

國立臺灣大學工學院機械工程學系
學士班學生論文

Department of Mechanical Engineering
College of Engineering
National Taiwan University
Bachelor Degree Thesis

利用粒子群演算法進行核反應爐組件之
最佳化切割設計

Use The Particle Swarm Optimization Algorithm to Find
Optimal Cutting Plan for Nuclear Reactor Components

鍾詔東

Chao-Tung Chung

指導教授：陳湘鳳 博士
Advisor: Shana Smith, Ph. D.

中華民國 109 年 4 月

April, 2020

口試委員審定書

國立臺灣大學學士班學生論文

口試委員會審定書

利用粒子群演算法進行核反應爐組件之

最佳化切割設計

Use The Particle Swarm Optimization Algorithm to Find
Optimal Cutting Plan for Nuclear Reactor Components

本論文係鍾詔東(B05502037)在國立臺灣大學機械工程
學系完成之學士班學生論文，於民國 109 年 4 月 15 日承下
列考試委員審查通過及口試及格，特此證明

口試委員(3 位)：

(簽名)

(指導教授)

系主任：_____ (簽名)

摘要

核反應為現代科技當中相當重要的非再生能源科技，相較其他種類的非再生能源如火力發電以及天然氣發電，核能發電的發電效率是遠大於其他發電方法。然而核電廠設備到達一定的使用年齡後必須要報廢拆除。針對如此高輻射量以及風險的拆除計畫並不能草率處理，必須有完整的輻射防止計畫，來避免其對於環境的破壞程度。而本研究目的為找出最佳拆除工程規劃，以降低切割成本以及環境成本。

本研究最佳拆除工程規劃主要方向為降低切割廢料體積，以延長切割刀具的生命週期以及減少廢料切割面積。而為了防止輻射外洩至自然環境，必須使用特殊容器裝載切割零件，再將運送至廢料放置中心。故在進行切割模擬最佳化時，必須注意零件是否會超過容器尺寸，並且在此切割條件下找出最佳切割路徑。

本研究使用 SolidWorks API (application programming interface)、Visual Studio C# 以及 Excel 開發一個最佳拆除工程規劃。以 STL 作為 3D 工程圖建模分析幾何圖形，並利用離散化資料以及粒子群演算法整合切割位置，最終使用 SolidWorks 模擬切割結果，並輸出的切割零件以及廢料體積分析。

整合演算法與 SolidWorks 後設計出自動化和最佳化的切割模擬軟體，使用者可自行輸入任意的工程圖模型、切割容器尺寸以及切割刀具厚度，分析切割工程廢料體積以及零件工程圖。如頂部導版切割模擬使用 6 mm 切割厚度以及 10 mm 的精準度，模擬結果花費 35 sec 切出下 18 個零件以及 2,709,059 mm³ 的切屑體積。

關鍵詞：核電廠除役、最佳化切割、SolidWorks 二次開發、粒子團演算法、
STL 截面分析

目錄

口試委員審定書	i
摘要	ii
第 1 章 緒論	1
1.1 研究動機	1
1.2 研究目的	2
1.3 研究方法	3
第 2 章 文獻參考	5
2.1 輻射危害	5
2.1.1 生物危害	5
2.1.2 機械危害	5
2.2 核電廠除役	6
第 3 章 設計方法	9
3.1 流程簡介	9
3.2 Visual Studio C#	10
3.3 SolidWorks 應用程式介面	11
3.3.1 SolidWorks 巨集	12
3.3.2 SolidWorks API VBA 介面	13
3.4 STL file	14
3.4.1 STL Binary Read	15
3.4.2 STL in SolidWorks	16
3.5 最佳化演算法	17
3.5.1 粒子群演算法	18
3.5.2 粒子群	19
3.5.3 向量參數	20
第 4 章 最佳化設計	21
4.1 設計條件	21
4.1.1 切割環境條件限制	21
4.1.2 切割假設條件	22

4.2	切割模型定義	22
4.3	截面積計算	23
4.3.1	截面分析	23
4.3.2	截面積計算	26
4.4	粒子群演算法	29
4.4.1	切割刀數	29
4.4.2	切割長度限制	29
4.4.3	切割精確度	31
4.4.4	初代切割最大化	32
4.4.5	變異度	33
4.4.6	Global Best	34
4.4.7	演算法整合	36
第 5 章	SolidWorks 切割模擬	38
5.1	切割流程簡介	38
5.2	切割模型方法	39
第 6 章	切割模擬結果	41
6.1	人機介面設計	41
6.2	模型切割結果展示	45
6.2.1	頂部導板切割結果	45
6.2.2	反應器壓力槽切割結果	47
6.2.3	爐心側板切割結果	49
6.2.4	上熱屏蔽層切割結果	51
第 7 章	結論與未來展望	54
7.1	結果討論	54
7.1.1	優勢分析	54
7.1.2	改良空間	55
7.2	未來展望	57
參考文獻	58
附錄	60
	附件一: STL 讀檔程式碼	60

附件二: STL 截面積計算程式碼.....	61
附件三: SolidWorks API C# 特定程式碼	63

圖 目 錄

Figure 2.1 2019 年美國商用運行反應爐	6
Figure 2.2 (左)VR 技術員工訓練 (右)工程環境可視化.....	7
Figure 2.3 (左)切割流程模擬 (右)工程環境模擬.....	8
Figure 3.1 簡介資料流程	9
Figure 3.2 Visual Studio 編輯介面	10
Figure 3.3 Visual Studio C# Window Application 介面編輯.....	10
Figure 3.4 SolidWorks 巨集功能	12
Figure 3.5 SolidWorks VBA 內建人機介面設計平台.....	13
Figure 3.6 SolidWorks VBA 內建程式編譯	13
Figure 3.7 Visual Studio 引入 SolidWorks API 步驟.....	14
Figure 3.8 STL 建模方式	15
Figure 3.9 STL 檔案資料結構與意義	16
Figure 3.10 STL 實驗 3D 工程圖樣本	16
Figure 3.11 群體粒子演算法運算流程	18
Figure 3.12 PSO 運用 Eqution(1)尋找最小值區塊(左)10 代(中)20 代(右)30 代	19
Figure 4.1 切割廢料計算簡化原理	22
Figure 4.2 三角形與平面相交之條件	24
Figure 4.3 三角形於平面相交(a)相交於兩點(b)相交於一條線(c)相交於一點	25
Figure 4.4 Matlab 分析 Figure 3.9 STL 檔在 $y=200$ 平面之截面	25
Figure 4.5 模型轉換為 STL 後法向量意義(a)範例原模型(b)STL 檔	26
Figure 4.6 STL 擷取線段後截面積計算範例.....	27
Figure 4.7 範例 Figure 4.6 的截面線段 ABCD 分析	27
Figure 4.8 範例 Figure 4.6 的截面線段 EFGH 分析	27
Figure 4.9 範例 Figure 4.6 最終截面積.....	28

Figure 4.10 截面積計算演算法.....	28
Figure 4.11 工件 XYZ 方向尺寸皆超過容器尺寸時的最大切割長度決定	30
Figure 4.12 工件 XYZ 方向尺寸有兩方向超過容器尺寸時的最大切割長度決定	30
Figure 4.13 工件 XYZ 方向尺寸僅有一方向超過容器尺寸時的最大切割長度決定	31
Figure 4.14 工件 XYZ 方向尺寸皆不超過容器尺寸時的最大切割長度決定	31
Figure 4.15 精準度概念圖	32
Figure 4.16 初始化最大化切割長度(a)隨機分佈(b)最大化分佈	32
Figure 4.17 變動度對粒子影響(a)產生粒子前(b)產生粒子後	33
Figure 4.18 粒子超出最大長度限制則更新為最大長度.....	34
Figure 4.19 PSO 初始條件設定	35
Figure 4.20 Local Best 更新方法	35
Figure 4.21 Global Best 更新方法	35
Figure 4.22 PSO 切割演算法設計流程圖	36
Figure 5.1 SolidWorks 切割模擬流程	38
Figure 5.2 核反應爐 Y 方向切割	39
Figure 5.3 分割功能操作介面	40
Figure 6.1 瀏覽模型檔案	41
Figure 6.2 選擇目標切割檔案.....	41
Figure 6.3 確認模型位置後點選下一步	42
Figure 6.4 切割資料輸入介面.....	42
Figure 6.5 自動化切割執行狀態	43
Figure 6.6 選擇是否開啟結果資料夾	43
Figure 6.7 最終切割資料介面.....	44
Figure 6.8 切割資料夾內容	44
Figure 6.9 切割數據 Cut_data 內容.....	45
Figure 6.10 最終切割結果零件.....	45
Figure 6.11 頂部導板檔案輸入.....	46
Figure 6.12 頂部導板 SolidWorks 模型.....	46
Figure 6.13 頂部導板切割參數.....	46

Figure 6.14 頂部導板切割結果.....	47
Figure 6.15 頂部導板切割截面積收斂圖	47
Figure 6.16 反應器壓力槽檔案輸入.....	48
Figure 6.17 反應器壓力槽 SolidWorks 模型	48
Figure 6.18 犯應器壓力槽切割參數.....	48
Figure 6.19 反應器壓力槽切割結果	49
Figure 6.20 反應器壓力槽切割截面積收斂圖	49
Figure 6.21 爐心側板檔案輸入.....	50
Figure 6.22 爐心側板 SolidWorks 模型.....	50
Figure 6.23 爐心側板切割參數.....	50
Figure 6.24 爐心側板切割結果.....	51
Figure 6.25 爐心側板切割截面積收斂圖	51
Figure 6.26 上熱屏蔽層檔案輸入.....	52
Figure 6.27 上熱屏蔽層 SolidWorks 模型	52
Figure 6.28 上熱屏蔽層切割參數.....	52
Figure 6.29 上熱屏蔽層切割結果	53
Figure 6.30 上熱屏蔽層切割截面積收斂圖	53
Figure 7.1 極座標切割反應器壓力槽	55
Figure 7.2 電腦卡式座標辨識是否可容於容器錯誤	56
Figure 7.3 演算法碎屑產生示意圖	56
Figure 7.4 演算法切割多餘微小切削現象.....	57

