

THE FLORIDA STATE UNIVERSITY
COLLEGE OF ENGINEERING

DEVELOPMENT OF INTEGRATED PROCESS DESIGN ENVIRONMENT
AND STATISTICAL ANALYSIS OF RTM PROCESS

By

JING LI

A Thesis submitted to the
Department of Industrial & Manufacturing Engineering
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Fall Semester, 2003

The members of the Committee approve the thesis of Jing Li defended on Oct 23, 2003.

Chuck Zhang
Professor Directing Thesis

Okenwa Okoli
Committee Member

Zhiyong Liang
Committee Member

Approved:

Ben Wang, Chair, Department of Industrial & Manufacturing Engineering

Ching-Jen Chen, Dean, College of Engineering

The Office of Graduate Studies has verified and approved the above named committee members.

ACKNOWLEDGEMENTS

The author would like to thank his major advisor, Dr. Chuck Zhang. During this thesis research, Dr. Zhang not only gave to the author guidance and constructive suggestions but also encouraged him to touch some research areas not been done before. Other committee members, Dr. Zhiyong Liang and Dr. Okenwa Okoli provided helpful advice for this research work. Special thanks are extended to them as well. Recognition and appreciation also goes to Dr. Wang, Chair of Department of Industrial Engineering, for his directions in the graduate study.

In addition, the author would like to thank Mr. James Horne for preparing the experiments apparatuses, Mr. Chensong Dong, and Mr. Gaurav Garg for assisting the author to conduct the experiments.

Finally, the author would like to thank his parents who cultivated the author's strong desire for knowledge. Their unconditional love and support inspired the author whenever difficulty was met.

TABLE OF CONTENTS

| | |
|---|-----|
| LIST OF TABLES | vi |
| LIST OF FIGURES | vii |
| ABSTRACT | ix |
| | |
| INTRODUCTION | 1 |
| 1.1 RTM Process Design | 1 |
| 1.2 Problem Statement | 3 |
| 1.3 Research Objectives | 4 |
| | |
| LITERATURE REVIEW | 6 |
| 2.1 RTM Mold Filling Simulation | 6 |
| 2.2 Optimal RTM Process Design | 7 |
| 2.3 Permeability Measurement and Characterization | 9 |
| 2.3.1 Theoretical Background | 9 |
| 2.3.2 Permeability Measurement Methods | 11 |
| 2.3.3 Issues in Permeability Measurement | 16 |
| 2.4 Summary | 17 |
| | |
| METHODOLOGY | 19 |
| 3.1 System Development | 19 |
| 3.1.1 Concept of Integrated Design Environment for RTM | 19 |
| 3.1.2 Implementation of the Integrated Design Environment | 22 |
| 3.2 Material Characterization | 43 |
| 3.2.1 Permeability Measurement Experiment | 44 |
| 3.2.2 Statistical Analysis of Permeability | 49 |
| 3.3 Robust Tooling Design with Virtual Manufacturing | 65 |
| 3.3.1 Case Study | 70 |
| 3.3.2 Process Parameters Generation | 71 |
| 3.3.3 Robustness Analysis | 73 |
| 3.3.4 Results and Discussion | 74 |

| | |
|-----------------------------------|-----|
| CONCLUSIONS..... | 81 |
| 4.1 Contributions..... | 81 |
| 4.2 Extension of Research..... | 82 |
| APPENDICES | |
| APPENDIX A. MATLAB CODES..... | 83 |
| APPENDIX B. VISUAL C++ CODES..... | 86 |
| REFERENCES..... | 103 |
| BIOGRAPHICAL SKETCH..... | 108 |

LIST OF TABLES

| | |
|--|----|
| Table 3.1 Summary of part definition..... | 29 |
| Table 3.2 RTMSim solver summary..... | 36 |
| Table 3.3 Database operation routines summary..... | 42 |
| Table 3.4 Experiment parameters..... | 54 |
| Table 3.5 Summary of race-tracking permeabilities (m ²)..... | 55 |
| Table 3.6 Summary of ratios..... | 56 |
| Table 3.7 Summary of parameters for fitted distributions..... | 61 |
| Table 3.8 Summary of statistics..... | 64 |
| Table 3.9 Summary of conclusion..... | 64 |
| Table 3.10 Deterministic parameters..... | 72 |
| Table 3.11 Results summary..... | 76 |
| Table 3.12 Summary of the optimal tooling design..... | 79 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1.1 Illustration of key components for RTM product design | 2 |
| Figure 2.1 Illustration of flow through porous medium | 10 |
| Figure 2.2 Mold design for one-dimensional permeability measurement | 12 |
| Figure 2.3 Illustration of flow front for one-dimensional permeability measurement | 13 |
| Figure 2.4 Mold design for two-dimensional permeability measurement..... | 14 |
| Figure 2.5 Illustration of flow front for two-dimensional permeability measurement | 15 |
| Figure 2.6 Mold design for out-of-plane permeability measurement | 15 |
| Figure 2.7 Illustration of flow front for out-of-plane permeability measurement | 16 |
| Figure 3.1 Framework of 3-tier client\server architecture | 21 |
| Figure 3.2 3-tier architecture of RTMSim | 23 |
| Figure 3.3 GUI of RTMSim..... | 25 |
| Figure 3.4 Client GUI: boundary condition definition | 26 |
| Figure 3.5 Client GUI: material selection..... | 27 |
| Figure 3.6 Client GUI: search engine for materials..... | 28 |
| Figure 3.7 Client GUI: search engine for parts..... | 28 |
| Figure 3.8 OpenGL programming pipeline | 31 |
| Figure 3.9 Wireframe model..... | 32 |
| Figure 3.10 Wireframe with hidden line model..... | 32 |
| Figure 3.11 Solid model..... | 33 |
| Figure 3.12 Illustration of linear interpolation algorithm | 34 |
| Figure 3.13 Pressure distribution | 35 |
| Figure 3.14 Filling time | 35 |
| Figure 3.15 Application: flow simulation analysis | 36 |
| Figure 3.16 Application: cost analysis..... | 37 |
| Figure 3.17 Part status review..... | 38 |
| Figure 3.18 Example of table..... | 39 |
| Figure 3.19 Example of field and datatypes | 40 |
| Figure 3.20 RTM database structure..... | 41 |
| Figure 3.21 GUI of Utility database | 41 |
| Figure 3.22 GUI of Weight database | 42 |
| Figure 3.23 Schematic of the one-dimensional permeability measurement setup..... | 45 |
| Figure 3.24 WR24-5X4 woven fiber | 46 |
| Figure 3.25 Mold design for one-dimensional permeability measurement | 46 |
| Figure 3.26 Pictures of the mold..... | 47 |
| Figure 3.27 Diagram of measured variables..... | 48 |
| Figure 3.28 Illustrations of in-plane and transverse permeabilities..... | 49 |
| Figure 3.29 Illustrations of edge effects | 50 |

| | |
|---|----|
| Figure 3.30 Illustrations of dry spot formation | 50 |
| Figure 3.31 Calculation of measured permeabilities | 51 |
| Figure 3.32 Histogram of measured permeabilities (m^2)..... | 52 |
| Figure 3.33 Procedures of race-tracking characterization | 53 |
| Figure 3.34 Meshed geometry including race-tracking | 53 |
| Figure 3.35 Flow process..... | 54 |
| Figure 3.36 Simulation result..... | 54 |
| Figure 3.37 Histogram of ratios of race-tracking permeabilities over average value..... | 57 |
| Figure 3.38 Scatter diagram for independence checking..... | 57 |
| Figure 3.39 Fitted gamma distribution..... | 62 |
| Figure 3.40 Fitted Weibull distribution | 62 |
| Figure 3.41 Fitted lognormal distribution..... | 63 |
| Figure 3.42 Illustration of proposed analysis method..... | 66 |
| Figure 3.43 Illustration of quality index | 67 |
| Figure 3.44 Illustration of improved quality index..... | 69 |
| Figure 3.45 3D model of designed part | 71 |
| Figure 3.46 Engineering drawing of designed part..... | 71 |
| Figure 3.47 FEM model of designed part | 72 |
| Figure 3.48 Histogram of generated k_r / k_a values | 73 |
| Figure 3.49 Possible locations of gates and vents..... | 74 |
| Figure 3.50 Dry spot formation ($q = 1.3$)..... | 75 |
| Figure 3.51 q^{all} plot..... | 76 |
| Figure 3.52 Plot of average q^{all} | 78 |
| Figure 3.53 Plot of variance q^{all} | 78 |
| Figure 3.54 Plot of range of q^{all} | 78 |
| Figure 3.55 Plot of average filling time..... | 78 |
| Figure 3.56 Robust tooling design..... | 79 |
| Figure 3.57 q^{all} plot of robust tooling design | 79 |
| Figure 3.58 Demonstration of a bad tooling design..... | 80 |

ABSTRACT

The resin transfer molding (RTM) process has been used in the composite industry for decades. However, several issues still exist and impede its wide applications. Some design tools for RTM parts have been developed but a more efficient design environment is lacking. Race-tracking is a common phenomenon that makes prediction in actual production difficult and makes current deterministic optimal tooling design unrepeatable.

This thesis integrates flow simulation and cost analysis modules together with database management system (DBMS) providing a prototype of the integrated design environment for RTM processes.

Preform permeability, especially race-tracking permeability that significantly affects not only simulated but also experimental results, was the factor being investigated. This thesis introduces a statistical approach utilizing statistically distributed variables to explain the race-tracking permeability values. One-dimensional flow experiments were conducted to obtain the permeability values. Three types of distribution (gamma distribution, Weibull distribution and lognormal distribution) were chosen as candidates. Experimental data were fitted for the three distributions. A goodness-of-fit test was performed to find the one that best describes the experimental data.

Taking into account the fact that the severe levels of race-tracking can be represented by statistically distributed variables, this thesis proposes an optimization approach to minimize the sensitivity of the mold design to uncertainty of race-tracking permeabilities by choosing the appropriate locations of gates and vents (robust tooling design). A sensitivity that indicates the process robustness was defined as objective and evaluated by RTMSim software both for 2D and 2.5D geometry. With the conclusion that the ratios of race-tracking permeability over average values can be described by Weibull distributed variables, a random number generator was employed to generate the input race-tracking permeability data for obtaining values of the objective. Locations of vents were determined via the assumption that vents should be assigned

at the locations where flow ends to avoid dry spot formation. Locations of gate were optimized from most possible locations.

CHAPTER I

INTRODUCTION

In recent years, composite materials have gained more and more attention in both commercial and industrial applications. Specifically, resin transfer molding (RTM) and its derivative processes, which were originally introduced in the mid 1940s, now appear uniquely capable of satisfying the demand of the automobile and aerospace industries for low-cost, complex structural parts. This thesis research develops an integrated environment, including a data base management system (DBMS) for product data management, RTM flow simulation analysis, visualization of results and cost analysis modules. For RTM parts manufacturing, existing simulation studies are deterministic in nature. The stochastic factors existing in manufacturing procedures need to be investigated. Through experiments and statistical analysis, this research work highlights the factors that significantly influence the RTM product quality and also determines how statistically distributed process parameters affect the RTM process. Finally, a case study demonstrates the robust tooling design concept.

1.1 RTM Process Design

Even though the RTM process is relatively new compared to traditional manufacturing processes, the general RTM product development process is similar to other products involving several key steps — product definition, geometry design, prototype production and process optimization. As the computer technology advances, to handle the related data more efficiently, a database management system (DBMS) is usually used to maintain product data and link these separated components to ensure that the data can arrive to different modules as required.

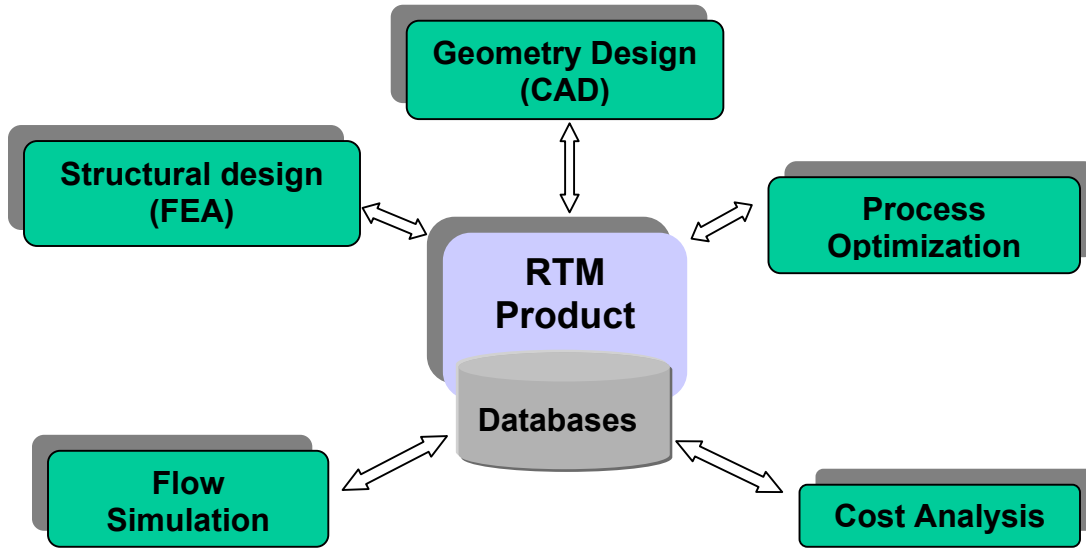


Figure 1.1 Illustration of key components for RTM product design

As shown in Figure 1.1, during processing and servicing, the activities involved in the RTM product development typically include, cost, static, dynamic and thermal analysis as well as geometry design. Despite the fact that integrated simulators are not new, the integrated simulator for the RTM process has not been widely used in the composite industry. Several RTM simulation software packages have been developed, such as LIMS, RTM-Worx, PAM-RTM [1-3]. All of these packages focus only on the flow simulation. None of them integrate other functions. In this RTMSim software system developed in this thesis work, the cost analysis developed by Dong *et al.* [4] was integrated and linked by a relational database, so that data flowed more efficiently within the program. A resin and fiber materials property database was developed and integrated in the RTMSim. Other functionalities may be integrated in future work.

Currently, most optimization work is reported to obtain the optimal operation setup in terms of the shortest filling time and minimizing injection pressure, as well as other objectives. In many cases, these optimized parameters cannot be accurately controlled in the manufacturing process due to inherent materials and process variations, which results in inconsistent part quality. Statistical distribution characteristics of the factors involved in the RTM process need to be investigated.

1.2 Problem Statement

The existing product data management (PDM) systems provide an effective tool to handle data. However, the PDM systems alone lack the capabilities of integrating CAD, CAE and CAM and are not adequate for effective product development. During the RTM product development process, the user must create multiple input files to manipulate the similar data with different programs. For geometry design, commercial software packages, such as Pro-E, AutoCAD, UG, and Solidworks are available. For CAE pre-process and post-processing, PATRAN and ANSYS are popular. Other auxiliary analysis tools, for example cost analysis, play an important role in product design. Although some studies have reported on integration with respect to CAD/CAM/CAE functions, they did not provide an adequate integrated environment for overall effective product design, which is also apparent in the realm of RTM product design. Still, with more and more product complexity, to reduce time to market and lower the cost of product development, simulation before manufacturing is becoming more popular. Simulation also allows products to be designed using an integrated method, taking advantage of available resources.

This research presents a computer-aided analysis system with a 3-tier client-server architecture consisting of several stand alone, task-specific computer programs that can meet the needs of integration of several components of RTM process design. When successfully developed, the system will result in reduced development time, cost saving, data security, improved product quality and faster response to the customer requirements.

In using this system, information flow is controlled and a minimum effort is required to run the application. At the server-tier, the crucial procedures of RTM product design, flow simulation and cost analysis are implemented, and the mechanism that acts as data bridge between client-tier and database-tier is achieved. At the database-tier, the component is a relational database, which provides an archive for RTM product related data. From the client-tier of the system, users can request their data from the DBMS, and place output data back into the database.

Considerable amount of work has been done on RTM processes modeling and optimization. This ranges from not only one-dimensional mold filling models but also three-dimensional models that simulate the mold filling, heat transfer and curing stages [5]. Spoerre *et al.* [6] utilized the genetic algorithms (GA) in conjunction with the cascade correlation neural

network architecture (CCA-NN) to build a model to predict and optimize performance and quality of RTM parts. Indeed, from simulation models, we can better understand the RTM process, and through process optimization, the factors that significantly affect the part quality are determined both quantitatively and qualitatively. However, in practice, variations of factor settings are inevitable that severely impedes the repeatability of optimal tooling design. Ranganathan *et al.* [7] reported that non-uniform raw material quality, improper preform preparation/loading, and mold assembling result in variations in preform microstructures and handling conditions, which often make the permeability largely different under the same theoretical circumstances. Other error sources in RTM processes may exist in the skill level of the operator, mold temperature or fiber material quality. Pan *et al.* [8] developed an experimental method to measure the fiber permeability and found that the probability distribution characteristics of perform permeability is actually normal distribution. Kang *et al.* [9] reported that in short fiber reinforced composite material, the distribution of the fibers is homogeneous and clustering of the fibers does not exist. Instead, the in-plane orientations of the fibers are almost distributed randomly as Weibull variables. Furthermore, the lengths of fiber are distributed normally. A sound assumption can be made that other factors have some kind of distribution functions as well.

In this thesis, a new data analysis technique for permeability measurement is proposed. The technique takes into account the combination effects of several factors in permeability measurement and analyzes the repeated experiment data statistically, i.e. permeability values for the same setting have some kind of distribution properties. A program generates the statistically distributed random variables that have the same statistical properties as experimental data for the processing parameters for a case study. After generating the stochastic permeability values, data are input into RTMSim, and simulations are run to obtain the filling times and flow patterns for the designed process settings. The process performance indices (Q's) are calculated using the simulation data and statistically analyzed to find the optimal tooling design from the statistical perspective. This approach will help designers achieve robust design of RTM processes.

1.3 Research Objectives

The goals of this thesis work are:

- Develop an integrated software system for life cycle development of RTM product. Accordingly, the components required in this system will be assessed and the relevant integration issues investigated. Finally, a development environment program will be established and the archive for the storage of RTM product data will be provided to allow connection between different server programs.
- Investigate the stochastic property of one of the key input parameters — permeability. Issues of measuring permeability will be discussed. Experimental validation will assess the proposed new data analysis technique.
- Introduce an optimal tooling design approach from a statistical perspective. The virtual experimental parameters will be simulated by a random number generator. An objective indicating the sensitivity of the process to uncertainty will be defined. Through a case study, robust tooling design will be performed and analyzed to find the optimal locations of gates and vents such that the optimized RTM process is insensitive to the statistically distributed race-tracking permeability values.

CHAPTER II

LITERATURE REVIEW

2.1 RTM Mold Filling Simulation

RTM has turned into an acronym to describe various resin transfer molding-type processes ranging from the traditional RTM to vacuum-assisted RTM (VARTM), resin film infusion (RFI), Seeman's Composite Resin Infusion Molding Process (SCRIMP) and other RTM variations [10]. No matter how complex these liquid composites molding (LCM) techniques are, they involve similar basic processes: mold filling and resin curing. The technical issues about how to achieve a good mold filling process make RTM process more complicated than it sounds. Since mold filling can be considered as the process of flow through porous media, which has a dominant effect on the final microstructure and overall quality of the composite parts, much research attention has been given to analyze, predict and simulate the behavior of resin flow inside the mold.

Starting with 1-D isotropic model, Gonzalez *et al.* and Chan *et al.* studied RTM in a disk-shaped mold and a rectangular mold, respectively [11, 12]. By neglecting the chemical reaction and heat transfer during the filling stage, both analytical and numerical methods were utilized to simulate flow process.

The porous media flow approach, i.e. applying Darcy's law, was used by many researchers [13 16] to model more complicated 2-D and 3-D flow problems. They took into account not only heat transfer but also curing and rheological changes for both isotropic and anisotropic scenarios. Experimental verifications of the current models also were performed to assess accuracy of the mathematical methods.

Corresponding to different mathematical models, which consist of a set of partial differential equations, several methods including finite different method (FDM), finite element

method (FEM) and boundary element method (BEM) were performed by researchers. FDM was the first attempt used to simulate a two-dimensional RTM process. By comparing with experimental results, it was proven that due to edge effects, the computing errors were over a reasonable range, which limited further application. Um *et al.* [16] applied the boundary element method. The case was two-dimensional flat molds where the permeability and the resin viscosity were constant. They reported that it took less time to generate mesh at each time step than required by FDM or FEM. Yoo *et al.* [17] and Osswald *et al.* [18] determined that under the limitations of simple geometry parts and isothermal Newtonian problems, the BEM method gave very accurate simulation results. Finite element method (FEM) and control volume (CV), i.e. FE/CV, are common methods that has been applied by some researchers [19-22]. Since the resin flow front advances, the calculation domain should be redefined and the numerical mesh should be regenerated, which results in a very time-consuming procedure. A major advantage of FE/CV is that simulating of the flow front can be carried out without re-meshing the filled node, although flow front changes continuously. Joshi *et al.* [22] concluded that three major steps are needed in the FE/CV flow simulation: (1) use the FE solution to obtain the pressure distribution in the resin-filled region; (2) calculate the resin flow rates; and (3) trace the resin flow front.

Some unique techniques emerged in RTM simulation research as well. Youssef *et al.* developed an interactive simulation technique in which during the simulation process, the user can: (1) change the locations of the inlet and vents; (2) remove, open and close inlet and vents; and (3) change the inlet pressure of flow rate at the inlets. Kanapady *et al.* [24, 25] developed a new, implicit, pure finite-element approach in symmetric multiprocessor machines for large-scale RTM process simulation which has been successfully applied to practical large-scale problems.

2.2 Optimal RTM Process Design

Mold filling is one of most important procedures in a typical RTM process cycle. The resin is injected and driven by the pressure to wet out the reinforcement preform. During this phase, many process factors are involved, such as location and size of gates and vent, injection pressure and mold temperature. Designing optimal RTM processes in terms of minimizing cycle time avoiding dry spots, and increasing the yield of successful parts has been done in this field.

Liu *et al* [15] implemented a mold filling simulation code to deal with the processing issues, including gate control, venting and dry spot formation. They used numerical sensors to design a control strategy to optimize mold filling time or reduce the required injection pressure. With specified vent location, they simulated the predicted changes in the pressure distribution and mold filling pattern. The simulation code checked if air was trapped in the mold and kept track of these trapped air pockets.

Lin *et al.* [26] discussed the strengths and weaknesses of the genetic algorithm and the gradient based algorithms. Two different types of RTM process optimization have been documented. In first case, the Quasi-Newtonian method was coupled in the code Global Local Optimizer (GLO), and gate locations were optimized to minimize the filling time. In the second case, a graphical search was explored for adding the varied high permeability layers to minimize resin waste in addition to minimizing the filling time. They reported that these two methods have their specialties, and if the design variables are discrete, for example number of gates and vents, the combination of two methods should be used. In addition, they also pointed out the limitation of finite element method used in analysis, i.e. the noticeable error was incurred if a single node was used to model the gate. Emphasis on adaptive meshing should obtain attention in future work.

Lawrence *et al.* [27] developed a design and control methodology. By taking advantage of sensor and actuators, the flow disturbance was identified and the resin flow was redirected to complete the mold filling without any void. They developed software for finding the position of the sensors in the mold to identify disturbances and suggest flow control actions for adding actuators at auxiliary locations to change the direction of flow. In their work, complex mold features, including tapered regions, rib structure, and thick regions, were tested to validate the effectiveness of the methodology. They documented that the feedback from sensors did have the ability to automate and actively control the flow of the resin, which led to consistently impregnating all the reinforcements even though disturbances were present in the process.

Jiang *et al.* [28] introduced a new mesh distance-based approach in genetic algorithm. The basic idea for this method was to find the optimum arrangement of gates and vents to achieve the objective of minimizing the maximum distance between gates and vents to avoid dry spot formation. By comparing with the examples in the literature, it was found that this method

was very efficient and effective in optimizing the locations of gates and vents and saving computation time.

2.3 Permeability Measurement and Characterization

Resin flow is governed by many process parameters, such as fiber reinforcement microscopic and macroscopic structure, resin chemistry and rheology, injection pressure, mold temperature, and mold complexity. All of these parameters make resin flow a critical part of RTM simulation. Among these parameters, permeability, the physical property of the fibrous material, which is measured by a mathematical model of Darcy's law, plays a crucial role in RTM simulation analysis. The filling time and flow pattern depend heavily on the preform property.

2.3.1 Theoretical Background

A porous medium is contained within a vessel, or some control volume consisting of pores between some particulate phase. The fluid flow rate through this vessel or control volume is $Q(m^3/s)$ and the cross sectional area is $A(m^2)$. Thus the superficial velocity U_0 is the total flow rate divided by the cross sectional area.

$$U_0 = \frac{Q}{A} \quad (2.1)$$

The particles existing within the vessel reduce the area available for fluid flow resulting in preserving fluid continuity with the entering superficial flow. Therefore, the fluid has to squeeze through a smaller area. This phenomenon makes velocity within the vessel greater than the superficial. The volume fraction has the most important effect compared to the mass fraction. The volume fraction of solids is usually referred to simply as the volume concentration or solids fraction, and the remaining fraction is that of the voids. The void fraction is also called the porosity. The dimensionless quantity porosity (ϕ) of a porous material is defined as the fraction of the vessel volume occupied by voids as shown in 2.2 and 2.3.

$$\phi = \frac{V_p}{V_c} = \frac{\text{Volume of Pores}}{\text{Vessel Volume}} \quad (2.2)$$

and

$$C + \varepsilon = 1 \quad (2.3)$$

where C is void fraction and ε is solid fraction.

The porosity is usually an isotropic property that means it is the same in all directions; therefore, the interstitial velocity is simply related to the superficial velocity by the expression 2.4, which comes from a consideration of fluid continuity. Figure 2.1 shows the relationship between superficial velocity and interstitial velocity.

$$U = \frac{U_0}{\varepsilon} \quad (2.4)$$

where U is interstitial velocity and U_0 is superficial velocity.

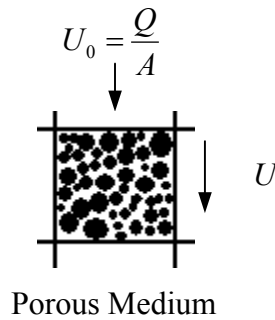


Figure 2.1 Illustration of flow through porous medium

The basic equation used to describe the flow behavior in porous media is Darcy's law (1856) [29], which states that the flow rate in a porous media is proportional to the pressure gradient in the medium. The constant of proportionality is defined as the permeability and the magnitude is a function of the pore structure (porosity or fiber volume content). The Darcy's law is valid as long as the following conditions are satisfied: (1) the flow should be a laminar flow that has a low Reynolds number; and (2) the fluid should behave according to the Newton's viscosity law.

$$\vec{v} = \frac{[k]}{\eta} \vec{\nabla} p \quad (2.5)$$

where: \vec{v} : Velocity of the flow front;

$[k]$: Permeability tensor;

η : Viscosity of the fluid;

$\vec{\nabla}p$: Pressure gradient.

From expression 2.5, permeability [k] characterizes how ease that a fluid goes through the porous material driven by an applied pressure gradient. The unit of permeability is a dimension of length scale squared and the most widely employed is the Darcy (D): one Darcy permeability, a pressure gradient of 1 atmosphere produces a flow rate of 1 cubic centimeter per second of a fluid with 1 centipoise viscosity through a 1cm^2 cross sectional area. Other units, such as m^2 , was widely found in literature as well.

$$1 \text{ darcy} = \frac{1(\text{cm}^3 / \text{sec}) \cdot 1(\text{cp})}{1(\text{cm}^2) \cdot 1(\text{atm} / \text{cm})} = 1\mu\text{m}^2 \quad (2.6)$$

2.3.2 Permeability Measurement Methods

Permeability is one of the most important factors governing resin flow through a composite preform, which makes it a critical input parameter for liquid composite molding manufacturing simulation and optimal tooling design. For most parts fabricated by RTM processes, the in-plane dimensions are noticeably greater than the thickness direction. Therefore, most research has focused on the in-plane, i.e. one-dimensional and two-dimensional flow experiments [29-37]. As the parts become more and more complex, large, thick components must be manufactured by RTM processes. Some attention has been drawn to three-dimensional flow experiments [38]. Three commonly used methods for one, two and three-dimensional flow experiments are as follows.

Unidirectional Flow Method

For one-dimensional flow, a rectangular cavity mold with an edge or line injection gate is a typical setup. Two techniques are widely applied, including saturated flow method and advancing flow front method.

For the saturated permeability measurement, one-dimensional Darcy's law is given as:

$$\frac{Q}{A} = -\frac{K}{\mu} \frac{dp}{dx} \quad (2.7)$$

where Q is flow rate; A is cross sectional area of the mold cavity; dp/dx is the pressure gradient across the length of the fabric; μ is the viscosity of the fluid, and K is an experimentally derived permeability constant. The permeability of the flow direction can be calculated by:

$$K = -\frac{Q \cdot \mu}{A \cdot \Delta p / L} \quad (2.8)$$

where L is length of the mold; $\frac{\Delta p}{L}$ is imposed pressure gradient on the flow direction.

For advancing flow front method, starting with the relationship between superficial velocity and interstitial velocity, the equation is stated as:

$$\frac{Q}{A} = q = V \cdot \frac{A_{flow}}{A_{total}} = V \cdot \phi = \frac{dx}{dt} \cdot \phi = -\frac{K}{\mu} \frac{dp}{dx} \quad (2.9)$$

where q is the superficial velocity, V is the interstitial velocity and ϕ is the porosity. So permeability can be calculated by the following equation:

$$\frac{dx}{dt} = \frac{K}{\phi \mu} \frac{(p_0)}{x} \Rightarrow \int_0^x x dx = \int_0^t \frac{K}{\phi \mu} p_0 dt \Rightarrow \frac{x^2}{2} = \frac{K}{\phi \mu} p_0 t$$

$$K = \frac{\phi \mu}{2 p_0 t} x^2 \quad (2.10)$$

where p_0 is the constant injection pressure, t is real filling time starting from the moment that test fluid begins to saturate the preform and x is the displacement corresponding to t .

