanted in the NFGL Framework to correlate the two density fields based on the value of the MSSIM index and the change in the unit-cell size ratio. This showed that there is a power relation between the two density fields as the size ratio increases, which allowed the NFGL Framework to determine the appropriate size ratio that falls within the desired threshold as shown in the application examples provided in this research.

Investigation and improvement of the performance of NFGL structures against FGL structures: One of the goals of this research was to investigate and improve the performance of NFGL structures with FGL structures since the research into this topic was lacking. So in this research, the example in section 4.1.2 focused on investigating the performance of NFGL structure with FGL structures based on their compliance values. And the results have shown that NFGL structures do give improved compliance compared to a regular FGL structure. The control arm example in section 04.2 aimed to also investigate the performance improvement from using NFGL structures over FGL structures and have shown similar results to the example in section 4.1.2. Furthermore, it also showed that performing size optimization on NFGL structures reduced the computational time of the optimization process significantly compared to FGL structures while producing similar results. This was because the NFGL structure had a reduced number of design variables as the unit-cell size ratio increases. Thus showing that NFGL structures are a more favorable choice when conducting size optimization for lattice structures. The example in section 4.3 investigated the mechanical and thermal compliance of an optimized FGL structure of an injection mold cooling channel using AH against NFGL structures without any modification to the SIMP optimization. The NFGL structure showed that it can reach comparable results to the FGL structure when using the same unit-cell size for the mechanical compliance and showed better results in some size ratios for the thermal compliance. Furthermore, when the unit-cell size was reduced and more unit-cells were generated, the results of the NFGL structure outperformed the FGL structure in both mechanical and thermal compliances. Thus, showing that structures with high performance can be generated without using computationally expensive topology optimization with AH.

5.2 Current Limitations

This section will outline the limitations of the work in this dissertation.

No control over struts orientation: The NFGL Framework currently cannot control the orientation of the generated struts in the NFGL structure. This would pose an issue in applications where a certain orientation is required to follow a certain load path, or in AM processes that require limited orientation angles for the manufactured struts. This is mainly because the nodes that were removed from the FEM grid were removed by an influence sphere. Another reason is that the Delaunay triangulation algorithm generates equiangular triangles as much as possible, so the orientation has no effect on the generated cells.

No additional nodes are placed/removed or adjustment of existing nodes: To reduce the computational cost of the generated NFGL structures, the NFGL Framework does not currently place or remove additional nodes, nor does it adjust the existing nodes it generates. This also contributes to the issue of orientation control on the generated NFGL struts.

Only tetrahedral or triangular unit-cells can be generated: The NFGL Framework currently generates tetrahedral or triangular unit-cells. This is because of the previous limitation of not being able to add or remove nodes in the generated NFGL nodes. This makes the generated NFGL nodes unsuitable to create quadrilateral or hexahedral elements. Thus limiting the generated unit-cells to tetrahedral or triangular.

Requires an iterative process to determine an appropriate unit-cell size ratio: As seen in the provided examples in this dissertation, the determination of an appropriate unit-cell size ratio requires an iterative process in which regression analysis is performed. This requires rerunning the NFGL Framework for each generated size ratio until the desired similarity threshold is achieved. If the NFGL Framework could determine the appropriate ratio without the need to iteratively run the generation process, this could save unneeded computational cost.

The performance of the generated NFGL structures is not the optimal: In the current dissertation, the value of S_R was determined based on the observation that in the threshold of 70%-75% similarity, the NFGL structure will have an improved performance compared to an FGL structure. However, the performance of the generated NFGL structure can still be improved as seen in some examples. So even though the value of S_R guarantees improved performance, it is not the optimal performance of the structure.

5.3 Future work

This research aims to build a platform for the generation of NFGL structures and close the gap in the research of these types of structures, which would facilitate their utilization in different engineering applications. The following suggestions can be beneficial for further research into the field of NFGL structures modeling and design.

