東　　海　　大　　學

工業工程與經營資訊研究所

碩士論文

以基因演算法結合類神經網路最佳化射出成型作業之翹曲與收縮值

研 究 生：張靜怡

指導教授：黃欽印 博士

陳武林 博士

中 華 民 國 九 十 八 年 七 月

Optimization of injection molding process parameters using combination of artificial neural network and genetic algorithm method to minimize the Warpage and Shrinkage value

By

Ching-Yi Chang

Advisor： Prof. Chin-Yin Huang

Prof. Wu-Lin Chen

A Thesis

Submitted to the Institute of Industrial Engineering and Enterprise Information at Tunghai University

in Partial Fulfillment of the Requirements

for the Degree of Master of Science

in

Industrial Engineering and Enterprise Information

July 2009

Taichung , Taiwan

以基因演算法結合類神經網路最佳化射出成型作業之翹曲與收縮值

學生：張靜怡　　指導教授： 黃欽印 博士
陳武林 博士

東海大學工業工程與經營資訊研究所

摘要

傳統上以技師進行試誤以找出塑膠射出成型生產之參數組合的方法，已無法因應產品生命週期短、資訊快速交換的時代，同樣也耗費時間以及金錢。又過去探討文獻中也普遍只針對單一品質特性進行分析討論或採用反應曲面法 Response Surface Method,RSM 以 model-base 的方式進行二個品質特性的分析。而透過 model-free（如類神經網路）的方法針對二項品質特性的分析仍未見於文獻。故此，本研究提出以基因演算法結合類神經網路找出射出成型作業之最佳化翹曲及收縮的方法，並找出最佳之參數組合。透過類神經網路(model-free)方法訓練出一射出成型輸入輸出關係之函數，透過輸入製程參數便可獲得一組對應之收縮翹曲輸出值，接著便以基因演算法搜尋出最小收縮及翹曲組合。最後透過 moldflow 驗證整體實驗之準確性。

本研究選用手機機殼模型進行分析，以射出時間、射出壓力、保壓時間、保壓壓力、冷卻時間、冷卻溫度、熔劑溫度、模具溫度、開模時間等九個製程參數作為初始控制因子。首先，以 Moldflow 模擬軟體進行模擬並搭配田口直交表及變異數分析找出製程之顯著控制因子（射出時間、保壓壓力、開模時間、溶劑溫度），接著透過類神經網路結合基因演算法搜尋出最佳化翹曲及收縮的兩組無異之最佳解組合。

關鍵字：射出成型、田口直交表、基因演算法、類神經網路、最佳化收縮與翹曲

Optimization of injection molding process parameters using combination of artificial neural network and genetic algorithm method to minimize the Warpage and Shrinkage value

Student: Ching-Yi Chang

Advisor：Prof. Chin-Yin Huang

Prof. Wu-Lin Chen

Department of Industrial Engineering and Enterprise Information

Tunghai University

ABSTRACT

In this study, optimum values of process parameters in injection molding of a cell phone thin shell to achieve both minimum warpage and shrinkage are determined. Nine process parameters are considered in the research: injection time, injection pressure, packing time, packing pressure, cooling time, cooling temperature, mold-open time, melt temperature, and mold temperature. In finding the optimum values, finite element software MoldFlow, ANOVA, full factorial experimental design, artificial neural network and genetic algorithm are exploited. A predictive model for warpage and shrinkage is created using artificial neural network exploiting finite element analysis results. Then, genetic algorithm is applied to find the optimum process parameter values for the minimum shrinkage only. By relaxing the shrinkage gradually and to minimize the warpage using genetic algorithm, the optimum process parameters are found for the two indifferent solutions. The two indifferent solutions means the two quality sets of shrinkage and warpage are equivalent minimum solutions. The results indicate that this research can optimize multiple quality criteria by using model free methods for the plastic injection molding process.

Keywords: Moldflow, Taguchi's orthogonal array, Artificial neural network, Genetic Algorithm, Optimization of warpage and shrinkage.

誌謝

畢業季節又到了，心情總是無比的複雜，六年在東海的日子讓我擁有無比美好的回憶。東海熟悉的校園、引領我們學習的師長們、從研究所考試就一起奮鬥的東海幫、研究所一起努力的好夥伴們都是我這輩子最重要的禮物。開心畢業的背後多少的回憶難以忘懷，充滿著歡笑、感動、壓力、悲傷的記憶都是未來我向前進的動力。特別是研究所的兩年我得到了更多珍貴的人生態度，黃欽印老師更是我人生階段中很重要的恩師，帶領著我成長並給予我開闊的眼界。半年的交換學生生活讓我擁有了很多精采的回憶。也讓我認識了南伊利諾大學的張豐昌老師，在陌生的美國生活裡多虧了張老師的關心及照顧，讓我學會獨立也增長了學問。感謝兩位恩師給予我自由思考及學習的空間，也帶領我走向人生另一個階段。

本研究論文之完成，首先要感謝黃欽印老師與陳武林老師無私的教誨、每周耐心指導與督促，藉由研究學習的過程中，不僅僅學到了解決問題的方法也慢慢訓練自己邏輯思考及口頭報告的能力，不論是課業或做人處事態度上，師長們總是不厭其煩的叮嚀與囑咐，讓我能夠持續地進步往前。另外還要感謝鄭辰仰博士與賴奕銓博士，針對論文初稿之問題與疏失，給予許多的幫助與寫作之建議，使我的論文架構能更為完整，對於觀念上的澄清有極大的幫助。

兩年的研究所生活中，承蒙王立志老師、啟偉學長、弼仁學長、敦婷學姐、建璋學長的帶領及指教，還有信宏、德芸、任志等研究室夥伴們的互相勉勵，靖雅、斯暢、怡芳、文冠、世倫、禺澐等學弟妹們給予我的支持與鼓勵，使我在學習的過程中仍然充滿著歡笑與喜悅。還要特別感謝系上助教默默的協助，讓我可以專心於論文研究，真的很感謝你們。

最後特別要感謝我最愛的爸爸媽媽姊姊廣瀚，謝謝你們一路以來的支持，我很愛你們，如果沒有你們的關愛與支持、沒有你們的陪伴與鼓勵，讓我能一心一意的專注於學業上，就無法成就今天的我，謝謝你們。

謹將這份成果獻給每一位幫助過我的貴人，有你們支持及鼓勵，才能完成此論文。

張靜怡　謹誌於

東海大學工業工程與經營資訊研究所

虛擬企業與資料探勘研究室

中華民國九十八年七月

目錄

圖目錄

表目錄

第一章、緒論

1.1 研究背景與動機

由於科技不斷的發展進步，使得資訊及通訊產品快速的推陳出新，消費性電子產品從設計、開模、原型直到產品實際上市的時間越來越短，而產品型態也以輕巧性、多功能性、可提升工作效率、堅固耐摔…等設計概念作為產品設計時的考量。在全球化競爭激烈的市場中，要如何透過各種方法來加快產品上市速度、降低成本、減少不良品浪費、增加利潤…等都是企業所關注的焦點。

產品的設計，逐漸採用塑膠材料(如 ABS、PC/ABS)或其他塑膠工程材料來取代其它的材料。塑膠材料的比重為 0.95~1.2 之間，遠比鐵、鋁等金屬材料之比重小，且塑膠發展至今種類繁多，具用途廣泛、質地輕、可塑性高、具高機械強度、永不生鏽、吸震性、成本低、製造取得方便、可回收、外觀色澤美…等優勢。因此，塑膠材料成為 3C (電腦、通訊、消費性電子)產品機殼的最佳選用材料，尤其是應用於筆記型電腦、手機、PDA、MP3…等產品的外殼及零件上。

根據「塑膠工業技術發展中心」之資料可知，目前，全球塑膠產量達 2 億多噸，實際消耗 1.8 億噸，預計 2010 年消耗量將達 2.5 億噸，其中亞洲為 0.9 億噸，占全球 36%。目前中國已經是名副其實的世界塑膠生產、消費和進口大國，預計 2010 年塑膠樹脂消費量將突破 6,000 萬噸，占全球 20%，年成長 15%以上，在全球塑膠製品產量排名穩居亞軍。可見塑膠產業前景之廣闊，但其也面臨巨大挑戰：1、原油價格持續高漲，塑膠價格隨之增漲；2、模具鋼材價格高漲；3、塑膠製品成形週期要求更短；4、塑膠製品價格更便宜，競爭越來越激烈。因此，要如何生產高附加價值製品，準確的把握塑膠製品之發展趨勢，如尺寸公差、結構強度，以先進的設計流程取代傳統過時的設計方式，透過電腦輔助工程(Computer Aided Engineering, CAE) 的引進結合品質最佳化方法以解決射出成型製程上面臨的問題已為趨勢。

塑膠材料的加工方法以射出成型為主，而在射出成型參數設定上主要可分為兩類: 1、以師傅或技師依賴過去之經驗來進行射出成型的參數設

定。2、工程師使用模流分析軟體找出射出成型參數的初始值，並透過模擬找出適合的參數組合。這兩類的方法都尚無法快速準確的找到參數的最佳組合，在試誤的過程中需花費相當多的時間及成本，已無法應付成本低、交期短的市場需求。如何快速的針對某項產品找出其塑膠射出成型最佳品質之製程參數設定以應付市場的生命週期為目前亟需克服的問題。

本研究之目標在於搜尋最佳化製程參數設定使得翹曲及收縮值達到最小，將針對塑膠射出成型之製程參數，先以田口直交表搭配變異數分析找出顯著之製程參數，接著應用倒傳遞類神經網路結合基因演算法以找出最佳之參數組合，建立一套射出成型作業之翹曲與收縮最佳化的搜尋方法。

1.2 研究目的與範圍

射出成型製程中，從塑膠原料加熱成熔融狀液體，再利用高速高壓射入模具中使其成形、冷卻，然後脫模使成為成品的這段過程中，有許多的影響因子都將導致產品產生不可預測的缺失，如收縮、翹曲、包風、流痕、下陷、剪應力…等。其中，Liao et al.(2004)提出射出成型條件設計中，減少收縮與翹曲量的產生為提高產品品質最重要的一部分。Jansen et al.(1998)提出保壓壓力及保壓時間為影響收縮很重要之因素。Huang and Tai(2001)也提出保壓壓力為影響塑件翹曲之最重要之參數，其次則是模具溫度、熔劑溫度及保壓時間等參數。而 Ozcelik and Sonat(2008) 則認為模具溫度、熔融溫度、注射時間、注射溫度為最小化翹曲之重要參數。因此，透過文獻之整理以注射時間、注射壓力、保壓壓力、保壓時間、冷卻時間、冷卻溫度、模具溫度、熔劑溫度、開模時間等九個重要的製程參數作為本研究之初始參數選擇，希望建立一最佳化射出成型製程之方法使得收縮及翹曲量降至最低，並找出其製程參數組合。

本研究之目的:

1. 找出實驗中手機機殼模型顯著之製程參數，並透過 moldflow 模擬取得與參數相對應之翹曲收縮數據，利用類神經網路訓練出一多重輸入多重輸出之函數。此函數為一 model-free 學習結果，有不受限制於模型且可搜尋高維度之解等特性，本研究欲證實此方法為較好之多目標解決方案。

2. 接著將隨機得到之參數代入類神經網路訓練函數以取得輸出值，最後，以基因演算法搜尋出製程之最佳參數組合，本研究針對多個品質特性及多個測量值進行考量，為一多目標問題。透過類神經網路結合基因演算法找出一個可以快速搜尋射出成型製程參數設定以達最佳化多目標之翹曲及收縮組合之搜尋方法，作為工程師進行整體考量時的參考依據。

本研究之詳細敘述如下:

1. 本研究選定九個初始參數各三個水準，故以 L_{27} 田口直交表找出初始之 27 組實驗參數組合，並利用此 27 組參數組合以 Moldflow 進行模擬實驗，得出四組收縮及四組翹曲值。接著以變異數分析表找出手機機殼製程顯著之參數組合。
2. 利用 Excel 依參數之上下界水準範圍隨機取得四位小數之 81(3^4)組參數設定值，並利用 Moldflow 進行模擬，取得品質特性收縮及翹曲之實驗數值。
3. 以免費軟體 Bloodshed Dev-C++ 5.0 Compiler 以 C 語言撰寫出一類神經網路程式，如附錄 A、附錄 B，並透過實驗數值採用 10-Fold Cross validation 方法訓練出一準確的多重輸入多重輸出之類神經網路。
4. 接著同樣以免費軟體 Bloodshed Dev-C++ 5.0 Compiler 以 C 語言寫出一基因演算法程式，如附錄 C，並透過基因演算法搜尋出最好的基因組合，找出最小翹曲之參數組合及最小收縮之參數組合，並搜尋出翹曲及收縮最佳化之參數組合。

本研究主要之研究限制與範圍如下:

1. 由於資源及成本上的限制，本研究以模流分析軟體 Moldflow 代替實際射出成型機台，實驗所需之數據皆由 Moldflow 軟體模擬取得。
2. 在塑膠射出成型中，控制參數可分為機器參數及製程參數兩類。本研究僅考量製程參數因子的部分，部份機器參數則以 Moldflow 預設值設定。
3. 本研究所考慮之九個製程參數皆假設為可控制因子，未考量之因子以 Moldflow 預設值設定。

1.3 研究架構與流程

本研究所使用之研究架構與流程主要分為四個階段: (1)射出成型流程之探討 (2)射出成型製程控制之發展文獻 (3)研究方法之發展 (4)系統實作及分析。研究架構圖可參考圖 1.1。

步驟一、射出成型流程之探討

說明本研究之動機與目的，點出目前塑膠射出成型產業面臨的挑戰及在參數設定上所遭遇之困難點及對於品質的影響，接著說明射出成型製造過程簡介及流程，透過熟悉射出成型之詳細步驟以了解各階段中眾多因素對於最終成品之影響。接著，說明射出成型產品的品質要素、機器參數、製程參數及其他製程流程對於產品品質特性影響的一系列過程。並說明塑膠射出成型產品導致不良品的缺陷情形。

步驟二、射出成型製程控制之發展文獻

首先針對過去應用於製程參數最佳化的文獻進行探討與分析，並進一步說明射出成型之機器控制與製程控制對於產品品質之影響及一般常用於射出成型之參數最佳化的方法。接著，收斂至本研究並將文獻作分析整理並進行討論，以作為未來的研究可選擇的方向。

步驟三、研究初始條件之建立

透過研究情境之建立，以 Moldflow 軟體模擬射出成型的實際製造過程，從成品模型之建立、機器參數之設置、製程參數之設定。接著敘述田口直交表、變異數分析法如何應用於本研究，並說明類神經網路及基因演算法之理論背景、重要概念及實驗實施之步驟。

步驟四、實驗與結果分析

此階段為實作部份，以 Moldflow 模擬實際製程所產生之 81 組亂數參數組合的實驗輸出值，透過倒傳遞類神經網路以 10-Fold Cross Validation 的方法訓練出一學習模式，接著應用基因演算法演化出最佳的參數組合。包

括實驗執行、資料分析及結果說明，並對本實驗所提出的方法進行驗證。最後為本研究做一結論，並建議未來的研究方向。

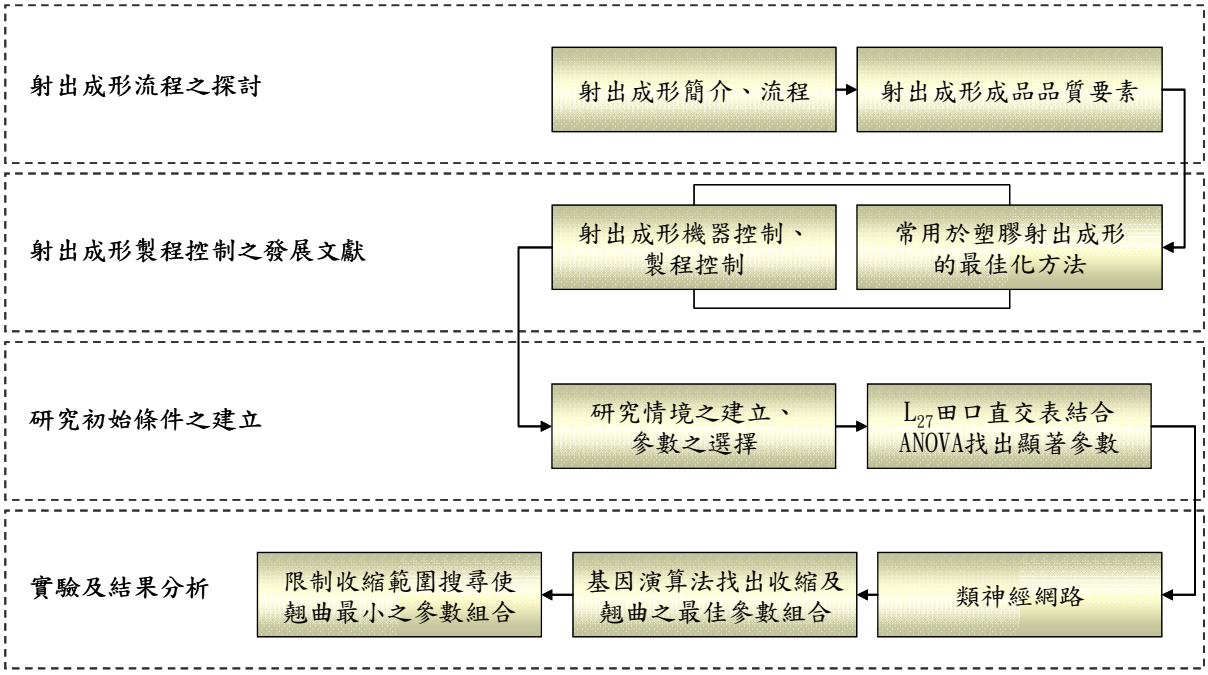


圖 1.1、本研究之架構與流程

第二章、文獻探討

2.1 射出成型簡介

射出成型方法為所有成型方法中效率最高、產量最大、自動化及適合用於複雜化產品之製造的一種加工方法。塑膠成型的步驟通常包含加熱塑料成熔融態、充填、保壓、冷卻和頂出等步驟(羅壬成,2006)，如圖 2.1，在過程中需透過控制大量的參數，使成品達到企業要求之品質，而控制的參數大致可分為兩類: 機器參數及製程參數。其中機器參數是指任何可從機台上設定或調整的參數；而製程參數是指任何能直接影響塑膠高分子的參數。

圖 2.1、塑膠射出成型步驟

一部完整的射出成型機主要分為油壓系統、射出系統、模具系統及鎖模系統單元，如圖 2.2(源自黃俊欽)。以下針對各個部分進行詳細之說明(陳良相 et al., 2005):

1. **油壓系統:** 射出成型機的油壓系統提供開啟與關閉模具的動力，蓄積並維持鎖模力頓數，旋轉與推動螺桿前進，並致動系統之頂出銷以及移動公模測。
2. **射出系統:** 包括了料斗(hopper)、迴轉螺桿、料筒(barrel)組合和噴嘴(nozzle)。主要的功能是存放及輸送塑料，使塑料歷經進料、壓縮、排氣、熔化、射出及保壓階段。
3. **模具系統:** 包括了導桿(tie bars)、固定模板(stationary platen)、移動模板(movable platen)和容納模穴、流道系統、頂出銷和冷卻管路的模板(molding plates)。主要功能是使熱塑性塑膠的熔膠在模穴內凝固成需要的形狀及尺寸。
4. **鎖模系統:** 用來開啟/關閉模具，支撐與移動模具組件，產生足夠的力量以防止模具被射出壓力推開。

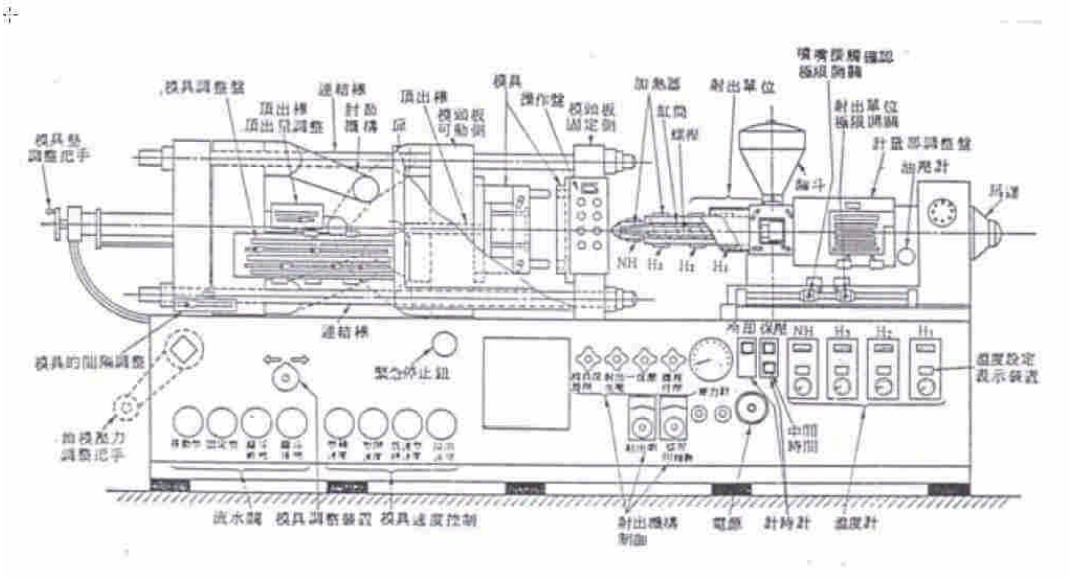
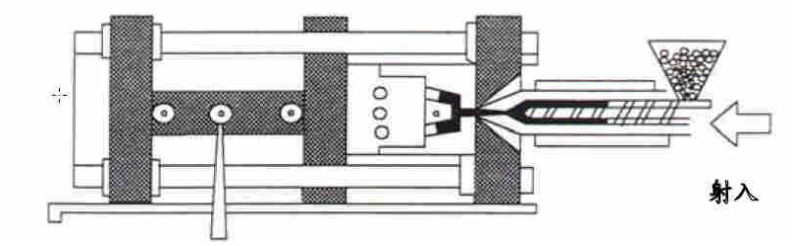


圖 2.2、射出成型機結構

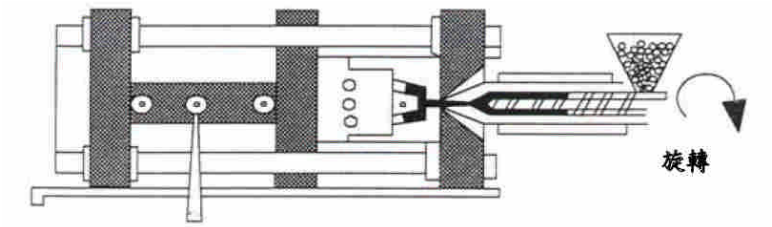
2.1.1 射出成型之流程

射出成型流程的整體循環週期如圖 2.3 所示，主要分為充填階段、保壓階段、冷卻階段和開模階段等四大階段 (羅壬成, 2006):

