Feature-based parametric design automation for complex geometry objects in CAD systems

by

Tianyu Zhou

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mechanical Engineering

University of Alberta

Abstract

With the rapid development of advanced manufacturing technologies, the fabrication of complex structures with enhanced functional properties has been realized. As a result, a wide exploration of the design space for functional geometric objects has been enabled in research and engineering applications. Due to the iterative construction/optimization process of functional structures, the geometries of these objects are complex. In the actual product development process of complex geometry objects, a parametric CAD model with good compatibility and editability is needed. However, the model construction efficiency for complex geometry objects in contemporary CAD systems is low, and the manual construction processes for complex geometry objects are repetitive, tedious, and time-consuming. Therefore, it is important to construct parametric models automatically for complex geometry objects to improve modeling efficiency in CAD systems, which is the aim of the proposed parametric design automation.

There are still some challenges to realize parametric design automation in CAD systems. For parametric modeling, feature-based method is a powerful approach, where the model is properly defined by several editable parameters so that the constructed CAD models are easy to be modified. A CAD model can be flexibly defined and modified in many different ways. However, the modeling efficiency will be low if all design parameters need to be carefully defined and set values, especially for objects with complex geometry features. For design automation, many approaches have been widely applied in product design and optimization processes, where the models of the product are constructed/updated automatically according to specific rules. For example, generative design is an iterative design process which can automatically update design models algorithmically. However, the models created by design automation approaches might not be parametric CAD models and post-processing operations and compatibility of these models in

computer-aided environments are limited. Therefore, it is important to combine the advantages of feature-based parametric modeling and design automation in product development processes.

In this research, feature-based parametric design automation methods in CAD systems were proposed, developed, and applied. In my methods, associative relationships among design parameters are built algorithmically for complex geometry objects so that the designers don't need to define and assign value for every parameter. The concept of associative features is extended to CAD models which are constructed iteratively or by a loop cycle. The modeling process is automatic, and the constructed models are parametric CAD models. A balance between modeling efficiency and design flexibility is made. The concept of generative design is applied to either create a design or build a CAD model iteratively with intelligent algorithms. In this research, two typical types of complex geometry objects, fractal geometry design and material distribution structures, were studied and their parametric design automation was realized in CAD systems.

In my case study, complex geometry models with a background in design and engineering applications were automatically constructed in a CAD system with editable design parameters. For CAD fractal modeling, a CAD fractal model of Koch Snowflake with variant indentation angles was realized in its parametric design automation. A comparison of modeling efficiency between traditional manual construction method (2880 seconds in average) and my proposed automatic method (6 seconds in average) was made in building a fractal CAD model with 768 edges. For modeling material distribution structures in CAD systems, several CAD models of topology optimized structures were constructed automatically. A complex multi-scale structure with 2800 elements of 5 parameters each was automatically constructed in 45 minutes. In this research, all CAD models built in my proposed methods were feature-based parametric models. My methods were found to be of sufficient efficiency and flexibility for parametric design automation.

Preface

This thesis is an original work by Tianyu Zhou, and the research is performed under the supervision of Dr. Carlos F. Lange and Dr. Yongsheng Ma. Some parts of this thesis have been published by or submitted to the journals listed below:

1. Tianyu Zhou, Hengxu Li, Xinming Li, Yongsheng Ma, Carlos F. Lange. "Feature-based Modeling for Variable Fractal Geometry Design Integrated into CAD System". *Advanced Engineering Informatics,* vol. 57, 102006, 2023, ISSN: 14740346. (Chapter 3) (I was responsible for conceptualization, literature review, algorithm design, and writing of the original manuscript; Hengxu Li assisted with CAD plug-in development and model construction; Xinming Li was responsible for reviewing the manuscript and improving the expression; Yongsheng Ma and Carlos F. Lange were the principal investigators and supervisors on this research)

2. Tianyu Zhou, Hengxu Li, Carlos F. Lange, Yongsheng Ma. "A Feature-based Automatic Model Construction Method in CAD for Material Distribution Structures". *CAD Conference and Exhibition*, 2023, pp. 200-205. (Chapter 4) (I was responsible for the conceptualization, literature review, optimization algorithm design, and writing of the original manuscript; Hengxu Li assisted in constructing CAD models, Yongsheng Ma and Carlos F. Lange were the principal investigators and supervisors on this research)

3. Tianyu Zhou, Hengxu Li, Carlos F. Lange, Yongsheng Ma. "Feature-based Modeling for Realizing Parametric Design Automation of Material Distribution Structures in CAD Systems*"*. (Manuscript is being prepared for final submission) (Chapter 5). (I was responsible for the conceptualization, literature review, optimization algorithm design, postprocessing the models, and writing of the original manuscript; Hengxu Li assisted with user interface design in CAD

systems and CAD model construction; Yongsheng Ma and Carlos F. Lange were the principal investigators and supervisors on this research)

Acknowledgments

First, I would sincerely thank my supervisors, Dr. Yongsheng Ma and Dr. Carlos F. Lange, for their kind supervision and encouragement. Without their guidance and help, I could not find the way to success of this study. Also, I would like to thank my supervisory committee member, Dr. Rafiq Ahmad, who has provided valued feedback and constructive suggestions to improve my research work.

I would like to express my gratitude to acknowledge Siemens for providing student license of NX, Southern University of Technology (Shenzhen, China) and the University of Alberta for academic collaboration support, and NSERC Discovery Grant from Prof. Ma's research funding (RGPIN-2020-03956) for the financial support.

My thanks also go to my colleagues from Dr. Ma's group and Dr. Lange's CFD lab, who contributed to a friendly research environment and provided collaboration and inspiration. They also enriched my life in Canada.

Last but not least, I appreciate the support from my family and friends. They helped me pass through many tough times. Without their encouragement, I would not have completed this journey.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AM	Additive manufacturing
BLFs	Buckling load factors
CAD	Computer aided design
CAE	Computer aided engineering
CAM	Computer aided manufacturing
CSG	Constructive solid geometry
EBHM	Energy based homogenization method
FEA	Finite element analysis
HASM	Hybrid additive-subtractive manufacturing
IFS	Iterated function system
IGM	Iterative generation method
LSTO	Lattice structure topology optimization
MMA	Method of moving asymptotes
NURBS	Non-uniform rational basis spline
OC	Optimality criteria
SIMP	Solid isotropic material with penalization

List of Symbols

$(\boldsymbol{X},d)$	A metric space with metric d
D_m	Machine resolution
D_p	The shortest dimension of the minimum bounding box of the fractal primitive
F	A family of functions
$f_1, f_2, \ldots, f_n$	Functions
floor( )	The floor function to round off the value to the nearest small integer
G	A family of fractal iterative generation functions
$g_1, g_2, \ldots, g_n$	Iterative generation functions
N	A natural number
N_{max}	Maximum iteration number of a CAD fractal model
$\boldsymbol{R}$	Real number
$\boldsymbol{S}$	A set
$\boldsymbol{X}$	A set in $\boldsymbol{R}^n$
$x_1, x_2, \ldots, x_m$	Extracted design parameters
ε, E	Fractal scaling factor and its reciprocal, $E=1/\varepsilon$

Chapter 1 Introduction

Chapter 1 provides an overview of this thesis research. This chapter contains five subsections: subsection 1.1 briefly introduces the research background and motivation, subsection 1.2 provides the scope of this research, subsection 1.3 presents the research objectives, subsection 1.4 provides methodologies proposed or applied in this research, and subsection 1.5 shows the outline of this thesis.

1.1 Background and motivation

With the rapid development of advanced manufacturing techniques, such as additive manufacturing [1,2] and hybrid additive-subtractive manufacturing [3,4], it is possible to manufacture complex structures with lightweight and high-performance designs. Integrated with advanced manufacturing technologies, structural design and optimization have been widely applied in research and industry, which are the processes of finding the optimal design of a structure to meet functional performance requirements while considering various constraints. Most research of structural design and optimization focuses on how to get the optimal design or most efficient material distribution of a structure, and the results of the design/optimization usually have complex geometry characteristics [5,6]. While the research of structural design and optimization are progressing fast with great achievements in the recent 15 years, developing an efficient model construction method for the relevant complex geometry objects is in great demand in research and industry.

In the actual product development processes, post-processing operations, such as surface smoothing, edge chamfering, and assembling with other components are needed to further modify the models to make a practical and manufacturable design scheme [7]. In addition, the models need

to be reused in different computer-aided platforms, such as CAE and CAM systems, for product performance simulation and manufacturing analysis [8,9], and design changes may be further required for these geometry objects after CAE and CAM analysis [10]. Therefore, during the product development processes of functional structural design and optimization, the models of these geometry objects are expected to have good compatibility with different computer-aided systems, and flexible model modifications are expected to be realized. A parametric model, especially a 3D solid model is needed. However, the research on efficiently constructing a CAD model for the relevant complex geometry objects was not a major focus and there are still a lot of challenges for realizing the automatic model construction processes for complex geometry objects. Most of the complex geometry CAD models from the results of innovative structural design/optimization are still constructed manually in current CAD systems. The manual CAD model construction processes for these structures are repetitive, tedious, and time-consuming. In many cases, the model construction processes even take much longer time than the structural design and optimization process.

An issue arises on how to efficiently model these complex geometry objects in CAD systems parametrically because there are gaps between parametric modeling and design automation. For parametric design or parametric modeling in CAD systems, the model is defined with sufficient and non-redundant parameters which are editable and may properly reflect the design intent [11,12]. The design is flexible with sufficient editable parameters and the constructed models can be easily modified. However, the model construction processes may be tedious and time-consuming. For design automation, the aim is to efficiently represent a model and realize an iterative update of the model during the generative design process[13]. The models are usually simplified or non-parametric computer graphic models. The design parameters of the models are

massively reduced, which may cause the constructed model hard to be further modified after getting the design resolution from the iterative design loop because of very few remaining editable parameters. As for the model types, there are two mainstreams of representation for a geometric object, a CAD model or a tessellation model [14,15]. The former is good for representing a solid geometry that is compatible with CAD operations, complex numerical simulations, and digital manufacturing while the latter is a model created by properly dividing datasets of polygons in computer graphics which is good for complex geometry visualization. In the product development processes, there should be a proper balance between model construction efficiency and design flexibility. The models used in product development processes may be complex in geometry, but a good CAD model is still preferable compared with computer graphics models.

In this research, automatic model construction methods for building parametric CAD models with complex geometry features were proposed in a conceptual framework, developed in a CAD environment, and applied in several design and engineering cases. The aim is to realize parametric design automation for complex geometry objects in CAD systems. To be more specific, the model construction processes of these objects are expected to be automatic to improve the modeling efficiency, and the models constructed with the proposed methods are expected to be parametric CAD models with sufficient parameters for flexible design changes.

1.2 Research scope

Figure 1.1 shows the scope of this proposed research, namely parametric design automation, within product development process. Parametric design automation covers the stages of product function design, product geometry design, product performance analysis, and product manufacturing in the product development process. In the product function design stage, parametric design automation can build a mapping between product functionality and design

parameters. Different values of design parameters lead to different geometric shapes, which can lead to different product functions. In the product geometry design stage, parametric design automation can realize automatic model construction of parametric product models in CAD systems. The model construction process is automatic, which can significantly improve the modeling efficiency. The constructed models are parametric CAD models, which have good editability for further modifications, and the models are compatible with different computer-aided systems. In the product performance analysis stage, parametric design automation can realize automatic design updates for design optimization based on numerical simulation results. The simulation result can be accurate and the design updates are efficient. In the product manufacturing stage, parametric design automation can realize a lightweight design and improve product performance. The digital product 3D models can be directly used in both additive manufacturing (AM) and subtractive manufacturing (SM) for process simulation and tool-path generation.

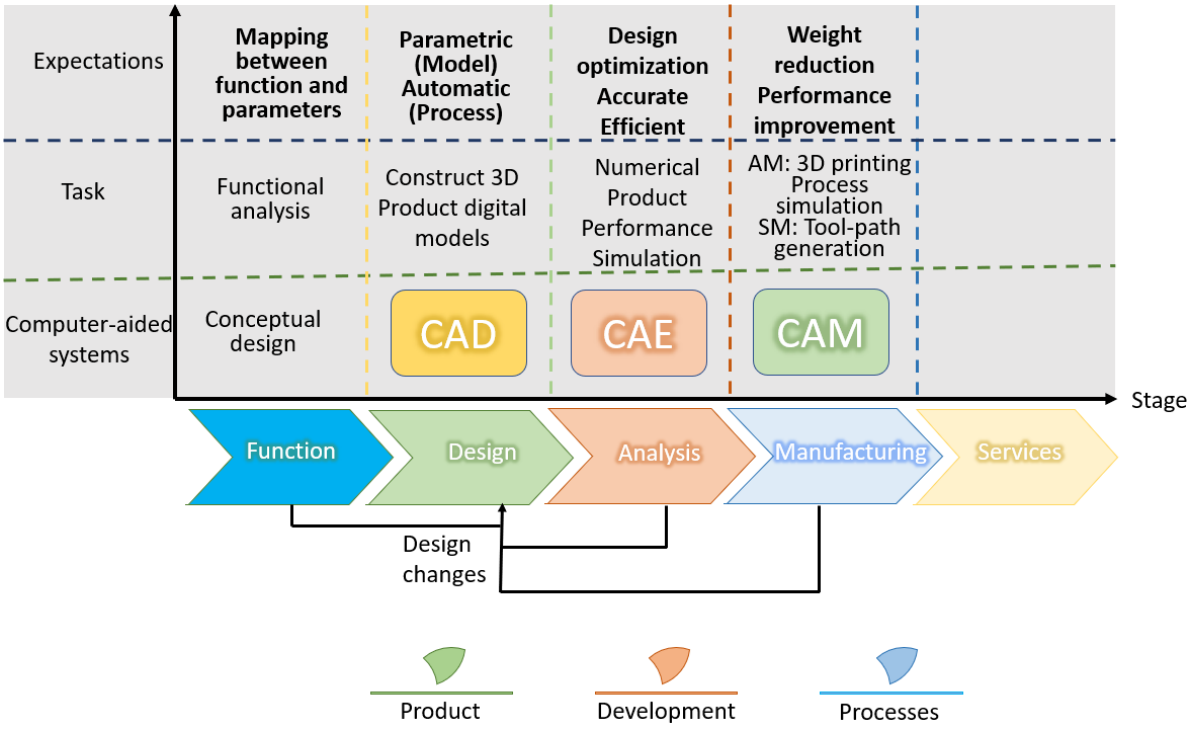


Figure 1.1 Scope of this research in product development processes

There are many types of geometry objects whose parametric design automation in CAD systems has been realized or needs to be realized in research and industry. The set of complex geometry objects is an important category among them, because the manual model construction processes for complex geometry objects in CAD systems usually take a long time, and the modeling efficiency needs to be improved to shorten the product development time. The fractal objects and the material distribution structures are two typical types of complex geometry objects. They both have a lot of practical application value in research and industry, and their parametric design automation in CAD systems have potential to be realized. In this research, the fractal objects and material distribution structures were studied and their parametric design automation in CAD systems were realized. Figure 1.2 shows the scope of the research and the sets' relationship of two categories studied with respect to complex geometry objects and parametric design automation.

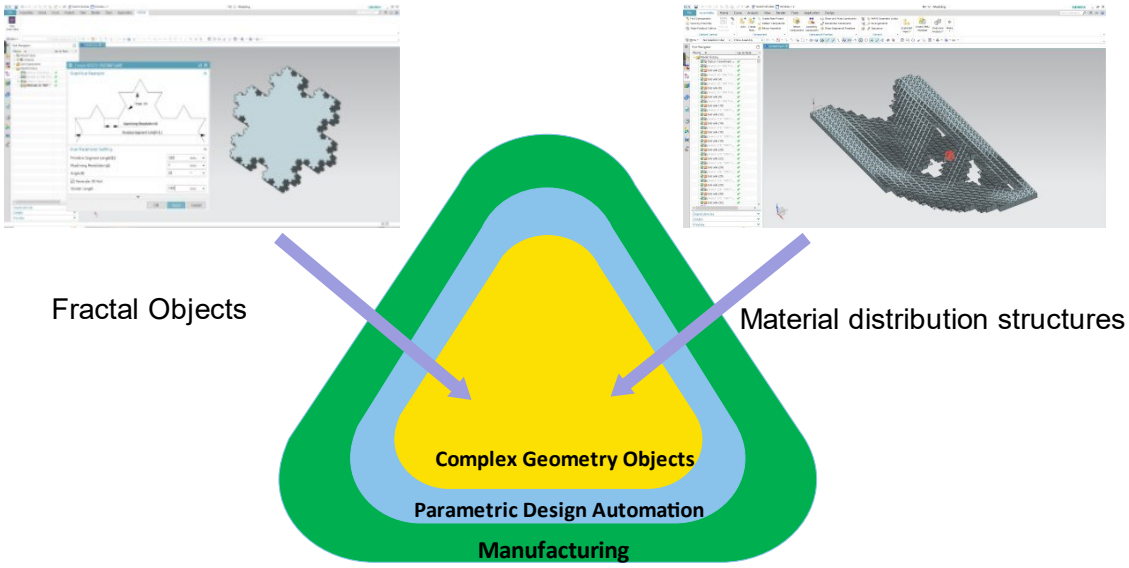


Figure 1.2 Scope of this research in geometry types: fractal objects (F), and material distribution structures (M)

1.3 Research objectives

The main objectives of this research is to develop methods to realize parametric design automation in CAD systems, especially for models with complex geometry features. Based on the above-mentioned scope, this research aims to realize parametric design automation for fractal geometry objects and material distribution structures. The research goals and the expected research outcomes are listed as follows:

(1) Develop a parametric modeling method for variable fractal geometry objects in a CAD system. Currently, fractal objects are mainly modeled in computer graphics for visual representation, the parameters of the fractal model are fixed, and fractal generators are statically constructed and non-editable. In contemporary CAD systems, the CAD models are not constructed iteratively and the fractal or fractal-like geometric objects are built manually in most cases. With the author's proposed method, fractal or fractal-like geometric objects are expected to be constructed automatically in the CAD system. The CAD fractal model is expected to be constructed iteratively and can be stored, represented, and edited in multilevel geometric complexities. Design parameters are expected to be extracted from the fractal model and fractal generators are expected to be constructed dynamically for each iteration cycle. As a result, the design parameters of the fractal or fractal-like objects can be variable and editable. The model is expected to be built as a feature-based CAD part model, which has rich information and is eligible for most CAD operations.

(2) Develop a method for automatic CAD model construction for material distribution structures. In structural design and optimization, material distribution structures are used to represent material allocation in space, and the final optimization result is an optimum arrangement of materials within a structural system to achieve the desired performance while minimizing

weight or satisfying other specific constraints. The global design domain of a material distribution structure is spatially divided into several subdomains and each subdomain acts as a design space for local material allocation. Currently, the data of the material distribution structures are represented in a 2D/3D matrix. The models of these structures are visually represented by tessellation models, such as STL format. The parametric CAD models of such structures are constructed manually. The proposed method aims to improve the CAD modeling efficiency of a complex material distribution structure with parameterized and spatial repetitive features of its microstructure geometry elements. The constructed CAD model is expected to be parametric, and every component microstructure is expected to be editable for its design parameters. The model should be well-compatible with a numerical simulation environment and further geometry modifications are easy to implement.

1.4 Research methodologies

To achieve the realization of the proposed work, this research proposed or adopted the methodologies as below:

(1) Feature-based modeling method. Feature modeling approaches have been developed for more than 40 years and the feature concept has been developed into different categories according to modeling domain perspective and engineering intents [16]. Feature technology has been applied in many areas related to computer-aided systems, however, its application in modeling fractal objects and material distribution structures in CAD systems are in relative infancy. In this research, the application of feature technology is to build associative relationships among component geometry elements for a complex model to realize parametric design automation in CAD systems. New feature types were proposed and applied in the automatic CAD model construction processes for fractal objects and material distribution structures.

(2) Iterative generation method (IGM). In fractal modeling, The Iterated Function System (IFS) is the most commonly used deterministic approach to generate or approximate a mathematical fractal object. An IFS is a family of specified contraction mappings that map a whole object onto the parts, unionize all the parts and iteration of these mappings will result in convergence to an invariant set [17]. However, traditional IFS has some limitations in building a fractal model in CAD systems. The proposed IGM is a built-in method of a CAD feature to iteratively generate a fractal model with editable parameters in CAD system. It is an extension of traditional IFS. As the IGM was a proposed method in this research, the detailed instruction of IFS and IGM will be presented in Chapter 3.

(3) Topology optimization. Topology optimization is a computational method that aims to find the optimal material layout for a given structure subject to certain design constraints [18]. The goal is to find the most efficient and effective design by redistributing material in the structure to reduce weight, increase stiffness, or minimize stress under various loading conditions. Topology optimization has applications in various fields, including aerospace, automotive, and mechanical engineering. It has the potential to improve the performance of existing designs while reducing material and manufacturing costs. The process of topology optimization involves defining the design space, loads, and constraints, and then using mathematical algorithms to iteratively remove and/or redistribute material from areas of low stress or strain energy while updating the structural design. The result is a structure with an optimized material distribution that meets the specified structural performance requirements and satisfies all the constraints. The optimization output of topology optimized structure is usually a 2D or 3D matrix representing the material distribution spatially for the global macrostructure. In this research, topology optimization was applied to

optimize structures, and the optimization results provided input data of material distribution structures for their automatic CAD model construction in my case studies.

1.5 Thesis outline

Chapter 1 provides a brief overview of this research. The background of parametric design automation is introduced. The current gaps and motivation of this research are presented. The scope of this research is provided, which is fractal objects and material distribution structures. The objectives of the research are shown, and the proposed and/or applied methodologies are briefly introduced.

In Chapter 2, a brief literature review of related research is introduced. Including fractal geometry, material distribution structures, contemporary CAD systems, feature-based modeling method, and topology optimization.

In Chapter 3, the research of modeling fractal geometry objects in CAD systems is presented. A new feature type, CAD fractal feature, is proposed, and the relevant built-in method, the iterative generation method is proposed and realized. An extruded Koch Snowflake CAD model with parametric geometric variants was built as a case study for demonstration of the proposed modeling method.

In Chapter 4, the research of an automatic CAD model construction method for material distribution structures is introduced. In the proposed method, there are three sub-steps to realize the automatic CAD model construction, which are shown in detail. A parametric CAD model of a topology-optimized cantilever was automatically constructed in CAD system with my proposed method in the case study. My method was proven to be of sufficient efficiency for CAD model construction.

In Chapter 5, the parametric design automation of material distribution structures in CAD systems is realized with feature-based modeling method. A new type of feature, named material distribution feature is proposed and applied to construct the parametric CAD model automatically. In addition, automatic CAD model construction of more complicated material distribution structures, such as multi-scale topology optimized structures are studied in this chapter. A case study of automatic model construction of a multiscale topology-optimized Michell structure with muti-variable lattices is presented, which proved the proposed method is of sufficient flexibility to model complicated material distribution structures.

In Chapter 6, the research is concluded, contributions are summarized, and the limitations and future work of the research are pointed out.

Chapter 2 Literature Review

Chapter 2 first provides an introduction to fractal geometry and material distribution structures, which are the scope of this research. Next, the chapter reviews contemporary CAD systems, which is the environment to realize the proposed parametric design automation. Then, the adopted modeling method in this research is introduced, which is feature-based modeling method. Finally, topology optimization is briefly introduced, which was the method applied in this research to get input data of material distribution structures.

2.1 Fractal geometry

Fractal geometry is an extension of classical geometry and has been employed for modeling geometrical objects which may not be easily described or efficiently modeled by Euclidean geometry. In the observation of "Geometry of Nature" [19], such as clouds, coastlines, and some biological structures, self-similarity exists widely when comparing the local geometric feature to the global shape (see Figure 2.1 as a demonstration). In the past few decades, the mathematical background, algorithm-based construction methods and the applications of fractal geometry and fractal analysis in some specific areas have been developing fast. In this subsection, a brief literature review of the mathematical description, construction methods, and industrial applications of fractal geometry is introduced related to this research.

2.1.1 Mathematical description of fractal geometry

The mathematical definition of fractal geometry has been discussed and developed by mathematicians and scientists for decades. The acknowledged founder mathematician, Benoit B. Mandelbrot, regarded fractals as sets with non-integral Hausdorff dimensions and such sets have the additional property of being self-similar in some sense, either strictly or statistically [14]. Then,

researchers have made great fundamental contributions in different fractal geometry areas such as the recognition of measure theory, iteration theory, dynamic systems [20,21]. Various attempts have been made to give a mathematical definition of a fractal, but such definitions have not proved satisfactory in a general context [22]. This research mainly focuses on how to efficiently and conveniently model fractal geometry objects in CAD system. Instead of trying to give a precise definition of fractal geometry, descriptive features of a fractal are provided to the best of the author's knowledge. A set $\boldsymbol{S}$ is regarded as a fractal if it has all or most of the following features [22–25]:

- $\boldsymbol{S}$ has a fine structure that is irregular in detail at arbitrarily small scales.
- $\boldsymbol{S}$ is too irregular to be described by traditional geometry or a set of solutions of any simple equation, either locally or globally.
- $\boldsymbol{S}$ often has self-similarity or self-affinity to some extent, which can be either strictly or in a statistical or approximate sense.
- $\boldsymbol{S}$ usually has its 'fractal dimension' (defined and measured in some ways) strictly greater than its topological dimension.
- $\boldsymbol{S}$ in many cases can be defined in a straightforward rule and can be obtained by recursive procedure, although it has an intricately detailed structure.
- $\boldsymbol{S}$ is in some ways quite a large set (it is uncountably infinite) and its size is not quantified by the usual measures such as length.
- $\boldsymbol{S}$ often has a 'natural' appearance.

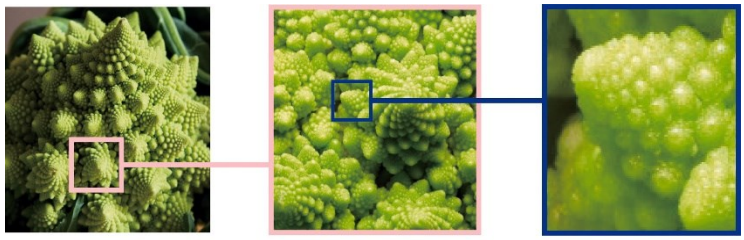


Figure 2.1 Self-similar geometry of Roman cauliflower [26].

In this research, the fractal object is defined as a geometric set, where each of its members satisfies the following geometric characteristic: the local shape has geometric similarity to some extent with the global shape.

2.1.2 Construction methods of fractal geometry

There are many methods to construct fractals. In computer graphics, typical methods include recursive method [19,27], Lindenmayer System (L-System) [28–30], and iterated function system (IFS) [21,31,32]. The recursion method is straightforward, and the recursive relation can be directly derived from the self-similar feature of a fractal. However, this method is difficult to manipulate in CAD system because the initiator is simply a set of simple geometric objects, such as edges and cubes; it is hard to recursively build CAD features or expressions and the direction of the fractal generation cannot be controlled explicitly. In the L-system method, the construction of a fractal is assisted with a turtle graphics interpretation [28]. The turtle's state is represented by its current coordinates and the angle that the turtle is facing. The method is not a good candidate for generating a fractal in CAD system because the current direction or orientation with respect to the global coordinate system in CAD environment has to be constantly calculated. The IFS is a good candidate for modeling fractal geometry in CAD system, where a set of affine transformations can be derived once after the initiator and the generator are determined. Then the IFS is determined, and the fractal is constructed by iterated steps (see Figure 2.3 for a

demonstration example). In IFS, the fractal is treated as a geometric structure set which will be copied and iterated with the transformations for higher-level generation. Thus, there is no need to find any orientation in each iterative calculation in this method [33]. Due to the above advantages of IFS, this method has been widely used and has become the most popular method in fractal modeling. However, traditional IFS still has some limitations for modeling variable fractal models parametrically. First, IFS is a function-based method, which is not object-oriented. The encapsulation, reusability and extensibility of IFS are restricted. Second, the iterated functions in a traditional IFS are usually fixed during the iterative construction process of a fractal model, which lacks design flexibility if the required model is not strictly fractal and design parameters need to be edited and may take different values for different iterations. To overcome these limitations of a traditional IFS, the iterative generation method (IGM) is proposed in this research, where variable design parameters are extracted and explicitly expressed as inputs of iterative generation functions and the geometry generators can be created/deleted/edited dynamically in different iteration cycles. The IGM is an extension, based on IFS, that has a higher level of encapsulation and is applied as a built-in fractal construction method in a CAD fractal feature. The theory of IFS and the proposed IGM will be explained in detail in Chapter 3.

2.1.3 Applications of fractal geometry in industry

Apart from a variety of applications in mathematics and physics, fractal geometry and fractal analysis have been widely applied in industry. Fractal concepts, fractal dimension and fractal analysis are useful tools for architectural and design criticism [25]. In architectural design, many architectural works are fractal-like, such as Gothic buildings [34], and some quantitative methods, such as the box-counting method [35] have been widely applied to analyze the fractal aspects of architecture. In surface evaluation, such as in class industry [36], fractal dimension in

combination with standard roughness parameters are applied for complex surface analysis [37]. In antenna industry, fractal concept has been applied to design antenna elements or antenna arrays in order to maximize the effective length or increase the perimeter of material that can receive or transmit electromagnetic radiation within a given total surface area or volume [38–41]. In energy storage and energy release, it is generally accepted that fractal geometry has the intrinsic advantages of minimized flow resistance and strong heat transfer capability [42,43], and fractal structures have been successfully applied to the design of highly efficient heat exchangers [44–46] or thermal energy storage units [47] (see Figure 2.2). Similarly, fractal geometry has been applied in micromixers [48,49], sensors [50,51] and drug delivery systems [52] for its potential to increase the specific contact area to improve the performances of these devices. In signal processing fields, such as image analysis and speech synthesis, fractal tools have been applied for studying complex signals and sometimes fractal approach provides new means to solve specific problems in signal processing, which might be with greater success than classical methods [53]. Also, from the engineering management point of view, fractal geometry in conjunction with statistics can be used as a useful and powerful tool for an explicit, objective and automatic description of production process data, such as structures (e.g., defects, surfaces, cracks, time series from dynamic processes), which are too complex and irregular to be described by conventional methods [54].

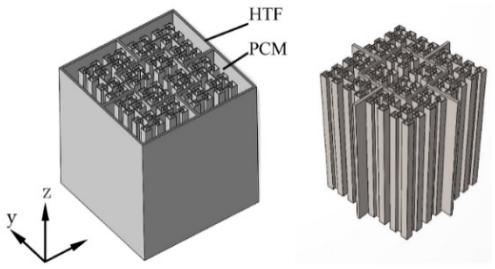


Figure 2.2 An application of fractal geometry in industry: Thermal energy storage unit with fractal net fins. HTF: heat transfer fluid; PCM: phase change material [47].

From the above applications, the author would like to point out that the fractal dimension is an important parameter for fractal analysis in industry, which can be calculated or measured by many methods such as the log-log method, and the box-counting method [55]. From the definition point of view, the relationship among the number of fractal pieces *Num* of an object, the scaling factor $\boldsymbol{\varepsilon}$, and the fractal dimension *Dim* satisfy the following equation:

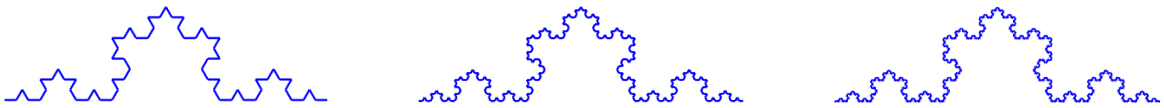
$$Num = \varepsilon^{-Dim} \tag{2.1}$$

Then, the fractal dimension *Dim* can be directly calculated as:

$$Dim = -\log_{\varepsilon} Num = -\frac{\log Num}{\log \varepsilon} \tag{2.1}$$

Take the Koch curve [56] (see Figure 2.3), for example, *Num*=4, $\boldsymbol{\varepsilon}=\frac{1}{3}$. Then the fractal dimension of Koch curve is $Dim=\frac{\log 4}{\log 3}\approx 1.2619$

(a) Step 0 (b) Step 1 (c) Step 2



(d) Step 3 (e) Step 4 (f) Step 5

Figure 2.3 Iterative construction of Koch Curve.

2.2 Material distribution structures

Material distribution structures are widely used in research and industry for structural design and optimization. In this subsection, a brief introduction to material distribution structures is presented, and the data representation method and current modeling method for material distribution structures are provided.

2.2.1 A brief introduction to material distribution structures

With the rapid development of advanced manufacturing technologies, especially for additive manufacturing, the fabrication of complex structures with enhanced functional properties has been realized. As a result, a wide exploration of the design space for functional material/structure has been enabled in research and engineering applications. Several examples below show innovative structural designs due to the benefits of emerging advanced manufacturing technologies. The design of cellular materials has recently undergone a paradigm shift, where cellular materials are no longer limited to traditional shapes such as honeycomb panels or stochastic foams [57]. The lattice structures can be designed with hybrid lattice types [58], and each type of lattice microstructure can have non-uniform and non-fixed geometries with multi-variable parameters [59,60]. The manufacturing of many fractal geometries becomes feasible, and

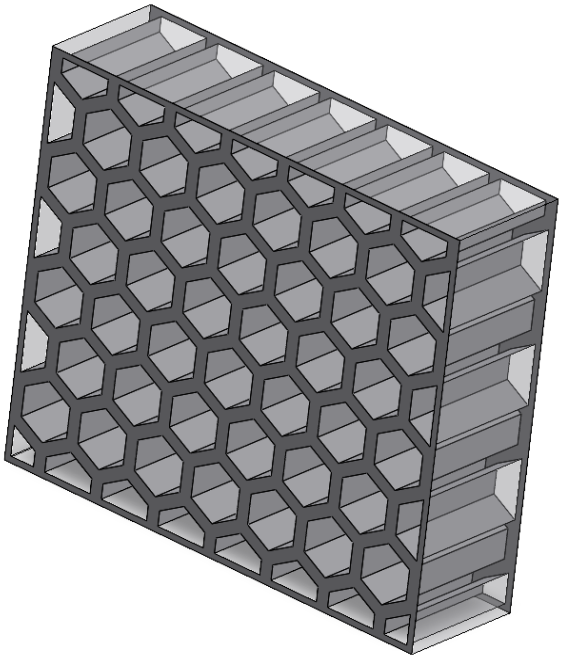
fractal structures can be produced with functional properties such as shock absorption [61]. Many structural optimization methods, such as size optimization [62,63], shape optimization [64,65] and topology optimization [66,67], can now have more design freedom because many manufacturability issues for the optimized structures have been resolved.

The above examples of innovative structural design can be directly integrated with advanced manufacturing technologies. Essentially, they are all aiming to find a feasible solution for material allocation or arrangement in space to achieve the desired performance of a structure. The structures in these examples can all be represented by spatial material distribution in discretized design domains, which are categorized as material distribution structures. In this research, the material distribution structure refers to a structure whose design domain is spatially divided into several subdomains and each subdomain acts as a design space for local material allocation. Figure 2.4 shows some examples of material distribution structures. In these material distribution structures, the design domains are called global macrostructures, which are discretized elementally, and each local element is a design space, which is called a local microstructure.