Controlling the orientation of generated struts: One approach for controlling the orientation of struts that can be suggested is the use of Solid Orthotropic Material Penalization (SOMP) [90] optimization. SOMP is an extension of the SIMP optimization, which includes the orientation of a material as a design variable in the optimization process. The SOMP formulation would have the following modifications to Eqs 1-3.

$$\min C(\boldsymbol{\rho}, \boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{N_e} \rho_i^p \mathbf{U}_i{}^T \mathbf{K}_i(\theta_i) \mathbf{U}_i \tag{63}$$

Subject to

$$\mathbf{U}(\boldsymbol{\rho}, \boldsymbol{\theta}) = \mathbf{K}(\boldsymbol{\rho}, \boldsymbol{\theta})^{-\mathbf{1}} \mathbf{F} \tag{64}$$

$$\rho_{min} \leq \rho \leq 1, -2\pi \leq \theta \leq 2\pi \tag{65}$$

where θ is the orientation of the material.

The NFGL Framework could be extended to allow for the inclusion of material orientation results. This further coupled with using ellipsoids rather than spheres could allow the generation of NFGL structures with struts of controlled orientation, which would help in addressing this limitation in the current NFGL Framework. A similar approach can be seen in the work in [68]. But the NFGL nodes are controlled by determining the k-nearest neighbors inside an ellipse surrounding a node to connect them and the orientation is only determined from stress fields, which becomes an issue when dealing with multiple loading conditions. Furthermore, the triangulation is done by considering the k-nearest neighbors only, which weakens the generated structures.

Nodal adjustments and generation: In this work, the nodes to generate the NFGL structure are fixed, so they are not adjusted, and no new nodes are generated. Allowing for the adjustments of generated nodes or the addition of new nodes can help not only control the orientation of the struts, but also the generation of hexahedral and quadrilateral elements. However, the computational cost of such an approach should be considered as to not drive execution time up.

The use of other types of unit-cells: The NFGL Framework in this dissertation only considered using Tetrahedral and triangular unit-cells, as they provide stretch dominated unit-cells. If the NFGL Framework was extended to include the generation of hexahedral and quadrilateral elements, these elements can be used as basic unit-cells to generate different types of stretch dominated unit-cells, such as the Octet unit-cell. This would further open the research into the effects of different unit-cell sizes on the performance NFGL structures and the generation of hybrid NFGL structures with multiple unit-cell types.

Determining an optimal unit-cell size ratio: In the current dissertation, the value of S_R was determined based on the observation that in the threshold of 70%-75% similarity, the NFGL structure will have improved performance compared to an FGL structure. However, the performance of the generated NFGL structure can still be improved as seen in some examples. The suggested threshold to determine the value of S_R does not guarantee an optimal solution. Further investigation can be directed into finding the optimal S_R value.

REFERENCES

[1]. Gibson, L.J. and Ashby, M.F., Cellular Solids: Structure and Properties, Cambridge University Press, (1999)

[2]. Fleck, N.A., Deshpande, V.S., and Ashby, M.F., “Micro-Architectured Materials: Past, Present and Future,” Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, **466**, No. 2121, 2495-2516 (2010).

[3]. Maskery, I., Hussey, A., Panesar, A., Aremu, A., Tuck, C., Ashcroft, I., and Hague, R., “An Investigation into Reinforced and Functionally Graded Lattice Structures,” Journal of Cellular Plastics, **53**, No. 2, 151-165 (2016).

[4]. Brennan-Craddock, J., Brackett, D., Wildman, R., and Hague, R., “The Design of Impact Absorbing Structures for Additive Manufacture,” Journal of Physics: Conference Series, **382**, (2012).

[5]. Banhart, J., “Manufacture, Characterisation and Application of Cellular Metals and Metal Foams,” Progress in Materials Science **46**, No. 6, 559-632 (2001).