表 目 錄

Table 1 STL 精準度比較表.....	17
Table 2 XYZ 軸方向的切割平面公式.....	24
Table 3 範例 Figure 4.6 之每一線段計算與總截面積關聯	28
Table 4 演算法最佳化變數比較	36
Table 5 PSO 演算法 pseudo code	37

第1章 緒論

能源在現代社會中扮演著重要的角色，不論是生活、交通、工業以及娛樂等都與能源息息相關。而人類自工業革命以來最常使用的能源方式為發電機。從最早的蒸氣發電機，利用蒸氣向上移動動能帶動渦輪轉動，再藉由磁能轉移至電能儲存，再到現代的再生能源發電的水力發電、風力發電、地熱發電、洋流發電以及非再生能源的火力發電、核能發電。而如今隨著社會人口不斷增加人類對於能源的需求不斷增加，以及環保意識抬頭的環境下，發電效率以及環境維護逐漸成為現代科技最主要的發展目標。其中又以近百年才出現的「核能發電」為最具代表性，其發電效率遠高於火力發電，且對於環境的影響也較小。但也因為其風險高，進而導致其產生的能源廢料危險性高，因此許多國家反對使用核能發電。故在現今社會中如何讓大眾接受核能發電所造成的影響？又如何利用較佳的工程方法降低風險，以及降低環境成本，成為這一世代相當重要的能源議題。

1.1 研究動機

世界上所有產品都有生命週期，不論是電腦、冰箱、汽車或是房屋都會隨著時間的推移在功能上漸漸的削弱，而逐漸的面臨到「廢棄」以及「淘汰」的命運。同樣的，核能發電廠如今已運作多年對於人類的能源貢獻龐大，但同樣的隨著時間的推移，發電廠中的設備已無法維持原先的發電效率，甚至危害到發電的安全性，故最終仍然須面臨到除役。

然而核電廠的除役是一件浩大的工程，首先是在核能發電時期反應爐中的許多零件都遭到中子輻射的危害，若任意棄置至自然環境中會造成需多的環境生態衝擊，甚至進入人類生活食物鏈而影響人類的健康生活。像是日本福島於2011年3月11日的核電廠因為地震所造成的爆炸毀損，而導致輻射外洩至自

然環境，造成當地的許多生態動物突變以及農作物毀損，甚至影響日本經濟發展。

故在拆除核電廠時需要注意不只是拆除時的安全性，也要注意在拆除後輻射危害對於環境的影響。實際的做法為利用防輻射容器包裹廢料以及廢料零件。故整體的核電廠拆除計畫應依序為，設計生產防輻射規格容器、規劃切割工程、準備切割機器以及人力、進行切割工程、將運送防輻射容器至廢料中心。而這項拆除工程必定會產生一定量的工程經濟成本和環境成本，故如何規劃最佳切割工程成為不可或缺的環節。例如用什麼樣的刀具才能有經濟效益的完成工程？什麼樣的切割路徑才能有效的延長刀具壽命？什麼樣的切割方式才能讓輻射量降至最低？這些工程問題是否可以利用適當的軟體設計來解決，也成了本研究的主要研究動機。

1.2 研究目的

對於一項拆除工程而言有無限多種的拆除工程及方法，且每一項拆除工程都有不同的優缺點。而本研究的最主要目的為降低工程所造成的經濟成本以及環境成本，故在這無限的工程以及方法當中要利用適當的方法篩選出最佳的工程方法。對於刀具消耗來說，切割所產生的廢料越少，可以使單一工程當中所使用的刀具數量越少。對於環境輻射來說若是切割的表面積越少，輻射外露的量會越少。故若假設零件的所有的材質皆相同、且輻射外洩量僅與表面積成正比，可以得知若選擇的切割方法中具有較少的切割廢料以及切割截面積，即可以有效的減少成本及汙染。

這一項工程若只是切除圓柱體，可以簡單的理解為將所有的切割距離拉到最大的容器範圍即可完成。若要一刀將啞鈴分為兩半可以理解為切除握把(截面積較小)即可。但隨著工程問題的擴張以及複雜化，人腦以及肉眼已無法負荷如此高的計算量。故本研究必須要設計出一套軟體能夠代替人腦，自動完成最佳

化的工程計算。故在本項研究當中將設計出一套能夠「自動化」以及「最佳化」工程切割的方法。

1.3 研究方法

本研究的工程條件為已有固定的防輻射容器來裝箱廢料零件，且擁有核電廠中的各項設施的 3D 工程圖以及切割刀具的厚度，故必須在這限制以及資料下完成資料讀取、資料分析、資料處理、資料輸出的功能。

資料讀取

目前有許多商用軟體可提供 3D 繪圖、顯示、以及編輯，像是 CREO、Inventor、SolidWorks 都是可以提供 3D 工程圖的軟體。然而由於切割時需要用到大量的計算，單純只是 3D 工程圖無法滿足如此龐大需求。故需要能夠連接外部程式語言的能力。本研究以 SolidWorks 為主要的建模軟體，利用其內建 SolidWorks 應用程式介面(API)連結至外部程式語言 C#，以做大量的資料處理，故需要輸入的檔案類型為 SLDPRT。

資料分析

在最佳化的過程中需要讀取大量的模型幾何特徵，雖然 SolidWorks 有內建量測體積、面積功能，但若進行大量計算截面積資料而言會費時太久。故在分析資料時，以較簡化的近似模型以獲得更快的面積計算速度。最終以 STL 檔案為近似模型去計算截面積。

資料處理

在取得截面積後，必須使用適當的演算法找出最佳切割方法，而切割條件受到容器大小的限制。切割方法需有三種特性，分別為切割刀數、切割方向、切割位置，才能表達出完整的工程模擬方法，在數學建模使用陣列 (Array) 表示切割三種特性。而在演算法上將使用較適合向量模型最佳化的粒子群演算法 (PSO)。

資料輸出

輸出資料為切割完成的零件、切割總廢料體積、最佳化切割收斂圖。故需藉由 SolidWorks API 自動化生成切割零件檔、以及利用 Excel API 輸出切割結果。

第2章 文獻參考

2.1 輻射危害

2.1.1 生物危害

輻射是一個具有高動能的電磁波或粒子，若暴露於自然界當中很容易造成物質變形，其中最著名案例為基因突變。^[1]普通生物學中描述到基因是由 DNA 構成，其中 DNA 是以核苷酸 (AMP) 構成，而在生物上的所有細胞中皆利用 DNA 轉換為 RNA，再利用轉譯蛋白生成胺基酸後，再藉由其他輔助蛋白來生成身上各部位所需要的蛋白，故可以說基因是生物運作資料庫中心。而單一胺基酸需要藉由三個核甘酸構成，且不同的順序數量可以生成不同的胺基酸，然而若 DNA 中的核甘酸排列順序遭到外部干擾或竄改損傷而破壞既有的順序，會造成這些蛋白質的變異以及功能異常，進而導致生物的特徵產生變異。而輻射就是自然環境最常造成 DNA 變異的外部因子，而稱此現象為基因突變。

雖然基因突變在生物演化史當中扮演著非常重要的角色，但事實上基因突變對於大部分的生物而言都是有害的，甚至會因此而導致生態系的崩毀。像是日本的福島的核電廠輻射外洩導致了許多生物的體型變異，最終無法負荷身體的變異而造成大量生物死亡。故在處理核廢料時也必須要能夠有效減少輻射外洩，以維護人類生活的安全性以及生態平衡。

2.1.2 機械危害

輻射除了在生態環境造成危害之外，輻射對於硬體也會造成危害。^{Hilsdorf, Kropp and Koch [2]}論文中提到核能反應當中最常出現的輻射為中子射線以及 γ 射線，兩者都是帶有高能量的輻射，極有可能導致原子變異。而大部分的核電廠都是以水泥作為防止輻射的材料，因其具有高聚合材料且成本較低。然而根

據研究，當中子以高速撞擊單一原子時會導致原子跳脫晶體束縛能，導致原子結構變形而造成材料脆化。除此之外高速的中子撞擊後的高能原子，會因為能量衰減而釋放 γ 射線 (Secondary Gamma Radiation)，除增加材料的輻射性外，也會導致材料溫度上升至約 250 度，對於水泥材料來說造成其 Tensile Strength 的強度下降至 50% (即便高溫導致的水蒸發會增加材料強度)。故在核能發電中冷卻水泥材料也是非常重要的一環，然而再強的冷卻劑仍然無法去除水泥材料內的輻射汙染，故在核電廠除役處理上必須使用更有效的防輻射容器去處理廢料。

2.2 核電廠除役

核能發電廠的除役是許多國家共同面臨的問題。以美國為例，在 2019 年總共 97 個反應爐 (Reactor Pressure Vessel, RPV) 在運行，佔全年總發電量的 19.7%，更佔無排放能源發電量的 60%，可見核能在美國的能源發展扮演重要的角色。然而 2013 年以來有 6 座核電廠成功除役且 8 座將在宣告在即將於 2025 年正式除役，到 2017 年更是有 10 座核電廠成功除役而 20 座正在除役階段當中。而每一座除役的成本都是龐大且危險的，故適當的除役計畫也成為美國近年來的重要議題。而從 Figure 2.1 中觀察到美國有數十反應爐正在運行當中，不久的將來核電廠的除役需求也會同步上升。