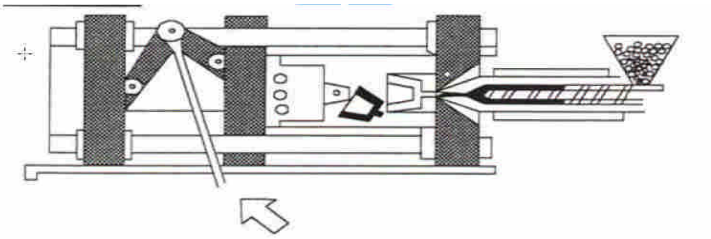
1. **充填階段(Filling):** 利用一壓力將高溫下之高分子熔融液注入冷卻模穴中，流體不斷擴展至模具整個區域，最後填滿模穴內之空間。
2. **保壓階段(Packing):** 當充填過程完畢時，此時模具仍然必須保持一個高壓狀態，因為熔融液於冷卻狀態時，會因密度變化而產生收縮，故維持一高壓以補充熔融液收縮時充填不足之膠料以達產品尺寸之要求。
3. **冷卻階段(Cooling):** 此階段即為物件持續進行冷卻，直到塑件成型，接著利用推出裝置將產品退出模穴。冷卻時間占成型週期相當高的比例，而成型品的冷卻時間依塑膠性質、成型品形狀、大小、尺寸、精度而有所不同。
4. **開模階段(Mold-Open):** 模穴打開，頂出成品，並準備下一循環。



流動充填階段，開始射出熔膠注入模穴中



保壓階段，熔膠持續擠入模穴以補充冷卻之收縮量



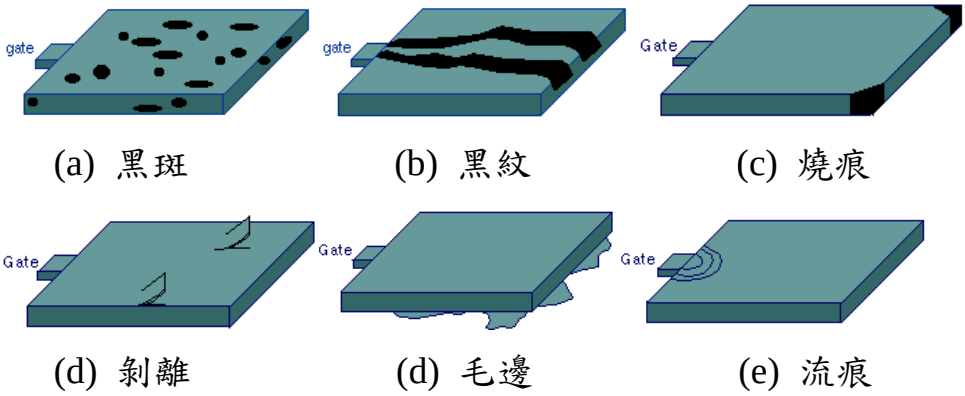
在熔膠冷卻固化成型後自模具頂出、開模，成品成型

圖 2.3、射出成型流程

2.2 射出成型成品品質因素

無論是何種型式的射出成型，最終影響成形品品質之因素，不外乎包括(1)原料的種類 (2)射出成型機之型式 (3)模具設計 (4)加工條件(溫度、壓力及時間…等) (陳劉旺 et al., 1989)

一般塑膠射出成型品常出現之瑕疵項目包括: 翹曲、收縮、尺寸精度不良、塑料充填不足、凹陷、燒痕、剝離、黑斑、黑紋、流痕、強度不足、毛邊等問題。其中 Yang and Gao(2006)認為產品重量是射出產品品質很重要的品質屬性，因為產品重量與其他的品質特性相關，特別是尺寸。而楊景程(2000)認為塑膠成形品之收縮、凹痕、翹曲變形為業界所面臨之最重要的問題。



(a) 黑斑　(b) 黑紋　(c) 燒痕

(d) 剝離　(d) 毛邊　(e) 流痕

圖 2.4、塑膠成形品可能產生之瑕疵

在射出成型的過程中，產品會因為成形材料本身性質、成形條件設定不當、模具設計或製作不完備、成形品設計不良及射出成型機成形能力不足等主要的加工要因對產品品質產生影響(張永彥, 2006)，當塑件成品產生以下缺陷時，可針對以下因素進行修正:

1. **設備不良:** 必須選用適合之射出成型機台大小、射出噸數及相關之塑料儲存、輸送等。
2. **材料不佳:** 塑料的選擇依照產品外殼之機械強度要求及作業需求，必須選擇合適之塑料來進行加工。材料不佳易造成成品的缺陷。
3. **製程不佳:** 射出成型中的速度、成形週期、充填時間、充填壓力、保壓時間、冷卻時間…等，皆為製程中重要的參數條件。塑膠成形加工之製程條件應隨時調整至最合適之參數組合。
4. **模具不良:** 模具的設計精度與製造因素將直接影響產品的品質。

2.3 射出成型機器控制及製程控制

Chen and Turng(2005)建議將變數分類為三個範疇: (1)機器變數　(2)製程變數 (3) 品質變數。而這些變數皆對於產品最終品質及製程的經濟性有顯著的影響。機器變數包含了所有製程設定階段的參數，像是料筒溫度、噴嘴溫度、冷卻溫度、保壓壓力、保壓時間、注射時間…等。製程變數則為對於製程階段至產品產出階段產生影響之參數，像是熔融溫度、熔融壓力、最大剪應力…等。品質參數包含所有與產品品質相關的量值，像是收縮、翹曲、重量、厚度及流痕…等。其研究針對此三個層級的參數對應出

三種回饋控制: 機器控制、製程控制、品質控制。機械控制確保製程的生產基臺能穩定的執行；製程控制確保實際製造情況能正常運作；品質控制則是以實際成品之品質做為控制對象。

透過良好的機器控制及製程控制，以確保機台能穩定的執行射出成型作業並使得製造的過程不產生任何的品質缺失為成形作業之最終目標。目前，我們只針對其中某幾個特定的參數進行分析並找出最佳的參數設定組合。其中 Zhao and Gao(1999)提出不同的製程參數會影響射出成型產品的品質，像是熔融溫度、模具溫度、注射壓力、注射速率、注射時間、保壓壓力、保壓時間、冷卻溫度。

未來,我們希望可以針對機器控制中的澆口位置及流道系統…等的配置做進一步的分析研究甚或是製程控制中的最大剪應力等特性進行整體的綜合分析，以獲得更好的品質結果。

2.4 射出成型常用之最佳化方法

近年來常用於最佳化射出成型製程之方法,一般為先利用模流分析軟體如 Moldflow、C-Mold、Moldex 3D…等，進行產品塑件之模擬產出，接著以單一或是結合兩種以上的最佳化方法進行實驗分析，其中可能包括田口品質方法、數值模擬分析、線性迴歸模型、反應曲面法、類神經網路、基因演算法…等方法，皆為射出成型製程常用之最佳化方法。

過去針對翹曲及收縮進行分析之文獻包括，Liao et al.(2004) 透過 Cyclone Scanner, PolyWorks 測量手機機殼翹曲及收縮值，利用田口品質方法及變異數分析進行分析，找出保壓壓力為影響翹曲及收縮最重要之影響因子。洪啟偉(2007)則利用反應曲面法及田口品質方法針對翹曲及收縮進行分析比較時，發現反應曲面法為一優於田口品質方法的最佳化方法。而 Chiang and Chang (2007)利用 PC/ABS 塑料之手機機殼模型，利用變異數分析及反應曲面法進行翹曲及收縮分析時，找出模具溫度、保壓時間、保壓壓力、冷卻時間此四個製程參數之最佳參數組合,使得翹曲及收縮值減少。但反應曲面法為 Model-based 之最佳化方法,在最佳解求解的過程中仍受限於模型之設定，無法搜尋到高緯度的可能的最佳結果。

而類神經網路為一種 Moldel-free 的求解方法，且過去尚未有文獻已類神經網路結合基因演算法進行翹曲及收縮兩種品質特性的分析研究。像是 Woll and Cooper(1996)僅比較線性迴歸模型、製程控制模型(SPC)及倒傳遞網路…等不同模型預測成品之重量與長度。Bewal and Toncich(1998)則是利用類神經網路模型，以塑料溫度及塑料壓力為控制對象，有效預測出塑膠成型成品之重量。而 Mok and Kwong(2001) 以手機模型利用模糊邏輯的相似性分析結合類神經網路，快速的找出射出成型製程的初始設定參數，僅研究說明類神經網路具有良好解決非線性問題的能力。

Shen et al.(2006) 針對品質特性收縮進行研究得出結合倒傳遞類神經網路及基因演算法之方法，可有效解決射出成型製程條件與其品質特性間複雜的非線性關係的問題。Kurtaran et al.(2005)以基因演算法及類神經網路僅針對最小化翹曲進行分析，發現類神經網路具有良好的預測能力且基因演算法為一個可有效找尋最佳解的方法。相關之文獻整理於表 2.1。

表 2.1、射出成型文獻之整理

作者(年份)	品質特性	研究方法	選擇之參數	材料	模型類型
多品質特性					Model -Based
Velia Garc´ıa et. al.(2008) [20]	Cycle time and Warpage	Data Envelopment analysis(DEA), Design of Experiment(DOE), Multiple-criteria optimization	PP, PT, MET, GTmp, NTmp, IS	PMMA,	
Ko-Ta Chiang, Fu-Ping Chang (2007) [3]	Shrinkage and Warpage	Response surface methodology(RSM), Sequential approximation optimization (SAO), Centered central composite design (CCD), ANOVA analysis	PT, PP, MOTE,COTE	PC/ABS	
單一品質特性					
Babur Ozcelik*, Ibrahim Sonat (2008) [15]	Warpage or Force	Moldflow, Regression analysis, ANOVA, Taguchi method, Structure analysis	Thickness (0.9, 1, 1.1 mm), PP, PT, MET, MOTE	PC/ABS	
Ko-Ta Chiang, (2006) [4]	Shrinkage, or Weld line	Taguchi Method, Orthogonal array, DOE, Grey relational analysis and fuzzy logic, ANOVA analysis (Strength is measure by ASTM D638)	Open Mold Cavity time, MOTE, MET, Filling Time, Filling pressure, PT, PP, CT	PC/ABS	

作者(年份)	品質特性	研究方法	選擇之參數	材料	模型類型
Yuehua Gao, Xicheng Wang*(2008) [7]	**Warpage**	MoldFlow, Kriging surrogate model, Design of Experiment(DOE),	IT, PP, PT, MET, MOTE	PC/ABS	
Wei-Jaw Deng, et. al.,(2008) [5]	**Weight** (measure by Mettler AE-100)	Taguchi method, Regression analysis, Davidon-Fletcher-Powell (DFP)method	IT, PP, IS, VP switch. Fixed MET:40°C IP: 150 MPa	PP	
Hasan Kurtaran ,et. al. (2006) [10]	**Warpage**	Integrating Finite Element(FE) ANOVA analysis, Design Of Experiment(DOE), Response Surface Method(RSM), Genetic Algorithm(GA), MoldFlow	PP, PT, COTE, MET, MOTE	ABS	
M.C. Song et. al (2005) [18]	**Filling volume**	Taguchi method, Numerical simulation, MoldFlow	Injection Rate, IP, MET, Metering size, Part thickness	PP	
Shen Changyu, et.al., (2005) [17]	**shrinkage**	Artificial Neural Network(BPNN), Genetic Algorithm(GA), Design of Experiment (DOE)	MET, MOTE, IT, PT, holding pressure	ABS	
Hasan Kurtaran et.al.(2005) [10]	**warpage**	Moldflow, Design of experiment(DOE), Artificial neural network, Genetic algorithm	MOTE, MET, PP, PT, COTE	ABS	
多品質特性					**Model -Free**
Liao et. al (2004) [11]	**Warpage, and Shrinkage**	Taguchi method (L27), ANOVA, F-test。C-MOLD	MOTE, MET, PP, IR	PC/ABS	
單一品質特性					
Tuncay Erzurumlu, Babur Ozcelik (2005) [6]	**Warpage or Sink index**	Taguchi method, ANOVA analysis, MoldFlow	PP, MOTE, MET, Rib cross-section type, Rib layout angle	PC/ABS, POM, PA66	
Huamin Zhou, Peng Zhao, Wei Feng (2006) [23]	**Part quality or cycle time**	Integrate case-base reasoning(CBR) and Fuzzy inference	Flow length, average thickness, cavity volume, IT, IP, PP, PT, CT, MET	ABS	
羅壬成(2005) [13]	**Warpage**	Taguchi method, Moldex3D	MOTE, IT, MET, Runner position, Plastic material	ABS	

2.5 小結

透過過去文獻的回顧，射出成型塑件之品質普遍為大家所關注的研究方向，研究者皆希望可以透過一良好且適當的方法使產品在快速生產之餘品質同樣具備優良的水準。而單一使用田口品質方法、類神經網路或是基因演算法等方法已無法滿足多個目標輸出之需求，林啟淙(2005)提出以基因演算法結合類神經網路可以加快最佳化的速度，且楊景程(2000)也說明了結合倒傳遞類神經網路與基因演算法具有方便性、泛用性及適應性、穩健性、實用性、全域性和平行性等優點，可解決田口品質方法只能針對單一性質探討的問題。而反應曲面法為 model-base 方法在搜尋的過程中受限於某個範圍，無法完整進行最佳解之搜尋。故本研究將採用倒傳遞類神經網路(Model-Free)結合基因演算法，並應用田口直交表之水準搭配變異數分析找出顯著之參數組合，藉此找出目標品質特性之最佳組合。

第三章、研究方法

3.1 模流分析

3.1.1 電腦輔助工程(Computer-Aided Engineering, CAE)

CAE(Computer-Aided Engineering) 模流分析技術日漸純熟，使用者可透過電腦進行產品開發設計前的模具、塑膠材料及射出機台之選擇，並對於成品製程操作條件、流道設計…等加工參數進行模擬，以減少產品於現場實際試誤時之成本與時間的浪費，以期達到 CAE 模流分析與現場實務同步化，進而達成快速及周密的生產流程藉此提升產業之競爭力。

本研究所採用的 CAE 模擬軟體是美商綺城科技公司(A.C Technology)所發展的軟體: Moldflow 塑膠模流分析之 MPI (Moldflow Plastic Insight)。MPI 是一套整合高分子射出成型專門之應用軟體，可分析流動情形、溫度分佈、鎖模力、縫合線、包封位置、剪應力、剪應變、翹曲及收縮分析等，是一直接整合至 CAD 使用環境之完整的模擬工具，可迅速產生澆口及流道系統，快速評估塑料流動及成品品質特性。使得使用者可在短時間內開發新產品且降低不良率、減少成本支出及製造品質優良之產品。

射出成型製程之電腦輔助設計分析可分為三大階段: 前置處理階段(Preprocessing)、模擬分析階段(Simulation)、後段處理階段(Postprocessing)(陳良相 et al., 2005)。如圖 3.1 所示。接下來的部份將針對各個階段設定進行說明。

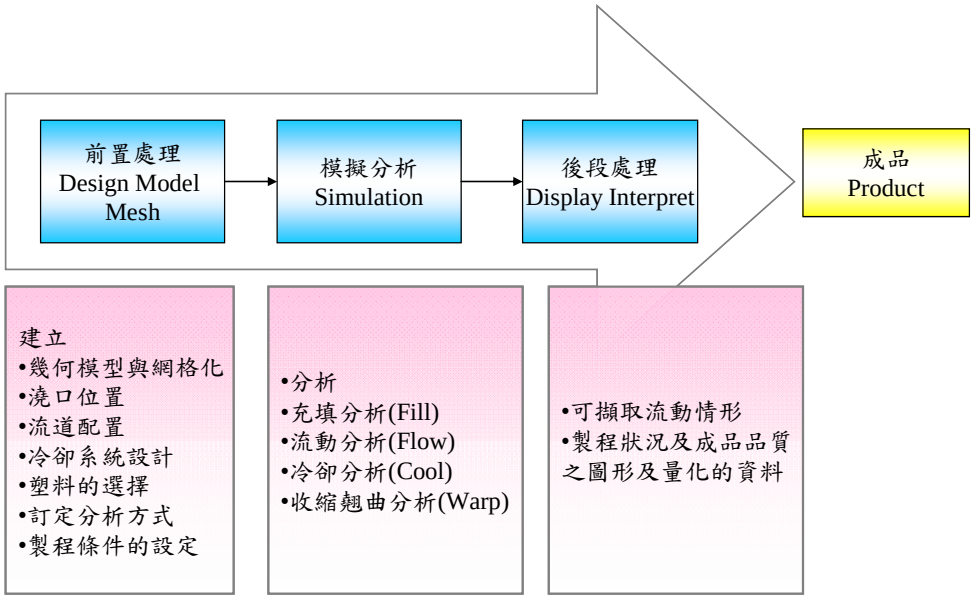


圖 3.1、塑膠射出成型 CAE 分析之三大階段

3.1.2 前置處理(Preprocessing)

前置處理階段為塑膠射出成型 CAE 分析中最重要的一個環節，成品塑件的模擬分析設定都在此時完成，若此階段發生差錯，則所有分析結果將不具任何意義。此階段主要之目的為建立幾何模型及建立網格、決定澆口位置、設定流道配置、設計冷卻系統、選擇塑膠射出材料、訂定欲採取之分析模式、製程條件設定等(陳良相 et al., 2005)。以下將針對本研究之模型建立、網格設定、材料選擇、模擬設定等前置階段的設定進行詳細說明。

3.1.2.1 模型建立

CAE 模擬軟體的使用方法中，通常幾何模型可以透過 Pro-Engineer、AutoCAD、SolidWork、Rhion 等製圖工具進行繪圖，再將檔案轉換成 STL、STP 或 IGES 等的檔案格式便可匯入 CAE 模擬工具進行設定及分析。

本研究採用 Moldflow MPI 軟體內建之手機薄殼模型如圖 3.2 所示。

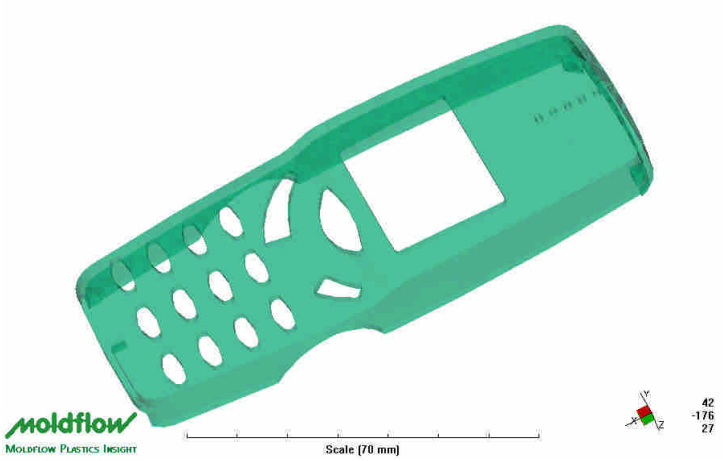


圖 3.2、本研究採用之手機薄殼模型

3.1.2.2 網格建立

網格建立是將初始建構之模型切割成元素(Element)與節點(Node)的組合，透過每一單位網格間之連接，將元素間的關係、材料屬性、流體力學等進行充填、保壓、冷卻和翹曲分析。

本研究利用 Moldflow 軟體內建之網格建立工具進行網格建立及修正，採用三角模式之網格，如圖 3.3 所示。透過符合網格(Match mesh)與網格平滑化(Smooth mesh)之網格的處理方法以達到較好的網格品質。並透過 Match ratio 作為檢視網格品質的重要指標。其中，若要進行流動分析，Match ratio 最少要到達 85%以上。若想得到較準確的翹曲結果，Match ration 最好在 90%以上。本研究之網格品質指標如圖 3.4 所示。

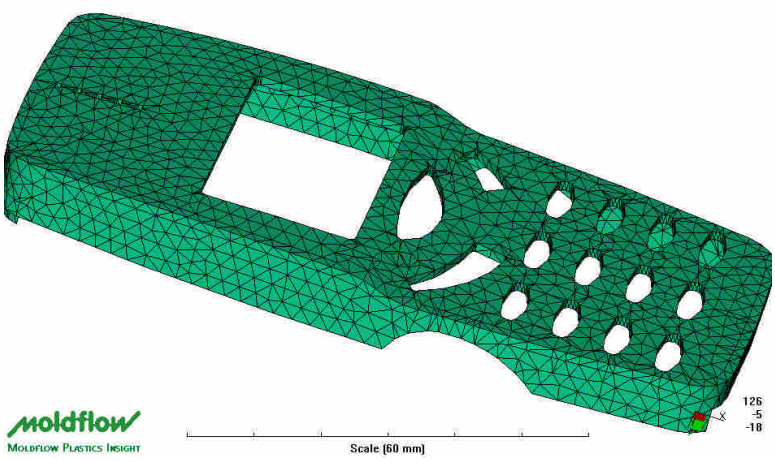


圖 3. 3、本研究模型之網格圖示

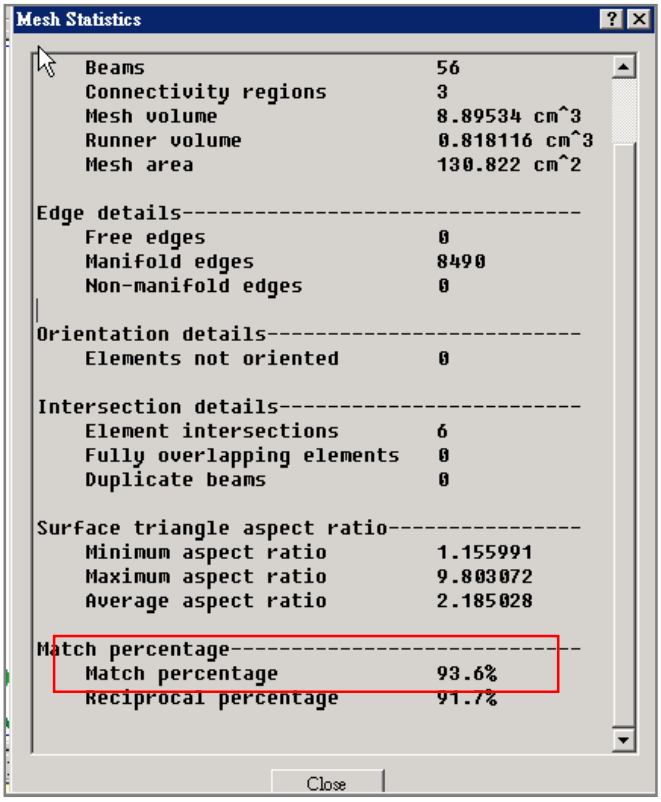


圖 3.4、本研究模型之詳細網格檢定

3.1.2.3 澆口位置及澆流道系統設置

模具之設定為一模一穴，進澆位置透過 Moldflow MPI 之 Gate Location 分析形式進行實驗分析，找出此手機模型之最佳澆口位置為 Node 369，如圖 3.5 所示。

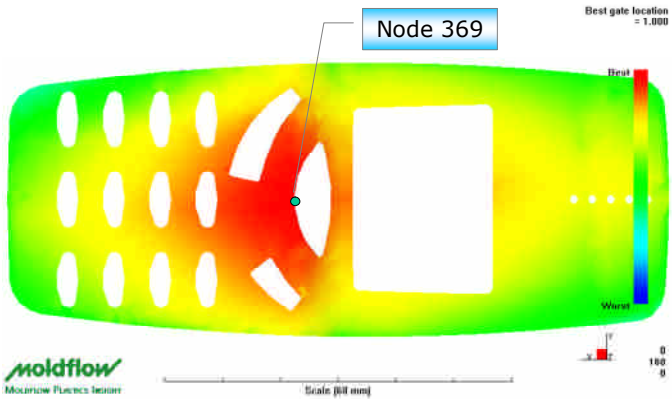


圖 3.5、Moldflow 建議之最佳澆口位置

本研究之初始澆流道採用潛狀澆口(Submarine)設定，詳細設計如圖 3.6 ~圖 3.9 所示。潛狀澆口為圓錐形澆口，又稱為隧道澆口或鑿子澆口，可使澆口分離自動化，且進澆位置可自由選擇在塑件表面、側面或裡面，可用於表面不容許留有澆口痕跡之成形品。方式簡單，唯有加工時須考慮死角容許尺寸問題。