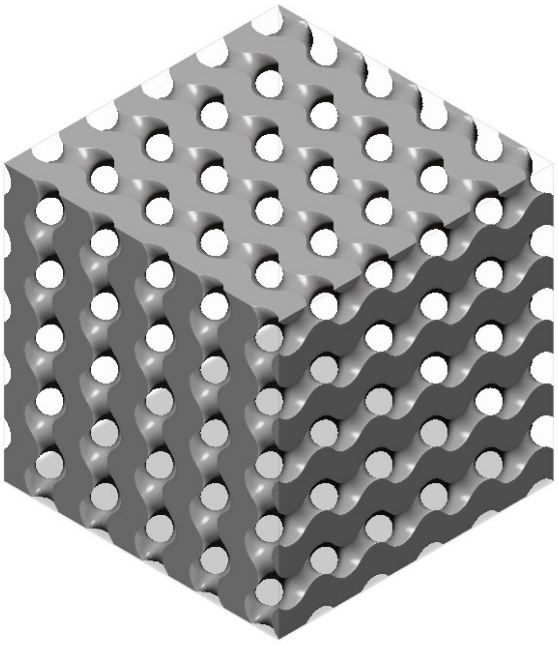
2.2.2 Current data representation method of material distribution structures

In current data representation methods of a material distribution structure, the matrix representation is mainstream, where a 2D or 3D matrix is used to represent material spatial distribution as the physical density field. In this research, it is called material distribution matrix. For each element of the matrix, its indices of row and column represent the relevant position of the local microstructure element with respect to the global macrostructure, and its value determines whether or not the local microstructure should be allocated with elemental infill patterns. Figure 2.4 shows an example of a material distribution structure and its data representation. In this

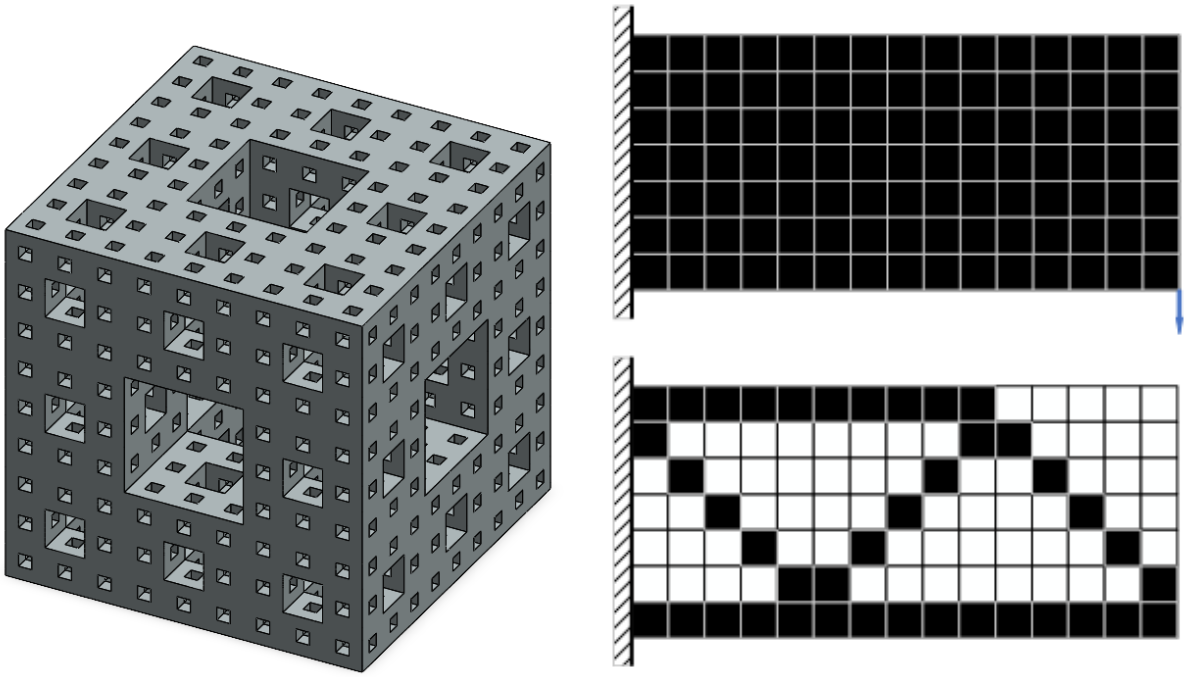
example, the model is a fractal structure named Sierpinski Carpet [68]. Figure 2.4 (b) is the binary matrix representation of its 1st iteration model (see Figure 2.4 (c)). Solid squares are used as infill for local microstructure elements. Each matrix element can take a value of either 0 or 1, where 0 means the local region should be left void and 1 means the region should be filled with a solid square.



(a)



(b)



(c) (d)

Figure 2.4 Examples of material distribution structures: (a) Honeycomb cellular structure, (b) Gyroid lattice structure, (c) Menger Sponge fractal structure, and (d) Topology optimized beam structure

(a)

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(b)

(c)

Figure 2.5 An example of a material distribution structure and its data representation: (a) The models of Sierpinski Carpet within 4 levels of iteration, (b) The binary matrix representation of the 1st iteration model, and (c) The 1st iteration model.

The matrix representation method of a material distribution structure is concise, and convenient to be initialized and updated. In addition, the model visualization from the data matrix is easy to be realized. Here, let us take topology optimization as an example. Before optimization,

the physical density field of the original design domain can be conveniently initialized by assigning a constant uniform value to all elements of its material distribution matrix[69]. During the iterative optimization loop, the elements' values of the matrix can be used to interpolate the mechanical properties of each local element [70], such as elemental elastic modulus E_e, for finite element analysis, and the matrix can be updated algorithmically for each iteration [71,72]. When the optimization loop ends, the final material distribution matrix represents the optimization result for the optimized structure. Software with matrix visualization functionality, such as Matlab, can be used to display the final structure by plotting its data matrix in a marching cubes based algorithm [69,73]. An STL writer can be applied to post-process the data matrix [69,74], and finally the structure model is created in STL format [75,76], which is friendly to additive manufacturing systems.

The above advantages of matrix representation make it become the mainstream representation method for a material distribution structure. During the initial structural design and optimization stage, the matrix representation method is a good option. However, during the model construction stage, the matrix representation method is not sufficient. First, as mentioned in the introduction chapter (Chapter 1), a parametric CAD model, rather than a matrix plot model is preferred in the structural development process. In current CAD systems, however, the material distribution matrix cannot be directly read or imported to guide CAD model construction for the relevant material distribution structure. Second, with the rapid development of advanced manufacturing technologies, design and optimization for material distribution structures may have more freedom, and the matrix representation method may not be flexible enough to represent innovative structural designs. For example, the subdomains of a material distribution structure may be non-equally divided, the local microstructure elements may have hybrid types of infill patterns,

and the elemental infill patterns may be non-uniform with unique values of design parameters for each specific local microstructure element. It is obvious that the matrix representation is not sufficient to cope with the above challenges. A new data format is needed to represent material distribution structures. The new data format is expected to be directly read by CAD systems as guidance for automatic CAD model construction, and it can be capable of including more information for each local microstructure element, such as types of infill patterns and relevant design parameters, to improve the flexibility to properly represent material distribution structures with more design freedom. In my proposed method, a new data format is proposed to represent material distribution structures in its model construction stage. A data processing program is designed, which can automatically process the original data from the material distribution matrix, transform the data into the new proposed data format, and store it in a CAD readable data file. The details for the proposed new data format and data processing program will be introduced in Chapter 4 and Chapter 5.

2.3 Contemporary CAD systems

Geometry modeling methods in CAD systems have been developing fast in the last few decades and great achievements have been made in constructing geometry efficiently and developing user-friendly operation interfaces. During the development of modern CAD systems, much attention has been paid to parametrically model a digital product and the communication/integration among different computer-aided product development systems, such as CAD/CAM/CAE communication/integration. In parametric modeling, the product model is defined, constrained, and represented by extracted geometry parameters, where the geometric parameters are carefully selected to make sure that the model is fully defined but not over-constrained or has ambiguity issues. Parameters should be defined in a user-friendly manner and

should be rollback editable for realizing rapid design modifications. Solid modeling method and surface modeling method are two mainstreams of geometry construction for contemporary CAD systems. In the integration/communication among different computer-aided systems, change propagation and information consistency should be carefully managed. Association and interoperability among different systems are key factors to shorten product development cycle time. In this subsection, a literature review of geometry construction methods in contemporary CAD systems, and a summary of fractal modeling research and research of modeling material distribution structures in CAD systems will be introduced.

2.3.1 Methods of geometry construction in contemporary CAD systems

Contemporary CAD systems nowadays are either based on solid modeling method or the surface modeling method for geometry construction [77]. In solid modeling, constructive solid geometry (CSG) has become the mainstream method of 3D model representation, where the CSG object is built from standard primitives, using regularized Boolean operations and rigid motions [78]. The CSG standard primitives are the parallelepiped (block), the triangular prism, the sphere, the cylinder, the cone, and the torus, and shapes represented by these primitives must be instantiated by the user to specific dimensions. The surface modeling method is used as a convenient way to model free-form curves and surfaces, where the non-uniform rational basis spline (NURBS) is modeled in CAD system to represent most of the surfaces [79]. Free-from deformation with respect to surface modeling has been studied by many researchers and the deformation can be applied either globally or locally [80,81]. Based on these great achievements, contemporary CAD systems can model Euclidean geometry efficiently and conveniently. And such CAD models are good enough to be reused in other product development steps, such as performance-based CAE simulation and manufacturing-based tool path planning. However, no

commercial CAD systems are dedicated to the efficient construction and manipulation of fractal geometry objects. For example, the fragmentation feature of the fractal makes it impossible for the NURBS method in surface modeling to model a fractal curve and the CSG-tree data structure in solid modeling will hierarchically represent an object but the self-similarity in a fractal object cannot be modeled in this way [33]. For the material distribution structures, current CAD systems cannot directly use the optimization result of a material distribution structure to automatically construct a parametric 3D solid model for it. In most cases, the feature-based CAD models of these structures are still constructed manually. In most of current CAD model reconstruction processes for material distribution structures, matrix plot models or non-parametric mesh models are first imported to a CAD system, then trace drawing process is implemented manually by designers to define the sketch, and finally, parametric 3D models are constructed. The processes are repetitive, tedious, and time-consuming, and a reasonable parameterization of these CAD models for further modifications needs advanced product development knowledge to reflect the design intent.

2.3.2 Research on modeling fractal geometry in CAD systems

Many research attempts have been made on modeling fractals in a computer-aided system and most of the research efforts focused on physically fabricating fractal geometry objects, either in the way of rapid prototyping (RP) methods, such as layer manufacturing (LM) technology [82,83] and additive manufacturing [84–86], or using traditional subtractive manufacturing methods, such as CNC machining [87]. Fractal objects were classified into several different subtypes and the manufacturability of each fractal class was analyzed [82]. Specific data structures based on iterated function system (IFS) were created and related algorithms were designed for rapid prototyping of fractal geometry represented objects. For example, the radial-annular tree (RAT) structure was developed to represent fractal curves in a computational form [33,88] and the

radial-blossoming tree (RBT) was proposed and implemented to represent self-similar fractal solid [89,90]. These specific data structures for fractal representation can be applied in the RP, and related operations, such as traversal algorithms, were designed for efficient toolpath generation processes. These research outcomes have bridged some of the gaps among CAD, RP, and fractal geometry and they provide a promising future for manufacturing fractal design patterns physically.

However, modeling and representing fractals parametrically is still an open issue in CAD and the communication between CAD and RP systems with respect to fractal models is still limited. The abovementioned manufacturing-based fractal modeling research focuses on building and representing fractal objects in RP systems, where the fractal models were generated as voxel-based models or STL models in most cases [83,84,91]. Based on the authors' knowledge, the following limitations might happen to such models when communicating with CAD systems:

- Model transformation might be needed, which means some important information, such as specific fractal representation data structures and fractal generation algorithms, might be stripped off during the transformation.
- Some of the CAD feature operations, such as adding fillet, chamfer, assembly constraints or Boolean operations with other CAD geometric objects, might not be realizable.
- The model imported into the CAD system might no longer be parametrically defined and editable.

Based on these inherent limitations, the design flexibility for these models in the CAD system is restricted and efficient management of change propagation and information consistency is hard to realize when communicating/integrating with RP systems.

The above limitations suggest that maybe a research attempt should be made to parametrically model and represent the fractal objects directly in a widely used CAD system first, such as Solidworks, Siemens NX, Catia, etc. Then it would be more convenient to perform the

modifications of the fractal design model and communication/integration between CAD and RP systems. To the best of the authors' knowledge, the modeling of fractal geometry objects in widely used CAD systems is still done mostly manually. Fractal geometry objects may consist of a tremendous number of geometric entities as the level of iteration increases, so the manual design process of CAD sketch-drawing or 3D modeling of a detail-designed fractal object is very tedious and time-consuming (see Figure 2.3 the Koch Curve as an example; the number of vertices and line segments increases sharply as the iteration step goes up). However, geometric entities of a fractal object are not independent, and their relationships can be represented by an iterative algorithm. Thus, the design automation of fractal geometry modeling in a CAD system is studied in this research. This research aims to propose an object-oriented approach for modeling parametric variable fractal design in a widely used CAD system. The feature-based modeling method is applied in Chapter 4 to generate fractal-like or self-similar objects with finite iteration steps. A brief review of feature-based modeling is in subsection 2.4.

2.3.3 Research on modeling material distribution structures in CAD systems

There are two mainstream platforms for material distribution optimization, and they use different relevant methods to construct CAD models for material distribution structures after getting the optimization results.

The first one is based on commercial structural optimization platforms (e.g.: OptiStruct and solidThinking by Altair Engineering, TOSCA by Dassault Systems). In these platforms, the material distribution structure is represented by a non-parametric mesh model. The mesh model can be directly used in finite element analysis (FEA) of the optimization loop, and the commercial software has a good design of user interfaces (UI). The users do not need advanced structural

optimization knowledge to prepare for optimization. Instead, they only need to use UI to select their goals and constraints. However, the final structure usually has rough surfaces, and the symmetric characteristics of a structure cannot be kept. Figure 2.6 is an example produced by me that shows the non-parametric mesh/faceted model of an optimized material distribution structure of a cantilevered beam. The prismatic design domain is 60×30×2 mm, which is fully constrained in one end, and a unit-distributed vertical load is applied downwards on the middle edge of the other end (see Figure 2.6(a)). The objective is to get the least compliance design with a material reduction of 50 percent. After topology optimization in a commercial structural optimization platform, the result is represented as a non-parametric mesh/faceted model (see Figure 2.6 (b)). It can be seen from Figure 2.6 (b) that the final structure has rough surfaces, which is not an expected design scheme. Also, even if the initial optimization problem is fully symmetric (both initial geometry and boundary conditions are symmetric), the result might not be exactly symmetric. Apart from the above drawbacks, there are two more disadvantages of using commercial software for material distribution optimization. First, the research work behind commercial software is not available in the literature, and commercial software limits the optimization problem to only a few relatively mature topics, such as maximum stiffness design under the constraint of a certain percentage of material remaining. Advanced topics for structural optimization cannot be implemented, such as maximization of buckling load factors (BLFs) under certain constraints to improve the stability behavior of the structure. Second, the models of solution from the commercial software cannot be directly smoothed and parameterized. The gaps between non-parametric mesh models and parametric CAD models are still huge. Some research has been done on automatic CAD model reconstruction from the mesh model to a parametric model. The majority is based on

getting the derived skeleton of the mesh models [92]. The CAD model construction processes (see Figure 2.7) can be described as the following steps [93–95]:

1. Raw structural optimization results are derived into an optimized shape with smooth surfaces. In this step, geometry processing algorithms are applied to the non-parametric mesh model, such as mesh smoothing, mesh contraction, and remeshing.

2. From the smoothed mesh model, the skeleton of the 3D shape is automatically derived. There are many related algorithms for obtaining a skeleton, such as medial axis transform [96–98]. The derived skeleton can either be a surface skeleton or a curve skeleton.

3. The skeleton is normalized, and the cross section of the structure is calculated.

4. A CAD model is reconstructed based on the information of the normalized skeleton and the calculated cross section.

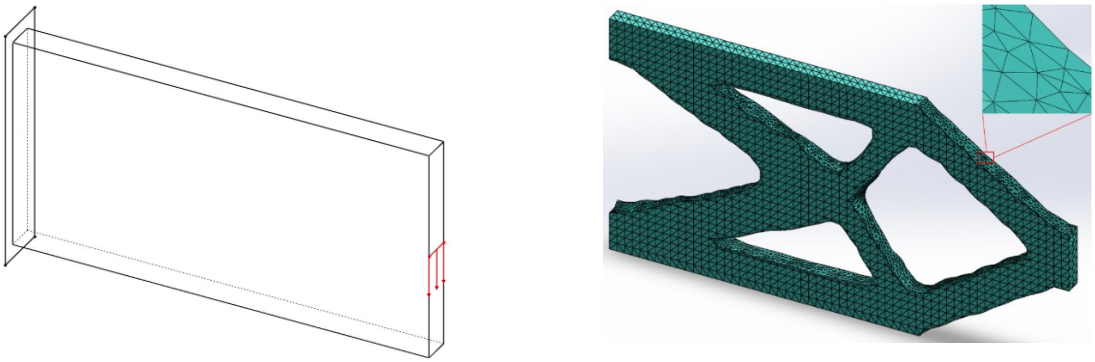
The above reconstruction processes of CAD models for material distribution structures can build a parametric 3D solid model. However, the limitations can be summarized below.

1. The users need to have advanced geometry processing knowledge to post-process the optimized structures for boundary smoothing and derivation of skeleton and cross section.

2. The methods are not applicable to all types of structures. Geometry is limited to beams or thin-walled structures.

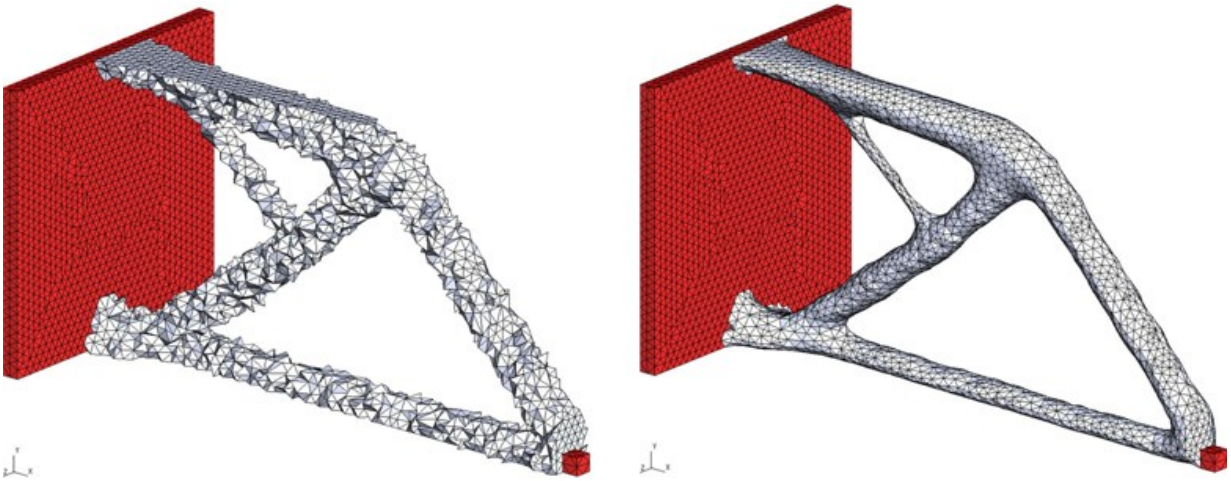
3. The skeleton normalization and calculation of cross section radius is based on the average of local region. Many design details may be stripped off.

4. The symmetric characteristics of the structure cannot be kept in both the optimization and the reconstructed CAD model.

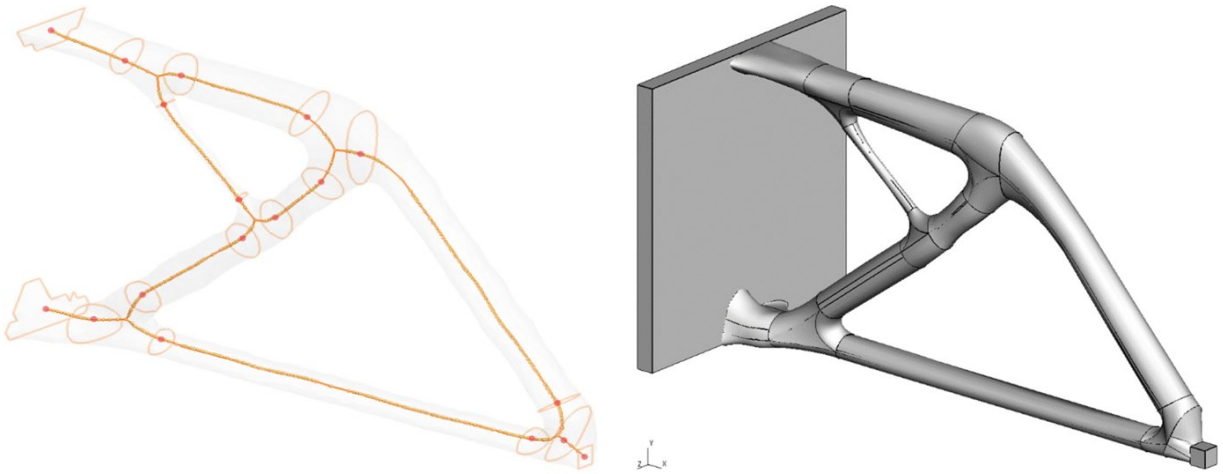


(a) (b)

Figure 2.6 An example of a non-parametric mesh model of an optimized material distribution structure of a cantilevered beam: (a) The design domain and boundary conditions, (b) The non-parametric mesh/faceted model (Optimization and final model were produced in the software Solidworks.)



(a) (b)



(c) (d)

Figure 2.7 A brief illustration of CAD model reconstruction processes for a material distribution structure from a mesh model (adapted from [94]): (a) Raw optimization result with rough surfaces, (b) Smoothed boundary triangulation of the optimized shape, (c) Skeleton extraction and cross-section calculation, and (d) CAD model reconstruction

The second platform for material distribution optimization is numerical computation software, such as Matlab. Specific optimization methods can be designed in detail in the numerical computation software and advanced topics of material distribution optimization can be explored. In this platform, the matrix representation is mainstream, where a 2D or 3D matrix is used to represent material spatial distribution as the physical density field (see Figure 2.5 as an example). As mentioned in subsection 2.2, the matrix representation is concise, but the data matrix cannot be directly read by CAD systems to guide the automatic CAD model construction. Some research [99–101] attempted to convert the matrix representation to a parametric CAD model. Translators were designed to process the matrix plotted models and spline curves were used to fit the boundaries of the structure. However, there are two major limitations of the translation methods, which are summarized below.

1. Most of the translators are image-based, which means the application is limited to 2D structures. For 3D shapes, most of the translators cannot be applied.

2. The control points and splines are used to fit the boundary curves, and the CAD models constructed are surface-based models with IGES format. The CAD models are not parametric 3D solid models, which means the further modifications of these structures may be difficult.

In summary, the current research results of relevant automatic CAD model construction methods for both platforms have limitations, and currently, there are no satisfactory methods for automatic CAD model construction. Therefore, the CAD model construction for material distribution structures is still done manually in most cases. In this research, a feature-based method is proposed to automatically construct CAD models for material distribution structures. The data of the matrix representation can be directly used, the models can be automatically constructed, and the constructed models are parametric 3D solid models with sufficient design parameters for flexible further modifications. A detailed introduction of the proposed method is in Chapter 4 and Chapter 5.

2.4 Feature-based modeling method

Feature modeling approaches have been developed for more than 40 years and the feature concept has been developed into different categories according to modeling domain perspective and engineering intents [16]. Different types of features, such as form feature, geometric feature, assembly feature, machining feature, CAE feature and functional feature have been developed and applied for their specific use in computer-aided systems [102,103]. Brunetti et al. [104] introduced a feature-based integrated product model that can capture product semantics in the conceptual

design phase and link design with part and assembly modeling with parametric modeling methodologies or parametric relationships. Ma et al [105,106] developed the associative feature concept, which is defined as a set of semantic relationships among product geometric entities, which may include assemblies, components, solids, faces, edges, vertices, surfaces, curves, points, vectors, datum references. Thereafter, many derived feature concepts or feature-based modeling methods were proposed and applied for their specific applications. Liu et al. proposed associative optimization feature for product design-optimization integration [107]. Ren et al. proposed process optimization feature for industrial processes optimization [108]. Li et al. applied [70] feature-based modeling method for CAD/CFD integration [109,110]. Feature technology has also been proven to be a very useful tool for complex system integration in a collaborative engineering environment, since it can act as a powerful tool to control information flow, manage change propagation, and maintain information consistency [111–116].

Although feature concepts or feature-based modeling methods are developing rapidly, understanding the essence of feature or feature-based modeling is still a challenge for engineers and researchers. Thus, instead of reviewing the evolution of feature modeling in detail or introducing the broadening application scopes of this approach, the authors would like to introduce the ontology description of feature and feature-based modeling approach: feature is defined as an object class with rich and associated properties of product and process engineering from both geometric and nongeometric aspects [117] and the feature-based modeling method is an object-oriented approach to model physical or abstract entities among different stages of product lifecycle management. The key advantages of feature-based modeling method are:

- Good for information encapsulation such as intent-based design [118].

- Friendly to information abstraction, reuse, and inheritance with respect to different types of features.
- A useful parametric modeling tool for parts, assemblies, optimization processes, etc.
- A powerful method to represent associative relationships, realize integration in a collaborative engineering environment and control/manage information flow among different product development stages.
- Provides a unified templated framework for systematic design and modeling.

Based on the above advantages, feature technology has been applied in many areas related to system integration, however, its application in modeling fractal objects and material distribution structures is in relative infancy. In this research, the application of feature technology is to build associative relationships among component geometry elements for a complex model to realize parametric design automation in CAD systems. New feature types were proposed and applied in the automatic CAD model construction processes for fractal objects and material distribution structures.

2.5 Topology optimization

In material distribution structure optimization, computer algorithms are used to determine the most efficient arrangement of materials for a given set of design and performance requirements. This approach involves removing material from non-critical areas of the structure and redistributing it to more critical areas, resulting in an optimal material distribution. Material distribution structure optimization is commonly used in engineering design, particularly in industries where weight reduction is critical, such as aerospace, automotive, and structural

engineering. By optimizing the material distribution, designers can create structures that are stronger and more durable, while minimizing their weight and cost.

In material distribution structure optimization, there are three typical methods (see Figure 2.8), including size optimization [62,63], shape optimization [64,65] and topology optimization [119,120]. The design pattern or topology of the structure is known for both size optimization and shape optimization. In size optimization, the sizes of the components which make up the structure are optimized. In shape optimization, the contour of some part of the boundary of a structural domain is optimized. Topology optimization is the most general structural optimization method, which enables the creation, merging, and splitting of the interior solids and voids during the optimization processes. Compared with size optimization and shape optimization, topology optimization has a larger design space, and superior structural performance is expected. In this research, material distribution structures based on topology optimization are focused.

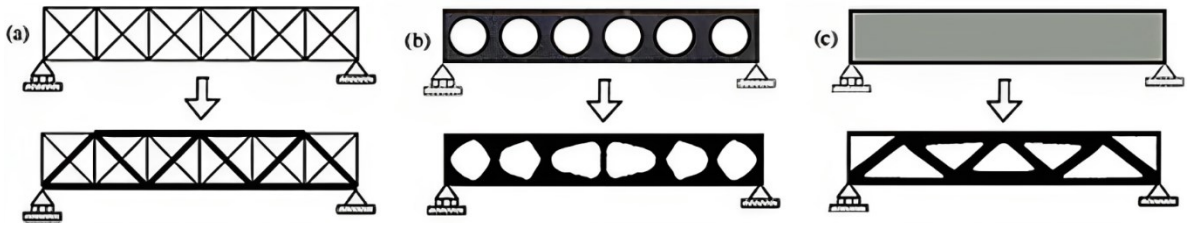


Figure 2.8 Three different structural optimization methods: (a) size optimization, (b) shape optimization, and (c) topology optimization

Topology optimization is a computational method that aims to find the optimal material layout for a given structure subject to certain design constraints. The goal is to find the most efficient and effective design by redistributing material in the structure to reduce weight, increase stiffness, or minimize stress under various loading conditions. Topology optimization has applications in various fields, including aerospace, automotive, and mechanical engineering. It has

the potential to improve the performance of existing designs while reducing material and manufacturing costs.

The process of topology optimization (see Figure 2.9 top as a demonstration) involves defining the design space, loads, and constraints, and then using mathematical algorithms to iteratively remove and/or redistribute material from areas of low stress or strain energy while updating the structural design. The result is a structure with an optimized material distribution that meets the specified structural performance requirements and satisfies all the constraints. The optimization output of topology optimized structure is usually a 2D or 3D matrix representing the material distribution spatially for the global macrostructure.

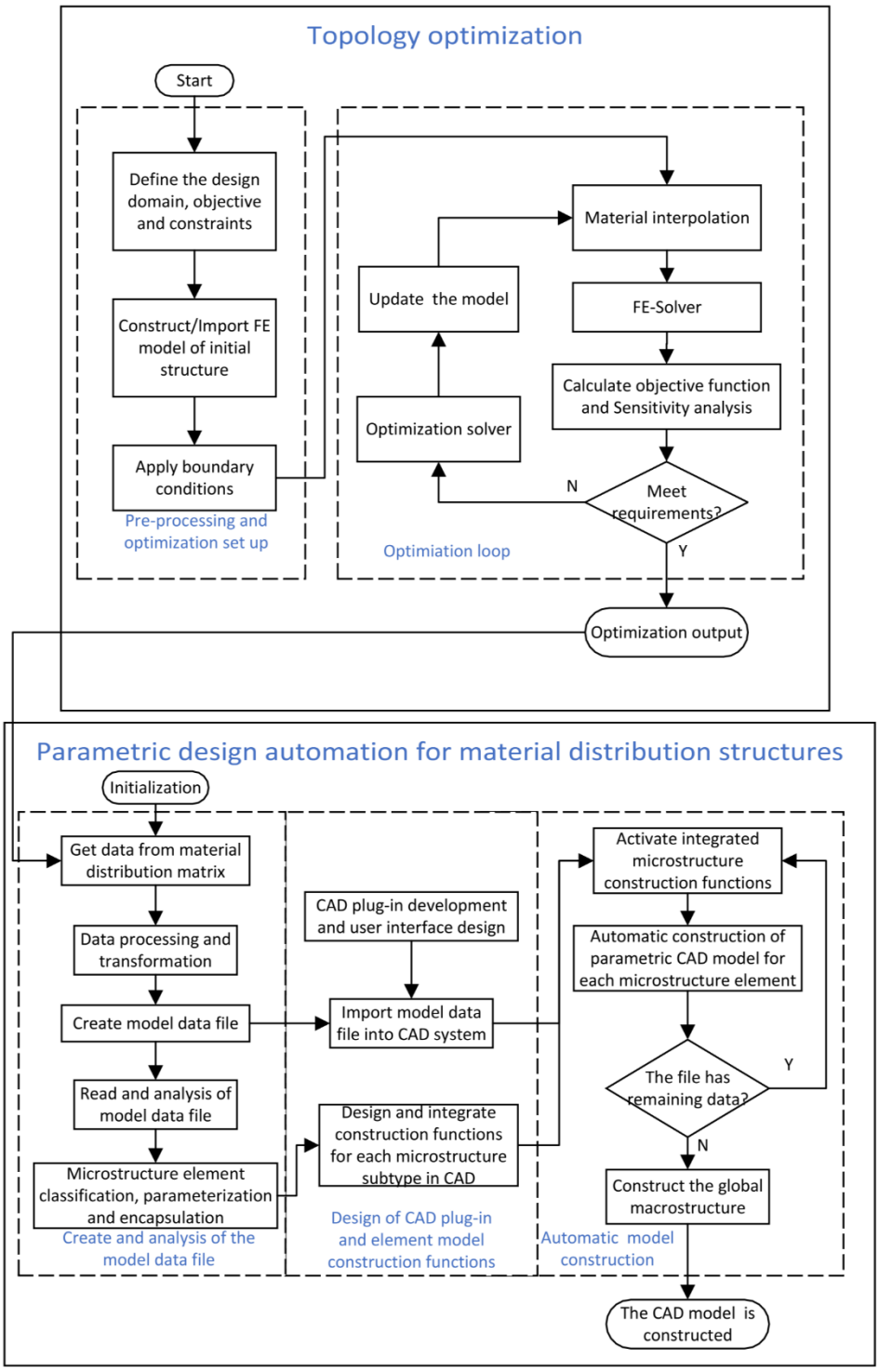


Figure 2.9 Flow charts of Topology optimization (top) and Parametric design automation for material distribution structures (bottom)

The relation between topology optimization and parametric design automation for material distribution structures can be represented in Figure 2.9. Topology optimization iteratively optimizes the structural design by reducing and/or redistributing its material with the updates of the values of design variables from the material distribution matrix in the optimization loop. The optimization output provides the original data of the material distribution structure for its parametric design automation in CAD. After processing and transforming the original data, the model data file is created and analyzed. Then, the model data file is imported into the CAD system and the CAD model will be constructed automatically with the designed plug-in and model construction functions in CAD. In summary, the output of the topology optimization result provides the original data of the material distribution structure as the input for its parametric design automation in CAD, and the latter provides CAD model reconstruction and post-processing operations for topology-optimized structures.

Chapter 3 Feature-based Modeling for Realizing Parametric Design Automation for Variable Fractal Geometry Design in CAD Systems

3.1 Chapter overview

Fractal geometry has been widely applied in computer graphics to visually represent natural objects, which cannot be easily represented by Euclidean geometry. Other than visual representation, parametric fractal-like CAD models have been used in industry for the design and manufacture of products with self-similarity geometric features. However, contemporary CAD systems cannot construct CAD fractal models efficiently, as most CAD fractal models are still built manually by designers. The construction and modifications for such CAD fractal models are tedious and time-consuming.

The research of this chapter aims to build and modify CAD fractal models more efficiently in CAD systems. A method for automatically modeling parametric fractal-like geometry objects in CAD systems was proposed. Feature-based modeling methods were applied to realize algorithm-based CAD fractal model generation. A new type of feature, called CAD fractal feature was proposed and implemented to bridge the gap between CAD and fractal geometry. The fractal-like CAD model was constructed iteratively with a dynamic generator for each iteration cycle. Design parameters were extracted as inputs for dynamic fractal generators, which allow users to define, parameterize, generate, and edit a fractal design with variable parameters. The parameters of the fractal generator can also take different values for different model construction iterations, which makes the generated models not necessarily to be strictly fractal but can also be fractal-like. The iterative generation method (IGM) was proposed as a built-in method in a CAD fractal feature for modeling complex but self-similar geometric models in an objected-oriented approach with high efficiency and sufficient design flexibility. As a result, designers do not need advanced knowledge

of fractal geometry to build and modify CAD models with self-similar features. An extruded Koch Snowflake CAD model with parametric geometric variants was built as a case study for demonstration of the proposed modeling method. The proposed method was proven to have sufficient efficiency and design flexibility in my case study.

The rest of this chapter is organized as below. Subsection 3.2 gives an introduction to this research, where the background, gaps, and motivations of this research are introduced. The related literature to this research has been reviewed in Chapter 2, including fractal geometry, contemporary CAD systems, and feature-based modeling method in subsections 2.1, 2.3, and 2.4. Subsection 3.3 gives a conceptual framework of the proposed method, and the partial relations of the proposed CAD fractal feature are represented in the form of a Unified Modeling Language (UML) diagram. A case study of generating a fractal feature in a CAD model with parametric geometric variants is demonstrated in subsection 3.4. Remarks are made in the discussion part in subsection 3.5. Subsection 3.6 concludes the research of this chapter. In the Appendix, a pseudo-code of the proposed modeling method in Siemens NX CAD system is explained.

3.2 Introduction

Geometric entities with self-similarity features widely exist in the real world and Euclidean geometry might not be adequate for efficiently describing and modeling such kinds of intricate and complex geometries [19]. Euclidean geometry is the natural way of representing and modeling man-made objects [121], however, it might be inept to describe some natural patterns, such as mountains, coastlines, and clouds with classical geometries such as straight lines, circles, ellipses, squares [122]. Then the concept of fractal geometry was proposed by mathematicians as an independent geometry branch to represent geometries with self-similarity and non-integer dimensionality [24].

Due to the self-similarity feature of fractal geometry, algorithm-based methods, such as recursion and iteration, have been widely applied in generating computer graphics models of fractal objects in a virtual environment. These algorithm-based methods have good model construction efficiency and have gradually become mainstream methods for visually representing fractal objects in computer graphics [28,123,124]. However, other than generating and representing a fractal pattern in a virtual visual device, such as a computer screen, the parametric design and physical manufacture of fractal geometry are also of great application value in industry. Computer graphics models of fractal objects are not sufficient to satisfy industrial design and manufacture requirements, and parametric CAD fractal models are needed for product lifecycle management in industry. For example, in aesthetic design industries, the fractal method has been widely applied in jewelry design [125], whose final purpose of the design outcome is to efficiently and conveniently design and manufacture the aesthetic physical design patterns. In such industrial cases, parametric CAD models of variable fractal patterns with solid geometry are preferred because of their better compatibility, reusability, and editability for different product development stages.