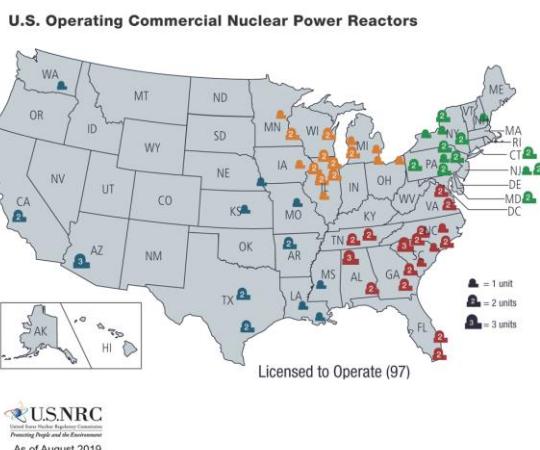


Figure 2.1 2019 年美國商用運行反應爐

Wald [3]計畫整套的除役工程，包括反應爐切除方法以及廢料零件運送流程地點，甚至還有熱處理方式來降低水泥輻射量後再進行運送。在核電廠除役中，人體曝曬於輻射的危險性過高，造成拆除核電廠的過程許多困擾，故許多國家開發電腦技術來取代人力曝曬於輻射的機率。例如 Solstad and Van Nieuwenhove [4]在挪威的 Halden 反應器除役計畫中開發一系列的 3D 軟體來輔助工作人員執行工作如 Figure2.2 所示，Halden Planner 用於最佳化 3D 除役流程、Halden Simulation Editor 用於 VR 員工訓練工作流程等等都是電腦輔助工程的軟體。



Figure 2.2 (左)VR 技術員工訓練 (右)工程環境可視化

Hyun, Kim, Lee, Kim, Jeong, Choi and Moon [5] 究韓國研究反應爐的拆解模擬，以 AutoCAD 以及 3DS Max 兩個軟體模擬核反應爐的除役工程，如 Figure 2.3 所示該軟體可以模擬機械手臂、人工切除的效能以及安全性，並可以計算整體工程所需耗費的時間成本、經濟成本以及人力成本等，對於除役工程有極大的貢獻。除此之外 Jeong, Choi, Moon, Hyun, Lee, Kim, Seo, Jeong, Lee, Song, Lee and Son [6] 也利用虛擬實境 (Virtual Reality) 技術模擬出各環境區塊的輻射量密度，對於工程人員可以提供更安全的保障空間。

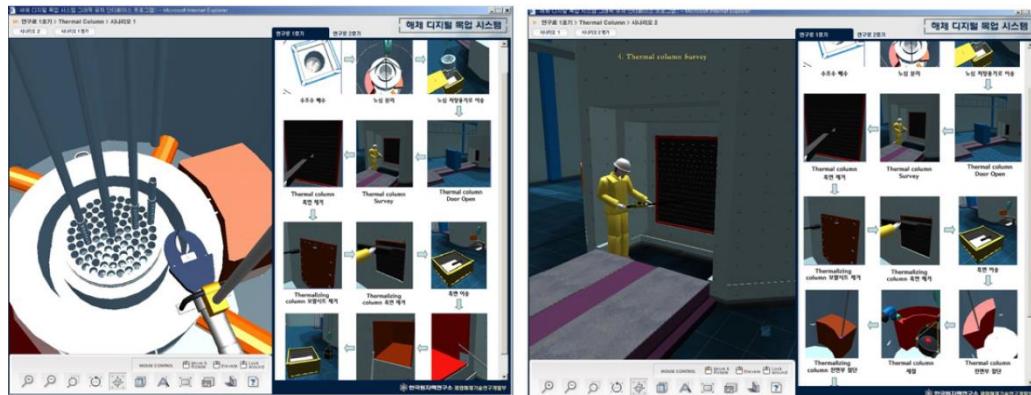


Figure 2.3 (左)切割流程模擬 (右)工程環境模擬

Fujiwara, Minowa and Munesawa [7]將擴增實境 (Augmented Reality) 應用於核電廠除役，先收集目標拆解物外觀尺寸等資料，再藉由 3D CAD 檔繪製出並建立虛擬空間模擬工程環境，最後將機器人置入核電廠後，利用 AR 技術同步操作員運行。如此的技術可以增加資料收集的速度，以及降低工作人員曝曬於危險輻射區域的時間，對於除役過程更能夠有效率的規劃除役工程。

除了除役的安全性之外，Tsai, Huang, Hung, Huang and Smith [8]利用 SolidWorks 二次開發以及基因演算法，計算最佳化切割工程。將切割目標的 3D CAD 檔輸入 SolidWorks，再輸入裝載廢料切割容器尺寸以及切割厚度，轉存 STL 檔案以計算切割截面積後，利用基因演算法 (Genetic Algorithm) 將整體的切割廢料最小化，最終模擬出所有切割廢料的總體積，結果可降低刀具的損耗成本以及環境污染成本。

第3章 設計方法

3.1 流程簡介

本研究使用 Visual Studio C#、Solidworks 2016、和 Microsoft Office Excel 開發切割規劃軟體。Visual Studio C#為主要的資料整合平台。先利用 C#內建人機介面設計資料視窗後，再利用 SolidWorks API 轉換目標檔案為 STL 檔。以截取大量截面積，最後利用粒子群演算法找出最佳切割方案後，藉由 SolidWorks API 自動模擬切割結果，並由 Excel 輸出模擬結果。整合上述內容可以得到 Figure 3.1 的流程圖，其中以粗黑框標記的流程為需要利用 SolidWorks API 操作的步驟。

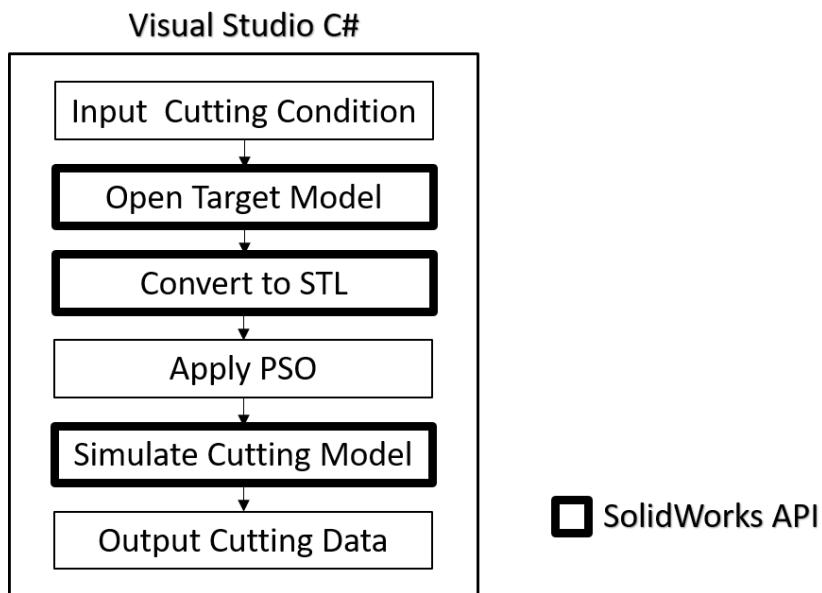


Figure 3.1 簡介資料流程

3.2 Visual Studio C#

Visual studio 屬於程式編譯器，可以處理 C/C++、C#、Python 等等時常會使用到的程式語言，且如 Figure 3.2 所示，Visual Studio 可以引用外部參考函式庫使用，並提供分離式的排版，將不同物件導向分裝於不同區塊，固在 Debug 或編輯程式時可以較清晰的排列方式編輯。

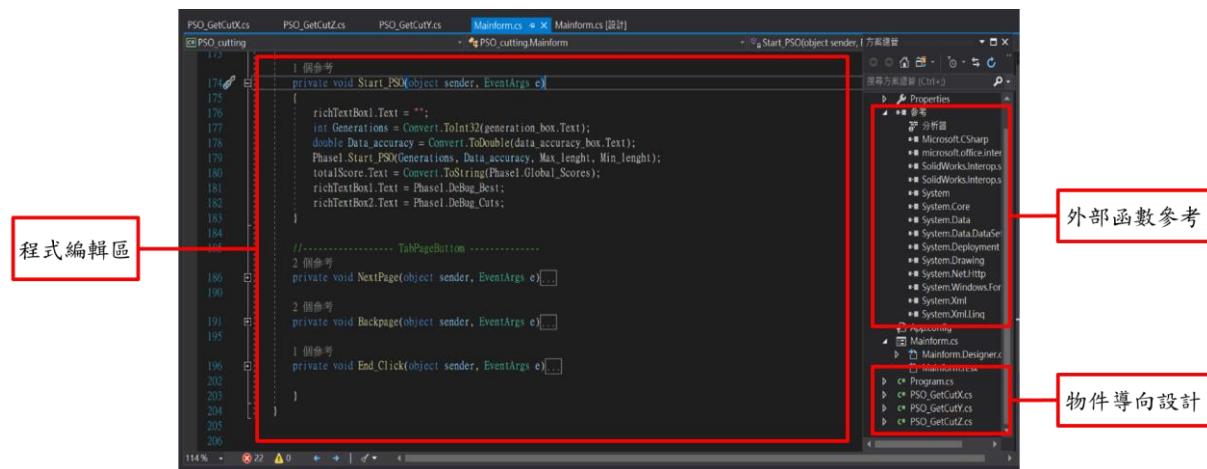


Figure 3.2 Visual Studio 編輯介面

本研究最主要的程式語言為 C#，相較於 C/C++有較豐富的函示庫供使用者直接使用，除此之外 Visual Studio C# 在 Windows Application 上提供選單式的方法設計人機介面如 Figure 3.3 所示，因此可以省去自己撰寫 GUI 程式的過程。利用手動拖曳、手動拉伸即可設計應用程式。

