Design of A Paint Simulation and Visualization Tool for Automotive Surfaces

Yuchen Luo

SURFACE AND TRIBOLOGY
FACULTY OF ENGINEERING TECHNOLOGY

Thesis committee:
Prof.dr.ir. D.J.Schipper
Dr. M.B.de Rooij
Dr. D.T.A. Matthews
Prof.dr.ir. T. Tinga
Dr. M. Toose

Design of A Paint Simulation and Visualization Tool for Automotive Surfaces

by

Yuchen Luo

PDEng Candidate
at the University of Twente,
to be defended publicly on Wednesday July 12, 2017 at 13:00.

Institute: University of Twente
Faculty: Engineering Technology
Trajectory: Maintenance
Case Study Organization: TATA Steel

Employee number:	M7662067	
Student number:	S1764527	
Project duration:	August 1, 2015 – July 31, 2017	
Thesis committee:	Prof. dr. ir. D. J. Schipper,	University of Twente, Programme director
	Dr. ir. M. B. de Rooij,	University of Twente, Daily supervisor
	Dr. D. T. A. Matthews,	TATA Steel & University of Twente, Company supervisor
	Prof. dr. ir. T. Tinga,	University of Twente, External examiner
	Dr. M. Toose,	TATA Steel, External examiner

This thesis is confidential and cannot be made public until July, 2017.

UNIVERSITY OF TWENTE. TATA STEEL

Preface

This report elaborates my two years PDEng project in University of Twente. Along this journey, I received plenty of help and support from my supervisors and my colleagues, I would like to express my deep gratitude at this graduation moment.

I would like to thank my supervisors in this project. I would like to thank Professor Dirk Schipper firstly, who always gives his support and the chance for me to pursue this graduation.

I would like to thank my daily supervisor Matthijn de Rooij, who always gives me his unreserved help in my daily project process. The fruitful discussions and the constructive suggestions from him are indispensable factors for the success of the project. He gives responsible and excellent supervision by always mastering the process and direction of the project and helping me make things possible with constructive and practical ideas.

I would also like to thank my company supervisor, David Matthews, who also helps and supports me a lot during the whole project from the company side. With his supervision, I can understand TATA steel's project goal and transfer it into concrete requirements. He is also responsible for arranging regular meetings with TATA steel, which makes the regular communication between company and me possible.

I would like to thank Matthijs Toose from TATA steel, who give me plenty of support at the early stage of my project. The important references he provided helps me to define the problem explicitly and get closer to the core of the project. I would also like to thank Eric, who helps me a lot during my validation stage. There would be much more difficult to handle the newly bought microscopy machine in our lab without Eric's help. He gives me necessary training and answering my questions regrading substrate measurement.

I would also like to thank my old and current office mates, especially Xiaver, Mohammed and Tanmaya, with whom I also had lots of fruitful discussions. I would thank Xiaver for his help and suggestions on COMSOL implementation and thank Mohanmmed for his help on mathematical model construction.

I would like to thank Aydar and Shivam for their help and working together on COMSOL-MATLAB interaction problem. It is our discussion and communication that helps me avoid unnecessary struggling.

At last, I would thank my parents who always support me from China, they trust me and my decisions, it was them who always make me believe that I can overcome any challenge and can always go further. Thanks!

Yuchen Luo
University of Twente, July 2017

Abbreviation List

- **SR**: Stakeholders' requirements
- **FR**: Functional requirements
- **A3AO**: A3 Architecture Overview
- **SA**: System Architecture
- **PSVT**: Paint Simulation and Visualization Tool
- **RSG**: Rough Substrate Generator

Contents

1

Project context

In this chapter, the context of my PDEng project will be firstly introduced. The background knowledge and several key conceptions will be explained in this chapter. Meanwhile, this chapter also focuses on the definition of the system goal and stakeholders requirements. Based on the relevant definition, the top requirements of my PDEng project will be extracted. Furthermore, in this chapter the structure of this report has been displayed, the introduction and main contents of each following chapter are concluded as well.

1.1. Introduction

In automotive industry, the surface aspect of a part or product is an important facet in terms of product performance and sale-ability. The performance of a surface is related to its functionality, including tribological contact ability, maintenance ability, the aesthetic appearance of a product and so on. Revealing the relation between the intrinsic steel surface products properties and its functionalities will provide necessary reference and support for improving products quality. To achieve this, our further discussion will be firstly based on an explicit introduction to several key concepts including **technological or engineering surface**, **surface functionality** and **surface texture**.

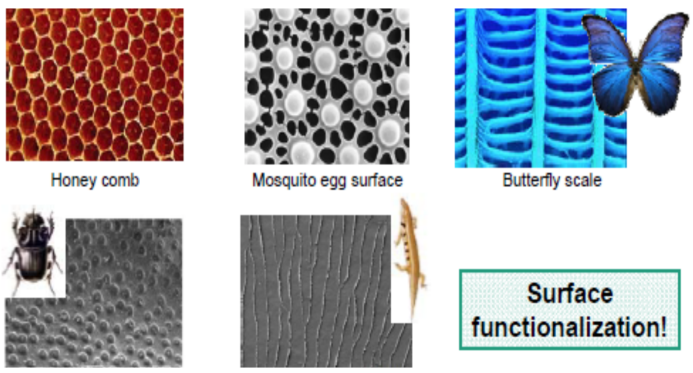


Figure 1.1: Functional surfaces in nature

Basically, a **technological or engineering surface** means any surface generated by manufacturing methods, such as cutting and grinding, forming and non-conventional material-removal processes (electro-discharge machining, water-jet, laser machining, etc.).[1][2] The

engineering surface achieves, after the relevant process, new properties and characteristics compared to the initial one. This definition of engineering surface can be also applied to automotive industry.[1]

The new properties and characteristics achieved for these manufactured surfaces will determine their **Surface functionalities**. A functional surface means a surface that can fulfill a certain functionality in nature or industry.[3] Figure,1.2 displays several surface functionalities that play important roles in automotive industry. The highlighted surface functionality, **Paint Appearance, which reflects the behavior of painted layer on certain textured surfaces from vision perspective**,will be the most interesting surface functionality of our PDEng project, this is because the paint appearance of an automotive product has the power to influence the perception of the consumers.

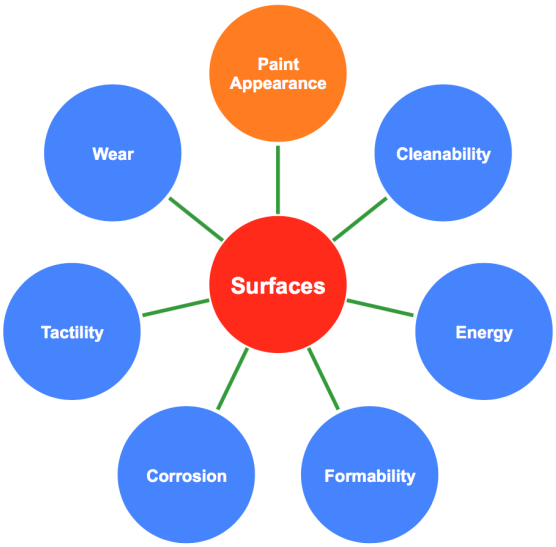


Figure 1.2: Various surface functionalities

Figure 1.3: Automotive product with excellent paint appearance

In most cases, one certain surface functionality is highly correlated to the surface's texture. **Surface texture**, also known as surface topography, is the nature of a surface as defined by the three characteristics of lay, surface roughness, and waviness. It comprises the small local deviations of a surface from the perfectly flat ideal. It is a foremost characteristic among the surface integrity magnitudes and properties imparted by the tools used in the processes, machining mostly, and especially their finishing versions [2]. The relevant parameters including surface roughness, surface waviness and plenty of other specific parameters are called **descriptors** of surface textures. The process of describing surfaces by utilizing these descriptors is called **surface characterization**. The surface characterization provides us a methodology to investigate our interested surface functionality, paint appearance, by a comprehensive process of simulation, visualization and analysis.

In summary, a PDEng project aiming at making a step change in paint appearance, from surface property perspective has been proposed. To be more precise, **at this stage the PDEng project will achieve this goal by revealing the relation between surface textures and our most interested surface functionality: surface paint appearance; a paint simulation and visualization tool will be developed to underpin such a process.**

1.2. Paint appearance of automotive surfaces

As mentioned in previous section, surface paint appearance is one of the most important surface functionalities that are interesting for automotive industry enterprise. The paint quality of automotive products will significantly influence customers' decision making. Therefore, the effort on improving paint appearance of automotive surfaces is consistent.

Figure.1.4 shows a comparison between automotive products with excellent paint appearance and another with poor paint appearance[4]. The quality of automotive paint can be described by parameters. Figure.1.5 displays two automotive paint surfaces with different Distinctness of Image (DOI), which is one typically used standard to judge surface paint quality[4]. Another example of poor paint quality is shown in Figure.1.6, which is called

(a) (b)

Figure 1.4: Car with different paint quality

"Orange peel". The unqualified paint process and poor quality paint substrate lead to a dimpled paint layer instead of smooth one. Compared to a poor paint process, "Orange peel" situation is more significantly connected to the improper or unqualified textures of the steel substrate. Therefore, the importance of investigating the relation between surface

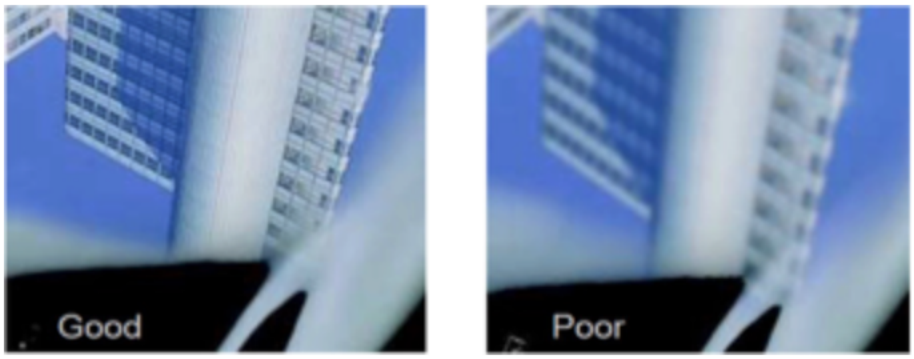


Figure 1.5: Comparison between good and poor Distinctness of Image

textures and its paint appearance is highlighted.

(a) (b)

Figure 1.6: "Orange peel" in automotive paint

1.3. System goal and stakeholders requirements

From the previous project context it's easy to define the three major stakeholders as follows:

- **The international automotive enterprise**
- **TATA steel**
- **University of Twente, surface and tribology group**

The original motivation of my PDEng project comes from the top stakeholder's, namely the international automotive enterprise's goal: **To improve the car manufacturing procedure by improving the relevant steel material properties and quality.** From TATA steel's perspective, the top stakeholder's goal is specified and has been translated into plenty of stakeholder's requirements. Several examples of stakeholder's requirements are shown as follows:

TATA steel shall be able to:

1. SR1: Generate tailored steel surfaces that fit the particular requirements of tasks without loss of their corrosion properties.

2. SR2: Realize the surface processing and manufacturing while a cost effective and environmental sound way is considered.

3. SR3: Control the steel products quality in order to fulfill certain customers' maintenance requirements.

4. SR4: Revealing the relation between steel surface textures and its certain surface functionalities.

5. SR5: Produce steel surfaces with certain surface textures to fulfill certain surface functionalities.

6. SR6: Improve the steel-based products with surface functionalities including paint appearance, press performance, friction performance and so on.

7. **SR7: Design a surface generator and paint appearance simulation & visualization tool to underpin the study of textured steel surface paint appearance.**

One key indicator of the steel products quality is whether the products can fulfill its required surface functionality. Therefore, the core of the stakeholders' requirements falls on SR4 and SR5. Meanwhile, SR7 stakeholders requirement:

SR7: Design a surface generator and paint appearance simulation & visualization tool to underpin the study of textured steel surface paint appearance.

is actually the top goal (top requirement) of my PDEng project. Therefore, it can be realized that compared to other surface functionalities, paint appearance is the major topic we are going to investigate in current time period of my PDEng project.

As discussed in Section.1.2, paint appearance is one of the most important indicators in automotive industry that can influence customers' decisions. In practice, this situation motivates the automotive manufacturing enterprise to propose rigorous requirements on steel products for paint. Figure.1.7 displays the bidirectional relation between stakeholders requirements and our PDEng-designed paint tool. From Figure.1.7 it can be seen that automotive manufacturing companies decide their potential consumption based on their requirements on steel surface's paint appearance. Ideally, TATA steel should be able to produce steel products that based on these stakeholders' requirements. However, in practice it is still difficult to realize this path since the problem that how to properly quantify steel products properties based on stakeholders abstract requirements is not completely solved yet. This also means some relations between surface functionalities and certain surface properties are still vague. Therefore, an inverse path is proposed in Figure.1.7. This path aims at providing reference, advices to stakeholders on their requirements intention by analyzing paint appearance of our finished products. In another word, the paint simulation tool helps us to find out which set of finished products can satisfy stakeholders' requirements on paint appearance to the maximum extent.

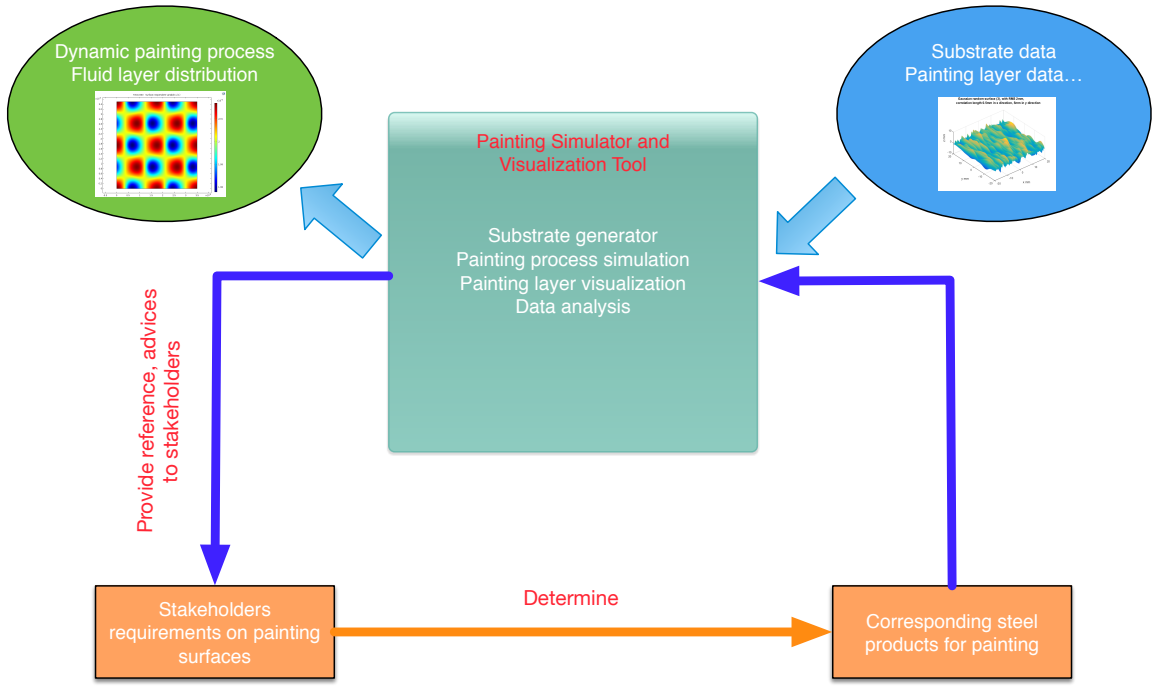


Figure 1.7: The relation between stakeholders requirements and paint simulation and visualization tool (PSVT)

1.4. Report Contents Overview

The project has been realized by following a design approach based on theories of system engineering. This report is organized as shown in Figure.1.8 , which also reflects our design approach.

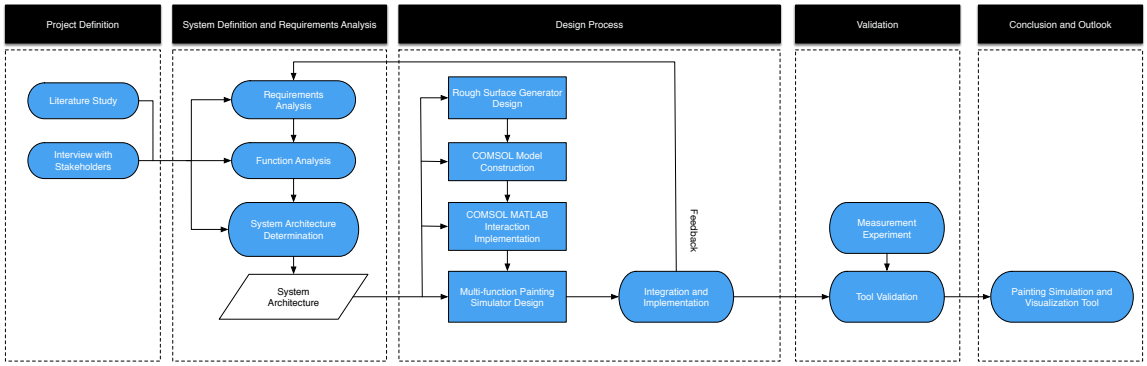


Figure 1.8: Design process and report structure overview

1. Chapter 1: Project definition and context
    - Literature study on surface functionalities
    - Stakeholders requirements and system goal definition

2. Chapter 2: Case study and background
    - Introduction on multi-layer paint system
    - Relation between paint layers and maintenance
3. Chapter 3: Design methodology for Paint Simulation and Visualization Tool (PSVT)
    - A plan of design process based on system engineering theory
    - Requirements definition based on the project goal
    - Propose a system (software) architecture
    - Function definition, categorization and analysis
4. Chapter 4: Mathematical model construction:
    - Construct mathematical models to describe painting process
    - Extract governing equations from mathematical models
5. Chapter 5: System architecture construction
    - Functional requirements definition and system decomposition
    - System architecture construction
6. Chapter 6: COMSOL model implementation
    - Build COMSOL model based on constructed mathematical models
    - Solve COMSOL model that simulating paint process on rough substrate
7. Chapter 7: Design of rough surface generator:
    - Investigation on descriptors (parameters, mathematical models, filters) that characterize a rough surface
    - Implementation of the rough surface generator
    - Implementation of data processing for later COMSOL-MATLAB interaction
8. Chapter 8: Design of painting simulator
    - Implementation of painting simulator with multi functions based on built software architecture
    - Implementation of painting simulator for multi-layer system
    - Realizing interaction between MATLAB and COMSOL
9. Chapter 9: Validation
    - Experimental validation
10. Chapter 10: Conclusion and outlook

2

Paint system case study

In this chapter, we are going to investigate the background knowledge of the automotive paint system. Multi-layer coating system for automotive will be firstly introduced. Furthermore, the typical E-coating process will also be introduced. Based on these knowledge, the role of paint system in maintenance will be introduced. The physical reality and relevant material properties for later simulation and validation will be investigated and defined in this chapter as well.

2.1. Automotive multi-layer coating system

Modern automotive coating process employs a multi-layer coating system to assure the performance of the automotive surfaces. Overall, the critical performance factors driving the development and use of advanced automotive coatings and coating technologies are **aesthetic characteristics**, **corrosion protection**, **mass production**, **cost and environmental requirements**, and **appearance and durability**.[6] Although the relative importance of each of these factors is debatable, the perfection of any one at the expense of another would be unacceptable. A multi-layer coating system considers the performance factors synthetically and usually each layer is able to fulfill one certain function so that improve and guarantee one corresponding performance factor.

Figure.2.1 shows a schematic drawing of an automotive multi-layer coating system. Corresponding to this defined multi-layer coating system, modern automotive coating methods consist of four main steps[6]:

1. The first step pretreatment removes and cleans excess metal and forms an appropriate surface structure enabling bonding of a corrosion protection layer. From Figure.2.1 it can be seen a zinc galvanized layer and a phosphating layer, which are used to prevent corrosion, are coated before E-coating layer.

2. The next step is electrodeposition (ED) of the anti-corrosion or rust prevention layer. This layer is also known as E-coating layer.

3. A primer is then applied to promote adhesion between the surface and the base coat; it also imparts a smoother surface for subsequent layers and has anti-chipping, leveling and UV resistance properties.

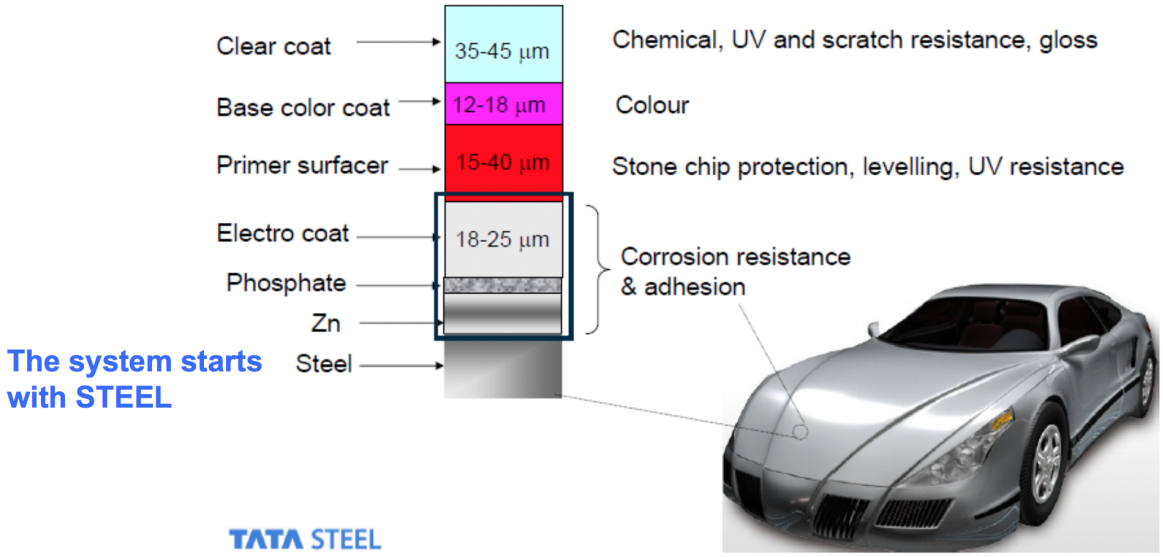


Figure 2.1: Multi-layer coating system

4. Finally, as shown in Figure.2.1, the topcoats that include a base color coat and clear coat are applied; they provide surface properties that are sought after, including color, appearance, gloss smoothness, and weather resistance.

2.2. E-coating process

As can be seen from Figure.2.1, and E-coat layer is on the top of phosphate layer. Together with phosphate layer and zinc galvanized layer, this three-layer system is used to realize the corrosion resistance & adhesion function.

E-coating is short for electro-coating, which is a method of painting by utilizing electrical current to deposit the paint. The process works on the principal of opposites attract, namely + charged particles and − charged particles will attract each other. A characteristic feature of this process is the colloidal particles suspended in a liquid medium migrate under the influence of an electric field and are deposited onto an electrode. The typical

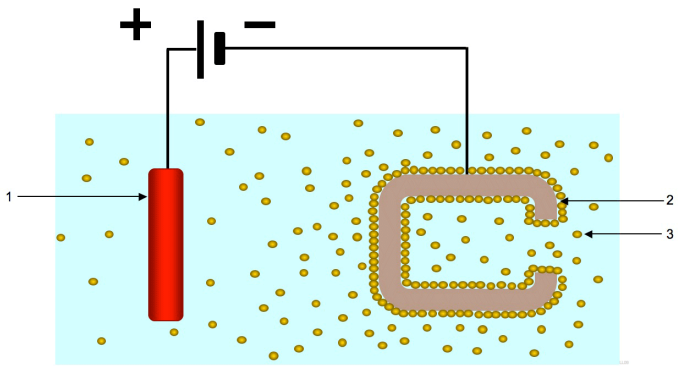


Figure 2.2: The opposites attract principal

e-coating process can be divided into four phases:

- **Pretreatment**
- **Electro-coat bath**
- **The post rinses**
- **The bake oven**

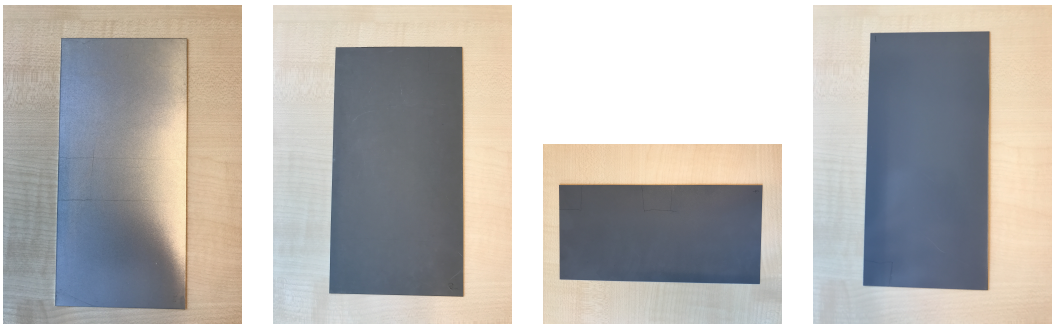
1. **Pretreatment:** The pretreatment zone cleans and phosphates the metal to prepare the surface for e-coating. Cleaning and phosphating are essential to achieving the performance requirements desired by end users of the product. This phase is also the very first phase for the whole surface coating procedure, as can be seen from Section.2.1.

2. **The Electro-coat bath:** The Electro-coat bath is where the coating is applied and the process control equipment operates. The e-coat bath consists of deionized water and paint solids. The deionized water acts as the carrier for the paint solids which are under constant agitation. Usually, the solids consist of resin and pigment. Resin is the backbone of the final paint film and provides corrosion protection, durability and toughness. Pigments are used to provide color and gloss [6].

3. **The post rinses:** The post rinses provide both quality and conservation. During the e-coat process, paint is applied to a part at a certain film thickness, regulated by the amount of voltage applied. Once the coating reaches the desired film thickness, the part insulates and the coating process slows down. As the part exits the bath, paint solids cling to the surface and have to be rinsed off to maintain efficiency and aesthetics. The excess paint solids are called "drag out" or "cream coat." These excess paint solids are returned to the tank to increase the coating application efficiency.

