EQUILIBRIUM TEMPERATURE ANALYSIS AND FILL PATTERN REASONING FOR DIE CASTING PROCESS

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Dongtao Wang, M.S.

* * *

The Ohio State University

2004

Dissertation Committee:

Dr. R. Allen Miller, Adviser

Dr. Jose M. Castro

Dr. Jerald R. Brevick

Approved By

______________________________

Adviser

Industrial and Systems Engineering Graduate Program
Die casting is one of the most important net shape manufacturing processes. There are usually two concerns for die casting designers, thermal characteristics and fill pattern because they are closely related to casting quality and die life. The traditional way to obtain the results is numerical simulation. However, due to the complexity of the equation system to be solved, the computational cost is high and numerical simulation is very time consuming. This makes numerical simulation an imperfect tool during the early stages of product development. It is very desirable for designers to have an efficient and reliable tool to quickly calculate the temperature distribution and fill pattern supporting interactive design.

In this study, a quick algorithm to compute the equilibrium temperature of the die and ejection temperature of the part is presented. The equilibrium temperature is defined as the time average temperature over a cycle after the process reaches the quasi steady state. This can help the cycle and die cooling/heating design. A few models to compute the heat released from part are tested and the combined asymptotic and surrogate model is applied. Special attention is paid to heat transfer calculation at the part-die interface and computational efficiency improvement. The algorithm also addresses the modeling of
cooling/heat line, spray effects and techniques for die splitting at the parting line. The algorithm has been implemented in the software CastView based on the finite difference method.

The previous algorithm used in CastView for fill pattern analysis based on geometric reasoning is redesigned. In this qualitative method, the flow behavior is calculated using the cavity geometric information. Many shortcomings in the old algorithm were fixed and improved. The new algorithm includes considerations which affect the flow behavior, such as flow resistance, more flow angle search and influence within neighborhood. Special attention is paid to computational efficiency improvement.

The fill pattern algorithm for die casting process is adapted for slow fill processes including gravity casting and squeeze casting. The dominant term for flow behavior for different process is defined from dimensionless Navier-Stokes equations. Based on this analysis, the fill pattern algorithm for die casting is modified for slow fill processes.

The analysis results using the algorithms presented in this dissertation are compared with those obtained from numerical simulations, historical data and experiments. The comparison shows good agreement. Given the typical computational time of a few minutes, the efficiency of the algorithms is remarkable.
Currently, the application of equilibrium temperature analysis is die casting and that of fill pattern reasoning is die casting, gravity casting and squeeze casting. However, the algorithms developed in this study can be adapted and applied to other net shape processes.
Dedicated to my parents and my wife.
ACKNOWLEDGMENTS

First and foremost I would like to thank Dr. R. Allen Miller as advisor for his support, guidance and above all, encouragement throughout the years of my PhD program. I have learned much under his instructions that has prepared me to make a contribution in the future.

I thank other members of my dissertation committee, Dr. Jose M. Castro and Dr. Jerald Brevick for their scientific inputs and advice, as well as members of my general exam committee, Dr. P. Sadayappan and Dr. Han-Wei Shen for their valuable comments and suggestions.

I also own thanks to the CastView group members, Yuming Ma, Li Cai, Xiaorui Chen, Lin Suo, Nishant Sharma, Parminder Gandhi, Li Shen and Zhaohui Ning for their help. Thanks also go to Adham Ragab, Abelardo Garza-Delgado, Haijing Mao and Dr. Walter Smith for kindly providing numerical simulation results.

Finally, I wish I could express my sincere gratitude to my wife, Lanchun Fei for her continuous love and support.
VITA

September 10, 1972   Born-Guangxi, P.R. China

1995   B.S., Mechanical Engineering, Tsinghua University, Beijing, P.R. China

1995 – 1998   M.S., Mechanical Engineering, Tsinghua University, Beijing, P.R. China

1998 – 1999   Engineer, Computer Integrated Foundry Technology Lab, Beijing, P.R. China

2000 – present   Graduate Research Associate, Department of Industrial, Welding and Systems Engineering, The Ohio State University

PUBLICATIONS


**FIELDS OF STUDY**

Major Field: Industrial & Systems Engineering

Major Area: CAD/CAE

Minor I: Computer Graphics

Minor II: Scientific Computing
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>DEDICATION</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>vi</td>
</tr>
<tr>
<td>VITA</td>
<td>vii</td>
</tr>
<tr>
<td>PUBLICATIONS</td>
<td>vii</td>
</tr>
<tr>
<td>FIELDS OF STUDY</td>
<td>viii</td>
</tr>
<tr>
<td><strong>CHAPTERS:</strong></td>
<td></td>
</tr>
<tr>
<td>1. BACKGROUND AND INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Die casting process</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem statement</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Heat balance at equilibrium status</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Fill pattern in die cavity</td>
<td>7</td>
</tr>
<tr>
<td>1.3 Literature review</td>
<td>11</td>
</tr>
<tr>
<td>1.3.1 Numerical simulation for die casting process</td>
<td>11</td>
</tr>
<tr>
<td>1.3.2 Geometric reasoning</td>
<td>14</td>
</tr>
<tr>
<td>1.4 Summary</td>
<td>17</td>
</tr>
<tr>
<td>2. EQUILIBRIUM TEMPERATURE ANALYSIS FOR DIE AND PART</td>
<td>18</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Heat balance for equilibrium process</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Models for Heat source calculation</td>
<td>28</td>
</tr>
<tr>
<td>2.4 Asymptotic model</td>
<td>32</td>
</tr>
<tr>
<td>2.5 Surrogate model for ejection temperature</td>
<td>37</td>
</tr>
</tbody>
</table>
3. EXAMPLES AND VERIFICATION OF EQUILIBRIUM TEMPERATURE ANALYSIS FOR DIE AND PART .................................................................71
  3.1 Implementation ..................................................................................71
  3.2 Case 1: flat part ..................................................................................71
  3.3 Case 2: Zinc part ..................................................................................85
  3.4 Conclusions .......................................................................................92

4. GEOMETRIC REASONING FOR FILL PATTERN .................................99
  4.1 Introduction .......................................................................................99
  4.2 Assumption and simplification ..........................................................100
  4.3 Discrete flow vector calculation .........................................................101
  4.4 Fill pattern calculation without obstruction .......................................102
  4.5 Flow obstruction calculation .............................................................103
  4.6 When flow front is blocked ...............................................................104
  4.7 Problems of original algorithm ........................................................105
  4.8 Summary .........................................................................................106
5. IMPROVEMENT OF GEOMETRIC REASONING ALGORITHM FOR FILL PATTERN

5.1 Introduction ........................................................................................................ 107
5.2 Shortcomings and problems of old model ....................................................... 108
5.3 Improvements in new algorithm ...................................................................... 112
5.4 Conclusions ...................................................................................................... 131

6. EXAMPLES AND VERIFICATION OF VISUALIZATION FOR FILL PATTERN

6.1 Implementation ................................................................................................... 133
6.2 Case 1: simple plate part ................................................................................... 133
6.3 Case 2: part with fingers .................................................................................. 138
6.4 Case 3: transfer case ........................................................................................ 142
6.5 Water analog study for verification .................................................................. 149
6.6 Conclusions ....................................................................................................... 157

7. VISUALIZATION FOR SLOW FILL PROCESSES

7.1 Introduction ....................................................................................................... 158
7.2 Consideration from Navier-Stokes equations ............................................... 159
7.3 Fill pattern of gravity casting .......................................................................... 165
7.4 Case study: gravity casting .............................................................................. 170
7.5 Fill pattern for squeeze casting ....................................................................... 176
7.6 Conclusions ....................................................................................................... 185

8. CONCLUSIONS AND FUTURE WORK ............................................................ 187

REFERENCE: ................................................................................................. 196
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1 Ejection Temperature Relationships</td>
<td>45</td>
</tr>
<tr>
<td>Table 2.2 Surrogate Constant Relationships</td>
<td>46</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 1.1 A typical cold chamber die casting machine (<a href="http://www.quantum-online.com">www.quantum-online.com</a>, 2002)</td>
<td>2</td>
</tr>
<tr>
<td>Fig. 1.2 Temperature change curve of a point at die over time. Each jag denotes a cycle.</td>
<td>6</td>
</tr>
<tr>
<td>Fig. 1.3 Temperature change of the same point over a cycle in equilibrium state. At the end of a cycle, temperature comes back to the level at the beginning of the cycle.</td>
<td>7</td>
</tr>
<tr>
<td>Fig. 1.4 Structure of research in this dissertation, which is associated to CastView project and previous work by other group member.</td>
<td>10</td>
</tr>
<tr>
<td>Fig. 2.1 Heat balance for a voxel. Q1 ~ Q6 are heat from the 6 neighboring voxels and Q is heat released from current voxel.</td>
<td>20</td>
</tr>
<tr>
<td>Fig. 2.2 Definition of thermal resistance</td>
<td>22</td>
</tr>
<tr>
<td>Fig. 2.3 A composite wall (without heat transfer coefficient between two materials).</td>
<td>23</td>
</tr>
<tr>
<td>Fig. 2.4. Thermal resistance with heat transfer coefficient at one side.</td>
<td>23</td>
</tr>
<tr>
<td>Fig. 2.5 Matrix form for equation system. T1 ~ Tn are unknowns. The values of m and n depend on the dimensions of computational domain.</td>
<td>26</td>
</tr>
<tr>
<td>Fig. 2.6 Computational domain and neighboring voxels around the target voxel. The labels of l, r, t, b, f, h represent the left, right, top, bottom, front and back voxels.</td>
<td>27</td>
</tr>
<tr>
<td>Fig. 2.7 Illustration of heat flux model. The part is divided into three regions according to part wall thickness.</td>
<td>30</td>
</tr>
<tr>
<td>Fig. 2.8 Illustration of skeleton model. The part is divided into three regions according part wall thickness and the skeleton for each region is computed.</td>
<td>31</td>
</tr>
</tbody>
</table>
Fig. 2.9 Temperature change of a part voxel in a cycle. The part may be ejected when its temperature is at any of three intervals.................................................................33

Fig. 2.10 Temperature change curves over a cycle for 4 different points at the part, which are from Abaqus numerical simulation.............................................................38

Fig. 2.11 Test of surrogate equation using data generated by numerical simulation. This is to demonstrate in principle the surrogate model is a good approximation.............48

Fig. 2.12 Direct search to solve \( h \) and \( g \) functions. The starting point is at \( O \) and the target is \( D \). Marching is made from \( O \) to \( D \) by choosing a direction at each step...............51

Fig. 2.13 A voxel at interface has different neighbors at different step during a cycle, (\( D \) depicts die, \( C \) casting)........................................................................................53

Fig. 2.14 Calculating average temperature of an interface part voxel at different cycle stage. There are two stages, contacting die surface and contacting air.................56

Fig. 2.15 A part with stepped parting plane.................................................................59

Fig. 2.16 Define a sketch plane and the intersection contour of part with parting. .........59

Fig. 2.17 Add a spray area at sketch plane by define a rectangle region (in blue color)...60

Fig. 2.18 An example of complex stepped parting plane. This stepped parting plane contains 6 separate planes.................................................................62

Fig. 2.19 Voxel model of cover die after spitting..........................................................64

Fig. 2.20 Asymptotic model and its linearized model for relationship between temperature of part voxel and heat it releases.........................................................67

Fig. 2.21 Store three data sets in a single buffer by using lower 2 bits as flag to distinguish die, cover insert and ejector insert. The 14 higher bits are used to store temperature data........................................................................................................69
Fig. 3.1 Geometry of flat part, rendering in CastView.................................72
Fig. 3.2 Calculate the time average temperature over a cycle from Abaqus result......75
Fig. 3.3 Equilibrium temperature of ejector insert from Abaqus..............................77
Fig. 3.4 Equilibrium temperature of ejector insert from CastView. Left: 200 voxel
resolution; Right: 300 voxel resolution.................................................................78
Fig. 3.5 Part ejection temperature from Abaqus .....................................................79
Fig. 3.6 Part ejection temperature from CastView using combined asymptotic and
surrogate model (Left: 200 voxel resolution; Right: 300 voxel resolution).............80
Fig. 3.7 Comparison of FDM data and FEM data by interpolation.............................82
Fig. 3.8 Part ejection temperature difference in percentage of Abaqus result and
CastView result (200 voxel resolution)................................................................83
Fig. 3.9 Die equilibrium temperature difference in percentage of Abaqus result and
CastView result (200 voxel resolution)................................................................84
Fig. 3.10 Geometry of the multipart cavity..............................................................88
Fig. 3.11 Cooling line pattern at cover and ejector side (totally 6 cooling lines)............89
Fig. 3.12 Spray pattern (whole area on die surface for both sides).........................90
Fig. 3.13 Die equilibrium temperature from Abaqus...............................................93
Fig. 3.14 Die equilibrium temperature from CastView (Pure asymptotic model)........94
Fig. 3.15 Die equilibrium temperature from CastView (Asymptotic + surrogate)......95
Fig. 3.16 Part ejection temperature from Abaqus ....................................................96
Fig. 3.17 Part ejection temperature from CastView (Pure asymptotic model).............97
Fig. 3.18 Part ejection temperature from CastView (Asymptotic + surrogate).........98
Fig. 4.1 Possible output vector from voxel a (Elfakharany, 1999).........................101
Fig. 4.2 The fifth set discrete flow vectors (Elfakharany, 1999)..............................102
Fig. 4.3 Volumetric distribution of a obstructed voxel (Elfakharany, 1999)..............103
Fig. 4.4 Flow front is completely obstructed (Elfakharany, 1999).........................105
Fig. 5.1 Actual flow pattern when there is an obstruction. Left: sketch; Right: numerical simulation from MagmaSoft.................................................................109
Fig. 5.2 Result calculated using old algorithm when there is an obstruction. Left: sketch; Right: CastView result from old algorithm..............................................110
Fig. 5.3 Melt should reach gate A, B and C simultaneously for a well designed runner system with multiple gates .................................................................111
Fig. 5.4 If flow coming from entrance A to fill ribs B and C. Flow will fill B earlier than fill C because B is more open and has small resistance...............................114
Fig. 5.5 Flow C will fill rib earlier because flow B has larger resistance than C due to the angle between flow B and the rib .........................................................115
Fig. 5.6 Second method to calculate flow resistance based on thickness and speed components along x, y and z directions .........................................................115
Fig. 5.7 New problems for second definition of flow resistance. In the left example, the flow resistance calculated using Eq (5-1) is larger than it should be. In the right example, both cases would have the same flow resistance, which is not correct....116
Fig. 5.8 Third definition for flow resistance. The resistance is calculated based on local geometry openness corresponding to flow vector .....................................117
Fig. 5.9 New vector calculation, where A is oncoming vector, B and C are vectors based on available voxels, D and E are calculated outgoing vectors .......................120
Fig. 5.10 Flow pattern when hitting an obstruction when using old algorithm for outgoing vector calculation........................................................................................................120

Fig. 5.11 Flow pattern when hitting an obstruction. Left: using new algorithm for outgoing vector calculation; Right: MagmaSoft result..........................................................121

Fig. 5.12 A fan gate to be filled and the result using original algorithm with only one angle search ..................................................................................................................123

Fig. 5.13 Fill pattern result using two angle searching. The flow front is flatter than using one angle searching..............................................................................................124

Fig. 5.14 Result of three angle searching which is better than two angle searching......125

Fig. 5.15 Result of four angle searching which is better than three angle searching. .....125

Fig. 5.16 Three gates are specified with red spots. The green circle denotes the biscuit where the initial flow front starts from. By specifying these, the user wishes the cavity fill start at multiple gates simultaneously. ..................................................128

Fig. 5.17 Result using the usual way. The cavity fill starts from central runner before other runners have not been filled.................................................................128

Fig. 5.18 Result if the user specifies the multiple gates. Cavity fill starts from three gates simultaneously after three runners are all filled.................................................129

Fig. 6.1 Simple plate part to be examined by MagmaSoft and CastView......................134

Fig. 6.2 MagmaSoft result for simple plate part (with courtesy of Mao)...................135

Fig. 6.3 MagmaSoft result for simple plate part (with courtesy of Mao). The two near corners are the last regions to be filled.................................................................135

Fig. 6.4 CastView result using old algorithm. The two near corners are filled too early. .................................................................................................................................136
Fig. 6.5 CastView result using old algorithm. The last region to be filled is the central region, which is not correct ................................................................. 136

Fig. 6.6 CastView result for simple plate part using new algorithm ...................... 137

Fig. 6.7 CastView result for simple plate part using new algorithm. The last region to be filled is the two near corners, which is similar to the result from MagmaSoft ...... 137

Fig. 6.8 A part with two fingers at different sides .................................................. 138

Fig. 6.9 MagmaSoft result for finger part (with courtesy of Mao). ......................... 139

Fig. 6.10 MagmaSoft result for finger part (with courtesy of Mao). The second finger is the last region to be filled ................................................................. 139

Fig. 6.11 CastView result for finger part using old algorithm ............................... 140

Fig. 6.12 CastView result for finger part using old algorithm. The second finger is not the last region to be filled ................................................................. 140

Fig. 6.13 CastView result for finger part using new algorithm .............................. 141

Fig. 6.14 CastView result for finger part using new algorithm. The second finger is the last region to be filled ................................................................. 141

Fig. 6.15 A very complex transfer case part for test. There are many thin ribs at around body surface ................................................................. 143

Fig. 6.16 Another view of the transfer case part for test. The part is hallow and the wall thickness is uneven ................................................................. 144

Fig. 6.17 Result from MagmaSoft (with courtesy of Dr. Smith) ............................. 144

Fig. 6.18 Result from MagmaSoft (with courtesy of Dr. Smith) ............................. 145

Fig. 6.19 Result from MagmaSoft (with courtesy of Dr. Smith) ............................. 145
Fig. 6.20 Result of transfer case from CastView using old algorithm. .........................146
Fig. 6.21 Result of transfer case from CastView using old algorithm. .........................146
Fig. 6.22 Result of transfer case from CastView using old algorithm. .........................147
Fig. 6.23 Result of transfer case from CastView using new algorithm.........................147
Fig. 6.24 Result of transfer case from CastView using new algorithm..........................148
Fig. 6.25 Result of transfer case from CastView using new algorithm..........................148
Fig. 6.26 Illustration of the gate, cavity, insert and assembly way of water analog model.  
(Rebello, 1997) ........................................................................................................150
Fig. 6.27 Dimensions of gate and runner (in inch). Upper: top view; Lower: side view.  
(Rebello, 1997) ........................................................................................................151
Fig. 6.28 Dimensions of cavity, die and insert in inch. The depth is 1 inch. Upper: cavity 
and die; Lower: insert. (Rebello, 1997) ..................................................................152
Fig. 6.29 Two cavities for water analog experiment and fill pattern reasoning. Upper: 
with insert; Lower: without insert..........................................................................153
Fig. 6.30 Experiment result and CastView results using old algorithm and new algorithm for cavity without insert. Left: old result; Middle: experiment result (Rebello, 1997); Right: new result.................................................................155
Fig. 6.31 Experiment result and CastView results using old algorithm and new algorithm for cavity with insert. Left: old result; Middle: experiment result (Rebello, 1997); Right: new result.................................................................156
Fig. 7.1 Relation between momentum vector and potential vector. Vectors (1) and (2) represent momentum vector and potential vector, respectively. Cases (a), (b) and (c) represent gravity casting, squeeze casting and die casting (Elfakharany, 1999). 165
Fig. 7.2 Illustration of Bernoulli’s equation.................................................................166

Fig. 7.3 Flow chart of calculating fill pattern of gravity casting based on combination of
die casting fill pattern algorithm and Elfakharany’s algorithm..............................169

Fig. 7.4 Model for experiment and simulation. The lengths are in mm and the third
dimension is 20 mm thick. (Ha, et. al. 1999).............................................................170

Fig. 7.5 Die cavity to be tested on CastView, shown in Unigraphics.........................171

Fig. 7.6 Die cavity in STL shown in CastView. ...........................................................172

Fig. 7.7 Numerical simulation and experiment results. (Ha, et. al. 1999)..............173

Fig. 7.8 CastView results of fill pattern for the same gravity casting part. Left:
Elfakharany’s algorithm; Right: new algorithm.....................................................174

Fig. 7.9 Dimensions of two plates for squeeze casting. The difference of the two parts is
the plate thickness and gate shape. (Wallace, et. al. 2000)...................................178

Fig. 7.10 Two plates with different thickness for squeeze casting. Upper: 0.5 inch plate;
lower: 0.25 inch plate.............................................................................................179

Fig. 7.11 Experimental results and simulation results from Procast for 0.5 inch plate.
(Wallace, et. al. 2000). .........................................................................................180

Fig. 7.12 CastView result for 0.5 inch plate. At the beginning of cavity fill, the flow
forms a bump from gate. The flow front then becomes flat and steady. .............181

Fig. 7.13 Fill pattern by numerical simulation using Procast for 0.25 inch plate. The flow
is in a uniform front to fill the whole cavity. (Wallace, et. al. 2000). ...............183

Fig. 7.14 Result of fill pattern for 0.25 inch plate from CastView. The flow front is
uniform and advances steadily until it hits the end.................................................184
CHAPTER 1

BACKGROUND AND INTRODUCTION

1.1 Die casting process

Die casting (or high pressure die casting in Europe) is one of the most important net shape manufacturing processes. In this process, metal liquid is injected at high pressure and high speed into metal die cavity, with subsequent solidification into useful shapes. Compared with other metal shaping processes such as sand casting, rolling and forging, the die casting process has some advantages such as capability to produce complex metal parts with clear surface and thin walls and with high productivity. These advantages make die casting a more and more important process in the modern industries. The applications of die castings increased significantly in the recent two decades, especially in the automobile, aerospace, electronics and medical apparatus industries. It was predicted that such applications will be further expanded in the future (Gu, 1995).