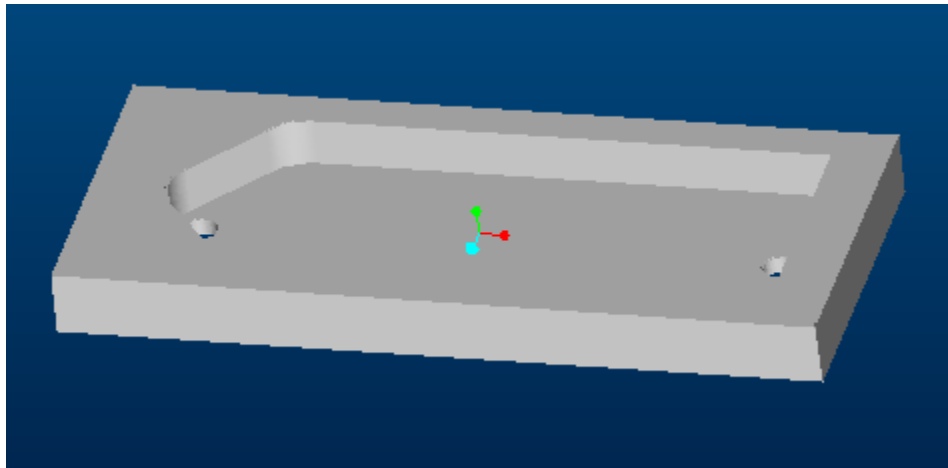
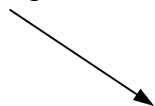


Figure 2.2 Mold design for one-dimensional flow experiment

Line injection gate



Flow front

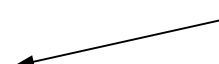




Figure 2.3 Illustration of flow front for one-dimensional flow

Bi-directional Flow Method

Since race-tracking is present in one-dimensional flow, researchers developed another experimental method: The test fluid was injected from the center of the mold and the analytical solution was solved based on Darcy's law. For isotropic case, the flow front advanced as a circular shape, and the following solution was proposed:

$$\left(\frac{R_f}{R_0}\right)^2 \left[2 \ln\left(\frac{R_f}{R_0}\right) - 1 \right] + 1 = \frac{4k\Delta p t}{\phi \mu R_0^2} \quad (2.11)$$

where: R_f : Flow front radius at time t ;

R_0 : Inlet radius, i.e. the radius of hold where the reinforcement stack is cut through at the center injection point;

Δp : Pressure gradient;

t : Elapsed time;

μ : Test fluid viscosity;

ϕ : Porosity;

k : Permeability ($k = k_x = k_y$).

For the anisotropic case, $k_x \neq k_y$ results in the flow front having an elliptical shape. Several simplification methods exist in the literature. The procedures for one of simplified solutions are shown, which transforms anisotropic permeability into an equivalent isotropic system, where Laplace's equation is utilized. A point in the original domain is transformed to a new coordinates system: $(x, y) \Rightarrow (x_e, y_e)$ by the relations:

$$x_e = \left(\frac{k_y}{k_x}\right)^{1/4} x \quad \text{and} \quad y_e = \left(\frac{k_x}{k_y}\right)^{1/4} y \quad (2.12)$$

In this equivalent isotropic system, the flow front is represented by a circular shape with $R = R_{xe} = R_{ye}$. Along with the relationship between x and y in original domain, the solution for permeabilities can be obtained by the following equations:

$$\left. \begin{aligned} k_e &= \sqrt{k_x k_y} \\ R_{x0,e} &= \left(\frac{k_y}{k_x} \right)^{1/4} R_0 \\ \left(\frac{R_x}{R_0} \right)^2 \left[2 \ln \left(\frac{R_x}{R_0} \right) - 1 \right] + 1 &= \frac{4k_e \Delta p t}{\phi \mu R_{x0,e}^2} \end{aligned} \right\} \quad (2.13)$$

where k_e and $R_{x0,e}$ are the transformed from there original coordinate system, respectively.

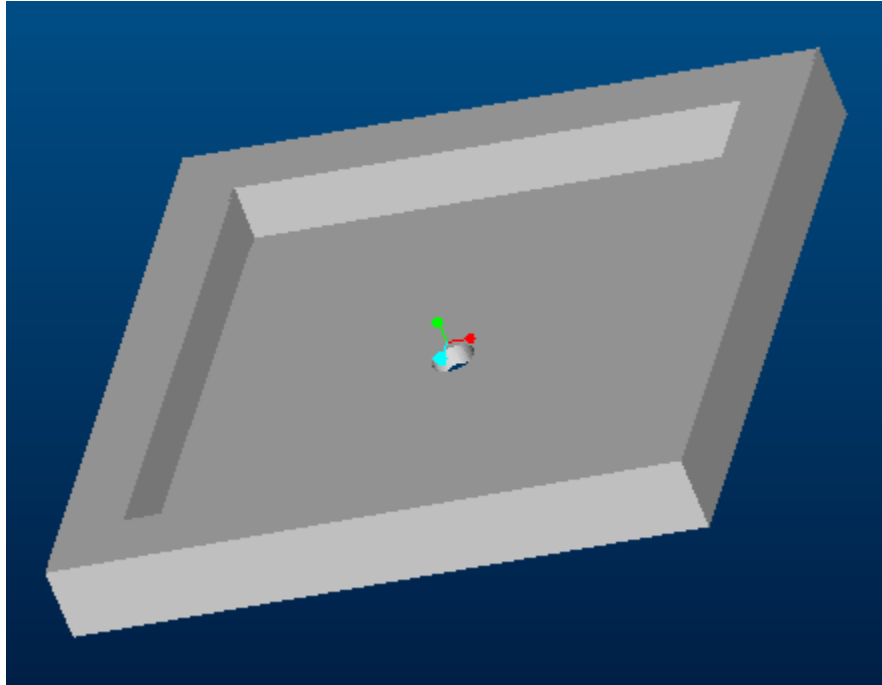


Figure 2.4 Mold design for two-dimensional flow experiment

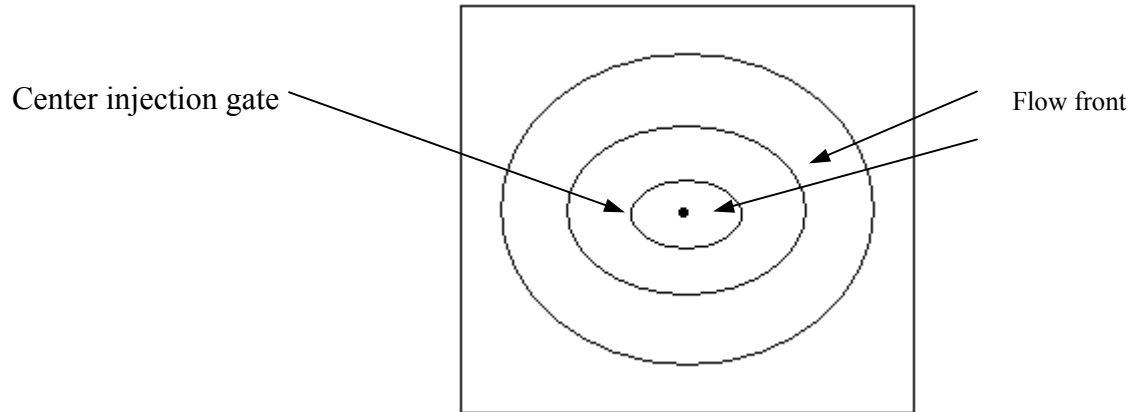


Figure 2.5 Illustration of flow front for two-dimensional flow experiment

Out-of-plane Flow Method

One-dimensional channel flow apparatus is the most common method to measure permeability in the out-of-plane direction. Darcy's law plays a governing role in computing the permeability as it does in one and two-dimensional flow.

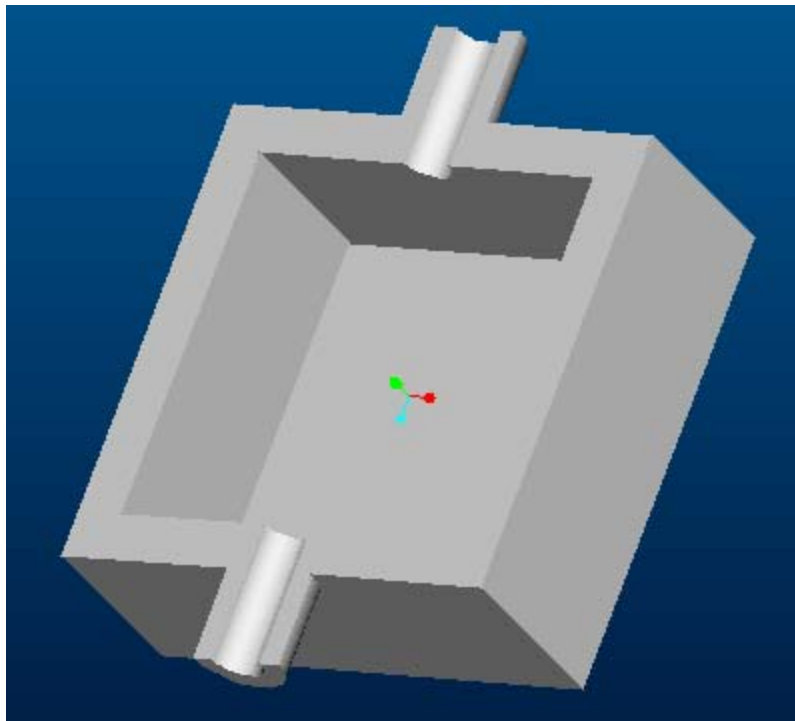


Figure 2.6 Mold design for out-of-plane flow experiment

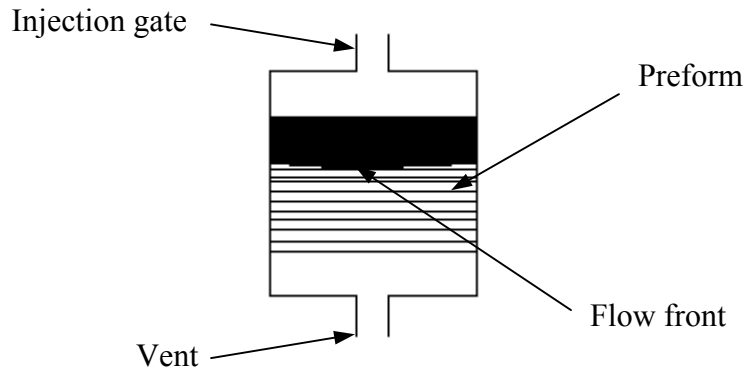


Figure 2.7 Illustration of flow front for out-of-plane flow

2.3.3 Issues in Permeability Measurement

As pointed out by Parnas *et al.* [35], even though permeability measurement is rather conceptually simple, three types of errors are likely to occur: (1) edge effect, (2) initial effect and (3) mold deflection.

In unidirectional flow experiments, achieving perfect fit of the test preform and mold edges is difficult, leading to the fluid leaking past the channel between preform and mold edges. Under this situation, the fluid flows faster along edges making the flow front a curvy shape instead of straight line. Some researchers have done extensively work on edge effect in permeability measurements. Hammami *et al.* [39] presented a model to describe the edge effect. The model derived using Navier-Stokes equations in the open channel and Darcy's law in the porous preform, simultaneously characterized the flow in the channel and through the reinforcement. They defined an equivalent porous media for which an equivalent permeability tensor can be computed as a function of the channel geometry.

$$k_{c,ave} = \frac{d^2}{12} \quad (2.14)$$

where $k_{c,ave}$: The equivalent average permeability of the channel;

d : Channel width.

This model was validated by their experiments in the case of neglecting the transverse flow.

S. Bickerton *et al.* [40] took into account a complex situation, studying the regions within the mold cavity where fiber free and high porosity exist, for example, near corners, bends, air-gaps and other features involving sharp curvatures. They demonstrated the role of flow channels in reducing the injection and mold pressures and redistributing the flow.

Due to the initial effect, unexpected air was trapped in the mold cavity during saturated flow experiments. The air expands and contracts resulting in non-Darcy behavior and measurement errors, which can be greater than 50%.

Transparent mold covers must be used in permeability measurements to monitor the flow front. Usually, the mold covers are lack the stiffness to resist the internal mold pressure, especially for a comparatively thin mold. This type of error has the similar mathematical magnitude as initial effect.

2.4 Summary

RTM process has drawn a considerable amount of attention from different perspectives, and several simulation software systems have been developed to simulate the RTM flow behavior, optimize the process and find the best tooling design. The difficulty of obtaining the accurate permeability of preform impedes the further development of RTM simulation packages.

For flow simulation analysis, the predicted results will be valid only if accurate permeability values for the actual process are supplied. Surface density variations, stacking sequences, compressing variations, race-tracking and even human errors significantly affect the permeability for the actual process. For example, to get the accurate permeability values, special care must be taken to avoid or minimize the edge effect or race-tracking; however, they are inevitable in actual manufacturing process. This reduces the repeatability of RTM process. Whenever the permeability changes for the same part, optimal set-up will also change. Researchers attempted to model the equivalent permeability for edge effect or rack-tracking, but how about the other regions? What is the exact permeability value for the regions where there is no edge effect. How much difference exists between permeability with edge effect and without edge effect? Existence of race-tracking increases the errors for permeability measurement, on the other hand, we do need race-tracking in real situations: the regions with race-tracking provides the low resistance during the course of resin flow and reduces the need of injection pressure and shortens the mold filling time.

In this thesis work, a simulation environment was developed and with the help of this simulation software, a new permeability data analysis technique and a new optimal RTM process design from statistical perspective are proposed.

CHAPTER III

METHODOLOGY

This thesis research involves three tasks: (1) development of an integrated RTM simulation software environment; (2) statistical characterization of permeability data; and (3) robust design of RTM processes with simulation and stochastic process parameters. Several software modules have been developed for RTM process design by previous researchers in the Department of Industrial Engineering at FAMU-FSU College of Engineering. This thesis research involved development of the integrated design environment with Microsoft Visual C++, OpenGL and Access database systems. To characterize permeability data, the commonly used one-dimensional method was used to measure the permeability of woven glass fabrics. The permeability data were characterized using statistical methods and the stochastic properties (distribution type and parameters). Based on the assumption that permeability values distribute statistically, optimal tooling design was simulated. In this research, robust tooling design was conducted so that the process was insensitive to the permeability variations for producing consistent part quality.

3.1 System Development

The activities involved in the RTM product development typically include geometric design; static, dynamic and thermal analysis during processing and servicing; process optimization and simulation; and cost analysis. To develop the system, all relevant issues, including the data requirement and possible system features were investigated. As these issues are resolved, a prototype for conceptual design environment became apparent.

3.1.1 Concept of Integrated Design Environment for RTM

Some researchers have attempted system integration. Klaas *et al.* [41] tried to embed numerical analysis capabilities into an enterprise-wide information system. They took into account many of product development tools that were developed independently and proposed an automated and adaptive geometric-based simulation processes associated into an engineering design process. This environment included a set of existing information management, computer aided design (CAD) modeling and computer aided engineering (CAE) analysis tools, which not only linked the existing tools together, but also provided additional technologies to improve the automation of the simulation processes and ensured accurate results. Commercial product data management (PDM) was developed to manage all the data related to a product, and workflow management system (WfMS) was used to coordinate and automate the execution of the processes.

Lu *et al.* [42] integrated a FEA code with pro/Engineer, a commercially available CAD system. They pointed out the difficulties in current practices and proposed an improved section-based FEA, which built an integrated design environment consisting of a CAD system and section analysis program along with other computer tools in sheet stamping process.

Several other researchers [43-46] did similar research on integration of CAD and CAE or utilized databases as connections and data storage for different applications. These integration methods occurred at multiple levels of detail. At the lowest level was the integration of data from one component or data repository to another. This transfer of data was accomplished either through direct integration of the components or through interface file transfers. At a higher level of detail, all of the functional components were integrated together in some manner such that the aim of developing and maintaining an integrated software system was achieved. These systems were modular and expandable.

A system is said to be modular if each component can be considered as a separate entity. In a modular system, each application might consist of a separate program or procedure block with calls to lower level procedures where required. Such a system would be easy to maintain because different components could be added or discarded with minimum disruption to the system.

Some integration models were available. In this work, a 3-tier client/server architecture was selected to obtain an integrated modular RTM design environment.

Client/server computing came about as a result of local-area-networks, leading to PCs coming out of their isolation. Servers that improve and maximize system performance have become more complicated and sophisticated, consisting of several branches with original file and database servers to large-scale computing application server. Currently, in a 2-tier architecture, only an architecturally tiered data server and client are still dominant. The main reason for widespread 2-tier model is due to the multiple applications and middleware that are commonly used since the 1990's, such as Remote-SQL or ODBC, as well as relatively inexpensive and easy-used PC development tools (visual basic, visual c++, Microsoft Access, etc.).

The 3-tier client/server architecture is a new trend of client/server system. The following diagram shows a simplified form of system architecture.

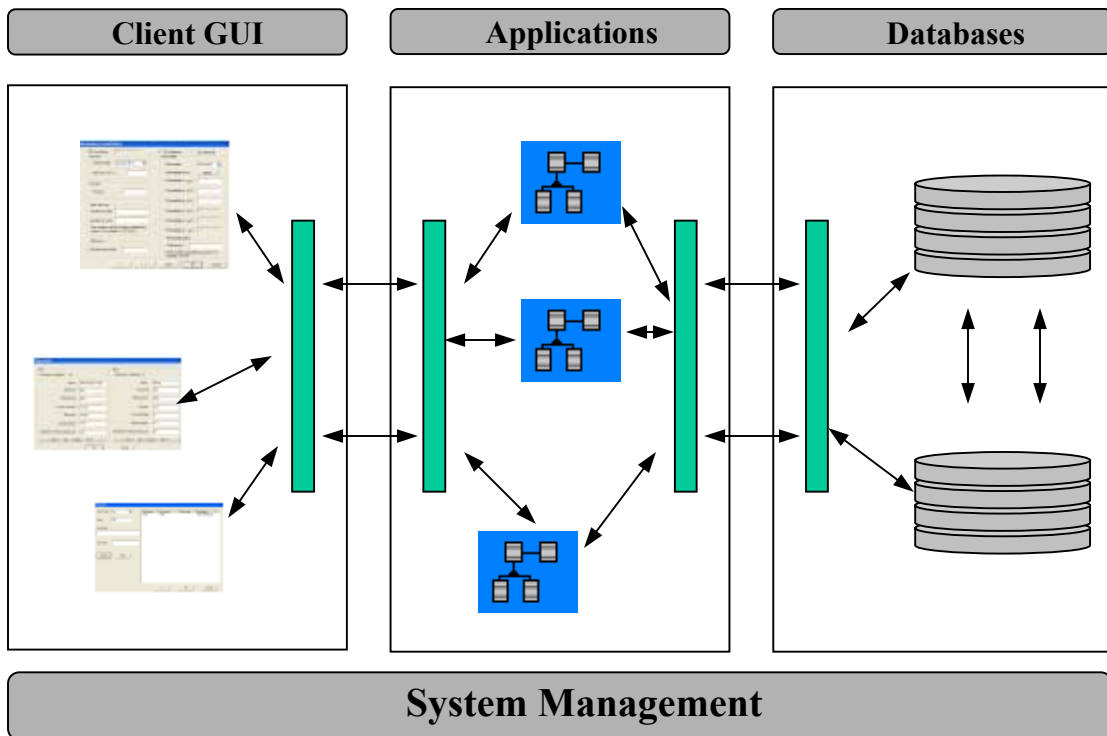


Figure 3.1 Framework of 3-tier client\server architecture

Client tier

The first tier or client tier is responsible for presenting data, receiving user events and controlling the user interface. The actual business application (e.g. calculating work) was moved to an application-server. Today, Java-applets offer an alternative to traditionally written PC-applications.

Application server tier

A new second tier is not present in a 2-tier architecture in this explicit form. Some functions originally belonging to the first tier of a 2-tier architecture now exist in the middle-tier. These middle-tier server application programs interact with clients by caching the data it extracts into a local object database for fast subsequent access and updating the third-tier data source directly with the newest results after the application is running.

The main purpose of this tier is to make the system modular. In this way, the term "component" means that different applications are linked and the system can add more components with less effort.

Data server tier

The third tier is responsible for data storage. In addition to the widespread relational database systems, existing legacy systems databases are also common. The boundaries between tiers are quite clear. The concept of 3-tier client/server can be implemented on the same machine or distributed computers. The main feature is that the system is neatly and conceptually structured by a well-planned definition of the software boundaries between different tiers.

3.1.2 Implementation of the Integrated Design Environment

In this thesis research, two key components in the integrated design environment were implemented: flow simulation and cost analysis. For flow simulation, users should provide the meshed geometry data file encompassing information about mesh nodes, elements and connectivity between nodes and elements. The client window displays the geometry file with three different models including wireframe, wireframe with hidden line and solid model. Geometry tools, like commercial CAD software, such as zoom in, zoom out, change view port, rotate and translate, are provided to manipulate the part. Boundary conditions definition is the most important step for a CAE analysis. The boundary conditions include reinforcement preform properties (permeability values, porosity and thickness), injection methods (constant pressure or constant flow) and tooling design (location of injection gates and vents). The next step is to

choose the materials that will be used for current part. After simulation, users should obtain the pressure distribution and flow patterns. The contour plots are provided by the visualization module. Another factor regarding RTM part is to lower the tooling cost as much as possible. Therefore, a cost analysis module, developed by Dong *et al.* [4], was modified to fit in the system. Basically, users provide some quantified key factors that influence the part development. The cost analysis module opens the cost expert database and fits user-provided data by the cost estimation solver. The rough cost can then be calculated.

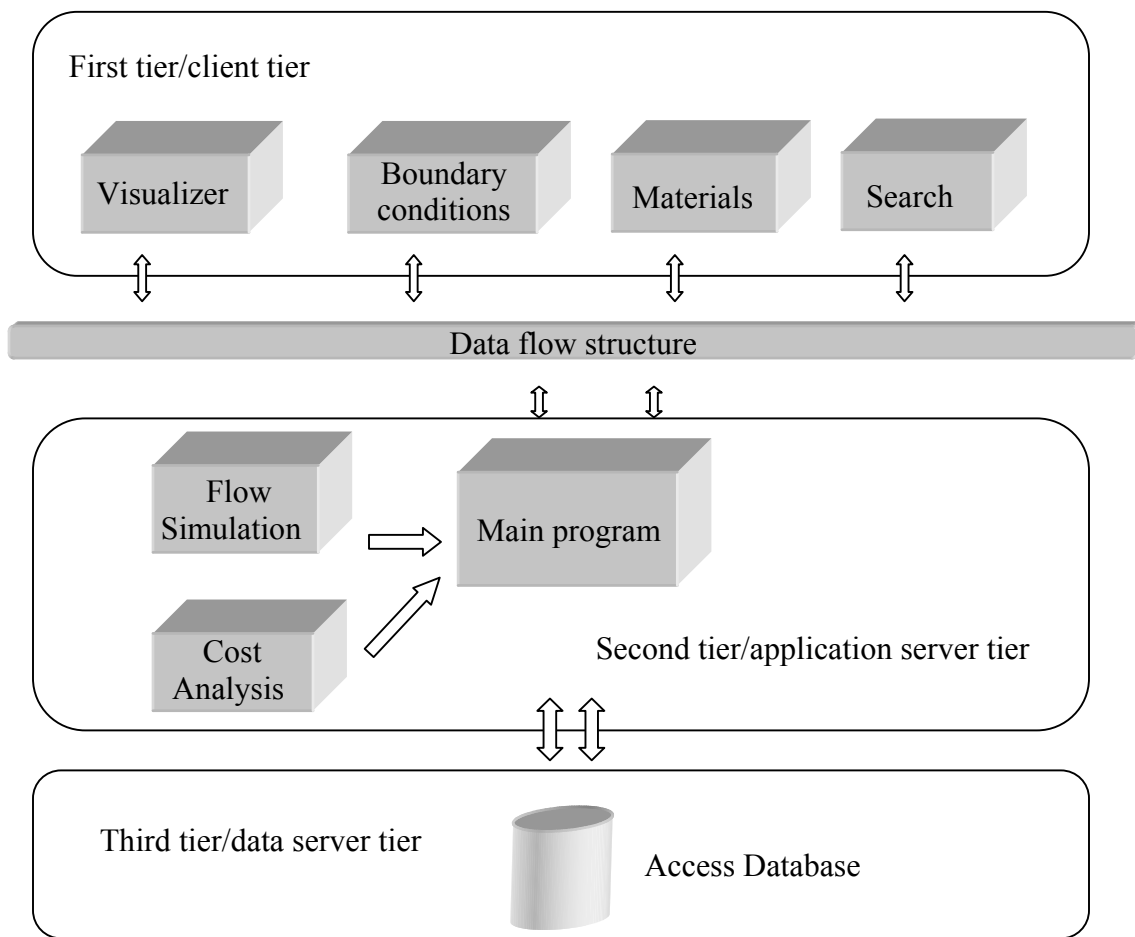


Figure 3.2 3-tier architecture of RTMSim

The above figure illustrates the detailed structure of the RTMSim software. The Microsoft Visual C++ 6.0 and the Microsoft Access database were the main tools for system development [47-49].

Microsoft Visual C++ is two complete Windows application development systems in one product (C and C++). Users can develop C-language Windows programs using only the Win32 API (Application Program Interface). This feature is one of the most useful benefits since the whole system coded in C++ can easily communicate with application servers coded in C. Many choices of programmable database management systems (DBMSs) are available, including Powersoft PowerBuilder, Borland Paradox, Microsoft Access and Microsoft FoxPro, as well as large-scale database server software, such as ORACLE or Microsoft SQL Server. Since Microsoft Access is a component of Microsoft Office, most users have Access, which is suitable for RTM product related data storage.

In this work, Microsoft Access database was chosen as the third tier connected through open database connectivity (ODBC). ODBC is based on a standardized version of structured query language (SQL). With ODBC and SQL, users have the ability to write database access codes that are independent of any database products.

A small top-level dynamic link library (DLL) ODBC32.DLL that defines the API makes the system fully modular. ODBC32.DLL loads database-specific DLLs, such as drivers, during the course of program running time. Taking advantage of Windows Registry maintained by the ODBC Administrator module in the Windows Control Panel, users can track database specific DLLs available and allow a single program to access data in several DBMSs simultaneously.

Figure 3.3 shows an example of the interface of RTMSim software. The client window is divided into several sections containing client tools for triggering the client tier data collection dialog boxes (boundary conditions definition, materials selection and parts data management) and application programs (files operation, flow analysis, cost analysis and results display). Users are also allowed to manipulate the geometry model, graph display window and the flow simulation results through geometry tools including zoom in, zoom out, translate and rotate. On the right side of the interface, users can choose the result they want to show from the list box.

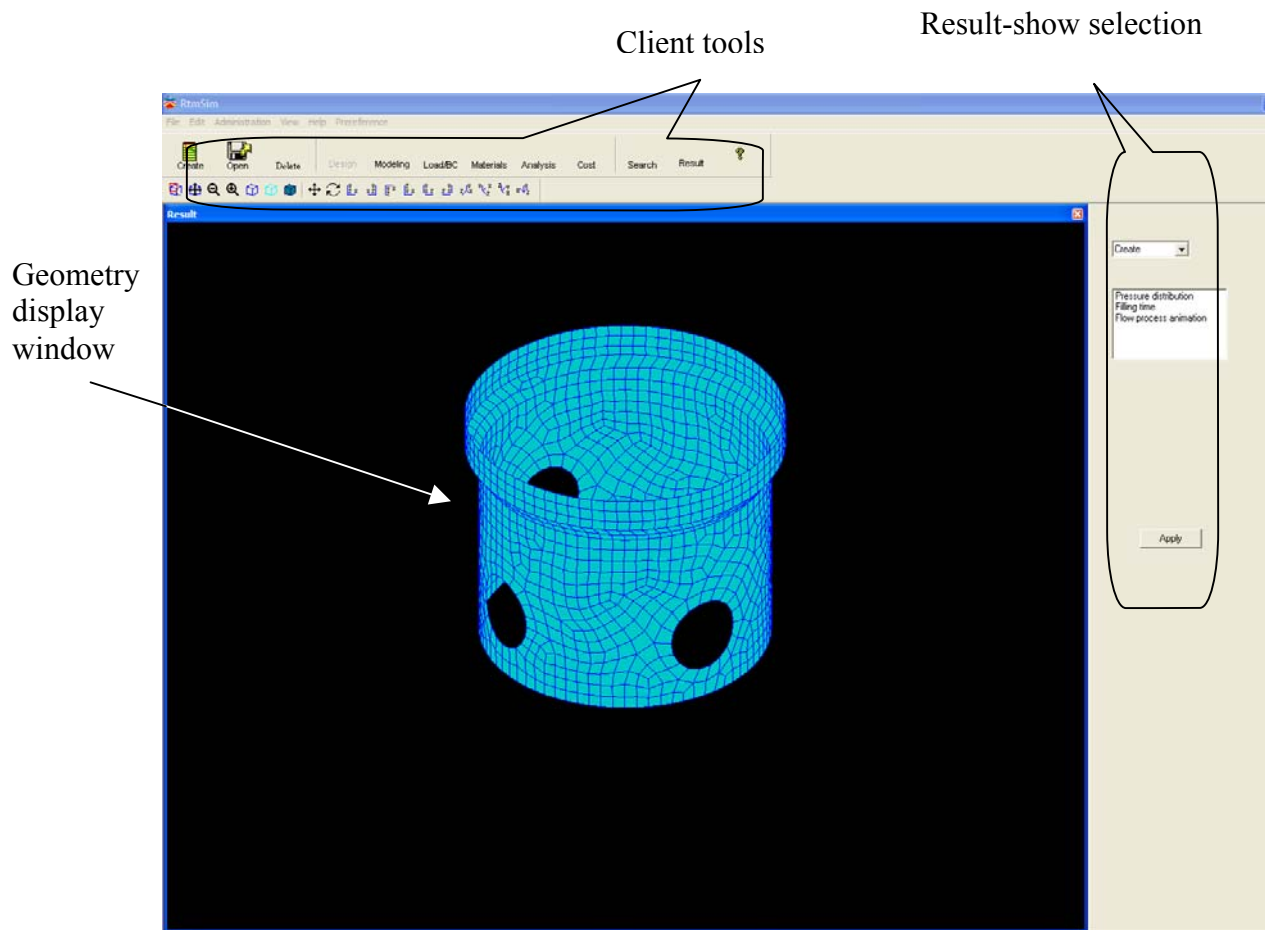


Figure 3.3 GUI of RTMSim

The following subsections describe the detailed functions of RTMSim software that are implemented through 3-tier client/server architecture approach.