4. **The bake oven:** The bake oven receives the parts after they exit the post rinses. The bake oven cross links and cures the paint film to assure maximum performance properties.

Figure.2.3 shows e-coating painted steel surfaces after certain phases.



(a) Original substrate (b) Phosphate substrate (c) Painted substrate (d) Painted substrate after bake oven

Figure 2.3: Samples in different phases of multi-layer paint process

2.3. Coating system and maintenance

Steel paint system, or coating system, is widely applied in various situations for achieving certain goals. Among these applications, maintenance is a key factor that will be considered. When talking about maintenance of steel coating system, the meaning can be understood from two perspectives:

- Maintenance relevant design has been considered as an important factor in a steel multi-layer coating system
- When considered as an independent system, steel multi-layer coating system itself also needs corresponding maintenance.

The first perspective explains why phosphating layer exists in our automotive coating system. The main effect of phosphating layer is preventing corrosion. Corrosion is known as one typical failure mechanism from maintenance perspective. Without such a phosphating corrosion resistant layer, undesired chemical reaction may occur between top paint layers (eg. color layers) and the steel substrate. Initialized by corrosion, the more serious failure mechanism like corrosion-fatigue will continue and eventually cause damage to the automotive outer surface structure. Therefore, with certain design the maintenance consideration has been reflected in our multi-layer coating system implementation.

From another perspective, the multi-layer coating system itself also needs maintenance. This is not the typical situation for automotive multi-layer coating system, since when the multi-layer coating system loses its function, the automotive products usually reaches the end of its lifetime. Most of the customers will consider replace the automotive products instead of professionally maintaining the surface paint system isolatedly.

However, situation will be quite different in other realms. One example is that steel multi-layer coating system is also applied on a large number of infrastructures including bridges, tunnels, and storm-surge barriers. In these situations, steel multi-layer coating is considered as an independent and complete system for which corresponding maintenance strategy will be design.

For infrastructures like bridges, the primary function of the coating system on steel is protect the steel form degradation in terms of corrosion.[7] The failure mechanism of the coating system is a combined process of several factors including corrosion, cracking, thinning and so on.[7] The process of corrosion and degradation will result in loss of functionality of the coating system, hence corresponding maintenance strategy is required to determine the repair or replacement procedure of the coating system as well as its lifetime.

In infrastructure coating system, two measures including Life-Extending Maintenance (LEM) and Coating Replacement(CR) are usually carried out. The condition of the coating can be determined by visual inspection and maintenance will be performed when the intervention level is exceeded.[7] By synthetically considering key parameters including **cost parameters**, **deterioration parametrs**, **lifetime-extension parameters**, an optimized LEM is more preferable than CR. It can be realized that the strategy consideration and the relevant maintenance decision here is quite different to automotive relevant coating system.

2.4. Conclusion

In this chapter, the contexts regarding steel coating or paint system are introduced. The functionalities of each layer in a multi-layer coating system and the typical E-coating paint

process are discussed as well. Meanwhile, the maintenance considerations integrated in steel coating design has been noted; the difference among various steel coating system applications from maintenance perspective are also investigated.

3

Design methodology for Paint Simulation and Visualization Tool (PSVT)

This chapter focuses on discussing design methodologies applied for the Paint Simulation and Visualization Tool. By splitting a design process into three phases, this chapter majorly discusses the methodologies and models attributing to the so called Conceptual Design Phase. Based on the discussed design methodologies, the project will proceed by carrying out an implementation of multiple level requirements, especially system requirements.

3.1. Design methodologies

3.1.1. Design phases

The application of system engineering theory is based on an understanding of the products life-cycle process.[5] This life-cycle theory aims at decomposing one design process into several design phases. When doing so, three design phases in the systems design life-cycle known as conceptual design, preliminary design and detail design are proposed. These three design phases are adapt to all systems while designing, hence they are quite important.

- **Conceptual design phase** focuses on defining requirements and early stage functional analysis. It also refers to evaluation and optimize the design synthesis.
- **Preliminary design phase** focuses on further functional analysis as well as allocation of functions to subsystem. Meanwhile, in this phase early prototyping may be realized.
- **Detail design phase** focuses on component design and building prototype models, the verification of products is also realized in this phase.

A typical system engineering theoretical design process is shown in Figure.3.1. As a software development project, the design process will be different to that mentioned in Figure.3.1. For example, the production process will be integrated with design and development phase and will be known as implementation. Therefore, when the three phases design strategy has been utilized on our own project, the contents for each design phase have been specified as follows:

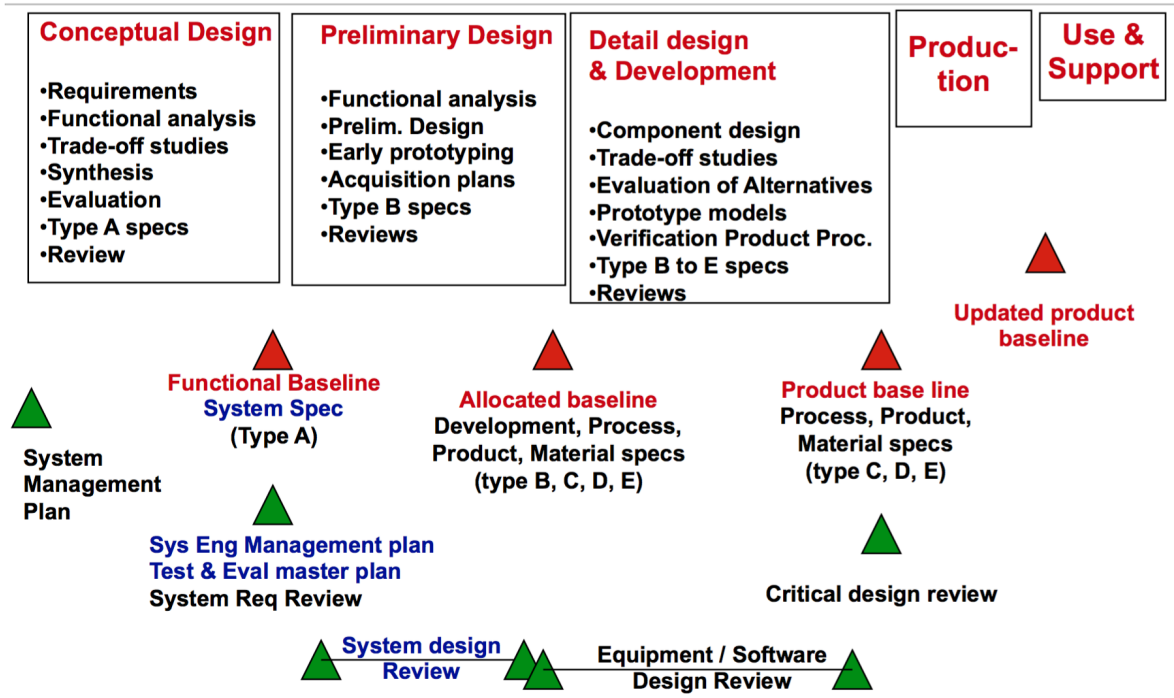


Figure 3.1: System engineering theoretical design process

- **Conceptual design phase**: In this phase, the multiple levels of requirements have to be defined explicitly. Typically, a project designed for a certain goal starts with stakeholders requirements. The well-defined stakeholders requirements will be extended to plenty of system requirements. Meanwhile, several methodologies, including A3AO diagram (as shown in Figure.3.3), will be utilized to realize the project's top-level design. Furthermore, the project implementation process will be organized by following selected certain design process models.

- **Preliminary design phase**: In this phase, we further investigate our defined system requirements in order to extract corresponding functional requirements. With clearly defined functional requirements, functional analysis will be carried out and the system will be decomposed based on categorization of various functions. This system decomposition provides us prerequisites for constructing a complete system architecture.

- **Detail design phase**: In this phase, detail design and development will be our major focus. This also means an component design as well as infrastructure implementation for our complete system.

In this chapter, it can be seen the topics discussed are mostly related to conceptual design phase contents. The preliminary design phase and detail design phase related contents will be discussed in later chapters.

3.1.2. Design models

The systems engineering process is usually organized by different kind of models. The common models often mentioned are Waterfall Process Model, Spiral Process Model, and Vee

Process Model.[5] As mentioned in Section.3.1, in principle, a certain design process model will be selected as a criterion in conceptual design phase. In our case, **we select Vee Process Model to guide our design process,** which is the most widely used design model nowadays.

As shown in Figure.3.2, the **Vee Process Model** starts with user needs on the upper left and ends with a user-validated system on the upper right. On the left side, decomposition and definition activities resolve the system architecture, creating details of the design. Integration and verification flows upward to the right as successively higher levels of subsystems are verified, culminating at the system level.

We select Vee model to guide our design process because firstly the system decomposition process can be clearly reflected from a Vee-model-utilized design process. As a software tool which consists of various modules to fulfill various system functions, how to allocate system functions to subsystems and construct a comprehensive system architecture is always one of our emphasis. Meanwhile, a Vee Process Model also proposes a verification sequence starting from infrastructure components to subsystems and ending with full system. This route is quite pragmatic and is also preferable for software development design process. Furthermore, a well-organized Vee Model can help us with more efficient time management, the simplified time period information for each phase is already reflected from the shape of the Vee Model. Therefore, Vee Process Model becomes our choice for design implementation.

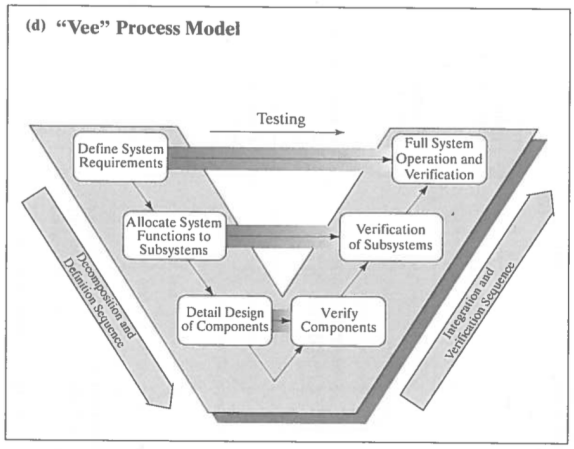


Figure 3.2: Vee Process Model [5]

Nevertheless, it should be noted that although a Vee Process Model provides good guidance and reference to our design process, necessary flexibility is also important when fulfilling our system implementation in practice.

3.1.3. A3AO diagram of the system

Several design methodologies have been utilized to realize the project's top-level design. In particular, from system thinking perspective, a so called A3AO diagram can be employed for early stage project design and management. A3AO is short for A3 Architecture Overview, it is a tool meant for effective communication of architecture knowledge. The A3AO method has the potential to give the software development a good understanding and insight for later construction of system architecture.

In Figure.3.3 an A3AO applied for our design project has been displayed. Regarding this A3AO diagram, some comments are stated as follows:

- As can be seen from Figure.3.3, a typical A3AO diagram includes **Functional flow**,**Visual aids**,**Design decisions and constraints**, **Quantification of key parameters** and **Physical view**.
- From this A3AO, it can be seen that the focus of our project consists of both surfaces generator design and paint simulation tool design, which will be two most important components later in our system.
- MATLAB will be our choice to implement the software tool, especially the user interface. However, the background computation may be fulfilled by some other softwares, which will be discussed in later Chap.4.
- The contents in the A3AO do not cover all aspects of our project, in most time, it gives an example for explaining certain alternatives or decisions.

As a design methodology applied at the very early stage of our design process, some detail decisions or alternatives implemented in the later practical design phases are not reflected in A3AO diagram. However, it gives a early stage guidance for our overall design as well as a clear top view of our system, which also provides valuable reference for later system architecture construction.

3.2. Requirements definition

In Chap.1, the stakeholders requirements are generally discussed. Among plenty of stakeholders requirements, SR7 becomes the top requirements for our PDEng project:

SR7: Design a surface generator and paint appearance simulation & visualization tool to underpin the study of textured steel surface paint appearance.

In Chap.2, a case study to steel surface paint process has been done. Therefore, based on the relevant physical reality and practical simulation and visualization goal we want to achieve, the corresponding system requirements are defined as follows:

1. **R1: The system should be able to generate various type of surfaces.**
2. **R2: The system should be able to read measured surfaces data.**
3. **R3: The system should be able to visualize generated surfaces.**
4. **R4: The system should be able to fulfill relevant data process, including data conversion and interaction between various software platform.**
5. **R5: The system should be able to analyze relevant statistics of paint process and fluid layer.**
6. **R6: The system should be able to simulate the paint process.**
7. **R7: The system should be able to visualize paint process.**

The system requirements will be the starting point of the PDEng project. In the later chapters, these system requirements will be specified and decomposed in to function requirements, which directly determine our design process and system architecture.

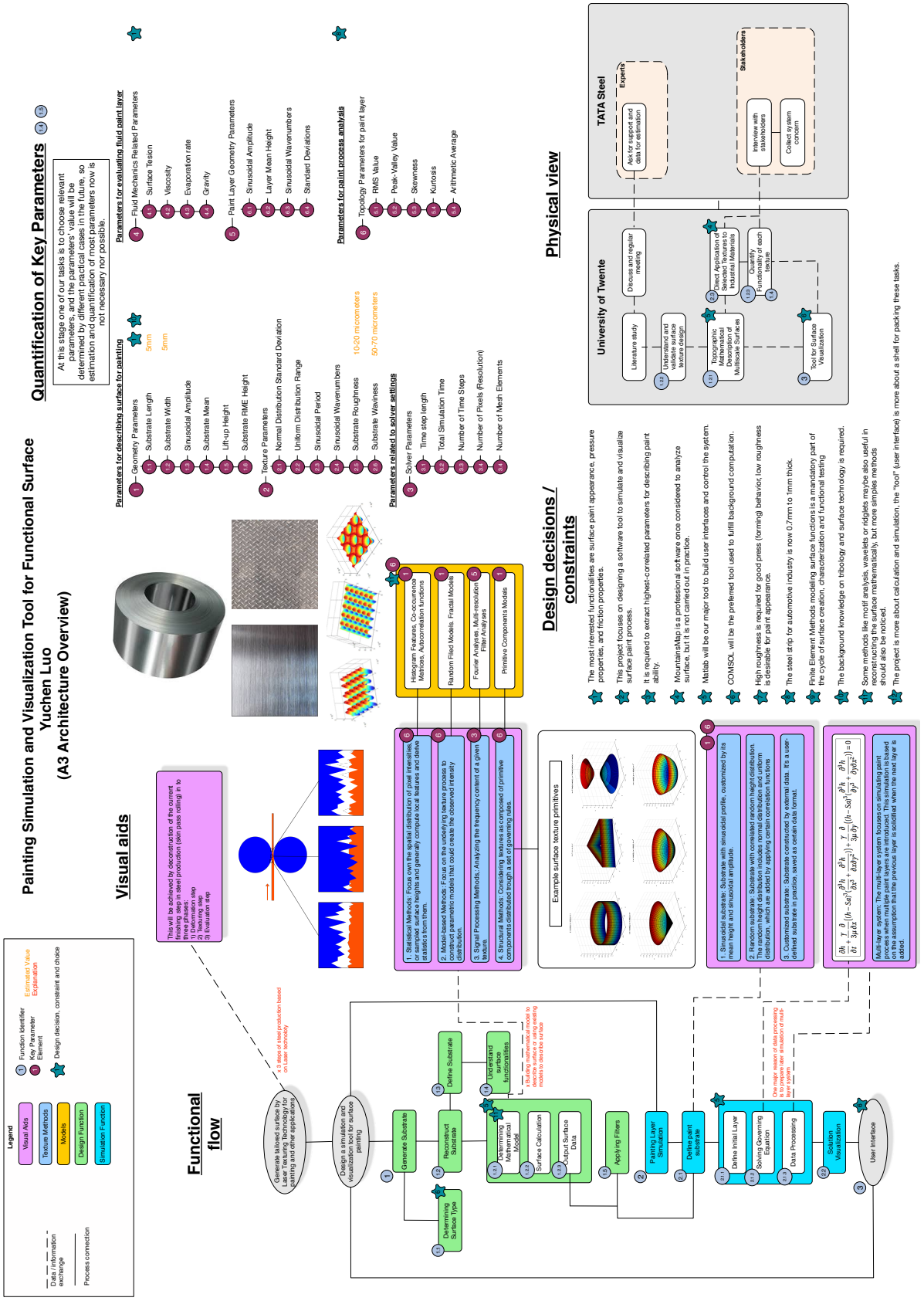


Figure 3.3: An A3AO diagram for early stage project design and management

3.3. Conclusion

In this chapter, we use system engineering theories and certain design methodologies to plan our design project. Contents in different design phases are specified and the advantages of utilizing Vee Process Models are also discussed. An A3AO diagram has been completed to give us top-design of my project at the early stage. Moreover, based on previous defined stakeholders requirements and physical reality the system requirements have been specified.

4

Model construction and platform selection

In this chapter, we start with an investigation on mathematical background used for describing paint process. Proper governing equations will be selected and corresponding physical models will be constructed based on this investigation. The properties of the governing equation and the complexity of the model will determine the software platform utilized for implementing paint process simulation.

4.1. Governing equation for paint process

The system requirement indicates that our system should be able to simulate a paint process on rough surfaces. A so-called paint process describes how a fluid layer distributes itself on a certain substrate after a certain time.[8][11] To be more precise, for observers, the initial layer, with a certain height (thickness) distribution on a rough substrate, will reshape itself due to the impact of surface tension, viscosity, gravity and other factors.[10] After a certain time, a new height distribution, which can be regarded as the most straightforward descriptor for this process, will be formed.

Therefore, mathematically, the system requirements are eventually translated into a problem: constructing a time-dependent model that can describe the fluid height distribution variation. Such a problem can be attributed to fluid mechanics realm, where Navier-Stokes equation is always regarded as an original point. The basic Navier-Stokes equations is shown in Eq.4.1:

$$\rho(\frac{\partial u}{\partial t} + u \cdot \nabla u) = \nabla p + \mu \nabla^2 u, \quad \nabla \cdot u = 0 \tag{4.1}$$

In Eq.4.1, we have:

- $u = (u_x, u_z)$ represents the velocity.
- μ is viscosity.
- p represents the pressure.
- ρ is the fluid density.
- t is represents the time.

For further discussion, we set the coordinates system: in 1D case, x coordinates represent the horizontal direction while z coordinates represent the vertical direction. In 2D case, the horizontal direction expands in to $x-y$ plane. The vertical direction is still represented by z coordinates.

Meanwhile, when considering the substrate with an inclination angle between horizontal surfaces, the new coordinate system will be set, the gravity acceleration g will be decomposed into new x,y and z direction. Therefore, we have the definition:

$$\begin{aligned} g_x &= g\sin\theta \\ g_y &= g\sin\theta \\ g_z &= g\cos\theta \end{aligned} \tag{4.2}$$

where θ is the inclination angle between substrate and horizontal surface, as shown in Figure.4.1.

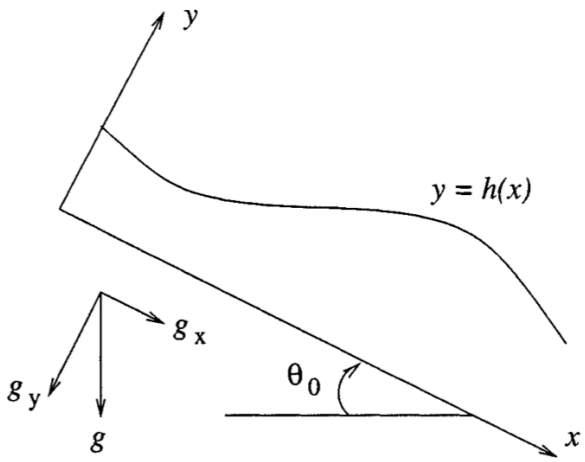


Figure 4.1: Coordinate system

4.1.1. Viscous fluid spreading under surface tension and gravity

Among several factors that can affect a paint process, surface tension and gravity are the most important two. Surface tension is certainly a key property of fluid while gravity, especially in vertical painting process, plays a very important role as well.[12]

To begin with, we rewrite the Navier-Stokes equation from Eq.4.1 into Eq.4.3 and Eq.4.4, which is a **steady state**, hence the term $\frac{\partial u}{\partial t}$ vanishes:

$$\rho(u_x\frac{\partial u_x}{\partial x}+u_z\frac{\partial u_x}{\partial z}) = -\frac{\partial p}{\partial x}+\mu(\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_x}{\partial z^2})+\rho g_x \tag{4.3}$$

$$\rho(u_x\frac{\partial u_z}{\partial x}+u_z\frac{\partial u_z}{\partial z}) = -\frac{\partial p}{\partial z}+\mu(\frac{\partial^2 u_z}{\partial x^2}+\frac{\partial^2 u_z}{\partial z^2})+\rho g_z \tag{4.4}$$

with continuity equation

$$\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}=0 \tag{4.5}$$

- The horizontal velocity u_x is much greater than the vertical velocity u_z and the perfect surface is horizontal, thus, terms with u_z can be neglected. Moreover, on the free surface the shear stress, which is approximately $\mu\frac{\partial u_x}{\partial z}$ vanishes.

- Since u_z is neglected, $\frac{\partial u_z}{\partial z} = 0$, together with the continuity equation given in Eq.4.5, we also obtain $\frac{\partial u_x}{\partial x} = 0$. Therefore, we made the assumptions that:

$$\begin{aligned} u_z &= 0, \\ \frac{\partial u_x}{\partial x} &= 0, \\ \frac{\partial u_z}{\partial z} &= 0 \end{aligned} \tag{4.6}$$

  or they are not comparable to the magnitude of other terms.

- The flow is driven by hydrostatic pressure, gravity, and resisted by viscous shear forces.[12][13] In another words, surface tension and gravity show their impact on fluid flow through pressure. [9] The pressure expression has the general form:

$$p = -\rho g_z h - \gamma\frac{\partial^2 h}{\partial x^2} \tag{4.7}$$

  The negative sign is used to express the pressure direction, and γ represents for surface tension. Then we have:

$$\frac{\partial p}{\partial x} = -\rho g_z\frac{\partial h}{\partial x} - \gamma\frac{\partial^3 h}{\partial x^3} \tag{4.8}$$

  In horizontal situation, the gravity driven pressure can be neglected. Therefore in horizontal situation we have:

$$\frac{\partial p}{\partial x} = -\gamma\frac{\partial^3 h}{\partial x^3} \tag{4.9}$$

Therefore, based on several assumptions [12][13], the whole left side of Eq.4.3 and Eq.4.4 can be neglected. So we reach:

$$0 = -\frac{\partial p}{\partial x} + \mu\frac{\partial^2 u_x}{\partial z^2} + \rho g_x \tag{4.10}$$

$$0 = -\frac{\partial p}{\partial z} + \rho g_z \tag{4.11}$$

The no-slip boundary condition requires that the velocity vanishes at the plane located at $z = 0$, namely the bottom surface, while the free-surface condition requires that the shear stress vanishes at the free-surface located at $z = h$, namely the film surface.[12][13] Therefore, the boundary condition for Eq.4.10 is given as:

$$\begin{aligned} u_x(z=0) &= 0 \\ \frac{\partial u_x}{\partial z}(z=h) &= 0 \end{aligned} \tag{4.12}$$

This boundary condition is applied for top and bottom of the thin layer, which is different to the periodic boundary conditions applied for left and right boundary of the thin layer.

Eq.4.11 shows that vertical direction pressure is only relevant to gravity. Eq.4.10 can be solved manually, and the solution is:

$$u(x,z,t) = \frac{1}{2\mu} z(2h(x,t) - z)(-\frac{\partial p}{\partial x} + \rho g_x) \tag{4.13}$$

In vertical situation, the flux is therefore

$$\begin{aligned} Q(x,t) &= \int_0^{h(x,t)} u(x,z,t)dz \\ &= -\frac{1}{3\mu}h^3\frac{\partial p}{\partial x} + \rho g_x \frac{1}{3\mu}h^3(x,t) \\ &= \frac{\rho g_z}{3\mu}h^3\frac{\partial h}{\partial x} + \frac{\gamma}{3\mu}h^3\frac{\partial^3 h}{\partial x^3} + \frac{\rho g_x}{3\mu}h^3(x,t) \end{aligned} \tag{4.14}$$

Meanwhile, since $\theta = \pi/2$, we have:

$$\begin{aligned} g_x &= g\sin\theta = g \\ g_z &= g\sin\theta = 0 \end{aligned} \tag{4.15}$$

hence:

$$Q(x,t) = \frac{\gamma}{3\mu}h^3\frac{\partial^3 h}{\partial x^3} + \frac{\rho g}{3\mu}h^3(x,t) \tag{4.16}$$

In horizontal situation, since the gravity driven pressure is neglected, the horizontal flux is therefore

$$\begin{aligned} Q(x,t) &= \int_0^{h(x,t)} u(x,z,t)dz \\ &= -\frac{1}{3\mu}h^3\frac{\partial p}{\partial x} + \rho g_x \frac{1}{3\mu}h^3(x,t) \\ &= \frac{\gamma}{3\mu}h^3\frac{\partial^3 h}{\partial x^3} + \frac{\rho g_x}{3\mu}h^3(x,t) \end{aligned} \tag{4.17}$$

Meanwhile, since $\theta = 0$, we also have:

$$\begin{aligned} g_x &= g\sin\theta = 0 \\ g_z &= g\sin\theta = g \end{aligned} \tag{4.18}$$

hence:

$$Q(x,t) = \frac{\gamma}{3\mu}h^3\frac{\partial^3 h}{\partial x^3} \tag{4.19}$$

The next step we consider **mass conservation** in the form

$$\frac{\partial h}{\partial t} + \frac{\partial Q}{\partial x} = 0 \tag{4.20}$$

and it gives us a nonlinear diffusion equation for $h(x,t)$:

Vertical case

$$\frac{\partial h}{\partial t} + \frac{1}{3\mu}\frac{\partial}{\partial x}\Big[h^3\big(\gamma\frac{\partial^3 h}{\partial x^3} + \rho g\big)\Big] = 0 \tag{4.21}$$

Horizontal case

$$\frac{\partial h}{\partial t}+\frac{\gamma}{3\mu}\frac{\partial}{\partial x}\Big(h^3\frac{\partial^3 h}{\partial x^3}\Big)=0 \tag{4.22}$$

With 1D result under gravity, it's easy to write down 2D situation. The equation is given as:

Vertical case

$$\boxed{\frac{\partial h}{\partial t}+\frac{1}{3\mu}\frac{\partial}{\partial x}\Big[h^3\big(\gamma(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})+\rho g\big)\Big]+\frac{\gamma}{3\mu}\frac{\partial}{\partial y}\Big(h^3(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})\Big)=0} \tag{4.23}$$

Horizontal case

$$\boxed{\frac{\partial h}{\partial t}+\frac{\gamma}{3\mu}\frac{\partial}{\partial x}\Big(h^3(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})\Big)+\frac{\gamma}{3\mu}\frac{\partial}{\partial y}\Big(h^3(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})\Big)=0} \tag{4.24}$$

These two equations can be governed by a general equation:

General case

$$\boxed{\frac{\partial h}{\partial t}+\frac{1}{3\mu}\frac{\partial}{\partial x}\Big[h^3\big(\gamma(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})+\rho g\sin\theta\big)\Big]+\frac{1}{3\mu}\frac{\partial}{\partial y}\Big[h^3\big(\gamma(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})+\rho g\cos\theta\big)\Big]=0} \tag{4.25}$$

Eq.4.23, Eq.4.24 and Eq.4.25 are models that describe the height distribution changing with time on a flat surface when surface tension and gravity are introduced. With such governing equations, several physical information can be extracted:

- The model is 2D case.
- The layer is painted on a perfect flat substrate.
- Gravity is considered, but in horizontal case, the gravity driven pressure can be neglected compared to surface tension driven pressure.
- Surface tension is considered as a constant.