[6]. Ryan, G., Pandit, A., and Apatsidis, D.P., “Fabrication Methods of Porous Metals for Use in Orthopaedic Applications,” Biomaterials, **27**, No. 13, 2651-70 (2006).

[7]. Wadley, H., “Fabrication and Structural Performance of Periodic Cellular Metal Sandwich Structures,” Composites Science and Technology, **63**, No. 16, 2331-2343 (2003).

[8]. Tao, W. and Leu, M.C., *Design of Lattice Structure for Additive Manufacturing*, in *International Symposium on Flexible Automation (ISFA)*. 2016: Cleveland, OH.

[9]. Gibson, I., Rosen, D.W., and Stucker, B., Additive Manufacturing Technologies, Springer,

[10]. Deshpande, V.S., Fleck, N.A., and Ashby, M.F., “E&Ective Properties of the Octet-Truss Lattice Material,” (2001).

[11]. Ashby, M.F., "The Properties of Foams and Lattices," Philos Trans A Math Phys Eng Sci, **364**, No. 1838, 15-30 (2006).

[12]. Austermann, J., Redmann, A.J., Dahmen, V., Quintanilla, A.L., Mecham, S.J., and Osswald, T.A., "Fiber-Reinforced Composite Sandwich Structures by Co-Curing with Additive Manufactured Epoxy Lattices," Journal of Composites Science, **3**, No. 2, (2019).

[13]. Deshpande, V.S., Ashby, M.F., and Fleck, N.A., "Foam Topology Bending Versus Stretching Dominated Architectures," Acta Materalia, **49**, No. 6, 1035-1040 (2001).

[14]. Dias, W. *Design and Optimization of Lattice Structures for 3d Printing Using Altair Optistruct*. 2015; Available from: https://insider.altairhyperworks.com/design-and-optimization-of-lattice-structures-for-3d-printing-using-altair-optistruct/. Retrieved November 1, 2019

[15]. Materialise. *Materialise 3-Matic Lattice Module*. Available from: https://www.materialise.com/en/software/3-matic/modules/lattice-module. Retrieved November 6, 2019

[16]. *Autodesk Netfabb*. Available from: https://www.autodesk.com/products/netfabb/overview. Retrieved November 1, 2019

[17]. ntopology. Available from: https://ntopology.com/. Retrieved November 9, 2019

[18]. Nguyen, J., Park, S.-i., Rosen, D., Folgar, L., and Williams, J., "Conformal Lattice Structure Design and Fabrication," Solid Freeform Fabrication Proceedings, 138-161 (2012).

[19]. Michell, A.G.M. and Melbourne, M.C.E., "The Limits of Economy of Material in Frame-Structures," **8**, No. 47, 589-597 (1904).

[20]. Zargham, S., Ward, T.A., Ramli, R., and Badruddin, I.A., "Topology Optimization: A Review for Structural Designs under Vibration Problems," Structural and Multidisciplinary Optimization, **53**, No. 6, 1157-1177 (2016).

[21]. Rozvany, G.I.N., Bendsøe, M.P., and Kirsch, U., “Layout Optimization of Structures,” Applied Mechanics Reviews, **48**, No. 2, 41-119 (1995).

[22]. Achtziger, W., “On Simultaneous Optimization of Truss Geometry and Topology,” Structural and Multidisciplinary Optimization, **33**, No. 4-5, 285-304 (2007).

[23]. Hagishita, T. and Ohsaki, M., “Topology Optimization of Trusses by Growing Ground Structure Method,” Structural and Multidisciplinary Optimization, **37**, No. 4, 377-393 (2008).

[24]. Wang, D., Zhang, W.H., and Jiang, J.S., “Truss Shape Optimization with Multiple Displacement Constraints,” Computer Methods in Applied Mechanics and Engineering, **191**, No. 33, 3597-3612 (2002).