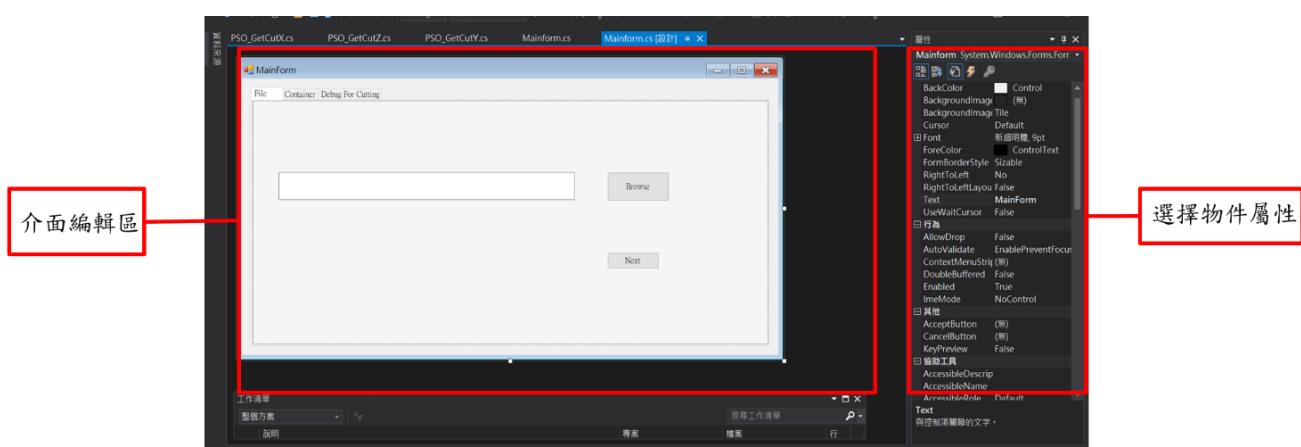


Figure 3.3 Visual Studio C# Window Application 介面編輯

3.3 SolidWorks 應用程式介面

SolidWorks 是一個 3D 的工程繪圖軟體，其具有豐富的功能供使用者繪製工程圖。然而有些工程圖具有許多重複特徵，例如切割、鑽孔，若利用手動模擬，將會耗費大量的時間。故 SolidWorks 有提供一個程式化管道來實踐「自動化」設計，稱為 SolidWorks 二次開發，而其中最主要的管道為 SolidWorks API。

API 為 Application Program Interface 英文縮寫，也就是軟體與外界的互動窗口。SolidWorks API 則是程式語言與 3D 工程圖互動的窗口介面，在使用 SolidWorks 時所使用到的功能幾乎皆可以透過 SolidWorks API 控制。而本研究就是要利用 SolidWorks API 實現自動化切割模擬的功能。

Dun and Chen [9]利用 SolidWorks 二次開發設計種子栽培盤，將種子栽培盤中的特徵尺寸設計為可調整參數後，利用 Visual Basic 以及 SolidWorks API 連接至 Excel 的xlsx 檔來達到各種不同的種子栽培盤的設計，並可以直接利用xlsx 表中的參數改變現有零件的尺寸參數，成功的完成自動化設計。

Dong and Song [10]研究利用 SolidWorks 二次開發模擬自動化焊接工程，利用 Visual Basic、SolidWorks API 以及 ROBOGUIDE 整合焊接機械手臂，以及輸入焊接目標工件的流程模擬，只要輸入預計焊接的目標，軟體將會自行模擬機械手臂加工路徑，並計算加工成本輸出、加工流程以及加工結果，對於工廠加工自動化非常有幫助。

3.3.1 SolidWorks 巨集

SolidWorks 內建的巨集(Macro)記錄工具，可以直接在 SolidWorks 自帶的 VBA(Visual Basic for Applications)中紀錄使用的操作命令，也可以藉由 Macro 來將重複簡單的操作步驟，編譯為 VBA 後重複執行，可以有效的增加繪圖效率。除此之外也可以利用錄製的方式，記錄操作過程中所執行的函示，並配合線上 SolidWorks API 說明，學習函示的正確參數設置及運行。而巨集功能在 SolidWorks 中的使用方式如 Figure 3.4 所示。

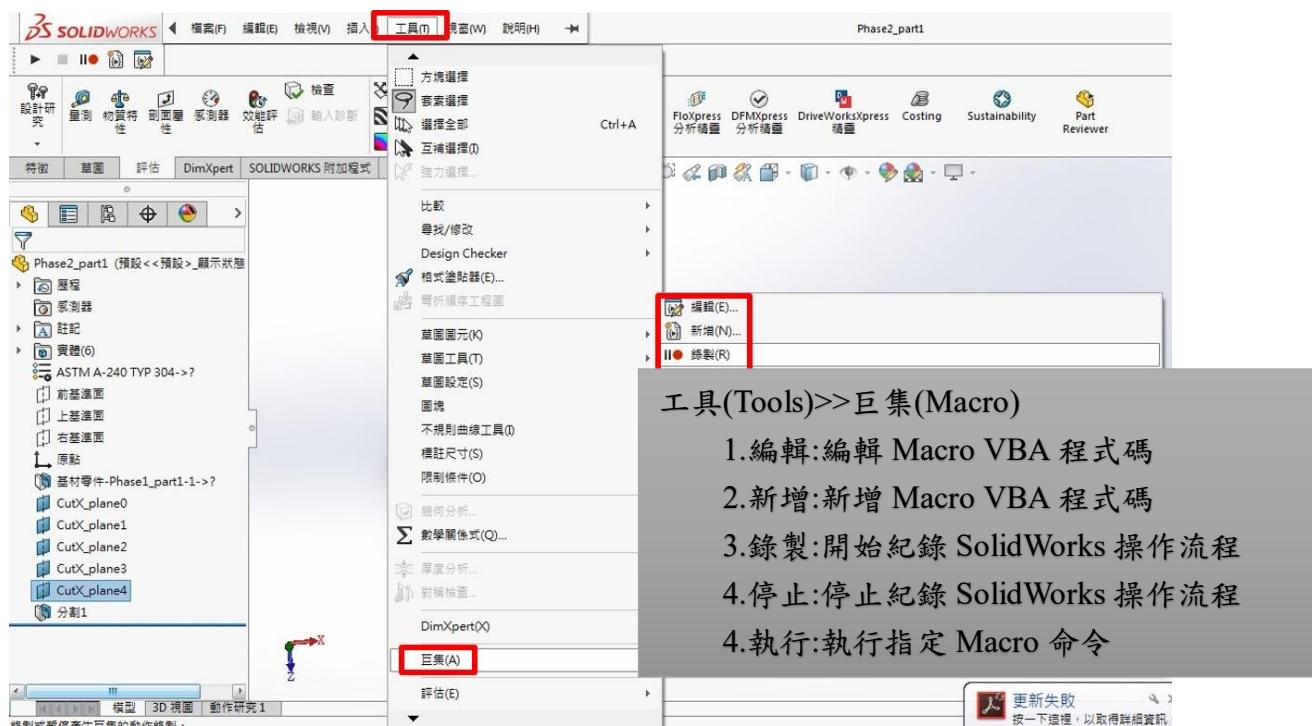


Figure 3.4 SolidWorks 巨集功能

1. 編輯:編輯 Macro VBA 程式碼

2. 新增:新增 Macro VBA 程式碼

3. 錄製:開始紀錄 SolidWorks 操作流程

4. 停止:停止紀錄 SolidWorks 操作流程

4. 執行:執行指定 Macro 命令

3.3.2 SolidWorks API VBA 介面

SolidWorks 2016 有內建的控制語言 VBA，也可以自行設置人機介面來進行功能操作，像是參數化的產品設計、展示產品尺寸參數或是應力分析等等功能都是可以透過 SolidWorks API 內建的人機介面來完成。而在 3.3.1 章節中所提到的錄製功能，錄製完畢後輸出的檔案類型就是 VBA 程式碼。