4.1.2. Governing equation on rough substrate

Eq.4.25 gives us a general model to describe paint process, however, such a model is merely able to describe the paint process on a perfect flat substrate. In practice, surface roughness exists everywhere and we care more about how fluid layer behaves on a substrate with certain surface topology, which enables certain surface functionalities.

We start with 2D case. To introduce surface roughness into our model, we start the derivation from Eq.4.10:

$$\begin{aligned}0&=-\frac{\partial p}{\partial x}+\mu\frac{\partial^2 u_x}{\partial z^2}+\rho g_x\\0&=-\frac{\partial p}{\partial y}+\mu\frac{\partial^2 u_y}{\partial z^2}+\rho g_y\end{aligned} \tag{4.26}$$

The substrate surface with a certain surface topology is represented by S_a, the velocity can be obtained by integrating the equation with the previous mentioned boundary conditions, where the bottom surface "0" has been replaced by S_a [8]:

$$\begin{aligned} u_x(z=S_a) &= 0 \\ \frac{\partial u_x}{\partial z}(z=h) &= 0 \end{aligned} \tag{4.27}$$

we will obtain:

$$\begin{aligned} u_x(x,y,z,t) &= \frac{1}{2\mu}\Big(\frac{\partial p}{\partial x} - \rho g_x\Big)\big(z^2 - 2h(z-S_a) - S_a^2\big) \\ u_y(x,y,z,t) &= \frac{1}{2\mu}\Big(\frac{\partial p}{\partial y} - \rho g_y\Big)\big(z^2 - 2h(z-S_a) - S_a^2\big) \end{aligned} \tag{4.28}$$

Similarly, we can obtain the local flow components on the layer thickness along x and y directions. Based on the coordinates system, in vertical case, $g_x = g, g_y = 0$; in horizontal case, $g_x = 0, g_y = 0$.

$$\begin{aligned} Q_x &= \int_{S_a}^{h} u_x dz = \frac{\gamma}{3\mu}(h-S_a)^3\Big(\frac{\partial^3 h}{\partial x^3} + \frac{\partial^3 h}{\partial x \partial y^2}\Big) + \frac{\rho g_x}{3\mu}(h-Sa)^3 \\ Q_y &= \int_{S_a}^{h} u_y dz = \frac{\gamma}{3\mu}(h-S_a)^3\Big(\frac{\partial^3 h}{\partial y^3} + \frac{\partial^3 h}{\partial y \partial x^2}\Big) + \frac{\rho g_y}{3\mu}(h-Sa)^3 \end{aligned} \tag{4.29}$$

Applying the mass conservation equation, the complete model equation is:
Vertical case

$$\boxed{\frac{\partial h}{\partial t} + \frac{1}{3\mu}\frac{\partial}{\partial x}\Big[(h-Sa)^3\Big(\gamma\Big(\frac{\partial^3 h}{\partial x^3} + \frac{\partial^3 h}{\partial x \partial y^2}\Big) + \rho g\Big)\Big] + \frac{\gamma}{3\mu}\frac{\partial}{\partial y}\Big((h-Sa)^3\Big(\frac{\partial^3 h}{\partial y^3} + \frac{\partial^3 h}{\partial y \partial x^2}\Big)\Big) = 0} \tag{4.30}$$

Horizontal case

$$\boxed{\frac{\partial h}{\partial t} + \frac{\gamma}{3\mu}\frac{\partial}{\partial x}\Big((h-Sa)^3\Big(\frac{\partial^3 h}{\partial x^3} + \frac{\partial^3 h}{\partial x \partial y^2}\Big)\Big) + \frac{\gamma}{3\mu}\frac{\partial}{\partial y}\Big((h-Sa)^3\Big(\frac{\partial^3 h}{\partial y^3} + \frac{\partial^3 h}{\partial y \partial x^2}\Big)\Big) = 0} \tag{4.31}$$

Eq.4.30 and Eq.4.31 can be combined and the general equation is:
General case

$$\begin{aligned} &\boxed{\frac{\partial h}{\partial t} + \frac{1}{3\mu}\frac{\partial}{\partial x}\Big[(h-Sa)^3\Big(\gamma\Big(\frac{\partial^3 h}{\partial x^3} + \frac{\partial^3 h}{\partial x \partial y^2}\Big) + \rho g \sin\theta\Big)\Big] + \ldots} \\ &\boxed{\frac{1}{3\mu}\frac{\partial}{\partial y}\Big[(h-Sa)^3\Big(\gamma\Big(\frac{\partial^3 h}{\partial y^3} + \frac{\partial^3 h}{\partial y \partial x^2}\Big) + \rho g \cos\theta\Big)\Big] = 0} \end{aligned} \tag{4.32}$$

Eq.4.30 and Eq.4.31 are the governing equations we are going to study further, they contain the following physical realities:

- The model is 2D case.
- Rough surface with surface topology S_a is introduced.
- Gravity is considered, but in horizontal case, the gravity driven pressure can be neglected compared to surface tension driven pressure.
- Surface tension is considered as a constant.

4.1.3. Evaporation

Evaporation is another factor that affects the paint process. It describes how does paint dry in a certain time period. In practice, the evaporation rate is considered as a constant number and it is linear combined with the surface tension and gravity driven paint process.[11] Therefore, with an evaporation term, Eq.4.30, Eq.4.31 and Eq.4.32 can be rewritten as:

Vertical case

$$\boxed{\frac{\partial h}{\partial t}+\frac{1}{3\mu}\frac{\partial}{\partial x}\Big[(h-Sa)^3\big(\gamma(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})+\rho g\big)\Big]+\frac{\gamma}{3\mu}\frac{\partial}{\partial y}\big((h-Sa)^3(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})\big)+E=0} \tag{4.33}$$

Horizontal case

$$\boxed{\frac{\partial h}{\partial t}+\frac{\gamma}{3\mu}\frac{\partial}{\partial x}\big((h-Sa)^3(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})\big)+\frac{\gamma}{3\mu}\frac{\partial}{\partial y}\big((h-Sa)^3(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})\big)+E=0} \tag{4.34}$$

General case

$$\begin{gathered}\boxed{\frac{\partial h}{\partial t}+\frac{1}{3\mu}\frac{\partial}{\partial x}\Big[(h-Sa)^3\big(\gamma(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})+\rho g\sin\theta\big)\Big]+\ldots}\\ \boxed{\frac{1}{3\mu}\frac{\partial}{\partial y}\Big[(h-Sa)^3\big(\gamma(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})+\rho g\cos\theta\big)\Big]+E=0}\end{gathered} \tag{4.35}$$

4.2. Platform selection

In Section.4.1, paint process models with their governing equations are put forward.

When flat substrates are introduced, a linearization process can be realized for governing equation so that a linear high-order partial differential equation will be formulated[8], as shown in Eq.A.1

$$\boxed{\frac{\partial\delta h(x,y,t)}{\partial t}=-\frac{\gamma}{3\mu}e_0^3\big(\frac{\partial^4\delta h(x,y,t)}{\partial x^4}+2\frac{\partial^4\delta h(x,y,t)}{\partial x^2\partial y^2}+\frac{\partial^4\delta h(x,y,t)}{\partial y^4}\big)} \tag{4.36}$$

When rough substrates are introduced, a non-linear high-order partial differential equation will be formulated. The expression for vertical situation and horizontal situation are different:

Vertical case

$$\boxed{\frac{\partial h}{\partial t}+\frac{1}{3\mu}\frac{\partial}{\partial x}\Big[(h-Sa)^3\big(\gamma(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})+\rho g\big)\Big]+\frac{\gamma}{3\mu}\frac{\partial}{\partial y}\big((h-Sa)^3(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})\big)=0} \tag{4.37}$$

Horizontal case

$$\boxed{\frac{\partial h}{\partial t}+\frac{\gamma}{3\mu}\frac{\partial}{\partial x}\big((h-Sa)^3(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})\big)+\frac{\gamma}{3\mu}\frac{\partial}{\partial y}\big((h-Sa)^3(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})\big)=0} \tag{4.38}$$

As discussed in Chap.3, MATLAB will be our first choice to implement the integrated software tool. When involving linearized PDEs, which is discussed in detail in Appendix.A,

a numerical solution scheme based on Fourier Transform can be built efficiently. However, when involving non-linear high-order PDEs (Eq.4.30 and Eq.4.31), it can be foreseen that the process of designing an accurate, efficient and comprehensive numerical scheme through MATLAB will be difficult and time-consuming. Therefore, **COMSOL Multiphysics**, which is capable of solving complicated PDEs, is chosen.

COMSOL Multiphysics is a finite element analysis, solver and simulation software/FEA software package for various physics and engineering applications, especially coupled phenomena, or multi-physics. In addition to conventional physics-based user interfaces, COMSOL Multiphysics also allows entering coupled systems of partial differential equations (PDEs). It's straightforward and efficient to utilize COMSOL to solve the non-linear, high-order PDEs mentioned before.

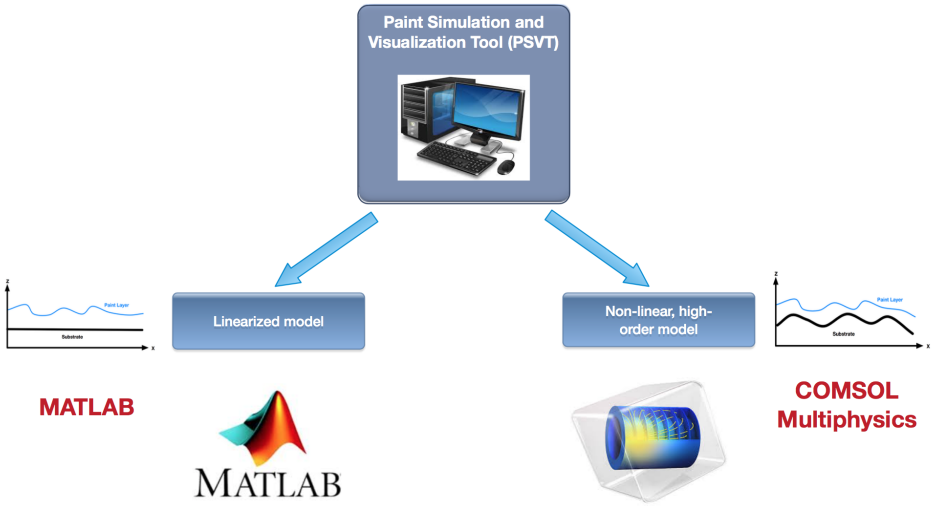


Figure 4.2: Platform selection for different mathematical models

COMSOL also provides another software, or known as a user accessible library, **COMSOL Livelink**, for users to realize interaction between COMSOL and other scientific softwares. COMSOL Livelink makes the data transfer between COMSOL and MATLAB possible, hence a software tool relies on the COMSOL-MATLAB cooperative work will be our preference.

Therefore, the following important **decisions** on platform selection for system implementation are made:

1. **The software tool user interface will be implemented through MATLAB. User will only get the access to the MATLAB-built user interface, which means all the I/O functions will be implemented through MATLAB.**

2. **A numerical scheme (FFT) for solving the linearized model, which describes paint layer distributes on a flat substrate, or on a tiny roughness substrate, will be built through MATLAB. However, this model will be only used for early stage model verification. It will not be employed in our software tool.**

3. **For solving the non-linear high-order model, which describes paint layer distributes on a rough substrate horizontally or vertically, COMSOL will be employed as a background solver.**

The aforementioned decisions will directly affect the system functions determination and system architecture construction.

4.3. Conclusion

In this chapter, the mathematical models of paint process on rough substrates are constructed. A linearized model has been built for describing paint layer distribution on a flat substrate. The linearized model, which can be solved by FFT scheme, has its certain meaning when facing a macro-scale problems and can be used as a reference for smaller-scale problems or an early stage verification source. Meanwhile, a non-linear, high-order model has been built for describing paint layer distribution on a rough substrate. The non-linear model accords with the realistic situation hence it will be our focus.

Based on the constructed models and the corresponding governing equations, decisions on platform selection have been made. The system then can be defined as a MATLAB-based, multi-platform involved software tool.

5

Function analysis and system architecture

In Chap.4, a mathematical model describing linear and non-linear paint process has been built. This mathematical model provides us an important reference to foresee possible difficulties and select the most proper software platform for implementing various background calculations and simulations, which is also a precondition and key consideration for constructing a system architecture.

In this chapter we are going to discuss the construction of the system architecture and relevant system functions. The various system functions are determined based on certain requirements, and a comprehensive system architecture will support the system to fulfill all of the determined functions.

5.1. Function analysis

As mentioned, the system functions can be defined based on system requirements. In some sense, system functions are direct translation of system requirements. The system requirements defined in Chap.3 are shown as follows:

1. **R1: The system should be able to generate various type of surfaces.**
2. **R2: The system should be able to read measured surfaces data.**
3. **R3: The system should be able to visualize generated surfaces.**
4. **R4: The system should be able to fulfill relevant data process, including data conversion and interaction between various software platform.**
5. **R5: The system should be able to analyze relevant statistics of paint process and fluid layer.**
6. **R6: The system should be able to simulate the paint process.**
7. **R7: The system should be able to visualize paint process.**

Taking R1 system requirement as an example, **the system should be able to generate various type of surfaces** is a requirement for our software system, meanwhile, **generating various type of surfaces** is also the system functions we are going to implement. In this sense,

one function is an expression of one certain requirement in a solution domain. Furthermore, the system functions can be as the supplement or explanation for the system requirements. For example, R1 requirement, **The system should be able to generate various type of surfaces**, can be more specific. Therefore, this requirement can be decomposed into:

- The system should be able to generate surface with regular surface textures.
- The system should be able to generate surface with random surface textures.

which means two new specific requirements are put forward, as well as the definitions of two system functions. The idea based on requirements decomposition can help us easily define system functions, at the same time, the system can be decomposed into various sub-systems. This decomposition provides us a possibility to categorize the system functions into groups based on their corresponding sub-systems, which is also the original point for us to build an overall system architecture.

Before constructing a complete system architecture, a general decomposition of the system can be implemented based on the information so far. As shown in Figure.5.1, the

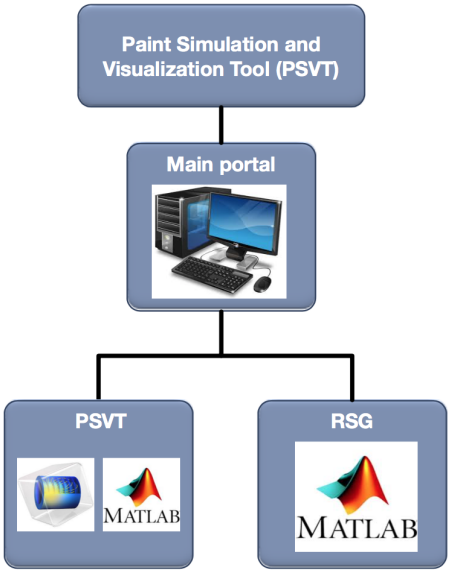


Figure 5.1: The preliminary decomposition

system will generally be considered as three major sub-systems, they are **PSVT**, **RSG**, and **Main portal**. According to the decisions made in Chap.4, the paint process will be simulated and calculated by COMSOL from background, but executed by MATLAB. The different software platform selections for background calculation also makes this decomposition reasonable. A rough surface generator will be implemented to provide substrates for paint process, which is consider as another independent module, while the main portal provides users an access to all these modules.

Figure.5.2 shows a table containing system functions categorized by considering above mentioned sub-systems.

To be more precise, some explanations for each system components from Figure.5.2 are shown as follows:

System Components	Sub-components	System Functions
Main portal	/	F1: Providing user access to the software tool
RSG	User data input and output module	F2: Data input and output
	Data check and error reporting module	F3: Generating substrate with regular primitives
	Random substrate generator	F4: Generating substrate with random topology
	Regular substrate generator	F5: Applying certain filters on generated substrate
	Statistic analysis module	F6: Display 2D substrate topology and 3D structure
	Visualization panel	F7: Calculating and showing statistics and key descriptors
		F8: Data conversion between Comsol and Matlab identified form
PSVT	User data input and output module	F9: Data input and output
	Setting module	F10: Realizing simulation with non-linear model
	Vertical model simulation branch	F11: Realizing the horizontal paint process simulation (rough substrat
	Horizontal model simulation branch	F12: Realizing the vertical paint process simulation (rough substrate)
	Data check and error reporting module	F13: Implementation of single-layer paint system
	Single layer system simulation module	F14: Implementation of multi-layer paint system
	Multi-layer system simulation module	F15: Displaying 2D paint layer height distribution (topology)
	Visualization panel	F16: Displaying 3D dynamic paint process
	Comsol & Matlab interaction module	F17: Calculating and showing process statistics and key descriptors
	Statistic analysis module	F18: Implementation of Comsol and Matlab background interaction
		F19: Data conversion between Comsol and Matlab identified form

Figure 5.2: System functions

1. **Main portal**

   - The whole user-interface is implemented with MATLAB.
   - The main portal, which can also be regarded as a main menu, provides user access to the software tool. Apparently, the main portal also enables user to enter the corresponding modules that can realize users' desired functions.

2. **Rough Substrate Generator (RSG)**

   - Two major sub-components of the RSG are known as regular substrate generator and random substrate generator. They correspond to the function **F3** and **F4**, respectively.
   - The function **F3** is realized by employing **Structural methods**[19] to construct a surface. This method, which will be discussed in Chap.7 in detail, considers surface textures as composed of primitives components distributed through a set of governing rules. The primitive usually has its certain geometry. In fact, a substrate with certain repeatable primitives is usually utilized to realize surface functionalities other than painting. The consideration for this function ensures the comprehensiveness of the software tool and provides possibilities for future upgrade.
   - A substrate with random topology are constructed based on data generated by random number with certain distributions, eg. uniform distribution, normal distribution. These data cannot be used to describe a surface in reality directly due to their discontinuity and singularity. Therefore, certain interpolation rule and filters have to be employed to calibrate the data. That's why function **F5** does exist.
   - The RSG will provide surface data as our substrate for paint process simulation, hence the data input and output function is indispensable. Meanwhile, since we employ both COMSOL and MATLAB to complete the background simulation and calculation, the data generated by RST should be accessible for both of them, especially COMSOL. Function **F8** will fulfill this data conversion task and make interaction between MATLAB built RSG and COMSOL possible.

3. **Non-linear PSVT**

   - A non-linear model simulates the paint process when rough substrates are introduced. As shown in Chap.4, the governing equation is a 4th order non-linear partial differential equation. COMSOL becomes our choice for solving the problem.
   - Non-linear PSVT is able to implement both vertical paint process simulation and horizontal paint process simulation. As discussed in Chap.3, vertical paint process is one of our focus since most of the industrial surface paint are realized with inclinable or vertical settings instead of flat settings.
   - The non-linear mathematical model is constructed and solved by COMSOL from background while the data analysis, visualization functions and user access are all realized by MATLAB. Therefore, this sub-system requires the most capacity

of interaction and data process function **F25** between MATLAB and COMSOL, COMSOL Livelink will play a very important role in the interaction implementation.

Figure.5.3 shows a **Function view** , which can provide insights in the functional elements, graphical interfaces, and primary interactions. Only key-features are shown here to keep the figure as simple as possible.

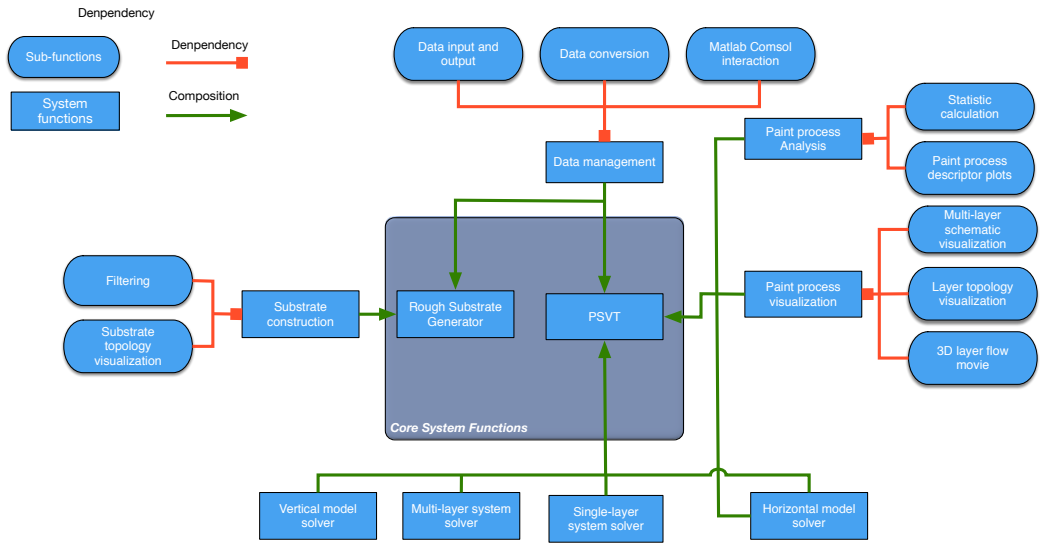


Figure 5.3: Function view of the software tool

5.2. System architecture

A system architecture will be constructed after definition of requirements and functions. The system architecture constructed in this section will focus more on the tool top level design instead of infrastructure. In the following chapters regarding the sub-systems, system architectures focusing more on infrastructures will also be constructed.

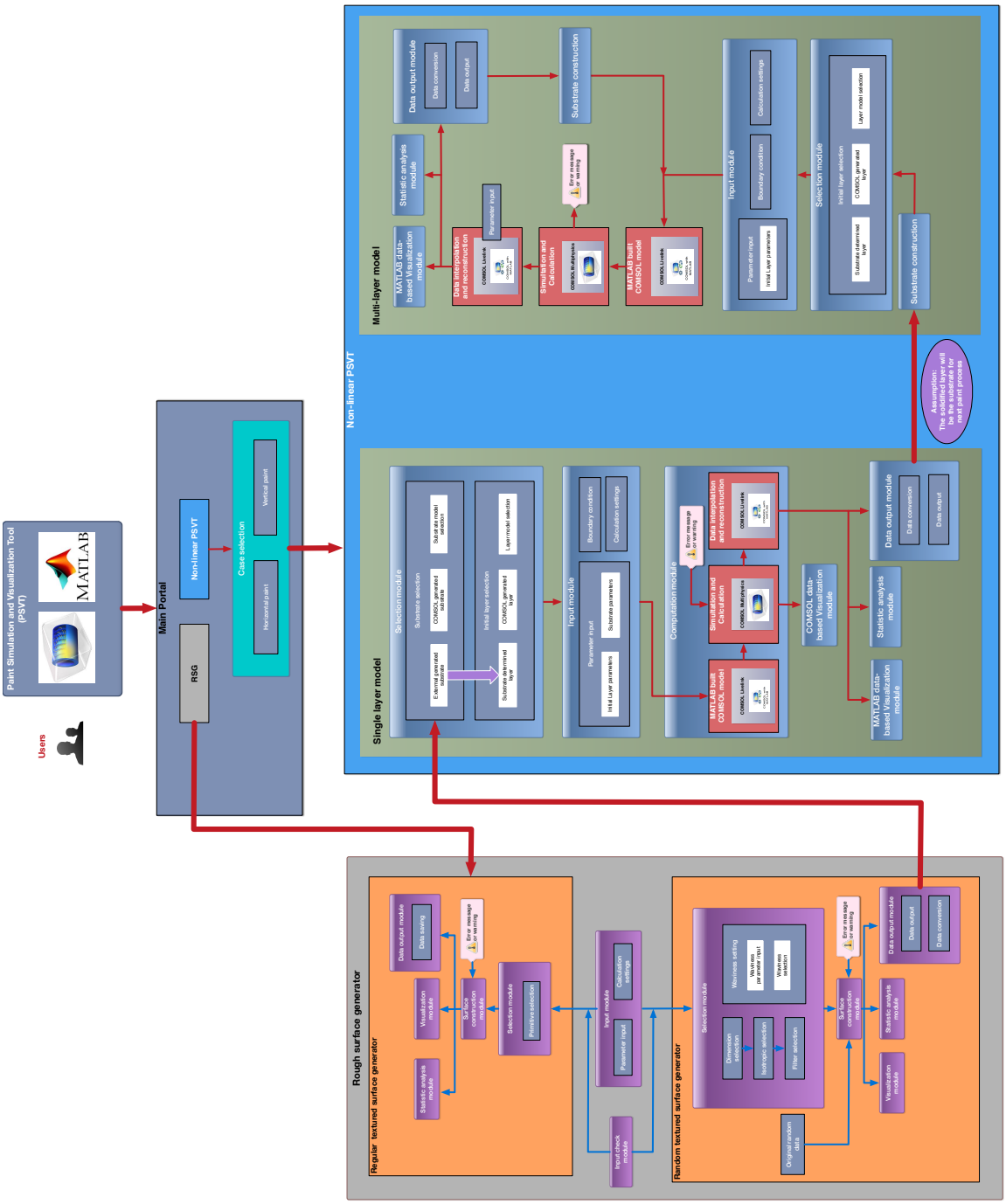


Figure 5.4: System architecture for Paint Simulation and Visualization Tool

6

COMSOL implementation

As discussed in Chap.4, COMSOL will be our choice to realize most of the background computation and simulation process. A COMSOL model can be built through either COMSOL GUI or MATLAB code. The first alternative will be the focus discussed in this chapter, while the second alternative, will be discussed specifically in later chapters. The COMSOL GUI built model will provide us obligatory reference and preliminary data for further investigation and implementation.