One requirement for metals to be die cast is they should have low melting point because the die and injection system are usually made by steel. The melting point of a metal to be die cast must be much lower than that of steel otherwise the die may be damaged or
distorted. Thus, at present, the majority of die casting alloys are aluminum alloys, zinc alloys and magnesium alloys because of their low melting points. A die casting machine is used to produce die casting parts. There are two types of die casting machines, hot chamber machine and cold chamber machine depending on the metal delivery method. A hot chamber machine has a crucible, where the alloy is melted and is delivered to injection system directly. In a cold chamber die casting machine, the melt is delivered from a separate furnace. Fig. 1.1 shows a typical cold chamber die casting machine.

![Typical cold chamber die casting machine](www.quantum-online.com, 2002)

It is generally acknowledged that the development of die casting began from the early type-letter production with lead alloys. In 1822, William Church introduced a machine
with an output of up to 20,000 letters per day, which is believed the first successful use of the die casting process (Barton, 1991). In 1839, the first patent of die casting using a piston die casting machine was issued (Kaye and Street, 1982). During the 19th century, the principles of die casting were being applied to the specialized field of printing, and the process was beginning to extend to the production of other articles. But only in the 20th century, especially in the World War II, die casting industry experienced its fast expansion period due to the demand from defense industry (Kaye and Street, 1982).

An example may be able to show the rapid expansion of die casting industry. From 1991 to 2001, the amount of magnesium used in passenger vehicles had an annual growth rate of 12%. In 2001, vehicles in production averaged 2.3 kg (5 lb) of magnesium and the most magnesium-intensive vehicles tipped the content scale at 25 kg (55 lb). "We expect magnesium to grow even faster in the next 10 years, and that's not even talking about powertrain (components)," said Bob Powell, Product Leader of the USCAR/U.S. Automotive Materials Partnership. (Kami Buchholz, 2001).

In the recent years, great progressive development has been made in die casting alloys, die casting technology, die casting machines and control devices. Die casting industry is now in its "golden age" and will see its expansion in the future.
1.2 Problem statement

During the die casting process, liquid or semi-solid metal is injected at high speed to fill a complex die cavity, and solidifies rapidly under high pressure. It is a complicated physical-chemical process. An incorrect process design often causes die casting defects, such as porosity and deformation, and also shortens the die life. To realize appropriate thermal balance and mold filling thus to ensure quality die casting products and prolong the die life, are two important issues for the die casting process design (Allchin, 1990).

1.2.1 Heat balance at equilibrium status

The thermal characteristics of the die and casting are always fundamental considerations for designers. At the conceptual design stage, the typical questions about thermal characteristics include: 1) What does the spatial temperature distribution of the die and part look like? 2) Where are the hot spots? 3) What is the effect of cooling lines and spray? 4) What is the ejection temperature of the part when a certain cycle is applied? Numerical simulation of the transient temperature is the typical way to obtain the answer. The typical procedure for numerical simulation is: build CAD models for part and dies – input thermal property data and cycle parameters – run simulation – view analysis results. The analysis results include the dynamic temperature change of part and dies. Thus the user can read temperature at any location and at any time from the results.
However, one disadvantage of transient solutions is that to obtain the thermal profile at quasi steady state literally hundreds of cycles are needed since the start-up is usually far away from the steady state. This is very time consuming and provides information that is beyond that needed for the cycle and die cooling design since cycle and cooling design only requires the overall temperature distribution and part ejection temperature at steady state. It does not require the dynamic temperature change information in a cycle. In addition, at conceptual design stage, there may be many alternate designs and exploration of the design space may be limited due to the time required to explore all alternatives. Thus it is very desirable to have a tool for thermal analysis that provides the needed information and runs very quickly supporting interactive redesign.

In quasi steady state, the temperature of any point at the die varies throughout the casting cycle but returns to the same value at the same relative time in each cycle. The equilibrium temperature is defined as the time average temperature over a cycle after the process reaches quasi steady state which is illustrated using Fig. 1.2 and Fig. 1.3. Considering any point in the die, its temperature change curve over time is shown in Fig. 1.2. After a number of start-up cycles, cycle to cycle change is small and the process reaches the quasi steady state. The temperature change of this point over a single cycle thus can be illustrated using Fig. 1.3. The cycle can be further separated into a few stages, die close, injection and solidification, die open, ejection, spray and idle stage. The temperature keeps changing in every stage. However, we define the equilibrium
temperature, which is hypothetical and does not actually exist. The equilibrium temperature is defined as time average temperature over a cycle for die and ejection temperature for part when the process reaches the quasi steady state.

Note that this is time average and not a spatial average so the temperature still varies spatially. This equilibrium temperature is helpful for cycle and cooling design, especially at conceptual design stage. A part of the research is to develop and implement a quick approach to compute the equilibrium temperature of die and part.

![Temperature change curve](image)

Fig. 1.2 Temperature change curve of a point at die over time. Each jag denotes a cycle.
Fig. 1.3 Temperature change of the same point over a cycle in equilibrium state. At the end of a cycle, temperature comes back to the level at the beginning of the cycle.

1.2.2 Fill pattern in die cavity

Another common concern of designers is the fill pattern of metal liquid as it enters the cavity. Under high speed and high pressure, the liquid metal flows through the narrow and complex shaped die cavity. This is a very fast transient process, which is usually atomized or turbulent flow. This filling process is one of the most important factors to determine the product quality. The metal flow, if not controlled precisely, can cause flow-related defects, which range from gas porosity and knit lines to incomplete-fill.
The most common flow-related defect might be gas porosity. During the cavity filling, ideally, the liquid metal pushes the cavity gas ahead to the vent as the flow front advances and the vent region is the last area to be filled. But if the flow is not controlled appropriately, the vent is sealed before the whole cavity is filled, thus some of the cavity gas is trapped in liquid metal. As the liquid metal solidifies and cools, the trapped gas in the product becomes the gas porosity. This defect usually results in poor product surface and low strength and is a common reason for rejection of die casting parts (Barone and Caulk, 2000).

Since the filling plays an important role in the product quality, many researchers put their effort on fill pattern prediction. At the present time, the prevailing method is still numerical simulation, which is based on mass conservation, momentum conservation and energy conservation (Lu, et. al. 1999, Khayat, 1998 and Sulaiman, et. al. 2000). Due to the complexity of the equations, the equation solving is usually a time consuming task. Furthermore, the simulation systems require that the user has much experience and understanding of fluid dynamics. These disadvantages limit the utility of numerical simulation and the number of “what-if” questions, especially at the early design stage.

An alternative method is qualitative reasoning for fill pattern prediction. This method is based on the geometric characteristics of the die cavity so it only requires the input of the part and gate geometry and does not require fluid dynamics background of the user. More important, there are no complex equations to be solved so the prediction can be obtained quickly. These reasons make it particularly suitable for the early stage design.
The CastView research group at The Ohio State University developed a qualitative reasoning algorithm for fill pattern prediction (Miller, 1998, Rebello, 1997 and Elfakharany, 1999). According to its application, it helps the improvement of gating system design at the qualitative design stage. There is, however, need to improve the algorithm for better results and for wider application. Another part of this research is to develop a better model for fill pattern to replace the existing one.

The problems for the old model include: 1) Velocity, pressure and resistance are not considered; 2) Calculation for multiple gate sometimes is wrong; 3) Flow pattern may be incorrect when there is an obstruction; 4) There is bias when filling a symmetric part. These problems, which will be discussed in more detail in chapter 4 and chapter 5, are contributes to an incorrect fill pattern prediction and should be addressed in the new model.

In addition, the calculation for fill pattern in CastView is limited to the high pressure die casting process and is not applicable for other similar processes, such as gravity die casting and squeeze casting. Study on filling of these processes shows that the major difference of them is the so-called dominating force (will be further discussed in Chapter 7). Thus it is possible to modify the approach of geometric reasoning on die casting fill pattern for other processes. Another part of this research is to find a general algorithm (or minor modification from die casting algorithm) for these processes.
Fig. 1.4 illustrates the structure of this research cooperating with other work. CastView is a comprehensive CAD/CAE project for die casting process while equilibrium temperature analysis and fill pattern computation are two sub-projects. Research in this dissertation includes equilibrium temperature analysis and redesign and improvement of old algorithm for fill.

---

**Work in this research**

---

**Work by other group members**

Fig. 1.4 Structure of research in this dissertation, which is associated to CastView project and previous work by other group member.
1.3 Literature review

1.3.1 Numerical simulation for die casting process

In the traditional die casting design, engineers optimize the die casting process through a “try-revise” loop, which results in an expensive trial fee and a long trial cycle. To overcome these problems, the die casting researchers have introduced computer technology into die casting design. The traditional design is being replaced by the die casting CAD (computer aided design) and CAE (computer aided engineering). Currently, numerical simulation is a hot topic in this field. In numerical simulation, a mathematical model based on physical-chemical phenomena in die casting process is set up. A collection of process data (including casting and gating system geometrical characteristics, material thermal properties and process parameters) is input into the computer then the simulation program is run and finally the result (prediction) is output (Chen, et. al. 1990, Rosbrook, et. al. 2000).

In the early 1960s, shortly after the appearance of computers, some researchers began studies on numerical simulation of metal solidification (Wang, 1992). According to the one-dimensional heat equilibrium equation, Tuten and Kaiser developed a program for optimizing one-dimensional cooling design with programmable calculators TI-58 and HP-67 in 1979 (Tuten, et. al. 1979). Booth and Allsop made a program for computing heat equilibrium of dies and cooling system parameters with a PET personal computer in
1981 (Booth and Allsop, 1981). Granchi computed the heat transfer between planar nodes thus obtained the two-dimensional temperature distribution of die sections by dividing the calculation field, which was based on energy conservation principle (Granchi and Vettori, 1983).

Most of the research on thermal analysis for die casting is focused on transient solution. But in recent years, some research has been carried out on the equilibrium thermal analysis at steady state. The objective is also to increase the efficiency of the simulation without sacrificing too much accuracy. Barone and Caulk (1993) presented a method to calculate the periodic die temperature at steady state without solving for the start-up transient. Rosidale and Davey's (1998) addressed the thermal behavior of the injection system of hot chamber machines. Their thermal analysis model was based on boundary element method and a series of papers were published. Ma (2000) computed the steady state thermal profile of part and die using purely geometric reasoning method.

In Barone and Caulk’s method, some assumptions were made, such as the casting being comprised of piecewise elements with constant element thickness. These assumptions decreased the accuracy of the result. Rosidale and Davey calculated the whole injection system including nozzle and adaptor, which may increase the computation time. They applied boundary element method, where mesh generation may be a difficulty and only the boundary temperature was represented. Ma’s method did not consider the actual heat transfer so the accuracy was quite poor. These observations make it necessary to develop a quick and easy-to-use tool to obtain equilibrium temperature with reasonable accuracy.
In the early research on die casting simulation, most of the efforts were placed on thermal profile analysis of the casting or die. But subsequently, some researchers carried out research on mold filling simulation due to the essential effect of the filling process on casting quality. There is not yet a satisfactory model describing the filling process accurately because of the complexity. Zhang (1996) developed a two-phase turbulence model, which successfully simulated some cases of filling process in die casting. Lu and Lee (1999) took the effects of the wall thickness and velocity profile between two bounding walls into account of governing equations. They simulated the metal flow in a cylindrical sleeve cavity using cylindrical coordinates system. Khayat (1998) proposed a three-dimensional boundary element method to confined free-surface flow for die casting. His approach is particularly applicable to flow driven by a plunger in cylindrical cavities. Lee and Sheu (2001) presented a new numerical method for incompressible viscous free surface flow without smearing the free surface. Both liquid and air are considered as the physical domain in their model. Though many researchers made much progress on their studies, at present, filling process simulation is still a difficulty in die casting simulation.

Some researchers have concentrated on other problems, such as thermal stress, distortion and peripheral equipment. In 1985, Kim and Ruhlanddt (1985) developed a model to compute thermal stress using the Finite Element Method. Rosindale and Davey (1998) studied the thermal behavior of injection system of a hot chamber machine using boundary element method (BEM).

For thermal analysis, these three packages compute transient temperature primarily due to the way the problem is set up. Thus, inevitably, start-up phase needs to be computed, which is very time consuming. For fill pattern prediction, both MagmaSoft and Procast have good reputation. Their methods are based on the conservation principle thus the analysis is also time consuming, which limits their utility at early design phase.

1.3.2 Geometric reasoning

Geometric reasoning is another important research topic in die casting CAD/CAE technology. Since the thermal profile, fill pattern and other process features are closely related to the geometry of casting and die, it is possible to obtain the feature information based on the pure geometric reasoning. Though the analysis accuracy of geometric
reasoning is lower than that of numerical simulation, geometric reasoning is particularly important at the early design phase since at this stage, only the geometric characteristics are known and other process parameters, which are usually required by numerical simulation, are still unknown.

The application of geometric analysis for casting process began in the 1940s. At that time, computers had not appeared yet. But the Chvoinov’s Rule had been widely known as (www.egr.msu.edu, 2002):

\[ t = C(V/A)^2 \]

where \( t \) is the solidification time for a point in the casting section; \( C \) is a constant relating to casting material, mould material and mould temperature; \( V \) is the volume of casting section; and \( A \) is the surface area of the casting section. Therefore, based on the geometrical characteristic of a casting section, the solidification time for this section can be estimated. Since the requirement for casting design is “sequential solidification” to reduce porosity, the Chvoinov’s Rule can help to optimize casting design. A simpler term “modulus” is thus defined as:

\[ M = V/A \]

Which means the ratio of volume over surface area in 3-D space or surface over perimeter in 2-D space. With the modulus of each casting section, the solidification
sequence of the casting can be calculated easily. Though the Chvoinov’s Rule was originally applied for sand casting, its application for die casting, which constructs the theoretic basis for computational geometric reasoning, is straightforward.

Heine and Uicker (1984) were the pioneers to apply computational geometric reasoning on casting analysis. They published their application and extension of section modulus approach in 1984 and 1985. In the following years, other researchers also carried out the research on geometric reasoning approaches, including Neises, Ravi, Kotschi, Upadhyay (Ma, 2000). The CastView research group at The Ohio State University is very active. In the early 1990s, they began the study and their software “CastView” has gained much attention from the die casting industry in recent years (www.oit.doe.gov, 2002).

As described in previous section (see Fig. 1.4), there are three projects for the research in this dissertation, 1) computation of equilibrium temperature of the die and ejection temperature of the part for die casting process; 2) fill pattern algorithm redesign for die casting process; 3) adaptation of fill reasoning algorithm for slow fill processes. The three projects are sub-projects of the CastView project. This dissertation will discuss the development, implement and analysis results of these three projects.
1.4 Summary

1) Die casting is an important net shape manufacturing process with rapid growth rate of the application in recent years.

2) Thermal characteristics and fill pattern are two common concerns for die casting engineers. The traditional way to obtain the solution is numerical simulation based on conservation equations.

3) Due to the complicated equation solving, numerical simulation is very time consuming and may be not suited in the conceptual design stage.

4) The research in this dissertation is to develop a mathematical model to calculate equilibrium temperature of die and ejection temperature of part. The purpose is to provide a quick tool to obtain the thermal profile of die/part.

5) Another portion of this research is to develop a fill pattern algorithm based on geometric reasoning for die casting process and further adapt the algorithm for slow fill processes. The objective is to predict the fill pattern efficiently without solving complicated equations.

6) The algorithms need to be compatible with current CastView program. The fill pattern algorithm should be based on the old fill pattern algorithm which exists in the current CastView.
CHAPTER 2

EQUILIBRIUM TEMPERATURE ANALYSIS FOR DIE AND PART

2.1 Introduction

As discussed in the previous chapter, prediction of equilibrium temperature of the die and ejection temperature of the casting is important for die cooling and cycle design. One part of this research is to develop a quick mathematical algorithm for equilibrium temperature computation. The objective is to provide a quick computer tool for steady state thermal analysis while providing reasonable accuracy. The intent is to help speed-up die and part design and help optimize cooling line placement and the casting cycle design. With the analysis results, a user can quickly know where the hot spot is and what the overall temperature looks like. This is very helpful at the conceptual design stage. A typical numerical simulation can also do this but it is too time consuming and the information provided is more than needed. Thus the traditional numerical simulation is not suited for this stage. In addition, the equilibrium results can provide a start-up profile for transient temperature simulation with greatly reducing total simulation time. For example, simulation software may have to finish 100 cycles from initial state to reach quasi steady
state for analysis of thermal stress or distortion, which may take days of computation. However, if the simulation starts from equilibrium temperature profile, it may need only 10 cycles to reach steady state, which takes much less time. This chapter discusses the algorithm to compute the equilibrium temperature in CastView.

### 2.2 Heat balance for equilibrium process

The numerical method selected is based on several considerations: 1) Compatibility with available software; 2) Speed; 3) Computer memory requirements. This study is a component of the existing CastView project thus compatibility with the CastView software is very important. Furthermore, pre-processor and post-processor functions of CastView can be used directly for thermal analysis. The CastView software uses the voxel model (a collection of uniform size cubes) to represent 3D geometric solids of die, part or other components. The voxel array is very similar to a finite difference grid. Therefore, finite difference method is employed as the numerical method.

If we consider a voxel in a true equilibrium process, the heat flows into this voxel plus the heat that it generates equals to the heat it flows out due to the energy balance. The die casting process is a quasi steady state and not in a true equilibrium. However, if we consider the whole cycle period, the heat balance still holds. The heat flows in plus heat generated should equal to heat flows out for a voxel when the process reaches quasi
steady state. Since we define the equilibrium temperature as time average temperature over the cycle, we can establish the energy balance equations by applying the equilibrium temperature.

Considering the heat transfer in an interior voxel (Fig. 2.1) in the equilibrium state, the well-known energy balance equation is:

\[ Q_1 + Q_2 + \ldots + Q_6 + Q = 0 \]  

(2-1a)

![Fig. 2.1 Heat balance for a voxel.](image)

Fig. 2.1 Heat balance for a voxel. Q1 ~ Q6 are heat from the 6 neighboring voxels and Q is heat released from the current voxel.

Q1 ~ Q6 are heat transferred from the 6 neighboring voxels and Q is heat released from the voxel itself. In another word, in equilibrium the heat flowing into a voxel plus the heat released must be balanced by the total heat flowing out of the voxel.
Considering the heat flow from face 1, we can express the heat amount in the whole cycle using time average temperature:

\[
Q_i = \int_{t_0}^{t_{cycle}} \frac{(T - T_1)}{R_1} \, dt = \frac{\bar{T} - \bar{T}_1}{R_1} (t_{cycle} - t_0)
\]  

(2-1b)

Where \(T\) and \(T_1\) are the transient temperatures of the target voxel and the neighboring voxel. \(R_1\) is thermal resistance of the target voxel and the neighboring voxel. \(\bar{T}\) and \(\bar{T}_1\) are time average temperatures of target voxel and neighboring voxel. Equation (2-1b) is for an interior voxel only where \(R_1\) does not change over the cycle. For the voxels at interface where heat transfer condition changes during the cycle, the heat flow needs special treatment which will be discussed in later section.

Equation (2-1b) is for heat flowing from a face. If we consider the 6 faces of the target voxel, we will have the sum that heat flows in. Thus Eq. (2-1a) can be written as (after rearrangement):

\[
\frac{\bar{T} - \bar{T}_1}{R_1} + \frac{\bar{T} - \bar{T}_2}{R_2} + \frac{\bar{T} - \bar{T}_3}{R_3} + \frac{\bar{T} - \bar{T}_4}{R_4} + \frac{\bar{T} - \bar{T}_5}{R_5} + \frac{\bar{T} - \bar{T}_6}{R_6} = q
\]  

(2-2)

Where \(R_1 \sim R_6\) are the thermal resistance between the target voxel and neighboring voxels and \(q\) is the interior heat rate. The thermal resistance to heat flow is defined by
analogy to Ohm’s law to current flow. To illustrate the definition of thermal resistance, let’s consider a 1D example.

In the example, the surface temperatures of the two sides of a wall are $T_1$ and $T_2$ with wall thickness $H$ and conductivity $k$. The heat flux is then:

$$ q = -k \frac{dT}{dx} = -\frac{k}{H} (T_2 - T_1) \quad (2-3) $$

If the surface area is $A$, in the thermal resistance form, we have,

$$ qA = -\frac{T_2 - T_1}{R} \quad (2-4) $$

The rate of heat flow is analogous to the current, and the temperature difference is analogous to the potential (voltage) difference. Thus the thermal resistance $R$ can be defined as,
\[ R = \frac{H}{Ak} \]  \hspace{1cm} (2-5)

It can be seen that to increase the thermal resistance, we can either increase the thickness of material or decrease the thermal conductivity. Using the similar analogy, we can develop heat resistance for a composite wall and complex heat transfer.

![Fig. 2.3](image1.png)

Fig. 2.3 A composite wall (without heat transfer coefficient between two materials).