[25]. Jamariya, V.N., Panchal, D.D., and Tare, S.R., *Structural Optimization of Truss Using Finite Element Analysis*, in *2018 International Conference on Smart City and Emerging Technology (ICSCET)*. 2018, IEEE: Mumbai, India.

[26]. Von Buelow, P., “Suitability of Genetic Based Exploration in the Creative Design Process,” Digital Creativity, **19**, No. 1, 51-61 (2008).

[27]. Bendsøe, M.P. and Kikuchi, N., “Generating Optimal Topologies in Structural Design Using a Homogenization Method,” Computer Methods in Applied Mechanics and Engineering, **71**, No. 2, 197-224 (1988).

[28]. Sigmund, O., “A 99 Line Topology Optimization Code Written in Matlab,” Structural and Multidisciplinary Optimization, **21**, 120-127 (2001).

[29]. Nagai, K., Igarashi, M., Gea, H.C., and Kikuchi, N., *Automotive Applications of Integrated Structural Optimization*, in *Computational Mechanics ’95*. 1995: Berlin, Heidelberg.

[30]. Bendsøe, M.P., “Optimal Shape Design as a Material Distribution Problem,” Structural optimization, **1**, No. 4, 193-202 (1989).

[31]. Rozvany, G.I.N., Zhou, M., and Birker, T., “Generalized Shape Optimization without Homogenization,” Structural optimization, **4**, No. 4, 250-252 (1992).

[32]. Bendsøe, M.P. and Sigmund, O., “Material Interpolation Schemes in Topology Optimization,” Archive of Applied Mechanics, **69**, No. 10, 635-654 (1999).

[33]. Norato, J.A., “Topology Optimization with Supershapes,” Structural and Multidisciplinary Optimization, **58**, No. 2, 415-434 (2018).

[34]. Gao, W., Zhang, Y., Ramanujan, D., Ramani, K., Chen, Y., Williams, C.B., Wang, C.C.L., Shin, Y.C., Zhang, S., and Zavattieri, P.D., “The Status, Challenges, and Future of Additive Manufacturing in Engineering,” Computer-Aided Design, **69**, 65-89 (2015).

[35]. Cheng, L., Bai, J., and To, A.C., “Functionally Graded Lattice Structure Topology Optimization for the Design of Additive Manufactured Components with Stress Constraints,” Computer Methods in Applied Mechanics and Engineering, **344**, 334-359 (2019).

[36]. Al-Saedi, D.S.J. and Masood, S.H., “Mechanical Performance of Functionally Graded Lattice Structures Made with Selective Laser Melting 3d Printing,” IOP Conference Series: Materials Science and Engineering, **433**, (2018).

[37]. Ramírez-Gil, F.J., Murillo-Cardoso, J.E., Silva, E.C.N., and Montealegre-Rubio, W., “Optimization of Functionally Graded Materials Considering Dynamical Analysis,” Computational Modeling, Optimization and Manufacturing Simulation of Advanced Engineering Materials **49**, 205-237 (2016).

[38]. Spricigo, P.C., Trento, J.P., Bresolin, J.D., Soares, V.F., Ferraz, L.F., and Ferreira, M.D., “Methods of Preparing Flower Stem Samples for Scanning Electron Microscopy,” Ornamental Horticulture, **21**, No. 1, 17-26 (2015).

[39]. Liu, Z., Meyers, M.A., Zhang, Z., and Ritchie, R.O., “Functional Gradients and Heterogeneities in Biological Materials: Design Principles, Functions, and Bioinspired Applications,” Progress in Materials Science, **88**, 467-498 (2017).

[40]. Harrison, N.M. and McHugh, P.E., “Comparison of Trabecular Bone Behavior in Core and Whole Bone Samples Using High-Resolution Modeling of a Vertebral Body,” Biomech Model Mechanobiol, **9**, No. 4, 469-80 (2010).