6.1. Model initialization

An example COMSOL model will be created for simulating horizontal case paint process mentioned in Chap.5, which has the governing equation Eq.4.31. The pre-settings for generating a COMSOL model is listing as follows:

- Space Dimension: A 2D space dimension is selected.
- Physics Interfaces: A **General Form PDE (g)** is selected for solving the governing equations.
- Study: A **Time Dependent Study** is selected since the governing equations are time-dependent PDEs

6.2. Model implementation

6.2.1. Parameters settings

Part of the required parameters can be found from Eq.4.30 and Eq.4.31, while others are determined by the form of substrates and the setting of initial conditions. Some of the key **input parameters** extracted from the model are explained here:
Input parameters

- μ: $(Pa \cdot s)$, Viscosity of the fluid
- γ: (N/m), Surface tension of the fluid
- ρ: (Kg/m^3), Density of the fluid

- g: (m/s^2), Gravity acceleration
- L: (mm), Sinusoidal waviness
- L_{sub}: (mm), Size of the substrate
- H_{sub}: (μm), Substrate mean height, the average height of the substrate above zero reference plane
- A_{sub}: (μm), Substrate amplitude in sinusoidal case, range or deviation of substrate roughness height in random case
- H_{layer}: (μm), Initial paint layer mean height, the average height of the paint layer above zero reference plane
- A_{layer}: (μm), Initial paint layer amplitude in sinusoidal case, range or deviation of paint layer roughness height in random case
- t: (s), Total time for simulation.
- t_{step}: (s), Time step length.

It can be seen H_{layer} and A_{layer} are parameters defined for setting initial conditions. With these parameters the initial shape of the paint layer can be described. More parameters may be introduced when various initial conditions or physical reality are applied[15].

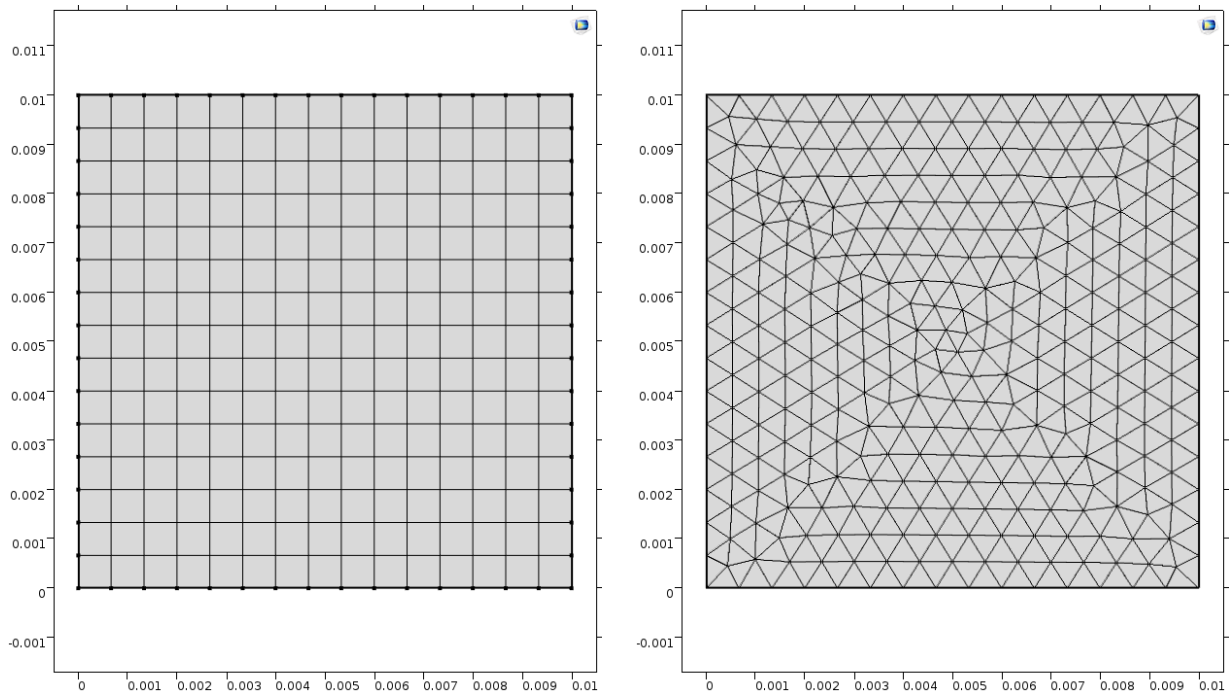
The **output parameter** of the simulation will be new height distribution of the paint layer, which is also in μm scale.

6.2.2. Geometry and mesh settings

Typically, the governing equations' corresponding physical entities or models are imported into COMSOL as a Geometry. In our case, instead of implementing the substrate texture in Geometry, we simply set the Geometry as a square or rectangle, which describes no more than the substrate size. To introduce the substrate texture, or the real "geometry", we connect texture mathematical expression with some defined variables appearing in the governing equations.

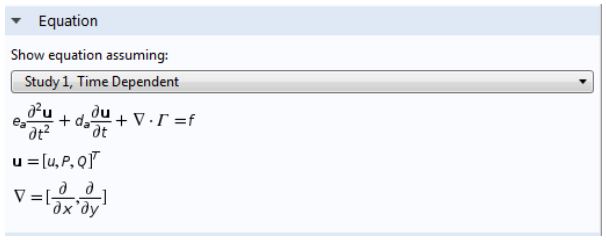
Another important factor is mesh settings on the defined geometry. COMSOL provides several kinds of mesh settings. Among all the options, **Mapped** mesh and **Free Triangular** mesh are going to be investigaed. Figure.6.1 shows these two types of mesh settings on a square geometry. **Free Triangular** is the default mesh setting of COMSOL geometry. However, it's difficult to quantify elements and nodes numbers in a triangular mesh. Moreover, the solution obtained with triangular mesh is asymmetric hence cannot be saved into an array, which makes further data process, especially data process in MATLAB, very difficult.

On the contrary, **Mapped** mesh makes it easy to set relevant mesh parameters and the solution data obtained will be symmetric, which can be export as matrices. Therefore, **Mapped** mesh setting becomes the preference in our case.

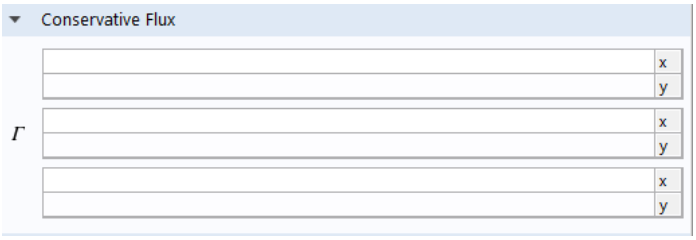


(a) (b)

Figure 6.1: Different mesh settings in COMSOL



(a) COMSOL-defined General Form PDE



(b) The empty Γ matrix

6.2.3. Governing equation

A General Form PDE in COMSOL definition is shown in Figure.6.2a Taking horizontal case governing equation as an example:

Horizontal case

$$\boxed{\frac{\partial h}{\partial t}+\frac{\gamma}{3\mu}\frac{\partial}{\partial x}\Big((h-Sa)^3(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})\Big)+\frac{\gamma}{3\mu}\frac{\partial}{\partial y}\Big((h-Sa)^3(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})\Big)=0} \tag{6.1}$$

The first step is to transfer our governing equations into COMSOL-defined General Form PDE. It can be realized that the mass coefficient e_a will be 0 for Eq.4.31. Meanwhile, Γ will be defined as 3 groups of 2×1 vector, as shown in Figure.6.2b By introducing two more dependent variables, P and Q, known as:

$$\nabla\Big(\frac{\partial h}{\partial x}\Big)=\frac{\partial^2 h}{\partial x^2}=P \tag{6.2}$$

$$\nabla\Big(\frac{\partial h}{\partial y}\Big)=\frac{\partial^2 h}{\partial y^2}=Q \tag{6.3}$$

the source term, can be written as:

$$f=\begin{bmatrix}0\\P\\Q\end{bmatrix} \tag{6.4}$$

By rewriting the governing equation, the empty Γ array will be filled in with:

$$\begin{bmatrix}\frac{\gamma}{3\mu}(h-Sa)^3(\frac{\partial^3 h}{\partial x^3}+\frac{\partial^3 h}{\partial x\partial y^2})\\ \frac{\gamma}{3\mu}(h-Sa)^3(\frac{\partial^3 h}{\partial y^3}+\frac{\partial^3 h}{\partial y\partial x^2})\end{bmatrix}\quad\begin{bmatrix}\frac{\partial h}{\partial x}\\0\end{bmatrix}\quad\begin{bmatrix}0\\\frac{\partial h}{\partial y}\end{bmatrix} \tag{6.5}$$

Eventually, the governing equation will be rewritten into a equation group including 3 equations:

$$\frac{\partial h}{\partial t}+\Big[\frac{\partial}{\partial x},\frac{\partial}{\partial y}\Big]\cdot\begin{bmatrix}\frac{\gamma}{3\mu}(h-Sa)^3(\frac{\partial P}{\partial x}+\frac{\partial Q}{\partial x})\\ \frac{\gamma}{3\mu}(h-Sa)^3(\frac{\partial P}{\partial y}+\frac{\partial Q}{\partial y})\end{bmatrix}=0 \tag{6.6}$$

$$\Big[\frac{\partial}{\partial x},\frac{\partial}{\partial y}\Big]\cdot\begin{bmatrix}\frac{\partial h}{\partial x}\\0\end{bmatrix}=P \tag{6.7}$$

$$\Big[\frac{\partial}{\partial x},\frac{\partial}{\partial y}\Big]\cdot\begin{bmatrix}0\\\frac{\partial h}{\partial x}\end{bmatrix}=Q \tag{6.8}$$

With above-mentioned settings, the governing equations can be input as General Form PDE with certain COMSOL grammar.

The PDE corresponding **initial conditions** and **boundary conditions** will be also determined:

Initial condition:

The initial condition describes the initial layer height distribution (the layer shape) for our simulation. Four initial condition options have been put forward:

- **Layer with sinusoidal profile:** A non-isotropic sinusoidal initial layer has the form:

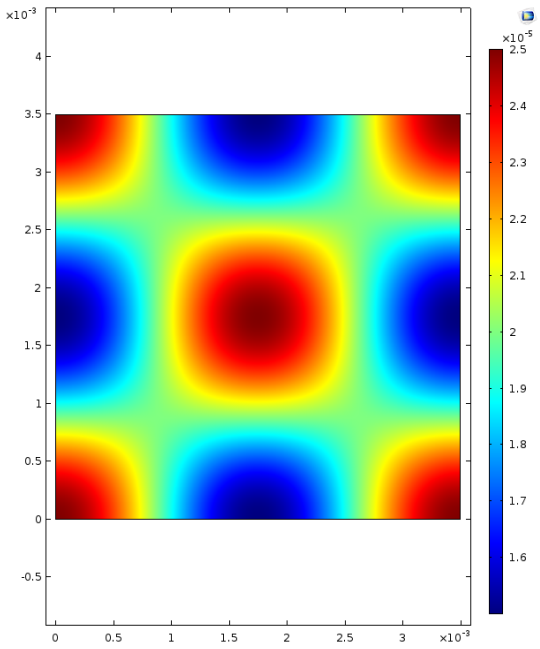
$$H_{initial} = H_{layer} + A_{layer}\cos(2\pi x/L_{layer})\cos(2\pi y/L_{layer}) \tag{6.9}$$

H_{layer} and A_{layer} are defined in parameter settings while L_{layer} is the waviness of the layer profile. An example is shown in Figure.6.2a. In practice, a sinusoidal profile is closest to the physical reality and it can be used to describe the shape of a smooth fluid layer by setting proper waviness and amplitude. In another word, sinusoidal profile can describe a approximately 'flat' initial layer if necessary. However, an absolute flat initial fluid layer does not make any sense in the simulation, since when the layer is flat the free energy is minimized and the layer already reaches a stable state. Therefore, no dynamic fluid flow will be observed no matter what substrates are introduced.

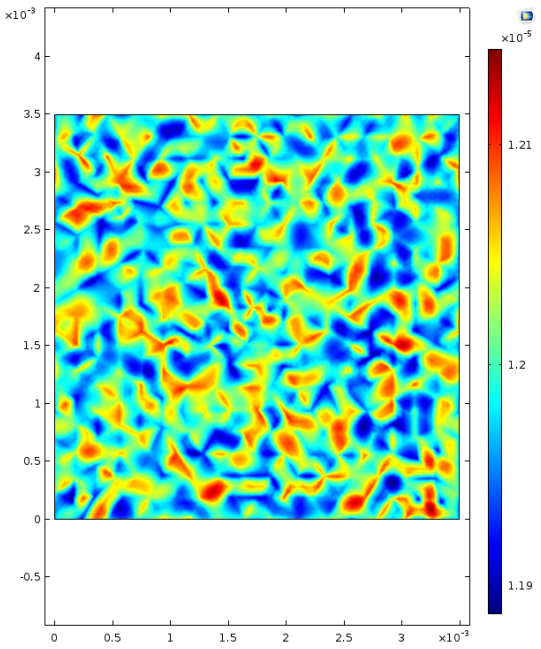
- **Layer with random profile:** A random initial layer is generated by utilizing COMSOL built-in rn function. The rn function will be also used for substrates generation internally. Such a initial layer has the form:

$$H_{initial} = H_{layer} + A_{layer}\,rn(x,y) \tag{6.10}$$

An example is shown in Figure.6.2b. One can image that it is hardly possible to find such a fluid layer in reality, since the fluid in this phase is so unstable that it will immediately flat itself in a very short time period, unless an extremely high viscosity is introduced. However, although such an initial condition is far from the reality, it's meaningful and necessary to take it into account to check the reliability of the simulation and ensure the comprehensiveness of the model.



(a) A sinusoidal initial layer with $20\mu m$ mean height, $5\mu m$ amplitude, $3.5mm$ waviness



(b) A random initial layer with $10\mu m$ mean height, $5\mu m$ range

- **Layer with a same profile to substrate:** In nano-scale fluid-substrate contact problem [17] the initial paint layer sometimes is defined as the same profile to the substrate roughness, but lifted with a certain height[18], as shown in Figure.6.2. Therefore, a new parameter lift hight H_{up} has to be introduced. Usually, such a H_{up} will be 10^{-3} small compared to the substrate length scale. In our case, this initial condition is also widely used since the paint layer height scale (mm) and the substrate length scale (μm) also show a 10^{-3} ratio.

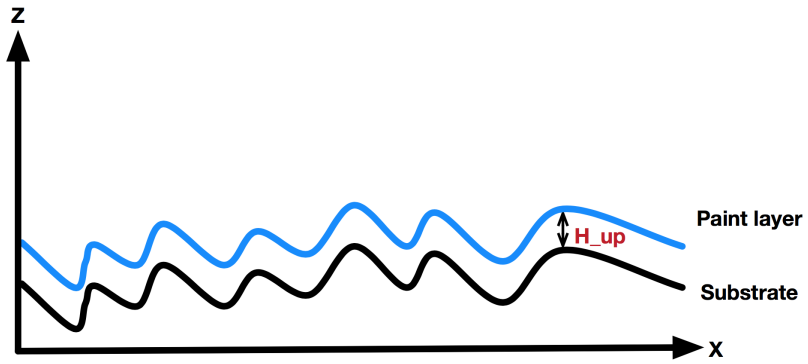


Figure 6.2: One particular initial condition

- **Measured layer data:** In practice, a real fluid layer can be measured with certain experimental set-ups. The thickness data of the layer can be sent into COMSOL as our initial condition.

Boundary condition:

Based on the assumption that our substrate is a selected part from one big steel surface, periodic boundary condition will be our choice. As shown in Figure.8.2, in horizontal case, periodic boundary condition will be applied to two couples of of boundaries, namely x-direction and y direction. In vertical case, periodic boundary condition will be applied to x-direction, but optional for z-direction (vertical direction). In vertical case, applying periodic boundary condition in z-direction means the substrate is selected from central part of the steel surface, while non-applying means the substrate bottom boundary is the whole surface boundary. This difference will certainly result in different solution. Furthermore, periodic boundary conditions will apply to all the three dependent variables defined in governing equations, namely h, P and Q.

6.2.4. Substrate implementation

In order to introduce rough substrates into the PDEs, a variable Sa representing the rough substrates has been defined. It appears in all the governing equations. This variable Sa will be connected to various substrate mathematical expression to introduce different substrates textures.

Internal built substrate:

Internal built substrates means substrates built within COMSOL. It is possible to build a substrate internally by direct input some COMSOL built-in functions. In this situation, two types of substrates are our interests: sinusoidal substrate and random data substrate, their mathematical expressions are similar to those mentioned in initial condition.

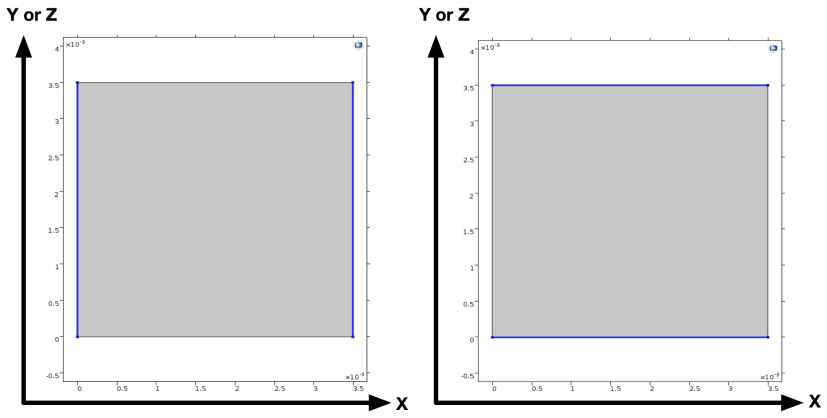


Figure 6.3: Periodic boundary conditions

- **Sinusoidal substrate** Since sin and cos functions are available in most of the scientific softwares including COMSOL, sinusoidal substrate can be generated by simply changing the expression of variable Sa. A sinusoidal substrate can be either isotropic or non-isotropic, which has the form:

$$Sa = H_{sub} + A_{sub}\cos(2\pi x/L) \qquad isotropic \tag{6.11}$$

$$Sa = H_{sub} + A_{sub}\cos(2\pi x/L)\cos(2\pi y/L) \qquad non-isotropic \tag{6.12}$$

  H is the substrate mean height, A is sinusoidal amplitude and L is sinusoidal waviness, which are all defined in parameter list. Figure.sincomsol shows two substrates, one with isotropic sinusoidal texture and another non-isotropic texture.

- **Random data substrate** By employing COMSOL built in random function rn, a random data substrate can be built. rn function can generate an array of norm distribution or uniform distribution numbers. The relevant deviation or mean height of these numbers can also be set within rn function property. By setting a proper deviation or mean height, such an array of data can be used to represent a rough substrate.

  With a well-defined rn function, the substrate variable Sa can be defined as:

$$Sa = H_{sub} + A_{sub}rn(x,y) \tag{6.13}$$

  Figure.6.4 shows a random data substrate with mean height $25\mu m$ and deviation $5\mu m$ It can be seen that the substrate data is quite discontinues and singular, which can hardly be a realistic substrate. However, it's sufficient to use this substrate to validate the mathematical model and observe the solution.

  In practice, filters or interpolations will be applied on random distribution data in order to create more realistic substrates. Such process will be implemented in RSG and PSVT. The substrates generated by RSG will be regarded as external built substrates and be imported into a COMSOL project.

External built substrate:

Substrates generated externally or measure surfaces are actually our major concern. Since the software tool employs COMSOL as background solver in most of situations, users

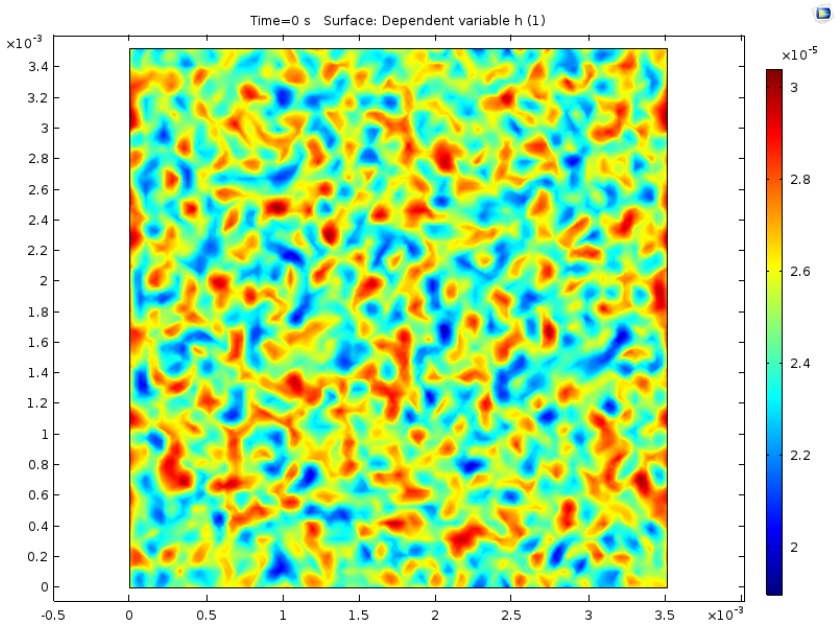


Figure 6.4: COMSOL generated random data substrate

are required to build their customized substrates through MATLAB user interface. Therefore, how to import external substrates data into a COMSOL model is going to be discussed. The problems about how to implement this import process through MATLAB code and the further interaction between COMSOL and MATLAB data will be discussed in RSG and PSVT implementation chapters.

Typically, there are three options to import external data: **Geometry**, **Material** and **Function**[16].

- **Geometry**: It is possible to import CAD file and COMSOL file as the COMSOL geometry. However, the substrate construction in our background COMSOL project is realized by setting variables instead of creating a COMSOL geometry directly. Specifically, the COMSOl geometry in our project is always a square or rectangle region which determines the basic dimensions of our substrate, while substrate textures are introduced by setting a variable with certain mathematical expression (function). Furthermore, it's difficult to realize the interaction between COMSOL geometry option and MATLAB data. COMSOL geometry does not provide options to introduce geometry data described by a $.mat$ or a $.txt$ file. With the aforementioned reasons, geometry is not our choice for importing external substrate data.

- **Material**: In a COMSOL model material is usually used to import isolated parameters of certain physical entities. It cannot fulfill the requirement that describing certain surface textures with some mathematical expressions or functions. Therefore, it is not our choice for importing external substrate data.

- **Function**: On one hand, COMSOL function option is available to generate built-in functions like random functions, waveform function, etc., on the other hand, it is also possible to import external data in the form of a certain function. Fox example, it's possible to build an interpolation function with external data, or even directly

call a MATLAB function to import data. Therefore, we will utilize function option in COMSOL to import our substrate.

Among all the built-in functions provided under **Function** option, we select interpolation function to import the substrates.

To realize this, the surface data must be saved as a file with certain COMSOL-identified formats, including "grid", "spreadsheet", and "section-wise". "Grid" format will be our choice since generating a text file in COMSOL-identified "grid" format from MATLAB matrix data is the easiest route. By applying "Grid" format, all the roughness data will be sent to COMSOL with no data lost[16] These data will be optimized by employing COMSOL internal interpolation algorithms during calculation and visualization. Figure.6.5 shows an example import substrate plot in COMSOL interface.

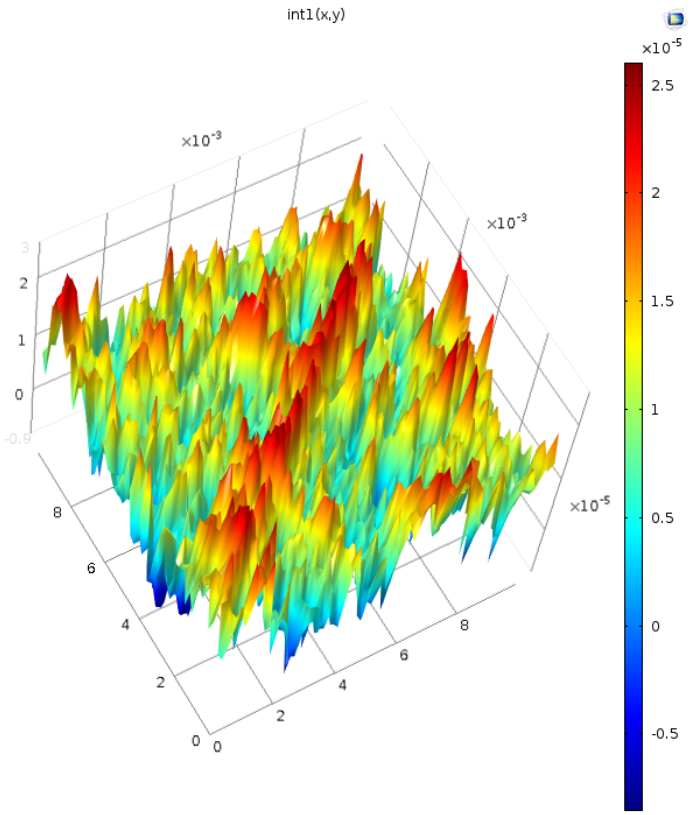


Figure 6.5: Substrate plot generated by interpolation function

6.3. Mathematical model validation

As discussed in Chap.4, we proposed several governing equations to describe or approximate different paint process. Meanwhile, MATLAB and COMSOL are selected to complete perturbed linear and non-linear simulation, respectively. In fact, the governing equations in different situations all can be concluded by Eq.4.31. Therefore, by setting some certain preconditions, results obtained by different governing equations from different solution schemes in MATLAB or COMSOL will be equivalent. This gives us the possibility to verify the reliability of our mathematical models by comparing relevant results.