![Fig. 2.4](image2.png)

Fig. 2.4. Thermal resistance with heat transfer coefficient at one side.
Suppose we have a composite wall with two different materials A and B shown in Fig. 2.3, the thermal resistance between the two surfaces (at $T_1$ and $T_2$) is

\[ R = \frac{h_1}{Ak_A} + \frac{h_2}{Ak_B} \]  \hspace{1cm} (2-6)

If there is convection at one side as shown in Fig. 2.4 with heat transfer coefficient $h$, the thermal resistance can be written as:

\[ R = \frac{H}{Ak} + \frac{1}{Ah} \]  \hspace{1cm} (2-7)

This shows that the heat transfer coefficient $h$ at the surface contributes a resistance to heat flow given by $1/Ah$. Therefore, if there is heat transfer coefficient $h$ between two materials in a composite wall, the thermal resistance should be

\[ R = \frac{h_1}{Ak_A} + \frac{h_2}{Ak_B} + \frac{1}{Ah} \]  \hspace{1cm} (2-8)

Back to Eq. (2-2), after rearranging, the equation can be rewritten as

\[-\left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4} + \frac{1}{R_6}\right)T + \frac{T_1}{R_1} + \frac{T_2}{R_2} + \frac{T_3}{R_3} + \frac{T_4}{R_4} + \frac{T_5}{R_5} + \frac{T_6}{R_6} = -q \]  \hspace{1cm} (2-9)\]
This is actually in the form (note that this is for interior voxels only and exterior voxels will be discussed later in this chapter):

\[
a^c \bar{T} + a^l \bar{T}_1 + a^b \bar{T}_4 + a^b \bar{T}_3 + a^r \bar{T}_6 + a^r \bar{T}_2 + a^H \bar{T}_5 = S
\]  \hspace{1cm} (2-10)

where \( \bar{T} \) is the equilibrium temperature of the voxel itself and \( \bar{T}_i \sim \bar{T}_6 \) are temperatures of the neighboring voxels. \( S \) is a term collecting all factors that do not depend on temperature and \( a^c \sim a^H \) are coefficients that depend on interface and material properties. For each voxel, there is an equation of the form of Eq. (2-10) and the total equations for all voxels form a large system of equations (as shown in Fig. 2.5). The values of \( m \) and \( n \) depend on the dimensions of computational domain.
Fig. 2.5 Matrix form for equation system. $T_1 \sim T_n$ are unknowns. The values of $m$ and $n$ depend on the dimensions of computational domain.

The relation of $m$ and $n$ with the computation domain can be illustrated using Fig. 2.6. In the figure, the three dimensions of the domain are $I_{\text{Max}}, J_{\text{Max}}$ and $K_{\text{Max}}$. This is a 3D array but we will use a 1D array to represent the voxels. We choose the marching direction on the three directions as $I \rightarrow J \rightarrow K$ for voxel index. This means we start from $(0, 0, 0)$ and walk to right until we reach $(I_{\text{Max}}-1, 0, 0)$. We then go to next row, starting from $(0, 1, 0)$ and walk to $(I_{\text{Max}}-1, 1, 0)$. This procedure continues until we finish the first layer where $K = 0$. Then we start the second layer $K = 1$. This marching scheme builds a relation between 1D array and 3D array. Considering a target voxel with 1D
index $p$ and the neighboring 6 voxels, the indices of left, right, front, back, bottom and top voxels are $p - 1$, $p + 1$, $p - I_{\text{Max}}$, $p + I_{\text{Max}}$, $p - I_{\text{Max}}*J_{\text{Max}}$ and $p + I_{\text{Max}}*J_{\text{Max}}$. Thus the values of $m$ and $n$ in Fig. 2.5 are $I_{\text{Max}} - 1$ and $I_{\text{Max}}*J_{\text{Max}} - I_{\text{Max}} - 1$.

![Diagram](image)

Fig. 2.6 Computational domain and neighboring voxels around the target voxel. The labels of l, r, t, b, f, h represent the left, right, top, bottom, front and back voxels.

Many existing solvers can solve the equation system (Fig. 2.5) in various ways such as Strongly Implicit Procedure (SIP), Alternating Direction Implicit (ADI) and CGSTAB (Ferziger and Peric, 1999). However, Successive Over Relaxation (SOR) is particularly suitable considering memory requirements. SOR has the simple iterative form:
where $\omega$ is the over relaxation factor which must be greater than 1 for acceleration and $t$ is the iteration counter and i, j, k are voxel position indices. The calculation continues for each voxel until it converges. SOR is a fast solver (though it is not the fastest one) and requires the minimum memory. Compared to other solvers, SOR only needs to store and use the non-zero coefficients ($a^c \sim d^l$), which increases the computation capability significantly.

\[ \overline{T}^{(l)}_{i,j,k} = \frac{\omega S_{i,j,k} - (a^c \overline{T}^{(l)}_{i-1,j,k} + a^H \overline{T}^{(l-1)}_{i,j,k} + a^F \overline{T}^{(l)}_{i,j,k+1} + a^T \overline{T}^{(l)}_{i,j,k-1} + a^R \overline{T}^{(l)}_{i+1,j,k} + a^L \overline{T}^{(l)}_{i,j+1,k}]}{a^c} \]

\[ + (1 - \omega)\overline{T}^{(l-1)}_{i,j,k} \]  \hspace{1cm} (2.11)

### 2.3 Models for Heat source calculation

The heat source of equilibrium is the heat released from part during the solidification or possible heating and cooling lines. The heat released from a part voxel is not independent of die, cooling, spray and other factors. Therefore, one difficulty is how to model the heat released from an individual part voxel. Several attempts were made to approximate the heat amount released, including heat flux model, skeleton model, linear average model, asymptotic model and surrogate model. The combination of asymptotic model and surrogate model was finally selected for the heat source calculation.

In the heat flux model, the part is first divided into regions according to the wall thickness. In CastView, there is a computation module named thick section analysis to
calculate the wall thickness of the part. As shown in Fig. 2.7, the part is divided into three regions, with wall thickness of 2, 3 and 4. It is assumed that at each region surface, the heat flux from part to die is constant. The procedure to calculate heat flux for each region is as follows. 1) A global ejection temperature is assumed, which means all part voxels are ejected at the same temperature. 2) Calculate voxel number and surface area for each region. 3) Based on these values, calculate the heat released from each region and the average heat flux from region surface.

However, there are two disadvantages for this method: 1) The part temperature cannot be calculated; 2) A global ejection temperature is assumed. 3) The heat flux is also related to die geometry even the part thickness is the same. For example, considering the left side and right side of region with wall thickness of 4, as shown in Fig. 2.7, the heat flux from left side should be larger than that from right side because the die temperature at left side is lower than that at right side. Therefore, the assumption is not correct even for a simple case like that in Fig. 2.7.
In the skeleton model, it is assumed that only the skeleton voxel of the part can release heat. Like heat flux model, the part is also divided into regions according to the wall thickness. The skeleton line (also known as the central voxels of a region) is found for each region using a method calling "onion peeling", which actually peels the voxel model from surface to the center. In CastView, there is a computation module named thin section analysis to perform this calculation. As shown in Fig. 2.8, the skeleton line is found and marked with the wall thickness. The heat released from each skeleton voxel thus can be calculated. The procedure for this is as follows. 1) Divide the part into
regions according to wall thickness. 2) Calculate the skeleton lines. 3) Calculate voxel number and the number of skeleton voxels for each region. 4) Assume a global ejection temperature and calculate total heat released from each region. 5) Assign the heat to skeleton voxel of each region. This method can overcome the disadvantages of unbalance heat flux in heat flux model. However, it sill makes an assumption for a global ejection temperature, which is not correct. Therefore, the result applying skeleton model is not good.

Fig. 2.8 Illustration of skeleton model. The part is divided into three regions according part wall thickness and the skeleton for each region is computed.
The shortcoming of these two models led to seeking methods linking the heat source and part average temperature. During the calculation, the approximate equilibrium temperature (time average temperature) of each part voxel is known. This equilibrium temperature should have a relation to part ejection temperature, which determines the heat released by each part voxel. The linear average model is a simple attempt for this idea. In this model, it is assumed that the equilibrium temperature of a part voxel is simply the mid point of the injection temperature and ejection temperature, i.e.

\[ T_{ave} = \frac{(T_{inj} + T_{ej})}{2} \]  

(2-12)

The mid point actually comes from the average assuming temperature in a linear formation of time. Thus the heat released by each part voxel can be easily calculated.

The linear average model is a good attempt but obviously the relation between equilibrium temperature and ejection temperature is not that simple. This led to the asymptotic model.

### 2.4 Asymptotic model

The basic idea of this model is to mimic the general form of the typical cooling curve for a point in the casting and assume a lower bound for the ejection temperature as shown in
Fig. 2.9. The minimum temperature of lower bound must be equal to the temperature of the environment or higher. If the cycle is very long, the ejection temperature will approach the environment temperature but for a practical cycle time, a higher temperature is reasonable.

Fig. 2.9 Temperature change of a part voxel in a cycle. The part may be ejected when its temperature is at any of three intervals.

For a part voxel, if the ejection temperature is below solidus temperature, the released heat during solidification and cooling can be represented by Eq. (2-13), which says that the heat released by a voxel over the cycle is composed of the super heat, the latent heat, and the heat released due to cooling after solidification.
\[ Q' = \frac{\rho \cdot V}{t_{\text{cycle}}} \cdot \left( C_p \cdot \left[ (T_{\text{inj}} - T_{\text{ej}}) \right] + L \right) \]  

(2-13)

Where,

- \( Q' \) – heat rate, (Watts)
- \( \rho \) -- density, (kg/m\(^3\))
- \( t_{\text{cycle}} \) -- cycle time, (sec)
- \( V \) -- volume of the part voxel, (m\(^3\))
- \( C_p \) -- specific heat, (J/kg\(^\circ\)C)
- \( T_{\text{inj}} \) -- injection temperature, (\(^\circ\)C)
- \( T_{\text{liq}} \) -- liquidus temperature, (\(^\circ\)C)
- \( T_{\text{sol}} \) -- solidus temperature, (\(^\circ\)C)
- \( T_{\text{ej}} \) -- ejection temperature, (\(^\circ\)C)
- \( L \) -- latent heat, (J/kg)

For the ejection temperature of a part voxel, there are three possibilities: \( T_{\text{ej}} > T_{\text{liq}} \), \( T_{\text{liq}} > T_{\text{ej}} > T_{\text{sol}} \) and \( T_{\text{ej}} < T_{\text{sol}} \). The relationship between \( T_{\text{ej}} \) and \( \bar{T} \) can be set up for each case.

1) \( T_{\text{ej}} > T_{\text{liq}} \)

The temperature is assumed to change in a linear way in this region. Hence the relationship is:

\[ \bar{T} = \frac{T_{\text{inj}} + T_{\text{ej}}}{2} \]  

(2-14)

\[ T_{\text{ej}} = 2 \cdot \frac{T}{T_{\text{inj}}} - T_{\text{inj}} \]  

(2-15)

2) \( T_{\text{liq}} > T_{\text{ej}} > T_{\text{sol}} \)
It is assumed that the heat rate is constant in this period, i.e., \( \frac{dQ}{dt} = C \). A little calculus and algebra leads to:

\[
T_{ej} = 2 \bar{T} \cdot \frac{Q}{Q - Q_{sh}} - (T_{inj} + T_{liq}) \cdot \frac{Q_{sh}}{Q - Q_{sh}} - T_{liq} \tag{2-16}
\]

\[
Q_{sh} = \rho \cdot V \cdot c_p \cdot (T_{inj} - T_{liq}) \tag{2-17}
\]

\[
Q = Q_{sh} + \rho \cdot V \cdot c_p \cdot \lambda \cdot \left\{ \left( T_{liq} - \bar{T} \right)^2 + \left[ (T_{inj} + T_{liq} - 2 \cdot \bar{T}) \cdot (T_{inj} - T_{liq}) \right] \cdot \left( T_{liq} - T_{sol} \right) \cdot \frac{c_p}{\lambda} + T_{liq} - \bar{T} \right\} \tag{2-18}
\]

Where \( Q \) is total heat released by this voxel and \( Q_{sh} \) represents its super heat.

3) \( T_{ej} < T_{sol} \)

In this period, it is assumed that rate of temperature change is proportional to its difference with the minimum temperature, i.e. Eq. (2-19) describes the time relationship between average temperature and minimum temperature and the proportionality constant is unknown.

\[
\frac{dT(t)}{dt} = \alpha \cdot (T(t) - T_{min}) \text{ with initial condition } T(t_{sh} + t_{lh}) = T_{sol} \tag{2-19}
\]

From this assumption, the ejection temperature is found as a function of other parameters
by solving the equation resulting in Eq. (2-19).

\[ T_{ej} = e^{-\alpha(t_c-t_{lh}-t_{sh})} * T_{sol} + (1 - e^{-\alpha(t_c-t_{lh}-t_{sh})}) * T_{min} \]  
(2-20)

where,

\( \alpha \) -- time constant  
\( t_{sh} \) -- time to release super heat  
\( t_c \) -- cooling time (from injection to ejection)  
\( T_{sol} \) -- Solidus temperature  
\( t_{lh} \) -- time to release latent heat

Starting from this equation and, after considerable algebra, the relationship of average temperature and heat released from the part voxel can be reduced to Eq. (2-21). The key property of Eq. (2-21) is that it relates heat released, \( Q \), to the average temperature, exactly the inverse of the relationship needed for the calculation. The heat released by a part voxel can be computed based on the equilibrium temperature of this part voxel. The calculated relation between heat released and ejection temperature using asymptotic model can be illustrated using Fig. 2.20.

\[
(T - T_{min}) = \left(\frac{T_{sh} - T_{min}}{Q_{sh}}\right) * Q_{sh} + \left(\frac{T_{lh} - T_{min}}{Q_{lh}}\right) * Q_{lh} + \left(\frac{Q - Q_{sh} - Q_{lh}}{T_{sol} - T_{min}}\right) * \ln\left[\frac{Q_{max} - Q}{Q_{max} - Q_{sh} - Q_{lh}}\right]
\]
(2-21)

\( \bar{T}_{sh} \) -- average temperature for release of superheat phase  
\( \bar{T}_{lh} \) -- average temperature for release of latent phase
\( Q_{\text{max}} \) -- maximum heat that can be released (if ejection temperature is the minimum temperature)

### 2.5 Surrogate model for ejection temperature

A drawback of the asymptotic model discussed previously is that a minimum temperature was assumed, which is the lower bound of the ejection temperature. Theoretically, this minimum temperature should be the environment temperature since the ejection temperature will never be lower than the environment temperature. However, it is found from computational experience that the environment temperature does not give a good result for the equilibrium temperature. The typical values for minimum temperatures in calculation of equilibrium temperature are 150 °C for an aluminum part and 130 °C for a zinc part. Occasionally, the minimum temperature needs to be adjusted after the first try.

In addition, the “optimum” minimum temperature varies with part geometry and heat transfer conditions. An example is shown in Fig. 2.10, which comes from numerical simulation. There are four temperature change curves over a cycle, corresponding to 4 different points on a part. It can be seen that at the ejection, the ejection temperatures of 4 points are very different. This indicates that the asymptotic minimum temperature should be locally defined.
Fig. 2.10 Temperature change curves over a cycle for 4 different points at the part, which are from Abaqus numerical simulation.

A surrogate model has been developed to overcome this problem, i.e. to remove the minimum temperature assumption for the calculation.

The rate of change of heat content equation for any voxel other than a part voxel during the period of time latent heat is being released can be described in general terms as follows:

\[
\frac{dQ(t)}{dt} = \rho V C_p \frac{dT_{jk}(t)}{dt} = \begin{cases} 
\left( T_{i+1,jk}(t) - T_{ijk}(t) \right) + \left( T_{i-1,jk}(t) - T_{ijk}(t) \right) + \\
\left( T_{ij+1,k}(t) - T_{ijk}(t) \right) + \left( T_{ij-1,k}(t) - T_{ijk}(t) \right) + \\
\left( T_{ijk+1}(t) - T_{ijk}(t) \right) + \left( T_{ijk-1}(t) - T_{ijk}(t) \right) 
\end{cases}
\]  

(2-22)
The terms $\psi_{lmn}$ in (2-22) are functions of heat transfer coefficients and conductivities depending on the specific interface involved. The subscript $l$, $m$ and $n$ are voxel indices.

This equation describes the relation of heat change and temperature difference with neighboring voxels. After rearrangement, the equation (2-22) becomes

$$\frac{dT_{ijk}(t)}{dt} = \frac{1}{\rho \delta^3 C_p} \left\{ -\left( \psi_{i+1,jk} + \psi_{i-1,jk} + \psi_{ij+1,k} + \psi_{ij-1,k} + \psi_{ijk+1} + \psi_{ijk-1} \right) T_{ijk}(t) ight\}$$

(2-23)

Using simple notation, we can rewrite equation (2-23) as:

$$\frac{dT_{ijk}(t)}{dt} = \left\{ -\left( \lambda_{i+1,jk} + \lambda_{i-1,jk} + \lambda_{ij+1,k} + \lambda_{ij-1,k} + \lambda_{ijk+1} + \lambda_{ijk-1} \right) T_{ijk}(t) ight\}$$

(2-24)

If we define

$$\lambda^T = \begin{bmatrix} \lambda_{i+1,jk} & \lambda_{i-1,jk} & \lambda_{ij+1,k} & \lambda_{ij-1,k} & \lambda_{ijk+1} & \lambda_{ijk-1} \end{bmatrix}$$

$$T_{ijk}^T = \begin{bmatrix} T_{i+1,jk} & T_{i-1,jk} & T_{ij+1,k} & T_{ij-1,k} & T_{ijk+1} & T_{ijk-1} \end{bmatrix}$$

Then we have (symbol $\mathbf{1}$ denotes a column vector of 1’s):
\[
\frac{dT_{ijk}(t)}{dt} = -\lambda \mathbf{1}_{ijk}^T(t) + \lambda \mathbf{T}_{ijk}(t) \quad (2-25)
\]

The differential equation for the part is a little more complex because of the latent heat release. We will use the concept of enthalpy, where the enthalpy content per unit volume of material is defined as (\(f_i\) is the fraction liquid):

\[
h = \int_{0}^{T} C_p(T) dT + Lf_i(T) \quad (2-26)
\]

If the specific heat is constant, the above equation can be written as:

\[
h = C_pT + Lf_i(T) \quad (2-27)
\]

Thus:

\[
\rho \delta^3 C_p \frac{dT_{ijk}(t)}{dt} + \rho \delta^3 L \frac{df_i(T(t))}{dt} = \begin{cases} 
\psi_{i+1,jk} \left( T_{i+1,jk}(t) - T_{ijk}(t) \right) + \psi_{i-1,jk} \left( T_{i-1,jk}(t) - T_{ijk}(t) \right) + \\
\psi_{j+1,ik} \left( T_{j+1,ik}(t) - T_{ijk}(t) \right) + \psi_{j-1,ik} \left( T_{j-1,ik}(t) - T_{ijk}(t) \right) + \\
\psi_{\delta+1,\delta} \left( T_{i+j+1,\delta}(t) - T_{ijk}(t) \right) + \psi_{\delta-1,\delta} \left( T_{i+j-1,\delta}(t) - T_{ijk}(t) \right)
\end{cases}
\]

\[
(2-28)
\]
Assuming that the fraction liquid is a linear function of the temperature difference from liquidus temperature:

\[
f_l(T_{ij}(t)) = \begin{cases} \frac{T_{ij}(t) - T_{sol}}{T_{liq} - T_{sol}} & \text{if } T_{sol} \leq T_{ij}(t) \leq T_{liq} \\ 0 & \text{otherwise} \end{cases}
\]  

(2-29)

Using the chain rule and we have:

\[
\frac{dT_{ij}(t)}{dt} = \frac{C_p*(T_{liq} - T_{sol})}{(C_p*(T_{liq} - T_{sol}) + L*\xi(T_{i,j,k}(t)))} \left\{ -\left(\lambda_{i+1,j,k} + \lambda_{i-1,j,k} + \lambda_{i,j+1,k} + \lambda_{i,j-1,k} + \lambda_{i,j,k+1} + \lambda_{i,j,k-1}\right)T_{ij}(t) \\
+\lambda_{i+1,j,k}T_{i+1,j,k}(t) + \lambda_{i-1,j,k}T_{i-1,j,k}(t) + \lambda_{i,j+1,k}T_{i,j+1,k}(t) \\
+\lambda_{i,j-1,k}T_{i,j-1,k}(t) + \lambda_{i,j,k+1}T_{i,j,k+1}(t) + \lambda_{i,j,k-1}T_{i,j,k-1}(t) \right\}
\]

(2-30)

Where \( \xi(T) = \begin{cases} 1 & \text{if } T_{sol} < T < T_{liq} \\ 0 & \text{otherwise} \end{cases} \)  

(2-31)

Combining (2-25) and (2-31), we have the differential equation that defines a part voxel temperature in the form:

\[
\frac{dT(t)}{dt} = \alpha(T(t))\left( -\lambda^T V(t) + \lambda^T \mathbf{T}_n(t) \right); \quad T(t_0) = T_{inj}
\]

(2-32)

\[
\alpha(T(t)) = \frac{C_p(T_{liq} - T_{sol})}{C_p(T_{liq} - T_{sol}) + L} \text{ if part voxel and } T_{sol} \leq T(t) \leq T_{liq}
\]

(2-33)

\[\alpha(T(t)) = 1 \text{ otherwise}\]
Thus, if the ejection temperature is above liquidus, we can write ($s$ denotes the dummy variable of integration for time),

$$T_{eject} - T_{inj} = \int_{t_{inj}}^{t_{eject}} \left( -\dot{\lambda} T \left( \mathbf{V}(s) - \mathbf{T}_e(s) \right) \right) ds$$  \hspace{1cm} (2-34)

If it is between liquidus and solidus

$$T_{liq} - T_{inj} = \int_{t_{inj}}^{t_{liq}} \left( -\dot{\lambda} T \left( \mathbf{V}(s) - \mathbf{T}_e(s) \right) \right) ds$$  \hspace{1cm} (2-35)

$$T_{eject} - T_{liq} = \left( \frac{C_p \left( T_{liq} - T_{sol} \right)}{C_p \left( T_{liq} - T_{sol} \right) + L} \right) \int_{t_{inj}}^{t_{eject}} \left( -\dot{\lambda} T \left( \mathbf{V}(s) - \mathbf{T}_e(s) \right) \right) ds$$

If it is below solidus

$$T_{liq} - T_{inj} = \int_{t_{inj}}^{t_{liq}} \left( -\dot{\lambda} T \left( \mathbf{V}(s) - \mathbf{T}_e(s) \right) \right) ds$$  \hspace{1cm} (2-36)

$$T_{sol} - T_{liq} = \frac{L}{C_p \left( T_{liq} - T_{sol} \right)} \int_{t_{inj}}^{t_{sol}} \left( -\dot{\lambda} T \left( \mathbf{V}(s) - \mathbf{T}_e(s) \right) \right) ds$$

$$T_{eject} - T_{sol} = \int_{t_{inj}}^{t_{eject}} \left( -\dot{\lambda} T \left( \mathbf{V}(s) - \mathbf{T}_e(s) \right) \right) ds$$

Using fraction solid, equation (2-34) ~ (2-36) can be combined as:
\[ T_{\text{eject}} - T_{\text{inj}} = \frac{L}{C_p} f_s + \int_{t_0}^{t_{\text{eject}}} -\lambda^T (\mathbf{T}(s) - \mathbf{T}_n(s)) ds \]  

(2-37)

\[ T_{\text{eject}} - T_{\text{inj}} = \frac{L}{C_p} f_s - \lambda^T (\mathbf{T} - \mathbf{T}_n)(t_{\text{eject}} - t_0) \]  

(2-38a)

Where \( f_s \) is fraction solid. This implies that the change in temperature is proportional to the average error between the voxel that is the target of the analysis and the surrounding voxels plus a term proportional to the latent heat (heat source).