[41]. Kalita, S.J., Bose, S., Hosick, H.L., and Bandyopadhyay, A., “Development of Controlled Porosity Polymer-Ceramic Composite Scaffolds Via Fused Deposition Modeling,” Materials Science and Engineering: C, **23**, No. 5, 611-620 (2003).

[42]. Leong, K.F., Chua, C.K., Sudarmadji, N., and Yeong, W.Y., "Engineering Functionally Graded Tissue Engineering Scaffolds," J Mech Behav Biomed Mater, **1**, No. 2, 140-52 (2008).

[43]. Burblies, A. and Busse, M., *Computer Based Porosity Design by Multi Phase Topology Optimization,* in *Multiscale & Functionally Graded Materials Conference (FGM2006)*. 2006: Honolulu, HW.

[44]. Nguyen, J., Park, S.-i., and Rosen, D., "Heuristic Optimization Method for Cellular Structure Design of Light Weight Components," International Journal of Precision Engineering and Manufacturing, **14**, No. 6, 1071-1078 (2013).

[45]. Brackett, D., Ashcroft, I., and Hague, R., *Topology Optimization for Additive Manufacturing,* in *Solid freeform fabrication symposium*. 2011: Austin, TX. p. 348-362.

[46]. Alzahrani, M., Choi, S.-K., and Rosen, D.W., "Design of Truss-Like Cellular Structures Using Relative Density Mapping Method," Materials & Design, **85**, 349-360 (2015).

[47]. Panesar, A., Abdi, M., Hickman, D., and Ashcroft, I., "Strategies for Functionally Graded Lattice Structures Derived Using Topology Optimisation for Additive Manufacturing," Additive Manufacturing, **19**, 81-94 (2018).

[48]. Menses, G.A., Pereira, A., and Menezes, I.F.M., "Lattice Structures Design by Means of Topology Optimization," Mecánica Computacional, **36**, 2111-2120 (2018).

[49]. Arabnejad Khanoki, S. and Pasini, D., "Fatigue Design of a Mechanically Biocompatible Lattice for a Proof-of-Concept Femoral Stem," J Mech Behav Biomed Mater, **22**, 65-83 (2013).

[50]. Arabnejad, S., Johnston, B., Tanzer, M., and Pasini, D., "Fully Porous 3d Printed Titanium Femoral Stem to Reduce Stress-Shielding Following Total Hip Arthroplasty," J Orthop Res, **35**, No. 8, 1774-1783 (2017).

[51]. Arabnejad, S. and Pasini, D., "Mechanical Properties of Lattice Materials Via Asymptotic Homogenization and Comparison with Alternative Homogenization Methods," International Journal of Mechanical Sciences, **77**, 249-262 (2013).

[52]. Masoumi Khalil Abad, E., Arabnejad Khanoki, S., and Pasini, D., "Fatigue Design of Lattice Materials Via Computational Mechanics: Application to Lattices with Smooth Transitions in Cell Geometry," International Journal of Fatigue, **47**, 126-136 (2013).

[53]. Matsui, K., Terada, K., and Yuge, K., "Two-Scale Finite Element Analysis of Heterogeneous Solids with Periodic Microstructures," Computers & Structures, **82**, No. 7-8, 593-606 (2004).

[54]. Wu, T., Liu, K., and Tovar, A., "Multiphase Topology Optimization of Lattice Injection Molds," Computers & Structures, **192**, 71-82 (2017).

[55]. Wu, T., Upadhyaya, N., Acheson, D., and Tovar, A., *Structural Optimization of Injection Molds with Lattice Cooling*, in *Volume 2B: 43rd Design Automation Conference*. 2017.

[56]. Jin, X., Li, G.X., and Zhang, M., "Optimal Design of Three-Dimensional Non-Uniform Nylon Lattice Structures for Selective Laser Sintering Manufacturing," Advances in Mechanical Engineering, **10**, No. 7, (2018).

