

COMPUTER AIDED ENGINEERING FOR INJECTION MOULD COOLING SYSTEM DESIGN

BY

NIALL MORAN

B.SC.(ENG.) DIP.ENG. MIEI

This thesis is submitted as the fulfilment of the requirement for the award of
Master of Engineering by research to:

DR. M. A. EL-BARADIE

SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING

DUBLIN CITY UNIVERSITY

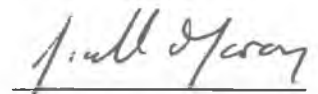
AUGUST 1998

DECLARATION

I hereby declare that all the work reported in this thesis was carried out by me at Dublin City University during the period from October 1995 to April 1998.

To the best of my knowledge, the results presented in this thesis originated from the presented study, except where references have been made. No part of this thesis has been submitted for a degree at any other institution.

Signature of Candidate

A handwritten signature in dark ink, appearing to read 'Niall Moran', is written over a horizontal line.

Niall Moran

ABSTRACT

COMPUTER AIDED ENGINEERING FOR INJECTION MOULD COOLING SYSTEM DESIGN

Niall Moran

The time taken in the cooling stage, of a typical injection moulding cycle, is a large factor in the productivity and efficiency of a plastic manufacturing process, and for this reason, must be minimised. In order to do this a cooling system is employed throughout the mould core.

This thesis describes the development and implementation of a PC based analysis system that can be used to optimise the size and position of injection mould cooling systems. The software is fully '32-Bit', operating on 'Windows' platforms, and uses graphical methods for input and output operations. The two-dimensional geometry of the mould is supplied using AutoCAD 14 and 'Active-X Automation'. The analysis programs were written using 'Fortran PowerStation' and the user interface using 'Visual Basic'.

To employ the optimisation process the 'Boundary Element Method' was used to predict the temperature profile throughout the mould. This method is compared to an analytical procedure and the "Finite Element Method", by analysing a simple benchmark problem. The results of the "Boundary Element Analysis" were extremely accurate and in close agreement with the analytical solutions.

This thesis presents the method by which the temperature profile, throughout an injection mould, can be predicted, and applies this method to a particular example. Also presented are the experimental results of a test mould that was manufactured to produce simple square plastic parts. The results of the numerical analysis agreed with experimental results to within 6%.

ACKNOWLEDGEMENT

The author wishes to express his appreciation to Dr. M.A. El-Baradie for his supervision and guidance during the course of this project.

Sincere thanks are also extended to the workshop technicians for their work on the test mould and their help in completing the testing.

The author would also like to thank Alpha Plastics for their help in the experimentation.

TABLE OF CONTENTS

Chapter	Description	Page
1	Introduction	
1.1	Scope of Present Work	1
1.2	Injection Moulding	2
1.2.1	Injection Moulds	4
1.2.2	Plastic Materials	6
2	Literature Survey	14
3	Injection Mould Cooling System Design	
3.1	Introduction	26
3.2	Defects in Plastic Parts	27
3.2.1	Warpage	27
3.2.2	Hot Spots and Sink Marks	28
3.3	Cooling System design	30
3.3.1	Cavity Surface	32
3.3.2	Cooling Lines	36
3.3.3	Mould Exterior	37
4	Computational Heat Transfer	
4.1	Introduction	41
4.2	The Finite Difference Method	42
4.3	The Finite Element Method	43
4.4	The Boundary Element Method	45
4.4.1	Steady-State Thermal Boundary Elements	46
4.4.2	Transient Thermal Boundary Elements	50
4.5	Computer Implementation of Boundary Element Method	53
5	MouldCOOL: Injection Mould Cooling System Analysis Software	
5.1	Introduction	60
5.2	Geometry Base (AutoCAD Interface)	64
5.3	Mesh Base	68

5.4	Analysis	70
5.5	Post-Processing	75
5.6	MouldCOOL Interface	78
5.6.1	File Menu	78
5.6.2	Edit Menu	79
5.6.3	View Menu	79
5.6.4	Pre-Processor Menu	79
5.6.5	Analysis Menu	80
5.6.6	Post-Processor Menu	80
5.6.7	Quick menu	80
6	Test Mould Design and manufacture	
6.1	Introduction	82
6.2	Mould Cavity and Core	87
6.3	Plastic Part Ejection	88
6.4	Feed System	88
6.5	Mould Cooling	89
7	Test Results and Analysis	
7.1	Introduction	91
7.2	Equipment and Instrumentation	92
7.3	Test Procedure and Results	94
7.4	Computer Simulation of Test Mould	97
7.5	Analysis of Results	108
8	Conclusions and Recommendation for Further Study	
8.1	System Limitations	109
8.2	Recommendations for Further Study	109

APPENDIX 1 - Gauss Quadrature weights for one-dimensional integration.

APPENDIX 2 - Program Listings

APPENDIX 3 - Mould Drawings

APPENDIX 4 – Bibliography

NOMENCLATURE

Symbol	Units	Description
H	m	Cavity thickness
T	K	Cavity wall temperature
T ₀	K	Melt temperature
α	m ² /s	Thermal diffusivity
h _{cv}	W/m ² K	Cycle averaged heat transfer coefficient
t _c	seconds	Cooling time of melt
Q	W	Heat transfer
q	W/m ²	Heat flux
K	W/mK	Thermal conductivity
S		Shape factor
u	K	Temperature

CHAPTER 1

INTRODUCTION

1.1 Scope of the Present Work

The objective of this research project was the development of software for the computer-aided analysis of injection mould cooling systems. The software can be used by an injection mould designer to optimise the size and position of cooling lines throughout the mould core. The system allows an injection mould manufacturer to produce better quality products at a more productive rate. In order to complete the objective, the research was divided into the following categories.

- Development of a theoretical model for the thermal analysis of injection moulds. A number of methods, such as the 'finite element method' [31], the 'finite differences method' [33] and the 'boundary element method' [35] are compared to establish which is most applicable. The comparison is discussed in chapter 4 of this thesis.
- Development of a computer program using the models established. The analysis programs were written in FORTRAN 90, with the main interface written using Visual Basic and AutoCAD as the pre-processor. The workings of these programs are detailed in chapter 5.
- Design and manufacture of a mould for experimental comparison between actual mould temperatures and those predicted by the software. The actual mould used is shown in Appendix 3. Chapter 6 describes the design and manufacture of this mould. Chapter 7 describes the tests carried out and the results obtained.

This thesis is divided into eight chapters. The first chapter presents an introduction to cooling system design of injection moulds. This chapter also looks at the different types of plastics used in the injection moulding industry as well as their thermal properties. The second chapter presents a survey of the work completed, in this area, by other authors. The third chapter looks at the accurate design of injection mould cooling systems and presents the mathematics needed to implement a complete design procedure. Chapter four presents the mathematics of a number of numerical methods that can be used to predict temperature profiles within an injection mould. This chapter presents a comparison between the different methods, from the point of view of applicability to an

injection mould analysis, and suggests the most appropriate method to be the 'Boundary Element Method'. A detailed look at the accuracy and convergence of the boundary element method is also presented, in this chapter. Chapter five gives a detailed description of the software developed to implement the procedures, discussed in previous chapters. In chapter six the design and manufacture of a double cavity test-mould is detailed. Chapter seven presents a comparison between results obtained from experiment and the present software. This chapter also describes the main factors influencing an efficient cooling system. Chapter eight details the main conclusions derived from the project and explains any limitations of the software, as well as recommendations for future work.

1.2 Injection Moulding

The injection moulding process is one in which hot molten polymer is injected at high pressure and temperature into a metal cavity to form the shape of the required part. This process can be used to produce a variety of complex objects from a number of different thermoplastic materials and is considered the most important industrial method for the production of plastic parts, for the following reasons:

- The process can be operated in a highly automated mass production environment.
- Highly complex shapes can be made with great speed.
- A high level of accuracy in repetitively producing the same object can be obtained.
- A number of different materials can be used to create parts with varying properties.

In general, an injection-moulding machine will consist of the following parts:

- Hopper. The conical container for feeding the solid plastic pellets into the injection-moulding machine.
- Plunger or Screw. The device used to force the plastic into a heating cylinder, for melting, and then into the mould.
- Platen. The back plates of the mould used to connect the mould to the machine.

The injection moulding process is completed in the following stages.

- **Mould Closing.** This stage should be as quick as possible, but not so fast as to cause damage to the parting surface of the injection mould.
- **Mould Filling.** The plastic melt is forced to fill the cavity of the mould.
- **Mould Packing.** After the plastic is injected, the pressure is increased to consolidate the plastic in the cavity.
- **Cooling.** This stage consists of cooling the plastic from its injection temperature to its ejection temperature.
- **Ejection.** In this stage the mould is opened and automatically ejected.

The approximate relative time spent in each of the stages of a typical moulding cycle is illustrated in figure 1.1.

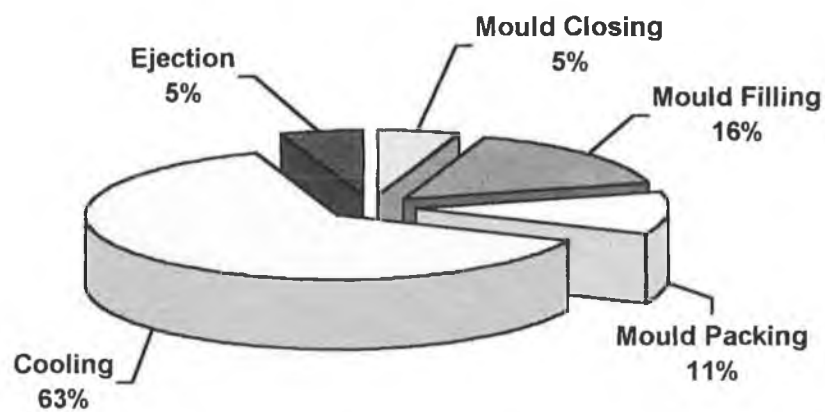


Figure 1.1 – Typical Injection Moulding Cycle

The areas of concern for mould designers are in the injection and cooling stages. Many authors, [1] to [10], have studied the mould filling process to avoid premature scorching of rubber compounds before the mould is completely filled. The main area of concern, however, is in the cooling stage, since this is the most time consuming. To cool the mould, cooling lines are drilled through the core and cavity plate, through which a coolant is conveyed to extract heat from the cavity.

In the years before the dawn of computer technology, the design of injection moulds out based on the experience of the designer. The lack of precise design techniques led to the following problems,

- Premature Scorch – solidification of plastic before the injection process is complete.
- Material Defects – defects due to the difference in cooling rates throughout the plastic part.

Computer technology allows us to analyse the injection moulding process in many different ways so that a better and more efficient part can be produced.

The optimisation of the size and position of the cooling lines, to give the most efficient cooling process is the main challenge of the computer-aided engineer in the design of the injection mould cooling system.

1.2.1 Injection Moulds

An injection mould consists of a number of parts, the design of each influencing the others. For this reason, the methodology of mould design must be understood so that an efficient cooling system can be incorporated.

The main elements of an injection mould (see figure 1.2) are as follows.

- Clamp Plates - one on each side of the mould attaching it to the injection-moulding machine.
- Cavity Plate - this plate contains the fixed side of the cavity.
- Core Plate - this plate contains the moving element of the cavity.
- Support Plates - these plates are used to guide and support the movement of the moving portion of the mould.
- Ejector Pins - these pins are used to eject the plastic part away from the moving side of the mould.
- Ejector Plate - this plate is used to support and guide the ejector pins.
- Sprue - the part of the mould through which the plastic is injected.

- Runners - the channels through which the plastic flows from the sprue to the cavity.
- Gates - the opening connecting the runner and the cavity.

Figure 1.2 shows a typical injection mould with some of the most important parts labelled.

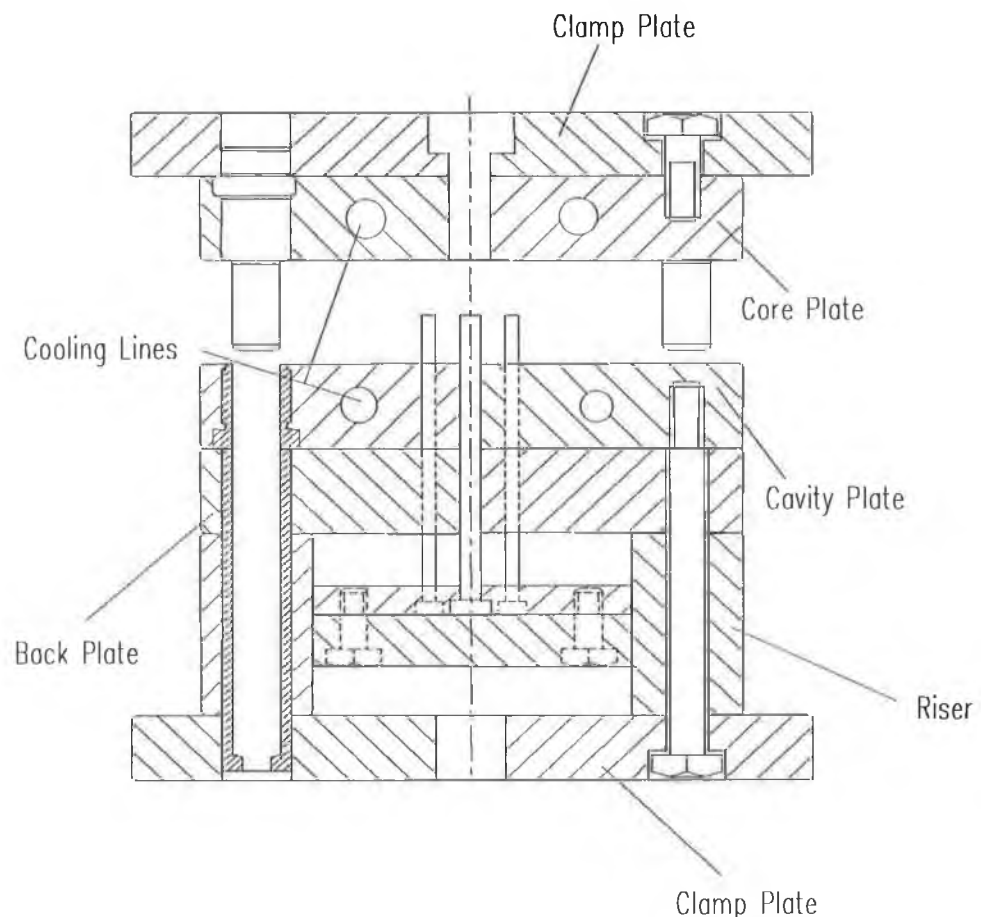


Figure 1.2 – Typical Injection Mould Parts

Plastic is injected through the sprue and flows through the runners into the cavity or cavities. When the plastic has reached its ejection temperature the moulded component is ejected. The ejection mechanism is operated when the moving half of the mould is retracted causing the ejection pins to push forward, forcing the plastic component away from the mould.

It is important to analyse the way in which the cooling system design is incorporated into the overall injection mould design. For a long time, the cooling system consisted of drilling holes for cooling lines, after the mould was

completed, hence producing an inefficient cooling system. Nowadays it is important to determine the optimum size and position of these cooling lines before the mould is made. Figure 1.3 shows the overall design methodology for an injection mould.

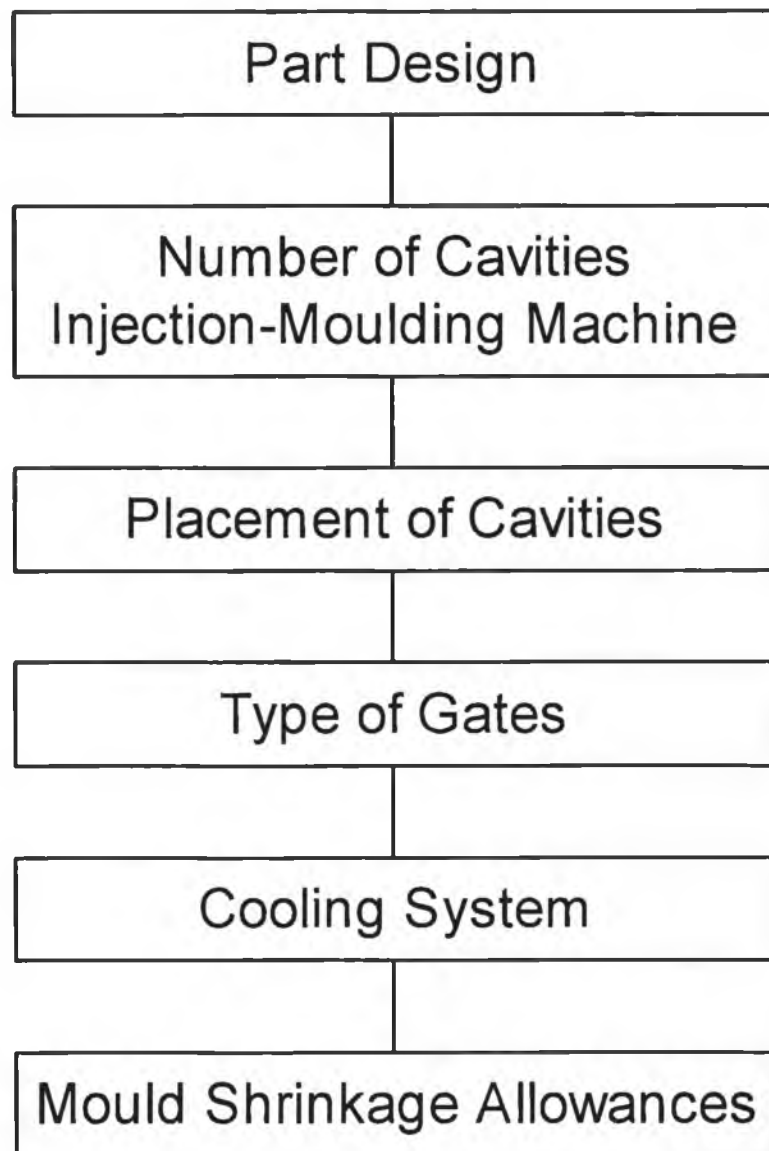


Figure 1.3 – Injection Mould Design Methodology

1.2.2 Plastic Materials

Before setting out on the design of an injection mould, it is important to first look at the plastic material to be used. Any number of factors can and will effect the decision, although mechanical properties are the most important. In many cases additives will be added to the plastic, for example rubber or glass, depending on

the properties required. Table 1.1 shows details for a number of thermoplastic materials [41].

Table 1.1 - Thermoplastic Materials

Name	Description	Applications
General-purpose polystyrene	hard, stiff, transparent, brittle	packaging, lighting fittings and toys
Toughened polystyrene (rubber-modified)	tougher than general-purpose polystyrene	vending cups, dairy produce containers, refrigerator liners, toys - particularly model kits for assembly
ABS	tough, stiff, abrasion resistant	dinghy hulls, telephone handsets, housings for vacuum cleaners and grass mowers
Un-plasticised PVC	hard, tough, strong and stiff, good chemical and weathering resistance, self-extinguishing and can be transparent	pipes, pipe fittings, and rainwater goods, wall cladding and curtain rails
Plasticised PVC	lower strength and increased flexibility depending on amount and type of	insulation of wire for domestic electricity

	plasticiser, compared with un-plasticised PVC	supply, domestic hose pipes, soles of footwear
Polyolefins	distinguished by excellent chemical resistance and electrical insulation properties	
Low-density polyethylene (918-935 kg/m ³)	exploits toughness at low temperatures, flexibility and chemical resistance in pipes for chemical plant	low-loss electrical wire covering, blow-moulded and large rotationally moulded containers, and packaging film
High-density polyethylene (935-965 kg/m ³)	much stronger and stiffer	dustbins, milk bottle crates and mechanical handling pallets
Polypropylene	has good fatigue resistance and can be used at higher temperatures than polyethylene; the copolymer version is more impact resistant than the homopolymer at low temperatures	pipes and pipe fittings, beer bottle crates, chair shells, capacitor dielectrics and cable insulation, twines and ropes
Acrylic	completely	domestic baths,

(PMMA)	transparent, not attacked by ultraviolet light (UV), stiff, strong and does not shatter	lenses, and illuminated signs
Modified PPO	tough, stiff, strong, transparent, and good electrical insulation properties	connectors and circuit breakers in electrical equipment, and for office machine housings
Polysulphones	stiff, strong, excellent dimensional stability, transparent, burns only with difficulty and without smoke	passenger service units in aircraft, components for high-temperature duty in electrical and electronic equipment
Nylons	stiff, strong, tough and abrasion resistant; absorption of moisture increases toughness, but reduces stiffness and dimensional stability	gears, bushes, cams and bearings; glass-filled nylon in power tool housings
Polyacetals	stiff, strong, extremely resistant, and abrasion resistant	taps and pipe fittings, light-duty beam springs, gears and bearings

PolyCarbonate	tough, stiff, strong, transparent, and good electrical insulation properties	street lamp covers, feeding bottles for babies, safety helmets
PTFE	outstanding electrical properties and corrosion resistant, exceptionally low coefficient of friction, tough, can be used continuously at 250°C	bearing surfaces of journal bearings, coatings for cooking utensils, high-frequency high-temperature cable insulation

Table 1.1 - Thermoplastic Materials

For the case of the cooling system, the thermal properties of the thermoplastic are of relevance. Table 1.2 shows the processing and mould temperatures along with shrinkage allowance for a number of different materials. The most important aspect of the thermal properties of injection moulding plastics is that they change with temperature. Table 1.2 shows average values although even these can vary as is shown by the specific heat capacity for polypropylene.

To complete any analysis of the cooling system, the variation of these properties with temperature must be known. The thermal properties of any polymer can be represented by a linear equation, such as those given in equation 1.1.

$$\begin{aligned}
 \rho &= mT + c \\
 C_p &= mT + c \\
 k &= mT + c
 \end{aligned}
 \tag{1.1}$$

The values of the constants, m and c , for a number of thermoplastics are shown in table 1.3.

- The change in water temperature should not exceed about three Celsius; this can be achieved by ensuring turbulent flow of coolant through the cooling lines.

In analysing an injection mould cooling system, the modes of heat transfer involved should be understood. As is the case with any heat transfer problem, the boundary conditions and methods of analysis can change due to assumptions that can be made. An example of this is the exterior of the mould. It could be assumed that the entire exterior is subject to natural convection with ambient air. The convection coefficient in this case can be evaluated using a nusselt relation. Another valid assumption may be to assume a temperature profile over this part of the mould or to assume an insulation condition over the parts in contact with the platen and convection over the rest. In analysing the different approaches, the following conditions are assumed.

- Natural convection between mould exterior and ambient air.
- Radiation from the mould exterior can be neglected since it is negligible compared with natural convection.
- Forced convection between cooling lines and coolant.
- The heat transfer within the cavity is transient cyclic and dependent on conduction within the mould core.

The final heat transfer mode is conduction within the mould core. This is modelled using a numerical technique incorporating the previously mentioned modes of heat transfer as boundary conditions. The different analysis techniques for doing this are detailed in section 3 of this report.

Hence, the general procedure for an injection mould cooling system analysis is as follows.

- A two-dimensional section of the mould is discretised into a number of linear elements.
- Boundary conditions are applied to the surfaces of the mould section.
- The temperatures and heat fluxes are calculated at the boundaries.

- The efficiency of the cooling system is calculated. The efficiency is calculated using the following formula.

$$\eta = \frac{E_c - E_e}{E_c}$$

Where E_e is the heat lost to the environment through the mould surfaces and E_c is the heat lost through the cavity walls.

- The size and positions of the cooling lines are the altered to increase the efficiency.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Until recently the design of injection mould cooling systems was reliant on the skill and judgement of the mould designer and not on any formal procedure. The size and position of the cooling circuit was limited by the mould design, that is, the position of the ejector pins, guide bars, etc. With the realisation that many of the defects found in plastic parts can be attributed to in-efficient cooling, the importance of a detailed analysis of the cooling system design was acknowledged by many mould designers. With this came the dawn of the computer age and an increasing demand for mould designers to develop software for analysing and optimising the injection mould cooling system.

2.2 Injection Mould Cooling System Analysis

One of the first applications of numerical mathematical modelling, for the solution of heat transfer within injection moulds, was by Kenig and Kamal [1] in 1970. In this paper, a single cylindrical mould was analysed using a polar finite difference approximation. The author looked mainly at temperature profiles within the mould for different thermoplastics. In addition, the procedure employed took into consideration the variation of polymer thermal properties with temperature and pressure. The governing equation suggested was.

$$(P + \pi)(V - \varpi) = RT \quad (2.1)$$

Where, P , V and T , represent pressure, specific volume and temperature, respectively, and, $\pi = 3282$ bar, $\varpi = 1143$ kg/m³, and $R = 296.5$ J/kgK.

The results of this numerical analysis compared accurately with experimental results, but the analysis was limited to this particular cylindrical mould.

In 1980, K.K. Wang [2] tackled the idea of developing a methodical approach to the overall design of injection moulds. This early work by Wang was mainly concerned with the simulation of plastic flow into the mould cavity. This analysis was based on a one-

dimensional representation of the cavity using a finite difference approach. The finite difference approach involved replacing differentials in the defining differential equation with differences. The method involves the solution of a system of equations that can be solved to produce the result over the entire domain. The approach also took account of the rheological properties of the polymer.

In order to allow for the variation in thermal properties of the polymer melt with respect to temperature and pressure Wang [2] suggested the following correlation for polystyrene.

$$(p + 27000)(v - 1.422) = 11.18T + 5134 \quad (2.2)$$

Where p , v and T represent pressure, specific volume and temperature, respectively.

One of the first applications of an integral method for mould analysis was by Barone and Caulk [3] who applied a boundary integral method to a simple mould. This application was for heating of mould cores.

In 1985, Colin Austin [4] introduced the idea of mould cooling and the effects that a poorly designed system could have on the finished plastic part. He explained the two main functions of a cooling system:

- To remove heat from the cavity at the required rate.
- To remove the heat at a uniform rate.

In his paper, Austin described the heat transfer mechanisms through which heat is transferred throughout a typical injection mould. The idea of turbulent coolant flow was also addressed. It was suggested that the coolant should be run in the turbulent range, otherwise the heat flow would be inefficient.

In 1986, Wang and Kwon [5] presented the first computer program for the analysis of injection mould cooling as part of the Cornell Injection Moulding program. This program analysed the steady state temperature profile throughout the mould using the 'boundary element method' [37] and a cycle averaged heat transfer coefficient along the cavity.

The boundary element method [37], [38], [39] is a mathematical method that is used to model engineering problems, such as heat transfer. The method consists of relating the temperatures at particular boundary points, by analytical functions. The analytical functions are called fundamental solutions and can be derived for the particular problem [38]. Once the analytical functions are developed for each boundary point, the solution can be solved for the entire boundary and any internal points for which the solution is required.

The cycle averaged heat transfer coefficient was introduced as a method of estimating the boundary condition at the cavity. This was essential since the heat transfer at the cavity was changing with respect to time within each injection moulding cycle. The idea was to average the heat flux at the cavity surface over the entire cooling time and represent this as a heat transfer coefficient. The coefficient could be derived using a one-dimensional analysis of the cavity and resulted in the following equation.

$$h_{cv} = \frac{\int_0^{t_c} q(t) dt}{t_c} \times \frac{1}{(T_m - T_w)} \quad (2.3)$$

Where T_m and T_w represent melt and cavity wall temperatures, respectively.

The software developed by Wang ET Al [7] used equation 2.3 as the boundary condition at the cavity surface and forced convection at the cooling lines to apply a boundary element solution to a two-dimensional section of the mould. The program optimised the cooling system using a two-dimensional analysis and then used a three-dimensional analysis to confirm the results.

In 1988, T.H. Kwon [6] published results of COOL2D and COOL3D a two-dimensional and three-dimensional 'boundary element analysis' program for analysing injection mould cooling systems. The method proceeded by application of the 'cycle average heat transfer coefficient' but analysed the results using the method of shape factors. The shape factor method involves calculating the ratio of heat flux to temperature difference at the cooling lines and at the cavity. Any difference in these values signifies a loss of heat to

the atmosphere and can be a very reliable way of indicating the performance of a cooling system.

Wang and Turng [7] presented a method of developing the cycle averaged heat transfer coefficient by the application of an analytical solution across the cavity. The method also took account of the fact that the thermal properties of the polymer will change with respect to temperature. This variation was taken account of by using the following procedure.

- The system is analysed using a constant specific heat capacity, determined as the slope of the straight part of the graph in figure 2.1.
- Using the cavity wall temperature and the initial melt temperature, the average polymer temperature at the end of cooling can be determined.
- With this knowledge, the change in enthalpy can be determined from figure 2.1 and hence a new value for the specific heat capacity can be evaluated and used for a new analysis.

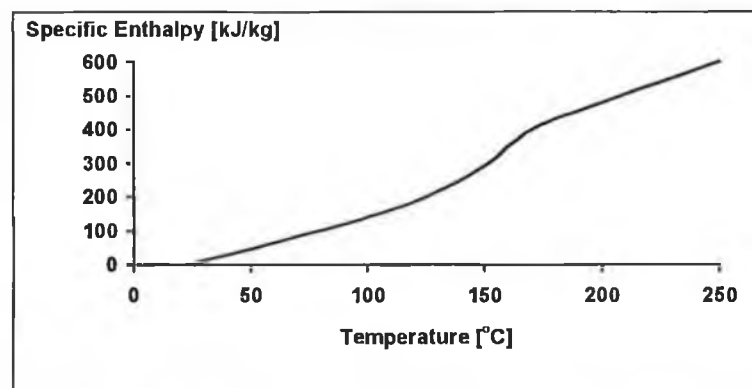


Figure 2.1 - Specific Enthalpy versus Temperature for Polypropylene

The transient analysis of injection mould cooling systems was completed by Chen and Chung [11] in 1994 using the 'dual reciprocity boundary element method' [39]. When the governing equation of heat diffusion is transferred into an integral equation a number of volume integrals are established. For the steady state analysis, the volume integrals are transformed into boundary integrals using Green's theorems [37].

When the transient conduction equation is transformed one volume integral is left. The dual reciprocity method is just one method of reducing this volume integral down to a boundary integral.

The analysis was completed by assuming the cavity and mould to be two separate domains dependent on each other. The mould analysis completed by boundary element method and the cavity analysis by finite difference in a coupled approach. This paper compared results of the two-dimensional program to those obtained experimentally.

2.3 Numerical Techniques

The decision on which numerical method to use is very important and should be governed by the following points.

- Applicability - It is very important that the numerical method used is applicable to a wide range of mould geometry and not just certain types.
- Accuracy - The accuracy of the method should be such that numerical results compare adequately to experimental results.
- Speed - It is important that the method used be as fast as possible, without compromising on the applicability or accuracy.

Three methods that are generally applicable to heat transfer problems, '*finite difference methods*', '*finite element methods*' and '*boundary element methods*'. Each of these methods set out to solve the general equation of heat diffusion, equation 2.4.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{\alpha} \frac{\partial T}{\partial t} \quad (2.4)$$

The finite difference method involves dividing the domain, to be analysed, into a rectangular grid. Then, by replacing the derivatives in equation 2.4 with differences, an equation can be derived for each node. Once these equations are gathered together in matrix form, a solution can be obtained for the entire domain. The 'finite element' and 'boundary element methods', work on converting equation 2.4 into an integral equation. Once this is done, the domain can be meshed using regularly shaped elements, over

which the integrals can be evaluated. The entire solution is obtained by summing these solutions. The only difference between the finite and boundary element methods is that the finite element method produces volume integrals, and hence the entire volume of the domain must be meshed. Whereas, with the boundary element method, these volume integrals are transferred into boundary integrals, and hence, only the boundary of the domain, must be meshed.

2.4 Boundary Element Method V's Finite Element Method

The comparison between the boundary element method and the finite element method, as applied to numerous problems, has been the topic of extensive research since the dawn of both methods. In 1977, Brebbia ET Al [12] presented results on the comparison of the two methods for potential problems. Results were produced for a number of potential problems, showing the boundary element method to have a much higher computational efficiency than the finite element method.

In 1989 Jon Trevelyan [13] produced a paper on the comparison between the boundary element method and the finite element method, suggesting that the boundary element method, when applicable, was a better choice for the following reasons:

- Ease of use - Since only the boundary of the domain is meshed, there are much less elements to prepare than those required for the finite element method. In addition, the mesh is much easier to create, this is because the dimension of the mesh is always one less than the defining problem, whereas, with the finite element method, the mesh and the problem have the same dimension. This is because, for the boundary element method, the boundary of the domain is only meshed, rather than the entire volume.
- Speed - The boundary element method is, in general, much faster than the finite element method, for the following reasons:
 - a) Only the boundary must be defined, hence, less time is taken in the data preparation stage. It is suggested [13] that the time saving is of the order of 10:1.

- b) Changes in the mesh can be done with simplicity as compared to the finite element method, where changing the mesh is usually impractical, because of its complex nature.
 - c) Since the analysis stage of the 'boundary element method' only uses a small number of elements, compared to the 'finite element method', the actual time spent in the analysis stage is far smaller than that of the finite element method.
- **More Accurate** – The boundary element method spends less time doing numerical approximations than the finite element method and hence, in general should be more accurate.

The boundary element method, when applicable, is by far the most favourable method to use, but it is not applicable to every problem. For such problems, more traditional methods, such as 'finite element methods', should be favoured.

2.5 The Boundary Element Method

A number of papers and books have been published dealing with the 'boundary element method' [11] to [31], the main area of interest being the improvement and stability of the transient analysis. A number of different techniques are documented for solving transient heat conduction using the 'boundary element method' [17], [19], [20], [22], [38], [39]. The method adopted for the current analysis is the 'dual reciprocity method' [39]. The 'dual reciprocity method' has gained wide popularity because of its ability to analyse transient problems while maintaining the full advantages of the boundary element method as explained in chapter 4 of this thesis.

The boundary element method has proved to be the most powerful method of solving steady-state thermal problems. The method can be used to provide a boundary-only solution or to provide a solution at certain internal points as well as at the boundary. However, the interior solution near the boundary can result in large inaccuracies. This effect is due to the nature of the method, which relies on a fundamental solution given by equation 2.5.

$$\hat{u} = \frac{1}{2\pi} \ln\left(\frac{1}{r}\right) \quad (2.5)$$

Where r is the distance between the node, at which the solution is required, and the collocation point. The method uses a collocation procedure, whereby, the solution at a point is derived in terms of the solution at all other points or nodes. When the value of r is zero a singularity occurs and an analytical technique is used to evaluate the integral [37]. When the solution of an internal node is of interest, the integration is evaluated numerically. If, however, the internal node is very close to the boundary, the value of r will approach zero and errors will occur.

In 1989, Paulsen et al [30] presented work on the problem of interior node calculations. The author highlights the problem, as applied to electro-magnetics, by conducting an analysis of concentric cylinders, with constant amplitude applied. The results of a number of internal nodes were compared to an analytical solution, showing increasing error as the nodes got closer to the boundary. The problem was reduced when an increasing number of elements were used, since the numerical integration was carried out over a smaller element. This method, however, increases the overall size of the problem, and hence the time taken in its solution. Another technique employed was to increase the level of numerical integration at the element and hence increase the accuracy. This proved to be an extremely efficient method of solving the problem and did not increase the computational cost of the procedure.

The same problem was dealt with by J.M. Sisson [31], in 1990. In this publication, the author highlights the problem and its effect on elastostatic stress analysis. The author presents a number of applications and concludes with the same solution to the interior point problem as Paulsen [30].

The steady-state boundary element technique is stable and accurate and has little room for improvement. The transient technique, however, presents a serious problem. Once the defining differential equation, equation 2.6, is transformed into an integral equation, equation 2.7 results, [20].

$$\Delta^2 T = 1/\alpha \frac{\partial T}{\partial t} \quad (2.6)$$

$$cT = \int_{\Omega} T_o T_o^* d\Omega + \alpha \int_{\Gamma} \int_{t_o}^t T^* \frac{\partial T}{\partial n} ds d\Gamma - \alpha \int_{\Gamma} \int_{t_o}^t T \frac{\partial T^*}{\partial n} ds d\Gamma \quad (2.7)$$

Where t_o is the initial time and T_o is the initial temperature distribution. Where Ω represents the volume of the domain and Γ represents the boundary.