In Eq. (2-19), \( T_{\text{min}} \) is an unknown constant and only is applied in the cooling phase. This is modified slightly for the entire interval as an estimator or surrogate for the ejection temperature.

\[ \frac{dz(t)}{dt} = -\alpha(z(t))\gamma(z(t) - T_c); \; z(t_0) = T_{\text{inj}} \]  

(2-39)

with \( T_c \) an unknown constant. The notation has been changed from the asymptotic model to minimize confusion, where \( T_c \) is the minimum temperature and \( z \) is surrogate of temperature. Equation (2-39) is an approximation of (2-32) but it does not depend on neighboring voxels. Using the assumption of \( \overline{z} = \overline{T} \) and \( z_{\text{eject}} = T_{\text{eject}} \), equation (2-38a) can be written as:
\[ z_{\text{eject}} - T_{\text{inj}} - f_s(z) \frac{L}{C_p} = -\lambda^T (\overline{T} - \overline{T_n}) (t_{\text{eject}} - t_0) \quad (2-38b) \]

Eq. (2-39) defines the relation of temperature change with difference of temperature and constant \( T_c \). It can be seen from Fig. 2.10 that the temperature change is initially slow and then fast. The difference between temperature and \( T_c \) is initially large and if the parameter \( \gamma \) were also large, the decay of difference between temperature and \( T_c \) would be very rapid. This is not a good representation. Therefore, the only way is that \( \gamma \) is relatively small but the two terms \( \gamma(z - T_c) \) and \( -\lambda^T (\overline{T} - \overline{T_n}) \) are approximately the same resulting in similar rates of change. Solving equation (2-39) with the similar procedure of Eq. (2-32) ~ Eq. (2-38a), we have solution in the form of Eq. (2-40). Rewriting Eq. (2-40), we will define the \( g \) function in Eq. (2-41) with assumptions \( \overline{z} = \overline{T} \) and \( z_{\text{eject}} = T_{\text{eject}} \) for convenience.

\[ z_{\text{eject}} - T_{\text{inj}} = f_s(z) \frac{L}{C_p} + \left( -\gamma z + \gamma T_c \right) \left( t_{\text{eject}} - t_0 \right) \quad (2-40) \]

\[ g(T_c, z_{\text{eject}}, \overline{z}) = z_{\text{eject}} - T_{\text{inj}} - f_s(z) \frac{L}{C_p} + \gamma \left( t_{\text{eject}} - t_0 \right) (\overline{z} - T_c) = 0 \quad (2-41) \]

Comparing equation (2-40) and (2-38), if we assume the surrogate is a good approximation, we can assume the left hand sides of both equations are approximately the same. With the assumption \( \overline{z} = \overline{T} \), we have equations (2-42) and (2-43). Similarly to the \( g \) function, we define the \( h \) function in the form of Eq. (2-44).
\[-\gamma(t_{\text{eject}} - t_0)(z - T_c) \approx \left(-\frac{2}{T} \ln T + \frac{2}{T} \ln T_n\right)(t_{\text{eject}} - t_0)\]  \hspace{1cm} (2-42)

\[T_c = \bar{T} - \frac{\left(-\frac{2}{T} \ln T + \frac{2}{T} \ln T_n\right)(t_{\text{eject}} - t_0)}{-\gamma(t_{\text{eject}} - t_0)}\]  \hspace{1cm} (2-43)

\[h(T_c, z_{\text{eject}}, \bar{T}, T_n) = T_c - \bar{T} + \frac{\left(-\frac{2}{T} \ln T + \frac{2}{T} \ln T_n\right)(t_{\text{eject}} - t_0)}{-\gamma(t_{\text{eject}} - t_0)} = 0\]  \hspace{1cm} (2-44)

Similar to asymptotic model, there are three cases for \(h\) and \(g\) functions for ejection temperature corresponding to super heat phase, latent heat phase and cooling phase, which are summarized in Table 1.1 and Table 1.2.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Ejection Temperature (source terms), (g(T_c, z_{\text{eject}}, \bar{T}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(z_{\text{eject}} \geq T_{\text{liq}})</td>
<td>(z_{\text{eject}} - T_{\text{inj}} - (\bar{T} - T_c)\left[\ln (z_{\text{eject}} - T_c) - \ln (T_{\text{inj}} - T_c)\right] = 0)</td>
</tr>
<tr>
<td>(T_{\text{sol}} \leq z_{\text{eject}} \leq T_{\text{liq}})</td>
<td>(z_{\text{eject}} - T_{\text{inj}} - \left(\frac{T_{\text{liq}} - z_{\text{eject}}}{T_{\text{liq}} - T_{\text{sol}}}\right)\frac{L}{C_p} - \left(\bar{T} - T_c\right)\left[\ln (z_{\text{eject}} - T_c) - \ln (T_{\text{inj}} - T_c)\right] + \frac{L}{C_p (T_{\text{liq}} - T_{\text{sol}})}\left[\ln (z_{\text{eject}} - T_c) - \ln (T_{\text{inj}} - T_c)\right] = 0)</td>
</tr>
<tr>
<td>(z_{\text{eject}} \leq T_{\text{sol}})</td>
<td>(z_{\text{eject}} - T_{\text{inj}} - \frac{L}{C_p} - \left(\bar{T} - T_c\right)\ln (z_{\text{eject}} - T_c) - \ln (T_{\text{inj}} - T_c) + \frac{L}{C_p (T_{\text{liq}} - T_{\text{sol}})}\left[\ln (T_{\text{sol}} - T_c) - \ln (T_{\text{liq}} - T_c)\right] = 0)</td>
</tr>
</tbody>
</table>

**Table 2.1 Ejection Temperature Relationships**
### Table 2.2 Surrogate Constant Relationships

<table>
<thead>
<tr>
<th>Condition</th>
<th>Ejection Temperature (source terms), ( g(T_c, z_{eject}, \overline{T}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( z_{eject} \geq T_{liq} )</td>
<td>( T_c - \overline{T} + \frac{(-\frac{1}{T_c} 1\overline{T} + \frac{1}{T} \overline{T})}{\ln(z_{eject} - T_c) - \ln(T_{inj} - T_c)} (t_{eject} - t_0) = 0 )</td>
</tr>
<tr>
<td>( T_{sol} \leq z_{eject} \leq T_{liq} )</td>
<td>( T_c - \overline{T} + \frac{(-\frac{1}{T_c} 1\overline{T} + \frac{1}{T} \overline{T})}{\ln(z_{eject} - T_c) - \ln(T_{inj} - T_c) + \frac{L}{C_p (T_{liq} - T_{sol})} [\ln(z_{eject} - T_c) - \ln(T_{sol} - T_c) - \ln(T_{liq} - T_c)]} (t_{eject} - t_0) = 0 )</td>
</tr>
<tr>
<td>( z_{eject} \leq T_{sol} )</td>
<td>( T_c - \overline{T} + \frac{(-\frac{1}{T_c} 1\overline{T} + \frac{1}{T} \overline{T})}{\ln(z_{eject} - T_c) - \ln(T_{inj} - T_c) + \frac{L}{C_p (T_{liq} - T_{sol})} [\ln(T_{sol} - T_c) - \ln(T_{liq} - T_c)]} (t_{eject} - t_0) = 0 )</td>
</tr>
</tbody>
</table>

The curves shown in Fig. 2.10 were used to test the surrogate equation as approximation. This test is not a computational test but a test of the approximation and demonstrates that the surrogate, in principle, is a good approximation. It does not demonstrate that a good result can be computed in our equilibrium temperature analysis.
The test actually substitutes the numerical simulation result back into the surrogate model and computes the temperature response with time. The procedure used is as follows:

1) A table of \( z \) values was computed as a function of \( z_{eject} \) and \( T_c \) using \( g \) function.

2) Given the ejection temperature and the average temperature from numerical simulation result, a value for \( T_c \) can be determined.

3) Once values for \( T_c \) and \( z_{eject} \) are available, \( \gamma \) can be computed since \( t_{eject} \) is known.

4) The time response can then be computed from the solutions of the differential equation.

The results are shown in Fig. 2.11. It can be seen that the two sets of curves match very much. This means that if we know the ejection temperature, average temperature and ejection time, we can compute the temperature change curve using surrogate model. The test shows that the surrogate model is a good approximation. However, since in our computation for equilibrium temperature, we do not have ejection temperature and average temperature (they are to be solved), the surrogate model does not guaranty to give very good results.
Fig. 2.11 Test of surrogate equation using data generated by numerical simulation. This is to demonstrate in principle the surrogate model is a good approximation.

### 2.6 Implementation of surrogate model

There are two functions $h(T_c, z_{eject}, \bar{T}, \bar{T}_n) = 0$ and $g(T_c, z_{eject}, \bar{T}) = 0$ in the surrogate model, which are listed in table 2-1 and 2-2. During the calculation, the average temperature $\bar{T}$ for each voxel (element) is available. The $h$ and $g$ functions then need to be solved to obtain $T_c$ and $z_{eject}$ for each voxel, corresponding to minimum temperature.
and ejection temperature. The heat released then can be calculated with ejection
temperature, which can be used to update the average temperature \( \bar{T} \). Such a procedure is
continued until the computation converges. To obtain the initial average temperature \( \bar{T} \),
the asymptotic model can be applied in the first few steps, say, 20 steps with an assumed
minimum temperature. This assumed temperature is not important because it is for the
initial guess for surrogate model and will be adjusted by surrogate model in later
calculation.

The procedure to solve \( h \) and \( g \) functions for each voxel at each step is to solve two non-
linear equations simultaneously, which is much more difficult than solving linear
equations. The first attempt is Newton-Raphson method which comes from Taylor
expansion. The Taylor series at first order for \( h \) function is:

\[
0 = h(T_c, z_{\text{eject}}, \bar{T}) = h(T'_c, z'_{\text{eject}}, \bar{T}) + \frac{\partial h(T_c, z_{\text{eject}}, \bar{T})}{\partial T_c} \Delta T_c + \frac{\partial h(T_c, z_{\text{eject}}, \bar{T})}{\partial z_{\text{eject}}} \Delta z_{\text{eject}}
\]  \( (2-45) \)

Similar equation can be obtained for \( g \) function. Both equations can be combined in a
matrix form:

\[
\begin{bmatrix}
    h(T_c, z_{\text{eject}}, \bar{T}, T_a)
g(T_c, z_{\text{eject}}, \bar{T})
\end{bmatrix}
+ \begin{bmatrix}
    \frac{\partial h(T_c, z_{\text{eject}}, \bar{T}, T_a)}{\partial T_c} & \frac{\partial h(T_c, z_{\text{eject}}, \bar{T}, T_a)}{\partial z_{\text{eject}}} \\
    \frac{\partial g(T_c, z_{\text{eject}}, \bar{T})}{\partial T_c} & \frac{\partial g(T_c, z_{\text{eject}}, \bar{T})}{\partial z_{\text{eject}}}
\end{bmatrix}
\begin{bmatrix}
    \Delta T_c \\
    \Delta z_{\text{eject}}
\end{bmatrix}
= \begin{bmatrix}
    0 \\
    0
\end{bmatrix}
\]  \( (2-46) \)
The equations of \( \frac{\partial h}{\partial T_c}, \frac{\partial g}{\partial T_c}, \frac{\partial h}{\partial z_{\text{eject}}}, \frac{\partial g}{\partial z_{\text{eject}}} \) are listed in Eq. (2-47) ~ (2-50):

\[
\frac{\partial h(T_c, z_{\text{eject}}, \bar{T}, \bar{T}_n)}{\partial T_c} = 1 + \frac{-(-\lambda^T 1 + \lambda^T \bar{T}_n)(t_{\text{eject}} - t_0)(\frac{-1}{z_{\text{eject}} - T_c} + \frac{1}{T_{\text{inj}} - T_c})}{(\ln(z_{\text{eject}} - T_c) - \ln(T_{\text{inj}} - T_c))^2} \\
(2-47)
\]

\[
\frac{\partial g(T_c, z_{\text{eject}}, \bar{T})}{\partial T_c} = (\ln(z_{\text{eject}} - T_c) - \ln(T_{\text{inj}} - T_c)) - (\bar{T} - T_c)(\frac{-1}{z_{\text{eject}} - T_c} + \frac{1}{T_{\text{inj}} - T_c}) \\
(2-48)
\]

\[
\frac{\partial h(T_c, z_{\text{eject}}, \bar{T}, \bar{T}_n)}{\partial z_{\text{eject}}} = \frac{-(-\lambda^T 1 + \lambda^T \bar{T}_n)(t_{\text{eject}} - t_0)(\frac{-1}{z_{\text{eject}} - T_c})}{(\ln(z_{\text{eject}} - T_c) - \ln(T_{\text{inj}} - T_c))^2} \\
(2-49)
\]

\[
\frac{\partial g(T_c, z_{\text{eject}}, \bar{T})}{\partial z_{\text{eject}}} = \frac{z_{\text{eject}} - \bar{T}}{z_{\text{eject}} - T_c} \\
(2-50)
\]

In this method, the two unknown \( \Delta T_c \) and \( \Delta z_{\text{eject}} \) are solved then \( T_c \) and \( z_{\text{eject}} \) can be updated. However, one problem with this method is that the convergence is slow. Since for each part voxel at each iteration, the \( h \) and \( g \) functions need to be solved, the overall convergence is much slower than that in asymptotic model.

Another way is the direct search method. Suppose the \( h \) and \( g \) relation are illustrated using Fig 2-12. For the \( g \) curve, \( g = 0 \) only happens on the curve and the upper side is for \( g > 0 \) and lower side is for \( g < 0 \). The situation is similar for \( h \) curve. The intersection
point D of the g curve and the h curve is the solution. At one iteration step, our starting point is O and the target is D. The search is started with a small step in each of 8 directions at O, with 45° angle between each two neighboring directions. By taking the minimum value of $|h| + |g|$ for each step, the next point can be found since at point D, the value of $|h| + |g|$ is zero. The search is repeated at the new point and this procedure is continued until the $|h| + |g|$ of the new point is close enough to zero.

The above method uses a fixed step size. By experience, it was found that the number of steps for searching varies with the convergence of global computation. For example, if we choose a fixed step size of 1 °C, early in the global computation, it usually takes ~6 steps of marching to reach D. As the computation advances, the number of steps to reach
D decreases to almost 3 or 4. When the computation is almost done, there is little marching from O. This observation indicates that at the beginning, the starting point O is far away from D but O is closer and closer to D during the computation though D is not a fixed point.

Based on this observation, the direct approaching can be further improved using fewer search directions and changing step size. The procedure is as follows:

1) Choose a relatively large step size at the beginning, say, 2 °C;

2) Change the step size to 1 °C, 0.5 °C and 0.2 °C when 20%, 50% and 80% of computation is done (this can be measured approximately through the convergence);

3) Not searching the 8 directions every time except the first time. In other words, the first step is taken by searching the 8 directions. The sequential search is only within last direction and the two neighboring directions. For example, in the first time, we search 8 directions and take the 5th direction. Next time, we only search the 4th, 5th and 6th directions and take the best one.

The direct search method is much faster than Newton-Raphson method and the total run time is only 10 – 20% more than that of the pure asymptotic model.
2.7 Composite heat transfer at interface

A typical die casting cycle consists of 6 steps: die close, injection and solidification, die open, ejection, spray and idle. An interface voxel, i.e., a voxel on the parting plane or cavity surface, has different neighbors at different steps during a cycle (Fig. 2.13). The middle die voxel in the upper row of the figure, located on the parting plane surface, has a casting voxel neighbor when the die is closed and has an air voxel neighbor when the die is open. Similar differences occur for any voxel on the parting surface or cavity surface. Thus, the heat balance equation needs to be modified for all interface voxels to account for the differences in neighbors during the cycle.

Fig. 2.13 A voxel at interface has different neighbors at different step during a cycle, (D depicts die, C casting)

Considering the composite heat over the cycle from both types of neighbors, it can be seen that the total heat is composed of the heat across the interface when the die is closed plus when open, see Eq. (2-51). It was approximated with two different heat transfer coefficients and different neighbor average temperatures and the appropriate average
temperatures Eq. (2-52). Dividing Eq. (2-52) by the cycle time gives the heat rate Eq. (2-53) and rearranging terms gives the expression in terms of the average voxel temperature, average part voxel temperature and air temperature Eq. (2-54). Each term is weighted by the fraction of the cycle that it applies. Eq. (2-54) is used in the heat balance equation Eq. (2-10) for all interface voxels. Essentially, a time weight factor is applied for interface voxels instead of the simpler original Eq. (2-10) that is based on fixed conditions between neighbors.

\[
Q_{\text{face}} = \left[ \int_{t_{\text{closed}}}^{t_{\text{open}}} q_{\text{face, closed}}(t) \, dt + \int_{t_{\text{open}}}^{t_{\text{closed}}} q_{\text{face, open}}(t) \, dt \right] * A \tag{2-51}
\]

\[
Q_{\text{face}} = h_c * A * (T_{i,j,k} - T_{\text{casting}}) * t_{\text{closed}} + h_o * A * (T_{i,j,k} - T_{\text{air}}) * t_{\text{open}} \tag{2-52}
\]

\[
Q'_{\text{face}} = h_c * A * (T_{i,j,k} - T_{\text{casting}}) * \frac{t_{\text{closed}}}{t_{\text{cycle}}} + h_o * A * (T_{i,j,k} - T_{\text{air}}) * \frac{t_{\text{open}}}{t_{\text{cycle}}} \tag{2-53}
\]

\[
Q'_{\text{face}} = \left( \frac{h_p * t_{\text{closed}}}{t_{\text{cycle}}} + \frac{h_o * t_{\text{open}}}{t_{\text{cycle}}} \right) A * T_{i,j,k} - h_p * \frac{t_{\text{closed}}}{t_{\text{cycle}}} * A * \bar{T} - h_o * \frac{t_{\text{open}}}{t_{\text{cycle}}} * A * \bar{T}_{\text{air}} \tag{2-54}
\]

In these equations:

\( h_c, h_o \) -- heat transfer coefficients between die and casting, die and air

\( t_{\text{cycle}}, t_{\text{open}}, t_{\text{close}} \) -- cycle time, die-open time and die-close time

\( Q_{\text{face}}, Q'_{\text{face}} \) -- heat and heat rate on interface
To obtain the equation in Eq. (2-10) form, let us consider a die voxel with 6 faces which is located at interface. Let us further suppose that face 1 is the interface and the time periods to contact air and part voxel are $t_{open}$ and $t_{closed}$. The heat balance equation can be written as:

$$\frac{T - T_1}{R_1} t_{closed} + \frac{T - T_{air}}{R_{air}} t_{open} + \frac{T - T_2}{R_2} t_{cycle} + \frac{T - T_3}{R_3} t_{cycle} + \frac{T - T_4}{R_4} t_{cycle} + \frac{T - T_5}{R_5} t_{cycle} + \frac{T - T_6}{R_6} t_{cycle} = q t_{cycle}$$  \hfill (2-55)

$$\frac{T}{R_1} \left( t_{closed} + \frac{t_{air}}{R_{air}} + \frac{t_{cycle}}{R_2} + \frac{t_{cycle}}{R_3} + \frac{t_{cycle}}{R_4} + \frac{t_{cycle}}{R_5} + \frac{t_{cycle}}{R_6} \right) - \frac{t_{closed}}{R_1} \frac{T_1 - T_2}{R_2}$$

$$- \frac{t_{cycle}}{R_3} \frac{T_3}{R_4} - \frac{t_{cycle}}{R_5} \frac{T_5}{R_6} - \frac{t_{cycle}}{R_6} \frac{T_6}{R_5} = q t_{cycle} + \frac{t_{open}}{R_{air}} T_{air}$$  \hfill (2-56)

Thus the equation can be written in the form of Eq. (2-10). If there are more faces on interface (contacting air or part), the heat balance equation needs to change accordingly.