Client tier

Since the first tier is responsible for presenting the data, receiving user events and controlling the user interface, it should have the capabilities to collect analysis related data, query related data, visualize the geometry model and show the results. RTMSim has four blocks: boundary conditions definition, material selections, search engine and visualizer.

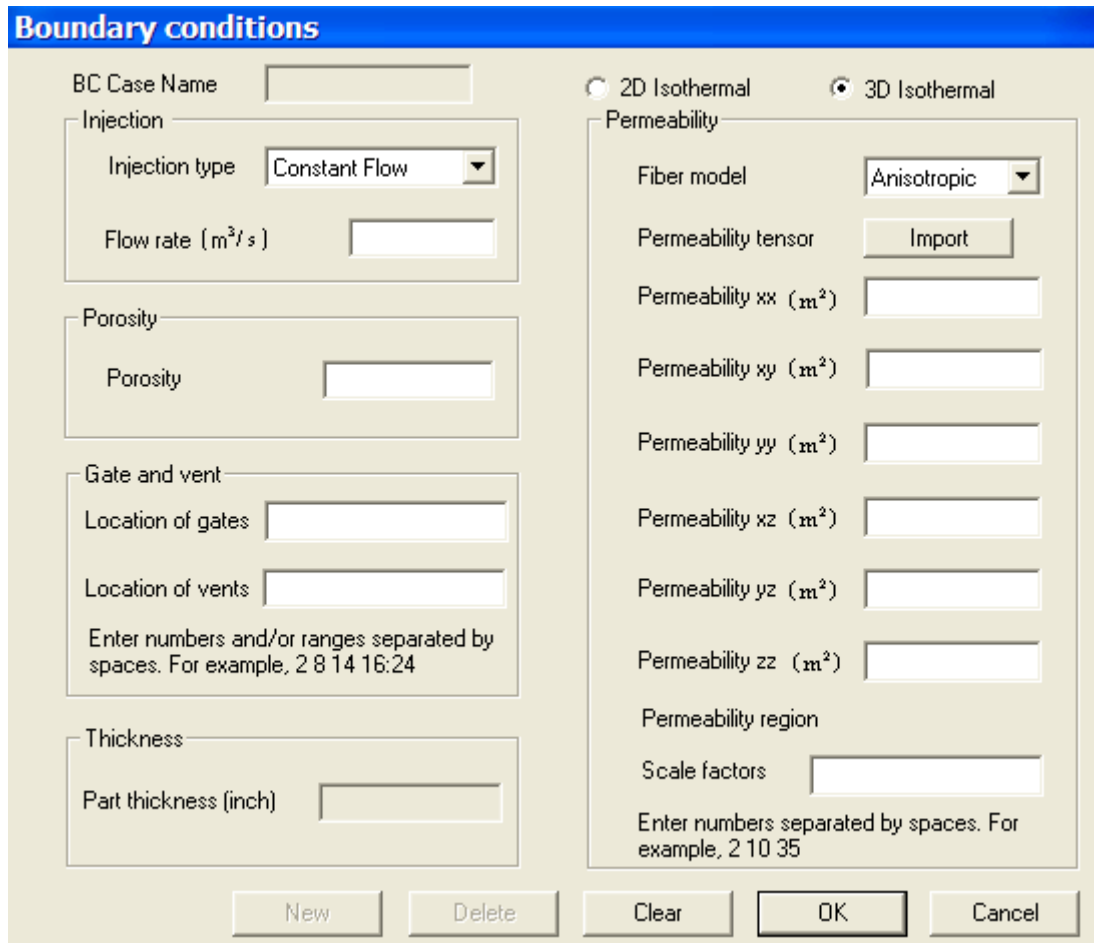


Figure 3.4 Client GUI: boundary condition definition

Boundary conditions collect the data needed in flow simulation analysis involving manufacturing parameters (choosing from constant flow or constant pressure injection for a given corresponding flow rate or injection pressure), tooling parameters (locations of injection gates and resin disperse vents), and reinforcement preform properties (2D or 3D models and their parameters, for example: thickness, permeability values). For the permeability definition, users can either input manually or import from a file that contains permeability data. After clicking the “OK” button, data collection is completed. The data input is stored into a boundary conditions database where it is available to the flow simulation analysis module.

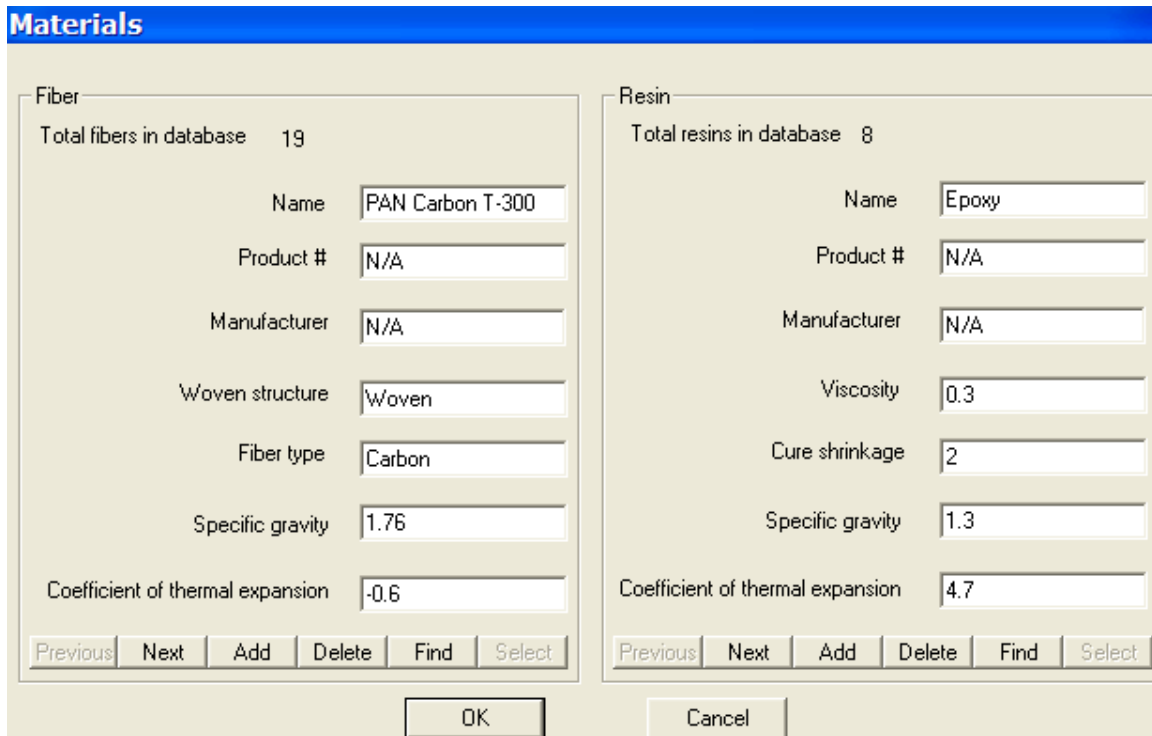


Figure 3.5 Client GUI: material selection

Because of the versatility and complexity of manufacturing process of RTM parts, simulations must be repeated numerous times for a single structure with different material configurations. As a result, a general material database will allow the user to take advantage of the database to obtain suitable materials faster. Actually, some researchers have attempted to build material databases [35, 46] in the composites manufacturing realm. In this module, users can browse and select the fibers and resins that they want for the current RTM part. They can also perform other operations, such as adding or deleting material records. After clicking “OK”, the selection is recorded for later use.

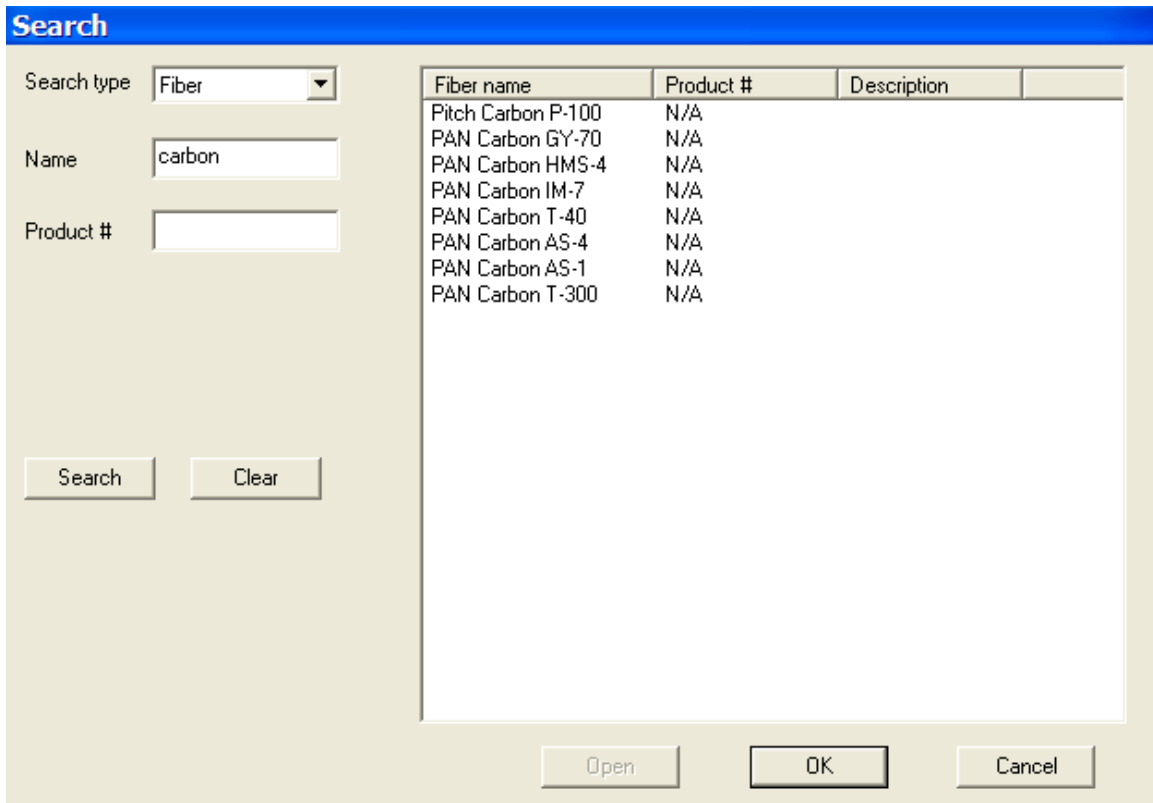


Figure 3.6 Client GUI: search engine for materials

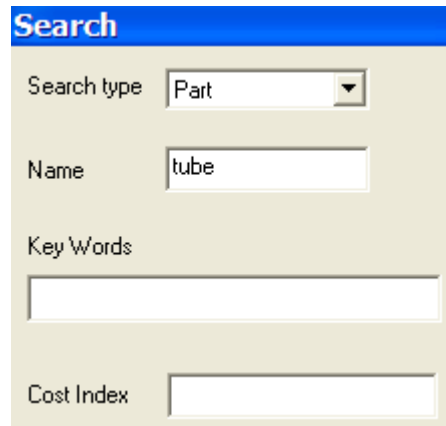


Figure 3.7 Client GUI: search engine for parts

Most of the part related data are stored in the database. Therefore, a powerful search engine is required to help users obtain information quickly and accurately. In the part search

engine, the user can search using any of the parameters or combining parameters together in the search dialogue box. If a field is left blank, all values for that field are displayed. The search parameters are as follows:

Table 3.1 Summary of part definition

| Search parameter | Note |
|------------------|--|
| Name | The names of parts, fibers and resins |
| Product # | Unique number for fibers and resins from manufacturer |
| Key Words | Created by designer from the beginning of part design |
| Cost Index | A real number between 0 and 1 indicating the part cost |

Visualizing the geometry data and displaying results requires a graphic library. Among the various choices available, such as ACIS, Open inventor, OpenGL etc, OpenGL was selected. OpenGL [50-52] is the premier environment for developing portable interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL also fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment. Main developer-driven advantages are list below [52]:

- Industry standard
An independent consortium, the OpenGL Architecture Review Board, guides the OpenGL specification. With broad industry support, OpenGL is the only truly open, vendor-neutral, multiplatform graphics standard.
- Stable
OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed

updates are announced in time for developers to adopt changes. Backward compatibility requirements ensure that existing applications do not become obsolete.

- Reliable and portable

All OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.

- Evolving

Because of its thorough and forward-looking design, OpenGL allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appear in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

- Scalable

OpenGL API-based applications can run on systems ranging from consumer electronics to PCs, workstations and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.

- Easy to use

OpenGL is well structured with an intuitive design and logical commands. Efficient OpenGL routines typically result in applications with fewer lines of code than those that make up programs generated using other graphics libraries or packages. In addition, OpenGL drivers encapsulate information about the underlying hardware, freeing the application developer from having to design for specific hardware features.

- Well-documented

Numerous books have been published about OpenGL, and a great deal of sample code is readily available, making information about OpenGL inexpensive and easy to obtain.

The procedures of using OpenGL for this work is illustrated as follows:

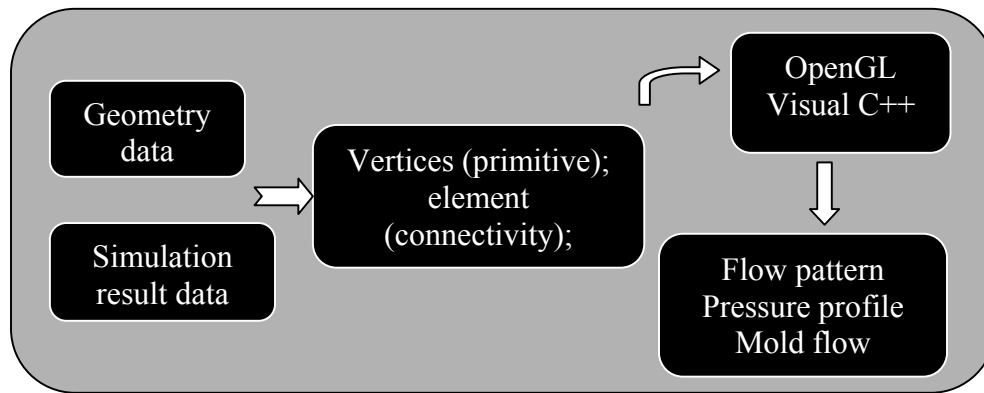


Figure 3.8 OpenGL programming pipeline

Users should provide meshed geometry file that includes coordinates of node and connectivity of element (element number and the nodes contained in this element). Current RTMSim software accepts PATRAN neutral export file. In this way, both 2D and 3D elements can be displayed. Segments of this type of files are shown below:

Node information:

```

.....
1  11  0  2  0  0  0  0  0
  5.794490337E+0 4.102369308E+0 -4.572124779E-1
1G  6  0  0 000000
1  12  0  2  0  0  0  0  0
  5.812500477E+0 7.597999096E+0 -2.269493962E-6
.....
  
```

Element information:

```

.....
2  29  4  2  0  0  0  0  0
  4  0  1  0 0.000000000E+00 0.000000000E+00 0.000000000E+00
  87 157 156 86
2  30  4  2  0  0  0  0  0
  4  0  1  0 0.000000000E+00 0.000000000E+00 0.000000000E+00
  88 158 157 87
.....
  
```

These data are then input into the main program that calls the OpenGL routines for visualization. Three models are currently provided: wireframe model, wireframe with hidden line model and solid model. The following figures show an example of the three display models.

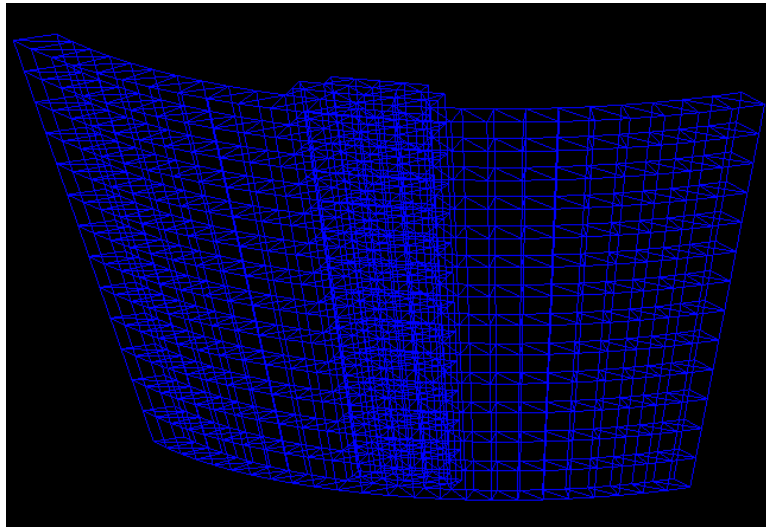


Figure 3.9 Wireframe model

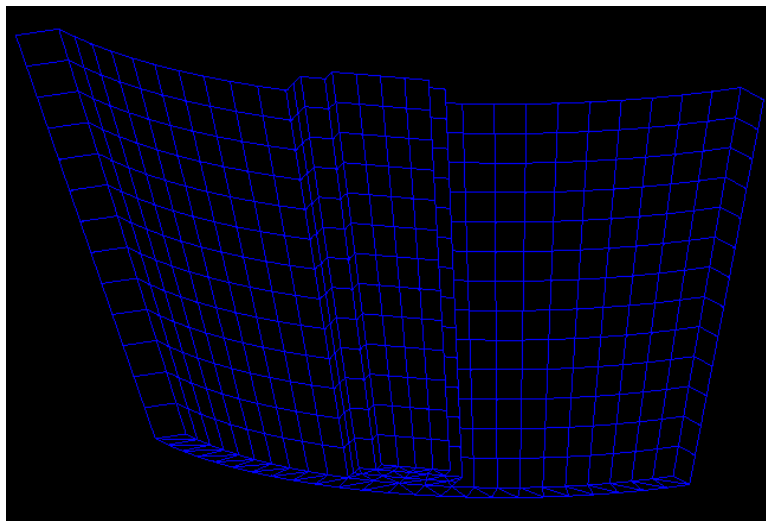


Figure 3.10 Wireframe with hidden line model

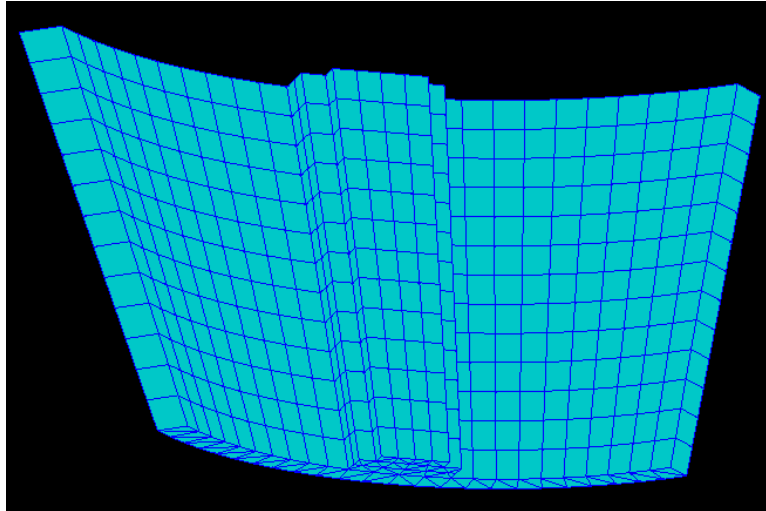


Figure 3.11 Solid model

In addition to geometry display and manipulation, the visualizer has the ability to interpolate the flow simulation data, i.e. computing contour line for the pressure distribution and flow pattern. Linear interpolation algorithm plays a key role in displaying the results.

A segment of the result file calculating by flow simulation codes is as follows:

.....

```
32.7613507e+066.9956898e+02  
42.5290284e+067.1538205e+02  
52.2967493e+067.3115408e+02  
63.2260489e+066.6785304e+02  
72.9936949e+066.8375893e+02  
82.7613573e+066.9956897e+02  
92.5290345e+067.1538203e+02  
102.2967549e+067.3115406e+02
```

.....

Number of node \swarrow 82.7613573e+066.9956897e+02
 └───┬───┘ └───┬───┘
 Pressure Filling time

where first two columns are used for number of node followed by pressure and filling time values.

To explain these node-based data, the contour lines have to be found. The contouring algorithm is given:

1. Supposing 15 different colors are needed, give all the nodes a unique color index according to their values:

$$\text{color index} = (\text{int}) (\text{current node value}/\text{max value}) \times 15$$

2. Starting from element 1, according to the connectivity, compare the color index between two nodes. If they are the same, then compare the next two nodes. If not, then do linear interpolation, until all the points needed are found.

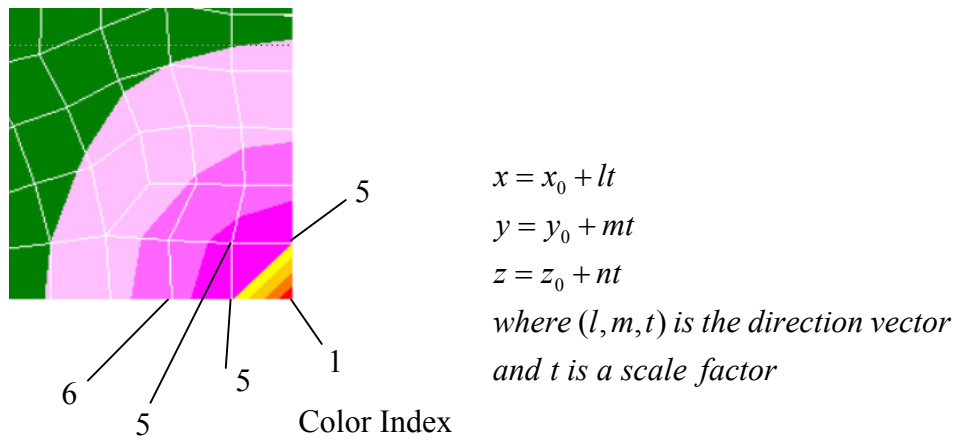


Figure 3.12 Illustration of linear interpolation algorithm

3. Regroup the elements based on the newly calculated nodes
4. Repeat Step 2 and 3 for all elements.

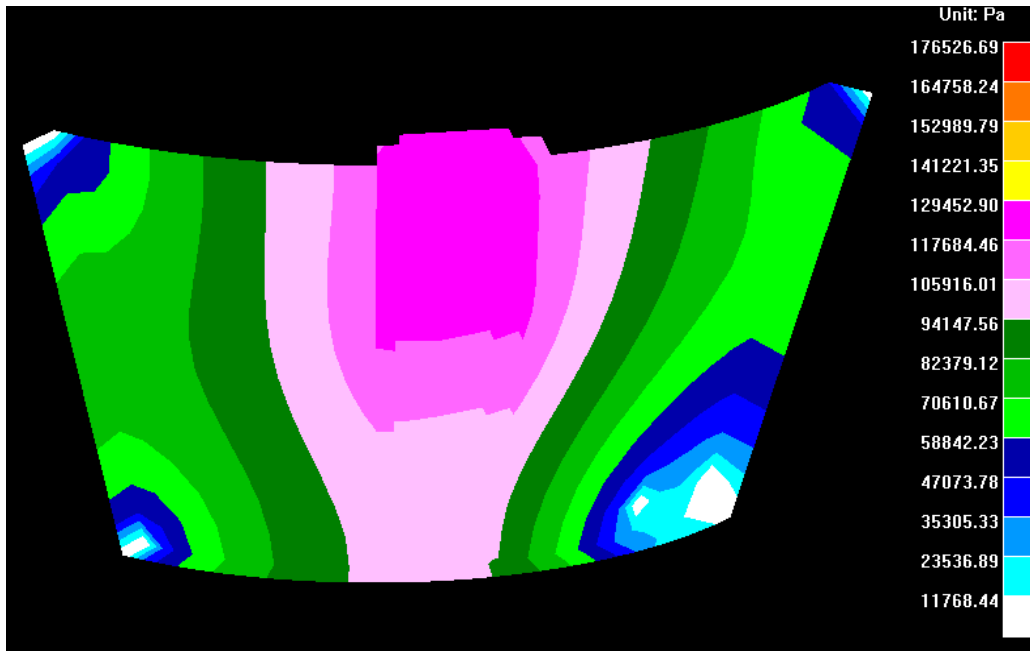


Figure 3.13 Pressure distribution

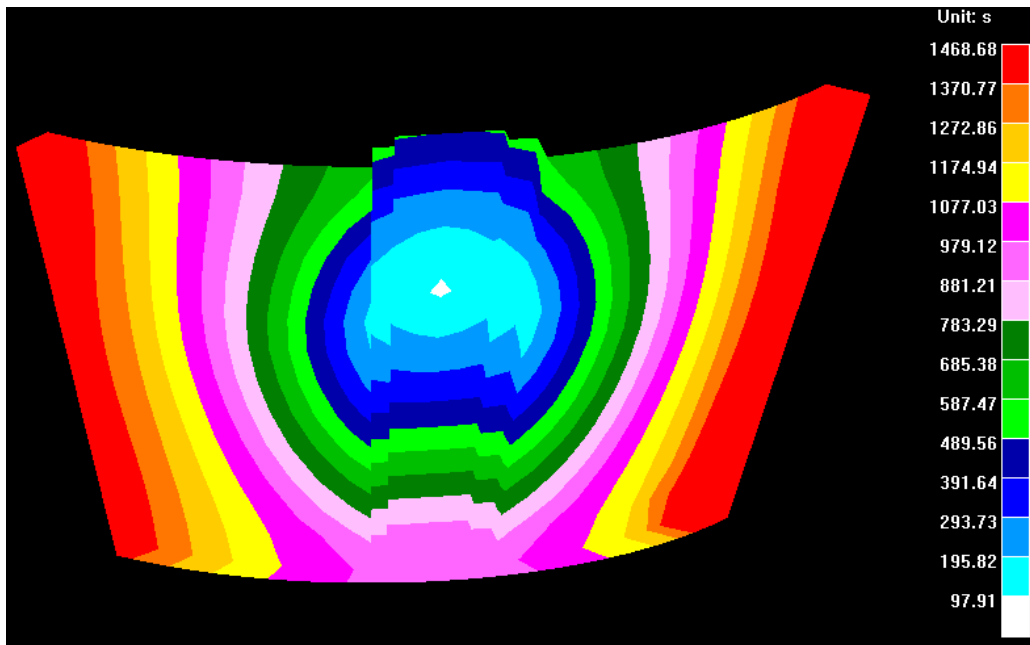


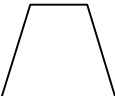
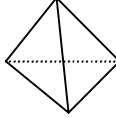

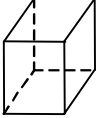


Figure 3.14 Filling time

Application server tier

The functions implemented in this tier include flow simulation, cost estimation, receiving events, accessing the database, and managing system. RTMSim flow simulation solver is a set of C-language codes based on control volume FEM method for simulating 2D, 2.5D (ignores the thickness) and 3D RTM flow problems. Most of commonly used element types are accepted by RTMSim. The key capabilities are shown in the following table:

Table 3.2 RTMSim solver summary

| Dimension | 1 | 2 and 2.5 | | 3 | | |
|------------------|---|---|---|--|---|---|
| Type of element | Beam | Triangular | Quadrangular | Tetrahedron | Wedge | Hexahedron |
| Shape of element |  |  |  |  |  |  |

After defining the input parameters, users can perform the simulation analysis application. During the computing process, the total number of nodes and the number of nodes that have been finished are shown in the GUI.

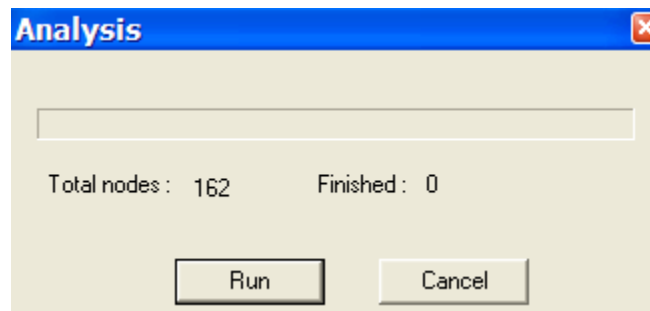


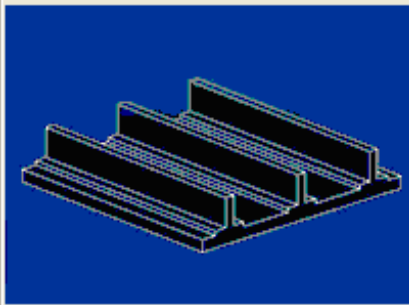
Figure 3.15 Application: flow simulation analysis

Cost analysis is another module included in the RTMSim software. With the help of a brief description, users can select the 13 factor levels for the part. Then a C-language based cost analysis codes, combined with expert database, is executed to estimate the cost of the part.

Cost

| | | | |
|----------------------|---|---------------------|---|
| Complexity | 2 | Curing Strategies | 4 |
| Quality Requirement | 1 | Fiber System | 3 |
| Fiber Volume Content | 2 | Resin System | 3 |
| Batch Size | 4 | Disposal Cost | 2 |
| Perform | 4 | Secondary Machining | 1 |
| Tooling Cost | 4 | Lightning Proof | 1 |
| Product Size | 3 | | |

Complexity



Flat shape with multiple cross stiffeners or ribs; Flat shape; Multiple cross stiffeners or ribs; Multiple inserts.

| | | | |
|-----------------------------|----------|----------------------------|---------|
| Cost Index: Central Gravity | 0.340159 | Cost (\$): Central Gravity | 3133.39 |
| Max Possibility | 0.340634 | Max Possibility | 3147.75 |

Expert Options Database

Utility Weight Regression Run OK Cancel

Figure 3.16 Application: cost analysis

Some other features, such as part status review, are provided. They can remind users about what has been done, what is still in process, and the parameter values in case they need to be redefined.

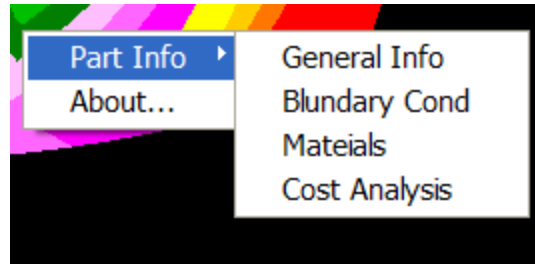


Figure 3.17 Part status review

Data server tier:

Several data models are widely used in different applications [47]. They can be classified into five general architectures: hierarchic or network (prerelational), relational, object-oriented or hybrid.