The problem with the solution of equation 2.7 is that it contains two volume integrals. This means the entire volume must be meshed. Hence, the method loses one of its main advantages over other numerical techniques. In order to regain this advantage, the volume integrals, present in equation 2.7 must be transformed into boundary integrals.

A number of methods have been employed to tackle this problem. The first was by Dargush ET. Al. [22], who solved equation 2.7 assuming a zero initial condition, hence, eliminating the volume integral. Once the solution was complete, the initial conditions were added to the solution. This method was accurate, but did not allow for problems with heat sources, meaning the method could not be universally used.

In 1989, Davey et al [21], presented a comparison between three methods of taking care of the volume integral, present in equation 2.7. The methods published were as follows.

- Domain meshing method.

This procedure consists of calculating the domain integral at each new time step, equation 2.8 shows the first two. This was done by meshing the entire volume and calculating the temperatures at the nodes produced. The temperature profile within the domain was then used to calculate the boundary temperature profile using equation 2.7. This method takes away from the main advantage of the boundary element method, that only the domain need be discretised, and hence is not practical.

$$\left. \begin{aligned} cT_1 &= \int_{\Omega} T_0 T_1^* d\Omega + \alpha \int_{\Gamma} \int_{t_0}^{t_1} T^* \frac{\partial T}{\partial n} ds d\Gamma - \alpha \int_{\Gamma} \int_{t_0}^{t_1} T \frac{\partial T^*}{\partial n} ds d\Gamma \\ cT_2 &= \int_{\Omega} T_1 T_1^* d\Omega + \alpha \int_{\Gamma} \int_{t_1}^{t_2} T^* \frac{\partial T}{\partial n} ds d\Gamma - \alpha \int_{\Gamma} \int_{t_1}^{t_2} T \frac{\partial T^*}{\partial n} ds d\Gamma \end{aligned} \right\} \quad (2.8)$$

▪ Wrobel's Method

This method was concerned with the transformation of the volume integrals by application of Green's second theorem. The method derived an equation for each time step, but presented the problem of requiring coefficient matrices at all previous time steps to solve for the current time step. This method was shown impractical for the following reasons.

1. Excessively large amounts of data storage are required.
2. The processor time required to solve the equations increases with time.
3. A new set of coefficient matrices must be calculated at each time step, increasing the processor time required even further.

The equations for the first two time steps are shown in equation 2.9.

$$\left. \begin{aligned} cT_1 &= \int_{\Omega} T_0 T_1^* d\Omega + \alpha \int_{\Gamma} \int_{t_0}^{t_1} T^* \frac{\partial T}{\partial n} ds d\Gamma - \alpha \int_{\Gamma} \int_{t_0}^{t_1} T \frac{\partial T^*}{\partial n} ds d\Gamma \\ cT_2 &= \int_{\Omega} T_0 T_2^* d\Omega + \alpha \int_{\Gamma} \int_{t_0}^{t_2} T^* \frac{\partial T}{\partial n} ds d\Gamma - \alpha \int_{\Gamma} \int_{t_0}^{t_2} T \frac{\partial T^*}{\partial n} ds d\Gamma \end{aligned} \right\} \quad (2.9)$$

The equation for time step two can be re-written as shown by equation 2.10.

$$\begin{aligned} cT_2 &= \int_{\Omega} T_0 T_2^* d\Omega + \alpha \int_{\Gamma} \int_{t_0}^{t_1} T^* \frac{\partial T}{\partial n} ds d\Gamma - \alpha \int_{\Gamma} \int_{t_0}^{t_1} T \frac{\partial T^*}{\partial n} ds d\Gamma \\ &\quad + \alpha \int_{\Gamma} \int_{t_1}^{t_2} T^* \frac{\partial T}{\partial n} ds d\Gamma - \alpha \int_{\Gamma} \int_{t_1}^{t_2} T \frac{\partial T^*}{\partial n} ds d\Gamma \end{aligned} \quad (2.10)$$

▪ Domain Approximation Method

This method uses Wrobel's method, discussed above, for the first three or four time steps. The next procedure is to compare the equations for the second time step in equations 2.10 and 2.8, giving equation 2.11.

$$\int_{\Omega} T_1 T_1^* = \int_{\Omega} T_0 T_2^* d\Omega + \alpha \int_{\Gamma} \int_{t_0}^{t_1} T^* \frac{\partial T}{\partial n} ds d\Gamma - \alpha \int_{\Gamma} \int_{t_0}^{t_1} T \frac{\partial T^*}{\partial n} ds d\Gamma \quad (2.11)$$

The procedure is then to estimate the volume integral for the current time step using the volume integral at the previous time step, which is known, and equation 2.11. This method eliminates the need for domain discretisation and doesn't require storage of all previous integrals. The method still, however, requires that Wrobel's method be used for the first number of time steps. The accuracy of the present method increases with increase in the number of time steps for which Wrobel's method is used.

Probably the most versatile and easy to implement methods of all is a method developed by Nardini and Brebbia [39] in 1982, called the 'Dual Reciprocity Boundary Element Method'. The method employs a fundamental solution, just like the steady-state method, and estimates all the other terms in the heat equation by a series expansion and global approximation functions. The full workings of the method can be found in [39].

The resulting matrix equation for transient heat conduction, without heat sources, is shown in equation 2.12.

$$\left(\frac{1}{\Delta t} C + \theta H \right) u^{m+1} - G q^{m+1} = \left[\frac{1}{\Delta t} C - (1 - \theta) H \right] u^m \quad (2.12)$$

It is important to note that the matrices H and G in equation 2.12 are the same coefficient matrices that are developed by the steady state method. This gives the dual reciprocity method the added advantage of supplying the steady state solution as well as the transient one. The transient solution is obtained by solving equation 2.13.

$$[H]\{u\} = [G]\{q\} \quad (2.13)$$

The value of θ in equation 2.12 is called the integration factor and is used to position the approximation of the current temperature between time steps, as shown by equation 2.14.

$$u = (1 - \theta)u^m + \theta u^{m+1} \quad (2.14)$$

As with any transient analysis the convergence of the solution will rely heavily on the time step employed. A number of authors have published work on the convergence of the method, [16], [17], [22], [23], [24], [25], [29], but probably the most reliable is the work done by Lahrman [17]. In this paper, the author developed an equation for relating the integration factor, θ , and the time step, equation 2.15.

$$\theta = \frac{Fo - 1 + e^{-Fo}}{Fo(1 - e^{-Fo})} \quad (2.15)$$

Where Fo represents the Fourier number and is given by equation 2.16.

$$Fo = \alpha \frac{\Delta t}{(\Delta x)^2} \quad (2.16)$$

CHAPTER 3

INJECTION MOULD COOLING SYSTEM DESIGN

3.1 Introduction

An injection moulding cycle consists of three distinct stages, injection, cooling and ejection. The cooling stage takes the greatest amount of time and is that part of the cycle where the plastic part is allowed to cool from its melt temperature to its ejection temperature. The time taken in the cooling stage has a large influence on the productivity and efficiency of a plastic manufacturing process, and for this reason, must be minimised. In order to do this a cooling system is employed throughout the mould core. A typical cooling system will consist of circular holes drilled at convenient places, within the mould, so that the coolant can extract the maximum possible amount of heat energy from the plastic. Figure 3.1 shows a simple, double cavity, system with sprue, runner and a four line cooling system.

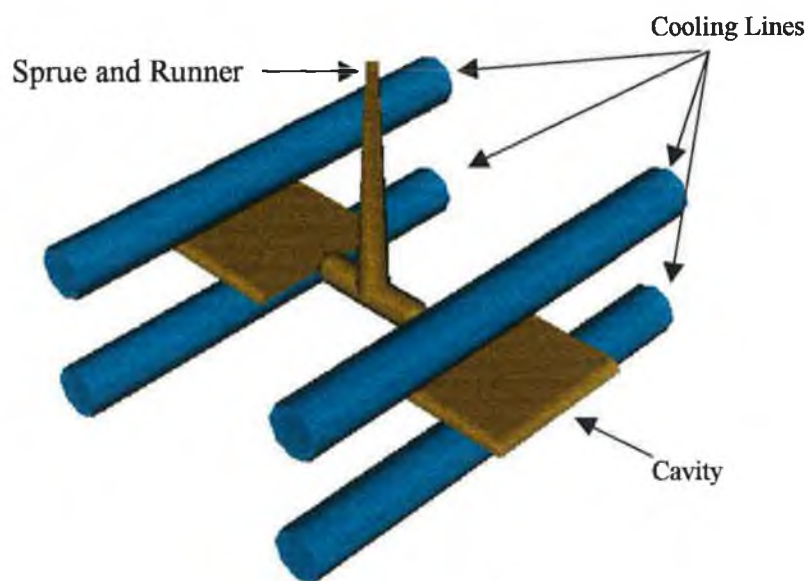


Figure 3.1 – Simple Cooling System

In some cases, however, it is necessary to make the cooling system more complicated, due to accessibility reasons, as shown in figure 3.2.

CHAPTER 3

INJECTION MOULD COOLING SYSTEM DESIGN

3.1 Introduction

An injection moulding cycle consists of three distinct stages, injection, cooling and ejection. The cooling stage takes the greatest amount of time and is that part of the cycle where the plastic part is allowed to cool from its melt temperature to its ejection temperature. The time taken in the cooling stage has a large influence on the productivity and efficiency of a plastic manufacturing process, and for this reason, must be minimised. In order to do this a cooling system is employed throughout the mould core. A typical cooling system will consist of circular holes drilled at convenient places, within the mould, so that the coolant can extract the maximum possible amount of heat energy from the plastic. Figure 3.1 shows a simple, double cavity, system with sprue, runner and a four line cooling system.

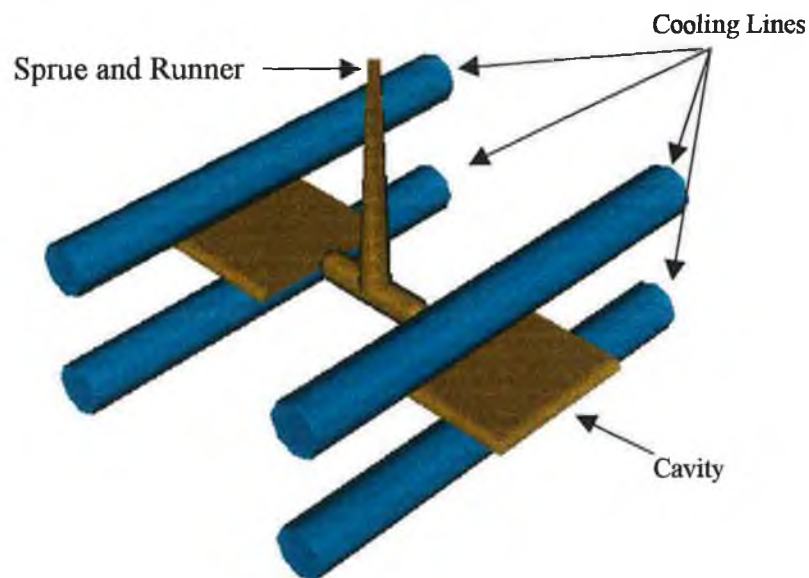


Figure 3.1 – Simple Cooling System

In some cases, however, it is necessary to make the cooling system more complicated, due to accessibility reasons, as shown in figure 3.2.

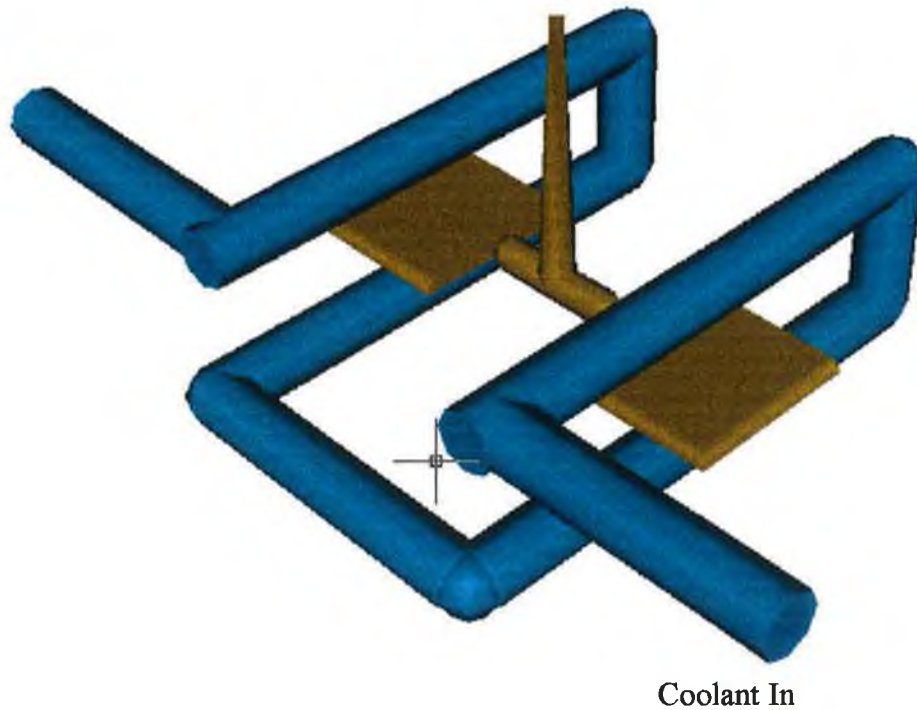


Figure 3.2 – Complicated Cooling System

In cases such as this, it is inevitable that parts of the cavity will be cooled at different rates than others. This can have the effect of leaving defects in the final product.

3.2 Defects in Plastic Parts

There are many types of defects that can occur in moulded thermoplastic parts. The causes of these defects can be due to the moulding machine, the injection mould, the material, or an inefficient cooling system. Some of the defects caused by in-efficient cooling are as follows:

3.2.1 Warpage

The most common defect found in plastic parts is *warpage*, which is primarily due to unbalanced cooling. Unbalanced cooling occurs when different sides of the cavity are cooled at different rates. This has the effect of inducing a bending moment on the part, figure 3.3. The hotter surface tends to shrink more once the part is ejected.

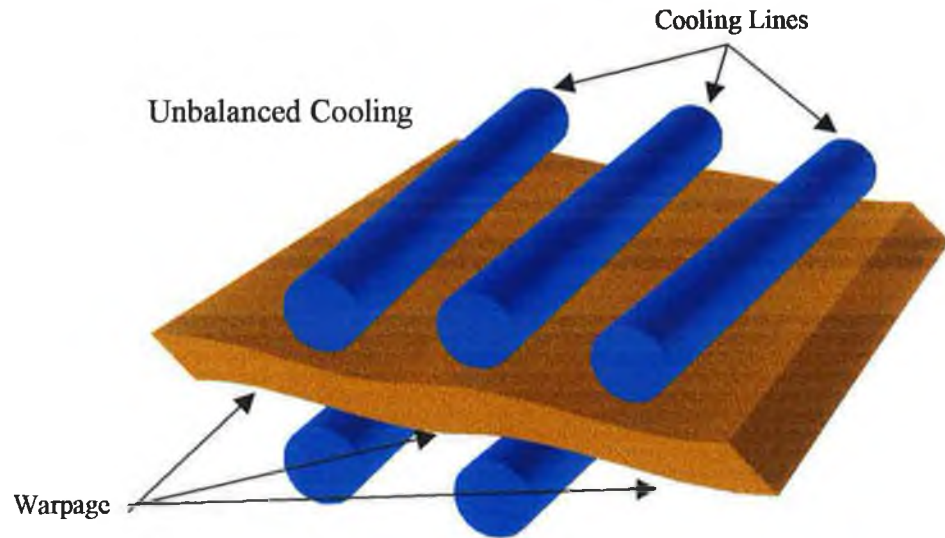


Figure 3.3 – Warpage on Plastic Part

This type of defect is due to inefficient cooling system design and can only be avoided by careful consideration of placement of cooling lines. In general, the number of cooling lines above the cavities should equal the number below.

3.2.2 Hot Spots and Sink Marks

In most complicated injection moulds, points within the plastic will be cooled at different rates to others. This usually happens in moulds with multiple or irregularly shaped cavities, resulting in parts of the cavity being inaccessible to the cooling lines. These *hot spots* will cause weaknesses in the product, resulting in *sink marks*, as shown in figure 3.4.

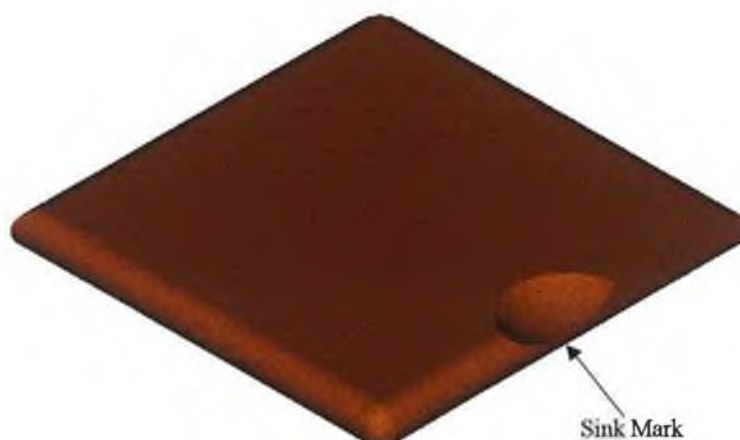


Figure 3.4 – Sink Mark

These defects can be avoided using heat pipes or bubblers, as shown in figures 3.5 and 3.6, respectively. A heat pipe is a tube that extends from the cooling line to the inaccessible area of the cavity. The tube has a wick that transports coolant in a liquid state, from the condenser end (cooling line) to the evaporator end (cavity). At this end the liquid evaporates, due to the heat from the plastic, and travels back to the condenser end to repeat the cycle and thus continuously transports heat from the cavity to the coolant. At the condenser end, the heat is transferred to the coolant flowing in the cooling lines.

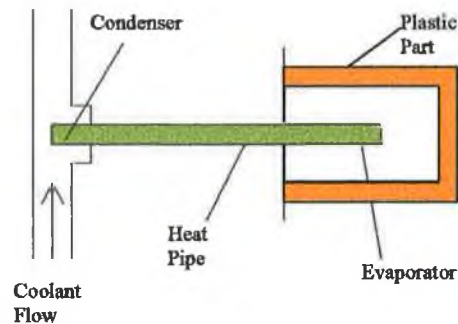


Figure 3.5 – Heat Pipe

A bubbler, also known as a fountain consists of concentric annuli. The coolant flows through the inner tube and returns through the annulus. For uniform flow of fluid, the internal diameter may be evaluated by [36], $D_1 = 0.70D_2 - t$.

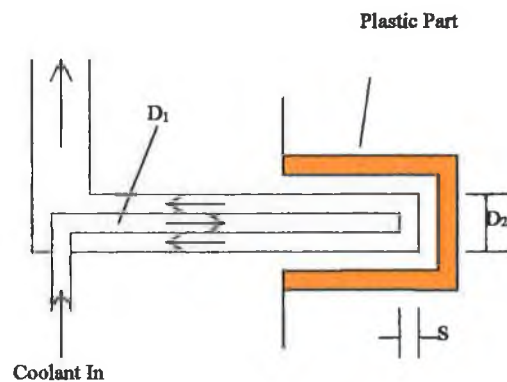


Figure 3.6 – Bubbler

Where t is the thickness of the inner pipe and D_2 is the diameter of the bubbler hole. For minimum pressure drop at the annular end, the distance S should be approximately $0.35D_2$ [36].

3.3 Cooling System Design.

The accurate design of injection mould cooling systems is required to achieve two major objectives:

- To obtain uniform cooling of the cavity.
- To extract the correct amount of heat in the minimum possible time.

In a typical injection mould, many variables can effect the efficiency of the cooling line mechanism within the mould core. These variables are as follows:

- Ambient air temperature.
- Mould surface area.
- Cooling line diameters.
- Coolant temperature.
- Coolant flow rate.
- Number of cooling lines.
- Cavity thickness.
- Polymer melt temperature.
- Polymer thermal properties.

The efficiency of an already implemented cooling system or one that has not yet been designed, may be altered by varying some, or all, of the above variables in order to minimise the heat lost to the environment. For a mould that has not yet been produced, this can be done by increasing the size and number of cooling lines. In practise, however, it is not possible to drill too many holes too close to each other because of mechanical failure. Hence, a number of rules must be taken into account when locating cooling lines [40].

- The cooling channels must not be machined too close to the cavity surface or thermal stresses could weaken the mould considerably. The cooling lines are kept 2-3 times the cooling line diameter away from the cavity.
- For the same reason, cooling lines must not be drilled too close to each other. Typically, the distance between cooling lines should be equal to their distance from the cavity.

A method used, mainly for moulds that are in operation, is to increase the coolant flow rate and/or to decrease the coolant temperature, using chillers. This will maximise the heat extracted from the cavity by the coolant and minimise the heat lost to the environment. It is important to note, however, that there is only a certain amount of heat that can be extracted by the cooling lines, since there will always be heat lost to the environment, and hence, if the flow of coolant is increased, past a critical amount, it will have little effect.

In order to develop a simulation methodology for the cooling system optimisation of an injection mould, the procedure shown in figure 3.7, is adopted.

When an injection mould is in use, the surfaces of the mould will undergo thermal loads. It is important to know the nature of these loads if a design methodology is to be developed. There are three types of loads.

- Constant temperature applied only when the temperature of a surface is known.
- Constant heat flux applied when the heat flux along a surface is known. This boundary condition is most commonly used when a surface is insulated or represents an axis of symmetry, in which case the flux is zero.
- Mixed, applied when the heat flux along a surface is dependent on the temperature of the surface. Examples of this are convection and radiation.

Once these boundary conditions are determined the temperature profile throughout the mould can be determined using a mathematical simulation technique.

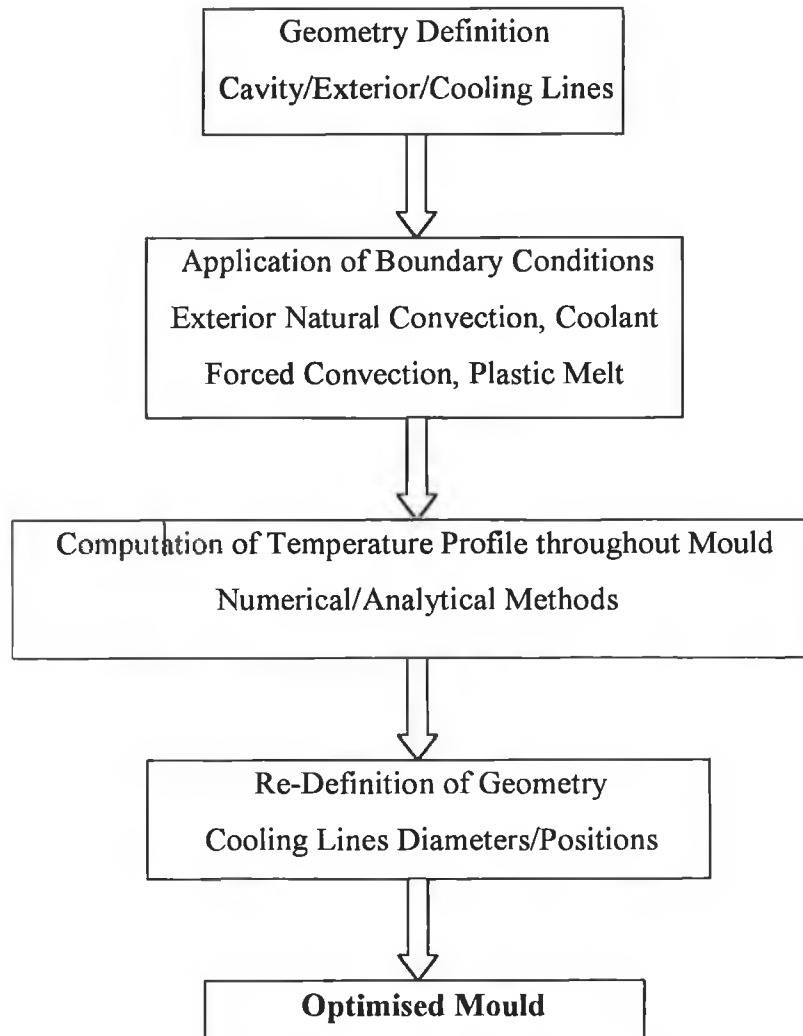


Figure 3.7 - Cooling System Optimisation Methodology

The schematic of a simple injection mould, as shown in figure 3.8, demonstrates the boundary conditions applied by the injection moulding cycle.

3.3.1 Cavity Surface

Hot polymer melt is injected into the cavity at high pressure, and is cooled down to its ejection temperature, by the cooling lines. If we consider a point on the cavity surface during this cycle, it is clear that the temperature at this point will be transient in behaviour, having a temporal profile like that shown in figure 3.9.

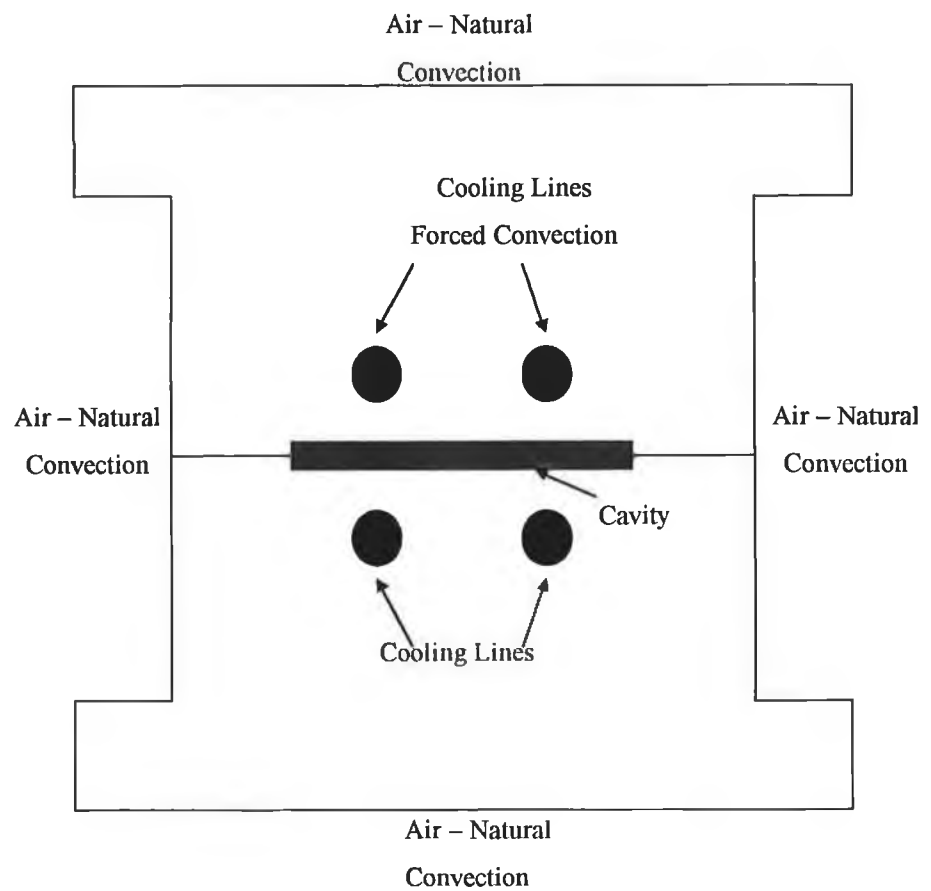


Figure 3.8 – Injection Mould Boundary Conditions

After the polymer-melt reaches its ejection temperature it is ejected and the cycle starts again. If this is repeated the temperature profile of the cavity surface will be transient cyclic in behaviour.

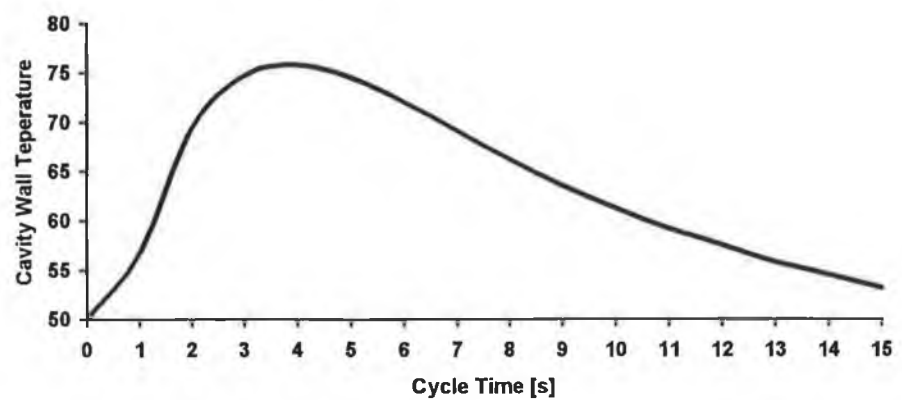


Figure 3.9 – Typical Cycle Temperature Profile

This means that the boundary condition at the cavity surface is continuously changing within the cycle. In order to determine the time-varying boundary condition, a one-dimensional analysis is carried out across the cavity wall. This analysis can be completed quite simply by making the following assumptions.

- The cavity is so thin that heat transfer occurs in the direction perpendicular to the cavity wall only.
- When the cavity is completely full, the heat transfer is assumed to be governed by conduction, and change of phase is neglected.

Figure 3.10 shows the thermal problem of one-dimensional conduction across a cavity of thickness, H , with a temperature of, T , at the boundaries. The temperature of the plastic is initially at T_0 .

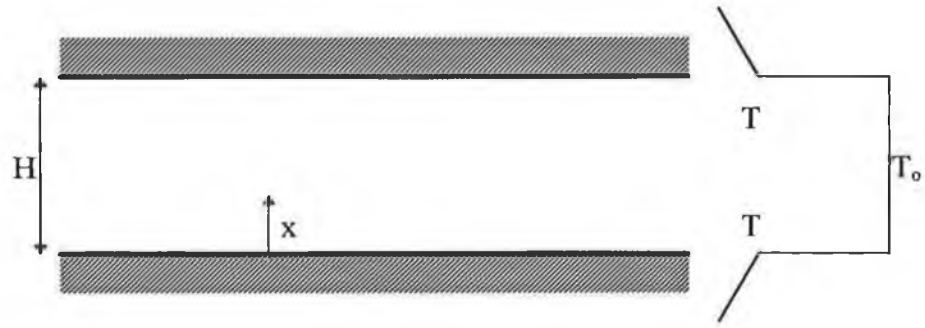


Figure 3.10 - Cavity Thermal Problem

To determine the transient temperature profile the equation for one-dimensional heat conduction, equation 3.1, must be solved.

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{1}{\alpha_p} \frac{\partial T(x,t)}{\partial t} \quad (3.1)$$

By applying the boundary conditions, shown in figure 3.10, and applying a Fourier technique [7], equation 3.2 can be derived for the temperature at any point, x , and time, t .

$$T(x,t) = T + \sum_{n=1,3,5,\dots}^{\infty} \frac{2}{n\pi} \left[(T_0 - T) - (-1)^n (T_0 - T) \right] \times \sin\left(\frac{n\pi x}{H}\right) e^{-\alpha_p n^2 \pi^2 t / H^2} \quad (3.2)$$

The average temperature of the polymer at any time, t , can be evaluated [7] by integrating the temperature profile, equation 3.2, over the entire thickness and dividing by the distance, H , resulting in equation 3.3.

$$T_{av}(t) = \sum_{n=0}^{\infty} \frac{8}{(2n+1)^2 \pi^2} (T_o - T) e^{\frac{-\alpha_p (2n+1)^2 \pi^2 t}{H^2}} \quad (3.3)$$

The heat flux can be derived, by making use of Fourier's equation, equation 3.4 [35], resulting in equation 3.5.

$$\frac{Q}{A} = \dot{Q} = k \frac{dT(x,t)}{dx} \quad (3.4)$$

$$Q(t) = K_p \sum_{n=1,3,5,\dots}^{\infty} \frac{2}{H} \left[(T_o - T) - (-1)^n (T_o - T) \right] \times e^{-\alpha_p n^2 \pi^2 t / H^2} \quad (3.5)$$

In order to apply a single boundary condition to the cavity, the heat flux is averaged over an entire cycle and applied in the form of a convection coefficient. The resulting coefficient is called the cycle-averaged convection coefficient and can be evaluated using equation 3.6.

$$h_{cv}(t_c) = \frac{\int_0^{t_c} Q(t) dt}{t_c (T_o - T)} \quad (3.6)$$

Combining equation 3.5 and 3.6 results in equation 3.7.

$$h_{cv} = \frac{2HK_p}{t_c (T_o - T) \alpha_p \pi^2} \times \sum_{n=1}^{\infty} \frac{1}{n^2} \left[(T_o - T) - (-1)^n (T_o - T) \right] \times \left(1 - e^{-\alpha_p n^2 \pi^2 t / H^2} \right) \quad (3.7)$$

It can be shown [7] that the terms with $n > 1$ are negligible compared to the first term. Hence, the equation can be reduced further to equation 3.8.

$$h_{cv} = \frac{4HK_p}{t_c \alpha_p \pi^2} \left(1 - e^{-\alpha_p \pi^2 t_c / H^2} \right) \quad (3.8)$$

The cooling time required to use equation 3.8 can be estimated, in terms of the plastic ejection temperature, T_e , using the following equation [7]:

$$t_c \approx \frac{4H}{\pi^2 \alpha_p} \ln \left[\frac{8}{\pi^2} \frac{T_m - T}{T_e - T} \right] \quad (3.9)$$

3.3.2 Cooling Lines

The most common type of cooling line is a simple circular hole drilled through the mould core so that the coolant can extract heat from the cavity. The flow through the cooling lines can be modelled using a forced convection correlation, and if turbulent flow occurs, the following correlation can be assumed [35].

$$\frac{hD}{k} = 0.023 \left(\frac{\rho v D}{\mu} \right)^{0.8} \left(\frac{\mu C_p}{k} \right)^{0.4} \quad (3.10)$$

In general, the flow of coolant should be kept turbulent. This will increase the value of $\left(\frac{\rho v D}{\mu} \right)$, or Reynolds's Number, in equation 3.1 and hence increase the heat transfer to the coolant.

Where,

ρ = Density of coolant [kg/m³]

v = Velocity of coolant, derived from mass flow rate, [m/s]

D = Diameter of cooling line [m]

μ = Coefficient of dynamic viscosity [kg/ms]

C_p = Specific heat capacity [J/kgK]

K = Thermal conductivity [W/mK]

h = Heat transfer coefficient [W/m²K]

3.3.3 Mould Exterior

The mould exterior is made up of two parts, those parts exposed to the atmosphere and those parts connected to the machine or platen. For the mould exterior the following correlations can be used for natural convection [35]:

Vertical side:

$$Nu = 0.677 Pr^{0.5} (0.952 + pr)^{-0.25} Gr^{0.25} \quad (3.11)$$

Horizontal Side:

$$Nu = C(Gr Pr)^m \quad (3.12)$$

Where C is 0.54, m is 0.25 for upper planes, C is 0.58, and m is 0.20 for lower horizontal planes.

Also,

$$Nu = \frac{hL}{k}, Gr = \frac{g\beta(T_w - T_a)L^3}{\nu}, Pr = \frac{\rho C_p}{k} \quad (3.13)$$

Where, L , represents the length of the vertical plane of the mould.

Once the boundary conditions have been determined, the next step is to determine the temperature and heat flux profile throughout the mould. Due to the irregular geometry of the mould, this cannot be done using analytical methods and a numerical method must be used. Three types will be considered:

- Finite Differences Method
- Finite Element Method
- Boundary Element Method

Whatever the analysis method used it must be noted that several of the boundary conditions, discussed previously, are dependent on the temperature of the mould. An example of this is the heat transfer coefficient for natural convection, which cannot be used to evaluate the temperature unless the temperature is known. This means an iterative approach is required to evaluate an accurate temperature profile. The steps in this approach are as follows.

- Initial estimate of global mould core temperature, may be taken from table 1.2.
- Calculation of cooling time, using equation 3.9.
- Calculation of cycle-averaged heat transfer coefficient, using equation 3.8.
- Calculation of natural and forced convection coefficients.
- Application of boundary conditions to numerical analysis.
- Re-iteration with new cavity and exterior temperature profiles.