### 2.8 Average temperature at different stages

In Eq. (2-56), temperature $\bar{T}$ is the average temperature of an interface voxel over the whole cycle. This temperature is used to calculate the heat transfer with other interface voxels in different cycle stages. However, careful inspection found there was error by using this global time average temperature.
Fig. 2.14 Calculating average temperature of an interface part voxel at different cycle stage. There are two stages, contacting die surface and contacting air.

Suppose there is a part voxel at interface (Fig. 2.14), during time period $0 \sim t_{open}$, the heat transfer happens between part voxel and die voxel. During time period $t_{open} \sim t_{ej}$, the heat transfer happens between part voxel and air voxel. In Eq. (2-56), the global equilibrium temperature $\overline{T}$ is used to calculate the heat transfer in both periods. This is not correct because neither the time average temperature $\overline{T}_{part\_closed}$ during the period $0 \sim t_{open}$ is the global temperature $\overline{T}$ nor the time average temperature $\overline{T}_{part\_open}$ during the period $t_{open} \sim t_{ej}$ is $\overline{T}$. In fact, it can be seen from Fig. 2.14 that the three average temperatures are different.
An observation from Fig. 2.14 is that $T_{part\_open}$ is very close to $T_{ej}$. Based on this, an approximation can be done to correct the error. Let us assume that $T_{part\_open} = T_{ej}$, then:

$$
\bar{T} = \frac{t_{open} \cdot T_{part\_closed} + (t_{ej} - t_{open}) \cdot \bar{T}_{part\_open}}{t_{ej}} \quad (2-57)
$$

$$
\bar{T} = \frac{t_{open} \cdot \bar{T}_{part\_closed} + (t_{ej} - t_{open}) \cdot \bar{T}_{ej}}{t_{ej}} \quad (2-58)
$$

$$
\bar{T}_{part\_closed} = \frac{\bar{T} \cdot t_{ej} - (t_{ej} - t_{open}) \cdot \bar{T}_{ej}}{t_{open}} \quad (2-59)
$$

Since $\bar{T}$ is known during computation, $T_{ej}$ can be calculated by asymptotic-surrogate model and $\bar{T}_{part\_closed}$ can be calculated. By this improvement, the error mentioned earlier for part voxels at interface can be corrected partially. There is the same situation for die interface voxels, i.e. parting place voxel and cavity surface voxels. The similar treatment should be applied for those voxels.

### 2.9 Cooling line and spray

Cooling lines and spray are two important ways for changing heat removal conditions and improving the design. Two issues are considered: 1) How to define the geometry; and 2) How to model their effects for heat transfer.
In this method, the opening direction and parting plane are defined before cooling line definition since a cooling line cannot go through parting plane or run into the part. One or more sketching planes are then defined on which cooling lines can be located. The diameter of the lines and the thermal data of the media can also be specified for the cooling lines (Chen, 2002).

The definition for spray is relatively simpler since spray is applied only on the parting surface and die cavity. Rectangular areas on the parting surface can be defined to represent spray areas. The procedure to define a spray area is:

1) Define the die configuration including parting plane (may be a stepped parting plane);
2) Construct a sketch plane perpendicular to the opening direction and at the middle of insert box;
3) Calculate the intersection contour of the part with parting plane;
4) Project the contour on the sketch plane for drawing reference;
5) Define rectangle spray areas on the sketch plane.

The drawing procedure can be illustrated by Fig. 2.15 ~ 2.17.
Fig. 2.15 A part with stepped parting plane

Fig. 2.16 Define a sketch plane and the intersection contour of part with parting.
Fig. 2.17 Add a spray area at sketch plane by define a rectangle region (in blue color).

Like other components (die or part), cooling lines are represented by voxels. Thus the heat transfer between cooling line and other components (die or insert) is modeled using the same finite difference equation with the assumption that the media temperature in the cooling line is constant.

Modeling actual spray effects is complicated but it is modeled approximately by simply increasing the heat transfer coefficient between the sprayed region and air. In the calculation, if the die surface is in spray area, its coefficient is increased to a certain value that is user selectable.
2.10 Handling parting surface

A parting surface is the contact surface between ejector die and cover die. According to the surface complexity, the parting surface can be classified into two types, a simple parting plane and stepped parting surface. A simple parting plane is a single plane that splits the cover half and ejector half and the whole surface is at the same plane. A stepped parting plane actually contains multiple planes connecting each other to form a complex surface. Fig. 2.18 shows an example of stepped parting plane.

Parting plane plays an important role in equilibrium temperature calculation since it determines the heat transfer condition at interface, i.e. heat transfer between steel-steel and steel-air. Detailed treatment for the calculation was discussed earlier. Therefore, an important work is to split the die shoe into two pieces, cover die and ejector die in the voxel model. In the current algorithm, only the part model is created by discretizing the CAD model into voxel model. The die voxel model is then constructed based on the part voxel model. Therefore, a question to be answered is how to split die into two halves if the user has defined a parting plane as Fig. 2.18.

Cai (2002) developed an algorithm to easily define stepped parting planes and internal parting plane. The output of the complex parting plane contains some loops, which are actually intersection vertexes of planes with the die box or part, with an indicator for external plane or internal plane. His technique can be used to split the die into two pieces.
Fig. 2.18 An example of complex stepped parting plane. This stepped parting plane contains 6 separate planes.

If the parting surface is a simple parting plane, the splitting procedure is:

1) Rotate the part model and parting plane so that Z direction is the opening direction;
2) For each voxel at X-Y bottom plane, march along Z direction to the top;
3) If no part voxel is encountered, die is split at simple parting plane;
4) If there is part voxel encountered, die is split the first time a part voxel is encountered.
For a stepped parting plane, it becomes a bit more complicated since the first task is to search which loop that each X-Y bottom voxel corresponds to. The procedure to split the die based on the loops information is then:

1) Rotate part so that Z direction is opening direction;
2) Project the loops to the bottom plane of die box;
3) Triangulate projected loops (polygons) which may be convex or concave polygons;
4) For each voxel at the bottom plane, calculate which triangle that the voxel center point is in. Thus the loop that the voxel belongs to is known since each triangle is associated with a loop.
5) For this voxel, calculate the height of the parting plane (since its loop is known from 3);
6) Split die box for this voxel using simple parting plane technique;
7) This sequence is used on external planes then on internal planes. External parting surfaces represent parting surfaces where the two die halves meet outside the part. Internal parting surfaces represent the parting surfaces that are inside the part.

By this method, the two pieces of die (cover and ejector) can be separated and its interface can be modeled. The cover die after splitting for example in Fig. 2.18 is shown in Fig. 2.19.
Fig. 2.19 Voxel model of cover die after spitting.
2.11 Computational efficiency

2.11.1 Two-step method

The SOR solver requires an initial value for iteration. The closer this initial value is to the final result, the faster the computation converges. In addition, designers are usually concerned with part and insert temperature more than die shoe temperature. Thus in a two-step method, more effort is put on initial temperature of insert and part to improve the computation efficiency. In the first step, a coarse mesh and loose convergence criterion are applied to the whole domain (die, insert and part). The result of first step is used as the initial value for the second step computation (by interpolation) where the mesh is refined and a tight convergence criterion is applied. In the second step, the computation is limited to the insert and part. Since the initial value is close to the final result, the computation converges quickly. Due to the application of a refined mesh and high convergence criterion, the accuracy for insert and part temperature can be ensured. Experiments showed that similar results could be obtained while the run time was reduce up to 40%.
2.11.2 Heat source linearization for asymptotic model

Due to the relationship between temperature and released heat for part voxels, the source term of a part voxel changes during the calculation. This change, which is non-linear (Fig. 2.20), causes two problems and increases run time. First, the source term needs to be computed for each part voxel at each iteration; and second, the changing source term slows convergence. To alleviate these problems, a linearized model is applied, in which a few line segments are used to approximate the heat released function. In Fig. 2.20, each segment represents the linearized relationship between temperature and heat. The source term then can be written as: \( S = a^*T + b \). The first term \( a^*T \) can be moved to left hand side in Eq. (2-10). The term \( b \) is left on the right hand side of Eq. (2-10) but it does not change in this interval. There is no need to re-compute the source term if the temperature stays in the same interval. The calculation also converges faster due to the more stable source term. Experiments showed that the run time could be reduced about 20% compared to the nonlinear representation for pure asymptotic model. For the combination model (asymptotic and surrogate), the run time is reduced about 5% since asymptotic model only takes effect in the early 20% computation.
Fig. 2.20 Asymptotic model and its linearized model for relationship between temperature of part voxel and heat it releases.
2.11.3 Symmetric part

Some parts are symmetric then only half or even quarter of them needs to be computed. Like transient solvers, for these cases, an isothermal layer is set at its central line and only one half is computed. The other half can be mirrored after the computation is finished. The run time is reduced proportionally.

2.12 Data storage

The result output is volume data, which actually contains three sets of data corresponding to temperatures of three components (part, cover insert and ejector insert). The volume data can be rendered by CastView post-processor where ray tracing is applied. The three sets of data can be output in three buffers to the post-processor for display. However, this will cause memory waste because some voxels in each buffer are not used.

A better way is to store the whole data in a single buffer but this requires some modification of the current post-processor. The original ray tracer in CastView is to render the whole volume data at once. Since the output data from equilibrium temperature analysis is an array of integer data, containing temperature of three components, if only the result of a component is desired, a filter between temperature volume data and ray tracer is necessary.
Fig. 2.21 Store three data sets in a single buffer by using lower 2 bits as flag to distinguish die, cover insert and ejector insert. The 14 higher bits are used to store temperature data.

Since the temperature range for die casting can not be out of 0 ~ 1000 and an integer type (16-bits) can stores a value between 0 ~ 65536, the lower 2 bits can be used as flags. Fourteen bits can still store a value between 0 ~ 16384, sufficient for temperature representation. The setting for the lower two bits is: 00 -- part voxel; 01 -- cover insert voxel and 10 -- ejector insert voxel. Thus, before the rendering, the filter fetches the appropriate data from volume data and sends it to the ray tracer.

### 2.13 Conclusions

1) An algorithm based on the Finite Difference Method for equilibrium temperature calculation for die casting process has been developed to speed up part and die design at early design stage and help optimize cooling, heating and cycle design.
2) The algorithm is built on the assumption that the heat released from part equals to the heat absorbed by the environment during a cycle after the production reaches the steady state. The solver SOR was chosen to solve the linear equation system.

3) A few models have been tried to compute the heat released from part voxels. The combination of asymptotic model and surrogate model are chosen because they can compute the ejection from average temperature.

4) Heat transfer at the interface, cooling line and spray effect are addressed in the algorithm.

5) Special attention is paid to computational efficiency and data storage to save run time and memory
CHAPTER 3

EXAMPLES AND VERIFICATION OF EQUILIBRIUM TEMPERATURE ANALYSIS FOR DIE AND PART

3.1 Implementation

The program for equilibrium temperature analysis is being integrated into the *CastView* software using Visual C++ on MS Windows platform. The examples in this chapter were run on a computer – 2.4 GHz Pentium IV PC with 1 GB memory. The typical run time was less than 1 minutes with resolution of 200 voxel at maximum dimension along X, Y and Z directions.

3.2 Case 1: flat part

The first part to be discussed is a flat part (shown in Fig. 3.1). For comparison, the transient temperature simulation of 100 cycles was run on Abaqus using the same parameters. It is assumed that at cycle 100, the production has reached the steady state and the time average temperature of die is the equilibrium temperature.
Fig. 3.1 Geometry of flat part, rendering in CastView.

The two sets of analysis parameters on Abaqus and CastView are the same and is list below, which comes from CastView data file. The meanings of most parameters are clear by their names. The parameters in Advanced section are for SOR solver and convergence tolerance.

General:

cycle_time: 51 sec
die_close_to_end_of_fill: 1 sec
end_of_fill_to_die_open: 9 sec
time_to_eject: 7 sec
spray_time: 0 sec
room_temp: 30 C
die_con_file: fp-part.cvdeg
cooling_file(0-no__1-yes): 1 fp-part.cvcln
spray_file(0-no__1-yes): 0

Casting:
material: Al380
density: 2.76 g/cm³
specific_heat: 0.963 J/g-K
conductivity: 109 W/m-K
latent_heat: 389 J/g
injection_temp: 620 C
liquidus_temp: 598 C
solidus_temp: 538 C

Die:
holder_block_material: H13
insert_material: ST4140
holder_block_conductivity: 40 W/m-K
insert_conductivity: 29 W/m-K

heat_transfer_coefficient_steel_to_steel: 5000 W/m^2-K

heat_transfer_coefficient_steel_to_air: 16 W/m^2-K

heat_transfer_coefficient_part_to_cavity: 5000 W/m^2-K

Others:

symmetry(0-none__1-x__2-y__3-xy): 1

part_temp(0-ejection__1-average): 0

output_die_temp(0-no__1-yes): 0

top_platen(0-no__1-yes): 0 0 C

bottom_platen(0-no__1-yes): 0 0 C

left_platen(0-no__1-yes): 0 0 C

right_platen(0-no__1-yes): 0 0 C

front_platen(0-no__1-yes): 0 0 C

back_platen(0-no__1-yes): 0 0 C

Advanced:

first_step: 0.01

second_step: 0.001

SOR_Omega: 1.9

asymptotic_temp: 150 C
Since Abaqus provides the option to output temperature change into a text file, the time average temperature, i.e. the equilibrium temperature can be obtained by calculating the temperature integration and dividing by the cycle time. The calculation can be shown by Fig. 3.2. The time average temperature at each nodal point is then:

\[ \bar{T} = \frac{\sum (T_1 + T_2) \Delta t}{2t_{cycle}} \]

Therefore, the Abaqus results (equilibrium temperature of the die and part) can be compared to CastView. Further, the comparison can be performed not only by temperature pattern but also the temperature values since the result data from Abaqus is available.

Fig. 3.2 Calculate the time average temperature over a cycle from Abaqus result
The analysis of flat part was run on CastView with two different voxel resolutions, 200 voxels and 300 voxels. The voxel resolution is referring to the voxel number on the maximum dimension among x, y and z direction of the part bounding box. For example, the dimension of xyz bounding box of the flat part is 152.4mm × 275.2mm × 56.8mm. The y dimension is larger than the other two. For the 200 voxels case, there are 200 voxels on this dimension. The voxel size is then $275.2/200 = 1.376 \text{ mm}$. The dimension of the whole die box is 300mm × 400mm × 500mm. The total voxel number in the computational domain is then $(300/1.376) \times (400/1.376) \times (500/1.376) = 218 \times 290 \times 363 = 22\,948\,860$. The voxel size for the 300 voxels case is $275.2/300 = 0.917 \text{ mm}$. The total voxel number is then $(300/0.917) \times (400/0.917) \times (500/0.917) = 327 \times 436 \times 545 = 77\,701\,740$. The run time for the 200 voxels case is 1 minute 4 seconds and the run time for the 300 voxels case is 4 minutes 12 seconds. The run time to finish 100 cycles on Abaqus on a similar PC is about 7 days.

Fig. 3.3 shows the equilibrium temperature of ejector die from Abaqus. As described earlier, this is from the average temperature and is read into Abaqus for display. It can be clearly seen that the hot spot is at the middle rib and the temperature range is 106 °C ~ 378 °C. The results of two resolutions from CastView are shown in Fig. 3.4. The temperature range of the 200 voxel resolution is 140 °C ~ 387 °C and the range of the 300 voxel resolution is 141 °C ~ 386 °C. This may indicate that the 200 voxel resolution is
enough for this part since to increase the voxel number does not change the results much. It can also be seen that the three temperature patterns are very similar and the hot spot in three results is at the middle rib.

Fig. 3.3 Equilibrium temperature of ejector insert from Abaqus
Fig. 3.4 Equilibrium temperature of ejector insert from CastView. Left: 200 voxel resolution; Right: 300 voxel resolution.
Fig. 3.5 Part ejection temperature from Abaqus
Fig. 3.6 Part ejection temperature from CastView using combined asymptotic and surrogate model (Left: 200 voxel resolution; Right: 300 voxel resolution).

The results for part ejection temperature from both CastView and Abaqus are also compared, which are listed in Fig. 3.5 and Fig. 3.6. The calculation from CastView is obtained using combined asymptotic and surrogate model. Unlike die average temperature, the part ejection temperature can be obtained from Abaqus at the cycle 100 without further treatment. It can be seen that the temperature range for Abaqus is 205 °C ~ 574 °C. The temperature range of 200 voxel resolution for CastView is 265 °C ~ 561 °C and that of 300 voxel resolution is 265 °C ~ 571 °C. It can be seen again that the
temperature values from two voxel resolutions are not much different. The lower bound from CastView is slightly higher than that from Abaqus with similar pattern in both cases.

A deeper comparison between both cases is the temperature value comparison. Since Abaqus can output temperature into text file, it is possible to compare the temperature value at the same location. However, there is a little work to be done for such comparison since Abaqus applies Finite Element Method and CastView applies Finite Difference Method and one mesh needs to be transformed into another mesh. Here we transform FEM mesh into FDM mesh (Fig. 3.7). The procedure is as follows;

1) FEM mesh and FDM mesh are aligned at low left corner of bounding box, which is the origin;
2) For center of each voxel, calculate the coordinate values. The temperature of this center from FDM is known from CastView results as T;
3) Calculate which FEM element this voxel center is in;
4) Calculate the temperature T’ at this point from FEM data by interpolation since the temperatures at the 4 vertices are known as T₁, T₂, T₃ and T₄;
5) Compare the difference between T and T’.

Such comparison on part ejection temperature and die average temperature are shown in Fig. 3.8 and Fig. 3.9 in difference percentage. Since the results of 200 voxel resolution and 300 voxel resolution are not different much, the comparison was done on 200 voxel
resolution. The range of difference for part is 0% ~ 23% and for die is 0% ~ 28%. For part, the maximum difference happens at the overflow tip and for die, the maximum difference happens at the corner of the box. It can be expected that the difference between two cases is partially due to error of interpolation.

Fig. 3.7 Comparison of FDM data and FEM data by interpolation.
Fig. 3.8 Part ejection temperature difference in percentage of Abaqus result and CastView result (200 voxel resolution).
Fig. 3.9 Die equilibrium temperature difference in percentage of Abaqus result and CastView result (200 voxel resolution).
3.3 Case 2: Zinc part

The die configuration is a little special for this part. There are four cavities in this die and no insert and the cavities are in the die directly. Another characteristic is there are six cooling lines in the die and a chiller is used in cooling lines, which helps maintain the cooling line temperature at ~10 °C. The melting point of zinc alloy is low (~380 °C) compared to other alloys. Since the dimension is small (only 90 mm on largest dimension for multipart cavity), the cycle time is only 9.35 seconds. The complete calculation parameters are listed below.

General:

- cycle_time: 9.35 sec
- die_close_to_end_of_fill: 0.23 sec
- end_of_fill_to_die_open: 6.08 sec
- time_to_eject: 0.2 sec
- spray_time: 1 sec
- room_temp: 10 °C
- die_con_file: Casting.cvdeg
- cooling_file(0-no__1-yes): 1 Casting.cvcln
- spray_file(0-no__1-yes): 1 casting.cvsp
Casting:

material: Zn

density: 6.6 g/cm^3

specific_heat: 0.419 J/g-K

conductivity: 109 W/m-K

latent_heat: 120 J/g

injection_temp: 417 C

liquidus_temp: 389 C

solidus_temp: 380 C

Die:

holder_block_material: H13

insert_material: H13

holder_block_conductivity: 29 W/m-K

insert_conductivity: 29 W/m-K

heat_transfer_coefficient_steel_to_steel: 5000 W/m^2-K

heat_transfer_coefficient_steel_to_air: 20 W/m^2-K

heat_transfer_coefficient_part_to_cavity: 5000 W/m^2-K

Others:

symmetry(0-none__1-x__2-y__3-xy): 0

part_temp(0-ejection__1-average): 0

output_die_temp(0-no__1-yes): 0
top_platen(0-no_1-yes): 0 0 C
bottom_platen(0-no_1-yes): 0 0 C
left_platen(0-no_1-yes): 0 0 C
right_platen(0-no_1-yes): 0 0 C
front_platen(0-no_1-yes): 0 0 C
back_platen(0-no_1-yes): 1 30 C

Advanced:
first_step: 0.01
second_step: 0.001

The 3D part geometry is shown in Fig. 3.10 and the cooling lines are shown in Fig. 3.11 (both sides). The spray areas are also shown in Fig. 3.12.
Fig. 3.10 Geometry of the multipart cavity
Fig. 3.11 Cooling line pattern at cover and ejector side (totally 6 cooling lines)
The Abaqus results and CastView results are compared, where CastView results are computed from pure asymptotic model and combined asymptotic and surrogate model. The die equilibrium temperature from Abaqus is shown in Fig. 3.13 with the range of 17 °C ~ 247 °C. Again, the equilibrium temperature from Abaqus is computed by taking average of dynamic temperature over a cycle. The corresponding CastView result is shown in Fig. 3.14 and Fig. 3.15, with range of 34 °C ~ 260 °C for pure asymptotic model
and range of 33 °C ~ 235 °C for mixed model. Both CastView results look like the pattern of Abaqus result. However, result from mixed model is better than that from pure asymptotic model since the temperature value is closer to Abaqus result.