In particular, one comparison of solutions to three simplified governing equations has be given. The three solutions are based on linearized solution scheme by MATLAB for flat substrate, non-linearized solution scheme (namely solving the PDE directly) by COMSOL for flat substrate, and non-linearized solution by COMSOL for substrate with very small amplitude compared to initial film thickness, respectively. The governing equations for these three situations are all 1D case extracted from Eq.4.24, Eq.4.31 and Eq.A.1, and they are shown as Eq.6.14, Eq.6.15 and Eq.6.16.

$$\boxed{\frac{\partial \delta h}{\partial t} + \frac{\gamma}{3\mu} e_0 \frac{\partial^4 \delta h}{\partial x^4} = 0} \tag{6.14}$$

$$\boxed{\frac{\partial h}{\partial t} + \frac{\gamma}{3\mu} \frac{\partial}{\partial x}\left(h^3 \frac{\partial^3 h}{\partial x^3}\right) = 0} \tag{6.15}$$

$$\boxed{\frac{\partial h}{\partial t} + \frac{\gamma}{3\mu} \frac{\partial}{\partial x}\left((h - S_a)^3 \frac{\partial^3 h}{\partial x^3}\right) = 0} \tag{6.16}$$

By utilizing S_a given in Table.6.1 and setting the initial condition as:

$$H_{initial} = 20\mu m + 5\mu m \sin(\frac{2\pi x}{L}) \tag{6.17}$$

solutions to Eq.6.14, Eq.6.15 and Eq.6.16 are shown in Figure.6.6. It can be observed that the solutions at 600s for COMSOL solution of flat substrate and COMSOL solution of tiny rough substrate almost coincide with each other. There is deviation between MATLAB linearized solution of flat substrate and two COMSOL solutions, but the deviation is as small as 0.8%. Therefore, the reliability of the mathematical models has been verified.

t

Parameter	Symbol	Value	Unit
Mean height of rough substrate	H_{sa1}	0.001	μm
Amplitude of rough substrate	A_1	0.0005	μm

Table 6.1: Parameters of tiny sinusoidal amplitude substrate chosen

6.4. Example solutions

Following the aforementioned route, we construct COMSOL models to simulate the paint process. Some example solutions are shown in this section. These example solutions are obtained in horizontal situations by utilizing sinusoidal initial layer and sinusoidal substrate.

In 1D COMSOL model, with the mathematical expression:

$$u = H + e \sin(\frac{2\pi \cdot 2x}{L}) \tag{6.18}$$

and the substrate with $n = 4$, which is:

$$Sa = H_{sa} + A \sin(\frac{2\pi \cdot 4x}{L}) \tag{6.19}$$

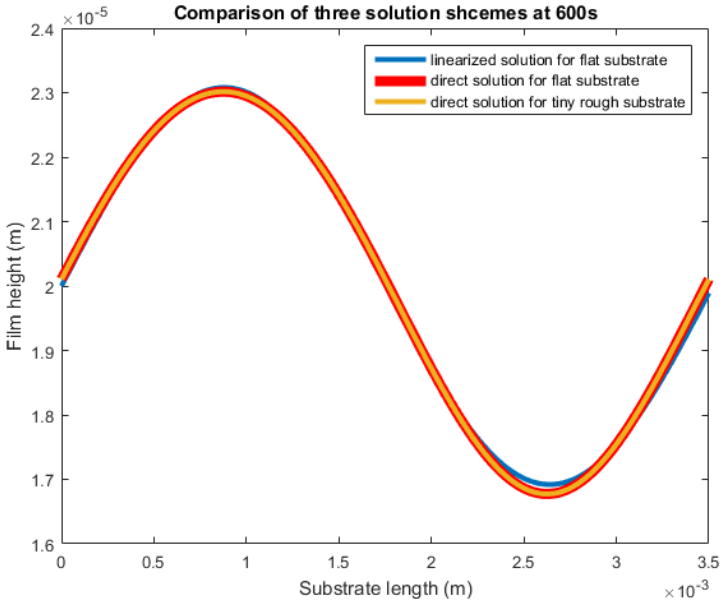


Figure 6.6: Solutions obtained by three schemes with different softwares

t

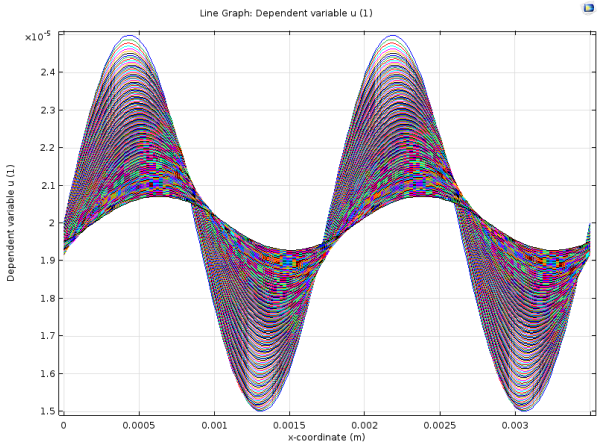
Parameter	Symbol	Value	Unit
Surcafe tension	γ	3×10^{-2}	N/m
Paint viscosity	μ	1	Pa.s
Mean thickness of the paint film	H	20.0	μm
Amplitude of the film	e	5.0	μm
Surface dimensions	L	3.52×3.52	mm^2

Table 6.2: Parameters of initial film properties

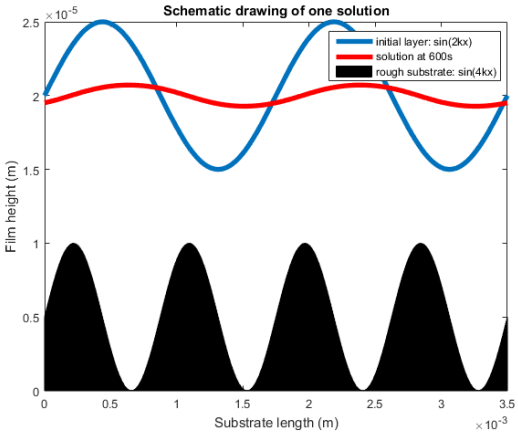
t

Parameter	Symbol	Value	Unit
Mean height of rough substrate 1	H_{sa2}	5	μm
Amplitude of rough substrate 1	A_2	5	μm
Mean height of rough substrate 2	H_{sa3}	10	μm
Amplitude of rough substrate 2	A_3	5	μm

Table 6.3: Parameters of three roughness substrates

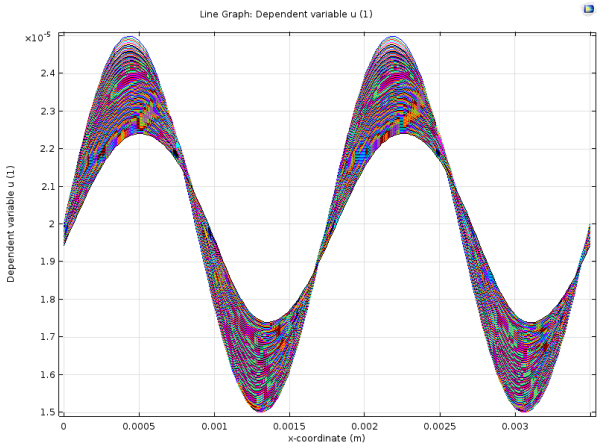


(a) Solution with substrate: $5\mu m + 5\mu m \cdot \sin(\frac{2\pi 4x}{L})$

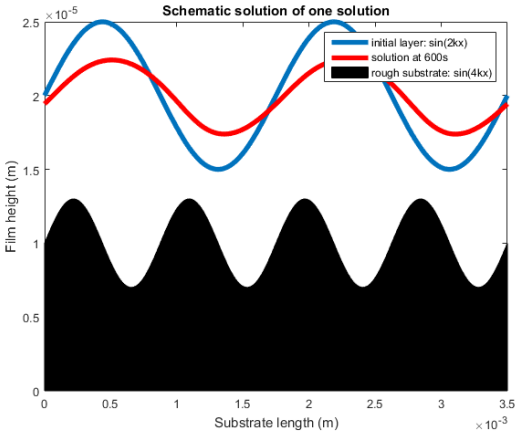


(b) schematic drawing

Figure 6.7: Comsol results and schematic plots



(a) Solution with substrate: $10\mu m + 5\mu m \cdot \sin(\frac{2\pi 4x}{L})$



(b) schematic drawing

Figure 6.8: Comsol results and schematic plots

Figure.6.7 and Figure.6.8 show two 1D solution obtained by COMSOL model with initial film parameters given in Table.6.2, and substrate parameters given in Table.6.3.

When comparing Figure.6.7 and Figure.6.8 it can be seen the amplitude of the substrate affects the paint process. With the same initial layer, the substrate with small roughness amplitude may results in a faster self-flat process while high amplitude corresponds to a slower self-flat process.

In 2D situation, the initial paint layer with sinusoidal profile are given as:

$$H + e\cos(\frac{2\pi nx}{L})\cos(\frac{2\pi ny}{L}) \tag{6.20}$$

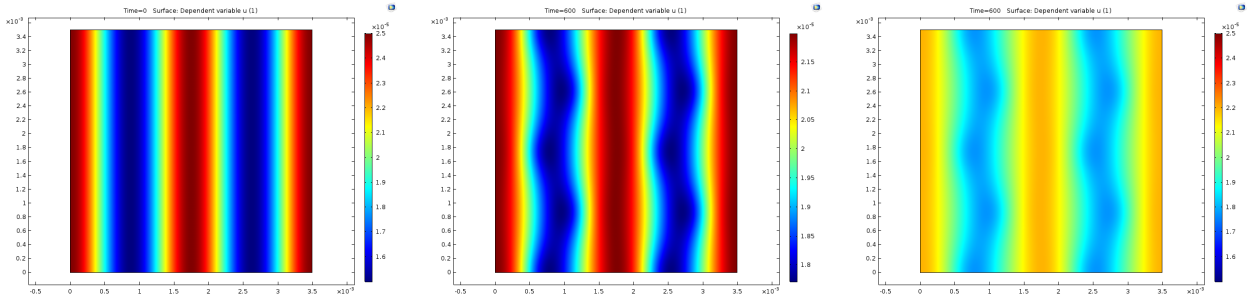
or:

$$H + e\sin(\frac{2\pi nx}{L}) \tag{6.21}$$

and the substrate is given as:

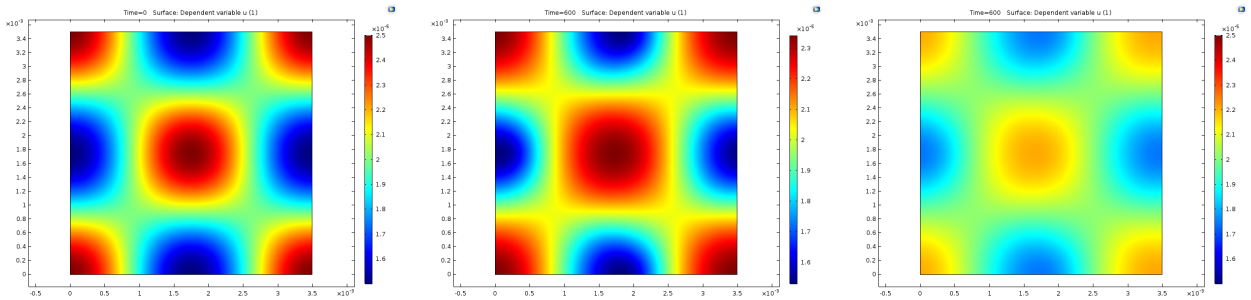
$$H_{sa} + A\sin(\frac{2\pi \cdot nx}{L})\cos(\frac{2\pi \cdot ny}{L}) \tag{6.22}$$

With these definition, we obtain several sets of solution as shown in Figure.6.9, Figure.6.10 and Figure.6.11.



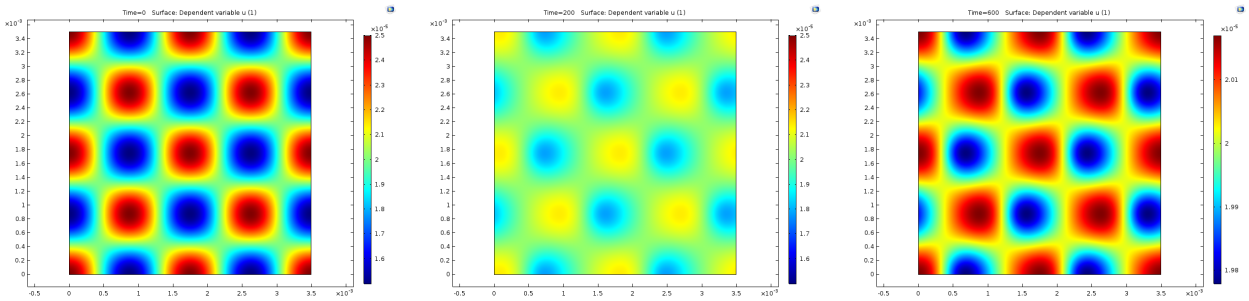
(a) Initial film (b) Solution at 600s (c) Solution with uniform colorbar

Figure 6.9: 2D results with initial film: $20\mu m + 5\mu m\sin(\frac{2\pi x}{L})$, substrate: $10\mu m + 5\mu m\sin(\frac{2\pi 2x}{L})\cos(\frac{2\pi 2y}{L})$



(a) Initial film (b) Solution at 600s (c) Solution at 600s with uniform colorbar

Figure 6.10: 2D results with initial film: $20\mu m + 5\mu m\cos(\frac{2\pi x}{L})\cos(\frac{2\pi y}{L})$, substrate: $10\mu m + 5\mu m\sin(\frac{2\pi 2x}{L})\cos(\frac{2\pi 2y}{L})$



(a) Initial film (b) Solution at 200s with uniform colorbar (c) Solution at 600s

Figure 6.11: 2D results with initial film: $20\mu m + 5\mu m \cos(\frac{2\pi 2x}{L})\cos(\frac{2\pi 2y}{L})$, substrate: $10\mu m + 5\mu m \sin(\frac{2\pi 2x}{L})\cos(\frac{2\pi 2y}{L})$

6.5. Conclusion

In summary, this chapter generally discusses how to create a COMSOL model to solve a paint process governing equation. Several key problems during COMSOL model implementations are discussed. The example solutions obtained by corresponding COMSOL model are also given in this chapter. In fact, based on the defined system architecture in Chap.4, COMSOL will work as a background solver and a COMSOL model will be implemented in the form of MATLAB code with the help of Livelink translation. Nevertheless, creating COMSOL model via COMSOL GUI is an indispensable part for our development process, since it provides reference, guidance and foundation for later implementation of MATLAB-based COMSOL solver.

7

Implementation of Rough Surface Generator

This chapter reports the design process of the Rough Surface Generator (RSG). With a rough surface generator, customized surfaces can be constructed as the painting substrates for our simulation.

7.1. Methods to realize surface textures

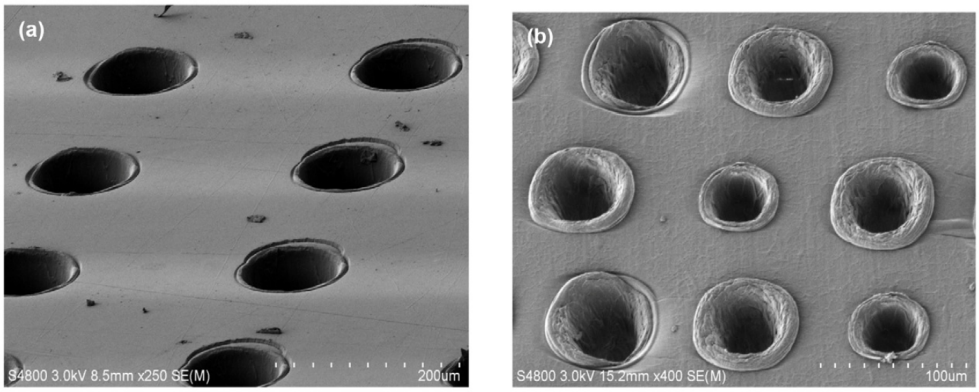
Several approaches to capture the surface properties such as regularity, directionality, and complexity have been put forward [19]. These approaches, which are shown as follows, indicate various methodology to construct a textured surface[19].

- **Statistical methods** focus on the spatial distribution of pixel intensities and generally calculate local features and derive statistics from them.
- **Model-based methods** focus on underlying texture process to construct parametric models that could create the observed intensity distribution.
- **Signal processing methods** usually analyze the frequency information of a given texture.
- **Structural methods** consider texture as composed of primitive components, they work best for describing periodically repetitive, or very regular textures.

In practice, these methodologies are considered synthetically and they provide us good references when implementing our own surface construction schemes.

7.2. Generation of rough surface with regular textures

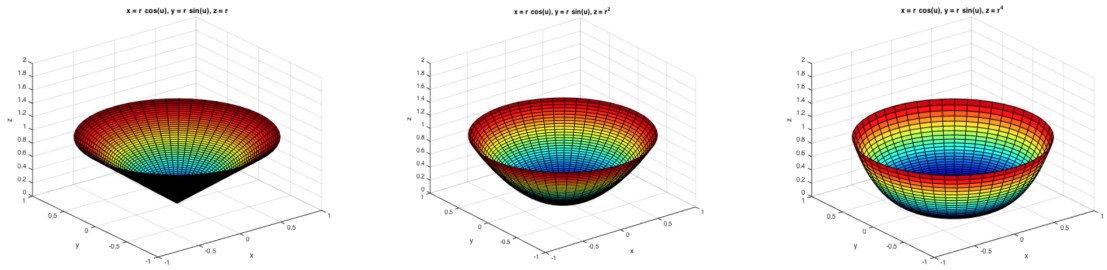
Surfaces have certain periodic, repetitive units as their surface textures can be constructed by **Structural methods**. Their relatively regular surface textures are considered as composed of primitive components. Such surfaces, as some examples displayed in Figure.7.1, are usually manufactured to realize functionalities other than paint, hence the generated surfaces discussed in this section will not be utilized as our paint substrates. The core of **structural methods** is to determine a reliable mathematical model to describe the tiny



(a) (b)

Figure 7.1: Primitives like surface textures [3]

repetitive structures on the surfaces. The 3D geometries extracted from these structures, which are known as primitives, are various, as examples given in Figure.7.2.



(a) (b) (c)

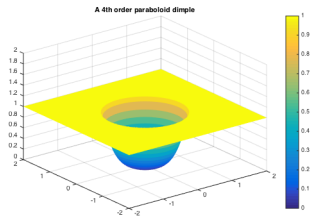
Figure 7.2: 3D primitives with certain mathematical expression

Figure.7.3 shows several surface primitives built by typical conic surfaces. They can be utilized to construct surfaces shown in Figure.7.1, since the dimples on the real surfaces in Figure.7.1 have quite similar shapes to the conic-surface-built primitives.

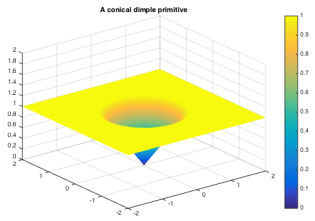
In practice, the primitives geometries usually require more complicated mathematical expressions than simple conic models. For example, one mathematical model of a spherical dent is given as:

$$Z(x,y) = -r_p \cdot \left(\sqrt{(\epsilon + \frac{1}{\epsilon})^2 - (x^2 + y^2)} - (\frac{1}{4\epsilon} - \epsilon) \right) \tag{7.1}$$

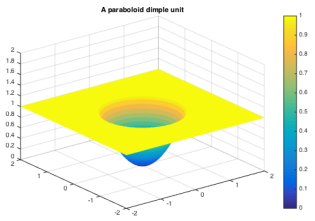
where $\epsilon = h_p/2r_p$, with h_p the center valley depth and r_p the sphere radius at the surface, and with feature density $S_p = \pi r_p^2/4r_L^2$ with r_L the feature area radius. Compared to a basic paraboloid dimple primitive, more physical quantities have been introduced in Eq.7.1 so that a more reliable and realistic textured surface can be described.



(a) A 4th-order paraboloid dimple primitive



(b) A conical dimple primitive



(c) A paraboloid dimple primitive

Figure 7.3: 3D primitives with certain mathematical expression

7.3. Generation of rough surface with statistical random textures

Apart from surface with regular primitives, there are more surfaces with relatively stochastic surface textures, which in fact, are more widely used in surface paint projects. These surfaces usually have textures that can be characterized by certain statistical laws or spatial distributions, therefore, **Statistical methods** will be one of the best choices to build such surfaces.

Statistical methods requires to describe surface textures from statistical perspective, hence relevant indicators in this perspective like spatial functions, correlation functions, etc. will be employed as mathematical tools to describe random textured surfaces. These conceptions will be introduced in this section, and how they are employed in our rough surface generator to construct a random textured surface will be discussed in Section.7.5.2.

7.3.1. Spatial functions

Typical spatial functions are known as the auto covariance (or autocorrelation) function, structure function, or power spectral density function. Spatial functions offer a means of representing the properties of all wavelengths, or spatial sizes of the feature.[20]

Auto Covariance Function has been the most popular way of representing spatial variation, it is defined as follows:

$$R(\tau) = \lim_{L\to\infty} \frac{1}{L} \int_0^L z(x)z(x+\tau)dx \tag{7.2}$$

where L is the sampling length of the profile. The normalized form of the auto covariance function is called auto correlation function and is given as:

$$C(\tau) = \lim_{L\to\infty} \frac{1}{L\sigma^2} \Big[z(x)-m\Big]\Big[z(x+\tau)-m\Big]dx = \Big[R(\tau)-m^2\Big]/\sigma^2 \tag{7.3}$$

The **Structure Function** in an integral form for a profile $z(x)$ is,

$$S(\tau) = \lim_{L\to\infty} \frac{1}{L} \int_0^L \Big[z(x)-z(x+\tau)\Big]^2 dx \tag{7.4}$$

The function represents the mean square of the difference in height expected over any spatial distance τ. The two principal advantages of SF are that its construction is not limited

to the stationary case, and it is independent of the mean plane.[20] Structure function is related to Auto covariance function as:

$$S(\tau) = 2\sigma^2[1 - C(\tau)] \tag{7.5}$$

The **Power Spectral Density Function** is another form of spatial representation and provides the same information as the auto covariance function or structure function, but in a different form. In fact, the power spectral density function is the Fourier transform of the auto covariance function:

$$P(\omega) = P(-\omega) = \int_{-\infty}^{\infty} R(\tau) exp(-i\omega\tau) d\tau \tag{7.6}$$

where ω is the angular frequency in $length^{-1}$. $P(\omega)$ is defined over all frequencies, both positive and negative, and is referred to as a two-sided spectrum. The power spectral density function is interpreted as a measure of frequency distribution of the mean square value of the function, that is the rate of change of the mean square value with frequency.

Among these three spatial functions, we will select auto-covariance function to characterize surface because of its Fourier transform, which is power spectral density function, containing all the frequency-domain information.

7.3.2. Auto-correlation function

In practice, auto-covariance function is related to the more commonly used **auto-correlation** function. An **auto-correlation function** is auto-covariance function normalized by dividing the variance σ^2.[2] The auto-covariance can be thought of as a measure of how similar a signal is to a time-shifted version of itself with an auto-covariance of σ^2 indicating perfect correlation at that lag. The normalization with the variance will put this into the range [-1, 1].

In our particular case, The surface height auto-correlation function is given as $C(R)$, where describes the extent to which knowledge of the height at one point on the surface does, on average, determine the height at some point R away[22]. The definition of auto-correlation function is:

$$C(R) = \frac{1}{\sigma^2} \langle h(r) h(r+R) \rangle \tag{7.7}$$

where $h(r)$ represents the height at point r.

The correlation function will always have the property that $C(0) = 1$, arising from the definition in Eq.7.7. It will usually also have the property that $C(\infty) = 0$, since points at a large distance apart are uncorrelated. In practice, it is interesting to consider height distributions based on Gaussian correlation function or Exponential correlation function. 2D Gaussian correlation function has the form

$$C(x, y) = \exp(-(x^2 + y^2)/\lambda_0^2) \tag{7.8}$$

while 2D exponential correlation function has the form

$$C(x, y) = \exp(-(|x| + |y|)/\lambda_0) \tag{7.9}$$

In our rough surface generator, the aforementioned correlations functions will be applied by doing convolution with some raw surface data, so that the new surface with certain correlation lengths and height distribution will be obtained. The detail will be discussed in Section.7.5.2.

7.4. Function and system architecture design

The system requirements for RSG are defined in Chap,5. With defined system requirements, the corresponding system functions to be implemented can be also found in Figure.5.2. Therefore, the corresponding software architecture for RSG module is built as shown in Figure.7.4.

