國 立 交 通 大 學

機 械 工 程 研 究 所

碩 士 論 文

金線密度對偏移之影響分析

Analysis on the Effect of Wire Density on Wire Sweep

研 究 生：簡維德

指導教授：洪景華　教授

中 華 民 國 九 十 四 年 六 月

金線密度對偏移之影響分析

Analysis on the Effect of Wire Density on Wire Sweep

研 究 生：簡維德　　**Student：Wei-Ti Chien**

指導教授：洪景華 教授　　**Advisor：Prof. Ching-Hua Hung**

國 立 交 通 大 學

機 械 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Mechanical Engineering

College of Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Mechanical Engineering

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

金線密度對偏移之影響分析

研究生：簡維德　　指導教授：洪景華 教授

國立交通大學機械工程研究所

摘要

現今電子產品多以朝向輕、薄、短、小與多功能趨勢發展，使得IC元件薄小化成為必要的發展技術，除了IC製程技術突破，提供晶片保護並作為訊號傳輸媒介的IC封裝技術也必須隨之跟進，對於金線偏移的預測與控制便是一例。當元件尺寸越小I/O數越高的情況，金線偏移量所引發的問題就漸漸浮上台面，成為封裝所必須克服的技術關鍵，其中以數值模擬預測並透過修正製程參數或模穴設計改善金線偏移情況，便是一快速並節省成本的方法。

本研究參考過去的文獻中所提出的金線偏移計算方法，針對高密度金線封裝提出一套可行的計算流程。另仿造過去傳統的金線偏移分析方法，計算不同金線分佈下的偏移量，與本研究中提出的金線偏移分析方法所得到的結果比較，討論不同金線密度下金線偏移情形。

Analysis on the Effect of Wire Density on Wire Sweep

Student: Wei-Ti Chien Advisor: Prof. Ching-Hua Hung

Institute of Mechanical Engineering

National Chiao Tung University

Abstract

Because the electric products trend to become lighter, thinner, smaller and multi-functional, semiconductor industry has to focus on IC chip scale-down. Meanwhile, IC packaging technology must also become more advanced to be compatible with this trend. Wires weep is one of the important issues in packaging technology. With the size-decreased chips and the number-increased I/Os, the problems due to wire sweep become more and more serious and often reduce the yield of products; so to control the amount of sweep within an acceptable range become one of the key points in advanced packaging technology. Currently, the most cost effective solution to predict and correct wire sweep is to use numerical simulation to obtain the optimum process parameters and mold cavity design.

A new simulation scheme for wire sweep analysis evolved from traditional methods was be discussed in this research. A comparison of results from both simulation methods was conducted for different wire distribution densities. The change of the wire sweep behavior according to the wire distribution density was concluded and discussed.

誌謝

在這短短兩年攻讀碩士學位的日子裡，首先要感謝指導老師洪景華教授。對於專業領域知識、日常生活常識、做人做事道理、攝影技術等等，老師從不吝於教導以及分享，對於學生我的怠惰，從未以嚴飭的言語責罵，而是在循循善誘下讓學生知道該積極進取，另外，老師十分尊重學生的想法，使我能夠在學習過程中能夠向著自己興趣的方向發展，遇到障礙時，老師也能夠點出問題的關鍵所在，讓研究能夠順利進行與完成。

在研究室裡，感謝榮崇、中興以及宇中學長平時的指導以及照顧；感謝琇晶與智偉兩位同學，在兩年內陪著我共患難，還有感謝學弟麒禎、嘉偉和銘傑，當我埋頭於研究及論文寫作時，能夠幫忙處裡手邊雜事。

在研究室外，感謝成功大學、交通大學與清華大學游泳隊的夥伴們，能夠在大學以及碩士班裡加入游泳校隊，和大家啃著訓練菜單的點點滴滴，將是這六年內，我最懷念也是最驕傲的事情。

另外，小小以及張傑鈞兩位從高中時代到現在的老朋友，感謝你們一直記得這位被遠放外地六年的朋友，雖然課業上無法替我分勞，但在精神上卻能替我解憂；還要感謝一位讓我立志繼續攻讀碩士學位的淑鈴學妹，不經意的一句話卻成為我的座右銘--我要做的，我都做得到。

最後，要感謝爸爸以及媽媽一直努力工作，讓我在求學過程中不曾為生活擔心，是我最可靠的後盾。我未來能有怎樣的成就，都是由你們的汗水所累積的。

感謝所有陪我走過這段時光以及曾經幫助過我的人，所有的文字都無法描述我對大家感激的心情。

目錄

表目錄

圖目錄

第一章　緒論

電晶體（Transistor）是電子產品最基本也是最重要的獨立元件，由Brattain、Bardeen以及Shockley於西元1949年在貝爾實驗室研發出。Jack Kilby在1959發展出包含二個電晶體以及一個電阻的微小電路，這是積體電子電路（Integrated circuit, IC）發展的開端。為了滿足電子產品功能與效能需求，不斷提升IC製程技術，使晶片上能夠容納更多的電晶體，從SSI（Small Scale Integration）、MSI（Medium Scale Integration）、LSI（Large Scale Integration）到VLSI（Very Large Integration），現在一晶片上已經能夠容納百萬以上的電晶體數。除了功能與效能提升，電子產品亦朝向輕薄短小的趨勢發展，這在消費性電子產品更為顯著，如行動電話、消費級數位相機等，不僅要求體積小，亦不斷增加功能。

IC元件就如同人類的大腦，對於支配裝置或個體扮演相當重要角色，大腦藉由神經傳導訊息、需要骨骼保護以及血管提供能量，而IC元件需要對外接收或發送訊息、需要牢靠的保護，也需要電能供給使其運作，這些就是封裝的功能，因此，沒有經過封裝的IC元件不會有任何作用。廣義的封裝包括：從完成研磨的晶圓上（Wafer）切割下的晶片封裝成一塊封裝模組之IC封裝；將單一或多個封裝模組及其他週邊所需的零組元件，焊接在一片設計製作完成的印刷電路板上之電路板組立；將電路板和其他週邊設備組合成為可使用的系統之系統組裝。其中IC封裝是最基層的步驟，也是封裝過程最重的一項製程。

1.1 IC封裝

IC封裝屬於產品後段的製程技術，常被誤認為並非重要的一環，實則不然。IC封裝技術所涵蓋的知識領域不亞於IC製程，包括電學、化學、機械、材料等，另外，所有的IC元件都需要經過封裝才能成為產品，IC封裝也支配著產品的效能、尺寸、可靠度以及成本，因此，IC封裝雖為後段製程，但卻也是具有相當影響且不可缺少的步驟。

IC封裝的功能包括：

1. 保護晶片，避免晶片受到外力破壞，或者遭外在環境污染。
2. 提供運作所需的電力。
3. 提供與外界間的訊號連結。
4. 幫助與外界的機械支撐結構。
5. 幫助晶片冷卻。

IC封裝接線方式可分為打線接合、捲帶自動接合與覆晶接合等三種方式，晶片包覆可分為積層陶瓷型、玻璃陶瓷型與塑膠封模型等三種方式。不同IC有不同的需求，因此必須依照封裝結構、材料、尺寸、I/O數及成本等提供最適當的封裝技術，滿足各種IC的需求。以打線接合（Wire Bonding）和塑料封模是目前最廣泛使用的封裝製程，配合各種封裝型態，如BGA（Ball Grid Array）、DIP（Dual In-line Package）等，甚至是目前最先進的SIP（System in Package），都能夠提供高可靠度低成本的製成方法。以下依此種封裝製程流程，大致說明其步驟：

1. 黏晶（Chip Attach）：黏晶之目的是將一顆顆之晶片置於基板上並以銀膠黏著固定。黏晶完成後之晶片則經由傳輸設備送至彈匣（Magazine）內，以送至下一製程進行打線接合。

2. 打線接合（Wire Bonding）：打線接合乃是將晶片上的接腳（Pad）以極細的金線（18～50μm），利用打線接合技術連接到基板之內引腳，藉此將IC晶片之電路訊號傳輸透過基板至外界。
3. 封模（Molding）：封模之主要目的為防止濕氣由外部侵入、提供機械支撐、內部產生熱量之去除及提供能夠手持之形體。其過程為將導線架置於框架上並預熱，將框架置於壓模機上的封裝模具上，再以樹脂充填並待硬化。
4. 剪切/成形：剪切之目的為將導線架上封模完成之晶片獨立分開，並把不需要的連接用材料及部份凸出之樹脂切除。成形之目的則是將外引腳壓成各種預先設計好之形狀，以便於裝置於電路版上使用。剪切與成形主要由一部衝壓機配上多套不同製程之模具，加上進料及出料機構所組成。
5. 印字：印字乃將字體印於封裝完的膠體之上，其目的在於註明商品之規格及製造者等資訊。
6. 檢驗：檢驗之目的為確定封裝完成之產品是否合於使用。典型的檢測包括拉力、剪力、溫度、壓力檢測、其他項目包括諸如外引腳之平整性、共面度、腳距、印字是否清晰及膠體是否有損傷等的外觀檢驗。

1.2 移轉模造成型

IC晶片大多以移轉模造（Transfer Molding）來進行封模的製程。在移轉模造成型程序（圖1-1）中，將預先成型的塊狀塑料—一般為環氧樹脂（Epoxy Molding Compound），放置於預熱箱中預熱一段時間到達某一溫度，再將塑料放入模缸內利用活塞加壓射出，塑料即開始流入澆道，通過澆口進入以置入晶片及基板的模穴中，塑料覆蓋晶片及基板，最後充滿整個模穴，待經冷卻硬化之後取出，初步的封裝即算完成。在填充的過程中，

常會有些缺陷產生的問題，包括：

1. 填充不完全：通常是因為流入模穴的前端塑料固化過快，使得後來進入的塑料無法完全填滿整個模穴。
2. 包風：當塑料留速不均時，流動過程會包住空氣進入模穴中，模造完成後，這些氣泡就成為沒有塑料的空洞。
3. 晶片偏移：塑料流經晶片時的壓力差造成晶片脫落黏貼的基板，或是晶片導架因塑料流過而發生變形。
4. 爆裂（Popcorn）現象：模造封裝時，晶片與樹脂，配線與塑料之間含有水分，當焊接時會加熱至100℃以上，水立即汽化膨脹，造成塑料爆裂而產生裂痕使晶片脫落。
5. 腐蝕現象：腐蝕現象也是因為模造時含有水分，長期使用下，水分會造成離子的遷移而發生腐蝕的現象。
6. 龜裂現象：因為塑料的熱膨脹係數較晶片大，當填充完全後冷卻時，塑料收縮時因為受到晶片阻礙而產生張力，嚴重時就會有塑料龜裂的現象。
7. 金線偏移現象：模造時的金線偏移現象對於製程良率影響甚深，是本文所要討論的主題，下一節中將會有詳細的說明。

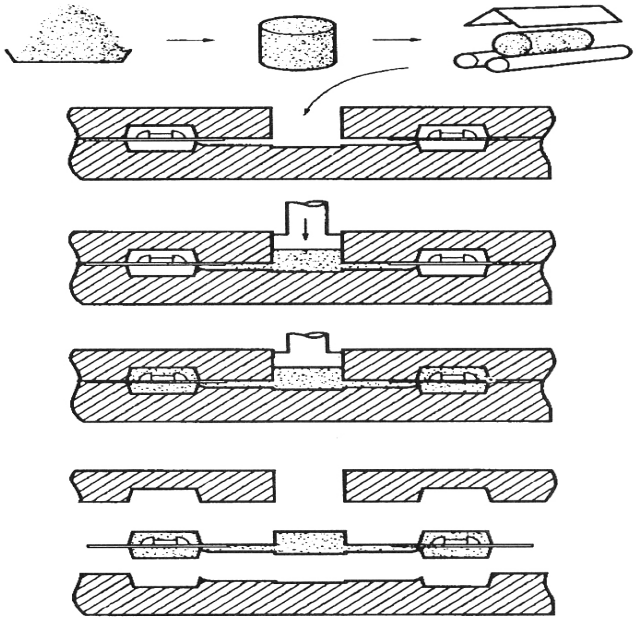


圖1-1、移轉模造成型程序示意圖

1.3 金線偏移

移轉模造封膠的過程中，因為塑料填充之模流會造成金線偏移的問題，金線偏移對於產品的影響，輕者造成金線間訊號相互干擾，影響產品的效能，重則導致短路，成為失效的不良品。隨著晶片功能增加，所需的I/O數隨之增加，再加上對IC封裝的尺寸要求越來越嚴苛，使得金線間的距離縮小，在這樣的情況下，金線的偏移就必須有更精準的控制。導致金線偏移的原因主要為：

1. 塑料引發的拖曳力（Viscous Drag Force）：金線偏移的主要原因就是因為塑料流過金線時產生拖曳力，拖曳力施加在金線上而使得金線變形，嚴重時更會在金線頸部斷裂，或者金線和接點脫離。當塑料流經金線時，若塑料流速過快或是塑料黏度過高，皆會使得金線偏移量過大。

2. 氣泡崩解或撞擊：在填充的過程中若有空氣進入模穴中，這些空氣形成氣泡後會在排氣孔處崩解，塑料必須填補先前被空氣所佔據的空間，造成流速瞬間上升使得拖曳力增加。或是金線也會因為氣泡的撞擊而產生偏移。
3. 導線架變形：若填充時塑料流速不均勻，在導線架上會產生不平衡力，使得導線架變形，因為金線是焊在晶片上的接點和導線架上，所以導線架變形也會使得金線偏移。
4. 填充物碰撞：封膠所用的塑料會添加一些添加物，如加入矽玻璃（Silica Glass）可以增加強度，若顆粒較大的添加物碰撞纖細的金線也會造成金線的偏移。
5. 保壓力過高：金線偏移後，會有一部分的彈性回彈減低偏移量，若是保壓力過高使得模穴壓力過大，使得彈性回彈困難。
6. 保壓力過小或保壓時間過長：若保壓時間過久，熟化反應將使得塑料黏度增大，亦使得金線偏移量加大。

1.4 文獻回顧

Nguyen[1]最早試著預測金線偏移，他利用網狀系統的流動模型（Network Flow Model）計算環氧樹脂從模缸到個模穴前膠口的溫度和熟化度（Degree of Curing），並利用動量與能量方程式，以有限差分法計算模穴中的溫度與熟化度分布。另外，Nguyen亦建立金線偏移模型，計算樹脂拖曳力造成金線變形的情況，其中，拖曳力係根據先前計算出的樹脂黏度、熟化度、流速以及金線幾何結構所算出。Nguyen針對雷諾數（Reynold Number）、金線結構，包括長度、高度、金線崩潰臨界點、剛性以及金線相對於流場波前的方向等因素，探討其對金線偏移的影響。Nguyen的金線模型限制在拋物線的數學模型，假設金線完美的焊接，並且只討論金線的彈性變形，以及使用簡單的流體模型，但這是首先定量分析金線偏移的研

究。

接著Nguyen與Lim[2]將金線輪廓改以高階指數描述，並將流體模型加入非牛頓流體性質以及包含化學反應，利用14條金線來進行金線偏移實驗，分別針對不同流速、模穴幾何形狀，金線幾何形狀，金線材料性質等進行引影響金線偏移的研究，之後將結果導入無因次分析，得到無因次偏移係數。

Nguyen[3]等提出高階指數方程式的改良金線模型，以及將流動模型加入非絕熱（Non-Isothermal）以及非牛頓（Non-Newtonian）流體，使用二維模流分析軟體C-MOLD來進行流場模擬，計算填充過程中金線索受到的拖曳力，並針對金線外型、金線相對流場波前角度和模造製程參數的影響作進一步的研究。

Tay[4]等將金線分成數小段，再以數學方程式描述個段輪廓近似真實金線輪廓，配合這些方程式給定其他相關形狀參數（焊接點跨距、金線高度、金線最高點位置等），方能定義金線輪廓。在文中將模穴中的模流以平面流（In-Plane Flow）簡化，並將黏度係數假設為常數，計算一些不同輪廓形狀的金線受流體平面負載（In-plane Load）的應力以及塑性應變率並做比較，提出較能抵抗拖曳力的金線輪廓。但是文中提到，由於真實的填充過程作用在金線的側向力所造成的偏移量遠大於先前提到的平面力所造成的偏移量，故單以平面流模擬設計出的金線輪廓並不能得到穩定的結構。

Wu[7]以Penalty Finite Element計算流場，再以Singularity Numerical Integration 的方法求出金線上的拖曳力，將求出的力代入有限元素軟體ABAQUS計算金線偏移。金線的輪廓沿用Tay的近似法[4]，並在金線的材料性質使用溫度相依的彈塑性；在計算流場方面，加入熟化度、溫度、黏度相關的非牛頓流體特性。Wu對模穴內金線的位置、局部流體速度場、局部溫度、熟化度、塑料黏度、雷諾速、澆口位置等影響金線偏移的因子作詳

細的探討，並對製程條件，如熟化時間、移轉速度（Transfer rate）和模溫的影響作一研究。

Su等[9]以分析軟體FIDAP作金線密度對金線偏移的影響。以簡單的二維平行板流動作為研究，討論圓柱在不同位置，圓柱間距離對於附近流場流速分佈之影響。

1.5 研究動機及目的

研究金線偏移的時間由文獻發表時間推算，已有十五年以上之久，對於傳統封裝製成的金線偏移問題已經可由經驗試誤或是模擬預測，透過製程上參數調整或是改變金線、導線架或模具的設計以及封膠材料的選用，使金線偏移量能後合乎標準。相對以經驗試誤來矯正金線偏移量耗時、耗成本，透過有效利用電腦輔助分析來模擬封膠製程及預測金線偏移量並加以修正將成為先進封裝技術不可或缺的工具。

近年來金線偏移模擬相關研究中，大致上都是將分析分成流場分析與金線受拖曳力變形分析兩步驟。在流場分析中並沒有將金線對流場造成的影響加入考量，即使有考慮到金線影響的研究，也是只有以等效體積[10]或是形狀因子[11]來修正流場分析結果，對於金線間的相互影響卻沒有較具體的研究。在Su[9]的研究中指出，當兩金線距離小於一限制時，處於上游之金線對於流場的干擾，將會對於下游金線附近的流場有所影響，因此推測下游的金線偏移量也會受到影響。

隨著3C產品不斷朝著輕薄短小的方向發展，IC的功能以及效率隨之提昇，相對的I/O數也隨之增加，欲在較小的體積內置入更多金線，這必定會使得金線與金線間的距離越形緊密，如此金線偏移控制的優劣更加影響成品良率，因此必須有更先進的封裝技術支援。

為了解金線間距對金線偏移量的影響以及金線密度造成流場的改變，進一步提高金線偏移模擬分析在高密度金線分佈情形下的準確性，本論文

將針對高密度金線封裝技術提出一套可行計算方法，用以預測封模流場以及金線偏移量，當作先進封裝製程中修正封模時金線偏移量的參考。

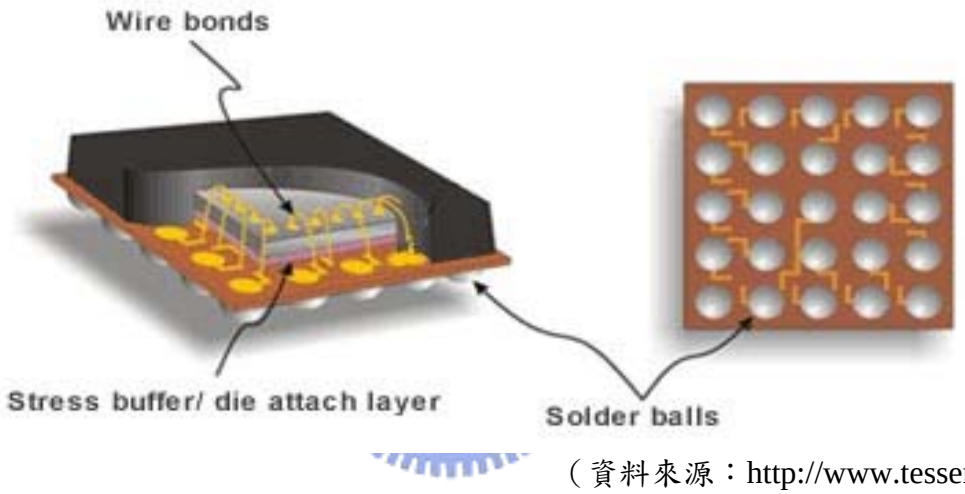
1.6 研究方法

本研究中將參考Han與Wang[6]提出的金線偏移模擬方法：將金線偏移分析分成整體流場、局部流場與金線偏移量計算三階段。為了將金線的影響加入流場分析中，在整體流場分析中，以3D模流分析軟體，並在建立流場網格時考慮金線之空間，進而計算該情況下金線偏移的狀況。另外也將仿造過去忽略金線對整體流場之影響的分析方式進行金線偏移分析，在不同的金線密度分佈下，比較前述兩方法偏移結果間的差異。

過去的文獻曾經提到計算拖曳力時需要以人工方式量測流場分析的數據，不僅費時而且容易人為誤差，並且，當金線數目多時，建立金線網格成為一項繁雜且重複性高的步驟。基於上述原因，本研究將建立前端輔助程式（程式原始碼參考附錄E），將大部份的工作由電腦來完成減輕建立模型網格以及計算拖曳立等耗人力工作。

第二章　網格模型建立

本文選用錫球凸塊陣列封裝（Ball Grid Array, BGA）為研究對象。BGA封裝（圖2-1）乃是只將晶片以打線接合、捲帶式接合或覆晶接合等方式和基板相連接，基板以面積陣列分佈排列的錫球凸塊（Solder Ball）代替引腳與電路板連結，BGA封裝最大的優勢是在於這樣的封裝型態易實現高密度的封裝，常運用在高I/O數、高功率的元件上，如DRAM（Dynamic Random-Access Memory）。



（資料來源：http://www.tessera.com）

圖2-1、BGA封裝示意圖

2.1　流場網格模型

2.1.1　流場網格建立

流場的區域包括模穴，因為塑料並不會流入晶片中，故實際流場區域是模穴與晶片及基板取差集部份，如圖2-2所示。建立流場網格的流程是以CAD軟體繪出流場區域幾何圖形並匯出IGS檔，再以網格建立軟體Hypermesh建立四面體元素流場網格並匯出UDM檔，最後由Moldflow承接此檔，經轉換後便完成用以計算整體流場分析的網格系統。

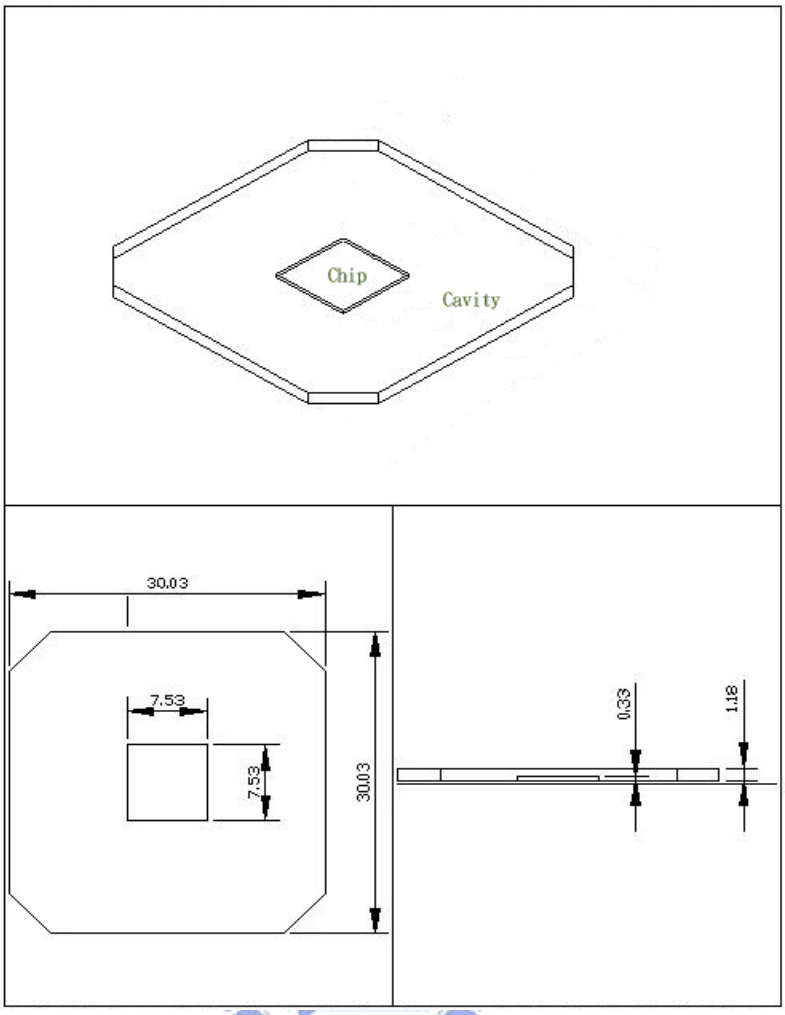


圖2-2、BGA封裝尺寸圖（單位：mm）

在本研究中，流場網格分為不包括金線與包括金線兩類：不包括金線的流場網格是在Hypermesh中先建立三角形面元素，再由這些面元素所圍繞出的體積建立四面體元素完成流場網格（圖2-3）；而包括金線的流場區域除了上述的區域外，尚須將金線體積部份扣除，在此，金線與流場尺寸相差甚巨，若以CAD軟體繪製金線轉出IGS檔並由Hypermesh加入，會有破面發生，且修正不易。因此，本研究以自行撰寫程式，根據金線接腳點與輪廓（金線輪廓給定詳見下節）產生能夠自動建立金線表面三角形元素的Hypermesh巨集，再自行完成其餘表面元素後建立出包括金線的流場網格（圖2-4）。