The part ejection temperature from Abaqus is shown in Fig. 3.16, with range of 126 °C ~ 234 °C. The corresponding CastView results from both models are also shown in Fig. 3.17 and Fig. 3.18. The pure asymptotic result has the temperature range of 194 °C ~ 316 °C and the mixed model has the temperature range of 222 °C ~ 284 °C. It can be seen again that these results have similar patterns. However, compared to die temperature, the difference between CastView part result and Abaqus part result is larger. This may be because 1) The thermal diffusivity of die is less than that of part; 2) Die is surrounded by air with constant temperature. These reasons make part temperature more sensitive to errors or other factors. In addition, the geometry of die in CastView and Abaqus is different since the geometry of die for Abaqus was generated by CAD system and the detailed feature can be modeled. The geometry of die for CastView can only be generated by CastView itself, which is actually a simple die box. The difference of die geometry should also have some impact for part temperature calculation.
3.4 Conclusions

1) Two cases were selected for test and verification of the algorithm for equilibrium temperature analysis. The analyses were run on CastView and on numerical simulation software Abaqus using the same parameters.

2) In the flat part case, the simulation of 100 cycles was run on Abaqus to reach quasi steady state and the run time was a few days. The run time on CastView for equilibrium temperature was only a few minutes.

3) The temperature patterns of ejector insert and part from two packages are compared. For flat part case, the temperature values of two results are compared by interpolation since the numerical methods of two packages are different (one is FDM and the other is FEM).

4) In both cases, the temperature patterns of die and part from both packages are very similar and the hot spot is at the same location, which indicates our algorithm for equilibrium temperature is valid and efficient.

5) In the comparison of zinc part, the part results from CastView and Abaqus are more different that the flat part. This may be caused by a few reasons including the different die geometry in two computations.
Fig. 3.13 Die equilibrium temperature from Abaqus.
Fig. 3.14 Die equilibrium temperature from CastView (Pure asymptotic model)
Fig. 3.15 Die equilibrium temperature from CastView (Asymptotic + surrogate)
Fig. 3.16 Part ejection temperature from Abaqus
Fig. 3.17 Part ejection temperature from CastView (Pure asymptotic model)
Fig. 3.18 Part ejection temperature from CastView (Asymptotic + surrogate)
CHAPTER 4

GEOMETRIC REASONING FOR FILL PATTERN

4.1 Introduction

Many studies have been carried out on mold filling of die casting due to the important role it plays for the product quality and the lifetime of dies. The prevailing method for fill pattern analysis is numerical simulation, where the disadvantages such as long run time limit the application. An alternative way to perform fill pattern analysis is qualitative reasoning which is simple but efficient. The CastView research group at The Ohio State University developed a qualitative reasoning algorithm for fill pattern prediction (Miller, et. al. 1998). The beauty of their method is the quickness to obtain the analysis results, which is greatly helpful at the conceptual design stage. Since at the conceptual design stage, there may be many alternative designs, a quick tool is important to run all designs and remove the poor designs. Thus at this stage, the level of accuracy is not necessarily very high. The traditional numerical simulation is not applicable because: 1) the data to run a numerical simulation is usually not available yet, such as runner geometry; 2) numerical simulation is too time consuming, usually hours or even days; 3) numerical
simulation provides a relatively high level accuracy, which is more than needed at conceptual design stage. So qualitative reasoning is a more desirable way at the early design stage. This chapter is to review the existing algorithm implemented in the CastView system since part of the proposed research is to improve the algorithm.

4.2 Assumption and simplification

- The flow does not change direction until it is obstructed by a cavity wall or a previously filled region.
- When the flow encounters an obstruction, the volume is uniformly distributed over the nearest unfilled regions.
- Conservation of volume is considered over the distribution within a neighborhood of the voxel that metal is exiting.
- An empty voxel can be filled by metal exiting a voxel or can be filled from multiple voxels, i.e., it can have one or more parent voxels.
- Due to the discrete voxel space, there are limited directions for flow from a voxel. In this algorithm, an interior voxel can have five sets of flow vectors but a gate voxel has less vectors
- If the flow front is completely obstructed, the flow direction is reversed and each voxel acts as a point source of a wave.

Based on these assumptions, the algorithm for fill pattern was set up.
4.3 Discrete flow vector calculation

In discrete voxel space, an input flow vector is "rounded" to the nearest discrete vector available. For example, in Fig. 4.1, the output vector of voxel a must be rounded to available $\overrightarrow{ab}$ or $\overrightarrow{ac}$.

![Initial flow vectors](image)

Fig. 4.1 Possible output vector from voxel a (Elfakharany, 1999).

In Fig. 4.1, the available vectors $\overrightarrow{ab}$ or $\overrightarrow{ac}$ are based on one layer outer voxels (first set). If the second outer layer is counted, the vectors $\overrightarrow{ad}$ or $\overrightarrow{ae}$ are also available (second set). In this algorithm, to keep both vector accuracy and a small computation domain, at most there are five sets of vector available (vectors pointing to the fifth outer layer, shown in Fig. 4.2).
4.4 Fill pattern calculation without obstruction

There are two cases during the calculation, a voxel has a single parent or has more parents. For the case that there is only a single parent and no obstruction, the flow front keeps moving forward at each iteration.

If there is more than one parent for a voxel and no obstruction, the voxel will be filled faster than if it would be filled by only one parent since each parent only needs to fill portion of the volume. The new flow vector will be calculated based on the volume fraction weight vector from its parents.
4.5 Flow obstruction calculation

If an obstruction is present, the new flow vector is based on the part geometry and the flow direction. In Fig. 4.3, the original flow vector is $\vec{ab}$. Black spots represent die voxels or filled part voxels while white spots represent empty part voxels. The flow is going to fill voxel b but b is an obstruction thus the flow will fill other available empty voxels.

Fig. 4.3 Volumetric distribution of an obstructed voxel (Elfakharany, 1999).
When the flow is obstructed, the algorithm will:

1) Search the first set of nearest voxels for empty voxels;
2) If there are empty voxels, distribute the voxel material over the empty voxels and calculate new flow vectors;
3) If there are no empty voxels in the first set, search the second set;
4) If there are no empty voxels in all five sets, consider the path to be completely blocked.

When distributing the voxel material over the empty voxels, the volume fraction distributed is based on the angles between original vector and the new vectors. In Fig. 4.3, since voxel b is an obstruction and voxel c, d, e and f are nearest empty voxels, the flow will fill voxel c, d, e and f. Each empty voxel will receive a fraction volume of a, which depends on the angle between flow vector (\(\overrightarrow{ac}, \overrightarrow{ad}, \overrightarrow{ae}\) or \(\overrightarrow{ae}\)) and the original vector \(\overrightarrow{ab}\) (based on cosine value).

4.6 When flow front is blocked

When the flow front is completely obstructed and the part is not completely filled (Fig. 4.4), the algorithm will do the following:

1) Reverse the flow direction;
2) Each constructed voxel acts as a point source of a wave;
3) From each source voxel, search nearest empty (available) voxels;
4) New flow vectors from new voxels are calculated;
5) Flow calculation continues.

Fig. 4.4 Flow front is completely obstructed (Elfakharany, 1999).

### 4.7 Problems of original algorithm

The algorithm described above was implemented in CastView. The application has shown its quickness and efficiency. However, occasionally analysis results for some cases were not correct. Inspection of this algorithm found that there exist some problems
which include: 1) Velocity and resistance consideration; 2) Flow obstruction calculation; 
3) Flow at multiple gates. These problems are the main reasons for the incorrect analysis 
results and should be fixed in the new algorithm.

4.8 Summary

This chapter reviews the old algorithm for fill pattern reasoning in CastView, which was 
developed by Rebello (1997) and Elfakharany (1999). The algorithm made some 
assumptions and simplifications to reduce computation. The discrete flow vector scheme, 
angle table and method to calculate new vector when obstruction is present were provide 
in the algorithm. However, there are some disadvantages in the old algorithm that 
sometimes result in incorrect analysis results. Next chapter will discuss these 
disadvantages in more detail and discuss the new algorithm.
CHAPTER 5

IMPROVEMENT OF GEOMETRIC REASONING ALGORITHM FOR FILL PATTERN

5.1 Introduction

As discussed in chapter 4, the original algorithm applied in CastView for fill pattern makes many assumptions and simplifications. The assumptions and simplifications are helpful to reduce calculation thus reduce computation time. However, they also cause some shortcomings. The major problem is sometimes the fill pattern predicted by CastView is not correct, compared to numerical simulation result by commercial software, especially for processes with complex cavity or multiple gates. The incorrectness may come from the assumptions and simplifications. Therefore, it is important to improve or replace the current algorithm for fill pattern reasoning. Today’s computers also allow using more complex calculation without much time penalty (however, to solve the conservation equations is still too time-consuming thus is not the direction of this research). This chapter describes an improved algorithm which overcomes some problems of the old algorithm but does not increase the computation time significantly.
5.2 Shortcomings and problems of old model

1) Metal speed not included

Velocity includes speed and direction. Generally speaking, the speed is determined by pressure gradient and other factors then speed determines the fill pattern. However, in the old model, speed is not considered and all moving voxels (flow front) therefore effectively have the same speed. As a result, some space may be filled (occupied) faster than it should be. These filled voxels then obstruct the flow channel for other moving voxels, which causes error for the whole fill pattern. This is a serious problem of the current model and needs to be corrected in new algorithm.

2) Flow obstruction calculation

In the old model, when there is an obstruction, the new flow front is calculated based on the available empty voxels. As a result, a flow will be reflected from a central obstruction, which is not correct (shown in Fig. 5.1 and Fig. 5.2). Fig. 5.1 shows the correct flow pattern when there is an obstruction. The left one is a sketch as demonstration and the right one is a numerical simulation for a simple part using MagmaSoft. Fig. 5.2 shows the result from old model. In the new method, this calculation will be improved. The new flow vector will be calculated not only based on the available empty voxels, but also based on the original flow vector and the near flow vectors.
Fig. 5.1 Actual flow pattern when there is an obstruction. Left: sketch; Right: numerical simulation from MagmaSoft.
Fig. 5.2 Result calculated using old algorithm when there is an obstruction. Left: sketch; Right: CastView result from old algorithm.

3) Cavity with multiple gates

For some parts with thin wall or have special shape, multiple gates are usually applied to avoid filling problem. A hypothetical example is shown in Fig 5-3. Generally, an ideal engineering design for multiple gates and the resistance at the gates themselves should ensure the melt enters the cavity simultaneously from each. In this situation, cavity fill will start after the runner system pressurizes which should not happen until the runner system is nearly full. Even if melt reaches gate C before the other two, the cavity will not start to fill because the gate resistance is significantly higher than the runner resistance.
However, the old model can not distinguish if this is a good design. Due to different fill distance of OA, OB and OC, the melt reaches B earlier than it reaches A and C. The melt will then fill the cavity at B before it reaches A and C even the thickness at C is thin enough, which will makes an incorrect prediction for the fill pattern.

![Diagram of melt flow](image)

**Fig. 5.3** Melt should reach gate A, B and C simultaneously for a well designed runner system with multiple gates

4) Flow resistance not considered.

Flow resistance is an important concept in qualitative reasoning for fill pattern since pressure is not calculated. Flow resistance is difference at different location due to
different geometric characteristics. The flow resistance affects the flow speed then affects the whole fill pattern.

5.3 Improvements in new algorithm

The basic idea is to correct the problems mentioned earlier but not to increase computation time or memory requirements significantly since the efficiency is a principal goal. The following improvements have been accomplished:

1) Speed calculation

In the fill pattern algorithm, a list is used to store the flow front voxels. The cavity voxels can be classified into three groups, filled voxels, empty voxels and flow front voxels. The initial flow front voxels are gate voxels with the initial incoming vector. In each calculation step, the flow front voxels will fill some empty voxels and generate new flow front voxels. This calculation is continued until the whole cavity is filled.

A flow front voxel had a direction (vector) but no speed in the original algorithm. In another word, all flow front voxels had the same speed. In the new algorithm, speed is included (qualitatively) and each flow front voxel may have a different speed. Both speed and the direction of flow may change during cavity filling. There are three possibilities that cause speed change:
a) When the flow hits an obstruction. As the flow vector changes, the speed drops due to energy loss;

b) When a voxel is filled by more than one voxel. The speed of outgoing voxel is calculated based on the speed of incoming voxels;

c) When flow resistance (discussed in detail later) changes due to local geometry the flow speed then may change.

The speed is also rounded to a small number of speed levels to save the computation and storage expense. This is reasonable since we are calculating the fill sequence and not the dynamic change with time. Furthermore, we can freely scale all the speeds of flow front voxels at the same time, which allows us to normalize the speeds to a fixed range. For example, if all the speeds are too small or too large, we can always normalize them to the range of 0 ~ 100.

2) Flow resistance

A flow resistance for each flow front voxel is defined based on local geometric characteristics. One example of flow resistance is shown in Fig. 5.4. Flow is coming from A and will fill B earlier than fill C since flow resistance at C is larger than that at B. Flow speed at C is reduced due to larger flow resistance if the thickness of C is small enough. Apparently the flow resistance is related to cavity openness. The more open, the smaller the resistance. In addition, it is also related to the flow vector. For example, in Fig. 5.5, flow B will have larger resistance than flow C when they are filling the two ribs because
there is an angle between flow B and the rib, which makes it hard for flow B to fill the rib.

An initial definition of cavity openness might be part wall thickness. However, part wall thickness does not represent the cavity openness very well. An example is the part shown in Fig. 5.4 where the part is flat and the thickness in Z direction (the direction coming out from page) is the same. The wall thickness will be the same everywhere but the openness is different in different regions. For example, the openness at B is larger than that at C. However, the variation in openness of different parts of the same shape with differing thickness would be captured by wall thickness.

![Diagram](image)

Fig. 5.4 If flow coming from entrance A to fill ribs B and C. Flow will fill B earlier than fill C because B is more open and has small resistance.
Fig. 5.5 Flow C will fill rib earlier because flow B has larger resistance than C due to the angle between flow B and the rib.

Fig. 5.6 Second method to calculate flow resistance based on thickness and speed components along x, y and z directions.
To calculate the flow resistance, we need to consider both the local openness and flow vector at the same time. The second attempt to calculate flow resistance is shown in Fig. 5.6. The thickness along x, y and z direction is calculated separately, denoting as $t_x$, $t_y$ and $t_z$. The speed components along x, y and z direction are also calculated as $v_x$, $v_y$ and $v_z$. The flow resistance is then defined as

$$r = \frac{1}{|t_x v_x| + |t_y v_y| + |t_z v_z|} \quad (5-1)$$

Fig. 5.7 New problems for second definition of flow resistance. In the left example, the flow resistance calculated using Eq (5-1) is larger than it should be. In the right example, both cases would have the same flow resistance, which is not correct.
Using this definition, the problem of the flat part with the first definition can be eliminated since the thickness at different location is distinguished. However, it would cause other problems, which are illustrated in Fig. 5.7. In the left example, if both the flow vector and part wall is inclined with respect to the coordinate axes, the calculation result from Eq. (5-1) is large because $t_x$ and $t_y$ are both small. However, the actual flow resistance is small because the wall open direction is the same of flow vector. In the right example, calculation results using Eq. (5-1) of both cases are the same because the thickness component at x and y direction are the same. However, it is apparent that the openness of two cases is different. Therefore, the second definition can not represent the flow resistance correctly.

Fig. 5.8 Third definition for flow resistance. The resistance is calculated based on local geometry openness corresponding to flow vector.
The third resistance thus is defined as inverse of the multiplication of part voxel number along flow vector and part voxel number at the plane perpendicular to flow vector. This can be shown by the example in Fig. 5.8 to calculate the flow resistance at flow front voxel A with a flow vector. A flow line is first constructed along flow vector (shown as the dash line in Fig. 5.8) and we count the number of part voxel along the local region at this flow line, denoting the number as \( m \). Then a flow plane through voxel A but perpendicular to the flow line is constructed, shown as an ellipse in Fig. 5.8. We count the number of part voxel in local region at the flow plane, denoting as \( n \). Note that voxel B is not counted because it is not in the local region even it is at the flow plane. A local region means there is no die voxel between the current voxel and the target voxel. Then the flow resistance is computed as \( r = 1/(nm) \). By this definition, we can address both local openness and flow vector. The flow resistance then can be used to compute flow speed.

Actually, we only need the inverse of flow resistance, \( m \times n \) during the calculation. We define the flow potential (not to be confused with potential energy) as \( p = s \times m \times n \), where \( s \) is the speed. In each calculation step, we first calculate \( p \) for every flow front voxel then take the overall average potential \( \bar{p} \) as the threshold. If the potential of a flow front is larger than the threshold, this voxel is said to have enough potential to move. If not, this voxel is held into the next step until it has larger potential than the threshold. In addition, since there are only limited flow vector in discrete space, we can pre-compute the locations of voxels for each flow line and each flow plane associated with each flow
vector and store them in a table. When computing flow resistance, we only need to check directly if the voxels in the table are cavity voxels. Further, we only check voxels in the flow plane within a certain range, say, 10×10 voxels around the flow front voxel to save computation.

3) Vector change at obstruction

When a flow hits an obstruction, the flow vector is going to change. In the previous algorithm, the new flow vector was calculated by simply constructing a vector from the current flow front voxel to the empty voxel. With the improvement, this is changed to the sum of original vector and position vector (vector from current voxel to empty voxel). In Fig 5-9, solid dots represent obstructions and blank dot represent empty voxels. A flow front voxel has the vector A and hits an obstruction. In the old algorithm, there were two new vectors, B and C while in the new one, the two new vectors are D (sum of A and C) and E (sum of A and B). In this way, the new direction is not only related to position but also related to the original direction, which gives a better way to calculate new direction. This can be considered to include the effect of inertial term, which is relative to the original direction. The calculation results using the old and new algorithms are shown in Fig. 5.10 ~ 5.11 using a plate part with a through hole, where the hole serves as the obstruction. When the flow hits the obstruction, it should disperse more like the pattern shown in Fig. 5.11 than Fig. 5.10.
Fig. 5.9 New vector calculation, where A is oncoming vector, B and C are vectors based on available voxels, D and E are calculated outgoing vectors.

Fig. 5.10 Flow pattern when hitting an obstruction when using old algorithm for outgoing vector calculation.
There is a pre-computed angle table to search empty voxels when a flow hits obstructions. The table contains voxel locations of 13 different angles, which is between current flow vector and vector to empty voxels, ranging from 35° to 180°. The neighboring empty voxels can be found through this angle table. The angle search starts from smallest angle to largest angle. Whenever there is an available angle, which means there is an empty voxel, the search stops and takes this empty voxel to calculate the new vector in the old algorithm. In other words, the search only selected one angel as the output. As a result, some available vectors are ignored and the new vectors are not the
complete possible vectors. Fig. 5.12 shows a fan gate to be filled. The flow should move smoothly from back to front and the flow front should be nearly a “flat line”, which is the design intent of the fan gate. However, as shown in Fig. 5.13, the result from original algorithm is not an anticipated.

The reason for this is that only one angle was considered to calculate new vectors. As the flow advances, the thickness is smaller and smaller then the resistance becomes larger and larger. The flow would move to any possible direction. If there is one angle limitation, the flow front becomes irregular, as shown in Fig. 5.12. In the new algorithm, more available angles are provided to be searched. The results for the fan gate for 2, 3 angles and 4 angle searching are shown in Fig. 5.13 ~ 5.15. It can be seen that result from 2 angle searching is much better than one angle searching. Result of 3 angle searching is better than 2 angle searching while result of 4 angle searching is better than 3 angle searching. However, there is efficiency penalty to search more angles. Compared to the original run time, the run time on 2, 3 and 4 angel searching are about 25%, 40% and 50% more, respectively. For the tradeoff, we choose 2 angel searching since it is much better than one angle searching and only slightly increases the run time.
Fig. 5.12 A fan gate to be filled and the result using original algorithm with only one angle search
Fig. 5.13 Fill pattern result using two angle searching. The flow front is flatter than using one angle searching.
Fig. 5.14 Result of three angle searching which is better than two angle searching.

Fig. 5.15 Result of four angle searching which is better than three angle searching.
Flow speed is also affected when flow hits obstruction since there is energy loss. The change of speed is related to the angle between the original vector and the new vector. The larger the angle is, the more the speed decreases. For example, the speed loss for the $180^\circ$ angle should be larger than that for the $35^\circ$ angle. There are totally 13 angles in the angle table in ascending order and they are grouped based on the angle value. The reduced speed for each group is also in ascending order.