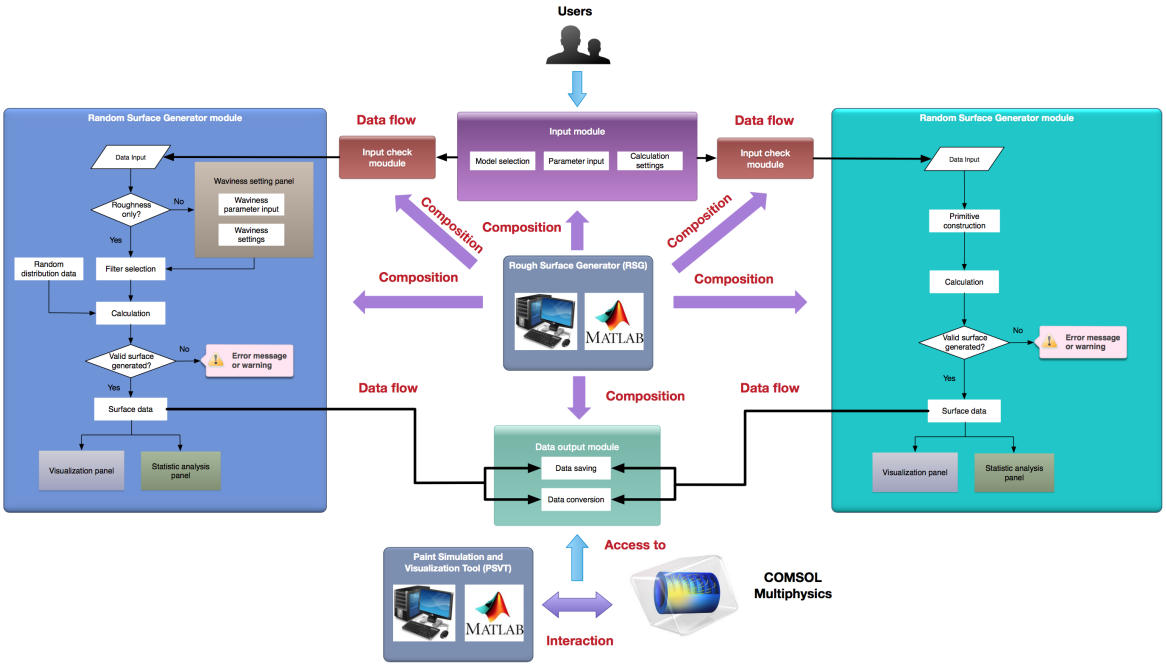


Figure 7.4: Architecture of RSG component

7.5. System implementation

With defined functions and built software architecture, system implementation process about several important modules will be discussed in this section. Two versions of RSG have been implemented. The prototype 1 includes random textured surface and regular textured surface generation function. The prototype 2 is integrated with PSVT, which improves random textured surface generation function but deletes regular textured surface generation function.

7.5.1. Regular textured surface generator

A flow chart can be extracted from Figure.7.4 as our guidance for implementation, as shown in Figure.7.5

It can be seen with well-defined primitives, it's easy to construct a regular surface. During the implementation process, the constructed primitives are placed repetitively with a certain period. Figure.7.7 displays several surfaces constructed by **structral method** with pre-defined primitives. These surfaces plots are also implemented in the RSG visualization panel as shown in Figure.7.6.

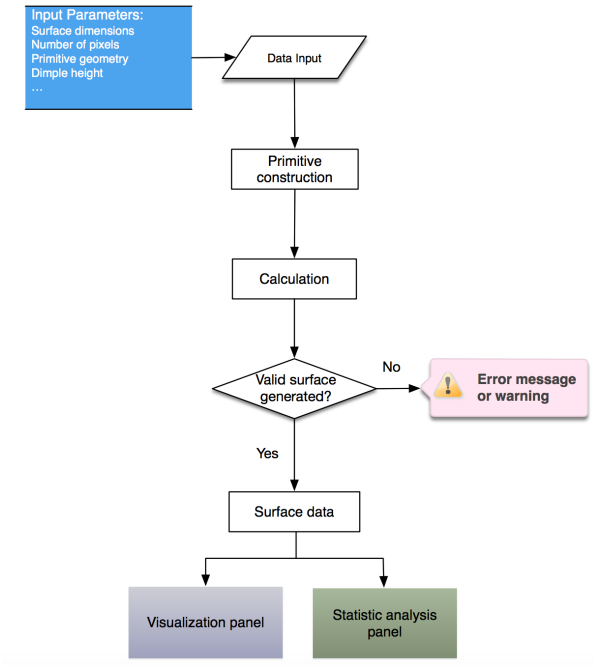


Figure 7.5: Flow chart of regular surface generator module

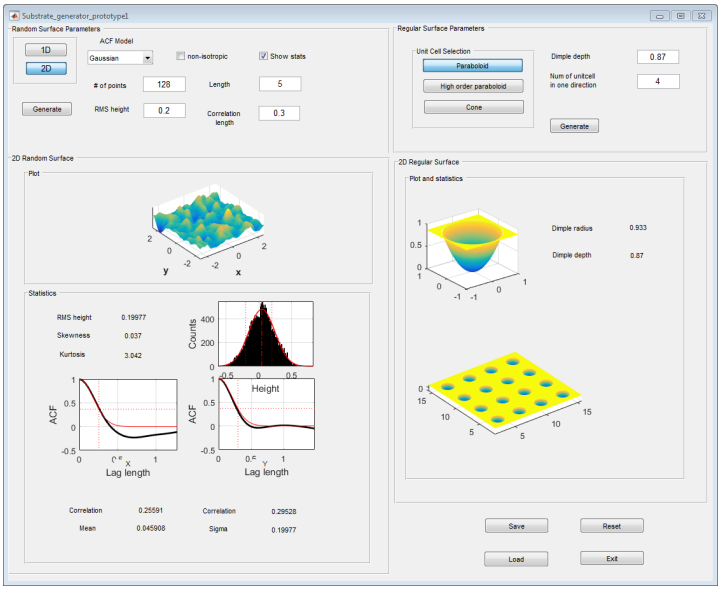
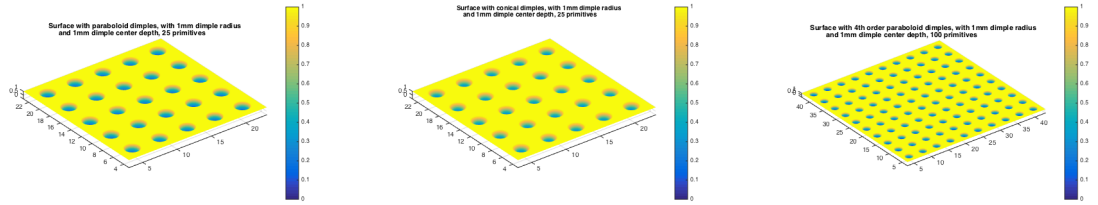


Figure 7.6: RSG prototype 1 with regular textured surface generation function



(a) Surface with paraboloid dimples

(b) Surface with conical dimples

(c) Surface with high order paraboloid dimples

Figure 7.7: 3D primitives with certain mathematical expression

7.5.2. Random textured surface generator

A flow chart can be extracted from Figure.7.4 as our guidance for implementation, as shown in Figure.7.8

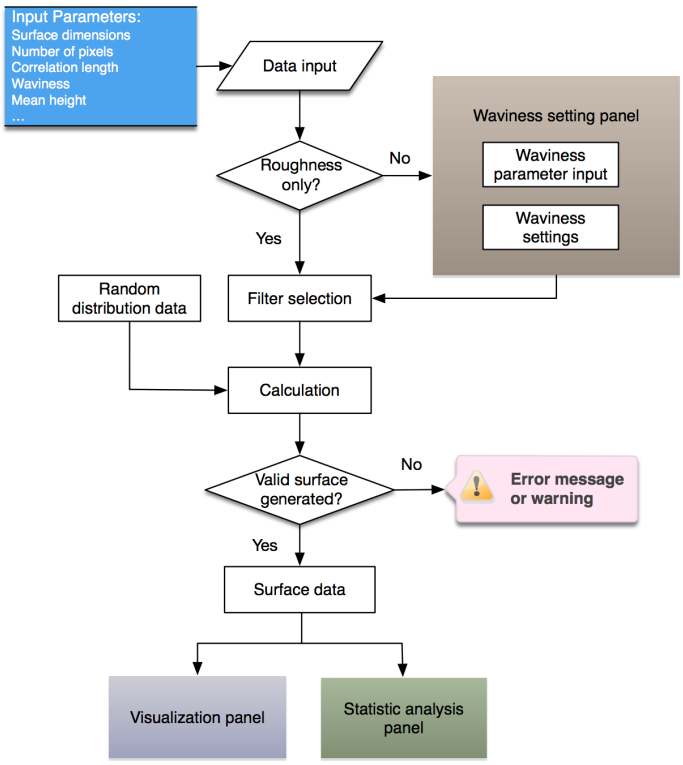


Figure 7.8: Flow chart of random surface generator module

The flow charts shows a decision making box when determine the length scale of the surface textures. The surface with only short-scale textures (roughness) or with both long-scale (waviness) and short-scale textures will be considered.

For a surfaces with only roughness, the theories mentioned in 7.3.2 and 7.3.1 are sufficient to help us generate random surfaces with statistical height distributions and certain correlation length. Suppose an uncorrelated stochastic rough surface Z (random distribution data in flow chart), which has norm distribution with mean 0 and standard deviation h, is employed as the original surface. Then a convolution between surface Z and the Gaussian correlation function has been performed to make the surface have a Gaussian height

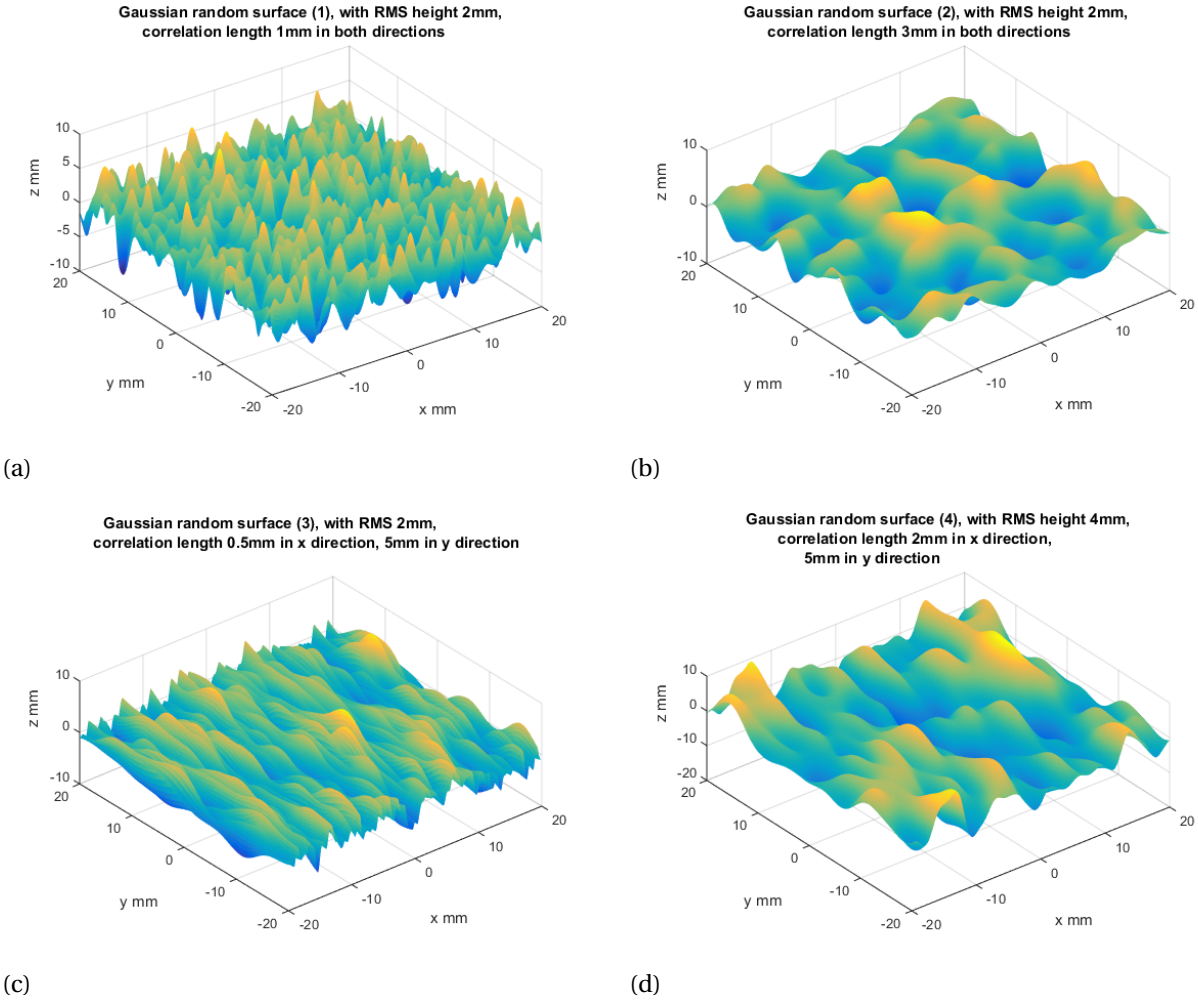
distribution. The convolution theorem is given as:

$$f * g = \mathscr{F}^{-1}\{\mathscr{F}\{f\} \cdot \mathscr{F}\{g\}\} \tag{7.10}$$

hence the new surface with normalizing factors in front can be obtained as

$$f = \frac{2L}{\sqrt{\pi} N l_c} \mathscr{F}^{-1}\{\mathscr{F}\{Z\} \cdot \mathscr{F}\{\exp(-(x^2 + y^2)/l_c^2)\}\} \tag{7.11}$$

where L is defined as the new surface length, N is the number of sample points, and l_c is the correlation length. Here the Gaussian correlation function plays a filter role [22], as discussed in 7.3.2. Figure.7.9 shows several examples of surfaces with Gaussian height distribution roughness.



(a) (b)

(c) (d)

Figure 7.9: (*a*) Gaussian random surface with 2mm RMS height, 1mm correlation length in both directions. (*b*) Gaussian random surface with 2mm RMS height, 3mm correlation length in both directions. (*c*) Gaussian random surface with 2mm RMS height, 0.5mm correlation length in x direction, 5mm correlation length in y direction. (*d*) Gaussian random surface with 4mm RMS height, 2mm correlation length in x direction, 5mm correlation length in y direction.

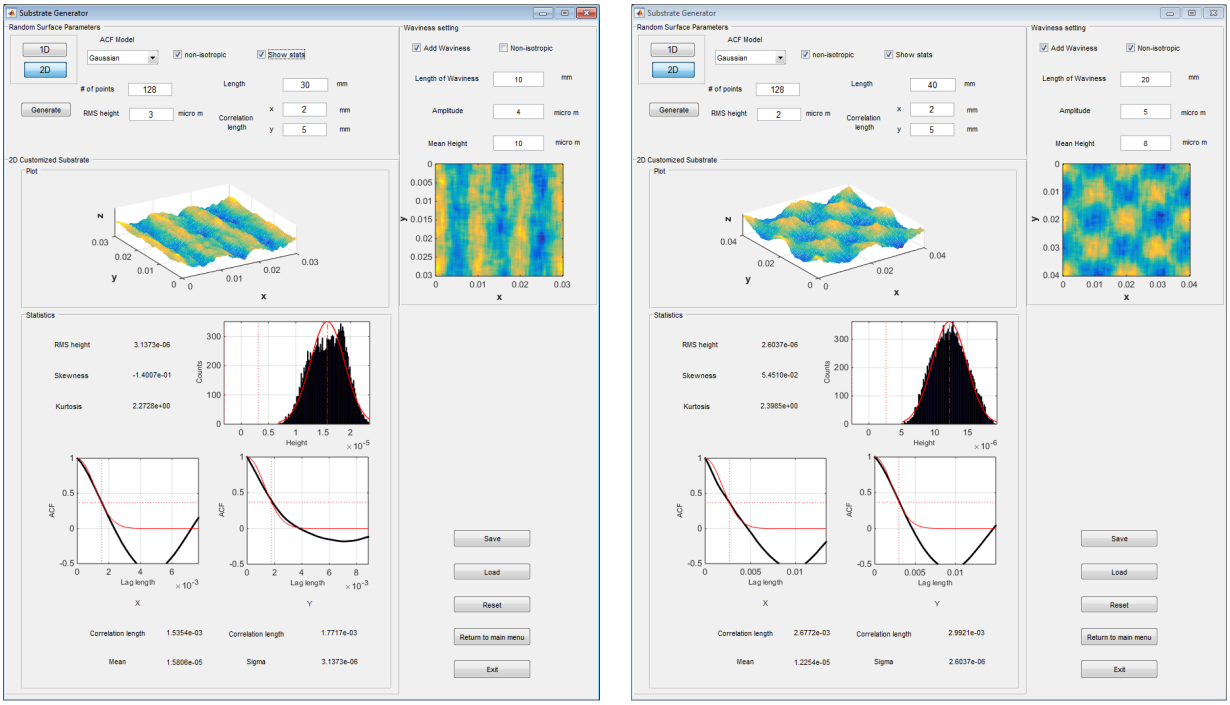
A texture with waviness L usually has a sinusoidal form:

$$f_W = \cos(2\pi x / L) \tag{7.12}$$

The surface combining roughness and waviness can be simply written as:

$$f = f_R + f_W; \tag{7.13}$$

Example surfaces with both short-scale roughness and long-scale sinusoidal waviness are constructed in RSG prototype 2 as shown in Figure.7.10.



(a) Substrate with non-isotropic sinusoidal waviness (b) Substrate with isotropic sinusoidal waviness

Figure 7.10: RSG prototype 2 with sinusoidal waviness and Gaussian roughness

Furthermore, several comments regarding implementation of random surface module are shown as follows:

1. The sinusoidal waviness can be either isotropic or non-isotropic, corresponding to two options implemented in waviness settings, as shown in Figure..

2. Filter selection box will provide filter options for generating surface roughness, gaussian filter and exponential filter will be the most widely used ones.

3. The surface roughness textures can be either isotropic or non-isotropic, which means the correlation length in x and y directions can be the same or be different. One selection box will be designed for implementing this difference.

7.5.3. Substrate data conversion for COMSOL

The generated surface will be utilized as the paint substrate in the simulation process. When employing COMSOL as our background solver, paint substrate data generated by MATLAB will become input for the built-in COMSOL project. Therefore, the data form of the substrate must be transformed into the form that COMSOL can identify.

As discussed in Chap.6, the external data import can be realized by several alternatives with COMSOL options **Geometry**, **Material** and **Function**. We will select interpolation function under **Function** option to import our generated substrate.

To be more precise, the RSG-generated substrate data conversion for COMSOL use will be implemented as follows:

1. The generated substrate data, which is in matrix form, will be saved as a text file first.
2. The data in text file will be converted into the form of 'grid', which can be identified by COMSOL. This process is still fulfilled by MATLAB
3. The text file with grid format data will be called by COMSOL interpolation function portal. This process will be executed through MATLAB-built PSVT, since it is PSVT that requires input substrate for paint simulation.
4. The import interpolation function will be used for defining the variable S_a, which will be used as the rough substrate in COMSOL built-in PDEs.

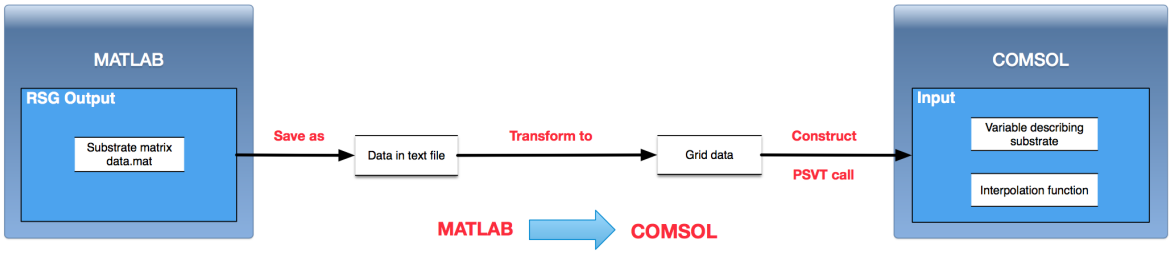


Figure 7.11: Unidirectional data exchange process between COMSOL and MATLAB

As shown in Figure.8.2.2, the data exchange implementation of RSG is an unidirectional process, namely MATLAB ⇒ COMSOL, which is different to the bidirectional data exchange process in PSVT. Meanwhile, the process calling and inputing text file grid data to COMSOL interpolation function portal is not executed by RSG but by PSVT. The detail will be discussed in Chap.8

7.6. Conclusion

In this chapter, the implementation process of RSG has been discussed. The chapter starts with discussing methods to realize surface textures. Meanwhile, some relevant theories on how to construct rough surfaces with certain surface textures, especially surfaces with statistical random textures and surfaces with regular textures, have been investigated. Based on methods, theories and system requirements, a software architecture has been built to guide the RSG implementation. Several implementation issues categorized by system components are also investigated in this chapter.

8

Implementation of Paint Simulator and Visualization Tool

This chapter reports the implementation process of paint simulator unit and visualization module of the software tool. Based on the constructed system architecture, PSVT will be implemented by components and some problems met during implementation process will be discussed.

8.1. System architecture design

To fulfill the system requirements defined in Chap.5, various corresponding system functions to be realized can be found in Figure.5.2 in Chap.5. Being different to the main system architecture shown in Figure.5.4, a sub-system architecture for PSVT has been built and shown in Figure.8.1.

8.2. PSVT

8.2.1. Background computation and simulation

As mentioned in Chap.6, a COMSOL model has to be translated into MATLAB code in order to couple with our MATLAB-based system. Hence, it can be seen in the sub-system architecture a MATLAB-built COMSOL model becomes the core of the computation and simulation module. In Chap.6 we created models in COMSOL GUI for solving our problems in advance, which actually aims at providing the best support and reference to help us fulfill this transferring.

The MATLAB-built COMSOL model is created with the help of COMSOL Livelink library. In general, the MATLAB-built COMSOL model is implemented by creating a typical MATLAB function[23]. User input will be added as the input of this function, while the output will be a 1×1 *ModelClient* containing all the results data from COMSOL computation[23]. An example is shown as follows:

```
function model = paint_simulation_v1(mu,length,gamma,H,A,e,ea,kx_sub,
    ky_sub,tt,t_step)
```

COMSOL Livelink contains necessary functions and commands to run COMSOL model from MATLAB terminal, which will be filled in this main MATLAB function.

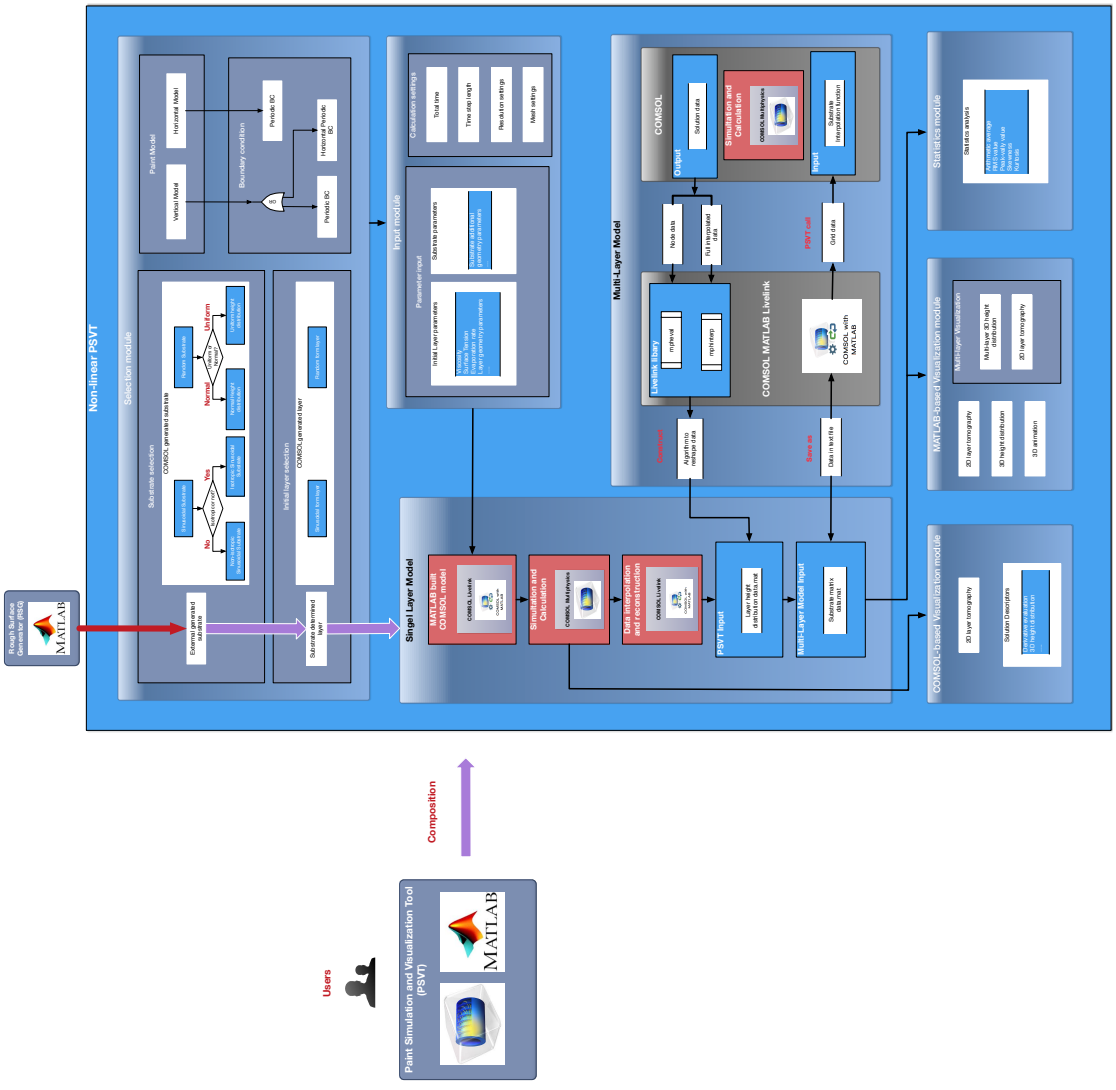


Figure 8.1: Architecture of PSVT component

The basic structure and the function division of a COMSOL model will be preserved in the main MATLAB function representing this COMSOL model. Specifically, the function realized by COMSOL GUI will be realized by corresponding MATLAB code instead. Several key nodes of the COMSOL GUI are list as examples, which will be translated into MATLAB code for discussion:

1. **Key parameter input**
   To translate COMSOL parameter input into MATLAB code, *model.param.set* command from Livelink library is obligatory[23]. Use *num2str* to build the required strings for input parameters, one example parameter setting section should then look like:

   ```
   % Parameter input
   model.param.set('gamma', num2str(gamma));
   ```

2. **Physical settings**
   Physical settings include the contents of geometry construction, mesh construction, boundary conditions, built-in function settings and so on. They can also be implemented with certain commands and functions from COMSOL Livelink library[23].