Two of these, hierarchic and network approaches, predate the relational approach. They provide users with a view of a structured database in which programs are written using a database language to search or navigate the data of interest to the application. They predefine access paths that are apparent at the application programming level. The main inconvenience is that whenever changes are required for production applications, the database should be changed accordingly.

The relational approach constructs a simple tabular data structure, i.e. more complex tree and network structures are replaced by simple tables. By programming language, referring multi-rows or multi-records becomes possible and searching the database is a function of the DBMS itself. This approach offers more powerful database languages and a higher degree of data independence than its predecessors. The object approach applies the concepts and techniques of object-oriented programming languages to the database environment. Users pay more attention to object, which is defined as data and code and inextricably connected to form objects.

By combining features and capabilities of both relational and the object-oriented approaches, the hybrid approach emerges as a more complex method. It supports query-based access (extended versions of SQL), as well as some forms of navigational access (typical object approach). This allows it to support nested tables and user-defined data type.

The data in RTMSim design environment include part general information (name, key words, description and locations of related files), boundary conditions (data needed in flow

simulation), material database (fibers and resins), and cost analysis related data. RTMSim is part-oriented, meaning that part-related data can be considered as the properties of the part, and different types of data do not interact each other. The data requirement, which determine the scheme of the system database, has several interrelated components, but they are independent each other. Therefore, a simple but reliable database will satisfy these requirements.

After reviewing the five types of database models, Microsoft Access relational database system was chosen for product related data storage. Microsoft Access uses a hierarchy in breaking down a database.



Database File: This is main file that encompasses the entire database and can be saved to hard-drive or floppy disk. For example: RTM.mdb

Table: A table is a collection of data about a specific topic. Multiple tables can be in a database. For example: (1) Partinfo; (2) Fiber.

| Boundarycond : Table | | | | | | |
|----------------------|-----|---------------|----------------|----------------|----------|--------------------------------|
| | Bou | Dimesion flag | Injection type | Flow rate | Porosity | Location of gates |
| ▶ | + | 1 | 2D Isothermal | Constant Flow | .000064 | .6 2 |
| | + | 2 | 3D Isothermal | Constant Flow | .000002 | .5 375 |
| | + | 3 | 2D Isothermal | Constant Flow | .000002 | .5 105 76 565 77 86 95 544 554 |
| | + | 4 | 3D Isothermal | Constant Press | 75000 | .5 2:66:8 8:72:8 |
| | + | 5 | 3D Isothermal | Constant Press | 75000 | .5 2 36 38 72 |

Figure 3.18 Example of table

Field: Fields are the different categories within a table. Tables usually contain multiple fields, for example: (1) Dimension flag; (2) Porosity.

Datatypes: Datatypes are the properties of each field. A field only has one datatype, for example: Field: Part Name; Datatype: Text.

| Boundarycond : Table | |
|----------------------|-----------|
| Field Name | Data Type |
| Boundary cond ID | Number |
| Dimesion flag | Text |
| Injection type | Text |
| Flow rate | Number |
| Porosity | Number |
| Location of gates | Text |
| Location of vents | Text |
| Part thickness | Number |
| Fiber model | Text |
| Permeability xx | Number |
| Permeability xy | Number |
| Permeability yy | Number |
| Permeability xz | Number |
| Permeability yz | Number |
| Permeability zz | Number |
| Scale factors | Text |

Figure 3.19 Example of field and datatypes

Another important concept is relationship. After setting up multiple tables in a Microsoft Access database, users need a way of telling Access how to bring that information back together. The first step in this process is to define relationships between the tables that help to create queries, forms and reports for displaying information from several tables at once. In this work, even though Visual C++ is the connection between applications and database instead of Microsoft Access itself, the relationships must still be defined. A relationship works by matching data in key fields — usually a field with the same name in both tables. In most cases, these matching fields are the primary key from one table, which provides a unique identifier for each record and a foreign key in the other table. For example, parts that can be associated with the boundary conditions are defined by creating a relationship between the Partinfo table and the Boundary conditions table using the Boundary cond ID field.

Two databases are involved in RTMSim software: RTM.mdb and Cost.mdb. In the RTM database, five tables were created for all product related data, including Partinfo, Boundarycond, Fiber, Resin and Costinfo. The Partinfo table is linked with the other four tables by a one-to-one relationship that means there is one and only one unique field corresponding each other, such as Boundary cond ID or FiberID. In this way, whenever a client application needs to modify required data, the data can be manipulated with minimal effort.

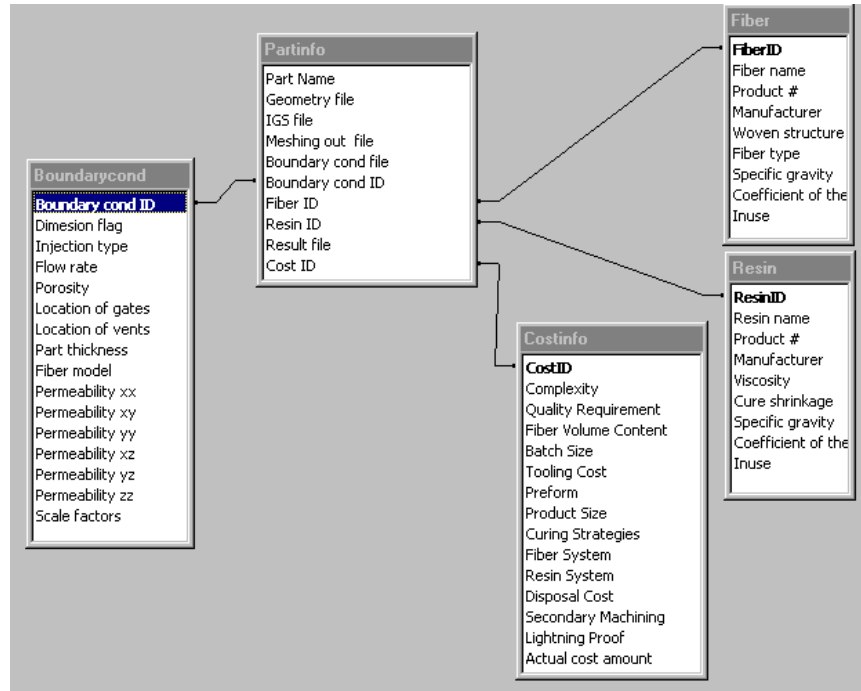


Figure 3.20 RTM database structure

The cost database is mainly used for cost analysis expert database divided into two portions: Utility database and Weight database.

| Utility | | | | | | | | | | | |
|---------|-------------------|---------|--------------|------------|--------------|------------|---------------|-----|------------------|-----|-------------|
| | Curing Strategies | | Fiber System | | Resin System | | Disposal Cost | | Second Machining | | Light Proof |
| | Complexity | Quality | Fiber VC | Product BS | Tooling Cost | Performing | Product Size | | | | |
| Level | OP1 | OP2 | OP3 | OP4 | OP5 | OP1 | OP2 | OP3 | OP4 | OP5 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 6 | 12 | 14 | 16 | 18 | 20 |
| 2 | 2 | 2.2 | 2.4 | 2.6 | 3 | 7 | 16 | 18 | 20 | 22 | 25 |
| 3 | 3 | 3.5 | 4 | 5 | 6 | 8 | 20 | 29 | 31 | 34 | 35 |
| 4 | 8 | 9 | 10 | 11 | 12 | 9 | 25 | 30 | 40 | 50 | 60 |
| 5 | 10 | 11 | 14 | 15 | 16 | 10 | 35 | 45 | 55 | 65 | 80 |

Figure 3.21 GUI of Utility database

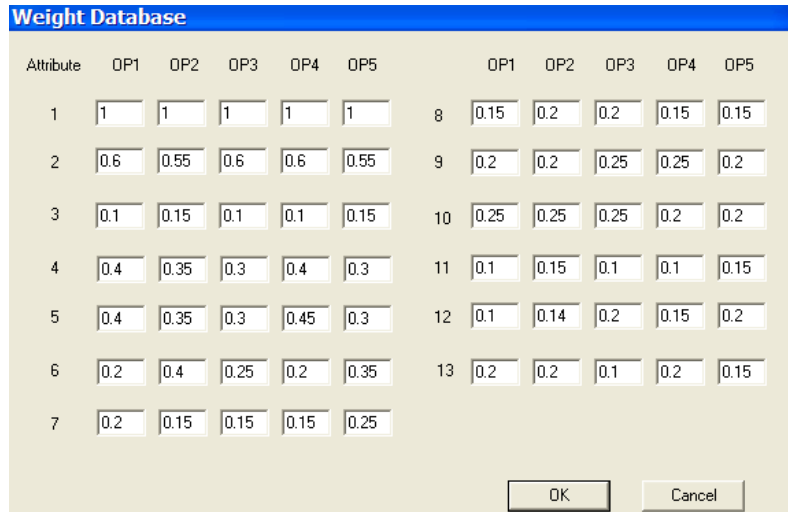


Figure 3.22 GUI of Weight database

As discussed before, to operate the database within the Visual C++ environment, ODBC was selected as the programming tool in RTMSim development. MFC ODBC Class *Crecordset* provides numerous member functions so that it can fully satisfy users' requirement for data record. These functions are summarized in Table 3.3[48].

Table 3.3 Database operation routines summary

| Function | Description |
|---------------|---|
| <i>Open</i> | Opens the recordset |
| <i>AddNew</i> | Prepares to add a new record to the table |
| <i>Update</i> | Completes an <i>AddNew</i> or <i>Edit</i> operation by saving the new or edited data in the data source |
| <i>Delete</i> | Deletes the current record from the record |
| <i>Edit</i> | Prepares to implement changes on the current record |
| <i>IsBOF</i> | Determines whether the recordset has been positioned before the first record |
| <i>IsEOF</i> | Determines whether the recordset has been positioned after the last record |

Table 3.3 continued

| Function | Description |
|--------------------------|---|
| <i>MoveNext</i> | Sets the current record to the next record or to the next rowset |
| <i>MoveFirst</i> | Sets the current record to the first record in the recordset |
| <i>MoveLast</i> | Sets the current record to the last record in the recordset |
| <i>Move prev</i> | Sets the current record to the previous record or to the previous rowset |
| <i>GetDefaultConnect</i> | Gets the default connect string for the data source on which the recordset is based |
| <i>GetDefaultSQL</i> | Gets the default SQL string |
| <i>DoFieldExchange</i> | Exchanges data between the recordset data field and the corresponding record on the data source |
| <i>GetStatus</i> | Gets the index of the current record in the recordset and the final count status |
| <i>GetrecordCount</i> | Determines the highest-numbered record yet encountered as the user moves through the records |
| <i>GetODBCFieldCount</i> | Determines the number of fields in the recordset object |
| <i>GetODBCFieldInfo</i> | Gets information about the fields in the recordset |

So far, a prototype of integrated RTM part design environment implemented by the 3-tier client/sever architecture has been achieved. Since the system is modular, other applications can be added with minimal disturbances to the whole system. The next step of this work is to analyze how to employ this software efficiently and effectively, such that more useful information could be derived from the process simulation.

3.2 Material Characterization

As discussed in the literature review, the RTM process has been gaining extensive attention for the complex structural parts made of fiber reinforced polymer matrix composites. Researchers have made significant progress in further understanding the process. Several

simulation tools have been developed and according to their reports, the predicted results matched well with experimental results. However, in the course of production process, some process parameters may change noticeably, which results in difficulty of maintaining consistent part quality.

Among these parameters, preform permeability, which significantly affects not only simulated but also experimental results, is a factor that must be investigated. Although the variation in preform permeability is inevitable, knowing the causes of this variation, the properties of the measurement errors and how the variation changes, will be helpful in RTM process development. Pan *et al.* [8] developed an automatic testing apparatus with automatic data acquisition and processing capabilities. Major factors, including preform edge protection, fiber weight control, complexity of mold shape and operator's skill level, were studied to determine how they affect the permeability values. They concluded that these factors, with the exception of operator skill level, made the permeability values diversified, and the detailed statistical analysis showed that varied permeability values can be treated as normally distributed variables. This was a new attempt to explain the permeability by statistical distribution. However, they did not consider other potential factors that may cause permeability variation, such as test liquid temperature or injection pressure variation. Another concern was that just assuming permeability values as normal variables seems inadequate. Other distributions may be more appropriate.

In the following section, repeated experiments were conducted under normal circumstance to simulate production conditions. Data analysis centered on the fiber preform permeability characterization from stochastic point of view was performed with different distribution assumptions.

3.2.1 Permeability Measurement Experiment

In this work, one-dimensional flow experiments were designed to study the statistical properties of permeability. The following equation was used to obtain the permeability along the direction of advancing flow front.

$$K = \frac{\phi\mu}{2p_0t}x^2 \quad (3.1)$$

where:

K : An experimentally derived permeability constant;

p_0 : The constant injection pressure;

μ : The viscosity of the fluid;

ϕ : The porosity of preform;

t : Real filling time starting from the moment that test fluid begins to saturate the preform;

x : The displacement corresponding to t .

The variables being measured are test fluid displacement and the filling time at each measuring moment. Other parameters should be kept as constant values. The apparatuses that satisfies above constrains are shown in Figure 3.23.

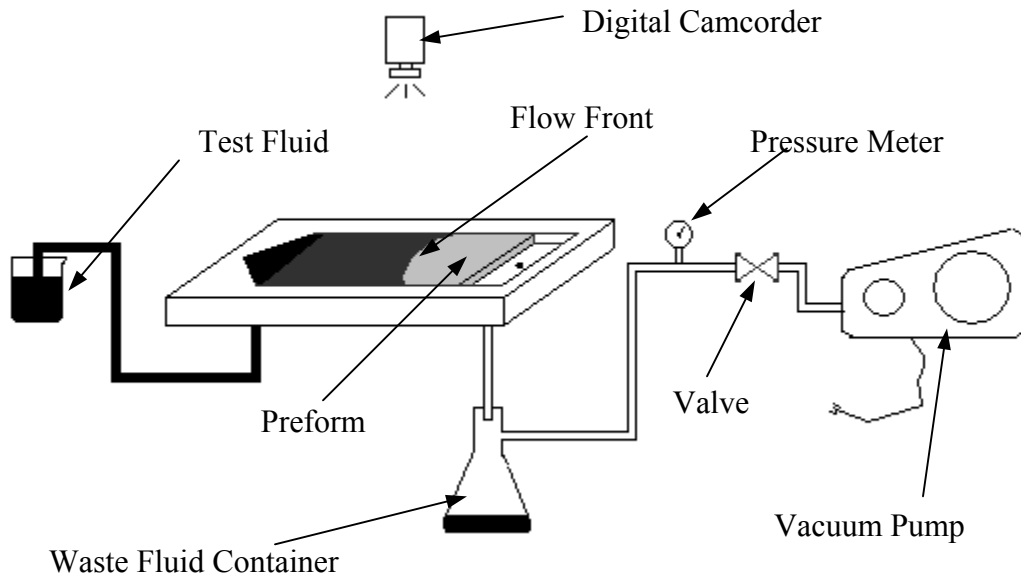


Figure 3.23 Schematic of the one-dimensional flow experiment setup

To maintain the constant pressure, a vacuum pump controlled by a valve instead of potentiometer was used, due to its simplicity of setup and control. WR24-5X4 woven fabrics (surface density = 810g/m^2 and fiber density = 2500kg/m^3) from Owens Corning Company were used as the test material for these experiments. The fabric is shown in Figure 3.24. The porosity

was also assumed as constant, even though the porosity was not exactly homogenous in the preform.

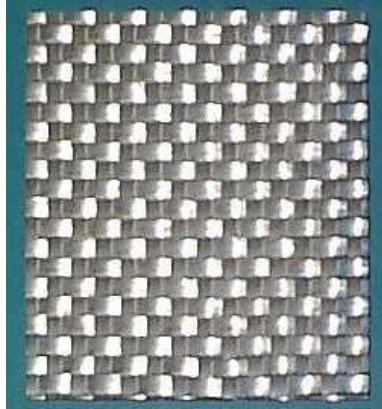


Figure 3.24 WR24-5X4 woven fiber

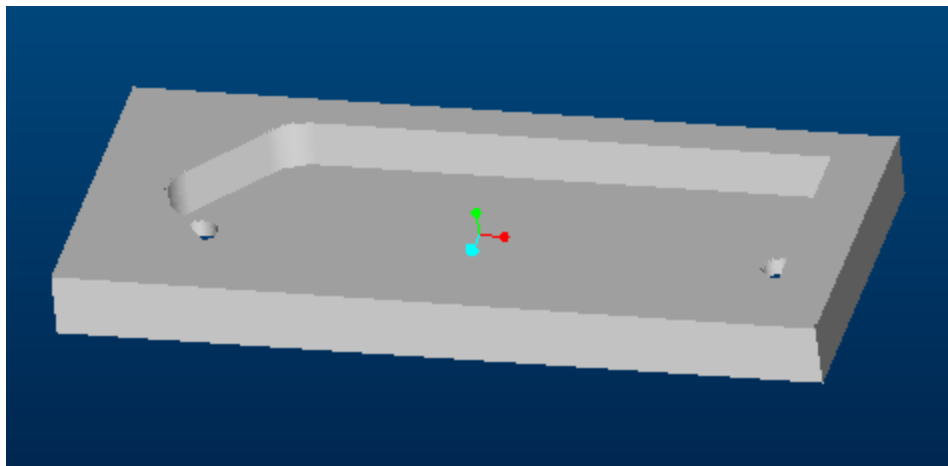


Figure 3.25 Mold design for one-dimensional flow experiment

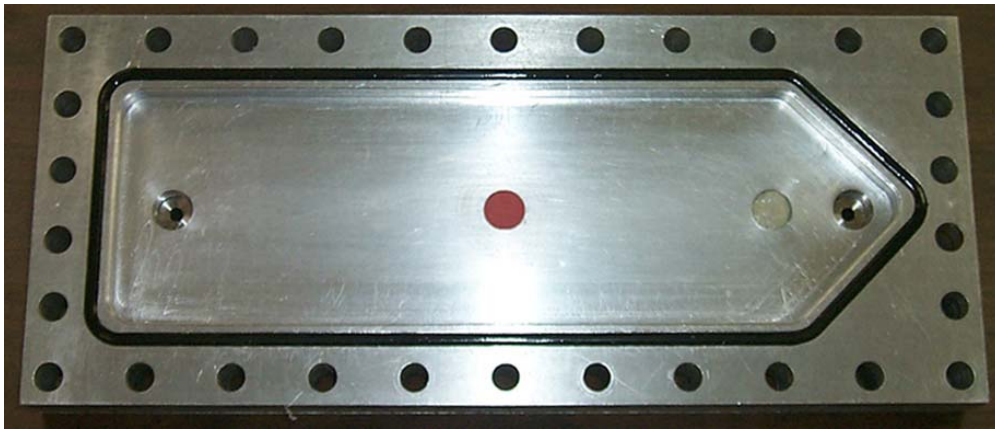


Figure 3.26 Pictures of the mold

In actual manufacturing process, liquid resins are injected into a mold cavity to obtain the reinforced part, but alternative test fluids, such as mineral oil and corn syrup are also acceptable since they are easy to handle and are chemically stable [53,55]. In Luo *et al.* and Hammond *et al.*'s experiments, even though there were different flow behaviors for different liquids, the measured permeability for both woven and knitted fibers did not exhibit significant differences. Steenmaker *et al.* [54] argued that they observed significant dependency of measured permeability on test liquids.

Adopting different test liquids only results in changes of mean values of measured permeability. As the study focuses on distribution properties, especially variability of the permeability, the selection of test liquids would not significantly affect the final conclusion for

the isothermal RTM process. Therefore, considering the advantages of alternative liquids, vegetable oil was chosen as test fluid, due to its merits mentioned above and its lower cost (1/10 of silicon oil). To ensure constant viscosity and avoid the influence of air bubbles in the oil, before each experiment, two important procedures were performed: (1) a vacuum expelled the air bubbles and (2) a viscosity meter measured the viscosity of the vegetable oil.

A digital camcorder was mounted on the top of the test frame to record the displacement and filling time for the advancing flow front. Figure 3.27 shows the variables measured in the experiments.

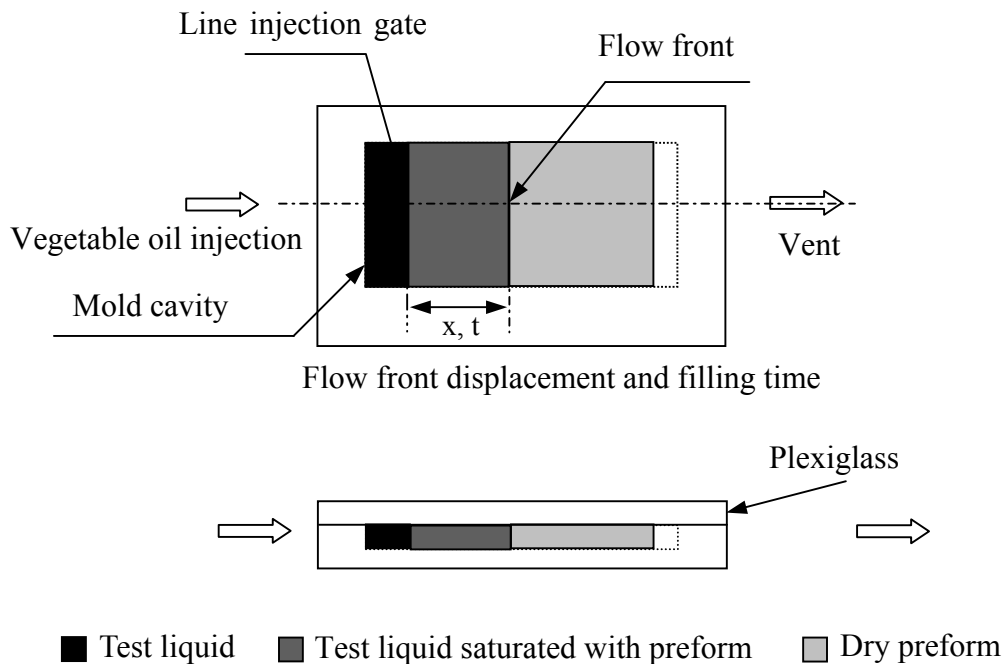


Figure 3.27 Diagram of measured variables

Since the work focuses on characterizing statistical properties of permeability for RTM process design, experimental circumstances were kept as close to real production situation as possible. The following procedures were performed:

1. Ten layers of WR24-5X4 woven fiber carefully cut with a shape cutter;
2. Fibers layed into the mold;
3. Vacuum grease applied along the o-ring;

4. Two halves closed with C-Clamps to prevent leakage;
5. Vacuum pump adjusted to desired pressure;
6. Digital camcorder started;
7. Injection process started;
8. Mold cleaned;
9. Next experiment conducted.

3.2.2 Statistical Analysis of Permeability

In the RTM process, under either isothermal or nonisothermal conditions, preform permeability is the most important factor that determines the resin flow behavior inside the mold and affects the final part quality, i.e. dry spot formation, inadequate wetting, etc. Basically, permeability is defined as a property of measuring the resistance of a resin flowing through a porous medium, which depends on the microscopic and macroscopic structure of the fiber preform. The higher the permeability, the easier the resin flows through the medium. Since the structures of the in-plane and the transverse directions of the fiber preform are quite different, the permeability values in these two directions are distinct. In-plane and transverse permeabilities are shown in the Figure 3.28. K_{11} and K_{22} denote the in-plane permeabilities and K_{33} denotes the transverse permeability.

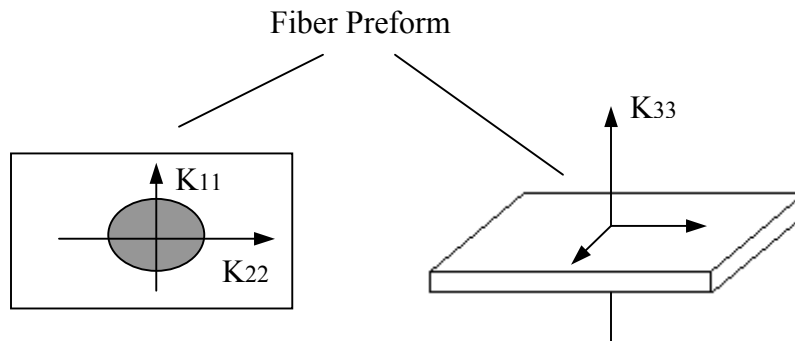


Figure 3.28 Illustrations of in-plane and transverse permeabilities

In the above example, the injection gate is located in the center of the preform. Therefore, the measurements of in-plane permeabilities are accurate enough, and measured permeabilities are valid in RTM simulation for similar tooling design (location of injection gate). However, in RTM manufacturing, since the injection gates are often positioned on the edge of mold, edge effect, commonly called race-tracking, may be encountered as shown in Figure 3.29.

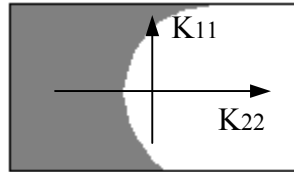


Figure 3.29 Illustrations of edge effects

When race-tracking occurs, the resin flows faster along the edges than in the rest of preform. In addition, during filling process, if the flow along the edges is too fast, dry spots at the end of the mold may be formed (see Figure 3.30).

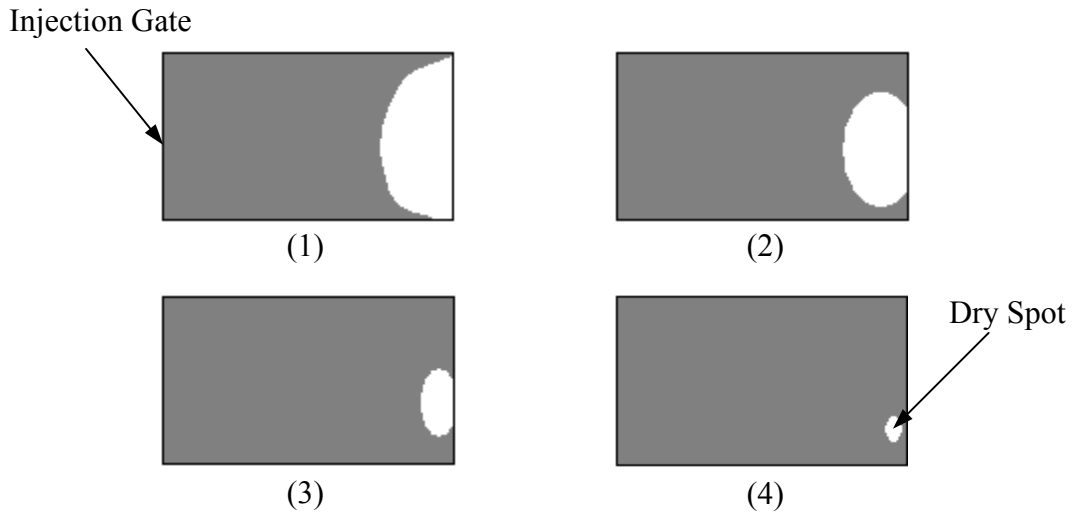


Figure 3.30 Illustrations of dry spot formation

Edge effect is due to lost fiber tows along the edges during fiber handling and misfit between preform and the mold cavity resulting in greater porosity than that in the main region or central area. Edge effect causes measured preform permeability to be higher than the actual value. In most permeability measurement experiments, special care was taken to avoid race-tracking to obtain an accurate permeability value. On the other hand, race-tracking is inevitable in actual RTM processes. The fact that measured permeability and actual permeability in production do not match impedes the application of flow simulation to actual RTM manufacturing. Even though several researchers have tried to model the race-tracking mathematically, the correlation between the permeability of the race-tracking area and that of the rest of fiber preform has not been successfully determined. The mathematical relationship between permeability values for these two regions and how race-tracking affects the flow in the rest of the region must be investigated.

In this thesis research, a correlation was established experimentally with the assistance of RTMSim. In experiments, no special care in transporting, cutting and laying up fibrics was given in order to make the experimental environment as close as possible to actual manufacturing processes, and inevitably, induced race-tracking. The calculations of measured permeability (formula 3.1) for the regions where the test fluid flowed slowest in the whole mold cavity were made under the effect of race-tracking. A total of 46 experiments were performed under the same circumstances.