Another problem that occurs with the analysis of the cavity is the fact that the thermal conductivity of the polymer is dependent on its temperature. This can be taken into account by assuming a linear variation, given by equation 3.14.

$$K = K_o(1 + \beta T) \quad (3.14)$$

In order to apply equation 3.14 the average polymer temperature must be evaluated using equation 3.3. It should be noted that in order to use equation 3.3 the thermal diffusivity, α_p , must be known, where $\alpha_p = K/\rho c$, and hence the thermal conductivity must be known, again presenting another iteration process. The overall analysis methodology is detailed in figure 3.11.

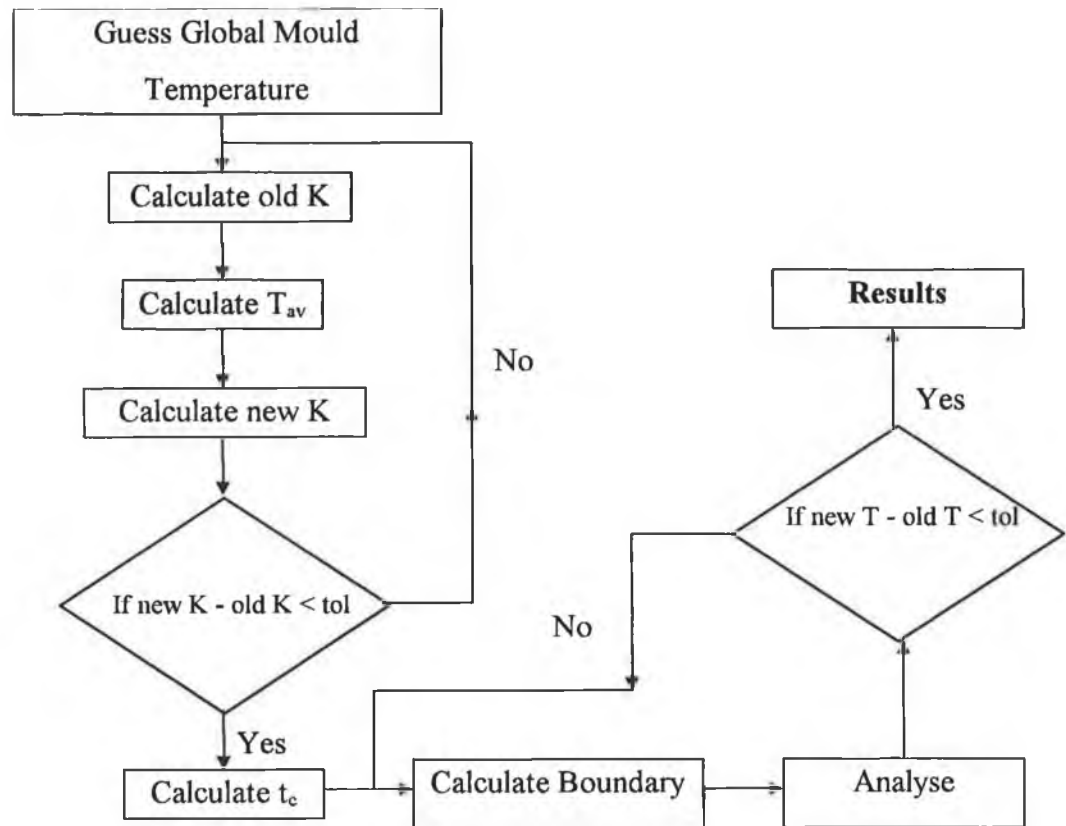


Figure 3.11 - Injection Mould Analysis Algorithm

Once the analysis is complete, the results must be interpreted, to determine the efficiency of the cooling system. A good indicator of this is the shape factor approach. Shape factors indicate the heat transfer due to a temperature difference and can be evaluated using equation (3.5).

As a way of indicating the overall performance of the cooling mechanism, two shape factors are defined,

$$S_1 = Q_m / K_m (T_m - T_c) \quad ; \quad S_2 = Q_c / K_m (T_m - T_c) \quad (3.15)$$

Where T_m and T_c represent the mould cavity and coolant temperatures, respectively.

S_1 and S_2 denote shape factors for the cavity surface and cooling surface, hence any difference in these will result in heat loss to the environment. Hence, the factor S_1/S_2 will give an indication of the cooling effect.

The cooling line system will be optimised to arrive at a value of $100 \times S_1/S_2$ closest to 100%.

When this analysis has been completed the boundary conditions are set up and a transient analysis can be performed, to see how the cycle-averaged temperature profile changes with time.

If the transient behaviour of the temperature profile within a steady cycle is required, the following procedure is adopted to give a cycle-transient solution.

- The cooling time is split up into a reasonable amount of time steps.
- At each time step, the heat transfer coefficient at the cavity wall is calculated using equation 3.5.
- A steady state analysis of the mould core will produce the temperature profile at each time step.

CHAPTER 4

COMPUTATIONAL HEAT TRANSFER

4.1 Introduction

Numerical methods are used to solve problems for which there are no exact mathematical solutions or for which the exact solutions are too complicated to derive. A classic example of this is the solution of partial differential equations. These equations can only be solved analytically for very simple physical problems with very simple boundary conditions. The problem with this is that most physical problems faced by engineers are governed by differential equations, for example, elastostatics, magnetics, fluid flow and heat transfer. These problems require a numerical method that can be used to approximate the solution to the governing differential equation.

The most common and most widely used methods for solving engineering problems are the finite difference and finite element method, the object of each is to reduce the given problem into a discrete mathematical model suitable for solution.

The finite difference method concentrates on replacing the differentials in the differential equation with differences, making it a very general and easy to implement. For this reason the method was adapted by engineers and widely used until the finite element method became popular in the 1950's. The finite element method offers many advantages over the finite difference method and is probably the most popular method used today.

Another method that has been around as long as the finite element method is called the boundary element method or the integral equation method. The method only became popular to engineers in the 1960's, when it emerged to be just as versatile and powerful a method as the finite element method.

It is not correct to say that any one method should be used universally over the others since each have their own particular advantages and disadvantages. It is therefore necessary to explain each method from a mathematical point of view to decide which method is more appropriate for the thermal analysis of injection moulds.

4.2 The Finite Difference Method.

The objective of the finite difference method is to represent the time-space continuum by a set of points. The variables required must then be derived for these points rather than the complete continuum. This is done by approximating the differential equation for each point or node and establishing a system of equations that are solved to find a solution at each node. The most effective way of doing this is to consider a node and its surrounding nodes. The nodes are related by a grid reference as shown in figure 4.1.

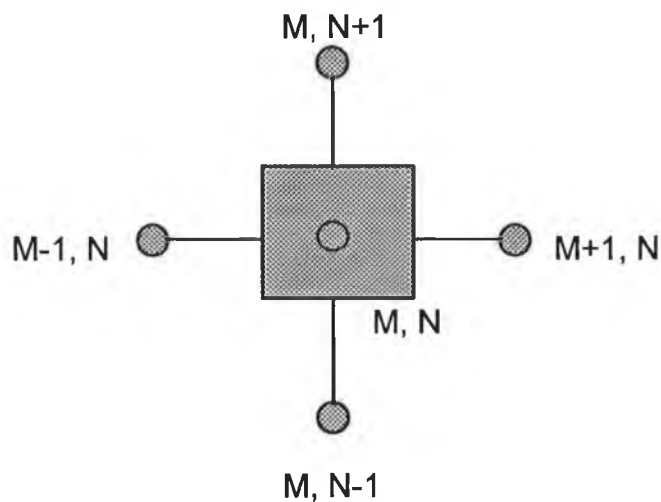


Figure 4.1 - Finite Difference Grid Reference

An energy balance on the node (M, N) will produce an equation for the temperature at that node in terms of the surrounding nodal temperatures. When an energy balance is completed for every node in the grid, a system of equations is produced which is solved to determine the temperature profile throughout the domain. The main disadvantage of the method is the need to derive equations for different systems, and hence the method is undesirable for computer programming of general-purpose codes. Another disadvantage is the difficulty involved when deriving finite difference models for irregularly shaped objects, since the accuracy of the method heavily relies on a rectangular grid.

This method is used when the problem in question is geometrically simple and the equations will not have to be re-derived, in the case of a change in boundary conditions or geometry.

In the case of injection mould analysis, the finite difference method is appropriate for the reasons stated above. The rest of this chapter concentrates on developing the finite element and boundary element model.

4.3 The Finite Element Method

The finite element method represents the geometry of the domain by a number of rectangular or triangular elements. Integral equations are derived using basic physical laws for each element and solved as a system for the entire domain. The method can represent any domain no matter how complicated, since it does not rely on rectangular grids, as did the finite difference method. Another advantage of the method over the finite difference method is that a simple equation is derived for the physical problem, for example heat transfer, and does not have to be re-derived for different domains.

The method was first put into practice by structural engineers in the analysis of complex structures but it was not long before it became available to field problems such as heat transfer.

Before describing the mathematics of the method, it is necessary to recall the differential and variational equations governing the conduction of heat through a solid with initial and boundary conditions.

For a three dimensional body, the heat fluxes in all three directions may be denoted by.

$$q_x = -k_x \frac{\partial u}{\partial x}; \quad q_y = -k_y \frac{\partial u}{\partial y}; \quad q_z = -k_z \frac{\partial u}{\partial z} \quad (4.1)$$

Where u denotes the temperature of the body at any point (x,y,z) and k denotes the thermal conductivity in a particular direction. Considering the heat flow equilibrium within the body, the following equation can be established for steady state heat transfer throughout the body [34].

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial u}{\partial z}\right) + \hat{q} = 0 \quad (4.2)$$

Where \hat{q} denotes heat generation per unit volume within the domain.

In any analysis, typical boundary conditions are applied to the mathematical equation. In general, there are three types of boundary condition.

- *Temperature Conditions:* A constant temperature may be applied to any surface element of the domain. In this case, the unknown for this element is heat flux.
- *Heat Flow Conditions:* A constant heat flow may be applied to any surface element of the domain. In this case, the unknown variable for this element is temperature.
- *Convection Boundary Conditions:* This boundary condition is of the mixed type, where neither the temperature nor the heat flow is known, but a relationship between them is.

The finite element method uses a *variational* approach, whereby the total potential Π is calculated and the stationarity of Π is invoked, that is, arbitrary variations in the state variables that satisfy the boundary conditions are negligible.

The functional governing heat conduction is given by equation 4.4 [32],

$$\Pi = \int_V \frac{1}{2} \left[k_x \left(\frac{\partial u}{\partial x} \right)^2 + k_y \left(\frac{\partial u}{\partial y} \right)^2 + k_z \left(\frac{\partial u}{\partial z} \right)^2 \right] dV - \int_V u \hat{q} dV - \int_S u q dS - \sum_i u_i Q_i \quad (4.4)$$

Letting this equal zero produces an integral equation that can be solved by writing the equation for each element and solving the system simultaneously.

The most important thing to note about equation (4.4) is that it contains volume integrals. This means that in order to establish the integrals involved, the complete volume of the domain must be discretised (broken down into elements).

4.4 The Boundary Element Method

In some engineering problems, the finite element method has proved inaccurate, inefficient, or too difficult to apply. It is for this reason that an alternative method was established. The method established uses Green's theorems to transform the volume integrals present in the finite element formulation into boundary ones. The advantage of this is that only the surface of the domain is discretised saving on data preparation time and creating a more efficient method. The main advantages of the method, over the finite element method, are as follows [38].

- Data preparation is minimised since no volume discretisation is required, creating a much faster method, over ten times faster than other methods.
- The method is more suitable to optimisation processes. For example if the position of the cooling line in an injection mould is changed then the elements concerned with the cooling line are simply moved and no other elements are disturbed. In the case of finite elements, any alteration to any part of the mesh requires a complete re-discretisation of the volume.
- The boundary element method only calculates the variables (temperature and heat flux) at the boundaries, by default, and not at points within the domain that may not be necessary, saving time. This makes the method more suitable to contact problems or problems where the variables are required at specific points only and not the entire domain. It is important to note, however, that the boundary element method is capable of calculating the variables at interior points within the domain, if required.
- Boundary element methods allow for elements that do not meet perfectly at corners, discontinuous elements, whereas finite elements do not, making mesh generation simpler again.
- In calculating the integrals involved in both the finite and boundary element methods a numerical integration scheme is considered. Since the finite element method uses more elements than the boundary element method the error introduced due to numerical integration should be greater, hence the boundary element method should be more accurate.

It is important to note that although boundary elements can prove more efficient for certain class of problems, it should not be seen as a method that completely replaces the finite element method. In the present case of analysing injection mould cooling systems,

the boundary element method would prove more suitable than other methods for the following reasons.

- The boundary element method can be used to solve both steady state and transient thermal problems in both two and three dimensions.
- The mould analysis is an optimisation process, which requires re-positioning of cooling lines, the boundary element method can do this without having to re-create the entire mesh.
- In the optimisation process, temperatures along the cavity surface are of interest only and temperatures at internal points are not required.

4.4.1 Steady State Thermal Boundary Elements

In order to predict the steady state temperature profile over a domain subject to the same boundary conditions as explained in section 4.3, a solution to the Laplace equation is sought [36].

$$\nabla^2 u = 0 \quad \text{in } \Omega \quad (4.5)$$

Where Ω represents the domain in question.

The basic procedure of the boundary element method is to transfer equation 4.5 into an integral equation and then to reduce all volume integrals to boundary integrals.

The first step in doing this is to approximate the function u within the domain. By doing this equation 4.5 will not be fully satisfied, instead a residual will be produced giving.

$$\nabla^2 u = R \neq 0 \quad \text{in } \Omega \quad (4.6)$$

Where R is the residual and u is an approximate solution.

The next step is to minimise the residual R by setting its weighted residual equal to zero, for various values of the weighting function, u^* .

$$\int_{\Omega} Ru^* d\Omega = \int_{\Omega} (\nabla^2 u) u^* d\Omega = 0 \quad (4.7)$$

Green's second identity is now applied, which states [36].

$$\int_V \phi \nabla^2 \chi dV = \int_S \left(\phi \frac{\partial \chi}{\partial n} - \chi \frac{\partial \phi}{\partial n} \right) dS \quad (4.8)$$

Where n denotes the normal to the surface S .

Applying equation 4.8 to 4.7, gives.

$$\int_{\Omega} u^* (\nabla^2 u) dV = \int_{\Gamma} \left(u^* \frac{\partial u}{\partial n} - u \frac{\partial u^*}{\partial n} \right) dS = 0 \quad (4.9)$$

The function u^* is known as the fundamental solution and satisfies Laplace's equation and represents the potential generated by a concentrated unit charge acting at a point ' i '. The value of u^* can be determined and is shown in reference [35] to be.

$$u^* = \begin{cases} \frac{1}{4\pi r}, & \text{for 3D} \\ \frac{1}{2\pi} \ln\left(\frac{1}{r}\right), & \text{for 2D} \end{cases} \quad (4.10)$$

Where r represents the distance between the point ' i ' and the point at which u is to be determined. A complete description of the mathematics is presented in reference [36] and results in the following 'boundary integral equation'.

$$c^i u^i + \int_{\Gamma} u q^* d\Gamma = \int_{\Gamma} q u^* d\Gamma \quad (4.11)$$

Where q and q^* represent $\frac{\partial u}{\partial n}$ and $\frac{\partial u^*}{\partial n}$, respectively.

The surface of the domain is discretised into N elements and equation 4.11 can then be written as follows.

$$c^i u^i + \sum_{j=1}^N \int_{\Gamma_j} u q^* d\Gamma = \sum_{j=1}^N \int_{\Gamma} q u^* d\Gamma \quad (4.12)$$

It is necessary now to consider how the temperature and flux $(\delta u / \delta n)$ vary over each element on the boundary, this is necessary in order to evaluate the integrals in equation 4.12. The variables can be assumed constant, linear, quadratic or of higher order. To maintain accuracy and simplicity linear elements are chosen. Reference [37] gives a good description of the different types of elements.

Consider two elements (two-dimensional) and their intersection as shown in figure 4.2.

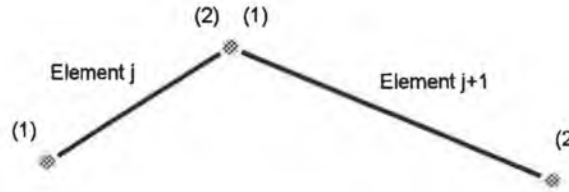


Figure 4.2. - Linear Element Intersection

In order to determine the integrals in equation 4.12 a local co-ordinate system is set up whereby $\eta = -1$ at point (1) of an element and $+1$ at point (2), that is $\eta = x/(l/2)$. Shape functions are defined such that.

$$\phi_1 = \frac{1}{2}(1 - \eta), \quad \phi_2 = \frac{1}{2}(1 + \eta) \quad (4.13)$$

The integrals in equation 4.12 become.

$$\int_{\Gamma_j} u q^* d\Gamma = \int_{\Gamma_j} [\phi_1 \phi_2] q^* d\Gamma \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = [h^1_{ij} h^2_{ij}] \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \quad (4.14)$$

Where,

$$h^1_{ij} = \int_{\Gamma_j} \phi_1 q^* d\Gamma, \quad h^2_{ij} = \int_{\Gamma_j} \phi_2 q^* d\Gamma \quad (4.15)$$

$$\int_{\Gamma_j} q u^* d\Gamma = \int_{\Gamma_j} [\phi_1 \phi_2] u^* d\Gamma \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} = [g^1_{ij} \ g^2_{ij}] \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix} \quad (4.16)$$

and,

$$g^1_{ij} = \int_{\Gamma_j} \phi_1 u^* d\Gamma, \quad g^2_{ij} = \int_{\Gamma_j} \phi_2 u^* d\Gamma.$$

The resulting equation can then be written for node 'i'.

$$c_i u_i + \begin{bmatrix} \hat{H}_{i1} & \hat{H}_{i2} & \dots & \hat{H}_{iN} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{Bmatrix} = \begin{bmatrix} G_{i1} & G_{i2} & \dots & G_{iN} \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{Bmatrix} \quad (4.17)$$

Where each \hat{H}_{ij} is equal to the h_2 term of element (j-1) plus the h_1 term of element (j), the same applying for G_{ij} . Equation 4.17 can be written for each node in the system resulting in a matrix equation as follows,

$$HU = GQ \quad (4.18)$$

Where,

$$H_{ij} = \begin{cases} \hat{H}_{ij} & \text{for } i \neq j \\ \hat{H}_{ij} + c_i & \text{for } i = j \end{cases}$$

Applying the boundary conditions to equation 4.18 can then solve the system. Once the variables at the boundary are known they can be evaluated at any internal point using equation 4.17.

The value of c can be evaluated by virtue of the fact that when a uniform potential is applied over the surface then the heat flux must be zero, producing, $HI = 0$, hence,

$$H_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^N H_{ij}, \quad i = 1, 2, \dots, N \quad (4.19)$$

4.4.2 Transient Thermal Boundary Elements

In order to predict the transient temperature profile over a domain subject to the same boundary conditions as explained in section 4.3, a solution to the diffusion equation is sought.

$$\nabla^2 u = \frac{1}{k} \frac{\partial u}{\partial t} \quad \text{in } \Omega \quad (4.20)$$

Where Ω represents the domain in question, k the thermal diffusivity of the material and t the temporal dimension.

A ‘time dependent fundamental solution’ can be established and an analysis completed as in section 4.4.1 to establish an integral equation. This, however, produces a volume integral equation, which reduces the main advantage of the boundary element method. A number of methods have been produced to transform the volume integrals into surface ones, and these are well documented in references [18], [20], [21], [23] and [25].

One such method, by Brebbia et. al. [23], can be used quite simply since it uses the same matrices established in the steady state analysis. This method is known as the dual reciprocity method.

The method approximates the right hand side of equation 4.20, using approximating functions f_{ij} and coefficients α_j .

$$\frac{\partial u}{\partial t} = \dot{u} \approx \sum_{j=1}^{N+L} f_{ij} \alpha_j \quad (4.21)$$

Where ' L ' represents the number of internal nodes, or poles, and N the number of boundary elements. It has been shown [38] that for diffusion problems a relatively simple function f_{ij} should be used, such as, $f_{ij} = 1 + r_{ij}$. Where r_{ij} denotes the distance between the source point i and the field point j .

This is written in matrix form as follows,

$$\{\dot{u}\} = [F]\{\alpha\}, \quad \{\alpha\} = [F^{-1}]\{\dot{u}\} \quad (4.22)$$

A particular solution \hat{u}_j is established and is related to f_j as follows.

$$\nabla^2 \hat{u}_j = f_j \quad (4.23)$$

Substituting equation 4.23 into equation 4.21 and using equation 4.20 gives.

$$\begin{aligned} \nabla^2 u &= \frac{1}{k} \sum_{j=1}^{N+L} \alpha_j (\nabla^2 \hat{u}_j) \\ \hat{u} &= r^2/4 + r^3/9 \end{aligned} \quad (4.24)$$

Multiplying by the fundamental solution, integrating over the domain, and integrating by parts as is done with the steady state problems, produces the equation.

$$c^i u^i + \int_{\Gamma} u q^* d\Gamma - \int_{\Gamma} q u^* d\Gamma = \frac{1}{\alpha} \sum_{j=1}^{N+L} \alpha_j \left(c^i \hat{u}^i + \int_{\Gamma} \hat{u} q^* d\Gamma - \int_{\Gamma} \hat{q} u^* d\Gamma \right) \quad (4.25)$$

After discretisation, equation 4.25 can be written in matrix form as.

$$Hu - Gq = \frac{1}{k}(H\hat{u} - G\hat{q})\alpha \quad (4.26)$$

Using equation 4.22, equation 4.27 is derived.

$$Hu - Gq = \frac{1}{k}(H\hat{u} - G\hat{q})F^{-1}\dot{u} \quad (4.27)$$

Resulting in,

$$C\dot{u} + Hu = Gq \quad (4.28)$$

where,

$$C = -\frac{1}{k}S = -\frac{1}{k}(H\hat{u} - G\hat{q})F^{-1}.$$

A two-level time integration scheme is applied and u and q are approximated as follows.

$$\begin{aligned} u &= (1 - \theta)u^m + \theta u^{m+1}, \quad q = q^{m+1} \\ \dot{u} &= \frac{1}{\Delta t}(u^{m+1} - u^m) \end{aligned} \quad (4.29)$$

Applying 4.29 to equation 4.25 gives.

$$\left(\frac{1}{\Delta t}C + \theta H\right)u^{m+1} - Gq^{m+1} = \left[\frac{1}{\Delta t}C - (1 - \theta)H\right]u^m \quad (4.30)$$

Once the matrices have been established and boundary conditions applied the temperature and flux profiles throughout the domain at any time step can be evaluated in terms of the profiles at the previous time.

4.5 Computer Implementation of Boundary Element Method.

It is necessary to note that the solution of the steady state or transient problem requires the manipulation of matrices, governed by equations 4.18 and 4.30. In order to evaluate the G and H matrices the integrals in equation 4.16 must be evaluated. Combining equations 4.13 and 4.16 the following integrals result.

$$\begin{aligned} g^1_{ij} &= \frac{1}{2} \int_{\Gamma_j} (1-\eta) u^*, & g^2_{ij} &= \frac{1}{2} \int_{\Gamma_j} (1+\eta) u^* \\ h^1_{ij} &= \frac{1}{2} \int_{\Gamma_j} (1-\eta) q^*, & h^2_{ij} &= \frac{1}{2} \int_{\Gamma_j} (1+\eta) q^* \end{aligned} \quad (4.31)$$

The integrals are evaluated using a Gauss Quadrature technique, keeping in mind the fundamental solution given by equation 4.10. Hence, the integrals are evaluated as follows.

$$\begin{aligned} g^1_{ij} &= \frac{1}{2} \cdot \frac{1}{2\pi} \cdot \frac{L}{2} \cdot \sum_{k=1}^K w (1-\eta) \ln\left(\frac{1}{R}\right), & g^2_{ij} &= \frac{1}{2} \cdot \frac{1}{2\pi} \cdot \frac{L}{2} \cdot \sum_{k=1}^K w (1+\eta) \ln\left(\frac{1}{R}\right) \\ h^1_{ij} &= \frac{1}{2} \cdot \frac{1}{2\pi} \cdot \frac{L}{2} \cdot \sum_{k=1}^K \frac{\partial R}{\partial n} (1-\eta) \left(\frac{1}{R}\right), & h^2_{ij} &= \frac{1}{2} \cdot \frac{1}{2\pi} \cdot \frac{L}{2} \cdot \sum_{k=1}^K \frac{\partial R}{\partial n} (1+\eta) \left(\frac{1}{R}\right) \end{aligned} \quad (4.32)$$

Where L denotes the length of the element ' j ', η represents a point along the element, w represents the weighting at that point and R represents the distance from source point ' i ' to point η . Values for η and w can be found in appendix 1, $\frac{\partial R}{\partial n}$ represents the slope of the distance R with respect to the element normal and can be evaluated considering the geometry of the element. It is obvious from equations 4.32 that when R is zero a singularity occurs making the numerical integration impossible. In this case, the integration can be carried out analytically, resulting in equation 4.33.

$$\begin{aligned} g^1_{ii} &= \frac{L}{4\pi} \left[\frac{3}{2} - \ln(L) \right], & g^2_{ii} &= \frac{L}{4\pi} \left[\frac{1}{2} - \ln(L) \right] \\ h^1_{ii} &= 0, & h^2_{ii} &= 0 \end{aligned} \quad (4.33)$$

Once the matrices G and H are evaluated, they can be re-arranged in accordance with boundary conditions resulting in a system of linear simultaneous equations. To apply the 'transient boundary element method' described above the same G and H matrices are used. The remaining matrices are calculated using a ' f ' function, given by $f = 1 + r$, where r is the distance between the source point and the 'dual reciprocity' collocation point. It is important to note that the 'dual reciprocity collocation points' can be any points on the boundary or internal to the domain, used to represent the domain, hence the boundary nodes should be used. This would also suggest that internal points should be used. This is the case, although in problems where there are sufficient numbers of degrees of freedom on the boundary the number of internal nodes (poles) can be small. The number of poles will effect the accuracy but will in no way jeopardise the existence of a solution.

In order to show the usefulness of the boundary element method the transient analysis of a benchmark problem was conducted. The program was written in Visual Basic to utilise the full power of Windows for input and output operations. The analysis subroutines were written in FORTRAN 90 and compiled into a DLL for use within the Visual Basic Program. The program listings can be seen in appendix 2.

The problem consists of the transient analysis of a one-dimensional slab with a temperature difference of 300°C applied across its width. The exact solution can be found using 'Fourier series analysis' of the boundary conditions, and an equation for the temperature at a distance x and at time t can be established, equation 4.34 [7].



Figure 4.3 - One Dimensional Thermal Problem

$$U(x, t) = 5x + \sum \frac{2}{n\pi} (U_0 - (-1)^n (U_0 - 300)) \sin\left(\frac{n\pi x}{6}\right) e^{-\frac{\alpha n^2 \pi^2 t}{36}} \quad (4.34)$$

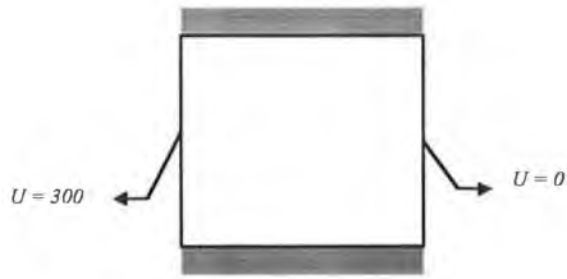


Figure 4.4 - Benchmark Problem

To model the one-dimensional problem a two-dimensional square is used. The square was insulated on two opposite sides providing one dimensional heat transfer between the other two sides. A temperature difference of 300°C is applied, as shown in figure 3.4. The square is of size 6 with a thermal diffusivity of 1.25. The initial global temperature was set as zero.

Table 4.1 shows how the boundary element estimation can be improved by increasing the number of elements used to estimate the boundary. Equation 4.30 was used with a Δt of 0.5 seconds, a θ of 0.5 and 5 internal nodes. The results show that the boundary element estimation converges on values slightly higher than the exact results.

TIME	EXACT	4 ELEMENTS	12 ELEMENTS	40 ELEMENTS
0.0	0.000	0.000	0.000	0.000
0.5	2.187	-1.462	-0.472	6.943
1.0	17.334	-16.477	-9.987	34.130
2.0	53.897	16.728	38.346	67.405
3.0	81.692	38.013	70.174	91.670
4.0	101.507	54.877	92.577	108.931
6.0	125.565	80.790	120.201	129.709
8.0	137.687	99.517	134.530	139.990
12.0	146.874	123.120	145.831	147.566
15.0	148.882	133.244	148.440	149.158
20.0	149.798	142.378	149.697	149.856
STEADY	150.000	150.000	150.000	150.000

Table 4.1 - Convergence with Refining Discretisations

Since the time differential is estimated using a finite difference method, as shown in equation 4.29, the approximation should be more accurate with decreasing time steps. This is shown in table 4.2 where an analysis was completed with 12 boundary elements, 5 internal nodes and a θ of 0.5.

TIME	EXACT	$\Delta T = 1$	$\Delta T = 0.5$	$\Delta T = 0.1$
0	0.000	0.000	0.000	0.000
1	17.334	0.950	-9.987	8.129
2	53.897	20.360	38.346	46.796
3	81.692	68.582	70.174	75.492
4	101.507	84.081	92.577	96.268
6	125.565	116.329	120.201	122.065
8	137.687	132.764	134.530	135.477
12	146.874	145.457	145.831	146.075
15	148.882	148.366	148.440	148.529
20	149.798	149.680	149.697	149.713

Table 4.2 - Convergence with decreasing time step

The accuracy of the dual reciprocity method relies heavily on the existence of internal nodes. Table 4.3 shows the convergence of the method with increasing numbers of internal nodes. This analysis was completed using 40 boundary elements, $\Delta t = 0.5$ and $\theta = 0.5$.

In order to estimate the temperature within two time steps an integration factor, θ , is used, as shown by equation 4.29. The final analysis was one of convergence and a suitable value of the time integration factor, θ to use. Table 4.4 shows an analysis completed with 40 boundary elements and 25 internal nodes with a time increment of 0.5 seconds. The 'weighted' value of θ is derived from a method mainly associated with finite elements. The weighted θ can be evaluated, for each element, using the following equation [16].

$$\theta_i = \frac{Fo_i - 1 + e^{-Fo_i}}{Fo_i(1 - e^{-Fo_i})} \quad (4.35)$$

Where, Fo_i is the Fourier number at the element 'i' and is equal to, $Fo_i = \alpha \Delta t / \Delta x_i^2$, where Δx is the element length.

TIME	EXACT	L = 5	L = 13	L = 25
0	0	0	0	0
1	17.33	34.13	25.26	14.60
2	53.90	67.41	60.65	55.23
3	81.69	91.67	86.63	83.29
4	101.51	108.93	105.15	102.86
6	125.57	129.71	127.56	126.34
8	137.69	139.99	138.78	138.10
12	146.87	147.57	147.19	146.98
15	148.88	149.16	149.01	148.92
20	149.80	149.86	149.83	149.81

Table 4.3 - Convergence with Increasing Number of Internal Nodes

TIME	EXACT	THETA=0.5	THETA=2/3	WEIGHTED
0	0.00	0.00	0.00	0.00
1	17.33	14.60	14.70	14.68
2	53.90	55.23	55.64	55.57
3	81.69	83.29	83.54	83.50
4	101.51	102.86	102.93	102.92
6	125.57	126.34	126.27	126.28
8	137.69	138.10	138.02	138.03
12	146.87	146.98	146.94	146.95
15	148.88	148.92	148.90	148.91
20	149.80	149.81	149.80	149.80

Table 4.4 Results for Different Integration Factors, θ .

It has been shown that the transient boundary element method produces quite accurate results when a substantial discretisation is used along with weighted time integration for convergence. In order to show the superiority of the method over the finite element method a comparison was made for the same discretisation of 40 boundary elements, the discretisations used are shown in figure 4.4. The finite element method used 40 domain elements. The boundary element analysis consisted of 40 elements and 25 nodes as before with a weighted time step.

The results show that for the present thermal problems the boundary element method can produce results that are far more accurate than those of the finite element method. It is important to note that the large initial errors are due to the fact that the temperature is assumed linear over the first time step rather than stepped as in the analytical solution.

TIME	EXACT	BEM	BEM Error (%)	FEM	FEM Error (%)
1	17.33	14.68	15	10.09	42
2	53.90	55.57	3	45.41	16
3	81.69	83.50	2	75.49	8
4	101.51	102.92	1	97.02	4
6	125.57	126.28	1	123.20	2
8	137.69	138.03	0	136.44	1
12	146.87	146.95	0	146.53	0
15	148.88	148.91	0	148.75	0
20	149.80	149.80	0	149.77	0

Table 4.5 - Boundary Elements versus Finite Elements

It is important to note the times taken for each analysis, 110 seconds for the finite element method and 5 seconds for the boundary element. In general, depending on the problem, the boundary element method can be 10 to 30 times faster than the finite element method.

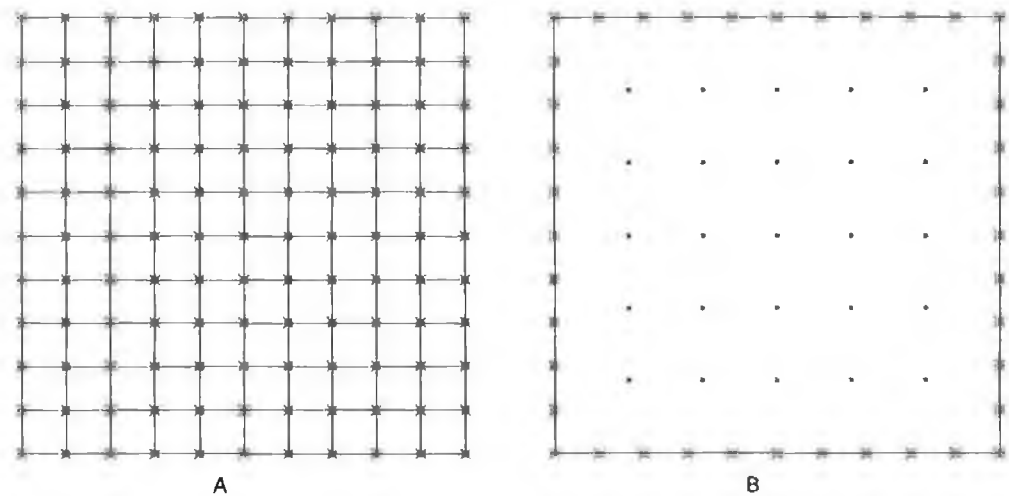


Figure 4.5 - (A) Finite Element Discretisation, (B) Boundary Element

CHAPTER 5

MOULD COOL: INJECTION MOULD COOLING SYSTEM ANALYSIS SOFTWARE

5.1 Introduction

In order to implement the methods and analysis procedures, set out in chapters 2 and 3, a two-dimensional, thermal analysis program was developed, called MouldCOOL. The software was developed to run on fully 32-bit operating systems, such as Windows 95, and uses the boundary element method for the two-dimensional steady state and transient analysis of injection mould cooling systems.

The software was developed using Visual Basic because of its excellent user friendly interface and collection of procedures and methods for input and output operations. Some of these features are listed below.

- Data File input and output.
- Input dialog boxes.
- Data grids for input and output.
- List boxes and combo boxes for output.
- Graphical picture boxes for output of geometry and contour plots.
- Graphs for plotting data.

Visual Basic, however, provides no support for complex mathematical methods, such as boundary element techniques. For this reason, the analysis subroutines, Appendix 2, were written in 'Fortran 90' using 'Microsoft FORTRAN PowerStation'. The subroutines were compiled and built as a 'Dynamic Link Library' (DLL) so that they could be made available for use by any other programming environment, such as Visual Basic.

MouldCOOL was developed to provide an analysis technique for the cooling system of injection moulds and to supply results in a graphical, user-friendly way. The resulting software is best illustrated by a flow diagram, as shown in figure 5.1.