   A geometry is defined with the command *model.geom()*:

   ```
   % Build geometry
   model.geom('geom1').create('sq1', 'Square');
   model.geom('geom1').feature('sq1').set('size', 'L');
   model.geom('geom1').run;
   ```

   With this module, a square geometry with size L^2 has been constructed. As discussed in Chap.6, we use the geometry property to define the dimension and the general shape of our substrate, while substrate textures are introduced by setting variables that will be contained in the governing equations, like variable S_a:

   ```
   % Substrate textures
   model.variable('var1').set('Sa', 'A+ea*(sin(2*pi*kx_sub*x/L)*cos
       (2*pi*ky_sub*y/L))');
   ```

   ,which defines a sinusoidal substrate texture.

   The mesh setting is considered simultaneously. The default mesh setting of a COMSOL model is:

   ```
   % Mesh settings
   model.mesh('mesh1').autoMeshSize(3);
   model.mesh('mesh1').run;
   ```

   , which generates meshes with triangle elements. As discussed before, such meshes are not our preference. Therefore, mapped meshes are constructed as follows:

   ```
   % Mesh settings
   model.mesh('mesh1').create('map1', 'Map');
   model.mesh('mesh1').feature('map1').create('dis1', 'Distribution')
       ;
   ```

```
model.mesh('mesh1').feature('map1').feature('dis1').set('numelem',
    '256');
model.mesh('mesh1').run('map1');
```

Via *model.mesh('mesh1').feature('map1').feature('dis1').set('numelem', '256');*, the elements number of the mesh has been fixed at 256. In principle, higher number of elements usually means a better resolution and more accurate results. However, due to COMSOL's built-in interpolation algorithms, the impact on the solutions caused by mesh settings has been minimized, which means, a finer mesh usually means longer computation time but no substantial improvements for solution accuracies[16][23]. Therefore, a fixed number of meshes can be utilized and the resolution and accuracy of the solution can be adjusted by later interpolation process[16].

As discussed in Chap.6, in our COMSOL model several built-in functions have been called to describe the substrate textures or fluid layer profiles. One example of implementing COMSOL built-in uniform distributed random functions is given as follows:

```
% Random function
model.func.create('rn1', 'Random');
model.func('rn1').set('nargs', '2');
model.func('rn1').set('mean', 'h_sub');
model.func('rn1').set('uniformrange', 'r');
```

Regarding boundary conditions, *model.physics().feature()* is the command prepared[23]. A periodic boundary condition has been set in the following code:

```
% Boundary conditions
model.physics('g').feature.create('pc1', 'PeriodicCondition', 1);
model.physics('g').feature.create('pc2', 'PeriodicCondition', 1);
model.physics('g').feature('pc1').selection.set([1 4]);
model.physics('g').feature('pc2').selection.set([2 3]);
```

With the above code two couples of periodic boundary conditions have been defined for boundary 1,4 and boundary 2,3, as shown in Figure.8.2.

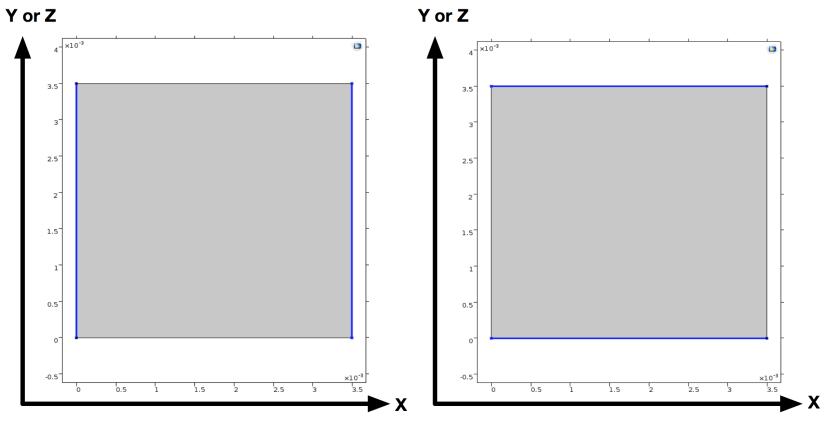


Figure 8.2: Periodic boundary conditions

3. **Governing equation input**

   Similarly, by utilizing *model.physics* commands, the governing equation can be expressed by MATLAB code as well. In COMSOL GUI, the governing equation is filled into a sheet with certain grammar. When implementing in MATLAB, the information of the equation and the structure of the sheet are all preserved in the MATLAB code.

4. **Plot groups for result display**

   As can be seen from the system architecture, a COMSOL data-based visualization panel is designed. The COMSOL built-in results plot will be displayed on this panel[23]. To create a plot group in COMSOL model, The following MATLAB code can be used:

```
% Plot group
model.result('pg12').create('surf1', 'Surface');
```

   Plot group "No.12" has been built via this piece of code. Various settings can be added to the plot group, in order to display the results in our desired form. For example, by introducing the corresponding command settings, plot group "No. 12" will show a 3D height distribution plot instead of 2D surface plot, as shown in Figure.2d3d. A gray scale color setting will be applied instead of default rainbow color setting if some proper settings are coded.

The above contents show several examples of implementation commands in one complete MATLAB-built COMSOL model. Usually, when the command

```
model.sol('sol1').runAll;
```

appears in one MATLAB-built COMSOL model, the computation process will be executed once by COMSOL.

8.2.2. Interaction and data processing between COMSOL and MATLAB

The interaction and data processing between COMSOL and MATLAB majorly include two aspects:

- Accessing MATLAB data in COMSOL
- Post-processing COMSOL results in MATLAB

Figure.8.3 extracted from PSVT architecture generally displays this bidirectional data processing procedure.

In Chap.7 a data conversion process has been investigated roughly for RSG-generated substrate data in order to further import these data into COMSOL. With "grid" form data after conversion, the further data exchange and interaction process between MATLAB and COMSOL will continue with PSVT. It can be realized that importing MATLAB customized substrate data will be one task of **"accessing MATLAB data in COMSOL"** aspect.

As shown in PSVT system architecture, the customized substrate saved as a text file will be read into MATLAB-built COMSOL model. In COMSOL GUI such a text file can be imported directly from interpolation function portal [16] while in MATLAB-built COMSOL model, this process can be done with certain commands:

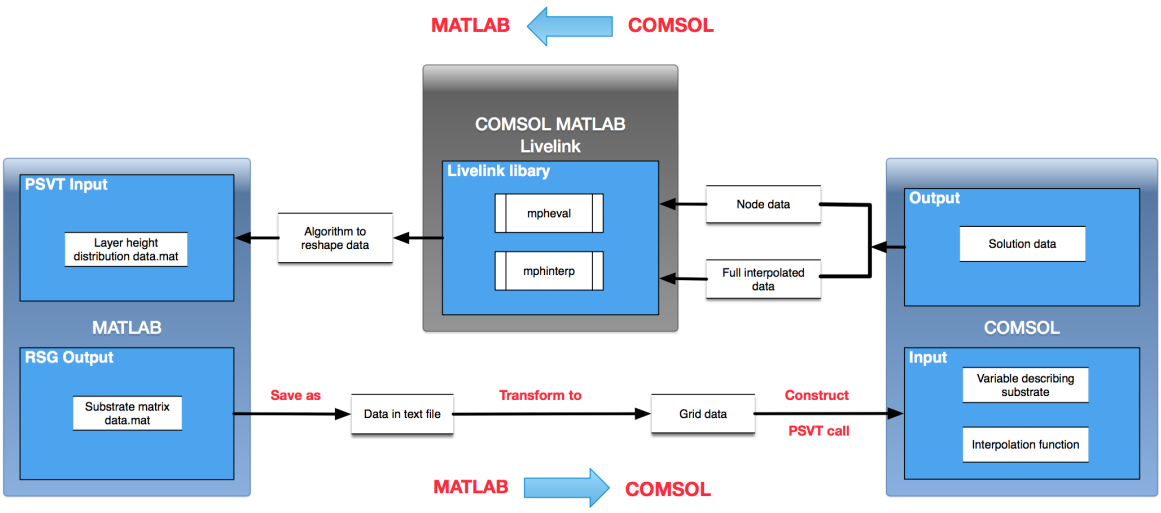


Figure 8.3: Bidirectional data exchange process between COMSOL and MATLAB

```
% Import customized substrate
model.func.create('int1', 'Interpolation');
model.func('int1').set('source', 'file');
model.func('int1').set('filename', pathname);
model.func('int1').importData;
```

Here "pathname" is a string variable containing the substrate text file name and its path. This string variable called in PSVT main script will be used as one input of the MATLAB-built COMSOL model function[16].

When COMSOL finishes its background computation, the solution data will be sent back to MATLAB terminal for further investigation. This process refers to the process **"post-processing COMSOL results in MATLAB"**. The solution data obtained by COMSOL will be called and applied by visualization module, statistic analysis module and multilayer module in the following phases.

As discussed in Section.8.2.1, the COMSOL model in MATLAB platform is a typical MATLAB function, with an 1×1 *ModelClient* output, namely:

```
function model = paint_simulation_v1(mu,length,gamma,H,A,e,ea,kx_sub,
    ky_sub,tt,t_step)
```

The solution data are all saved in this output "model". To extract the data, functions from COMSOL Livelink library have to be employed. Two most important functions we are going to discussed are *mpheval* and *mphinterp*[16].

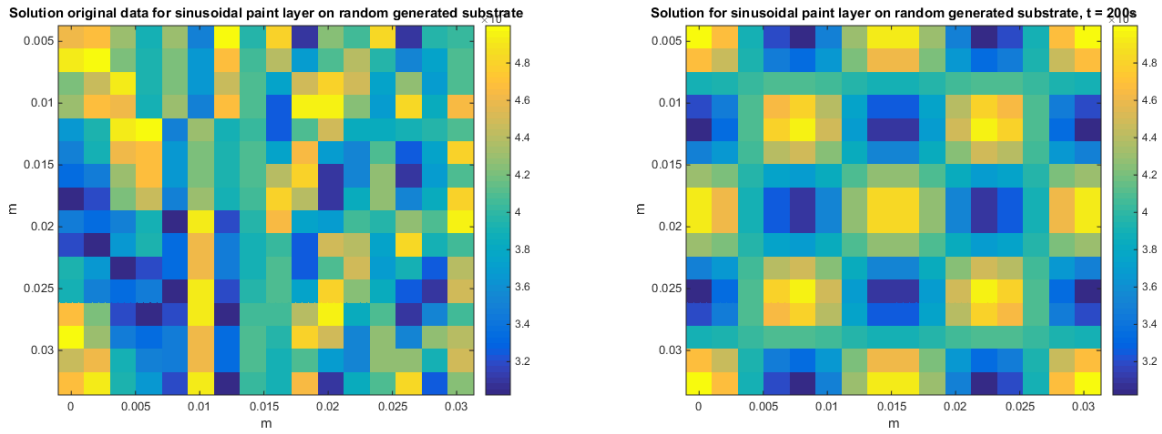
- **mpheval**: The general purpose of this function is to evaluate solution expressions on node points. Typically, the solution data (on node points) from COMSOL model regarding variable u will be all saved in a new MATLAB struct named $data$ with the command[16]:

  ```
  data = mpheval(model,'u')
  ```

- **mphinterp**: The general purpose of this functions is to evaluate solution expressions in arbitrary points or data sets. The solution data can be also saved in a new MATLAB cell[16].

Both these two commands focus on send COMSOL solution data to a MATLAB struct, but the solution data are organized in quite different ways in a MATLAB struct. Plenty of work has to be done to reform the coordinates system and reshape the layer height data.

As can be seen from Figure.8.4, the coordinates system and height distribution saved in $data$ struct by function $mpheval$ is scrambled. By applying some reshape algorithms, the organized solution data can be seen from Figure.node2. However, it can be seen the plots suffers from a very low resolution because only nodes point of the solution data are interviewed when calling $mpheval$.



(a) $mpheval$ generated original data

(b) Data after reshape algorithm

Figure 8.4: Solution data processed by $mpheval$

Figure.8.5 shows the solution data plots obtained by function $mphinterp$. The results shows correct data arrangement and the resolution is satisfied. This result is also obtained by applying certain reshape codes that are different to those applied with $mpheval$. The solution plots obtained in this way will be utilized as the final results shown in MATLAB-based visualization module. Meanwhile, its validity can be verified by comparing with COMSOL internal solution plots obtained by command $mphplot$.

By transforming solution data obtained by COMSOL into MATLAB recognizable array form, these data can be further processed by MATLAB. For example, as can be seen from Figure.8.1, the statistics that will be displayed on statistics panel including *Arithmetic average, RMS value, Peak-Valley value, Skewness, and Kurtosis*. These statistics are all calculated by interviewing COMSOL send-out data in MATLAB form.

Meanwhile, for further calculation, especially the multi-layer system calculation, the data that previously describes fluid paint layer will now describes the solidified new substrate. Therefore, the solidified new substrate will send back to COMSOL again and hence an iteration is constructed as shown in Figure.8.3. The details about multi-layer system implementation will be discussed in later sections.

8.2.3. Results display and visualization module

As shown in the system architecture, for single layer model, two visualization modules are realized, one based on COMSOL built-in results and another based on processed data in MATLAB.

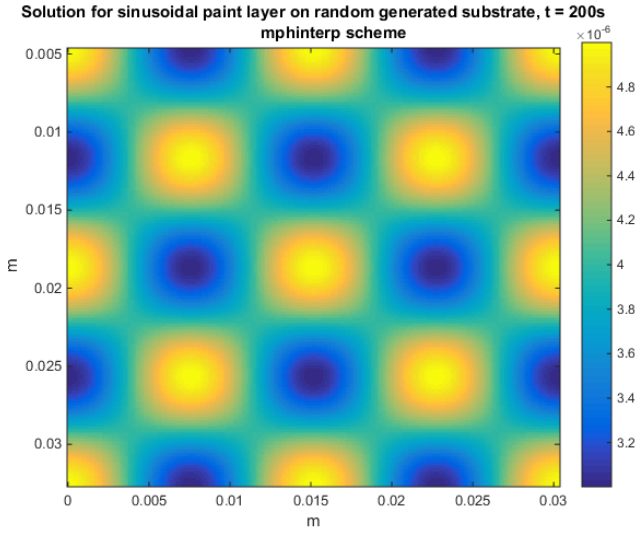


Figure 8.5: Solution data processed by *mphinterp*

COMSOL built-in results are solution plots generated by COMSOL software internally. The COMSOL-based visualization module focuses on displaying the plots generated in COMSOL plot group on the MATLAB user interface. As discussed in Chap.8.2.1, plenty of plot groups are created by corresponding commands in the MATLAB-built COMSOL model, when the COMSOL model is executed, certain solution plots will be generated and saved in these plot groups[16]. By applying certain functions, these solution plots can be easily interviewed from MATLAB user interface.

The function *mphplot* is a widely used function to call COMSOL generated plots from MATLAB user interface[16][23]. An example of its application is given as follows:

```
% Call results plot in MATLAB
mphplot(model,'pg1','rangenum',1);
```

In this function, the solution plot in COMSOL plot group 1 has been interviewed. Therefore, the image in plot group 1 will be displayed in a MATLAB figure. During this process, no data processing is executed since the solution plot is sent to MATLAB as a constructed image directly.

On the COMSOL-based visualization panel, several paint process descriptors including arithmetic average, RMS value, Peak-Valley Value, Skewness, and Kurtosis, as shown in Figure.8.7, are also displayed. Once defined, it is possible to export simulation process descriptor plots directly from COMSOL plot groups to MATLAB terminal. As can be seen from Figure.8.6, these descriptors are usually derivatives or second derivatives of the paint layer, which represent the velocities or the acceleration along coordinates directions.

Besides COMSOL-based visualization module, MATLAB visualization module is also implemented by interviewing processed data and reconstructing the desired plots with these data. The data interaction process has already been discussed in Section.8.2.2, it can be seen that before visualizing paint layer with MATLAB, data reconstruction and interpolation have been done.

As can be seen from Figure.8.7, with MATLAB-based visualization panel, the paint layer at arbitrary time step can be visualized. This is different to COMSOL-based visualization panel, since the paint layer plots displayed on COMSOL-based visualization panel are lim-

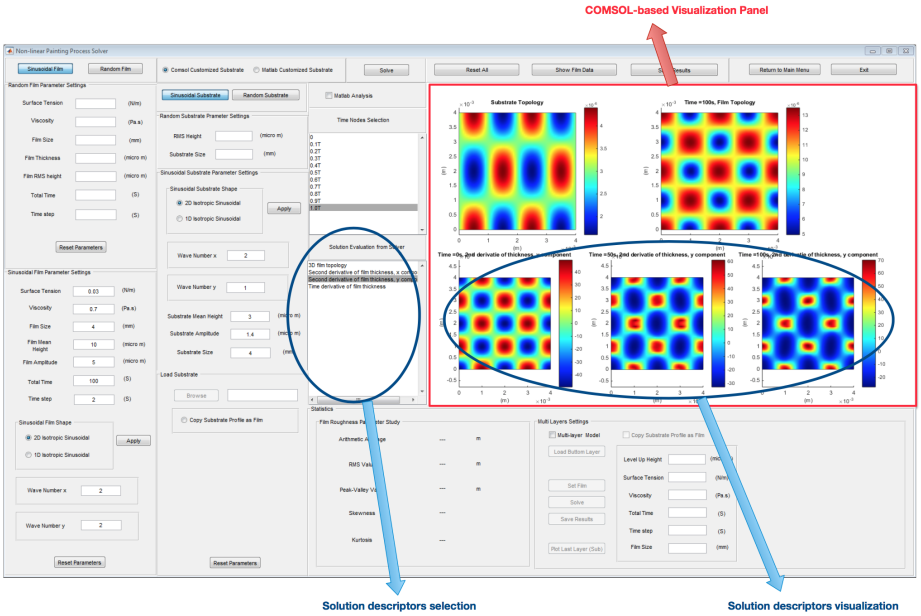


Figure 8.6: COMSOL-based visualization panel of PSVT *mphinterp*

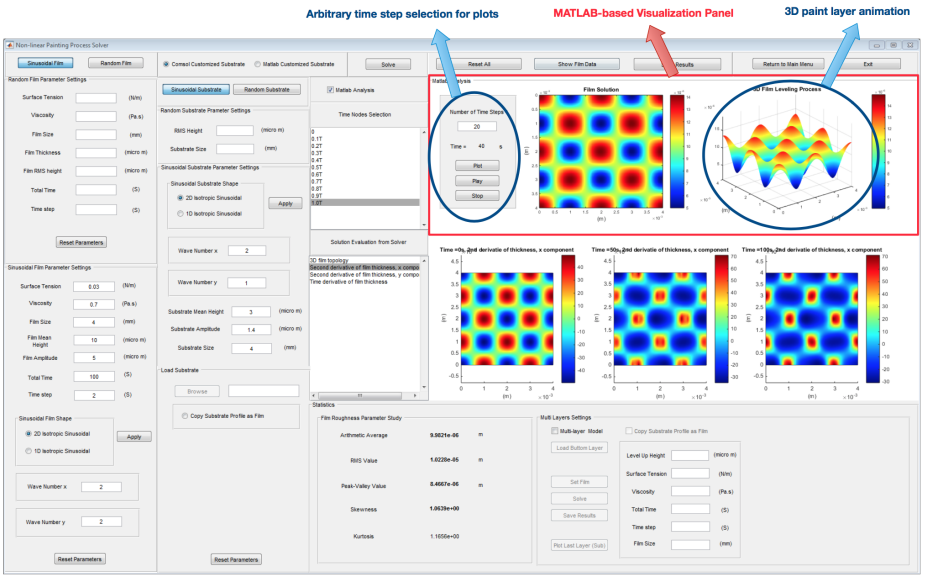


Figure 8.7: MATLAB-based visualization panel of PSVT *mphinterp*

ited by the number of plot group constructed. Meanwhile, since interviewing the solution data at each time step is available from MATLAB, a 3D animation can be also generated and shown on MATLAB-base visualization panel.

8.2.4. Multilayer model implementation

As discussed in Chap.2, the automotive paint system is a multi-layer system, while different layers are endowed with different functions. In our PSVT, the simulation of multi-layer paint process has been implemented.

The basic assumption of realizing multi-layer paint system in our situation is:

The previous paint layer was completely solidified before next paint process hence can be regarded as a solid substrate for next paint process.

With this assumption, a new simulation can be implemented by inputting new fluid layer parameters and setting new simulation parameters. Meanwhile, the data processing generally follows the iteration shown in Figure.8.3, namely:

1. Converting previous stable paint layer solution data into MATLAB array data and saving into text file.
2. Calling the array data from MATLAB and send it into COMSOL as next paint substrate via interpolation function portal.
3. Fulfilling the simulation process in COMSOL.
4. Extracting the solution data from COMSOL and doing data processing to generate new paint layer solution data in MATLAB array form.
5. Fulfilling this iteration to realize multi-layer paint simulation process.

Figure.8.8 shows PSVT running with multi-layer simulation model. The visualization panel of multi-layer paint process is MATLAB-based. A general trend that the paint layer will become more and more flat will be observed, when a large number of paint layers are introduced.

8.3. Conclusion

In this chapter, the implementation of PSVT has been discussed. The chapter starts with displaying sub-system architecture design. Following this architecture, several topics including data processing between MATLAB and COMSOL, results visualization, multi-layer paint simulation have been investigated.

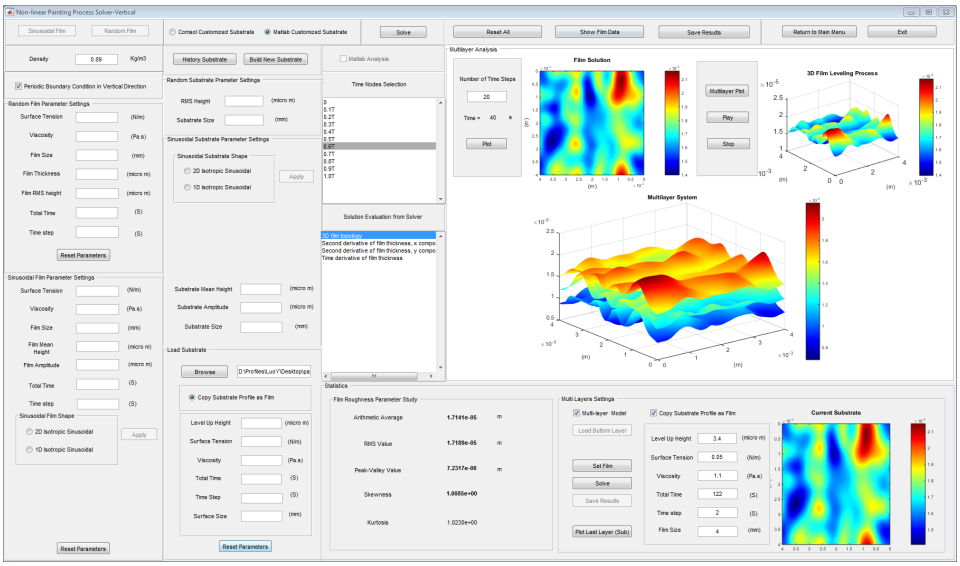


Figure 8.8: PSVT multi-layer simulation model

9 Validation

This chapter discusses the validation process of the project. Basically, the measurement data obtained from the designed experiments have been compared with the results obtained by PSVT.

9.1. Experimental validation

Two schemes of validation experiments are designed during the validation phase. The first validation experiment is realized by the application of a mineral oil layer on a rough surface. The oil is chosen as a liquid to be studied as, contrary to paints, oils are much less influenced by chemical reactions and evaporation. The new formed oil layer will be observed after a certain time. The second validation experiment is realized by employing TATA steel painted samples; a comparison between measured paint layer data of the sample and the simulation results of the same painted sample has been carried out.

9.1.1. Validation with oil

As mentioned, our first validation experiment is realized by wiping certain amount of oil on a rough steel substrate. After a certain time period, the new oil layer distribution will be observed and then compared with the simulation results. The oil selected is *Shell Vitrea Oils M 320*, which is usually used for lubrication of heavy duty industrial bearings and circulating systems. The substrate sample provided by TATA steel is shown in Figure.9.1, and the microscope we used for measurement is shown in Figure.9.2.

Figure.9.3 show two areas from the steel sample, the first measurement size is $8.068mm \times 6.072mm$ and the second measurement size is $15.936mm \times 2.507mm$. As can be seen from Figure.9.3, no waviness information can be observed from this sample.

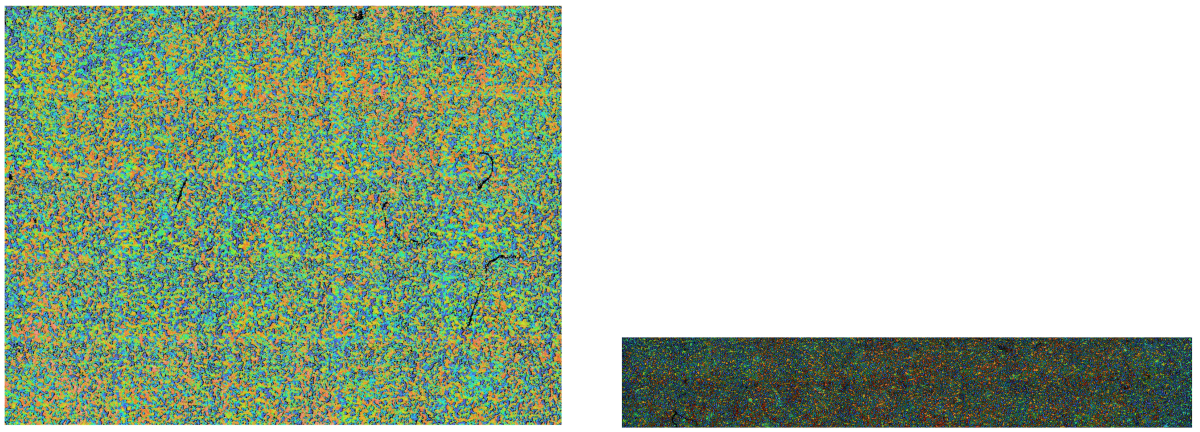
Due to the limitation of experimental setups, it is difficult to control the initial fluid layer thickness and topography, especially when the required layer thickness is merely several micrometers. Therefore, in practice the layer thickness is estimated by following formula:

$$H = \frac{m}{\rho \cdot S} = \frac{V}{S} \tag{9.1}$$

which means the layer thickness is obtained by dividing the oil volume with the surface area size. During the experiments, suppose $5\mu L$ oil is wiped on a $1.5cm \times 1.5cm$ surface,

Figure 9.1: Steel sample as substrate

Figure 9.2: Microscope used for measurement



(a) Size $8.068mm \times 6.072mm$ measurement result (b) Size $15.936mm \times 2.507mm$ measurement result

Figure 9.3: Substrate measurement results for validation

the average layer thickness is therefore:

$$H = \frac{m}{\rho \cdot S} = \frac{V}{S} = \frac{5\mu L}{1.5cm \times 1.5cm} = 22.22\mu m \tag{9.2}$$

For validation, we cut an area with the size of $2.1 \times 2.1mm$ from the big wiped area as our measured substrate area. Due to the pixel lost shown in dark blue color and several singularities with extreme values, a spline interpolation has been carried out to fill in those NAN (Not A Number) points and discard singularities. The substrate data after interpolation is shown in Figure.9.4.