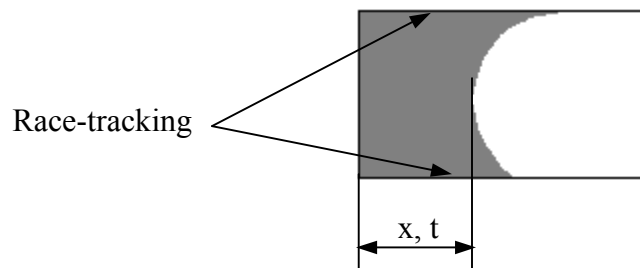


Figure 3.31 Calculation of measured permeabilities

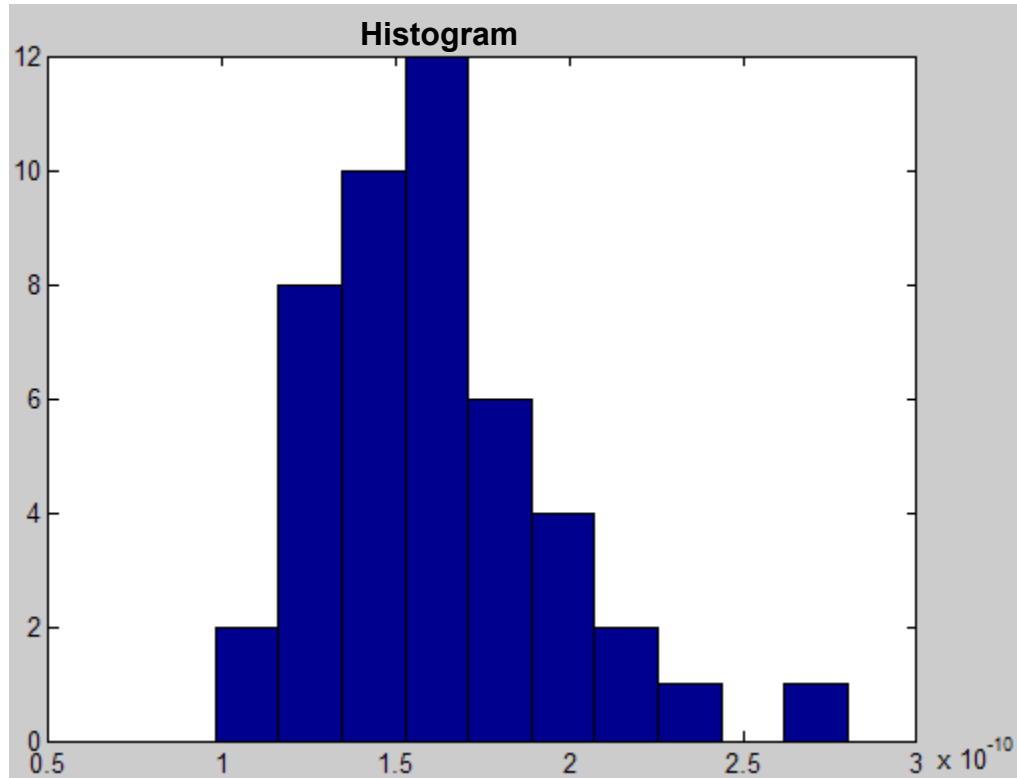


Figure 3.32 Histogram of measured permeabilities (m^2)

Obviously, those measured permeability values are not real permeabilities of the fiber preform; therefore, no effort was made to analyze the statistical distribution of these values. However, they do provide us some information on how race-tracking affects the flow behavior. As shown in Figure 3.32, existence of race-tracking causes the test liquid to flow faster at center area resulting in permeability measurement errors. The permeability for determining the resin flow behavior in the mold should be in the range from $0.98 \times 10^{-10} m^2$ to $2.7 \times 10^{-10} m^2$. Since the experiment setup was designed as a one-dimensional flow, the flow without race-tracking should be the valid one for computing the actual preform permeability. The values located on the left side are preferable for the reason that no or very slight edge effect was observed in these experiments. In simulation, $0.98e-10 m^2$ was selected as the average permeability for the main regions of the fiber preform. Using this real permeability combined with the race-tracking permeabilities as input parameters, the simulation should achieve accurate results. To obtain race-tracking permeability, an inverse approach, as shown in Figure 3.33, was employed.

Simulation results were used to infer the race-tracking permeability instead of current method of mathematically characterizing the race-tracking permeability by channel width.

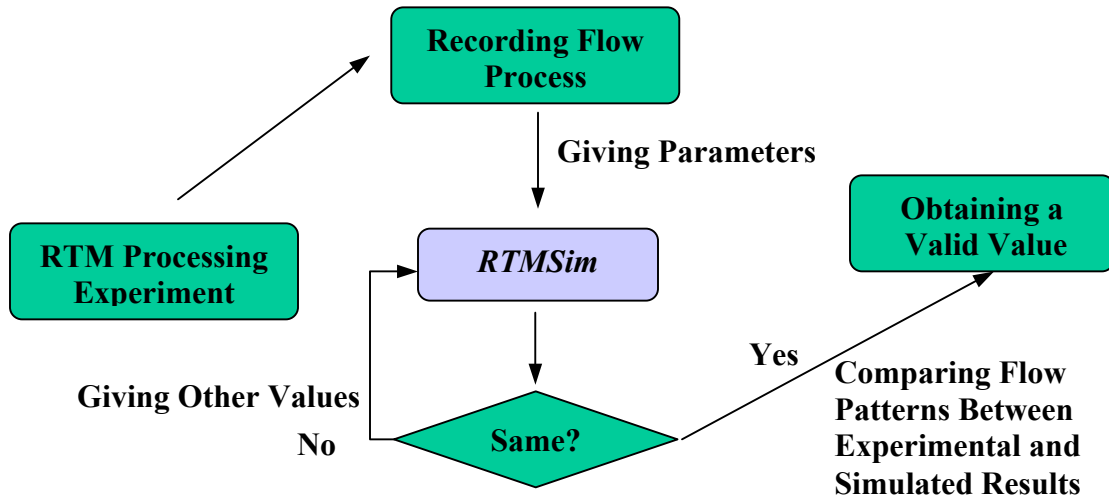


Figure 3.33 Procedures of race-tracking characterization

In the boundary condition definition process of RTMSim, users can define the regions where they have different permeability values. A meshed geometry for the preform, as shown in Figure 3.34, was created with divided regions for separating the race-tracking regions with others. With the ability to divide the regions, race-tracking permeability can be obtained by several trial runs. Inputting the measured permeability value of $0.98e-10 \text{ m}^2$ for the center region and estimating rack-tracking permeabilities into RTMSim, a flow pattern can be obtained. The simulated results were compared with the flow processes recorded by the digital camcorder. If the two results of the experiment, both filling time and flow pattern, were located in a similar range, the actual race-tracking permeabilities for both edges are acquired.

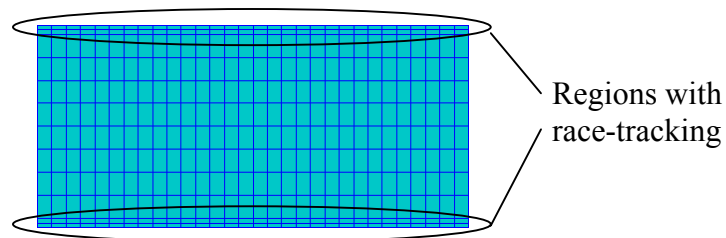


Figure 3.34 Meshed geometry including race-tracking

A detailed example is shown below. Experiment data are listed in Table 3.4.

Table 3.4 Experiment parameters

| Filling time | Injection pressure | Porosity | Test fluid viscosity | Permeability |
|--------------|--------------------|----------|----------------------|-------------------------|
| 53 s | 72807.3 pa | 0.5 | 48 cp | 0.98e-10 m ² |

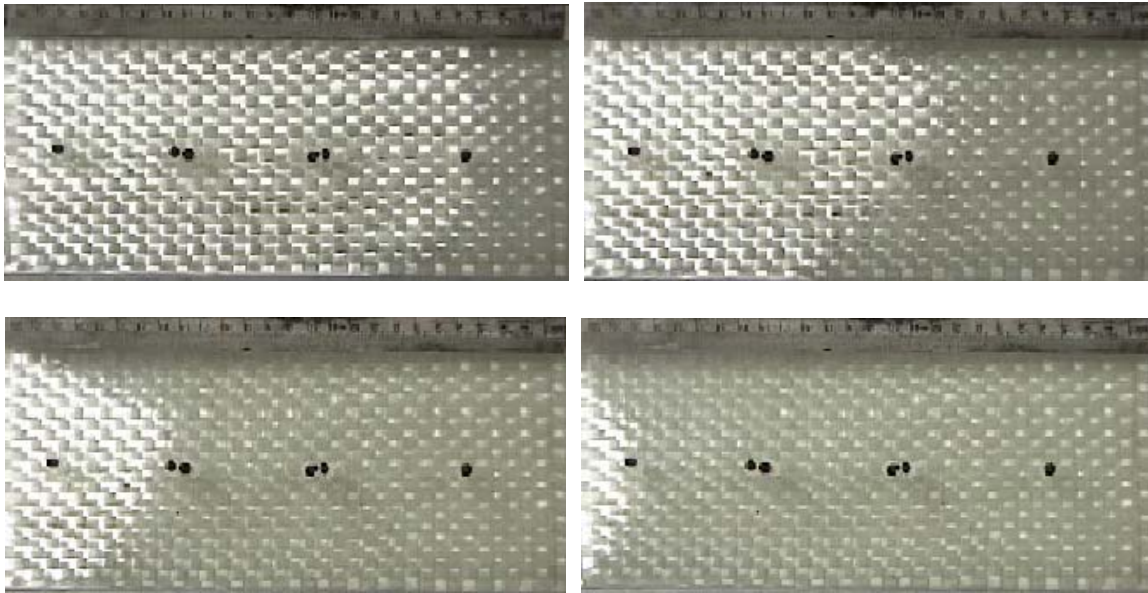


Figure 3.35 Flow process

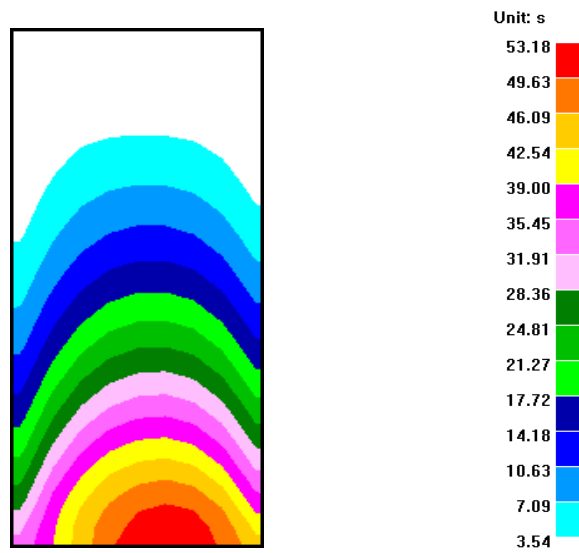


Figure 3.36 Simulation result

Comparing the experimental flow process with simulation result shown in Figure 3.35 and Figure 3.36, respectively, a pair of race-tracking permeabilities was obtained, i.e. $1.13\text{e-}09\text{ m}^2$ and $1.57\text{e-}09\text{ m}^2$. This procedure was repeated 46 times, resulting in 46 pairs of permeability values being obtained (Table 3.5).

Table 3.5 Summary of race-tracking permeabilities (m^2)

| | | | | |
|-----|-------------------|-------------------|-------------------|-------------------|
| Ex# | 1 | 2 | 3 | 4 |
| | 1.13e-09/1.57e-09 | 1.03e-09/1.23e-09 | 3.33e-10/5.88e-10 | 1.52e-09/1.18e-09 |
| Ex# | 5 | 6 | 7 | 8 |
| | 9.31e-10/9.31e-10 | 7.25e-10/9.8e-10 | 6.76e-10/6.76e-10 | 5.78e-10/5.78e-10 |
| Ex# | 9 | 10 | 11 | 12 |
| | 5.88e-10/6.71e-10 | 2.94e-10/5.19e-10 | 7.06e-10/7.64e-10 | 1.13e-09/1.32e-09 |
| Ex# | 13 | 14 | 15 | 16 |
| | 6.76e-10/6.76e-10 | 5.68e-10/7.64e-10 | 2.06e-10/2.55e-10 | 1.08e-09/1.37e-09 |
| Ex# | 17 | 18 | 19 | 20 |
| | 1.13e-09/2.16e-09 | 1.52e-09/1.52e-09 | 3.92e-10/6.37e-10 | 5.39e-10/7.35e-10 |
| Ex# | 21 | 22 | 23 | 24 |
| | 1.27e-09/2.55e-09 | 4.51e-10/6.96e-10 | 6.08e-10/8.23e-10 | 4.02e-10/4.8e-10 |
| Ex# | 25 | 26 | 27 | 28 |
| | 1.76e-10/1.76e-10 | 2.35e-10/2.94e-10 | 5.78e-10/7.01e-10 | 7.84e-10/1.22e-09 |
| Ex# | 29 | 30 | 31 | 32 |
| | 5e-10/7.94e-10 | 4.02e-10/1.76e-10 | 1.03e-10/1.08e-10 | 3.14e-10/3.23e-10 |
| Ex# | 33 | 34 | 35 | 36 |
| | 3.14e-10/3.23e-10 | 2.35e-10/2.94e-10 | 1.23e-10/1.23e-10 | 2.35e-10/2.94e-10 |
| Ex# | 37 | 38 | 39 | 40 |
| | 4.21e-10/4.61e-10 | 5e-10/6.76e-10 | 5e-10/3.13e-10 | 5.88e-10/5.88e-10 |
| Ex# | 41 | 42 | 43 | 44 |
| | 2.84e-10/4.8e-10 | 1.37e-10/6.01e-10 | 7.64e-10/1.18e-09 | 3.43e-10/3.92e-10 |
| Ex# | 45 | 46 | | |
| | 7.64e-10/9.01e-10 | 9.51e-10/1.14e-09 | | |

To obtain more precise analysis results in avoiding truncation or rounding error, a ratio was defined as follows for quantifying the severe level of race-tracking permeability:

$$Ratio = \frac{k_r}{k_a} = \frac{\text{race - tracking permeability}}{\text{average permeability}} \quad (3.2)$$

Table 3.6 Summary of ratios

| | | | | |
|-----|---------|-----------|-----------|-----------|
| Ex# | 1 | 2 | 3 | 4 |
| | 11.5/16 | 10.5/12.5 | 3.4/6 | 15.5/12 |
| Ex# | 5 | 6 | 7 | 8 |
| | 9.5/9.5 | 7.4/10 | 6.9/6.9 | 5.9/5.9 |
| Ex# | 9 | 10 | 11 | 12 |
| | 6/6.85 | 3/5.3 | 7.2/7.8 | 11.5/13.5 |
| Ex# | 13 | 14 | 15 | 16 |
| | 6.9/6.9 | 5.8/7.8 | 2.1/2.6 | 11/14 |
| Ex# | 17 | 18 | 19 | 20 |
| | 11.5/22 | 15.5/15.5 | 4/6.5 | 5.5/7.5 |
| Ex# | 21 | 22 | 23 | 24 |
| | 13/26 | 4.6/7.1 | 6.2/8.4 | 4.1/4.9 |
| Ex# | 25 | 26 | 27 | 28 |
| | 1.8/1.8 | 2.4/3 | 5.9/7.2 | 8/12.5 |
| Ex# | 29 | 30 | 31 | 32 |
| | 5.1/8.1 | 4.1/1.8 | 1.5/1.1 | 3.2/3.3 |
| Ex# | 33 | 34 | 35 | 36 |
| | 3.2/3.3 | 2.4/3 | 1.15/1.25 | 2.4/3 |
| Ex# | 37 | 38 | 39 | 40 |
| | 4.3/4.7 | 5.1/6.9 | 5.1/3.2 | 6/6 |
| Ex# | 41 | 42 | 43 | 44 |
| | 2.9/4.9 | 1.4/6.2 | 7.8/12 | 3.5/4 |
| Ex# | 45 | 46 | | |
| | 7.8/9.2 | 9.7/11.6 | | |

In the following section, these ratio data were analyzed instead of original race-tracking permeability values.

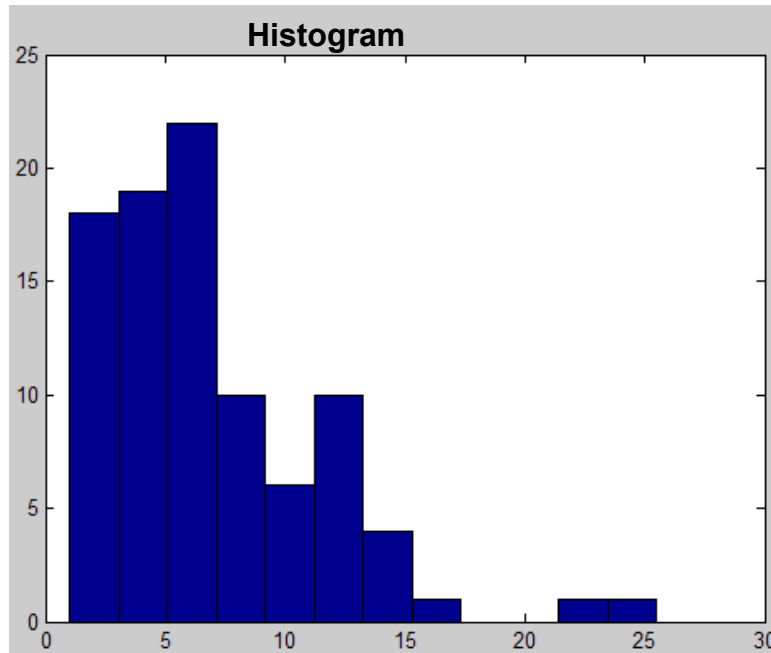


Figure 3.37 Histogram of ratios of race-tracking permeabilities over average value

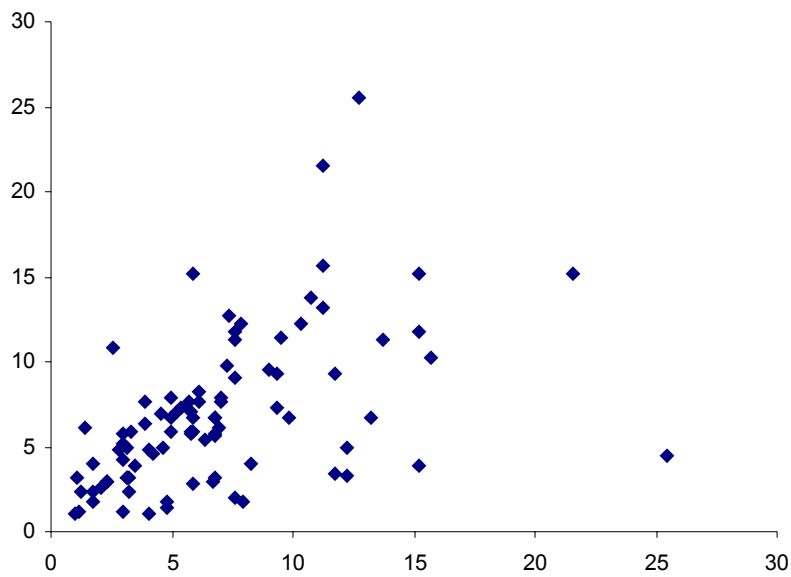


Figure 3.38 Scatter diagram for independence checking

Since permeability is a parameter determined by many random factors, statistical analysis is an effective approach for data analysis. Researchers [54, 56] attempted to utilize statistically distributed variables describing permeability values for their experiments and simulation analysis. This method is becoming more encouraging. In this thesis work, the histogram of ratios (race-tracking permeabilities over average value) shows some type of recognizable shape. In addition, the data points in independence checking diagram, plot of the pairs (x_i, x_{i+1}) for $i = 1, 2, \dots, n - 1$, are scattered randomly throughout the first quadrant.

Based on these results, the assumption that the experimental data were independently and identically distributed (IID) random variables was confirmed. Thus, the next step was to choose a suitable distribution model to fit the data. In some situations, adequate prior knowledge of random variables is helpful for selecting a distribution model. However, for acquiring race-tracking permeability, we did not have sufficient theoretical prior information for determining the type of statistical distribution.

Hypothesizing a distribution family for observation data was structured and results were compared to determine the best model. Three candidate distributions [57, 58] were used to explain the k_r/k_a values for the following three main criteria: (1) continuous distributions should be selected; (2) since ratio values are greater than 1, the defined variables should be positive; and (3) because extremely severe race-tracking happens rarely in real production, histogram of k_r/k_a values tends to have a skew instead of symmetric shape. The shape of distribution function should have the ability to change as estimates of parameters change. To achieve a proper physical or geometric interpretation of experimental data, three types of basic parameters should be defined or fitted correctly: location, scale and shape parameters.

The *location parameter* γ specifies an abscissa location point of a distribution's range of variables. It could be the middle point or the lower endpoint of the distribution's range. The distribution merely shifts either left or right without other changes as γ alters. By changing the *scale parameter* β that determines the unit of measurement of associated values, compression or expansion of corresponding distribution is fulfilled without altering the basic form of distribution. The *shape parameter* α , which affects a distribution's property, such as skewness, determines the basic form of shape of a distribution within the general family of distributions. Some

distributions, for example exponential and normal distributions, do not have a shape parameter; however, others, such as beta distribution, have more than one.

The detailed properties of the three continuous distributions fitted for the k_r/k_a values are listed below:

(1) Gamma distribution: $\text{gamma}(\alpha, \beta)$

$$\text{Density function: } f(x) = \begin{cases} \frac{\beta^{-\alpha} x^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

$$\text{where } \Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1} e^{-t} dt \quad \alpha > 0$$

Parameters: shape parameter $\alpha > 0$, scale parameter $\beta > 0$

Range: $[0, \infty)$

Mean: $\alpha\beta$

Variance: $\alpha\beta^2$

(2) Weibull distribution: $\text{Weibull}(\alpha, \beta)$

$$\text{Density function: } f(x) = \begin{cases} \alpha\beta^{-\alpha} x^{\alpha-1} e^{-(x/\beta)^\alpha} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

Parameters: shape parameter $\alpha > 0$, scale parameter $\beta > 0$

Range: $[0, \infty)$

Mean: $\frac{\beta}{\alpha} \Gamma\left(\frac{1}{\alpha}\right)$

Variance: $\frac{\beta^2}{\alpha} \left\{ 2\Gamma\left(\frac{2}{\alpha}\right) - \frac{1}{\alpha} \left[\Gamma\left(\frac{1}{\alpha}\right) \right]^2 \right\}$

(3) Lognormal distribution: $LN(\mu, \sigma^2)$

$$\text{Density function: } f(x) = \begin{cases} \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.5)$$

Parameters: shape parameter $\sigma > 0$, scale parameter $\mu \in (-\infty, \infty)$

Range: $[0, \infty)$

Mean: $e^{\mu+\sigma^2/2}$

Variance: $e^{2\mu+2\sigma^2}(e^{\sigma^2} - 1)$

Several methods are available for specifying a numerical value for an unknown parameter, i.e. estimating parameters from data, including maximum-likelihood estimation (MLE), least-squares estimation and method of moment. Among these approaches, MLE was selected for its desirable statistical properties. The maximum-likelihood estimator (MLEs) for each candidate distribution and their estimated results are listed as follows: (MATLAB was used for computing these estimators.)

(1) Gamma distribution: $gamma(\alpha, \beta)$

Approximations to $\hat{\alpha}$ and $\hat{\beta}$ can be obtained by the equations and table 6.19 from [57] by considering $\hat{\alpha}$ as a function of T

$$\left\{ \begin{array}{l} T = \left[\ln \bar{X} - \sum_{i=1}^n \ln X_i / n \right]^{-1} \\ \hat{\alpha}\hat{\beta} = \bar{X} \end{array} \right. \quad (3.6)$$

(2) Weibull distribution: $Weibull(\alpha, \beta)$

By solving the following equations, $\hat{\alpha}$ and $\hat{\beta}$ can be computed.

$$\left\{ \begin{array}{l} \frac{\sum_{i=1}^n X_i^{\hat{\alpha}} \ln X_i}{\sum_{i=1}^n X_i^{\hat{\alpha}}} - \frac{1}{\hat{\alpha}} = \frac{\sum_{i=1}^n \ln X_i^{\hat{\alpha}}}{n} \\ \hat{\beta} = \left(\frac{\sum_{i=1}^n X_i^{\hat{\alpha}}}{n} \right)^{1/\hat{\alpha}} \end{array} \right. \quad (3.7)$$

(3) Lognormal distribution: $LN(\mu, \sigma^2)$

$$\left\{ \begin{array}{l} \hat{\mu} = \frac{\sum_{i=1}^n \ln X_i}{n} \\ \hat{\sigma} = \left[\frac{\sum_{i=1}^n (\ln X_i - \hat{\mu})^2}{n} \right]^{1/2} \end{array} \right. \quad (3.8)$$

Table 3.7 Summary of parameters for fitted distributions

| Distribution | Gamma distribution | Weibull distribution | Lognormal |
|--------------|---|---|--|
| MLEs | $\hat{\alpha} = 2.48 \quad \hat{\beta} = 2.759$ | $\hat{\alpha} = 1.638 \quad \hat{\beta} = 7.6829$ | $\hat{\mu} = 1.71 \quad \hat{\sigma} = 0.6869$ |

The fitted distributions are given as follows:

(1) Gamma distribution: *gamma* (2.48, 2.759)

$$\text{Density function: } f(x) = \begin{cases} \frac{2.759^{-2.48} x^{1.48} e^{-x/2.759}}{\Gamma(2.48)} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.9)$$

where $\Gamma(2.48) = \int_0^{\infty} t^{1.48} e^{-t} dt \approx 1.31$

(2) Weibull distribution: *Weibull* (1.638, 7.6829)

$$\text{Density function: } f(x) = \begin{cases} 1.638 \times 7.6829^{-1.638} x^{0.638} e^{-(x/7.6829)^{1.638}} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.10)$$

(3) Lognormal distribution: *LN* (1.71, 0.6869)

$$\text{Density function: } f(x) = \begin{cases} \frac{1}{x\sqrt{0.9437\pi}} e^{\frac{-(\ln x - 1.71)^2}{0.9437}} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.11)$$

In general, since none of these three distributions fits the data perfectly, we have to closely examine them to determine the one that best represents the true underlying distribution for the k_r / k_a values.

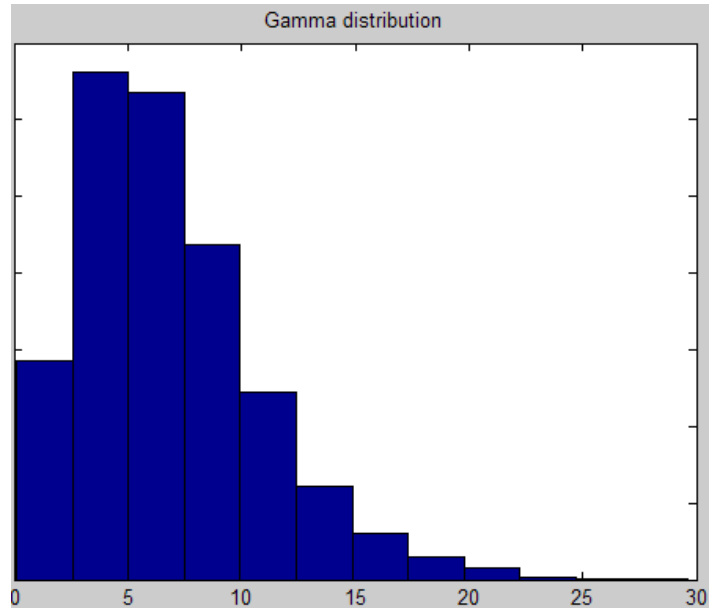


Figure 3.39 Fitted gamma distribution

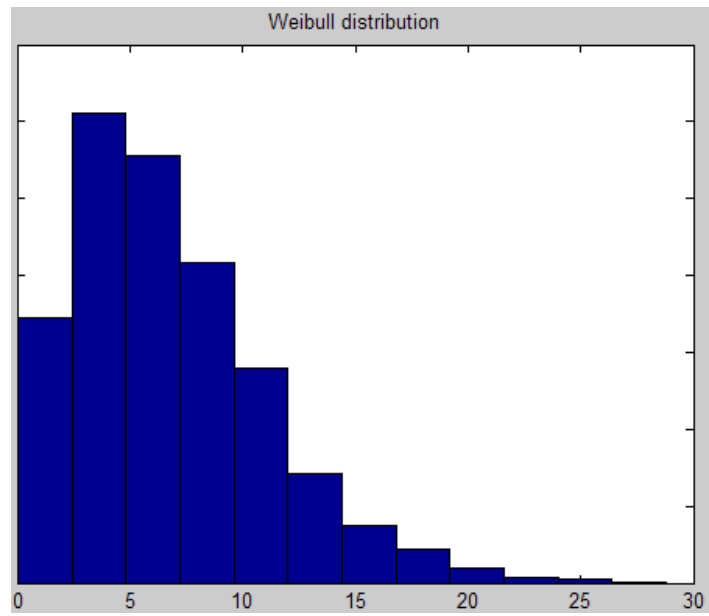


Figure 3.40 Fitted Weibull distribution

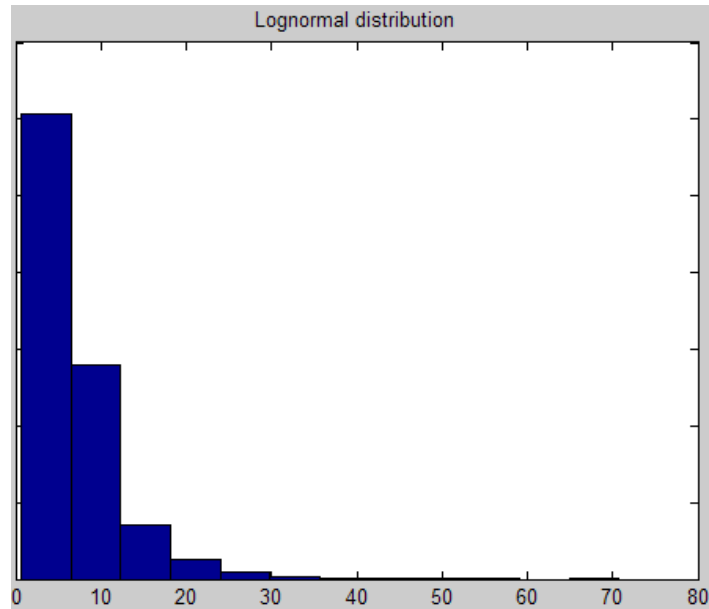


Figure 3.41 Fitted lognormal distribution

Several approaches are available for evaluating the fitted distributions, including heuristic procedures and goodness-of-fit hypothesis tests. Before performing these procedures, the histograms of fitted distributions were plotted. The fitted gamma distribution agreed with experimental data very well, as did the fitted Weibull distribution, including both shape of distribution and range of variables. With the fitted lognormal distribution, even though the variables were located within the same range of experimental data, the shape differed significantly. Thus, more procedures for checking the quality of fitted distributions were necessary for comparing the fitted gamma and Weibull distributions.

The goodness-of-fit test is an effective method to assess whether the observations, such as X_1, X_2, \dots, X_n are independent samples from a particular distribution. A null hypothesis was tested:

$$H_0 : \text{The } X_i\text{'s are IID random variables with distribution function } \hat{F}$$

where \hat{F} is distribution function of *gamma* (2.48, 2.759) or *Weibull* (1.638, 7.6829) .

Chi-Aquare Tests was performed as goodness-of-fit hypothesis test [57]. The test statistic is:

$$\chi^2 = \sum_{i=1}^k \frac{(N_i - np_i)^2}{np_i} \quad (3.12)$$

where:

$$p_i : \int_{a_{i-1}}^{a_i} \hat{f}(x)dx;$$

N_i : Number of X_i 's in the i^{th} interval $[a_{i-1}, a_i)$;

$$n: \sum_{i=1}^k N_i;$$

k: Number of observations.