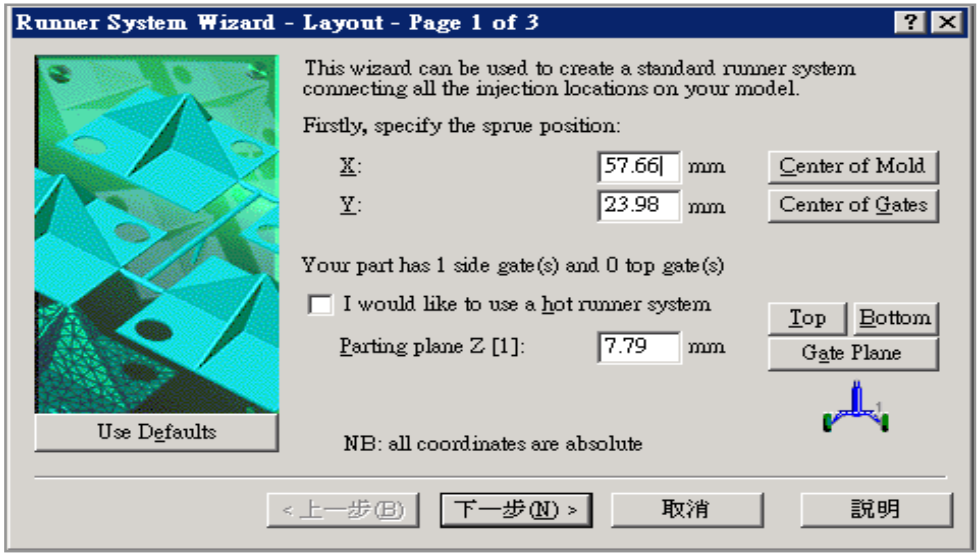


圖 3.6、本研究模型之澆流道設定-1

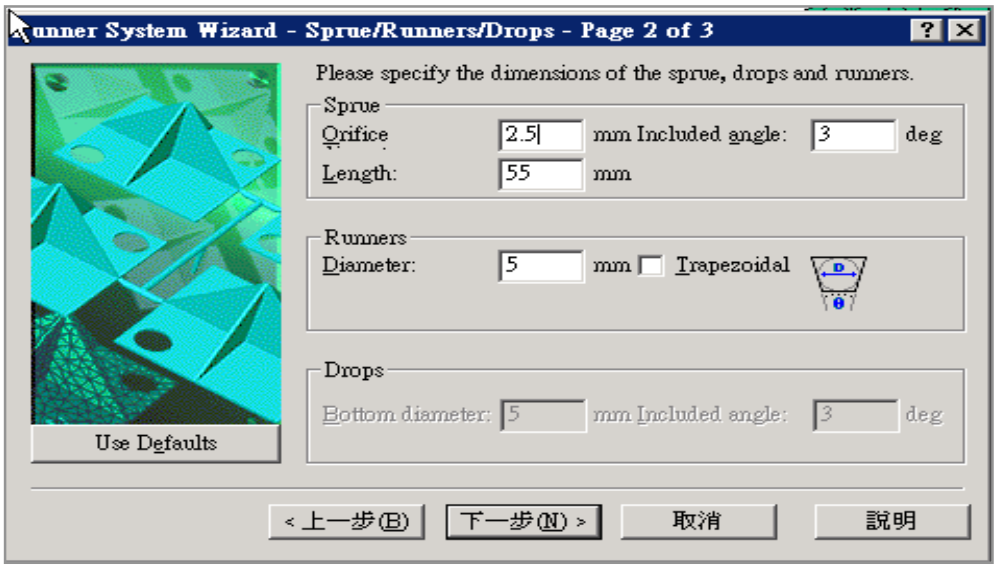


圖 3.7、本研究模型之澆流道設定-2

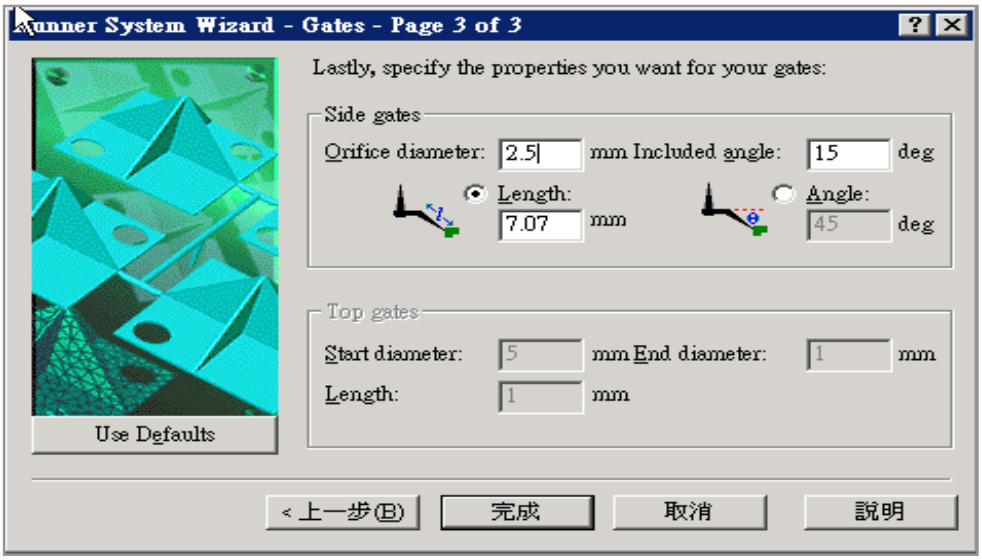


圖 3.8、本研究模型之澆流道設定-3

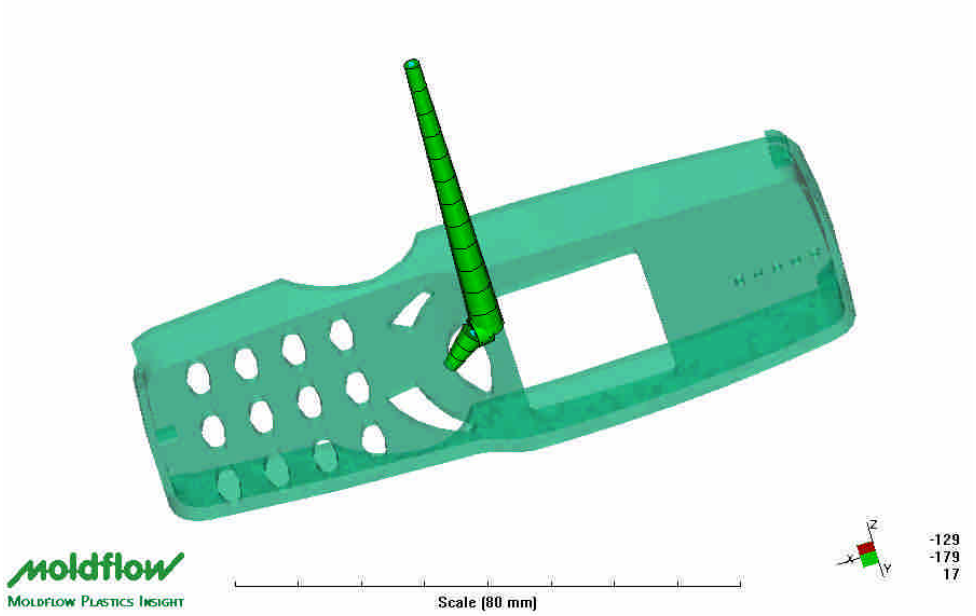


圖 3.9、本研究模型之澆流道圖示

完成澆流道系統之建立後，下一步則是建立模穴冷卻系統，冷卻系統對塑件的影響非常大，冷卻的好壞將直接影響塑件的表面質量、機械性能和結晶度。且冷卻時間也決定了塑件成形周期的長短，直接影響產品的成本。

本研究採用冷卻水作為降低模型溫度之冷卻液，將模具內熱量有效的減少。冷卻系統之詳細設定如圖 3.10、圖 3.11 所示。建置完成後之冷卻流道系統如圖 3.12 所示。

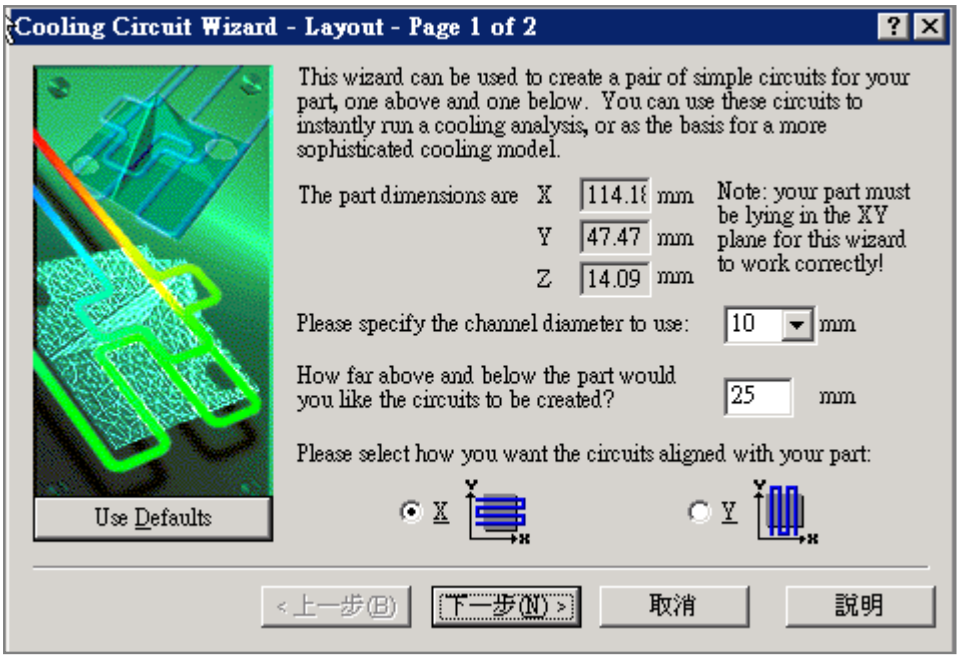


圖 3.10、本研究模型之冷卻系統設計-1

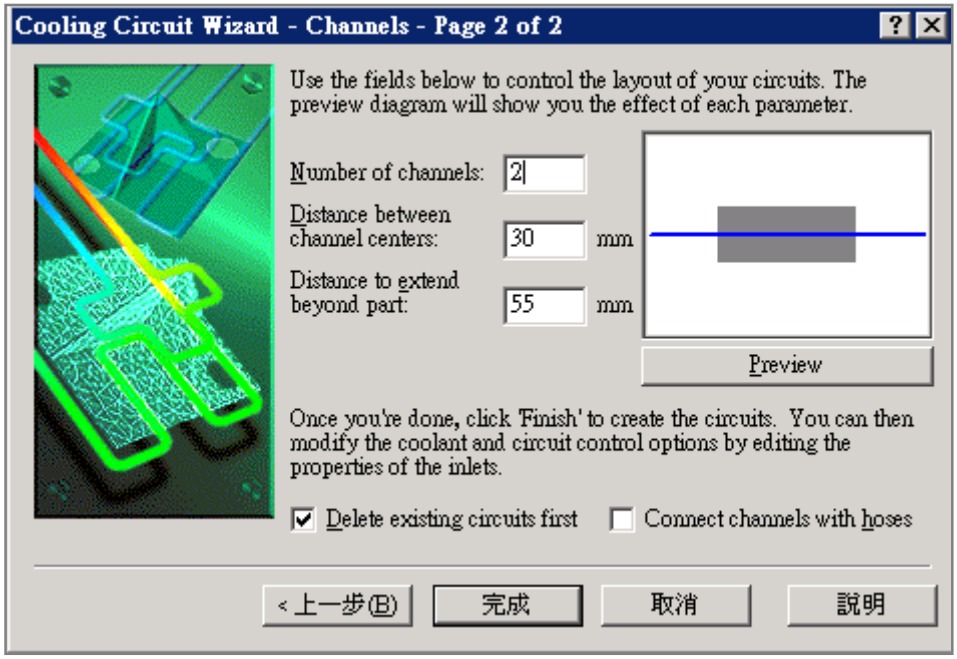


圖 3.11、本研究模型之冷卻系統設計-2

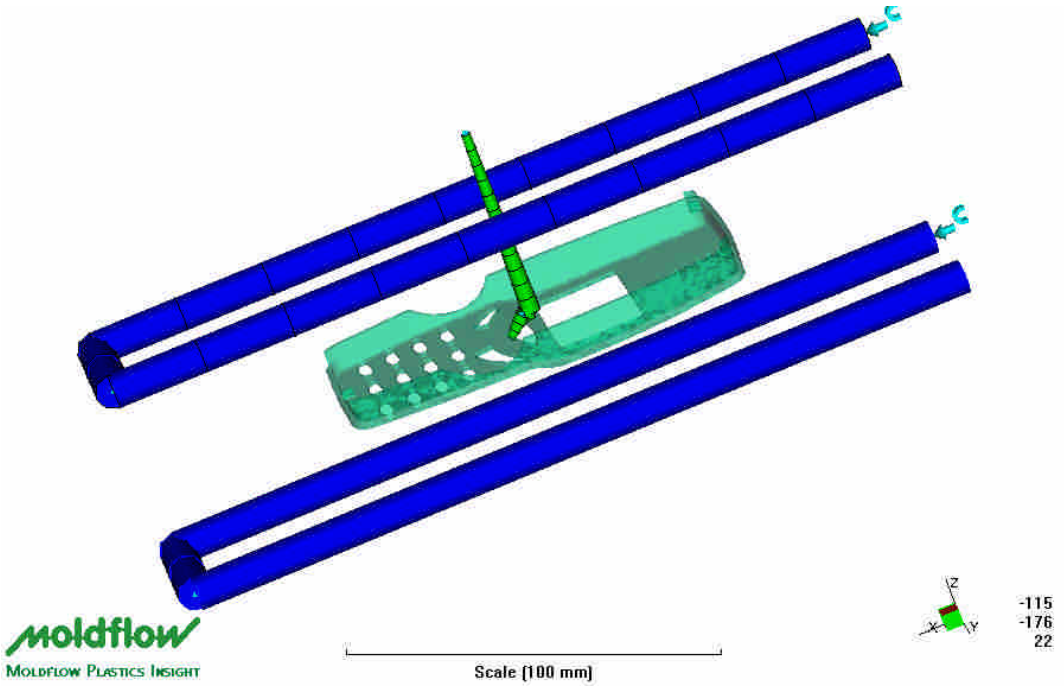


圖 3.12、本研究成品之冷卻系統

3.1.2.4 材料選擇

材料選擇的部份，將針對塑膠材料及模具的選擇進行說明，本研究選用之塑膠材料為美國奇異公司所生產之型號 GE C1000HF PC/ABS 之塑料，模具材料則採用 TOOL STEEL P-20。

目前，手機常用塑膠材料主要有 PC、ABS 和 PC/ABS 三大類。由於 PC/ABS 具 PC、ABS 材料之特點，具有優良的成型加工性能、流動性好、強度較高(抗拉伸強度 56MPa，抗彎曲強度 86MPa)等特性，如表 3.1。常用於直板機外殼和一般外觀、色彩要求高而對環境無特殊要求的摺疊機外殼。故本研究採用 PC/ABS 塑料進行分析，以期更貼近實際市場之需求。

表 3.1、PC、ABS、PC/ABS 材料特性表

材料	材料強度	硬度	溫度	其它特性
PC	69 MPa	120 R	130 °C	無毒、透光、低温特性較好
ABS	43 MPa	100 R	60 °C	配色、噴塗、電鍍等特性好
PC+ABS	56 MPa	110 R	60 °C	各項性能居中，加工、流動性好

PC/ABS 材料建議之射出成型製程條件為熔劑溫度於 230˚C ~310˚C、

模具溫度於 50˚C ~110˚C、注射速度越高越好…等。本研究採用 PC/ABS 塑料之詳細物理特性及製造特性如圖 3.13、圖 3.14，壓力比容溫度(PVT)資料如圖 3.15 所示，模具材料製造特性如圖 3.16。其中， PVT 圖是指塑膠材料隨著壓力及溫度的變化而發生體積上的變化。在充填及保壓階段時，塑膠材料會隨著壓力的增加而膨脹，而在冷卻的階段時，塑膠隨著溫度的降低而收縮。

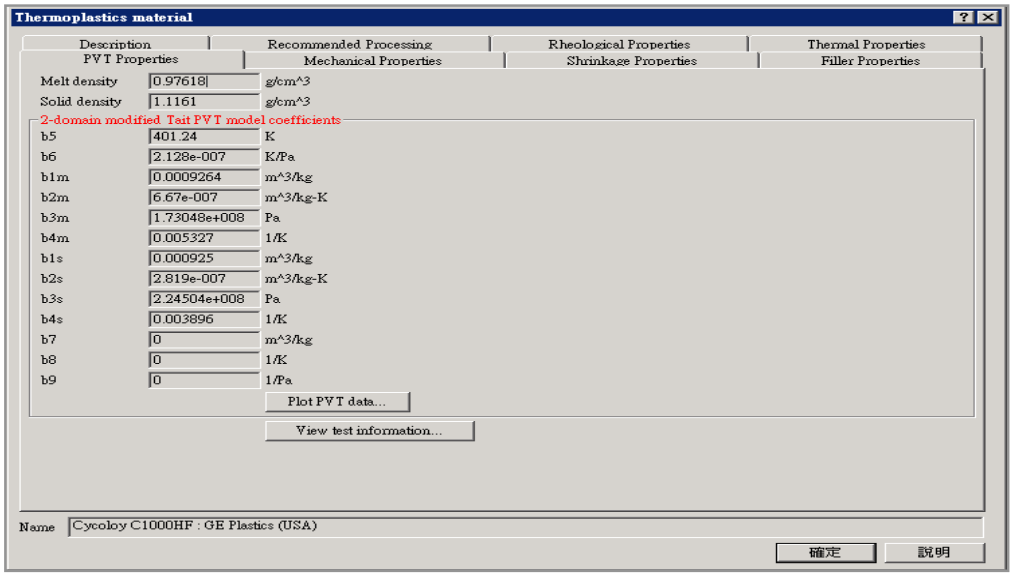


圖 3.13、本研究塑料之詳細物理特性資料

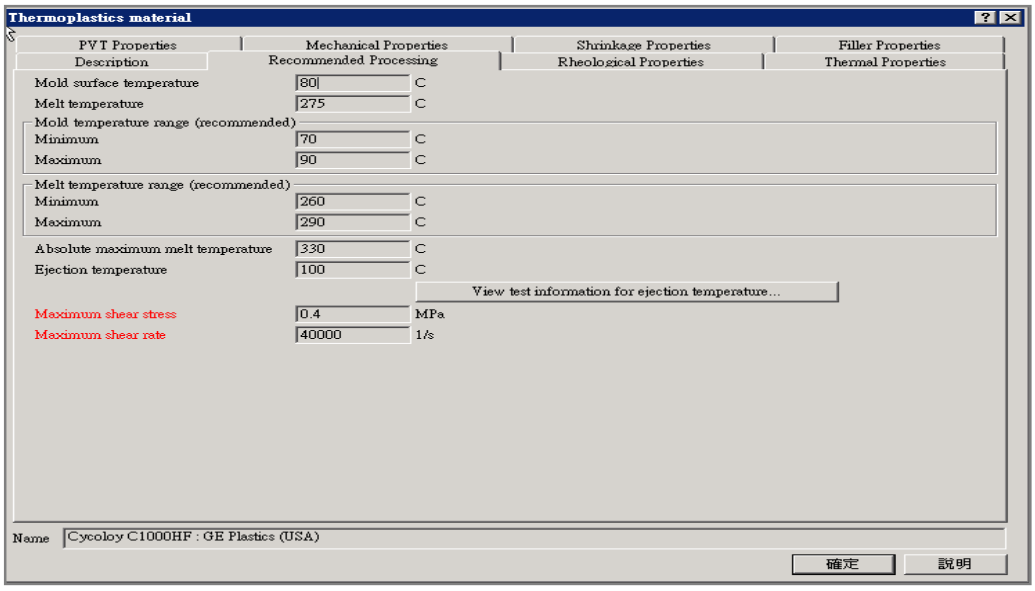


圖 3.14、本研究塑料之製造特性

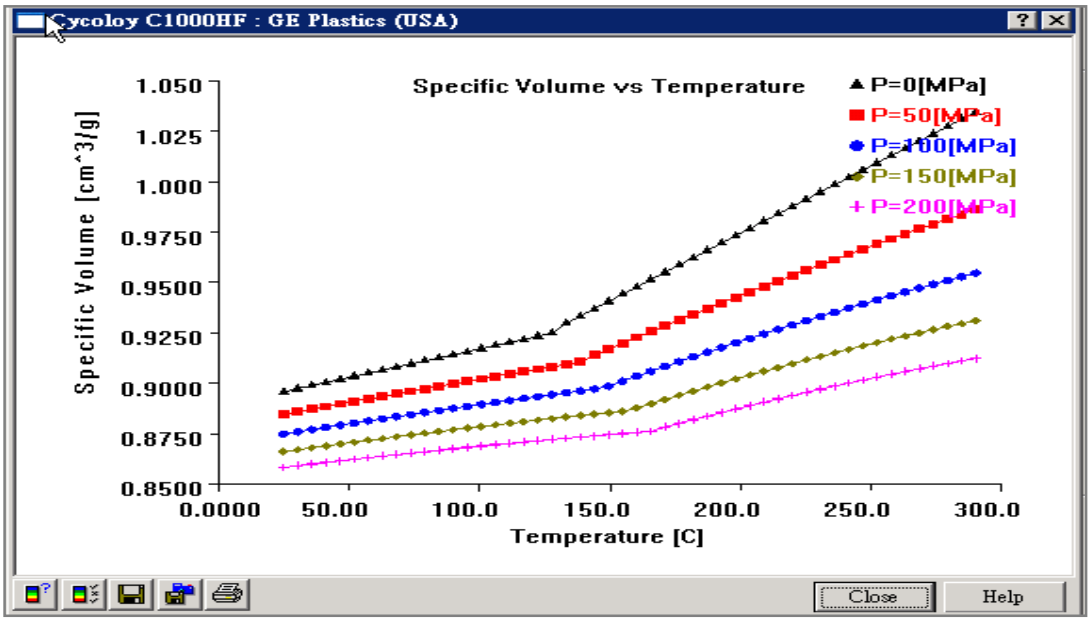


圖 3.15、本研究塑料之 PVT 圖

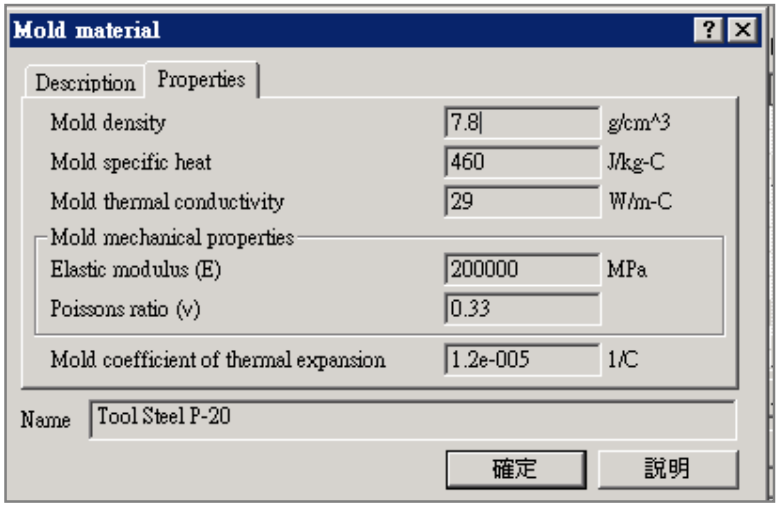


圖 3.16、本研究模具材料之製造特性資料

3.1.2.5 模擬設定

完成模型建立及機器參數設置之後，前置階段的最後步驟即為模擬設定，進行模擬製造前必須先設定製程參數。本研究將針對注射時間(Injection Time)、注射壓力(Injection Pressure)、保壓壓力(Packing Pressure)、保壓時間(Packing Time)、冷卻時間(Cooling Time)、冷卻液溫度(Coolant Temperature)、開模時間(Mold-Open Time)、溶劑溫度(Melt Temperature)、模穴溫度(Mold Surface Temperature)等九個製程參數進行研究分析，期望找出塑膠射出成型製程中，使得翹曲及收縮品質特性最佳之參數組合。製程