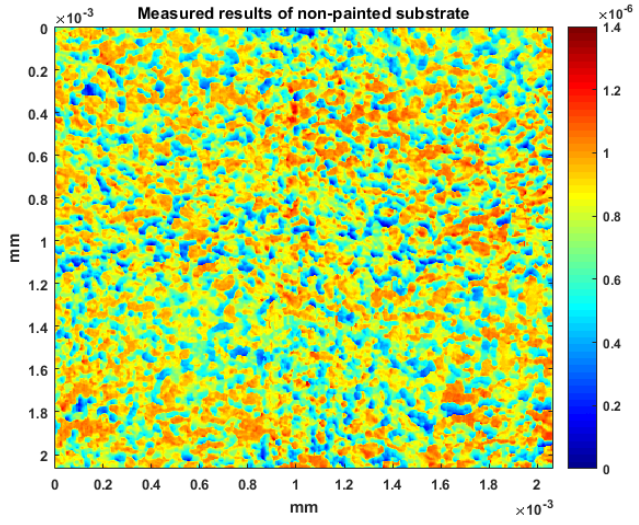


Figure 9.4: Surface used for validation, interpolated measured substrate data, size $2.1mm \times 2.1mm$

The experiment is realized by simulating and measuring the oil layer spreading process on this $2.1mm \times 2.1mm$ area. The relevant experimental parameters are shown in Table.9.1.

t

Parameter	Symbol	Value	Unit
Oil volume	V	5	μL
Oil density	ρ	870	Kg/m^3
Average layer thickness	H	22.22	μm
Surface tension of the oil	γ	0.026	N/m
Viscosity of the oil	μ	320	$mPa \cdot s$
Substrate size	S	4.41	mm^2

Table 9.1: Parameters for validation

Firstly, the initial paint layer is measured by the microscope, which is shown in Figure.9.5. The data will be used as our initial condition for starting the paint simulation. Figure.9.6 shows the validation results displayed on PSVT. To zoom in, Figure.9.7a shows the simulation results at 30s after initial layer measurement, while Figure.9.7b shows the measured solidified layer on the substrate.

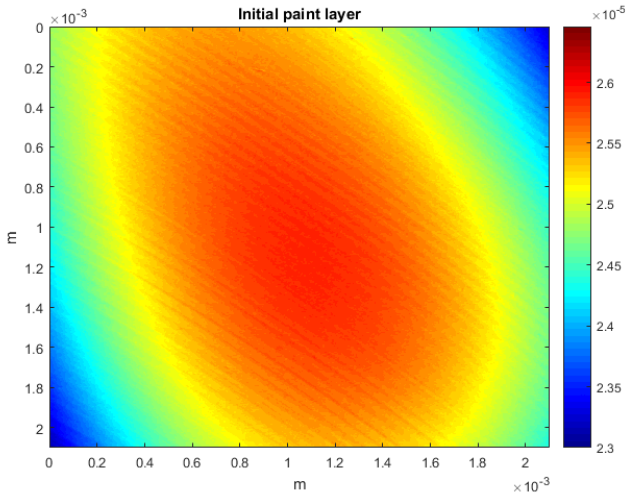


Figure 9.5: Initial layer measured as the input of simulation

It can be seen that there are considerable similarities between the simulation results and measured results, especially the new paint layer's height range. The two plots show their peaks at almost the same location, namely between x-coordinate $1mm-1.2mm$ and y coordinate $1.2mm-1.4mm$ (or $0.6mm-0.8mm$ from inverse y coordinate). The difference is possibly introduced by unpredictable setups displacement or even wind.

Meanwhile, it can be also recognized that in this situation the substrate topography (roughness) is not reflected in the eventual paint layer topography, caused by a thick layer of liquid. Based on some reference[4][8][15], we can also infer that the substrate waviness may have more significant effect than substrate roughness on paint process when a thick paint layer has been introduced.

9.1.2. Validation with painted substrate

Another validation route is based on TATA steel painted steel substrate, which is shown in Figure.9.8. In this experiment, instead of using real liquid, we set a new initial condition by

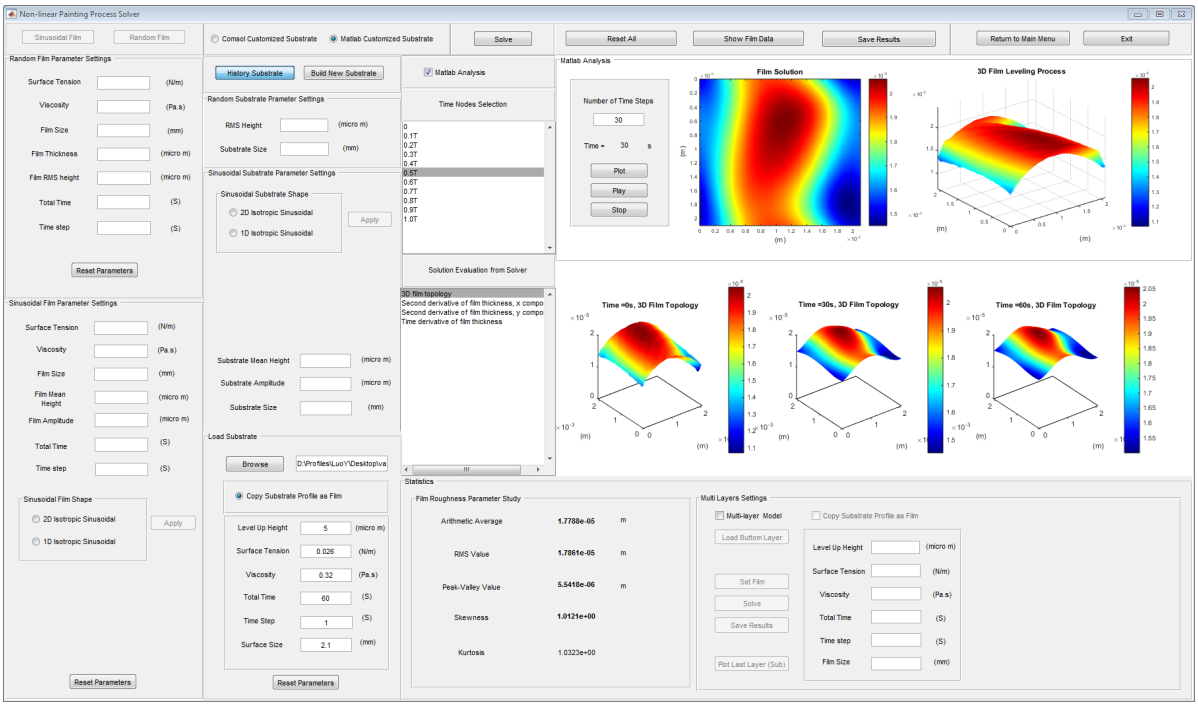
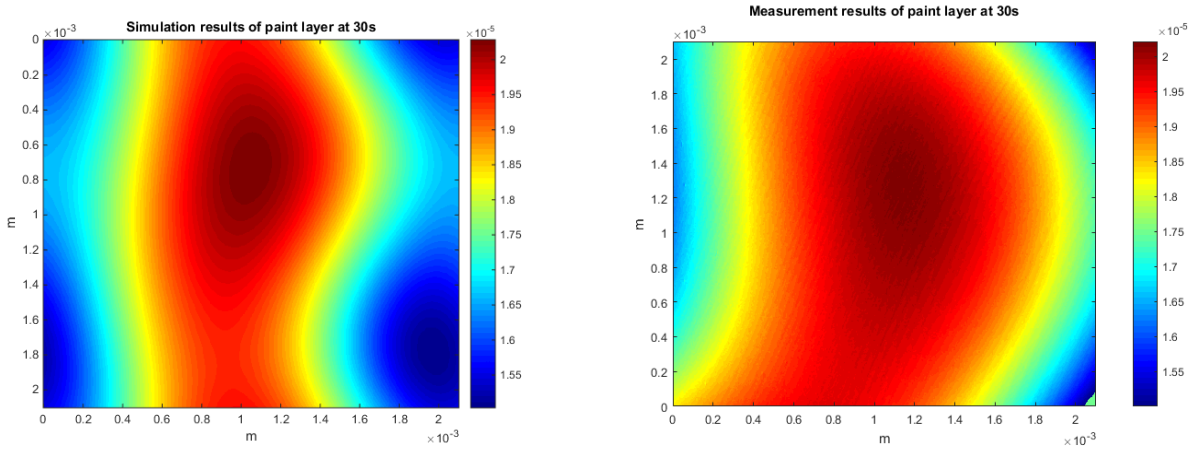


Figure 9.6: Simulation results displayed on PSVT



(a) Simulation result at 30s

(b) Measurement result at 30s

Figure 9.7: Measurement and simulation results with thick paint layer

assuming the initial layer follows the shape of the given substrate data, and with a much thinner average thickness, eg., several micrometers(about $5\mu m$). With this initial layer and given substrate data as input, a new simulation result based on the experimental setting shown in Table.9.2 is obtained and shown in Figure.9.9a. Meanwhile, the real painted layer on the sample after solidification in Figure.9.8 is measured and shown in Figure.9.9b.

Figure 9.8: Painted sample

t

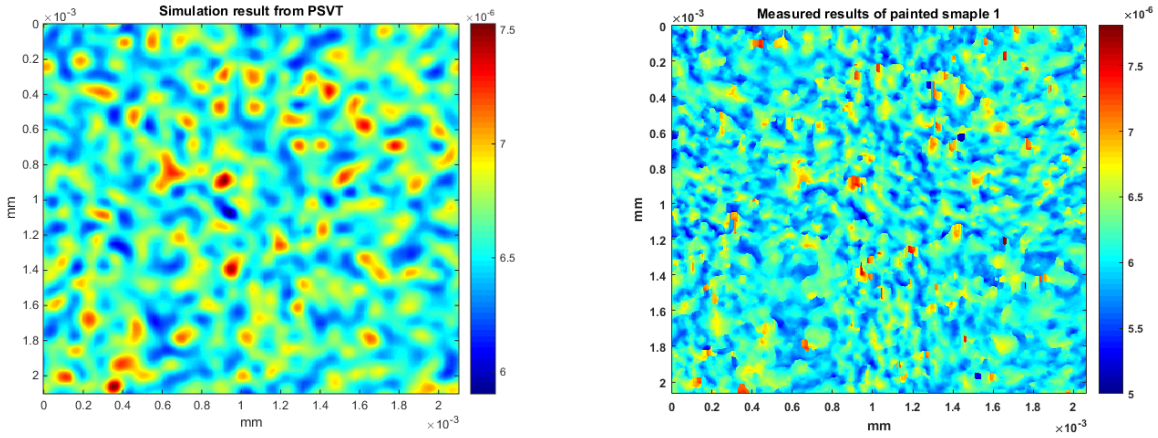
Parameter	Symbol	Value	Unit
Average layer thickness	H	5.5	μm
Surface tension of the liquid	γ	0.035	N/m
Viscosity of the liquid	μ	520	$mPa \cdot s$
Substrate size	S	4.41	mm^2

Table 9.2: Parameters for validation

Although the simulation result shows a slight higher thickness ($6-7.5\mu m$ compared to $5-7.5\mu m$) than the measurement, similarities are observed from the layer profile between the simulation results and the measured results shown in Figure.9.9b. The substrate topography is reflected in the eventual paint layer when the paint layer thickness is small, which indicates that the layer thickness and substrate roughness are both key factors that determine the paint process.

9.2. Conclusion

By comparing with the two validation results, it can be concluded that when the fluid layer thickness is comparable to the substrate roughness, the substrate roughness will have a considerable effect on the paint process, which means the topography of the substrate will be reflected in the stable paint layer in some sense. On the contrary, when the layer thickness is much bigger than the substrate roughness, there will be a flattening process and the effect from substrate to the eventual layer shape is limited.



(a) Simulation result

(b) Measurement result

Figure 9.9: Measurement and simulation results with thin paint layer

10

Conclusion and outlook

10.1. Conclusion

To conclude, this report elaborates the whole process of the PDEng project *Design of A Paint Simulation and Visualization Tool for Automotive Surfaces*. Several important outcomes and conclusions regarding the project and paint process from this study are listed as follows:

1. A high-order non-linear mathematical model has been constructed to describe paint process on rough surfaces.

2. A PSVT has been developed to simulate paint process on rough surfaces in various cases.

3. A RSG has been developed to generate rough surfaces that can be used as substrates for paint process simulation.

4. The paint simulation has been validated by comparing the simulation results with measurement results.

5. Compared to surface roughness, long-scale surface waviness will have a more significant effect on paint process, especially when the paint layer is thick. The series of sample used for validation did not show waviness information, but the PSVT shows such simulation results and this conclusion is supported by lots of reference.[4][8][15][21]

6. The layer thickness has a certain effect on paint process. When the layer thickness is relatively high compared to the substrate roughness, the surface roughness topography will not be reflected in the eventual layer shape, which means the paint layer will be almost flat regardless of the substrate roughness. On the contrary, when the layer thickness is comparable to the substrate roughness, the eventual layer height distribution will be significantly affected by substrate roughness.

In summary, our designed PSVT together with supporting components RSG successfully revealing the basic relation between surface textures and fluid layer properties regarding the surface paint appearance. With its certain potentials, the PSVT and validation experiments are also available for further improvements in the future applications, the relevant contents will be discussed in next section.

10.2. Recommendations and outlook

Based on the outcomes obtained and conclusion drawn, it can be realized that there is still potential for our designed system to improve and some optimizations can be achieved in the near future to obtain a better performance. Starting with this, several recommendations are given as follows:

1. The sample used for validation did not show a lot of waviness as roughness details were restricted to high frequencies. In the future simulation and experiments, waviness is recommended to be one focus for further investigation, validation based on waviness-dominated substrate is expected to be realized.

2. In our current validation measurement procedures, measuring the liquid layer on the steel substrate is always a big problem. Although the confocal microscopy used has certain functions to measure transparent layers, in practice the measured results shows plenty of pixels lost and the accuracy is not always satisfied. Therefore, the validation experiment can be improved in several ways. For example, by investigating different kinds of liquid and select the most suitable one for both paint process and experimental setups (eg., non-transparent liquid), the measurement results can be more accurate and the validation can be more reliable.

3. In our current simulation model, evaporation is considered as an additional linear term that has a very limited effect on our simulation results. However, in reality the evaporation term is more complicated and it affects the paint process considerably, especially when the painted samples are put into oven for further layer forming. Therefore, in the future work, a more accurate mathematical model that describes temperature-dependent evaporation process has to be implemented.

4. The phosphating layer is not considered during our simulation and validation process. However, the phosphating layer exists on some of our measured samples, which may affects the validation process by introducing unpredictable difference between simulation and measurement. Therefore, a more comprehensive simulation model is expected to be constructed to describe paint process when phosphating layer is taken into account. To realize this, solidified phosphating layers can be measured in advance and added into our PSVT as substrate layers. Alternatively, the effect of the phosphating process on the surface topography should be modeled. However, this is complex, involving processes like crystallization. As far as is known to the author, no suitable models are available in the literature.

A

Perturbation theory and linearized equations

Eq.4.30 and Eq.4.31 are both high-order non-linear partial differential equations. which are difficult to solve. In practice, such a model can be simplified when introducing some certain assumptions. The simplified model can be solved easier and will provide us some early stage reference for discussing.

We take Eq.4.31, which describes the fluid layer distributes on a substrate horizontally, as an example. Assuming that the amplitude of surface modulation is relatively small ($10^{-3} smaller$) compared to the layer thickness and substrate dimensions, then Eq.4.31 can be linearized by setting $h = h_0 + \delta h$, expanding it in powers of δh, and keeping lowest order terms.[8] Denoting the mean paint thickness as e_0, the linearized equation can be obtained as:

$$\frac{\partial \delta h(x,y,t)}{\partial t} = -\frac{\gamma}{3\mu} e_0^3 \Big(\frac{\partial^4 \delta h(x,y,t)}{\partial x^4} + 2\frac{\partial^4 \delta h(x,y,t)}{\partial x^2 \partial y^2} + \frac{\partial^4 \delta h(x,y,t)}{\partial y^4} \Big) \tag{A.1}$$

This method simplifies the by employing perturbation δh, hence avoid some difficulties when considering rough surface substrates. The surface schematic plot is shown in Figure A.1.

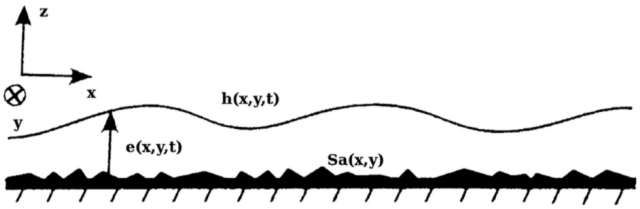


Figure A.1: Thin paint film flow, linearized

The simplest case of this Eq.A.1 is 1D case, which can be written as:

$$\frac{\partial \delta h(x,t)}{\partial t} = -\frac{\gamma}{3\mu} e_0^3 \frac{\partial^4 \delta h(x,t)}{\partial x^4} \tag{A.2}$$

It can be realized that the analytical solution for Eq.A.2 is not difficult to find. Since $\frac{\partial^4 sin(x)}{\partial x^4} = sin(x)$, the solution of Eq.A.2 has a sinusoidal form with an exponential decay

factor. Precisely, the solution for Eq.A.2 can be written as:

$$\delta h(x,t) = sin(kx)e^{-\lambda t} \tag{A.3}$$

with

$$k = \frac{2\pi}{L} \tag{A.4}$$

$$\lambda = \frac{\gamma}{3\mu} e_0^3 k^4 \tag{A.5}$$

Here L is defined as the period of the surface unit, and λ is the time decay factor. Moreover, it is also possible to find the analytical solution for Eq.A.1. Suppose Eq.A.1 has a solution in $L^2(R)$, in the corresponding Fourier space it fulfills:

$$\frac{\partial \hat{\delta h}}{\partial t} = \frac{\gamma}{3\mu} e_0^3 (\xi_x^4 + \xi_x^2 \xi_y^2 + \xi_y^4) \hat{\delta h} \tag{A.6}$$

Hence:

$$\hat{\delta h}(t) = \hat{h}(0) \exp\left(\frac{\gamma}{3\mu} e_0^3 (\xi_x^4 + \xi_x^2 \xi_y^2 + \xi_y^4)\, t\right) \tag{A.7}$$

where $\xi = (\xi_x, \xi_y)$ is the Fourier wave vector. Therefore[8],

$$\delta h(t) = \frac{1}{4\pi^2} \int_{R^2} \hat{h}(0) \exp\left(\frac{\gamma}{3\mu} e_0^3 (\xi_x^4 + \xi_x^2 \xi_y^2 + \xi_y^4)\, t\right) \exp(i(\xi_x x + \xi_y y)) d\xi \tag{A.8}$$

Eq.A.8 gives the analytical solution for Eq.A.1 via Fourier transform scheme. Although a perfect flat surface does not exist in reality, in most situation the substrate roughness is relatively small compared to layer thickness and substrate dimensions. Therefore, it's sufficient to use this model to approximate some certain circumstance. In principle, this approximation will demonstrate the common sense that the fluid layer will try to flat itself on a flat substrate. From academic perspective, this approximation provides a possible alternative, namely a Fast Fourier Transform (FFT) scheme, to give the numerical solution for the paint process models. From physical reality perspective, this approximation fits the physical reality when a macro-scale paint problem has been introduced. In automotive paint, when we zoom out and observe a large scale of paint surface, eg, a car door, the substrate is actually quite smooth.

Furthermore, this assumption also supports our validation results when thick paint layer is introduced. As discussed in Chap.9, when the paint layer is as thick as about $48\mu m$ and the substrate roughness is just several μm big, the substrate topography is not reflected in the eventual paint layer since the thick layer tries to flat itself in general.

Bibliography

[1] Georgios P. Petropoulos, Constantinos N. Pandazaras, J. Paulo Davim, *"Surface Texture Characterization and Evaluation Related to Machining"*, Department of Mechanical and Industrial Engineering, University of Thessaly, Department of Mechanical Engineering, University of Aveiro, pp. 37-65.

[2] David. J. Whitehouse, *"Handbook of Surface and Nanometrology"*, University of Warwick, The Institute of Physics, London, pp.8-26, 2003.

[3] A.A.G. Bruzzone, H.L. Costa b, P.M. Lonardo, D.A. Lucca, *"Advances in engineered surfaces for functional performance"*, Dipartimento di Ingegneria della Produzione, Termoenergetica e Modelli Matematici, University of Genoa, Genoa, Italy, School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, USA, Manufacturing Technology 57, pp. 750-769, 2008.

[4] O. Deutscher, D. Paesold, K. Korner, H.-G. Weyen, S. P. Jupp, *"Characterising the surface waviness of hot dip galvanised steel sheets for optical high-quality paintability (Carsteel)"*, Luxembourg: Office for Official Publications of the European Communities, 2009.

[5] Blanchard Fabrycky, *"System Engineering and Analysis"*, 5th edition, 2014, ISBN 10: 1-292-02597-2.

[6] Nelson K. Akafuah, Sadegh Poozesh, *"Evolution of the automotive body coating process-a review"*, College of Engineering, University of Kentucky, USA, Toyota Motor Engineering & Manufacturing North America, Inc., *Coating* 2016.

[7] A.Heutink, A.van Beek, J.M.van Noortwijk, *"Environment-friendly maintenance of protective paint system at lowest costs"*, Ministry of Transport, Public Works and Water Management Civil Engineering Division, HKV Consultants, Delft University of Technology, XXVII FATIPEC Congress, 19-21, p.351-364, April 2004

[8] Bruno Figliuzzi, Dominique Jeulin, Anael Lemaitre, *"Numerical simulation of thin paint film flow"*, Journal of Mathematics in Industry, doi:10.1186/2190-5983-2-1, 2012,2:1.

[9] M.H.Eres, *"Three-Dimensional Direct Numerical Simulation of Surface-Tension-Gradient Effects on the Leveling of an Evaporating Multicomponent Fluid"*, Department of Mechanical Engineering, University of Delaware, Newark, 1998.

[10] T. G. Myers, *"Thin Films with High Surface Tension"*, Society for Industrial and Applied Mathematics 001, SIAM REV. Vol.40, No.3, pp. 411-462, September 1998.

[11] Richard. R. Eley, Leonard. W. Schwartz, *"Interaction of Rheology, Geometry, and Process in Coating Flow"*, ICI Paints, University of Delaware, Technical Aricles, Vol.74, No.932, September, 2002.

[12] C. Pozrikidis, *"Fluid Dynamics Theory Computation and Numerical Simulation"*, University of California, San Diego, La jolly, California 92093-0411 U.S.A, p 412-p 413, Chapter 9.

[13] Howison, *Lubrication theory for fluids*, p 265-p 280, Chapter 20.

[14] J. B. Sweeney, T. Davis, L. E. Scriven, *"Equilibrium Thin Films on Rough Surfaces, Capillary and Disjoining Effects"*, Department of Chemical Engineering and Material Science, University of Minnesota, Department of Chemical Engineering, University of California at Santa Barbara, 1993.

[15] Eduardo Nunes, *"Inter-relationship of skin pass, 2D and 3D roughness parameters, stampability and paintability on cold rolled steel sheets for the automotive industry"*, Thesis, Metallurgical and Material Engineering, Universidade de Sao Paulo, 2014.

[16] *"Livelink for Matlab User's Guide"*, COMSOL Multiphysics, Version October 2014, COMSOL 5.0.

[17] Shanti Vusirikala, *"CFD simulation of contact planarization"*, Master thesis, Chemical Engineering, University of Missouri-Rolla, 2007.

[18] Bazr Afshan Fadafan, M.de Rooij, M.B.Valefi, *"Comparison of Lennard-Jones interaction and Maugis-Dugdale models for adhesion for the adhesive contact analysis for a bisinusoidal interface"*, 17th Nordic Symposium on Tribology-Nordtrib, pp. 1-7, Aulanko, Finland, 2016.

[19] Duysters, *"State of the art in surface texture synthesizing"*, TATA steel gatekeeper report, The Netherlands, 2015.

[20] Bharat Bhushan, *"Surface Roughness Analysis and Measurement Techniques"*, The Ohio State University, Chapter 2, 2001.

[21] Matthias Schneider, Christian Hager, *"Causes for the formation of surface structure on paint films"*, Department of Coating Systems and Painting Technology, Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Stuttgart, Germany, BYK-Gardner User Conference, 16-17 April 2013, Innsbruck, Austria.

[22] J. A. Ogilvy and J. R. Foster, *"Rough surfaces: gaussian or exponential statistics?"*, Theoretical Physics Division, Harwell Laboratory, Didcot, Oxon OX11 ORA, UK, J. Phys. D: Appl. Phys, pp.1243-1251, 1989.

[23] Dominik Kern, Nils-Henning Framke, *"Automation of COMSOL multiphysics parameter studies using the MATLAB livelink"*, TU Chemnitz, Chemnitz, Germany, September 26, 2012.