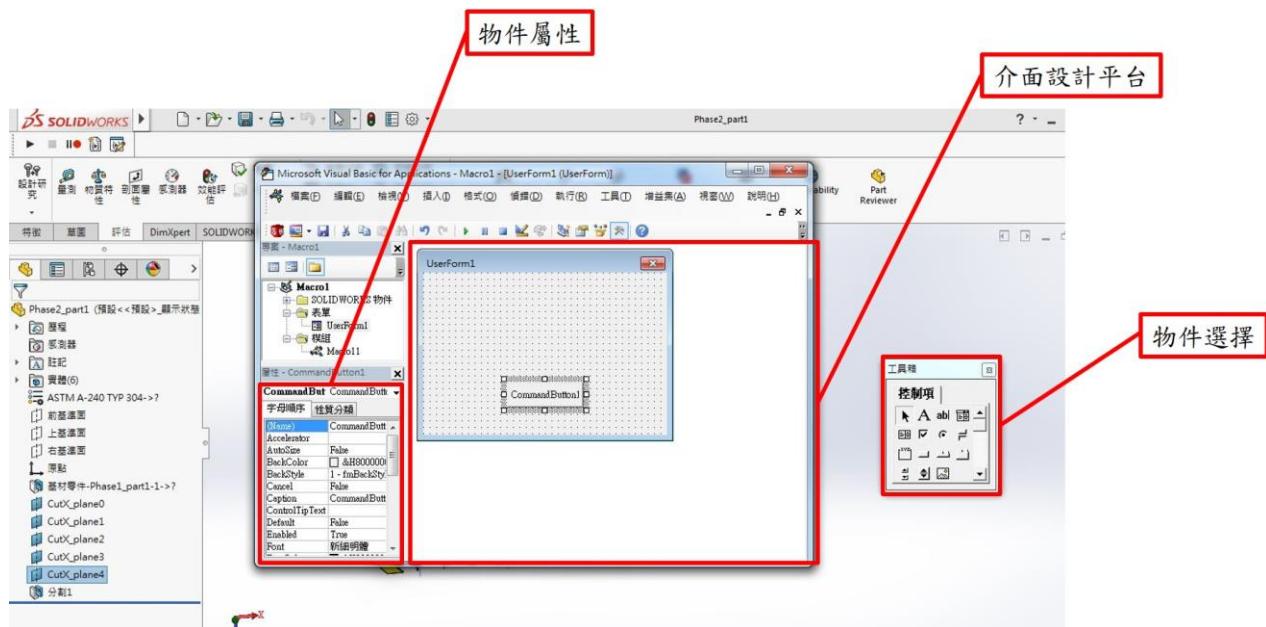


Figure 3.5 SolidWorks VBA 內建人機介面設計平台

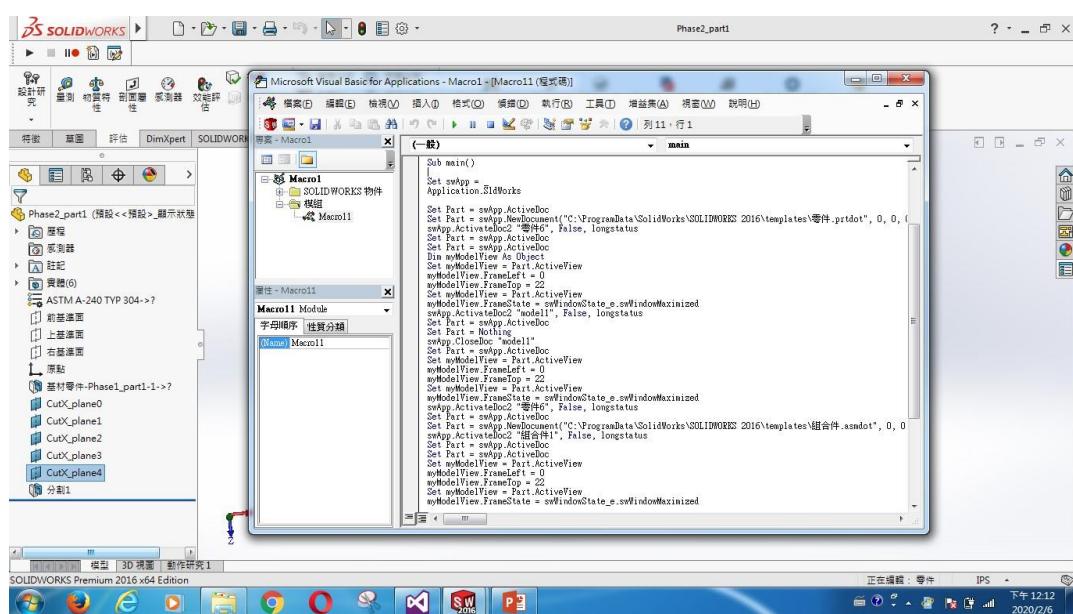


Figure 3.6 SolidWorks VBA 內建程式編譯

Figure 3.5 以及 Figure 3.6 VBA 已經可以在 SolidWorks 中達到自動化設計以及人機介面處理，然而使用 SolidWorks 內建的 VBA 人機介面的功能有限，對於前端程式設計、資料處理以及資料輸出至 Excel 相對困難，固主要編輯程式語言依然以具有較多控制功能的 Visual Studio C#為主，固在使用 SolidWorks API 之前必須將 SolidWorks API 列入 Visual Studio C#當中的參考資料才能引用 API 函式庫(操作流程如 Figure 3.7 所示)。

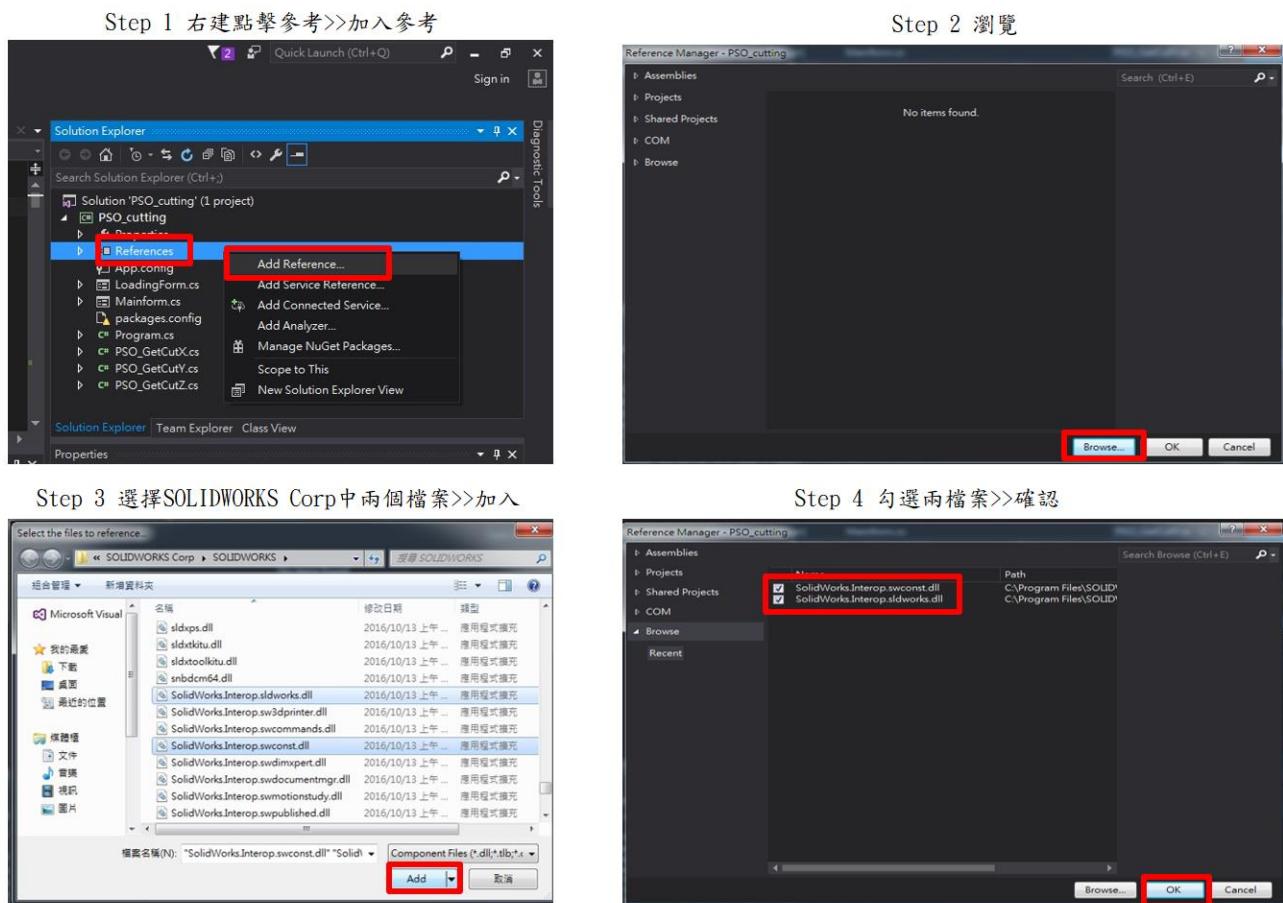


Figure 3.7 Visual Studio 引入 SolidWorks API 步驟

3.4 STL file

電腦輔助設計系統 (Computer-Aided Design, CAD) 與快速成型技術 (Rapid prototyping, RP) 的資料交換介面格式 STL (STereoLithography)，是美國 3D System 公司於 1987 年提出的，目前已成為 RP 工業界的標準。運作原理為將複雜的工件模型簡化儲存為許多三角形所構成的資料檔(如 Figure 3.8 所示)，而

STL 檔最為廣泛使用的地方為 3D 列印，原因為 STL 檔相較於 3D 工程軟體所輸出的零件檔有較少的檔案空間，相較之下做 3D 列印的加工規劃較為快速。

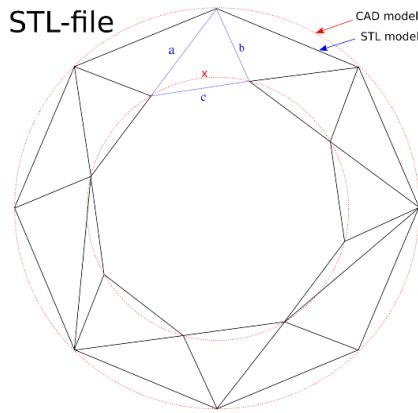


Figure 3.8 STL 建模方式

M. and Gy [11]研究關於 STL 檔案以及真實工件的誤差，發現精度越高雖然可以讓 STL 越趨近於真實 3D 工程圖，但同時也會導致許多不連續邊界的位置會有空洞或是重疊的現象產生，故若工程需求的誤差容忍度足夠時，可使用適量的精準度即可，避免資料量多卻沒有減少誤差的窘境。