[57]. Kang, D., Park, S., Son, Y., Yeon, S., Kim, S.H., and Kim, I., "Multi-Lattice Inner Structures for High-Strength and Light-Weight in Metal Selective Laser Melting Process," Materials & Design, **175**, (2019).

[58]. Li, D., Liao, W., Dai, N., Dong, G., Tang, Y., and Xie, Y.M., "Optimal Design and Modeling of Gyroid-Based Functionally Graded Cellular Structures for Additive Manufacturing," Computer-Aided Design, **104**, 87-99 (2018).

[59]. Maskery, I., Aboulkhair, N.T., Aremu, A.O., Tuck, C.J., Ashcroft, I.A., Wildman, R.D., and Hague, R.J.M., "A Mechanical Property Evaluation of Graded Density Al-Si10-Mg Lattice Structures Manufactured by Selective Laser Melting," Materials Science and Engineering: A, **670**, 264-274 (2016).

[60]. Al-Saedi, D.S.J., Masood, S.H., Faizan-Ur-Rab, M., Alomarah, A., and Ponnusamy, P., "Mechanical Properties and Energy Absorption Capability of Functionally Graded F2bcc Lattice Fabricated by Slm," Materials & Design, **144**, 32-44 (2018).

[61]. Choy, S.Y., Sun, C.-N., Leong, K.F., and Wei, J., “Compressive Properties of Functionally Graded Lattice Structures Manufactured by Selective Laser Melting,” Materials & Design, **131**, 112-120 (2017).

[62]. Pasko, A., Vilbrandt, T., Fryazinov, O., and Adzhiev, V., *Procedural Function-Based Spatial Microstructures,* in *2010 Shape Modeling International Conference*. 2010. p. 47-56.

[63]. Fryazinov, O., Vilbrandt, T., and Pasko, A., “Multi-Scale Space-Variant Frep Cellular Structures,” Computer-Aided Design, **45**, No. 1, 26-34 (2013).

[64]. Brackett, D.J., Ashcroft, I.A., Wildman, R.D., and Hague, R.J.M., “An Error Diffusion Based Method to Generate Functionally Graded Cellular Structures,” Computers & Structures, **138**, 102-111 (2014).

[65]. Lou, Q. and Stucki, P., *Fundamentals of 3d Halftoning,* in *7th International conference on electronic publishing*. 1998: St. Malo, France. p. 224-39.

[66]. Lu, L., Sharf, A., Zhao, H., Wei, Y., Fan, Q., Chen, X., Savoye, Y., Tu, C., Cohen-Or, D., and Chen, B., “Build-to-Last,” ACM Transactions on Graphics, **33**, No. 4, 1-10 (2014).

[67]. Kuipers, T., Wu, J., and Wang, C.C.L., “Crossfill: Foam Structures with Graded Density for Continuous Material Extrusion,” Computer-Aided Design, **114**, 37-50 (2019).

[68]. Martínez, J., Dumas, J., and Lefebvre, S., “Procedural Voronoi Foams for Additive Manufacturing,” ACM Transactions on Graphics, **35**, No. 4, 1-12 (2016).

[69]. Wang, G., Shen, L., Zhao, J., Liang, H., Xie, D., Tian, Z., and Wang, C., “Design and Compressive Behavior of Controllable Irregular Porous Scaffolds: Based on Voronoi-Tessellation and for Additive Manufacturing,” ACS Biomaterials Science & Engineering, **4**, No. 2, 719-727 (2018).

[70]. Wu, J., Wang, C.C.L., Zhang, X., and Westermann, R., “Self-Supporting Rhombic Infill Structures for Additive Manufacturing,” Computer-Aided Design, **80**, 32-42 (2016).

[71]. Lee, M., Fang, Q., Cho, Y., Ryu, J., Liu, L., and Kim, D.-S., “Support-Free Hollowing for 3d Printing Via Voronoi Diagram of Ellipses,” Computer-Aided Design, **101**, 23-36 (2018).