MouldCOOL Algorithm

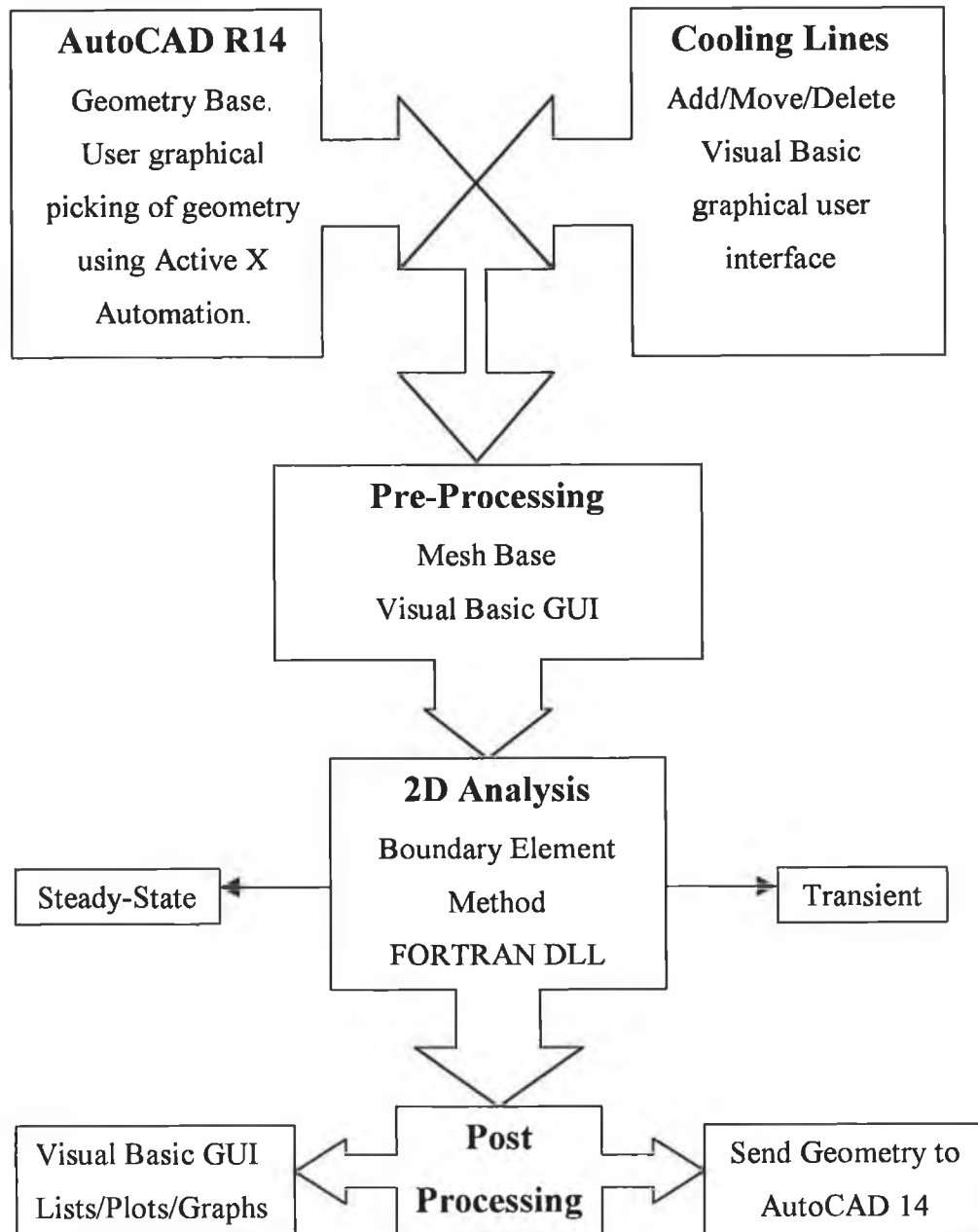


Figure 5.1 – MouldCOOL Program Structure

In order to complete a thermal analysis of an injection mould, a two-dimensional section of its geometry must be specified. Once the geometry is supplied, it is broken down into elements or discretised so that the boundary element method can be applied. The geometry is supplied to MouldCOOL via 'AutoCAD Release 14', which is used because of its widespread use and flexibility. This is done by accessing the AutoCAD database via object orientated programming, from within visual basic, and Active X automation. Active X automation is a procedure whereby an application lets its objects, procedures and methods become available for use by any other application. The main functions of the program can be defined within the following categories.

- AutoCAD Interface
- Pre - Processing - mesh generation.
- Analysis - application of boundary element method.
- Post-Processing - displaying of results for interpretation by the user.

A complete listing of the visual basic programs and the Fortran programs can be found in appendix 2. A snap shot of the MouldCOOL interface is shown in figure 5.2.

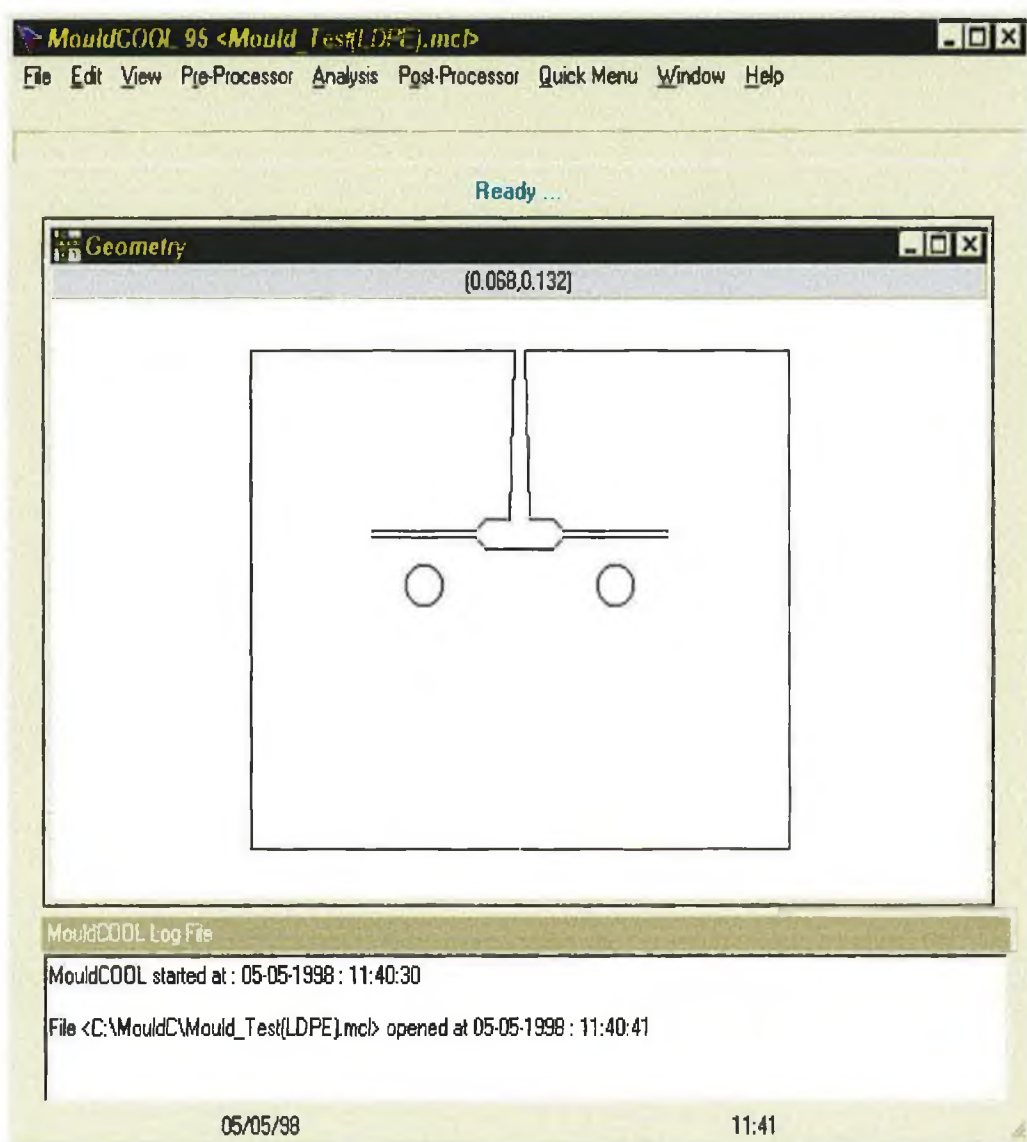


Figure 5.2 – MouldCOOL Interface

5.2 Geometry Base (AutoCAD Interface)

In order to supply the geometry for the injection mould analysis, it is useful to use a well-established and tried graphics or 'Computer Aided Design' (CAD) package. In this way, users can conduct an analysis using their current drawings, eliminating the need for developing an understanding of graphics systems that may be shipped with regular analysis programs. One of the most popular CAD packages available is AutoCAD Release 14, which contains a number of capabilities for programming, such as.

- AutoLisp – programming language used within AutoCAD for accessing the geometry database. All programs written in AutoLisp are run via the command line from within AutoCAD.
- ADS – 'AutoCAD Development System', capabilities to run compiled 'C' programs, that have full access to the AutoCAD database, from within the AutoCAD environment.
- Active X Automation – object oriented method for supplying other 32-bit Windows applications with information about its database.

ActiveX Automation is a new programming interface for AutoCAD, providing a means for developing scripts, macros and third-party applications using programming environments, such as Visual Basic 5.0. Through Active X Automation, AutoCAD exposes programmable objects, which can be manipulated by Visual Basic. Thus, 'Active X Automation' enables cross-application programming, a capability that does not exist in AutoLisp. The exposed objects are called Automation objects, which have and expose methods and properties. Methods are functions that perform an action on an object. Properties are functions that set or return information about the state of an object.

This means that applications, requiring geometry as input, can be developed to use AutoCAD 14 for the contents of its database. The current software, MouldCOOL, uses AutoCAD to supply a two-dimensional section of an injection mould. In order to do this, within the Visual Basic environment, AutoCAD must be initialised so that its objects can be accessed. This can be done using the following code.

```
Set AcadAPP = GetObject (, "AutoCAD.Application")
```

This code reserves a place within the Visual Basic application for information on AutoCAD, for example, the path of its executable program. If AutoCAD is not running, this code returns an error, in which case AutoCAD can be started and initialised using the following code.

```
Set AcadAPP = SetObject (, "AutoCAD.Application")
```

Now AutoCAD's database can be accessed by the application by referring to the properties of 'AcadAPP'. For example a line can be drawn, between two points, pt1 and pt2, on the current AutoCAD document screen using the following code.

```
AcadAPP.ActiveDocument.ModelSpace.AddLine (pt1, pt2)
```

The same procedure can be used to represent any AutoCAD entity, in AutoCAD, using Visual Basic. The interface can also be used as an input device; for example, the following code shows how the application jumps to AutoCAD so that the user can select a number of objects.

```
Set MouldSet = AcadApp.ActiveDocument.SelectionSets.Add("MouldSet")
```

This line defines a variable, MouldSet, so that the objects selected by the user can be accessed from within the application.

```
Call MouldSet.SelectOnScreen
```

This line sets AutoCAD to expect the user to select a number of objects. Once the objects are selected, a selection set is created which contains the properties of all entities selected by the user.

Once the geometry has been selected from AutoCAD, MouldCOOL sets up variables so that the geometry can be stored and displayed when required. The variables are defined by declaring four new object types within Visual Basic, a 'Point' object, a 'Line' object, an 'Arc' object and a 'CoolingLine' object. The code for these declarations is as follows.


```
Type point
X As Variant
Y As Variant
End Type
```

```
Type line
pt1 As point
pt2 As point
End Type
```

```
Type arc
pt1 As point
pt2 As point
Centre As point
Radius As Variant
angle1 As Variant
angle2 As Variant
End Type
```

```
Type CoolingLine
Centre As point
Radius As Variant
NElements As Integer
End Type
```

The first line of each declaration defines the new object type. The lines of code, between the first line and the 'End Type' statement, define separate properties of the object. An example of this is a line which is defined by two points, whilst a point is defined by an 'X co-ordinate' and a 'Y – co-ordinate'.

Once the geometry is defined new variables are set up and put into memory for further use. MouldCOOL can then display the geometry in its own graphics window whenever required by the user, as shown in figure 5.3.

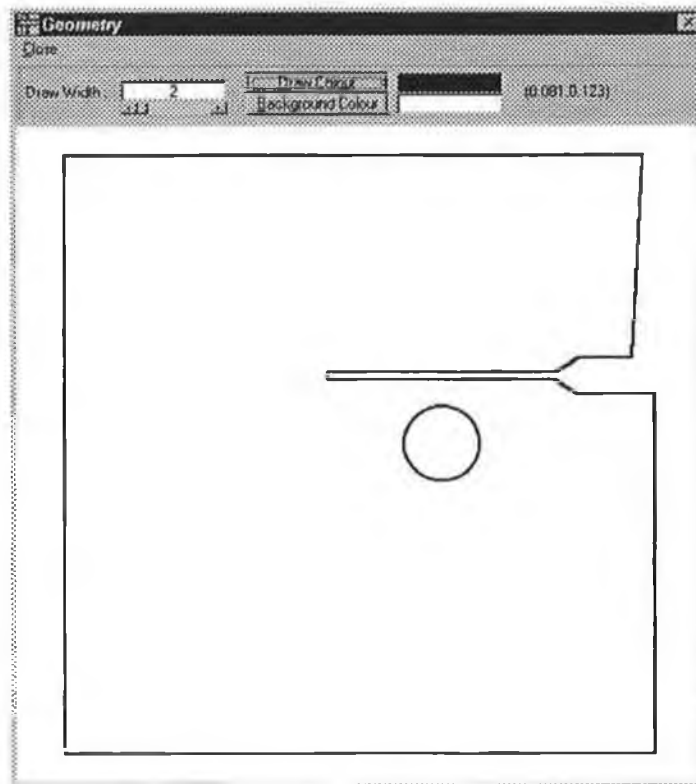


Figure 5.3 – MouldCOOL Geometry View

The geometry base also allows for deleting moving or adding cooling lines. This is especially important since the cooling system design process is an optimisation one, where geometry may be changed many times before deciding on a sufficient arrangement.

This is done by adding or deleting, to or from, the cooling line variable already set up. Moving a cooling line simply changes an existing cooling line's parameters. The cooling line dialog box within MouldCOOL is shown in figure 5.4.

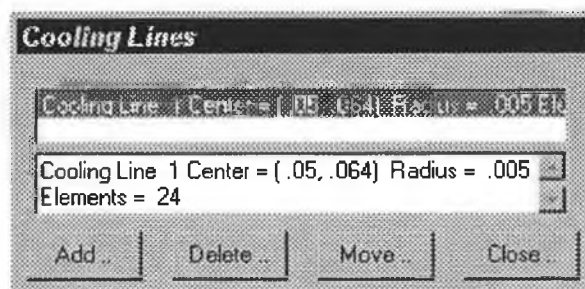


Figure 5.4 – MouldCOOL Cooling Line Editor

5.3 Mesh Base

Before the analysis of the mould can be completed, the geometry must be subdivided into a number of elements. In the case of two-dimensional boundary element techniques, the elements required are one-dimensional lines. This means that lines and arcs/circles are simply divided up into a number of lines, the fineness of the mesh being specified by the user. The mesh base is that part of memory, which holds the details of the geometry discretisation or mesh created. MouldCOOL provides a graphical interface between the user, the geometry base and the mesh base, which consists of the following.

- A utility to let the user specify the number of divisions for all entities and/or for picked entities. This is done by displaying the contents of the geometry base, as shown in figure 5.3, and allowing the user to graphically pick entities on the screen and set mesh divisions.
- A meshing utility: divides lines, arcs and cooling lines into elements.
- A utility to delete the mesh.
- A utility to save the mesh to a file for future use.
- A utility to read a file for saved meshes.
- Utilities to add, delete and modify cooling lines.
- A utility for adding, deleting, saving and recovering internal poles.

Once the mesh has been created and placed in memory, it can be displayed at any time, in the form of linear elements, boundary nodes and internal poles, as shown in figure 5.5. It is also possible to list all nodes defining the mesh or the connectivity of the elements, as shown in figure 5.6.

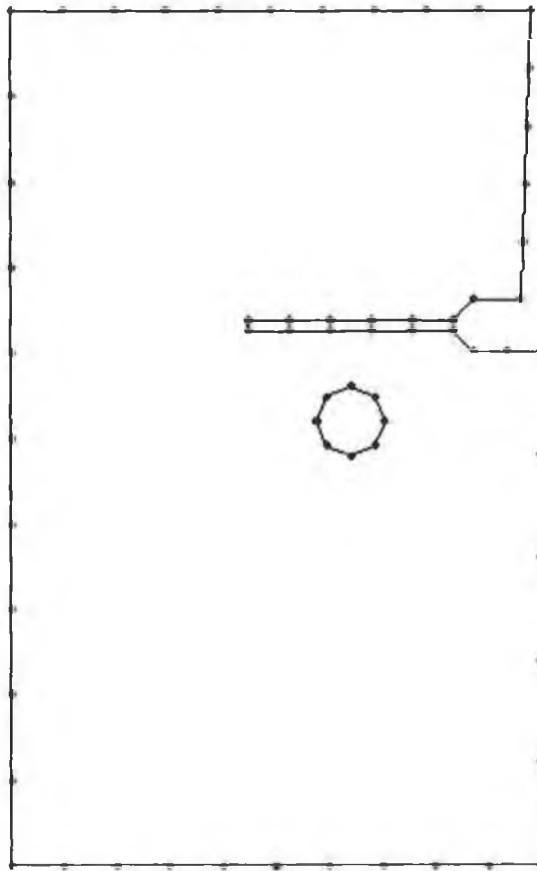


Figure 5.5 – Mesh Display (Discretisation)

List View			
Total number of elements defining boundary: 154			
Boundary Elements			
Element	Node 1	Node 2	Length
1	1	1	0.006
2	2	2	0.006
3	3	3	0.006
4	4	4	0.006
5	5	5	0.006
6	6	6	0.006
7	7	7	0.006
8	8	8	0.006
9	9	9	0.006
10	10	10	0.006
11	11	11	0.006
12	12	12	0.006
13	13	13	0.006
14	15	15	0.006

Figure 5.6 – Element Connectivity Listings

5.4 Analysis

The analysis section of MouldCOOL is actually carried out within a program that was written in FORTRAN and compiled as a 'Dynamic Link Library' or DLL. This was done to make full use of FORTRAN's ability to process complex mathematical code in a very short time. In order to maintain 32-bit operation FORTRAN 90 was used. Table 5.1 shows the subroutines available within the DLL.

Ordinal	Name	Description
1	_ASSEMBGH@28	Assembles [G] and [H] matrices, defined in chapter 3.
2	_BOUNDSS@28	Applies boundary conditions, in steady-state analyses.
3	_BOUNDTR@60	Applies boundary conditions, in transient analyses.
4	_INVERSEF@16	Gets inverse of [F] matrix, as defined by equation 3.22.
5	_RHSMATRIX@36	Evaluates matrix [C] in equation 3.28.
6	_RHSVEC@36	Evaluates vector on right hand side of equation 3.30.
7	_SOLVEBEM@32	Solves set of simultaneous equations.

Table 5.1 – Analysis Sub-Routines

The ordinal positions of the sub-routines were evaluated using the 'Dumpbin.exe' program supplied with the 'Microsoft FORTRAN PowerStation' development studio.

In order to make the sub-routines available to the Visual Basic code the following declarations were made.

Declare Sub assembgh Lib "BEM.DLL" Alias "_ASSEMBGH@28" (Domain As Single, con As Single, le As Single, x As Single, y As Single, G As Single, H As Single)

Declare Sub BOUNDSS Lib "BEM.DLL" Alias "_BOUNDSS@44" (Domain As Single, KODE As Single, G As Single, H As Single, A As Single, B As Single, HC As Single, TA As Single, con As Single, U As Single, Q As Single)

Declare Sub SOLVEBEM Lib "BEM.DLL" Alias "_SOLVEBEM@32" (Domain As Single, KODE As Single, U As Single, Q As Single, HC As Single, TA As Single, A As Single, B As Single)

Declare Sub RHSMATRIX Lib "BEM.DLL" Alias "_RHSMATRIX@36" (Domain As Single, con As Single, x As Single, y As Single, le As Single, S As Single, FINV As Single, H As Single, G As Single)

Declare Sub INVERSEF Lib "BEM.DLL" Alias "_INVERSEF@16" (Domain As Single, FINV As Single, x As Single, y As Single)

Declare Sub RHSVEC Lib "BEM.DLL" Alias "_RHSVEC@36" (MATPROP As Single, Domain As Single, S As Single, H As Single, UP As Single, QP As Single, XY As Single, THETAU As Single, THETAQ As Single)

Declare Sub BOUNDTR Lib "BEM.DLL" Alias "_BOUNDTR@60" (MATPROP As Single, Domain As Single, S As Single, H As Single, G As Single, KODE As Single, con As Single, A As Single, B As Single, HC As Single, TA As Single, U As Single, Q As Single, THETAU As Single, THETAQ As Single)

Before the analysis can be completed, boundary conditions must be applied to the surfaces of the mould. MouldCOOL allows the user to do this by graphical picking of the entities in the geometry base and applying one of six different boundary conditions.

- Constant Temperature.
- Constant Heat Flux.
- Convection.
- Injection Mould Exterior (Natural Convection).
- Injection Mould Cavity.
- Cooling Line (Forced Convection).

The user can only apply boundary conditions to geometry entities if a mesh has been declared. Once the boundary conditions are applied to the entities, the program automatically applies them to the correct elements.

MouldCOOL also allows material properties and analysis options to be input by the user. This is done using a graphical form as shown in figure 5.7.

Figure 5.7 – Analysis Options Dialog

Material Properties

The material properties section allows the user to input the thermal conductivity of the mould metal and the thermal diffusivity. The conductivity is required by steady state and transient analyses programs for the evaluation of heat fluxes. The heat flux is given in terms of the temperature derivative by equation 5.1.

$$q = k \frac{\partial T}{\partial n} \quad (5.1)$$

Where, k , is the thermal conductivity, usually measured in W/mK.

The thermal diffusivity is used only for transient analyses and is the main factor in determining the rate at which a mould reaches its steady state temperature profile. The thermal diffusivity of the metal is given in terms of the material's thermal properties by equation 5.2.

$$\alpha = \frac{k}{\rho c} \quad (5.2)$$

Where ρ is the density of the metal and c the specific heat capacity, usually measured in kg/m^3 and J/kgK , respectively.

Injection Mould Properties

The mould properties, that must be input by the user, are those values required to calculate the heat transfer coefficient at the cavity wall, as described in chapter 2. The input variables, along with their SI units, include the following.

- Cavity Thickness, [m].
- Average Mould Temperature, [$^{\circ}\text{C}$].
- Coolant Temperature, [$^{\circ}\text{C}$].
- Plastic Ejection Temperature, [$^{\circ}\text{C}$].
- Cooling Time, [s].

The cooling time will be estimated by the analysis program if a value of zero is entered in the options dialog.

Plastic Properties

This section allows the user to input values for the following variables. The values are stored in a separate database and hence can be used for any user session. The thermal properties of plastics vary extensively with temperature so the following assumptions were made.

- Linear variation of thermal conductivity with temperature, [W/mK].
- Linear variation of specific heat capacity with temperature, [J/kgK].
- Linear variation of density with temperature, [kg/m^3].
- Plastic melt temperature, [$^{\circ}\text{C}$].

The thermal properties of each polymer in the database can be represented by a linear equation, like those shown in equation 1.1.

The values of these constants for a number of thermoplastics are shown in table 1.3.

The user can use the options dialog box to select any of the thermoplastics in the database. The properties of this plastic will automatically be used. The user can also alter the database by adding or deleting records in the database. Figure 5.8 shows the database operations allowed.

The screenshot shows a software window titled "Thermoplastic Properties". At the top, there is a navigation bar with four buttons: a double left arrow, a single left arrow, a single right arrow, and a double right arrow. Below this, a text field contains the name "Polypropylene". The window displays three sets of linear equations for thermal properties, each with input fields for the slope (m) and intercept (c):

- Thermal Conductivity:** $K = mT + c$
m = 0.0005, c = 0.1343
- Specific Heat Capacity:** $C_p = mT + c$
m = 0, c = 1926
- Density:** $\rho = mT + c$
m = 0.000125, c = 897.5

Below these, there is a "Melt Temperature:" label followed by a text field containing the value "230". At the bottom of the window, there is a row of four buttons: "Add", "Delete", "Refresh", and "Update".

Figure 5.8 – Polymer Database

5.5 Post-Processing

The post-processing part of the software is that part, that supplies a link between the results of the analysis section and the user. The post-processor uses a number of methods, to do this, as illustrated in figure 5.9.

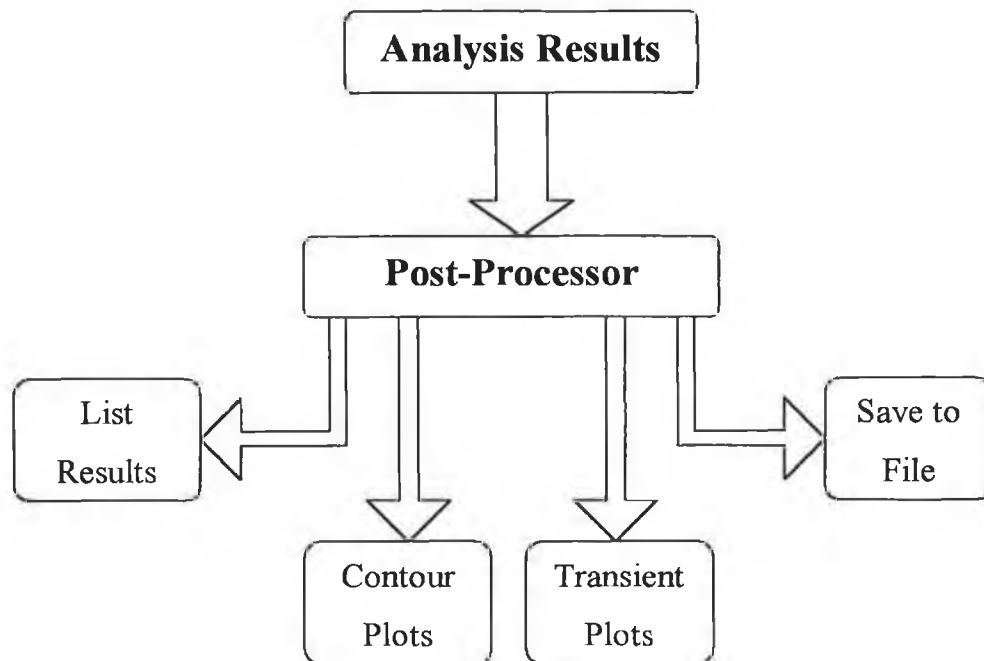


Figure 5.9 – MouldCOOL Post-Processing Abilities

The post-processor can be used to display any part of the overall analysis procedure through, lists, plots and/or graphs. The following is a list of the full capabilities of the post-processor.

- Plot nodes, elements, temperatures and fluxes. This part of the post-processor will display the graphics screen along with the required display. Figure 5.10 shows an example of a temperature plot of a square, subject to a temperature difference of 300°C. Figure 5.11 shows the flux plot.

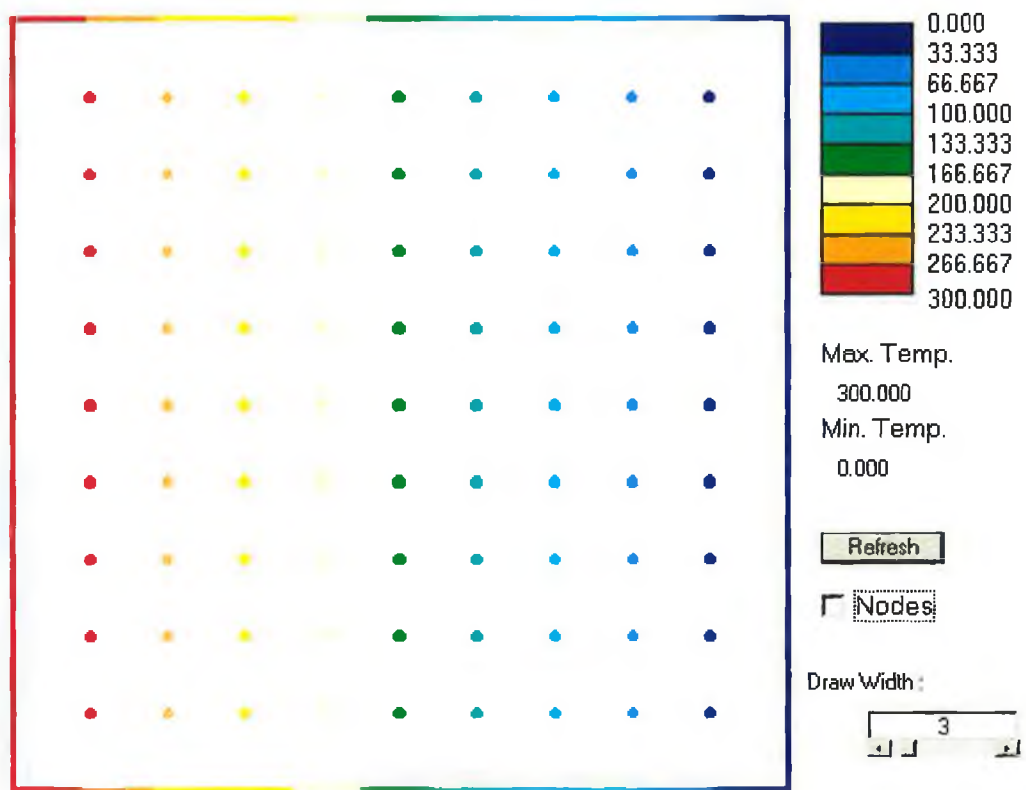


Figure 5.10 – MouldCOOL Temperature Plot

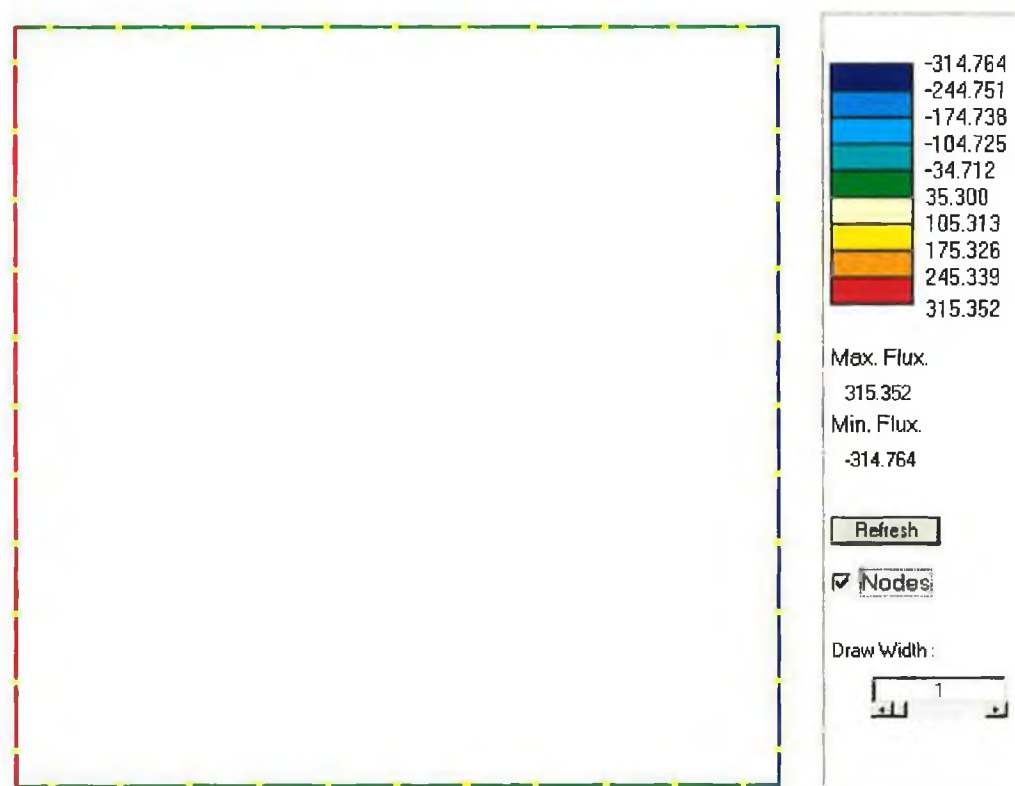


Figure 5.11 – MouldCOOL Flux Plot

- List nodes, elements, temperatures, fluxes and boundary conditions. This part of the post-processor will display the list screen along with the required data.
- Graph transient temperature and flux. Figure 5.12 shows an example of a transient temperature plot of the example shown in figure 5.10 and 5.11.

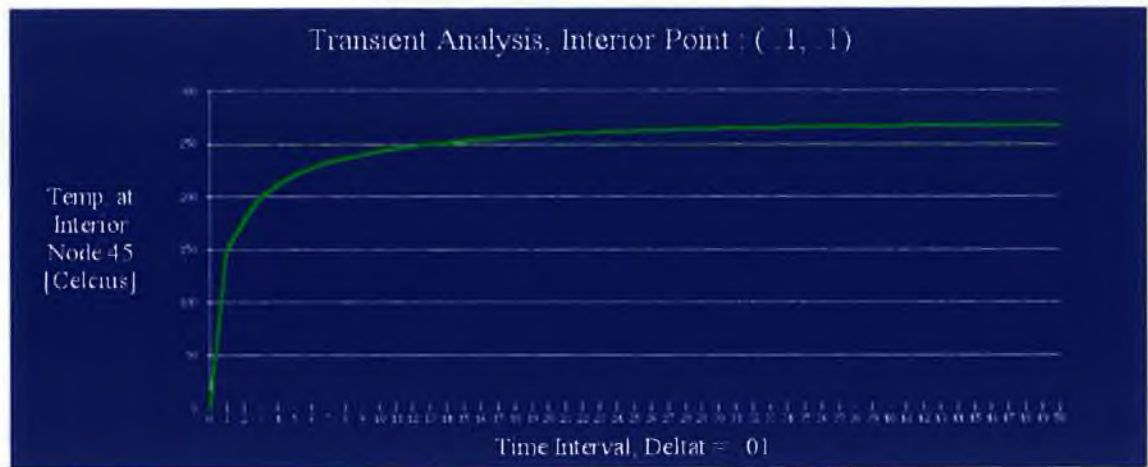


Figure 5.12 – MouldCOOL Transient Temperature Plot

The last part of the post-processor, and probably the most important, is the MouldCOOL conclusion. This gives the user the following information.

- Heat lost to atmosphere.
- Heat extracted from cavity.
- Cooling system efficiency.
- Cooling time used.
- Minimum possible cooling time.

Using this information, the user can re-arrange the cooling system to increase the efficiency and/or to decrease the minimum possible cooling time. An example of the MouldCOOL conclusion dialog is shown in figure 5.13.