Eragubi [12]利用 STL 檔規劃雷射 3D 列印。利用 STL 檔取得加工件的截面型態後，再利用演算法模擬該截面的雷射加工路徑，有效的利用 STL 檔完成自動化雷射加工的規劃。而本研究會需要使用讀取 STL 檔以及利用 STL 的資料作幾何分析，故需要設計有效的資料結構以及演算法來處理 STL 檔。

3.4.1 STL Binary Read

STL 檔案的儲存格式為二進碼，也因此其佔有的容量空間小於原先 3D 工程圖。而這二進碼中前 80 Bytes 為檔案辨識碼，與原先工程圖的幾何意義毫無相關，而其後 4 Bytes 為三角形總數的 4 位元組整數碼，可以藉由 C# 中 Binary Read 的 ReadInt32() 來讀取數值。而往後則是所有三角片的資訊，且每一個三角片所存取的資料有三個頂點座標向量以及一個法向量，且每一個向量皆以三個 4 位元浮點數的型態存放在檔案中。最後在每一三角片資料的後方皆有 2 位元的結束指令，故每一個三角片所佔用的位元數為 50 Bytes。完整的 STL 資料結

構如 Figure 3.9 所示(讀取程式碼可參考附錄-附件一)。

```
UINT8[80] - Header  
UINT32 - Number of triangles  
  
foreach triangle  
REAL32[3] - Normal vector  
REAL32[3] - Vertex 1  
REAL32[3] - Vertex 2  
REAL32[3] - Vertex 3  
UINT16 - Attribute byte count  
end
```

Figure 3.9 STL 檔案資料結構與意義

3.4.2 STL in SolidWorks

SolidWorks 可以將 3D 的工程圖儲存為 STL 檔，而三角形的總數量可以藉由調整精度來決定，越高的精度就以越多的三角形來儲存資料，而 STL 檔就越趨近於原先 3D 工程圖。而為了觀察 STL 的精度與 STL 的相關性質，將 3D 物件(實驗物件如 Figure 3.10 所示)用 SolidWorks 內建的存檔設定中調整不同精度參數儲存 STL 檔，並利用 MatLab 繪圖觀察變化。

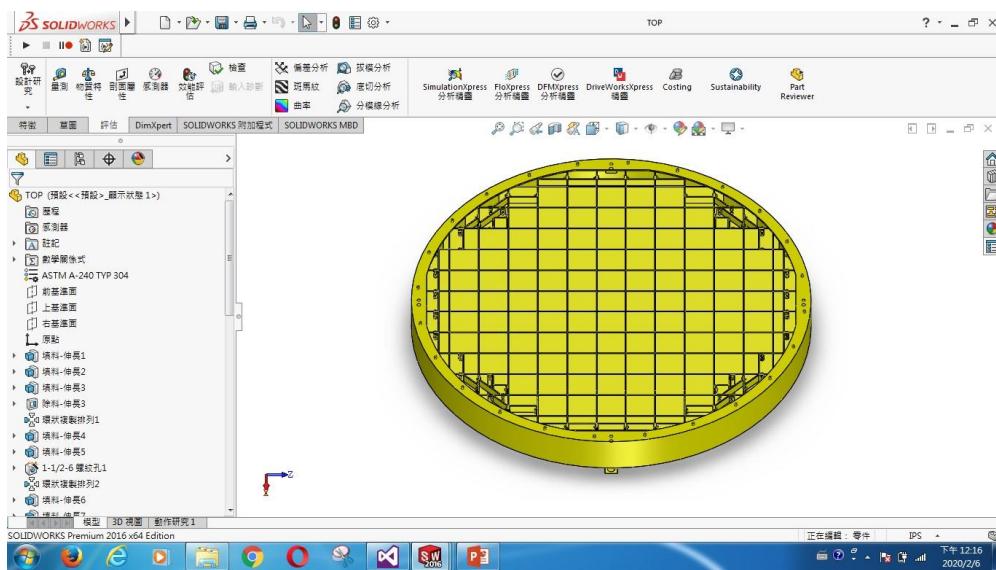
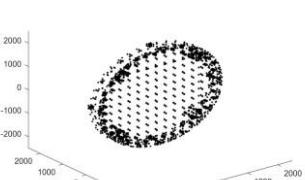
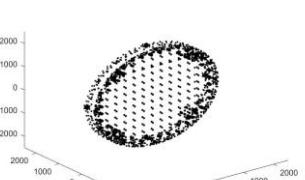
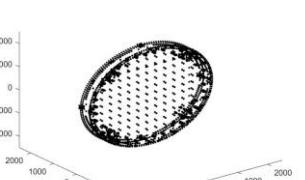


Figure 3.10 STL 實驗 3D 工程圖樣本

Table 1 STL 精準度比較表

精度	粗糙	良好	優異
精度參數設定			
三角形數量	24822	50758	90430
點狀分布圖			

從 Table 1 發現隨著三角片的數量增加，STL 精準度的增加不少，但從三角形的頂點分布上可以發現到隨著精密度的增加，頂點分布並不會均勻成長而是會在圓形區塊有較明顯的增加。我認為造成此現象的原因為三角形的建模對於線狀特徵可以較輕易的建模，但對於圓形來說需要無限多個三角形才能趨近於圓形，故調整精準度會較針對非線狀特徵進行更精確的建模，故精確度並不會讓增加的三角形均勻遞增。

3.5 最佳化演算法

最佳化演算法是運用程式語言去增加目標的效益，可以利用於找出函數的最高點、Curve Fitting、成本分析等各式各樣可量化問題。要證明最佳化演算法為有效演算法的方式為繪製出收斂圖 (Fitness vs iteration Graph)，以證明於演算法前以及演算法後目標的效益有變化。而最佳化演算法當中最具代表性的演算法為 Particle swarm optimization (PSO) 以及 Genetic algorithm (GA)。Shi, Liang, Lee, Lu and Wang [13]發展出整合性的演算法。GA 是以基因代碼作為變數變換的演算法，也就是 1 與 0 的方式調控變數，較適合使用於較數位的問題 e.g 工

廠加工配置(flowshop scheduling problem)、物流中心配置。PSO 則是向量為主的演算法，較適用於連續的問題 e.g 尋找函數最低點、Curve fitting。而在此研究中將基因演算法中的交配以及突變的特性結合到向量解中，也成功增加全域解中的收斂速度。但在本研究中主要是利用群體粒子演算法為最主要的最佳化演算法。

3.5.1 粒子群演算法

粒子群最佳化 (Particle Swarm Optimization, PSO)，是由 J. Kennedy 和 R. C. Eberhart 於 1995 年開發的一種演化計算技術，來源於對比群集動物行為模擬的社會行為。群居動物共同的目標為尋找水源或食物，找到大量的食物後會利用生物賀爾蒙或其他生物訊號來吸引其他夥伴的到來，像是以蜜蜂為例，在出巢後所有的蜜蜂都在尋找適合採蜜的區域，當尋找到較佳的區塊採蜜後會以特殊的飛行方式通知其他的蜜蜂來到已採取更多花蜜回蜂巢。

而粒子群演算法也是利用相同的模式在尋找最佳解，演算法開始在限制的範圍中任意的撒下許多粒子，且每一個粒子都有屬於自己的數據，而在每一次的移動前都會將這筆數據與處理的問題做評分，開始移動後這些數據會以向量的形式向最佳的數據前進。

PSO 演算法流程圖

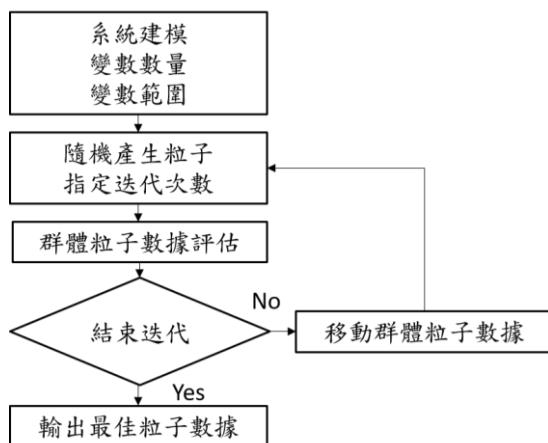


Figure 3.11 群體粒子演算法運算流程

3.5.2 粒子群

粒子移動的向量需要根據最佳的數據去決定，然而若所有的數據往目前最佳的數據前進，而這目前的最佳數據不一定是全域數據中的最佳解，故可能導致最終的演算法無法找到真正的最佳解。除此之外若最佳解並非唯一解，會導致每次做出來的結果可能不同。故在 PSO 演算法當中引進「群」的概念，所有的粒子中為一大族群，而這大族群中存在著小族群並利用這兩族群中分別的最佳資料來得到下一步的向量模型，如此可以增加向量的變異性以及多重最佳解的特性。

$$F(x_1, x_2) = \left(x_2 - \frac{5}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10 \quad (1)$$

以 Equation (1) 公式為例， $F(x_1, x_2)$ 最小值包括 $(\pi, 2.275)$ 、 $(9.4278, 2.475)$ 、 $(-\pi, 12.275)$ 。而利用 PSO 演算法去處理此問題，在運算過程中發現隨著移動次數的增加，漸漸地發現所有粒子逐漸分成 3 大區塊且往最小值的座標點集中。