參數之初始值以 Moldflow 預設值作為起始設定值，如表 3.2 所示。

表 3.2、本研究之製程參數初始設定值

射出成型參數	射出成型參數	參數簡稱	起始設定值
Injection Time (S)	注射時間	INT (X_1)	**1**
Injection Pressure (MPa)	注射壓力	INP (X_2)	**120**
Packing Pressure (MPa)	保壓壓力	PP (X_3)	**80**
Packing Time (S)	保壓時間	PT (X_4)	**10**
Cooling Time (S)	冷卻時間	COTI (X_5)	**19**
Coolant Temperature (℃)	冷卻溫度	COTE (X_6)	**25**
Mold-Open Time (S)	開模時間	MOO (X_7)	**5**
Melt Temperature (℃)	熔劑溫度	MET (X_8)	**275**
Mold Temperature (℃)	模穴溫度	MOTE (X_9)	**80**

其中，保壓時間為充填時間壓力速度控制切換成保壓壓力控制上的時間；保壓壓力為充填壓力由速度控制切換成保壓壓力控制；冷卻時間為保壓結束後會有一段時間不再對塑件施加壓力，等待塑件凝固直到頂出的時間；開模時間為成形終了取出成形品時，模具打開至成品彈出的時間，成品與模具不再進行熱量的交換；熔劑溫度為熔劑在進入澆流道前，塑膠熔融狀態之溫度值；模穴溫度為當塑料注入模穴時，接觸模穴壁前之模穴表面溫度，此溫度值將會影響冷卻流程之冷卻率。

3.1.3 模擬分析(Simulation)

Moldflow 模流分析軟體中，可依據不同的需求提供不同的分析模組，其中分析形式包括 Flow 流動充填分析、Cool 模具冷卻分析、Warp 翹曲變形分析、Stress 塑膠應力分析、GAS 氣輔中空成形…等。在分析過程中，須先對冷卻、充填及保壓進行最佳化，唯有如此才能獲得準確的翹曲及收縮值。

故本研究選用熱塑性射出模組中的 Cool+Flow+Warp 之連串流程分析形式進行分析。表 3.3 為此三種分析形式可得之分析結果與應用效益。

表 3.3、Cool、Flow、Warp 模組之分析結果及應用效益

	分析結果	應用效益
Cool	1. 計算模具截面溫度分佈，提供冷卻管路配置時參考 2. 計算冷卻管路之溫度、雷諾數、冷劑流率及模壁面之溫度分佈、溫差、凝固時間	1. 設計均勻有效之冷卻系統 2. 減少成品扭曲、改善品質 3. 找出局部熱點或冷點 4. 縮短成型週期
Flow	1. 有限元素數值計算，可提供模穴及澆道之充填時間、溫度分怖、壓力分怖、剪力分佈。 2. 有限差分數值分數，可計算保壓壓力、保壓時間、體積收縮率等。	1. 縮短成型週期及交貨時間 2. 選擇澆口位置及數目 3. 防止不均勻收縮 4. 降低不良率 5. 流動平衡
Warp	1. 讀取流動、冷卻之分析結果，配合材料物性，計算成品之各方向收縮率、面積收縮率、體積收縮率。 2.計算各方向之翹曲率	1. 模擬可能之翹曲情況 2. 尋找造成翹曲之原因，以降低變形量

(源自林瑞璋, 2001)

透過流動分析結果觀察融膠是否同時到達模穴各個角落；而冷卻分析結果為判斷冷卻系統之冷卻效果，且所得之冷卻時間可用來判斷塑件的成形周期；翹曲分析主要針對模型翹曲及收縮的類型、範圍及原因進行分析。其中塑件體積的收縮與線性收縮受模具的限制、結晶、配向性所影響，而翹曲則與收縮的變異有關(Jay Shoemaker, 2006)。陳良相 et al.(2005)認為最主要影響翹曲產生的因素主要可分為三大類: (1)冷卻性差異(Differential Cooling)、(2)收縮不均(Differential Shrinkage)、(3)配向性影響(Orientation Effects)。

3.1.4 後段處理(Postprocessing)

經過一連串的冷卻、充填、保壓及翹曲分析之後，便完成塑膠成品整體之模擬工作。接著，我們可從 Moldflow 中擷取需要之成品分析的資料，例如: 熔融液流動情形、製程狀況、成品品質翹曲及收縮值…等圖形或量化的資訊。而後段處理階段主要之目的即為取得每一次實驗之製程資訊及成品結果等資料，進行記錄及分析進行回饋，使得成品之資訊及結果可作為下一次生產製造的參考依據。

本研究採用之塑件為手機薄殼模型，對於手機產品汰換率如此高的時代來說，產品的美觀及堅固耐用為消費者的優先考量，手機機殼整體的品質特性也同樣是製造商最關注的目標。因次，本研究將針對手機整體之翹曲及收縮進行分析，透過擷取手機薄殼模型的四個角點之結點(以 N11、N2203、N941、N55 表示)及四個角點所連成的邊長(以 E1、E2、E3、E4 表示)作為翹曲及收縮之分析資料取得考量基準。如圖 3.17。

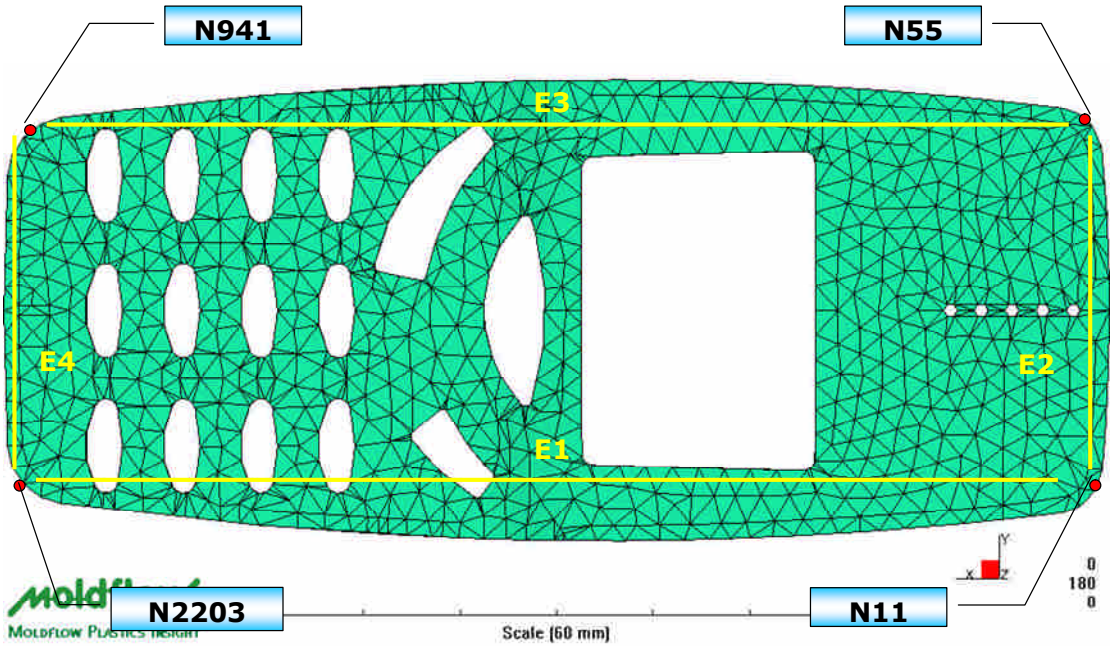


圖 3.17、成品手機模型之製程參數測量點

3.2 田口直交表結合變異數分析

本研究應用田口直交表結合變異數分析工具之目的為找出 CAE 模流分析初始選取之九個製程參數中，何者為手機薄殼射出成型製程中之顯著參數。田口品質方法的特色在於利用直交表規劃實驗以進行數據分析。使

用直交表設計實驗可以使設計者以快速且經濟的方式，同時研究多個可控制因子對於品質特性之平均值及變異數的影響(楊景程, 2000)。本研究之田口直交表結合變異數分析實驗步驟包含以下幾個步驟:

1. 選定品質特性:

品質特性會直接影響產品品質。從過去的文獻中，Liao et al.(2004)進行研究之後發現收縮及翹曲為影響塑膠射出成品之重要影響因素。故本研究選用翹曲及收縮作為欲分析之品質特性。

2. 選擇控制因子:

在此步驟中，可由設計者自由設定及選擇控制因子的水準，並可藉由調整這些控制因子的水準，找出最佳的參數組合，使品質特性符合產品要求。接下來，透過控制因子的數目及水準選出合適之直交表，並安排完整的實驗。

3. 執行實驗:

設計者依照直交表上各因子之水準組合，透過 Moldflow CAE 模流分析軟體進行完整之模擬分析，取得實驗之翹曲及收縮數據。

4. 變異數分析:

變異數分析的主要目的為應用統計檢定方式來辨別各別因素之影響效果，彌補田口品質方法無法研判各實驗參數對品質特性之影響能力及誤差程度。此階段的目的在於找出對於品質特性翹曲及收縮有顯著影響之製程參數。

3.3 類神經網路

3.3.1 類神經網路原理

3.3.1.1 生物神經網路

類神經網路為模仿人類大腦思考與判斷所建構而成的一個由許多神經元所組成之網路，透過適當的學習與訓練之後，便可建立出高準確之輸入與輸出之間的關係。(林啟淶, 2005)。

類神經網路透過模仿生物神經網路作為資訊處理的系統。因此，必須對於生物神經網路有基本的瞭解才可能清楚類神經網路的原理與概念。人類的大腦是由以下幾種神經元互相連結成一個具有高度非線性且具有平形處理資訊能力之神經網路，可分為以下幾個重要的部份，如圖 3.18 (Negnevitsky, 2005):

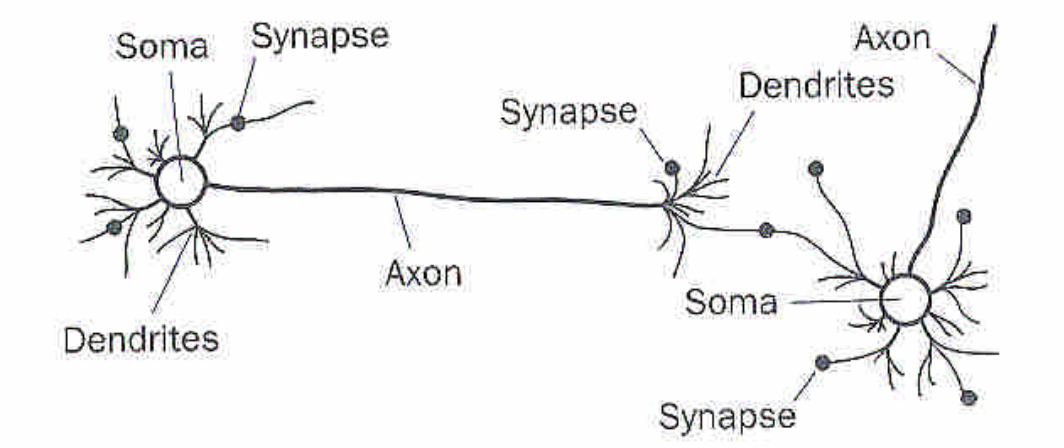


圖 3.18、生物神經元之示意圖

1. 神經細胞核(Soma): 神經細胞的中心體
2. 神經軸(Axon): 連接在神經細胞核上，用來傳送由神經細胞核產生的信號至其他的神經細胞中
3. 神經樹(Dendrites): 神經樹分為輸入神經樹與輸出神經樹
   (1) 輸入神經樹: 用以接收其他神經細胞傳來的信號
   (2) 輸出神經樹: 用以傳送信號至其他神經細胞

4. 神經節(Synapse)

(1) 輸入神經樹與輸出神經樹相連接的點稱為神經節

(2) 表示兩個神經細胞間的聯結強度，以一數值來表示，並稱之為加權值(Weight)。(王進德, 2006)

一般而言，當神經網路在進行學習時，外界刺激神經細胞所產生的電流會去改變神經節上的加權值，在學習的過程中，外界刺激所產生的電流反覆在神經網路上流動，神經節上的加權值也反覆地改變，最後會慢慢趨向穩定，此時表示學習即完成。

3.3.1.2 人工神經元之模型

以人工神經元作為生物神經細胞的簡單模擬，透過外界環境或輸入神經元取得資訊，神經細胞核會將由其他神經細胞輸入的脈波訊號進行收集並做加總，再做一次非線性轉換後，產生一個新的脈波訊號。如果此訊號夠強，則新的脈波訊號會由神經軸傳送到輸出神經樹，再透過神經節將此訊號傳給其他神經細胞如圖 3.19。

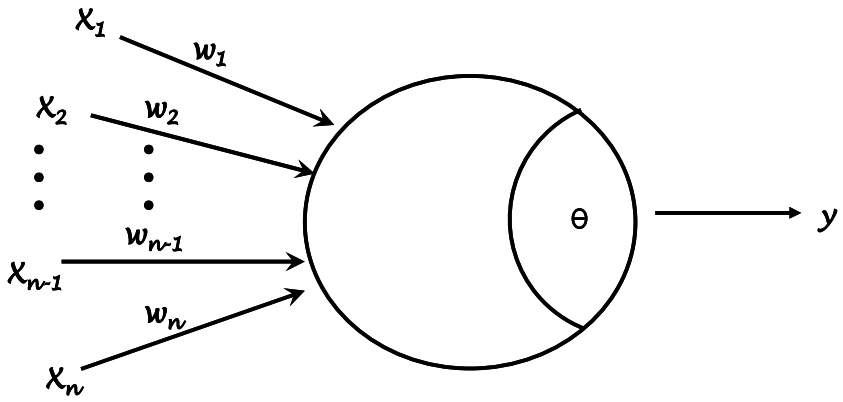


圖 3.19、人工神經元模型

每個人工神經元有多個輸入 $X_1, X_2, \ldots, X_n$ 及一個輸出 y，輸入值與輸出值的關係式，一般可用輸入值的加權乘積和的函數來表示(王進德, 2006)，即

$$y(t) = f(\sum_{i=1}^{n} w_i \cdot x_i(t) - \theta)$$

其中

w_i :模仿生物神經細胞的神經節加權值

θ :模仿生物神經細胞的細胞核偏權值(bias),即輸入訊號的加權乘積和必須要大於偏權值後，才能被傳輸至其他人工神經元中。

f(θ) :模仿生物神經細胞的細胞核非線性轉換函數

t :時間

n :人工神經元輸入數目

3.3.2 類神經網路之基本架構

類神經網路基本架構可分為兩大類: 回饋式網路(Recurrent network)及前饋式網路(Feed-forward network)。其中，回饋式網路最具代表性的是霍普菲爾(Hopfield network)，而前饋式網路為一種階層式的網路，最具代表性的則為倒傳遞類神經網路。

倒傳遞類神經網路的架構概分為三層: 輸入層、隱藏層與輸出層。其中，輸入層負責接收外界的資訊或變數，以線性轉換的方式轉換後再交由隱藏層處理；而隱藏層使用非線性轉換的方式處理資訊或變數間的相互關係。隱藏層可以視需求或設計，分為單層或多層式；輸出層則是將隱藏層處理後的資訊以線性轉換的方式輸出。

本研究以倒傳遞神經網路(Back-Propagation Neural Network, BPNN)作為實驗之方法，此網路為一種具有學習能力的多層前饋型網路。其學習過程事實上是一種監督式的學習過程，透過從問題領域中取得訓練範例及目標輸出值，然後將這些訓練範例當作網路的輸入，利用最陡坡降法(The Gradient descent method)反覆的調整網路連結的加權值。其中，修正網路加權值及偏權值的方法，是將目標輸出值與網路的推論輸出值之間的誤差，一邊向後傳播一邊加以修正。當網路訓練完成後將進行測試，網路會接受外來輸入，並依回想演算法經反覆運算後，由輸出神經元將結果送出，為一種「分類」或「預測」的過程。(王進德, 2006)

倒傳遞網路中的神經元最常使用的非線性函數為雙彎曲函數(Sigmoide

Function)，如圖 3.20，此函數的特性為當 x 趨近於正負無窮大時，f(x)趨近於 0 或 1，而 f(x)的值介於(0,1)之間:

$$f(x)=\frac{1}{1+e^{-x}}$$

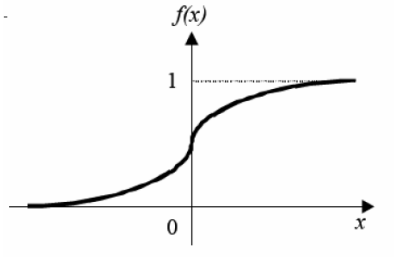


圖 3.20、雙彎曲函數

類神經網路處理資料時，須先將資料分成訓練組(Training set)與測試組(Testing set)，訓練目的在決定網路的權重值與偏權值；而測試的目的在於判斷網路的預測輸出值或驗證網路的準確度

首先將訓練組的資料匯入輸入節點後，透過學習訓練法則調整各個節點的權重比值，讓類神經網路的計算輸出能夠符合所設定的要求並且大於門檻值。接著，利用測試組的資料驗證模型的準確度，並可利用此一模型進行預測的任務。

3.3.3 類神經網路的學習訓練法則

當網路有 1 層輸入層 N 個神經元數目、1 層隱藏層 H 個神經元數目、1 層輸出層 O 個神經元數目時。網路中，每個神經元接收前一階層之輸入值如公式(1)、輸出值如公式(2) 所示:

$$net_j=\sum_{j=0}^{N} w_{ij}x_i \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(1)$$

$$\mathrm{Out}_j=f(net_j)=\frac{1}{1+e^{net_j}} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(2)$$

其中

net_j :整體或淨輸入值

N :至隱藏層之輸入神經元數目

w_{ij} :第i個神經元連結至第j個隱藏層之神經元的權重

x_i :從前一層之第i個神經元之輸入值

Out_j :網路訓練過程之第j個神經元之輸出值

f :雙彎曲函數

接著，以最陡坡降法(The Gradient Descent Method)，透過不斷的改變權重，直到使得預測值與訓練值之間的均方誤差值(Mean Square error, MSE)達到最小，利用公式(3)、(4)找出最佳的權重組合:

$$W_{ij}^{new} = W_{ij}^{old} + \Delta W_{ij} \quad \cdots\cdots(3)$$

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} out_j \quad \cdots\cdots(4)$$

其中

E為MSE

η為學習率

3.3.4 類神經網路之特性及優點

1. **平行處理:** 採大量平行計算。類神經網路在運作時具有很高的錯誤容忍度，如果輸入資料混雜少許的雜訊干擾，仍然不會影響其運作的正確性。而且即使有部分人工神經元失效，整個類神經網路仍能有效運作。
2. **聯想式記憶:** 在針對輸入訊號進行運算時，整個網路藉由運算過程可聯想出相對應的輸出值，此種記憶方式，我們稱為“聯想式記憶”。
3. **解決最佳化問題:** 可藉助類神經網路解決某一問題領域中的最佳化問題。我們期望找出在一滿足設計限制下的設計變數，並使得設計目標達到最佳化。
4. **過濾功能:** 在神經網路中某一個輸入與某一個輸出關係，並不是直接由

網路中某個節點所單獨負責。事實上，每一個節點只會映射出輸入-輸出模式的一個特徵值(Micro Feature)。當網路所有特徵組合在一起時，才能呈現完整的輸入-輸出模式。因此當某一個節點所輸入待處理信號有雜訊或數據不完整時，此一輸入對網路所造成的影響，將不會如想像中那麼嚴重。

5. **適應性學習能力:** 每種類型的類神經網路都有特定的學習演算法，經由演算法可調整節點與節點之間的連結權重值。透過不斷調整權重值而得到正確的輸入-輸出模式，這種能力稱為適應性學習能力。
6. **多輸入與多輸出系統:** 輸入輸出層可具有任意的節點數目。

根據過去文獻，可得知類神經網路為一個具有多項優點之可解決多重輸入多重輸出問題的方法，透過平行處理以及適應性學習能力可以快速的找出參數與品質特性之間輸入及輸出的關係，並且擁有一定程度的錯誤容忍度。我們藉由訓練出此一準確的類神經網路函數，可以快速的預測參數組合對應之品質特性值。希望透過與基因演算法可搜尋出最佳解之特性結合，以快速的找出射出成型製程的最佳參數組合。

3.4 基因演算法

基因演算法為近年來成功應用於各種領域之最佳解搜尋方法，其理論是源自於自然生物界的「適者生存，不適者淘汰」的自然選擇及自然遺傳的搜尋法則。根據達爾文的「物競天擇」理論，若物種越能適應環境則生存的機會就越高，透過基因的複製(Reproduction)、交配(Crossover)、突變(Mutation)等演化，最後只會留下最適合的物種。而基因演算法則是仿造這樣的理論進行最佳解的搜尋。透過對於欲研究之目標的參數，進行複製、交配及突變的演化，找出一組適應函數最好的解，即達到參數最佳化。

3.4.1 基因演算法原理

基因演算法透過物競天擇、優勝劣敗的自然進化法則，選擇物種中具優良特性的母代(Parent Generation)，透過隨機交換彼此的基因以產生更優秀的子代，子代成為下一母代，經由基因演算法再找出下一子代，如此重覆下去，直到找出適應性最佳的族群數為止。

應用基因演算法時，為便於交配與突變之進行，通常以二進位的方式進行運算，運算的方法為隨機選取參數搜尋範圍中的各參數值編碼成二進位字串，本研究將參數編碼成字串長度為 10 之二進位字串，並隨機產生 12 個成員，形成母代。接著透過基因演算法中主要的三種運算: 複製(Reproduction)、交配(Crossover)，以及突變(Mutation)進行最佳基因的搜尋。

1. **複製運算:** 本研究在此採用競爭式選擇，在每一代的演化過程，隨機選擇抓出兩個或多個基因字串，進行適應函數的計算，具有最大適應函數值的基因即會被選中送到交配池。
2. **交配運算:** 隨機選取族群中的 2 個基因字串，彼此交換位元資訊，進而組成另外兩個新的基因字串。本研究以單點交配方法作為基因交配的選擇，隨機選取兩個基因字串中的交配點，再交換兩字串中此交配點後的所有位元。
3. **突變運算:** 隨機的選取某一基因字串並隨機選取突變點，再改變基因字串中的位元資訊，本研究採用將位元反相，即將位元由 0 變 1，1 變 0 進行突變。