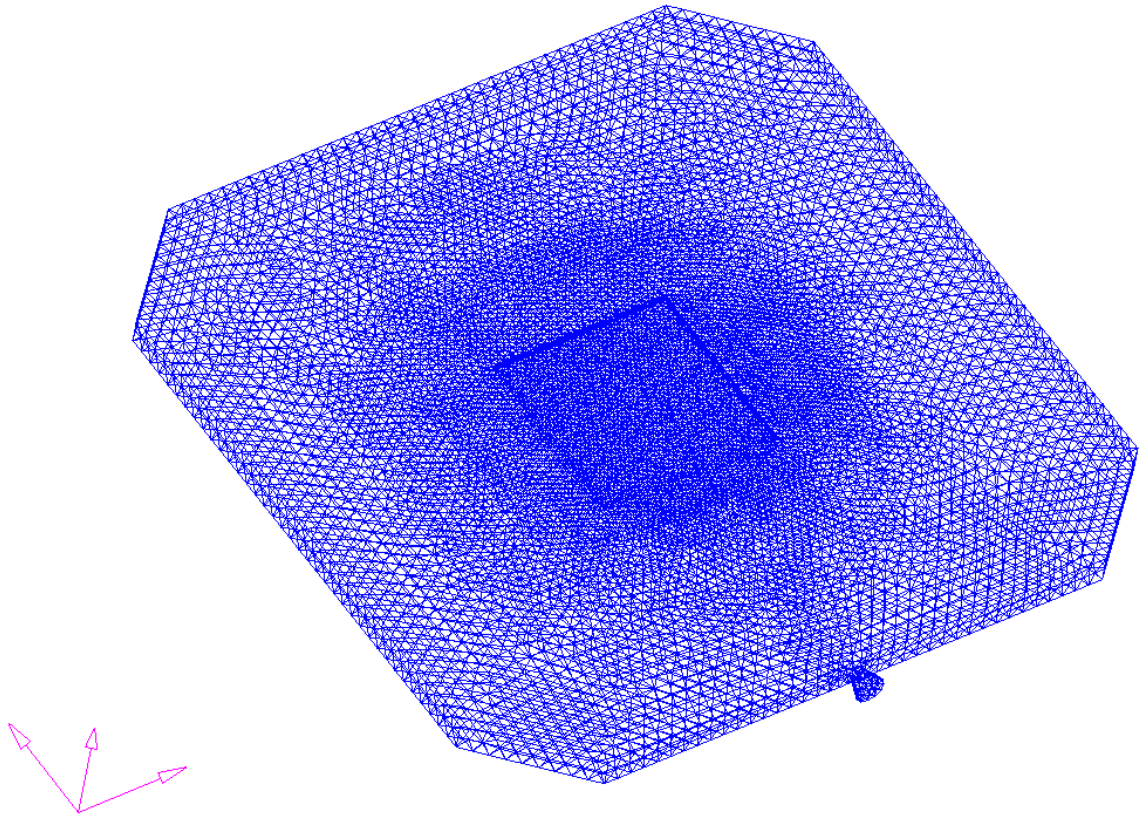


圖2-3、不包含金線之流場網格

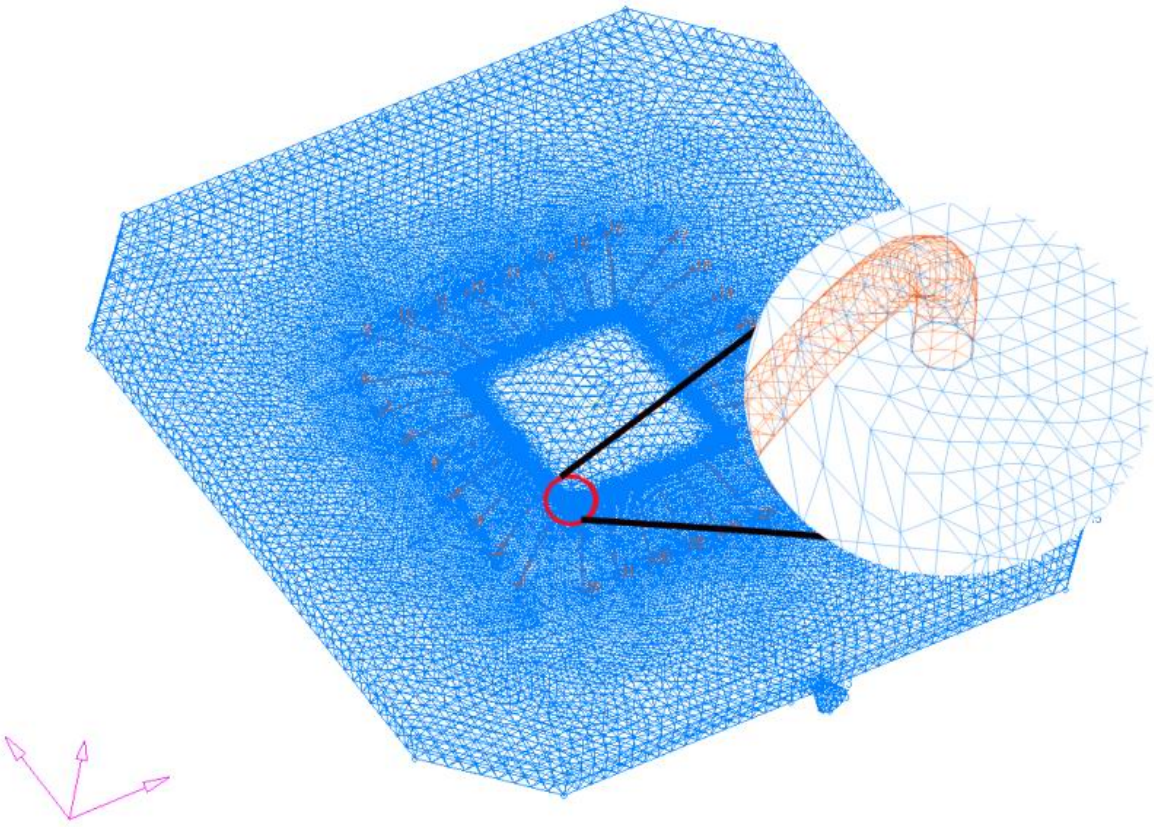


圖2-4、包含金線之流場網格

2.1.2 塑料材料性質

封裝用的塑料大多使用含環氧基（-C-C-）之環氧樹脂，環氧樹脂的優點包括易成型、易注模、加工時間短、韌性強、強度高、接著性良好、低收縮、氣體發生少，另外熱膨脹係數（約50～100×10^{-6}℃）與矽十分接近，當溫度變化時，塑料與矽晶片的熱應力較小。

環氧樹脂屬於非牛頓流體，其黏度不僅隨著剪切率（Shear Rate）變化，亦隨著溫度、熟化度改變。溫度越高黏度越低，但是高溫會使得分子間的鏈結反應加速而加快熟化反應，使得黏度上升。為了減少填充過熟化度，除了調降模溫以外，亦可加快填充速度，使熟化時間減少，但是卻使得塑料流速增加造成再金線上受到的拖曳力上升。

研究中所用的塑料為Moldflow Database裡提供的Hitachi Chemical公司，型號為CEL-9200的環氧樹脂（材料性質如表2-1），其黏度由Macosko黏度數學模型表示：

$$\eta(\alpha,T,\dot{\gamma})=\frac{\eta_0(T)}{1+\left(\frac{\eta_0(T)\dot{\gamma}}{\tau^*}\right)^{1-n}}\left(\frac{\alpha_{gel}}{\alpha_{gel}-\alpha}\right)^{(C_1+C_2\alpha)} \tag{2-2}$$

$$\eta_0=(T)=B\exp\left(\frac{T_b}{T}\right)$$

其中熟化度α以Kamal方程式表示：

$$\frac{d\alpha}{dt}=\left(K_1+K_2\alpha^{m_1}\right)\left(1-\alpha^{m_2}\right)$$
$$K_1=A_1\exp\left(\frac{-E_1}{T}\right) \tag{2-3}$$
$$K_2=A_2\exp\left(\frac{-E_2}{T}\right)$$

表2-1、 CEL-9200材料性質

項目	材料特性	數值
材料性質	熔融密度	1.995 g/cm^3
	固化密度	2 g/cm^3
	比熱	1100 $J/kg-^\circ C$
	熱傳導係數	0.9 $W/m-^\circ C$
	反應熱	20220 J/kg
反應黏度參數	n	0.7348
	B	8.32865e-5
	C_1	5.326
	C_2	0.1937
	凝膠熟化度（α_{gel}）	0.387
	溫度靈敏因子（T_b）	12038.7 k
	臨界剪應力（τ^*）	7.3e-5 Pa
反應熟化度參數	m_1	0.6015
	m_2	1.252
	A_1	1.7577e+9 $1/s$
	A_2	2.7836e+5 $1/s$
	E_1	12415.7 K
	E_2	7314.07 K

2.2 金線輪廓與有限元素網格

2.2.1 金線輪廓

打線接合完成的典型金線輪廓分為三種（圖2-5），本文以標準輪廓為研究對象。依照Tay [4]的概念，將金線於最高點切分成二段（圖2-6），分別以三次方程式與橢圓方程式描述其輪廓（式2-3），於第二接腳與最高處金線處於水平，而在第一接腳金線為垂直，因此設定金線數學邊界條件（式2-4），經求解推倒便得金線輪廓近似方程式（式2-5）。

$$\begin{cases} y = a_1x^3 + a_2x^2 + a_3x + a_4, & 0 \le x < l \\ \dfrac{(x-b_1)^2}{b_2} + \dfrac{(y-b_3)^2}{b_4} = 1, & l < x \le L \end{cases} \tag{2-4}$$

$$\begin{cases} y(0)=0, & y'(0)=0 \\ y(l)=H, & y'(l)=0 \\ y(L)=h, & y'(L)=\infty \end{cases} \tag{2-5}$$

$$\begin{cases} y(x)=-\dfrac{2H}{l^3}x^3+\dfrac{3H}{l^2}x^2, & 0\le x<l \\ y(x)=h+(H-h)\sqrt{1-\left(\dfrac{x-l}{L-l}\right)^2}, & l<x\le L \end{cases} \tag{2-6}$$

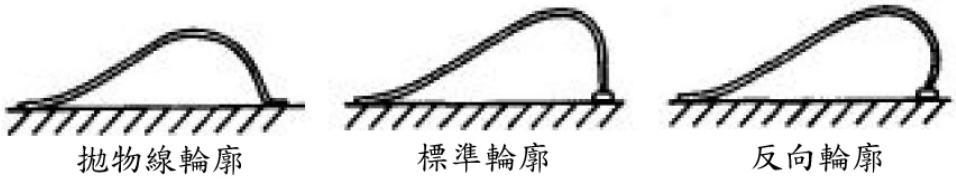


圖2-5、典型金線輪廓

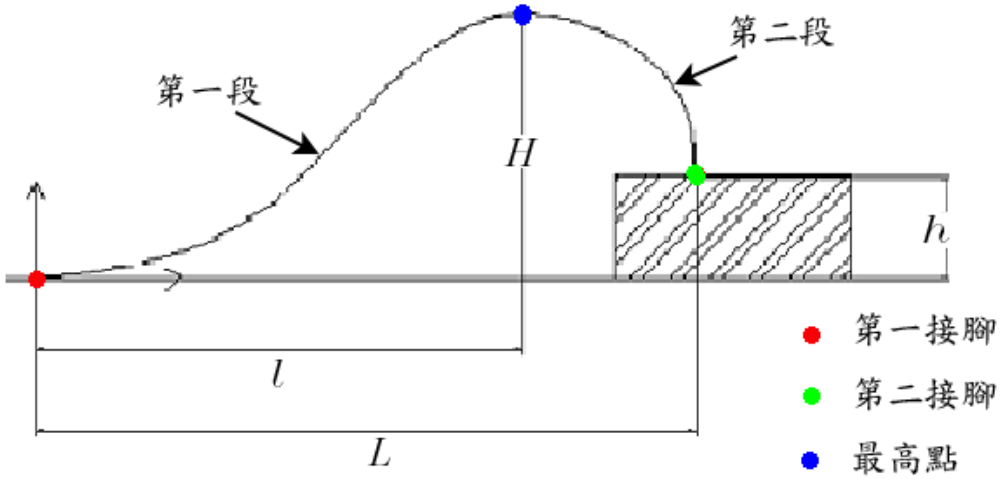


圖2-6、金線輪廓示意圖

2.2.2 金線有限元素網格

根據描述金線輪廓的方程式（式2-6），建立二節點Beam元素ABAQUS金線網格，以及計算元素上的拖曳力。在本文中，以自行撰寫的程式進行網金線網格建立，為了有效控制金線元素長度，在程式中以類似牛頓近似的方法控制金線元素長度，如圖2-7所示，線段L為欲建立之金線元素，長度為S，第一步，先計算在金線元素第一個節點a上的金線切線與X軸之夾角θ，另外在切線上作一長度為S之線段L1，以夾角θ計算L1在X軸上之投影

點X1並求出中點mid；第二步，計算金現在mid上的切線與X軸之夾角φ，並以a為起點，與X軸夾角為φ且長度為S之線段L2。計算L2在X軸上之投影點X2，並求出金線元素上的第二個節點b，則線段a b即為長度接近S之金線元素。

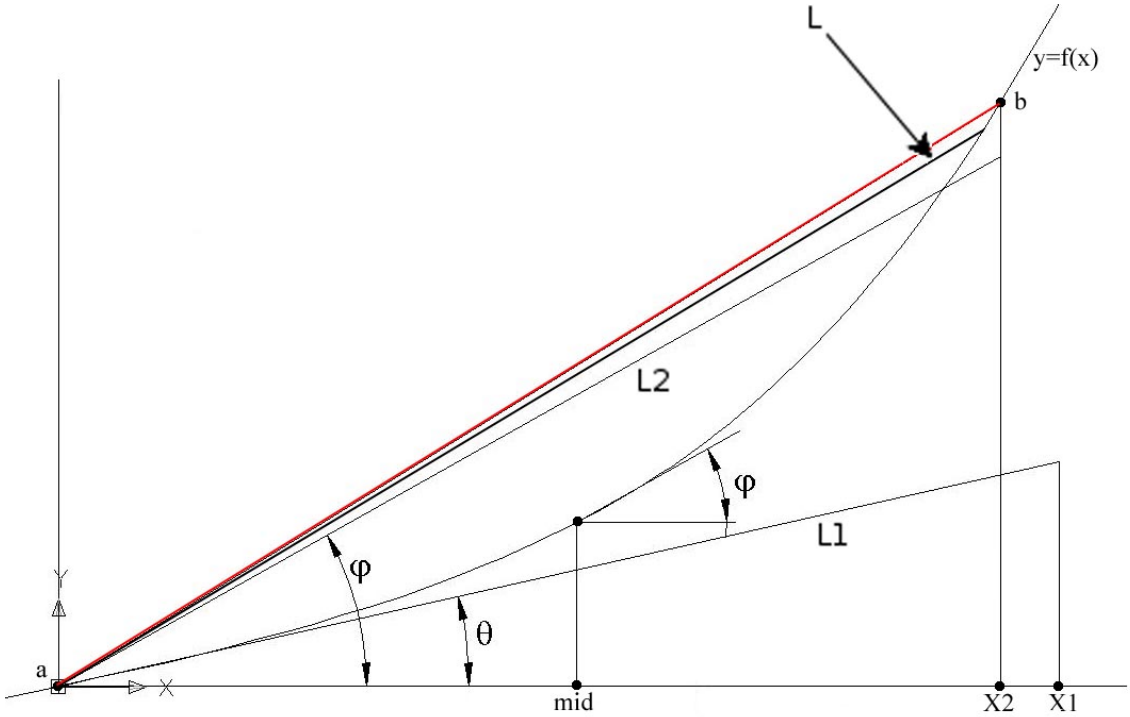


圖2-7、金線元素長度計算示意圖

金線材料性質是從MOLDFLOW Database中所去得，其為Tanaka公司編號為M3的金線，其彈性係數為60.51GPa，浦松比為0.33，其應力應變圖如下：

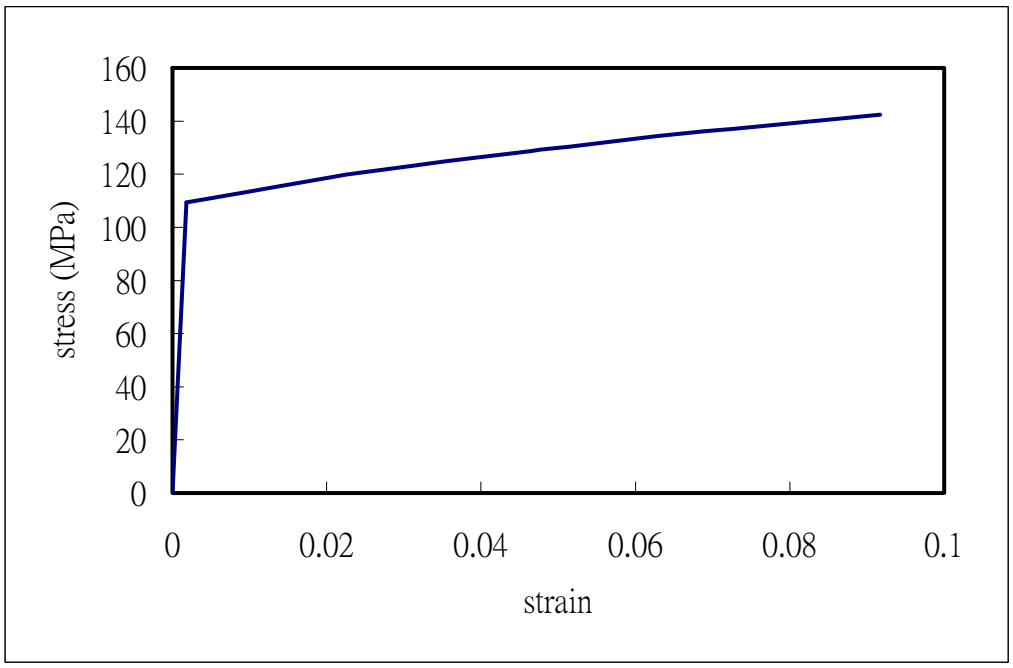


圖2-8、金線應力應變圖

2.3 邊界條件與假設

表2-2為模擬填膠製程參數，其中給定填充時間，Moldflow程式會依據行程與活塞速度關係（圖2-9）以及模穴容積計算膠口處的體積流率。在模擬時經過以下假設：

1. 塑料流動加入了慣性力的影響。
2. 因為模穴厚度小，不考慮重力對塑料的作用。
3. 晶片與基板間接合處在填充過程中不發生滑動或脫離。
4. 在填充過程中，金線偏移的現象不影響塑料流動，因此在加入金線的整體流場分析中，金線網格僅用以模擬金線所佔之體積，並不發生偏移。
5. 金線接腳處的接合完美，填充過程中不發生脫離。

表2-2、 移轉模造成型填膠製程參數

模穴壁溫度	180 ℃
熔膠溫度	90 ℃
初始熟化度	0
填充時間	5 seconds

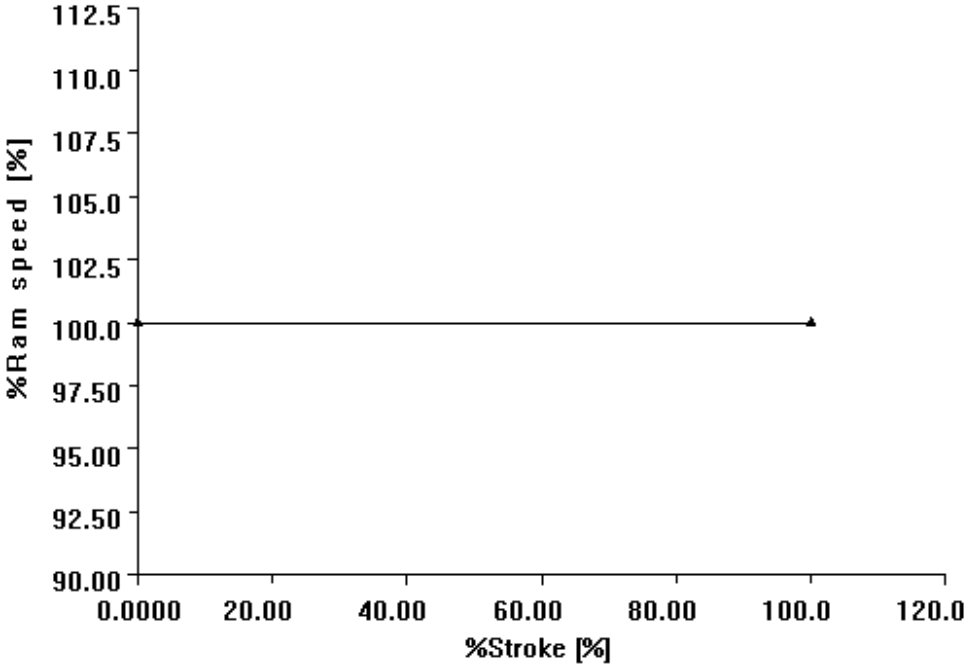


圖2-9、行程與活塞速度關係圖

第三章　分析步驟

金線偏移主要是因為移轉模造過程中受到塑料流動所引發的拖曳力而發生金線結構的變形，因此，分析過程中牽涉到模流、固體力學與流固之間的反應問題，在本文中將分析分成整體流場、局部流場與金線偏移三階段，並將依照流場是否加入金線網格分成兩種，以觀察金線密度對偏移量的影響。

3.1　整體流場分析

整體流場分析用以模擬轉移模造成形時，塑料填充模穴時的塑料流動情形，計算出模穴裡個節點上的流速、黏度、溫度、壓力等數據，在本研究中，此分析是以三維模流分析軟體Moldflow作計算。

Moldflow是Moldflow Corporation專門用來分析塑膠模流的軟體，其幾何構型寬度，可跨越2D，2.5D，3D不同模型方式的整合方式，涵蓋線架構、曲面、薄殼、厚件實體。Moldflow也針對IC轉移模造成型提供可用的分析模組（Microchip Encapsulation），也包含許多塑料材料資料庫。

3.1.1　模流理論

本研究以Navier-Stokes方程式求解在流場，包括：

1. 質量守衡方程式（3-1式）：塑料在控制體積內的密度隨時間增量等於塑料流入體積內的質量流率，亦可稱為連續方程式。

$$\frac{d\rho}{dt} = \rho \nabla \cdot \vec{u} \tag{3-1}$$

2. 動量守恆方程式（3-2式）：根據牛頓第二運動定律，塑料動量變化率等於外力作用於流體的動量。

$$\rho \frac{d\vec{u}}{dt} + \nabla p = \nabla \cdot (\mu \quad def \ \vec{u}) + \vec{F} \tag{3-2}$$

$def\ \vec{u}$ 為塑料的形變率（Rate of Deformation）表示為：

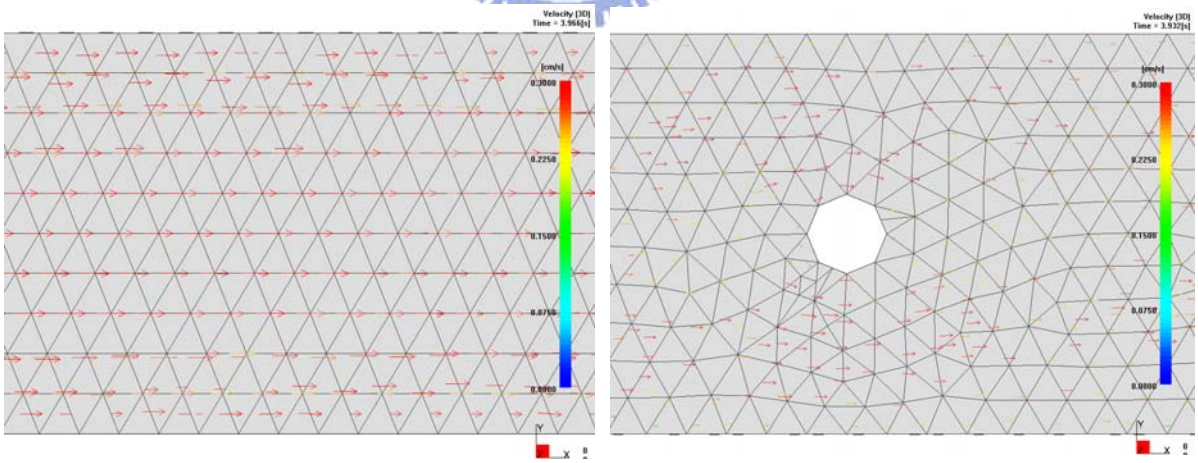
$$def\ \vec{u} = \begin{bmatrix} 2u,_x & v,_x + u,_y & w,_x + u,_z \\ u,_y + v,_x & 2v,_y & w,_y + v,_z \\ u,_z + w,_z & v,_z + w,_y & 2w,_z \end{bmatrix} \tag{3-3}$$

3. 能量守衡方程式（3-4式）：根據熱力學第一定律，控制體積內增加的儲存能量等於邊界對控制體積內所做的功加上從邊界傳入體積內的熱量。

$$\rho c_p \frac{dT}{dt} = \beta T \frac{dp}{dt} + \frac{\mu}{2} def\ \vec{u} \cdot def\ \vec{u} + \nabla \cdot (k \nabla T) \tag{3-4}$$

3.2 局部流場分析

局部流場分析依據整體流場分析中所得到的流場流速、黏度、壓力等結果，計算在金線上所引發的拖曳力。金線所在位置附近的局部流場分佈情形會因為流場網格中是否加入金線而有所不同（圖3-1），因此，在此階段分析，計算拖曳力的方法分成二類。



（左：未含金線；右：包含金線）

圖3-1、金線影響流場示意圖

3.2.2 流場網格不考慮金線存在

當整體流場中忽略金線影響，也就是流場網格中不包括金線。此時金線周圍局部流場流速均勻，塑料在金線上所引發的拖曳力以Sherman（3-5式）拖曳力方程式計算之。

$$D_t = \frac{4\pi\mu V_t}{2\ln(\varepsilon) - 1}$$
$$D_n = \frac{8\pi\mu V_n}{2\ln(\varepsilon) + 1} \quad (3\text{-}5)$$

D_t、D_n分別為每單位長度金線切線方向和正交方向的拖曳力，以及V_t、V_n分別為金線切線方向和正交方向的塑料流速如圖3-2所示；μ為流場黏度；ε細長比則為金線長度與半徑比，在Sherman的推導中，將細長比設定在$\varepsilon >> 2$，因此以此方程式計算拖曳力時，必須注意到金線取樣長度。