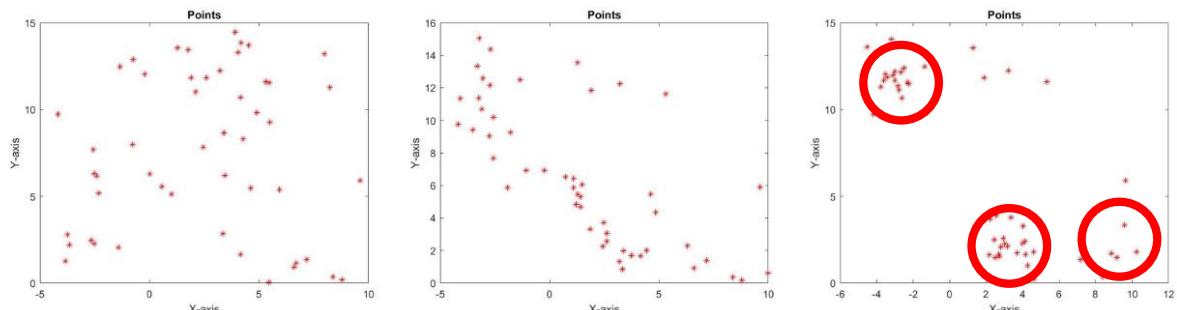


Figure 3.12 PSO 運用 Eqution(1)尋找最小值區塊(左)10 代(中)20 代(右)30 代

而其中可以發現在 $(9.4278, 2.475)$ 的區塊粒子分布較小，是因為在該區塊的斜率值較大，故粒子有較低的範圍在該區塊找到相對小的量值，因此較少粒子有機會能夠分布在此區塊中。

3.5.3 向量參數

在最佳化演算法中時常以隨機變數來尋找最佳解，像是基因演算法中時常以交配(Crossover)、突變(Mutation)來將現有的資料量複雜化來擴大尋找的資料。而同樣的在群體粒子演算法當中也有相同的功能，是利用向量參數變異來增加粒子的搜索範圍，越大的向量變異能力會造就越強的粒子搜索能力，而增加全域最佳解的搜索能力。而越小的變異能力，可能形成局部解。因此在粒子群演算法的向量移動中，須調整參數來處理不同的題目。

$$V_i^* = k * \left(V_i + rand * c_1 * (P_{i\text{best}} - P_i) + rand * c_2 * (G_{\text{best}} - P_i) \right) \quad (2)$$

$$k = \left| \frac{2}{2-\varphi-\sqrt{\varphi^2-4\varphi}} \right| \quad , \quad \varphi = c_1 + c_2 \quad \&& \quad \varphi > 4 \quad (3)$$

$$P_i^* = P_i + V_i^{**} \quad (4)$$

V_i^* ∈ 更新向量 V_i ∈ 原先向量

c_1 ∈ local向量參數 c_2 ∈ Global向量參數

P_i ∈ 目前粒子位置 $P_{i\text{best}}$ ∈ Local最佳數據 G_{best} ∈ Global最佳數據

$rand$ ∈ 0~1 的隨機變數 k ∈ 向量變異數

其中 c_1 為調整局部解的能力， c_1 若越大可以使的演算法更容易找到局部解。而 c_2 則是調整全域最佳解的能力， c_2 越大可以讓演算法更容易找到全域最佳解。而其他的參數則是增加粒子的變異程度，其中 $rand$ 是一個 1~0 的隨機數來增加粒子的亂度，而 k 則是增加收斂速度的參數經驗值。

第4章 最佳化設計

4.1 設計條件

4.1.1 切割環境條件限制

在一項切割工程當中，切割環境是假設不變的，故在進入模擬之前必須先定義哪些切割條件會影響到切割工程。

(一) 切割工具

本項工程中最重要的是切割工具。切割時所去除掉的廢料量與切割刀具有關，而廢料指的是水泥材料受到切割刀具切割後產生的切屑，越厚的切割刀具所生成的廢料會越多。而廢料會增加輻射外洩總表面積而增加輻射量。除此之外，切割工具也是一種耗材，其加工效率會隨著加工的時間逐漸下降，而當效率低到一定的程度後導致整體工程品質降低，必須要做適當的刀具更換來恢復工程品質。故可得知：

- (1)切割刀具的尺寸會影響廢料的數量
- (2)切割的廢料可以被視為自然成本以及社會成本的一種。

而為了滿足各種不同的工程需求，切割道具厚度可由使用者自行輸入。

(二) 切割容器

由於本項工程可能具有高輻射量，故切割廢料與工件需要以特殊的容器盛裝。故切割時須注意到切割下來的工件尺寸必須小於容器的尺寸，否則將無法放入該容器當中。故可得知

- (1)在設計切割路徑時需要考慮到切割工件的最大尺寸
- (2)容器為三維空間物體，故切割工件的最大尺寸與切割方向有關。

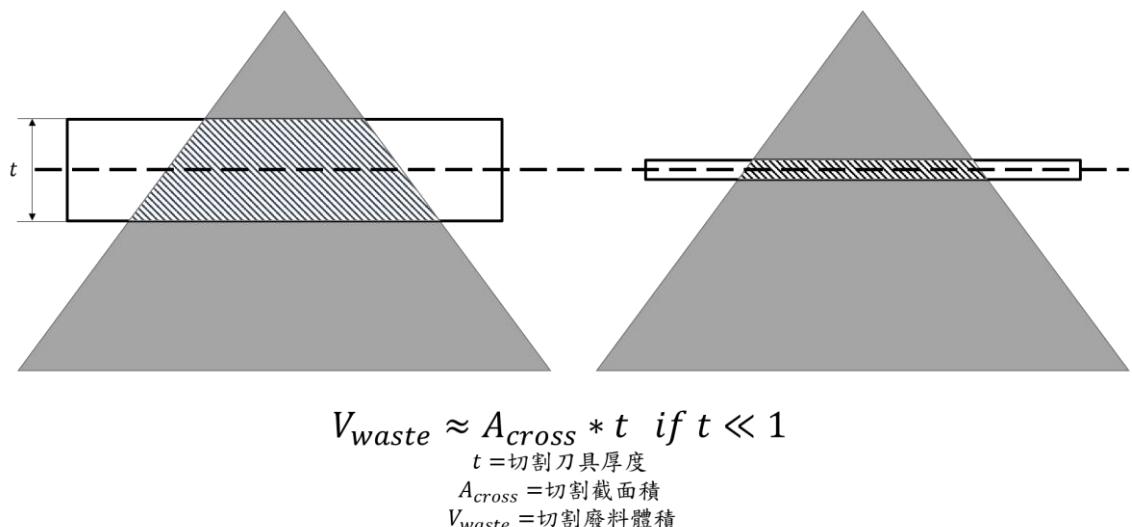
而在本研究中容器為長方體，可以自行輸入長寬高作為工程容器。

4.1.2 切割假設條件

由於在切割核反應爐組件時，除了需要考慮到的切割刀具以及時間之外，還需要考慮工件當中每一個區塊的輻射量強度，才能較全面的計算出最佳切割路徑。然而這些資料在取得上有較大的困難。故在設計最佳化的方法之前，必須做一些先決條件假設，來降低系統的複雜度。假設條件如下：

- (1) 假設工件當中所有部分材料皆相同
- (2) 假設輻射量僅與切割面積以及廢料有關
- (3) 切割刀具遠小於工件尺寸，故廢料體積約等於切割截面積乘以刀具厚度(如

Figure 4.1 所示)



$$V_{waste} \approx A_{cross} * t \quad if \quad t \ll 1$$

t = 切割刀具厚度
 A_{cross} = 切割截面積
 V_{waste} = 切割廢料體積

Figure 4.1 切割廢料計算簡化原理

4.2 切割模型定義

在開始進行最佳化前必須將所有的變數以及目標函數定義。本研究的研究目的為降低切割廢料體積維持刀具壽命，且最終切割下的零件必須要符合容器尺寸大小，故先以定義簡易的數學型量化切割工程。

$$p = \min \left(\sum_{i=1}^{N_x} \text{Area}(X_i) + \sum_{j=1}^{N_y} \text{Area}(Y_j) + \sum_{k=1}^{N_z} \text{Area}(Z_k) \right) \times t \quad (5)$$

$$x_{i+1} - x_i \leq \text{Container Length} \quad \forall i = 1, 2, \dots, N_x - 1 \quad (6)$$

$$y_{j+1} - y_j \leq \text{Container Length} \quad \forall j = 1, 2, \dots, N_y - 1 \quad (7)$$

$$z_{k+1} - z_k \leq \text{Container Length} \quad \forall k = 1, 2, \dots, N_z - 1 \quad (8)$$

$p \in$ 目標函數

$t \in$ 切割刀具厚度 $\text{Area}(a) \in$ a 平面截面積

$X_i \in X$ 軸方向距離原點 x_i 的切割平面 $N_x \in X$ 方向的切割刀數

$Y_i \in Y$ 軸方向距離原點 y_i 的切割平面 $N_y \in Y$ 方向的切割刀數

$Z_i \in Z$ 軸方向距離原點 z_i 的切割平面 $N_z \in Z$ 方向的切割刀數

上述的數學模型中 Equation (5) 為目標函數，為切割工程所產生的切割廢料體積。而 Equation (5)(6)(7) 則是限制在 XYZ 三切割方向中兩者的切割距離不能超過容器的尺寸。故接下來以上述的數學模型作為後端演算法的開發軸心。

4.3 截面積計算

4.2 中的 Equation (5) 中需要能夠讀取切割平面的截面積。在 SolidWorks 中的「評估」當中具有量測體積、面積以及重心等的功能，但截面積的量測方法需要使用到剖面圖的功能，才能獲得特定區段的截面積。利用 SolidWorks API 確實可以辦到，但在後段的演算法處理中需要計算上千次不同區段的截面積，會耗費較長的時間才能完成計算。為了增加截面積的計算速度，本研究選擇使用結構較簡單的 STL 檔，試著快速獲得截面積。