當搜尋出目標之最佳基因字串後，這些字串為對應參數區間範圍之某一實數值。接著須將此二進位數值轉換至十進位數值，轉換方法如下: 首先，我們先將基因字串表示為 $S_{N-1}S_{N-2}\cdots S_1S_0$ ，S_0 為最小的位元，S_{N-1} 為最大的位元。接著將基因字串利用公式 $m=\sum_{i=0}^{N-1}s_i 2^i$ 轉成十進位，但此時之十進位數值不一定會在參數的限制範圍內，所以我們再依公式 $x=a+m\frac{b-a}{2^N-1}$ 將十進位數值轉換成[a,b]之間的某一數值。經過適當的轉換之後 x=a 時，基因字串會表示成”000…0000”，當 x=b 時基因字串會表示成”111…1111”。

3.4.2 適應值原理

求解最佳化問題時，我們需要設定一個指標來判斷結果是否已達可接受之最佳解。因此，在基因演算法過程中需事先定義一個適應函數，以此適應函數與最佳化問題中的目標函數相對應作為最佳化程度的指標。而適應函數的建構需依照問題的特性而有所調整。當最佳化問題之目標函數為求解最大值問題時，適應函數即為目標值；而當目標函數為最小值問題時，需適當的調整適應函數，將某個適當正數減去目標函數值作為適應函數，以滿足基因演算法中進化的過程恆向極大值逼進的需求，而適當正數的選取必須滿足所計算的適應值恆大於或等於零。而透過適應函數計算所得之適應值(Fittness)則為基因演算法中用以描述個體優劣程度的指標，適應值越大則此個體越趨近於最佳解。本研究為搜尋最小翹曲及收縮值之問題，且在運算的過程中將資料皆正規化至 0~1 區間之數值，因此在此將適當正數設為 1 以調整適應函數，以適應值作為複製、交配、突變過程淘汰或選擇的標準。如圖 3.21，為基因演算法演化之流程。

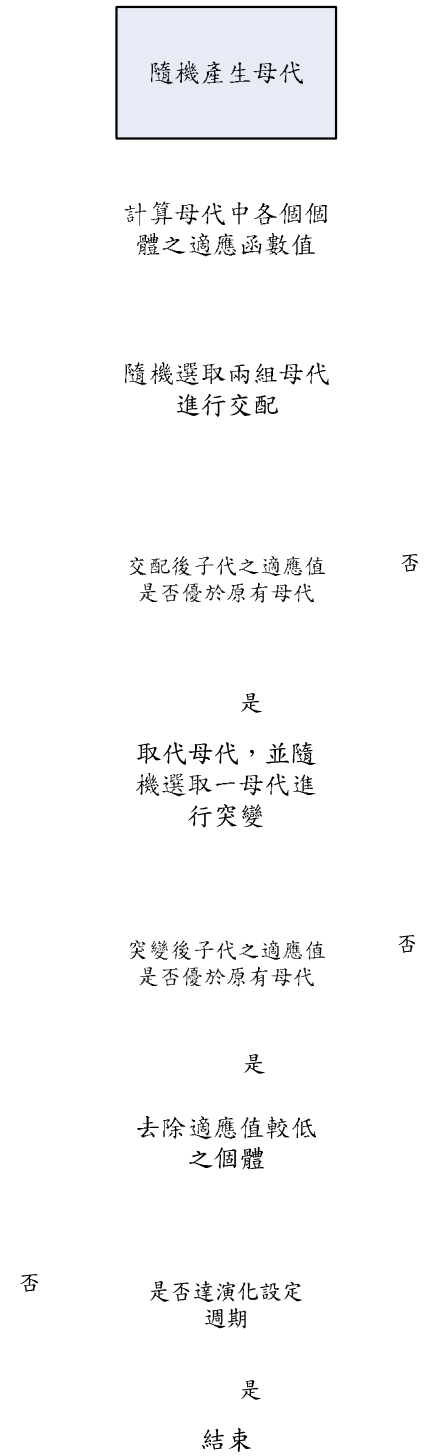


圖 3.21、基因演算法之演化步驟

3.4.3 定義目標函數及適應函數

本研究欲搜尋翹曲及收縮最小值之最佳化參數組合，但由於基因演算法僅能針對單一目標進行搜尋，且過去文獻中曾採用將多目標問題給予權重的方式，並無一法則或統計理論給予我們權重的分配規則，故給予權重並非我們所期望得到之結果。因此，第一階段將先針對目標函數為最小收縮值 S=f(INT, INP, PP, PT, COTI, COTE, MOO, MET, MOTE) 進行基因演算法的搜尋，找出收縮最小值及翹曲最小之最佳解。接著在第二階段中，我們放寬一個範圍的收縮值並以此範圍作為限制條件進行基因演算法運算以搜尋最佳化翹曲及收縮之組合。以下將定義出此步驟中，目標函數、適應函數及限制條件之呈現方式。

目標函數： S=f(INT, INP, PP, PT, COTI, COTE, MOO, MET, MOTE)

適應函數： F =1-f

限制條件：

$$INT_{\min} \le INT \le INT_{\max}$$
$$INP_{\min} \le INP \le INP_{\max}$$
$$PP_{\min} \le PP \le PP_{\max}$$
$$PT_{\min} \le PT \le PT_{\max}$$
$$COTI_{\min} \le COTI \le COTI_{\max}$$
$$COTE_{\min} \le COTE \le COTE_{\max}$$
$$MOO_{\min} \le MOO \le MOO_{\max}$$
$$MET_{\min} \le MET \le MET_{\max}$$
$$MOTE_{\min} \le MOTE \le MOTE_{\max}$$

3.4.4 基因演算法之主要特性

1. 基因演算法是以參數集合的編碼進行運算，而不是參數本身，所以可以跳脫搜尋空間分析上的限制。
2. 基因演算法同時考慮搜尋空間上的多個點，而不限於單一個點，因此可以較有機會獲得整體最佳解，避免陷入區域最佳解的困境，此項特性也是基因演算法的最大優點。
3. 基因演算法只用適應函數的資訊來求解，而不需要其他的輔助資訊，如梯度值，所以可以避免複雜的數學運算。

透過 moldflow 實驗之參數與相對應塑件品質之數據，以類神經網路訓練的過程中，我們可取得不同的權重值與偏權值組合，此權重值及偏權值即為倒傳遞類神經網路裡決定輸出結果之關鍵，透過輸入欲預測之參數組合便可得知塑件之品質結果。透過這樣的輸入輸出關係，將此權重值及偏權值寫入基因演算法程式語法中，在進行基因演算法搜尋過程時，當演化出不同參數組合時即可得相對應品質特性值，如此反覆的進行最小化適應函數 之演化過程，便可以達到快速的進行最佳品質特性的搜尋。

第四章、實驗模擬與分析

4.1 田口直交表結合變異數分析

4.1.1 選定欲研究之製程參數及品質特性

本研究選用 Moldflow 軟體內建之手機薄殼模型、塑膠材料 PC/ABS、模具材料 TOOL STEEL P-20 進行塑膠射出成型最佳化收縮及翹曲值之分析。首先，我們選取射出成型製程中重要的九個製程參數及兩個品質特性進行 Moldflow 之模擬實驗，如表 4.1。接著，將實驗數據利用變異數分析找出與品質特性具顯著相關之參數進行後續的最佳化分析，期望找出使得翹曲及收縮最小的參數組合。其中，實驗數值的測量透過擷取分析完成之手機薄殼模型的四個角點之結點值(以 N11、N2203、N941、N55 表示)及四個角點所連成的邊長(以 E1、E2、E3、E4 表示)作為翹曲及收縮之實驗數據。因此，測量結果包含四個翹曲測量值及四個收縮測量值。

表 4.1、本研究選用之製程參數及品質特性

射出成型參數	射出成型參數	參數代碼
Injection Time	注射時間	INT
Injection Pressure	注射壓力	INP
Packing Pressure	保壓壓力	PP
Packing Time	保壓時間	PT
Cooling Time	冷卻時間	COTI
Coolant Temperature	冷卻溫度	COTE
Mold-Open Time	開模時間	MOO
Melt Temperature	熔劑溫度	MET
Mold Temperature	模穴溫度	MOTE
品質特性	**品質特性**	**特性代碼**
Warpage	翹曲	N11, N2203, N55, N941
Shrinkage	收縮	E1, E2, E3, E4

4.1.2 選定因子水準

選定欲研究之製程參數及品質特性之後，將根據 Moldflow 之參數預設值訂定各參數之因子水準，如表 4.2。本實驗中共有九個實驗因子，設定每個實驗因子具有三個水準。因此本研究選用 L_{27} 直交表，如表 4.3 所示。

表 4.2、本研究之各參數因子水準

射出成型參數	射出成型參數	參數代碼(變數)	L1	L2	L3
Injection Time (S)	注射時間	INT (X_1)	0.7	**1**	1.3
Injection Pressure (MPa)	注射壓力	INP (X_2)	84	**120**	156
Packing Pressure (MPa)	保壓壓力	PP (X_3)	56	**80**	104
Packing Time (S)	保壓時間	PT (X_4)	7	**10**	13
Cooling Time (S)	冷卻時間	COTI (X_5)	13.3	**19**	24.7
Coolant Temperature (℃)	冷卻溫度	COTE (X_6)	17.5	**25**	32.5
Mold-Open Time (S)	開模時間	MOO (X_7)	3.5	**5**	6.5
Melt Temperature (℃)	熔劑溫度	MET (X_8)	247.5	**275**	302.5
Mold Temperature (℃)	模穴溫度	MOTE (X_9)	56	**80**	104

由於 PC/ABS 塑膠材料的熔劑溫度建議值介於 230˚C~310˚C 之間，因此，熔劑溫度以參數預設值±10%之界限作為因子水準選取的範圍，而其餘的八個製程參數之 L1 及 L3 因子水準範圍則以參數預設值±30%之界限做為因子水準的範圍。接著，將直交表對應之參數設定值如表 4.4 代入 Moldflow 進行模擬分析，取得翹曲及收縮實驗值。

表 4.3、本研究之 L_{27} 直交表

L_{27}	INT	INP	PP	PT	COTI	COTE	MOO	MET	MOTE
1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	2	2	2	2	2
3	1	1	1	1	3	3	3	3	3
4	1	2	2	2	1	1	1	2	2
5	1	2	2	2	2	2	2	3	3
6	1	2	2	2	3	3	3	1	1
7	1	3	3	3	1	1	1	3	3
8	1	3	3	3	2	2	2	1	1
9	1	3	3	3	3	3	3	2	2
10	2	1	2	3	1	2	3	1	2
11	2	1	2	3	2	3	1	2	3
12	2	1	2	3	3	1	2	3	1
13	2	2	3	1	1	2	3	2	3
14	2	2	3	1	2	3	1	3	1
15	2	2	3	1	3	1	2	1	2
16	2	3	1	2	1	2	3	3	1
17	2	3	1	2	2	3	1	1	2
18	2	3	1	2	3	1	2	2	3
19	3	1	3	2	1	3	2	1	3
20	3	1	3	2	2	1	3	2	1
21	3	1	3	2	3	2	1	3	2
22	3	2	1	3	1	3	2	2	1
23	3	2	1	3	2	1	3	3	2
24	3	2	1	3	3	2	1	1	3
25	3	3	2	1	1	3	2	3	2
26	3	3	2	1	2	1	3	1	3
27	3	3	2	1	3	2	1	2	1

表 4.4、本研究之 L_{27} 直交表對應之參數組合

L_{27}	INT	INP	PP	PT	COTI	COTE	MOO	MET	MOTE
1	0.7	84	56	7	13.3	17.5	3.5	247.5	56
2	0.7	84	56	7	19	25	5	275	80
3	0.7	84	56	7	24.7	32.5	6.5	302.5	104
4	0.7	120	80	10	13.3	17.5	3.5	275	80
5	0.7	120	80	10	19	25	5	302.5	104
6	0.7	120	80	10	24.7	32.5	6.5	247.5	56
7	0.7	156	104	13	13.3	17.5	3.5	302.5	104
8	0.7	156	104	13	19	25	5	247.5	56
9	0.7	156	104	13	24.7	32.5	6.5	275	80
10	1	84	80	13	13.3	25	6.5	247.5	80
11	1	84	80	13	19	32.5	3.5	275	104
12	1	84	80	13	24.7	17.5	5	302.5	56
13	1	120	104	7	13.3	25	6.5	275	104
14	1	120	104	7	19	32.5	3.5	302.5	56
15	1	120	104	7	24.7	17.5	5	247.5	80
16	1	156	56	10	13.3	25	6.5	302.5	56
17	1	156	56	10	19	32.5	3.5	247.5	80
18	1	156	56	10	24.7	17.5	5	275	104
19	1.3	84	104	10	13.3	32.5	5	247.5	104
20	1.3	84	104	10	19	17.5	6.5	275	56
21	1.3	84	104	10	24.7	25	3.5	302.5	80
22	1.3	120	56	13	13.3	32.5	5	275	56
23	1.3	120	56	13	19	17.5	6.5	302.5	80
24	1.3	120	56	13	24.7	25	3.5	247.5	104
25	1.3	156	80	7	13.3	32.5	5	302.5	80
26	1.3	156	80	7	19	17.5	6.5	247.5	104
27	1.3	156	80	7	24.7	25	3.5	275	56

表 4.5、實驗 27 組之翹曲及收縮模擬結果

	N11	N2203	N55	N941	E1	E2	E3	E4
1	0.461	0.4109	0.4506	0.4193	0.7986	0.3997	0.7994	0.8514
2	0.3947	0.3586	0.3805	0.3675	0.6868	0.3453	0.6899	0.7393
3	0.363	0.33	0.3533	0.3375	0.6309	0.3127	0.6323	0.6788
4	0.3055	0.2733	0.2977	0.2794	0.5245	0.2728	0.5262	0.5623
5	0.2931	0.2628	0.284	0.2688	0.5036	0.2618	0.504	0.5439
6	0.334	0.2984	0.3262	0.3044	0.5747	0.2995	0.5757	0.614
7	0.2342	0.205	0.2273	0.2106	0.3937	0.2174	0.3955	0.4296
8	0.2374	0.2006	0.2305	0.206	0.3892	0.2276	0.3891	0.4202
9	0.2387	0.2069	0.2319	0.2123	0.4001	0.2208	0.4012	0.4327
10	0.2559	0.2138	0.248	0.2182	0.4145	0.244	0.4132	0.443
11	0.291	0.258	0.2837	0.2639	0.4923	0.2673	0.4931	0.5269
12	0.2699	0.238	0.2621	0.244	0.4541	0.2523	0.4545	0.489
13	0.2143	0.1798	0.2085	0.1847	0.3449	0.2098	0.3448	0.3725
14	0.2227	0.1921	0.2165	0.1974	0.369	0.2107	0.3695	0.3996
15	0.2135	0.1795	0.2069	0.1825	0.3397	0.2149	0.3375	0.365
16	0.3402	0.3082	0.3325	0.3159	0.5863	0.306	0.5885	0.6302
17	0.4054	0.3563	0.3954	0.3626	0.6868	0.356	0.6854	0.7225
18	0.3752	0.3347	0.365	0.3429	0.6426	0.336	0.6435	0.6865
19	0.2481	0.207	0.241	0.2085	0.3943	0.2424	0.3888	0.4188
20	0.1877	0.1527	0.1818	0.1546	0.2831	0.1976	0.2794	0.303
21	0.1995	0.1671	0.1945	0.1708	0.315	0.1997	0.3145	0.3373
22	0.3641	0.322	0.3551	0.3271	0.6011	0.3268	0.5995	0.6276
23	0.3306	0.2921	0.3227	0.298	0.5468	0.3059	0.5474	0.5756
24	0.3855	0.3349	0.3739	0.3336	0.6037	0.3475	0.5972	0.6203
25	0.2625	0.2317	0.2557	0.2368	0.4374	0.2452	0.4376	0.4689
26	0.2792	0.2402	0.2687	0.2412	0.4473	0.2731	0.4405	0.4736
27	0.2657	0.2314	0.2585	0.2356	0.4329	0.2548	0.4316	0.4612

4.1.3 變異數分析

透過上個階段取得實驗數據結果之後，我們須進一步找尋九個參數因子中哪些是影響品質特性的顯著因子。故採用變異數分析法來做顯著因子的搜尋。表 4.6~ 表 4.9 為以翹曲實驗值做為反應變數 y 所得之變異數分析表，其中注射溫度、保壓壓力、開模時間及熔劑溫度為 N11、N2203、N55 之顯著因子(p 值<0.1)。而注射溫度、保壓壓力、開模時間則為 N941 之顯著因子(p 值<0.1) 。

表 4.6、N11 之變異數分析表

Source	DF	Seq SS	Adj SS	Adj MS	F	P
INT	**2**	**6.0672**	**6.0672**	**3.0336**	**12.28**	**0.004**
INP	2	0.0793	0.0793	0.0396	0.16	0.854
PP	**2**	**98.3094**	**98.3094**	**49.1547**	**198.97**	**0**
PT	2	0.0958	0.0958	0.0479	0.19	0.827
COTI	2	0.117	0.117	0.0585	0.24	0.795
COTE	2	1.301	1.301	0.6505	2.63	0.132
MOO	**2**	**1.752**	**1.752**	**0.876**	**3.55**	**0.079**
MET	**2**	**3.4505**	**3.4505**	**1.7253**	**6.98**	**0.018**
MOTE	2	0.5027	0.5027	0.2514	1.02	0.404
Error	8	1.9764	1.9764	0.247		
Total	26	113.6512				

表 4.7、N2203 之變異數分析表

Source	DF	Seq SS	Adj SS	Adj MS	F	P
INT	**2**	**9.247**	**9.2466**	**4.6233**	**14.36**	**0.002**
INP	2	0.223	0.223	0.1115	0.35	0.717
PP	**2**	**118.158**	**118.158**	**59.079**	**183.46**	**0**
PT	2	0.119	0.119	0.0595	0.18	0.835
COTI	2	0.117	0.1167	0.0583	0.18	0.838
COTE	2	1.865	1.8651	0.9325	2.9	0.113
MOO	**2**	**2.203**	**2.2027**	**1.1014**	**3.42**	**0.084**
MET	**2**	**2.009**	**2.0094**	**1.0047**	**3.12**	**0.1**
MOTE	2	0.621	0.6211	0.3105	0.96	0.422
Error	8	2.576	2.5762	0.322		
Total	26	137.138				

表 4.8、N55 之變異數分析表

Source	DF	Seq SS	Adj SS	Adj MS	F	P
INT	2	6.0582	6.0582	3.0291	11.96	**0.004**
INP	2	0.0922	0.0922	0.0461	0.18	0.837
PP	2	99.137	99.137	49.5685	195.75	**0**
PT	2	0.1156	0.1156	0.0578	0.23	0.801
COTI	2	0.1574	0.1574	0.0787	0.31	0.741
COTE	2	1.4272	1.4272	0.7136	2.82	0.118
MOO	2	1.8176	1.8176	0.9088	3.59	**0.077**
MET	2	3.3297	3.3297	1.6649	6.57	**0.02**
MOTE	2	0.4791	0.4791	0.2395	0.95	0.428
Error	8	2.0258	2.0258	0.2532		
Total	26	114.6399				

表 4.9、N941 之變異數分析表

Source	DF	Seq SS	Adj SS	Adj MS	F	P
INT	2	10.2382	10.2382	5.1191	15.48	**0.002**
INP	2	0.2217	0.2217	0.1108	0.34	0.725
PP	2	117.0064	117.0064	58.5032	176.91	**0**
PT	2	0.1112	0.1112	0.0556	0.17	0.848
COTI	2	0.133	0.133	0.0665	0.2	0.822
COTE	2	1.8718	1.8718	0.9359	2.83	0.118
MOO	2	2.1927	2.1927	1.0964	3.32	**0.089**
MET	2	1.5371	1.5371	0.7686	2.32	0.16
MOTE	2	0.5165	0.5165	0.2583	0.78	0.49
Error	8	2.6456	2.6456	0.3307		
Total	26	136.4743				

而表 4.10~ 表 4.13 為以收縮實驗值做為反應變數 y 所得之變異數分析表，其中注射溫度、保壓壓力為 E1、E3 之顯著因子(p 值<0.1)。注射溫度、保壓壓力、開模時間為 E2 之顯著因子(P 值<0.1)。保壓壓力為 E4 之顯著因子(P 值<0.1)。以表 4.14 呈現整理之結果，聯集之後取注射時間、保壓壓力、開模溫度、熔劑溫度為整體分析之顯著因子。

表 4.10、E1 之變異數分析表

Source	DF	Seq SS	Adj SS	Adj MS	F	P
INT	**2**	**12.5059**	**12.5059**	**6.2529**	**16.92**	**0.001**
INP	2	0.1467	0.1467	0.0734	0.2	0.824
PP	**2**	**115.5739**	**115.5739**	**57.787**	**156.4**	**0**
PT	2	0.1743	0.1743	0.0871	0.24	0.795
COTI	2	0.2247	0.2247	0.1123	0.3	0.746
COTE	2	2.2044	2.2044	1.1022	2.98	0.108
MOO	2	2.1687	2.1687	1.0843	2.93	0.111
MET	2	2.0541	2.0541	1.0271	2.78	0.121
MOTE	2	0.454	0.454	0.227	0.61	0.565
Error	8	2.9559	2.9559	0.3695		
Total	26	138.4625				

表 4.11、E2 之變異數分析

Source	DF	Seq SS	Adj SS	Adj MS	F	P
INT	2	1.8251	1.8251	0.9125	5.26	**0.035**
INP	2	0.0074	0.0074	0.0037	0.02	0.979
PP	2	67.715	67.715	33.8575	195.05	**0**
PT	2	0.1411	0.1411	0.0705	0.41	0.679
COTI	2	0.0283	0.0283	0.0141	0.08	0.923
COTE	2	0.5158	0.5158	0.2579	1.49	0.283
MOO	2	0.8949	0.8949	0.4474	2.58	0.137
MET	2	4.1843	4.1843	2.0922	12.05	**0.004**
MOTE	2	0.375	0.375	0.1875	1.08	0.384
Error	8	1.3887	1.3887	0.1736		
Total	26	77.0755				

表 4.12、E3 之變異數分析表

Source	DF	Seq SS	Adj SS	Adj MS	F	P
INT	2	13.6132	13.6132	6.8066	17.65	**0.001**
INP	2	0.1698	0.1698	0.0849	0.22	0.807
PP	2	116.7026	116.7026	58.3513	151.3	**0**
PT	2	0.151	0.151	0.0755	0.2	0.826
COTI	2	0.2492	0.2492	0.1246	0.32	0.733
COTE	2	2.2372	2.2372	1.1186	2.9	0.113
MOO	2	2.2517	2.2517	1.1259	2.92	0.112
MET	2	1.6932	1.6932	0.8466	2.2	0.174
MOTE	2	0.3822	0.3822	0.1911	0.5	0.627
Error	8	3.0854	3.0854	0.3857		
Total	26	140.5355				

表 4.13、E4 之變異數分析表

Source	DF	Seq SS	Adj SS	Adj MS	F	P
INP	2	0.1367	0.1367	0.0684	0.18	0.836
PP	**2**	**109.2292**	**109.2292**	**54.6146**	**145.89**	**0**
PT	2	0.2399	0.2399	0.1199	0.32	0.735
COTI	2	0.2566	0.2566	0.1283	0.34	0.72
COTE	2	2.1062	2.1062	1.0531	2.81	0.119
MOO	2	2.0191	2.0191	1.0095	2.7	0.127
MET	2	1.4476	1.4476	0.7238	1.93	0.207
MOTE	2	0.4415	0.4415	0.2208	0.59	0.577
Error	8	2.9948	2.9948	0.3743		
Total	26	133.6059				

表 4.14、整理之顯著因子結果

	N11	N2203	N55	N941	E1	E2	E3	E4	U
INT	◎	◎	◎	◎	◎	◎	◎		●
INP									
PP	◎	◎	◎	◎	◎	◎	◎	◎	●
PT									
COTI									
COTE									
MOO	◎	◎	◎	◎					●
MET	◎	◎	◎			◎			●
MOTE									