Table 3.8 Summary of statistics

| | Mean | Variance | χ^2 | Degrees of freedom |
|--------------------------------|--------|----------|----------|--------------------|
| Experimental data | 6.8419 | 19.996 | N/A | N/A |
| <i>gamma</i> (2.48, 2.759) | 6.8423 | 18.878 | 32.78 | 11 |
| <i>Weibull</i> (1.638, 7.6829) | 6.8742 | 18.537 | 21.66 | 11 |

The hypothesis that the data are from a population with the specified distribution is rejected if:

$$\chi^2 > \chi_{k-1,1-\alpha}^2 \quad (3.13)$$

where $\chi_{k-1,1-\alpha}^2$ is the chi-square percent point function with $k - 1$ degrees of freedom and a significant level of α .

Table 3.9 Summary of conclusion

| Alpha level | Cutoff | <i>gamma</i> (2.48, 2.759) $\chi^2 = 32.78$ | <i>Weibull</i> (1.638, 7.6829) $\chi^2 = 21.66$ |
|------------------|--------|--|--|
| $\alpha = 5\%$ | 19.675 | Reject H_0 | Reject H_0 |
| $\alpha = 2.5\%$ | 21.920 | Reject H_0 | Accept H_0 |
| $\alpha = 1\%$ | 23.209 | Reject H_0 | Accept H_0 |

From the above summary, the Weibull distribution was found to perform better than the gamma distribution as the statistical model for characterizing the k_r/k_a values. Since it was case

specific, i.e. woven fiber, one dimensional experiment setup, etc, it cannot be concluded that the k_r/k_a values for different cases can be represented by Weibull distributed variables. In the remainder of this thesis work, Weibull distribution was used to simulate the input parameters of a two-dimensional flow mold cavity for virtual tooling design for RTM processes.

3.3 Robust Tooling Design with Virtual Manufacturing

Mold filling is the core phase of the RTM process. Much work regarding RTM process design and optimization has been done. However, many researchers made some strict assumptions, such as permeability of the preform is uniform and isotropic across the part, which may not be true in actual production cases. In some cases of current design practice, optimal design based on these assumptions will not be valid if race-tracking is present. In addition, since race-tracking has a significant effect on resin flow behavior, both adverse and favorable, variations of race-tracking permeability often lead to inconsistent part quality. This thesis research introduces a new optimization approach to minimize the sensitivity of the mold design to uncertain material properties by choosing the appropriate locations of gates and vents. This new tooling design scheme can improve the robustness of RTM tooling and processes design.

Taguchi [59] realized the importance of low sensitivity in product quality and introduced innovative concepts and approaches for reducing variations in process development. Even though debated, his methodology has gained wide usage in industries. Basically, by conducting planned experiments under some assumptions that uncontrollable or noise variable can be precisely controlled, the designer chooses the levels of controllable variables to accomplish a robust system that is insensitive to inevitable changes of the noise variables [60, 61]. However, this traditional robust design approach is not applicable in RTM mold filling process design, even though locations and number of gates and vents can be considered as controllable variable, and race-tracking permeability can be considered as a noise variable. The reason is that the assumption mentioned above is unsatisfied, since race-tracking permeability cannot be controlled in the experiments.

The ratios indicating the severe level of the race-tracking for WR24-5X4 woven fiber with porosity 50% were statistically characterized as Weibull variables. In addition, among all the process parameters, race-tracking and the gate/vent locations are the governing factors for

resin flow inside the mold. Therefore, a combined method: a virtual experiment coupled with Monte-Carlo simulation model, was introduced.

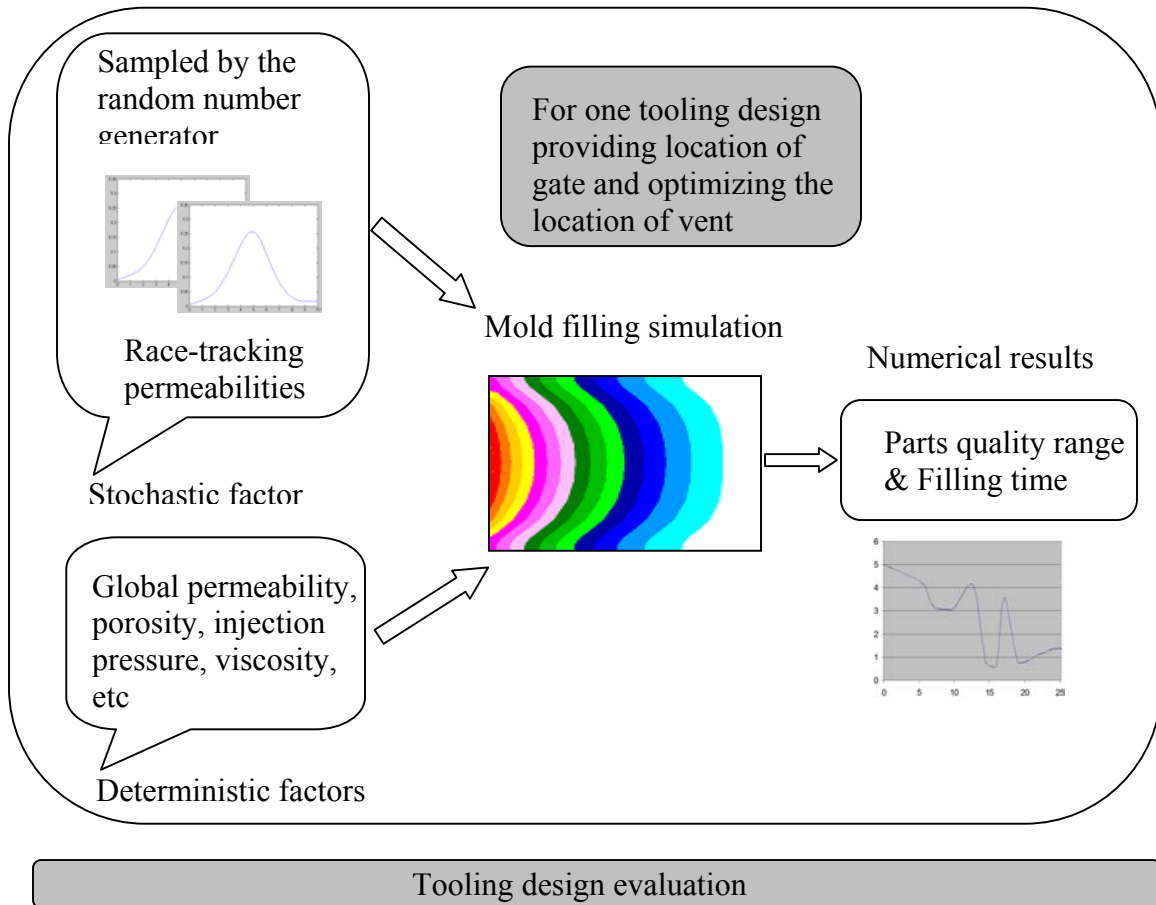


Figure 3.42 Illustration of proposed analysis method

In the RTM process, the principal factors that determine the resin flow process and final part quality can be grouped into two types: deterministic factors and stochastic factors. Injection pressure, flow rate, mold temperature, etc are the deterministic factors, which means they can be measured or controlled as desired. The primary sources of uncertainty are the preform permeability dominated by its microstructure and the variability of rheological and kinetic properties of the resin. Those uncertainties presented by the above factors occur in different magnitudes. For example, for different mold sizes, race-tracking permeability occurring in the edges or sharp corners of preform usually ranges from 2 to 3 up to over hundred times of the

main region permeability. However, the rheological and kinetic characterization of resin, i.e. viscosity, does not change as much as race-tracking permeability if filling time is not excessively high. Therefore, the sources of crucial uncertain input variables were narrowed. Only race-tracking permeability was considered as stochastic factor. The same material described in the subsection 3.2.1 was used for the robust RTM process design so as to utilize the conclusion that the k_r/k_a values distributed as Weibull random variables (*Weibull* (1.638, 7.6829)). The density function is listed as expression 3.10. With the help of virtual manufacturing, many combinations of statistically generated input parameters and deterministic parameters can be used as inputs for the flow simulation software to assess the current tooling design by well-defined objectives.

A quantitative index must be defined and used as an objective for an optimized design of a RTM process. Usually, a proper flow should avoid dry spot formation and result in comparatively less filling time. The original quality index was defined as follows [62]:

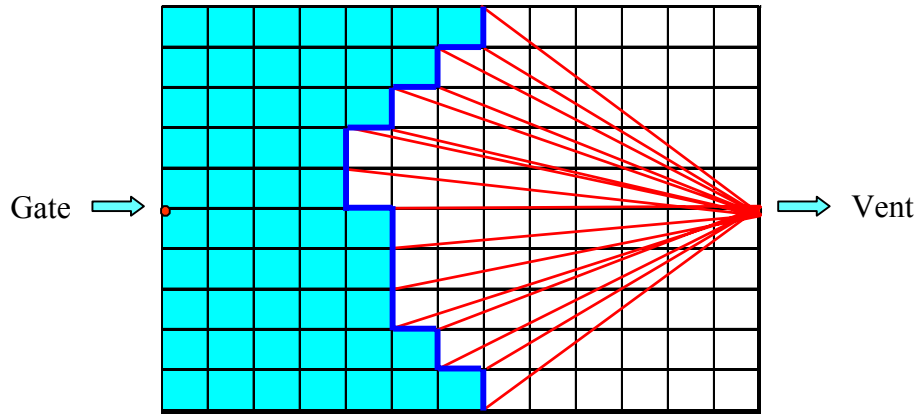


Figure 3.43 Illustration of quality index

$$q_k = \sqrt{\frac{\sum_{i_k=1}^{n_k} (d_{i_k} - \bar{d}_k)^2}{n-1}}; \quad Q = \frac{T \times \sum_{k=1}^m q_k}{m} \quad (3.14)$$

where:

q_k : Intermediate resin flow index for flow front k ;

m : Number of flow fronts taken from the simulation model;

- n_k : Number of nodes on flow front k ;
- d_{ik} : Distance from the node i on the flow front k to the vent;
- \bar{d}_k : Average distance from nodes on flow front k to the vent;
- Q : Overall resin flow index;
- T : Total filling time.

The mold filling process was divided into m time steps (m flow fronts). For each of the m flow fronts, an intermediate resin flow index value (q_k) that characterizes the flow front shape in time step k could be calculated. The physical meaning of q_k denotes the standard deviation of distances between flow front and the vent. A flow front centering around the vent that leads to smooth flow and fine part quality is preferable. Due to the short filling time requirement, the overall performance of tooling design could be assessed. Therefore, the overall resin flow index of the mold filling process (Q) was calculated from all q_k and total filling time T .

Jiang *et al.* [28] provided a mesh distance-based method of quantifying the part quality.

$$P_l = \frac{d_{vent}}{d_{max}} + k \frac{A_v}{A} \quad (3.15)$$

where:

- P_l : Process performance index;
- d_{vent} : Maximum of the distances between any gate and vent;
- d_{max} : Maximum distance between two nodes on the model;
- k : Penalty weight;
- A_v : Area of the regions in which vents should be placed but are not;
- A : Total area of the model.

The primary advantages of this definition are: (1) the ability to evaluate the part with cutoffs and (2) easily dealing with multiple gates and vents.

From this work, an improved distance based quality performance index for quantifying the resin flow quality was established as the output of RTMSim program associated with each combination of input parameters.

$$q_k = \sqrt{\frac{\sum_{i_k=1}^{n_k} (d_{i_k} - \bar{d}_k)^2}{n-1}}; \quad Q = \frac{\sum_{k=1}^m k \times q_k}{\sum_{k=1}^m k} \quad (3.16)$$

where:

q_k : Intermediate resin flow index for flow front k ;

m : Total number of flow fronts taken from the simulation model;

k : k^{th} flow front;

n_k : Number of contouring points on flow front k ;

d_{ik} : Distance from the node i on the flow front k to the vent;

\bar{d}_k : Average distance from nodes on flow front k to the vent;

Q : Overall resin flow index.

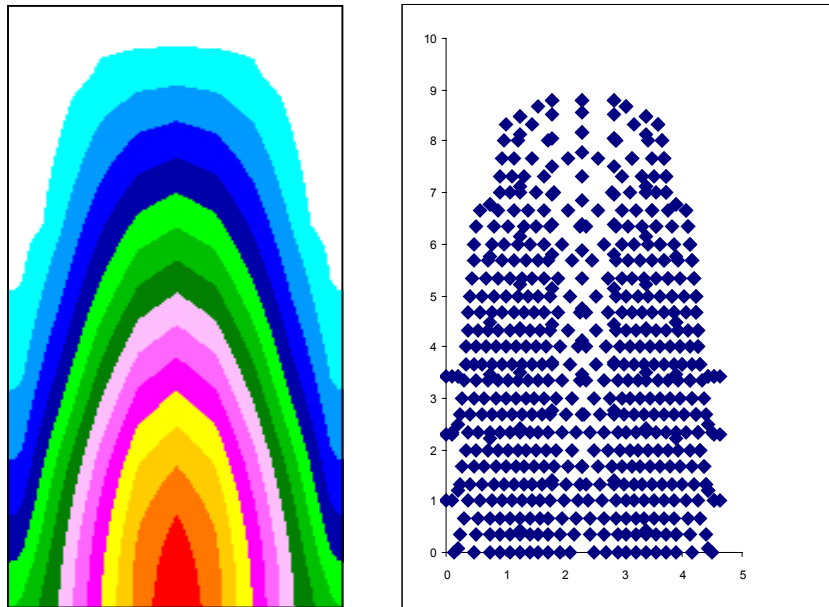


Figure 3.44 Illustration of improved quality index

Three changes were made from the original definition of distance-based performance index:

- Points on the contour line computed by RTMSim program instead of mesh nodes are used to calculate the distances for intermediate resin flow index to obtain more accurate distance variance;
- The presence of race-tracking will shorten the mold filling time. Thus, the total filling time is removed from overall performance index and listed as the second objective;
- A sound flow pattern requires that resin flow fronts approach the vent at the same speed so as to obtain a dry spot free part. However, the shape of the prime flow fronts is not as important as the shape of last flow fronts. Consequently, a weighting parameter k denoting k^{th} flow front is multiplied by each corresponding intermediate resin flow index and averaged by a total number of flow fronts, m , to obtain a overall performance index.

For each tooling design, 100 simulated input combinations are provided and accordingly, several sets of resulting data consisting of overall performance indices and filling times are acquired as evaluating responses.

Gates and vents of the mold are commonly considered as the most crucial factors contributing to the mold filling process; Therefore, they are the factors investigated by virtual experiments. In this work, virtual experiment designs that involves several joint-effect factors to one or more responses were conducted to evaluate the best possible tooling designs. In the following case study, both the number of gates and vents were fixed as one, and distinct arrangement of gate locations and vent locations were classified as categorical quantities. The simulation was first run without giving the location of vent. According to the flow patterns, the best location of vent was optimized such that objectives were satisfied.

3.3.1 Case Study

A rectangle plate part with two rectangular holes was designed to assess the robust analysis methodology. Since the dimension of thickness is far less than the other two directions, a 2D FEM model neglecting the thickness was built to simulate the resin flow.

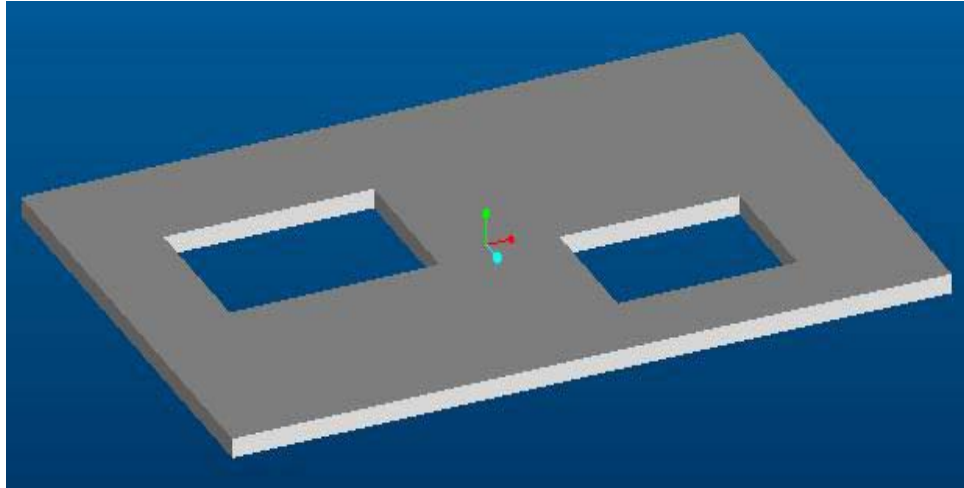


Figure 3.45 3D model of designed part

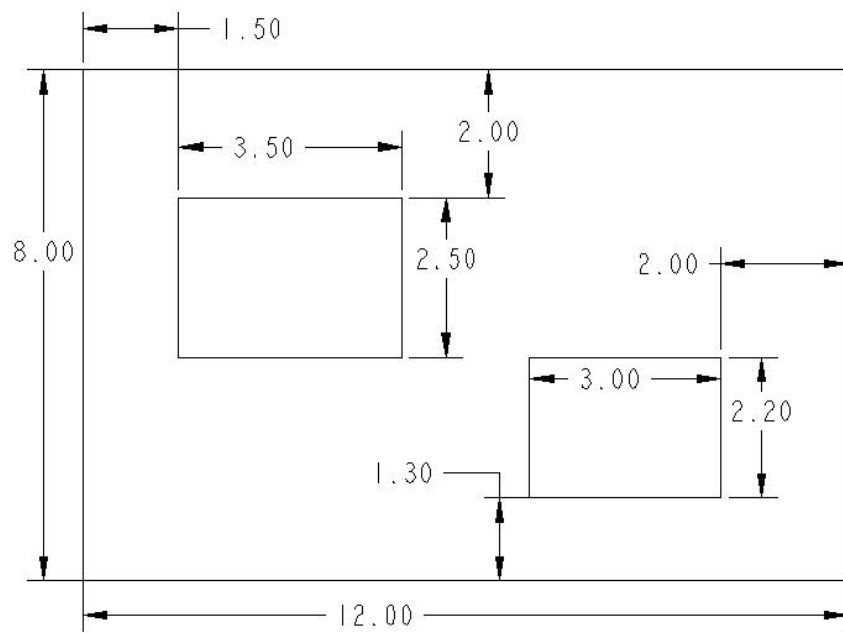


Figure 3.46 Engineering drawing of designed part

3.3.2 Process Parameters Generation

Both deterministic and stochastic parameters were necessary for simulation. The deterministic simulation parameters selected were close to those used in experiments conducted for material characterizations to create a real production circumstances.

Table 3.10 Deterministic parameters

| Fiber preform | Injection type | Injection pressure | Porosity | Resin viscosity | Main region permeability |
|---------------|-------------------|--------------------|----------|-----------------|--------------------------|
| WR24-5X4 | Constant pressure | 72500 pa | 0.5 | 48 cp | 0.98e-10 m ² |

From material characterization analysis, the race-tracking permeability values were affected by several sources, including fiber tows lost at edges, stacking sequence of the preform, and misfit between preform and mold cavity. The k_r/k_a values from previous experiments were characterized as Weibull variables (*Weibull* (1.638, 7.6829)). Therefore, we can utilize this statistical property to model the race-tracking permeability values for the production part, if the parts have a similar geometry and manufacturing environment. For the designed part, four outside edges and eight interior edges contact the mold cavity. Twelve independent race-tracking permeability values were needed for each simulation, and since 100 simulations were performed to evaluate the robustness of current tooling design, 1200 values were generated from the random number generator.

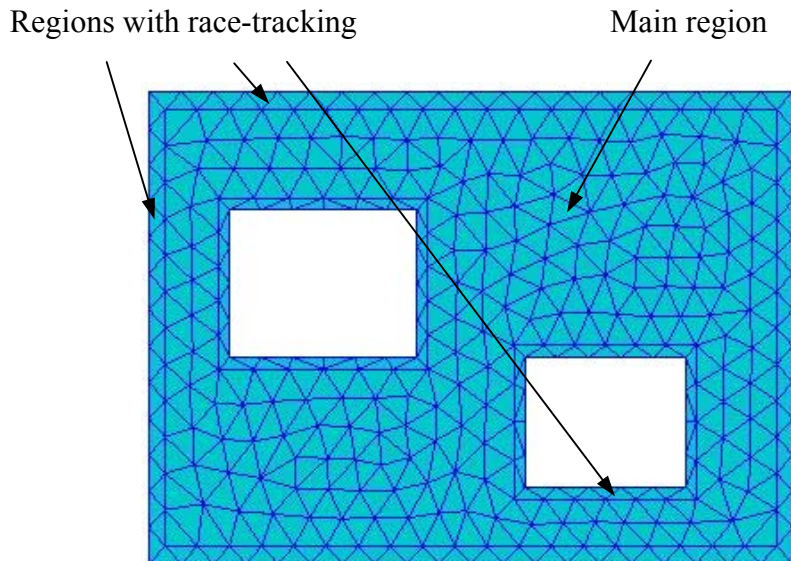


Figure 3.47 FEM model of designed part

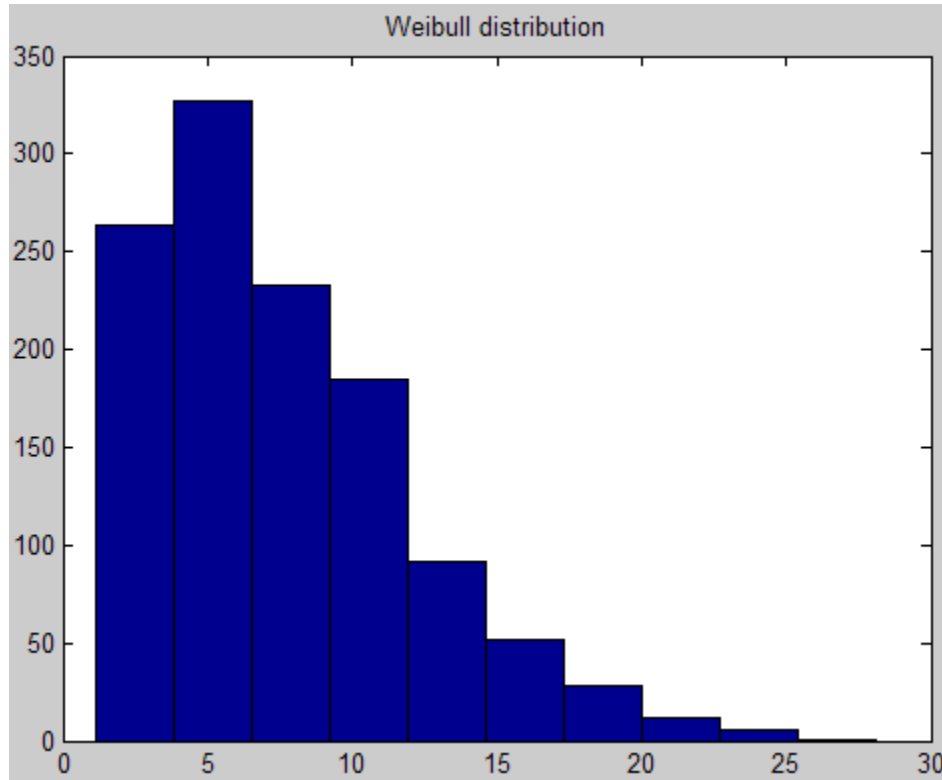


Figure 3.48 Histogram of generated k_r/k_a values

3.3.3 Robustness Analysis

Results from RTMSim consist of mold filling time contouring plot and pressure field contouring plot. Quality performance indices were calculated with a given location of vent and filling time contouring plot. The variations of parts' quality were denoted by the statistical properties of performance index values, such as the range and the variance of these values. A proper indicator of system robustness can be formulated as follows:

Let φ represent the sensitivity of the current system and be defined as follows:

$$\varphi = f(g, v) \quad (3.17)$$

where g and v are the location of gate and vent (location of gate is assigned and the location of vent is to be optimized).

Meanwhile, some other constraints apply. The values of filling time and performance index should be as small as possible to shorten the cycle time and obtain a better quality part. Thus, the optimal tooling design is one that has comparatively short average filling time, small

average performance index and narrow range, as well as a variance of performance indices for 100 simulated filling experiments. To minimize sensitivity φ and other constrains, the following optimization formulations were solved:

$$\begin{aligned}
 & \min \quad \text{var}(q^{all}) = f(g, v) \\
 & \min \quad \text{range}(q^{all}) = g(g, v) \\
 & \text{subject to} \quad \text{low average filling time \& average } q^{all}
 \end{aligned}
 \tag{3.18}$$

The most possible locations of gates and vents are illustrated in Figure 3.49. Thirty-two cases were studied and compared to identify the best tooling design.

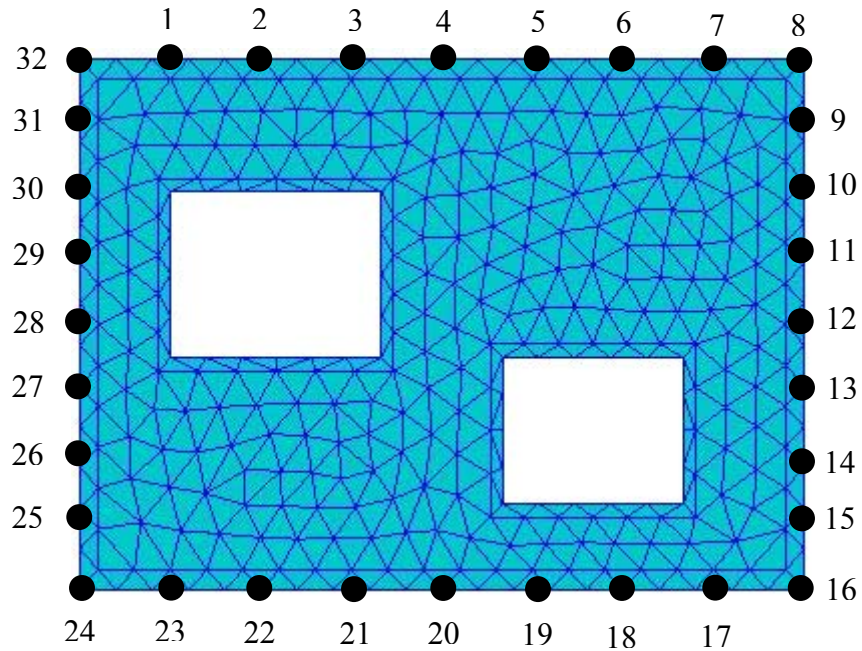


Figure 3.49 Possible locations of gates and vents

3.3.4 Results and Discussion

The robustness analysis framework previously described was utilized to determine how the uncertain process parameters affect the mold filling process. Among the four objectives, the average q^{all} was the most important due to its direct connection with part quality. A constant for q denoting the threshold between a good part and a waste part should be found prior to optimization.

Void formation during the RTM flow process leads to poor part quality. The performance index defined in this thesis work is a sound indicator for quantifying the part quality. After several case studies, the cut-off value for void formation was determined as shown in Figure 3.50. If q is greater than 1.3, there is an increased possibility that a dry spot will form.

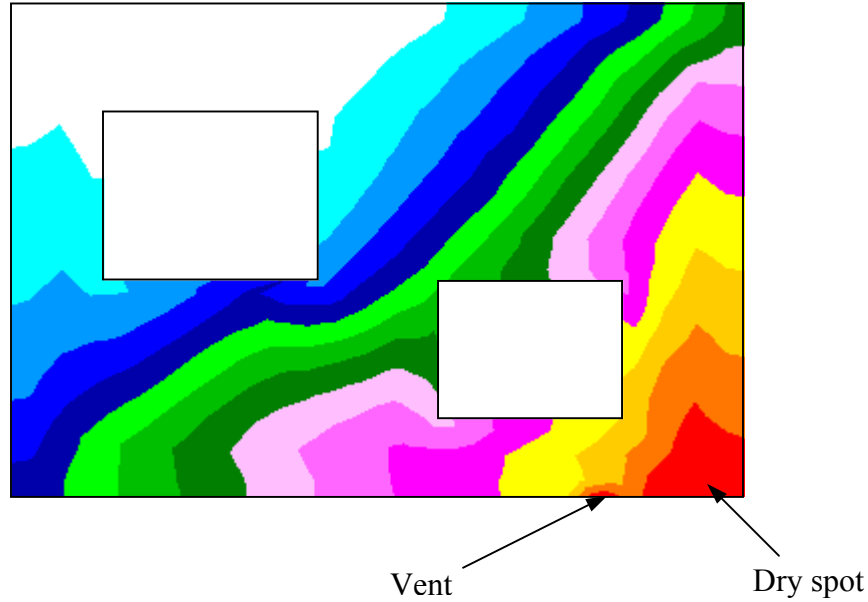


Figure 3.50 Dry spot formation ($q = 1.3$)

A robust tooling design should have the outputs that are insensitive to uncertainties. Minimizing sensitivity suggests both variance and range of outputs to be as small as possible.

$$\text{var}(q^{all}) = \frac{\sum_{i=1}^{100} (q_i - \bar{q})^2}{99} \quad (3.19)$$

$$\text{range}(q^{all}) = \max(q^{all}) - \min(q^{all}) \quad (3.20)$$

Figure 3.51 shows q^{all} for a typical tooling design. The distributed race-tracking permeability values treated as an uncertain process parameter influence the final parts' quality. The part quality changes corresponding to altering process parameters. Due to the extensive changes of race-tracking permeability, obtaining unqualified parts, even with optimal tooling design, is inevitable.

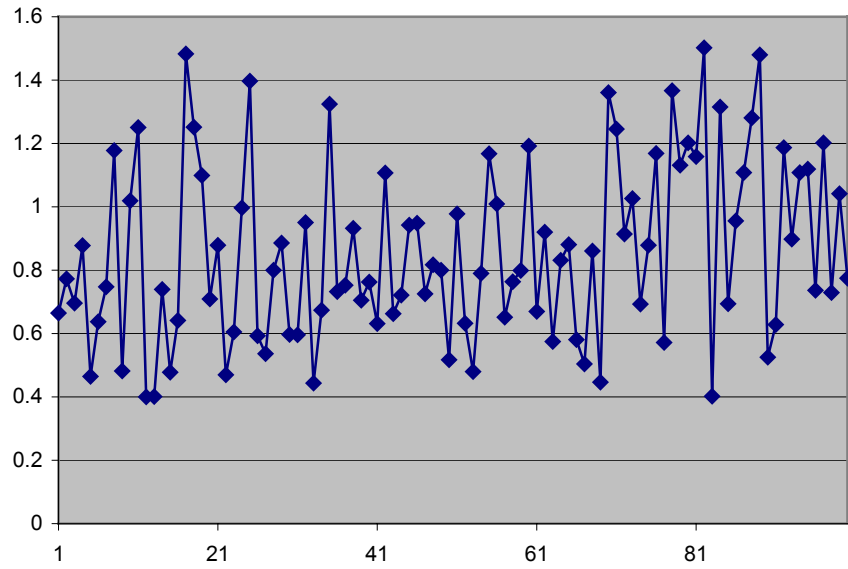


Figure 3.51 q^{all} plot

Besides finding an optimal tooling design by minimizing the process output variabilities and maintaining the parts' quality, it is desirable to minimize the average cycle time as well.