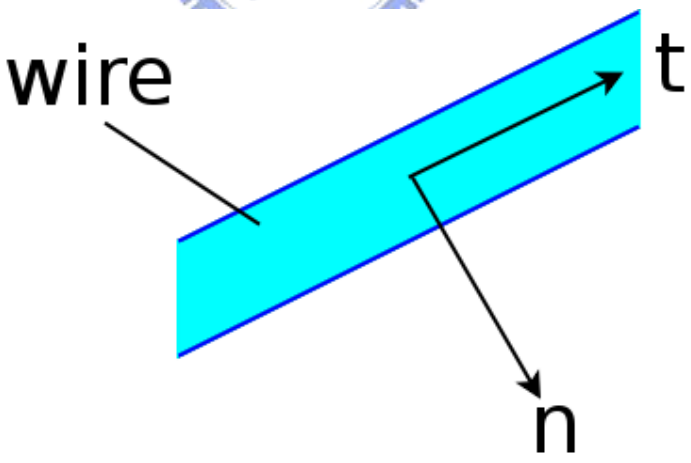


圖3-2、Sherman拖曳力座標系

計算拖曳力的方式是自行撰寫程式來進行，（程式流程如圖請參考附錄B）當整體流場分析完成後，根據Moldflow匯出之網格、節點座標數據，

由程式搜尋各個金線Beam元素中點位置所在之流場網格。為了增加搜尋速度，搜尋方式是先找出距離金線元素中點最近之流場節點，依此節點可由網格資訊裡找出數個備尋流場網格，這些網格的組成節點中必須包含該節點。一一檢查金線元素中點是否位於上述之備尋網格中。在某流場網格中，對應於金線元素中點，可定義四個形狀函數（Shape Function）：

$$N_i = \frac{vol_i}{vol}, \qquad i = 1 \sim 4 \tag{3-6}$$

式中 vol 為該流場網格體積，vol_i 為網格內編號 i 以外三個組成網格節點與金線元素中點構成之四面體體積，如圖3-3所示。若金線元素中點位於流場網格中，則 $N_1 + N_2 + N_3 + N_4 = 1$。完成流場網格搜尋後，由Moldflow匯出的塑料流場分佈資料中讀取各流場節點上的流速以及黏度，並內插（2-7式）至金線元素中點上，最後在以Sherman拖曳力方程式計算在該金線元素之拖曳力。

$$\Phi = N_1\Phi_1 + N_2\Phi_2 + N_3\Phi_3 + N_4\Phi_4 \tag{3-7}$$

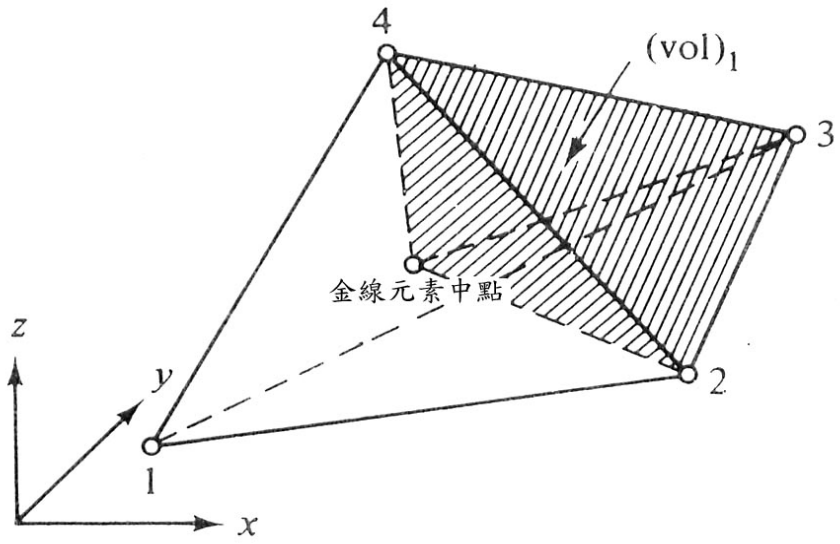


圖3-3、流場網格形狀函數計算示意圖

3.2.3 流場網格考慮金線存在

當流場網格中加入金線，因為塑料與金線間的相互關係，在金線附近有流速減慢的的情形，因此，局部位置的流場並非均勻分佈，故在此類金線上的拖曳力計算方式無法如同上述簡化公式計算，因此在這情形下計算拖曳力是以數值方法計算之。

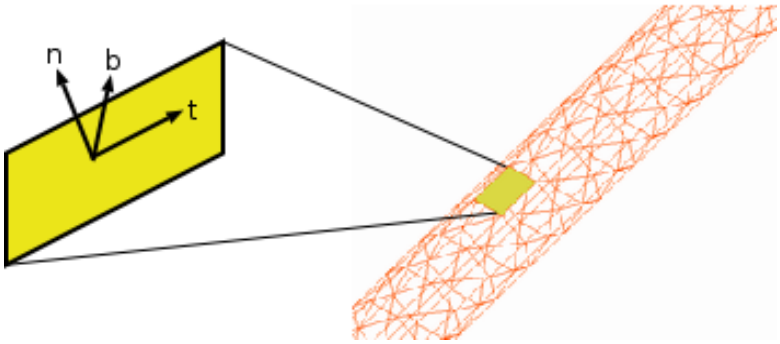


圖3-4、金線網格上拖曳力座標系

在流場中與金線介面的網格上（圖3-4），所受到的作用力 $\vec{\sigma}$ 為：

$$\vec{\sigma} = -\vec{n}\,p + \vec{\tau} \tag{3-8}$$

其中 p 為壓力，$\vec{\tau}$ 為偏差應力（Deviatoric Stress），將 $\vec{\tau}$ 分成在平面的法線方向 $\vec{n}$、金線切線方向 $\vec{t}$ 與副法線（Bi-Normal）方向 $\vec{b}$ 上的應力分量 $\vec{\tau}_n$、$\vec{\tau}_t$ 與 $\vec{\tau}_b$，其大小為：

$$\begin{cases} \tau_n = 2\mu\dfrac{\partial w}{\partial n} \\ \tau_t = \mu\left(\dfrac{\partial u}{\partial n} + \dfrac{\partial w}{\partial t}\right) \\ \tau_b = \mu\left(\dfrac{\partial v}{\partial n} + \dfrac{\partial w}{\partial b}\right) \end{cases} \tag{3-9}$$

式中w、u、v分別為 $\vec{n}$、$\vec{t}$、$\vec{b}$ 三方向的塑料流速分量，因此由3-8式及3-9式該金線網格所受到塑料引發的作用力 $\vec{\sigma}$ 為：

$$\vec{\sigma} = \left(-p + 2\mu\frac{\partial w}{\partial n}\right)\vec{n} + \mu\left(\frac{\partial u}{\partial n} + \frac{\partial w}{\partial t}\right)\vec{t} + \mu\left(\frac{\partial v}{\partial n} + \frac{\partial w}{\partial b}\right)\vec{b} \tag{3-10}$$

用以計算拖曳力的程式中（程式流程如圖參考附錄C），當整體流場分析完成後，由Moldflow匯出之網格、節點座標、速度場、壓力及黏度分佈等數據，程式先讀取網格與節點座標資訊，並搜尋流場網格中位於塑料與金線介面上之節點與網格，再讀取在該節點上之壓力、流速與黏度；在各網格中找出網格中點並計算法線、切線與副法線向量，接著找出流場網中最接近中點的三個節點；為了計算3-10式中流速在個方向上之偏微分項，假設在流場與金線介面上之一網格附近的流速場V為座標位置之函數：

$$V = V(x,y,z) = a_1 x + a_2 y + a_3 z + a_4 \tag{3-11}$$

在此網格上中點與最接近中點之三流場節點流速V_1、V_2、V_3、V_4為：

$$\begin{Bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{Bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix} \tag{3-12}$$

則：

$$\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}^{-1} \begin{Bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{Bmatrix} \tag{3-13}$$

因此由3-11式及3-13式得此網格附近的流速為：

$$V = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}^{-1} \begin{Bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{Bmatrix} \tag{3-14}$$

速度場計算完成後，在以數值偏微分法計算於中點上之偏差應力，最後在以3-10式計算該網格上之拖曳力。

3.3 金線偏移分析

當局部流場分析根據整體流場分析結果計算出金線上的拖曳力，便以有限元素軟體計算金線上因作用力所造成的形變量。在此計算金線偏移量所用的有限元素軟體為ABAQUS/Standard。

ABAQUS是由ABAQUS公司所發展的有限元素軟體，在線性或非線性的工程運用上具有相當可靠的能力。其中的Standard是其一隱性（Implicit）迭代求解器，用以計算靜態或擬動態等工程結構問題。

在本文中以Beam作為金線網格所用的元素，在ABAQUS中的Beam元素依元素階次可分為二節點線性（B31）、三節點二階（B32）以及兩節點三階等三類（B33）元素、並且提供多樣的截面種類（表3-1、3-2），本文以二節點線性截面為實心圓的Beam元素。

在ABAQUS中，Beam元素的定義為：元素中任意點與中心軸的最短距離小於元素軸向長度，其主要的求解變數只為中心軸上位置之函數。最簡單的Beam元素理論是Euler-Bernoulli假設：最初和Beam元素中心軸垂直的任意截斷平面永遠保持平面且垂直中心軸也不發生扭曲（Undistorted）。

二節點線性元素是延伸Euler-Bernoulli假設的Timoshenko Beam元素理論：元素允許橫向剪切變形，原本與中心軸垂直之截斷平面不須保持與中心軸垂直。這對於無論厚短或細長的元素都是相當有用的理論，能夠承受大變形或是彎曲，當元素截面尺寸高達八分之一軸向長度，其求出的解也具有相當程度的準確性。

表3-1、 ABAQUS軟體中Beam元素截面種類及其積分點

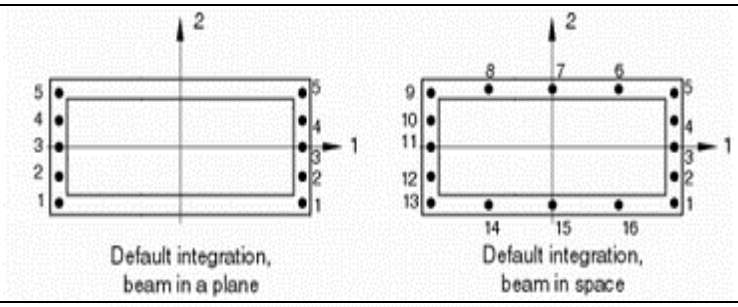
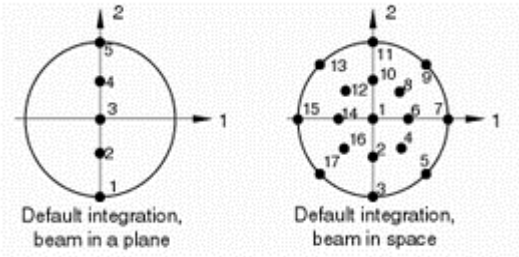
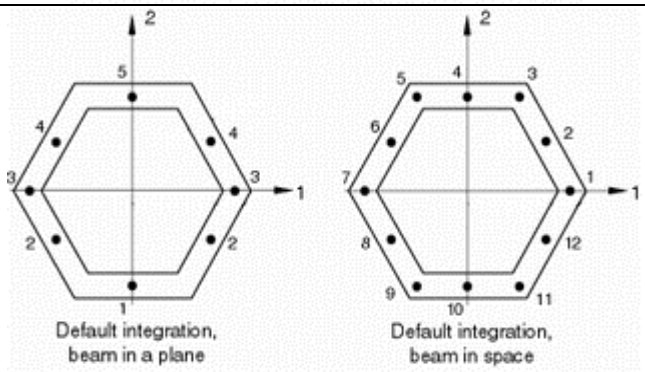
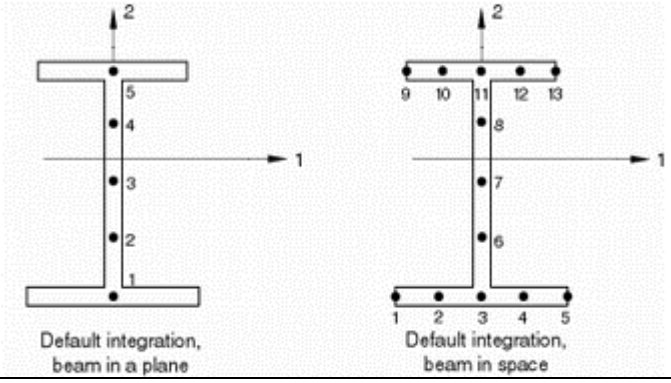
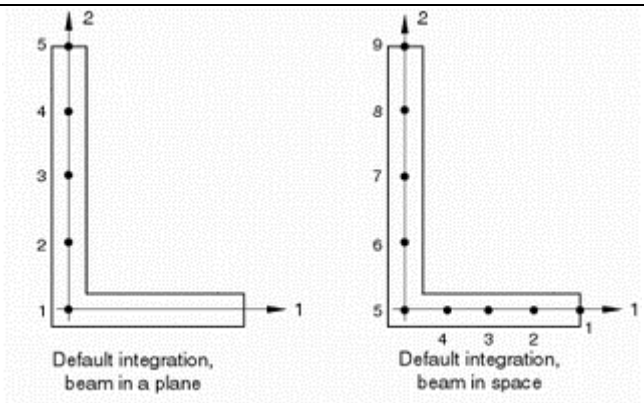
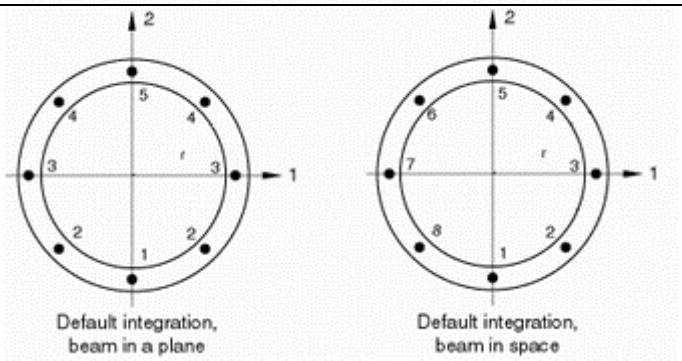
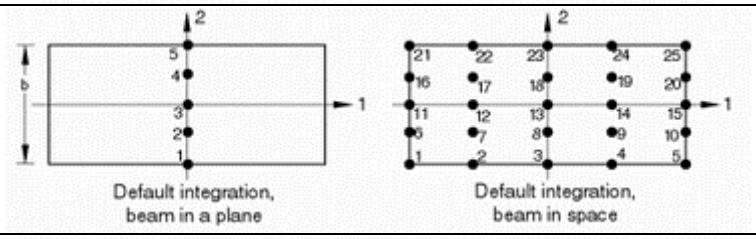
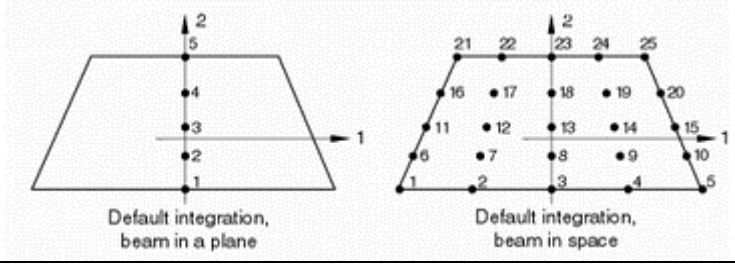
	Box section
	Circular section
	Hexagonal section
	I-section

表3-2、 ABAQUS軟體中Beam元素截面種類及其積分點（續）

	L-section
	Pipe section
	Rectangular section
	Trapezoidal section

第四章　金線密度影響分析

金線密度影響之分析以本研究提出將金線網格建立在流場網格中的進階金線偏移分析方式，另一方面也沿用過去沒有將金線建立在流場網格中的傳統金線偏移分析方式，比較兩種金線偏移分析方式所得到的結果。為了金線密度對於其偏移量的影響，在不改變其他變數的前提下，藉由改變金線位置調整金線間的密度，以一系列不同金線密度分別以兩種方式進行金線偏移量計算，並討論金線以及金線密度對於流場以及金線偏移量之影響。

4.1　金線分佈規劃（一）

前文曾提及金線與模穴尺寸相差甚巨，這不僅使得網格建立困難，亦使得建立完成的網格在金線附近必須相當緊密（圖4-1），以致網格數動輒百萬，這樣的情況使得計算機在資源有限及運算速度有限的情況下，計算整體流場需要耗費相當多的時間。雖然建立網格的工作流程可以自行撰寫程式簡化，但仍無法改善計算機的能力，因此，加入流場網格的金線僅以位於晶片左側之簡化方式來分析。

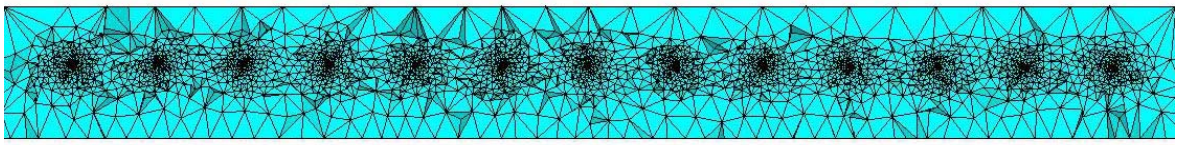


圖4-1、金線周圍之流場網格分佈情形

於晶片左側平均分佈13條金線，金線高度為0.8，並且與第二接腳距離皆設定為第一接腳和第二接腳間距離之0.75倍，所有金線逐次靠近向七號金線改變金線間距，依接腳間距離不同共分成六組（表4-1），其金線間第一接腳與第二接腳間距及座標見附錄D，各組金線分佈情形如圖4-2所示。

表4-1、 金線接腳間距

組別	一	二	三	四	五	六
第一接腳間距（mm）	1.00	0.85	0.70	0.55	0.40	0.25
第二接腳間距（mm）	0.60	0.51	0.42	0.33	0.24	0.15

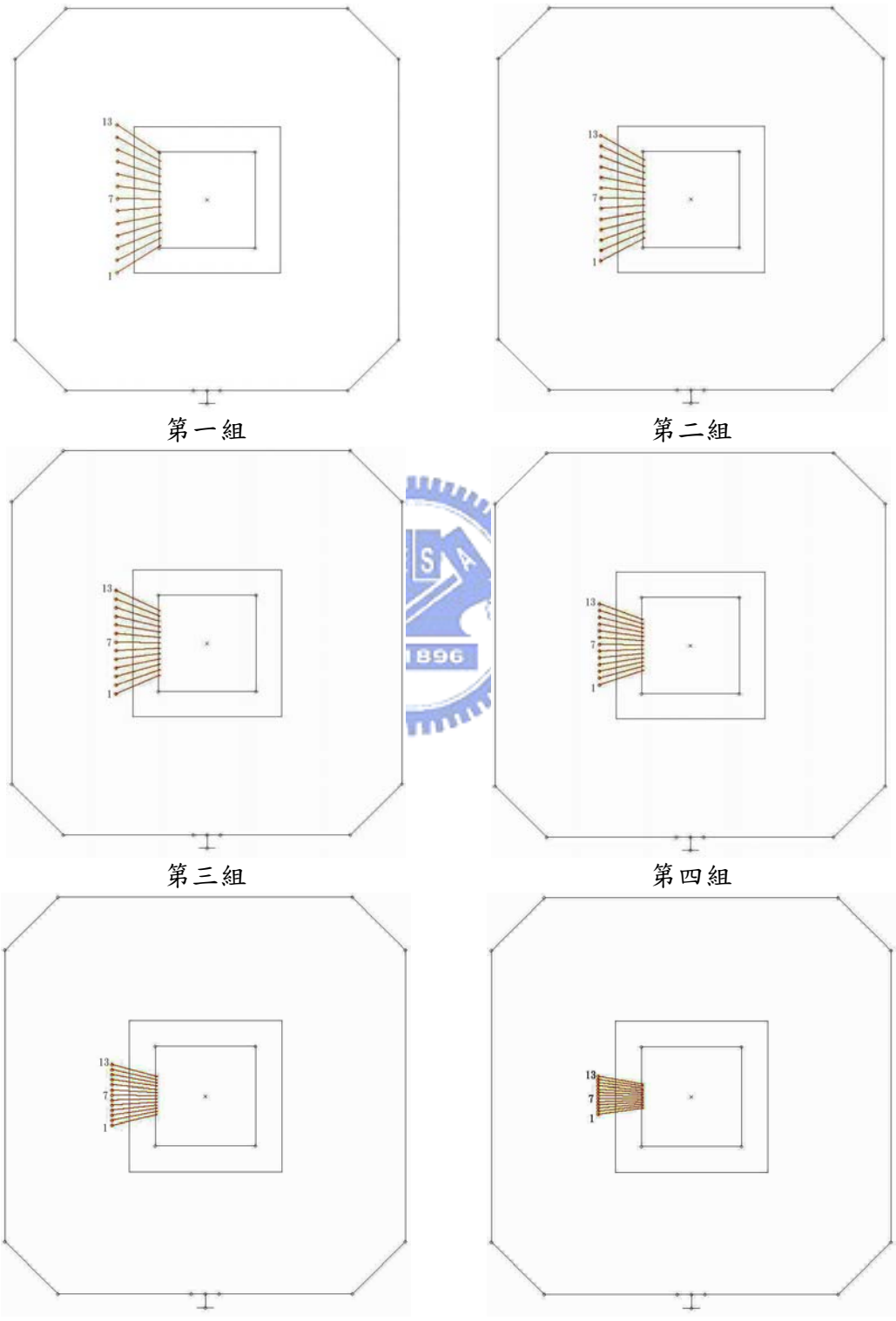


圖4-2、金線分佈圖

4.2 模擬結果（一）

4.2.1 金線對塑料波前之影響

由塑料流動波前對時間的關係圖（圖4-3）觀察，不含金線和其他含金線的流動情形作比較可明顯看出，當塑料流經金線時都會有波前落後的情形，且含有金線的流場，在晶片左側的波前會稍微落後於晶片右側。而在不同金線密度下的整體波前行為並無太大差異，僅在金線區因為金線位置不同而有所變化，通過金線後的波前大致相同。因此可推論，金線造成流動阻礙，使得波前落後；但不同的金線密度不會明顯影響波前行為。

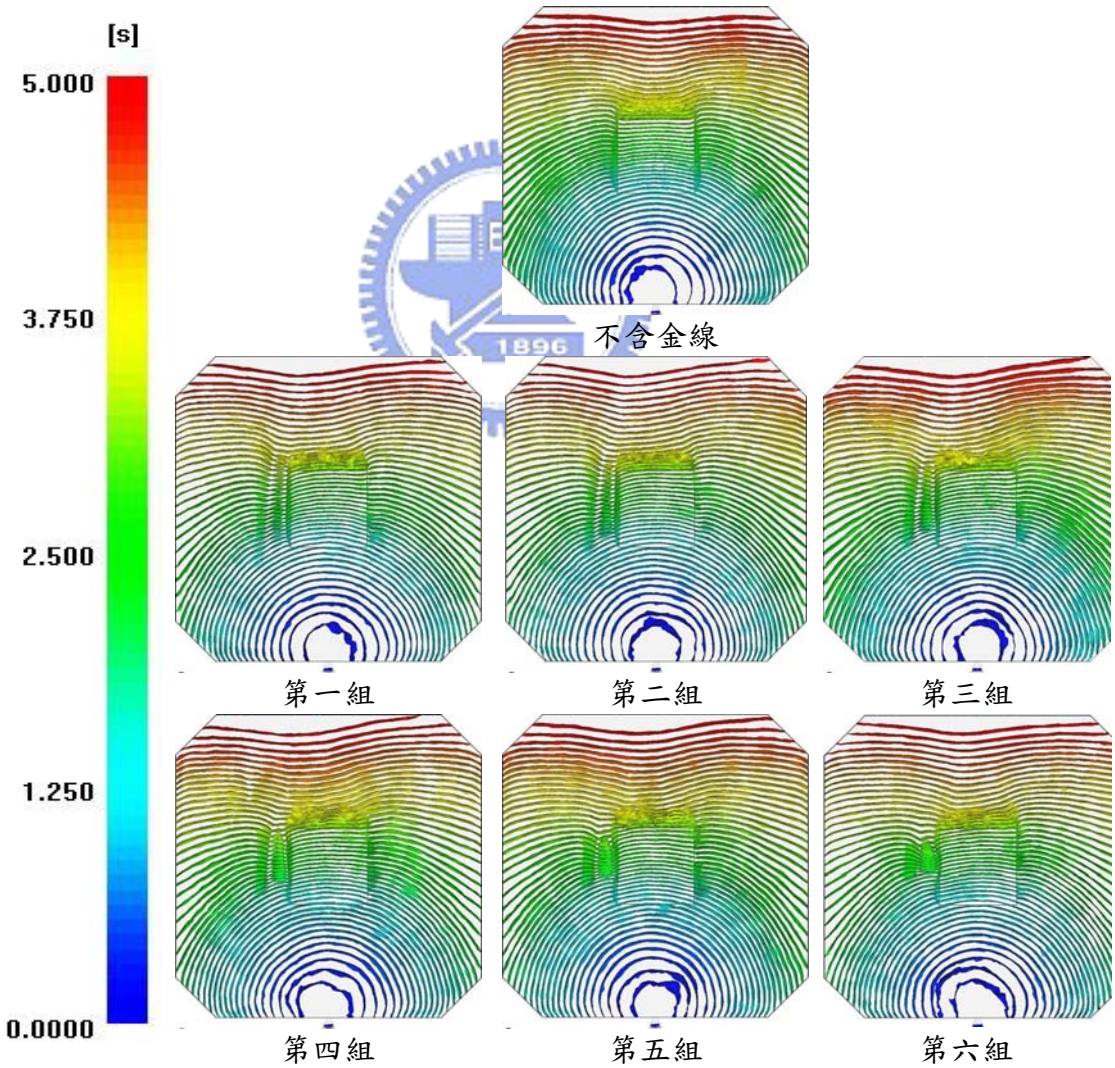


圖4-3、塑料流動波前對時間的關係圖

4.2.2　金線對模穴中壓力之影響

圖4-4為當塑料完全填充模穴時之壓力分佈情形，當流場分析加入金線的影響時，流場受到阻礙而壓力降提前在金線區域發生，這不僅發生在晶片左側，在晶片右側亦受到影響，並且金線密度增加時，晶片兩側壓力分佈更加相似。在固定金線數量下提高金線密度，使得金線分部區域較為集中，因此影響壓力變化的金線範圍縮小。故當金線密度高時，加入金線的晶片左側與右側之壓力分佈較為一致。