4.2 訓練倒傳遞類神經網路

從變異數分析的結果得知注射時間、保壓壓力、開模時間及熔劑溫度四個製程參數為品質特性的顯著因子，因此在這個階段中我們將利用參數範圍內隨機產生 81(3^4)組的參數組合進行 Moldflow 的模擬實驗，並取得翹曲及收縮之實驗值。再從四個翹曲及四個收縮值中，取最大翹曲值及最大收縮值作為輸出值進行實驗。進行網路訓練之前，我們須將參數及實驗值利用以下公式進行數值正規化，以數據資料之 L1、L3 水準界限將數值的範圍限制在 0~1 之間。

$$INT_{Normalize}=\frac{x_{1i}-0.7}{1.3-0.7} \qquad MET_{Normalize}=\frac{x_{4i}-247.5}{302.5-247.5} \qquad Warpage_{Normalize}=\frac{x_{5i}-0.5}{0.5}$$

$$PP_{Normalize}=\frac{x_{2i}-56}{104-56} \qquad MOO_{Normalize}=\frac{x_{3i}-3.5}{6.5-3.5} \qquad Shrinkage_{Normalize}=\frac{x_{6i}-0.75}{0.75}$$

透過研究方法中指出的網路學習法則進行倒傳遞類神經網路的建構，利用 C 語法建立一層輸入層、一層隱藏層及一層輸出層之訓練網路，其中輸入層包含四個輸入結點、輸出層包含兩個輸出結點、隱藏層則為六個結點，如圖 4.1 所示。其中設定學習率為 0.5、慣性因子為 0.2 及訓練次數 10000 次。

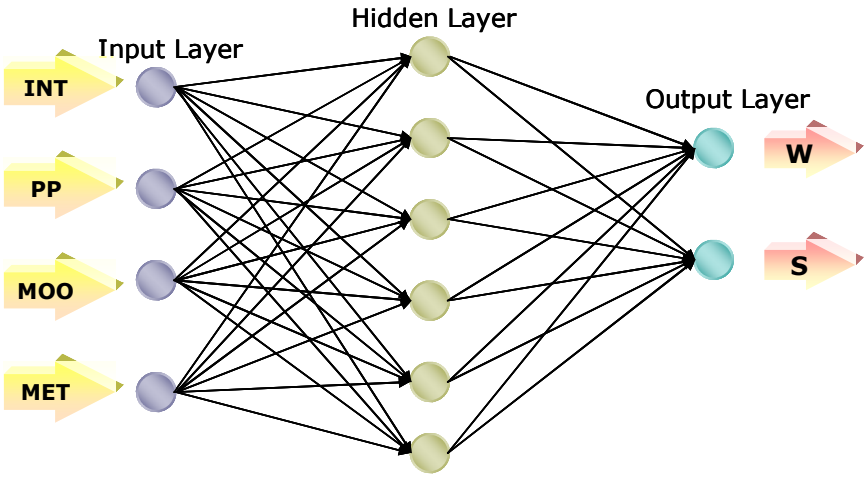


圖 4.1、本研究類神經網路示意圖

接著，我們將 81 組的之實驗數據利用 10-Fold Cross Validation 方法處理數據資料，將數據分成十份，輪流將九份資料進行訓練，以一份資料進行驗證，並將實驗數據中的製程參數、翹曲及收縮正規化至 0~1 之數值，透過此方法將可以得到精確的訓練網路。因此，本研究以 730 組的訓練資料及 81 筆的驗證資料進行類神經網路訓練及驗證。從網路訓練之 MSE 會收斂至 0.0002 可知，此網路之配適狀態良好，如圖 4.2 及圖 4.3。

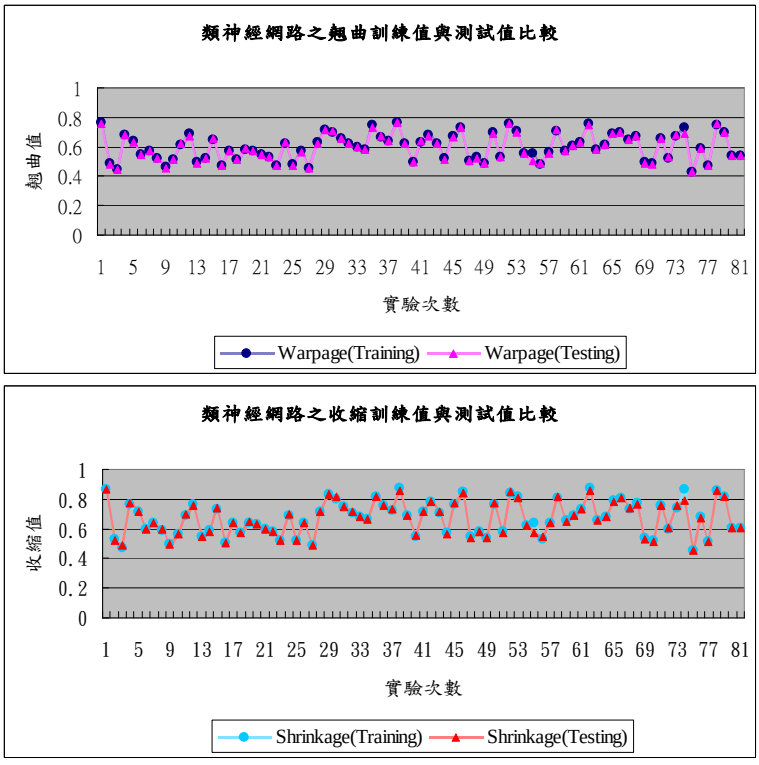


圖 4.2、翹曲之訓練值與測試值比較及收縮之訓練值與測試值比較

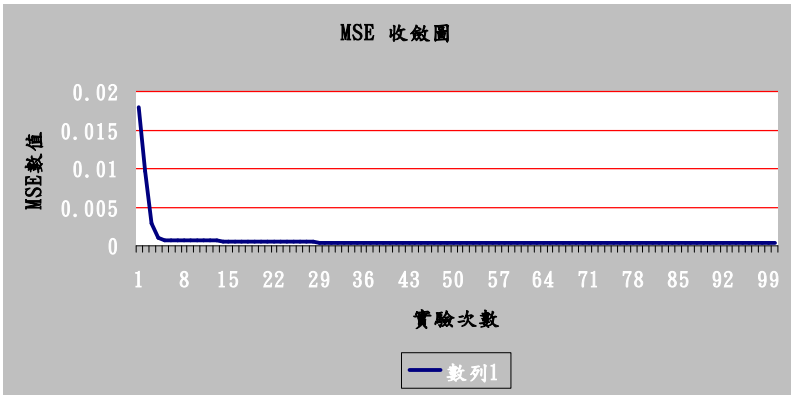


圖 4.3、類神經網路訓練 MSE 收斂示意圖

81 組數據驗證之結果透過 SPSS 統計軟體計算所得之 ROC 曲線如圖 4.4 可看出訓練網路所得知目標值與實際模擬取得之目標值誤差相當小。其中，翹曲之 ROC 曲線下面積為 0.977；收縮之 ROC 曲線下面積為 0.994。

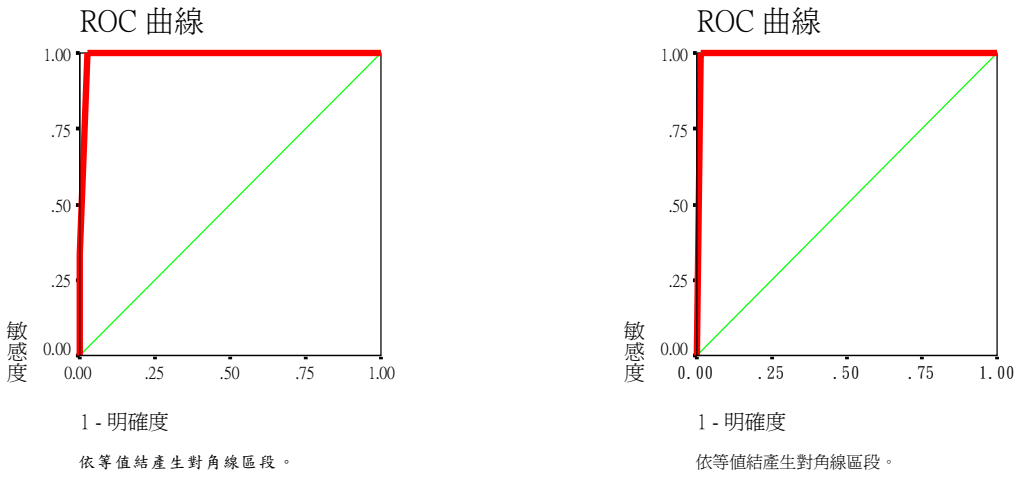


圖 4.4、翹曲(左)及收縮(右)之 ROC 曲線

網路訓練完畢之後，我們可取得網路之權重(Weight)及偏權值(Bias)，利於下一階段基因演算法之運算實驗。如附錄 E 所示。

4.3 基因演算法

在實驗的過程中，我們以基因演算法複製、交配、突變過程中所產生之參數組合，代入類神經網路訓練之函數並得到翹曲值及收縮值。接著，反覆利用此過程進行最小翹曲或最小收縮之搜尋以找出最佳化收縮及翹曲之參數組合結果。最後，將所得之最佳參數組合以 Moldflow 進行結果驗證的動作，確認實驗之誤差大小及精確性。在應用基因演算法時，個體之位元長度將決定製程參數的精度，即解析度。本研究以位元長度為 10，分為 2^{10}(1024)區間，以達較好之精度。依照基因演算法之法則，初始群組之參數組合從參數水準之限制範圍內隨機選取，本研究基因演算法之設定以母代群組規模為 12、各個參數之位元長度為 10、交配率為 1.0、突變率為 1.0。實際之目標函數如下，以收縮為例:

目標函數： S=f(INT, PP, MOO, MET)

適應函數： F =1-f

限制條件： $0.7\ s \le INT \le 1.3\ s$

$56\ MPa \le PP \le 104\ MPa$

$3.5\ s \le MOO \le 6.5\ s$

$247.5\ ^{\circ}C \le MET \le 302.5\ ^{\circ}C$

當我們利用基因演算法不給予任何限制下，個別搜尋此塑件之最小翹曲值及最小收縮值時，我們得到正規化數值結果如下:

最小化翹曲 ： (Warpage, Shrinkage) = (0.3779, 0.3957)，如表 4. 16。

最小化收縮 ： (Warpage, Shrinkage) = (0.3801, 0.3943)，如表 4. 17。

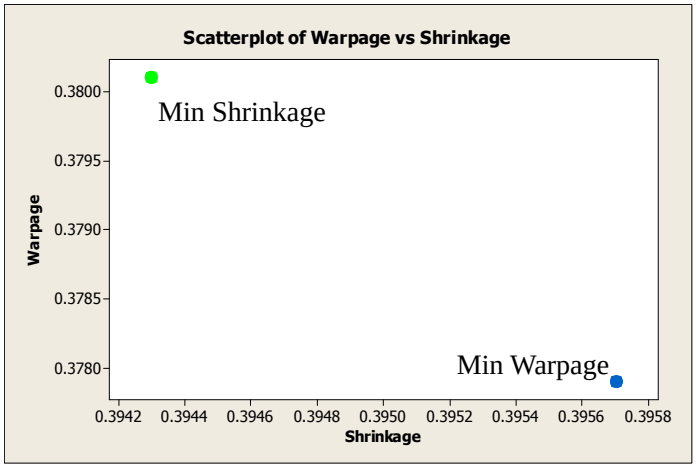


圖 4.5、最小翹曲及最小收縮示意圖(正規化數值)

將實驗所得之參數組合轉換為原始實驗數值之後，透過 Moldflow 實際模擬之後，可得知最小收縮值及最小翹曲值之實驗值與模擬值誤差甚小，如表 4. 15 所示。

表 4.15、GA 預測值與 Moldflow 模擬數值之比較

分析方法	實驗數值	翹曲	收縮
最小翹曲	GA預測值	0.18895	0.296775
	Moldflow實際值	0.1910	0.2909
	差值	0.00205	0.0059
最小收縮	GA預測值	0.19005	0.295725
	Moldflow實際值	0.1904	0.2891
	差值	0.00035	0.0066

表 4.16、基因演算法搜尋之最小翹曲結果

顯著參數	一般化之數值	GA_原始之數值	MoldFlow模擬數值
INT	1	1.3	1.3
PP	1	104	104
MOO	0	3.5	3.5
MET	0.4066	269.863	269.863
Max Warpage	0.3779	0.18895	0.1910
Max Shrinkage	0.3957	0.296775	0.2909

表 4.17、基因演算法搜尋之最小收縮結果

顯著參數	一般化之數值	GA_原始之數值	MoldFlow模擬數值
INT	1	1.3	1.3
PP	1	104	104
MOO	0	3.5	3.5
MET	0.2493	261.2115	261.2115
Max Warpage	0.3801	0.19005	0.1904
Max Shrinkage	0.3943	0.295725	0.2891

透過表 4. 16 及表 4. 17 之結果比較我們發現: 當翹曲值達最小時，其收縮值相對於實驗之最小收縮值還來得大，且當收縮值達最小時，其翹曲值也同樣相對於實驗之最小翹曲來的大。然而，在塑件實際產出時，我們認為收縮與翹曲皆越小所得塑件品質越好。因此，我們希望搜尋出一組參數組合其收縮及翹曲兩者數值皆小且在我們允許的範圍內的結果。

本研究將透過下一小節之搜尋方法，以限制收縮值的範圍利用 GA 進行最小翹曲值搜尋，期望達到最佳的收縮及翹曲組合。並透過流程圖清楚說明如何決定收縮限制之範圍，以快速達到目標之搜尋。

4.3.1 限制收縮之範圍，搜尋較佳的翹曲與收縮組合

經由基因演算法之演化得到正規化數值收縮值最小組合為(Warpage, Shrinkage) = (0.3801, 0.3943)。接下來，以”已知收縮最小值，欲搜尋較佳之翹曲值”作為目標進行下一步之搜尋，搜尋的過程如圖 4. 6。

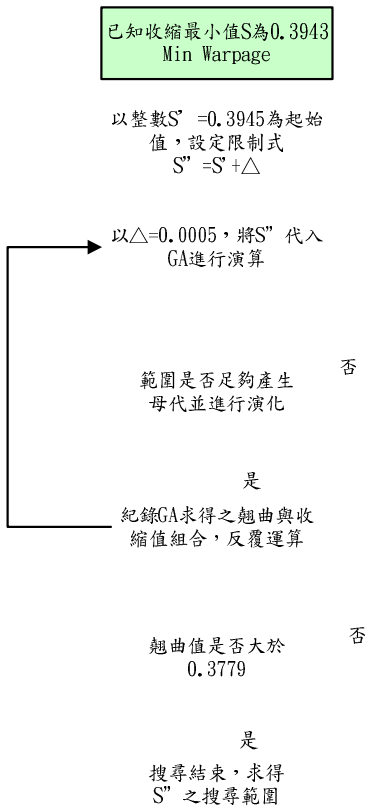


圖 4.6、以收縮為限制搜尋最小翹曲之流程

在已知正規化數值收縮最小值為 0.3943 的情況下，我們在 GA 下設定一個收縮限制值進行翹曲最小值之搜尋。由於在最小收縮值的狀態下已無法找出更佳翹曲值，因此我們透過逐步放寬收縮之限制來找出可能存在之較好的翹曲值。但由於本研究 GA 之設計為在母代時即進行收縮限制的篩選，而母代一次需產生 12 組之群組，所以必須將收縮之限制放寬至某適當範圍才可進行 GA 之運算。本實驗設定當翹曲值大於 0.3779 時即停止搜尋，此數值為搜尋最小翹曲時所得之翹曲值。因此，透過圖 4. 6 流程之搜尋方法便可找出收縮限制下建議搜尋的範圍。

進行搜尋過程時，在已知最小收縮值為 0.3943 下，我們以整數值 0.3945、△=0.0005 放寬收縮開始搜尋，直到收縮限制放寬至 0.3985 時 GA 便可產生足夠的母代進行複製、交配及突變，取得基因演算之結果。

經由基因演算法反覆的搜尋過程，如附錄 D，我們發現當收縮值放寬至 0.4545 時，會得到(Warpage,Shrinkage)=(0.3787,0.3990)的翹曲收縮組合，此時翹曲大於 0.3779，為 GA 演化較差且不被接受的翹曲及收縮組合，故使實驗停止放寬收縮值搜尋。透過這樣搜尋的過程，我們建議僅將收縮放寬以 0.3943≦Shrinkage≦0.4545 作為搜尋範圍即可，如圖 4. 7。而在此範圍下搜尋出的翹曲及收縮組合如下所示。

目標函數： W=f(INT, PP, MOO, MET)

適應函數： F =1-f

限制條件： **0.3943≦Shrinkage≦0.4545**

$$0.7\ s \le INT \le 1.3\ s$$

$$56\ MPa \le PP \le 104\ MPa$$

$$3.5\ s \le MOO \le 6.5\ s$$

$$247.5\ ^{\circ}C \le MET \le 302.5\ ^{\circ}C$$

$$Level\ 1 \le INT, PP, MOO, MET \le Level\ 3$$

其中，GA 搜尋產生之正規化數值收縮翹曲組合如下:

$(\text{Warpage, Shrinkage})_1 = (0.3801, 0.3943)$

$(\text{Warpage, Shrinkage})_2 = (0.3779, 0.3949)$

$(\text{Warpage, Shrinkage})_3 = (0.3779, 0.3954)$

$(\text{Warpage, Shrinkage})_4 = (0.3779, 0.3955)$

$(\text{Warpage, Shrinkage})_5 = (0.3779, 0.3956)$

$(\text{Warpage, Shrinkage})_6 = (0.3779, 0.3957)$

$(\text{Warpage, Shrinkage})_7 = (0.3787, 0.3990)$

正規化數值經轉換為原始數值後之收縮翹曲組合如下:

$(\text{Warpage, Shrinkage})_1 = (0.19005, 0.295725)$

$(\text{Warpage, Shrinkage})_2 = (0.18895, 0.296175)$

$(\text{Warpage, Shrinkage})_3 = (0.18895, 0.29655)$

$(\text{Warpage, Shrinkage})_4 = (0.18895, 0.296625)$

$(\text{Warpage, Shrinkage})_5 = (0.18895, 0.2967)$

$(\text{Warpage, Shrinkage})_6 = (0.18895, 0.296775)$

$(\text{Warpage, Shrinkage})_7 = (0.18935, 0.29925)$

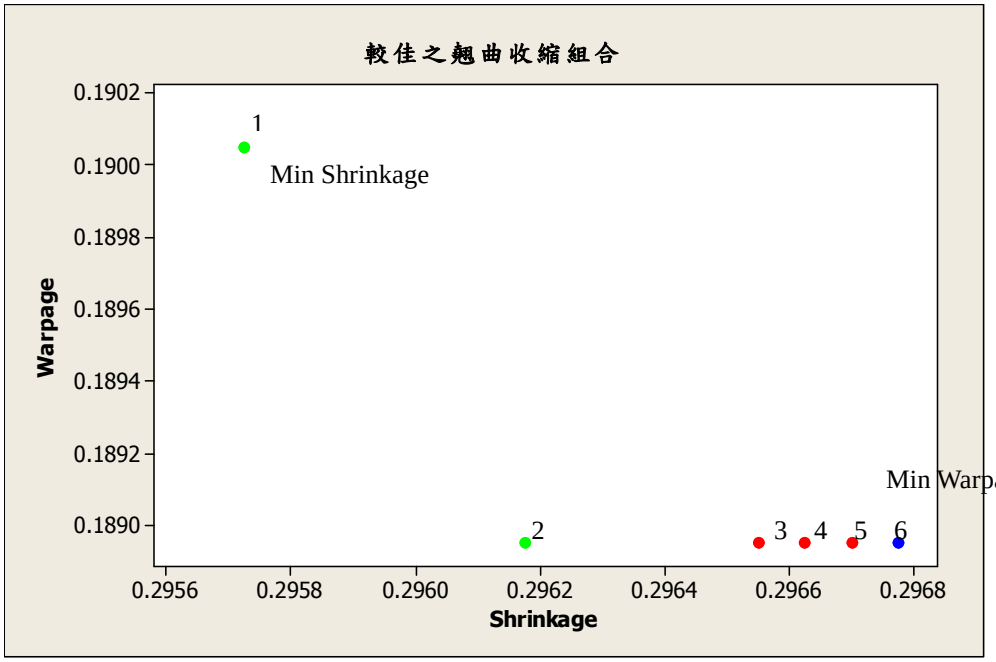


圖 4.7、搜尋範圍內較佳之翹曲收縮組合(原始數值)

在此搜尋範圍所產生的翹曲收縮組合中，我們可以明顯看出組合 7 為較差之翹曲收縮組合故不予以討論。事實上，組合 1 至組合 6 之收縮翹曲組合結果皆已呈現良好的品質特性，但其中組合 1 及組合 2 我們又可視為無異之最佳解決方案(Indifference solutions)，透過放寬組合 1 之收縮而得到一較佳之翹曲收縮組合，使用者可藉由對於翹曲或收縮之重視程度自行選擇最適合之參數組合。接下來利用 Moldflow 進行此六組結果之驗證，以確保實驗之準確性。參數設定及模擬結果如表 4.18 所示。

表 4.18、Moldflow 參數設定及模擬結果

NO.	INT	PP	MOO	MET	Warpage	Shrinkage
1	**1.3**	**104**	**3.5**	**261.2115**	**0.1904**	**0.2891**
2	**1.3**	**104**	**3.5**	**268.092**	**0.1915**	**0.2922**
3	1.3	104	3.5	269.379	0.1921	0.2931
4	1.3	104	3.5	269.544	0.1928	0.294
5	1.3	104	3.5	269.6485	0.1924	0.2932
6	1.3	104	3.5	269.863	0.1910	0.2909

接著透過表 4.19 整理比較 Moldflow 模擬與 GA 實驗搜尋之結果，我們可以發現兩者之平均誤差皆僅介於千分之二至千分之三之間。其中，翹曲之平均誤差為 0.002567，收縮之平均誤差為-0.003975。因此我們可以相信此實驗過程之準確性及實際應用之可能性。

表 4.19、Moldflow 模擬值與 GA 實驗值之比較

No.	**Moldflow 模擬**		**GA 搜尋範圍內較佳之參數組合 (如 圖 4.7)**		**Moldflow 模擬與 GA 搜尋之差值**	
	Warpage	Shrinkage	Warpage	Shrinkage	Warpage	Shrinkage
1	**0.1904**	**0.2891**	**0.19005**	**0.295725**	**0.00035**	**-0.006625**
2	**0.1915**	**0.2922**	**0.18895**	**0.2811**	**0.00255**	**0.0111**
3	0.1921	0.2931	0.18895	0.29655	0.00315	-0.00345
4	0.1928	0.2940	0.18895	0.296625	0.00385	-0.002625
5	0.1924	0.2932	0.18895	0.2967	0.00345	0.0035
6	0.1910	0.2909	0.18895	0.296775	0.00205	-0.005875
Average difference					**0.002567**	**-0.003975**