Contemporary CAD systems have developed a variety of ways to model or represent three-dimensional (3D) geometry objects or surfaces [7]. The solid modeling method [126] can be applied as an information complete and unambiguous solution for defining and modeling 3D shapes, while the surface modeling method [127] is often used as an efficient and convenient way to model free-form curves or surfaces [128]. In other words, CAD systems nowadays are sufficient and mature in general to model Euclidean geometric objects which can be defined in an analytic or semi-analytic equation or in the way being represented by non-uniform rational basis spline (NURBS). However, current CAD systems still cannot generate fractal objects algorithmically. As

a result, the design processes of fractal objects in CAD systems have some challenges in efficiency and convenience for the construction and modification of such models. In CAD environment, geometry objects are built in a model tree structure, which contains a hierarchical list of features or components in the order in which they were created. The model tree structure is good for showing model construction history, representing geometric dependency, and adjusting feature display. However, such a model construction approach might not be beneficial to realize some algorithmic methods to iteratively generate a fractal-like CAD model. As a result, most of the CAD fractal models are still constructed manually by designers. The construction and modification of such models still require a lot of repetitive work and are extremely time-consuming, especially for complex fractal objects with high iteration levels. Therefore, the study and development of efficient parametric methods for the automatic construction of fractal models in CAD systems have great potential to significantly improve the efficiency and convenience of the construction and modification processes of fractal-like CAD models.

The research of this chapter aims to develop an automatic model construction method that employs an algorithmic approach to iteratively construct fractal-like models in CAD systems. A feature-based method for modeling parametric variable fractal design was proposed. An efficient and flexible modeling interface was built for designers to generate CAD models with a self-similarity design feature. The iteration generation method (IGM) was proposed to make the creation, representation, and manipulation of the algorithmically constructed geometry objects applicable to a CAD system. The concept of CAD fractal feature is proposed, which encapsulates fractal geometry generation methods and associative relationships among composed geometry elements within a CAD fractal model. Associativity among models with different iteration levels was also realized and integrated into this CAD fractal feature. Thus, designers do not need to have

advanced expertise on how to generate a fractal pattern or geometry objects with self-similarity features with respect to its specific algorithm background. Additionally, the generator was constructed dynamically in a parametric way to make design parameters variable and editable in each iterative generation cycle for the final geometry. This results in the "self-similarity" property being variable for greater design flexibility.

3.3 Conceptual framework

This subsection introduces the conceptual framework and the methodology of the proposed feature-based modeling for variable fractal geometry objects in a CAD system. First, requirements for the fractal objects to be modeled by the proposed method are discussed, as not all fractal solids or surfaces can be modeled in CAD system with the proposed feature-based method. Second, the proposed CAD fractal feature is introduced and represented as an object class. Next, the proposed related object-oriented fractal generation method, namely the iterative generation method (IGM), is presented as an extension of traditional IFS. Then, the modeling processes based on the proposed modeling method are represented by a flow chart and a UML diagram. The relationships among designers, CAD system and fractal geometry objects are explained.

3.3.1 Requirements for fractal objects

In this research, the proposed modeling method is limited to representing and modeling fractal objects which must satisfy the following requirements:

(1) Regular fractal primitive.

(2) Contraction mapping.

(3) Self-connected during any iteration step.

(4) No self-intersecting or self-overlapping during iterative generation processes.

(5) Finite iteration.

(1) Regularity

The regularity of the model must be fulfilled to generate a model in a CAD system, which is one of the requirements in conventional geometrical modeling. The regularity of a set can be defined as follows.

Let S be a subset in X. The *interior* of S, $i(S)$ is a set of points that belongs to the union of all open subsets of S. The closure of S, $c(S)$, is a set of points that belongs to the intersection of all closed supersets of S. Then, the definition of a regular set can be stated as [77]:

A set S is *regular* if and only if $S=c(i(S))$.

This definition states that the set is regular if the closure of the interior of a given set yields that same given set. Isolated, detached, dangling sets, and sets with open boundaries are regarded as irregular sets [129]. Also, this regular set definition excludes geometries with non-orientable manifolds such as Möbius strip and Klein bottle (see Figure 3.1 for an illustrative example).

It has been proven that if a set is regular, the relevant generated fractal objects with finite iterations of contraction mappings from the set are regular [90].

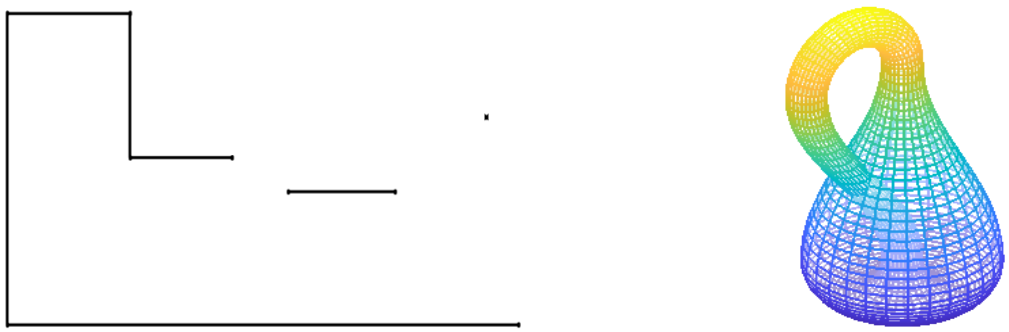


Figure 3.1 Irregular sets examples: A set with isolated, detached, dangling and open boundaries (left), and a set (Klein bottle) with non-orientable manifolds (right).

(2) Contraction mapping

The contraction mapping is the basic requirement for the IFS theory which maps the whole onto the parts and will result in convergence to an invariant set [21]. A transformation $f: X \rightarrow X$ on a metric space ($\boldsymbol{X}$,d) is called a *contraction mapping* if there is a contractivity factor such that

$$d(f(x), f(y)) \leq r \cdot d(x, y), \quad \forall\, x, y \in X \tag{3.1}$$

The proposed iterative generative method (IGM) is an extension of IFS which consists of a family of contraction mappings $F=\{f_1, f_2, \ldots, f_n\}$, $n \in \boldsymbol{N}$ on ($\boldsymbol{X}$,d) with corresponding contractivity factors $\{r_1, r_2, \ldots, r_n\}$ with $r_i \in (0,1)$.

Thus, for all $f_i: X \rightarrow X$,

$d(f_i(x), f_i(y)) \leq r_i \cdot d(x, y)$, where $i \in N$.

In existing CAD modeling, most of the geometric sets are closed and bounded, which are called *compact* sets. It is necessary to introduce the Hausdorff metric of ($\boldsymbol{X}$,d) for the contraction mapping to deal with a compact subset in ($\boldsymbol{X}$,d). For two non-empty compact sets S_1, S_2 in ($\boldsymbol{X}$,d), the *Hausdorff metric* d_H is defined as [24]

$$d_H(S_1, S_2) = \max\{\max_{x \in S_1} \min_{y \in S_2} d(x, y), \max_{y \in S_2} \min_{x \in S_1} d(x, y)\}$$

Thus, for all $f_i: X \rightarrow X$,

$$d_H(f_i(S_1), f_i(S_2)) \leq r_i \cdot d_H(S_1, S_2), \quad \text{where } i \in N \text{ and } r_i \in (0,1) \tag{3.2}$$

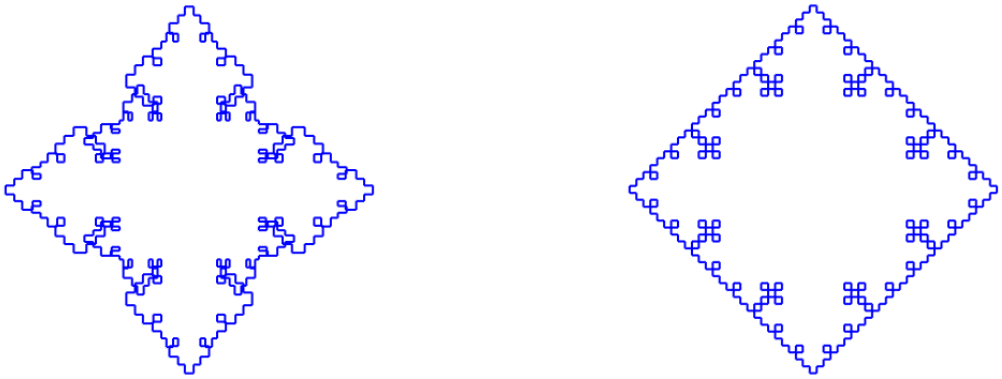
(1) Connectivity

It has been proposed by S. C. Soo et al [90] that three different types of connected models can be generated by IFS, which are self-connected fractal models, host-connected fractal models, and cumulatively connected fractal models. In my proposed method, I focus on modeling self-

connected fractal models and more constraints have been made for self-connectivity. For any intermediate state of the fractal object with finite iteration step k, it is required that the model $F^k(\boldsymbol{S})$ is self-connected.

(2) No self-intersecting or self-overlapping during iterative generation processes

During the iterative generation steps of a fractal object, it may be possible that the primitives of a fractal model at the higher level are intersected or overlapped by other primitives in the same or lower levels (see Figure 3.2 for an example). The topology and connectivity of the fractal model would change when such an intersection or overlap happens. For these fractal objects with features of self-overlap or self-intersection during the iterative generation processes, a specific data structure is recommended to address this information during the CAD modeling, and intersection/overlap inspection/detection should be rendered during the preprocessing stage of manufacturing. It is a research topic to study the modeling and manufacturing of these self-overlapped or self-intersect fractal objects, but it is not within the scope of this work. In this study, I focus on CAD modeling fractal objects that do not have self-intersection or self-overlap features during the iterative model generation processes.



(a) (b)

Figure 3.2 Variant Koch curves with (a) self-overlap, or (b) self-intersection.

(3) Finite iteration

Mathematically speaking, a pure fractal object is in some ways quite a large set (it is uncountably infinite) and its size is not quantified by the usual measures such as length, which should go through infinite iteration steps if generated iteratively. In reality, however, fractal objects should be generated through finite iteration. In the aspect of CAD modeling, the resolution of the model representation is limited, and the time and memory needed for generating a fractal model should be taken into account. In the aspect of manufacturing, machine resolution should be considered, which limits the minimum machinable dimension. Stopping criteria for fractal generation steps or a rule to decide the maximum level of iteration should be considered. In my modeling method, the level of iteration N can either be user-defined or be calculated according to the relationship between the dimensions of the fractal primitive and the machine resolution, which could be expressed as follows:

D_p is the shortest dimension (length, width, or height) of the minimum bounding box of the fractal primitive. D_m is the machine resolution, which represents the minimum machinable dimension. E is a real number larger than 1, which represents the reciprocal of the scaling factor ε, $E=1/\varepsilon$. For a

regular fractal object with a constant scaling factor ε for each iteration, the maximum level of iteration N_{max} should satisfy the following condition:

$$E^{N_{max}} \leq \frac{D_p}{D_m}$$

Then, N_{max} can be calculated as:

$$N_{max} = \text{floor}\left(\frac{\log\left(\frac{D_p}{D_m}\right)}{\log E}\right) \tag{3.3}$$

Where floor(·) is the floor function to round off the value to the nearest small integer.

For a variable fractal-like object which might have variable scaling factor ε_i and the reciprocal E_i for the ith iteration step, the condition for N_{max} should be satisfied as:

$$E_1 \cdot E_2 \cdot \ldots \cdot E_{N_{max}} \leq \frac{D_p}{D_m} < E_1 \cdot E_2 \cdot \ldots \cdot E_{N_{max}} \cdot E_{N_{max}+1} \tag{3.4}$$

Then, it is easy to calculate N_{max} for a variable fractal-like object with variable scaling factors.

3.3.2 CAD fractal feature

A CAD fractal model is constructed iteratively. Using the model of an extruded and chamfered Koch snowflake as an example, the steps are exemplified as follows (see in Figure 3.3):

1. First, a fractal primitive is created by the built-in fractal primitive constructor.
2. Next, the initiator is activated to act on the fractal primitive to construct its initial fractal pattern before iteration and initialize the related fractal data structure.
3. Then, IGM is applied to the fractal data structure with several iteration steps.

4. Finally, CAD postprocessing operations are performed to generate the CAD fractal model, such as extrusion, chamfering, filleting.

During the iterative generation processes, the geometric entities, such as points and line segments are built via a specific parametric generation algorithm and their associative relationships, such as the sequence of points in the generated point set and the order of connectivity are built by dynamically adding/deleting/editing feature expressions. After all iteration steps, a validator is activated to check the connectivity and closure of the generated fractal model and the inspection process of self-intersection or self-overlap is implemented.

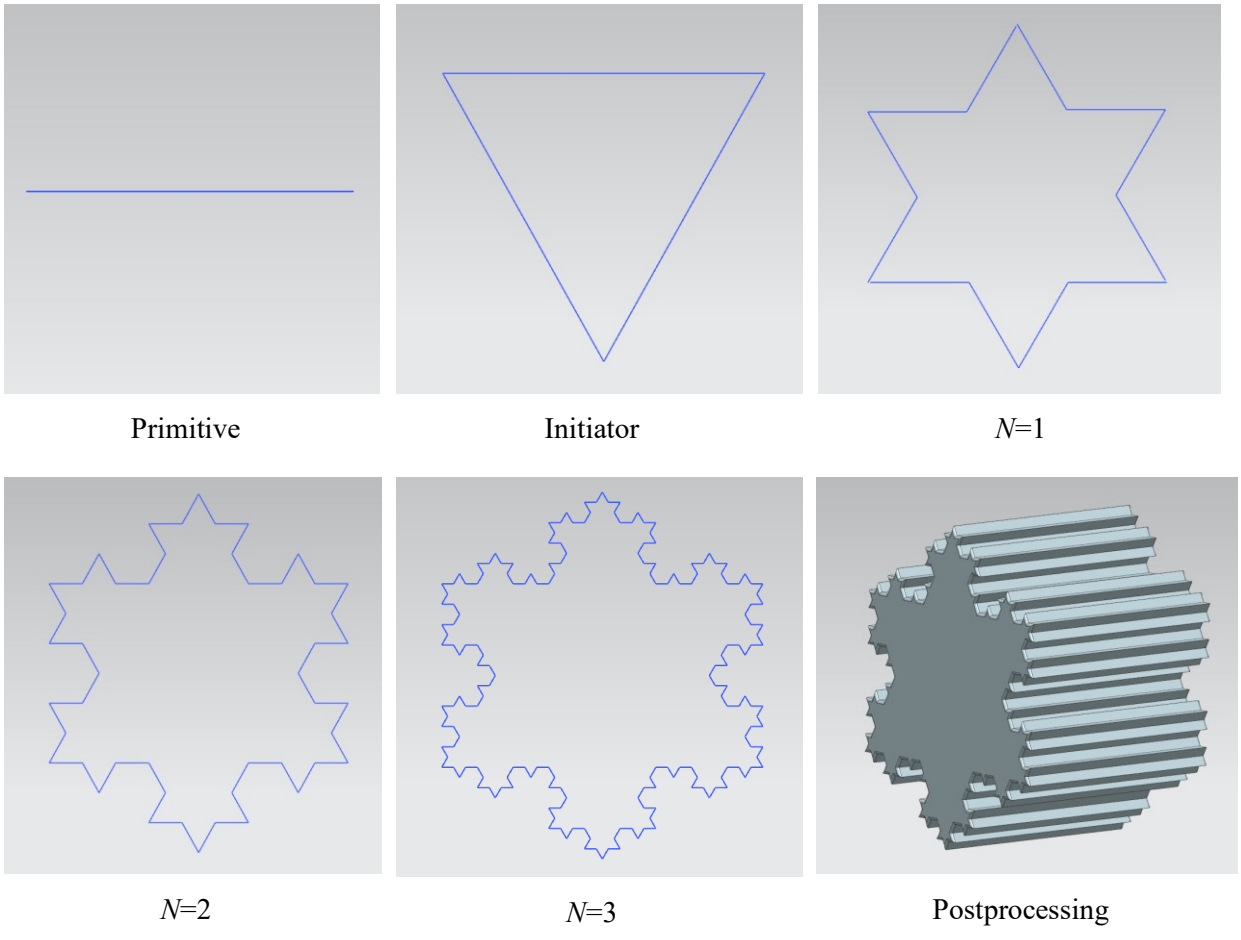


Figure 3.3 CAD modeling process of an extruded Koch Snowflake model with the first 3 iteration steps (N=3).

It can be observed that such design intent of iteratively generating fractal objects in CAD with editable variable parameters cannot be predefined or represented by existing feature types, and yet significant commonality exists among different instances. The templated framework of modeling a fractal or fractal-like geometric object with variable design parameters using an iterative generation modeling method in CAD is essential. It is named as CAD fractal feature.

The CAD fractal feature is very difficult to be represented with the traditional feature concept because the model is constructed iteratively, the geometric entities and feature expressions are created/deleted/edited dynamically, and the high-level knowledge of algorithm-based fractal generation method are integrated whenever such features are involved.

To represent CAD fractal features consistently, the object-oriented approach offers an excellent solution. The fractal primitive, the specific data structure, the parametric iterative generation method, and the related validator could be integrated into an object class in CAD after encapsulation which can conveniently and efficiently represent the iterative design intent, algorithm-based model generation knowledge, and the associative relationships or topological configuration of the geometric entities. It has been summarized by Ma et al [105] that a feature has to be flexible, self-contained and consistent to integrate different applications. Here, “flexible” means the data structure of the CAD fractal feature can be created, edited, and deleted by the end user dynamically through the proposed iterative generation method and the variable design parameters for the fractal object generator can be defined and roll-back edited through user interface. “Self-contained” means the fractal feature can keep its validation and integrity before and after any interaction with any integrated applications. The requirements of the fractal objects in subsection 3.1 and the built-in validator for the inspection/detection of self-intersection and self-overlap for the fractal model make it well-defined and represented by a self-contained object. The

CAD fractal model is constructed iteratively with the built-in method to integrate associative relationships among component geometric elements and the model has rich information to be consistently integrated with CAE analysis or CAM processes.

To address the requirements for parametrically modeling variable fractal objects in CAD system and keep the information consistency for the CAD fractal model to be integrated with other applications, the proposed CAD fractal feature should have the following key characteristics:

- Built-in fractal primitive constructor to generate its related fractal primitives.
- Built-in initiator to construct its initial fractal pattern before iteration and initialize the related fractal data structure.
- Methods available for iteratively generate a fractal model with editable parameters.
- Methods available for calculating the maximum iteration number N_{max} and multi-level representation of the model.
- Self-validation for the consistency of its geometrical requirements.
- Methods available for communication with end users for adding/deleting/editing design parameters.
- Methods available for constructing, storing, displaying, editing, indexing and destroying its instances.

Based on the above concept, a fractal modeling toolbox for modeling fractal or fractal-like objects integrated in a CAD system is developed in NX Open in this work, which is the Application Programming Interface (API) of Siemens NX for design customization.

The IGM is an essential built-in method for the CAD fractal feature to construct a CAD fractal model iteratively and parametrically. It is an extension of the IFS with explicit expressions of extracted design parameters to realize feature-based parametric modeling of fractal and fractal-like geometric objects in a CAD system. The introduction of the concept of the IGM is found in the next subsection.

3.3.3 Iterative generation method (IGM)

The IGM is an extension of the IFS. In this section, a brief review of traditional IFS is introduced first and then the proposed IGM is explained in detail.

(1) Iterated Function System (IFS)

The Iterated Function System (IFS) is the most commonly used deterministic approach to generate or approximate a mathematical fractal object. An IFS is a family of specified contraction mappings that map a whole object onto the parts, unionize all the parts, and iteration of these mappings will result in convergence to an invariant set [17]. Let $\boldsymbol{S}$ be a non-empty compact set in $(\boldsymbol{X},d)$, a combined transformation $F: \boldsymbol{X} \rightarrow \boldsymbol{X}$ can be defined by

$$F(S) = f_1(S) \cup f_2(S) \cup f_3(S) \cup \ldots \cup f_n(S) = \bigcup_{i=1}^{n} f_i(S) \tag{3.5}$$

Hence, the kth iteration of F (i.e. F^k) will be

$$\begin{cases} F^0(S) = S \\ F^1(S) = F(F^0(S)) = \bigcup_{i=1}^{n} f_i(F^0(S)) \\ F^2(S) = F(F^1(S)) = \bigcup_{i=1}^{n} f_i(F^1(S)) \\ \quad \vdots \qquad\qquad \vdots \qquad \vdots \\ F^k(S) = F(F^{k-1}(S)) = \bigcup_{i=1}^{n} f_i(F^{k-1}(S)) \end{cases} \tag{3.6}$$

Where n is the number of functions f_i in the function group F and k is a positive integer which represents the number of iterations. Suppose that k is large enough and the fractal is represented in a very detailed and complex model, then the fractal set becomes invariant. That is, there exists a unique non-empty compact set $\boldsymbol{A} \subset \boldsymbol{X}$ that is invariant for F, i.e. which satisfies

$$\boldsymbol{A}=F(\boldsymbol{A})= \bigcup_{i=1}^{n} f_i(\boldsymbol{A}) \tag{3.7}$$

Taking the Koch snowflake as an example (see Figure 3.4), its fractal primitive is a line segment and the triangle initiator is applied which includes three transformations I_1, I_2, and I_3. The IFS has four functions in total which are: $F = \{f_1, f_2, f_3, f_4\}$ and Figure 3.4(c) illustrates the transformations. The IFS for the Koch curve can be represented as the following:

$$F^k(S) = F(F^{k-1}(S)) = \bigcup_{i=1}^{4} f_i(F^{k-1}(S)) \tag{3.8}$$

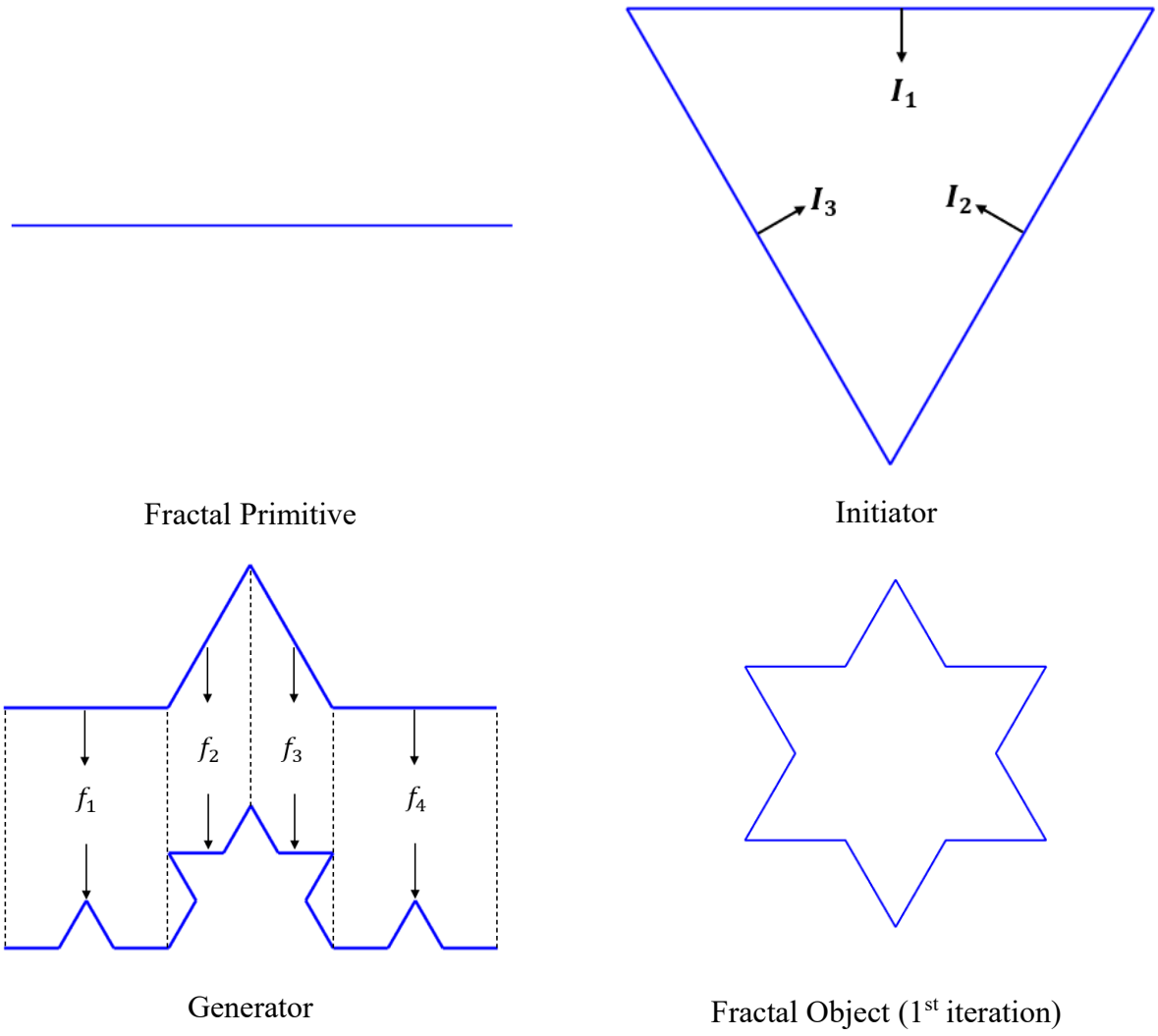


Figure 3.4 IFS: Koch Curve example: (a) The Fractal Primitive, (b) The Initiator, (c) The Generator, and (d) The Fractal Object (1st iteration).

(2) Iterative Generation Method (IGM)

Based on the theory of IFS and feature-based parametric CAD modeling, I propose the Iterative Generation Method (IGM) for modeling fractal or fractal-like geometric objects with variable design parameters in a CAD system. The mathematical description of IFS and IGM for modeling fractals is similar, which is iteratively acting on a regular set $\boldsymbol{S}$ with a family of finite contraction mappings. As an extension to IFS, the iterative generator of IGM includes variable

design parameters which affect the "behavior" of the generator and the final constructed fractal model.

Definition. Let $x_1, x_2, \ldots, x_m$ be the total number of m extracted variable parameters for the fractal generator. $\boldsymbol{S}$ is a regular compact set and $g_i: \boldsymbol{X} \rightarrow \boldsymbol{X}$ is an iterative generation function which is a compact mapping of $\boldsymbol{S}$ on the metric space $(\boldsymbol{X},d)$ with variables $x_1, x_2, \ldots, x_m$. Then, the iterative generation function g_i can be denoted as:

$$g_i = g_i(\boldsymbol{S}, x_1, x_2, \ldots, x_m) \qquad for\ \forall\ i\ \in \{1,2,\ldots,n\} \tag{3.9}$$

The iterative generation function group G includes a total number of n iterative generation functions $\{g_1, g_2, \ldots, g_n\}$. So the combined transformation G can be expressed as:

$$\begin{aligned} G(S) &= G(\boldsymbol{S}, x_1, x_2, \ldots, x_m) \\ &= g_1(\boldsymbol{S}, x_1, x_2, \ldots, x_m) \cup g_2(\boldsymbol{S}, x_1, x_2, \ldots, x_m) \cup \ldots \\ &\qquad \cup g_n(\boldsymbol{S}, x_1, x_2, \ldots, x_m) \\ &= \bigcup_{i=1}^{n} g_i(\boldsymbol{S}, x_1, x_2, \ldots, x_m) \end{aligned} \tag{3.10}$$

The main differences between IGM and IFS have three points. First, IGM is mainly based on object-oriented programming/CAD feature modeling, which means the IGM is a built-in method of a class/CAD feature type. This is why it should be called a "method". A proper level of encapsulation and reusability of IGM is guaranteed by the object-oriented approach. Second, variable parameters are extracted in IGM, and the iterative generator will be explicitly affected by the value of these parameters. This makes IGM keep more design flexibility for constructing the fractal object by editing design parameters. The extracted design parameters $x_1, x_2, \ldots, x_m$ are not

limited to be geometry dimensions, but can also be parametric equations such as expressions of constraints or associative relationships among parts/faces/interfaces. Additionally, the fractal generators are constructed dynamically for each iteration cycle and the variable design parameters may take different values for different iteration steps, which will then lead to different iterative generation functions for different iteration steps during the model construction processes. This gives the IGM the capacity to model not only strictly fractal objects but also fractal-like objects or objects with the feature of self-similarity but still can keep some variance of the "similarity".

Apart from constructing a model iteratively, IGM takes advantage of parametric modeling in CAD. In addition, efficient communications between the CAD system and designers or other end users are implemented by adding/deleting/editing extracted design parameters with the dynamic fractal generators. In a broad sense, IGM is an object-oriented method for parametrically constructing a fractal or fractal-like object iteratively. The IGM is essential for a CAD fractal feature, but its application is not limited to a CAD system.

Here, let us take an example of generating a parametric Koch snowflake with my proposed IGM. Figure 3.5 shows a parametric snowflake generator. In this generator, the indentation angle θ, the orientation angle α, and parent segment length L are independent variable parameters. Other than these independent parameters, there are 3 associative parameters in this generator, which are children segment length l, scaling factor ε, and fractal dimension Dim. Traditionally, Koch curve and Koch snowflake have a fixed indentation angle of 60°, and their attributes, such as fractal dimension and scaling factor are fixed. Koch curve with a varying indentation angle was proposed by Vinoy et al. [130] for antenna design and its scaling factor ε and fractal dimension Dim were then expressed by Vinoy et al. as a function of θ:

$$\varepsilon=\frac{1}{2(1+cos\theta)}=\frac{l}{L} \tag{3.11}$$

$$Dim=\frac{log\ Num}{log\frac{1}{\varepsilon}}=\frac{log4}{log[2(1+cos\theta)]} \tag{3.12}$$

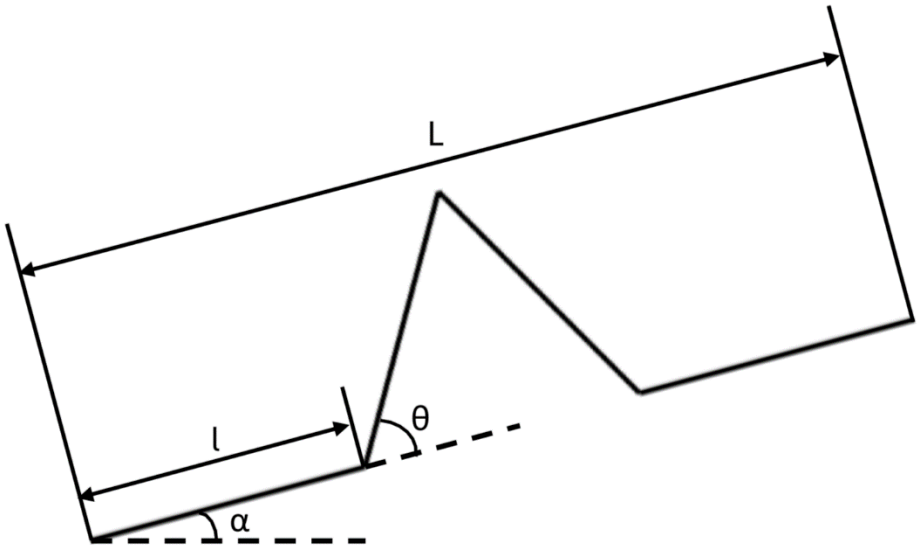


Figure 3.5 Koch curve generator with a variable indentation angle θ.

In the fractal modeling point of view by IGM, the indentation angle θ is extracted to be the variable design parameter for the parametric modeling of a generalized Koch snowflake. Then, each of its iterative generation function g_i and the function group G can also be expressed as functions of θ:

$$g_i = g_i(S,\theta) \quad \text{for } i \in \{1,2,3,4\} \tag{3.13}$$

$$G = G(S,\theta) = \bigcup_{i=1}^{4} g_i(S,\theta) \tag{3.14}$$

In the CAD modeling process, the indentation angle θ will be an input parameter by the end user for the fractal generator and the geometry objects of different shapes will be constructed automatically for different θ. If the values of θ are taken to be 45° and 75° respectively, for instance, the iterative generator is shown in Figure 3.6. It is obvious that different indentation angles θ will lead to different fractal dimensions and sizes of the geometry. If θ is within the range

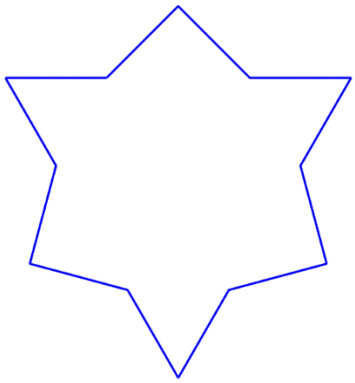
(0°,90°), the smaller the indentation angle, the constructed geometry will have a lesser fractal dimension and more compact size. Then, after applying these generators to two equal initiators (two equilateral triangles of the same size as Figure 3.5(b)) the Koch snowflakes of the first four iterations are constructed with two different indentation angles (see Figure 3.7). It is noteworthy that the length of each line segment component of the generators and the Koch snowflakes are exactly the same, which can be expressed as a function of the indentation angle θ for a specific iteration:

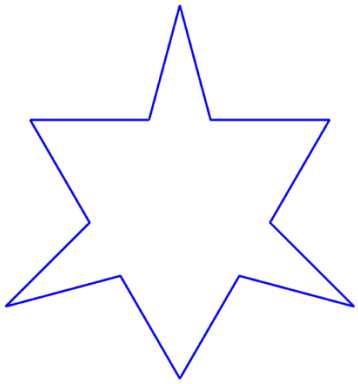
$$l^N = \left[\frac{1}{2(1+cos\theta)}\right]^N L \tag{3.15}$$

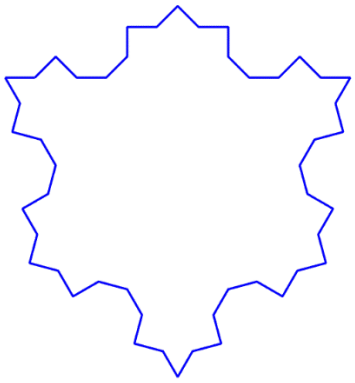
where l^N is the current line segment length, L is the primitive segment length, θ is the indentation angle, N is the number of iterations.