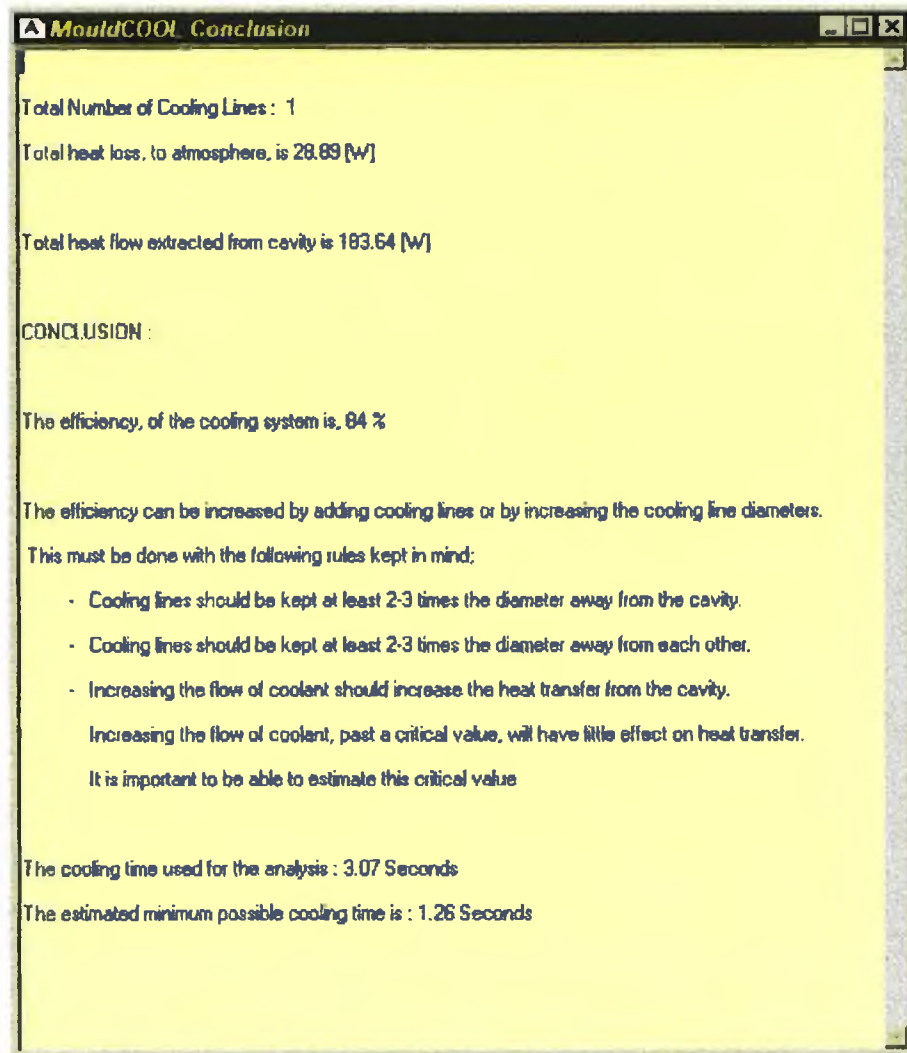


Figure 5.13 – MouldCOOL Conclusion Dialog

5.6 MouldCOOL Interface

The interface between the methods and procedures, just described, consists of a menu system, as shown, in figure 5.2. A description of the complete menu systems is as follows.

5.6.1 File Menu

Close – Closes the currently open MouldCOOL file and unloads all variables from memory.

Open – Displays a file dialog box so that the user can resume a previously saved MouldCOOL file. MouldCOOL files have the extension *.mcl.

Save – Saves the current file.

Save As – Displays a dialog box so that the current file can be saved with any name.

Exit – Unloads MouldCOOL.

5.6.2 Edit Menu

Password – Allows the user to set a password for start-up.

5.6.3 View Menu

Geometry – Lets the user view the contents of the geometry base.

Elements – Displays the contents of the mesh base.

Details – Lists the entities in the geometry base.

5.6.4 Pre-Processor Menu

AutoCAD – Allows the user to initialise AutoCAD.

Geometry – Allows the user to use AutoCAD to select geometry and to write geometry to AutoCAD's database. Also lets user save or open data files containing geometry.

Divisions – Allows user to set the number of divisions allowed by the mesh generator for individual or all entities.

Mesh – Allows the user to mesh the geometry or to delete a current mesh. Lets the user open or save data files containing details of a mesh.

Cooling Lines – Displays cooling line dialog box, so that cooling lines can be added, deleted or modified.

Internal Poles – Presents options for selecting internal points, using AutoCAD or an external data file.

5.6.5 Analysis Menu

Options – Displays dialog box to set analysis options and material properties.

Boundary Conditions – Displays dialog to set up boundary conditions on entities in geometry base.

Analyse – Let's the user start any of the analyses detailed in chapter 2.

5.6.6 Post-Processing Menu

Plot – Allows the user to graphically plot elements, nodes temperatures or fluxes.

List – Allows the user to list elements, nodes, temperatures, fluxes or boundary conditions.

Graph – Displays a line graph of temperature versus time for any node in the mesh. This command can only be used if a transient analysis has been completed.

Conclusion – Describes the efficiency of the cooling system. This command can only be used if a steady-state injection mould analysis has been completed.

5.6.7 Quick Menu

This menu provides a number of commands to change specific variables. These variables are as follows.

- Polymer melt temperature.
- Cavity Thickness.

- Coolant Flow Rate.
- Coolant Temperature.
- Ambient Temperature.

CHAPTER 6

DESIGN AND MANUFACTURE OF THE TEST MOULD

6.1 Introduction

The software discussed in chapter 5 was developed to analyse the cooling system of injection moulds. This is done by first developing a two-dimensional section of the injection mould. The software then applies boundary conditions and analyses the mould core using a 'boundary element' technique. The purpose of the analysis is to predict a temperature and flux profile throughout the mould core. Once the temperature and flux profile is known, the efficiency of the cooling system and the minimum possible cooling time can be calculated. In chapter 4 of this thesis, the boundary element method (BEM) was used to solve a simple heat transfer problem and the results compared to those of an analytical method and a finite element analysis. In order to further prove the validity of the BEM and the injection moulding software a test mould was built and tested.

A simple square-plate injection mould consisting of two cavities was manufactured. The mould would produce simple plastic plates 30mm square and 1.5 mm thick. Photographs of the test mould and the injection-moulding machine are shown in figures 6.9 and 6.10. The mould utilised a sprue and runner system supplying two cavities. The plastic part produced by this mould is shown in figure 6.1.

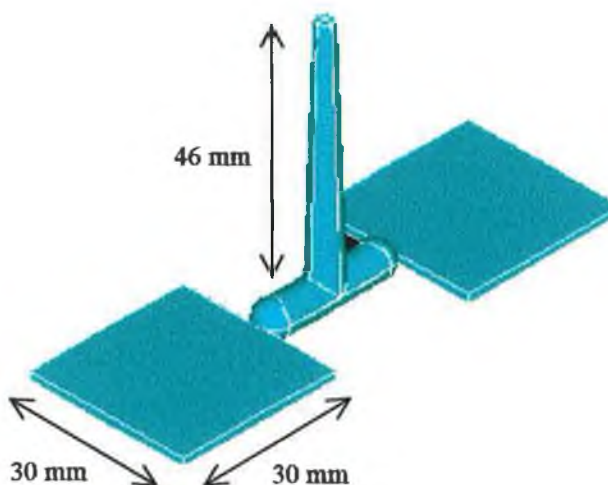


Figure 6.1 – Plastic Part

The first problem in any injection mould design is the limitation imposed by the size of the injection-moulding machine to be used. The injection-moulding machine supports the mould on platens, using tie bars, and hence, the mould must be small enough to fit between the tie bars. The machine used for the current problem had a tie bar arrangement as shown in figure 6.2.

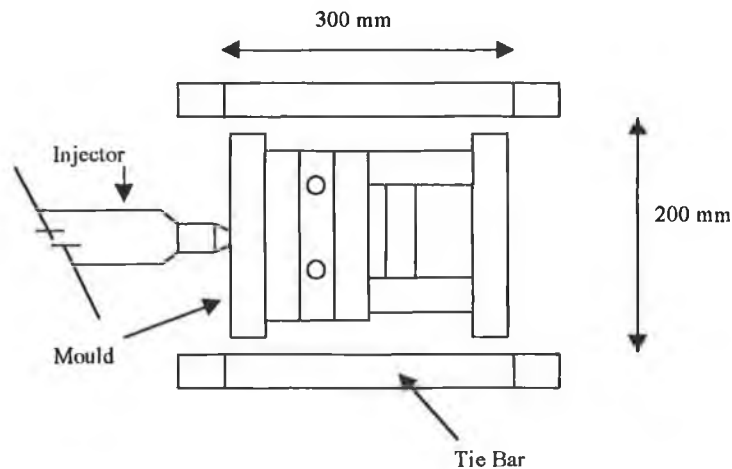


Figure 6.2 – Tie Bar Arrangement

A typical injection mould will consist of a number of distinct parts, the core and cavity plate, the ejector system and the clamp plates, as shown in figure 1.2. This structural similarity of injection moulds means that they can be made using standard sized plates. The mould system can be purchased in “kit form” or as an assembled unit. This practice eliminates the, relatively unimportant, procedure of starting from scratch with bare sheets of metal. The plates come with holes included for guide pins, ejector bars and ejector pins.

Once the injection machine has been properly sized, the dimensions of the mould plates can be determined. Determining the size mould that can be fit within the platen is the first step. The main specifications for a mould are the dimensions of the clamp plate. The standard sizes range from 095mm × 095mm to 796mm × 996mm. The distance between tie bars will determine the maximum plate size that can be used. The thickness of the plates can be determined by considering the distance between platens and the length of the ejector stroke. The general dimensions that can be specified for a typical ‘injection mould kit’ are shown in figure 6.3.

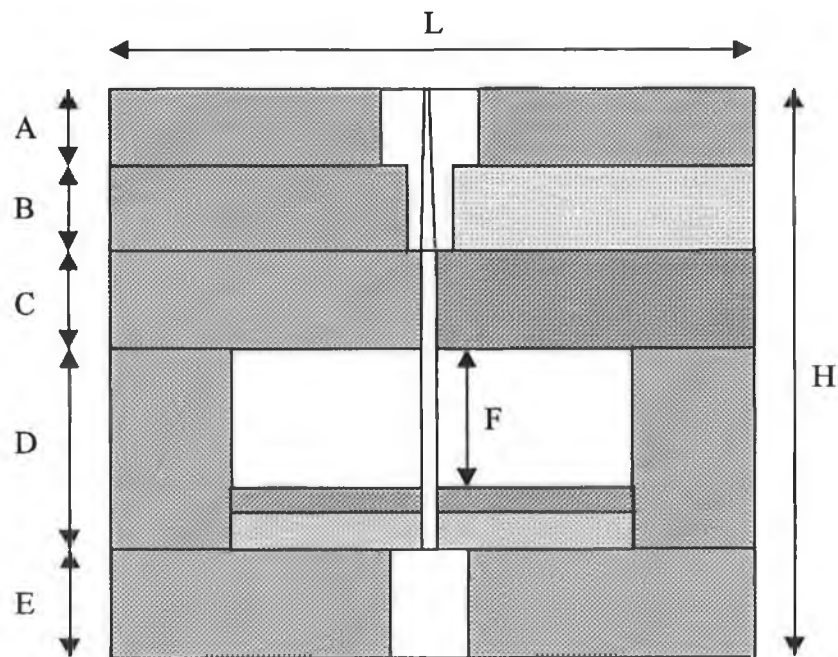


Figure 6.3 – Standard Injection Mould Parts

The dimensions, shown in figure 6.3, represent the following.

Length of mould	L
Height of mould	H
Depth of clamp plate (usually same as E)	A
Depth of core plate	B
Depth of cavity plate	C
Depth of ejector space	D
Depth clamp plate (usually same as A)	E
Ejector stroke	F

Due to the size of the injection-moulding machine, as shown in figure 6.2, a 156mm by 196mm mould kit was shown. An assembly drawing, including the 'DMS' purchase code for each element of the mould is shown in figure 6.4. A detailed drawing of the machined parts of the mould is shown in figure 6.5 and detailed part drawings of the mould can be found in appendix 2.

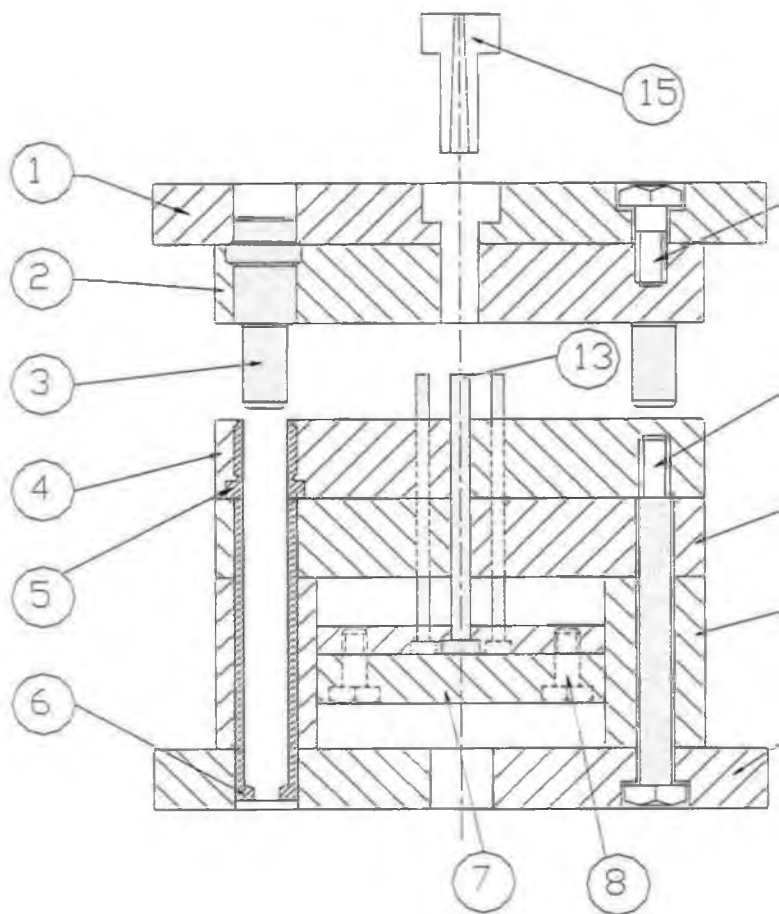


Figure 6.4 – Test Mould Assembly Drawing

14

12

11

10

9

Part	Qty.	Name	DMS Code
1	1	Clamp Plate	CP1519H
2	1	Core Plate	A151926N8
3	4	Guide Pillar	GP142635
4	1	Cavity Plate	B151926N8
5	4	Guide Bush	GB1426
6	4	Liner	L2080
7	1	Ejector Set	-st-
8	4	M8 cap screw	820CS
9	1	Clamp Plate	CP1519H
10	2	Riser	R151956
11	1	Backing Plate	BP1519
12	4	M10 cap screw	1090CS
13	9	Ejector Pins	-ns-
14	4	M10 cap screw	1020CS
15	1	Sprue Bush	-ns-

Drawing Name:

"Test Injection Mould Assembly"

Drawing By:

Niall Moran

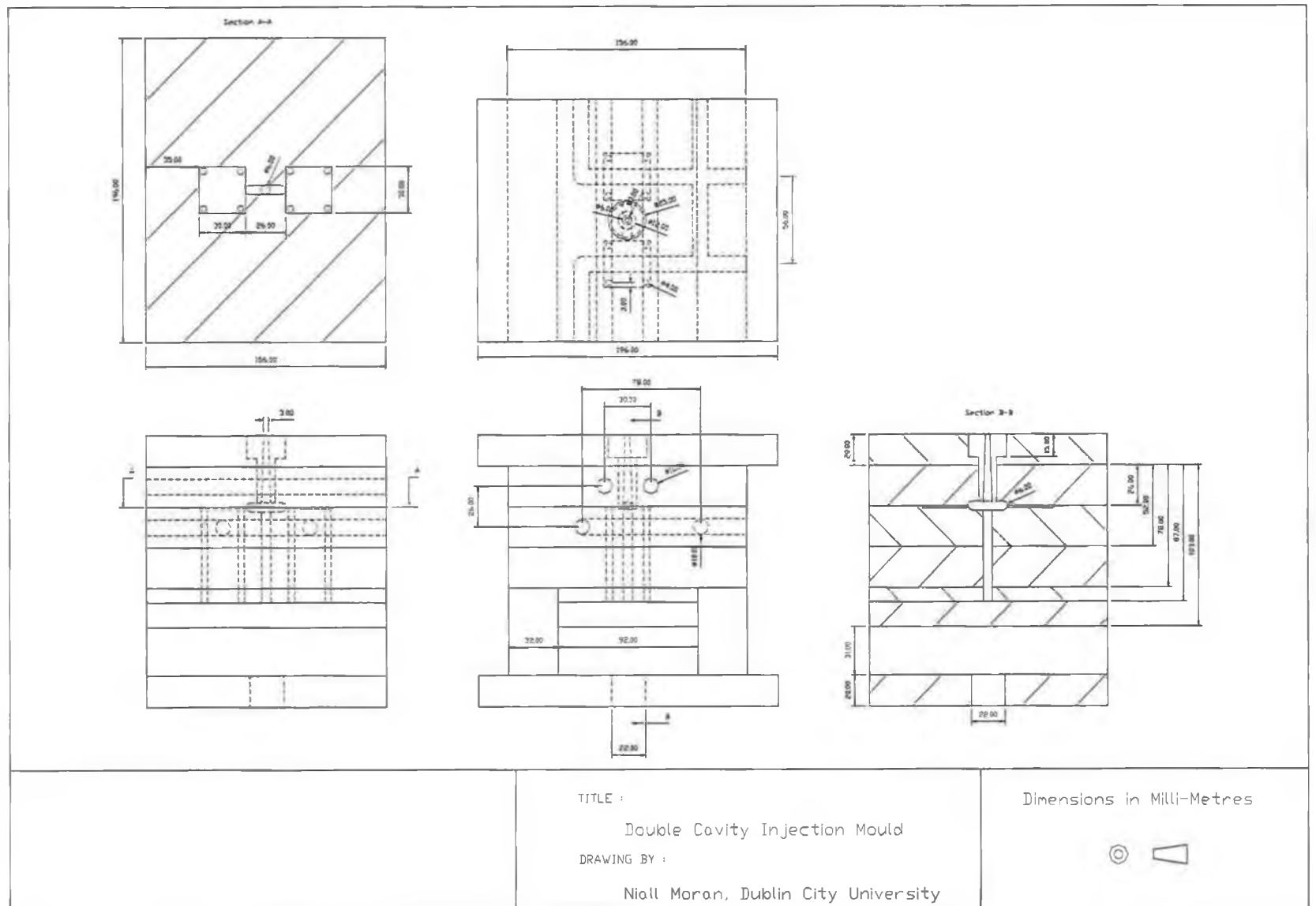
Dublin City University

All Parts Supplied by DMS

-ns- : Not Supplied

-st- : Standard Size

Figure 6.5 – Detailed Mould Drawing



6.2 Mould Cavity and Core

An injection mould cavity is that part of the mould which is filled with plastic to form the shape of the required part. The plate that contains this cavity is called the cavity plate. In general the thickness of plastic parts are quite small and hence a typical cavity will have a core, as illustrated in figure 6.6.

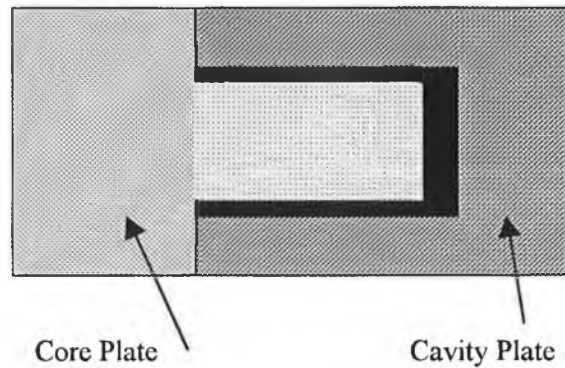


Figure 6.6 – Cavity and Core of Injection Mould

The cavity can be incorporated into the mould in two different ways.

Integer Type

This type exists when the cavity and core are machined directly into the plates of the mould. This means that in order to change the cavity two new plates must be inserted.

Inserts

‘Inserts’ consist of small blocks of metal with the cavity and core machined into them. These inserts can be inserted into the core and cavity plates. This method is best used for multi-cavity moulds where a number of different parts are to be produced. The main advantage of this mould is that by changing the part to be produced the mould does not have to be re-machined just the inserts.

The test mould manufactured was of the integer type with the cavity cut directly into the cavity plate. A computer model of the plastic part produced is shown in figure 6.1.

6.3 Plastic Part Ejection

After the plastic has cooled to its solidifying temperature, it will have contracted from its original volume. This will result in the plastic part clinging to the mould. In order to remove the part and keep the process automatic an ejection system is incorporated. A typical ejection system will consist of a number of ejector bars that will move forward and push the part away from the mould. It is important that this is not done manually, as the cycles must run automatically if a uniform temperature profile is to be maintained. The ejection system is usually incorporated within the moving half of the mould, but is itself kept stationary using an ejector bar. When the moving half is retracted, after cooling, the stationary ejector pins push the part away from the mould's parting-surface.

The test mould consisted of five ejector pins, as shown in figure 6.7, four for each of the squares and one at the centre of the runner.

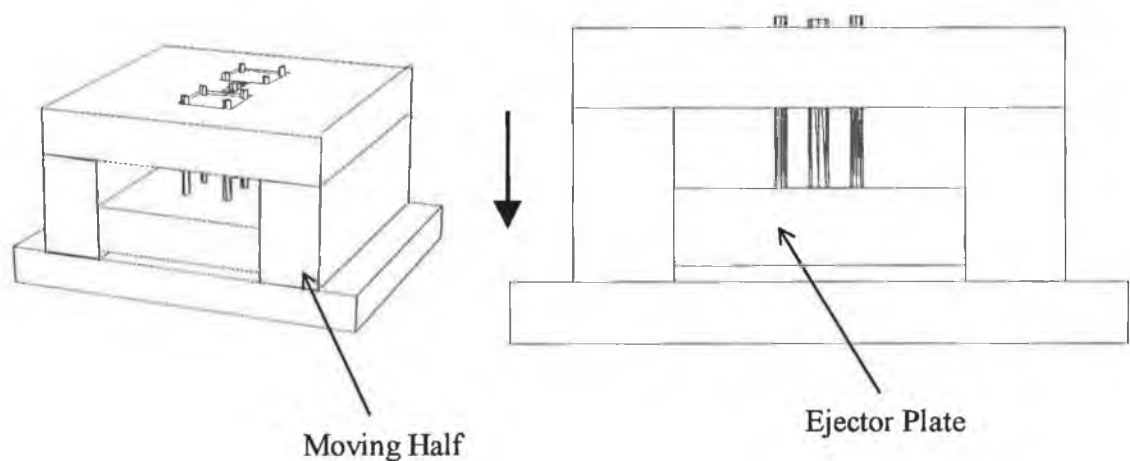


Figure 6.7 – Ejection System

6.4 Feed System

In order to supply the cavities with the plastic material a channel is provided between the cavities and the injector nozzle. This channel is termed a feed system. Normally a feed system will comprise of a sprue, a runner and a gate. A sprue is a frustum shaped channel that conveys the molten plastic from the nozzle to the cavity and core plate.

The runner is the channel connecting the sprue to the gates that allow the plastic to flow into the cavities. The overall feed system for the test mould is illustrated in figure 6.8.

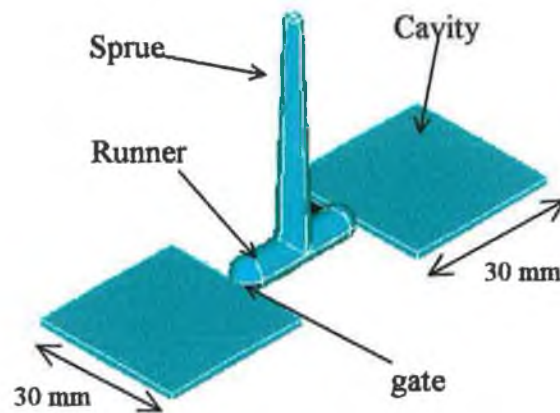


Figure 6.8 – Feed System

The test mould was manufactured with two impressions and hence a very simple runner system was designed. The runner was made larger in diameter than the thickness of the cavity to ensure uniform flow into the cavity. The runner was made 3mm in radius and the cavity 1.5mm thick.

6.5 Mould Cooling

After the main parts of the injection mould were machined, a cooling system was to be incorporated. It was important to make the cooling channels accessible to all parts of the cavities. A standard 10mm-bore cooling-channel was adopted. The cooling network consisted of two channels in the fixed half and a single channel in the moving half, as illustrated by figure 6.9.

The final mould parting surfaces are shown in figure 6.10.

Cavity Plate (Moving)

Core Plate (Fixed)

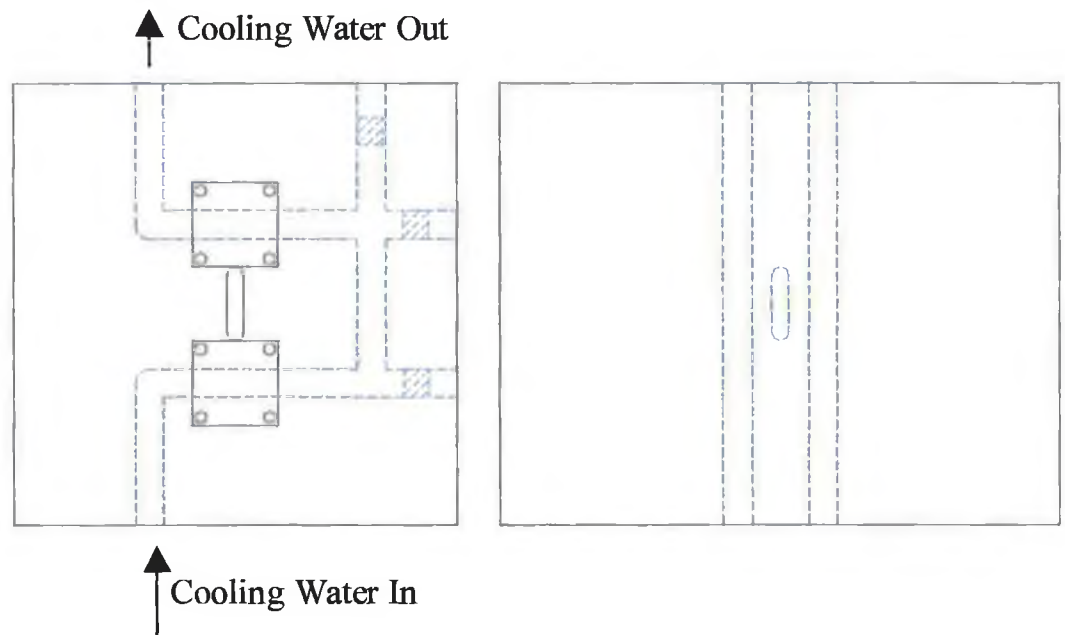


Figure 6.9 Test Mould Cooling System

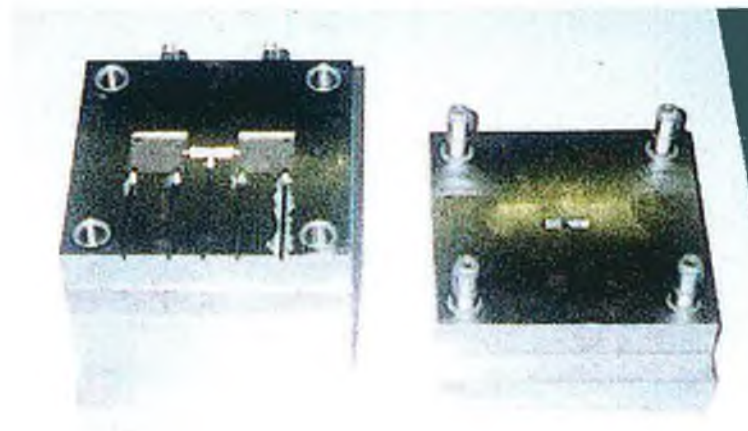


Figure 6.10 – Finished Test Mould

CHAPTER 7

TEST RESULTS AND ANALYSIS

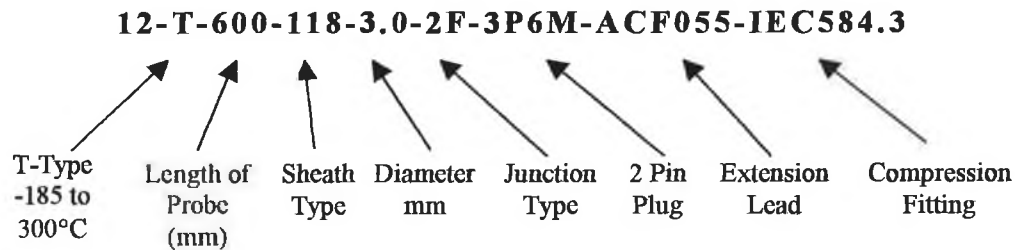
7.1 Introduction

In predicting the optimum size and position of cooling lines within a typical injection mould core, the temperature profile throughout the core must also be predicted. Before the analysis software can be reliably used, it must be shown that the temperature profile predicted by the software agrees with practical results. To show this, temperatures within the test mould, as described in the previous chapter, were recorded for a number of cycles. This was done using 3mm diameter thermocouples located at different locations across a section of the mould. Due to the thickness of the thermocouples and the size of the mould, it was only possible to locate five holes for the thermocouples, as shown in figure 7.2. To show the effect of different thermoplastic materials, two were used, 'Low Density Polyethylene' and 'Polypropylene Copolymer'.

The following chapter describes the test procedure and all the equipment and instrumentation used. The mould is set up for numerical analysis and the temperature profile predicted and compared with the test results. A sample optimisation is then shown to emphasise the factors influencing injection mould cooling systems design.

7.2 Equipment and Instrumentation

In order to take temperature measurements, thermocouples were purchased from TC Ltd. The thermocouples were of type '12' with the following purchase code:



This thermocouple configuration had a response time of 0.8 seconds. The thermocouples were connected to the parallel port of a PC via a 'Pico TC-08' data logger, as shown in figure 7.1.

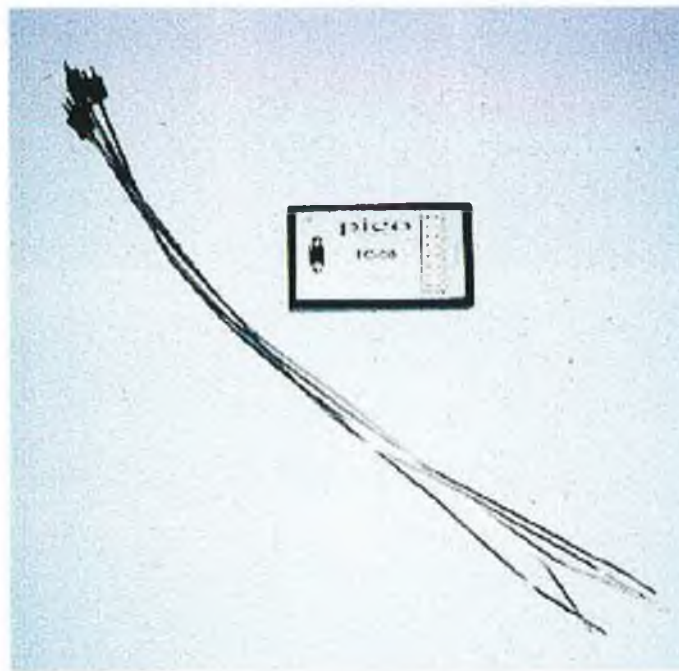


Figure 7.1 – Thermocouples and PC Data Logger

The data logger allowed temperatures to be read every 0.5 seconds for 500 readings. The temperature values were displayed on the screen as they were recorded. To incorporate the thermocouples into the core 3-mm holes were drilled through the metal of the cavity plate. These holes allowed the thermocouple probes to be positioned without the need for cement. The positions of these holes are shown in figure 7.2.

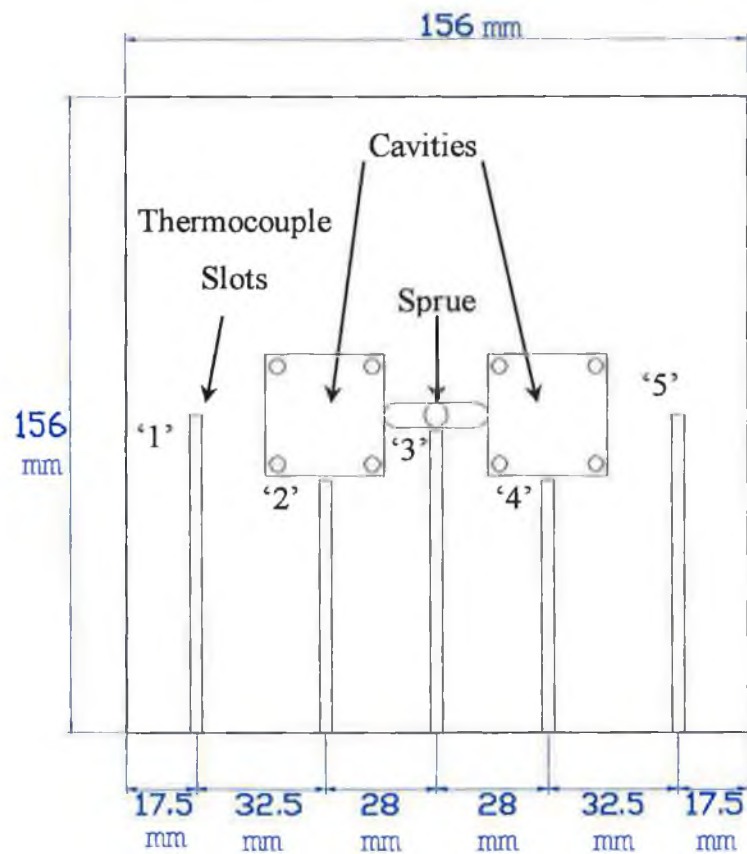


Figure 7.2 – Thermocouple Positions

The actual injection-moulding machine used for the tests is shown in figure 7.3

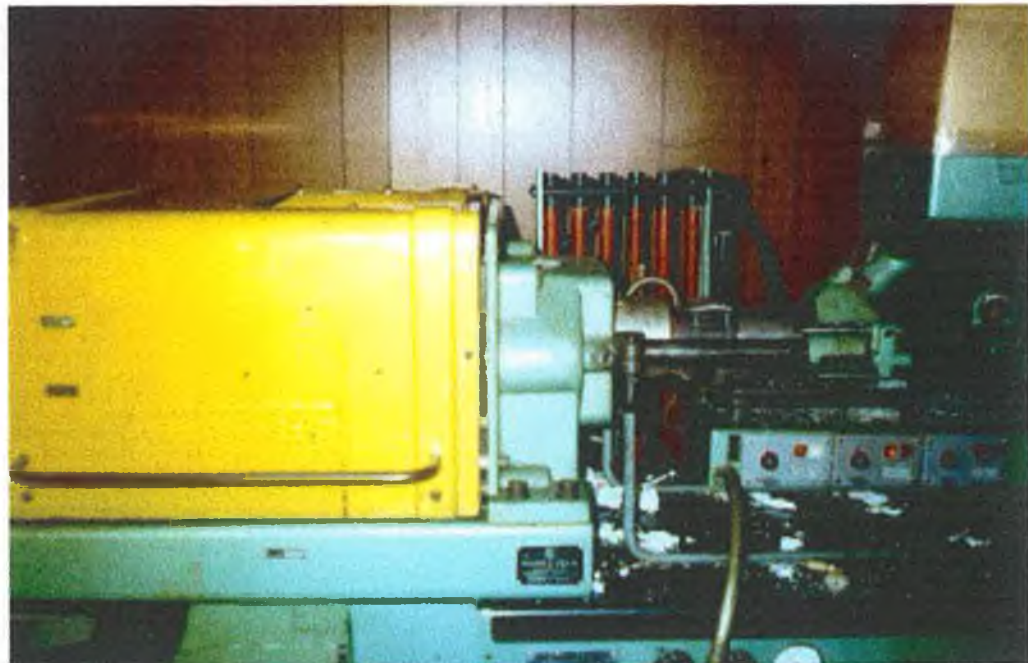


Figure 7.3 – Injection Moulding Machine used for Testing

7.3 Test Procedure and Results

Once the injection mould was hoisted on to the moulding machine, the following procedure was adopted to carry out the testing.

- The plastic filler was filled with low-density polyethylene pellets and the melt temperature set to 190°C.
- The coolant was connected and set to 20°C.
- Once the mould was perfectly centred the thermocouples were attached and the data logger initialised.
- The logger continued to monitor the temperature profile for approximately 25 cycles. This was to ensure that the mould cyclic behaviour had reached steady state.
- The test was repeated for polypropylene copolymer, with a melt temperature of 230°C.

The results logged for the five points, shown in figure 5.2, for polyethylene and polypropylene are shown in figures 7.4 and 7.5, respectively.