4.3.2 小結

透過以上之實驗，藉由限制收縮的範圍去搜尋較佳的翹曲，利用犧牲部份的收縮值來找出更好的翹曲值，因而得到以下兩組無異之最佳解決方案 $(Warpage,\ Shrinkage)_1 = (0.19005,\ 0.295725)$、$(Warpage,\ Shrinkage)_2 = (0.18895,\ 0.296175)$。此兩組無異解所對應之參數設定值為注射時間 1.3 s、注射壓力 120MPa、保壓時間 10s、保壓壓力 104 MPa、冷卻時間 19s、冷卻溫度 25°C、開模時間 3.5 s、熔劑溫度 261.2115°及熔劑溫度 268.092、模穴溫度 80°C，此時塑件可得最佳化之收縮翹曲組合，達到最佳品質結果。

第五章、結論及未來研究方向

5.1 結論

在射出成型產業中塑件品質優劣及產品生命週期速度為公司降低成本之關鍵，本研究以手機機殼製程之顯著因子進行實驗分析，透過模擬取得與參數相對應之翹曲收縮數據，透過類神經網路訓練出一多重輸入多重輸出之函數，接著透過將參數代入類神經網路訓練函數以取得輸出值，最後，以基因演算法搜尋出製程之最佳參數組合，證實此方法可行且可快速的找出最佳多個品質特性結果，找出最佳化之製程參數組合。在此實驗搜尋的範圍中，我們得到兩組收縮翹曲結果為無異之最佳解決方案。

針對此多目標問題，本研究提出一個可行的實驗步驟方法，透過類神經網路結合基因演算法找出一個可以快速搜尋射出成型製程參數設定以達最佳化多目標之翹曲及收縮組合之搜尋方法，作為工程師進行整體考量時的參考依據。透過基因演算法結合類神經網路，並給予限制條件，以解決多目標函數難以求解之困難。提供射出成型工程師未來在進行塑膠產品參數設定時，有一參數選擇之參考依據。

5.2 未來之研究方向

1. 在射出成型製程中，Chen and Turng(2005)建議將變數分類為三個範疇: (1)機器參數 (2)製程參數 (3) 品質參數。而這些變數皆對於產品最終品質及製程的經濟性有顯著的影響。本研究主要只針對製程參數進行分析探討，未來可增加機器參數分析以更進一步的改善塑件品質。
2. 當劣品產生時，浪費已經發生，故若能在製程中間發現一個準確判斷品質優劣之指標或是方程式，更能進一步的減少企業之不必要的浪費。
3. 塑件之關鍵品質特性為收縮及翹曲，故本研究僅針對此兩個重要之品質特性進行分析，但實際上縫合線、剪應力、包風等特性同時也為塑件品質之考量因素，未來若能有一多目標函數可整合多個品質特性進行分析研究，相信可使機台在實際操作時得到更全面性的考量。

參考文獻

中文

[1] 王進德，*ANN 介紹-類神經網路與模糊控制理論-入門與應用*，全華科技圖書股份有限公司，2007

[2] 吳俊煌，*塑膠射出成型模具設計*，復文書局，2001。

[3] 林啟淙，「*一般最佳化方法在塑膠射出成型之應用*」，機械工程系，台灣科技大學，2005。

[4] 林瑞璋，*塑膠膜電腦輔助設計-MoldFlow 軟體應用*，全威圖書有限公司，2003

[5] 洪啟偉，「*以雙反應曲面法與非線性規劃進行塑膠射出成形作業之最佳化設計*」，工業工程與經營資訊學系，東海大學，2007

[5] 財團法人塑膠工業技術發展中心 http://www.pidc.org.tw/Pages/default.aspx

[6] 陳劉旺、丁金超，*高分子加工*，高立圖書有限公司，1989。

[7] 陳良相、黃子健、劉昭宏，*Moldflow MPI 實用基礎*，全華科技圖書股份有限公司，2005。

[8] 黃俊欽，*射出成型各階段與操作條件的關係(塑膠射出成型加工流程)*，高雄應用科技大學模具工程系，射出成型講義 P.5

[9] 張云濤、龔玲，*資料探勘原理與技術 Data Mining、AI、Algorithm*，五南圖書出版股份有限公司，2007

[10] 楊景程，「*射出成型機最佳參數之預測*」，纖維高分子工程，台灣科技大學，2000。

[11] 廖述賢、溫志浩，*資料採礦與商業智慧*，雙葉書廊有限公司，2009

[12] 羅華強，*類神經網路-Matlab 的應用*，高立圖書有限公司，2005

[13] 羅壬成，「*模流分析與社出成型控制參數的優化*」，工學院精密與自動化工程，交通大學，2006。

英文

[1] Bloodshed Dev-C++ 5.0 Compiler 軟體下載 http://www.bloodshed.net/devcpp.html

[2] Chen, Z., Turng, L.S., "A Review of Current Developments in process and Quality Control for Injection Molding,"*Advances in Ploymer Technology*, 24, 165-182, 2005.

[3] Chiang, K.T., Chang F.P., "Analysis of shrinkage and warpage in an injection-molded part with a thin shell feature using the response surface methodology, " *International Journal of Advance Manufacturing Technology*, 35, 468-479, 2007.

[4] Chiang K.T.,"The Optimal process conditions of an injection-molded thermoplastic part with a thin shell feature using grey-fuzzy logic: A case study on machining the PC/ABS cell phone shell," Material and Design, 28, 1851-1860, 2007

[5] Deng, W.J., Chen, C.T., Sun, C.H., Chen, W.C., Chen, C.P., "An Effective Approach for Process Parameter Optimization in Injection Molding of Plastic Housing Components," *Polymer Plastics Technology and Engineering*, 47, 910-919, 2008

[6] Erzurumlu, T., Ozcelik, B., " Minimization of Warpage and Sink Index in Injection-molded Thermoplastic parts using Taguchi optimization method," *Materials & Design*, 27, 853-861, 2006

[7] Gao, Y., Wang, X., "Surrogate-based process optimization for reducing warpage in injection molding" *Journal of materials processing technology*, 209, 1302-1309, 2009

[8] Huang, M.C., Tai, C.C., "The Effective Factors in the Warpage Problem of an Injection-Molded part with a Thin Shell Feature," *Journal of Material Process Technology*, 110(1), 1-9, 2001

[9] Jansen, K.M.B., Pantani, R., Titomanlio, G., "As-molded shrinkage measurement on polystyrene injection molded products, " *Polymer Engineering Science*, 38, 254-264, 1998

[10] Kurtaran, H., Ozcelik, B., Tuncay, E., "Warpage optimization of a bus ceiling lamp base using neural network model and genetic algorithm," *Journal of Materials Processing Technology*, 169, 314-319, 2005

[11] Liao, S.J., Chang, D.Y., Chen, H.J., Tsou, L.S., "Optimal Process Condition of Shrinkage and Warpage of Thin-Wall Parts," *Polymer Engineering Science*, 44(5), 917-928, 2004

[12] Mok, S.L., Kwong, C.k., "Application of artificial neural network and fuzzy logic in a case-based system for initial process parameter setting of injection molding," *Journal of Intelligence manufacturing*, 13(3), 165-176, 2002

[13] Jay Shoemaker (Ed.), "Moldflow Design Guide- A resource for plastics Engineers, " *Hanser Gardner Publications, Incorporation*, 2006

[14] Negnevitsky M., “Artificail Intelligence: A Guide to intelligent systems, Second Edition, “*ADDISON WESLEY*, 2005

[15] Ozcelik, B., Sonat, I., “Warpage and structural analysis of thin shell plastic in the plastic injection molding, ” *Material and design*, 30, 367-375, 2009

[16] Rewal, N., Toncich, D., Friedl, C., "Predicting part quality in injection molding using artificial neural networks," *Journal of Injection Molding Technology*, 2, 109-119, 1998

[17] Shen, C., Wang, L., Li, Q., “Optimization for injection molding process conditions of refrigerator top cover using combination method of artificial neural network and genetic algorithms,” *Polymeric Plastic Technique Engineering*, 46(2), 105-112, 2006

[18] Song, M.C., Liu, Z., Wang, M.J., Yu, T.M., Zhao, D.Y., ”Research on effects of injection process parameters on the molding process for ultra-thin wall plastic parts, ” *Journal of Materials Processing Technology*, 187-188, 668-671, 2007.

[19] Shen, Y.K., Wu, C.W., Yu, Y.F., Chung, H.W., “Analysis for optimal gate design of thin-walled injection molding, ” *International Communication in Heat and Mass Transfer*, 35, 708-734, 2008.

[20] Velia Garc´ıa Loera, Jos´e M. Castro, Jes´us Mireles Diaz, O´ scar L. Chaco´n Mondrago´n, *“*Setting the Processing Parameters in Injection Molding Through Multiple-Criteria Optimization: A Case Study,” *IEEE Transcation on Systems, Man and Cybernetics-part C Application and Review*, 38(5), 2008

[21] Woll, S. L. B., Cooper, D. J., "On-line pattern-based part quality monitoring of injection molding process," *Polymer Engineering and Science*, 36, 1477-1488, 1996.

[22] Yang, Y., Gao, F., “Injection Molding Product Weight: Online Prediction and Control Based on a Nonlinear principle Component Regression Model,” *Polymeric Science*, 46(4), 540-548, 2006

[23] Zhai, M., Lam, Y.C., Au, c.k., “Runner sizing and weld line positioning for plastics injection moulding with multiple gates, ” *Engineering with Computers*, 21, 218–224, 2006.

[24] Zhou, H., Zhao, P., Feng, W., “ An Intergrated Intelligent System for Injection Molding Process Determination,” *Advances in ‘Polymer Technology*, 26(3), 191-205, 2007

[25] Zhao, C., Gao, F., “Melt Temperature Profile Prediction for Thermoplastic Injection Molding,” *Polymer Engineering Science*, 39(9) 1787-1801, 1999

附錄 A: 類神經網路之 C 程式語言 (Training)

```
#include <cstdlib>
#include <iostream>

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
using namespace std;
float random_value(void);
int main(int argc, char *argv[])
{       if(argc == 1)
        {       printf("Usage: \n");
                printf(">NNTRAIN.exe (-c/-t/-i/-h/-o/-e/-a) train_file weight_file mse_file \n");
                printf("      where: -c = training cycle [default = 1000] \n");
                printf("                -t = # of training sample [default = AUTO] \n");
                printf("                -i = # of input nodes [default = 4]\n");
                printf("                -h = # of hidden nodes [default = 6] \n");
                printf("                -o = # of output nodes [default = 2]\n");
                printf("                -e = ETA [default = 0.5] \n");
                printf("                -a = alpha [default = 0.2] \n\n");

                system("PAUSE");
                return EXIT_SUCCESS;
        }
        FILE *train_fp, *weight_fp, *mse_fp;

        if(argc >= 4)
        {       train_fp = fopen(argv[argc-3], "r");
                weight_fp = fopen(argv[argc-2], "w");
                mse_fp = fopen(argv[argc-1], "w");

                if(train_fp == NULL)
                {
                        cout<<"\nError! Training file not exist!"<<endl;
                        system("PAUSE");
                        return EXIT_FAILURE;
                }
```

```
}
else
{       cout<<"\nError! Please indicate the files. See Usage.\n"<<endl;
        system("PAUSE");
        return EXIT_FAILURE;
}
int Ncycle = 1000;
int Ntrain = -1;
int Ninp = 4;
int Nout = 2;
int Nhid = 6;
float eta = 0.5;
float alpha = 0.2;

for(int i=1;i<argc-3;i++)
{
        if(argv[i][0] == '-')
        {
                switch(argv[i][1])
                {
                case 'c':
                        Ncycle = atoi(argv[i+1]);
                        break;
                 case 't':
                        Ntrain = atoi(argv[i+1]);
                        break;
                 case 'i':
                        Ninp = atoi(argv[i+1]);
                        break;
                 case 'h':
                        Nhid = atoi(argv[i+1]);
                        break;
                 case 'o':
                        Nout = atoi(argv[i+1]);
                        break;
                 case 'e':
                        eta = atof(argv[i+1]);
                        break;
```

```
                case 'a':
                        alpha = atof(argv[i+1]);
                        break;
                default:
                        break;
                }
        }
}

char c = 0x00;

if(Ntrain == -1)
{
        Ntrain = 0;

        fseek(train_fp, 0, SEEK_SET);

        while(fscanf(train_fp, "%c", &c) != -1)
        {
                if(c == 0x0A)
                        Ntrain++;
        }

        fseek(train_fp, 1, SEEK_END);
        fscanf(train_fp, "%c", &c);
        if(c != 0x0A && c != 0x0D)
                Ntrain++;
        Ntrain--;     //// only for specific data file format!!!
}
printf("\nNcycle=%d Ntrain=%d Ninp=%d Nhid=%d Nout=%d ", Ncycle, Ntrain, Ninp, Nhid, Nout);
printf("\neta=%f alpha=%f \n", eta, alpha);

float X[Ninp], T[Nout], H[Nhid], Y[Nout];
float W_xh[Ninp][Nhid], W_hy[Nhid][Nout];
float dW_xh[Ninp][Nhid], dW_hy[Nhid][Nout];
float Q_h[Nhid], Q_y[Nout];
float dQ_h[Nhid], dQ_y[Nout];
float delta_h[Nhid], delta_y[Nout];
```

```
float sum, mse;
int Icycle, Itrain;
int x, y, h;

srand((int)time(0));

for(h=0;h<Nhid;h++)
{
	for(x=0;x<Ninp;x++)
	{
		W_xh[x][h] = random_value();
		dW_xh[x][h] = 0;
	}
}
for(y=0;y<=Nout;y++)
{
	for(h=0;h<Nhid;h++)
	{
		W_hy[h][y] = random_value();
		dW_hy[h][y] = 0;
	}
}
for(h=0;h<Nhid;h++)
{
	Q_h[h] = 0;
	dQ_h[h] = 0;
	delta_h[h] = 0;
}
for(y=0;y<Nout;y++)
{
	Q_y[y] = random_value();
	dQ_y[y] = 0;
	delta_y[y] = 0;
}
	for(Icycle=0;Icycle<Ncycle;Icycle++)
{
	mse = 0.0;
```

```
// input training sample...
fseek(train_fp, 0, 0);

//// only for specific data file format!!!
do
{
	fscanf(train_fp, "%c", &c);
}while(c != 0x0A);
//// only for specific data file format!!!

for(Itrain=0;Itrain<Ntrain;Itrain++)
{
	for(x=0;x<Ninp;x++)
		fscanf(train_fp, "%f", &X[x]);

	for(y=0;y<Nout;y++)
		fscanf(train_fp, "%f", &T[y]);

	// compute H, Y...
	for(h=0;h<Nhid;h++)
	{
		sum = 0.0;

		for(x=0;x<Ninp;x++)
			sum = sum + X[x]*W_xh[x][h];

		H[h] = (float)1.0 / (1.0 + exp(-(sum-Q_h[h])));
	}

	for(y=0;y<Nout;y++)
	{
		sum = 0.0;

		for(h=0;h<Nhid;h++)
			sum = sum + H[h]*W_hy[h][y];

		Y[y] = (float)1.0 / (1.0 + exp(-(sum-Q_y[y])));
	}
```

```
// compute delta...
for(y=0;y<Nout;y++)
	delta_y[y] = Y[y] * (1.0-Y[y]) * (T[y]-Y[y]);

for(h=0;h<Nhid;h++)
{
	sum = 0.0;

	for(y=0;y<Nout;y++)
		sum = sum + W_hy[h][y]*delta_y[y];

	delta_h[h] = H[h] * (1.0-H[h]) * sum;
}

// compute dW, dQ...
for(y=0;y<Nout;y++)
	for(h=0;h<Nhid;h++)
		dW_hy[h][y] = eta*delta_y[y]*H[h] + alpha*dW_hy[h][y];

for(y=0;y<Nout;y++)
	dQ_y[y] = (-eta)*delta_y[y] + alpha*dQ_y[y];

for(h=0;h<Nhid;h++)
	for(x=0;x<Ninp;x++)
		dW_xh[x][h] = eta*delta_h[h]*X[x] + alpha*dW_xh[x][h];

for(h=0;h<Nhid;h++)
	dQ_h[h] = (-eta)*delta_h[h] + alpha*dQ_h[h];

// compute new W, Q...
for(y=0;y<Nout;y++)
	for(h=0;h<Nhid;h++)
		W_hy[h][y] = W_hy[h][y] + dW_hy[h][y];

for(y=0;y<Nout;y++)
	Q_y[y] = Q_y[y] + dQ_y[y];
```

```
            for(h=0;h<Nhid;h++)
                for(x=0;x<Ninp;x++)
                    W_xh[x][h] = W_xh[x][h] + dW_xh[x][h];

            for(h=0;h<Nhid;h++)
                Q_h[h] = Q_h[h] + dQ_h[h];

            for(y=0;y<Nout;y++)
                mse += (T[y]-Y[y])*(T[y]-Y[y]);
        }
        mse = mse / Ntrain;

        if(Icycle == 0)
            printf("\nIcycle = %4d, mse = %-8.4f", Icycle+1, mse);

        if((Icycle%10) == 9)
            printf("\nIcycle = %4d, mse = %-8.4f", Icycle+1, mse);

        fprintf(mse_fp, "%4d %-8.4f\n", Icycle+1, mse);
}
printf("\n");

for(h=0;h<Nhid;h++)
{       for(x=0;x<Ninp;x++)
        {
            fprintf(weight_fp, "%-8.2f", W_xh[x][h]);
        }
        fprintf(weight_fp, "\n");
}
fprintf(weight_fp, "\n");

for(y=0;y<Nout;y++)
{
        for(h=0;h<Nhid;h++)
        {
            fprintf(weight_fp, "%-8.2f", W_hy[h][y]);
        }
        fprintf(weight_fp, "\n");
```

```
        }
        fprintf(weight_fp, "\n");

        for(h=0;h<Nhid;h++)
        {
                fprintf(weight_fp, "%-8.2f", Q_h[h]);
        }
        fprintf(weight_fp, "\n\n");

        for(y=0;y<Nout;y++)
        {
                fprintf(weight_fp, "%-8.2f", Q_y[y]);
        }
        fprintf(weight_fp, "\n\n");
        fclose(train_fp);
        fclose(weight_fp);
        fclose(mse_fp);

        cout<<endl<<"Training successful!"<<endl;

      return EXIT_SUCCESS;
}

float random_value(void)
{
        return ((rand()/(RAND_MAX+1.0))-0.5);
}
```

附錄 B: 類神經網路之 C 程式語言 (Testing)

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

using namespace std;
int main(int argc, char *argv[])
{
      if(argc == 1)
       {
            printf("Usage: \n");
            printf(">NNRECALL.exe (-t/-i/-h/-o) weight_file test_file recall_file \n");
            printf("      where: -t = # of test sample [default = AUTO] \n");
            printf("                  -i = # of input nodes [default = 4]\n");
            printf("                  -h = # of hidden nodes [default = 6] \n");
            printf("                  -o = # of output nodes [default = 2]\n\n");

            system("PAUSE");
            return EXIT_SUCCESS;
       }
            FILE *weight_fp, *test_fp, *recall_fp;

       if(argc >= 4)
       {
            weight_fp = fopen(argv[argc-3], "r");
            test_fp = fopen(argv[argc-2], "r");
            recall_fp = fopen(argv[argc-1], "w");

            if(weight_fp == NULL)
            {
                  cout<<"\nError! Weight file not exist!"<<endl;
                  system("PAUSE");
                  return EXIT_FAILURE;
            }
            if(test_fp == NULL)
            {
```

```
            cout<<"\nError! Test file not exist!"<<endl;
            system("PAUSE");
            return EXIT_FAILURE;
        }
}
else
{
        cout<<"\nError! Please indicate the files. See Usage.\n"<<endl;
        system("PAUSE");
        return EXIT_FAILURE;
}
int Ntest = -1;
int Ninp = 4;
int Nout = 2;
int Nhid = 6;

for(int i=1;i<argc-3;i++)
{
        if(argv[i][0] == '-')
        {
            switch(argv[i][1])
            {
             case 't':
                    Ntest = atoi(argv[i+1]);
                    break;
             case 'i':
                    Ninp = atoi(argv[i+1]);
                    break;
             case 'h':
                    Nhid = atoi(argv[i+1]);
                    break;
             case 'o':
                    Nout = atoi(argv[i+1]);
                    break;
             default:
                    break;
             }
        }
```

```
}

char c = 0x00;
	if(Ntest == -1)
{
	Ntest = 0;

	fseek(test_fp, 0, SEEK_SET);

	while(fscanf(test_fp, "%c", &c) != -1)
	{
		if(c == 0x0A)
			Ntest++;
	}

	fseek(test_fp, 1, SEEK_END);
	fscanf(test_fp, "%c", &c);
	if(c != 0x0A && c != 0x0D)
		Ntest++;
	Ntest--;	//// only for specific data file format!!!
}

printf("\nNtest=%d Ninp=%d Nhid=%d Nout=%d \n\n", Ntest, Ninp, Nhid, Nout);

float X[Ninp], T[Nout], H[Nhid], Y[Nout];
float W_xh[Ninp][Nhid], W_hy[Nhid][Nout];
float Q_h[Nhid], Q_y[Nout];
float sum, mse;
float avg_mse = 0.0;
int Itest;
int x, h, y;

fseek(weight_fp, 0, 0);

for(h=0;h<Nhid;h++)
	for(x=0;x<Ninp;x++)
		fscanf(weight_fp, "%f", &W_xh[x][h]);
for(y=0;y<Nout;y++)
```

```
        for(h=0;h<Nhid;h++)
                fscanf(weight_fp, "%f", &W_hy[h][y]);

for(h=0;h<Nhid;h++)
        fscanf(weight_fp, "%f", &Q_h[h]);
for(y=0;y<Nout;y++)
        fscanf(weight_fp, "%f", &Q_y[y]);

fseek(test_fp, 0, 0);

//// only for specific data file format!!!
do
{
        fscanf(test_fp, "%c", &c);
}while(c != 0x0A);
//// only for specific data file format!!!

for(Itest=0; Itest<Ntest; Itest++)
{
        //input test data
        for(x=0; x<Ninp; x++)
                fscanf(test_fp, "%f", &X[x]);
        for(y=0; y<Nout; y++)
                fscanf(test_fp, "%f", &T[y]);

        for(h=0; h<Nhid; h++)
        {
                sum = 0.0;
                for(x=0; x<Ninp; x++)
                        sum = sum + X[x]*W_xh[x][h];
                H[h] = (float)1.0 / (1.0 + exp(-(sum-Q_h[h])));
        }
        for(y=0; y<Nout; y++)
        {
                sum = 0.0;
                for(h=0; h<Nhid; h++)
                        sum = sum + H[h]*W_hy[h][y];
                Y[y] = (float)1.0 / (1.0 + exp(-(sum-Q_y[y])));
```

```
        }
                mse = 0.0;

        for(y=0; y<Nout; y++)
                mse += (T[y]-Y[y])*(T[y]-Y[y]);

        avg_mse += mse;

        printf("%3d: Test = ", Itest+1);
        fprintf(recall_fp, "%3d      T      ", Itest+1);
        for(y=0; y<Nout; y++)
        {
                printf("%-8.4f", T[y]);
                fprintf(recall_fp, "%-8.4f", T[y]);
        }

        printf("Recall = ");
        fprintf(recall_fp, "R      ");
        for(y=0; y<Nout; y++)
        {
                printf("%-8.4f", Y[y]);
                fprintf(recall_fp, "%-8.4f", Y[y]);
        }

        printf("    mse = %-8.4f\n", mse);
        fprintf(recall_fp, "    m      %-8.4f\n", mse);
    }

    avg_mse = avg_mse / Ntest;

    printf("\nAverage mean square error (mse) = %-8.4f\n", avg_mse);
    fprintf(recall_fp, "\navg_mse    %-8.4f\n", avg_mse);
    fclose(weight_fp);
    fclose(test_fp);
    fclose(recall_fp);
    cout<<endl<<"Recalling successful!"<<endl;
   return EXIT_SUCCESS;
}
```

附錄 C: 基因演算法之 C 程式語言