Table 3.11 Results summary

| Location of gate | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Optimal location of vent | 16 | 16 | 17 | 20 | 23 | 24 | 24 | 26 |
| Variance of q^{all} | 0.089 | 0.059 | 0.069 | 0.019 | 0.046 | 0.068 | 0.103 | 0.055 |
| Range of q^{all} | 1.24 | 1.46 | 1.58 | 0.62 | 1.23 | 1.30 | 1.22 | 1.32 |
| Average q^{all} | 1.16 | 1.21 | 1.47 | 1.71 | 1.16 | 1.05 | 1.05 | 0.95 |
| Average filling time (s) | 157 | 142 | 131 | 118 | 132 | 143 | 161 | 199 |
| Location of gate | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

Table 3.11 continued

| | | | | | | | | |
|--------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| Optimal location of vent | 24 | 26 | 27 | 28 | 29 | 30 | 32 | 30 |
| Variance of q^{all} | 0.139 | 0.080 | 0.038 | 0.034 | 0.021 | 0.090 | 0.021 | 0.060 |
| Range of q^{all} | 1.51 | 1.10 | 0.81 | 0.78 | 0.63 | 1.31 | 0.63 | 1.10 |
| Average q^{all} | 1.13 | 0.86 | 0.76 | 0.84 | 0.89 | 1.33 | 0.89 | 1.12 |
| Average filling time | 171 | 157 | 150 | 148 | 154 | 162 | 154 | 197 |
| Location of gate | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Optimal location of vent | 32 | 2 | 1 | 2 | 8 | 8 | 7 | 6 |
| Variance of q^{all} | 0.095 | 0.072 | 0.064 | 0.027 | 0.108 | 0.091 | 0.118 | 0.110 |
| Range of q^{all} | 1.21 | 1.12 | 1.21 | 0.95 | 1.64 | 1.47 | 1.42 | 1.45 |
| Average q^{all} | 1.12 | 1.37 | 1.55 | 2.46 | 1.61 | 1.01 | 1.10 | 1.38 |
| Average filling time (s) | 160 | 158 | 129 | 116 | 125 | 144 | 166 | 211 |
| Location of gate | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| Optimal location of vent | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Variance of q^{all} | 0.122 | 0.078 | 0.042 | 0.055 | 0.086 | 0.092 | 0.130 | 0.137 |
| Range of q^{all} | 1.53 | 1.30 | 0.99 | 1.01 | 1.15 | 1.22 | 1.46 | 1.54 |
| Average q^{all} | 0.95 | 0.77 | 0.69 | 0.84 | 1.00 | 1.02 | 1.21 | 1.27 |
| Average filling time | 169 | 157 | 150 | 150 | 152 | 159 | 173 | 197 |

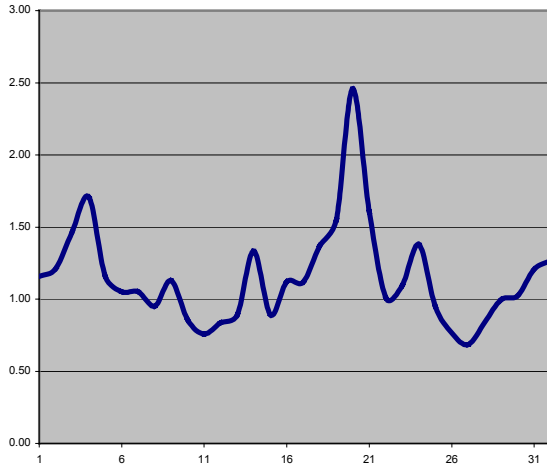


Figure 3.52 Plot of average q^{all}

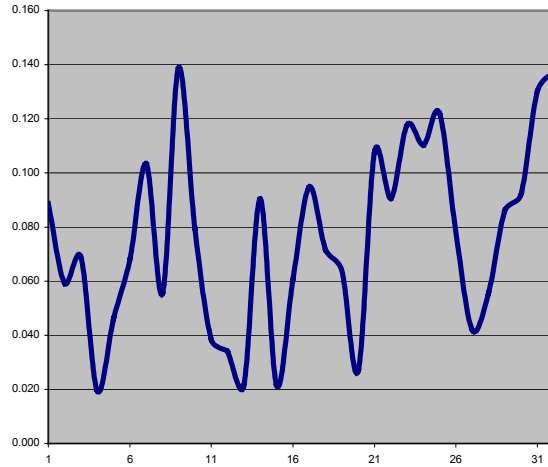


Figure 3.53 Plot of variance of q^{all}

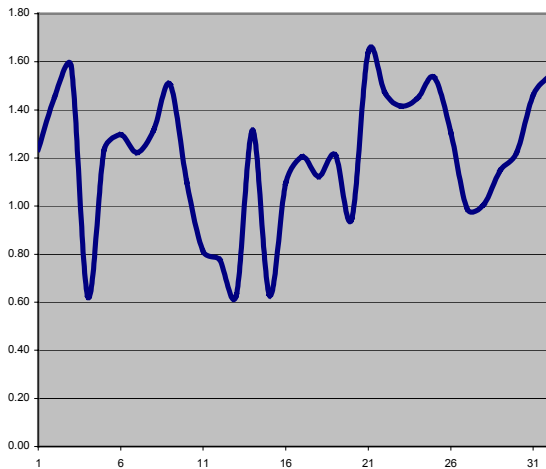


Figure 3.54 Plot of range of q^{all}

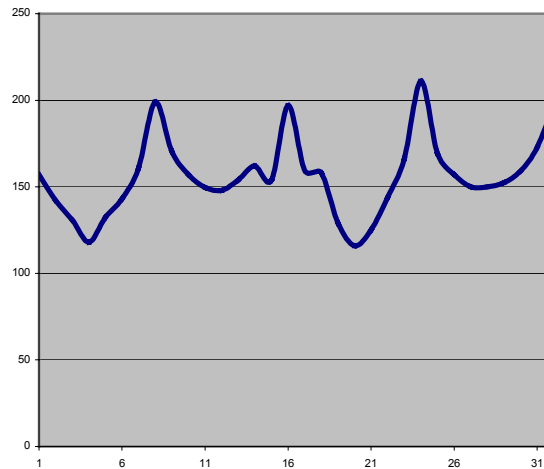


Figure 3.55 Plot of average filling time

The robust tooling design should maintain parts quality. From Figure 3.52, the possible optimal tooling designs are those whose q^{all} are far less than 1.3. Among these tooling designs, 12 (gate 12 and vent 28) would achieve the best outputs (see Table 3.12).

Table 3.12 Summary of the optimal tooling design

| | |
|--------------------------|-------|
| Variance of q^{all} | 0.034 |
| Range of q^{all} | 0.78 |
| Average q^{all} | 0.84 |
| Average filling time (s) | 148 |

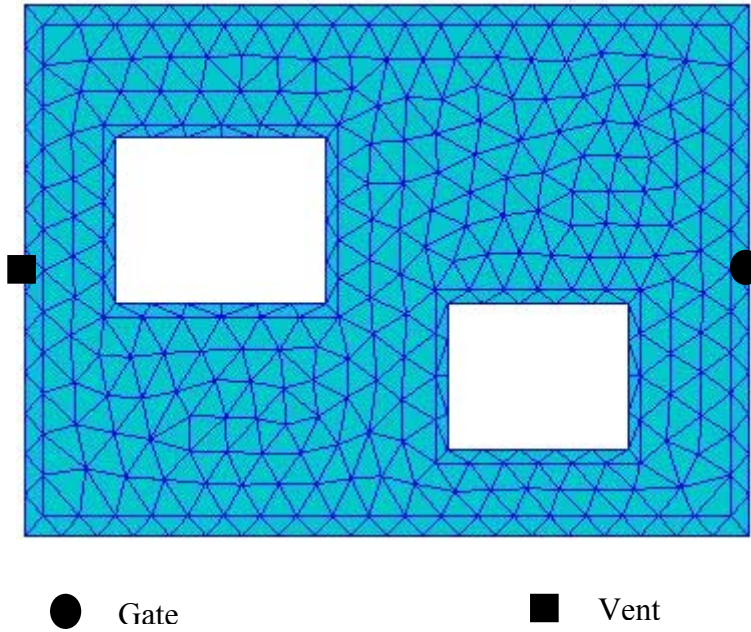


Figure 3.56 Robust tooling design

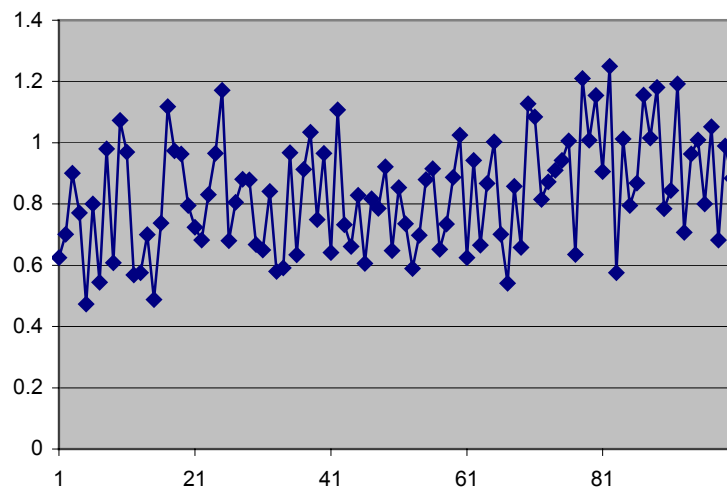


Figure 3.57 q^{all} plot of robust tooling design

The above analysis shows that even though the part has asymmetric geometry, the symmetric arrangement of gates and vents still performs very well. However, as illustrated in Figure 3.58, resin flows faster along edges due to presence of race-tracking resulting in large possibility of dry spot formation, so the gate and vent should be assigned along the direction with longer edge to avoid the dry spot formation.

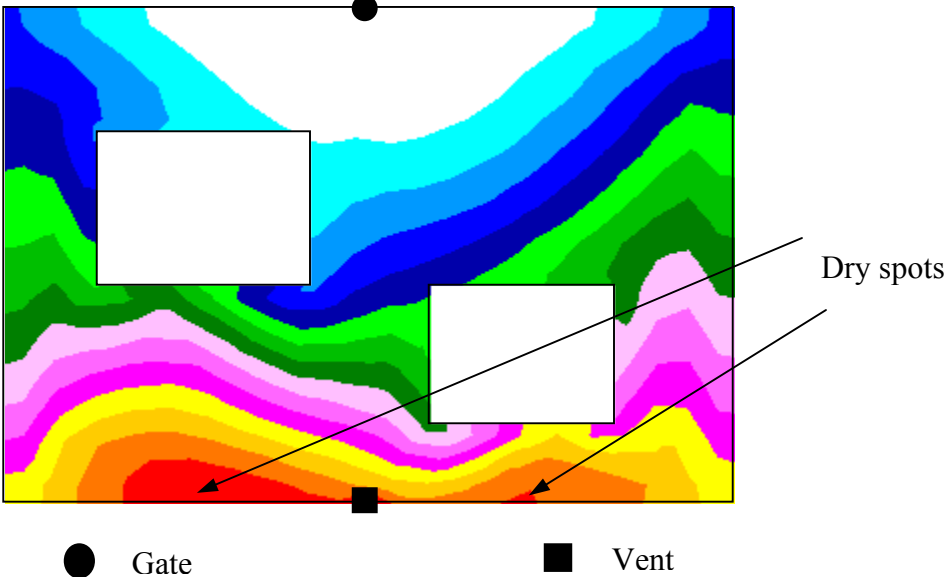


Figure 3.58 Demonstration of a bad tooling design

CHAPTER IV

CONCLUSIONS

The objectives of this thesis work were: (1) develop an integrated design environment for RTM process, (2) study the statistical properties of raw materials using in RTM manufacturing process and combine a simulation tool provided with characterized process parameters, and (4) perform the robust tooling design.

Although RTM processes have been used by industries for decades and different aspects of RTM process have been researched. A number of variations during manufacturing make the traditional RTM trial-and-error process costly and time consuming. The simulation-based optimization methodology from statistical point of view provided by this thesis work reveals that variations of process variables have a great impact on the final parts not only cycle time but also quality of the parts. Stochastic factors must be used for RTM optimization design.

4.1 Contributions

After extensive research work, several contributions were achieved:

1. A prototype of integrated design environment for RTM processes has been built. Components encompassing flow simulation and cost analysis were implemented into this modular software system. Databases were used as data storage, as well as the link for different modules.
2. Sources of variations were analyzed and the most important variable, i.e. preform race-tracking permeability values resulting in inconsistent parts' quality, were characterized statistically through repeated one-dimensional flow experiments. It was found that the ratios (race-tracking permeability over average value) can be represented by Weibull variables.

3. A simulation based robust tooling analysis taking into account the statistically distributed process variables was introduced and performed by the simulation tool. Through case studies, we concluded that if there are only one gate and one vent, they should be positioned symmetrically along the length direction of the part.
4. An improved objective (flow performance index) for quantifying part quality was successfully defined and a contour points based algorithm was implemented into RTMSim for computing flow performance index more accurately.

4.2 Extension of Research

Although this thesis work optimized the RTM process from a distinct view, the research process poses several questions that must be investigated to achieve even better results. Future work for this realm is suggested as follows:

1. Characterize preform permeability at main regions. A one-dimensional flow method is not accurate enough for the main region permeability measurement due to edge effect. Instead, even though more complex, two-dimensional method can be employed to perform repeated experiments so as to obtain the statistical properties of main region permeability values.
2. Characterize of race-tracking permeability values for different materials, preform geometry and porosity. The statistical properties may vary whenever materials, preform geometry and porosity change. Mathematical models for predicting variations could be constructed through more experiments and theory analysis.
3. Design an effective and efficient optimization approach for robust design. Since a great deal of computing for evaluating objectives is entailed for each tooling design, a method that converges faster should be employed.
4. Produce a better flow performance index. Even though some improvements regarding the definition and computation of performance index were made, it is still limited in some cases. A performance index that has ability to handle real 3D geometry and multiple vents is necessary.

APPENDIX A

MATLAB CODES

1. The following codes are for parameter estimations:

```
clear all;
load racedata.txt;
X=racedata;
mx=mean(X);
varx=var(X);
% hist(X,12);
```

% The following codes are for gamma dist.

```
for i=1:92
    x(i)=log(X(i));
    x2(i)=x(i)^2;
end
M=log(mx)-sum(x)/92;
T=inv(log(mx)-sum(x)/92);
```

%The following codes are for weibull dist.

```
a(1)=((6/pi^2)*(sum(x2)-(sum(x))^2/92)/91)^(-0.5);
A=sum(x)/92;
for jj=1:8
    for j=1:92
        xa(j)=X(j)^a(jj);
        xal(j)=(X(j)^a(jj))*(log(X(j)));
        xah(j)=(X(j)^a(jj))*((log(X(j)))^2);
    end
    B(jj)=sum(xa);C(jj)=sum(xal);H(jj)=sum(xah);
    a(jj+1)=a(jj)+(A+1/a(jj)-C(jj)/B(jj))/(1/a(jj)^2+(B(jj)*H(jj)-C(jj)^2)/(B(jj)^2));
end
for i=1:92
    xb(i)=X(i)^a(6);
end
b=(sum(xb)/92)^(1/a(6));
b1=1/(b^a(6));
```

```
[WA, WB]=weibfit(X);
```

%The following codes are for lognormal dist.

```
mu=sum(x)/92;  
for i=1:92  
    xlv(i)=(x(i)-mu)^2;  
end  
varln=(sum(xlv)/92)^0.5;
```

2. The following are for input parameters generation:

```
clear all;  
fid = fopen('racedatawei.txt','w');  
for i=1:1200  
    d(i)=random('Weibull',0.0355,1.638);  
    while (d(i)>40|d(i)<1.1)  
        d(i)=random('Weibull',0.0355,1.638);  
    end  
    ds(i)=d(i)/0.98;  
    fprintf(fid,'%6.2f\t',d(i));  
end  
end  
hist(ds);title('Weibull distribution');  
fclose(fid);  
mean(d)  
var(d)
```

Note: the formulas for density function of Weibull distribution are different between those used in subsection 3.2 and MATLAB. The transformation is given below:

Formula in subsection 3.2: *Weibull* (α, β) \rightarrow *Weibull* (1.638, 7.6829)

Formula in MATLAB: *Weibull* (a, b) \rightarrow *Weibull* (0.0355, 1.638)

$$a = \frac{1}{\beta^\alpha}$$
$$b = \alpha$$

3. The following codes are for the goodness-of-fit test:

```
clear all;  
load racedata.txt;  
X=racedata;
```

```

mx=mean(X)
varx=var(X)
[a,b]=hist(X,12);
g(1)=gamcdf(b(1),2.759,2.48);
w(1)=weibcdf(b(1),0.0355,1.638);
gkq=0;wkq=0;
for i=2:12
    g(i)=gamcdf(b(i),2.759,2.48)-gamcdf(b(i-1),2.759,2.48);
    w(i)=weibcdf(b(i),0.0355,1.638)-weibcdf(b(i-1),0.0355,1.638);
end
gg=92*g;ww=92*w;
for j=1:12
    gkq=gkq+((a(j)-gg(j))^2)/gg(j);
    wkq=wkq+((a(j)-ww(j))^2)/ww(j);
end
gkq
meang=2.759*2.48
varg=2.48*2.759*2.759
wkq
meanw=(7.6829/1.638)*gamma(1/1.638)
varw=((7.6829^2)/1.638)*(2*gamma(2/1.638)-(1/1.638)*(gamma(1/1.638))^2)

```

APPENDIX B

VISUAL C++ CODES

Due to the complexity of the whole program, it is impossible to list all the codes for RTMSim. If need, please contact the author.

1. The following codes are for building OpenGL environment (SetWindowPixelFormat, RenderScene and OnSize):

//The following codes is for SetWindowPixelFormat function.

```
bool CGeodisplaywnddlg::SetWindowPixelFormat(HDC hDC)
{
    PIXELFORMATDESCRIPTOR pixelDesc;

    pixelDesc.nSize = sizeof(PIXELFORMATDESCRIPTOR);
    pixelDesc.nVersion = 1;

    pixelDesc.dwFlags = PFD_DRAW_TO_WINDOW |
                       PFD_SUPPORT_OPENGL |
                       PFD_DOUBLEBUFFER |
                       PFD_STEREO_DONTCARE;

    pixelDesc.iPixelFormat = PFD_TYPE_RGBA;
    pixelDesc.cColorBits = 32;
    pixelDesc.cRedBits = 8;
    pixelDesc.cRedShift = 16;
    pixelDesc.cGreenBits = 8;
    pixelDesc.cGreenShift = 8;
    pixelDesc.cBlueBits = 8;
    pixelDesc.cBlueShift = 0;
    pixelDesc.cAlphaBits = 0;
    pixelDesc.cAlphaShift = 0;
    pixelDesc.cAccumBits = 64;
```

```

pixelDesc.cAccumRedBits = 16;
pixelDesc.cAccumGreenBits = 16;
pixelDesc.cAccumBlueBits = 16;
pixelDesc.cAccumAlphaBits = 0;
pixelDesc.cDepthBits = 32;
pixelDesc.cStencilBits = 8;
pixelDesc.cAuxBuffers = 0;
pixelDesc.iLayerType = PFD_MAIN_PLANE;
pixelDesc.bReserved = 0;
pixelDesc.dwLayerMask = 0;
pixelDesc.dwVisibleMask = 0;
pixelDesc.dwDamageMask = 0;

m_GLPixelFormat = ChoosePixelFormat(hDC,&pixelDesc);
if(m_GLPixelFormat==0) // Choose default
{
    m_GLPixelFormat = 1;
    if(!DescribePixelFormat(hDC,m_GLPixelFormat,sizeof(PIXELFORMATDESCRIPTOR),
&pixelDesc))
        return FALSE;
}

if(!SetPixelFormat(hDC,m_GLPixelFormat,&pixelDesc))
    return FALSE;

return TRUE;

}

```

// The following codes is for RenderScene function.

```

void CGeodisplaywnddlg::RenderScene()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glPushMatrix();

    // Position / translation (mouse rotation)
    glTranslated(m_xTranslate/10,m_yTranslate/10,0.0);
// glTranslated(0.0,m_yTranslate,0.0);
    glRotated(m_xRotate, 1.0, 0.0, 0.0);
    glRotated(m_yRotate, 0.0, 1.0, 0.0);
    glScalef(m_ScaleX,m_ScaleY,m_ScaleZ);

    int mode;
}

```

```

glGetIntegerv(GL_RENDER_MODE,&mode);
if (mode==GL_SELECT)
{

glInitNames();
glPushName(0);
glPointSize(10.0f);
for(int i=1;i<=nodenum;i++)
    {
        glLoadName(i);
        glPushMatrix();

        glBegin(GL_POINTS);
        glColor3d(0.5,0.0,0.5);
        glVertex3d(m_Nodeset[i][1],m_Nodeset[i][2], m_Nodeset[i][3]);
        glEnd();
        glPopMatrix();
    }
    int numm;
    glGetIntegerv(GL_NAME_STACK_DEPTH ,&numm);

}

else
{
// wireframe model with line shown
if(displaymode==1)
    {
        ::glCallList(1);
    }
//wireframe model with hidden line
if(displaymode==2)
    {
        glEnable(GL_POLYGON_OFFSET_FILL);
        glPolygonOffset(1.0,1.0);
        glColor3ub(0,0,0);
        ::glCallList(2);
        glDisable(GL_POLYGON_OFFSET_FILL);

        glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
        glColor3ub(0,0,255);
        ::glCallList(2);
        glPolygonMode(GL_FRONT_AND_BACK,GL_FILL);
    }
}

```

```

    }
    //solid model
    if(displaymode==3)
    {
// glEnable(GL_LIGHT0);
// glEnable(GL_LIGHTING);
    glEnable(GL_POLYGON_OFFSET_FILL);
        glPolygonOffset(1.0,1.0);
        glColor3ub(0,200,200);
        ::glCallList(2);
        glDisable(GL_POLYGON_OFFSET_FILL);

// glDisable(GL_LIGHT0);
// glDisable(GL_LIGHTING);

        glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
        glColor3ub(0,0,255);
        ::glCallList(2);
        glPolygonMode(GL_FRONT_AND_BACK,GL_FILL);
    }
//result pressure contour plot
if(displaymode==4||displaymode==5||displaymode==7||displaymode==8)
{
    ::glCallList(3);
    CRect windowreg;
    GetClientRect (&windowreg);
    CRect redrawreg;
    redrawreg.left=windowreg.left;redrawreg.bottom=windowreg.bottom;
    redrawreg.top=windowreg.top;redrawreg.right=windowreg.right-32;
        InvalidateRect(redrawreg,FALSE);
}
//flow animation
if(displaymode==6)
{

    glEnable(GL_POLYGON_OFFSET_FILL);
    glPolygonOffset(1.0,1.0);
    glColor3ub(0,0,0);
    ::glCallList(2);
    glDisable(GL_POLYGON_OFFSET_FILL);

    glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
    glColor3ub(0,0,255);
    ::glCallList(2);
    glPolygonMode(GL_FRONT_AND_BACK,GL_FILL);
}

```

```

//the following codes read animation information
    CString aninode=((CRtmApp*)AfxGetApp())-
>home+"\\Aniresults\\"+"ani"+((CRtmApp*)AfxGetApp())->Partname+".nod";
    CString aniele=((CRtmApp*)AfxGetApp())-
>home+"\\Aniresults\\"+"ani"+((CRtmApp*)AfxGetApp())->Partname+".ele";
    FILE *tempaninode;//in is the pointer of boundary cond file
    tempaninode=fopen(aninode,"r");
    FILE *tempaniele;//in is the pointer of boundary cond file
    tempaniele=fopen(aniele,"r");
////////////////////////////////////
    int aniresele=((CRtmApp*)AfxGetApp())->result.reselenium-1;
    int m_AniContourReseaset[400][8];
    float m_AniContourResnodset[1500][4];
        for (int mmm=1;mmm<aniresele+1;mmm++)
            {
                fscanf(tempaniele,"%d\t",&((CRtmApp*)AfxGetApp())-
>result.m_ContourReseaset[mmm][0]);
                for(int jj=1;jj<((CRtmApp*)AfxGetApp())-
>result.m_ContourReseaset[mmm][0]+1;jj++)
                    {
                        fscanf(tempaniele,"%d\t",&((CRtmApp*)AfxGetApp())-
>result.m_ContourReseaset[mmm][jj]);
                    }
                fscanf(tempaniele,"\n");
            }
        int mm=1;
        int test=0;
    while(test==0)
        {
            test=feof(tempaninode);
            if(test!=0)
                break;
            fscanf(tempaninode,"%f\t%f\t%f\t%f\n",&((CRtmApp*)AfxGetApp())-
>result.m_ContourResnodset[mm][0],
                &((CRtmApp*)AfxGetApp())-
>result.m_ContourResnodset[mm][1],
                &((CRtmApp*)AfxGetApp())-
>result.m_ContourResnodset[mm][2],
                &((CRtmApp*)AfxGetApp())-
>result.m_ContourResnodset[mm][3]);
            mm=mm++;
        }
    fclose(tempaninode);
    fclose(tempaniele);
    ::glCallList(4);

```

```

SetTimer(0,300,NULL);
CRect windowreg;
GetClientRect (&windowreg);
CRect redrawreg;
redrawreg.left=windowreg.left;redrawreg.bottom=windowreg.bottom;
redrawreg.top=windowreg.top;redrawreg.right=windowreg.right-32;
        InvalidateRect(redrawreg,FALSE);
        if(aninum>15)
            KillTimer(0);
    }

}
glPopMatrix();
}

```

//The following codes are for OnSize function.

```

void CGeodisplaywnddlg::OnSize(UINT nType, int cx, int cy)
{
    CDialog::OnSize(nType, cx, cy);

    // TODO: Add your message handler code here
    GLsizei width,height;
    GLdouble aspect;

    // width = cx;
    // height = cy;
    width=((CRtmApp*)AfxGetApp()->Clientwind.Width()-182;
    height=((CRtmApp*)AfxGetApp()->Clientwind.Height()-4;

    if(cy==0)
        aspect = (GLdouble)width;
    else
        aspect = (GLdouble)width/(GLdouble)height;

    glViewport(0,0,width,height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // gluPerspective(45,aspect,1,10.0);
    glOrtho(-width/60,width/60,-height/60,height/60,0,3000);
    /*
    // the following codes determine the initial window size
    float m_xwin=(m_MaxNodeset[1][1]-m_MaxNodeset[1][2])*1.4;
    float m_ywin=(m_MaxNodeset[2][1]-m_MaxNodeset[2][2])*1.4;

```

```

float m_zwin=(m_MaxNodeset[3][1]-m_MaxNodeset[3][2])*1.4;
      glOrtho(-(width/height)*(m_ywin/2),(width/height)*m_ywin/2,-
(m_ywin/2),(m_ywin/2),0,3000);
*/
/* gluLookAt(WindowDef.eye[0],WindowDef.eye[1],WindowDef.eye[2],
      WindowDef.center[0],WindowDef.center[1],WindowDef.center[2],
      WindowDef.up[0],WindowDef.up[1],WindowDef.up[2]);
*/
gluLookAt(0.,0.,400.,0.,0.,0.,0.,1.,0.);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

glDrawBuffer(GL_BACK);

// Lights, material properties
GLfloat ambientProperties[] = {0.7f, 0.7f, 0.7f, 1.0f};
GLfloat diffuseProperties[] = {0.8f, 0.8f, 0.8f, 1.0f};
GLfloat specularProperties[] = {1.0f, 1.0f, 1.0f, 1.0f};

glLightfv( GL_LIGHT0, GL_AMBIENT, ambientProperties);
glLightfv( GL_LIGHT0, GL_DIFFUSE, diffuseProperties);
glLightfv( GL_LIGHT0, GL_SPECULAR, specularProperties);
glLightModel(GL_LIGHT_MODEL_TWO_SIDE, 1.0);
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

// Default : lighting
// glEnable(GL_LIGHT0);
// glEnable(GL_LIGHTING);

glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, diffuseProperties);

glEnable(GL_DEPTH_TEST);

}

```

2. The following codes are for Implementation of contouring algorithm:

```

int CResultdlg::ColorIndex(double max, double val)
{
    for(int ind=1;ind<=16;ind++)
    {
        if((max/15)*(ind)>=val&&val>=(max/15)*(ind-1))
            return 16-ind;
    }
}

```

```

    }
}
//m_CountourResnodset 0\colorindex 1\x 2\y 3\z
//the following codes calculate the points which have the same value and group result element
void CResultdlg::Contouring(double maxval, int elenumber[],int elenum,CString restype)
{
    int j=1;int es;int ee;
    //the following codes calculate the points which have the same value
    int t=1;float tempnode[5][5];double intsvale;
    es=n;
    while(t!=elenum+1)
    {
        if (restype=="pressure")
        {
            m_CountourResnodset[n][0]=m_Rescolorind[elenumber[t]][1];
            tempnode[t][0]=m_Rescolorind[elenumber[t]][1];
            m_CountourResnodset[n][4]=m_Resultset[elenumber[t]][1];
            tempnode[t][4]=m_Resultset[elenumber[t]][1];
        }
        if (restype=="time")
        {
            m_CountourResnodset[n][0]=m_Rescolorind[elenumber[t]][2];
            tempnode[t][0]=m_Rescolorind[elenumber[t]][2];
            m_CountourResnodset[n][4]=m_Resultset[elenumber[t]][2];
            tempnode[t][4]=m_Resultset[elenumber[t]][2];
        }
        if (restype=="permeability")
        {
            m_CountourResnodset[n][0]=m_Rescolorind[elenumber[t]][3];
            tempnode[t][0]=m_Rescolorind[elenumber[t]][3];
            m_CountourResnodset[n][4]=m_Resultset[elenumber[t]][3];
            tempnode[t][4]=m_Resultset[elenumber[t]][3];
        }
        if (restype=="airpressure")
        {
            m_CountourResnodset[n][0]=m_Rescolorind[elenumber[t]][4];
            tempnode[t][0]=m_Rescolorind[elenumber[t]][4];
            m_CountourResnodset[n][4]=m_Resultset[elenumber[t]][4];
            tempnode[t][4]=m_Resultset[elenumber[t]][4];
        }
    }

    m_CountourResnodset[n][1]=((CRtmApp*)AfxGetApp())->geodiswnd.m_Nodeset[elenumber[t]][1];
}