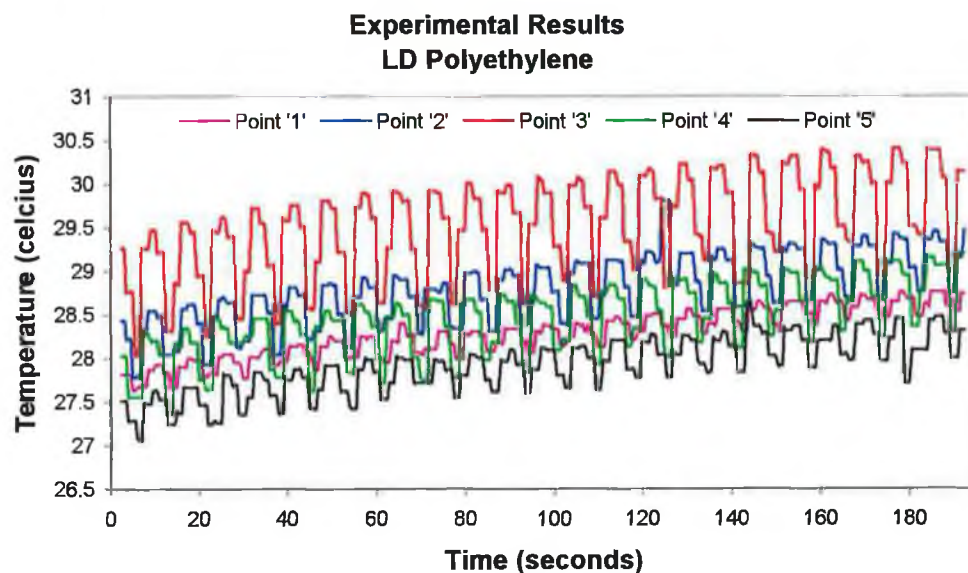


Figure 7.4 – Experimental Data for Low Density Polyethylene

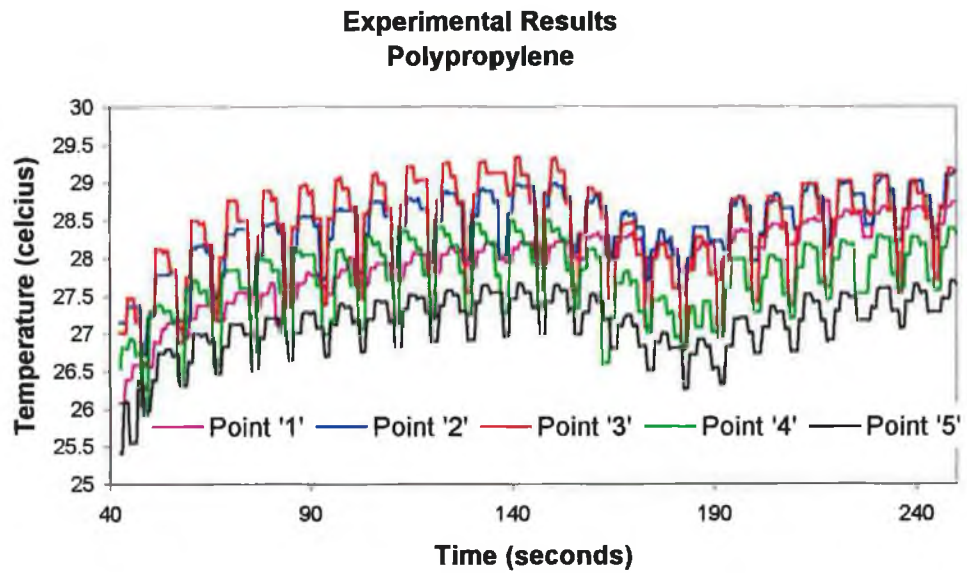


Figure 7.5 – Experimental Data for Polypropylene (Copolymer)

Figures 7.6 and 7.7 show the experimental temperatures at the five points for a typical cycle.

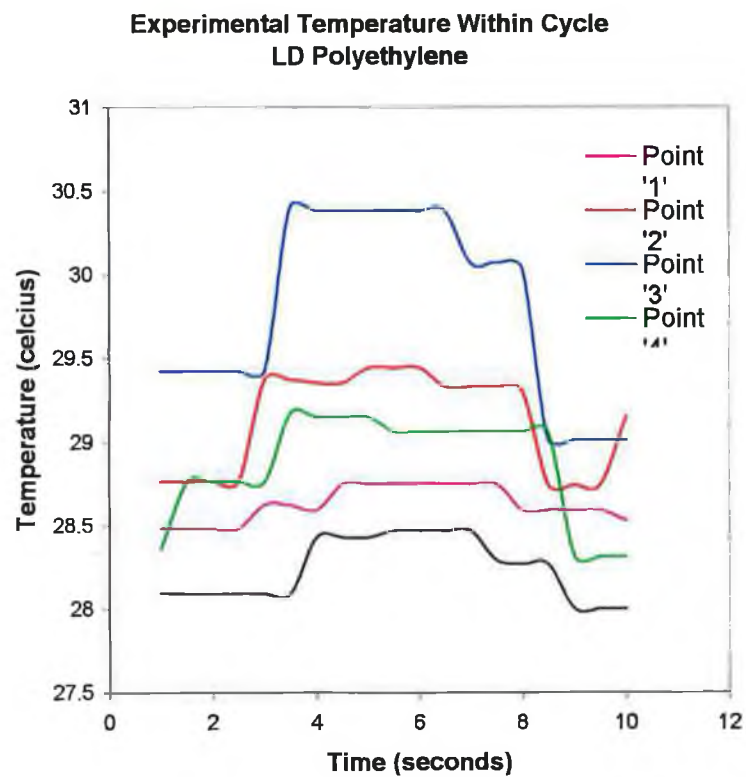


Figure 7.6 – Experimental Data for Typical Cycle - Polyethylene

Experimental Temperature Within Cycle Polypropylene

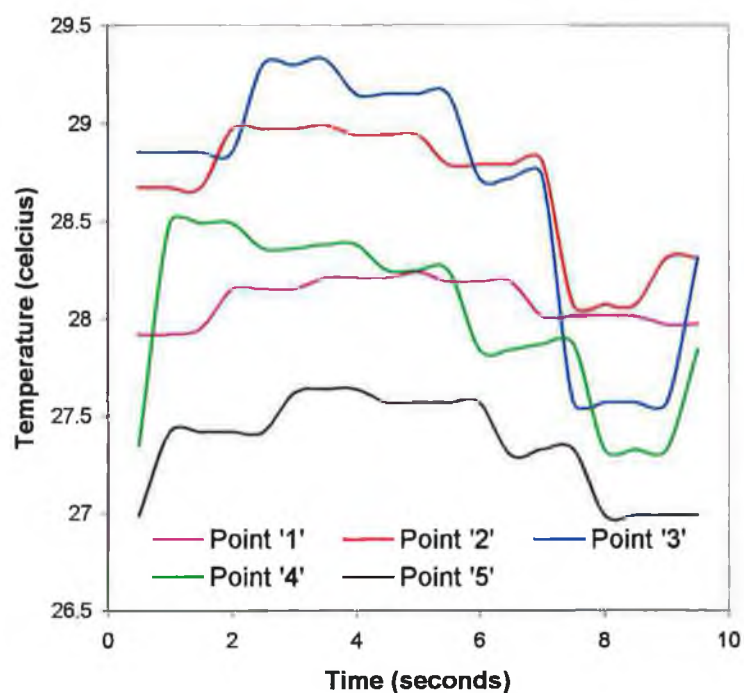


Figure 7.7 - Experimental Data for Typical Cycle - Polypropylene

The experimental cyclic averaged temperatures for each of the five points are shown in table 7.1.

	Temperature (°C)				
	Point '1'	Point '2'	Point '3'	Point '4'	Point '5'
Thermoplastic					
LD Polyethylene	28.63	29.13	29.79	28.86	28.24
Polypropylene	28.09	28.67	28.67	28.02	27.36

Table 7.1 – Cycle-Averaged Temperatures at Five Points

7.4 Computer Simulation of Test Mould

In order to conduct the analysis the following steps were taken:

- A two-dimensional section representing the cavities and cooling lines was drawn, as shown in figure 7.8.

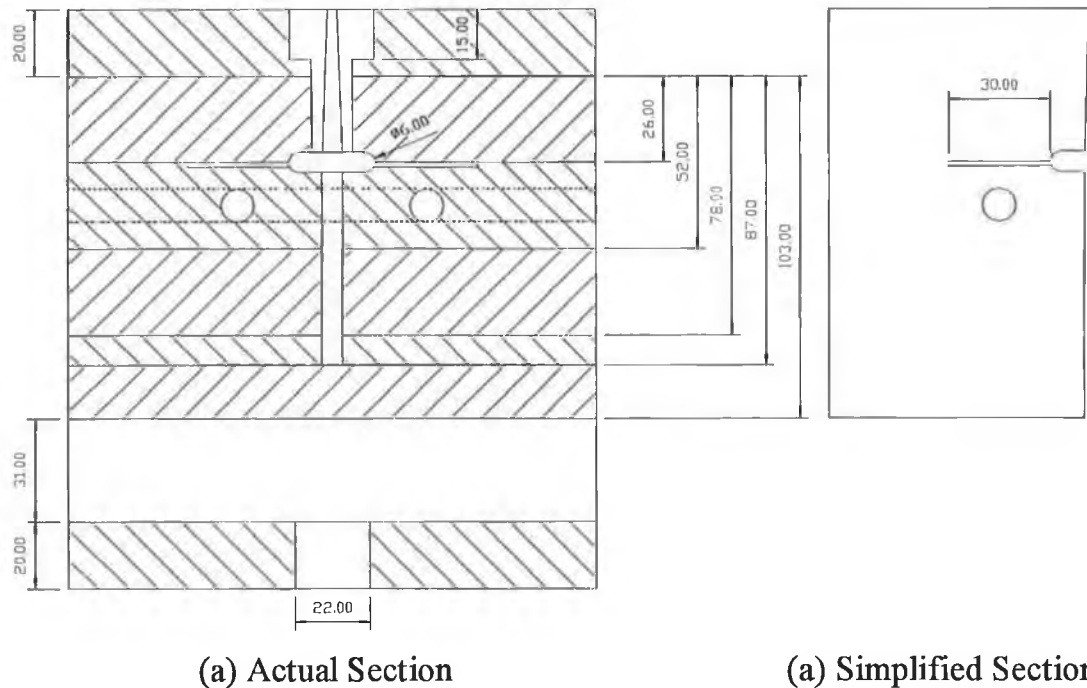


Figure 7.8 – Simplified 2D Mould Section

Figure 7.8 shows the actual section of the mould and the outline representation that would be used for the analysis system. This section incorporates the two cavities, but cannot incorporate the cooling channels completely, because of their three-dimensionality, see appendix 3. Figure 7.8 shows the hidden cooling line by dotted lines. In order to allow for this, a cooling channel is added to take account of the channels that cannot be seen by the section. Considering symmetry the section used for the analysis is shown in figure 7.9.

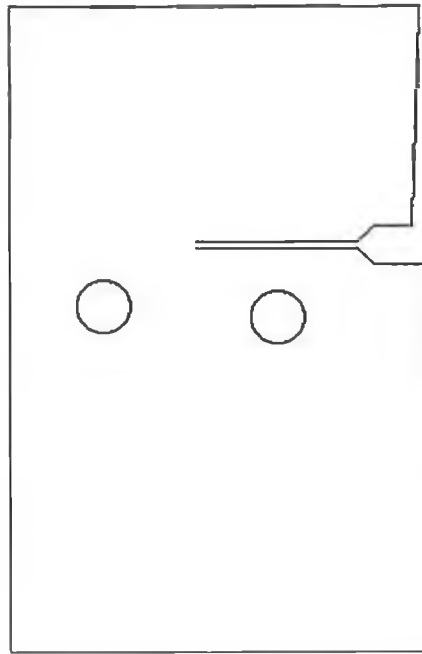


Figure 7.9 – Final Mould Section used for Analysis

- The section shown in figure 7.9 was discretised. The mesh created by this discretisation is shown in figure 7.10.

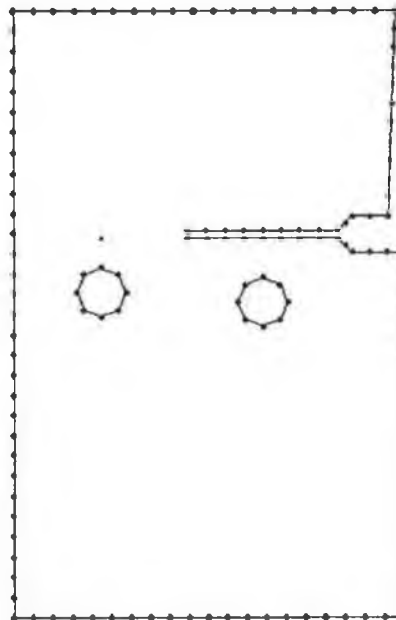


Figure 7.10 – Mould Section Discretisation

- The boundary conditions were set up using the following data and material properties.

Mould Material:

Thermal Conductivity: 46.7W/mK

Thermal Diffusivity: 0.0000142m²/s

Injection Moulding Cycle Properties:

Cavity Thickness: 1.5 mm

Cooling Time: 10s.

Coolant Temperature: 20°C.

Polyethylene Thermal Properties:

$$k = 0.00217T + 0.2453 \text{ [W/mK]}$$

$$\rho = 0.0003375T + 906.25 \text{ [kg/m}^3\text{]}$$

$$C_p = 7.851T + 2355.2 \text{ [J/kgK]}$$

Melt Temperature = 190°C

Polypropylene Thermal Properties

$$k = 0.0005T + 0.1343 \text{ [W/mK]}$$

$$\rho = 0.000125T + 897.5 \text{ [kg/m}^3\text{]}$$

$$C_p = 1926 \text{ [J/kgK]}$$

Melt Temperature = 230°C

- A cycle-averaged analysis, as described in chapter 3, was completed. The completed analysis resulted in the temperature plot, as shown by figures 7.11 and 7.12 for polyethylene and polypropylene, respectively.

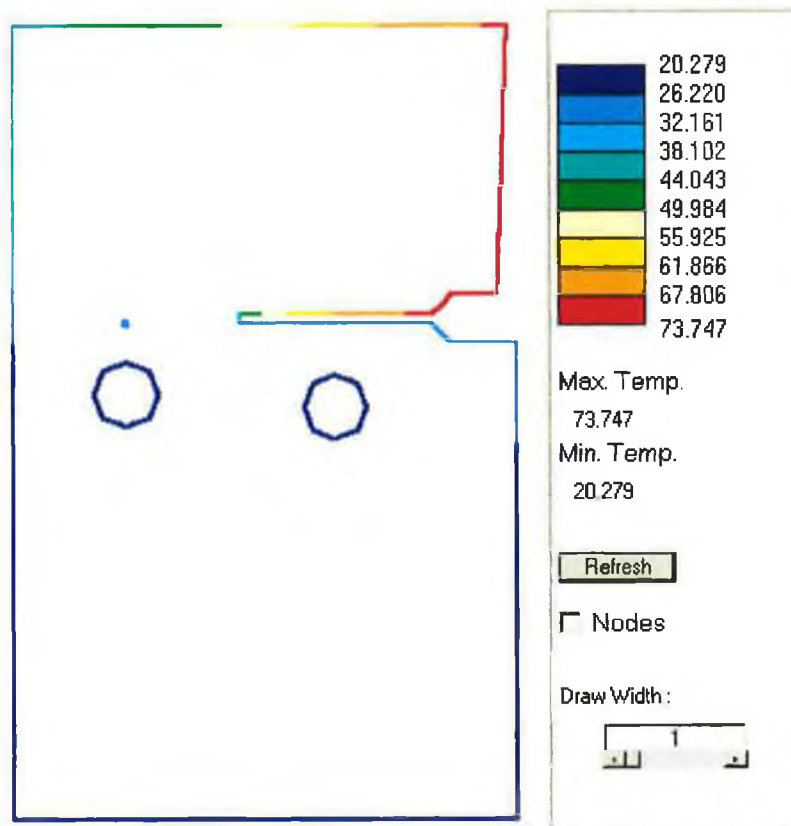


Figure 7.11 – Cycle-Averaged Temperature Plot for Polyethylene

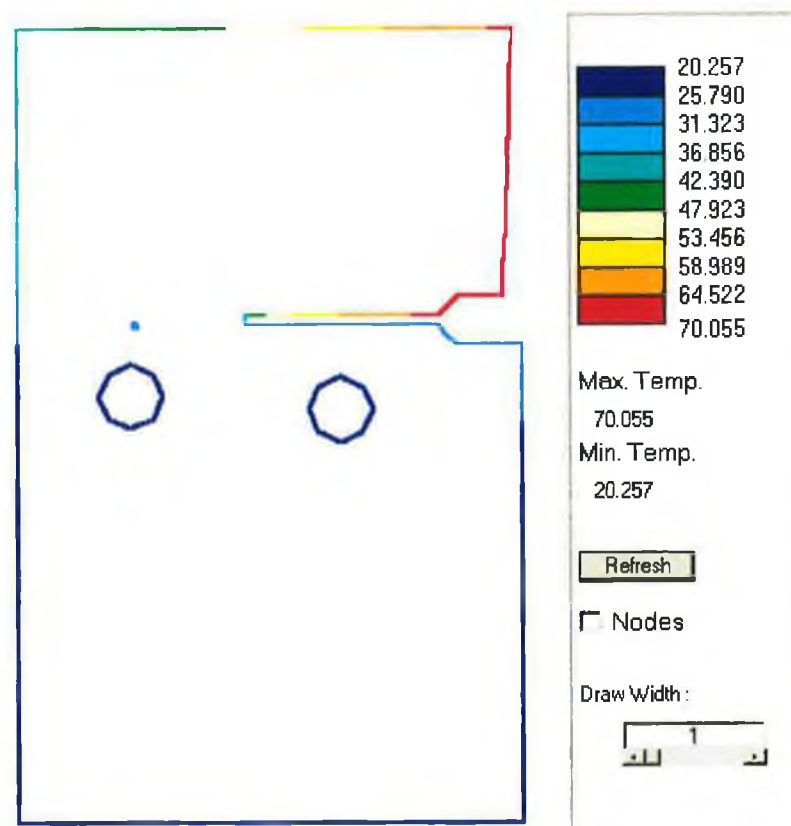


Figure 7.12 – Cycle-Averaged Temperature Plot for Polypropylene

The temperatures at the required points were taken directly from the MouldCOOL display screen, shown in figures 7.11 and 7.12. Since the section of the mould was simplified using symmetry, the temperatures at points '4' and '5' were assumed equal to the temperatures at points '1' and '2'. Tables 7.2 and 7.3 show the comparison between the numerical results of the cycle-averaged analysis and the results of experimentation.

	Temperature (°C)				
	Point '1'	Point '2'	Point '3'	Point '4'	Point '5'
Experiment	28.63	29.13	29.79	28.86	28.24
Numerical Results	28.9	28.5	31.65	28.5	28.9
% Error	0.94	2.2	6.24	1.25	2.34

Table 7.2 – Comparison of Experimental/Numerical Results for Polyethylene

	Temperature (°C)				
	Point '1'	Point '2'	Point '3'	Point '4'	Point '5'
Experiment	28.09	28.67	28.67	28.02	27.36
Numerical Results	28.22	28.04	30.14	27.48	28.22
% Error	0.46	2.2	5.13	1.93	3

Table 7.3 – Comparison of Experimental/Numerical Results for Polypropylene

After the temperature profiles were predicted MouldCOOL gave a breakdown of the heat losses throughout the mould. The breakdown, for both polypropylene and polyethylene, as well as the efficiency (as explained in chapter 1 of this thesis) are shown in table 7.4.

Plastic	Heat Loss from Cavity(W)	Heat Gained by Coolant (W)	Heat Lost to Environment (W)	Efficiency %
Polypropylene	72.51	70.35	2.16	97
Polyethylene	80.48	78.11	2.37	97

Table 7.4 – Cooling Line Efficiency

In order to increase this efficiency an extra cooling line was added in to extract more heat from the cavity. The geometry and a discretisation of 149 elements are shown in figure 7.13.

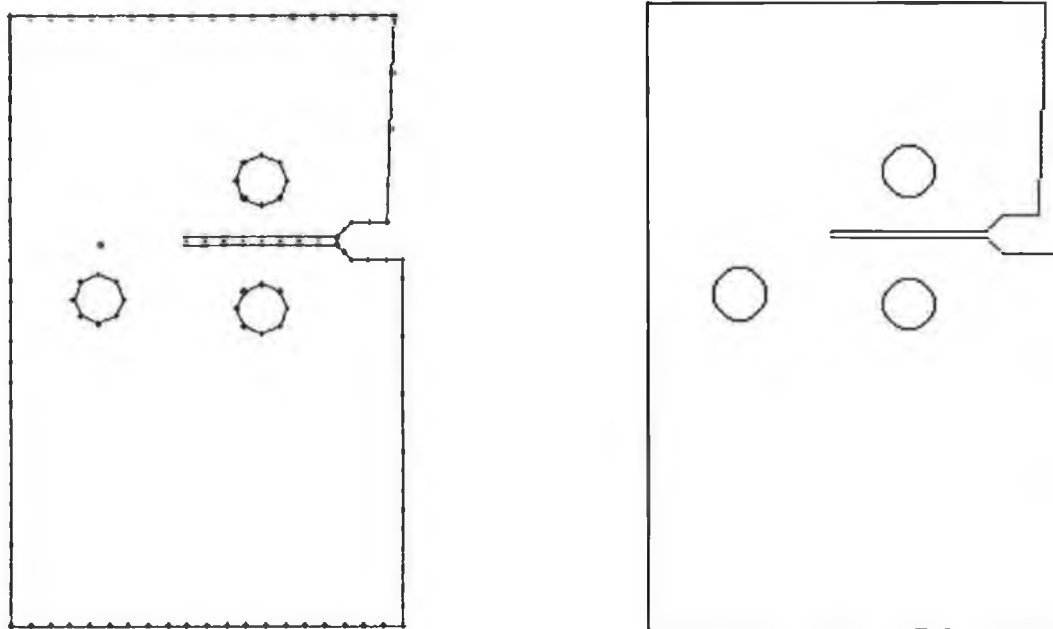


Figure 7.13 – Mould Discretisation with extra Cooling Line

The heat loss to the environment was calculated as 0.36W compared with 2.16W for the mould with only two cooling lines.

It can be seen from table 7.2 and 7.3 that the predicted mould temperatures are slightly higher than the experimental values. It is, however, not safe to say that the analysis over predicts the temperatures until the analysis is repeated using a finer mesh. Table 7.5 shows the results for the polypropylene analysis using a number of different meshes. This table also shows the time taken for the analysis to complete.

No. of Elements	Predicted Temp. at Point '3'	Experimental Temp. at Point '3'	% Error	Time Taken for Analysis (s)
53	34.079	28.67	18.87	62
141	30.141	28.67	5.13	19
185	29.936	28.67	4.4	8
228	29.99	28.67	4.6	2

Table 7.5 – Mould Temperature Prediction for Different Meshes

It can be seen from table 7.5 that the mould temperature predictions converge on values slightly greater than the experimental values. It also may be concluded from table 7.5 that increasing the fineness of the mesh past a certain level will only increase the analysis time without significant increase in accuracy.

7.5 Analysis and Discussion of Results

When a thermoplastic material is first injected into an injection mould the cavity wall will start to increase in temperature, whilst the plastic decreases in temperature. At this stage the mould is gaining heat lost by the plastic. At some stage after this the plastic and cavity wall will reach the same temperature. As further cooling of the plastic occurs, heat will be extracted from the mould core until the plastic is ejected. This expected behaviour of mould temperatures is verified by the experimental data in figures 7.4 and 7.5.

It can also be noted that the cyclic behaviour of the mould temperatures, shown by figures 7.4 and 7.5, is slightly increasing with time. This is due to the fact that the steady-state cyclic temperature profile had not quite been reached and the temperature profile was still converging on a higher value. This was due to the fact that the data logging software was only capable of logging 500 readings (250 seconds). This problem resulted in the recorded cycle-average temperatures being slightly less than they should have been.

These temperature profiles will vary with time until they reach a steady state cyclic behaviour, as shown by figures 7.6 and 7.7. The steady-state cycle-averaged temperature can then be evaluated as the mean value taken from figures 7.6 and 7.7.

It can be seen from tables 7.2 and 7.3 that the cycle-averaged approach, as detailed in chapter 3 of this thesis, is a very accurate method of predicting the temperature profile throughout a mould core, the maximum error being 6%.

It is also worth noticing that the error closest to the sprue was greater than that further from the sprue. This was due to the inability of the two-dimensional section taken to incorporate the entire cooling system as well as the cavities. In the case of the present analysis a cooling line ran just behind the cavities and sprue, shown in figure 7.8 by the dotted lines, and hence cooled the sprue lower than was predicted by the computer model. This also explains the lack of symmetry in the results, that is, why the temperatures at points '2' and '4', for example, are not equal.

The test was completed for two different thermoplastics, low-density polyethylene and polypropylene. In the case of polypropylene, the melt temperature is higher, but the thermal properties are lower, table 5.2, and hence the temperature profile for polypropylene is slightly lower than that of polyethylene.

Once this temperature profile has been predicted the efficiency of the cooling system can be determined. The conclusion given by 'MouldCOOL' for the test mould is as follows:

Total Number of Cooling Lines: 2
Total heat loss, to atmosphere, is 2.38 [W]
Total heat flow extracted from cavity is 80.48 [W]

CONCLUSION:

The efficiency, of the cooling system is, 97 %

The efficiency can be increased by adding cooling lines or by increasing the cooling line diameters.

This must be done with the following rules kept in mind;

- Cooling lines should be kept at least 2-3 times the diameter away from the cavity.
- Cooling lines should be kept at least 2-3 times the diameter away from each other.
- Increasing the flow of coolant should increase the heat transfer from the cavity.
- Increasing the flow of coolant, past a critical value, will have little effect on heat transfer.
- It is important to be able to estimate this critical value

The cooling time used for the analysis: 10.00 Seconds

It is obvious from these figures that the efficiency of this particular mould is very high, since very little heat is lost to the environment. It is, however, necessary to highlight the factors that influence the efficiency so that it may be increased. The first factor is the mass flow-rate of coolant. Equation 3.10 shows how the heat transfer coefficient can be determined for the flow of coolant. Since the coolant extracts the heat from the cavity it is important to try to increase the heat transfer coefficient, which can be done by increasing the mass flow. However, since there is only a finite amount of heat that can be extracted from the cavity, there must be a limit to the heat transfer to the cooling lines. Hence, increasing the mass flow of coolant past some critical amount will have no effect on efficiency. In order to prove this, the test, as described above, was repeated for a number of different coolant flow rates.

The results of this, as shown in figure 7.13, show that after a flow rate of about 0.5 kg/s, the heat loss and hence efficiency, will be little affected by any further increase in flow rate.

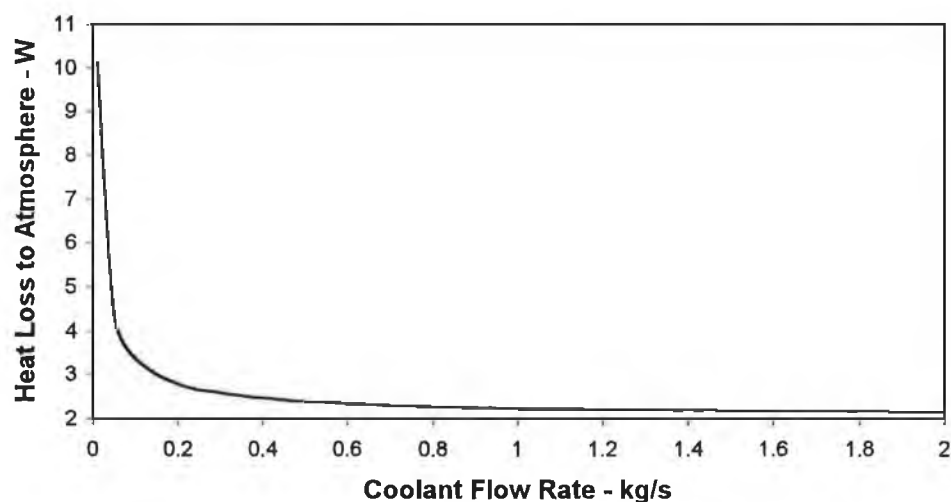


Figure 7.13 – Heat Loss V's Coolant Flow Rate

Another factor that determines the efficiency of the cooling system is the temperature of the coolant. By decreasing the temperature of the coolant the heat transfer from the cavity should be increased. To see the effect of decreasing the coolant temperature the analysis was repeated for a number of temperatures. The results of these analyses showed that the relationship between coolant temperature and heat loss to the

environment was linear, as shown in figure 7.14. Therefore, probably the best method of reducing heat lost to the atmosphere is to decrease the coolant flow rate using chillers.

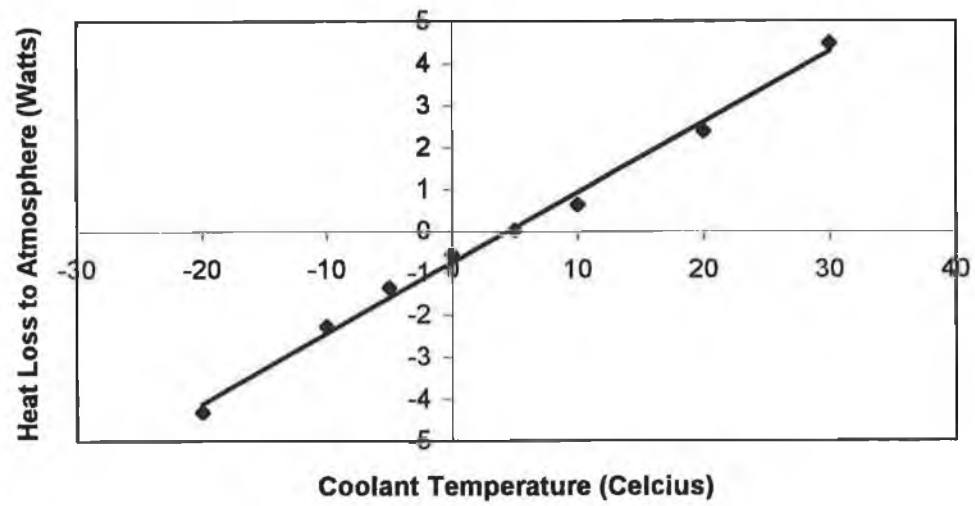


Figure 7.14 – Heat Loss V's Coolant Temperature

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDY

The application of computer simulation techniques to engineering problems is one of the greatest achievements of this century. With the power of modern day computers, software can be developed to simulate almost any process, making the design and manufacture of components simple and cost effective.

In this study, software has been developed to calculate the heat losses from injection moulds and hence calculate the efficiency of the cooling system employed. The software utilises the capabilities of a CAD system, "AutoCAD", to supply a two-dimensional section of the mould geometry. The analysis system, as a result, gains in the following ways.

- There is no need to develop a geometry processor for the system, which would take considerable time and effort.
- The CAD system employed has excellent capabilities for generating geometry quickly and simply and has the advantage of zooming and panning.
- The CAD system is well known and can be used by a large number of people.

In this study the numerical methods available to conduct the simulation of injection mould cooling systems are described and compared. One such method, which has gained widespread popularity, especially in the injection moulding industry, is the "Boundary Element Method (BEM)". This method has been shown to have the following advantages over other methods, such as the "Finite Element Method (FEM)".

- Data preparation is small compared to other methods, since only the boundary of the object is used.
- The mesh associated with the BEM does not have to be changed every time a cooling line is changed or added, since there are no internal elements.
- The BEM only calculates temperatures at internal points, if required by the user, and not as a necessity, as is the case with the FEM.

- The method has been shown to be, in general, more accurate, since less numerical approximation is being carried out.

The software was developed based on the BEM and resulted in a system with the following main features.

- The system was developed for use on a PC and is fully 32 Bit, running on Windows operating systems.
- The system utilises AutoCAD Release 14 for the mould geometry using Active X automation.
- The program was written in Visual Basic and utilises a number of user-friendly methods for input and output operations.
- The analysis programs were written in FORTRAN 90.
- The analysis allows the cycle-averaged temperature and heat flux profiles throughout the mould core to be predicted. These profiles can be used to estimate the efficiency of the cooling system.
- Cooling lines can be added, deleted or moved.
- A quick menu is incorporated. This allows the user to repeat the analysis with different values of certain variables, for example, mass flow rate of coolant.

In order to show the usefulness of the system a test mould was designed, manufactured and tested. The results of this test are shown in chapter 7 of this thesis. It can be seen from these results, table 7.2 and table 7.3, that the temperature predictions were very accurate, with the maximum error being 6%.

After predicting the temperature profiles, tests were conducted on the results, using the software developed, to show the effects of coolant flow rate and coolant temperature on cooling system efficiency. The conclusions drawn from these tests were as follows.

- The flow of coolant should be increased until turbulent flow occurs. Increasing the flow rate over this will have little effect on the efficiency of the cooling system.
- The efficiency will increase linearly with decrease in coolant temperature. Where possible chillers should be used to decrease the temperature of the coolant.

The software developed has been shown to have great usefulness in optimising the cooling system of injection moulds, but does have certain limitations.

8.1 System Limitations

The main limitations of the software are as follows:

- The analysis is two-dimensional, and only analyses a section of a mould.
- The software uses a cycle-averaged approach and does not take the phase change of the plastic into account.
- The system will only allow circular cooling lines to be incorporated.
- The software allocates storage to all variables in memory and not on disk, using a database system. This means that system resources limit the maximum number of elements that can be used, and hence a limit of 500 was set within the code.

8.2 Recommendations for Further Study

The system developed and described in this thesis is based on a two-dimensional analysis routine. This means that the geometry supplied by the CAD system is made up of lines and arcs and the mesh is made up of lines.

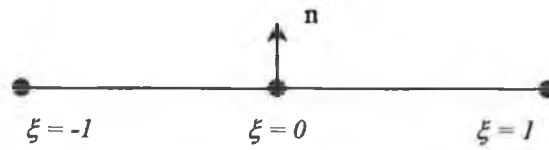
In many cases the mould in question will have many cavities and cooling circuits, and a two-dimensional section will not give a good picture of the mould. In this case, the system would have to be extended to solve three-dimensional problems. The steps involved in this would include:

- A new link would be set for the CAD system to supply a solid model of the geometry.
- The mesh base would have to deal with three-dimensional elements. This includes the discretisation of a three dimensional model being incorporated into the system.
- The analysis routines would have to be re-written for three-dimensional problems.