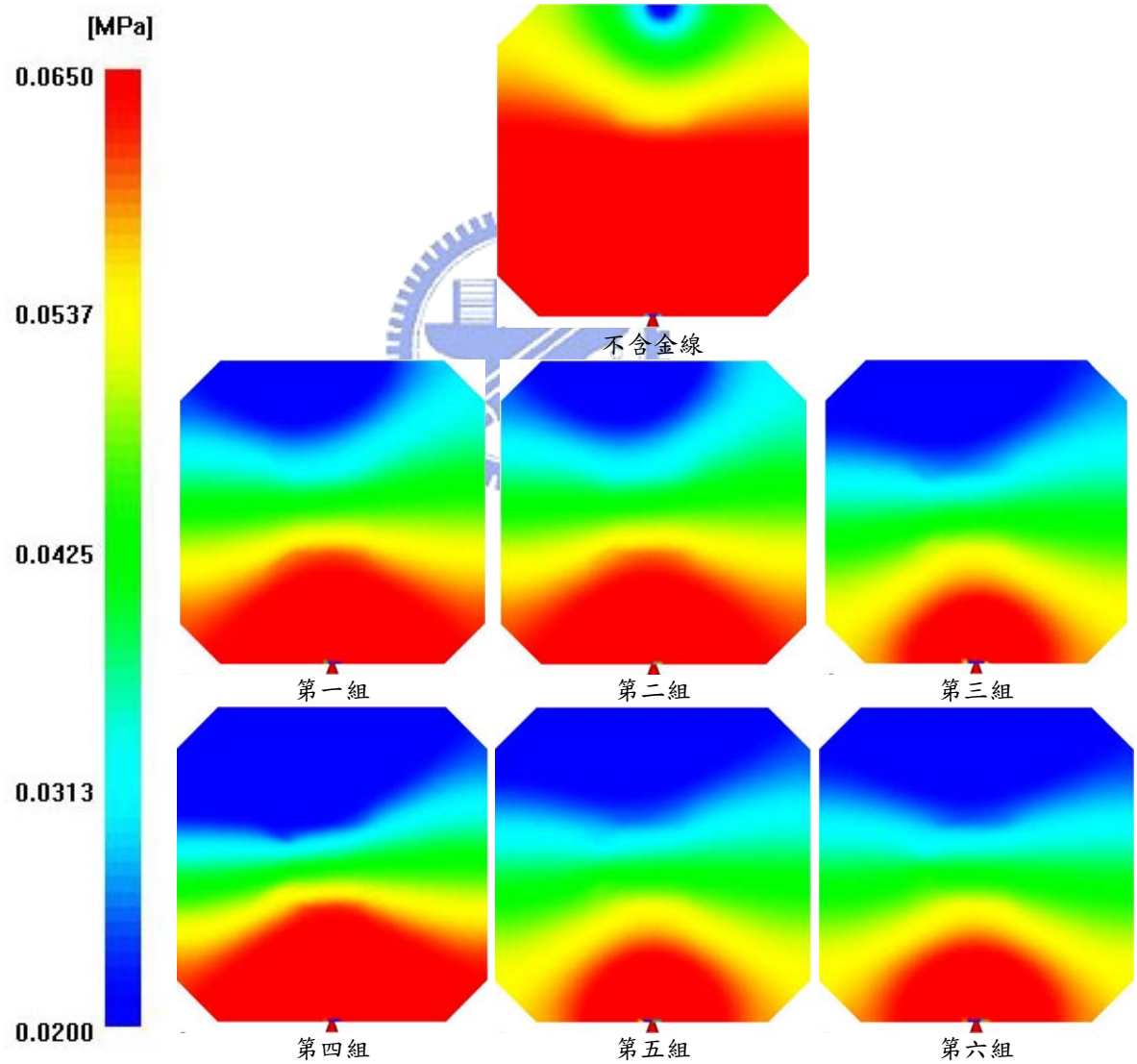


圖4-4、壓力分佈圖

4.2.3 金線對流速之影響

圖4-5為塑料完全填充模穴時在二分之一模穴厚度切面之速度分佈情形，整體的流速分佈情形大致不受金線影響，唯其在模穴上方處（波前到達末端處）有較明顯的差異，而在金線位置的局部區域受影響較大，當塑料流經金線時，金線周圍的流速大幅下降。由於流向晶片左側之塑料經過金線時，塑料受到金線阻礙而部份向更左側區域流動，使得在金線區域左側延伸處之流速有稍微的增加。

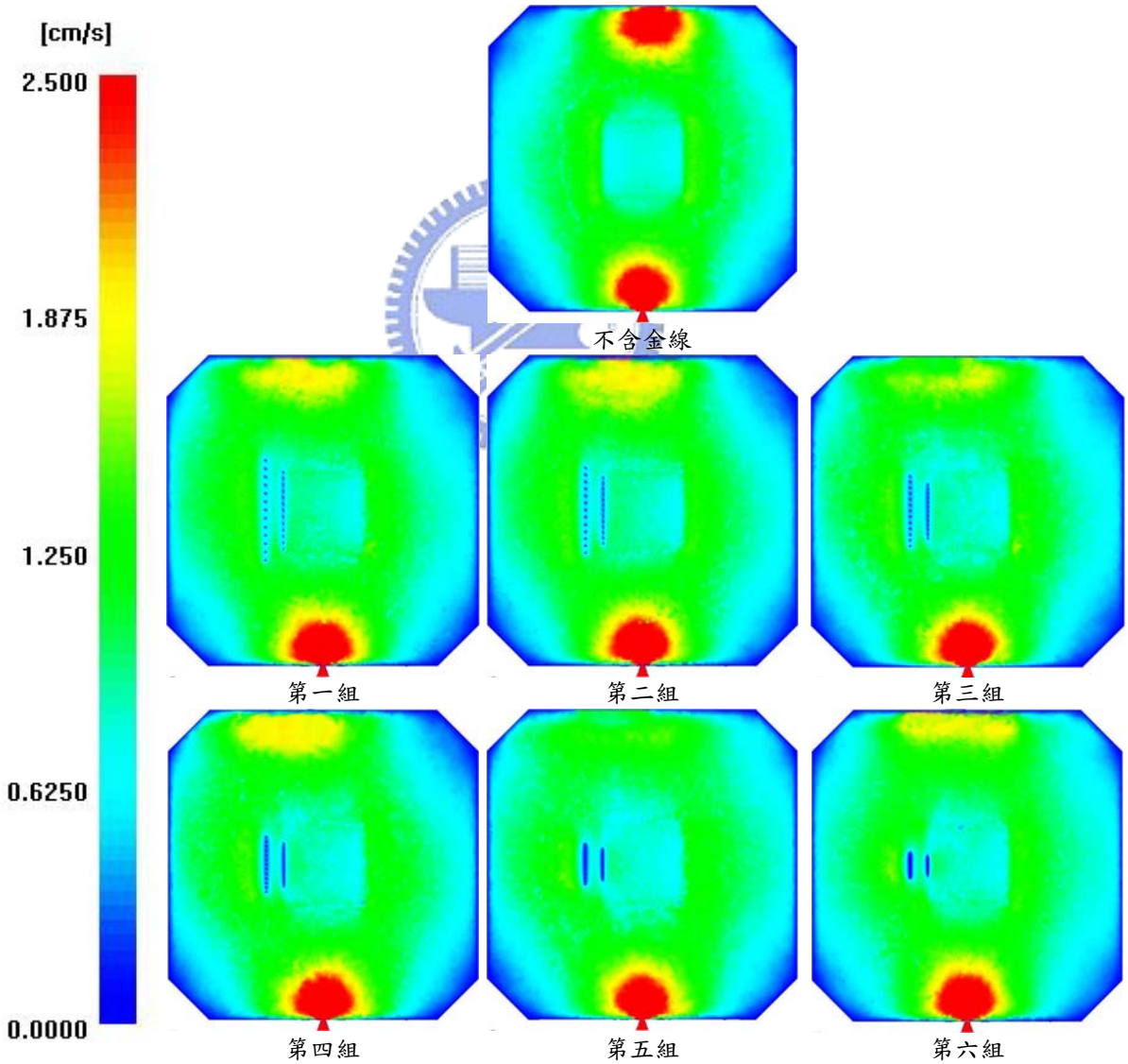


圖4-5、速度分佈圖

4.2.4　金線偏移結果

圖4-6為各組不同金線密度下，取金線節點中最大水平偏移量（xy平面上之位移量）之位移曲線圖，由圖中可看出以本文所提出的方法所計算出的金線偏移量大致小於傳統的分析方法所計算出的結果，並且當金線間距越小，偏移量差異越大。

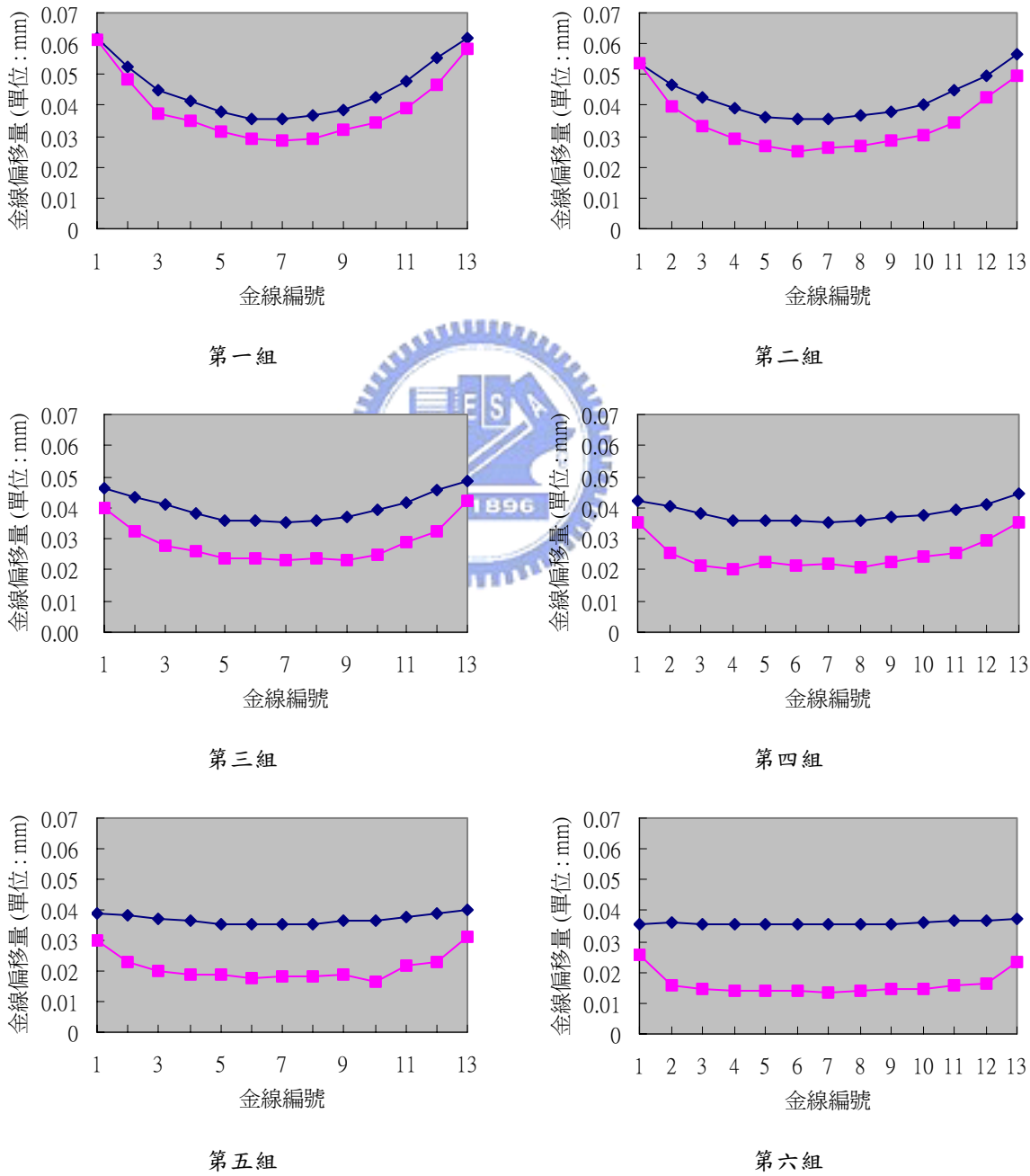


圖4-6、金線位移曲線圖

另外，當中有兩項值得注意的現象：

1. 位於較外側之兩種分析方法偏移量差異較小
2. 第一組與第二組中編號第一號之金線，兩種分析方法計算出的偏移量幾乎沒有差異，但第三組後差異便明顯增加

探求其原因，將觀察模穴於金線處ZY截面流速分佈情形（圖4-7），金線對流場的影響範圍在金線周圍，向上游處影響範圍較近而對下游處流場影響範圍較遠，因此，金線間的影響不僅受上游一金線影響，可能還會包括更前方的金線影響（視流場和金線間距而定），甚至，若金線距離更接近時也可能受到下游金線影響。因此，影響兩測金線的金線數較少，所以在二側的金線偏移量和傳統不考慮金線影響流場的方法所計算出的結果較為接近。此外，當金線間距較大時，上游金線不受下游金線影響，這也是第一組和第二組中第一號金線的偏移量與傳統分析方法相當接近之原因。

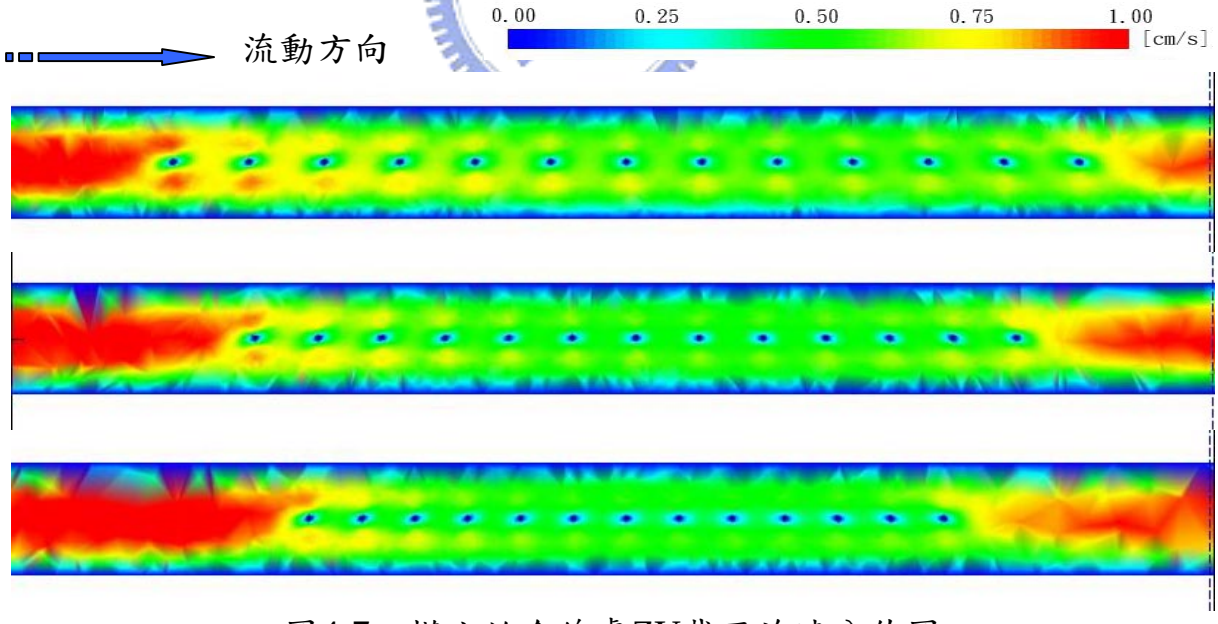


圖4-7、模穴於金線處ZY截面流速分佈圖

4.3 結果討論（一）

在兩種分析方法結果的比較中可發現，當整體流場分析中加入金線後，金線會造成流動阻礙，使得塑料在金線周圍的局部流場明顯改變，但是金線對於整體流場的影響並不明顯。當塑料流經金線時，塑料在金線周圍會產生一受影響範圍，範圍內流速減慢，當不同金線影響範圍重疊時，金線間便會影響個別的編移量，使得分析時金線偏移量減小，因此，流場分析中考慮金線影響後，偏移量大致都比傳統的分析方法小。並且，當金線越密集時，金線間的相互影響更加顯著，因此金線間距越小時，兩種方法所得到的偏移量差距越大。

4.4 金線分佈規劃（二）

先前文中的金線分佈規劃，因為考量計算機以及軟體的運算極限，僅在相同金線數目下，改變金線間距進行分析，為了實際了解金線密度對於偏移量的影響，進一步以增加金線數目的方式提高金線密度，進行金線偏移分析。

在此階段，金線的分佈，如同第一部份金線分佈規劃，僅將金線擺設於晶片左側，並且逐次將金線增加提高金線密度，金線分佈共分五組，金線數目分別為1、3、5、7、13條，以本研究所提出的金線偏移分析方法進行金線偏移分析；另外，傳統的金線偏移分析方法方面，由於整體流場不受金線的影響，各條金線偏移量並不受到金線彼此間的分不影響，所以此分析方法僅以一組13條金線進行偏移計算做為對照比較。金線接腳位置以及各金線編號位置圖如表4-2以及圖4-8所示。

表4-2、 各組金線接腳位置表

第一組						
NO.	第一接腳座標			第二接腳座標		
	x	y	z	x	y	z
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
第二組						
NO.	第一接腳座標			第二接腳座標		
	x	y	z	x	y	z
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
1	8.7713	14.7213	0.3418	12.5023	15.2523	0.672
13	8.7713	17.7213	0.3418	12.5023	17.0523	0.672
第三組						
NO.	第一接腳座標			第二接腳座標		
	x	y	z	x	y	z
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
1	8.7713	14.7213	0.3418	12.5023	15.2523	0.672
13	8.7713	17.7213	0.3418	12.5023	17.0523	0.672
4	8.7713	15.4713	0.3418	12.5023	15.7023	0.672
10	8.7713	16.9713	0.3418	12.5023	16.6023	0.672
第四組						
NO.	第一接腳座標			第二接腳座標		
	x	y	z	x	y	z
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
1	8.7713	14.7213	0.3418	12.5023	15.2523	0.672
13	8.7713	17.7213	0.3418	12.5023	17.0523	0.672
3	8.7713	15.2213	0.3418	12.5023	15.5523	0.672
9	8.7713	16.7213	0.3418	12.5023	16.4523	0.672
5	8.7713	15.7213	0.3418	12.5023	15.8523	0.672
11	8.7713	17.2213	0.3418	12.5023	16.7523	0.672
第五組						
NO.	第一接腳座標			第二接腳座標		
	x	y	z	x	y	z
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
1	8.7713	17.4713	0.3418	12.5023	16.9023	0.672
13	8.7713	17.7213	0.3418	12.5023	17.0523	0.672
2	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
8	8.7713	16.4713	0.3418	12.5023	16.3023	0.672
3	8.7713	16.4713	0.3418	12.5023	16.3023	0.672
9	8.7713	16.7213	0.3418	12.5023	16.4523	0.672
4	8.7713	16.7213	0.3418	12.5023	16.4523	0.672
10	8.7713	16.9713	0.3418	12.5023	16.6023	0.672
5	8.7713	16.9713	0.3418	12.5023	16.6023	0.672
11	8.7713	17.2213	0.3418	12.5023	16.7523	0.672
6	8.7713	17.2213	0.3418	12.5023	16.7523	0.672
12	8.7713	17.4713	0.3418	12.5023	16.9023	0.672

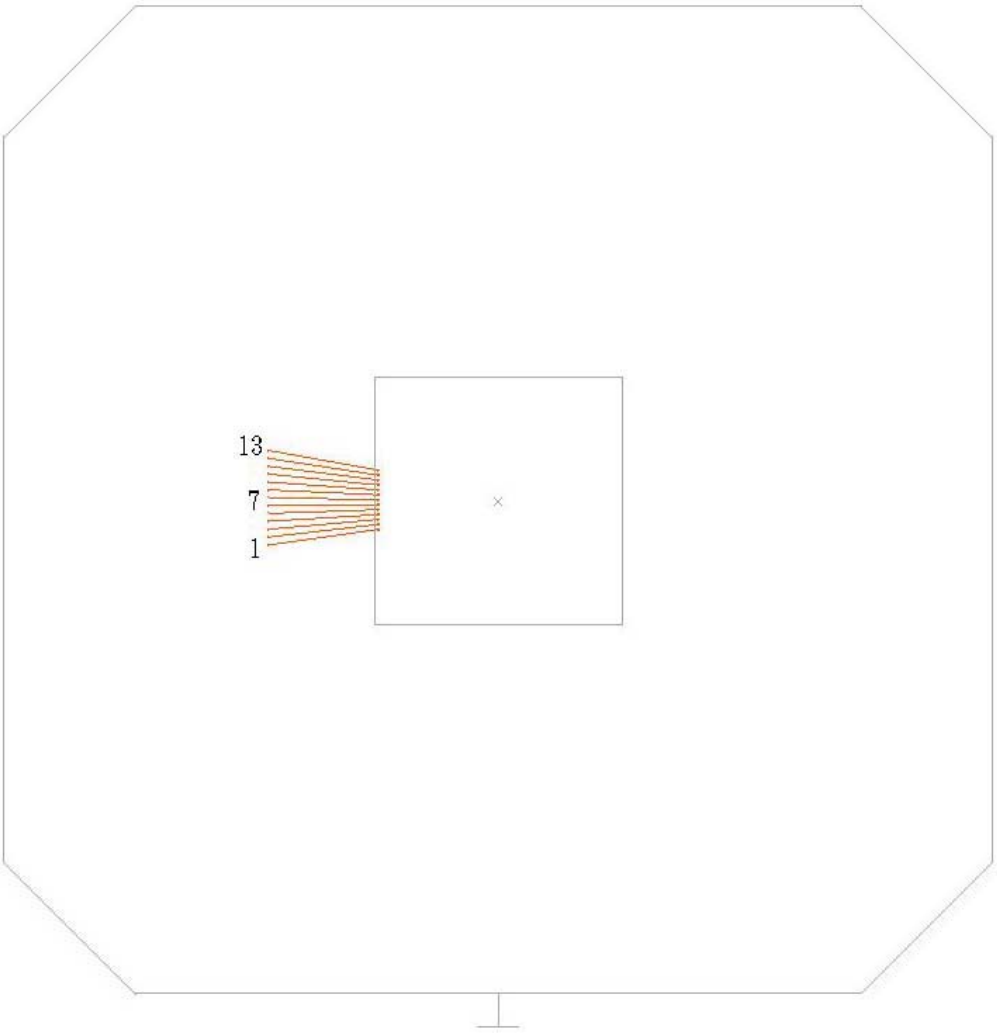


圖4-8、金線分佈及編號圖

4.5 結果討論（二）

圖4-9為各組金線偏移量計算結果中，取各金線節點中最大水平偏移量（xy平面上之位移量）之位移區線圖，由圖中可發現，當金線數越多密度越高時，兩種不同方法所得到的金線偏移結果差異越大，並且，當金線數目越多，本研究提出的分析方法所得到之金線偏移結果越小。因此可以確定，金線密度增加，將使得金線偏移偏移量分析結果減小。

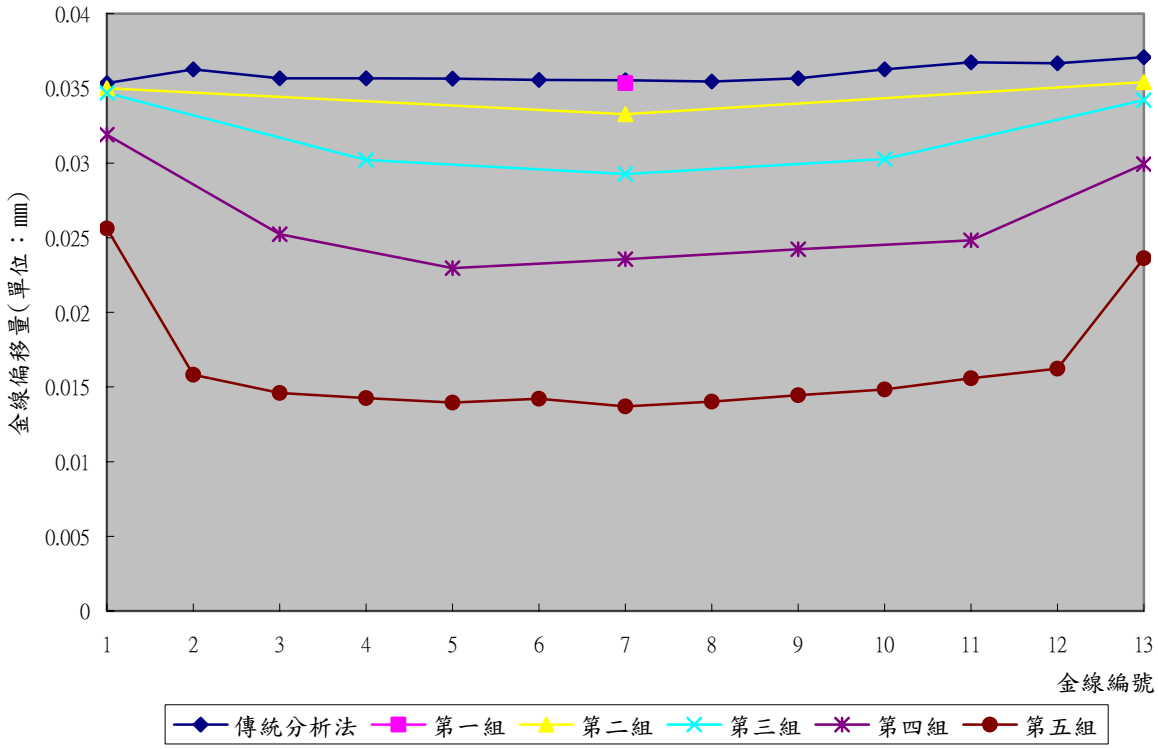


圖4-9、各組金線偏移結果曲線圖

4.6 金線回彈

在封模過程中，塑料流動使得金線受力產生偏移的現象，填充過程中，塑料熟化度隨塑料分子間的鏈結反應而上升，使得塑料黏度上升。塑料填充完全時，模穴中的塑料停止流動，金線上不再有拖曳力，這使得之前受塑料流動而變形的金線產生部份彈性回彈，使實際金線偏移量減小。完全填充後的冷卻過程，塑料黏度將隨著溫度下降而提高，直至塑料硬化，因此在塑料冷卻過程中，金線彈性回彈量將受到限制。本文中先前探討的金線偏移量，僅分析塑料填充過程中的金線偏移情形，對於填充成後的金線偏移回彈現象並未加以討論，故在此節將針對金線彈性回彈作初步探討。