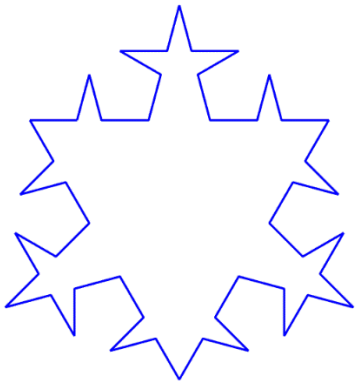
Figure 3.6 Different Koch curve/snowflake generators with respect to different indentation angle θ. Left: $\theta = 45°$ Right: $\theta = 75°$

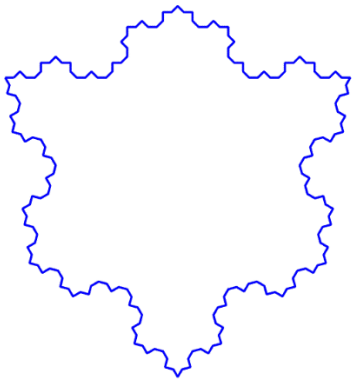
θ=45°
θ=75°

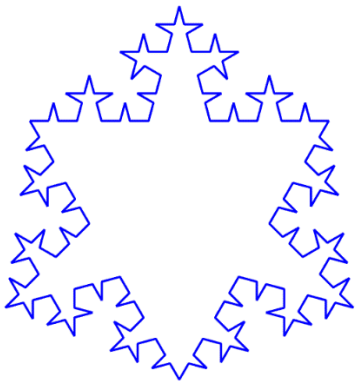












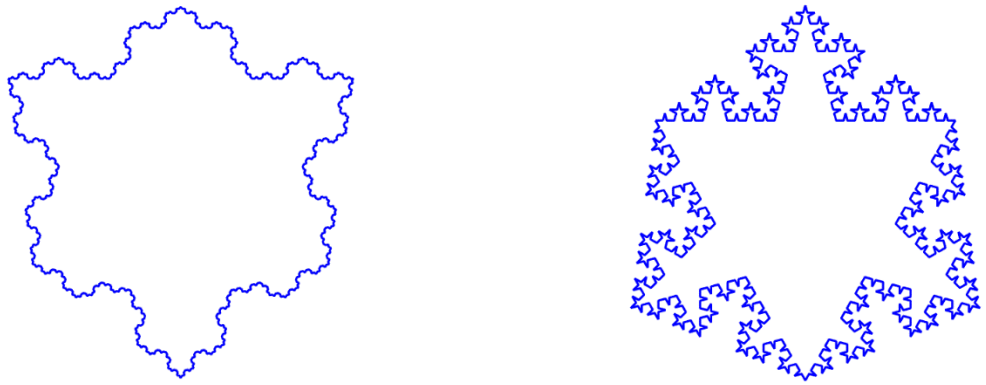


Figure 3.7 Generalized Koch snowflakes of the first four iterations with two different indentation angles. Left: $\theta = 45°$ Right: $\theta = 75°$

In IGM, the extracted parameters may also take different values with different iteration steps. Thus, fractal-like geometry objects with not strictly self-similar features can be constructed with IGM. In the construction processes of a Koch-like snowflake with variable indentation angles by IGM, the value of the indentation angle θ may change with respect to different iterations. Here, θ^k is denoted as the indentation angle for kth iteration. Then, for the kth iteration, the related iterative generation function g_i^k, function group G^k, current segment length l^k can also be expressed as functions of θ^k:

$$g_i^k = g_i^k(S, \theta^k) \quad for\ i \in \{1,2,3,4\}\ and\ \forall\ k \in \boldsymbol{Z}^+ \tag{3.16}$$

$$G^k = G^k\ (S, \theta^k) = \bigcup_{i=1}^{4} g_i^k(S, \theta^k) \tag{3.17}$$

$$l^k = \frac{1}{2(1+cos\theta^k)}\, l^{k-1} = \left[\frac{1}{2(1+cos\theta^k)}\right] \times \left[\frac{1}{2(1+cos\theta^{k-1})}\right] \times \ldots \times \left[\frac{1}{2(1+cos\theta^1)}\right] L \tag{3.18}$$

Figure 3.8 shows the construction processes of a generalized Koch-like snowflake of the first four iterations, where the indentation angles are different for different iterations and are set as θ^1=30°, θ^2=45°, θ^3=60°, θ^4=75°, respectively. The constructed snowflake is not a strictly self-similar or a strictly fractal geometry object according to the definition. Thus, more design

flexibility can be realized by IGM without excessive increase of too much algorithm complexity and computational cost.

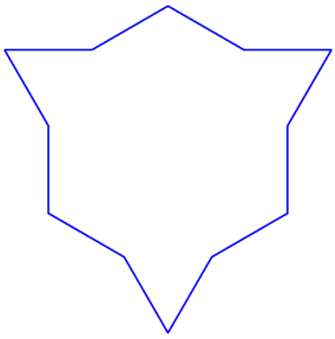
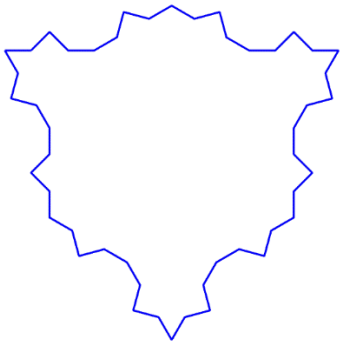
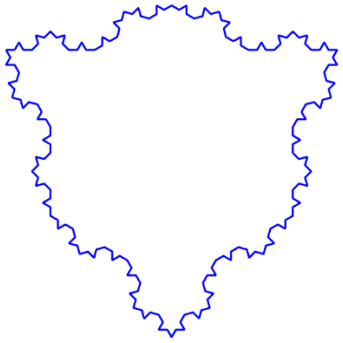
Current Iteration	Current generator	Current Snowflake
1 θ^1=30°		
2 θ^2=45°		
3 θ^3=60°		

Figure 3.8 Construction processes of a generalized Koch-like snowflake of the first four iterations with variant indentation angles as: θ^1=30°, θ^2=45°, θ^3=60°, θ^4=75°.

If the indentation angles have a periodicity feature, for example, θ^{4k+1} =30°, θ^{4k+2}=45°, θ^{4k+3}=60°, θ^{4k+4}=75° for $\forall\ k \in \boldsymbol{N}$, then, according to the property above, the constructed geometry object will converge to an invariant set. It is also noteworthy that the sequence for the parameter values with respect to iteration numbers will significantly affect the final shape of the geometry. In Figure 3.9, two snowflakes from the first two iterations are constructed with θ^1=30°, θ^2=75° (left) and θ^1=75°, θ^2=30° (right). It is obvious that they have different shapes simply because the sequence of their variable parameter values is different. However, according to the segment length in equation (3.18), the two models have the same segment length.

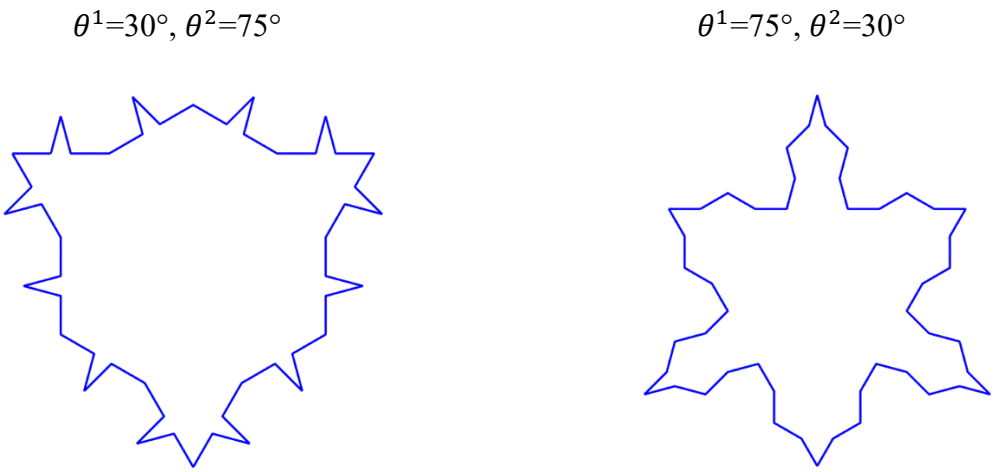


Figure 3.9 Two different shapes of generalized Koch-like snowflakes of the first two iterations with indentation angles as: θ^1=30°, θ^2=75° (left) and θ^1=75°, θ^2=30° (right).

3.3.4 Processes for the construction of a CAD fractal model

In this subsection, the construction processes of a CAD fractal model is introduced. First, the CAD fractal modeling process is described in detail, including the procedure flow and the information flow. Then, a partial class relation diagram in UML format is shown to represent each component and their association relationships for the CAD fractal modeling framework.

Figure 3.10 shows the procedure flow and the information flow for the construction process of a CAD fractal model. The procedure flow shows the sequence of steps, and the information flow represents data transformation, dependency, and associative relationships among different components. In the customized user interface (UI), the user needs to first select a fractal type. Next, based on the fractal type selected, relevant available fractal initiators appear in the UI and the user needs to select a fractal initiator to initialize the CAD fractal model. Then, the design variables are demonstrated in a sample graph and the user needs to input their values. After that, the maximum iteration number N_{max} is calculated using either equation (3.3) or (3.4) based on the input parameters. The iterative generation method is applied to the initialized model in a loop of N_{max}

times and a new model is constructed and stored in each iteration cycle. When the model construction process is completed, a built-in validator for checking self-intersection and self-overlapping is enabled. If self-intersection or self-overlapping exists in the model, the user will receive a warning and redesign procedures need to be performed by either changing a fractal initiator or modifying relevant values of the design parameters. The CAD fractal model will be reconstructed and checked again by the validator. Finally, CAD postprocessing is performed according to the users' needs, such as filleting, chamfering, and extrusion.

Figure 3.11 shows the UML diagram which represents the partial relations defined in the CAD fractal feature. Four types of associations, namely aggregation, composition, navigation, and dependency exist in the fractal CAD modeling framework. Each 'CAD fractal feature' has one 'Fractal initiator' as a one-to-one composition, but possibly a few 'Dynamic parametric generators' depending on the objective geometry elements. For each iteration, there is a current fractal generator, which is constructed dynamically and parametrically. The maximum number of iterations is an important attribute for a 'CAD fractal feature', which has a direct effect on the complexity level of the constructed CAD fractal model and the number of the composed 'Dynamic parametric generators'. The maximum number of iterations can either be user-defined or calculated by a specific rule depending on attributes of a related 'CAM manufacturing feature' such as the machine resolution according to the parameter list input by the user. Each 'CAD fractal feature' has one parameter list which can be directly created/edited by the user and related design parameters may have default values and warning of exception value errors for the robustness of the CAD fractal feature. A 'CAD fractal feature' may have a few 'Fractal model validators' for the validation of self-intersection and self-overlapping in different regions.

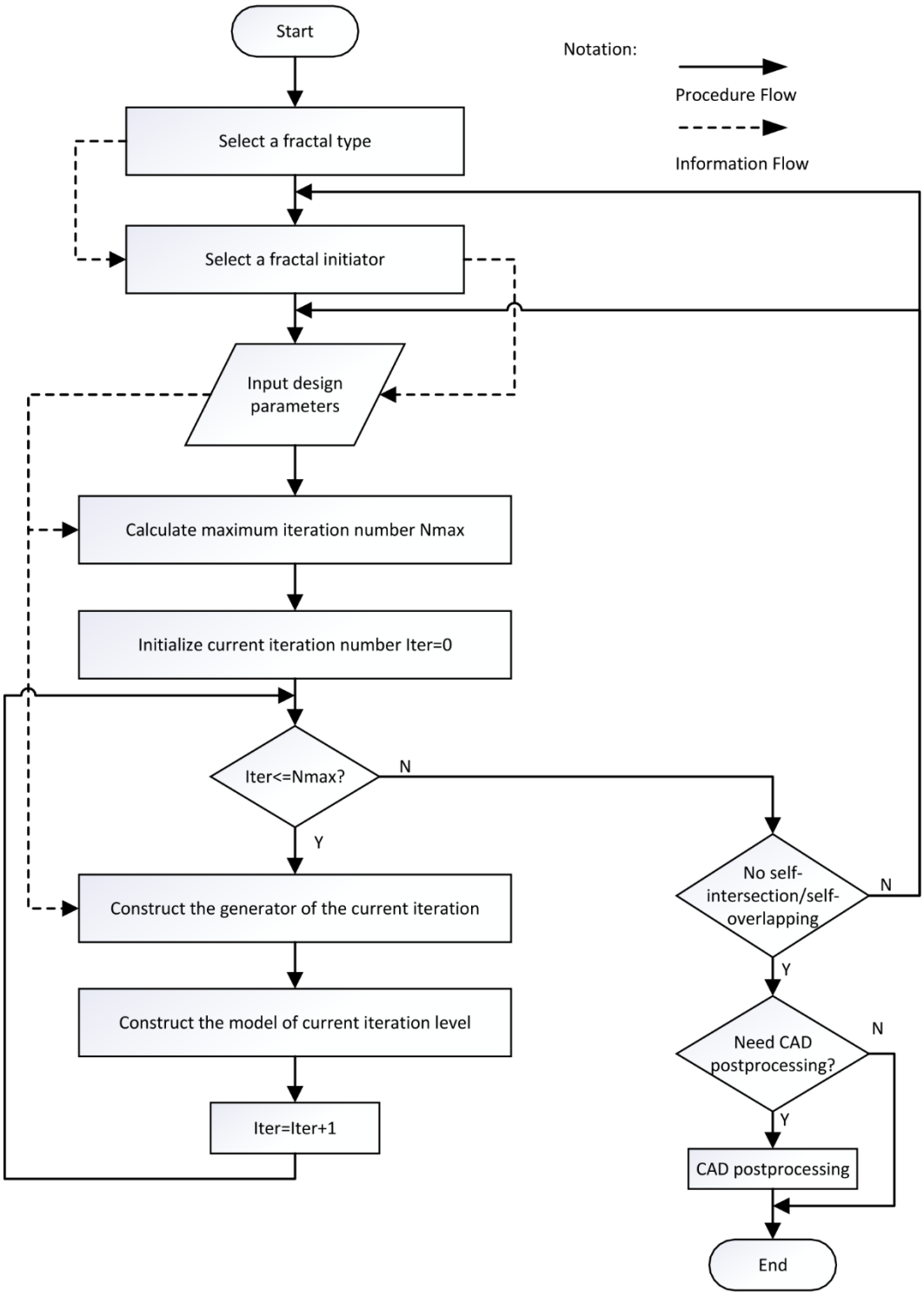


Figure 3.10 Procedure and information flow for the construction of a CAD fractal model.

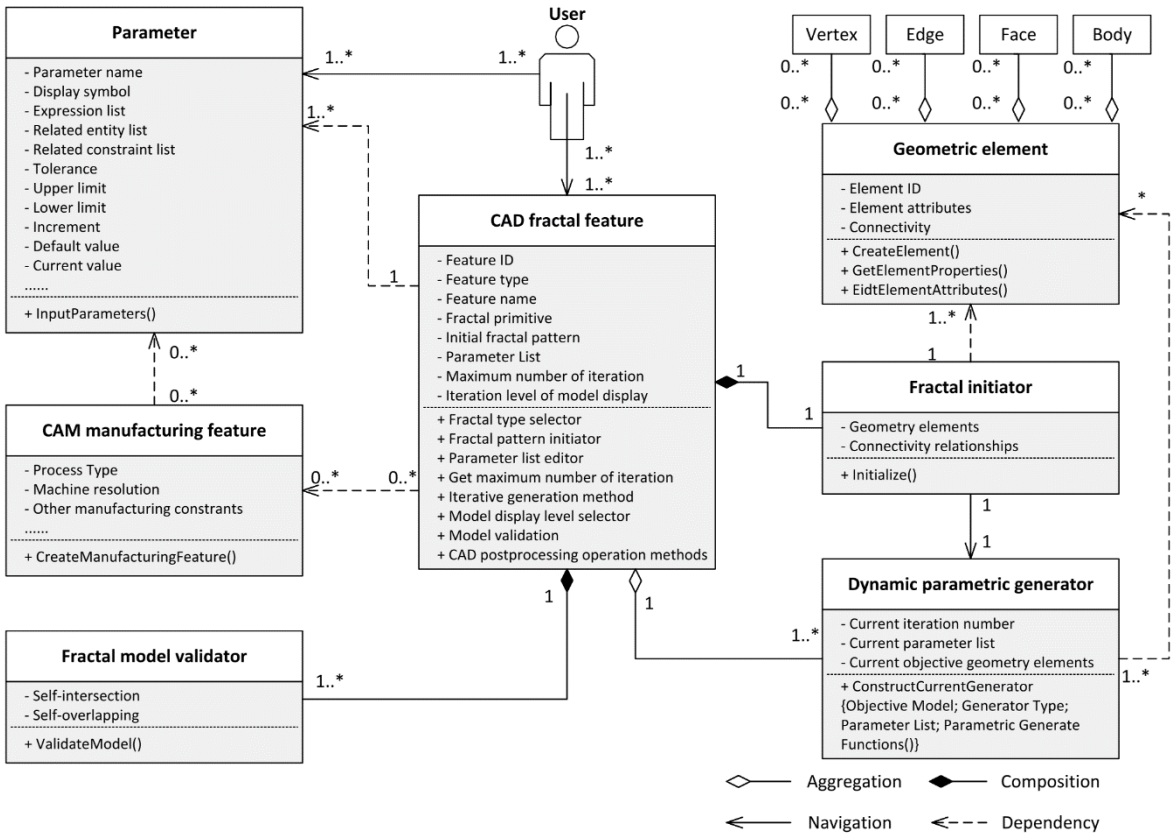


Figure 3.11 UML diagram of partial relations defined in the CAD fractal feature.

3.4 Case study

In this subsection, an extruded Koch Snowflake CAD model with parametric geometric variants has been constructed in Siemens NX for the demonstration of my proposed feature-based variable fractal geometry modeling method. In my case study, a plug-in named “Koch Snowflake” is developed in the NX 12 system using NX Open API (see Figure 3.12). Feature-based modeling is used as an object-oriented approach to construct the snowflake model and the proposed iterative generation method (IGM) is applied as a built-in method for the CAD snowflake fractal feature. My main contributions in this case include:

(1) Automatic construction of an extruded Koch Snowflake CAD model with extracted variable design parameters.

(2) Application of the proposed IGM to construct dynamic fractal generators and the parametric CAD Koch-like Snowflake feature.

(3) Multilevel representation of the Koch Snowflake CAD model with different geometric complexities.

(4) Rich information and available postprocessing operations for the constructed CAD model.

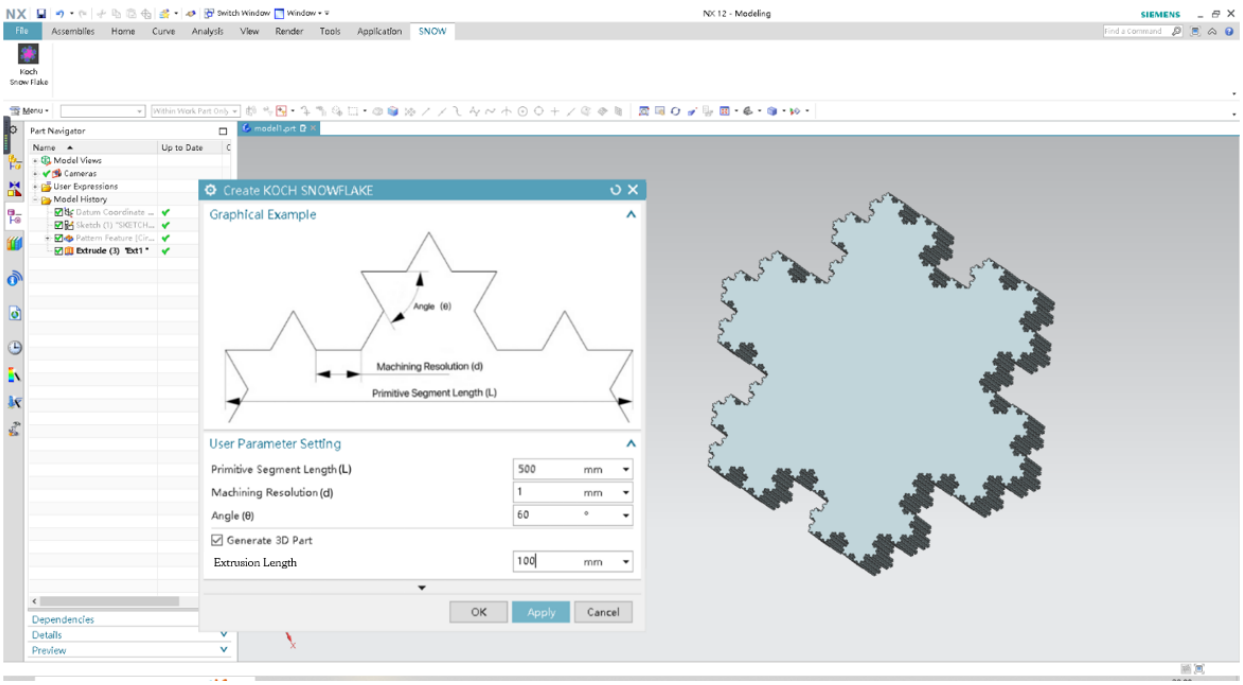


Figure 3.12 “Koch Snowflake” plug-in and its user interface in NX 12.

3.4.1 Automatic construction of a parametric extruded Koch Snowflake CAD model

In this case, three parameters are extracted to act as variable design parameters, which are primitive segment length L, machining resolution l, and angle θ. The primitive segment length L represents the length of the initial Koch line segment, the machining resolution d represents the minimum precision length for the available manufacturing machine to make the fractal prototype, and angle θ represents the indentation angle of the Koch generator (see Figure 3.5). Figure 3.13

shows the customized UI of this model creator, where a graphical example is on the top for an illustration of these three design parameters. The end user will then input the values of these parameters. An option for whether to extrude the snowflake sketch to generate a 3D extruded part model is provided and the user can extrude the sketch to the desired length.

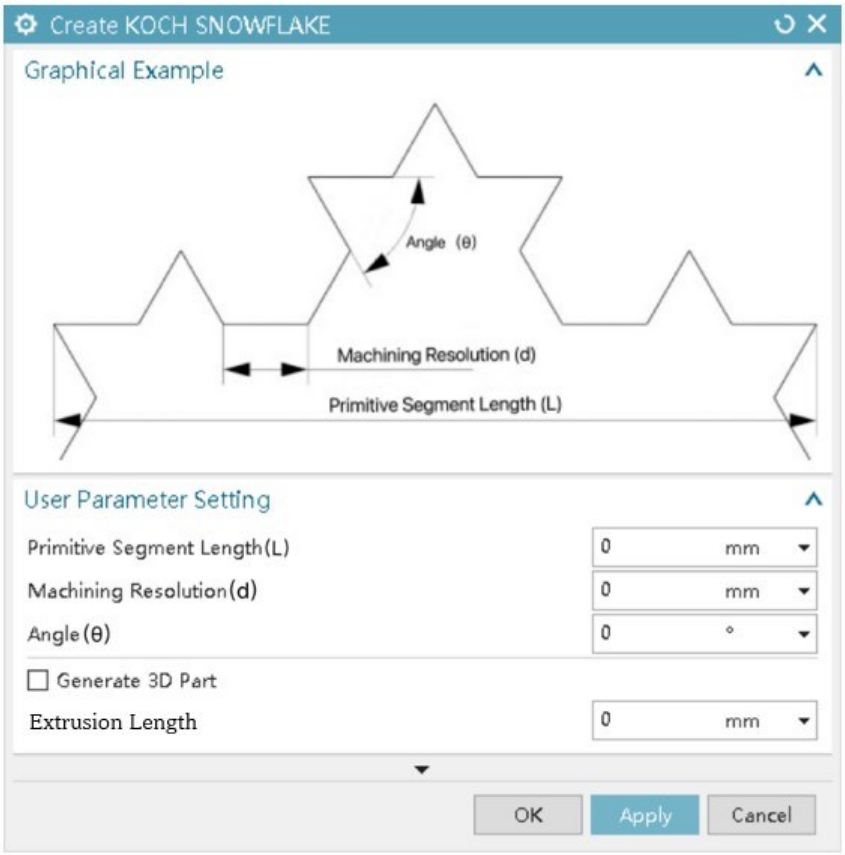


Figure 3.13 Customized UI for automatically constructing a variable Koch Snowflake CAD model.

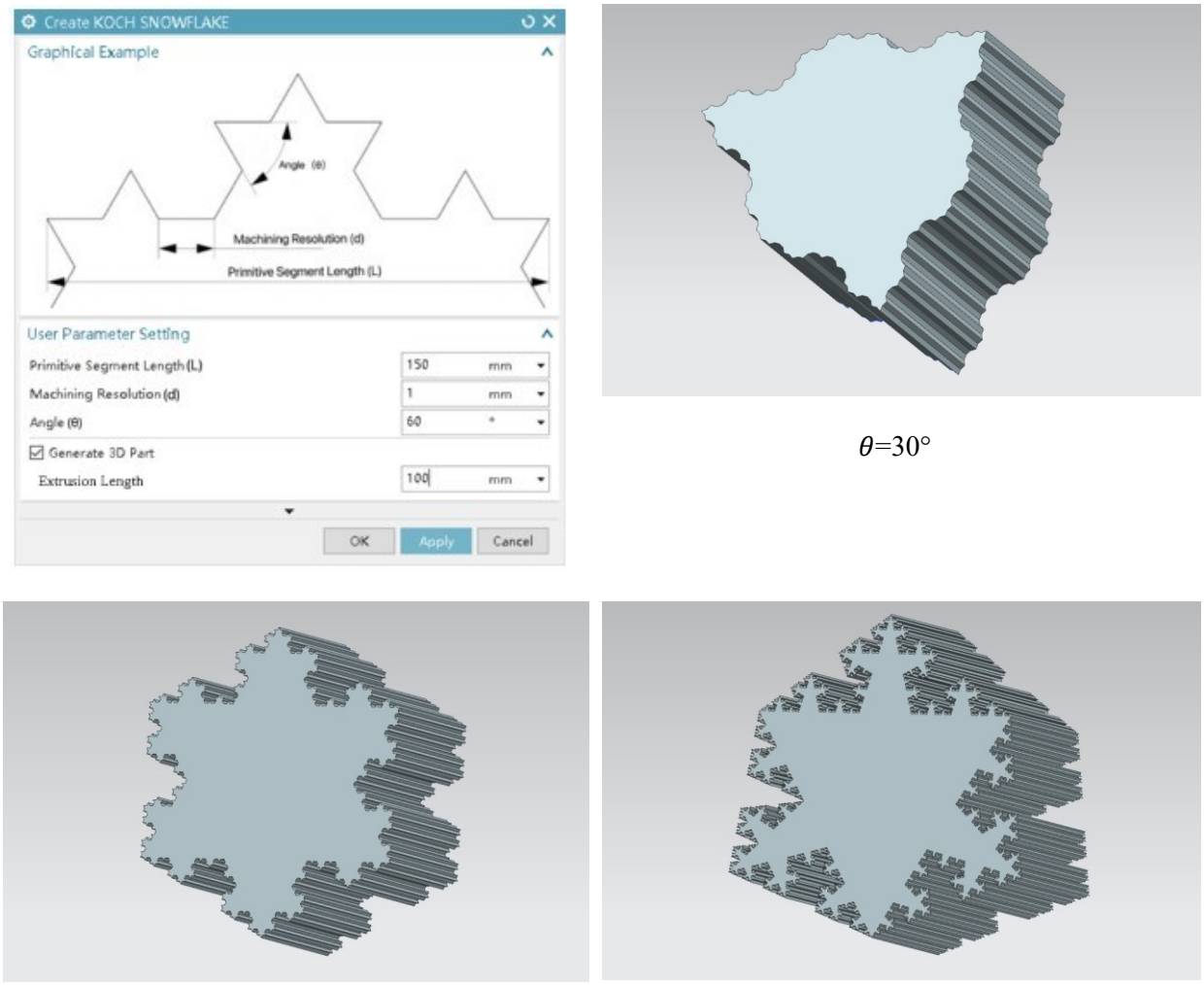
Based on the input values of L, d, and θ, the maximum number of iterations is calculated in the background according to equation (3.3). To be more specific, the N_{max}, in this case, can be calculated by:

$$N_{max} = floor\left(\frac{log\left(\frac{L}{d}\right)}{log[2(1+cos\theta)]}\right) \tag{3.19}$$

Then, the IGM is applied to construct the model iteratively and the general process is described in Figure 3.10. A loop of N_{max} iteration cycles for the model generation is executed. In each iteration cycle, a Koch generator for the current iteration is constructed according to the input angle θ and is then applied to each line segment of the snowflake model constructed in the previous iteration cycle to construct a new snowflake model for the current iteration. It can be seen that in each iteration cycle, mappings are built for each parent line segment to generate 4 connected children line segments with a contraction factor of $\frac{1}{2(1+cos\theta)}$.

Finally, relevant validators are enabled to check whether the sketch curve is closed and whether the constructed model has self-intersection or self-overlapping issues. A warning may appear if the above issues happen, which will remind the user to adjust the input parameters for the redesign process. It is noteworthy that the values of these design parameters in the customized UI are rollback editable and a new model will be constructed and overwrite the previous one if any of the values are changed.

Figure 3.14 shows three extruded Koch Snowflake CAD models. The primitive segment length L is set as 150 mm, the machining resolution d is set as 1 mm, and the extrusion length is set as 100 mm for extrusion. Only the parameter angle θ is different among three models, which are 30°, 60°, and 75° respectively. Apart from the difference in the indentation angle θ, the three constructed models also have different geometric complexities because the values of their maximum number of iterations N_{max} are calculated to be 3, 4, and 5 according to the equation (3.19).



θ=30°

θ=60°

θ=75°

Figure 3.14 Extruded Koch Snowflake CAD models with L=150 mm, d=1 mm and different indentation angles θ as 30°, 60° and 75°.

It can be seen that if one design parameter has been changed, many associated/derived parameters or properties of the geometry will change accordingly, which will significantly affect the shape of the final constructed model. Thus, the various combinations of the extracted design parameters will guarantee that my snowflake constructor has sufficient design flexibility. For all fractal CAD models in Figure 3.14, the total time for background calculation and final model construction for each model is within 20 seconds for most contemporary desktop computers. These models contain 192, 768, and 3072 edges respectively on their end faces. The pseudo-code of the

snowflake creator is in the Appendix. It is also noteworthy that the methods of design parameters extraction are not unique and there are many other parameters and combinations for modeling a variable fractal-like CAD model.

3.4.2 Applications of IGM to construct dynamic fractal generators and CAD snowflake feature

In the present case, IGM is applied as a built-in method for the CAD snowflake feature. In this subsection, the construction method for this CAD snowflake model is introduced. For an understandable demonstration, three defined classes in my plug-in are introduced, including *Directed line segment*, *Dynamic parametric generator*, and *Snowflake initiator* (see Figure 3.15).

The *Directed line segment* class (see Figure 3.15 top) is defined to construct and represent the constitutive geometric element for the sketch of the CAD snowflake model. The snowflake sketch is built on the XY plane so the *Directed line segment* and its start and end points *StartPoint* and *EndPoint* are 2D elements with coordinates (*x*,*y*). Here, l_i^k is denoted as the *i*th directed line segment for the *k*th iteration. Apart from the start point and the end point, this class also has attributes of length, orientation angle, and next connected line. The length attribute, *Length*, represents the length of the directed line segment, which is the Euclidean distance between its start point and end point. The orientation angle, *Angle*, means the orientation angle of this directed line segment with respect to the XY plane of the global coordinate system and the range is regularized between [$-\pi$, π]. The *Next connected line*, **next*, is a pointer for class *Directed line segment*, which points to the address of the next connected instance of *Directed line segment*. In the *k*th iteration, the *i*th line segment and the *i+1*th line segment are connected in sequential order, thus, $l_i^k -> next = l_{i+1}^k$. In this way, a sequenced list of the *Directed line segment* can be built to represent a

snowflake sketch model and relevant traversal algorithms can be designed. Also, this class has three built-in methods, which are *CreateLine(), GetLineProperties(), and EditLineAttributes()*. *CreateLine()* constructs an instance of a *Directed line segment* with the input of the point elements *StartPoint* and *EndPoint*. *GetLineProperties()* calculates the derived attributes *Length* and *Angle*. *EditLineAttributes()* is a method for editing the attributes of a constructed *Directed line segment* instance, such as changing the position of its *StartPoint* and *EndPoint* and the connectivity relationships of its next instance.

The class *Dynamic parametric generator* (see Figure 3.15 middle) is defined to generate the CAD snowflake model iteratively, which creates a dynamic mapping from a parent *Directed line segment* instance as its *objective line segment* from the model of the last iteration to four connected *Directed line segment* children instances in the current iteration. From Figure 3.15 top to middle, the directed line segment l_i^k is mapped into $\{l_{4i-3}^{k+1}, l_{4i-2}^{k+1}, l_{4i-1}^{k+1}, l_{4i}^{k+1}\}$ during the current *k+1*th iteration by the generator, which means that the *i*th line segment for the model in the *k*th iteration is now used to generate the *4i-3*th, *4i-2*th, *4i-1*th, *4i*th line segment for the model of current *k+1*th iteration. The *Dynamic parametric generator* has a *Current indentation angle* as θ^{k+1}, whose value can be defined/edited by end user to ensure design flexibility and avoid self-intersection/sell-overlapping issues of the model if redesign processes are necessary. The relevant associative relationships are built by the generator, which are as follows:

For line segment l_{4i-3}^{k+1}, the *StartPoint* and the *Angle* attributes are the same as l_i^k, and its *Length* has been contracted by the scaling factor of $\frac{1}{2(1+cos\theta^{k+1})}$ according to the equation (3.18). Also, the next connected line segment is l_{4i-2}^{k+1}.

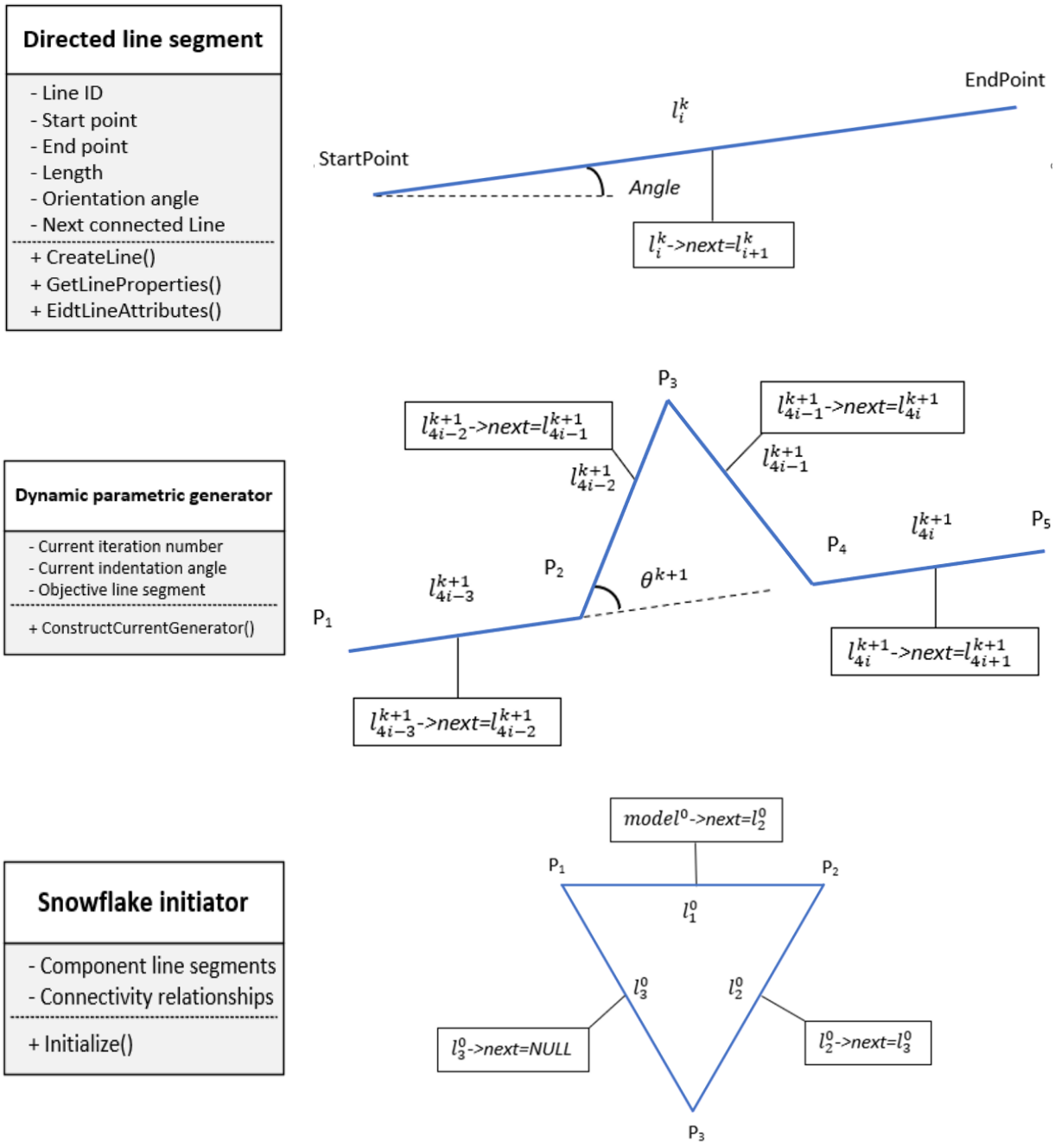


Figure 3.15 Three defined classes: Directed line segment (top), Dynamic parametric generator (middle), Snowflake initiator (bottom).