APPENDIX 1

ONE-DIMENSIONAL GAUSS QUADRATURE

Gauss Quadrature weights.



$$I = \int_{-1}^{+1} f(\xi) d\xi = \sum_{i=1}^{ni} w_i f(\xi_i)$$

NI = 4

$$\xi_i(1) = 0.861136311594053$$

$$\xi_i(2) = -0.861136311594053$$

$$\xi_i(3) = 0.339981043584856$$

$$\xi_i(4) = -0.339981043584856$$

$$w_i(1) = 0.347854845137454$$

$$w_i(2) = 0.347854845137454$$

$$w_i(3) = 0.652145154845137$$

$$w_i(4) = 0.652145154845137$$

NI = 6

$$\xi_i(1) = 0.932469514203152$$

$$\xi_i(2) = -0.932469514203152$$

$$\xi_i(3) = 0.661209386466265$$

$$\xi_i(4) = -0.661209386466265$$

$$\xi_i(5) = 0.238619186083197$$

$$\xi_i(6) = -0.238619186083197$$

$$w_i(1) = 0.17132449237917$$

$$w_i(2) = 0.17132449237917$$

$$w_i(3) = 0.360761573048139$$

$$w_i(4) = 0.360761573048139$$

$$w_i(5) = 0.467913934572691$$

$$w_i(6) = 0.467913934572691$$

NI = 8

$$\xi_i(1) = 0.960289856497536$$

$$\xi_i(2) = -0.960289856497536$$

$$\xi_i(3) = 0.796666477413627$$

$$\xi_i(4) = -0.796666477413627$$

$$\xi_i(5) = 0.52553241$$

$$\xi_i(6) = -0.52553241$$

$$\xi_i(7) = 0.1834346425$$

$$\xi_i(8) = -0.1834346425$$

$$w_i(1) = 0.10122854$$

$$w_i(2) = 0.10122854$$

$$w_i(3) = 0.222381$$

$$w_i(4) = 0.222381$$

$$w_i(5) = 0.31370665$$

$$w_i(6) = 0.31370665$$

$$w_i(7) = 0.3626837834$$

$$w_i(8) = 0.3626837834$$

$$NI = 10$$

$$\xi_i(1) = 0.97391$$

$$\xi_i(2) = -0.97391$$

$$\xi_i(3) = 0.86506337$$

$$\xi_i(4) = -0.86506337$$

$$\xi_i(5) = 0.67941$$

$$\xi_i(6) = -0.67941$$

$$\xi_i(7) = 0.4334$$

$$\xi_i(8) = -0.4334$$

$$\xi_i(9) = 0.14887434$$

$$\xi_i(10) = -0.14887434$$

$$w_i(1) = 0.06667134431$$

$$w_i(2) = 0.06667134431$$

$$w_i(3) = 0.1495$$

$$w_i(4) = 0.1495$$

$$w_i(5) = 0.2191$$

$$w_i(6) = 0.2191$$

$$w_i(7) = 0.26927$$

$$w_i(8) = 0.26927$$

$$w_i(9) = 0.295524225$$

$$w_i(10) = 0.295524225$$

$$NI = 12$$

$$\xi_i(1) = 0.98156063425$$

$$\xi_i(2) = -0.97391$$

$$\xi_i(3) = 0.90411726$$

$$\xi_i(4) = -0.86506337$$

$$\xi_i(5) = 0.7699026742$$

$$\xi_i(6) = -0.67941$$

$$\xi_i(7) = 0.58731795428662$$

$$\xi_i(8) = -0.4334$$

$$\xi_i(9) = 0.3678315$$

$$\xi_i(10) = -0.14887434$$

$$\xi_i(11) = 0.1252334085115$$

$$\xi_i(12) = -0.14887434$$

$$w_i(1) = 0.04717533639$$

$$w_i(2) = 0.04717533639$$

$$w_i(3) = 0.107$$

$$w_i(4) = 0.107$$

$$w_i(5) = 0.1601$$

$$w_i(6) = 0.1601$$

$$w_i(7) = 0.20316743$$

$$w_i(8) = 0.20316743$$

$$w_i(9) = 0.2335$$

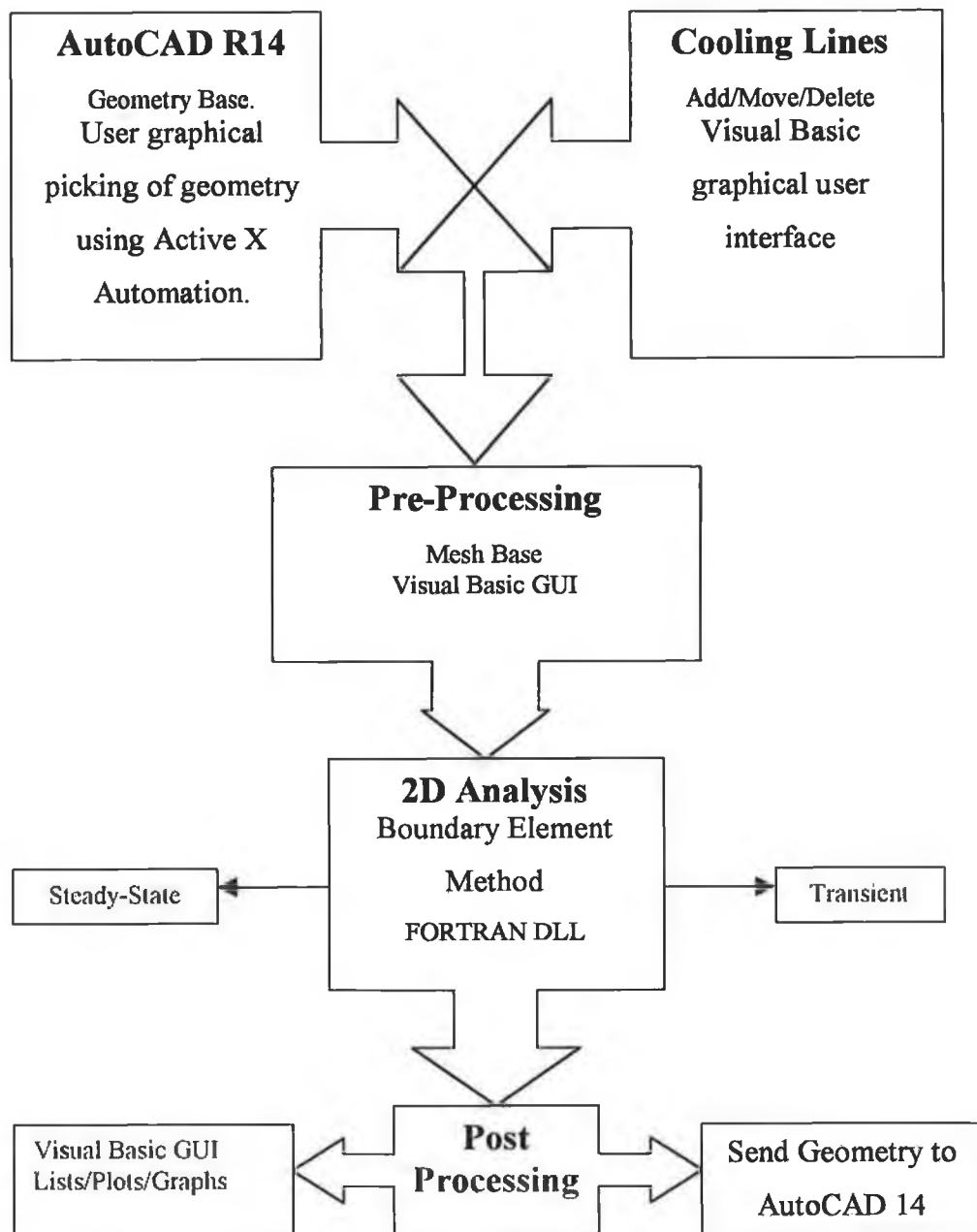
$$w_i(10) = 0.2335$$

$$w_i(11) = 0.25$$

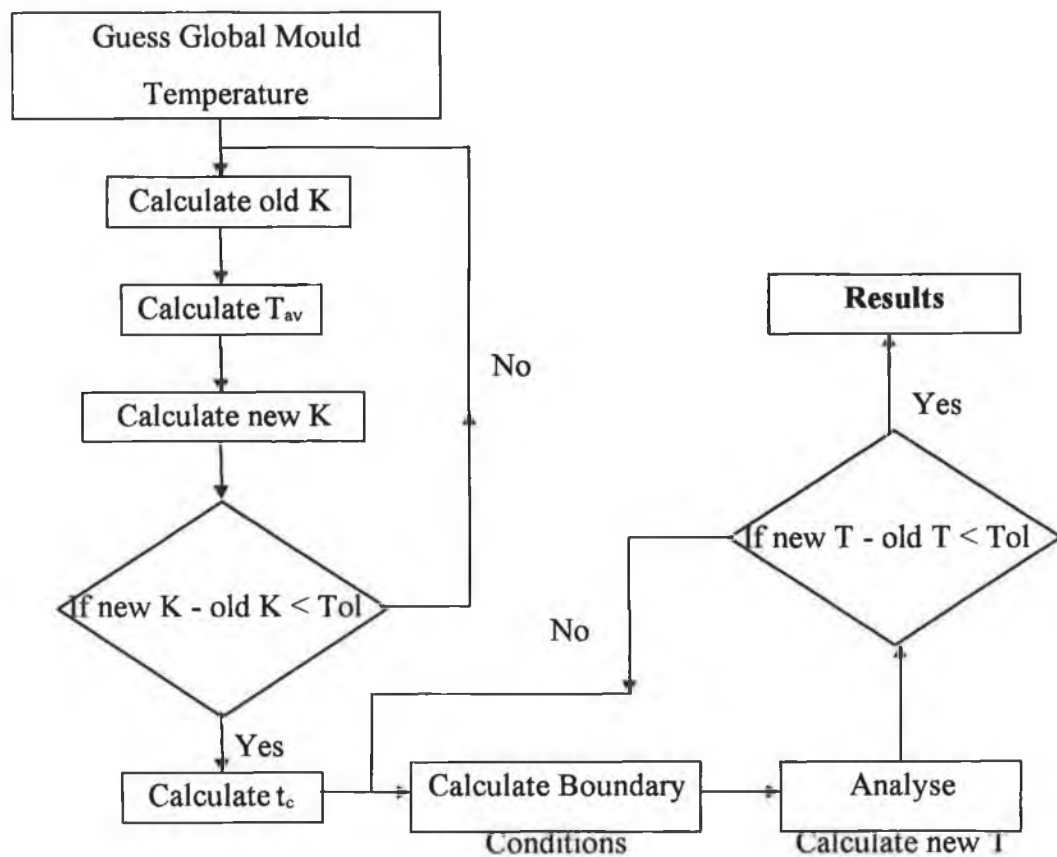
$$w_i(12) = 0.25$$

APPENDIX 2

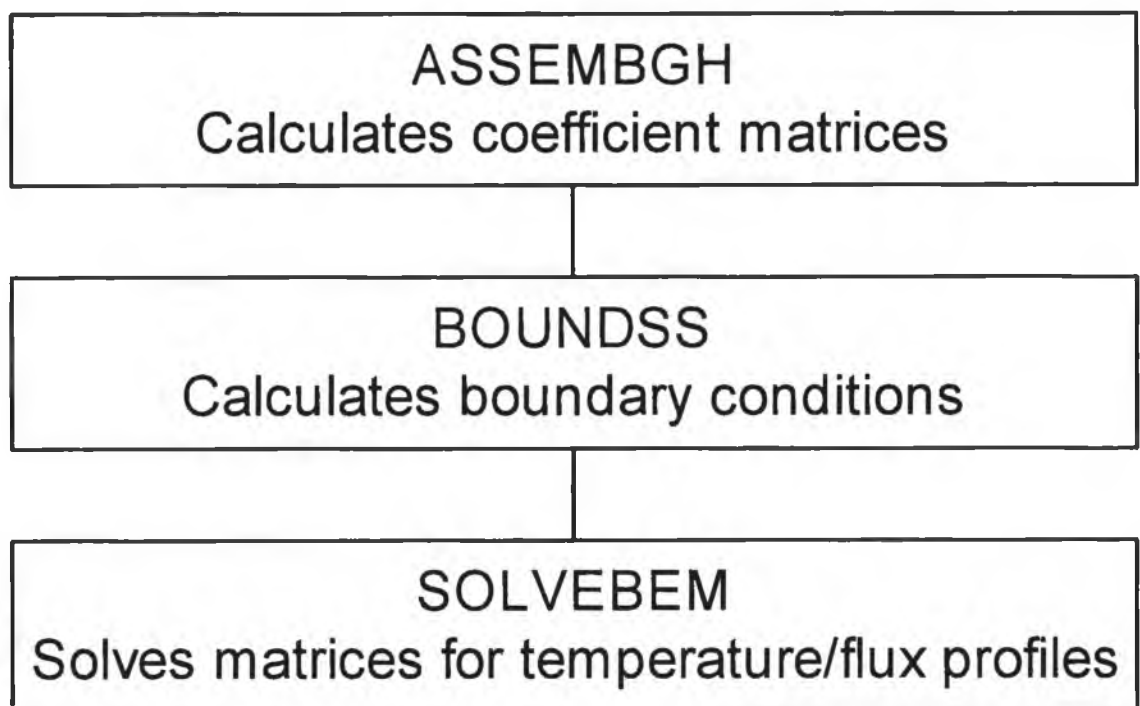
PROGRAM LISTINGS



MouldCOOL Program Structure



Injection Mould Analysis Algorithm



Steady-State Cycle-Averaged Analysis

***FORTTRAN 90 Subroutines for boundary element analysis of thermal problems.
Subroutines are compiled as a DLL for use within other programming environments
such as Visual Basic.***

Note : All of the following programs are 32-bit

SUBROUTINE ASSEMBGH(DOMAIN,ELEMCON,ELEML,X,Y,G,H)

***! SUBROUTINE FOR CONSTRUCTING COEFFICIENT MATRICES G AND H
! FOR USE WITH STEADY STATE OR TRANSIENT BOUNDARY ELEMENT TECHNIQUE
! WRITTEN : NIAL MORAN***

**!ms\$if .not. defined(LINKDIRECT)
!ms\$attributes dllexport :: ASSEMBGH
!ms\$endif**

**REAL*4 DOMAIN(4)
REAL*4 ELEMCON(300,2),ELEML(300),X(300),Y(300),G(300,600),H(300,300)
REAL*4 EI(4),WI(4),PI,R,LJ**

EXTERNAL WRRRN

INTEGER NN,NE,L

**DATA EI/0.86113631,-0.86113631,0.33998104,-0.33998104/
DATA WI/0.34785485,0.34785485,0.65214515,0.65214515/**

**NN = DOMAIN(1)
NE = DOMAIN(2)
L = DOMAIN(3)**

**PI = 3.141592654
DO J1 = 1,NN+L**

**XI = X(J1)
 YI = Y(J1)
 DO J = 1,NN
 H(J1,J)=0.
 END DO
 CC = 0.**

DO J2 = 1,NE

```

LJ = ELEML(J2)
N1 = ELEMCON(J2,1)
N2 = ELEMCON(J2,2)
X1 = X(N1)
X2 = X(N2)
Y1 = Y(N1)
Y2 = Y(N2)
H1 = 0.
H2 = 0.
G1 = 0.
G2 = 0.
      DO J3 = 1,4
        E = EI(J3)
        W = WI(J3)
        XX = X1 + (1.+E)*(X2-X1)/2.
        YY = Y1 + (1.+E)*(Y2-Y1)/2.
        R = SQRT((XI-XX)**2 + (YI-YY)**2)
        PP = ((XI-XX)*(Y1-Y2)+(YI-YY)*(X2-X1))/(R*R*4.*PI)*W
        H1 = H1 + (1.-E)*PP/2.0
        H2 = H2 + (1.+E)*PP/2.0
        PP = LOG(1./R)/(4.*PI)*LJ*W
        G1 = G1 + (1.-E)*PP/2.
        G2 = G2 + (1.+E)*PP/2.
      END DO
CC = CC - H1 - H2
GE = LJ*(3./2.-LOG(REAL(LJ)))/(4.*PI)
IF(N1.EQ.J1) G1=GE
IF(N2.EQ.J1) G2=GE
H(J1,N1)=H(J1,N1)+H1
H(J1,N2)=H(J1,N2)+H2
G(J1,2*J2-1) = G1
G(J1,2*J2) = G2
      END DO
H(J1,J1) = CC
      END DO
END SUBROUTINE ASSEMBGH

```

SUBROUTINE BOUNDSS(DOMAIN,KODE,G,H,A,B,HC,TA,ELEMCON,U,Q)

*! SUBROUTINE FOR APPLYING STEADY STATE BOUNDARY CONDITIONS
! FOR USE WITH STEADY STATE BOUNDARY ELEMENT TECHNIQUE
! WRITTEN : NIAL MORAN*

```
!ms$if .not. defined(LINKDIRECT)
!ms$attributes dllexport :: BOUNDSS
!ms$endif
```

```
REAL*4 DOMAIN(3),G(300,200),H(300,300),A(300,300),B(300),KODE(300)
REAL*4 HC(300),TA(300),U(300),Q(200),ELEMCON(300,2)
```

```
NN = DOMAIN(1)
NE = DOMAIN(2)
L = DOMAIN(3)
```

```
DO I = 1,NN+L
  B(I) = 0.
```

```
    DO J = 1,NN+L
      A(I,J) = 0.
      KK=KODE(J)
      IF(KK.EQ.0.OR.KK.EQ.2)THEN
        A(I,J) = H(I,J)
      ELSEIF (KK.EQ.1) THEN
        B(I) = B(I) - H(I,J)*U(J)
      Elself (KK.EQ.3) Then
        A(I, J) = H(I, J)
      END IF
    END DO
```

```
    DO J = 1,NE
      N1 = ELEMCON(J,1)
      N2 = ELEMCON(J,2)
      KK = KODE(N1)
      IF(KK.EQ.0.OR.KK.EQ.2) THEN
        B(I) = B(I) + G(I,2*J-1)*Q(N1)
      ELSEIF (KK.EQ.1) THEN
        A(I,N1) = A(I,N1) - G(I,2*J-1)
      Elself (KK.EQ.3) Then
        A(I, N1) = A(I, N1) - G(I, 2 * J - 1) * HC(N1)
        B(I) = B(I) - G(I, 2 * J - 1) * HC(N1) * TA(N1)
      END IF
      KK = KODE(N2)
      IF(KK.EQ.0.OR.KK.EQ.2) THEN
        B(I) = B(I) + G(I,2*J)*Q(N2)
```

```

        ELSEIF (KK.EQ.1) THEN
        A(I,N2) = A(I,N2) - G(I,2*J)
        Elself (KK.EQ.3) Then
        A(I, N2) = A(I, N2) - G(I, 2 * J) * HC(N2)
        B(I) = B(I) - G(I, 2 * J) * HC(N2) * TA(N2)
        END IF
    END DO

END DO

END SUBROUTINE BOUNDSS

SUBROUTINE SOLVEBEM(DOMAIN,KODE,U,Q,HC,TA,A,B)

! SUBROUTINE FOR SOLVING BOUNDARY ELEMENT SYSTEM OF EQUATIONS
! FOR USE WITH STEADY STATE OR TRANSIENT BOUNDARY ELEMENT TECHNIQUE
! WRITTEN : NIAL MORAN

!ms$if .not. defined(LINKDIRECT)
!ms$attributes dllexport :: SOLVEBEM
!ms$endif

REAL*4 DOMAIN(3),KODE(300),U(300),Q(200),HC(300),TA(300),A(300,300),B(300)
DIMENSION XX(300)

NN = DOMAIN(1)
NE = DOMAIN(2)
L = DOMAIN(3)

CALL LSLRG(NN+L,A,300,B,1,XX)

DO J1=1,NN+L
KK=KODE(J1)
    IF(KK.EQ.0.OR.KK.EQ.2)THEN
        U(J1) = XX(J1)
    ELSEIF (KK.EQ.1) THEN
        Q(J1) = XX(J1)
    ELSEIF (KK.EQ.3) THEN
        U(J1) = XX(J1)
        Q(J1) = HC(J1)*(U(J1)-TA(J1))
    END IF
END DO
END DO

```


END SUBROUTINE SOLVEBEM

SUBROUTINE RHSMATRIX(DOMAIN,ELEMCON,X,Y,ELEML,S,FINV,H,G)

! SUBROUTINE FOR MATRIX S

! FOR USE WITH TRANSIENT BOUNDARY ELEMENT TECHNIQUE

! WRITTEN : NIAL MORAN

!ms\$if .not. defined(LINKDIRECT)

!ms\$attributes dllexport :: RHSMATRIX

!ms\$endif

REAL*4 DOMAIN(3),ELEMCON(300,2),X(300),Y(300),H(300,300),G(300,200)

REAL*4 ELEML(300),S(300,300),FINV(300,300),HAT(300,300)

NN = DOMAIN(1)

NE = DOMAIN(2)

L = DOMAIN(3)

DO J1=1,NN+L

DO J2 = 1,2*NE

HAT(J2,J1) = 0.

END DO

END DO

DO J1 = 1,NE

N1 = ELEMCON(J1,1)

N2 = ELEMCON(J1,2)

X1 = X(N1)

X2 = X(N2)

Y1 = Y(N1)

Y2 = Y(N2)

DD = ELEML(J1)

JP1 = 2*J1 - 1

JP2 = 2*J1

DO J2 = 1,NN+L

XP = X(J2)

YP = Y(J2)

R1 = SQRT((X1-XP)**2 + (Y1-YP)**2)

R2 = SQRT((X2-XP)**2 + (Y2-YP)**2)

QHAT1 = (0.5+R1/3)*((X1-XP)*(Y1-Y2)+(Y1-YP)*(X2-X1))/DD

```

      QHAT2 = (0.5+R2/3)*((X2-XP)*(Y1-Y2)+(Y2-YP)*(X2-X1))/DD
      HAT(JP1,J2) = QHAT1
      HAT(JP2,J2) = QHAT2
    END DO
  END DO

  DO J1 = 1,NN+L
    DO J2 = 1,NN+L
      S(J1,J2)=0.
      DO J3 = 1,2*NE
        S(J1,J2) = S(J1,J2) + G(J1,J3)*HAT(J3,J2)
      END DO
    END DO
  END DO

  DO J1=1,NN+L
    XI=X(J1)
    YI=Y(J1)
    DO J2=1,NN+L
      HAT(J1,J2)=0.
      XP = X(J2)
      YP = Y(J2)
      R=SQRT((XI-XP)**2+(YI-YP)**2)
      UHAT = (R**3)/9.+(R**2)/4
      HAT(J1,J2)=UHAT
    END DO
  END DO

  DO J1=1,NN+L
    DO J2=1,NN+L
      DO J3=1,NN+L
        S(J1,J2)=S(J1,J2)-H(J1,J3)*HAT(J3,J2)
      END DO
    END DO
  END DO

  DO I=1,NN+L
    DO J=1,NN+L
      HAT(I,J)=-S(I,J)
    END DO
  END DO

```

```

DO I = 1,NN+L
  DO J=1,NN+L
    S(I,J)=0.
    DO J1=1,NN+L
      S(I,J) = S(I,J)+HAT(I,J1)*FINV(J1,J)
    END DO
  END DO
END DO

```

END SUBROUTINE RHSMATRIX

SUBROUTINE INVERSEF(DOMAIN,FINV,X,Y)

! SUBROUTINE FOR CONSTRUCTING COEFFICIENT MATRICES F AND INVERSE F
! FOR USE WITH TRANSIENT BOUNDARY ELEMENT TECHNIQUE
! WRITTEN : NIAL MORAN

```

!ms$if .not. defined(LINKDIRECT)
!ms$attributes dllexport :: INVERSEF
!ms$endif

```

```

REAL*4 DOMAIN(3),FINV(300,300),F(300,300),X(300),Y(300)

```

```

NN = DOMAIN(1)
NE = DOMAIN(2)
L = DOMAIN(3)

```

```

DO I = 1,NN+L
  XI = X(I)
  YI = Y(I)
  DO J = 1,NN+L
    XJ = X(J)
    YJ = Y(J)
    R = SQRT((XI-XJ)**2 + (YI-YJ)**2)
    F(I,J) = 1 + R
  END DO
END DO

```

```

CALL LINRG(NN+L,F,300,FINV,300)

```

END SUBROUTINE INVERSEF

SUBROUTINE RHSVEC(MATPROP,DOMAIN,S,H,UP,QP,XX,THEU,THEQ)

*! SUBROUTINE FOR CONSTRUCTING RHS VECTOR OF DUAL RECIPROCITY
! FOR USE WITH TRANSIENT BOUNDARY ELEMENT TECHNIQUE
! WRITTEN : NIAL MORAN*

*!ms\$if .not. defined(LINKDIRECT)
!ms\$attributes dllexport :: RHSVEC
!ms\$endif*

REAL*4 DOMAIN(3),MATPROP(5),S(300,300),H(300,300),UP(300),QP(300),XX(300)
REAL*4 CK,DT,NUMTS,TI,KM,THEU,THEQ

NN = DOMAIN(1)
NE = DOMAIN(2)
L = DOMAIN(3)
NUMTS = MATPROP(1)
DT = MATPROP(2)
TI = MATPROP(3)
CK = MATPROP(4)
KM = MATPROP(5)

CT = -1./(CK*DT)
DO I = 1,NN+L
XX(I) = 0.
DO J = 1,NN+L
XX(I) = XX(I) + ((CT*S(I,J)-((1-THEU)*H(I,J))*UP(J))+((1-THEQ)*QP(J))
END DO
END DO

END SUBROUTINE RHSVEC

SUBROUTINE

BOUNDTR(MATPROP,DOMAIN,S,H,G,KODE,ELEMCON,A,B,HC,TA,U,Q,THEU,THEQ)

*! SUBROUTINE FOR APPLYING TRANSIENT BOUNDARY CONDITIONS
! FOR USE WITH TRANSIENT BOUNDARY ELEMENT TECHNIQUE
! WRITTEN : NIAL MORAN*

*!ms\$if .not. defined(LINKDIRECT)
!ms\$attributes dllexport :: BOUNDTR
!ms\$endif*

```

REAL*4 DOMAIN(3),MATPROP(5),S(300,300),H(300,300),G(300,200)
REAL*4 KODE(300),ELEMCON(300,2),A(300,300),B(300),HC(300),TA(300),U(300),Q(200)
REAL*4 CK,DT,NUMTS,TI,KM

```

```

NN = DOMAIN(1)
NE = DOMAIN(2)
L = DOMAIN(3)
NUMTS = MATPROP(1)
DT = MATPROP(2)
TI = MATPROP(3)
CK = MATPROP(4)
KM = MATPROP(5)
CT = -1./(CK*DT)
DO I = 1,NN+L
    DO J = 1,NN+L
        A(I,J) = 0.
        KK=KODE(J)
        IF(KK.EQ.0.OR.KK.EQ.2)THEN
            A(I,J) = CT*S(I,J)+THEU*H(I,J)
        ELSEIF (KK.EQ.1) THEN
            B(I) = B(I) - (CT*S(I,J)+THEU*H(I,J))*U(J)
        Elself (KK.EQ.3) Then
            A(I, J) = CT * S(I, J) + THEU*H(I, J)
        END IF
    END DO
    DO J = 1,NE
        N1 = ELEMCON(J,1)
        N2 = ELEMCON(J,2)
        KK = KODE(N1)
        IF(KK.EQ.0.OR.KK.EQ.2) THEN
            B(I) = B(I) + THEQ*G(I,2*J-1)*Q(N1)
        ELSEIF (KK.EQ.1) THEN
            A(I,N1) = A(I,N1) - THEQ*G(I,2*J-1)
        Elself (KK.EQ.3) Then
            A(I, N1) = A(I, N1) - THEQ* G(I, 2 * J - 1) * HC(N1)
            B(I) = B(I) - THEQ* G(I, 2 * J - 1) * HC(N1) * TA(N1)
        END IF
        KK = KODE(N2)
        IF(KK.EQ.0.OR.KK.EQ.2) THEN
            B(I) = B(I) + THEQ*G(I,2*J)*Q(N2)
        ELSEIF (KK.EQ.1) THEN

```

$A(I, N2) = A(I, N2) - THEQ * G(I, 2 * J)$

Elseif (KK.EQ.3) Then

$A(I, N2) = A(I, N2) - THEQ * G(I, 2 * J) * HC(N2)$

$B(I) = B(I) - THEQ * G(I, 2 * J) * HC(N2) * TA(N2)$

END IF

END DO

END DO

END SUBROUTINE BOUNDTR

Visual basic declarations for initialising the subroutines from within the DLL.

Declarations are global.

Declare Sub assembgh Lib "BEM.DLL" Alias "_ASSEMBGH@28" (ByRef domain As Single, ByRef CON As Single, ByRef LE As Single, ByRef X As Single, ByRef Y As Single, ByRef G As Single, ByRef H As Single)

Declare Sub BOUNDSS Lib "BEM.DLL" Alias "_BOUNDSS@44" (domain As Single, KODE As Single, G As Single, H As Single, A As Single, B As Single, HC As Single, TA As Single, CON As Single, U As Single, Q As Single)

Declare Sub SOLVEBEM Lib "BEM.DLL" Alias "_SOLVEBEM@32" (domain As Single, KODE As Single, U As Single, Q As Single, HC As Single, TA As Single, A As Single, B As Single)

Declare Sub RHSMATRIX Lib "BEM.DLL" Alias "_RHSMATRIX@36" (domain As Single, CON As Single, X As Single, Y As Single, LE As Single, S As Single, FINV As Single, H As Single, G As Single)

Declare Sub INVERSEF Lib "BEM.DLL" Alias "_INVERSEF@16" (domain As Single, FINV As Single, X As Single, Y As Single)

Declare Sub RHSVEC Lib "BEM.DLL" Alias "_RHSVEC@36" (MATPROP As Single, domain As Single, S As Single, H As Single, UP As Single, QP As Single, XY As Single, THETAU As Single, THETAQ As Single)

Declare Sub BOUNDTR Lib "BEM.DLL" Alias "_BOUNDTR@60" (MATPROP As Single, domain As Single, S As Single, H As Single, G As Single, KODE As Single, CON As Single, A As Single, B As Single, HC As Single, TA As Single, U As Single, Q As Single, THETAU As Single, THETAQ As Single)

Visual Basic program for opening MouldCOOL files.

```
Private Sub filemnuopen_Click()
```

```
On Error GoTo openerr:
```

```
frmmdi.MsgBar.Caption = "Loading File, Please Wait!"
```

```
frmmain.MousePointer = 11
```

```
Screen.MousePointer = 11
```

```
Me.Refresh
```

```
CommonDialog2.InitDir = App.Path
```

```
CommonDialog2.Action = 1
```

```
Myf = CommonDialog2.filename
```

```
MyFile = Myf
```

```
frmmdi.Caption = MyCap + " <" + CommonDialog2.FileTitle + ">"
```

```
If MyFile <> "" Then
```

```
st = "File <" + MyFile + "> opened at "
```

```
AddText st
```

```
End If
```

```
Open Myf For Input As #1
```

```
Nn = 0
```

```
Ne = 0
```

```
L = 0
```

```
Input #1, GeometryFlag, geoflag, BcFlag, SSFlag, AnalFlag, InitFlag
```

```
If GeometryFlag = 1 Then
```

```
Input #1, Nlines, Narcs
```

```
For i = 1 To Nlines
```

```
Input #1, MyLine(i).pt1.x, MyLine(i).pt1.y, MyLine(i).pt2.x, MyLine(i).pt2.y
```

```
Input #1, Ndivl(i)
```

```
Next i
```

```
For i = 1 To Narcs
```

```
Input #1, MyArc(i).pt1.x, MyArc(i).pt1.y, MyArc(i).pt2.x, _
```

```
MyArc(i).pt2.y, MyArc(i).Radius, MyArc(i).angle1, MyArc(i).angle2 _
```

```
, MyArc(i).Center.x, MyArc(i).Center.y
```

```
Input #1, Ndiva(i)
```

```
Next i
```

```
End If
```

```
If geoflag = 1 Then
```



```

Input #1, Nn, Ne, L, Ncool
For i = 1 To Nn + L
Input #1, x(i), y(i)
Next i
For i = 1 To Ne
Input #1, ExteriorFace(i), FL(i)
Input #1, con(i, 1), con(i, 2)
x1 = x(con(i, 1))
y1 = y(con(i, 1))
N2 = con(i, 2)
X2 = x(N2)
Y2 = y(con(i, 2))
le(i) = Sqr((X2 - x1) ^ 2 + (Y2 - y1) ^ 2)
Next i
For i = 1 To Ncool
Input #1, MyCool(i).Center.x, MyCool(i).Center.y, MyCool(i).Radius, MyCool(i).NElements
Next i
End If

```

nulls = 0

```

If BcFlag = 1 Then
For i = 1 To Ne
Input #1, KODE(i), val1, val2
If KODE(i) = 1 Then
U(i) = val1
Elseif KODE(i) = 2 Then
Q(i) = val1
Elseif KODE(i) = 3 Then
HC(i) = val1
TA(i) = val2
Elseif KODE(i) = 4 Then
HC(i) = val1
TA(i) = val2
Elseif KODE(i) = 5 Then
HC(i) = val1
TA(i) = val2
Elseif KODE(i) = 6 Then
HC(i) = val1
TA(i) = val2
End If
Next i

```

End If

Input #1, Km, CKM, KPO, BETA, ROP, CPP, TP, hm, AvMoldT, TempC, HA, NUMTS, DT,
THETAU, TI, TE, TAIR, DCav, DMould, TCC

If AnalFlag = 1 Then

 If SSFlag = 1 Or SSFlag = 2 Then

 For i = 1 To Nn + L

 Input #1, U(i), Q(i)

 Next i

 Else

 For i = 1 To Nn + L

 For j = 1 To NUMTS + 1

 Input #1, TEMPT(i, j), FLUXT(i, j)

 Next j

 Next i

 End If

End If

Close #1

frmmdi.MsgBar.Caption = "Ready ..."

frmmain.MousePointer = 0

Screen.MousePointer = 0

Exit Sub

openerr:

frmmdi.MsgBar.Caption = "Ready ..."

frmmain.MousePointer = 0

Screen.MousePointer = 0

Close #1

showerror

End Sub

Visual Basic program for saving MouldCOOL files.