4.6.1 金線回彈模型

從實際的封模製程推測，金線的偏移行為大致可以分為量階段，首先

是塑料填充時，在金線上引發的拖曳力所造成的偏移；接著，是塑料完全填充後，塑料冷卻過程中金線的彈性回彈。實際上，金線回彈並非單純和金線材料性質以及金線偏移量有關，金線在充滿塑料的模穴中回彈，亦受到塑料阻力牽制。塑料對於金線的回彈行為之影響，推測與塑料隨著時間變化的黏度和金線回彈速度有關，因此要建立金線在塑料中的回彈模型十分複雜，是一項值得進一步探討研究的議題，但由於此議題並非本研究中所討論的重點。在此僅希望了解金線變形後，在金線上是否發生塑性變形與回彈後金線偏移情形之關係，所以，假設塑料在過程中黏度不變亦不固化，並且，回彈時間足以讓金線彈範圍應變完全恢復。在這樣的假設條件下，是在金線偏移模擬後，將金線上所有塑料造成之拖曳力移除，並計算金線偏移恢復狀況。若是在這樣假設下，在充模過程中發生塑性變形之金線，回彈後仍存在偏移量，則可確定，在實際情況下或將塑料對回彈阻礙考慮時，金線回彈後必定無法恢復到未受拖曳力作用前的形狀而存在偏移量。

4.6.2 網格模型、材料及製程條件設定

在金線回彈模擬分析中，所使用的塑料網格，塑料以及金線材料均與本文先前之分析模擬相同，並且以分佈如同4.1節中第一組的13條金線進行分析。在製程條件設定方面，為了使得模擬中的塑性變形量增加以方便觀察，僅將充填時間所短為4秒，以增加塑料流經金線時的速度，使拖曳力提高，其餘製程參數仍與本文先前之分析模擬的設定相同。

4.6.3 分析結果與討論

圖4-10為塑料填充完全時，以ABAQUS計算出的金線偏移以及金線上主軸塑性應變分佈圖，由圖中可看出，編號1、2、3、4、11、12及13的金線上，接近接腳處應變超過降服應變而造成永久的塑性應變。金線於塑料完全填充時的編移量以及回彈後的編移量曲線圖中（圖4-11），當金線回彈

後，同樣在編號1、2、3、4、11、12及13的金線上仍然存在偏移量，其餘編號的金線回彈後偏移量分析結果的數值皆小於10^{-13} mm，為數值上之誤差，因此可當成不具偏移量。因此可以肯定，當金線在充模過程中若應變量超過降服，則回彈後，金線無法恢復到位受力偏移的狀態。

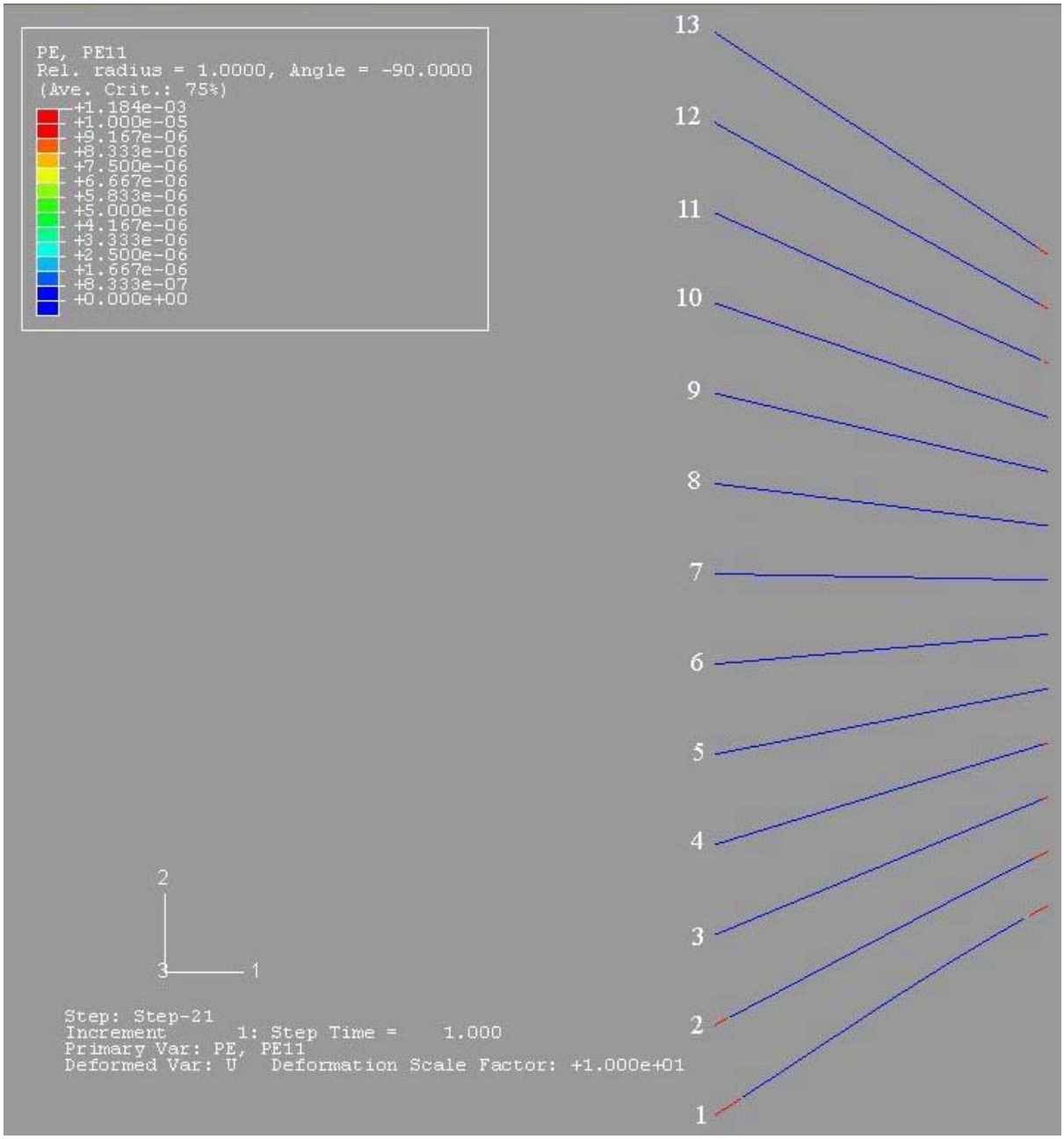


圖4-10、金線偏移及主軸塑性應變分佈圖

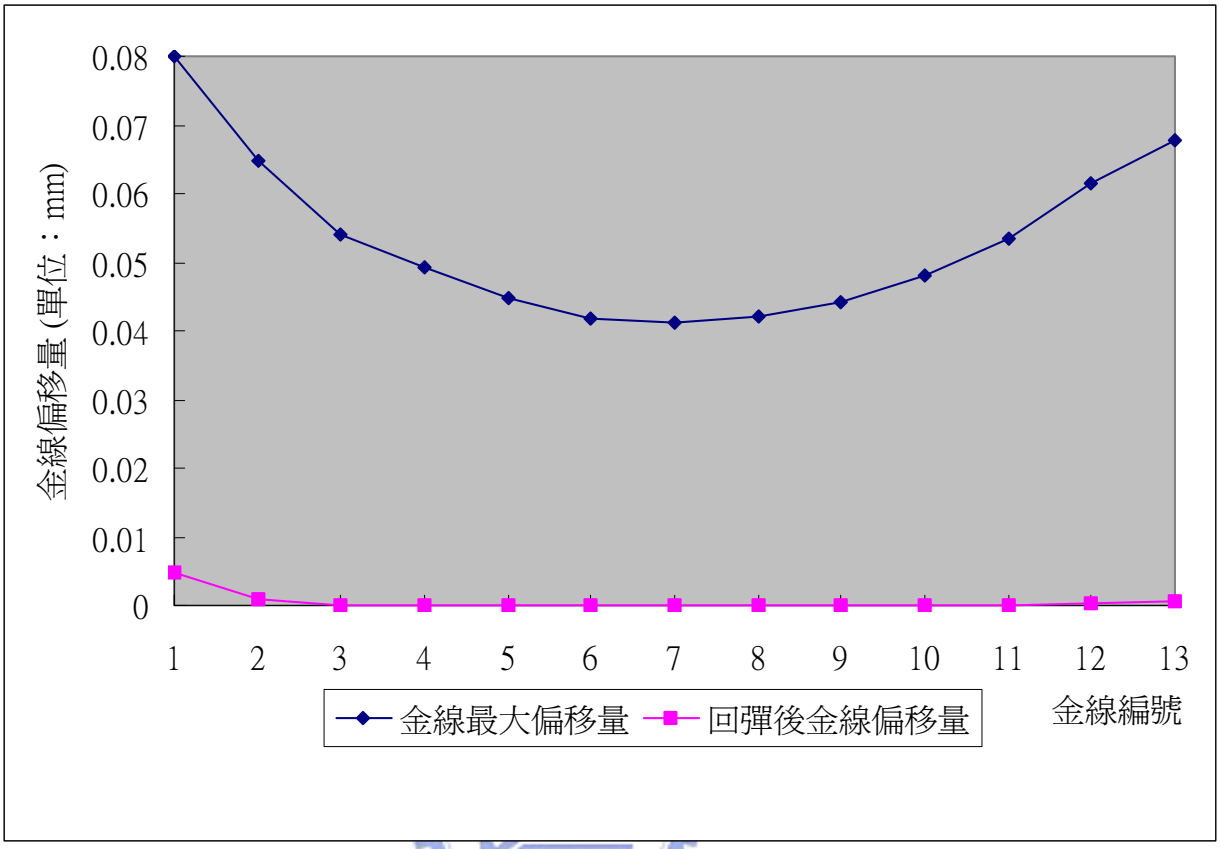


圖4-11、金線回彈後與回彈前偏移曲線圖

第五章　總結與未來展望

5.1　研究總結

本研究目的在探討以透過數值模擬金線偏移的情形時，對於金線間的相互影響。為了計算流場中加入金線考量，必須將直徑相當細小的金線網格加入整體流場網格中，這對於建立網格工作造成相當的困難度，若直接從CAD軟體繪製金線模型，金線複雜的輪廓曲度以及細小的直徑都會造成檔案在軟體間的承接問題，本研究利用自行撰寫的前端程式，用來幫助研究中建立包含金線之整體流場網格，使得建立網格的工作得以簡化並完成之，並確定了流場中加入金線的分析方法之可行性。另外，本研究亦自行撰寫程式用來計算流場在金線上所引發之拖曳力，並匯出用以計算金線偏移量的有限元素軟體匯入檔，改善了以往研究人員必須以耗時的人工作業方式。

本文參考並改進過去金線偏移的計算流程，提出針對高密度金線封裝之金線偏移量計算方法，以此方法計算出的結果和傳統的方法之結果作比較，歸納下列結論：

1. 金線會對流場造成局部波前落後的影響，但是同一金線數量下，金線的密集度影響波前前進情形不明顯。
2. 金線不僅影響金線區域範圍的壓力分布，對於距離此區域範圍較遠的流場也會有同樣的影響。
3. 金線僅使局部的流場流速減慢，對於整體流速分佈並無明顯影響。
4. 當金線對於流場的影響加入整體流場分析，所得到的金線偏移量小於不考慮金線對流場影響所得到的結果。
5. 當金線密度越高，則兩種方法所得到的結果差異越大。

6. 在一排金線分佈中，受彼此間的影響以中間處的金線效應較兩側金線大。
7. 本研究中，一排金線上以兩側的金線偏移量較大，也較容易在接腳處發生塑性變形，因此回彈後仍有少量的偏移量。

5.2 未來展望

本研究建立了考慮金線對流場影響的金線偏移分析方法，亦分析了金線密度對金線偏移之影響，但是在這樣的方法建立後，尚有許多的研究等待進行：

1. 本文研究過程中，尚無法以真實的金線偏移實驗作為模擬結果驗證，未來可以透過實驗來確定本文所提出的方法與傳統的方法之準確性比較。
2. 本文中的金線密度影響僅是一小部分的研究，能有相當多樣研究來探討金線密度的影響程度，如改變製程參數、塑料材料性質、不同模穴設計、不同金線擺設位置等。
3. 實際金線受力偏移後，在塑料尚未固化前會有彈性回彈的現象。本本文中以極簡化的模型初步分析回彈的情形，若可建立金線在塑料中更符合實際狀況的回彈模型並加入金線偏移後段分析，可使得金線偏移量更接近實際量測到之結果。
4. 實際金線接合後，在金線中會存有殘留應力，這對於金線受拖曳力後的的偏移量應該會有些許的影響，如何計算金線殘留應力以及加入初始殘留應力於金線偏移分析中，亦是值得深入探討的議題。

參考文獻

[1] L. T. Nguyen, “Wire Bond Behavior during Molding Operations of Electronic packages”, Polymer Eng. and Science, July 1988, vol. 28, pp. 926-943.

[2] L.T. Nguyen and F. J. Lim, “Wire Sweep During Molding of Integrated Circuits”, in IEEE 40th Proc. Electron. Comp. Conf., 1990, vol. 1, pp. 777-785.

[3] L. T. Nguyen, A. Danker, N. Santhiran, and C. R. Shervin, “Flow Modeling of Wire Sweep during Molding of Integrated Circuits”, ASME Writer Annual Meeting, 1992.

[4] A. A. O. Tay, K. S. Yeo, and J. H. Wu, “ The Effect of Wirebond Geometry and Die Setting on Wire Sweep”, IEEE Transactions on Components, Packaging, and Manufacturing Technology--Part B, 1995, vol. 18, pp. 201-209.

[5] A. A. O. Tay, K. S. Yeo, and J. H. Wu, “Numerical Simulation of Three-Dimensional Wirebond Deformation during Transfer Molding”, in Proc. 45th ECTC, 1995, pp. 999-1004.

[6] S. Han and K. K. Wang, “A Study on Wire Sweep in Encapsulation of Semiconductor Chip Using Simulated Experiments”, Journal of Electric Packaging, 1995, vol. 117, pp. 178-184.

[7] J. H. Wu, A. A. O. Tay, K. S. Yeo, and T. B. Lim, “A Three-Dimensional Modeling of Wire Sweep Incorporating Resin Cure”, IEEE Transactions on Component, Packaging, and Manufacturing Technology –Part B, 1998, vol. 21, pp. 65-72.

[8] L. Nguyen, H. Q. Yang, S. Bayyuk, S. Mazumder, S. Lowry, A.Krishnan, and A. Przekwas, “Time-Accurate, 3-D Computation of Wire Sweep During Plastic Encapsulation of Electric Component”, Journal of Pressure Vessel Technology, 2001, vol. 123, pp. 501-509.

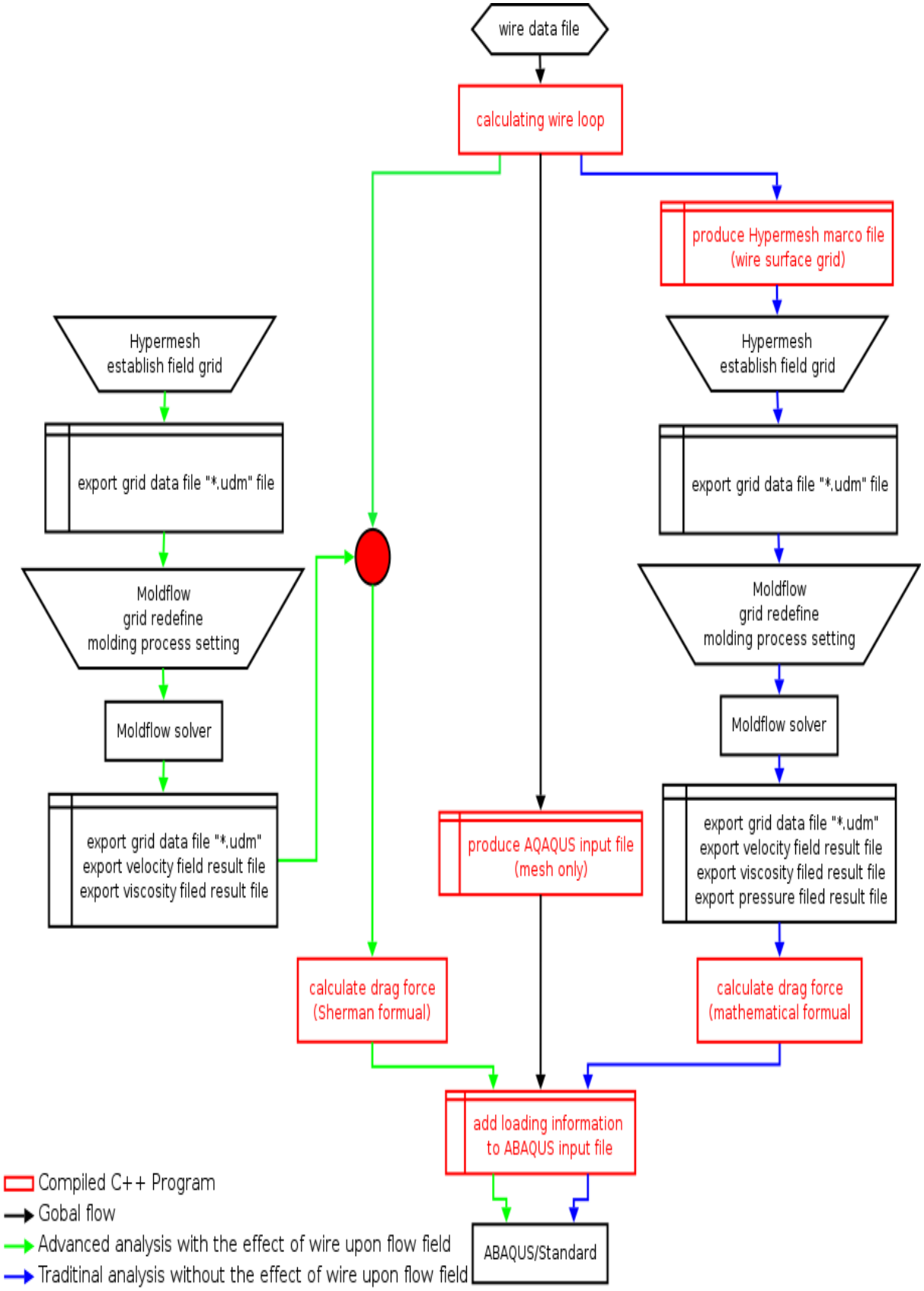
[9] C. C. Pei, Francis Su, S. J. Hwang, D. Y. Huang, and H. H. Lee, “Wire Density Effect on Wire Sweep Analysis for IC Packaging”, 8th International Symposium on Adavanced Packaging Materials, pp. 160-165.

[10] S. Han and K. K. Wang, “Flow Analysis in a Chip Cavity During Semiconductor Encapsulation”, Journal of Electronic Packaging, 2000, vol 122, pp. 160-167.

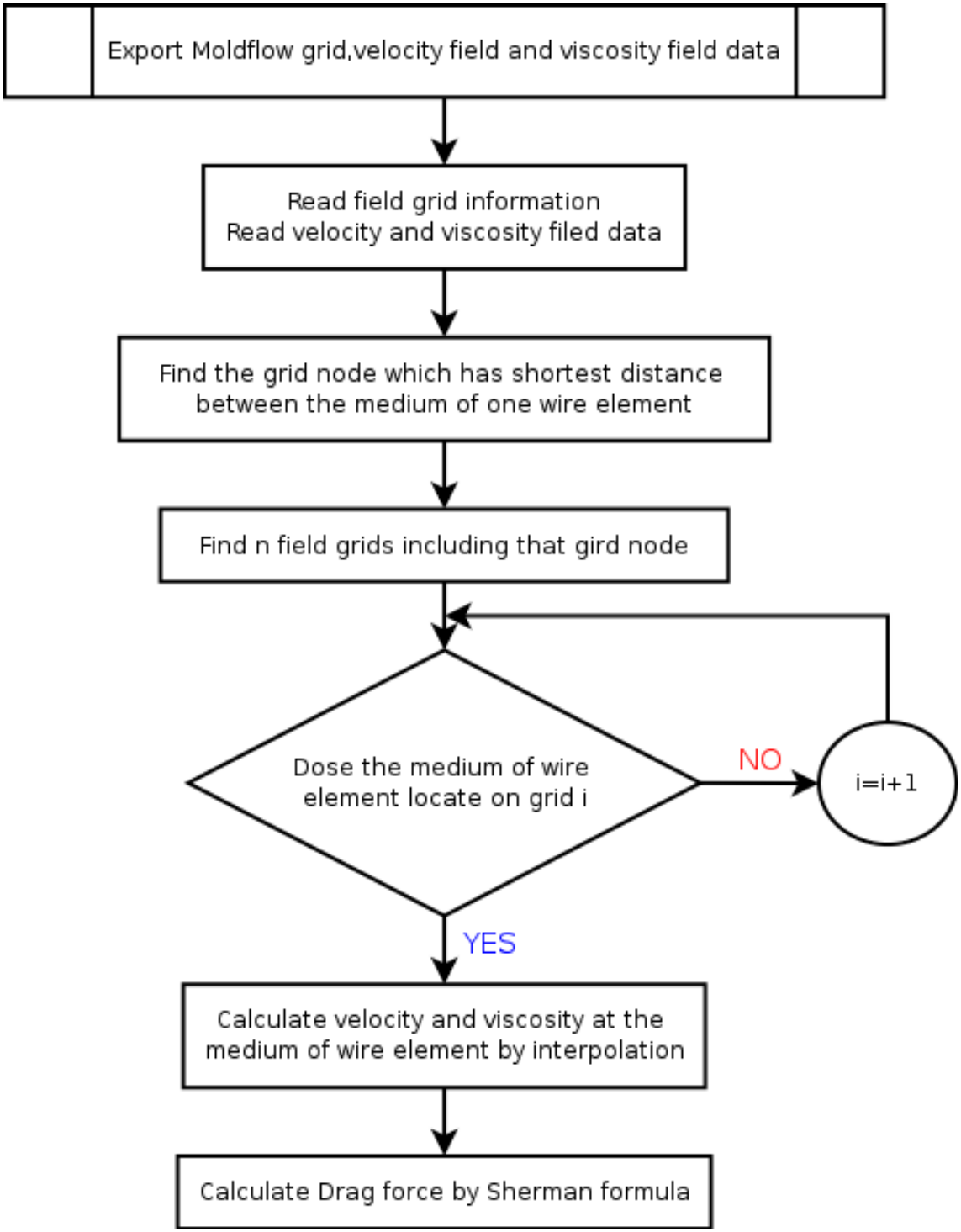
[11] 劉鴻汶，“IC 封裝製程之模流與金線偏移分析”，碩士論文，私立中原大學，2003。

[12] Frederick, S. Sherman, Viscous Flow, McGraw-Hill, New York, 1990.