```

```

    m_ContourResnodset[n][2]=((CRtmApp*)AfxGetApp()-
>geodiswnd.m_Nodeset[elenumbr[t]][2];
    m_ContourResnodset[n][3]=((CRtmApp*)AfxGetApp()-
>geodiswnd.m_Nodeset[elenumbr[t]][3];
    tempnode[t][1]=((CRtmApp*)AfxGetApp()->geodiswnd.m_Nodeset[elenumbr[t]][1];
    tempnode[t][2]=((CRtmApp*)AfxGetApp()->geodiswnd.m_Nodeset[elenumbr[t]][2];
    tempnode[t][3]=((CRtmApp*)AfxGetApp()->geodiswnd.m_Nodeset[elenumbr[t]][3];
        n++;t++;
    }
    int ts=1;int te=2;
    while (ts!=elenum+1)
    {
        if(abs(tempnode[ts][0]-tempnode[te][0])!=0)
            {
                double startingpt[5];double endingpt[5];

                startingpt[1]=tempnode[ts][1];startingpt[2]=tempnode[ts][2];startingpt[3]=tempnode[ts][
3];
                    startingpt[0]=tempnode[ts][0];startingpt[4]=tempnode[ts][4];

                endingpt[1]=tempnode[te][1];endingpt[2]=tempnode[te][2];endingpt[3]=tempnode[te][3];
                    endingpt[0]=tempnode[te][0];endingpt[4]=tempnode[te][4];
                for (int tt=1;tt<=abs(tempnode[ts][0]-tempnode[te][0]);tt++)
                    {

                        if(tempnode[ts][0]>tempnode[te][0])
                            intsvale=(maxval/15.0)*(15+tt-tempnode[ts][0]);
                        else
                            intsvale=(maxval/15.0)*(16-tt-tempnode[ts][0]);
                    double nodecord[4];
                    Linintepo(startingpt,endingpt,intsvale,nodecord);

                    m_ContourResnodset[n][0]=startingpt[0];
                    m_ContourResnodset[n][1]=nodecord[1];
                    m_ContourResnodset[n][2]=nodecord[2];
                    m_ContourResnodset[n][3]=nodecord[3];

//////////////////////////////////////
                    bool notin=true;int ncline;
                        // store the contouring line information
                            if(startingpt[0]<endingpt[0])
                                ncline=15-int(startingpt[0]);
                            else
                                ncline=16-int(startingpt[0]);
                        for (int jj=1;jj<=contorder[ncline];jj++)
                            {

```

```

float xx=nodecord[1];
float yy=nodecord[2];
float zz=nodecord[3];
    float ttx1=m_ContourResnodset[contnoed[ncline][jj]][1];
    float tty1=m_ContourResnodset[contnoed[ncline][jj]][2];
    float ttx=fabs(xx-m_ContourResnodset[contnoed[ncline][jj]][1]);
    float tty=fabs(yy-m_ContourResnodset[contnoed[ncline][jj]][2]);
    if (fabs(xx-m_ContourResnodset[contnoed[ncline][jj]][1])<0.00001&&
        fabs(yy-m_ContourResnodset[contnoed[ncline][jj]][2])<0.00001&&
        fabs(zz-m_ContourResnodset[contnoed[ncline][jj]][3])<0.00001)
        {
            notin=false;
        }
    }
if(notin==true)
    {
if(startingpt[0]<endingpt[0])
    {
        contnoed[15-int(startingpt[0])][contorder[15-int(startingpt[0])]=n;
        contorder[15-int(startingpt[0])]=contorder[15-int(startingpt[0])+1];
    }
    else
        {
        contnoed[16-int(startingpt[0])][contorder[16-int(startingpt[0])]=n;
        contorder[16-int(startingpt[0])]=contorder[16-int(startingpt[0])+1];
        }
    }

////////////////////////////////////
if(startingpt[0]>endingpt[0])
{startingpt[0]=startingpt[0]-1;}
else
{startingpt[0]=startingpt[0]+1;}

startingpt[4]=intsvalue;
startingpt[1]=nodecord[1];startingpt[2]=nodecord[2];startingpt[3]=nodecord[3];
n++;
m_ContourResnodset[n][0]=startingpt[0];
m_ContourResnodset[n][1]=nodecord[1];
m_ContourResnodset[n][2]=nodecord[2];
m_ContourResnodset[n][3]=nodecord[3];
n++;
    }
}

ts++;
if(ts!=elenum)

```

```

        te++;
    else
        te=1;
    }
    //merge the same point

    //////////
    //group result element
    ee=n-1;int maxcol=1;int mincol=15;int tempes=es;
    for(int m=1;m<=(ee-es+1);m++)
        {
            if(maxcol<=m_ContourResnodset[tempes][0])
                maxcol=m_ContourResnodset[tempes][0];
            if(mincol>=m_ContourResnodset[tempes][0])
                mincol=m_ContourResnodset[tempes][0];

        }
    tempes++;
    }
    tempes=es;
    for (int th=1;th<=(ee-es+1)/elenum;th++)
        {
            m_ContourReseaset[reselenum][0]=0;
            int tempelenum=1;
            while(tempes<=ee)
                {
                    if (mincol==m_ContourResnodset[tempes][0])
                        {
                            m_ContourReseaset[reselenum][tempelenum]=tempes;

                            m_ContourReseaset[reselenum][0]=m_ContourReseaset[reselenum][0]+1;
                            tempelenum++;
                        }
                    tempes++;
                }
            mincol++;
            reselenum++;
            tempes=es;
    //renumber the nodes in element
    //the following code are for 4 nodes case
        if(tempelenum==5)
            {
                float spoint[4];float epoint[4];float testpoint[4];
                int tempelenode[7];
                tempelenode[1]=m_ContourReseaset[reselenum-1][1];
                for (int i=1;i<=3;i++)
                    {

```

```

    spoint[1]=m_ContourResnodset[tempelenode[i]][1];
    spoint[2]=m_ContourResnodset[tempelenode[i]][2];
    spoint[3]=m_ContourResnodset[tempelenode[i]][3];
    bool validpk=false;int validnode;int pk=1;
while(!validpk)
    {
        int count=10;
        while(count!=0)
            {count=0;
            for(int iii=1;iii<=4;iii++)
                {
                    if(m_ContourReseaset[reselenum-
1][pk]==tempelenode[iii])
                        {
                            count=count+1;
                        }
                }
            if (count!=0)
                pk++;
            }
        tempelenode[i+1]=m_ContourReseaset[reselenum-1][pk];
        epoint[1]=m_ContourResnodset[tempelenode[i+1]][1];
        epoint[2]=m_ContourResnodset[tempelenode[i+1]][2];
        epoint[3]=m_ContourResnodset[tempelenode[i+1]][3];
        validnode=pk;

        float nveang[6][4];int jj=1;
        for(int j=1;j<=4;j++)
            {
                testpoint[1]=m_ContourResnodset[m_ContourReseaset[reselenum-1][j]][1];
                testpoint[2]=m_ContourResnodset[m_ContourReseaset[reselenum-1][j]][2];
                testpoint[3]=m_ContourResnodset[m_ContourReseaset[reselenum-1][j]][3];
                if(tempelenode[i]!=m_ContourReseaset[reselenum-
1][j]&&
tempelenode[i+1]!=m_ContourReseaset[reselenum-1][j])
                    {float tempang[4];
//                    nveang[jj]=Normvecang(spoint,epoint,testpoint);
                    Normvecang(spoint,epoint,testpoint,tempang);
                    nveang[jj][1]=tempang[1];
                    nveang[jj][2]=tempang[2];
                    nveang[jj][3]=tempang[3];
                    jj++;
                    }
            }
    }

```

```

        if(abs(nveang[1][1]-nveang[2][1])<=0.0001&&abs(nveang[1][2]-
nveang[2][2])<=0.0001&&abs(nveang[1][3]-nveang[2][3])<=0.0001)
            {
                validpk=true;pk=1;
                break;
            }
        else
            {
                pk++;
            }
    }

    for(int nn=1;nn<=4;nn++)
        {
            m_ContourReseaset[reselenium-1][nn]=tempelenode[nn];
        }
}
//the following code are for 5 nodes case
if(tempelenium==6)
    {
        float spoint[4];float epoint[4];float testpoint[4];
        int tempelenode[7];
        tempelenode[1]=m_ContourReseaset[reselenium-1][1];
        for (int i=1;i<=4;i++)
            {
                spoint[1]=m_ContourResnodset[tempelenode[i]][1];
                spoint[2]=m_ContourResnodset[tempelenode[i]][2];
                spoint[3]=m_ContourResnodset[tempelenode[i]][3];
                bool validpk=false;int validnode;int pk=1;
                while(!validpk)
                    {
                        int count=10;
                        while(count!=0)
                            {count=0;
                            for(int iii=1;iii<=5;iii++)
                                {
                                    if(m_ContourReseaset[reselenium-
1][pk]==tempelenode[iii]&&pk<8)
                                        {
                                            count=count+1;
                                        }
                                }
                            }
                    }
    }

```

```

        if (count!=0)
            pk++;
        }
tempelenode[i+1]=m_ContourReseleset[reselenum-1][pk];
epoint[1]=m_ContourResnodset[tempelenode[i+1]][1];
epoint[2]=m_ContourResnodset[tempelenode[i+1]][2];
epoint[3]=m_ContourResnodset[tempelenode[i+1]][3];
validnode=pk;

float nveang[6][4];int jj=1;
for(int j=1;j<=5;j++)
{
testpoint[1]=m_ContourResnodset[m_ContourReseleset[reselenum-1][j]][1];
testpoint[2]=m_ContourResnodset[m_ContourReseleset[reselenum-1][j]][2];
testpoint[3]=m_ContourResnodset[m_ContourReseleset[reselenum-1][j]][3];
if(tempelenode[i]!=m_ContourReseleset[reselenum-
1][j]&&
tempelenode[i+1]!=m_ContourReseleset[reselenum-1][j])
{float tempang[4];
// nveang[jj]=Normvecang(spoint,epoint,testpoint);
Normvecang(spoint,epoint,testpoint,tempang);
nveang[jj][1]=tempang[1];
nveang[jj][2]=tempang[2];
nveang[jj][3]=tempang[3];
jj++;
}
}
if(abs(nveang[1][1]-nveang[2][1])<=0.0001&&abs(nveang[1][1]-
nveang[3][1])<=0.0001&&abs(nveang[3][1]-nveang[2][1])<=0.0001&&
abs(nveang[1][2]-
nveang[2][2])<=0.0001&&abs(nveang[1][2]-nveang[3][2])<=0.0001&&abs(nveang[3][2]-
nveang[2][2])<=0.0001&&
abs(nveang[1][3]-
nveang[2][3])<=0.0001&&abs(nveang[1][3]-nveang[3][3])<=0.0001&&abs(nveang[3][3]-
nveang[2][3])<=0.0001)
{
validpk=true;pk=1;
break;
}
else
{
pk++;
}
}
}

```

```

for(int nn=1;nn<=5;nn++)
    {
        m_ContourReseaset[reselenium-1][nn]=tempelenode[nn];
    }
}
//the following code are for 6 nodes case
if(tempelenium==7)
    {
        float spoint[4];float epoint[4];float testpoint[4];
int tempelenode[7];
    tempelenode[1]=m_ContourReseaset[reselenium-1][1];
for (int i=1;i<=5;i++)
    {
        spoint[1]=m_ContourResnodset[tempelenode[i]][1];
        spoint[2]=m_ContourResnodset[tempelenode[i]][2];
        spoint[3]=m_ContourResnodset[tempelenode[i]][3];
        bool validpk=false;int validnode;int pk=1;
while(!validpk)
    {
        int count=10;
        while(count!=0)
            {count=0;
            for(int iii=1;iii<=6;iii++)
                {
                    if(m_ContourReseaset[reselenium-
1][pk]==tempelenode[iii])
                        {
                            count=count+1;
                        }
                }
            if (count!=0)
                pk++;
        }
        tempelenode[i+1]=m_ContourReseaset[reselenium-1][pk];
        epoint[1]=m_ContourResnodset[tempelenode[i+1]][1];
        epoint[2]=m_ContourResnodset[tempelenode[i+1]][2];
        epoint[3]=m_ContourResnodset[tempelenode[i+1]][3];
        validnode=pk;
        float nveang[6][5];int jj=1;
        for(int j=1;j<=6;j++)
            {
                testpoint[1]=m_ContourResnodset[m_ContourReseaset[reselenium-1][j]][1];
                testpoint[2]=m_ContourResnodset[m_ContourReseaset[reselenium-1][j]][2];
                testpoint[3]=m_ContourResnodset[m_ContourReseaset[reselenium-1][j]][3];
            }
    }
}

```

```

        if(tempelenode[i]!=m_ContourReseaset[reselenium-
1][j]&&
        tempelenode[i+1]!=m_ContourReseaset[reselenium-1][j])
            {float tempang[4];
            Normvecang(spoin,epoin,testpoint,tempang);
            nveang[jj][1]=tempang[1];
            nveang[jj][2]=tempang[2];
            nveang[jj][3]=tempang[3];
            jj++;
            }
        }
    if(abs(nveang[1][1]-nveang[2][1])<=0.0001&&abs(nveang[1][1]-
nveang[3][1])<=0.0001&&
        abs(nveang[3][1]-
nveang[2][1])<=0.0001&&abs(nveang[1][1]-nveang[4][1])<=0.0001&&
        abs(nveang[4][1]-
nveang[2][1])<=0.0001&&abs(nveang[3][1]-nveang[4][1])<=0.0001&&
        abs(nveang[1][2]-
nveang[2][2])<=0.0001&&abs(nveang[1][2]-nveang[3][2])<=0.0001&&
        abs(nveang[3][2]-
nveang[2][2])<=0.0001&&abs(nveang[1][2]-nveang[4][2])<=0.0001&&
        abs(nveang[4][2]-
nveang[2][2])<=0.0001&&abs(nveang[3][2]-nveang[4][2])<=0.0001&&
        abs(nveang[1][3]-
nveang[2][3])<=0.0001&&abs(nveang[1][3]-nveang[3][3])<=0.0001&&
        abs(nveang[3][3]-
nveang[2][3])<=0.0001&&abs(nveang[1][3]-nveang[4][3])<=0.0001&&
        abs(nveang[4][3]-
nveang[2][3])<=0.0001&&abs(nveang[3][3]-nveang[4][3])<=0.0001)
        {
            validpk=true;pk=1;
            break;
        }
    else
        {
            pk++;
        }
    }
}
for(int nn=1;nn<=6;nn++)
    {
        m_ContourReseaset[reselenium-1][nn]=tempelenode[nn];
    }
}
}

```

```

}

void CResultdlg::Linintepo(double p1[], double p2[],double v,double nodecords[])
{
    double cof=fabs((p1[4]-v)/(p1[4]-p2[4]));
    nodecords[1]=p1[1]+(p2[1]-p1[1])*cof;
    nodecords[2]=p1[2]+(p2[2]-p1[2])*cof;
    nodecords[3]=p1[3]+(p2[3]-p1[3])*cof;
}

float CResultdlg::Distance(float ps[], float pe[])
{
    return sqrt((ps[1]-pe[1])*(ps[1]-pe[1])+(ps[2]-pe[2])*(ps[2]-pe[2])+(ps[3]-pe[3])*(ps[3]-pe[3]));
}

float CResultdlg::Ang(float pc[], float p1[], float p2[])
{
    double cp1=Distance(pc,p1);
    double cp2=Distance(pc,p2);
    double p1p2=Distance(p1,p2);
    double cosv=(cp1*cp1+cp2*cp2-p1p2*p1p2)/(2*cp1*cp2);
    return (acos(cosv));
}

void CResultdlg::Normvecang(float ve1[], float ve2[], float ve3[], float angs[])
{
    float a[4]; float b[4];float crossve[4];
    a[1]=ve2[1]-ve1[1];a[2]=ve2[2]-ve1[2];a[3]=ve2[3]-ve1[3];
    b[1]=ve3[1]-ve1[1];b[2]=ve3[2]-ve1[2];b[3]=ve3[3]-ve1[3];
    crossve[1]=a[2]*b[3]-a[3]*b[2];crossve[2]=a[3]*b[1]-a[1]*b[3];crossve[3]=a[1]*b[2]-a[2]*b[1];
    double
    cosvec1=crossve[1]/sqrt(crossve[1]*crossve[1]+crossve[2]*crossve[2]+crossve[3]*crossve[3]);
    double
    cosvec2=crossve[2]/sqrt(crossve[1]*crossve[1]+crossve[2]*crossve[2]+crossve[3]*crossve[3]);
    double
    cosvec3=crossve[3]/sqrt(crossve[1]*crossve[1]+crossve[2]*crossve[2]+crossve[3]*crossve[3]);
    angs[1]=acos(cosvec1);
    angs[2]=acos(cosvec2);
    angs[3]=acos(cosvec3);
}

```

REFERENCES

- [1] <https://www.ccm.udel.edu/reports-pubs/tech-briefs/102.html>
- [2] <http://www.polyworx.com/doc/>
- [3] <http://www.esi-group.com/Products/RTM/>
- [4] Dong, C., Zhang, C., and Wang, B., "Integration of Green Quality Function Deployment and Fuzzy Multi-Attribute Utility Theory-Based Cost Estimation for Environmentally Conscious Product Development," *International Journal of Environmentally Conscious Design & Manufacturing*, 2002 p 12-28
- [5] Phelan Jr., F.R. "Simulation of the Injection Process in Resin Transfer Molding," *Polymer Composites* Aug., 1997, 18(4), p 460-476.
- [6] Spoerre, J., Zhang, C., Wang, B. and Parnas, R., "Integrated Product and Process Design for Resin Transfer Molded Parts," *Journal of Composite Materials* Vol 32 No.13, 1998, p 1244-1272
- [7] Ranganathan S., Easterling R.G., Advani S.G. and Phelan F.R. Jr., "Effect of Microstructure Variations on the Permeability of Preform Materials," *polymers & Polymer Composites*, v6, n2, Feb 1998, p 63-73
- [8] Pan R., Liang Z., Zhang C., and Wang B., "Statistical Characterization of Fiber Permeability for Composite Manufacturing," *Polymer Composites* Dec 2000, vol. 21, No. 6, p 996-1006
- [9] Kang Guo-Zheng, Yang C. and Zhang Ji-xi "Tensile Properties of Randomly Oriented Short $\delta - Al_2O_3$ Fiber Reinforced Aluminum Alloy Composites .I. Microstructure Characteristics, Fracture Mechanisms and Strength Prediction," *Composite. Part A* 33, 2002, p 647-656
- [10] Scott W. Beckwith and Craig R. Hyland, "Resin Transfer Molding (RTM) Technology Overview," *Composites Fabrication*, March 1998, p24-27
- [11] Gonzalez-Romero and Macosko C. W., "Process Parameters Estimation for Structural Reaction Injection Molding and Resin Transfer Molding," *Polymer*

Engineering and Science, Vol. 30 Mid-Feb., 1990, p 142-146

- [12] Chan, A.W., et al. "Molding of Impregnation Process During Resin Transfer Molding," *Polymer Engineering and Science*, Mid-Aug., Vol. 31, 1991, p 1149-1156
- [13] Lee, James, Young W.B., and Lin R.J., "Mold Filling and Cure Modeling of RTM and SCRIM Processes," *Composites Structure*, 27, 1994, p 109-120
- [14] Young Wen-Bin, "Three-dimensional Nonisothermal Mold Filling Simulations in Resin Transfer Molding," *Polymer Composites*, Vol. 15, No. 2, April 1994, p 118-127
- [15] Liu Baichen, Bickerton Simon, Advani Suresh G., "Modeling and Simulation of Resin Transfer Molding (RTM) – Gate Control, Venting and Dry Spot Prediction," *Composites. Part A*, Vol. 27, No. 2, 1996, p 135-141
- [16] Um MK and Lee WI, "A Study on Mold Filling Process in Resin Transfer Molding," *Polymer Engineering and Science*, Vol. 31, 1991, p 765-771
- [17] Yoo yeong-eun and Lee Woo II, "Numerical Simulation of the Resin Transfer Mold Filling Process Using Boundary Element Method," *Polymer Composites*, Vol. 17, No. 3, 1996, p 368-374
- [18] Osswald, T. and Tucker, C., "A Boundary Element Simulation of Compression Mold Filling," *Polymer Engineering and Science*, Vol. 28, No. 7, Mid-Apr. 1998, p 413-420
- [19] Brusckhe MV., "A Predictive Model for Permeability and Non-isothermal Flow of Viscous and Shear-thinning Fluids in Anisotropic Fibrous Media" CCM Report 92-56, 992
- [20] Varma RR., "Three-dimensional Simulations of Filling in Resin Transfer Molding" *Advances in Finite Element Analysis in Fluid Dynamics (ASME): FED* 1994, 200:21-7
- [21] Phelan Jr. FR. "Simulation of the Injection Process in Resin Transfer Molding" *Polymer Composites* 1997, 4(18), p 460-76
- [22] Joshi Sunil C., Lam Y.C. and Liu X.L., "Mass Conservation in Numerical Simulation of Resin Flow" *Composites. Part A*, Vol. 31, 2000, p 1061-1068
- [23] Ismail Y., Springer G.S., "Interactive Simulation of Resin Transfer Molding," *Journal of Composite Materials*, Vol. 31, No. 10, 1997, p 954-980
- [24] Kanapady R., Tamma K. K., Baddourah M. and Mark A., "High Performance

Computing on a Symmetric Multiprocessor (SMP) Environment for RTM Process Modeling of Large Complex Structural Geometries,” *Advances in Engineering Software*, Vol. 29, 3-6, 1998 p 399-408

- [25] Kanapady R., Tamma K., and Mark A., “Highly Scalable Parallel Computational Models for Large-scale RTM Process Modeling Simulation, Part 3: Validation and Performance Results,” *Numerical Heat Transfer, Part V: Fundamentals*, Vol. 36, 4, 1999 p 351-386
- [26] Lin M. and Hahn H.T., “Resin Transfer Molding Process Optimization,” *Composites. Part A*, Vol. 31, 2000 p 361-371
- [27] Lawrence Jeffrey M., Hsiao Kuang-Ting, Don Roderic C., Simacek Pavel, Estrada Gonzalo, Sozer E. Murat, Stadtfeld Hubert C. and Advani Suresh G. “An Approach to Couple Mold Design and On-line control to Manufacture Complex Composite Parts by Resin Transfer Molding,” *Composites. Part A*, Vol. 33, 2002 p 981-990
- [28] Jiang S., Zhang C. and Wang B., “Optimum Arrangement of Gate and Vent Locations for RTM Process Design Using a Mesh Distance-based Approach,” *Composites. Part A*, Vol. 33, 2002 p 471-481
- [29] Scheidegger A.E., *The Physics of Flow Through Porous Media*, MacMillan, New York (1974)
- [30] Adams K.L., *et al. Int. J. Multiphase Flow*, 14, 203 (1988)
- [31] Gauvin R. and Trochu F., “Key Issues in Numerical Simulation for Liquid Composite Molding Processes,” *Polymer Composites*, June 1998, vol. 19, No. 3, p 233-240
- [32] Gauvin, R., Trochu F., Lemenn Y., and Diallo L., “Permeability Measurement and Flow Simulation Through Fiber Reinforcement,” *Polymer Composites*, Feb. 1996, vol. 17, No. 1, p 34-42
- [33] Hammami A., Trochu F., Gauvin R. and Wirth S., “Directional Permeability Measurement of Deformed Reinforcement,” *Journal of Reinforced Plastics and Composites*, June 1996, vol. 15, p 552-562
- [34] Lekakou C., Johari M. A. K., Norman D. and Bader M. G., “Measurement Techniques and Effects on In-plane Permeability of Woven Cloths in Resin Transfer Molding” *Composites. Part A*, 1996, Vol. 27, Issue 5, p 401-408
- [35] Rechard S. Parnas, Kathleen M. Flynn, and Mary E. Dal Favero, “A Permeability Database for Composites Manufacturing,” *Polymer Composites*, Oct. 1997, Vol. 18, No. 5, p 623-632

- [36] Carter EJ, Fell AW, Griffin PR and Summerscales J., "Data Validation Procedures for the Automated Determination of the Two-dimensional Permeability Tensor of a Fabric Reinforcement," *Composites. Part A*, 1996, Vol. 27, Issue. 4, p 255-261
- [37] Ferland P., Guitard D., and Trochu F., "Concurrent Methods for Permeability Measurement in Resin Transfer Molding," *Polymer Composites*, Feb. 1996, Vol. 17, No. 1, p 149-158
- [38] Weitzenbock J. R., Sheno, R.A., Wilson, P.A., "Measurement of Three-dimensional Permeability," *Composites Part A*, 1998, Vol. 29A, p 159-169
- [39] Hammami A., Gauvin R., Trochu F., Truret O. and Ferland P., "Analysis of the Edge Effect on Flow Patterns in Liquid Composites Molding," *Applied Composite Materials 5*: 1998, p 161-173
- [40] Bickerton S., Advani S.G., Mohan R.V. and Shires D.R., "Experimental Analysis and Numerical Modeling of Flow Channel Effects in Resin Transfer Molding," *Polymer Composites*, Feb. 2000, Vol 21, No. 1, p 134-153
- [41] Klaas O. and Shephard M.S., "Embedding Reliable Numerical Analysis Capabilities into an Enterprise-wide Information System," *Engineering with Computers* 2001 17, p 151-161
- [42] Lu S.C., Saran M.J. and Miller R.A., "Integration of CAD and FEA for Concurrent Engineering Design of Sheet Stamping," *Transactions of the ASME*, Vol. 118, Aug. 1996, p 310-317
- [43] Gerda Gemser, Mark A. A. M. Leenders, "How Integrating Industrial Design in the Product Development Process Impacts on Company Performance," *The Journal of Product Innovation Management*, 18 (2001) p 28-38
- [44] C. Ou-yang and Jiang T.A., "Developing an Integration Framework to Support the Information Flow between PDM and MRP," *International Journal of Advance Manufacturing technology*, 2002 19, p 131-141
- [45] Bchenek G. M., Rangusea J. M. and Malone L. C., "Integration Virtual 3-D Display Systems into Product Design Reviews: Some Insights from Empirical Testing," *International Journal of Technology Management*, Vol. 21, 2001, p 340-352
- [46] Ismail Y. M. and Springer G. S., "Interactive Simulation of Resin Transfer Molding" *Journal of Composite Materials*, Vol. 31, No. 10, 1997, p 954-980
- [47] Charles J. Bontempo and Cynthia Maro Saracco, *Database Management Principles and Products*, Prentice Hall PTR, 1996
- [48] David J. Gruglinski, *Inside Visual C++ Fourth Edition*, Microsoft Press, 1997

- [49] Mike Blaszcak, *Professional MFC with Visual C++ 5*, Wrox Press, 1997
- [50] Richard S. Wright, Jr. and Michael Sweet, *OpenGL Super Bible Second Edition*, Waite Group Press, 1999
- [51] OpenGL Architecture Review board, Editor: Dave Shreiner, *OpenGL Reference Manual third Edition The Official Reference Document to OpenGL, Version 1.2*, Addison Wesley Press, 2002
- [52] <http://www.opengl.org>
- [53] Luo Yiwen, Verpoest Ignaas, Hoes Kris, Vanheule Marleen, Sol Hugo and Cardon Albert, "Permeability Measurement of Textile Reinforcements with Several Test Liquids," *Composites. Part A*, 32 (2001), p 1797-1504
- [54] Steenmaker D.A. *et al.*, "Experimental Characterization of Permeability and Fiber Wetting for Liquid Moulding," *Journal of Material Science*, 1995, 30, p 3207-3215
- [55] Hammond VH, and Loos AC, "The Effects of Fluid Type and Viscosity on the Steady-state and Advancing Front Permeability Behavior of Textile Preforms," *Journal of Reinforced Plastic Composite*, 1997, 16(1), p 50-72
- [56] Padmanabhan S.K. and Pitchumani R., "Stochastic Modeling of Nonisothermal Flow During Resin Transfer Molding," *International Journal of Heat and Mass Transfer*, 42 (1999) p 3057-3070
- [57] Averill M. Law and W. David Kelton, *Simulation Modeling & Analysis Second Edition*, McGRAW-Hill International Editions, 1991
- [58] Dennis D. Wackerly, William Mendenhall III and rechar L. Scheaffer, *Mathematical Statistics with Applications Sixth Edition*, Duxbury, 2002
- [59] Taguchi, G., 1985, "Quality Engineering in Japan", *Proc. CAD/CAM, Robotics and Automation International Conference*, The university of Arizona, Tucson, AZ, Feb.
- [60] Douglas C. Montgomery, *Design and Analysis of Experiments 5th Edition*, John Wiley & Sons Inc., 2001
- [61] Raymond H. Myers Douglas C. Montgomery, *Response Surface Methodology Process and Product Optimization Using Designed Experiments 2nd Edition*, John Wiley & Sons Inc., 2002
- [62] Luo J., Liang Z., Zhang C., Wang B., "Optimum Tooling Design for Resin Transfer Molding with Virtual Manufacturing and Artificial Intelligence," *Composites. Part A*, 32 (2001) p 877-888

BIOGRAPHICAL SKETCH

Jing Li studied in Beijing University of Aeronautics and Astronautics and received his B.S in 1999. After one year full-time employment in the field of CAD and CAM process development design for mechanical parts, he decided to pursue a higher degree. He started his M.S.I.E program in Fall 2001 at FAMU-FSU College of Engineering specializing in composite manufacturing process optimization. Other research fields include computer graphic and FEM method. He will continue his Ph. D. studies in Industrial Engineering after graduation.

Born in 1976, the author came from Lanzhou, capital city of Gansu province, P.R. China. His hobby since primary school is building and flying model airplane. He took part in region-wide and nation-wide competitions and was awarded several honors. He enjoys traveling, tennis and fishing.