4) Multiple gates

If a part has multiple gates with different distances to the biscuit, the flow may or may not start to fill cavity simultaneously. If the runner system is well designed, the cavity fill will start from gates simultaneously due to the large resistance at gates. For the example shown in Fig. 5.3, cavity fill will start after the runner system pressurizes which should not happen until the runner system is nearly full if this is a properly designed runner system. Even if melt reaches gate C before the other two, the cavity will start to fill only to a small degree because the gate resistance is significantly higher than the runner resistance. However, in the original algorithm, due to different fill distance of OA, OB and OC, the melt reaches C earlier than it reaches A and B. The melt will then fill the cavity at C before it reaches A and B, which will makes an incorrect prediction for the fill pattern if this is a well designed runner system.
The new algorithm provides an option to allow the user to specify if he wants the cavity fill to start at multiple gates simultaneously. In the example of Fig. 5.16, three gates (identified with a red dot) and the biscuit (in green circle) are specified, which indicates this is a properly designed runner system. During the fill pattern calculation, the surface analysis is done first to find the wall thickness information. For each gate point, the local region with the same thickness is found and is set to be a gate region. All gate voxels are assigned a very large flow resistance, which will force the flow to stop at gate region until the runner system is filled. The flow can start to fill cavity simultaneously from gate regions. Fig. 5.17 shows the result if the user does not specify the multiple gates while Fig. 5.18 shows the result if he does. It is clearly shown that in Fig 5-18, the three flows start to fill cavity almost simultaneously.
Fig. 5.16 Three gates are specified with red spots. The green circle denotes the biscuit where the initial flow front starts from. By specifying these, the user wishes the cavity fill start at multiple gates simultaneously.

Fig. 5.17 Result using the usual way. The cavity fill starts from central runner before other runners have not been filled.
Fig. 5.18 Result if the user specifies the multiple gates. Cavity fill starts from three gates simultaneously after three runners are all filled.

5) Bias removal

The fill pattern for a symmetric part should also be symmetric (for example, the flat part with hole in Fig. 5.10). However, the analysis result from old algorithm was not symmetric. Careful inspection found that it was caused by the calculation sequence. As mentioned earlier, there is a list to store the flow front voxels. At each calculation step, the new flow front voxels are generated based on the sequence of old flow front voxels, i.e. from the front to the end. Since the number of voxel can be filled in a step is fixed due to mass conservation, there is fill competition for flow front voxels, which may
generate fill bias. For example of Fig. 5.10, if in the initial flow front list, voxels at left side are the front and voxels at right side are the end, the voxels at left side will always be filled earlier than those at right side. As the calculation continues, there is more and more bias. The solution is to reverse the sequence at each step. At an odd step, the calculation is from the front of flow front list to its end while at an even step, it is from the end to the front. Thus the bias can be removed.

6) Influence of neighboring voxels

In the old algorithm, the calculation for each flow front voxel was independent of its neighboring flow front voxels. There was no influence between neighboring voxels. As a result, the flow lines were often very dispersed. This differs from numerical simulation. Furthermore, occasionally there were small flows advancing too far compared to others, which should be incorrect. An example of old algorithm was shown in Fig. 5.10. Even after we apply the new calculation of vector (shown in Fig. 5.11), the flow lines are still too dispersed. In the new algorithm, the influence from neighboring voxels is addressed by grouping flow front voxels and balancing their vector and speed, which helps reduce the dispersal. The flow front voxels are first grouped based on their neighbor relationship. In each group, the average vector and speed are calculated. Each flow front voxel is then slightly adjusted to the average vector and speed. This way does not eliminate the problem completely but can help balance the behavior of voxels within neighborhood.
7) Efficiency issue

Special attention was paid to efficiency for implementation of new algorithm. It is undesired that run time increases significantly due to the added computation. Some methods are applied to reduce computation: 1) Pre-computed tables of geometry information are used. 2) Real data are rounded to integers. 3) Variables are saved for later use. 4) Data structure and computation are optimized. With these treatments, the run time increase for the new algorithm is 20% ~ 50% compared to the old.

5.4 Conclusions

A new algorithm for visualization for fill pattern was designed based on the old algorithm. There are some improvements:

1) Speed calculation is included. The change of speed of flow front is affected by local flow resistance.

2) Flow resistance and flow potential are calculated during the analysis. Three definitions have been tried to model flow resistance. Flow potential determines whether a flow front has enough “potential” to move.

3) Vector change calculation at obstruction was redesigned. The simple examples of plate with hole and fan gate demonstrate the improvement.
4) The user is provided the option to specify multiple gates, which ensures cavity fill start at multiple gates simultaneously.

5) Fill pattern bias due to competition between flow fronts to fill empty voxels is removed by switching the sequence of flow front list.

6) Influence of neighboring voxels is considered. Special attention is paid to computational efficiency issue.
CHAPTER 6

EXAMPLES AND VERIFICATION OF VISUALIZATION FOR FILL PATTERN

6.1 Implementation

The program for fill pattern visualization has been integrated into the CastView software using Visual C++ on MS Windows platform. The examples in this chapter were run on a computer – 2.4 GHz Pentium IV PC with 1 GB RAM. The typical run time of CastView is about 10 minutes with resolution of 200 voxel along the maximum dimension. Four cases are demonstrated in this chapter. Three of them are compared to numerical simulation, including a simple part, a medium complex part and a very complex part. The other is compared to water analog.

6.2 Case 1: simple plate part

The comparison was done between numerical simulation and qualitative reasoning since the accuracy of numerical simulation is higher. The first part is a simple plate part. The numerical simulation was performed on MagmaSoft by Ms. Haijing Mao. The run time on
MagmaSoft was ~2 hours and the run time on CastView was only ~ 2 minutes. The part geometry is shown in Fig. 6.1, in CastView rendering. The results from MagmaSoft are shown in Fig. 6.2 and Fig. 6.3 with the courtesy of Ms. Mao. From the MagmaSoft results, it can be seen that the flow fills the central region first and then the two far corners. The near two corners are the last regions to be filled. The visualization results from CastView using old algorithm are shown in Fig. 6.4 and Fig. 6.5 while the results using new algorithm are shown in Fig. 6.6 and 6.7. It can be seen that in the new results, the flow also fills the central region first and then far corners, finally near corners. Both patterns of MagmaSoft result and new algorithm results are very similar. However, the old results are not similar to those of MagmaSoft. In the old results, the flow fills the near corners earlier and the last region to be filled is the central region, which is not correct. This comparison suggests the improvement.

Fig. 6.1 Simple plate part to be examined by MagmaSoft and CastView.
Fig. 6.2 MagmaSoft result for simple plate part (with courtesy of Mao)

Fig. 6.3 MagmaSoft result for simple plate part (with courtesy of Mao). The two near corners are the last regions to be filled.
Fig. 6.4 CastView result using old algorithm. The two near corners are filled too early.

Fig. 6.5 CastView result using old algorithm. The last region to be filled is the central region, which is not correct.
Fig. 6.6 CastView result for simple plate part using new algorithm.

Fig. 6.7 CastView result for simple plate part using new algorithm. The last region to be filled is the two near corners, which is similar to the result from MagmaSoft.
6.3 Case 2: part with fingers

The second part to be compared is a part with two fingers at both sides. The 3D part geometry is shown in Fig. 6.8. Again the numerical simulation was done on MagmaSoft. The run time on MagmaSoft was ~3 hours and the run time on CastView was ~6 minutes. The results from MagmaSoft are shown in Fig. 6.9 and Fig. 6.10. It can be seen that the first finger is filled simultaneously with the second half of part and the second finger is the last region to be filled. The results from CastView using old algorithm are shown in Fig. 6.11 and Fig. 6.12 while the results using new algorithm are shown in Fig. 6.13 and 6-14. In the old results, the second half starts to be filled before the center of first half is filled, which is not similar to that of MagmaSoft. The results using new algorithm has similar pattern with that of numerical simulation. The first finger is not filled until the second half of part is filled. The second finger is the last region to be filled.

Fig. 6.8 A part with two fingers at different sides
Fig. 6.9 MagmaSoft result for finger part (with courtesy of Mao).

Fig. 6.10 MagmaSoft result for finger part (with courtesy of Mao). The second finger is the last region to be filled.
Fig. 6.11 CastView result for finger part using old algorithm.

Fig. 6.12 CastView result for finger part using old algorithm. The second finger is not the last region to be filled.
Fig. 6.13 CastView result for finger part using new algorithm.

Fig. 6.14 CastView result for finger part using new algorithm. The second finger is the last region to be filled.
6.4 Case 3: transfer case

The third part for test is a relatively complex part with relatively thin wall. The 3D part geometry is shown in Fig. 6.15 and Fig. 6.16. The part is hollow with a number of very thin ribs while the part wall at bottom is thick, which may cause fill related problems in production. The results from MagmaSoft are shown in Fig. 6.17 ~ 6.19 (kindly provided by Dr. Walter Smith, DCD Technology Inc.). The results from CastView using old algorithm are shown in Fig. 6.20 ~ 6.22 while the results using new algorithm are shown in Fig. 6.23 ~ 6.25. It can be seen that the results using old algorithm does not have similar fill pattern with that from numerical simulation in that the last region to be filled is not the same. However, the results using new algorithm have similar pattern to those of MagmaSoft. The flow fills central region and ribs first then the upper left region is the last one to be filled. These results for the complex part show again the improvement for geometric reasoning algorithm for fill pattern.
Fig. 6.15 A very complex transfer case part for test. There are many thin ribs at around body surface.
Fig. 6.16 Another view of the transfer case part for test. The part is hallow and the wall thickness is uneven.

Fig. 6.17 Result from MagmaSoft (with courtesy of Dr. Smith)
Fig. 6.18 Result from MagmaSoft (with courtesy of Dr. Smith)

Fig. 6.19 Result from MagmaSoft (with courtesy of Dr. Smith)
Fig. 6.20 Result of transfer case from CastView using old algorithm.

Fig. 6.21 Result of transfer case from CastView using old algorithm.
Fig. 6.22 Result of transfer case from CastView using old algorithm.

Fig. 6.23 Result of transfer case from CastView using new algorithm.
Fig. 6.24 Result of transfer case from CastView using new algorithm.

Fig. 6.25 Result of transfer case from CastView using new algorithm.
6.5 Water analog study for verification

Since it is extremely difficult to run a real experiment on die casting process to obtain fill pattern result, water analog study is an important way to verify the fill pattern algorithm. Though the water analog model can not represent the heat transfer and solidification effects, it still has good similarity with the flow behavior of liquid metal in die cavity.

The water analog model results in this section are from Rebello’s dissertation (1997). Two cases of experiment with different cavity were run. One has an insert in the cavity and the other does not. The gate, cavity, insert and the assembly way are showed in Fig. 6.26. The dimensions of gate and runner are shown in Fig. 6.27 and those of cavity and insert are shown in Fig. 6.28.

The CAD model of was built using Unigraphics NX and the STL model was exported for CastView analysis. The rendering of the two cavities (with and without insert) are shown in Fig. 6.29.
Fig. 6.26 Illustration of the gate, cavity, insert and assembly way of water analog model.

(Rebello, 1997)
Fig. 6.27 Dimensions of gate and runner (in inch). Upper: top view; Lower: side view.

(Rebello, 1997)
Fig. 6.28 Dimensions of cavity, die and insert in inch. The depth is 1 inch. Upper: cavity and die; Lower: insert. (Rebello, 1997)
Fig. 6.29 Two cavities for water analog experiment and fill pattern reasoning. Upper:
with insert; Lower: without insert.
The experiment results and CastView results using old algorithm and new algorithm for the case without insert are listed in Fig. 6.30, (Left: old algorithm; Middle: water analog result from Rebello’s dissertation; Right: new algorithm). It can be seen from the photo taken from the experiment that the flow fills the central region first. Some flow is touching the end wall and starts to fill the end region with the region close to end wall unfilled. Both old result and new result follow this pattern but the new result is more similar to the experiment result. For example, the experiment result shows that the region close to gate is completely filled at this stage. The new result matches this pattern but the old result does not. The region close to gate has not been filled completely yet in old result. In addition, both experiment result and new result show that the small corner at the end has not been filled at this stage. However, the old result shows this corner has been filled, which is not correct.

The similar comparison is made on the case with insert, which is listed in Fig. 6.31. The fill patterns of three results are similar. Flow hits the 45° angle wall then changes the direction. The small tip region is the last region to be filled. Comparing the old result and the new result, a slight difference is that in the new result, the flow fills the central region directly but in the old result, the flow fills the central region and the edge region at the same time. In addition, the flow lines in the old result are very dispersed. Judging from the experiment result, the new result is slightly better than the old result.
Fig. 6.30 Experiment result and CastView results using old algorithm and new algorithm for cavity without insert. Left: old result; Middle: experiment result (Rebello, 1997); Right: new result.
Fig. 6.31 Experiment result and CastView results using old algorithm and new algorithm for cavity with insert. Left: old result; Middle: experiment result (Rebello, 1997); Right: new result.
6.6 Conclusions

1) Some cases are chosen for test and verification for the new fill pattern algorithm. These include three cases compared with numerical simulation results and two cases compared with water analog results.

2) The three parts compared to MagmaSoft simulation include a simple part, a medium complex part and a very complex part. The run time for a reasoning analysis is minutes and the run time for a numerical simulation is hours.

3) The results from new algorithm and results from numerical simulation have good agreement but there is some difference for the results from old algorithm. This shows the improvement of the new algorithm.

4) In the two cases compared to water analog results, one has an insert in the cavity and the other one does not. The three results from old algorithm, new algorithm and experiment are similar. However, the results from new algorithm are slightly better than those from old algorithm. The comparison shows the improvement of the new algorithm again.
7.1 Introduction

Some metal forming processes other than die casting, such as gravity casting, low pressure die casting, squeeze casting and semi-solid casting also have cavity filling process. From the fluid dynamics point of view, their filling characteristics are different from die casting. For example, in die casting, the viscous term and body force term usually are both negligible due to its small viscosity and high velocity. However, for gravity casting, the body force term is the most important term and for semi-solid casting, the viscous term is important but inertial term only plays minor role due to the large viscosity and low velocity. Nevertheless, it is still possible to apply the similar model to calculate the fill patterns for those different filling processes and only minor modification is required during the implementation.

The current fill pattern model is for die casting process only and cannot be used on other processes since the filling characteristics of other processes are not considered. In this
research, the fill model for die casting process is modified to calculate fill pattern of those slow fill processes. The purpose is to build a “general” algorithm of fill pattern analysis for different processes with minor modification. The basic idea is to consider the so-called dominant term in calculation because the dominant term is different with different processes. At this stage, only two slow fill processes are tried, gravity casting and squeeze casting.

7.2 Consideration from Navier-Stokes equations

The well known Navier-Stokes equations are widely applied to solve fluid flow problems in numerical simulation. Starting from Navier-Stokes equations in our research can provide some useful hints for dominant terms in our modeling.

The Navier-Stokes equations in rectangular coordinates are written as:

\[
\rho \left( \frac{\partial v_x}{\partial t} + v_x \frac{\partial v_x}{\partial x} + v_y \frac{\partial v_x}{\partial y} + v_z \frac{\partial v_x}{\partial z} \right) = -\frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} + \frac{\partial^2 v_x}{\partial z^2} \right) + \rho b_x \tag{7-1}
\]

\[
\rho \left( \frac{\partial v_y}{\partial t} + v_x \frac{\partial v_y}{\partial x} + v_y \frac{\partial v_y}{\partial y} + v_z \frac{\partial v_y}{\partial z} \right) = -\frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 v_y}{\partial x^2} + \frac{\partial^2 v_y}{\partial y^2} + \frac{\partial^2 v_y}{\partial z^2} \right) + \rho b_y \tag{7-2}
\]

\[
\rho \left( \frac{\partial v_z}{\partial t} + v_x \frac{\partial v_z}{\partial x} + v_y \frac{\partial v_z}{\partial y} + v_z \frac{\partial v_z}{\partial z} \right) = -\frac{\partial p}{\partial z} + \mu \left( \frac{\partial^2 v_z}{\partial x^2} + \frac{\partial^2 v_z}{\partial y^2} + \frac{\partial^2 v_z}{\partial z^2} \right) + \rho b_z \tag{7-3}
\]
Since the terms in the equations have dimensions, it is not easy to see the dominant terms (or the important terms). Some algebra of scaling could help us make decisions.

To do the scaling analysis, we need to define the dimensionless variables as follows:

\[ t^* = \frac{t}{t_c} \]
\[ x^* = \frac{x}{L} \]
\[ y^* = \frac{y}{L} \]
\[ z^* = \frac{z}{L} \]
\[ v_x^* = \frac{v_x}{V} \]
\[ v_y^* = \frac{v_y}{V} \]
\[ v_z^* = \frac{v_z}{V} \]
\[ p^* = \frac{p}{\Delta p} \]
\[ b_x^* = \frac{b_x}{B} \]
\[ b_y^* = \frac{b_y}{B} \]
\[ b_z^* = \frac{b_z}{B} \]

Where:

\( t_c \) --- cycle time;

\( L \) --- characteristic dimension. It can be the length of part;

\( V \) --- characteristic velocity; It can be the gate velocity;

\( \Delta p \) --- characteristic pressure gradient.

\( B \) -- gravity acceleration;

The Navier-Stokes equations now can be changed from dimensional to dimensionless variables. Making use of the chain rule for differentiation:

\[
\frac{\partial}{\partial t} = \frac{dt^*}{dt} \frac{\partial}{\partial t^*} = \frac{1}{t_c} \frac{\partial}{\partial t^*} , \text{ similarly:}
\]

\[
\frac{\partial}{\partial x} = \frac{1}{L} \frac{\partial}{\partial x^*}
\]

\[
\frac{\partial}{\partial y} = \frac{1}{L} \frac{\partial}{\partial y^*}
\]

\[
\frac{\partial}{\partial z} = \frac{1}{L} \frac{\partial}{\partial z^*}
\]

\[
\partial v_x = V \partial v_x^*
\]

\[
\partial v_y = V \partial v_y^*
\]
\[ \partial v_z = v \partial v_z^* \]

Using these relations to introduce the dimensionless variables into the governing equations, with \( b_x^* = b_y^* = 0 \), (supposing z direction is pointing up) yields:

\[
\left[ \frac{\rho V}{t_c} \right] \left( \frac{\partial v_z^*}{\partial t} \right) + \left[ \frac{\rho V^2}{L} \right] \left( v_y^* \frac{\partial v_z^*}{\partial x} + v_z^* \frac{\partial v_y^*}{\partial x} + v_z^* \frac{\partial v_z^*}{\partial x} \right) = -\left( \frac{\Delta p}{L} \right) \left( \frac{\partial p^*}{\partial x} \right) + \left[ \frac{\mu V}{L} \right] \left( \frac{\partial^2 v_y^*}{\partial x^2} + \frac{\partial^2 v_z^*}{\partial x^2} \right) \tag{7-4}
\]

\[
\frac{\partial^2 v_y^*}{\partial y^2} + \frac{\partial^2 v_z^*}{\partial z^2} \]

\[
\left[ \frac{\rho V}{t_c} \right] \left( \frac{\partial v_y^*}{\partial t} \right) + \left[ \frac{\rho V^2}{L} \right] \left( v_y^* \frac{\partial v_y^*}{\partial x} + v_y^* \frac{\partial v_y^*}{\partial y} + v_z^* \frac{\partial v_y^*}{\partial z} \right) = -\left( \frac{\Delta p}{L} \right) \left( \frac{\partial p^*}{\partial y} \right) + \left[ \frac{\mu V}{L} \right] \left( \frac{\partial^2 v_y^*}{\partial x^2} + \frac{\partial^2 v_y^*}{\partial z^2} \right) \tag{7-5}
\]

\[
\frac{\partial^2 v_y^*}{\partial y^2} + \frac{\partial^2 v_z^*}{\partial z^2} \]

\[
\left[ \frac{\rho V}{t_c} \right] \left( \frac{\partial v_z^*}{\partial t} \right) + \left[ \frac{\rho V^2}{L} \right] \left( v_y^* \frac{\partial v_z^*}{\partial x} + v_z^* \frac{\partial v_y^*}{\partial x} + v_z^* \frac{\partial v_z^*}{\partial x} \right) = -\left( \frac{\Delta p}{L} \right) \left( \frac{\partial p^*}{\partial z} \right) + \left[ \frac{\mu V}{L} \right] \left( \frac{\partial^2 v_z^*}{\partial x^2} + \frac{\partial^2 v_z^*}{\partial z^2} \right) + \left[ \rho B \right] p^* \tag{7-6}
\]

All terms in brackets are dimensionless. Thus we only need to compare the coefficients in the square brackets. If we have values typical for die casting of aluminum alloy,

\[
\mu = 3 \times 10^{-3} \text{ Pa} \cdot \text{s} --- \text{Viscosity}
\]

\[
\rho = 2.7 \times 10^3 \text{ kg/m}^3 --- \text{Density}
\]

\[
V = 10 \text{m/s} --- \text{Velocity}
\]
\[ L = 0.1m \text{ --- Fill distance} \]

\[ B = 9.8m/s^2 \text{ --- Z component of Gravity} \]

Substituting these values into Eq. (7-4) ~ (7-6) and we have

Viscous term: \( \frac{\mu V}{L^2} = \frac{3 \times 10^{-3} \times 10}{0.1^2} = 3. \)

Inertial term: \( \frac{\rho V^2}{L} = \frac{2.7 \times 10^3 \times 10^2}{0.1} = 2.7 \times 10^6. \)

Gravity term: \( \rho B = 2.7 \times 10^3 \times 9.8 = 2.6 \times 10^4. \)

These data show the viscous term is negligible in die casting. Even the gravity term is also negligible compared to inertial term. Clearly, the dominant term is inertial term. For gravity casting and squeeze casting, the velocity is 0.5 m/s and 1 m/s, the corresponding data is (suppose the material property remains the same):

Gravity casting:

Viscous term: \( 3 \times 10^{-3} \times 0.5 / 0.1^2 = 0.15. \)

Inertial term: \( 2.7 \times 10^3 \times 0.5^2 / 0.1 = 6.8 \times 10^3. \)

Gravity term: \( 2.7 \times 10^3 \times 9.8 = 2.6 \times 10^4. \)

Squeeze casting:

Viscous term: \( 3 \times 10^{-3} \times 1 / 0.1^2 = 0.3. \)
Inertial term: $2.7 \times 10^3 \times 1^2 / 0.1 = 2.7 \times 10^4$.