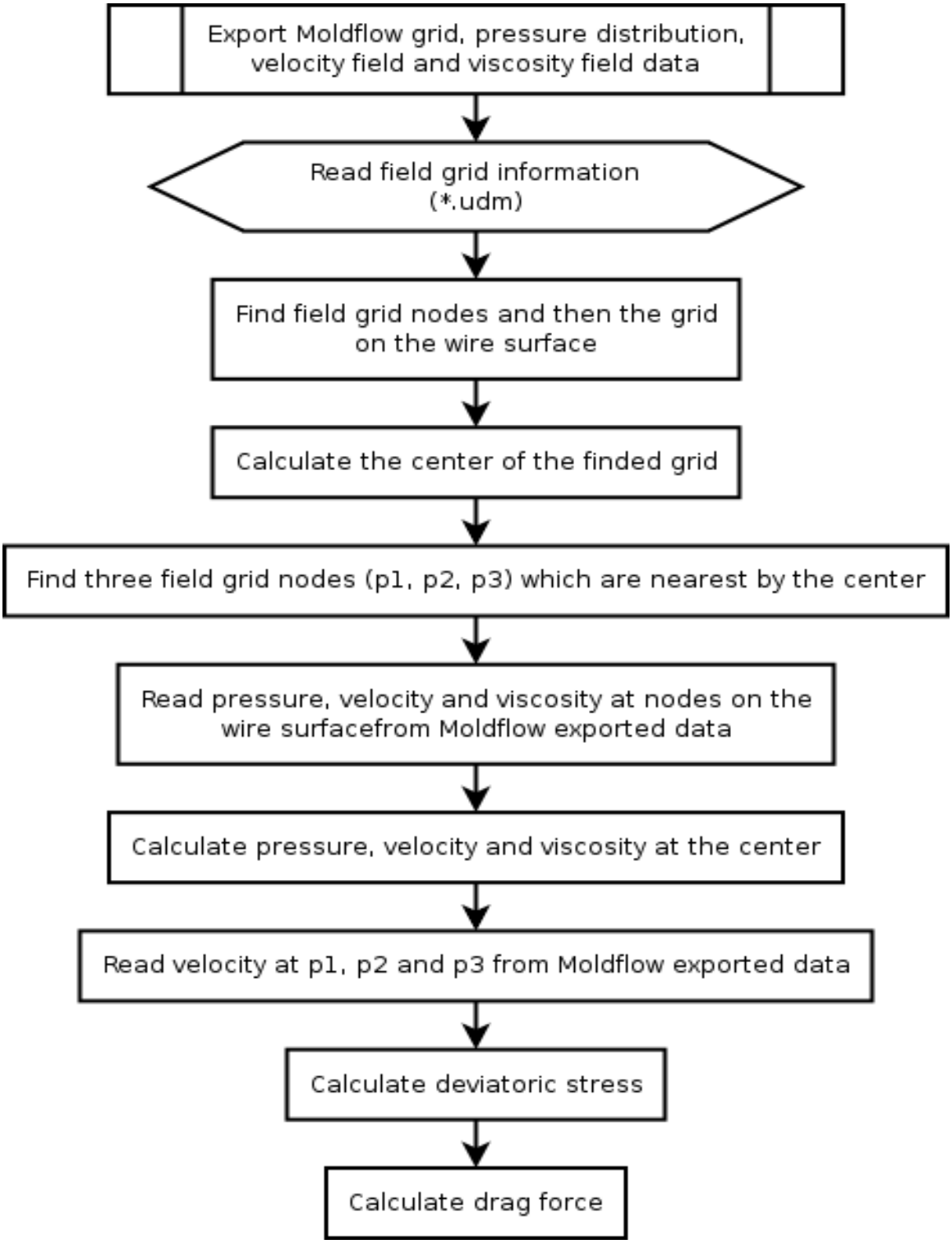
附錄A、 金線偏移分析流程圖



附錄B、 以Sherman方程式計算拖曳力程式流程圖



附錄C、 以數值方法計算拖曳力程式流程圖



附錄D、 金線接腳座標表

第一組						
NO.	第一接腳座標 (pitch=1mm)			第二接腳座標 (pitch=0.6mm)		
	x	y	z	x	y	z
1	8.7713	10.2213	0.3418	12.5023	12.5523	0.672
2	8.7713	11.2213	0.3418	12.5023	13.1523	0.672
3	8.7713	12.2213	0.3418	12.5023	13.7523	0.672
4	8.7713	13.2213	0.3418	12.5023	14.3523	0.672
5	8.7713	14.2213	0.3418	12.5023	14.9523	0.672
6	8.7713	15.2213	0.3418	12.5023	15.5523	0.672
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
8	8.7713	17.2213	0.3418	12.5023	16.7523	0.672
9	8.7713	18.2213	0.3418	12.5023	17.3523	0.672
10	8.7713	19.2213	0.3418	12.5023	17.9523	0.672
11	8.7713	20.2213	0.3418	12.5023	18.5523	0.672
12	8.7713	21.2213	0.3418	12.5023	19.1523	0.672
13	8.7713	22.2213	0.3418	12.5023	19.7523	0.672

第二組						
NO.	第一接腳座標 (pitch=0.85mm)			第二接腳座標 (pitch=0.51mm)		
	x	y	z	x	y	z
1	8.7713	11.1213	0.3418	12.5023	13.0923	0.672
2	8.7713	11.9713	0.3418	12.5023	13.6023	0.672
3	8.7713	12.8213	0.3418	12.5023	14.1123	0.672
4	8.7713	13.6713	0.3418	12.5023	14.6223	0.672
5	8.7713	14.5213	0.3418	12.5023	15.1323	0.672
6	8.7713	15.3713	0.3418	12.5023	15.6423	0.672
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
8	8.7713	17.0713	0.3418	12.5023	16.6623	0.672
9	8.7713	17.9213	0.3418	12.5023	17.1723	0.672
10	8.7713	18.7713	0.3418	12.5023	17.6823	0.672
11	8.7713	19.6213	0.3418	12.5023	18.1923	0.672
12	8.7713	20.4713	0.3418	12.5023	18.7023	0.672
13	8.7713	21.3213	0.3418	12.5023	19.2123	0.672

第三組						
NO.	第一接腳座標 (pitch=0.7mm)			第二接腳座標 (pitch=0.42mm)		
	x	y	z	x	y	z
1	8.7713	12.0213	0.3418	12.5023	13.6323	0.672
2	8.7713	12.7213	0.3418	12.5023	14.0523	0.672
3	8.7713	13.4213	0.3418	12.5023	14.4723	0.672
4	8.7713	14.1213	0.3418	12.5023	14.8923	0.672
5	8.7713	14.8213	0.3418	12.5023	15.3123	0.672
6	8.7713	15.5213	0.3418	12.5023	15.7323	0.672
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
8	8.7713	16.9213	0.3418	12.5023	16.5723	0.672
9	8.7713	17.6213	0.3418	12.5023	16.9923	0.672
10	8.7713	18.3213	0.3418	12.5023	17.4123	0.672
11	8.7713	19.0213	0.3418	12.5023	17.8323	0.672
12	8.7713	19.7213	0.3418	12.5023	18.2523	0.672
13	8.7713	20.4213	0.3418	12.5023	18.6723	0.672

第四組						
NO.	第一接腳座標 (pitch=0.55mm)			第二接腳座標 (pitch=0.33mm)		
	x	y	z	x	y	z
1	8.7713	12.9213	0.3418	12.5023	14.1723	0.672
2	8.7713	13.4713	0.3418	12.5023	14.5023	0.672
3	8.7713	14.0213	0.3418	12.5023	14.8323	0.672
4	8.7713	14.5713	0.3418	12.5023	15.1623	0.672
5	8.7713	15.1213	0.3418	12.5023	15.4923	0.672
6	8.7713	15.6713	0.3418	12.5023	15.8223	0.672
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
8	8.7713	16.7713	0.3418	12.5023	16.4823	0.672
9	8.7713	17.3213	0.3418	12.5023	16.8123	0.672
10	8.7713	17.8713	0.3418	12.5023	17.1423	0.672
11	8.7713	18.4213	0.3418	12.5023	17.4723	0.672
12	8.7713	18.9713	0.3418	12.5023	17.8023	0.672
13	8.7713	19.5213	0.3418	12.5023	18.1323	0.672

第五組						
NO.	第一接腳座標 (pitch=0.4mm)			第二接腳座標 (pitch=0.24mm)		
	x	y	z	x	y	z
1	8.7713	13.8213	0.3418	12.5023	14.7123	0.672
2	8.7713	14.2213	0.3418	12.5023	14.9523	0.672
3	8.7713	14.6213	0.3418	12.5023	15.1923	0.672
4	8.7713	15.0213	0.3418	12.5023	15.4323	0.672
5	8.7713	15.4213	0.3418	12.5023	15.6723	0.672
6	8.7713	15.8213	0.3418	12.5023	15.9123	0.672
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
8	8.7713	16.6213	0.3418	12.5023	16.3923	0.672
9	8.7713	17.0213	0.3418	12.5023	16.6323	0.672
10	8.7713	17.4213	0.3418	12.5023	16.8723	0.672
11	8.7713	17.8213	0.3418	12.5023	17.1123	0.672
12	8.7713	18.2213	0.3418	12.5023	17.3523	0.672
13	8.7713	18.6213	0.3418	12.5023	17.5923	0.672

第六組						
NO.	第一接腳座標 (pitch=0.25mm)			第二接腳座標 (pitch=0.15mm)		
	x	y	z	x	y	z
1	8.7713	14.7213	0.3418	12.5023	15.2523	0.672
2	8.7713	14.9713	0.3418	12.5023	15.4023	0.672
3	8.7713	15.2213	0.3418	12.5023	15.5523	0.672
4	8.7713	15.4713	0.3418	12.5023	15.7023	0.672
5	8.7713	15.7213	0.3418	12.5023	15.8523	0.672
6	8.7713	15.9713	0.3418	12.5023	16.0023	0.672
7	8.7713	16.2213	0.3418	12.5023	16.1523	0.672
8	8.7713	16.4713	0.3418	12.5023	16.3023	0.672
9	8.7713	16.7213	0.3418	12.5023	16.4523	0.672
10	8.7713	16.9713	0.3418	12.5023	16.6023	0.672
11	8.7713	17.2213	0.3418	12.5023	16.7523	0.672
12	8.7713	17.4713	0.3418	12.5023	16.9023	0.672
13	8.7713	17.7213	0.3418	12.5023	17.0523	0.672

附錄E、 程式原始碼

E.1 不考慮金線對塑料流動影響

E.1.1 使用者介面程式碼

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include "dforce.cpp"
#include "wire.cpp"
#include "ABAQUS_gen.cpp"
int main(){
        flow fl;
        wires wi;
        fl.fnodege("cavity.udm"); //read cavity grid data from Moldflow export file
        wi.getwire("wire.txt", 0.1, 0.025 ); //read wire informations for prepared file
        ABAQUS_MODEL_GEN(wi,"wiresweep.inp","property.txt"); //output wire FEM mesh data to ABAQUS input file
                                                             //read wire property from prepared file and output to input file
        wi.search_element(fl); // Haedle mesh configuration.
        //read Moldflow calculated velocity and viscosity at one time, calculate drag force, and then output to inp file.
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.001","viscosity.nod.001","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.002","viscosity.nod.002","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.003","viscosity.nod.003","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.004","viscosity.nod.004","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.005","viscosity.nod.005","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.006","viscosity.nod.006","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.007","viscosity.nod.007","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.008","viscosity.nod.008","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.009","viscosity.nod.009","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.010","viscosity.nod.010","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.011","viscosity.nod.011","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.012","viscosity.nod.012","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.013","viscosity.nod.013","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.014","viscosity.nod.014","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.015","viscosity.nod.015","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.016","viscosity.nod.016","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.017","viscosity.nod.017","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.018","viscosity.nod.018","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.019","viscosity.nod.019","wiresweep.inp");
        ABAQUS_STEP_GEN(wi,fl,"velocity.nod.020","viscosity.nod.020","wiresweep.inp");
        return 0;
};
```

E.1.2 主程式碼

```
#include <math.h>
class fnode
{
        public:
```

```
                int lab; //The label of flow nodes.
                double x[3]; //The coordinates of flow node.
                double ve[3], vi;//The velocity and viscosity on node.
};

class element
{
        public:
                int node_lab[4];
};

class flow
{
        public:
                int num; //the number of fnode.
                int elnum; //the number of element.
                int nofnsearch(fstream &);
                int fnodege(char *);
                fnode *n;
                element *el;
                int velocityread(char *);
                int viscosityread(char *);
                float time;
                ~flow(){
                        delete [] n;
                };
};

int flow::nofnsearch(fstream &input){
        char command[200];
        int tmp;
        while(!input.eof()){
                input.getline(command,200);
                if(check("SUMR{\0",command)==1){
                        while(!input.eof()){
                                tmp=input.tellg();
                                input>>command;
                                if(check("NOND{",command)==1){
                                        input.seekg(tmp+8,ios::beg);
                                        input>>num;
                                        break;
                                };
                        };
                        while(!input.eof()){
                                tmp=input.tellg();
                                input>>command;
                                if(check("NOT4{",command)==1){
                                        input.seekg(tmp+8,ios::beg);
                                        input>>elnum;
                                        break;
                                };
                        };
                        break;
                };
        };
        return 0;
};
```

```
int flow::fnodege(char *ifile){
        int i,j;
        char command[200];
        int tmp;
        double ftmp;
        fstream input;
        input.open(ifile, ios::in);
        if(!input){
                cerr<<"Can't open flow node input    file!!"<<endl;
                exit(1);
        };
        // Search the number of nodes.
        nofnsearch(input);
        cout<<"The number of flow node is "<<num<<endl;
        cout<<"The number of flow element is "<<elnum<<endl;
        n=new fnode[num];
        el=new element [elnum];
        // read the coordinates of flow nodes.
        while(!input.eof()){
                tmp=input.tellg();
                input.getline(command,200);
                if(check("NODE{",command)){
                        input.seekg(tmp,ios::beg);
                        break;
                };
        };
        for(i=0;i<num;i++){
                input.get(command,6);
                input>>n[i].lab>>tmp>>tmp>>tmp>>ftmp;
                input>>n[i].x[0]>>n[i].x[1]>>n[i].x[2];
                input.getline(command,2);
        };
        // read the coordinates of flow nodes.
        while(!input.eof()){
                tmp=input.tellg();
                input.getline(command,200);
                if(check("TET4{",command)){
                        input.seekg(tmp+6,ios::beg);
                        for(i=0;i<elnum;i++){
                                input>>tmp>>tmp>>tmp>>tmp>>tmp>>command>>tmp>>tmp;
                                input>>el[i].node_lab[0]>>el[i].node_lab[1]>>el[i].node_lab[2]>>el[i].node_lab[3];
                                input>>command>>command;
                        };
                        break;
                };
        };
        cout<<"fnode is dernerated!!\n";
        return 0;
};

int flow::velocityread(char *ifile){
        int lab;
        char null[80];
        int tmp;
        int i,j;
        fstream input;
        for(i=0;i<num;i++){
                for(j=0;j<3;j++)
```

```
                        n[i].ve[j]=0;
        };
        input.open(ifile,ios::in);
        if(!input){
                cerr<<"Can't open the velocuty input file"<<endl;
                exit(1);
        };
        while(!input.eof()){
                tmp=input.tellg();
                input>>null;
                if(null[0]>='0' && null[0]<='9'){
                        input.seekg(tmp,ios::beg);
                        input>>time;
                        input>>null;
                        break;
                };
        };
        input.getline(null,80);
        input.getline(null,80);
        input.getline(null,80);
        while(!input.eof()){
                input>>lab;
                input>>n[lab-n[0].lab].ve[0]>>n[lab-n[0].lab].ve[1]>>n[lab-n[0].lab].ve[2];
                input.getline(null,80);
        };
        input.close();
        cout<<"velocutyread is done!!\n";
        return 0;
};

int flow::viscosityread(char *ifile){
        int lab;
        char null[80];
        int i;
        int tmp;
        fstream input;
        for(i=0;i<num;i++){
                n[i].vi=0;
        };
        input.open(ifile,ios::in);
        if(!input){
                cerr<<"Can't open the viscosity input file"<<endl;
                exit(1);
        };
        while(!input.eof()){
                tmp=input.tellg();
                input>>null;
                if(null[0]>='0' && null[0]<='9'){
                        input.seekg(tmp,ios::beg);
                        input>>time;
                        break;
                };
        };
        input.getline(null,80);
        input.getline(null,80);
        input.getline(null,80);
        while(!input.eof()){
                input>>lab;
```

```
                input>>n[lab-n[0].lab].vi;
                input.getline(null,80);
        };
        input.close();
        cout<<" viscosityread is done!!\n";
        return 0;
};

double volume(double *n1, double *n2, double *n3, double *n4){
        double v1[3],v2[3],v3[3];
        double vol;
        int i;
        for(i=0;i<3;i++){
                v1[i]=(n2[i]-n1[i]);
                v2[i]=(n3[i]-n1[i]);
                v3[i]=(n4[i]-n1[i]);
        };
        vol=0;
        for(i=0;i<3;i++){
                vol=v1[i]*(v2[int(fmod(i+1,3))]*v3[int(fmod(i+2,3))]-v2[int(fmod(i+2,3))]*v3[int(fmod(i+1,3))])+vol;
        };
        vol=sqrt(pow(vol,2));
        return vol;
};

class node
{
        public:
                double x[3];
                double fn[3],ft[3];
                int FN[4];  //Flow node where wire element mids locate.
                double shape[4];
                double EPS;
};

class wire
{
        public:
                char null;
                int nu;
                int NOWN;
                double p1[3],p2[3];
                double l,H;
                node *n;
                int cal(double);
                int read(fstream &);
                double func1(double x){
                        double y;
                        y=-2*H*pow(x,3)/pow(l,3)+3*H*pow(x,2)/pow(l,2);
                        return y;
                };
                double pfunc1(double x){
                        double y;
                        y=-6*H*pow(x,2)/pow(l,3)+6*H*pow(x,1)/pow(l,2);
                        return y;
                };
                double func2(double x){
                        double y;
```

```
                        double L;
                        L=sqrt(pow((p2[0]-p1[0]),2)+pow((p2[1]-p1[1]),2));
                        if(x==l)
                                y=H;
                        else if(x>L)
                                y=p2[2]-p1[2];
                        else
                                y=(p2[2]-p1[2])+(H-(p2[2]-p1[2]))*sqrt(1-pow((x-l)/(L-l),2));
                        return y;
                };
                double pfunc2(double x){
                        double y,yp,xp;
                        double slop;
                        xp=x*(1+1e-5);
                        y=func2(x);
                        yp=func2(xp);
                        slop=(yp-y)/(xp-x);
                        return slop;
                };
                ~wire(){
                        delete [] n;
                };
};

class wires
{
        public:
                wire *wi;
                int NOWI;
                double time;
                double radius;
                int getwire(char *, double, double );
                int dforce(flow &, char *, char *);
                int search_element(flow &);
                int moldflowvbs(char *);
};


int wire::read(fstream &input ){
        input>>nu>>null;
        input>>null>>p1[0]>>null>>p1[1]>>null>>p1[2]>>null>>null;
        input>>null>>p2[0]>>null>>p2[1]>>null>>p2[2]>>null>>null;
        input>>H>>null>>l;
        return 0;
};

int wire::cal(double esize){
        double x,y,xp,yp,slop;
        int i;
        int lab;
        double tmp;
        fstream pout,inp;
        pout.open("tmp",ios::out|ios::trunc);
        NOWN=0;
        x=sqrt(pow((p2[0]-p1[0]),2)+pow((p2[1]-p1[1]),2));
        y=p2[2]-p1[2];
        xp=0;
        pout<<NOWN<<" "<<0<<" "<<0<<endl;
```

```
		NOWN++;
		while(1){
			slop=pfunc1(xp);
			tmp=esize/sqrt(pow(slop,2)+1);
			slop=pfunc1(xp+tmp/2);
			tmp=esize/sqrt(pow(slop,2)+1);
			slop=pfunc1(xp+tmp/2);
			tmp=esize/sqrt(pow(slop,2)+1);
			xp=xp+tmp;
			if(xp>l){
				xp=xp-tmp;
				break;
			};
			yp=func1(xp);
			pout<<NOWN<<" "<<xp<<" "<<yp<<endl;
			NOWN++;
		};
		xp=l;
		yp=func1(xp);
		pout<<NOWN<<" "<<xp<<" "<<yp<<endl;
		NOWN++;
		while(1){
			slop=pfunc2(xp);
			tmp=esize/sqrt(pow(slop,2)+1);
			slop=pfunc2(xp+tmp/2);
			tmp=esize/sqrt(pow(slop,2)+1);
			slop=pfunc2(xp+tmp/2);
			tmp=esize/sqrt(pow(slop,2)+1);
			xp=xp+tmp;
			if(xp>x){
				xp=xp-tmp;
				break;
			};
			yp=func2(xp);
			pout<<NOWN<<" "<<xp<<" "<<yp<<endl;
			NOWN++;
		};
		xp=x;
		yp=func2(xp);
		pout<<NOWN<<" "<<xp<<" "<<yp<<endl;
		NOWN++;
		pout.close();
		n = new node[NOWN];
		inp.open("tmp",ios::in);
		do{
			inp>>lab>>xp>>yp;
			n[lab].x[0]=xp*(p2[0]-p1[0])/x+p1[0];
			n[lab].x[1]=xp*(p2[1]-p1[1])/x+p1[1];
			n[lab].x[2]=yp+p1[2];
		}while(!inp.eof());
		inp.close();
		return 0;
};

int wires::getwire(char *ifile, double esize, double rad){ //"NOWN" is the number of nodes of each wire
		radius=rad;
		time=0;
		char null;
```

```
float tmp;
int wn=0; //the number of wires.
int i;
fstream input,output;
//check the number of wire from "wire.txt".
input.open(ifile,ios::in);
do{
        input>>tmp>>null;
        if(null!=':'){
                cout<<"Wire input file error!!\n"<<endl;;
                exit(0);
        };
        input>>null>>tmp>>null>>tmp>>null>>tmp>>null>>null;
        if(null!=','){
                cout<<"Wire input file error!!\n"<<endl;
                exit(0);
        };
        input>>null>>tmp>>null>>tmp>>null>>tmp>>null>>null;
        if(null!=','){
                cout<<"Wire input file error!!\n"<<endl;
                exit(0);
        };
        input>>tmp>>null;
        if(null!=','){
                cout<<"Wire input file error!!\n"<<endl;
                exit(0);
        };
        input>>tmp;
        wn++;
        input>>null;
        if(null=='*'){
                break;
        };
        input.seekg(-1,ios::cur);
}while(!input.eof());
input.close();
NOWI=wn;
cout<<"Number of wire ="<<NOWI<<endl;
wi=new wire[wn];
// Reading each wire data
input.open(ifile,ios::in);
for(i=0;i<NOWI;i++){
        wi[i].read(input);
};
input.close();
cout<<"wire data reading is done\n";
// Calculating the coordinates of nodes of each wire.
for(i=0;i<NOWI;i++){
        wi[i].cal(esize);
};
cout<<"Calculation of coordinates of node is done!!\n";
output.open("wire.scr", ios::out);
int j;
for(i=0;i<NOWI;i++){
        output<<"spline\n";
        for(j=0;j<wi[i].NOWN;j++){
        output<<wi[i].n[j].x[0]<<","<<wi[i].n[j].x[1]<<","<<wi[i].n[j].x[2]<<endl;
        };
```

```
                output<<endl<<endl<<endl;
        };
        output.close();
        return 0;
};

int wires::dforce(flow &fl, char *velocity, char *viscosity){
        int i,j,k,m;
        double p1[3],p2[3];
        double vi, ve[3], vt[3], vn[3];
        double shape[4];
        double tmp,tmp1;
        int s[4];
        fstream out;
        fl.velocityread(velocity);
        fl.viscosityread(viscosity);
        out.open("report.msg",ios::out|ios::app);
        out<<"\n\nAt Time = "<<fl.time<<endl;
        for(i=0;i<NOWI;i++){
                for(j=0;j<wi[i].NOWN-1;j++){
                        for(k=0;k<3;k++){
                                p1[k]=wi[i].n[j].x[k];
                                p2[k]=wi[i].n[j+1].x[k];
                        };
                        l=0;
                        for(k=0;k<3;k++)
                                l=pow(p2[k]-p1[k],2)+l;
                        l=sqrt(l);
                        EPS=l/radius;
                        for(k=0;k<3;k++){
                                ve[k]=0;
                                vt[k]=0;
                        };
                        for(k=0;k<4;k++){
                                s[k]=wi[i].n[j].FN[k];
                                shape[k]=wi[i].n[j].shape[k];
                        };
                        vi=0;
                        for(k=0;k<4;k++)
                                vi=shape[k]*fl.n[s[k]].vi+vi;
                        for(k=0;k<3;k++){
                                for(m=0;m<4;m++)
                                        ve[k]=shape[m]*fl.n[s[m]].ve[k]+ve[k];
                        };
                        out<<"\tAt wire "<<i<<", element "<<j+1<<":";
                        tmp1=0;
                        for(k=0;k<3;k++){
                                tmp1=ve[k]*(p2[k]-p1[k])/l+tmp1;
                        };
                        for(k=0;k<3;k++){
                                vt[k]=tmp1*(p2[k]-p1[k])/l;
                                vn[k]=ve[k]-vt[k];
                        };
                        out<<"\t\tViscosity = "<<vi<<endl;
                        out<<"\t\tTangent velocity = "<<sqrt(pow(vt[0],2)+pow(vt[1],2)+pow(vt[2],2))<<endl;
                        out<<"\t\tNormal velocity = "<<sqrt(pow(vn[0],2)+pow(vn[1],2)+pow(vn[2],2))<<endl;
                        for(k=0;k<3;k++){
                                wi[i].n[j].ft[k]=4*3.1416*vi*vt[k]/(2*log(EPS)-1)/1000;
```

```
                                        if(log(EPS)<=1){
                                                cerr<<"wire element length too short!!\n";
                                                wi[i].n[j].ft[k]=0;
                                        };
                                        wi[i].n[j].fn[k]=8*3.1416*vi*vn[k]/(2*log(EPS)+1)/1000;
                                };
                                out<<"\t\tTangent force =
"<<sqrt(pow(wi[i].n[j].ft[0],2)+pow(wi[i].n[j].ft[1],2)+pow(wi[i].n[j].ft[2],2))<<endl;
                                out<<"\t\tNormal force =
"<<sqrt(pow(wi[i].n[j].fn[0],2)+pow(wi[i].n[j].fn[1],2)+pow(wi[i].n[j].fn[2],2))<<endl;
                        };
        };
        out.close();
        return 0;
};

int wires::search_element(flow &fl){
        double delta=0;
        double mid[3];
        double p1[3],p2[3];
        double vol[5];
        int i,j,k,m,l;
        int *smallest,step;
        int s[4];
        double tmp;
        int itmp;
        double *rp;
        double max_residual;
        cout<<"Elements containing wire element mid point is searching...\n";
        smallest=new int[fl.num];
        rp =new double[fl.num];
        for(i=0;i<NOWI;i++){                    // for each wire.
                for(j=0;j<wi[i].NOWN-1;j++){    // for each wire element.
                        // Define the mid of wire element.
                        for(k=0;k<3;k++){
                                p1[k]=wi[i].n[j].x[k]/1e3;
                                p2[k]=wi[i].n[j+1].x[k]/1e3;
                                mid[k]=(p2[k]+p1[k])/2;
                        };
                        max_residual=pow(pow(p1[0]-p2[0],2)+pow(p1[1]-p2[1],2)+pow(p1[2]-p2[2],2),3.0/2.0)/1e5;
                        // Calaulate the distances of each flow node to wire element mid.
                        for(k=0;k<fl.num;k++){
                                rp[k]=0;
                                for(m=0;m<3;m++)
                                        rp[k]=rp[k]+pow(mid[m]-fl.n[k].x[m],2);
                                rp[k]=sqrt(rp[k]);
                        };
                        // Search the three nearest flow node of the wire element mid.
                        for(k=0;k<fl.num;k++)
                                smallest[k]=k;
                        for(k=0;k<50;k++){
                                for(m=fl.num-1;m>k;m--){
                                        if(rp[smallest[m]]<rp[smallest[m-1]]){
                                                itmp=smallest[m];
                                                smallest[m]=smallest[m-1];
                                                smallest[m-1]=itmp;
                                        };
                                };
```

```
                                        };
                                        step=1;
here:
                                        delta=max_residual/1000*step;
                                        for(l=0;l<50;l++){
                                                for(k=0;k<fl.elnum;k++){
                                                        if(fl.n[smallest[l]].lab==fl.el[k].node_lab[0]||
                                                           fl.n[smallest[l]].lab==fl.el[k].node_lab[1]||
                                                           fl.n[smallest[l]].lab==fl.el[k].node_lab[2]||
                                                           fl.n[smallest[l]].lab==fl.el[k].node_lab[3]){
                                                                for(m=0;m<4;m++)
                                                                        s[m]=fl.el[k].node_lab[m]-fl.n[0].lab;
                                                                vol[0]=volume(fl.n[s[0]].x,fl.n[s[1]].x,fl.n[s[2]].x,fl.n[s[3]].x);
                                                                vol[1]=volume(fl.n[s[3]].x,fl.n[s[1]].x,fl.n[s[2]].x,mid);
                                                                vol[2]=volume(fl.n[s[0]].x,fl.n[s[3]].x,fl.n[s[2]].x,mid);
                                                                vol[3]=volume(fl.n[s[0]].x,fl.n[s[1]].x,fl.n[s[3]].x,mid);
                                                                vol[4]=volume(fl.n[s[0]].x,fl.n[s[1]].x,fl.n[s[2]].x,mid);
                                                                tmp=vol[0]-(vol[1]+vol[2]+vol[3]+vol[4]);
                                                                if(tmp>=-delta){
                                                                        cout<<"For wi["<<i<<"], node["<<j<<"]:    ";
                                                                        cout<<"Residual size="<<tmp<<endl;
                                                                        cout<<"Wire element mid
point:"<<mid[0]<<","<<mid[1]<<","<<mid[2];
                                                                        cout<<endl;
                                                                        cout<<"flow element is "<<k+1<<endl<<endl;
                                                                        goto there;
                                                                };
                                                        };
                                                };
                                        };
                                        if(k>=fl.elnum){
                                                step++;
                                                cout<<step<<endl;
                                                if(step>1000){
                                                        cout<<"Can't find any flow element containing the wire element mid.=>\n";
                                                        cout<<"wire number:"<<i<<";"<<"node_lab:"<<j<<endl;
                                                        exit(1);
                                                };
                                                goto here;
                                        };
there:
                                        for(k=0;k<4;k++){
                                                wi[i].n[j].FN[k]=s[k];
                                                wi[i].n[j].shape[k]=vol[k+1]/vol[0];
                                        };
                        };
                };
                fstream out;
                out.open("check.cmf",ios::out);
                k=0;
                for(i=0;i<NOWI;i++){
                        for(j=0;j<wi[i].NOWN;j++){

                out<<"*createnode("<<wi[i].n[j].x[0]/1e3<<","<<wi[i].n[j].x[1]/1e3<<","<<wi[i].n[j].x[2]/1e3<<",0,0,0)\n";
                                        k++;
                        };
                };
                for(i=0;i<NOWI;i++){
```