From the above associativities, the following equations can be derived:

$$
\begin{aligned}
&l_{4i-3}^{k+1}->StartPoint=l_i^k->StartPoint\\
&l_{4i-3}^{k+1}->Angle=l_i^k->Angle\\
&l_{4i-3}^{k+1}->Length=\frac{1}{2(1+cos\theta^{k+1})}\times l_i^k->Length\\
&l_{4i-3}^{k+1}->next=l_{4i-2}^{k+1}
\end{aligned}
\tag{3.20}
$$

Then, an incremental method is applied to construct line segment l_{4i-2}^{k+1}, $l_{4i-1}^{k+1}, and\ l_{4i}^{k+1}$.

For their length and connectivity, the associative relationships remain the same, which are:

$$
\begin{aligned}
&l_{4i-n+2}^{k+1}->StartPoint=l_{4i-n+1}^{k+1}->EndPoint\\
&l_{4i-n+2}^{k+1}->Length=l_{4i-n+1}^{k+1}->Length\\
&l_{4i-n+2}^{k+1}->next=l_{4i-n+3}^{k+1}
\end{aligned}
\tag{3.21}
$$

where *n* takes the value of 4, 3, and 2 in turn to construct line segment l_{4i-2}^{k+1}, $l_{4i-1}^{k+1}, and\ l_{4i}^{k+1}$ in order.

For the orientation angle of these children line segments, the incremental relationships depend on the *Current indentation angle* θ^{k+1} of the generator, which can be expressed as:

$$
\begin{aligned}
&l_{4i-2}^{k+1}->Angle=rectify(l_{4i-3}^{k+1}->Angle+\theta^{k+1})\\
&l_{4i-1}^{k+1}->Angle=rectify(l_{4i-2}^{k+1}->Angle-2\times\theta^{k+1})\\
&l_{4i-2}^{k+1}->Angle=rectify(l_{4i-3}^{k+1}->Angle+\theta^{k+1})
\end{aligned}
\tag{3.22}
$$

where the function $rectify(\bullet)$ operates on an angle and will regularize its range to [$-\pi$, π]

Based on the above associative relationships, the generator will be applied to each line segment of the previous model and the model of the current iteration will be constructed.

The class *Snowflake initiator* (see Figure 3.15 bottom) is defined to initialize an original snowflake sketch, which is an equilateral triangle in this case. *Directed Line Segment* is used as its

component element and pointers are used to construct and represent their connectivity. It is noteworthy that during the initialization process of this case, the first element and the last element need to be identified. Here, the initial address of the initialized sketch is named $model^0$, where *0* means it is an initialized model before iteration, which is also the identifier of the first element. For the last element of the initialized sketch, the pointer to the next is set as NULL. Then, for the model constructed in iteration *k*, the initial element/ initial address of the model is identified as $model^k$, and the last element is identified with the pointer to the next as NULL. In this way, traversal algorithms can be designed for desired operations of the model.

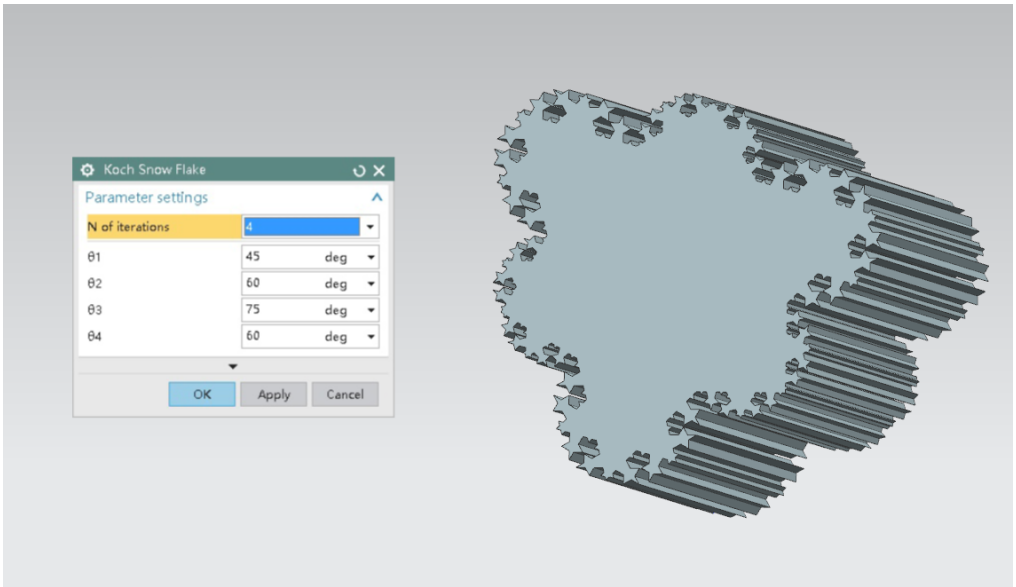


Figure 3.16 An extruded CAD snowflake model with different indentation angles for different iterations: θ^1=45°, θ^2=60°, θ^3=75°, θ^4=60°.

Figure 3.16 shows a model of 4 iteration steps created with the indentation angles of θ^1=45°, θ^2=60°, θ^3=75°, θ^4=60°, respectively, for each iteration step. It is noteworthy that the IGM allows the parameters for generators of each iteration step to take different values and efficient interactions between end users and the CAD system is available by setting/editing parameters for fractal generators. When the user inputs the values of the parameters, the built-in

IGM will select the type of geometric entities and calculate the positions in the background with the relevant construction algorithm, and then the CAD fractal model will be constructed automatically. The users do not need to have advanced knowledge of how the fractal-like CAD model is constructed.

A detailed comparison of performance improvement with my proposed method is made for the model construction of Figure 3.16. In Figure 3.16, the fractal-like CAD model contains 768 edges in each of its end faces. A desktop computer with a common configuration is used for the model construction, which has the 10th Gen Intel® Core™ i5-10400 processor, 16 GB DDR4 RAM of 2666 MHz, Windows 10 operating system, and CAD software of Siemens NX 12. Six volunteer CAD designers are invited to model this fractal-like CAD model with the same desktop computer. All of them have at least a BSc degree in Mechanical Engineering and over three years of experience in using NX in product design. The invited CAD designers are divided into two groups, three people each, with/without advanced experience in fractal modeling in CAD. The invited designers are allowed to use all available model construction methods in CAD. The average time used for CAD designers with/without fractal modeling experience is 1560 seconds and 2880 seconds, respectively. With my proposed method, the automatic construction of the same model takes only 6 seconds on average. This method is found to be of significant improvement in the modeling efficiency of fractal-like CAD models. Therefore, the integration of computational geometry algorithms with my proposed IGM in the CAD fractal feature plug-in is found to be of sufficient efficiency for design automation. My new method may have even greater potential for performance improvement for modeling more complex fractal-like CAD models, however, I was not able to find CAD designers who are willing to take a long time to model more complex CAD models as volunteers for comparison.

3.4.3 Multilevel representation of a CAD fractal model with different geometric complexities

As described in subsection 3.3, the CAD fractal model is constructed iteratively. In each iteration, a model instance is constructed and stored in the database. The user can then select one stored model with respect to a certain iteration number to display, edit and postprocess. It is common to use CAD models with different levels of details and geometric complexities among different product development stages. For example, in the numerical simulation stage, the CAD model should not be too complex or contain too many unnecessary details, a simplified model might be used or defeaturing processes might be implemented. Thus, the multilevel representation of a CAD fractal model makes it robust and widely used.

Figure 3.17 shows two sketches of a traditional Koch snowflake model with respect to different iteration numbers and geometric complexities. The primitive line segment is set to 100 mm, and the maximum iteration number, N_{max} is set to 6. The indentation angle θ remains constant and is set to 60°. Then, the snowflake sketches for iteration step N= 1-6 are constructed and stored. The end user can select the iteration level of the sketch to display, edit, and post-process from the customized UI and then the current sketch properties, such as current segment length will be shown. In Figure 3.17, the sketches of the snowflake for the 3rd iteration and 5th iteration are selected to be shown as an illustration, which contain 192 and 3072 line segments, respectively. It is obvious that the number of iterations will significantly influence the geometry complexity of the CAD fractal model, so the multilevel representation of the model can save significant time and computational cost when the detailed design for the model is not necessarily needed.

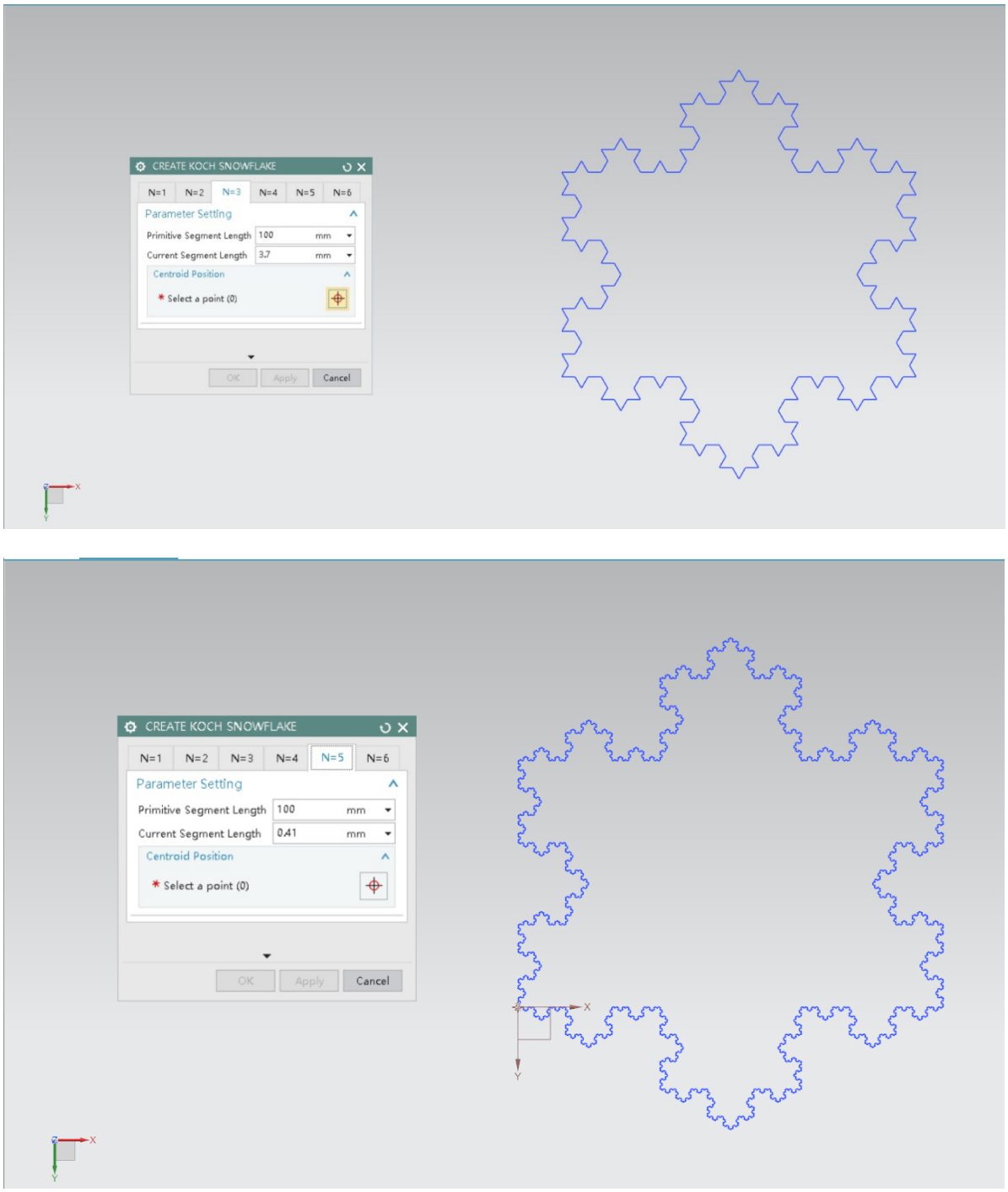


Figure 3.17 Multilevel representation of a Koch snowflake sketch: Primitive Segment Length=100 mm, *Nmax*=6. Sketches of *N*=3 (top) and *N*=5 (bottom) are displayed.

3.4.4 Operability of the constructed CAD fractal model

In this research, the fractal model is constructed in a CAD system. The constructed model is a CAD part model, which has rich information and is eligible for most CAD operations. The part model can also be added in an assembly environment and assembly features can be created.

Figure 3.18 shows a Koch Snowflake CAD part model (left) and its assembly (right) with other components. For the part model (Figure 3.18 left), a through hole is made and the edges of the end faces are chamfered, which demonstrates the proposed CAD fractal feature can interact with other CAD features. Reference geometries, such as reference points and reference planes can be created based on the model, and Boolean operations are available. Non-geometric attributes can also be added/deleted/edited, such as color and material. The part model can also be added to the assembly environment and assembly features with other parts can be added/deleted/edited. In the assembly model (Figure 3.18 right), coaxial and symmetric assembly features are created between the snowflake part and the shaft part, which demonstrates the usability of my CAD fractal model in an assembly environment. The CAD fractal model can also be exported and reused in other environments, such as CAM and CAE.

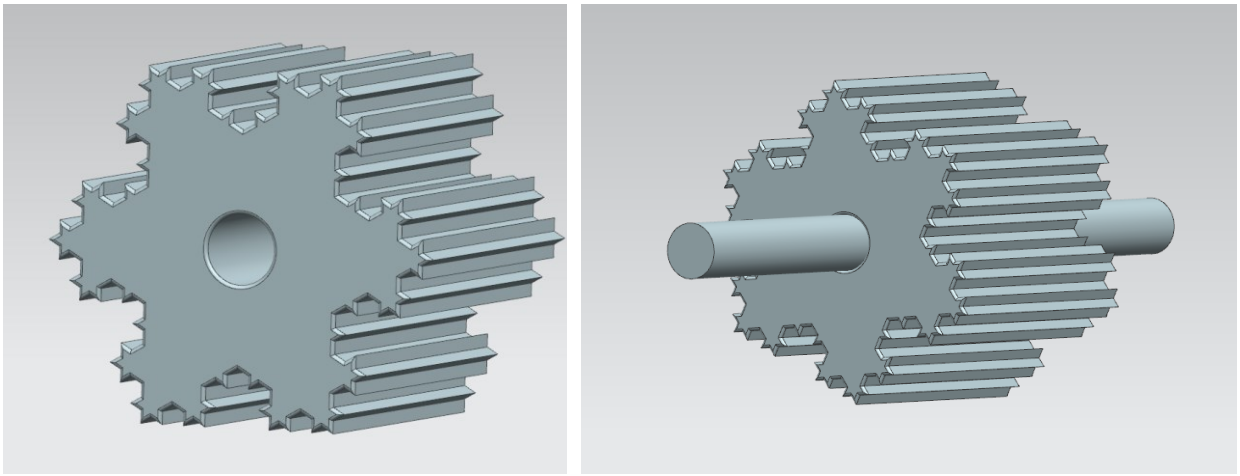


Figure 3.18 Operability of the constructed snowflake model: Part (left), Assembly (Right).

3.4.5 Potential application of this case study

This subsection gives an example of the potential application value of my case study in research and industry. A Koch-based fractal metasurface antenna was proposed and studied by Liu et. al. [131]. A brief overview of the application research is as below.

Wideband antennas have become more and more popular recently in lots of fields such as wireless communication, radar systems, and satellite navigation [131]. In advanced antenna design, Koch curve has been used to develop wideband, multiband, or miniaturized antennas [132–135]. Liu et. al. [131] proposed and successfully implemented a novel wideband slot antenna with Koch fractal metasurface structure as a boundary. The iteration factor (IF), iteration angle (IA), and iteration order (IO) were used as design variables for testing the performance of the fractal antennas. These variables were scaling factor ε, indentation angle θ, and level of iteration N in this case study.

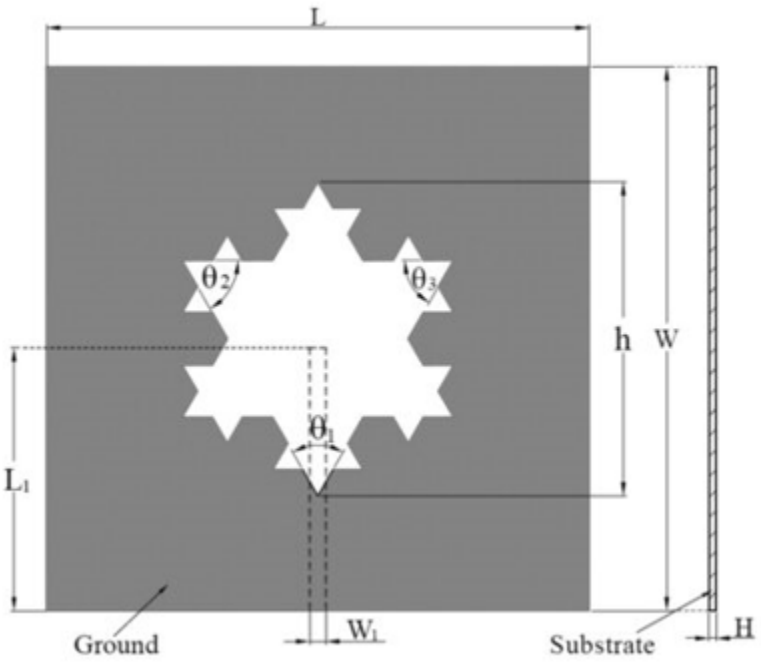


Figure 3.19 Geometry of the proposed fractal metasurface antenna by [131]

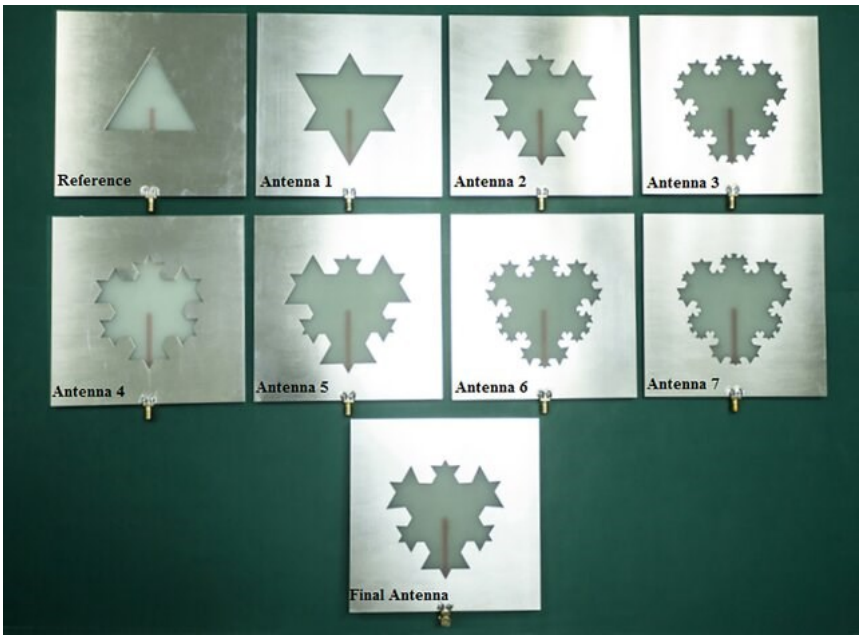


Figure 3.20 Photographs of the fabricated fractal metasurface antennas [131]

Table 3.1 Performance comparisons between the proposed fractal metasurface antennas with different values of parameters [131]

Antenna	IO	IF	IA of three Koch edges	Frequency range (GHz)	BW (GHz)
Reference	1	0	0°, 0°, 0°	1.78-2.46	0.68
Antenna 1	1	0.45	60°, 60°, 60°	1.63-4.78	3.15
Antenna 2	2	0.45	60°, 60°, 60°	1.48-4.81	3.33
Antenna 3	3	0.45	60°, 60°, 60°	1.01-3.98	2.97
Antenna 4	2	0.4	60°, 60°, 60°	1.36-4.49	3.13
Antenna 5	2	0.5	60°, 60°, 60°	1.22-2.19, 2.81-5.01	3.17
Antenna 6	3	0.45	55°, 55°, 55°	0.98-4.02	3.04
Antenna 7	3	0.45	65°, 65°, 65°	1.23-4.26	3.03
Final Antenna	2	0.45	60°, 55°, 55°	1.45-4.86	3.41

Figure 3.19 shows the geometry of the proposed fractal metasurface antenna. The indentation angle, scaling factor, and level of iteration were design variables. Figure 3.20 shows the photographs of the fabricated antennas and Table 3.1 shows their performance, frequency range and bandwidth (BW), with respect to different combination values of these parameters. It can be

seen that different values of parameters will lead to different shapes of the metasurface antennas, and the performance will then vary from the shapes.

The case study of my research has good potential to be applied in advanced antenna design because my proposed method can realize automatic model construction for Koch-like fractal objects with variable design parameters. Additionally, the constructed parametric CAD models in my proposed method can be directly used in CAM for tool-path generation of the metasurfaces, which can be used as antennas with the required performance. My method can significantly reduce the product development time for advanced antenna design. A mapping between design parameters, product shapes, and functional behaviors can be built.

3.5 Discussion

In this research, a method for automatically modeling parametric fractal or fractal-like geometry objects in the CAD system is proposed. Previously, fractal objects were mainly modeled in computer graphics for visual representation, the parameters of the fractal model were fixed, and fractal generators were statically constructed and non-editable. In traditional CAD systems, the CAD models were not constructed iteratively and the fractal or fractal-like geometric objects were built manually in most cases. Now, with my proposed method, fractal or fractal-like geometric objects can be constructed automatically in the CAD system. Design parameters can now be extracted from the fractal model and fractal generators are constructed dynamically for each iteration cycle. As a result, the design parameters of the fractal or fractal-like objects can be variable and editable. The CAD fractal model is constructed iteratively and can be stored, represented, and edited in multilevel geometric complexities. The model is built as a feature-based CAD part model, which has rich information and is eligible for most CAD operations. The model can also be directly imported to other computer-aided environments for further use, such as in

CAM for tool-path generation. The proposed method has been proven in the case study for its sufficient efficiency and design flexibility, where an extruded Snowflake CAD model with variable design parameters has been built automatically.

The time complexity of my proposed method in constructing a variable Koch snowflake is O(4^N), and this complexity belongs to an exponential algorithm O(2^n).Unfortunately, it is inevitable to use an exponential algorithm for modeling any fractal objects because of its intrinsic fractal characteristics. Compared with building a computer graphics model of a fractal object for visual representation, my proposed method has the same time complexity for the built-in algorithm, but the constructed models are parametric CAD models, which can be easily modified and used in many different stages of product development process. Although the exponential algorithms usually have a very long running time when the level of iteration N is high, it can stay at an acceptable running time if the level of iteration N is under control. The practical machining resolution in manufacturing can control the maximum level of iteration N_{max}, and the actual level of iteration N will not be more than 6 in most cases in the CAD model construction processes. With the designed UI, the machining resolution has been considered as one of the input parameters, and this guaranteed my proposed method in actual CAD model construction processes would not reach a very high level of iteration. The CAD model construction time for a fractal object in my proposed method has been proven to be acceptable in most cases when machining resolution has been considered (usually within 2 minutes).

3.6 Conclusion

3.6.1 Summary and contribution of this research

This research developed an automatic model construction method that employs an algorithmic approach to iteratively construct fractal-like models in CAD systems. With my proposed method, the design processes of fractal objects in CAD systems can now be algorithmically and iteratively. As a result, the efficiency and convenience of constructing and modifying fractal-like models in CAD systems have been revolutionarily changed. The feature-based fractal models in this research have better compatibility, reusability, and editability for product development. Such models are preferred in industry over computer graphics models because they can be directly used in CAM tool-path generation. Then, new product development time with self-similarity geometric features will be reduced dramatically. A new type of feature, named CAD fractal feature, was proposed to bridge the gap between CAD and fractal geometry. The feature-based modeling method was used, and the prototype can parametrically and automatically construct a fractal or fractal-like object in a CAD system. A key built-in method of the CAD fractal feature, the iterative generation method (IGM) was proposed as an extension of Iterated Function System (IFS) to construct CAD fractal models with extracted variable design parameters with a developed fractal generator, where different parameter values for different iteration cycles can be edited by end users. The algorithms for constructing this complex computational geometry have been integrated into a CAD system with my proposed IGM, so the designers do not need advanced knowledge of fractal geometry to efficiently and conveniently build a fractal CAD model or a complex CAD model with self-similarity features. The proposed approach was proved in the case study, which was found to be of sufficient efficiency and design flexibility for design automation and automatic CAD model construction of complex geometry

objects with associated inner geometric relationships. For example, an extruded Koch Snowflake CAD model with parametric geometric variants was built in Siemens NX 12 for demonstration. The novelty of this work is the demonstration of an algorithmic approach employed to iteratively construct geometrical entities within a 3D modeling software environment and the built-in dynamic feature generation that is self-contained and modularized, such that a reusable parametric generator for complex geometry types can be embedded into a convenient toolkit. This work is an extension of the generic feature-based modeling [106,136] method. The scientific value of this research lies in the practical availability of fractal definitions in new product development, where CAM tool-path generation can be conveniently carried out. From the perspective of industrial management and applications, the dramatic improvement in efficiency will lead to the broader use of such fractal geometry designs.

3.6.2 Limitations and future work

It should be noted that the proposed method currently requires the fractal-like geometry objects to be self-connected and self-intersection/self-overlapping is not allowed to appear for any iteration. As a result, my method may not be suitable for constructing some types of fractal objects that may be disconnected or have self-intersection/self-overlapping geometric features when the iteration level increases. To automatically construct CAD fractal models in a broader scope with less geometric requirements, intelligent algorithms to split disconnected fractal objects and properly merge/unite self-intersect/self-overlapped fractal models in CAD systems are expected to be developed and integrated with my proposed method in future research.

Chapter 4 A Feature-based Automatic Model Construction Method in CAD for Material Distribution Structures

4.1 Chapter overview

The material distribution structure has been widely applied in research and industry, where the entire domain of the macrostructure has been equally or non-equally divided into several subdomains and each subdomain is regarded as a design space for local material distribution in the microstructure. In structural design and optimization, material distribution structures are used to represent material allocation in space, and the final optimization result is an optimum arrangement of materials within a structural system to achieve the desired performance while minimizing weight. The optimized material distribution structures usually have complex geometry characteristics, and it is repetitive, tedious, and time-consuming to manually construct parametric CAD models after getting the structural optimization result.

The research of this chapter aims to improve the modeling efficiency for material distribution structures in CAD systems. An automatic model construction method was proposed, and feature-based modeling method was adopted. In the proposed method, there were three sub-steps to realize the automatic CAD model construction. In the case study, a parametric CAD model of a topology-optimized cantilever was automatically constructed in CAD system with my proposed method. My method was proven to be of sufficient efficiency for CAD model construction, and the constructed model is editable for each elemental component, which has sufficient flexibility for further model modifications.

The rest of this chapter is organized as below. Subsection 4.2 gives an introduction to this research, where the background, gaps, and motivations of this research are introduced. The related

literature to this research has been reviewed in Chapter 2, including material distribution structures, contemporary CAD systems, feature-based modeling method, and topology optimization in subsections 2.2, 2.3, 2.4, and 2.5. Subsection 4.3 gives a conceptual framework of the proposed method. In this subsection, the proposed method is represented in a flowchart, and three constituent sub-steps are explained in detail. A case study of automatic model construction of a topology-optimized cantilever beam in a widely used CAD system, Siemens NX 12, is demonstrated in subsection 4.4. Subsection 4.5 concludes the research of this chapter.

4.2 Introduction

The material distribution structure has been widely applied in structural design and optimization in research and industry. Over the past few decades, the data of material distribution structures were stored in a 2D or 3D matrix as the physical density field and such data can be plotted in software with matrix visualization functionality such as Matlab [69]. The design domain was discretized into small squares or cubes as basic elements and a marching cube algorithm [73] was designed for the plot. In this way, the visualization of material distribution structure models was realized. Then, an STL writer was applied to the matrix, where the matrix data was post-processed and the model was converted to STL format, which is friendly to manufacturing such as 3D printing [137]. However, for structural behavior analysis such as finite element analysis and thermal expansion simulation, the data and the model are not sufficient in numerical simulation environments. First, a good CAD geometry file is preferable in a simulation system, where a model with solid geometry will be much more compatible to be imported, simulated, and exported than the tessellation model. Second, the structure geometry might be further modified based on the simulation results. Although some software provides users with auxiliary model conversion functions which may help to convert an STL file to a CAD geometry model, the converted model

is not feature-based, which makes it not well-parameterized for model modification and not compatible with some CAD post-processing operations such as blending and chamfering. Therefore, a feature-based parametric CAD model of a material distribution structure is preferred in the numerical simulation environment for structure behavior analysis and further modifications of the model geometry. To the authors' best knowledge, such parametric CAD models are still constructed manually in most cases, which is tedious and time-consuming.

Considering manufacturability constraints [138] and periodic boundary conditions [139], the geometries of microstructure design are usually parameterized and may have repetitive features in global spatial distribution for the macrostructure. Therefore, it might be possible to take advantage of the associative relationships of spatial allocation positions among elemental component microstructures to realize automatic model construction for material distribution structures in CAD systems.

In the research of this chapter, an automatic model construction method for a feature-based parametric CAD model of the material distribution structure was proposed. The proposed method aims to improve the CAD modeling efficiency of a complex structure geometry with parameterized and spatial repetitive features of its microstructure geometry elements. The constructed CAD model is parametric, and every component microstructure is editable for its design parameters. The model is also well-compatible with a numerical simulation environment and further geometry modifications are easy to be implemented.

4.3 Conceptual framework

This subsection introduces the conceptual framework of the proposed automatic modeling method for material distribution structures in CAD systems. First, the overall proposed modeling

processes for material distribution structures are presented as a flowchart. Then, three constitutive sub-steps are introduced in detail.

4.3.1 Overall modeling processes of a material distribution structure in CAD systems

The proposed model construction method for material distribution structures can be divided into 3 sub-steps, which are shown below.

(1) Model data file creation.

(2) Microstructure element categorization, parameterization, encapsulation, and integration.

(3) Automatic CAD model construction.

Figure 4.1 shows the flowchart of the proposed modeling processes. In my proposed method, the modeling processes can be realized in the following steps. First, the data of the material distribution structure is mapped in a new format and a data file is created to store the information. This data file should be directly read by CAD systems and guide the automatic model construction processes. Second, the microstructure geometry elements are categorized into several subtypes according to their geometric characteristics and are properly parameterized, encapsulated, and integrated. A microstructure construction function is defined and integrated into a CAD system to automatically construct a microstructure model with required input parameters. Finally, a plug-in of a CAD system is developed to read the data file and activate the integrated microstructure construction function to automatically construct the CAD model. Figure 4.1 shows the relationship among the 3 constitutive sub-steps. The output of sub-step 1 is a data file, which is used as the input data for the developed CAD plug-in sub-step 3. The result from sub-step 2 is a user-defined microstructure construction function, which is activated in the developed CAD plug-in in sub-step 3 to automatically construct each microstructure element for the global macrostructure.

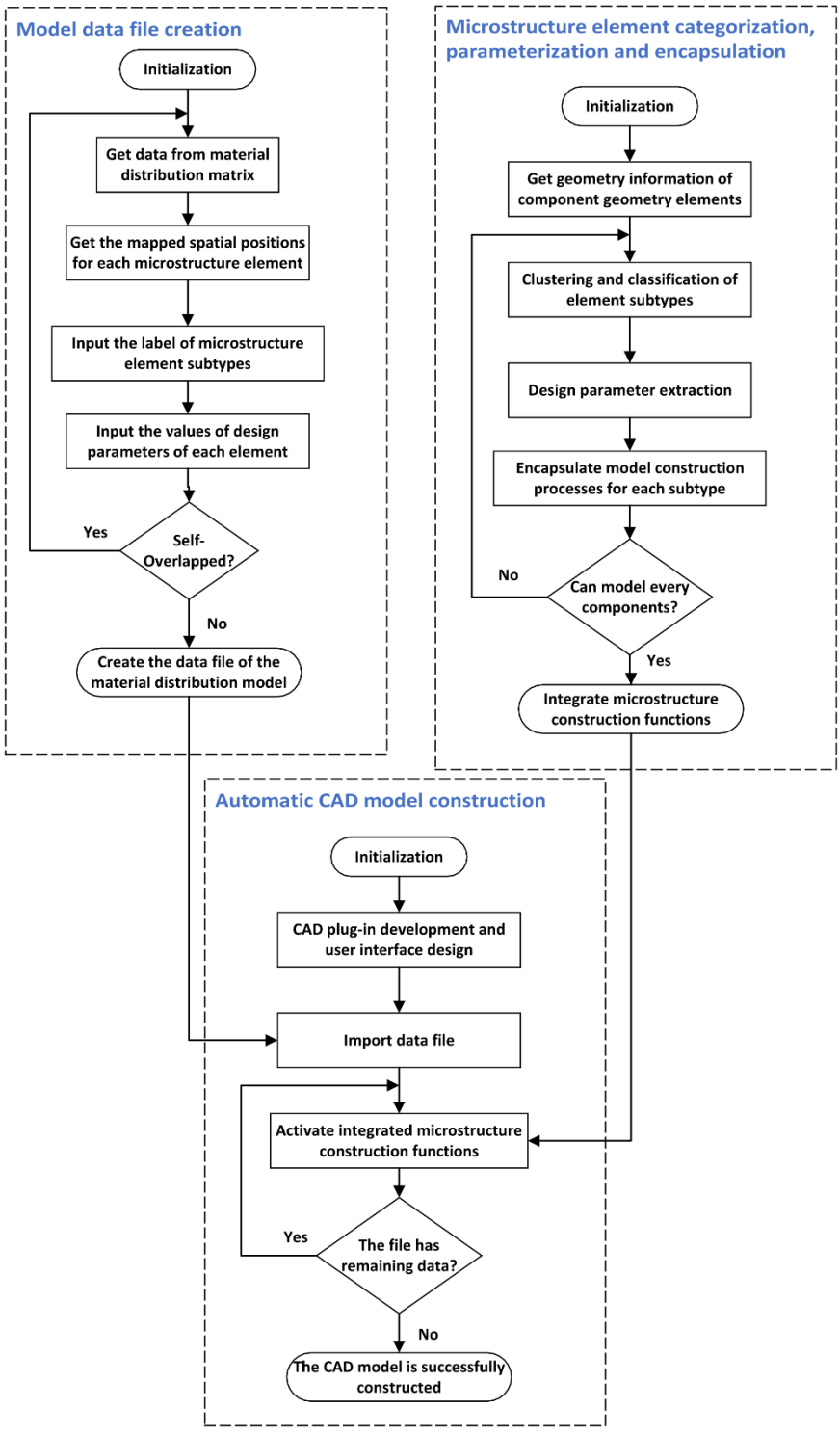


Figure 4.1 Flow chart of the proposed modeling processes of a material distribution structure

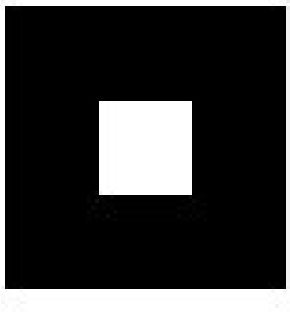
4.3.2 Model data representation and data file creation (sub-step 1)

As mentioned in subsection 2.2, the traditional data representation of a material distribution structure was a 2D or 3D matrix. The matrix was a binary matrix with all the elements taking the value of either 0 or 1. Usually, the value 0 means the local region should be left void without being filled with material, and value 1 means the local region should be filled with material.

The traditional data representation was simple, and the modeling process was of high efficiency. However, microstructure elements were not parameterized, and their geometries would be fixed after being defined. To improve the design flexibility of the constructed model, the component microstructure elements are classified into many subtypes according to their geometry feature, and each subtype is parameterized with its unique design variables. In my proposed method, the material distribution model is mapped by a material distribution list with the information on spatial distribution, microstructure element subtype, and relevant parameter list of the related design variables for each of the component microstructure elements. The new data format requires more storage space because more parameters are included. However, compared with the size of the model, such an increase in storage resource consumption is negligible.