Sub SaveFile(Myf)

Open Myf For Output As #1

Write #1, GeometryFlag, geoflag, BcFlag, SSFlag, AnalFlag, InitFlag

If GeometryFlag = 1 Then

Write #1, Nlines, Narcs

For i = 1 To Nlines

If Ndivl(i) = 0 Then Ndivl(i) = 10

Write #1, MyLine(i).pt1.x, MyLine(i).pt1.y, MyLine(i).pt2.x, MyLine(i).pt2.y

Write #1, Ndivl(i)

Next i

For i = 1 To Narcs

If Ndiva(i) = 0 Then Ndiva(i) = 10

Write #1, MyArc(i).pt1.x, MyArc(i).pt1.y, MyArc(i).pt2.x, _

MyArc(i).pt2.y, MyArc(i).Radius, MyArc(i).angle1, MyArc(i).angle2 _

, MyArc(i).Center.x, MyArc(i).Center.y

Write #1, Ndiva(i)

Next i

End If

If geoflag = 1 Then

Write #1, Nn, Ne, L, Ncool

For i = 1 To Nn + L

Write #1, x(i), y(i)

Next i

For i = 1 To Ne

Write #1, ExteriorFace(i), FL(i)

Write #1, con(i, 1), con(i, 2)

Next i

For i = 1 To Ncool

Write #1, MyCool(i).Center.x, MyCool(i).Center.y, MyCool(i).Radius, MyCool(i).NElements

Next i

End If

nulls = 0

If BcFlag = 1 Then

For i = 1 To Ne

If KODE(i) = 1 Then

Write #1, KODE(i), U(i), nulls

Elseif KODE(i) = 2 Then

Write #1, KODE(i), Q(i), nulls

Elseif KODE(i) = 3 Then

Write #1, KODE(i), HC(i), TA(i)

Elseif KODE(i) = 4 Then

```

    Write #1, KODE(i), HC(i), TA(i)
    Elself KODE(i) = 5 Then
    Write #1, KODE(i), HC(i), TA(i)
    Elself KODE(i) = 6 Then
    Write #1, KODE(i), HC(i), TA(i)
    End If
  Next i
End If
Write #1, Km, CKM, KPO, BETA, ROP, CPP, TP, hm, AvMoldT, TempC, HA, NUMTS, DT,
THETAU, TI, TE, TAir, DCav, DMould, TCC
If AnalFlag = 1 Then
  If SSFlag = 1 Or SSFlag = 2 Then
    For i = 1 To Nn + L
      Write #1, U(i), Q(i)
    Next i
  Else
    For i = 1 To Nn + L
      For j = 1 To NUMTS + 1
        Write #1, TEMPT(i, j), FLUXT(i, j)
      Next j
    Next i
  End If
End If
Close #1

End Sub

```

Visual Basic program for drawing the contents of the geometrey base to a picture box called MyPic.

```

On Error Resume Next
MyPic.Cls
MyPic.Refresh

```

```

minx = 9.99E+101
miny = 9.99E+101
maxx = 0
maxy = 0

```

```

For i = 1 To Nlines
x1 = MyLine(i).pt1.x
y1 = MyLine(i).pt1.y
X2 = MyLine(i).pt2.x
Y2 = MyLine(i).pt2.y
    If x1 > maxx Then maxx = x1
    If x1 < minx Then minx = x1
    If y1 > maxy Then maxy = y1
    If y1 < miny Then miny = y1
    If X2 > maxx Then maxx = X2
    If X2 < minx Then minx = X2
    If Y2 > maxy Then maxy = Y2
    If Y2 < miny Then miny = Y2
Next i

```

```

For i = 1 To Narcs
    t1 = MyArc(i).angle1
    t2 = MyArc(i).angle2
    If t2 > t1 Then
        TH = ((t2 - t1) / 32)
        For j = 1 To 32
            th3 = (t1 + TH * (j - 1))
            x1 = MyArc(i).Center.x + MyArc(i).Radius * Cos(th3)
            y1 = MyArc(i).Center.y + MyArc(i).Radius * Sin(th3)
            If x1 > maxx Then maxx = x1
            If x1 < minx Then minx = x1
            If y1 > maxy Then maxy = y1
            If y1 < miny Then miny = y1
        Next j
    Else
        TH = ((t2 - t1 + (2 * pi)) / 32)
        t1 = t1 - (270 * pi / 180)
        For j = 1 To 32
            th3 = (t1 + TH * (j - 1))
            x1 = MyArc(i).Center.x + MyArc(i).Radius * Sin(th3)
            y1 = MyArc(i).Center.y - MyArc(i).Radius * Cos(th3)
            If x1 > maxx Then maxx = x1
            If x1 < minx Then minx = x1
            If y1 > maxy Then maxy = y1
            If y1 < miny Then miny = y1
        Next j
    End If

```

Next i

WX = maxx - minx

WY = maxy - miny

dx = WX / 20

dy = WY / 20

MyPic.ScaleLeft = 0

MyPic.ScaleTop = 0

MyPic.ScaleWidth = WX + 2 * dx

MyPic.ScaleHeight = WY + 2 * dy

If MyPic.ScaleWidth >= MyPic.ScaleHeight Then

 MyPic.Width = Me.ScaleWidth

 MyPic.Height = MyPic.Width * (MyPic.ScaleHeight / MyPic.ScaleWidth)

Else

 MyPic.Height = Me.ScaleHeight - MsgBox2.Height

 MyPic.Width = MyPic.Height * (MyPic.ScaleWidth / MyPic.ScaleHeight)

End If

MyPic.Left = (Me.ScaleWidth - MyPic.Width) / 2

MyPic.Top = (Me.ScaleHeight - MsgBox2.Height - MyPic.Height) / 2 + MsgBox2.Height

MyPic.Refresh

For i = 1 To Nlines

 x1 = MyLine(i).pt1.x - minx + dx

 y1 = MyPic.ScaleHeight - (MyLine(i).pt1.y - miny + dy)

 X2 = MyLine(i).pt2.x - minx + dx

 Y2 = MyPic.ScaleHeight - (MyLine(i).pt2.y - miny + dy)

 If MyIndex = i And MyEnt = "line" Then

 MyPic.Line (x1, y1)-(X2, Y2), RGB(0, 0, 255)

 Else

 MyPic.Line (x1, y1)-(X2, Y2)

 End If

Next i

```

For i = 1 To Narcs
x1 = MyArc(i).Center.x - minx + dx
y1 = MyPic.ScaleHeight - (MyArc(i).Center.y - miny + dy)
TH1 = MyArc(i).angle1
TH2 = MyArc(i).angle2
r = MyArc(i).Radius
  If MyIndex = i And MyEnt = "arc" Then
    MyPic.Circle (x1, y1), r, RGB(0, 0, 255), TH1, TH2
  Else
    MyPic.Circle (x1, y1), r, , TH1, TH2
  End If

Next i

For i = 1 To Ncool
  If MyIndex = i And MyEnt = "cool" Then
    MyPic.Circle (MyCool(i).Center.x - minx + dx, (MyPic.ScaleHeight - (MyCool(i).Center.y -
miny + dy))), MyCool(i).Radius, RGB(0, 0, 255)
  Else
    MyPic.Circle (MyCool(i).Center.x - minx + dx, (MyPic.ScaleHeight - (MyCool(i).Center.y -
miny + dy))), MyCool(i).Radius
  End If
Next i

```

Visual Basic program for displaying mesh on picture box

```

Sub Draws()
Picture1.Cls
Picture1.Refresh

```

```

If Ne > 0 Then

```

```

ReDim x1(Ne), y1(Ne), X2(Ne), Y2(Ne)

```

```

  minx = 1E+99
  miny = 1E+99
  maxx = 0
  maxy = 0
  For i = 1 To Ne
    If x(i) < minx Then minx = x(i)
    If y(i) < miny Then miny = y(i)
    If x(i) > maxx Then maxx = x(i)
    If y(i) > maxy Then maxy = y(i)
  
```

```

Next i
dx = maxx / 20
dy = maxy / 20
frmgraphics.Picture1.ScaleWidth = maxx + dx - minx
frmgraphics.Picture1.ScaleHeight = maxy + dy - miny
rad = (frmgraphics.Picture1.ScaleWidth + frmgraphics.Picture1.ScaleHeight) / 50
delx = dx / 2
dely = dy / 2
If PlotNodesFlag = 0 Then
    For i = 1 To Ne
        XVAL1 = x(con(i, 1)) + delx - minx
        XVAL2 = x(con(i, 2)) + delx - minx
        YVAL1 = frmgraphics.Picture1.ScaleHeight - y(con(i, 1)) - dely + miny
        YVAL2 = frmgraphics.Picture1.ScaleHeight - y(con(i, 2)) - dely + miny
        frmgraphics.Picture1.Line (XVAL1, YVAL1)-(XVAL2, YVAL2), RGB(0, 0, 0)
    Next i
End If
If nflag = 1 Then
    For i = 1 To Nn
        XVAL = x(i) + delx - minx
        YVAL = frmgraphics.Picture1.ScaleHeight - y(i) - dely + miny
        Picture1.FillColor = RGB(0, 0, 255)
        Picture1.FillStyle = 0
        frmgraphics.Picture1.Circle (XVAL, YVAL), rad, RGB(0, 0, 255)
    Next i
    For i = Nn + 1 To Nn + L
        XVAL1 = x(i) + delx - minx
        YVAL1 = frmgraphics.Picture1.ScaleHeight - y(i) - dely + miny
        Picture1.FillStyle = 0
        Picture1.FillColor = RGB(255, 0, 0)
        frmgraphics.Picture1.Circle (XVAL1, YVAL1), rad, RGB(255, 0, 0)
    Next i
End If
End If
End Sub

```


Visual Basic program for conducting a standard steady state analysis

On Error GoTo analerr

If geoflag = 0 Then

MsgBox "No Mesh Defined"

Exit Sub

End If

If BcFlag = 0 Then

MsgBox "No Boundary Conditions Defined"

Exit Sub

End If

If Km = 0 Then

msg = "There is a problem with input data - Thermal Conductivity"

msg = msg + "Check out Analysis -> Options Menu!"

MsgBox msg

Exit Sub

End If

Domain(1) = Nn: Domain(2) = Ne: Domain(3) = L: Domain(4) = Ng: Domain(5) = Km

AddText "Steady State Analysis initialised at "

Me.Refresh

frmmdi.MsgBar.Caption = "Analysing, Please Wait!"

Me.MousePointer = 11

Screen.MousePointer = 11

Time1 = Timer

MATPROP(1) = NUMTS: MATPROP(2) = DT: MATPROP(3) = TI: MATPROP(4) = CKM

Call assembgh(Domain(1), con(1, 1), le(1), x(1), y(1), G(1, 1), H(1, 1))

frmmdi.MsgBar.Caption = "Solving Equations, Please Wait!"

Call BOUNDSS(Domain(1), KODE(1), G(1, 1), H(1, 1), A(1, 1), B(1), HC(1), TA(1), con(1, 1),
U(1), Q(1))

Call SOLVEBEM(Domain(1), KODE(1), U(1), Q(1), HC(1), TA(1), A(1, 1), B(1))

AddText "Steady State Analysis completed at "

Time2 = Timer

CpuTime = Time2 - Time1

If CpuTime < 1 Then

msg = "Previous analysis took : " + Str\$(CpuTime * 1000) + " Milli-Seconds"

ElseIf CpuTime > 1 And CpuTime < 60 Then

msg = "Previous analysis took : " + Str\$(CpuTime) + " Seconds"

ElseIf CpuTime > 60 And CpuTime < 3600 Then

msg = "Previous analysis took : " + Str\$(CpuTime / 60) + " Minutes"

Else

msg = "Previous analysis took : " + Str\$(CpuTime / 3600) + " Hours"

End If

frmmain.txt = frmmain.txt + Chr\$(13) + Chr\$(10) + Chr\$(13) + Chr\$(10) + msg

AnalFlag = 1

Me.MousePointer = 0

Screen.MousePointer = 0

frmmdi.MsgBar.Caption = "Ready ..."

SSFlag = 1

Exit Sub

analerr:

SSPanel2.FloodType = 0

Me.MousePointer = 0

Screen.MousePointer = 0

frmmdi.MsgBar.Caption = "Ready ..."

'Me.Refresh

msg = "There was a problem with the analysis." + Chr\$(13)

msg = msg + "Please check all input variables and try again."

MsgBox msg

AddText "Steady State Analysis crashed at "

Close

Exit Sub

Visual Basic program for conducting a standard transient analysis

On Error GoTo analerr

Static ARR As String

If geoflag = 0 Then

MsgBox "No Geometry Defined"

Exit Sub

End If

If BcFlag = 0 Then

MsgBox "No Boundary Conditions Defined"

Exit Sub

End If

'Ng = 4

If CKM = 0 Or DT = 0 Or NUMTS = 0 Then

MsgBox "There is a problem with input data"

MsgBox "Check out Analysis -> Options Menu!"

Exit Sub

End If

If Km = 0 Then

MsgBox "There is a problem with input data - Thermal Conductivity"

MsgBox "Check out Analysis -> Options Menu!"

Exit Sub

End If

Domain(1) = Nn: Domain(2) = Ne: Domain(3) = L: Domain(4) = Ng: Domain(5) = Km

AddText "Transient Analysis initialised at "

Me.Refresh

frmmdi.MsgBar.Caption = "Analysing, Please Wait!"

Me.MousePointer = 11

Screen.MousePointer = 11

Time1 = Timer

MATPROP(1) = NUMTS: MATPROP(2) = DT: MATPROP(3) = TI: MATPROP(4) = CKM

Call assembgh(Domain(1), con(1, 1), le(1), x(1), y(1), G(1, 1), H(1, 1))

```

For i = 1 To Nn + L
If KODE(i) = 1 Then
UP(i) = U(i)
Else
UP(i) = TI
End If
QP(i) = 0
TEMPT(i, 1) = TI
FLUXT(i, 1) = 0
Next i

```

```
frmmdi.MsgBar.Caption = "Calculating Matrices, Please Wait!"
```

```

Call INVERSEF(Domain(1), FINV(1, 1), x(1), y(1))
Call RHSMATRIX(Domain(1), con(1, 1), x(1), y(1), le(1), S(1, 1), FINV(1, 1), H(1, 1), G(1, 1))

```

```

For its = 1 To NUMTS
THETAQ = 1
frmmdi.MsgBar.Caption = "Solving Equations for Time Step" + Str$(its) + ", Please Wait!"
Call RHSVEC(MATPROP(1), Domain(1), S(1, 1), H(1, 1), UP(1), QP(1), B(1), THETAU,
THETAQ)
'PRTCOL B(), nn + 1, "b"
Call BOUNDTR(MATPROP(1), Domain(1), S(1, 1), H(1, 1), G(1, 1), KODE(1), con(1, 1),
A(1, 1), B(1), HC(1), TA(1), U(1), Q(1), THETAU, THETAQ)
Call SOLVEBEM(Domain(1), KODE(1), U(1), Q(1), HC(1), TA(1), A(1, 1), B(1))

```

```

For i = 1 To Nn + L
TEMPT(i, its + 1) = U(i)
FLUXT(i, its + 1) = Q(i)
UP(i) = U(i)
QP(i) = Q(i)
' Q(i) = KM * Q(i)
Next i

```

```
frmmdi.ProgressBar1.Value = its * 100 / NUMTS
```

```
Next its
```

```

Time2 = Timer
CpuTime = Time2 - Time1

```

```
If CpuTime < 1 Then
msg = "Previous analysis took : " + Str$(CpuTime * 1000) + " Milli-Seconds"
ElseIf CpuTime > 1 And CpuTime < 60 Then
msg = "Previous analysis took : " + Str$(CpuTime) + " Seconds"
ElseIf CpuTime > 60 And CpuTime < 3600 Then
msg = "Previous analysis took : " + Str$(CpuTime / 60) + " Minutes"
Else
msg = "Previous analysis took : " + Str$(CpuTime / 3600) + " Hours"
End If
```

```
AddText "Transient Analysis Completed at "
SSFlag = 0
frmmain.txt = frmmain.txt + Chr$(13) + Chr$(10) + Chr$(13) + Chr$(10) + msg
frmmdi.ProgressBar1.Value = 0
Me.MousePointer = 0
Screen.MousePointer = 0
frmmdi.MsgBar.Caption = "Ready ..."
AnalFlag = 1
Exit Sub
analerr:
Me.MousePointer = 0
Screen.MousePointer = 0
frmmdi.MsgBar.Caption = "Ready ..."
'Me.Refresh
frmmdi.ProgressBar1.Value = 0
Me.MousePointer = 0
Screen.MousePointer = 0
frmmdi.MsgBar.Caption = "Ready ..."
showerror
Close
Exit Sub
```

Visual Basic program for conducting a cycle-average steady state analysis

***** - If an error occurs, go directly to eh AnbalErr Flag - *****

On Error GoTo analerr

***** - If no discretisation has been completed, notify user and exit sub-routine - *****

If geoflag = 0 Then

MsgBox "No Mesh Defined"

Exit Sub

End If

***** - If no boundary conditions have been applied, notify user and exit sub-routine -

If BcFlag = 0 Then

MsgBox "No Boundary Conditions Defined"

Exit Sub

End If

***** - If no thermal conductivity of mould material, has been defined, notify user and exit
sub-routine - *****

If Km = 0 Then

Me.MousePointer = 0

Screen.MousePointer = 0

frmmdi.MsgBar.Caption = "Ready ..."

msg = "There is a problem with input data - Thermal Conductivity"

msg = msg + "Check out Analysis -> Options Menu!"

MsgBox msg

Exit Sub

End If

***** - Set up arrays for analysis programs - *****

Domain(1) = Nn: Domain(2) = Ne: Domain(3) = L: Domain(4) = Ng: Domain(5) = Km

MATPROP(1) = NUMTS: MATPROP(2) = DT: MATPROP(3) = TI: MATPROP(4) = CKM

***** - Notify that analysis is about to start - *****

AddText "Injection Mould Analysis initialised at "

Me.Refresh

frmmdi.MsgBar.Caption = "Analysing, Please Wait!"

Me.MousePointer = 11

Screen.MousePointer = 11

***** - Note current time - *****

Time1 = Timer

***** - Check to see if a cavity has been defined, if not, notify and exit the sub-routine -

FdL = 0

For i = 1 To Ne

 kk = KODE(i)

 If kk = 4 Then

 FdL = 1

 End If

Next i

If FdL = 0 Then

 Me.MousePointer = 0

 Screen.MousePointer = 0

 frmmdi.MsgBar.Caption = "Ready ..."

 msg = "There is no cavity defined." + Chr\$(13) + "There must be a cavity defined!"

 MsgBox msg

 AddText "Injection Mould Analysis crashed at "

Exit Sub

End If

***** - Calculate the coefficient matrices, [G] and [H] - *****

Call assembgh(Domain(1), con(1, 1), le(1), x(1), y(1), G(1, 1), H(1, 1))

***** - Set up convergence tolerances and initial cavity and exterior wall temperatures -

tol = 0.001

DIFF = 2

twallold = AvMoldT

tcavityold = AvMoldT

ap = KPO / (CPP * ROP)

ab = Log((Abs(TP - tcavityold)) / Abs((TE - tcavityold)))

If TCC = 0 Then

 TC = (hm ^ 2 / (pi * pi * ap)) * ab

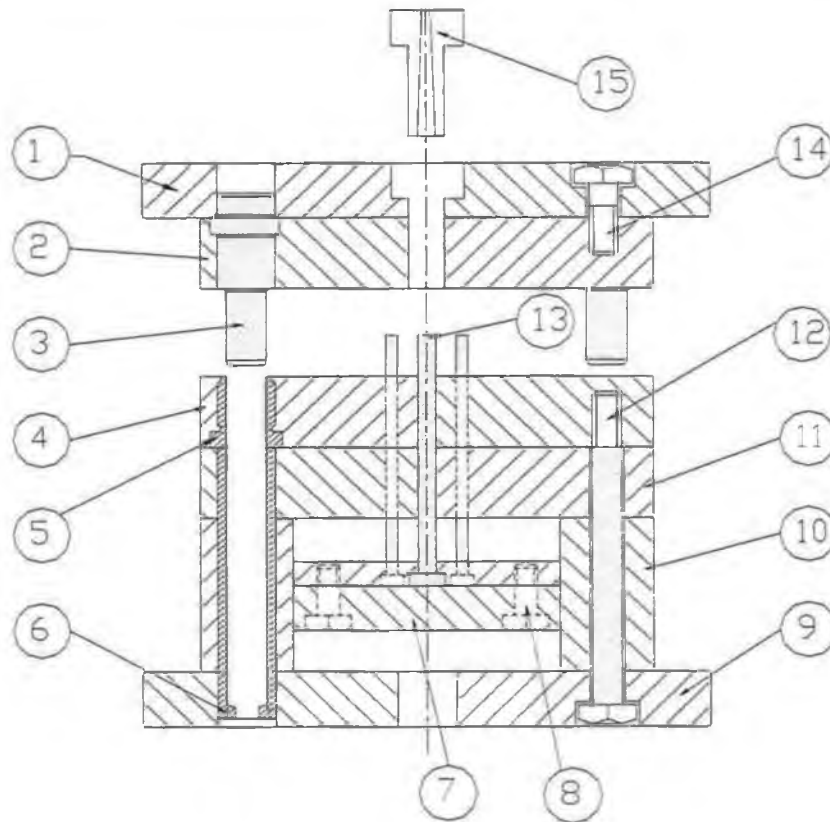
Else

 TC = TCC

End If

APPENDIX 3

TEST MOULD DRAWINGS



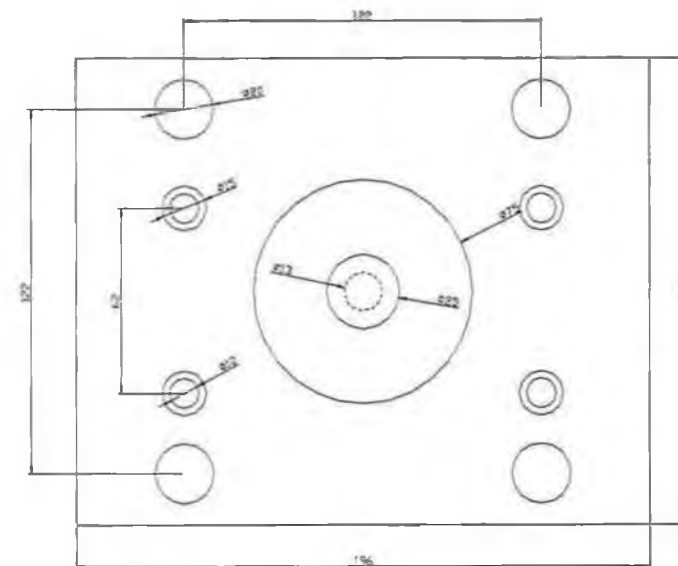
Part	Qty.	Name	DMS Code
1	1	Clamp Plate	CP1519H
2	1	Core Plate	A151926N8
3	4	Guide Pillar	GP142635
4	1	Cavity Plate	B151926N8
5	4	Guide Bush	GB1426
6	4	Liner	L2080
7	1	Ejector Set	-st-
8	4	M8 cap screw	820CS
9	1	Clamp Plate	CP1519H
10	2	Riser	R151956
11	1	Backing Plate	BP1519
12	4	M10 cap screw	1090CS
13	9	Ejector Pins	-ns-
14	4	M10 cap screw	1020CS
15	1	Sprue Bush	-ns-

Drawing Name:
"Test Injection Mould Assembly"

Drawing By:
Niall Moran
Dublin City University

All Parts Supplied by DMS

-ns- : Not Supplied
-st- : Standard Size



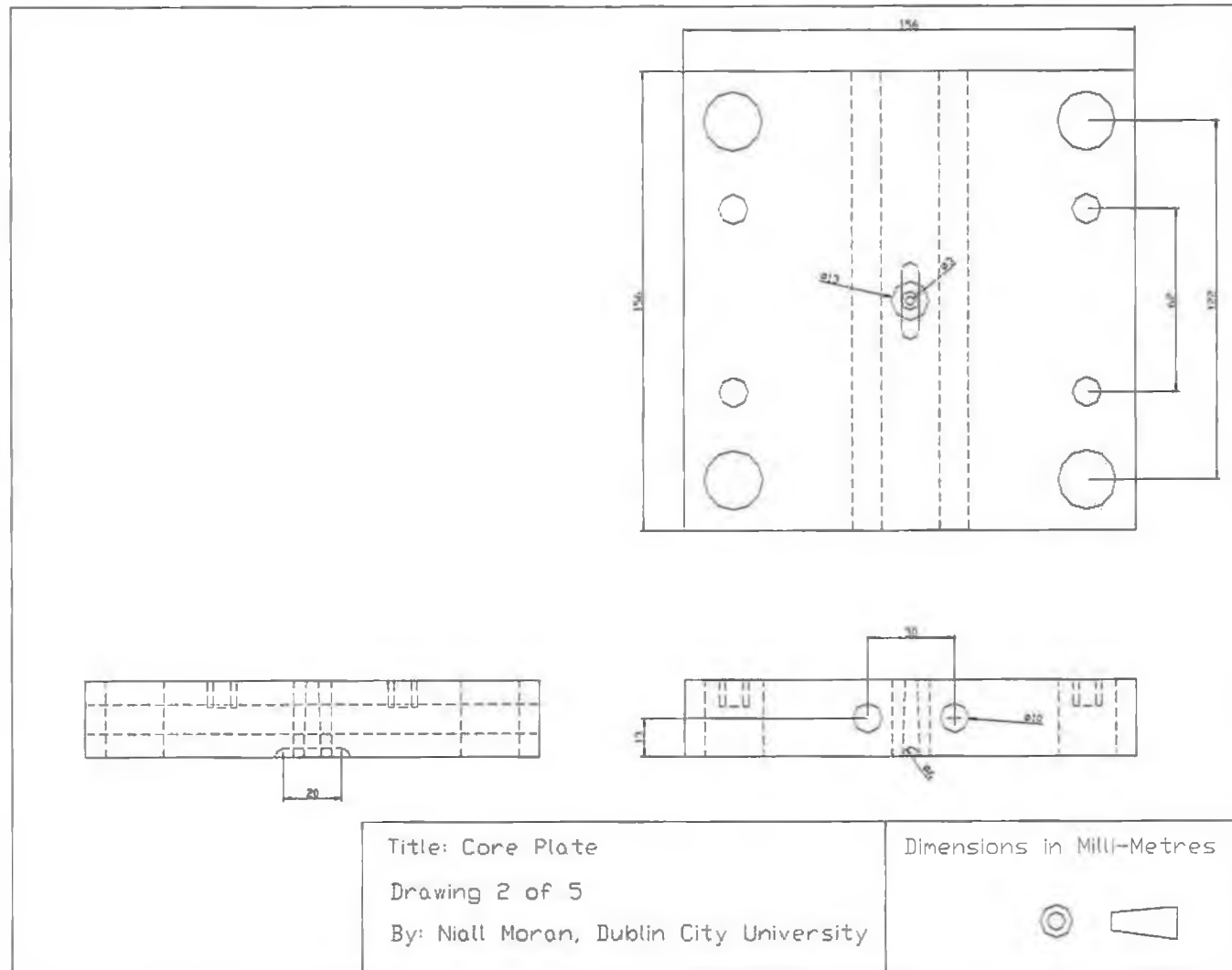
Title: Injector Clamp Plate

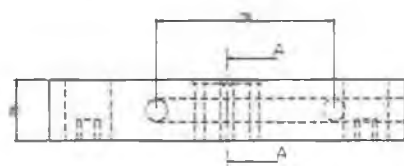
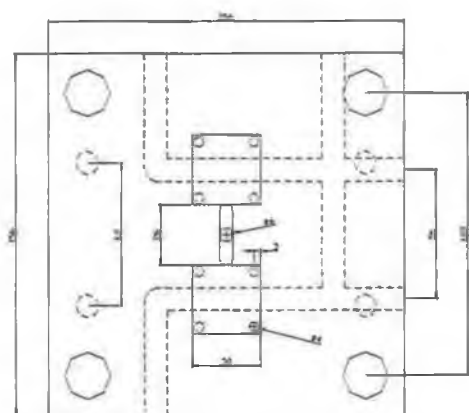
Drawing 1 of 5

By: Niall Moran, Dublin City University

Dimensions in Milli-Metres







Section A-A



Title: Cavity Plate

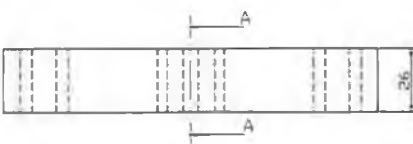
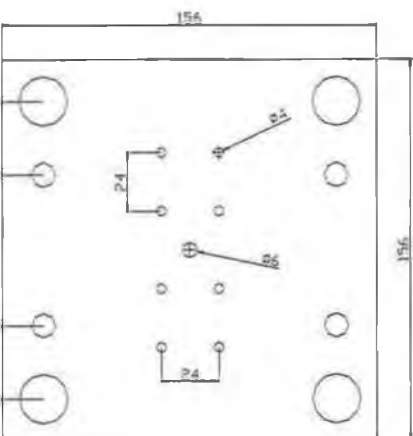
Drawing 3 of 5

By: Niall Moran, Dublin City University

Dimensions in Milli-Metres







Section A-A



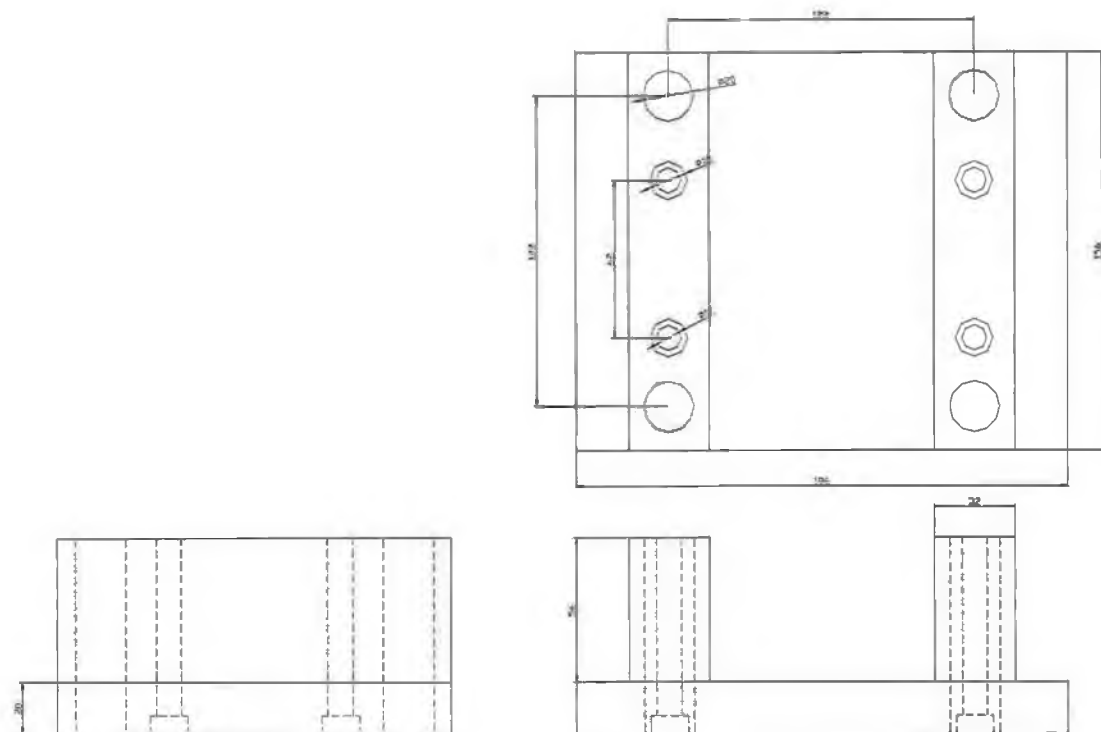
Title: Back Plate

Drawing 4 of 5

By: Niall Moran, Dublin City University

Dimensions in Milli-Metres





Title: Ejector Clamp Plate
 Drawing 5 of 5
 By: Niall Moran, Dublin City University

Dimensions in Milli-Metres



APPENDIX 4

BIBLIOGRAPHY

References

1. S. Kenig and M.R. Kamal, "Cooling Molded Parts - A Rigorous Analysis", SPE Journal, Vol.26, pp 50-57, 1970.
2. K.K. Wang, "A System Approach to Injection Molding Process", Polymer and Plastic Technology Engineering", Vol.14(1), pp75-93, 1980.
3. M.R.Barone and D.A.Caulk, "Optimal Thermal Design of Injection Molds for Filled Thermosets", Polymer Engineering and Science, July, 1985, Vol. 25(10), pp 608-617.
4. Colin Austin, "Mold Cooling", ANTEC, pp 764-766, 1985.
5. T.H. Kwon, S.F. Shen and K.K.Wang, "Computer-Aided Cooling-System Design for Injection Molding", ANTEC, pp 110-115, 1986.
6. T.H. Kwon, "Mold Cooling System design Using Boundary Element Method", Transactions of ASME, Vol. 110, pp384-394, 1986.
7. L.S.Turng and K.K. Wang, "A Computer-Aided Cooling-Line System for Injection Molds.", ASME, Chicago, Illinois, 1988.
8. Igor Catic and Pero Raos, "Theoretical Approach to Injection Mould Design Using Partial Functions and a Morphological Matrix", Plastics and Rubber Processing and Applications, Vol. 11(3), 1989.
9. S.C. Chen, S.M.Wang, Y.L.Chang and C.H. Wang, "A Study of Computer-Aided Mold Cooling Simulations Based on Different Methods", ANTEC, 1990.
- 10.H.H. Chiang, K. Himasekhar, N. Santhanam and K.K.Wang, "Integrated Simulation of Fluid Flow and Heat Transfer in Injection Molding for the Prediction of Shrinkage and Warpage", Journal of Engineering Materials and Technology, Vol. 115, pp 37-47, 1993.
- 11.Shia Chung Chen and Yung Chien Chung, "Simulation of the Cyclic Injection Mold-Cooling Process Using Dual Reciprocity Boundary Element Method", Transactions of the ASME, Vol. 117, pp 550-553, 1995.
- 12.C.Brebbia and J. Dominguez, "Boundary Element Methods Versus Finite Elements", International Conference on Applied Numerical Modelling, pp 571-586, 1977.

13. Jon Trevelyan, "Boundary Elements: The Other Analysis", Computer-Aided Engineering, pp70-74, 1989.
14. G. Athanasiadis, "Direct and Indirect Boundary Element Methods for Solving the Heat Conduction Problem", Computer Methods in Applied Mechanics and Engineering, Vol. 49, pp 37-54, 1985.
15. C.A. Brebbia and T.G.B. DeFigueiredo, "A New Variational Boundary Element Model for Potential Problems", Engineering Analysis with Boundary Elements, Vol. 8(1), 1991.
16. K.M. Singh and M.S. Kalra, "Least Squares Finite Element Formulation in the Time Domain for the Dual Reciprocity Boundary Element Method in Heat Conduction", Computer Methods in Applied Mechanics and Engineering, Vol. 104, pp 147-172, 1993.
17. A. Lahramann, "Stable Boundary Element Formulation for the Determination of Transient Temperatures with a Weighted Time Stepping Solution", 2nd International Conference on Advanced Computational Methods in Heat Transfer, pp715-733, 1992.
18. A.C. Neves and C.A. Brebbia, "The Multiple Reciprocity Method Applied to Thermal Stress Problems", International Journal for Numerical Methods in Engineering, Vol. 35, pp 443-455, 1992.
19. Weifeng Tang and C.A. Brebbia, "Critical Comparison Between Two Transformation Methods for Taking BEM Domain Integrals to the Boundary", Engineering Analysis with Boundary Elements, Vol. 6(4), 1989.
20. C.A. Brebbia and L.C. Wrobel, "Steady and Unsteady Potential Problems using the Boundary Element Method", Recent Advances in Numerical Methods in Fluids, 1979.
21. K. Davey and S. Hinduja, "An Improved Procedure for Solving Transient Heat Conduction Problems Using the Boundary Element Method", International Journal for Numerical Methods in Engineering, Vol. 28, pp 2293-2306, 1989.
22. G.F. Dargush and P.K. Banerjee, "Application of the Boundary Element Method to Transient Heat Conduction", International Journal for Numerical Methods in Engineering, Vol. 31, pp 1231-1247, 1991.

23. A. H-D. Cheng, S. Grilli and O. Lefe, "Dual Reciprocity Boundary Element Based on Complete Set Global Shape Functions", International Conference on Boundary Element Methods, pp 343-357, 1993.
24. P.W. Partridge and C.A. Brebbia, "The BEM Dual Reciprocity Method for Diffusion Problems", 8th International Conference on Computational Methods in Water Resources, pp 389-396, 1990.
25. P.W. Partridge and C.A. Brebbia, "Computer Implementation of the BEM Dual Reciprocity Method for the Solution of General Field Problems", Communications in Applied Numerical Methods, Vol.6, pp 83-92, 1990.
26. M.S. Ingber, "A Triple Reciprocity Boundary Element Method for Transient Heat Conduction", 9th International Conference on Boundary Element Technology, pp41-49, 1994.
27. Yinglong Zhang and Songping Zhu, "On the Choice of Interpolation Functions Used in the Dual Reciprocity Boundary Element Method", Engineering Analysis with Boundary Elements, Vol. 13, pp387-396, 1994.
28. Songping Zhu, Pornchai Satravaha & Xiaoping Lu, "Solving Linear Diffusion Equations with the Dual Reciprocity Method in Laplace Space", Engineering Analysis with Boundary Elements, Vol. 13, pp1-10, 1994.
29. T. Yamada, L.C. Wrobel and H. Power, "On the Convergence of the Dual Reciprocity Boundary Element Method", Engineering Analysis with Boundary Elements, Vol. 13, pp291-298, 1994.
30. Keith Paulsen and Daniel Lynch, "Calculation of Interior Values by the Boundary Element Method", Communications in Applied Numerical Methods, Vol. 5, pp 7-14, 1989.
31. J.M. Crotty Sisson, "Accurate Interior Point Computations in the Boundary Integral Equation Method", Computer Methods in Applied Mechanics and Engineering, Vol. 79, pp 281-307, 1990.
32. N. Zabaraz, S. Mukherjee and O. Richmond, "An Analysis of Inverse Heat Transfer Problems with Phase Changes Using an Integral Method", Transaction of ASME, Vol. 110, pp 554-561, 1988
33. J.S. Hsiao and B.T.F. Chung, "An Efficient Algorithm for Finite Element Solution to Two-Dimensional Heat Transfer with Melting and Freezing", Transactions of ASME, Vol. 108, pp 462-464, 1986.

34. Xinghong Li, Rockson Huang, and Davor Juricic, "Application Programs Written by Using Customising Tools of a Computer-Aided Design System", Tribology Symposium ASME, Vol. 72, 1995.
35. Incropera, F.P. and De Witt, D.P., Fundamentals of Heat Transfer, Wiley & Sons, 1981, ISBN 0-471-08961-3.
36. Isayev, Avraam I., Injection and Compression Molding Fundamentals, 1987, Dekker New York, ISBN: 0824776704.
37. Brebbia, C.A. and Dominguez, J., Boundary Elements - An Introductory Course, Computational Mechanics, 1992, ISBN : 1-85312-160-6
38. Brebbia, C.A., Telles, J.C.F. and Wrobel, L.C., Boundary Element Techniques - Theory and Applications in Engineering, Springer-Verlag, 1984, ISBN : 3-540-12484-5.
39. Partridge, P.W., Brebbia, C.A., and Wrobel, L.C., The Dual Reciprocity Boundary Element Method, Computational Mechanics, 1992.
40. Gastrow, Injection Molds 102 Proven designs, Hanser Publishers, 1983, ISBN : 0-02-949440-0.