```
                for(j=0;j<wi[i].NOWN-1;j++){

        out<<"*createnode("<<fl.n[wi[i].n[j].FN[0]].x[0]<<","<<fl.n[wi[i].n[j].FN[0]].x[1]<<","<<fl.n[wi[i].n[j].FN[0]].x[2]<<",0,0,0)\n
";

        out<<"*createnode("<<fl.n[wi[i].n[j].FN[1]].x[0]<<","<<fl.n[wi[i].n[j].FN[1]].x[1]<<","<<fl.n[wi[i].n[j].FN[1]].x[2]<<",0,0,0)\n
";

        out<<"*createnode("<<fl.n[wi[i].n[j].FN[2]].x[0]<<","<<fl.n[wi[i].n[j].FN[2]].x[1]<<","<<fl.n[wi[i].n[j].FN[2]].x[2]<<",0,0,0)\n
";

        out<<"*createnode("<<fl.n[wi[i].n[j].FN[3]].x[0]<<","<<fl.n[wi[i].n[j].FN[3]].x[1]<<","<<fl.n[wi[i].n[j].FN[3]].x[2]<<",0,0,0)\n
";
                        out<<"*createlist(nodes,1) "<<k+1<<" "<<k+2<<" "<<k+3<<" "<<k+4<<endl;
                        k+=4;
                        out<<"*createelement(204,1,1,1)\n";
                };
        };
        out.close();
        delete [] rp;
        return 0;
};

int ABAQUS_MODEL_GEN(wires w, char *ifile, char *pfile){
        int i,j,k;
        char null[200];
        int lab;
        fstream out;
        fstream in;
        out.open(ifile,ios::out|ios::trunc);
        in.open(pfile,ios::in);
        out<<"*HEADING"<<endl;
        //NODE;
        out<<"*NODE"<<endl;
        for(i=0;i<w.NOWI;i++){
                for(j=0;j<w.wi[i].NOWN;j++){
                        lab=i*100+j+1;
                        out<<lab;
                        for(k=0;k<3;k++)
                                out<<","<<w.wi[i].n[j].x[k];
                        out<<endl;
                };
        };
        //ELEMENT
        out<<"*ELEMENT, ELSET=WIRE, TYPE=B31"<<endl;
        for(i=0;i<w.NOWI;i++){
                for(j=0;j<w.wi[i].NOWN-1;j++){
                        lab=i*100+j+1;
                        out<<lab;
                        out<<","<<i*100+j+1<<","<<i*100+j+2<<endl;
                };
        };
        //NODE SET
        for(i=0;i<w.NOWI;i++){
                out<<"*NSET, nset=wire"<<i<<", generate"<<endl;
                out<<i*100+1<<","<<i*100+w.wi[i].NOWN<<",1"<<endl;
        };
        //ELEMENT SET
```

```
	for(i=0;i<w.NOWI;i++){
		for(j=0;j<w.wi[i].NOWN-1;j++){
			lab=i*100+j+1;
			out<<"*Elset,\telset=el"<<lab<<endl;
			out<<lab<<endl;
		};
	};
	// Material
	out<<"*MATERIAL, NAME=gold-wire";
	while(!in.eof()){
		out<<endl;
		in.getline(null,200);
		out<<null;
	};
	//SECTION
	out<<"*BEAM SECTION, SECTION=CIRC, Elset=WIRE, Material=gold-wire\n";
	out<<w.radius<<endl;
	//Initial Boundary Conditions.
	//Fixed at wire ends.
	out<<"*BOUNDARY\n";
	for(i=0;i<w.NOWI;i++){
		out<<i*100+1<<", 1, 6\n";
		out<<i*100+w.wi[i].NOWN<<", 1, 6\n";
	};
	out.close();
	in.close();
	return 0;
};

void ABAQUS_STEP_GEN(wires &wi, flow &fl, char *velocity, char *viscosity, char *ifile){
	int i,j;
	double time;
	wi.dforce(fl, velocity, viscosity);
	time=fl.time-wi.time;
	fstream out;
	out.open(ifile,ios::out|ios::app);
	out<<"*Step,		Amplitude=RAMP\n";
	out<<"*Static\n";
	out<<time/5<<","<<time<<","<<time/100000<<","<<time/5<<endl;
	out<<"*DLOAD\n";
	for(i=0;i<wi.NOWI;i++){
		for(j=0;j<wi.wi[i].NOWN-1;j++){
			if(wi.wi[i].n[j].ft[0]+wi.wi[i].n[j].fn[0]!=0)
			out<<"el"<<i*100+j+1<<","<<"PX"<<","<<wi.wi[i].n[j].ft[0]+wi.wi[i].n[j].fn[0]<<endl;
			if(wi.wi[i].n[j].ft[1]+wi.wi[i].n[j].fn[1]!=0)
			out<<"el"<<i*100+j+1<<","<<"PY"<<","<<wi.wi[i].n[j].ft[1]+wi.wi[i].n[j].fn[1]<<endl;
			if(wi.wi[i].n[j].ft[2]+wi.wi[i].n[j].fn[2]!=0)
			out<<"el"<<i*100+j+1<<","<<"PZ"<<","<<wi.wi[i].n[j].ft[2]+wi.wi[i].n[j].fn[2]<<endl;
		};
	};
	for(i=0;i<wi.NOWI;i++){
		out<<"*Node Print, nset=wire"<<i<<",frequency=1000"<<endl;
		out<<"U"<<endl;
	};
	out<<"*END STEP\n";
	out.close();
	wi.time=fl.time;
};
```

E.2 考慮金線對塑料流動影響

E.2.1 使用者介面程式碼

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include "wire.cpp"
int main(){
        wires wi;
        wi.getwire("wire.txt", 0.1, 0.025,8,4);//read wire informations for prepared file
        ABAQUS_MODEL_GEN(wi,"wiresweep.inp","property.txt"); //output wire FEM mesh data to ABAQUS input file
                                                              //read wire property from prepared file and output to input file
        wi.hypermeshcommend("wire.cmf"); //output Hypermesh command file to build wire surface grid
        wi.wireelement("cavity.udm","wire.udm"); //handle flow grid configuration.
        //read Moldflow calculated pressure, velocity and viscosity at one time, calculate drag force, and then output to inp file.
        wi.dforce("pressure.nod.001","velocity.nod.001","viscosity.nod.001","wiresweep.inp");
        wi.dforce("pressure.nod.002","velocity.nod.002","viscosity.nod.002","wiresweep.inp");
        wi.dforce("pressure.nod.003","velocity.nod.003","viscosity.nod.003","wiresweep.inp");
        wi.dforce("pressure.nod.004","velocity.nod.004","viscosity.nod.004","wiresweep.inp");
        wi.dforce("pressure.nod.005","velocity.nod.005","viscosity.nod.005","wiresweep.inp");
        wi.dforce("pressure.nod.006","velocity.nod.006","viscosity.nod.006","wiresweep.inp");
        wi.dforce("pressure.nod.007","velocity.nod.007","viscosity.nod.007","wiresweep.inp");
        wi.dforce("pressure.nod.008","velocity.nod.008","viscosity.nod.008","wiresweep.inp");
        wi.dforce("pressure.nod.009","velocity.nod.009","viscosity.nod.009","wiresweep.inp");
        wi.dforce("pressure.nod.010","velocity.nod.010","viscosity.nod.010","wiresweep.inp");
        wi.dforce("pressure.nod.011","velocity.nod.011","viscosity.nod.011","wiresweep.inp");
        wi.dforce("pressure.nod.012","velocity.nod.012","viscosity.nod.012","wiresweep.inp");
        wi.dforce("pressure.nod.013","velocity.nod.013","viscosity.nod.013","wiresweep.inp");
        wi.dforce("pressure.nod.014","velocity.nod.014","viscosity.nod.014","wiresweep.inp");
        wi.dforce("pressure.nod.015","velocity.nod.015","viscosity.nod.015","wiresweep.inp");
        wi.dforce("pressure.nod.016","velocity.nod.016","viscosity.nod.016","wiresweep.inp");
        wi.dforce("pressure.nod.017","velocity.nod.017","viscosity.nod.017","wiresweep.inp");
        wi.dforce("pressure.nod.018","velocity.nod.018","viscosity.nod.018","wiresweep.inp");
        wi.dforce("pressure.nod.019","velocity.nod.019","viscosity.nod.019","wiresweep.inp");
        wi.dforce("pressure.nod.020","velocity.nod.020","viscosity.nod.020","wiresweep.inp");
        return 0;
};
```