Here, let us take an example of the model of a material distribution structure named Sierpinski Carpet (see Figure 2.5). For a better illustration, Figure 4.2 extracts the 1st iteration of the model and its data representation in the form of a binary matrix. Figure 4.2 (a) is the model of the structure. The design domain of the structure is equally divided into 9 elements and each element is a square region. Figure 4.2 (b) is the material distribution matrix of the structure, where the 0 element in the middle means there's no material allocated in the central element region for the structure. In this matrix representation method, a CAD system cannot directly read and use the

matrix data to guide a model construction process. Moreover, this representation only allows the structure to be equally divided and each microstructure element is limited to be a solid square with a specific fixed edge length.



(a)

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(b)

Figure 4.2 An example of matrix representation for a material distribution structure: (a) The 1st iteration model of Sierpinski Carpet, (b) The binary matrix representation of the model's data

In my proposed method, the design domain can be non-equally divided, and the component microstructure elements are allowed to be rectangles with variable lengths and widths. The spatial distribution information is extracted from the original matrix data, where the (i, j) element of the original matrix is regarded as a rectangle element with its centroid position of (i-1,j-1) in the 2D Cartesian coordinate system. The material distribution list of the 1st iteration model of Sierpinski Carpet is shown in Table 4.1. A data file is then created and stores such information about the model. Although the proposed data format might not be as simple as the traditional matrix representation, it has good potential to include more information and each local microstructure element can have variable parameters/dimensions.

Table 4.1 Material distribution list for data representation of the 1st iteration model of Sierpinski Carpet

Element Number	*Centroid Position*	*Element Type*	*Length (mm)*	*Width (mm)*
1	(0,0)	Rectangle	1	1
2	(1,0)	Rectangle	1	1
3	(2,0)	Rectangle	1	1
4	(0,1)	Rectangle	1	1
5	(1,1)	Rectangle	Null	Null
6	(2,1)	Rectangle	1	1
7	(0,2)	Rectangle	1	1
8	(1,2)	Rectangle	1	1
9	(2,2)	Rectangle	1	1

4.3.3 Microstructure element categorization, parameterization, encapsulation, and integration (sub-step 2)

In the structural optimization process, multi-variable hybrid geometric elements might be used as candidates to obtain the final material distribution structures. The component microstructure elements are categorized into many subtypes according to their geometry characteristics and each subtype is parameterized with its unique design variables. For example, Figure 4.3 shows three subtypes of basic 3D geometry elements, a block, a sphere and a honeycomb cell, and Table 4.2 shows their design parameters respectively. For each type of geometry element, an associative parameter list is encapsulated as the input parameter set for a microstructure model construction, and these design parameters are initialized with default values if not input by the user to improve the robustness of the model construction algorithm. Finally, the construction functions of all the subtypes are integrated as a function named

BuildAMicrostructure() with input data as follows: *BuildAMicrostructure(Element Type, Centroid Position, Sketch Plane, Design Parameter List)*.

(a) (b) (c)

Figure 4.3 Three subtypes of basic 3D geometry elements: (a) A block, (b) A sphere, and (c) A honeycomb cell

Table 4.2 Design parameters of three geometry element subtypes

Type	*Design Parameters*		
Block	Length(L)	Width(W)	Hight(H)
Sphere	Radius(R)		
Honeycomb cell	Incircle radius(r)	Circumcircle radius(R)	Extruded Length(L)

4.3.4 Automatic CAD model construction (sub-step 3)

A plug-in is developed in a CAD system, which can read the data file of the material distribution list. A traverse algorithm is designed for the CAD system to read the material distribution list line by line. While reading each line of the data file, the *BuildAMicrostructure()* function is activated with the inputs as the current line of the data and a parameterized microstructure CAD model is constructed automatically.

In my proposed method, more information is added and stored which allows the microstructure elements to have more design parameters. The model constructed is a feature-based parametric model, which has better compatibility, editability and design flexibility.

4.4 Case study

In this subsection, the automatic CAD model construction of a topology-optimized 3D cantilever is realized to validate the proposed method. A plug-in named "Create MD Model" is developed in the NX 12 system using NX Open API. Feature-based modeling is used as an object-oriented approach to automatically construct the CAD model of the material distribution structure. A UI was designed in the developed plug-in which can directly read the data file of the proposed material distribution list. In this case study, efficient 3D topology optimization proposed by Liu et al [69] was applied to get the optimized structure, and the optimization output provided the input data of the material distribution structure to be processed and imported into the CAD system. In this subsection, the optimization problem is stated, the specific case is introduced, the result of the final material distribution matrix is plotted, and the automatically constructed CAD model is shown.

4.4.1 Optimization problem formulation: minimum compliance

The design domain is discretized by eight-noded cubic elements (see Figure 4.4 and Figure 4.5). Each node in the structure has three degrees of freedom (DOFs) corresponding to linear displacements in *x*-*y*-*z* directions (one element has 24 DOFs). The degrees of freedom are organized in the nodal displacement vector **U**.

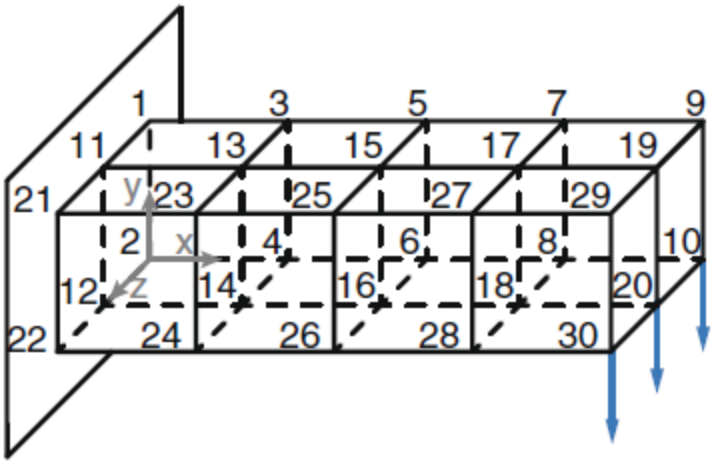


Figure 4.4 Global node IDs in a prismatic structure discretized by eight-noded cubic elements [69]

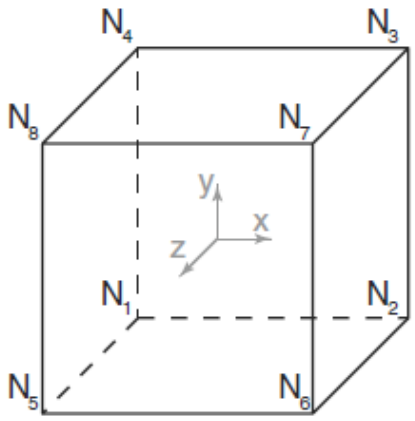


Figure 4.5 Local node numbers within a cubic element

The objective of the minimum compliance problem is to find the material density distribution $\tilde{\mathbf{x}}$ that minimizes the structure's deformation under the prescribed support and loading conditions. The structure's compliance, which provides a global measure of deformation, is defined as [69] :

$$c(\tilde{\mathbf{x}}) = \mathbf{F}^{\mathrm{T}}\mathbf{U}(\tilde{\mathbf{x}}) \tag{4.1}$$

where **F** is the vector of nodal forces and $\mathbf{U}(\tilde{\mathbf{x}})$ is the vector of nodal displacements. Incorporating a volume constraint, the minimum compliance optimization problem is

$$
\begin{aligned}
&\text{find} && \mathbf{x}= [x_1, x_2, \dots, x_e, \dots, x_n]^{\mathrm{T}} \\
&\text{minimize} && c(\tilde{\mathbf{x}}) = \mathbf{F}^{\mathrm{T}}\mathbf{U}(\tilde{\mathbf{x}}) \\
&\text{subject to} && \mathbf{K}(\tilde{\mathbf{x}})\mathbf{U}(\tilde{\mathbf{x}})=\mathbf{F} \\
& && v(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}^{\mathrm{T}}\mathbf{v} - \bar{v} \leq 0 \\
& && \mathbf{x} \in \chi,\ \chi = \{\mathbf{x} \in \mathbf{R}^n : 0 \leq \mathbf{x} \leq 1\}
\end{aligned}
\tag{4.2}
$$

where the physical densities $\tilde{\mathbf{x}}=\tilde{\mathbf{x}}(\mathbf{x})$ are defined by applying a basic density filter function [140] to design variables **x**, n is the number of elements used to discretize the design domain, $\mathbf{v}= [v_1, v_2, \dots, v_e, \dots, v_n]^{\mathrm{T}}$ is a vector of element volume, and $\bar{v}$ is the prescribed volume limit of the design domain. The nodal force vector **F** is independent of the design variables and the nodal displacement vector $\mathbf{U}(\tilde{\mathbf{x}})$ is the solution of $\mathbf{K}(\tilde{\mathbf{x}})\mathbf{U}(\tilde{\mathbf{x}})=\mathbf{F}$, where the tensor **K** is the global stiffness matrix.

4.4.2 Specific case: minimum compliance problem of a cantilevered beam

In this case study, the design domain is a 60mm×20mm×4mm block which is fully constrained in one end, and a unit distributed vertical load is applied downwards on the lower free edge (see Figure 4.6).

The domain is meshed by 1mm size cubes of 60×20×4. The maximum volume fraction is set to be 0.3 and the filter radius is 1.5 element sizes. The structure is a single material structure and Solid Isotropic Material with Penalization (SIMP) method [141] is applied in the topology

optimization processes as the material's Young's modulus interpolation. The Optimality Criteria (OC) [142] is applied as the optimization solver. Figure 4.7 shows the MATLAB plot of the optimized result of the physical density field. The optimized structure has been tested to be mesh independent. Figure 4.8 shows the MATLAB plot of the optimized result of the physical density field for the same domain with refined 0.667mm size cubes of 90×30×6, which is used for mesh independence test. It can be clearly seen that the optimized structures have very similar geometry in Figure 4.7 and Figure 4.8 with different sizes of cubic mesh.

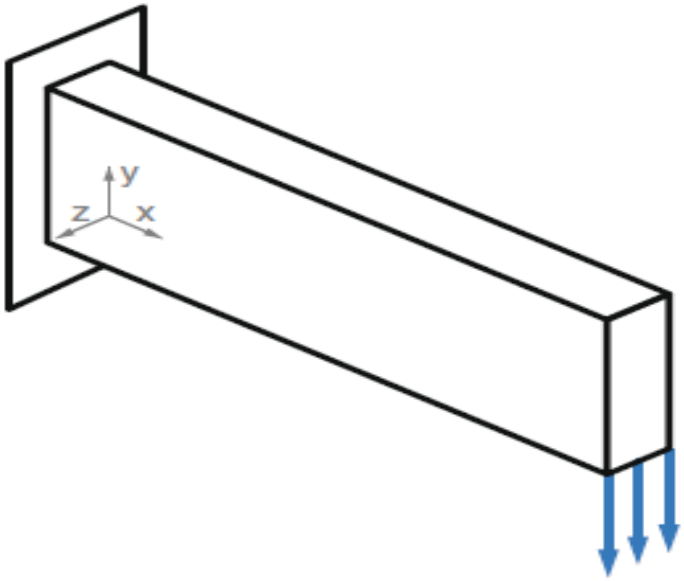


Figure 4.6 Topology optimization of a 3D cantilever beam: Initial design domain

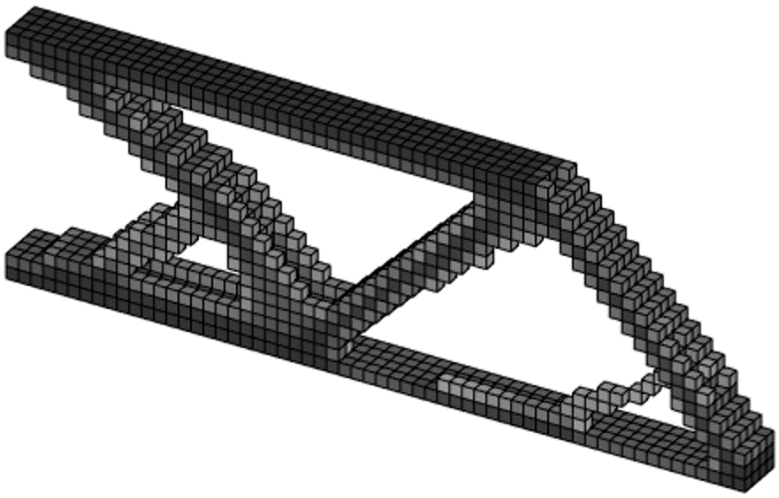


Figure 4.7 Matrix plotted model of the topology optimized beam in Matlab (1mm size cubes of 60×20×4, reproduced from [69])

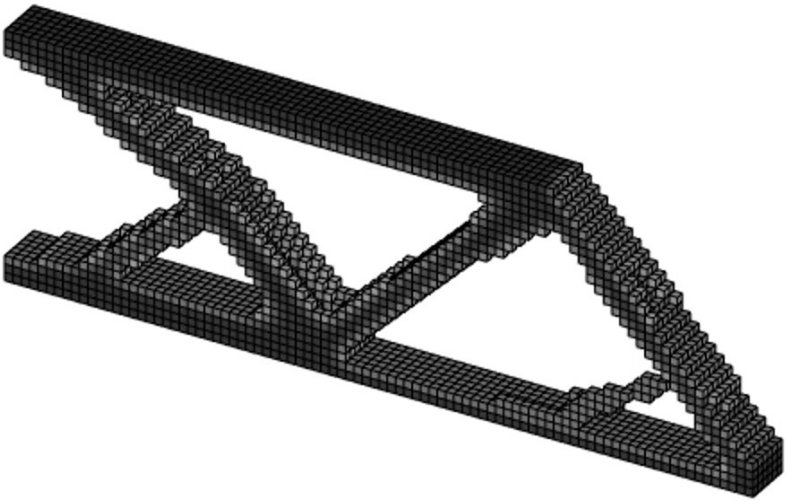


Figure 4.8 Matrix plotted model of the topology optimized beam in Matlab for mesh independence test (0.667mm size cubes of 90×30×6)

4.4.3 Automatic CAD model construction in CAD system: batch modeling mode

In my proposed CAD model construction processes, two modes are realized and optional for users, which are batch modeling and sequential modeling. The batch modeling mode is suitable for CAD model construction of macrostructures which have the same type and same parameters for all microstructure elements. This mode will set the parameter list of the microstructure elements

as default and then only the element index and spatial distribution information is read and passed to the integrated construction function. The batch modeling mode is very efficient but has less flexibility for the initial model construction of the optimized material distribution structures. The sequential modeling mode will be introduced in detail in Chapter 5.

In this case study, all the elements are 1mm size cubes and the batch modeling mode is activated to construct the CAD model automatically. Figure 4.9 is the automatically constructed CAD model of the cantilever beam with my proposed method.

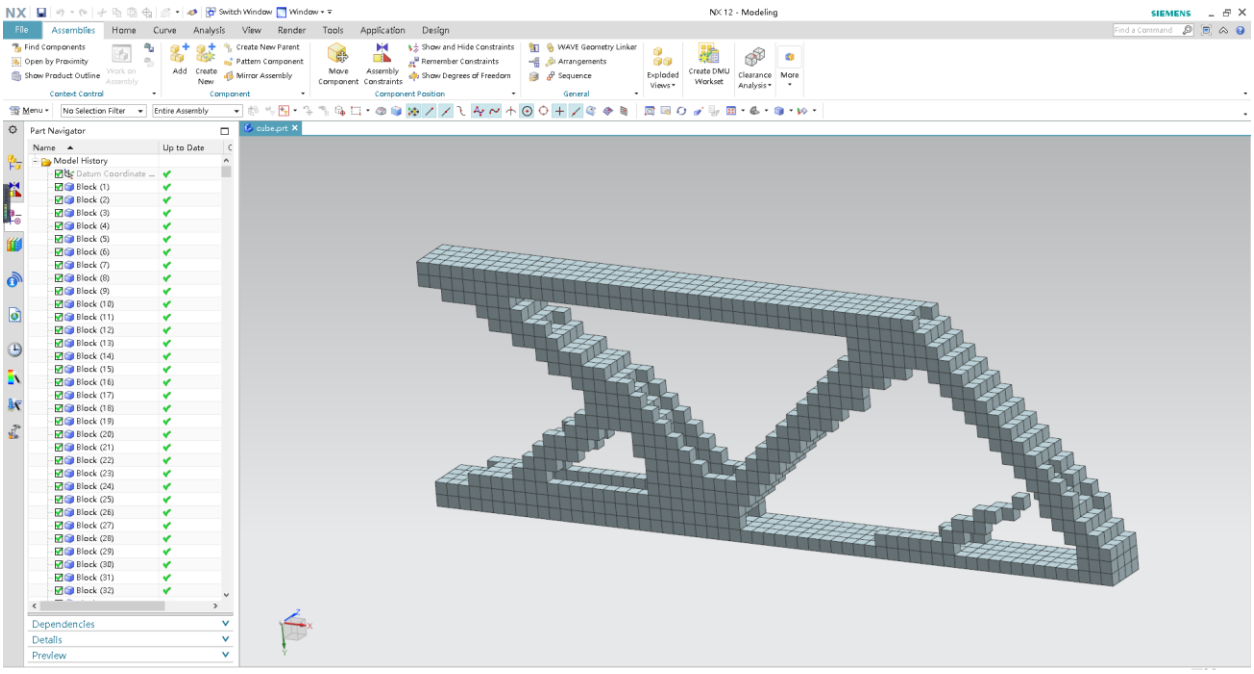


Figure 4.9 Automatic CAD model construction of the optimized cantilever beam

In Figure 4.9, every geometry element of the CAD model is set as a block feature of 1mm×1mm×1mm. The model constructed is a feature-based parametric CAD model with 1434 block features and the parameters (length, width, and height) of each block element can be edited by the user for post-processing. The CAD model is automatically constructed within 45 seconds which demonstrates my proposed method is of sufficient modeling efficiency. Compared with the

Matlab plotted optimized structure, the constructed CAD model has the same geometry which validates my method is applicable.

4.5 Conclusion

4.5.1 Summary and contribution of this research

In this research, an automatic CAD model construction method for material distribution structures is proposed as conceptual framework, developed in a CAD system, and validated by a case study. A new data format is applied for the representation of material distribution structures. Microstructure elements are categorized, parameterized, encapsulated, and integrated in a CAD local microstructure construction function. A plug-in is developed in a CAD system to read the proposed data file and activate the integrated microstructure construction function to realize automatic CAD model construction for the entire material distribution structure. The CAD model constructed is a feature-based parametric model which is compatible with a numerical simulation environment and the model is easy to be further modified. In my case study, the proposed method is validated to be of sufficient design flexibility and modeling efficiency.

4.5.2 Limitations and future work

The research of this chapter focuses on material distribution structures composed of simple geometric microstructure elements. The proposed modeling method has the potential to handle more complicated cases for microstructures with more design parameters. In the future, a library of commonly used parametric geometry elements will be integrated into the microstructure construction function and parameterized multi-variable lattice structure [143] CAD models will be automatically constructed for further validation and application of my proposed method.

Chapter 5 Feature-based Modeling for Realizing Parametric Design Automation of Material Distribution Structures in CAD Systems

5.1 Chapter overview

The research of this chapter is the subsequent research followed by Chapter 4. This research aims to realize parametric design automation of material distribution structures in CAD systems. To be more specific, the output data of material distribution optimization results are expected to be directly used to realize integration between structural optimization platforms and CAD systems, the CAD model construction processes for material distribution structures are expected to be automatic to drastically improve the modeling efficiency, and the constructed models are expected to be 3D solid models with sufficient editable design parameters for flexible model modifications. In this chapter of research, feature-based modeling method is adopted to realize the expected parametric design automation. A new type of feature, named material distribution feature is proposed and applied to construct the parametric CAD model automatically. In addition, automatic CAD model construction of more complicated material distribution structures, such as multi-scale topology optimized structures is studied in this chapter. The parametric design automation of multi-scale structures has been realized in CAD systems.

The rest of the chapter is organized as follows. Subsection 5.2 gives an introduction to this research, where the background, gaps, and motivations of this research are introduced. The related literature to this research has been reviewed in Chapter 2, including material distribution structures, contemporary CAD systems, feature-based modeling method, and topology optimization in subsections 2.2, 2.3, 2.4, and 2.5. Subsection 5.3 gives a conceptual framework of the proposed method. In this subsection, the proposed material distribution feature is introduced. A case study of automatic model construction of a multiscale topology-optimized Michell structure with lattices

of multi-variables is presented in subsection 5.4. The CAD model is constructed in a widely used CAD system, Siemens NX 12. My method is validated to be of sufficient editability for further modifications. Subsection 5.5 concludes the research of this chapter, where the research is summarized, and future work is pointed out.

5.2 Introduction

The fast development of advanced manufacturing techniques has made it possible to manufacture complex structures to realize lightweight and high-performance designs. As a result, a wide exploration of the design space for functional material/structure has been enabled in research and engineering applications. Structural optimization is not limited to single-scale problems. Concurrent multi-scale optimization methods for optimizing both macrostructures and elemental microstructures have been developed. However, the optimized multi-scale structures have complex geometry characteristics, and there are many challenges for constructing their CAD models at an acceptable efficiency.

The research of this chapter is the subsequent research followed by Chapter 4. In this research, I aim to study parametric design automation for more complicated material distribution structures. In addition, the conceptual framework of CAD model construction for material distribution structures is extended to a more generic way with a new type of feature defined. The proposed method is applied to construct a CAD model of a non-uniform lattice structure automatically in this case study. Sequential modeling mode is used to construct microstructure elements with different values of design parameters. The modeling efficiency has slowed down compared with the batch modeling mode applied in Chapter 4, but more design parameters can be included, and the parametric design automation is validated to be much more flexible for complicated cases in the sequential modeling mode.

5.3 Conceptual framework

5.3.1 Material distribution structures with multi-scale geometric characteristics

With the development of advanced manufacturing technologies, material distribution structures have the potential to have multi-scale geometric characteristics. Take lattice structures, for example, the lattice structures can be designed with hybrid lattice types [58], and each type of lattice microstructure can have non-uniform and non-fixed geometries with multi-variable parameters [59,60].

Figure 5.1 shows an example of a multi-variable lattice structure. In the global macrostructure, some regions have infills allocated with local microstructure lattices, while other regions are left void without any materials arranged. The lattice may have many types and for each type of the lattice, it can be parameterized with unique parameters. For each local microstructure, its geometric feature is specific and unique which is mapped with a set of values of its design parameters. For example, in Figure 5.1, the candidate lattice has 4 design parameters α, β, γ, and δ. Each parameter represents the ratio between the illustrated width of the lattice and the side length of the squared microstructure element. For the lattices located in different regions, it might have different values for the parameters, which will lead to different geometric shapes. In addition, each lattice might have its unique non-geometry features, such as material density and color.

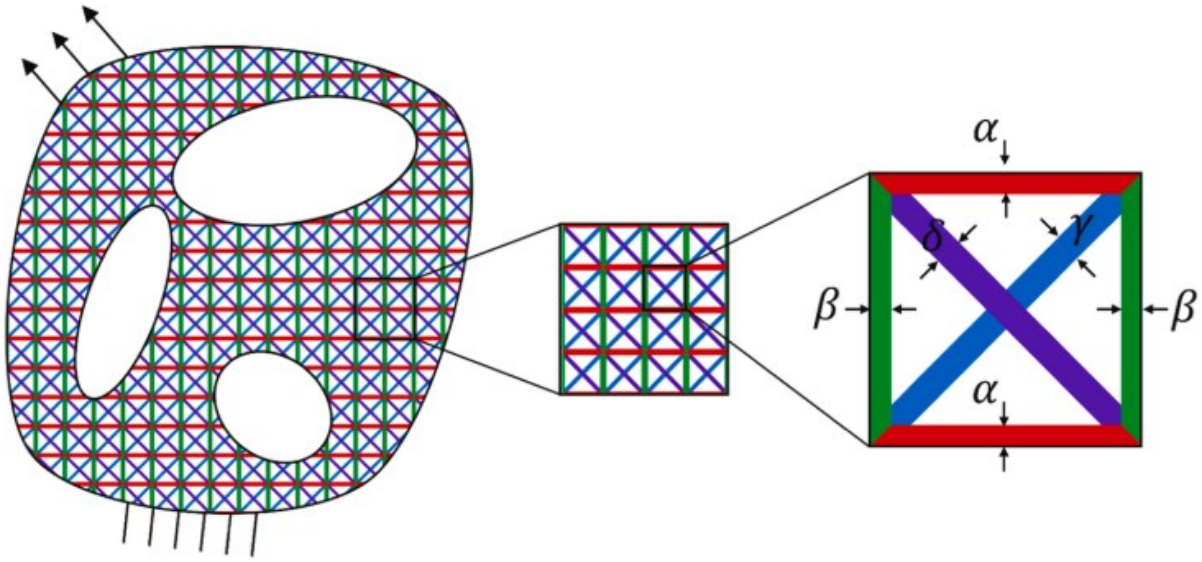


Figure 5.1 Multi-variable lattice structure [138]

5.3.2 Material distribution feature

In order to represent complex multi-scale material distribution structures, the material distribution feature is proposed and applied. The proposed material distribution feature should have the following key characteristics:

- The index or reference number of each spatially distributed microstructure element.
- The belonged subtype of each microstructure element.
- The parameter list for each microstructure element. The parameters can include both geometric parameters and non-geometric parameters.
- The spatial distribution information which represents the relative position between each local microstructure element and the global macrostructure.
- Methods available for communication with end users for adding/deleting/editing design parameters for each microstructure element.

- Methods available for constructing, storing, displaying, editing, and deleting the global material distribution macrostructure.

The proposed material distribution feature is applied in my case study in subsection 5.4 to realize parametric design automation for material distribution structures in CAD systems.

5.4 Case study

In this subsection, the automatic CAD model construction of a topology-optimized Michell structure [144] with lattices of multi-variables is realized to validate the proposed method. The same plug-in named "Create MD Model" developed in Chapter 4 is used to read the data file and activate microstructure model construction function in the model construction loop. The sequential modeling mode is realized in the research of this chapter and applied to automatically construct the CAD model of a multi-scale structure. Feature-based modeling method is adopted. In my case study, lattice structure topology optimization (LSTO) [138] was applied to get the optimized structure, and the optimization output provided the input data of the multi-scale material distribution structure. While in this case, a structure with symmetric shape and boundary conditions is optimized, my method has also been shown to be applicable in asymmetric cases. In this subsection, the optimization problem is stated, the specific case is introduced, the result of the final material distribution matrix is plotted, and the automatically constructed CAD model is shown.

5.4.1 Formulation of the problem: minimum compliance for a lattice structure

Figure 5.1 shows the multi-variable lattice used for LSTO with 4 design parameters α, β, γ and δ. Each parameter represents the ratio between the illustrated width of the lattice and the side

length of the squared microstructure element. The LSTO aims to determine the macro-structure topology and the optimal distribution of the varying size lattices.

In my case, the minimum compliance problem is considered, which can be formulated as below [138] :

$$
\begin{aligned}
&\text{find:} && \boldsymbol{x}^M = \left[\boldsymbol{\rho}_1^{\boldsymbol{M}}, \boldsymbol{\rho}_2^{\boldsymbol{M}}, \ldots, \boldsymbol{\rho}_N^{\boldsymbol{M}}\right]^{\mathrm{T}}, \\
& && \boldsymbol{x}^L = [\alpha_1, \alpha_2, \ldots, \alpha_N; \beta_1, \beta_2, \ldots, \beta_N; \gamma_1, \gamma_2, \ldots, \gamma_N; \delta_1, \delta_2, \ldots, \delta_N]^{\mathrm{T}} \\
&\text{minimize} && C = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{K} \boldsymbol{U} = \sum_{i=1}^{N} \boldsymbol{u}_i^T \, k_i \boldsymbol{u}_i \\
&\text{subject to} && \boldsymbol{F} = \boldsymbol{K}\boldsymbol{U} \\
& && G^M = \sum_{i=1}^{N} \rho_i^M \rho_i^L \leq N f^M \\
& && \mathbf{0} \leq \rho_{\boldsymbol{i}}^{\boldsymbol{M}} \leq \mathbf{1}, \boldsymbol{i} = \mathbf{1}, \mathbf{2}, \ldots, N
\end{aligned}
\tag{5.1}
$$

where vector $\boldsymbol{x}^M$ denotes macro-element densities, vector $\boldsymbol{x}^L$ denotes lattice geometric parameters, C denotes the global compliance of the macro-structure, **U** and **F** are the nodal displacement and force vectors of the macro-structure, $\boldsymbol{u}_i$ is the elemental displacement vector, G^M denotes the volume constraint function, and f^M is the volume fraction upper bound.

5.4.2 Specific case: minimum compliance of a Michell-type structure

Figure 5.2 is the flowchart which shows the processes of LSTO. In my case, an energy-based homogenization method (EBHM) [139,145] is applied to calculate the effective material properties of each cell element, and the adjoint method [18] is used for sensitivity analysis. To be more specific, the sensitivity analysis is implemented as follows:

As a first step, second order polynomials are used to interpolate the relationships between the lattice effective properties and its geometric parameters [138]. Then, the relative density ρ^L or the elastic components D_{ij}^L can be expressed as:

$$\rho^L \text{ or } D_{ij}^L = a_1 + a_2\alpha + a_3\beta + a_4\gamma + a_5\delta + a_6\alpha^2 + a_7\alpha\beta + a_8\alpha\gamma + a_9\alpha\delta + a_{10}\beta^2 + a_{11}\beta\gamma + a_{12}\beta\delta + a_{13}\gamma^2 + a_{14}\gamma\delta + a_{15}\delta^2 \quad (5.2)$$

Then, the PAMP method [146] is adopted to interpolate the elastic matrix D_i^M of an arbitrary macro-element, written as

$$D_i^M = [d + (\rho_i^M)^s(1 - d)]\, D_i^L \quad (5.3)$$

The global stiffness matrix $\boldsymbol{K}$ of the macro-structure can be obtained by assembling the macro-element stiffness matrices k_i, as:

$$\boldsymbol{K} = \sum_i^N \boldsymbol{k}_i = \sum_i^N \int_{\Omega_i} \boldsymbol{B}^T \boldsymbol{D}_i^M \boldsymbol{B}\, d\Omega_i \quad (5.4)$$

where $\boldsymbol{B}$ is the strain-displacement matrix, and Ωi is the domain for macro-element i.

According to the adjoint method [18], the derivatives of the objective function with respect to the macro-element density ρ_i^M is calculated as:

$$\frac{\partial C}{\partial \rho_i^M} = -\boldsymbol{u}_i^T \frac{\partial \boldsymbol{k}_i}{\partial \rho_i^M} \boldsymbol{u}_i \quad (5.5)$$

Combining (5.3), (5.4), and (5.5), the sensitivity of the objective function with respect to ρ_i^M is expressed as:

$$\frac{\partial C}{\partial \rho_i^M} = -\boldsymbol{u}_i^T \frac{\partial}{\partial \rho_i^M}\left(\int_{\Omega_i} \boldsymbol{B}^T \boldsymbol{D}_i^M \boldsymbol{B} d\Omega_i\right)\boldsymbol{u}_i = -\boldsymbol{u}_i^T \left(\int_{\Omega_i} \boldsymbol{B}^T \frac{\partial \boldsymbol{D}_i^M}{\partial \rho_i^M} \boldsymbol{B} d\Omega_i\right)\boldsymbol{u}_i$$
$$= -s(\rho_i^M)^{s-1}(1-d)\boldsymbol{u}_i^T \left(\int_{\Omega_i} \boldsymbol{B}^T \boldsymbol{D}_i^M \boldsymbol{B} d\Omega_i\right)\boldsymbol{u}_i \tag{5.6}$$

Meanwhile, the derivative of the structure volume function G^M is given by:

$$\frac{\partial G^M}{\partial \rho_i^M} = \rho_i^L \tag{5.7}$$

Similarly, the sensitivity of the objective function with respect to the lattice structure design variables, e.g. α_i, can be expressed as follows:

$$\frac{\partial C}{\partial \boldsymbol{\alpha}_i} = -\boldsymbol{u}_i^T \frac{\partial \boldsymbol{k}_i}{\partial \boldsymbol{\alpha}_i} \boldsymbol{u}_i \tag{5.8}$$

Combing (5.3), (5.4), and (5.8), it can be obtained that:

$$\frac{\partial C}{\partial \boldsymbol{\alpha}_i} = -\boldsymbol{u}_i^T \frac{\partial}{\partial \boldsymbol{\alpha}_i}\left(\int_{\Omega_i} \boldsymbol{B}^T \boldsymbol{D}_i^M \boldsymbol{B} d\Omega_i\right)\boldsymbol{u}_i = -\boldsymbol{u}_i^T \left(\int_{\Omega_i} \boldsymbol{B}^T \frac{\partial \boldsymbol{D}_i^M}{\partial \boldsymbol{\alpha}_i} \boldsymbol{B} d\Omega_i\right)\boldsymbol{u}_i$$
$$= [d+(\rho_i^M)^s(1-d)]\boldsymbol{u}_i^T \left(\int_{\Omega_i} \boldsymbol{B}^T \frac{\partial \boldsymbol{D}_i^M}{\partial \boldsymbol{\alpha}_i} \boldsymbol{B} d\Omega_i\right)\boldsymbol{u}_i \tag{5.9}$$

where the derivative of the lattice effective elastic matrix $\frac{\partial \boldsymbol{D}_i^M}{\partial \boldsymbol{\alpha}_i}$ can be calculated from (5.2).

The sensitivities of the structure volume function with respect to the lattice structure design variables are expressed as:

$$\frac{\partial G^M}{\partial \boldsymbol{\alpha}_i} = \rho_i^M \frac{\partial \rho_i^L}{\partial \boldsymbol{\alpha}_i} \tag{5.10}$$

In (5.10), the derivative of the relative density with respect to the lattice design variables $\frac{\partial \rho_i^L}{\partial \boldsymbol{\alpha}_i}$ can be calculated from the interpolation function (5.2)

The method of moving asymptotes (MMA) optimizer [142] is used to solve the optimization problem. Finally, the optimization result is a group of distribution of parameters, $\boldsymbol{x}^M$ and $\boldsymbol{x}^L$, for each cell element.

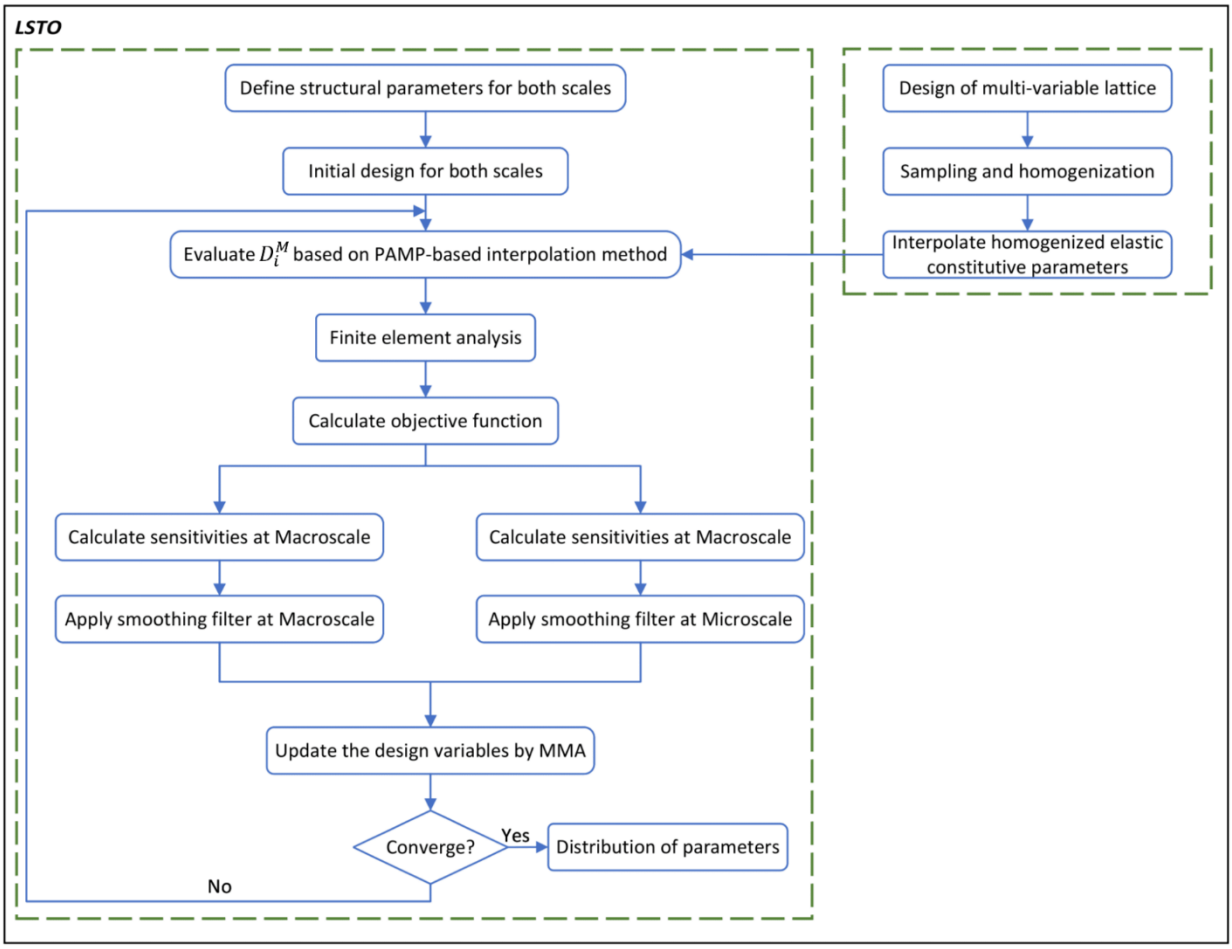


Figure 5.2 Flowchart of LSTO (adapted from [138])

A Michell-type structure is used for LSTO and demonstration of my automatic CAD model construction method. Figure 5.3 shows the original structure and boundary conditions of the simply supported beam. The domain is meshed by 1mm size squares of 100×40. The maximum volume fraction is set to be 0.4 and the filter radius is 1.5 element sizes.