```
#include <cstdlib>
#include <iostream>

#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <math.h>

using namespace std;

float a = 0.0;
float b = 1.0;

unsigned int g_cycle;
float m_rate;
int m_cycle;
int population, length;
int Ninp, Nhid, Nout;
float w_warp, w_shrin;
float x_warp, x_shrin;
int *farm;
float *x_value, *y_value, *f_value;
int f_cycle;

float *H, *W_xh, *W_hy, *Q_h, *Q_y;
FILE *weight_fp, *output_fp;
void setup(void)
{
        time_t t;
        int i, j, k, x, h, y;
        srand((unsigned) time(&t));
        for(i=0; i<population; i++)
                for(j=0; j<Ninp; j++)
                        for(k=0; k<length; k++)
                                farm[i*Ninp*length + j*length + k] = rand() % 2;
        fseek(weight_fp, 0, 0);
```

```
	for(h=0;h<Nhid;h++)
		for(x=0;x<Ninp;x++)
			fscanf(weight_fp, "%f", &W_xh[x*Nhid + h]);
	for(y=0;y<Nout;y++)
		for(h=0;h<Nhid;h++)
			fscanf(weight_fp, "%f", &W_hy[h*Nout + y]);

	for(h=0;h<Nhid;h++)
		fscanf(weight_fp, "%f", &Q_h[h]);
	for(y=0;y<Nout;y++)
		fscanf(weight_fp, "%f", &Q_y[y]);
}

void print_farm(void)	{
	int i, j, k;

	for(i=0; i<population; i++)
	{
		printf("farm[%2d] = ", i+1);
		for(j=0; j<Ninp; j++)
		{
			for(k=0; k<length; k++)
				printf(" %d ", farm[i*Ninp*length + j*length + k]);
			printf("\n                    ");
		}
		printf("\n");
	}
}

void compute_f_value(int pop)
{
	int i, j, k, x, h, y;
	float m;
	float sum;
	for(i=0; i<Ninp; i++)
	{
		m = 0.0;
		for(j=0; j<length; j++)
```

```
            {
                k = j;
                m += farm[pop*Ninp*length + i*length + (length-j-1)] * pow(2.0, k);
            }
            x_value[pop*Ninp + i] = a + m*(b-a)/(pow(2.0, length)-1.0);
        }

        for(h=0; h<Nhid; h++)
        {
            sum = 0.0;
            for(x=0; x<Ninp; x++)
                sum = sum + x_value[pop*Ninp + x] * W_xh[x*Nhid + h]; //X[x]*W_xh[x][h];
            H[h] = (float)1.0 / (1.0 + exp(-(sum - Q_h[h])));
        }
        for(y=0; y<Nout; y++)
        {
            sum = 0.0;
            for(h=0; h<Nhid; h++)
                sum = sum + H[h] * W_hy[h*Nout + y]; //H[h]*W_hy[h][y];
            y_value[pop*Nout + y] = (float)1.0 / (1.0 + exp(-(sum - Q_y[y])));
        }

        f_value[pop] = (w_warp*y_value[pop*Nout + 0] + w_shrin*y_value[pop*Nout + 1]) / (w_warp +
w_shrin);
}
void print_value(void)
{
        int i, j, k;
        for(i=0; i<population; i++)
        {
            printf(" INPUT[%2d] =", i+1);
            for(j=0; j<Ninp; j++)
            {
                for(k=0; k<length; k++)
                    printf(" %d ", farm[i*Ninp*length + j*length + k]);
                printf("= %-2.4f ", x_value[i*Ninp + j]);
                if(j != Ninp-1)
                    printf("\n                    ");
```

```
            }
            printf("\nOUTPUT[%2d] = %-2.4f * %-2.2f\n                    %-2.4f", i+1, y_value[i*2+0],
(w_warp/(w_warp+w_shrin)), y_value[i*2+1]);
            printf(" * %-2.2f += %-2.4f \n\n", (w_shrin/(w_warp+w_shrin)), f_value[i]);
        }
}

void crossing(void)
{       int r0, r1, rc, j;
        r0 = rand() % population;
        r1 = rand() % population;
        while(r0 == r1)
                r1 = rand() % population;
        for(j=0; j<(Ninp*length); j++)
        {
                farm[(population+0)*(Ninp*length) + j] = farm[r0*Ninp*length + j];
                farm[(population+1)*(Ninp*length) + j] = farm[r1*Ninp*length + j];
        }
        rc = rand() % (Ninp*length - 2) + 1;
        for(j=rc; j<(Ninp*length); j++)
        {
                farm[(population+0)*(Ninp*length) + j] = farm[r1*Ninp*length + j];
                farm[(population+1)*(Ninp*length) + j] = farm[r0*Ninp*length + j];
        }
        compute_f_value(population+0);
        compute_f_value(population+1);
        if(f_value[population+0] < f_value[r0])
        {
                if((x_shrin < 0.0) && (x_warp < 0.0))                   {
                        for(j=0; j<(Ninp*length); j++)
                                farm[r0*(Ninp*length) + j] = farm[(population+0)*(Ninp*length) + j];
                        compute_f_value(r0);
                }
                else if((x_warp >= 0.0) && (x_shrin < 0.0) && (y_value[(population+0)*Nout + 0] <= x_warp))
                {
                        for(j=0; j<(Ninp*length); j++)
                                farm[r0*(Ninp*length) + j] = farm[(population+0)*(Ninp*length) + j];
                        compute_f_value(r0);
```

```
            }
            else if((x_shrin >= 0.0) && (x_warp < 0.0) && (y_value[(population+0)*Nout + 1] <= x_shrin))
            {
                for(j=0; j<(Ninp*length); j++)
                    farm[r0*(Ninp*length) + j] = farm[(population+0)*(Ninp*length) + j];
                compute_f_value(r0);
            }
            else
            {
                if((y_value[(population+0)*Nout + 0] <= x_warp) && (y_value[(population+0)*Nout + 1] <=
x_shrin))
                {
                    for(j=0; j<(Ninp*length); j++)
                        farm[r0*(Ninp*length) + j] = farm[(population+0)*(Ninp*length) + j];
                    compute_f_value(r0);
                }
            }
        }

        if(f_value[population+1] < f_value[r1])
        {
            if((x_shrin < 0.0) && (x_warp < 0.0))      // no limitation
            {
                for(j=0; j<(Ninp*length); j++)
                    farm[r1*(Ninp*length) + j] = farm[(population+1)*(Ninp*length) + j];
                compute_f_value(r1);
            }
            else if((x_warp >= 0.0) && (x_shrin < 0.0) && (y_value[(population+1)*Nout + 0] <= x_warp))
            {
                for(j=0; j<(Ninp*length); j++)
                    farm[r1*(Ninp*length) + j] = farm[(population+1)*(Ninp*length) + j];
                compute_f_value(r1);
            }
            else if((x_shrin >= 0.0) && (x_warp < 0.0) && (y_value[(population+1)*Nout + 1] <= x_shrin))
            {
                for(j=0; j<(Ninp*length); j++)
                    farm[r1*(Ninp*length) + j] = farm[(population+1)*(Ninp*length) + j];
                compute_f_value(r1);
```

```
            }
            else   // x_warp >= 0.0 && x_shrin >= 0.0
            {
                if((y_value[(population+1)*Nout + 0] <= x_warp) && (y_value[(population+1)*Nout + 1] <=
x_shrin))
                {
                    for(j=0; j<(Ninp*length); j++)
                        farm[r1*(Ninp*length) + j] = farm[(population+1)*(Ninp*length) + j];
                    compute_f_value(r1);
                }
            }
        }
}

void mutate(void)
{
        int i, rp, rm;
        rp = rand() % population;
        rm = rand() % (Ninp*length);
        for(i=0; i<(Ninp*length); i++)
        farm[(population+0)*(Ninp*length) + i] = farm[rp*Ninp*length + i];
        farm[(population+0)*(Ninp*length) + rm] = 1 - farm[(population+0)*(Ninp*length) + rm];
        compute_f_value(population+0);

        if(f_value[population+0] < f_value[rp])
        {
            if((x_shrin < 0.0) && (x_warp < 0.0))                    {
                for(i=0; i<(Ninp*length); i++)
                    farm[rp*(Ninp*length) + i] = farm[(population+0)*(Ninp*length) + i];
                compute_f_value(rp);
            }
            else if((x_warp >= 0.0) && (x_shrin < 0.0) && (y_value[(population+0)*Nout + 0] <= x_warp))
            {
                for(i=0; i<(Ninp*length); i++)
                farm[rp*(Ninp*length) + i] = farm[(population+0)*(Ninp*length) + i];
                compute_f_value(rp);
            }
            else if((x_shrin >= 0.0) && (x_warp < 0.0) && (y_value[(population+0)*Nout + 1] <= x_shrin))
```

```
            {
                for(i=0; i<(Ninp*length); i++)
                    farm[rp*(Ninp*length) + i] = farm[(population+0)*(Ninp*length) + i];
                compute_f_value(rp);
            }
            else                {
                if((y_value[(population+0)*Nout + 0] <= x_warp) && (y_value[(population+0)*Nout + 1] <=
x_shrin))
                {
                    for(i=0; i<(Ninp*length); i++)
                        farm[rp*(Ninp*length) + i] = farm[(population+0)*(Ninp*length) + i];
                    compute_f_value(rp);
                }
            }
        }
}

void print_best(int cycle, int fout)
{       int i, j;
        float x_best[Ninp], y_best[Nout], f_best;
        for(i=0; i<Ninp; i++)
            x_best[i] = x_value[0*Ninp + i];
        for(i=0; i<Nout; i++)
            y_best[i] = y_value[0*Nout + i];
        f_best = f_value[0];
        for(j=1; j<population; j++)
        {
            if(f_value[j] < f_best)
            {
                for(i=0; i<Ninp; i++)
                    x_best[i] = x_value[j*Ninp + i];
                for(i=0; i<Nout; i++)
                    y_best[i] = y_value[j*Nout + i];
                f_best = f_value[j];
            }
        }

        printf("%6d: ", cycle);
```

```
        printf("X ");
        for(i=0; i<Ninp; i++)
        printf("%-2.4f ", x_best[i]);
        printf("Y ");
        for(i=0; i<Nout; i++)
        printf("%-2.4f ", y_best[i]);
        printf("F %-2.4f\n", f_best);
        if(fout == 1)
        {
                fprintf(output_fp, "%8d: ", cycle);
                fprintf(output_fp, "X ");
                for(i=0; i<Ninp; i++)
                        fprintf(output_fp, "%-2.4f ", x_best[i]);
                fprintf(output_fp, "Y ");
                for(i=0; i<Nout; i++)
                        fprintf(output_fp, "%-2.4f ", y_best[i]);
                fprintf(output_fp, "F %-2.4f\n", f_best);
        }
}
void limitation_check(void)
{
        int i, j, k;
        if((x_warp >= 0.0) && (x_shrin >= 0.0))
        {
                for(i=0; i<population; i++)
                {
                        while((y_value[i*Nout + 0] > x_warp) || (y_value[i*Nout + 1] > x_shrin))
                        {
                                for(j=0; j<Ninp; j++)
                                        for(k=0; k<length; k++)
                                                farm[i*Ninp*length + j*length + k] = rand() % 2;
                                compute_f_value(i);
                        }
                }
        }
        else if(x_shrin < 0.0)
        {
                for(i=0; i<population; i++)
```

```
        {
            while(y_value[i*Nout + 0] > x_warp)
            {
                for(j=0; j<Ninp; j++)
                    for(k=0; k<length; k++)
                        farm[i*Ninp*length + j*length + k] = rand() % 2;
                compute_f_value(i);
            }
        }
    }
    else if(x_warp < 0.0)
    {
        for(i=0; i<population; i++)
        {
            while(y_value[i*Nout + 1] > x_shrin)
            {
                for(j=0; j<Ninp; j++)
                    for(k=0; k<length; k++)
                        farm[i*Ninp*length + j*length + k] = rand() % 2;
                compute_f_value(i);
            }
        }
    }
    else
    {}
}

int main(int argc, char *argv[])       // main program
{   if(argc == 1)
    {
        printf("Usage: \n");
        printf(">NNGA.exe (-c/-p/-l/-m/-h/-ww/-ws/-xw/-xs/-f) NNweight_file GA_output_file \n");
        printf("     where: -c = # of generation cycle [default = 10000] \n");
        printf("            -p = # of genes population [default = 12]\n");
        printf("            -l = # of genes length [default = 10] \n");
        printf("            -m = # of mutation rate [default = 1.0] \n");
        printf("            -h = # of NN hidden nodes [default = 6] \n");
        printf("            -ww = weight of the warp output parameter [default = 1.0] \n");
```

```
        printf("        -ws = weight of the shrin output parameter [default = 1.0] \n");
        printf("        -xw = max limit of the warp output [default = -1 (no limitation)] \n");
        printf("        -xs = max limit of the shrin output [default = -1 (no limitation)] \n");
        printf("        -f = file output cycle [default = 10] \n\n");

        system("PAUSE");
        return EXIT_SUCCESS;
}
if(argc >= 3)
{
        weight_fp = fopen(argv[argc-2], "r");
        output_fp = fopen(argv[argc-1], "w");
        if(weight_fp == NULL)
        {
                cout<<"\nError! Weight file not exist!"<<endl;
                system("PAUSE");
                return EXIT_FAILURE;
        }
}
else
{
        cout<<"\nError! Please indicate the files. See Usage.\n"<<endl;
        system("PAUSE");
        return EXIT_FAILURE;
}
g_cycle = 10000;
m_rate = 1.0;
population = 12;
length = 10;
Ninp = 4;
Nhid = 6;
Nout = 2;
w_warp = 1.0;
w_shrin = 1.0;
x_warp = -1.0;
x_shrin = -1.0;
f_cycle = 10;
```

```
for(int i=1;i<argc-2;i++)
{
	if(argv[i][0] == '-')
	{
		switch(argv[i][1])
		{
		 case 'c':
			g_cycle = atoi(argv[i+1]);
			break;
		 case 'p':
			population = atoi(argv[i+1]);
			break;
		 case 'l':
			length = atoi(argv[i+1]);
			break;
		 case 'm':
			m_rate = atof(argv[i+1]);
			break;
		 case 'h':
			Nhid = atoi(argv[i+1]);
			break;
		 case 'w':
			switch(argv[i][2])
			{
			case 'w':
				w_warp = atof(argv[i+1]);
				break;
			case 's':
				w_shrin = atof(argv[i+1]);
				break;
			default:
				break;
			}
			break;
		 case 'x':
			switch(argv[i][2])
			{
				case 'w':
```

```
                        x_warp = atof(argv[i+1]);
                        break;
                    case 's':
                        x_shrin = atof(argv[i+1]);
                        break;
                    default:
                        break;
                }
                break;
            case 'f':
                f_cycle = atoi(argv[i+1]);
                break;
            default:
                break;
            }
        }
}

m_cycle = (int)(1.0 / m_rate);
if(m_cycle == 0)
m_cycle = 1;
farm = new int[(population+2) * Ninp * length];
x_value = new float[(population+2) * Ninp];
y_value = new float[(population+2) * Nout];
f_value = new float[population+2];
H = new float[Nhid];
W_xh = new float[Ninp * Nhid];
W_hy = new float[Nhid * Nout];
Q_h = new float[Nhid];
Q_y = new float[Nout];

printf("\n GAcycle = %d, GPopulation = %d, GLength = %d \n", g_cycle, population, length);
printf(" MutationCycle = %d, WarpWeight = %f, ShrinWeight = %f \n", m_cycle, w_warp, w_shrin);
printf(" MaxWarpLimitaion = ");
if(x_warp == -1.0)
        printf("None, ");
else
        printf("%f, ", x_warp);
```

```
	printf("MaxShrinLimitation = ");
	if(x_shrin == -1.0)
		printf("None ");
	else
		printf("%f ", x_shrin);
	printf("\n");
	printf(" NN_Input = %d, NN_Hidden = %d, NN_Output = %d, FileOutput_cycle = %d\n\n", Ninp, Nhid,
Nout, f_cycle);
	int i, j;

	setup();

	for(i=0; i<population; i++)
		compute_f_value(i);

	if(x_warp >= 0.0 || x_shrin >= 0.0)
		limitation_check();

	print_value();
	print_best(1, 0);
	system("PAUSE");

	for(j=0; j<g_cycle; j++)
	{
		crossing();
		if(j % m_cycle == 0)
			mutate();
		if(j % f_cycle == 0)
			print_best(j+1, 1);
	}

	printf("\n");
	print_value();
	print_best(g_cycle, 1);

	free(farm);
	free(x_value);
	free(y_value);
```

```
        free(f_value);
        free(H);
        free(W_xh);
        free(W_hy);
        free(Q_h);
        free(Q_y);

        fclose(weight_fp);
        fclose(output_fp);

    return EXIT_SUCCESS;
}
```

附錄 D: 收縮限制下之最小翹曲搜尋過程

收縮限制值	MET	Warpage	Shrinkage
0.3945			
0.3950			
0.3955			
0.3960			
0.3965			
0.3970			
0.3975			
0.3980			
0.3985	0.2493	0.3801	0.3943
0.3990	0.3744	0.3779	0.3949
0.3995	0.3978	0.3779	0.3954
0.4000	0.3744	0.3779	0.3949
0.4005	0.3744	0.3779	0.3949
0.4010	0.4008	0.3779	0.3955
0.4015	0.3744	0.3779	0.3949
0.4020	0.3744	0.3779	0.3949
0.4025	0.3744	0.3779	0.3949
0.4030	0.4008	0.3779	0.3955
0.4035	0.3744	0.3779	0.3949
0.4040	0.3744	0.3779	0.3949
0.4045	0.3978	0.3779	0.3954
0.4050	0.3744	0.3779	0.3949
0.4055	0.3744	0.3779	0.3949
0.4060	0.3744	0.3779	0.3949
0.4065	0.4008	0.3779	0.3955
0.4070	0.3744	0.3779	0.3949
0.4075	0.3744	0.3779	0.3949
0.4080	0.3744	0.3779	0.3949
0.4085	0.3744	0.3779	0.3949
0.4090	0.4008	0.3779	0.3955
0.4095	0.4066	0.3779	0.3957
0.4100	0.4008	0.3779	0.3955
0.4105	0.4008	0.3779	0.3955
0.4110	0.3744	0.3779	0.3949

收縮限制值	MET	Warpage	Shrinkage
0.4115	0.3744	0.3779	0.3949
0.4120	0.3978	0.3779	0.3954
0.4125	0.4027	0.3779	0.3956
0.4130	0.4008	0.3779	0.3955
0.4135	0.3744	0.3779	0.3949
0.4140	0.3744	0.3779	0.3949
0.4145	0.4008	0.3779	0.3955
0.4150	0.3744	0.3779	0.3949
0.4155	0.4066	0.3779	0.3957
0.4160	0.3744	0.3779	0.3949
0.4165	0.3744	0.3779	0.3949
0.4170	0.3744	0.3779	0.3949
0.4175	0.4008	0.3779	0.3955
0.4180	0.4008	0.3779	0.3955
0.4185	0.4008	0.3779	0.3955
0.4190	0.3744	0.3779	0.3949
0.4195	0.4066	0.3779	0.3957
0.4200	0.3744	0.3779	0.3949
0.4205	0.4008	0.3779	0.3955
0.4210	0.3744	0.3779	0.3949
0.4215	0.4008	0.3779	0.3955
0.4220	0.4008	0.3779	0.3955
0.4225	0.3978	0.3779	0.3954
0.4230	0.3744	0.3779	0.3949
0.4235	0.3744	0.3779	0.3949
0.4240	0.3744	0.3779	0.3949
0.4245	0.4008	0.3779	0.3955
0.4250	0.3744	0.3779	0.3949
0.4255	0.3744	0.3779	0.3949
0.4260	0.3978	0.3779	0.3954
0.4265	0.4027	0.3779	0.3956
0.4270	0.4008	0.3779	0.3955
0.4275	0.4066	0.3779	0.3957
0.4280	0.3978	0.3779	0.3954
0.4285	0.3978	0.3779	0.3954
0.4290	0.3744	0.3779	0.3949

收縮限制值	MET	Warpage	Shrinkage
0.4295	0.3744	0.3779	0.3949
0.4300	0.4008	0.3779	0.3955
0.4305	0.3978	0.3779	0.3954
0.4310	0.4008	0.3779	0.3955
0.4315	0.4066	0.3779	0.3957
0.4320	0.3744	0.3779	0.3949
0.4325	0.3744	0.3779	0.3949
0.4330	0.3744	0.3779	0.3949
0.4335	0.4008	0.3779	0.3955
0.4340	0.4027	0.3779	0.3956
0.4345	0.3744	0.3779	0.3949
0.4350	0.3744	0.3779	0.3949
0.4355	0.4008	0.3779	0.3955
0.4360	0.3744	0.3779	0.3949
0.4365	0.3744	0.3779	0.3949
0.4370	0.4066	0.3779	0.3957
0.4375	0.4066	0.3779	0.3957
0.4380	0.3744	0.3779	0.3949
0.4385	0.3978	0.3779	0.3954
0.4390	0.3744	0.3779	0.3949
0.4395	0.3744	0.3779	0.3949
0.4400	0.4008	0.3779	0.3955
0.4405	0.3744	0.3779	0.3949
0.4410	0.3744	0.3779	0.3949
0.4415	0.3744	0.3779	0.3949
0.4420	0.3744	0.3779	0.3949
0.4425	0.3744	0.3779	0.3949
0.4430	0.3744	0.3779	0.3949
0.4435	0.4008	0.3779	0.3955
0.4440	0.4066	0.3779	0.3957
0.4445	0.3744	0.3779	0.3949
0.4450	0.3744	0.3779	0.3949
0.4455	0.3744	0.3779	0.3949
0.4460	0.3744	0.3779	0.3949
0.4465	0.4008	0.3779	0.3955
0.4470	0.4027	0.3779	0.3956

收縮限制值	MET	Warpage	Shrinkage
0.4475	0.4008	0.3779	0.3955
0.4480	0.4008	0.3779	0.3955
0.4485	0.3978	0.3779	0.3954
0.4490	0.4008	0.3779	0.3955
0.4495	0.3978	0.3779	0.3954
0.4500	0.4008	0.3779	0.3955
0.4505	0.3744	0.3779	0.3949
0.4510	0.3744	0.3779	0.3949
0.4515	0.3744	0.3779	0.3949
0.4520	0.4008	0.3779	0.3955
0.4525	0.4008	0.3779	0.3955
0.4530	0.4008	0.3779	0.3955
0.4535	0.4066	0.3779	0.3957
0.4540	0.4008	0.3779	0.3955
0.4545	**0.5005**	**0.3787**	**0.399**

附錄 E:　BPNN 訓練之權重值及偏權值

Weight

Input Layer → Hidden Layer

0.2043　0.8241　-0.6716 -0.8055

-1.1695 -2.7530 -0.2209 -0.8753

0.2753　-2.1565 -0.4676 -1.8845

-1.2005 -1.0815 0.1116　1.7150

-0.6788 -0.8951 0.1270　-0.5685

-0.2486 -0.9225 -0.1917 -0.5601

Hidden Layer → Output Layer

-0.9218　0.3865　1.6880　0.8776　0.6036　0.7087

-0.8619　2.2297　1.8189　1.2512　0.7130　0.3096

Bias

Input Layer → Hidden Layer

0.6579　-0.0276　-0.9248　-0.9266　-0.0439　0.0039

Hidden Layer → Output Layer

0.8168　0.8829