[72]. Zhang, X., Fang, G., Dai, C., Verlinden, J., Wu, J., Whiting, E., and Wang, C.C.L., "Thermal-Comfort Design of Personalized Casts," Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, Québec City, Canada, 243-254 (2017).

[73]. Wu, J., Aage, N., Westermann, R., and Sigmund, O., “Infill Optimization for Additive Manufacturing-Approaching Bone-Like Porous Structures,” IEEE Trans Vis Comput Graph, **24**, No. 2, 1127-1140 (2018).

[74]. Wu, J., “Continuous Optimization of Adaptive Quadtree Structures,” Computer-Aided Design, **102**, 72-82 (2018).

[75]. Choi, S.-K., Grandhi, R.V., and Canfield, R.A., Reliability-Based Structural Design, Springer, London (2006)

[76]. Gorguluarslan, R.M., Choi, S.K., and Saldana, C.J., “Uncertainty Quantification and Validation of 3d Lattice Scaffolds for Computer-Aided Biomedical Applications,” J Mech Behav Biomed Mater, **71**, 428-440 (2017).

[77]. Mangado, N., Piella, G., Noailly, J., Pons-Prats, J., and Ballester, M.A., “Analysis of Uncertainty and Variability in Finite Element Computational Models for Biomedical Engineering: Characterization and Propagation,” Front Bioeng Biotechnol, **4**, 85 (2016).

[78]. Rozvany, G.I.N., “A Critical Review of Established Methods of Structural Topology Optimization,” Structural and Multidisciplinary Optimization, **37**, No. 3, 217-237 (2008).

[79]. Floyd, R.W. and Steinberg, L., “An Adaptive Algorithm for Spatial Grayscale,” Proceedings of the Society of Information Display, **17**, No. 2, 75-77 (1976).

[80]. Huang, P., Li, Y., Chen, Y., and Zeng, J., "A Digital Material Design Framework for 3d-Printed Heterogeneous Objects," ASME IDETC/CIE 2016 Charlotte, NC, (2016).

[81]. Lo, S.H., Finite Element Mesh Generation, CRC Press, London (2014)

[82]. Lo, S.H. and Wang, W.X., “Generation of Anisotropic Mesh by Ellipse Packing over an Unbounded Domain,” Engineering with Computers, **20**, No. 4, 372-383 (2005).

[83]. Wang, Z. and Bovik, A.C., Modern Image Quality Assessment, Modern Image Quality Assessment, (2006)

[84]. Wang, Z., Bovik, A.C., Sheikh, H.R., and Simoncelli, E.P., “Image Quality Assessment: From Error Visibility to Structural Similarity,” IEEE Trans Image Process, **13**, No. 4, 600-12 (2004).

[85]. Yvinec, M. *Cgal 5.0.2 - 2d Triangulation*. 2020; Available from: https://doc.cgal.org/latest/Triangulation_2/. Retrieved April 7th 2020

[86]. Avanesov, E.T., “Solution of a Problem on Figurate Numbers,” Acta Arithmetica, **12**, 409-420 (1967).

[87]. Jain, R.C., Katsuri, R., and Schunck, B.G., Machine Vision, McGraw-Hill Science/Engineering/Math, (1995)

[88]. Gentle, J.E., Numerical Linear Algebra for Applications in Statistics, Springer, New York, NY (1998)

[89]. Aldous, M. *Lattice Optimisation Setup*. 2015; Available from: https://connect.altair.com/CP/SA/hwhelp/2019.1/os/topics/solvers/os/control_arm_with_draw_direction_constraints_r.htm?zoom_highlightsub=control+arm. Retrieved October 20, 2019

[90]. Jia, H.P., Jiang, C.D., Li, G.P., Mu, R.Q., Liu, B., and Jiang, C.B., “Topology Optimization of Orthotropic Material Structure,” Materials Science Forum, **575**, 978-989 (2008).