5.4.3 Optimization result: a multi-scale Michell-type structure

Figure 5.4 is the optimization result plotted by Matlab. The optimized structure is verified to be mesh independent. The optimized structure has 2830 elements filled with multi-variable lattices and each lattice cell has its own unique parameters' values, which also means each element has its own specific geometric shape, porosity, density, and mechanical properties.

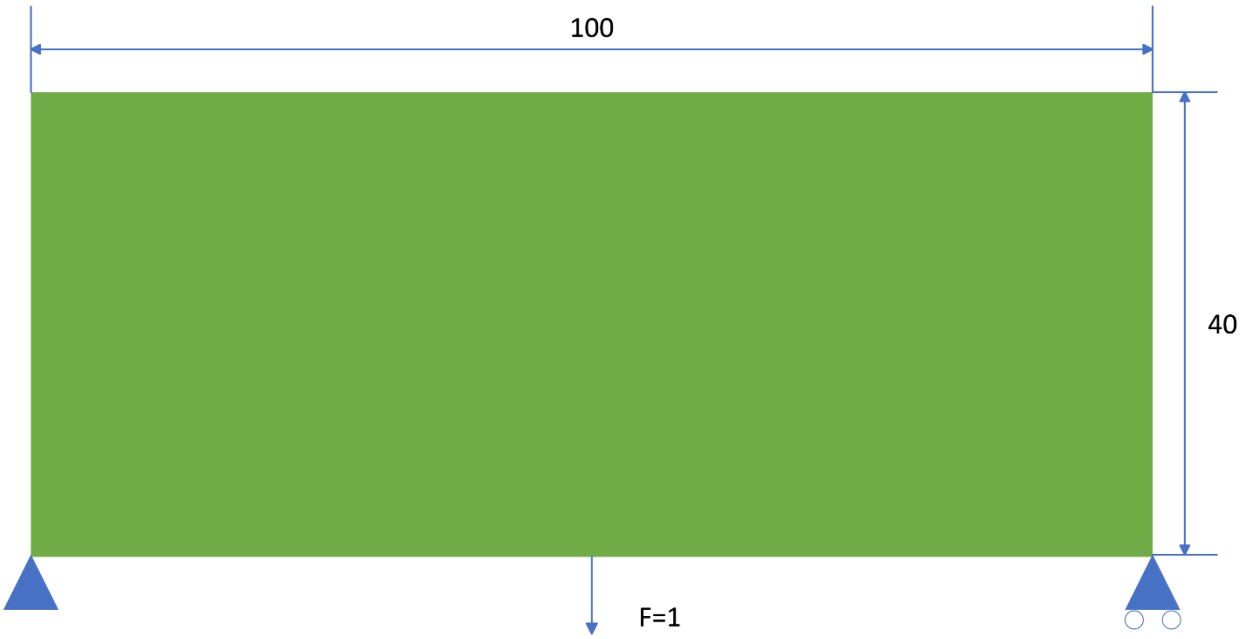


Figure 5.3 LSTO of a Michell-type structure: Initial design domain and boundary conditions

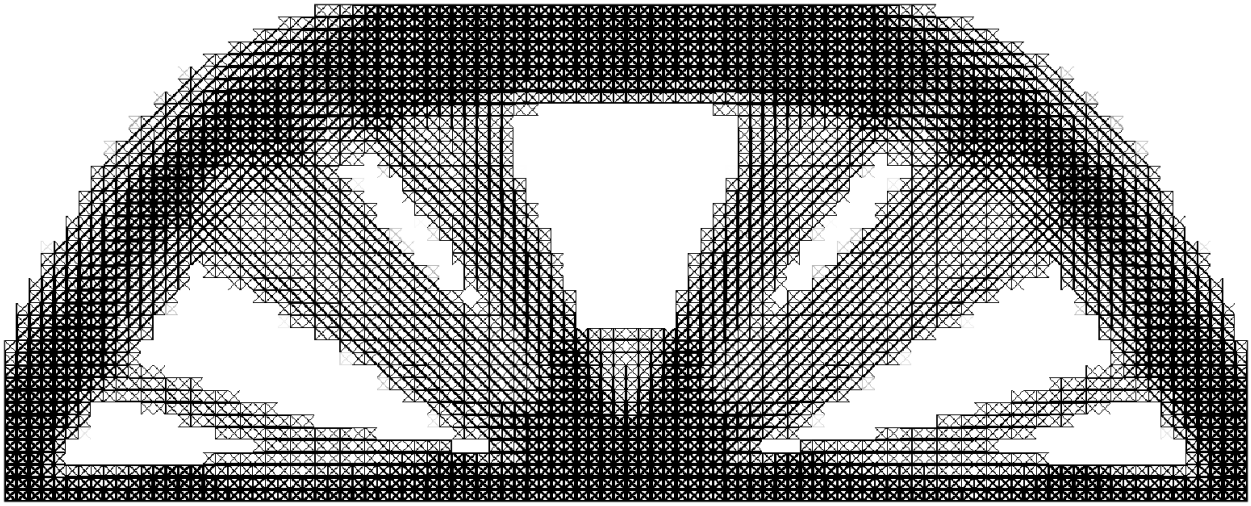


Figure 5.4 The LSTO result plotted by Matlab

Figure 5.5 and Figure 5.6 are the stress plot (von Mises) and elemental density plot of the optimized structure. It can be seen that the regions with high values of von Mises stress are allocated with lattices of high density, which demonstrates the optimization result is a reasonable material distribution in space for the entire structure.

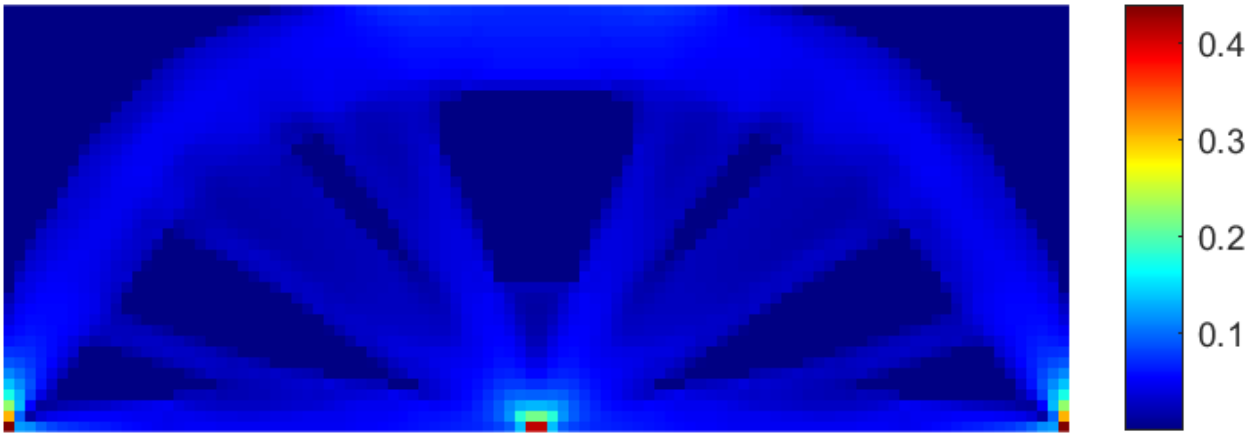


Figure 5.5 Stress distribution plot of the optimized structure (von Mises).

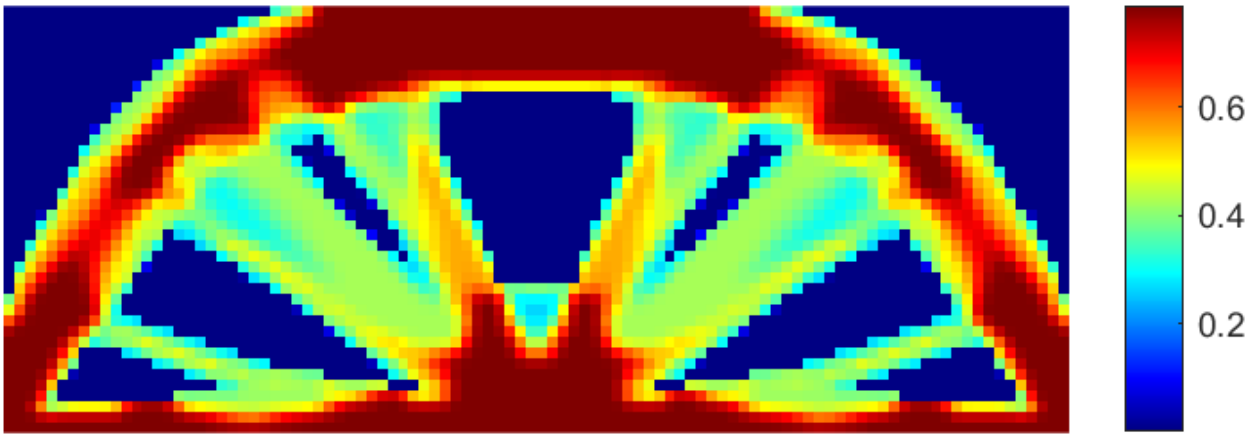


Figure 5.6 Local elemental density plot of the optimized structure.

5.4.4 Data file creation

The original data from the optimization result is processed and a data file in the format of comma-separated values (CSV) is created to store the information of the structure. Based on the proposed material distribution feature concept, the element index, centroid position, element type, and design parameters for each microstructure element are included. Table 5.1 shows the data for a small region in the left part of the structure. The "Index" column is the elemental index for each microstructure feature. (X, Y, Z) is the centroid position for each lattice. The candidate lattice type in Figure 5.1 is regarded as type 3. The values of the parameters extracted from $\boldsymbol{x}^L$ are stored in the columns of alpha, beta, gamma, and delta, which determine the lattices' geometries. The values of the parameters extracted from $\boldsymbol{x}^M$ are stored in the column of "depth", which determine the extrusion length for each local lattice. The data can be directly read by my developed plug-in in Siemens NX 12.

Table 5.1 Data file of the optimized structure

Index	X	Y	Z	Type	alpha	beta	gamma	delta	depth
1	0	0	0	3	0.240	0.240	0.240	0.240	1
2	0	1	0	3	0.240	0.240	0.240	0.240	1
3	0	2	0	3	0.240	0.240	0.240	0.240	1
4	0	3	0	3	0.240	0.240	0.240	0.240	1
5	0	4	0	3	0.240	0.240	0.240	0.240	1
6	0	5	0	3	0.240	0.240	0.240	0.240	1
7	0	6	0	3	0.240	0.240	0.240	0.240	1
8	0	7	0	3	0.240	0.240	0.240	0.240	1
9	0	8	0	3	0.240	0.240	0.240	0.240	1
10	0	9	0	3	0.208	0.240	0.240	0.208	1
11	0	10	0	3	0.110	0.240	0.210	0.108	1
12	0	11	0	3	0.057	0.210	0.125	0.045	1
13	0	12	0	3	0.042	0.119	0.119	0.040	1
14	1	0	0	3	0.240	0.240	0.240	0.240	1
15	1	1	0	3	0.240	0.240	0.240	0.240	1
16	1	2	0	3	0.240	0.240	0.240	0.240	1
17	1	3	0	3	0.240	0.240	0.240	0.240	1
18	1	4	0	3	0.240	0.240	0.240	0.240	1
19	1	5	0	3	0.240	0.240	0.240	0.240	1
20	1	6	0	3	0.240	0.240	0.240	0.240	1
21	1	7	0	3	0.240	0.240	0.240	0.240	1
22	1	8	0	3	0.240	0.240	0.240	0.240	1
23	1	9	0	3	0.236	0.240	0.240	0.236	1
24	1	10	0	3	0.193	0.240	0.236	0.183	1
25	1	11	0	3	0.130	0.236	0.212	0.101	1
26	1	12	0	3	0.082	0.211	0.213	0.070	1
27	1	13	0	3	0.054	0.204	0.212	0.044	1
28	1	14	0	3	0.042	0.176	0.161	0.040	1
29	1	15	0	3	0.040	0.097	0.095	0.040	0.943

5.4.5 Automatic CAD model construction: sequential modeling mode

Based on the created data file, a 3D CAD model is automatically constructed in NX 12 by the proposed sequential modeling mode (see Figure 5.7 and Figure 5.8). Figure 5.9 shows the local details of the structure cell, where every cell element has its unique parameter values and lattice geometry. In addition, every cell element can be selected and edited by users for further modifications of the model. The original structure has 4000 elements, and the optimized structure has 2830 elements filled with multi-variable lattices. It took about 1 second for model construction

of each cell element and the entire CAD model was constructed in 45 minutes. Compared with the batch modeling mode applied in Chapter 4, the sequential modeling mode needs to read and process more parameters, the model construction process is slower, but it is more flexible. Although the model construction process may take a longer time, the complexity of the model is significantly increased, and the modeling efficiency is still acceptable for most users.

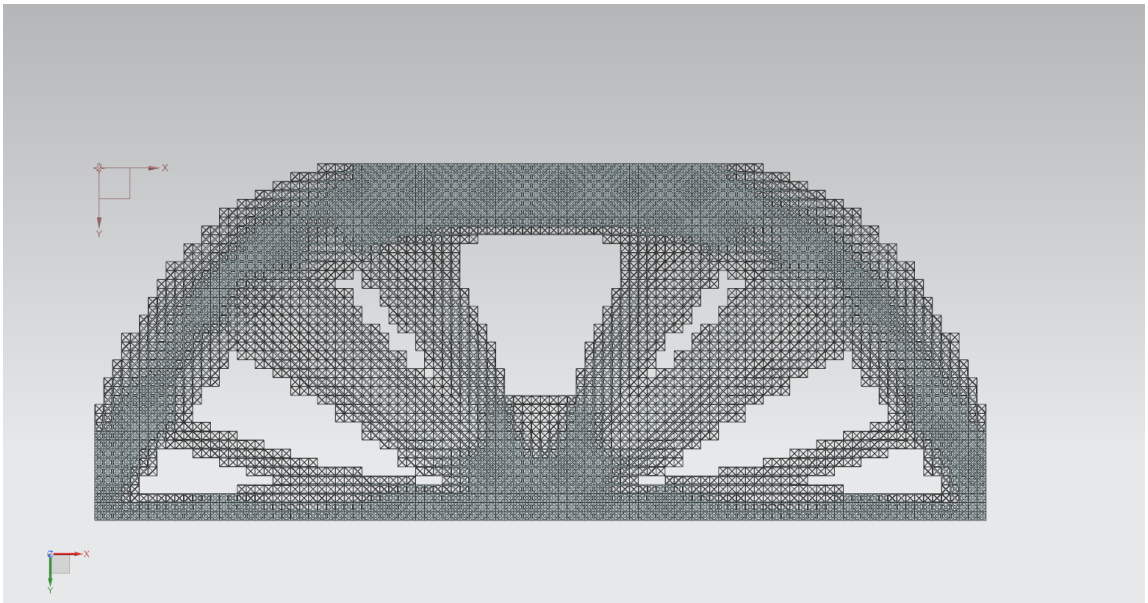


Figure 5.7 The automatically constructed CAD model in NX 12 (front view).

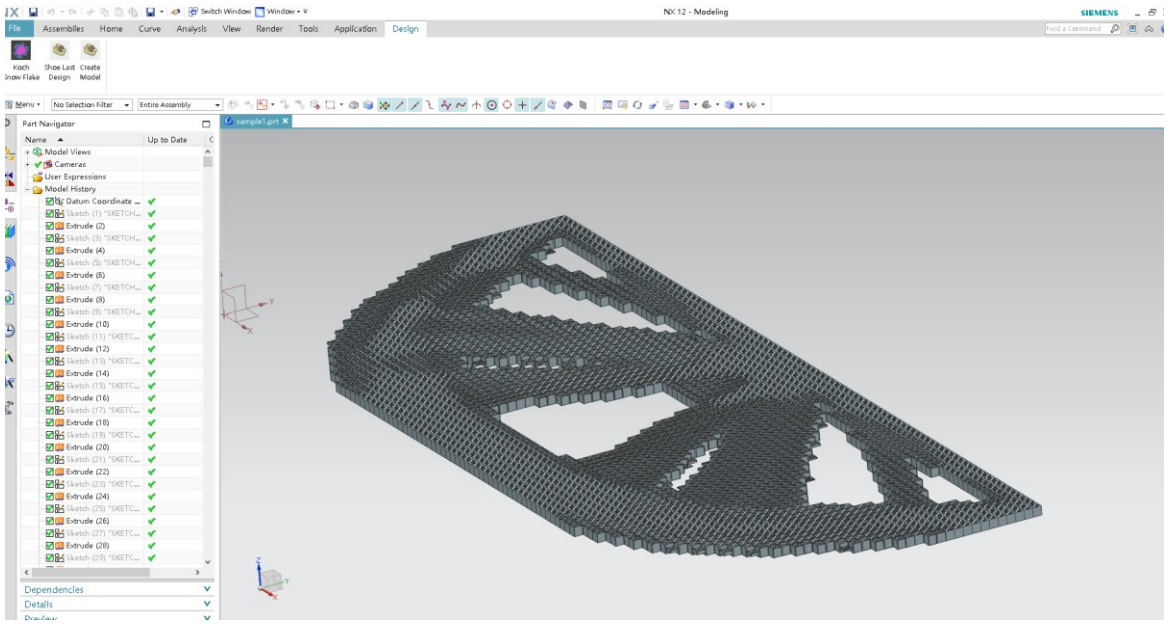


Figure 5.8 The automatically constructed CAD model in NX 12 (isometric view).

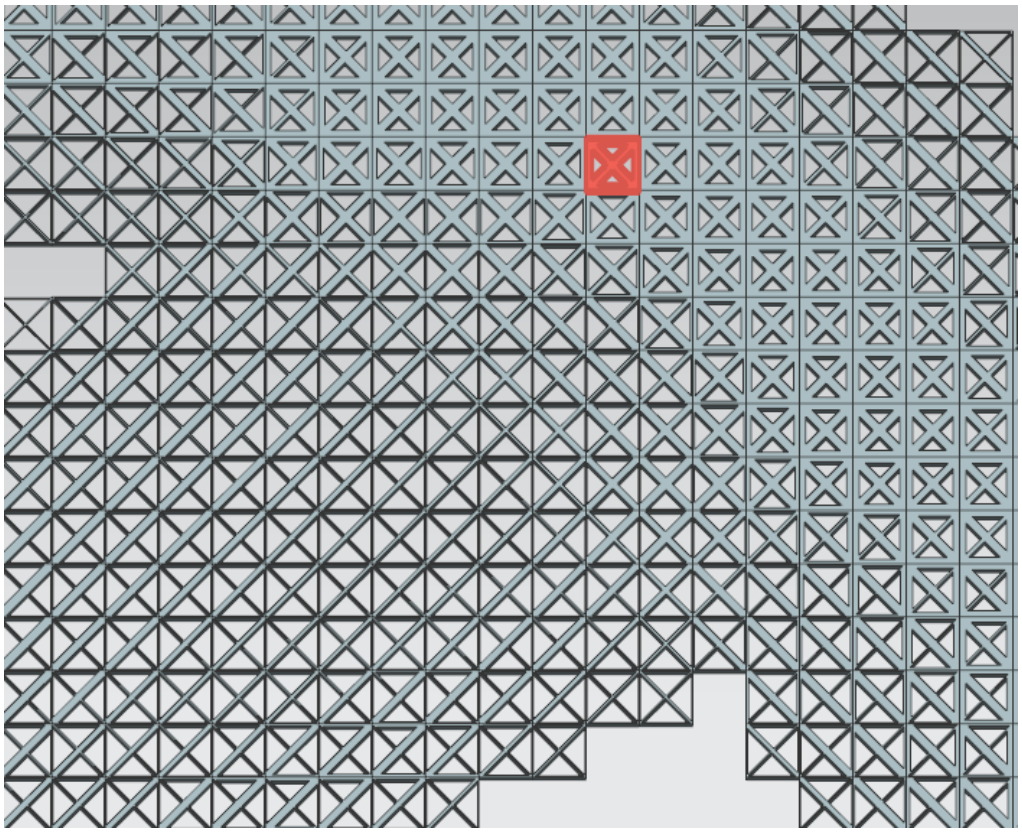


Figure 5.9 Local details of the constructed model

5.5 Conclusion

5.5.1 Summary and contribution of this research

In this research, parametric design automation for material distribution structures was realized in CAD systems. Feature-based modeling method was adopted and a new type of feature, named material distribution feature was proposed and applied. The data from the structural optimization result was processed and stored in a new format, which can be directly read by CAD systems. The sequential modeling mode was applied in a developed CAD plug-in to automatically construct a CAD model for a multi-scale structure. Compared with the batch modeling mode applied in Chapter 4, the sequential modeling mode was slower but had more flexibility. The CAD model constructed is a feature-based parametric model. Each element was parametrically defined and constructed, which allowed the user to select and edit for further model modifications.

Previously, the tessellation models of the material distribution structures were created statically with fixed parameters. Such models have a lower level of information and bad compatibility. The parametric CAD models of such structures were constructed manually. Now, with my proposed method, parametric CAD models of material distribution structures can be constructed automatically. The parametric design automation of complex material distribution structures can be realized.

5.5.2 Limitations and future work

The research of this chapter focuses on realizing parametric design automation for material distribution structures composed of complex geometric microstructure elements in CAD systems. In this case study, the sequential modeling mode was applied and the overall result is acceptable.

Based on the proposed material distribution feature concept and modeling modes, there are more research work to be explored in the future.

For the modeling mode, it can be optimized to further improve the modeling efficiency. A mixed modeling mode might be developed in the future which is a combination of batch modeling mode and sequential modeling mode. With the mixed modeling mode, the data needs to be pre-processed, and uniform and non-uniform regions are expected to be recognized before starting the model construction process. Then, the batch modeling mode will be applied in the uniform regions and the sequential modeling model will be applied in the non-uniform regions to further improve the modeling efficiency.

For the storage consumption, the models constructed with my proposed method currently take up a relatively large storage space. For example, the CAD model of Figure 5.7 has 2830 microstructure elements, which takes up about 200 MB storage space. Although it is acceptable for most users with modern PC configurations, the size of the model is expected to be further reduced. A smart union algorithm for merging neighboring microstructure elements during the loop construction process of a material distribution structure is expected to be developed in the future as an option for the users to reduce the storage consumption of the constructed model.

To further improve the design flexibility, a library of commonly used parametric geometry elements will be integrated and multi-scale structures with hybrid types of microstructure cells will be explored for their parametric design automation. In addition, non-geometric characteristics have the potential to be included in the data file to further improve the flexibility of the proposed method. For example, different microstructure elements might have different colors, material densities, and other non-geometric properties. Such information will be included in the data file as future work

to guide the automatic CAD model construction of material distribution structures with multi-materials.

Chapter 6 Conclusion and future work

6.1 Conclusion

In this research, parametric design automation methods for complex geometry models in CAD systems have been proposed as the conceptual framework, developed in CAD systems, and validated by case studies. The automatic CAD model construction for fractal objects and material distribution structures has been realized. Associative relationships between geometric elements have been built to capture design intent and improve modeling efficiency. The constructed models are feature-based CAD models with editable parameters, which have good compatibility in numerical simulation environments and are easy to be further modified for post-processing.

The novelty of this work is to integrate algorithmic approaches to construct geometrical entities of fractal objects and material distribution structures into a 3D modeling software environment. The models can either be constructed iteratively with dynamic fractal generators or be built in a loop construction approach that realizes automatic local microstructure construction for the global macrostructure. A balance between model construction efficiency and design flexibility has been made for the parametric design automation processes. The models constructed with my proposed methods are parametric solid CAD models, which have good compatibility and reusability for product life cycle management. With my new proposed methods, the product development processes for complex geometric objects can be modified to perform more efficiently, and the product development time can be significantly reduced.

6.2 Research contributions

The main contributions of this research are summarized as follows:

- New methods of automatically constructing parametric CAD models for complex geometric objects in CAD systems were proposed and applied. The modeling efficiency for complex geometry objects in CAD systems was improved.
- Advanced feature concepts were extended to realize parametric design automation in CAD systems. CAD fractal feature was proposed to bridge the gaps between CAD and fractal geometry. Material distribution feature was proposed to realize automatic CAD model construction for material distribution structures.
- The concept of iteratively constructing a complex CAD model was proposed and realized in CAD systems. The proposed iterative model construction method may also have the potential to be applied in generative design.
- The design intent is captured and knowledge for building associative relationships among geometry entities is encapsulated in built-in methods of proposed new feature types.
- Algorithmic approaches to construct geometrical entities within a 3D modeling software environment were proposed and realized.
- A mapping between product physics optimization and digital CAD modeling was realized.

6.3 Limitations and future work

For fractal modeling in CAD systems, the proposed method currently requires the fractal-like geometry objects to be self-connected and self-intersection/self-overlapping is not allowed to appear for any iteration. As a result, my method may not be suitable for constructing some types of fractal objects that may be disconnected or have self-intersection/self-overlapping geometric features when the iteration level increases. To automatically construct CAD fractal models in a broader scope with less geometric requirements, intelligent algorithms to split disconnected fractal

objects and properly merge/unite self-intersect/self-overlapped fractal models in CAD systems are expected to be developed and integrated with my IGM method in future research.

For parametric design automation of material distribution structures in CAD systems, the proposed method currently has not realized automatic model construction for material distribution structures with non-geometric characteristics. Non-geometric characteristics have the potential to be included in the data file to further improve the flexibility of the proposed method. For example, different microstructure elements might have different colors, material densities, and other non-geometric properties. Such information will be included in the data file as future work to guide the automatic CAD model construction of material distribution structures with multi-materials.

In addition, the manufacturing processes of models constructed with the proposed method are expected to be systematically studied in the future. Intelligent decision-making algorithms are expected to be integrated with the proposed parametric design automation to automatically choose the most suitable manufacturing process type, such as additive manufacturing (AM) and subtractive manufacturing (SM). Simulations of related manufacturing processes are expected to be implemented automatically, such as tool-path planning for SM and simulations of 3D printing process for AM.

Bibliography

[1] Y.Y.C. Choong, Chapter 4 - Additive manufacturing for digital transformation, in: C.D. Patel, C.-H. Chen (Eds.), Digital Manufacturing, Elsevier, 2022: pp. 145–182. https://doi.org/https://doi.org/10.1016/B978-0-323-95062-6.00002-4.

[2] I. Gibson, D. Rosen, B. Stucker, M. Khorasani, Additive Manufacturing Technologies, Third Edition, Springer, Switzerland, 2021. https://doi.org/https://doi.org/10.1007/978-3-030-56127-7.

[3] W. Grzesik, Hybrid additive and subtractive manufacturing processes and systems: A review, Journal of Machine Engineering. 18 (2018) 5–24. https://doi.org/10.5604/01.3001.0012.7629.

[4] E. Weflen, M.C. Frank, Hybrid additive and subtractive manufacturing of multi-material objects, Rapid Prototyp J. 27 (2021) 1860–1871. https://doi.org/10.1108/RPJ-06-2020-0142.

[5] J. Liu, A.T. Gaynor, S. Chen, Z. Kang, K. Suresh, A. Takezawa, L. Li, J. Kato, J. Tang, C.C.L. Wang, L. Cheng, X. Liang, A.C. To, Current and future trends in topology optimization for additive manufacturing, Structural and Multidisciplinary Optimization. 57 (2018) 2457–2483. https://doi.org/10.1007/s00158-018-1994-3.

[6] J. Lim, C. You, I. Dayyani, Multi-objective topology optimization and structural analysis of periodic spaceframe structures, Mater Des. 190 (2020). https://doi.org/10.1016/j.matdes.2020.108552.

[7] Bi Zhuming, Wang Xiaoqin, Computer Aided Design and Manufacturing, John Wiley & Sons Ltd, 2020. https://doi.org/10.1002/9781119667889.

[8] C. Elanchezhian, G.S. Sundar, Computer aided manufacturing, Firewall Media, 2007.

[9] K.-H. Chang, e-Design: computer-aided engineering design, Academic Press, 2016.

[10] B. Louhichi, G.N. Abenhaim, A.S. Tahan, CAD/CAE integration: updating the CAD model after a FEM analysis, International Journal of Advanced Manufacturing Technology. 76 (2015) 391–400. https://doi.org/10.1007/s00170-014-6248-y.

[11] J. Frazer, Parametric Computation: History and Future, Architectural Design. 86 (2019) 18–23. https://doi.org/https://doi.org/10.1002/ad.2019.

[12] R. Rempling, A. Mathern, D. Tarazona Ramos, S. Luis Fernández, Automatic structural design by a set-based parametric design method, Autom Constr. 108 (2019). https://doi.org/10.1016/j.autcon.2019.102936.

[13] K. Amadori, M. Tarkian, J. Ölvander, P. Krus, Flexible and robust CAD models for design automation, Advanced Engineering Informatics. 26 (2012) 180–195. https://doi.org/10.1016/j.aei.2012.01.004.

[14] F. Cucinotta, M. Raffaele, F. Salmeri, A stress-based topology optimization method by a Voronoi tessellation Additive Manufacturing oriented, International Journal of Advanced Manufacturing Technology. 103 (2019) 1965–1975. https://doi.org/10.1007/s00170-019-03676-4.

[15] H. Altendorf, F. Latourte, D. Jeulin, M. Faessel, L. Saintoyant, 3D reconstruction of a multiscale microstructure by anisotropic tessellation models, Image Analysis and Stereology. 33 (2014) 121–130. https://doi.org/10.5566/ias.v33.p121-130.

[16] J. Ren, T. Zhou, Y. Rong, Y. Ma, R. Ahmad, Feature-based modeling for industrial processes in the context of digital twins: A case study of HVOF process, Advanced Engineering Informatics. 51 (2022). https://doi.org/10.1016/j.aei.2021.101486.

[17] John E. Hutchinson, Fractals and Self Similarity, Indiana University Mathematics Journal. 30 (1981) 713–747. https://www.jstor.org/stable/24893080.

[18] M.P. Bendsøe, O. Sigmund, Topology Optimization: Theory, Methods, and Applications, Springer, Berlin, Heidelberg, 2004. https://doi.org/https://doi.org/10.1007/9 78-3-662-05086-6.

[19] B.B. Mandelbrot, J.A. Wheeler, The Fractal Geometry of Nature, Am J Phys. 51 (1983) 286–287. https://doi.org/10.1119/1.13295.

[20] Jacques Belair, Serge Dubuc, Fractal Geometry and Analysis, 1991. https://doi.org/10.1007/978-94-015-7931-5.

[21] Michael Barnsley, Fractals Everywhere, Academic Press, Inc., 1989. https://doi.org/10.1080/09332480.1989.11882331.

[22] Kenneth Falconer, Introduction, in: Techniques in Fractal Geometry, John Willey &Sons Ltd, 1997: pp. 7–15.

[23] K.J. Falconer, Introduction, in: Fractal Geometry: Mathematical Foundations and Applications (Third Edition), Wiley, 2014: pp. 16–27.

[24] E. Gerald, Measure, Topology, and Fractal Geometry Second Edition, 2008.

[25] C. Bovill, Fractal Geometry in Architecture and Design, 1996. https://doi.org/10.1007/978-1-4612-0843-3.

[26] H.K. Zhang, W.J. Wu, Z. Kang, X.Q. Feng, Topology optimization method for the design of bioinspired self-similar hierarchical microstructures, Comput Methods Appl Mech Eng. 372 (2020). https://doi.org/10.1016/j.cma.2020.113399.

[27] B.B. Mandelbrott, T. Vicsekto, Directed recursive models for fractal growth, J. Phys. A: Math. Gen. 22 (1989) 377–383.

[28] J. Mishra, S.N. Mishra, Fractals and L-System, in: L-System Fractals, First edition, Elsevier B.V., 2007: pp. 23–48.

[29] D. Fang, X. Li-Fneg, An Application of L-system and IFS in 3D Fractal Simulation, WSEAS TRANSACTIONS on SYSYEMS. 7 (2008).

[30] M. Alfonseca, A. Ortega, A study of the representation of fractal curves by L systems and their equivalences, IBM J Res Dev. 41 (1997) 727–736. https://doi.org/10.1147/rd.416.0727.

[31] M.F. Barnsley, S. Demko, Iterated Function Systems and the Global Construction of Fractals, Source: Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences. 399 (1985) 243–275. https://about.jstor.org/terms.

[32] S. Demko, L. Hodges, B. Naylor, Construction of Fractal Objects with Iterated Function Systems, SIGGRAPH. 19 (1985) 22–26.

[33] S.C. Soo, K.M. Yu, Tool-Path Generation for Fractal Curve Making, Int J Adv Manuf Technol. 19 (2002) 32–48.

[34] Y. Joye, A review of the presence and use of fractal geometry in architectural design, Environ Plann B Plann Des. 38 (2011) 814–828. https://doi.org/10.1068/b36032.

[35] W.E. Lorenz, Fractal Geometry of Architecture Implementation of the Box-Counting Method in a CAD-Software, 27th ECAADe Conference Proceedings. (2009) 697–704.

[36] V. Hotař, Application of Fractal Dimension in Industry Practice, in: Fractal Analysis - Applications in Physics, Engineering and Technology, InTech, 2017: pp. 137–172. https://doi.org/10.5772/intechopen.68276.

[37] V. Hotar, P. Salac, Surface Evaluation by Estimation of Fractal Dimension and Statistical Tools, Scientific World Journal. 2014 (2014). https://doi.org/10.1155/2014/435935.

[38] D.H. Werner, S. Ganguly, An overview of fractal antenna engineering research, IEEE Antennas Propag Mag. 45 (2003) 38–57. https://doi.org/10.1109/MAP.2003.1189650.

[39] J.P. Gianvittorio, Y. Rahmat-Samii, Fractal antennas: A novel antenna miniaturization technique, and applications, IEEE Antennas Propag Mag. 44 (2002) 20–36. https://doi.org/10.1109/74.997888.

[40] D.H. Werner, R.L. Haup, P.L. Wernej, Fractal Antenna Engineering: The Theory and Design of Fractal Antenna Arrays, IEEE Antennas Propag Mag. 41 (1999) 37–59. https://doi.org/10.1109/74.801513.

[41] J. Anguera, A. Andújar, J. Jayasinghe, V.V.S.S. Sameer Chakravarthy, P.S.R. Chowdary, J.L. Pijoan, T. Ali, C. Cattani, Fractal antennas: An historical perspective, Fractal and Fractional. 4 (2020) 1–26. https://doi.org/10.3390/fractalfract4010003.