E.3 主程式碼

```
#include <math.h>
#include "matrix.cpp"

// check if two characters are the same
int check(char *a, char *b){
        int j=0, n=0;
        int i;
        while(int(a[n])!=0){
                n++;
        };
```

```
        for(i=0;i<n;i++){
                if(a[i]==b[i]){
                        j++;
                };
        };
        if (j==n){
                return 1; //if char a = char b, return 1 (true)
        };
        return 0;
};

//calculate the volume of tetrahedral element according four nodes
double volume(double *n1, double *n2, double *n3, double *n4){
        double v1[3],v2[3],v3[3];//the vectors of the three edges of tetrahedral element
        double vol; //volume of tetrahedral
        int i;
        //calculate the vectors of the three edges of tetrahedral element
        for(i=0;i<3;i++){
                v1[i]=(n2[i]-n1[i]);
                v2[i]=(n3[i]-n1[i]);
                v3[i]=(n4[i]-n1[i]);
        };
        vol=0;
        for(i=0;i<3;i++){
                vol=v1[i]*(v2[int(fmod(i+1,3))]*v3[int(fmod(i+2,3))]-v2[int(fmod(i+2,3))]*v3[int(fmod(i+1,3))])+vol;
        };
        vol=sqrt(pow(vol,2));
        return vol;
};

// one class of flow field grid node
class fnode
{
        public:
                int num;
                double x[3];//the coord. of node in globe system
                double vel[3]; // the velocity in three coord. of this node
                fnode(){
                        vel[0]=0;
                        vel[1]=0;
                        vel[2]=0;
                };
};

//the class of node on the interface between flow and wire
class snode
{
        public:
                int num;
                double x[3]; //the coord. of node in globe system
                int neb[50]; //the nebor point coordinates used to determine shear force;
                double vel[3]; // the velocity in three coord. of this node
                double vis; // the viscosity of this node.
                double p; //the hydro-dynamic pressure.
                double shear; // the shear force
                int readdata(fstream &);
};
```

```
// the class of node of wire FEM beam element
class node
{
        public:
                double x[3]; //the coord. of this node
                double F[3]; //the drag force at this node.
                double area; //the area of grid on the interface between
                                //flow and wire.
                node(){
                        area=0;
                        F[0]=0;
                        F[1]=0;
                        F[2]=0;
                };
};

// the class of each wire
class wire
{
        public:
                char null;
                int nu; // the number of wire
                int NOWN;
                int stp; // the end ignored node at wire mesh when buliding flow grid.
                double p1[3],p2[3]; //the first and second bond point
                double l,H; //l: the distance between the highest point of wire and p1.
                                //H: the height of the highest point of wire
                node *n;
                int cal(double); // the function which calculates wire curve
                int read(fstream &); // the function which read wire information data file
                double func1(double x){ //wire curve math function 1
                        double y;
                        y=-2*H*pow(x,3)/pow(l,3)+3*H*pow(x,2)/pow(l,2);
                        return y;
                };
                double pfunc1(double x){ //the partial of function 1
                        double y;
                        y=-6*H*pow(x,2)/pow(l,3)+6*H*pow(x,1)/pow(l,2);
                        return y;
                };
                double func2(double x){//wire curve math function 2
                        double y;
                        double L;
                        L=sqrt(pow((p2[0]-p1[0]),2)+pow((p2[1]-p1[1]),2));
                        if(x==l)
                                y=H;
                        else if(x>L)
                                y=p2[2]-p1[2];
                        else
                                y=(p2[2]-p1[2])+(H-(p2[2]-p1[2]))*sqrt(1-pow((x-l)/(L-l),2));
                        return y;
                };
                double pfunc2(double x){//the partial of function 2
                        double y,yp,xp;
                        double slop;
                        xp=x*(1+1e-5);
                        y=func2(x);
                        yp=func2(xp);
```

```
                slop=(yp-y)/(xp-x);
                return slop;
        };
        ~wire(){
                delete [] n;
        };
};

class selement
{
        public:
                int num;
                int wirenu;
                int welement;
                int node[4];
                double normal[3],binormal[3],tangent[3];
                double **shape;
                double integratepoint[3];
                double norpoint[3],tangpoint[3],bipoint[3];
                int norelem[3];
                double area;
                double dforce[3];
                int readdata(fstream &);
                int calnormal(snode *);
                int findnorelement(snode *,fnode *);
                int findbelongelement(wire &,int);
                int caldforce(snode *,fnode *);
                selement(){
                        shape=new double *[4];
                        for(int i=0;i<4;i++)
                                shape[i]=new double [4];
                };
                ~selement(){
                        for(int i=0;i<4;i++)
                                delete [] shape[i];
                        delete [] shape;
                };
};

class wires
{
        public:
                wire *wi;
                snode *sn;
                fnode *fn;
                selement *selem;
                int NOWI;
                double radius;
                int cirsep;
                int dn;
                double time;
                int getwire(char *, double, double,int,int);
                int wireelement(char *, char *);
                int hypermeshcommend(char *);
                int dforce(char *,char *,char *,char *);
                int readvelocity(char *, double &);
                int readviscosity(char *, double &);
                int readpressure(char *, double &);
```

```
                int readshear(char *, double &);
                int ABAQUS_MODEL_GEN(char *,char *);
                int ABAQUS_STEP_GEN(char *,double);
                wires(){
                        dn=150;// "dn" means the density of element on wire;
                        cirsep=8;
                        time=0;
                };
                ~wires(){
                        delete [] wi;
                        delete [] sn;
                        delete [] fn;
                };
};

int snode::readdata(fstream &input) {
        int tmp;
        double ftmp;
        int i;
        char command[200];
        input>>command>>num>>tmp>>tmp>>tmp>>ftmp;
        input>>x[0]>>x[1]>>x[2];
        input>>command;
        return 0;
};

int selement::readdata(fstream &input){
        int tmp;
        double ftmp;
        int i;
        char command[200];
        input>>command>>num>>tmp>>tmp>>tmp>>command>>tmp>>tmp;
        input>>node[0]>>node[1]>>node[2];
        for(i=0;i<3;i++)
                node[i]=node[i]-1;
        input>>command;
        return 0;
};

int selement::findnorelement(snode *n, fnode *fn){
        double rp,c;
        double cl[3]={1,1,1};
        double **a;
        int i,j,k;
        double T;
        a=new double *[4];
        for(i=0;i<4;i++)
                a[i]=new double [4];
        T=pow(normal[0],2)+pow(normal[1],2)+pow(normal[2],2);
        for(i=0;i<3;i++)
                norelem[i]=-1;
        for(j=1;j<50;j++){
                for(i=0;i<4;i++){
                        rp=0;
                        c=0;
                        for(k=0;k<3;k++){
                                rp=normal[k]*(fn[n[node[i]].neb[j]].x[k]-n[node[0]].x[k])+rp;
                                c=pow((integratepoint[k]-fn[n[node[i]].neb[j]].x[k]),2)+c;
```

```
                        };
                        c=sqrt(c);
                        if(rp>T){
                                if(c<cl[0]){
                                        cl[2]=cl[1];
                                        cl[1]=cl[0];
                                        cl[0]=c;
                                        norelem[2]=norelem[1];
                                        norelem[1]=norelem[0];
                                        norelem[0]=n[node[i]].neb[j];
                                }
                                else if(c>cl[0]&&c<cl[1]){
                                        cl[2]=cl[1];
                                        cl[1]=c;
                                        norelem[2]=norelem[1];
                                        norelem[1]=n[node[i]].neb[j];
                                }
                                else if(c>cl[1]&&c<cl[2]){
                                        cl[2]=c;
                                        norelem[2]=n[node[i]].neb[j];
                                }
                                else{
                                        continue;
                                };
                        };
                };
        };
        if(norelem[0]==-1||norelem[1]==-1||norelem[0]==-1)
                return 0;
        for(i=0;i<3;i++){
                for(j=0;j<3;j++){
                        a[i][j]=fn[norelem[i]].x[j];
                };
        };
        for(i=0;i<3;i++)
                a[3][i]=integratepoint[i];
        for(i=0;i<4;i++)
                a[i][3]=1;
        INVERSE(a,shape,4,4);
        return 1;
};

int selement::findbelongelement(wire &wi, int wino){
        int i,j;
        double v1[3],v2[3];
        double dot,null;
        for(i=wi.stp;i<wi.NOWN-1;i++){
                dot=0;
                null=0;
                for(j=0;j<3;j++){
                        v1[j]=wi.n[i+1].x[j]-wi.n[i].x[j];
                        v2[j]=integratepoint[j]*1000-wi.n[i].x[j];
                };
                for(j=0;j<3;j++){
                        null=v1[j]*v1[j]+null;
                };
                null=sqrt(null);
                for(j=0;j<3;j++){
```

```
                        dot=v1[j]*v2[j]/null+dot;
                };
                if(dot>=0&&dot<null){
                        wirenu=wino;
                        welement=i;
                        return 0;
                };
        };
        return 1;
};

int selement::calnormal(snode *n){
        double v1[3],v2[3];
        int i,j;
        double distance1=0;
        double distance2=0;
        double T=0;
        for(i=0;i<3;i++){
                v1[i]=n[node[1]].x[i]-n[node[0]].x[i];
                v2[i]=n[node[3]].x[i]-n[node[0]].x[i];
        };
        area=(pow(v1[0],2)+pow(v1[1],2)+pow(v1[2],2))*(pow(v2[0],2)+pow(v2[1],2)+pow(v2[2],2));
        area=area-pow(v1[0]*v2[0]+v1[1]*v2[1]+v1[2]*v2[2],2);
        area=sqrt(area);
        T=(sqrt(pow(v1[0],2)+pow(v1[1],2)+pow(v1[2],2))+sqrt(pow(v2[0],2)+pow(v2[1],2)+pow(v2[2],2)))/10;
        normal[0]=v1[1]*v2[2]-v1[2]*v2[1];
        normal[1]=v1[2]*v2[0]-v1[0]*v2[2];
        normal[2]=v1[0]*v2[1]-v1[1]*v2[0];
        for(i=0;i<3;i++){
                distance1=distance1+pow(normal[i],2);
                distance2=distance2+pow(v1[i],2);
        };
        distance1=sqrt(distance1);
        distance2=sqrt(distance2);
        for(i=0;i<3;i++){
                normal[i]=normal[i]/distance1;
                tangent[i]=v1[i]/distance2;
        };
        binormal[0]=normal[1]*tangent[2]-normal[2]*tangent[1];
        binormal[1]=normal[2]*tangent[0]-normal[0]*tangent[2];
        binormal[2]=normal[0]*tangent[1]-normal[1]*tangent[0];
        for(i=0;i<3;i++){
                normal[i]=normal[i]*T;
                tangent[i]=tangent[i]*T;
                binormal[i]=binormal[i]*T;
        };
        // determind first order gauss integrate point
        integratepoint[0]=(n[node[0]].x[0]+n[node[1]].x[0]+n[node[2]].x[0]+n[node[3]].x[0])/4;
        integratepoint[1]=(n[node[0]].x[1]+n[node[1]].x[1]+n[node[2]].x[1]+n[node[3]].x[1])/4;
        integratepoint[2]=(n[node[0]].x[2]+n[node[1]].x[2]+n[node[2]].x[2]+n[node[3]].x[2])/4;
        for(i=0;i<3;i++){
                norpoint[i]=integratepoint[i]+normal[i];
                tangpoint[i]=norpoint[i]+tangent[i];
                bipoint[i]=norpoint[i]+binormal[i];
        };
        return 0;
};
```

```
int selement::caldforce(snode *n,fnode *fn){
        int i,j;
        double T;
        double dot1,dot2;
        double vn[3],vt[3],vb[3];
        double **xn,**xt,**xb,**v,**tmp,**tmp1;
        double vis=0,p=0,tshear=0,bshear=0;
        double pressure[3],sf[3];
        xn=new double *[1];xn[0]=new double [4];
        xt=new double *[1];xt[0]=new double [4];
        xb=new double *[1];xb[0]=new double [4];
        v=new double *[4];for(i=0;i<4;i++) v[i]=new double [1];
        tmp=new double *[4];for(i=0;i<4;i++) tmp[i]=new double [1];
        tmp1=new double *[1];tmp1[0]=new double [1];
        T=sqrt(pow(normal[0],2)+pow(normal[1],2)+pow(normal[2],2));
        for(i=0;i<3;i++){
                xn[0][i]=norpoint[i];xt[0][i]=tangpoint[i];xb[0][i]=bipoint[i];
        };
        xn[0][3]=1;xt[0][3]=1;xb[0][3]=1;
        for(i=0;i<3;i++){ //for three coordinates
                for(j=0;j<3;j++){ //for three normal point
                        v[j][0]=fn[norelem[j]].vel[i];
                };
                v[3][0]=0;
                MULTI(shape,v,tmp,4,4,1);
                MULTI(xn,tmp,tmp1,1,4,1);
                vn[i]=tmp1[0][0];
                MULTI(xt,tmp,tmp1,1,4,1);
                vt[i]=tmp1[0][0];
                MULTI(xb,tmp,tmp1,1,4,1);
                vb[i]=tmp1[0][0];
        };
        dot1=0;dot2=0;
        for(i=0;i<3;i++){
                dot1=dot1+tangent[i]/T*vn[i];
                dot2=dot2+normal[i]/T*(vt[i]-vn[i]);
        };
        tshear=(dot1+dot2)/T;
        dot1=0;dot2=0;
        for(i=0;i<3;i++){
                dot1=dot1+binormal[i]/T*vn[i];
                dot2=dot2+normal[i]/T*(vb[i]-vn[i]);
        };
        bshear=(dot1+dot2)/T;
        for(i=0;i<4;i++){
                vis=n[node[i]].vis/3+vis;
                p=n[node[i]].p/3+p;
        };
        dot1=0;
        for(i=0;i<3;i++)
                dot1=dot1+normal[i]/T*vn[i];
        for(i=0;i<3;i++){
                pressure[i]=(-p+dot1/T)*normal[i]/T;
                sf[i]=vis*(tshear*tangent[i]+bshear*binormal[i])/T;
                dforce[i]=(sf[i]+pressure[i])*area;
        };
        for(i=0;i<4;i++){
                delete [] v[i];
```

```
                delete [] tmp[i];
        };
        delete [] xn[0];
        delete [] xt[0];
        delete [] xb[0];
        delete [] tmp1[0];
        delete [] xn;
        delete [] xt;
        delete [] xb;
        delete [] v;
        delete [] tmp;
        delete [] tmp1;
        return 0;
};

int wire::read(fstream &input ){
        input>>nu>>null;
        input>>null>>p1[0]>>null>>p1[1]>>null>>p1[2]>>null>>null;
        input>>null>>p2[0]>>null>>p2[1]>>null>>p2[2]>>null>>null;
        input>>H>>null>>l;
        return 0;
};

int wire::cal(double esize){
        double x,y,xp,yp,slop;
        int i,highest=0;
        double tmp;
        fstream pout,inp;
        pout.open("tmp",ios::out|ios::trunc);
        NOWN=1;
        x=sqrt(pow((p2[0]-p1[0]),2)+pow((p2[1]-p1[1]),2));
        y=p2[2]-p1[2];
        xp=0;
        pout<<0<<" "<<0<<endl;
        while(1){
                slop=pfunc1(xp);
                tmp=esize/sqrt(pow(slop,2)+1);
                slop=pfunc1(xp+tmp/2);
                tmp=esize/sqrt(pow(slop,2)+1);
                slop=pfunc1(xp+tmp/2);
                tmp=esize/sqrt(pow(slop,2)+1);
                xp=xp+tmp;
                if(xp>l){
                        xp=xp-tmp;
                        break;
                };
                yp=func1(xp);
                pout<<xp<<" "<<yp<<endl;
                NOWN++;
        };
        xp=l;
        yp=func1(xp);
        pout<<xp<<" "<<yp<<endl;
        NOWN++;
        while(1){
                slop=pfunc2(xp);
                tmp=esize/sqrt(pow(slop,2)+1);
                slop=pfunc2(xp+tmp/2);
```

```
                tmp=esize/sqrt(pow(slop,2)+1);
                slop=pfunc2(xp+tmp/2);
                tmp=esize/sqrt(pow(slop,2)+1);
                xp=xp+tmp;
                if(xp>x){
                        xp=xp-tmp;
                        break;
                };
                yp=func2(xp);
                pout<<xp<<" "<<yp<<endl;
                NOWN++;
        };
        xp=x;
        yp=func2(xp);
        pout<<xp<<" "<<yp<<endl;
        NOWN++;
        pout.close();
        n = new node[NOWN];
        inp.open("tmp",ios::in);
        for(i=0;i<NOWN;i++){
                inp>>xp>>yp;
                n[i].x[0]=xp*(p2[0]-p1[0])/x+p1[0];
                n[i].x[1]=xp*(p2[1]-p1[1])/x+p1[1];
                n[i].x[2]=yp+p1[2];
        };
        inp.close();
        return 0;
};

int wires::getwire(char *ifile, double esize, double radi, int cir, int s){ //"NOWN" is the number of nodes of each wire
        radius=radi;
        cirsep=cir;
        char null;
        float tmp;
        int wn=0; //the number of wires.
        int i,j;
        fstream input;
        //check the number of wire from "wire.txt".
        input.open(ifile,ios::in);
        do{
                input>>tmp>>null;
                if(null!=':'){
                        cout<<"Wire input file error!!\n"<<endl;;
                        exit(0);
                };
                input>>null>>tmp>>null>>tmp>>null>>tmp>>null>>null;
                if(null!=','){
                        cout<<"Wire input file error!!\n"<<endl;
                        exit(0);
                };
                input>>null>>tmp>>null>>tmp>>null>>tmp>>null>>null;
                if(null!=','){
                        cout<<"Wire input file error!!\n"<<endl;
                        exit(0);
                };
                input>>tmp>>null;
                if(null!=','){
                        cout<<"Wire input file error!!\n"<<endl;
```

```
                        exit(0);
                };
                input>>tmp;
                wn++;
                input>>null;
                if(null=='*'){
                        break;
                };
                input.seekg(-1,ios::cur);
        }while(!input.eof());
        input.close();
        NOWI=wn;
        cout<<"The number of wires = "<<NOWI<<endl;
        wi=new wire[wn];
        for(i=0;i<NOWI;i++)
                wi[i].stp=s;
        // Reading each wire data
        input.open(ifile,ios::in);
        for(i=0;i<NOWI;i++){
                wi[i].read(input);
        };
        input.close();
        // Calculating the coordinates of nodes of each wire.
        for(i=0;i<NOWI;i++){
                wi[i].cal(esize);
        };
        cout<<" Wire loop calculation was completed!!"<<endl;
        return 0;
};

int wires::wireelement(char *ifile1, char *ifile2){
        char command[200];
        int i,j,k,l,tmp,num,num1, nono[4];
        int m;
        double x[3];
        double ftmp;
        double *rp;
        int *smallest;
        cirsep=8;
        fstream inp,pout;
        inp.open(ifile2,ios::in);
        // cheak the number of node and element on the wire surface
        sn=new snode[NOWI*cirsep*(dn+1)];
        selem=new selement[NOWI*cirsep*dn];
        //read data of the grid on the surface of wire
        while(!inp.eof()){
                tmp=inp.tellg();
                inp.getline(command,200);
                if(check("NODE{",command)){
                        inp.seekg(tmp,ios::beg);
                        for(i=0;i<NOWI*cirsep*(dn+1);i++){
                                sn[i].readdata(inp);
                        };
                        break;
                };
        };
        for(i=0;i<NOWI;i++){
                for(j=0;j<dn;j++){
```

```
                        for(k=0;k<cirsep-1;k++){
                                selem[i*cirsep*dn+j*cirsep+k].node[0]=i*cirsep*(dn+1)+j*cirsep+k;
                                selem[i*cirsep*dn+j*cirsep+k].node[1]=i*cirsep*(dn+1)+j*cirsep+k+1;
                                selem[i*cirsep*dn+j*cirsep+k].node[2]=i*cirsep*(dn+1)+j*cirsep+k+cirsep+1;
                                selem[i*cirsep*dn+j*cirsep+k].node[3]=i*cirsep*(dn+1)+j*cirsep+k+cirsep;
                        };
                        selem[i*cirsep*dn+(j+1)*cirsep-1].node[0]=i*cirsep*(dn+1)+(j+1)*cirsep-1;
                        selem[i*cirsep*dn+(j+1)*cirsep-1].node[1]=i*cirsep*(dn+1)+j*cirsep;
                        selem[i*cirsep*dn+(j+1)*cirsep-1].node[2]=i*cirsep*(dn+1)+(j+1)*cirsep;
                        selem[i*cirsep*dn+(j+1)*cirsep-1].node[3]=i*cirsep*(dn+1)+(j+2)*cirsep-1;
                };
};
inp.close();
cout<<"The data of nodes and elements on the interface between wire and fluid reading was compeletes!!\n";
//adject the node number form hypermesh to moldflow udm file
inp.open(ifile1,ios::in);
while(!inp.eof()){
                inp.getline(command,200);
                if(check("SUMR{\0",command)==1){
                        while(!inp.eof()){
                                tmp=inp.tellg();
                                inp>>command;
                                if(check("NOND{",command)==1){
                                        inp.seekg(tmp+8,ios::beg);
                                        inp>>num;
                                        break;
                                };
                        };
                        break;
                };
};
cout<<"The number of nodes in the cavity = "<<num<<endl;
fn = new fnode[num];
rp = new double[num];
smallest=new int[num];
while(!inp.eof()){
                tmp=inp.tellg();
                inp.getline(command,200);
                if(check("NODE{",command)){
                        inp.seekg(tmp,ios::beg);
                        break;
                };
};
for(i=0;i<num;i++){
                inp.get(command,6);
                inp>>fn[i].num>>tmp>>tmp>>tmp>>ftmp;
                inp>>fn[i].x[0]>>fn[i].x[1]>>fn[i].x[2];
                inp.getline(command,2);
};
inp.close();
cout<<"The data of nodes in the cavity reading was compeletes!!\n";
                // serach the top 50 nearest node to each wire surface node
                // the nearest node will be map to that wire surface node
cout<<"Searching 50 nearest node to each interface nodes....."<<endl;
pout.open("step.txt",ios::out);
for(j=0;j<NOWI*cirsep*(dn+1);j++){
                for(k=0;k<num;k++)
                        smallest[k]=k;
```

```
                for(k=0;k<num;k++){
                        rp[k]=0;
                        for(i=0;i<3;i++)
                                rp[k]=rp[k]+pow(sn[j].x[i]-fn[k].x[i],2);
                        rp[k]=sqrt(rp[k]);
                };
                for(k=0;k<50;k++){
                        for(i=0;i<num-k-1;i++){
                                if(rp[smallest[i]]<rp[smallest[i+1]]){
                                        tmp=smallest[i];
                                        smallest[i]=smallest[i+1];
                                        smallest[i+1]=tmp;
                                };
                        };
                        sn[j].neb[k]=smallest[num-k-1];
                        pout<<sn[j].neb[k]<<" ";
                };
                pout<<endl;
                cout<<"Node "<<j<<" completed!!"<<endl;
                sn[j].num=fn[sn[j].neb[0]].num;
                for(k=0;k<3;k++){
                        sn[j].x[k]=fn[sn[j].neb[0]].x[k];
                };
        };
        cout<<"Calulating the normal of each elements on the interfaces"<<endl;
        for(i=0;i<NOWI*cirsep*dn;i++){
                selem[i].calnormal(sn);
        };
        // classify the wire surface element accrodine to wire FEM element
        cout<<"Projecting the elements on the interfaces to the elements of wire beam elements!!"<<endl;
        for(i=0;i<NOWI;i++){
                for(j=0;j<cirsep*dn;j++){
                        tmp=selem[i*cirsep*dn+j].findbelongelement(wi[i], i);
                        if(tmp==1){
                                selem[i*cirsep*dn+j].wirenu=i;
                                selem[i*cirsep*dn+j].welement=selem[i*cirsep*dn+j-1].welement;
                        };
                };
        };
        for(i=0;i<NOWI*cirsep*dn;i++){
                wi[selem[i].wirenu].n[selem[i].welement].area=selem[i].area+wi[selem[i].wirenu].n[selem[i].welement].area;
        };
        cout<<"Searching the tet4 elemets on the interfaces!!"<<endl;
        for(i=0;i<NOWI*cirsep*dn;i++){
                tmp=selem[i].findnorelement(sn,fn);
                if(tmp==0){
                        cerr<<"element "<<i+1<<" eorror!!"<<endl;
                        //exit(1);
                };
        };
        cout<<"Searching was completed!!"<<endl;
        delete [] rp;
        delete [] smallest;
        return 0;
};

int wires::readvelocity(char *ifile, double &time){
        int i,j;
```

```
        char null[200];
        int lab;
        int tmp;
        double vel[3];
        fstream inp;
        inp.open(ifile,ios::in);
        if(!inp){
                cerr<<"Can't open the velocity result file "<< ifile<<endl;
                exit(1);
        };
        while(!inp.eof()){
                tmp=inp.tellg();
                inp>>null;
                if(null[0]>='0' && null[0]<='9'){
                        inp.seekg(tmp,ios::beg);
                        inp>>time;
                        inp>>null;
                        break;
                };
        };
        inp.getline(null,200);
        inp.getline(null,200);
        inp.getline(null,200);
        while(!inp.eof()){
                inp>>lab>>vel[0]>>vel[1]>>vel[2];
                for(i=0;i<NOWI*cirsep*dn;i++){
                        for(int k=0;k<3;k++){
                                if(lab==fn[selem[i].norelem[k]].num){
                                        fn[selem[i].norelem[k]].vel[0]=vel[0];
                                        fn[selem[i].norelem[k]].vel[1]=vel[1];
                                        fn[selem[i].norelem[k]].vel[2]=vel[2];
                                        goto flag1;
                                };
                        };
                };
                for(i=0;i<NOWI*cirsep*(dn+1);i++){
                        if(lab==sn[i].num){
                                sn[i].vel[0]=vel[0];
                                sn[i].vel[1]=vel[1];
                                sn[i].vel[2]=vel[2];
                                break;
                        };
                };
flag1:
                inp.getline(null,200);
        };
        inp.close();
        return 0;
};

int wires::readviscosity(char *ifile, double &time){
        int i;
        char null[200];
        int lab;
        int tmp;
        double vis;
        fstream inp;
        inp.open(ifile,ios::in);
```

```
	if(!inp){
		cerr<<"Can't open the velocity result file "<< ifile<<endl;
		exit(1);
	};
	while(!inp.eof()){
		tmp=inp.tellg();
		inp>>null;
		if(null[0]>='0' && null[0]<='9'){
			inp.seekg(tmp,ios::beg);
			inp>>time;
			inp>>null;
			break;
		};
	};
	inp.getline(null,200);
	inp.getline(null,200);
	inp.getline(null,200);
	while(!inp.eof()){
		inp>>lab>>vis;
		for(i=0;i<NOWI*cirsep*(dn+1);i++){
			if(lab==sn[i].num){
				sn[i].vis=vis;
				break;
			};
		};
		inp.getline(null,200);
	};
	inp.close();
	return 0;
};

int wires::readpressure(char *ifile, double &time){
	int i,j;
	char null[200];
	int lab;
	int tmp;
	double p;
	fstream inp;
	inp.open(ifile,ios::in);
	if(!inp){
		cerr<<"Can't open the velocity result file "<< ifile<<endl;
		exit(1);
	};
	while(!inp.eof()){
		tmp=inp.tellg();
		inp>>null;
		if(null[0]>='0' && null[0]<='9'){
			inp.seekg(tmp,ios::beg);
			inp>>time;
			inp>>null;
			break;
		};
	};
	inp.getline(null,200);
	inp.getline(null,200);
	inp.getline(null,200);
	while(!inp.eof()){
		inp>>lab>>p;
```

```
                for(i=0;i<NOWI*cirsep*(dn+1);i++){
                        if(lab==sn[i].num){
                                sn[i].p=p;
                                break;
                        };
                };
                inp.getline(null,200);
        };
        inp.close();
        return 0;
};

int wires::readshear(char *ifile, double &time){
        int i;
        char null[200];
        int lab;
        int tmp;
        double shear;
        fstream inp;
        inp.open(ifile,ios::in);
        if(!inp){
                cerr<<"Can't open the velocity result file "<< ifile<<endl;
                exit(1);
        };
        while(!inp.eof()){
                tmp=inp.tellg();
                inp>>null;
                if(null[0]>='0' && null[0]<='9'){
                        inp.seekg(tmp,ios::beg);
                        inp>>time;
                        inp>>null;
                        break;
                };
        };
        inp.getline(null,200);
        inp.getline(null,200);
        inp.getline(null,200);
        while(!inp.eof()){
                inp>>lab>>shear;
                for(i=0;i<NOWI*cirsep*(dn+1);i++){
                        if(lab==sn[i].num){
                                sn[i].shear=shear;
                                break;
                        };
                };
                inp.getline(null,200);
        };
        inp.close();
        return 0;
};

int wires::dforce(char *ifile1,char *ifile2, char *ifile3,char *ofile){
        double time0,time1;
        int i,j,k;
        double ln,ln1;
        cout<<"Reading pressure data from "<<ifile1<<"...!!"<<endl;
        readpressure(ifile1,time0);
        cout<<"Reading velocity data from "<<ifile2<<"...!!"<<endl;
```

```
        readvelocity(ifile2,time0);
        cout<<"Reading viscosity data from "<<ifile3<<"...!!"<<endl;
        readviscosity(ifile3,time0);
        cout<<" Data reading was completed!!"<<endl;
        time1=time0-time;
        cout<<" Calculating Drag force ....."<<endl;
        for(i=0;i<NOWI;i++){
                for(j=0;j<wi[i].NOWN;j++){
                        wi[i].n[j].F[0]=0;
                        wi[i].n[j].F[1]=0;
                        wi[i].n[j].F[2]=0;
                };
        };
        for(i=0;i<NOWI*cirsep*dn;i++){
                selem[i].caldforce(sn,fn);
                for(j=0;j<3;j++)

        wi[selem[i].wirenu].n[selem[i].welement].F[j]=selem[i].dforce[j]+wi[selem[i].wirenu].n[selem[i].welement].F[j];
        };
        for(i=0;i<NOWI;i++){
                for(j=wi[i].stp;j<wi[i].NOWN-1;j++){
                        ln=0;
                        for(k=0;k<3;k++)
                                ln=ln+pow(wi[i].n[j].x[k]-wi[i].n[j+1].x[k],2);
                        ln=sqrt(ln);
                        for(k=0;k<3;k++)
                                wi[i].n[j].F[k]=wi[i].n[j].F[k]/ln;
                };
                for(j=0;j<wi[i].stp;j++){
                        ln=0;
                        ln1=0;
                        for(k=0;k<3;k++){
                                ln=ln+pow(wi[i].p1[k]-wi[i].n[wi[i].stp].x[k],2);
                                ln1=ln1+pow((wi[i].n[j].x[k]+wi[i].n[j+1].x[k])/2-wi[i].n[wi[i].stp].x[k],2);
                        };
                        ln=sqrt(ln);
                        ln1=sqrt(ln1);
                        for(k=0;k<3;k++){
                                wi[i].n[j].F[k]=wi[i].n[wi[i].stp].F[k]/ln*(ln-ln1);
                        };
                };
        };
        cout<<" STEP output.."<<endl;
        ABAQUS_STEP_GEN(ofile, time1);
        time=time0;
        return 0;
};

int wires::ABAQUS_MODEL_GEN(char *ifile, char *pfile){
        int i,j,k;
        char null[200];
        int lab;
        fstream out;
        fstream in;
        out.open(ifile,ios::out|ios::trunc);
        in.open(pfile,ios::in);
        out<<"*HEADING"<<endl;
        //NODE;
```

```
	out<<"*NODE"<<endl;
	for(i=0;i<NOWI;i++){
		for(j=0;j<wi[i].NOWN;j++){
			lab=i*100+j+1;
			out<<lab;
			for(k=0;k<3;k++)
				out<<","<<wi[i].n[j].x[k];
			out<<endl;
		};
	};
	//ELEMENT
	out<<"*ELEMENT, ELSET=WIRE, TYPE=B31"<<endl;
	for(i=0;i<NOWI;i++){
		for(j=0;j<wi[i].NOWN-1;j++){
			lab=i*100+j+1;
			out<<lab;
			out<<","<<i*100+j+1<<","<<i*100+j+2<<endl;
		};
	};
	//NODE SET
	for(i=0;i<NOWI;i++){
		out<<"*NSET, nset=wire"<<i<<", generate"<<endl;
		out<<i*100+1<<","<<i*100+wi[i].NOWN<<",1"<<endl;
	};
	//ELEMENT SET
	for(i=0;i<NOWI;i++){
		for(j=0;j<wi[i].NOWN-1;j++){
			lab=i*100+j+1;
			out<<"*Elset,\telset=el"<<lab<<endl;
			out<<lab<<endl;
		};
	};
	// Material
	out<<"*MATERIAL, NAME=gold-wire";
	while(!in.eof()){
		out<<endl;
		in.getline(null,200);
		out<<null;
	};
	//SECTION
	out<<"*BEAM SECTION, SECTION=CIRC, Elset=WIRE, Material=gold-wire\n";
	out<<radius<<endl;
	//Initial Boundary Conditions.
	//Fixed at wire ends.
	out<<"*BOUNDARY\n";
	for(i=0;i<NOWI;i++){
		out<<i*100+1<<", 1, 6\n";
		out<<i*100+wi[i].NOWN<<", 1, 6\n";
	};
	out.close();
	in.close();
	return 0;
};

int wires::ABAQUS_STEP_GEN(char *ifile,double time){
	int i,j;
	fstream out;
	out.open(ifile,ios::out|ios::app);
```

```
		out<<"*Step,		Amplitude=RAMP\n";
		out<<"*Static\n";
		out<<time/5<<","<<time<<","<<time/100000<<","<<time/5<<endl;
		out<<"*DLOAD\n";
		for(i=0;i<NOWI;i++){
			for(j=0;j<wi[i].stp;j++){
				if(wi[i].n[j].F[0]!=0)
				out<<"el"<<i*100+j+1<<","<<"PX"<<","<<wi[i].n[j].F[0]<<endl;
				if(wi[i].n[j].F[1]!=0)
				out<<"el"<<i*100+j+1<<","<<"PY"<<","<<wi[i].n[j].F[1]<<endl;
				if(wi[i].n[j].F[2]!=0)
				out<<"el"<<i*100+j+1<<","<<"PZ"<<","<<wi[i].n[j].F[2]<<endl;
			};
			for(j=wi[i].stp;j<wi[i].NOWN-1;j++){
				if(wi[i].n[j].area!=0){
					if(wi[i].n[j].F[0]!=0)
					out<<"el"<<i*100+j+1<<","<<"PX"<<","<<wi[i].n[j].F[0]<<endl;
					if(wi[i].n[j].F[1]!=0)
					out<<"el"<<i*100+j+1<<","<<"PY"<<","<<wi[i].n[j].F[1]<<endl;
					if(wi[i].n[j].F[2]!=0)
					out<<"el"<<i*100+j+1<<","<<"PZ"<<","<<wi[i].n[j].F[2]<<endl;
				};
			};
		};
		for(i=0;i<NOWI;i++){
			out<<"*Node Print, nset=wire"<<i<<",frequency=1000"<<endl;
			out<<"U"<<endl;
		};
		out<<"*END STEP\n";
		out.close();
		return 0;
};

int wires::hypermeshcommend(char *ofile){
		int i,j,l,n;
		int line,node;
		int plane1,plane2;
		double tmp;
		int system=0;
		float angel=0;
		double p1[3],p2[3],p[3];
		double x;
		double theta;
		fstream pout;
		pout.open(ofile,ios::out);
		line=0;
		node=0;
		pout<<"*collectorcreate(components,\"wires\",\"wires\",14)\n";
		pout<<"*currentcollector(componenet,\"wires\")\n";
		for(i=0;i<NOWI;i++){
			l=node;
// create curves of each wire;
			for(j=wi[i].NOWN-1;j>=wi[i].stp;j--){
				node++;
				pout<<"*createnode(";
				pout<<wi[i].n[j].x[0]<<",";
				pout<<wi[i].n[j].x[1]<<",";
				pout<<wi[i].n[j].x[2]<<",";
```

```
                        pout<<0<<",0,0)\n";
                        p[0]=wi[i].n[j].x[0];
                        p[1]=wi[i].n[j].x[1];
                        p[2]=wi[i].n[j].x[2];
                        n=j;
                        if(wi[i].n[j].x[2]<(wi[i].p1[2]+radius)){
                                break;
                        };
                };
                pout<<"*createlist(nodes,"<<1<<")";
                for(j=l+1;j<=node;j++){
                        pout<<" "<<j;
                };
                pout<<endl;
                pout<<"*linecreatefromnodes(1,2,0,0,180)"<<endl;
                line++;
//
                p1[0]=wi[i].p1[0];
                p1[1]=wi[i].p1[1];
                p1[2]=wi[i].p1[2];
                p2[0]=wi[i].p2[0];
                p2[1]=wi[i].p2[1];
                p2[2]=wi[i].p2[2];
                x=sqrt(pow((p2[0]-p1[0]),2)+pow((p2[1]-p1[1]),2));
// calculate the angle of wire with aspect to goble coordinary system.
                tmp=(p1[0]-p2[0])*(p1[1]-p2[1]);
                if(tmp>0)
                        theta=asin((p2[1]-p1[1])/x);
                else if(tmp<0){
                        if(p2[0]-p1[0]<0)
                                theta=acos((p2[0]-p1[0])/x);
                        else
                                theta=acos(-1)*2-acos((p2[0]-p1[0])/x);
                }
                else
                        theta=acos(p2[0]-p1[0]);
// create the cross section edge of wire at the first bonding point
                pout<<"*createnode(";
                pout<<p[0]-radius*sin(theta)<<","<<p[1]+radius*cos(theta)<<","<<p[2]<<","<<0<<",0,0)\n";
                node++;
                pout<<"*systemsize(1)\n";
                pout<<"*systemcreate3nodes(0,"<<node-1<<",\"z-axis\","<<node-2<<"\",yz-plane\","<<node<<")"<<endl;
                system++;
                pout<<"*createplane(1,";
                pout<<wi[i].n[n+1].x[0]-p[0]<<",";
                pout<<wi[i].n[n+1].x[1]-p[1]<<",";
                pout<<wi[i].n[n+1].x[2]-p[2]<<",";
                pout<<p[0]<<","<<p[1]<<","<<p[2]<<")\n";
                l=line;
                pout<<"*createnode(";
                pout<<radius<<","<<0<<","<<0<<","<<system<<",0,0)\n";
                node++;
                pout<<"*createlist(nodes,"<<1<<")"<<node<<endl;
                pout<<"*createcirclefrompointplane(1,1,"<<360<<","<<angel<<")\n";
                line++;
// Trim lines
                pout<<"*createplane(1,0,0,1,"<<p2[0]<<","<<p2[1]<<","<<p2[2]<<")\n";
                pout<<"*createnode(";
```

```
		pout<<p2[0]+radius*cos(theta)<<",";
		pout<<p2[1]+radius*sin(theta)<<",";
		pout<<p2[2]<<","<<"0,0,0)\n";
		node++;
		pout<<"*createlist(nodes,"<<1<<")"<<node<<endl;
		pout<<"*createcirclefrompointplane(1,1,"<<360<<","<<angel<<")\n";
		line++;
};
for(i=0;i<NOWI;i++){
		// drag wire surfaces
		pout<<"*surfacemode(3)\n";
		pout<<"*createlist(lines,1)";
		pout<<" "<<i*3+2;
		pout<<endl;
		pout<<"*createlist(nodes,1)"<<endl;
		pout<<"*createlist(lines,2)"<<1+i*3<<endl;
		pout<<"*createlist(nodes,2)"<<endl;
		pout<<"*createplane(1,1,0,0,0,0,0)\n";
		pout<<"*linedraglinetoformsurface(1,1,2,2,1,1)\n";
		// wire surface mesh
			//setedgemeshparams(edge index,element density,bias style, bias)
		pout<<"*setedgemeshparams(0,"<<cirsep<<",0,0)\n";
		pout<<"*setedgemeshparams(1,"<<dn<<",0,0)\n";
		pout<<"*setedgemeshparams(2,"<<cirsep<<",0,0)\n";
		pout<<"*setedgemeshparams(3,"<<dn<<",0,0)\n";
		pout<<"*setedgemeshparams(4,"<<dn<<",0,0)\n";
		pout<<"*hmmeshlinedrag(0)\n";
		pout<<"*storeAMelemstodatabase(0,0)\n";
		// front surface
		pout<<"*surfacemode(4)\n";
		pout<<"*createplane(1,1,0,0,0,0,0)\n";
		pout<<"*createmark(lines,1) ";
		//for(j=0;j<cirsep;j++){
			pout<<" "<<i*3+2;
		//};
		pout<<endl;
		pout<<"*splinesurface(lines,1,1,1,0)\n";
};
pout<<"*collectorcreate(components,\"cavity\",\"cavity\",9)\n";
pout<<"*currentcollector(componenet,\"cavity\")\n";
pout<<"*createmark(systems,1) \"all\"\n";
pout<<"*deletemark(systems,1)\n";
pout<<"*mergefile(\"Z:/Program/cavity.hm\",1,1)\n";
//pout<<"*feinput(\"#iges\\iges\",\"Z:/Prrgram/BGA.IGS\",1,0,0.01,0,1)\n";
pout<<"*createmark(surfaces,1) \"all\"\n";
pout<<"*numbersmark(surfaces,1,1)\n";
pout<<"*createmark(nodes,1) \"all\"\n";
pout<<"*nodemarkcleartempmark(1)\n";
pout<<"*createmark(surfaces,1) \"all\"\n";
pout<<"*renumber(surfaces,1,1,1,0,0)\n";
pout.close();
pout.open("trim.cmf",ios::out);
cout<<"Input the number of plane to be trim:\n";
cin>>plane1;
cout<<"Input the number of plane to be deleted:\n";
cin>>plane2;
pout<<"*createmark(surfaces,1) \"all\"\n";
pout<<"*renumber(surfaces,1,1,1,0,0)\n";
```

```
		pout<<"*createvector(1,0,0,1)\n";
		for(i=0;i<NOWI;i++){
			pout<<"*createmark(surfaces,1) "<<plane1<<endl;
			pout<<"*createmark(lines,2) "<<i*3+3<<endl;
			//pout<<"*surfacemarkclipwithlines(1,2,0.01)\n";
			pout<<"*surfacemarksplitwithlines(1,2,1,1,0)\n";
			pout<<"*createmark(surfaces,1) \"all\"\n";
			pout<<"*renumber(surfaces,1,1,1,0,0)\n";
			//pout<<"*renumberall(1,1,0,0)\n";
			pout<<"*createmark(surfaces,1) "<<plane2<<endl;
			pout<<"*deletemark(surfaces,1)\n";
		};
		pout.close();
		return 0;
};
```