Gravity term: $2.7 \times 10^3 \times 9.8 = 2.6 \times 10^4$.

It can be seen that for gravity casting, the dominant term is gravity term but inertial term also takes some effect though viscous term is still negligible. For squeeze casting, both the inertial term and gravity term are dominant terms.

Considering the above typical values, the order of processes for importance of gravity is: die casting, squeeze casting and gravity casting. The gravity effect in die casting is negligible but along this sequence, the gravity plays a more and more important role. For gravity casting, the momentum effect is nearly negligible but gravity term is the dominant term.

To model the fill pattern of gravity casting and squeeze casting, two component vectors are applied to represent the effects of inertial term and gravity term, namely a momentum vector and a potential vector. The momentum vector depends on the flow velocity while the potential vector depends on the flow location. The combination of these two vectors determines the flow vector. In die casting, momentum vector is the dominant vector since the filling process is under high pressure with high velocity while in gravity casting, gravity vector is the dominant vector since the velocity is small (Fig. 7.1). If both flows are dominating (for squeeze casting), the actual flow is the sum of both flows.
The calculation shows that it is possible to visualize the fill pattern for slow processes using a single algorithm. The major difference is to determine the dominant flow or to calculate the contribution of different flows. Then the resistance and speed for each flow front voxel can be calculated and the fill pattern analysis can be performed.

7.3 Fill pattern of gravity casting

The previous analysis based on dimensionless Navier-Stokes equations proves the dominant term in gravity casting is the gravity term, while inertial term takes some effect.

Elfakharany developed an algorithm for fill pattern reasoning for gravity casting. His
model is based on the analysis of Bernoulli’s equation. The Bernoulli’s equation states that the sum of potential head, the velocity head and the pressure head of a flowing liquid is constant and is in this form:

\[
\frac{V_1^2}{2g} + \frac{P_1}{\rho} + h_1 = \frac{V_2^2}{2g} + \frac{P_2}{\rho} + h_2
\]

Where \( V_1 \) = metal velocity at point 1.

\( g \) = acceleration due to gravity.

\( P_1 \) = static pressure in the liquid at point 1;

\( h_1 \) = height of liquid at point 1.

\( \rho \) = density of the liquid.

Fig. 7.2 Illustration of Bernoulli’s equation
The Bernoulli’s equation is a simplified energy balance equation, which divides the flow energy at a given point into three parts, kinetic energy, pressure energy and potential energy and the sum of these three energies is constant. Though Bernoulli’s equation only applies in equilibrium, Elfakharany uses the concept to define the momentum vector and potential vector.

In Elfakharany’s model, the sum of momentum value and potential value is always equal to a constant. The potential vector is always pointing in the –Z direction and the value is relative to the height (Z coordinate value) of current flow front voxel. If the maximum Z value of voxel model is \( n \), a flow front voxel with Z coordinate of \( n \) has the maximum potential value. Similarly, a flow front voxel with Z coordinate of 0 has minimum potential value. The direction of the momentum vector is the current flow front vector. During the computation, since the Z coordinate value is known, the value of potential can be calculated. Since the sum of potential and momentum values is known, the value of momentum vector can be calculated. Given the potential direction and momentum direction are known, the sum of two vectors can be calculated, which is the new flow front direction.

Elfakharany’s model provides a basic idea to calculate the effect of gravity vector. However, there are some problems in his model, which include:
1) Elfakharany’s model was based on the old algorithm for fill pattern for die casting. As discussed previously, there were some problems for the old fill pattern model, such as no speed, no flow resistance, etc. These problems usually cause analysis errors. Thus, the old model should be improved.

2) As discussed in section 7.2, the dominant term in gravity casting is the gravity term. The ratio of importance between gravity term and inertial term is about 10:3. However, in Elfakharany’s model, the ratio used to calculate new vector is 1:1, which is not correct and should be fixed.

3) Since the gravity vector is always pointing down, the part should be well oriented so that the sprue is facing up to positive Z direction. However, in the old algorithm, the orientation is dependent on the CAD model. This is not correct and should be fixed. The part should be oriented so that the sprue is facing positive Z direction.

The improvement is made on the combination of new algorithm for die casting and Elfakharany’s algorithm for gravity casting. The new algorithm is implemented and integrated into CastView and the flow chart is shown in Fig. 7.3.
Fig. 7.3 Flow chart of calculating fill pattern of gravity casting based on combination of die casting fill pattern algorithm and Elfakharany’s algorithm.
7.4. Case study: gravity casting

The example is taken from the paper “Simulation of Die Filling in Gravity Die Casting Using SPH and MagmaSoft” by Ha, et. al. (1999), where the die model dimension, experiment results, MagmaSoft results and SPH results are included. This provides a way to compare CastView results with results from other approaches.

Fig. 7.4 shows the cavity geometry and gate as well as runner system. The water analogue modeling was also performed as experiment. Ha and his associates built a transparent die to record the filling of water.

The CAD model was built using Unigraphics NX and the STL model was exported for CastView analysis. The rendering of part in Unigraphics and STL are shown in Fig. 7.5 and Fig. 7.6. The experiment results and numerical simulation results are shown in Fig.
7.7. The left result is from MagmaSoft while middle result is from experiment and right result is from SPH simulation. It can be seen that the flow fills the wide rib first then fill the narrow rib along the channel. Both simulation results match the experiment very well.

The geometric reasoning is performed on this model for fill pattern by Elfakharany’s algorithm and the new algorithm. Results from both algorithms are listed in Fig. 7.8.

Fig. 7.5 Die cavity to be tested on CastView, shown in Unigraphics.
Fig. 7.6 Die cavity in STL shown in CastView.
Fig. 7.7 Numerical simulation and experiment results. (Ha, et. al. 1999)
Fig. 7.8 CastView results of fill pattern for the same gravity casting part. Left: Elfakharany’s algorithm; Right: new algorithm.
Fig. 7.8 continued
Comparing the geometric reasoning results from both algorithms and those from Ha’s paper, we can see that the results match quite well. The flow fills the wide rib then the narrow rib, which demonstrates the suitability of geometric reasoning for gravity casting. However, comparison of the details shows some difference. When the flow is entering the wide rib, it forms a fountain due to gravity effect. This fountain can be seen at the results of numerical simulation and experiment but not in the result from Elfakharany’s algorithm. However, this fountain can be clearly seen in the results from the new algorithm. That is because in the new algorithm, the weight of gravity is larger, as proved that it should be (from section 7.2). Another improvement is the pattern after the wide rib has been completely filled and the flow starts to fill the horizontal narrow rib. The results of simulation and experiment show that the left end of horizontal rib is filled before flow reaches the right end, which exactly matches the result from new algorithm. In results from Elfakharany’s algorithm, the flow starts to fill the final vertical rib before the left end of horizontal rib is filled, which is not correct. This comparison shows the improvement of fill pattern for gravity casting.

7.5 Fill pattern for squeeze casting

As discussed previously, the dominant terms for die casting, gravity casting and squeeze casting are different, which can be seen from the dimensionless Navier-Stokes equations. In die casting process, the dominant term is the inertial term and the gravity term is
negligible. In gravity casting, the dominant term is gravity term and inertial term only plays a minor role for the fill pattern. However, in squeeze casting, there is not a single dominant term. Gravity term and inertial term play almost equal roles in determining the fill pattern.

In our algorithm, we choose the ratio of gravity term and inertial term as 1:1 to calculate vector for the new flow front. The algorithm is implemented in CastView based on the fill pattern algorithm for the die casting process.

Two examples are chosen for test. The experiment and numerical simulation are from the paper by Wallace, et. al. (2000). Both parts are flat plates with different thickness. One is in 0.5 inch and the other one is in 0.25 inch. The dimensions of the two parts for squeeze casting are shown in Fig. 7.9. The flow pattern of the molten metal into the die cavity was measured by the use of incomplete shots and the numerical simulation was performed using Procast.

The CAD models of these two parts are built on Unigraphics using the dimensions in Fig. 7.9. The 3D rendering pictures in Unigraphics is showed in Fig. 7.10.

The experimental results and Procast results of 0.5 inch plate are listed in Fig. 7.11. It can be seen that at the early stage when melt fills the cavity, the flow advances like an arc with some bias at one side due to gravity effect. The flow front then becomes even and advances steadily until it reaches the end.
The CastView results are shown in Fig. 7.12. When the flow starts to enter the cavity, it forms a big bump around the gate with some bias at one side, which is very similar to the pattern in Fig. 7.11. The flow then advances more evenly and steadily to the end. It can be seen that the results in Fig. 7.11 and Fig. 7.12 have good agreement, which indicates the suitability of fill pattern reasoning for squeeze casting.

Fig. 7.9 Dimensions of two plates for squeeze casting. The difference of the two parts is the plate thickness and gate shape. (Wallace, et. al. 2000).
Fig. 7.10 Two plates with different thickness for squeeze casting. Upper: 0.5 inch plate; lower: 0.25 inch plate.
Fig. 7.11 Experimental results and simulation results from Procast for 0.5 inch plate.

(Wallace, et. al. 2000).
Fig. 7.12 CastView result for 0.5 inch plate. At the beginning of cavity fill, the flow forms a bump from gate. The flow front then becomes flat and steady.

Another example is the 0.25 inch flat part. The paper only provides the simulation result for the 0.25 inch flat part without experimental result. Fig. 7.13 shows the numerical simulation result of the thin plate by Procast. The flow fills the fan gate then starts to fill the cavity in a much more uniform front. The flow front then advances steadily to fill the whole cavity. The corresponding results from CastView for the same part are shown in Fig. 7.14. It can be seen that once the flow enters the cavity, the flow front is in a uniform
The flow front then fills the cavity steadily from left to right until it hits the end line. Though there is still a little disturbance at the flow corner, the overall flow pattern from CastView is very similar to that from Procast. The paper does not provide the run time for the numerical simulation by Procast but a reasonable guess is 2 ~ 3 hours on a typical PC. Given the analysis time for this part by CastView is 2 ~ 3 minutes, the efficiency of geometric reasoning for fill pattern is remarkable.
Fig. 7.13 Fill pattern by numerical simulation using Procast for 0.25 inch plate. The flow is in a uniform front to fill the whole cavity. (Wallace, et. al. 2000).
Fig. 7.14 Result of fill pattern for 0.25 inch plate from CastView. The flow front is uniform and advances steadily until it hits the end.
7.6 Conclusions

1) It is possible that the fill pattern algorithm for die casting can be applied to slow fill processes with slight modification.

2) It can be seen from the Navier-Stokes equations that the major difference between die casting, gravity casting and squeeze casting is the dominant term in flow
calculation. The algebra further shows that the dominant term is the combination of gravity term and inertial term.

3) There are problems for Elfakharany’s algorithm for gravity casting. This algorithm was redesigned and implemented based on the algorithm for die casing process.

4) A gravity casting case was chosen to test the algorithm improvement. Analysis is performed using the old algorithm and new algorithm then the results are compared to experiment results and numerical simulation results. The comparison clearly shows the validity and efficiency of improvement in new algorithm.

5) Two squeeze casting examples are shown with comparison between short shot results, numerical simulation results and CastView results. The fill patterns are very similar, which indicates the efficiency of applying geometric reasoning technique on slow fill processes.
CHAPTER 8

CONCLUSIONS AND FUTURE WORK

Die casting is an important net shape manufacturing process. In this process, liquid metal is injected at high pressure and high speed into a metal die cavity, with subsequent solidification into useful shapes. The applications of die castings have increased significantly in the recent years, especially in the automobile, aerospace, electronics and medical apparatus industries. The die casting industry is now in fast development and will see its further expansion in the future.

There are usually two concerns for die casting designers and engineers, the thermal characteristics and fill pattern. The temperature distributions of die and part are closely related to stress/strain, distortion, casting quality, die life, etc. In a good process design, the die temperature distribution should be as even as possible to reduce stress/strain and the part ejection temperature should be below the melting point of casting alloy for safe ejection. It is very desirable to have temperature pattern of die and ejection temperature of part when the process reaches the quasi steady state because they help the cooling/heating design and cycle design.
The typical way to obtain the thermal profile of die/part is numerical simulation for dynamic temperature change. This technique usually sets up equation system based on dynamic heat transfer and solidification then solves the equation system for the temperature change. However, this is very time consuming since to reach the quasi steady state, literally hundreds of cycles of simulation are needed. In addition, the information that numerical simulation provides is beyond that needed for cycle and die cooling/heating design. Thus, numerical simulation is not suitable in the conceptual design stage.

Equilibrium temperature solution may be better than dynamic temperature solution for cycle and die cooling/heating design. The equilibrium temperature is defined as the time average temperature over a cycle of die and the ejection temperature of part after the process reaches the quasi steady state. To obtain the solution, the overall heat balance over the whole cycle is considered. The heat transfer concepts in real steady state are introduced into the quasi steady stage of die casting process. Since it does not solve the heat transfer in the cycles before the process reaches quasi steady state, this method is very quick and supports the interactive process design. This is particularly important in early design stage because there may be many alternative designs and the engineer wants to explore all designs using a quick computer tool.
Another concern of die casting engineers is the fill pattern of liquid metal in die cavity. The fill pattern also plays a very important role for casting quality. If the flow is not controlled well, it would cause many flow-related defects. One of the common defects is gas porosity, which usually happens at the last region to be filled. Thus, it is very important to evaluate the fill pattern of a process before the production.

Currently, the prevailing method to obtain the fill pattern is numerical simulation. This technique utilizes principles of heat transfer, solidification and fluid mechanics and is usually based on Finite Element Method (FEM), Finite Difference Method (FDM) or Boundary Element Method (BEM). Due to the complexity of the equation system to be solved, numerical simulation is very time consuming. The computation time for a case using numerical simulation is usually hours or days. This is not suitable for quickly evaluating designs and limits the utility in the early stages of development.

An alternative method to obtain fill pattern is qualitative analysis which is based on geometric reasoning. In this method, the flow behavior is calculated using the cavity geometric information. The flow vector is recomputed when the flow hits obstruction. The mass conservation is considered within neighborhood. Some assumptions and simplifications are made to reduce the computation. Though the accuracy is not as high as numerical simulation, this method provides a quick and efficient way to predict the fill pattern in cavity.
The Ohio State University developed an algorithm based on geometric reasoning previously. This algorithm has proved a simple but efficient way for fill pattern. However, there are some shortcomings in this algorithm and occasionally the analysis results are not correct. Thus it is very necessary to correct the problems and improve the old algorithm.

Another objective of this research is to develop an algorithm for fill patterns of slow fill processes, such as gravity casting and squeeze casting. The flow behavior of these slow fill processes is different from that of die casting process. However, from the qualitative analysis point of view, the majority difference is the driven force for flow. Thus it is possible to apply the algorithm for die casting process on these slow fill processes by making small modification.

In this research, an efficient algorithm to compute equilibrium temperature of die and ejection temperature of part has been developed. This algorithm was implemented in the program CastView based on Finite Difference Method (FDM). Compared to days of run time for numerical simulation for dynamic temperature solution, the typical run time using this algorithm for equilibrium temperature is only a few minutes. The results from this algorithm have been compared to those from numerical simulation. Both results have good agreement, which indicate the validity and efficiency of the algorithm.

The old fill pattern algorithm for die casting process based on geometric reasoning has been redesigned. Many shortcomings in the old algorithm were fixed and improved. The
new algorithm includes some other considerations which affect the flow behavior. The results produced by the new algorithm have been compared to those from numerical simulation and water analog model. The comparison clearly shows the improvement of the new algorithm.

The fill pattern algorithm for die casting process has been modified to applied on the slow fill processes. At this stage, two processes are considered, gravity casting and squeeze casting. The dominant terms for these two processes are determined and reflected in their algorithms. The analysis results generated by these algorithms have been compared with experiment results and the results from literature resource and they showed good agreement.

The specific contributions made by this research are:

- Developed a mathematical algorithm to compute the equilibrium temperature of die and ejection temperature of part. The heat transfer concepts in steady state are introduced to quasi steady state to calculate the heat balance over a cycle.

- Special attention was paid to the calculation of heat released from the part. A few models have been tried but finally the combined asymptotic and surrogate model was chosen.

- Composite heat transfer at interface, average temperature at different stages,
cooling line effect, spray effect and die splitting at parting surface were addressed. Computational efficiency was considered by applying some special methods to reduce the computation.

- Implemented the algorithm in the program CastView, providing a quick tool to evaluate the cycle and die cooling/heating design. Compared to days of run time using numerical simulation for dynamic solution, the run time of CastView for equilibrium temperature is only a few minutes.

- Developed a method to transfer date between FDM model and FEM model, thus related the equilibrium temperature data with numerical simulation temperature data.

- Redesigned the old fill pattern algorithm for die casting processes. Flow speed, flow resistance, flow potential and influence within neighboring flows were included. The method to compute new vector when flow hits an obstruction and the searching way for available vectors were improved.

- Implemented the new algorithm for fill pattern in the program CastView, providing a quick and efficient tool to evaluate fill pattern in cavity. The run time for a typical case on CastView is only ~10 minutes while that on numerical simulation packages is hours or days.
• Relating the fill pattern results from geometric reasoning with numerical simulation results.

• Relating the fill pattern results from geometric reasoning with water analog studies.

• Determined the dominant terms on flow behavior for die casting process, gravity casting process and squeeze casting process from Navier-Stokes equations.

• Developed fill pattern algorithms for gravity casting process and squeeze casting process by modifying the algorithm for die casting process.

• Implemented the algorithms in the program CastView. The typical run time is only a few minutes.

• Relating the fill pattern results of gravity casting and squeeze casting using geometric reasoning with the information in literature resource.

The tool for equilibrium temperature can quickly compute the overall temperature distribution of die and ejection temperature of part. This helps the user evaluate the effect of cooling and heating and verify the cycle design. Due to the efficiency of this tool, the user can explore numerous alternative designs, which is very difficult for numerical simulation packages.
The tool for fill pattern analysis provides an alternative means other than numerical simulation. The advantage of geometric reasoning is the computational efficiency. It can produce a fill pattern prediction in a few minutes. This is particularly useful in the early design stage. Compared with the old fill pattern algorithm, the new algorithm improves the calculation and considers more factors. The analysis results are more accurate but the computation time is only increased slightly.

Based on the analysis of Navier-Stokes equations and geometric reasoning technique, the fill pattern algorithms for gravity casting and squeeze casting can provides quick evaluation for fill patterns of these two processes. Furthermore, the development of the algorithms suggests the similar analysis procedure can be applied on other slow fill processes, such as low pressure die casting and semi-solid casting.

The limitation of the analysis for equilibrium temperature is that we applied the steady state on the quasi steady state. This is to simplify the computation but also introduces some problems. The heat transfer is computed based on the time average temperature. However, the actual heat transfer happens on the dynamic changing temperature. The discussion in Section 2.8 makes some remedy but it does not eliminate the problem.

The limitation for fill pattern analysis is that this method is solely based on geometric reasoning thus there is no or only little physics behind it. There is no strict consideration
of mass conservation, momentum conservation and energy conservation. There is no calculation for heat transfer and solidification during flow analysis. This lack has inevitable effect on the result accuracy.

Extensions to the research on computing equilibrium temperature include the improvement of the current method to calculate heat released from part and better way to address the average temperatures in different cycle stages. This would improve the accuracy of the calculation, especially for the ejection temperature of the part.

Future work for fill pattern reasoning includes better definitions for flow resistance and flow potential. For example, flow resistance calculation should address the threshold of wall thickness which can affect the resistance. In the example of Fig. 5.4, if the thickness of C is large enough, it would not have resistance for the flow. Future work also includes the development of functions to define gate, biscuit and runner system. The fill pattern analysis will be performed on the whole cavity including part, gate, runner system and biscuit.

Other future work includes the further development of fill pattern algorithms for other slow fill processes, such as low pressure die casting and semi-solid casting. Following the procedure to build algorithms for gravity casting and squeeze casting, the dominant terms for those processes can be determined and their algorithm can be developed.
REFERENCE:

1. Allchin, T., Quality assurance with modern CNC die casting system, Die Casting Engineer, 1990, March/April, p34-38.


8. Chen, Xiaorui, Graphical user interface for cooling line functions and surface rendering, Thesis for Master’s degree, The Ohio State University, 2002.