[42] C. Zhang, Y. Chen, Z. Deng, M. Shi, Role of rough surface topography on gas slip flow in microchannels, Phys Rev E Stat Nonlin Soft Matter Phys. 86 (2012). https://doi.org/10.1103/PhysRevE.86.016319.

[43] C. Zhang, Y. Chen, M. Shi, Effects of roughness elements on laminar flow and heat transfer in microchannels, Chemical Engineering and Processing: Process Intensification. 49 (2010) 1188–1192. https://doi.org/10.1016/j.cep.2010.08.022.

[44] Z. Huang, Y. Hwang, V. Aute, R. Radermacher, Review of Fractal Heat Exchangers, in: International Refrigeration and Air Conditioning Conference, 2016. http://docs.lib.purdue.edu/iracc/1725.

[45] Z. Huang, Y. Hwang, R. Radermacher, Review of nature-inspired heat exchanger technology, International Journal of Refrigeration. 78 (2017) 1–17. https://doi.org/10.1016/j.ijrefrig.2017.03.006.

[46] X. Liu, F. Liu, Q. Dai, F. Yao, T. Zeng, Y. Zhang, C. Yu, NUMERICAL STUDY on the THERMAL PERFORMANCE of A PHASE CHANGE HEAT EXCHANGER (PCHE) with INNOVATIVE FRACTAL TREE-SHAPED FINS, Fractals. 28 (2020). https://doi.org/10.1142/S0218348X20500838.

[47] J. Zheng, C. Yu, T. Chen, Y. Yu, F. Wang, Optimization of the melting performance of a thermal energy storage unit with fractal net fins, Processes. 7 (2019). https://doi.org/10.3390/pr7010042.

[48] S. Xiong, X. Chen, Simulation and experimental research of an effective SAR multilayer interlaced micromixer based on Koch fractal geometry, Microfluid Nanofluidics. 25 (2021). https://doi.org/10.1007/s10404-021-02495-y.

[49] X. Chen, S. Zhang, Z. Wu, Y. Zheng, A novel Koch fractal micromixer with rounding corners structure, Microsystem Technologies. 25 (2019) 2751–2758. https://doi.org/10.1007/s00542-019-04296-4.

[50] F. Tian, A. Jiang, T. Yang, J. Qian, R. Liu, M. Jiang, Application of Fractal Geometry in Gas Sensor: A Review, IEEE Sens J. 21 (2021) 14587–14600. https://doi.org/10.1109/JSEN.2021.3072621.

[51] K. Hassan, T.T. Tung, P.L. Yap, H. Rastin, N. Stanley, M.J. Nine, D. Losic, Fractal Design for Advancing the Performance of Chemoresistive Sensors, ACS Sens. 6 (2021) 3685–3695. https://doi.org/10.1021/acssensors.1c01449.

[52] V. Linares, Á. Aguilar-De-Leyva, M. Casas, I. Caraballo, 3D Printed Fractal-like Structures with High Percentage of Drug for Zero-Order Colonic Release, Pharmaceutics. 2022 (2022) 2298. https://doi.org/10.3390/pharmaceutics.

[53] J.L. Véhel, J. Lévy, V. Fractal, S. Processing, H. Peitgen, Fractal Approaches in Signal Processing To cite this version : HAL Id : inria-00593322, (2011).

[54] V. Hotař, Fractal geometry for industrial data evaluation, Computers and Mathematics with Applications. 66 (2013) 113–121. https://doi.org/10.1016/j.camwa.2013.01.015.

[55] G.A. Losa, D. Ristanović, D. Ristanović, I. Zaletel, S. Beltraminelli, From Fractal Geometry to Fractal Analysis, Appl Math (Irvine). 07 (2016) 346–354. https://doi.org/10.4236/am.2016.74032.

[56] P. Paramanathan, R. Uthayakumar, Fractal interpolation on the Koch Curve, Computers and Mathematics with Applications. 59 (2010) 3229–3233. https://doi.org/10.1016/j.camwa.2010.03.008.

[57] D. Bhate, Four questions in cellular material design, Materials. 12 (2019). https://doi.org/10.3390/ma12071060.

[58] N. Wei, H. Ye, X. Zhang, J. Li, Y. Sui, Topology Optimization for Design of Hybrid Lattice Structures with Multiple Microstructure Configurations, Acta Mechanica Solida Sinica. 35 (2022) 367–383. https://doi.org/10.1007/s10338-021-00302-3.

[59] C. Wang, X. Gu, J. Zhu, H. Zhou, S. Li, W. Zhang, Concurrent design of hierarchical structures with three-dimensional parameterized lattice microstructures for additive manufacturing, Structural and Multidisciplinary Optimization. 61 (2020) 869–894. https://doi.org/10.1007/s00158-019-02408-2.

[60] C. Wang, J.H. Zhu, W.H. Zhang, S.Y. Li, J. Kong, Concurrent topology optimization design of structures and non-uniform parameterized lattice microstructures, Structural and Multidisciplinary Optimization. 58 (2018) 35–50. https://doi.org/10.1007/s00158-018-2009-0.

[61] M. Viccica, M. Galati, F. Calignano, L. Iuliano, Design, additive manufacturing, and characterisation of a three-dimensional cross-based fractal structure for shock absorption, Thin-Walled Structures. 181 (2022). https://doi.org/10.1016/j.tws.2022.110106.

[62] A. Kaveh, S. Talatahari, Size optimization of space trusses using Big Bang-Big Crunch algorithm, Comput Struct. 87 (2009) 1129–1140. https://doi.org/10.1016/j.compstruc.2009.04.011.

[63] M. Khatibinia, H. Yazdani, Accelerated multi-gravitational search algorithm for size optimization of truss structures, Swarm Evol Comput. 38 (2018) 109–119. https://doi.org/10.1016/j.swevo.2017.07.001.

[64] I. Marinić-Kragić, D. Vučina, Z. Milas, Concept of flexible vertical-axis wind turbine with numerical simulation and shape optimization, Energy. 167 (2019) 841–852. https://doi.org/10.1016/j.energy.2018.11.026.

[65] L.F.F. Miguel, L.F. Fadel Miguel, Shape and size optimization of truss structures considering dynamic constraints through modern metaheuristic algorithms, Expert Syst Appl. 39 (2012) 9458–9467. https://doi.org/10.1016/j.eswa.2012.02.113.

[66] J. Wu, O. Sigmund, J.P. Groen, Topology optimization of multi-scale structures: a review, Structural and Multidisciplinary Optimization. 63 (2021) 1455–1480. https://doi.org/10.1007/s00158-021-02881-8.

[67] J. ZHU, H. ZHOU, C. WANG, L. ZHOU, S. YUAN, W. ZHANG, A review of topology optimization for additive manufacturing: Status and challenges, Chinese Journal of Aeronautics. 34 (2021) 91–110. https://doi.org/10.1016/j.cja.2020.09.020.

[68] V.E. Veen, S. Yuan, M.I. Katsnelson, M. Polini, A. Tomadin, Quantum transport in Sierpinski carpets, Phys Rev B. 93 (2016) 115428. https://doi.org/10.1103/PhysRevB.93.115428.

[69] K. Liu, A. Tovar, An efficient 3D topology optimization code written in Matlab, Structural and Multidisciplinary Optimization. 50 (2014) 1175–1196. https://doi.org/10.1007/s00158-014-1107-x.

[70] W. Zuo, K. Saitou, Multi-material topology optimization using ordered SIMP interpolation, Structural and Multidisciplinary Optimization. 55 (2017) 477–491. https://doi.org/10.1007/s00158-016-1513-3.

[71] L. Yin, W. Yang, Optimality criteria method for topology optimization under multiple constraints, Comput Struct. 79 (2001) 1839–1850. https://doi.org/https://doi.org/10.1016/S0045-7949(01)00126-2.

[72] K. Svanberg, The method of moving asymptotes—a new method for structural optimization, Int J Numer Methods Eng. 24 (1987) 359–373. https://doi.org/10.1002/nme.1620240207.

[73] D.A. Rajon, W.E. Bolch, Marching cube algorithm: Review and trilinear interpolation adaptation for image-based dosimetric models, Computerized Medical Imaging and Graphics. 27 (2003) 411–435. https://doi.org/10.1016/S0895-6111(03)00032-6.

[74] S. Liu, Q. Li, J. Liu, W. Chen, Y. Zhang, A Realization Method for Transforming a Topology Optimization Design into Additive Manufacturing Structures, Engineering. 4 (2018) 277–285. https://doi.org/10.1016/j.eng.2017.09.002.

[75] I. Stroud, P.C. Xirouchakis, STL and extensions, Advances in Engineering Software. 31 (2000) 83–95. https://doi.org/https://doi.org/10.1016/S0965-9978(99)00046-0.

[76] M. Szilvśi-Nagy, Gy. Mátyási, Analysis of STL files, Math Comput Model. 38 (2003) 945–960. https://doi.org/https://doi.org/10.1016/S0895-7177(03)90079-3.

[77] Ibrahim Zeid, R Sivasubramanian, CAD/CAM Theory and Practice, Second Edition, Tata McGraw Hill Education Private Limited, 2009.

[78] Christoph M. Hoffmann, Geometric and Solid Modeling: An Introduction, Morgan Kaufmann Pub, 1989.

[79] L. Piegl, W. Tiller, The NURBS Book, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. https://doi.org/10.1007/978-3-642-59223-2.

[80] Sabine Coquillart, Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling, SIGGRAPH. 24 (1990) 187–196.

[81] Thomas W. Sederberg, Scott R. Parry, Free-Form Deformation of Solid Geometric Models, SIGGRAPH. 20 (1986) 151–160.

[82] Y.C. Yeung, K.M. Yu, Manufacturability of fractal geometry, Materials Science Forum. 471–472 (2004) 722–726. https://doi.org/10.4028/www.scientific.net/msf.471-472.722.

[83] W.K. Chiu, Y.C. Yeung, K.M. Yu, Toolpath generation for layer manufacturing of fractal objects, Rapid Prototyp J. 12 (2006) 214–221. https://doi.org/10.1108/13552540610682723.

[84] A.M.M.S. Ullah, D.M. D'addona, Y. Seto, S. Yonehara, A. Kubo, Utilizing fractals for modeling and 3d printing of porous structures, Fractal and Fractional. 5 (2021). https://doi.org/10.3390/fractalfract5020040.

[85] M. Viccica, M. Galati, F. Calignano, L. Iuliano, Design, additive manufacturing, and characterisation of a three-dimensional cross-based fractal structure for shock absorption, Thin-Walled Structures. 181 (2022) 110106. https://doi.org/https://doi.org/10.1016/j.tws.2022.110106.

[86] S.Y. Jun, B. Sanz-Izquierdo, E.A. Parker, D. Bird, A. McClelland, Manufacturing Considerations in the 3-D Printing of Fractal Antennas, IEEE Trans Compon Packaging Manuf Technol. 7 (2017) 1891–1898. https://doi.org/10.1109/TCPMT.2017.2730366.

[87] C.-C.A. Chen, Y.-S. Juang, W.-Z. Lin, Generation of fractal toolpaths for irregular shapes of surface finishing areas, J Mater Process Technol. 127 (2002) 146–150. https://doi.org/https://doi.org/10.1016/S0924-0136(02)00115-2.

[88] S.C. Soo, K.M. Yu, Rapid prototyping using fractal geometry, Solid Freeform Fabrication Symposium. (2001) 424–431.

[89] S.C. Soo, K.M. Yu, Rapid prototyping for self-similarity design, J Mater Process Technol. 139 (2003) 219–225. https://doi.org/10.1016/S0924-0136(03)00223-1.

[90] S.C. Soo, K.M. Yu, W.K. Chiu, Modeling and fabrication of artistic products based on IFS fractal representation, CAD Computer Aided Design. 38 (2006) 755–769. https://doi.org/10.1016/j.cad.2006.04.003.

[91] K. Guan, T. Mei, F. Kong, X. Li, Fractal Object Generation Based on CAD and CAM, International Conference on Mechatronics and Automation. (2007) 2210–2215. https://doi.org/10.1109/ICMA.2007.4303895.

[92] N.D. Cornea, D. Silver, P. Min, Curve-skeleton properties, applications, and algorithms, IEEE Trans Vis Comput Graph. 13 (2007) 530.

[93] J.C. Cuillière, V. François, A. Nana, Automatic construction of structural CAD models from 3D topology optimization, Comput Aided Des Appl. 15 (2018) 107–121. https://doi.org/10.1080/16864360.2017.1353726.

[94] A. Amroune, J.C. Cuillière, V. François, Automated Lofting-Based Reconstruction of CAD Models from 3D Topology Optimization Results, CAD Computer Aided Design. 145 (2022). https://doi.org/10.1016/j.cad.2021.103183.

[95] T. Stangl, S. Wartzack, Feature based interpretation and reconstruction of structural topology optimization results, in: DS 80-6 Proceedings of the 20th International Conference

on Engineering Design (ICED 15) Vol 6: Design Methods and Tools-Part 2 Milan, Italy, 27-30.07. 15, 2015: pp. 235–244.

[96] E.C. Sherbrooke, N.M. Patrikalakis, F.-E. Wolter, Differential and Topological Properties of Medial Axis Transforms, Graphical Models and Image Processing. 58 (1996) 574–592. https://doi.org/https://doi.org/10.1006/gmip.1996.0047.

[97] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, A. Telea, 3D skeletons: A state-of-the-art report, in: Computer Graphics Forum, Blackwell Publishing Ltd, 2016: pp. 573–597. https://doi.org/10.1111/cgf.12865.

[98] M. Foskey, M.C. Lin, D. Manocha, Efficient computation of a simplified medial axis, in: Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications, 2003: pp. 96–107. https://doi.org/10.1115/1.1631582.

[99] O. Ibhadode, Z. Zhang, A. Bonakdar, E. Toyserkani, IbIPP for topology optimization—An Image-based Initialization and Post-Processing code written in MATLAB, SoftwareX. 14 (2021). https://doi.org/10.1016/j.softx.2021.100701.

[100] S. Liu, Q. Li, J. Liu, W. Chen, Y. Zhang, A Realization Method for Transforming a Topology Optimization Design into Additive Manufacturing Structures, Engineering. 4 (2018) 277–285. https://doi.org/10.1016/j.eng.2017.09.002.

[101] J.M. Chacón, J.C. Bellido, A. Donoso, Integration of topology optimized designs into CAD/CAM via an IGES translator, Structural and Multidisciplinary Optimization. 50 (2014) 1115–1125. https://doi.org/10.1007/s00158-014-1099-6.

[102] L. Li, Y. Zheng, M. Yang, J. Leng, Z. Cheng, Y. Xie, A survey of feature modeling methods : Historical evolution and new development, Robotics and Computer Integrated Manufacturing. 61 (2020) 101851. https://doi.org/10.1016/j.rcim.2019.101851.

[103] Z. Cheng, Y. Ma, A functional feature modeling method, Advanced Engineering Informatics. 33 (2017) 1–15. https://doi.org/10.1016/j.aei.2017.04.003.

[104] G. Brunetti, B. Golob, A feature-based approach towards an integrated product model including conceptual design information, Computer-Aided Design. 32 (2000) 877–887. https://doi.org/https://doi.org/10.1016/S0010-4485(00)00076-2.

[105] Y.S. Ma, T. Tong, Associative feature modeling for concurrent engineering integration, Comput Ind. 51 (2003) 51–71. https://doi.org/10.1016/S0166-3615(03)00025-3.

[106] Y.S. Ma, G.A. Britton, S.B. Tor, L.Y. Jin, Associative assembly design features: Concept, implementation and application, International Journal of Advanced Manufacturing Technology. 32 (2007) 434–444. https://doi.org/10.1007/s00170-005-0371-8.

[107] J. Liu, Z. Cheng, Y. Ma, Product design-optimization integration via associative optimization feature modeling, Advanced Engineering Informatics. 30 (2016) 713–727. https://doi.org/10.1016/j.aei.2016.09.004.

[108] J. Ren, G. Zhang, Y. Rong, Y. Ma, A feature-based model for optimizing HVOF process by combining numerical simulation with experimental verification, J Manuf Process. 64 (2021) 224–238. https://doi.org/10.1016/j.jmapro.2021.01.017.

[109] L. Li, C.F. Lange, Y. Ma, Association of design and computational fluid dynamics simulation intent in flow control product optimization, Proc Inst Mech Eng B J Eng Manuf. (2017) 095440541769735. https://doi.org/10.1177/0954405417697352.

[110] L. Li, C.F. Lange, Z. Xu, P. Jiang, Y. Ma, Feature-based intelligent system for steam simulation using computational fluid dynamics, Advanced Engineering Informatics. 38 (2018) 357–369. https://doi.org/10.1016/j.aei.2018.08.011.

[111] Y. Xie, Y. Ma, Design of a multi-disciplinary and feature-based collaborative environment for chemical process projects, Expert Syst Appl. 42 (2015) 4149–4166. https://doi.org/10.1016/j.eswa.2015.01.009.

[112] Y. Ma, G. Chen, G. Thimm, Change propagation algorithm in a unified feature modeling scheme, Comput Ind. 59 (2008) 110–118. https://doi.org/10.1016/j.compind.2007.06.006.

[113] L. Li, J. Liu, Y. Ma, R. Ahmad, A. Qureshi, Multi-view feature modeling for design-for-additive manufacturing, Advanced Engineering Informatics. 39 (2019) 144–156. https://doi.org/10.1016/j.aei.2018.12.004.

[114] N. Geren, O. Oktay Akçalı, E. Unver, J. Allport, Automated sizing of automotive steering ball joints in parametric CAD environment using expert knowledge and feature-based computer-assisted 3D modelling, Advanced Engineering Informatics. 52 (2022) 101630. https://doi.org/https://doi.org/10.1016/j.aei.2022.101630.

[115] R.K. Gupta, B. Gurumoorthy, Feature-based ontological framework for semantic interoperability in product development, Advanced Engineering Informatics. 48 (2021) 101260. https://doi.org/https://doi.org/10.1016/j.aei.2021.101260.

[116] S. Benjamin, R. Christopher, H. Carl, Feature modeling for configurable and adaptable modular buildings, Advanced Engineering Informatics. 51 (2022) 101514. https://doi.org/https://doi.org/10.1016/j.aei.2021.101514.

[117] Tianyu Zhou, Weidan Xiong, Yuki Obata, Carlos Lange, Yongsheng Ma, Chapter 2 - Digital product design and engineering analysis techniques, in: Digital Manufacturing The Industrialization of "Art to Part" 3D Additive Printing, Elsevier, 2022: pp. 58–96. https://doi.org/https://doi.org/10.1016/B978-0-323-95062-6.00003-6.

[118] C. Patel, The Case of Art to Part: The Rise of Systemic Intent Based Design with 3D Digital Manufacturing Engineering, (2019). https://www.engineering.com/story/the-case-of-art-to-part-the-rise-of-systemic-intent-based-design-with-3d-digital-manufacturing.

[119] J. Wu, O. Sigmund, J.P. Groen, Topology optimization of multi-scale structures: a review, Structural and Multidisciplinary Optimization. 63 (2021) 1455–1480. https://doi.org/10.1007/s00158-021-02881-8.

[120] J. ZHU, H. ZHOU, C. WANG, L. ZHOU, S. YUAN, W. ZHANG, A review of topology optimization for additive manufacturing: Status and challenges, Chinese Journal of Aeronautics. 34 (2021) 91–110. https://doi.org/10.1016/j.cja.2020.09.020.

[121] D. Hearn, M.P. Baker, Computer Graphics - C version, (1996) 652.

[122] B.B. Mandelbrot, Fractal geometry: what is it, and what does it do ?, Fractals in the Natural Sciences. (2014) 3–16. https://doi.org/10.1515/9781400861040.3.

[123] K. Falconer, Fractal geometry : mathematical foundations and applications., 3rd edition, Wiley, 2014.

[124] N. Aslan, M. Saltan, B. Demir, A different construction of the classical fractals via the escape time algorithm, Journal of Abstract and Computational Mathematics. 3 (2018) 1–15. http://www.ntmsci.com/jacm.

[125] S. Wannarumon, An aesthetics driven approach to jewelry design, Comput Aided Des Appl. 7 (2010) 489–503. https://doi.org/10.3722/cadaps.2010.489-503.

[126] Vadim Shapiro, Solid Modeling, in: Handbook of Computer Aided Geometric Design, 1st ed., North Holland, 2002: pp. 473–518.

[127] W. Welch, A. Witkin, Variational Surface Modeling, Comput Graph (ACM). 26 (1992). papers2://publication/uuid/857D7828-119D-4A41-9F8F-F37956F63E8B.

[128] G. Elber, Metamorphosis of Freeform Curves and Surfaces, ACADEMIC PRESS LIMITED, 1995. https://doi.org/10.1016/b978-0-12-227741-2.50007-4.

[129] K.M. Yu, K.M. Lee, K.M. Au, Regularized set operations in solid modeling revisited, Comput Aided Des Appl. 17 (2020) 1084–1100. https://doi.org/10.14733/cadaps.2020.1084-1100.

[130] K.J. Vinoy, K.A. Jose, V.K. Varadan, Multi-band characteristics and fractal dimension of dipole antennas with Koch curve geometry, in: IEEE Antennas and Propagation Society, AP-S International Symposium (Digest), 2002: pp. 106–109. https://doi.org/10.1109/aps.2002.1016937.

[131] G. Liu, J. Gu, Z. Gao, M. Xu, Wideband printed slot antenna using Koch fractal metasurface structure, International Journal of RF and Microwave Computer-Aided Engineering. 30 (2020). https://doi.org/10.1002/mmce.22058.

[132] C. Sharma, D. Vishwakarma, Miniaturization of logarithmic spiral antenna with modified Koch curve, Microw Opt Technol Lett. 60 (2018) 2167–2172. https://doi.org/https://doi.org/10.1002/mop.31321.

[133] M. Manohar, Miniaturised low-profile super-wideband Koch snowflake fractal monopole slot antenna with improved BW and stabilised radiation pattern, IET Microwaves, Antennas & Propagation. 13 (2019) 1948–1954. https://doi.org/https://doi.org/10.1049/iet-map.2019.0116.

[134] D.E. Anagnostou, J. Papapolymerou, M.M. Tentzeris, C.G. Christodoulou, A printed log-periodic Koch-dipole array (LPKDA), IEEE Antennas Wirel Propag Lett. 7 (2008) 456–460. https://doi.org/10.1109/LAWP.2008.2001765.

[135] G. Liu, L. Xu, Z. Wu, Miniaturised wideband circularly-polarised log-periodic Koch fractal antenna, Electron Lett. 49 (2013) 1315–1316. https://doi.org/https://doi.org/10.1049/el.2013.2418.

[136] Y. Ma, S.B. Tor, G.A. Britton, The development of a standard component library for plastic injection mould design using an object-oriented approach, Int J Adv Manuf Technol. 22 (2003) 611–618. https://doi.org/10.1007/s00170-003-1555-8.

[137] K. Rajaguru, T. Karthikeyan, V. Vijayan, Additive manufacturing-State of art, in: Mater Today Proc, Elsevier Ltd, 2020: pp. 628–633. https://doi.org/10.1016/j.matpr.2019.06.728.

[138] C. Zhang, S. Xu, J. Liu, Y. Ma, Comprehensive clustering-based topology optimization for connectable multi-scale additive manufacturing structures, Addit Manuf. 54 (2022). https://doi.org/10.1016/j.addma.2022.102786.

[139] J. Gao, Z. Luo, L. Xia, L. Gao, Concurrent topology optimization of multiscale composite structures in Matlab, Structural and Multidisciplinary Optimization. 60 (2019) 2621–2651. https://doi.org/10.1007/s00158-019-02323-6.

[140] T.E. Bruns, D.A. Tortorelli, Topology optimization of non-linear elastic structures and compliant mechanisms, n.d. www.elsevier.com/locate/cma.

[141] O. Sigmund, A 99 line topology optimization code written in matlab, Structural and Multidisciplinary Optimization. 21 (2001) 120–127. https://doi.org/10.1007/s001580050176.

[142] S. Rojas-Labanda, M. Stolpe, Benchmarking optimization solvers for structural topology optimization, Structural and Multidisciplinary Optimization. 52 (2015) 527–547. https://doi.org/10.1007/s00158-015-1250-z.

[143] Y. Agrawal, G.K. Ananthasuresh, Towards optimal heterogeneity in lattice structures, Structural and Multidisciplinary Optimization. 64 (2021) 2489–2512. https://doi.org/10.1007/s00158-021-03003-0.

[144] T. Lewiński, T. Sokół, C. Graczykowski, Michell structures, Springer, 2018.

[145] L. Xia, P. Breitkopf, Design of materials using topology optimization and energy-based homogenization approach in Matlab, Structural and Multidisciplinary Optimization. 52 (2015) 1229–1241. https://doi.org/10.1007/s00158-015-1294-0.

[146] L. Liu, J. Yan, G. Cheng, Optimum structure with homogeneous optimum truss-like material, Comput Struct. 86 (2008) 1417–1425. https://doi.org/https://doi.org/10.1016/j.compstruc.2007.04.030.

Appendix Pseudocode for automatically constructing a CAD fractal model with variable design parameters

Algorithm: Automatic CAD model construction of an extruded Koch Snowflake with variable parametric angle of the snow twig using NX Open API

Inputs:

L	Length of the primitive segment for the fractal part
N	Machine resolution
angleA	Parametric angle of the snow twig
StretchLength	Length of extrusion from the sketch

Output: A parametric CAD fractal part model with desired iteration level

//Prepare useful data structures and related functions

//function A(float a) checks if a float number a is close enough to an integer with the predefined very small threshold

Function bool A(float a)

{

return True if (abs(a-(int)a) < threshold)

return False in other cases

}

EndFunction

//data structure vPoint represents a 2D point (x, y) with the default initialized value (0,0)

Struct vPoint

{

double x=0;

double y=0;

}

EndStruct

//data structure vLine represents and initializes a 2D line segment with attributes of start point, end point, orientation angle, connected next line segment

Struct vLine

{

```
        vPoint start, end;
        double angle=0, length=0;
        struct vLine * next = NULL;
    }
```

EndStruct

//function getPoint(vPoint * currentPoint, double assignX, double assignY) assigns the current 2d point with position(assign X, assignY)

```
    Function vPoint getPoint(vPoint * currentPoint, double assignX, double assignY)
    {
        currentPoint->x = assignX;
        currentPoint->y = assignY;
        return *currentPoint;
    }
```

EndFunction

//function getDist(vPoint P1, vPoint P2) calculates the Euclidean distance between point P1 and P2

```
    Function double getDist(vPoint P1, vPoint P2)
    {
        double dist = 0;
```

```
        Calculate dist as the Euclidean distance between P1 and P2;
        return dist;
    }
    EndFunction
```

//function getAngle(vPoint P1, vPoint P2) calculates the orientation angle of a line segment starts with P1 and ends with P2, the range is between [-pi/2, pi/2]

```
    Function double getAngle(vPoint P1, vPoint P2)
    {
        double angle;
        Calculate angle as the orientation angle of the line segment;
        Rectify the angle within the range [-pi/2, pi/2];
        return angle;
    }
    EndFunction
```

//function getLine(vLine * currentLine, vPoint assignStart, vPoint assignEnd) sets up the object line segment with the start point as assignStart and end point as assignEnd and calculates the attributes of the line segment with respect to length and orientation angle

```
    Function vLine getLine(vLine * currentLine, vPoint assignStart, vPoint assignEnd)
```

```
        {
            currentLine->start = assignStart;
            currentLine->end = assignEnd;
            currentLine->length = getDist(currentLine->start, currentLine->end);
            currentLine->angle = getAngle(currentLine->start, currentLine->end);
            return *currentLine;
        }
        EndFunction


//Initialize NXOpen environment
    Initialize NXOpen::Session *theSession
    Initialize NXOpen::Part *workPart
    Initialize NXOpen::Part *displayPart


//Get number of Features
    b=allfeatures.size()  //Set allfeatures.size to b


//Reinitialize and update the sketch if the design parameters have been updated
    If (b>3)
```

Delete all *Features* of *workpart*

Initialize *NXOpen::Sketch *sketch1*

EndIf

//Initialize the sketch parameters of the fractal model

Set *NXOpen::Point3d origin1* to *(0.0, 0.0, 0.0)*

Set *NXOpen::Vector3d normal1* to *(0.0, 0.0, 1.0)*

Initialize *CreatePlane*

Initialize *NXOpen::Unit *unit1*

Initialize *SetRightHandSide("0")*

Initialize *NXOpen::Sketch *sketch1*

Initialize *NXOpen::Features::Feature *feature1*

Set name of *ActiveSketch* to *SKETCH_SNOW*

Set *LValue* to *L*

Set *NValue* to *N*

Set *AngleValue* to *angleA*

//Calculate the maximum iteration number as *NNewValuie*

//Use the equation $N_{max} = \text{floor}\left(\frac{\log\left(\frac{D_p}{D_m}\right)}{\log E}\right)$

Set *tmp* to log(L/N)/log[2(1+cos(angleA))]

If *A(tmp) != 0*

NNewValue = (int)tmp + 1;

Else

NNewValue = (int)tmp;

EndIf

//Initialize length of extrusion, length of the fractal primitive line segment, switch unit of the parametric angle of the snow twig from degree to radius

Set *ExtrudeLength* to *StretchLength*

Set *length* to *LValue*

Set *theta* to *AngleValue*pi/180*

//Apply the iterative generation method to each line segment of the current fractal sketch and generate a new fractal sketch after the iteration

data structures:

currentLine an existing array of sequential connected line segments of the current fractal sketch model

newLines a new array of sequential connected line segments of the new generatated fractal

sketch model after an iteration step

variables:

count the number of line segments of the current fractal sketch model

newStart start point of a line segment of the newly generated fractal sketch

newAngle orientation angle of a line segment of the newly generated fractal sketch

newLength length of a line segment of the newly generated fractal sketch

newEnd end point of a line segment of the newly generated fractal sketch

i the (i+1)th line segment of the current fractal sketch model

//Calculate the total number of line segments of the current fractal sketch and set the value to the variable *count*

Set 0 to the variable *count* and *i*

While *(currentLine + i)->next != NULL* //There exists a line segment that connects to the current line segment in the desired sequence

Add 1 to i

Add 1 to count

EndWhile

//Dynamically allocate memory to the new fractal sketch model *newLines* as *(vLine *)malloc(4*

** count * sizeof(vLine))*

newLines = (vLine *)malloc(4 * count * sizeof(vLine));

//Dynamically apply the current iteration generators with the design parameter variable *theta* of snow twig angle to each sequential connected line segments of the current fractal sketch *currentLine* to generate the fractal sketch *newLines* after an iteration step

For (i = 0; (currentLine + i)->next != NULL; i++)

//Construct the 1st line segment of the new fractal from the current fractal

Set *newStart* to *(currentLine + i)->start*

Set *newAngle* to *(currentLine + i)->angle*

Set *newLength* to *((currentLine + i)->length) / (2 * (1 + cos(theta)))*

Set *newEnd* to *(newStart.x + newLength * cos(newAngle), newStart.y + newLength * sin(newAngle))*

//contruct the 1st line segment

*(newLines+4*i+0)=getLine((newLines+4*i+0), newStart, newEnd);

(newLines + 4 * i + 0)->next = (newLines + 4 * I + 1);

//Construct the 2nd line segment of the new fractal from the current fractal

Set *newStart* to *(newLines +4* i+0)->end*

Set *newAngle* to *(currentLine + i)->angle+theta*

Rectify *newAngle* to range [-pi/2, pi/2]

Set *newEnd* to *(newStart.x + newLength * cos(newAngle), newStart.y + newLength * sin(newAngle))*

//contruct the 2nd line segment

*(newLines+4*i+1)=getLine((newLines+4*i+1), newStart, newEnd);

(newLines + 4 * i + 1)->next = (newLines + 4 * i + 2);

//Construct the 3rd line segment of the new fractal from the current fractal

Set *newStart* to *(newLines + 4*i+1)->end*

Set *newAngle* to *(newLines + 4*i+1)->angle-2*theta*

Rectify *newAngle* to range [-pi/2, pi/2]

Set *newEnd* to *(newStart.x + newLength * cos(newAngle), newStart.y + newLength * sin(newAngle))*

//contruct the 3rd line segment

*(newLines+4*i+2)=getLine((newLines+4*i+2), newStart, newEnd);

(newLines + 4 * i + 2)->next = (newLines + 4 * i + 3);

//Construct the 4th line segment of the new fractal from the current fractal

Set *newStart* to *(newLines + 4*i+2)->end*

Set *newEnd* to *(currentLine + i)->end*

//contruct the 4th line segment

*(newLines+4*i+3)=getLine((newLines+4*i+3), newStart, newEnd);

(newLines + 4 * i + 3)->next = (newLines + 4 * i + 4);

EndFor

Set *(newLines + 4 * i + 3)->next* to NULL for the last line segment of the current fractal

//Get a point set as sequencial vertices of the current fractal sketch model

data structures:

pointSet an array of sequential vertices of the current fractal sketch model

currentLine an existing array of sequential connected line segments of the current fractal sketch model

variables:

count the number of vertices of the current fractal sketch model

i the (i+1)th line segment of the current fractal sketch model

//Count the number of vertices of the current fractal model

For (i = 0, count=2; (currentLine + i)->next != NULL; i++)

{

Add 1 to *count*

```
    }
    EndFor


//Gather and store the sequential vertices of the fractal model to array pointSet
    Allocate memory (vPoint *)malloc(count * sizeof(vPoint) to pointset
    If (count==2)  //There is only one line segment (2 vertices) in the model
    {
    //Add the start point and end point of the only line segment to the sequential point set.
        *(pointSet+0)=currentLine->start;
        *(pointSet+2)=currentLine->end;
    }
    Else
    {
    //Add the start point of each line segment (except for the last line segment) to the
sequential point set
        For (i = 0; (currentLine + i)->next != NULL; i++)
        {
            *(pointSet + i) = (currentLine + i)->start;
        }
```

EndFor

// Add the start point and end point of the last line segment to the sequential point set

*(pointSet + i) = (currentLine + i)->start;

*(pointSet + i + 1) = (currentLine + i)->end;

}

EndIf

//Insert the fractal curve according to the *pointset*

For (i=0;i<count-1;i++)

{

Set x1 to (pointSet + i)->x;

Set y1 to (pointSet + i)->y

Set x2 to (pointSet + i + 1)->x

Set y2 to (pointSet + i + 1)->y

Insert x1,x2,y1,y2 to the sketch curve

}

EndFor

//pattern the curve to form a snowflake

Initialize *NXOpen::Features::PatternFeatureBuilder* patternFeatureBuilder1*

Set the origin of pattern to *(0, 0, 0)*

Set the vector of pattern to *(0, 0, 1)*

Set pattern as circle *PatternEnunCircular*

Set the copies of circle as *3*

Set the angle of pattern as *120* deg

//Select sketch feature to pattern

Set *NXOpen::Features::SketchFeature* sketchFeature1* to*FindObject("SKETCH(1)")*

Set reference point as *coordinates1(0.0, 0.0, 0.0)*

Set the center point of pattern as *helpPoint1(LValue/2, -(LValue / 2/ tan(pi / 3)), 0)*

//Finish sketch

Set *NXOpen::NXObject* nXObject1a* as *patternFeatureBuilder1->Commit()*

//Extrude the sketch to construct a 3D extrusion model if required

If (Extrude the sketch==True) //An extrusion is required by the end user

{

If (*Check3DValue==True*) //The sketch is well defined and closed for extrusion

{

Select sketch features *NXOpen::Features::Feature* nullNXOpen_Features_Feature1*

Create extrude features *NXOpen::Features::ExtrudeBuilder* extrudeBuilder1*

Set *extrudeBuilder1* to *CreateExtrudeBuilder(nullNXOpen_Features_Feature1)*

Select the type of operation

SetType(NXOpen::GeometricUtilities::BooleanOperation::BooleanTypeCreate)

Set *startExtend* to *(0)*

Set *endExtend* to *(to_string(ExtrubeLength))*

//Select the features to extrude

workPart->Features()->FindObject("SKETCH(1)")

}

EndIf

}

EndIf