4.3.1 截面分析

於第 3 章的 3.4 小節已知 STL 是由大量的三維空間三角片所構成，故所有三維模型的表面都會被這些三角片取代或近似。故若取得特定截面所經過的三角片，且做適當的處理則可以得到此截面的幾何特徵。故加入截面積計算的三

角片首要條件為通過該截面平面。將三角片中的三個頂點代入指定平面公式(如 Figure 4.2 $y-4=0$)中，觀察是否全部為正負同號，若三者的結果皆為同號，則表示此三角形位於平面的同側，若有其中任兩對為異號則表示此三角形有通過該平面，而若有 0 則表示此三角形的頂點位於該平面上。

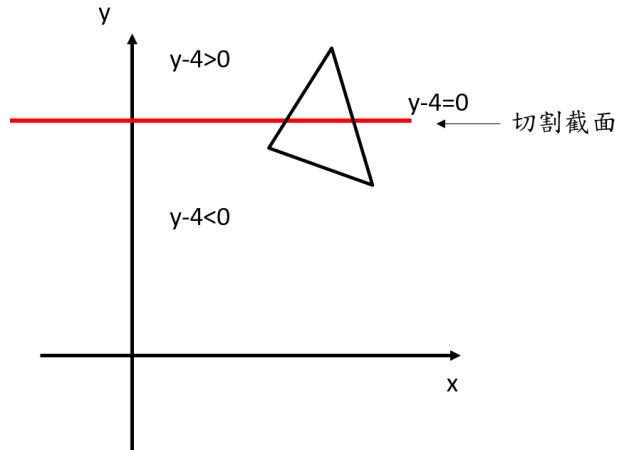


Figure 4.2 三角形與平面相交之條件

而本研究目前所處理的切割方式為 XYZ 方向的切割方向，故切割平面公式分別在 X、Y、Z 軸方向的公式如 Table 2 所示：

Table 2 XYZ 軸方向的切割平面公式

X 軸向切割	$x - x_0 = 0$	x_0 = 距離 YZ 平面距離
Y 軸向切割	$y - y_0 = 0$	y_0 = 距離 XZ 平面距離
Z 軸向切割	$z - z_0 = 0$	z_0 = 距離 XY 平面距離

而在三維空間中平面與三角形有許多相交的方式則有許多種，其中完全重疊，就是切割平面就是模型表面，故可以被視為不會被刀具切割，因此不納入計算考量。故在截面積分析中主要以 Figure 4.2 中的三種情況做討論：(a)相交於兩點(b)相交於一條線(c)相交於一點。

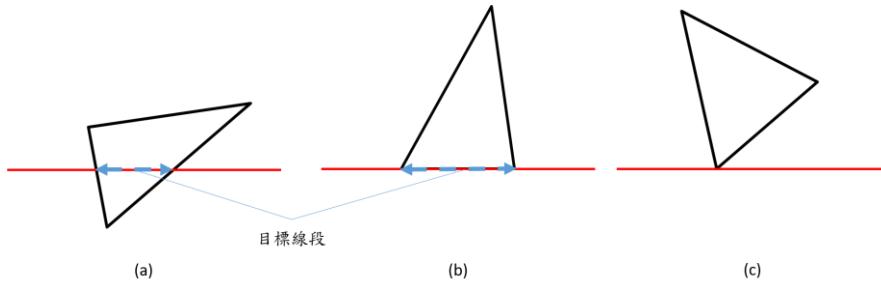


Figure 4.3 三角形於平面相交(a)相交於兩點(b)相交於一條線(c)相交於一點

Figure 4.3 中(a)的兩個交點可以藉由內差法求出兩交點座標，再得到相交線段，而(b)則可以直接利用相交兩頂點，計算目標線段，但(c)由於只相交於一點，可以被視為目標線段的長度趨近於 0，故也可以不納入計算考量。而為了證明理論的正確性，同樣的利用 Matlab 做實驗分析觀察，利用 Figure 3.10 的工程圖樣本所輸出的 STL 檔，且存檔方式為 Table 1 的粗糙參數設定來做實驗。而切割方法選擇以 X-Z 平面做為切割平面方向，故輸入 Y 軸方向的切割高度即可做出該截面的模型樣本。

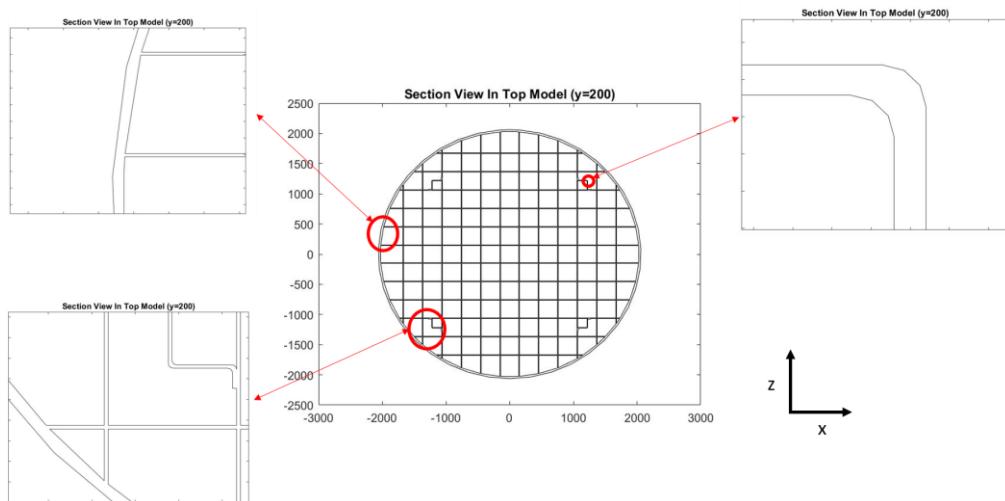


Figure 4.4 Matlab 分析 Figure 3.9 STL 檔在 $y=200$ 平面之截面

從 Figure 4.4 可以發現到從 STL 檔案中截下來的線段彙整到同一張圖後確實能夠得到原模型的截面樣貌，雖然在許多圓形處仍然有不連續的現象可能導致在後端的截面積處理產生誤差，但這誤差不足以影響到整體的分析結果。

4.3.2 截面積計算

在前小節後已經可以得到在截面線段，若要計算截面積則需設計適當演算法整合這些線段資訊。在 3.4 章節中提到 STL 檔中三角形資料會有三個頂點及一個法向量座標，本小節即是配合法向量來取得截面積的計算。而在 STL 檔中三角形法向量是表示模型在此三角形平面中哪一側是實體(如 Figure 4.5 所示)，故可以藉由此特徵判斷截面積的計算方式。

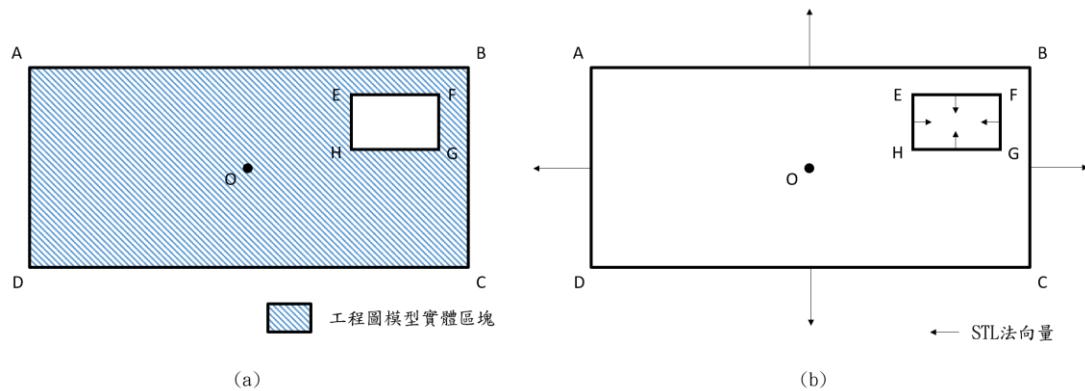


Figure 4.5 模型轉換為 STL 後法向量意義(a)範例原模型(b)STL 檔

在計算截面積時將所有於先前被截取下來的線段進行處理，而處理每一線段的演算法如下：

- (1) 計算該線段的中點，並與原點形成一個中點向量(如 Figure 4.7 紅色向量)
- (2) 計算該線段的法向量(如 Figure 4.7 藍色向量)與中點向量(如 Figure 4.7 紅色向量)的內積
- (3) 若兩向量內積為正則利用該線段兩端點與原點(截面中心)計算三角形面積，並將此三角形面積加入總截面積(參考 Figure 4.7 - Figure 4.8 以及 Table 3)
- (4) 若兩向量內積為負則利用該線段兩端點與原點(截面中心)計算三角形面積，並將此三角形面積減去總截面積(參考 Figure 4.8 (e)(f) 以及 Table 3)

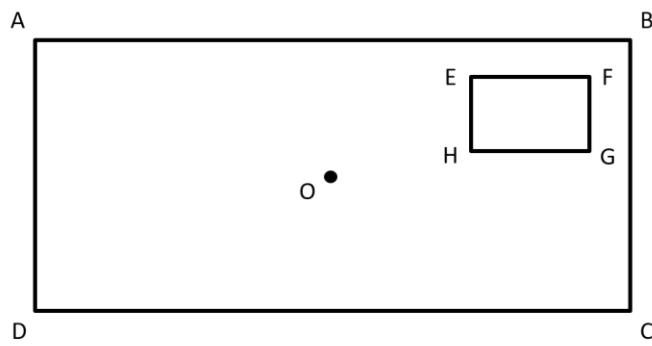


Figure 4.6 STL 撷取線段後截面積計算範例

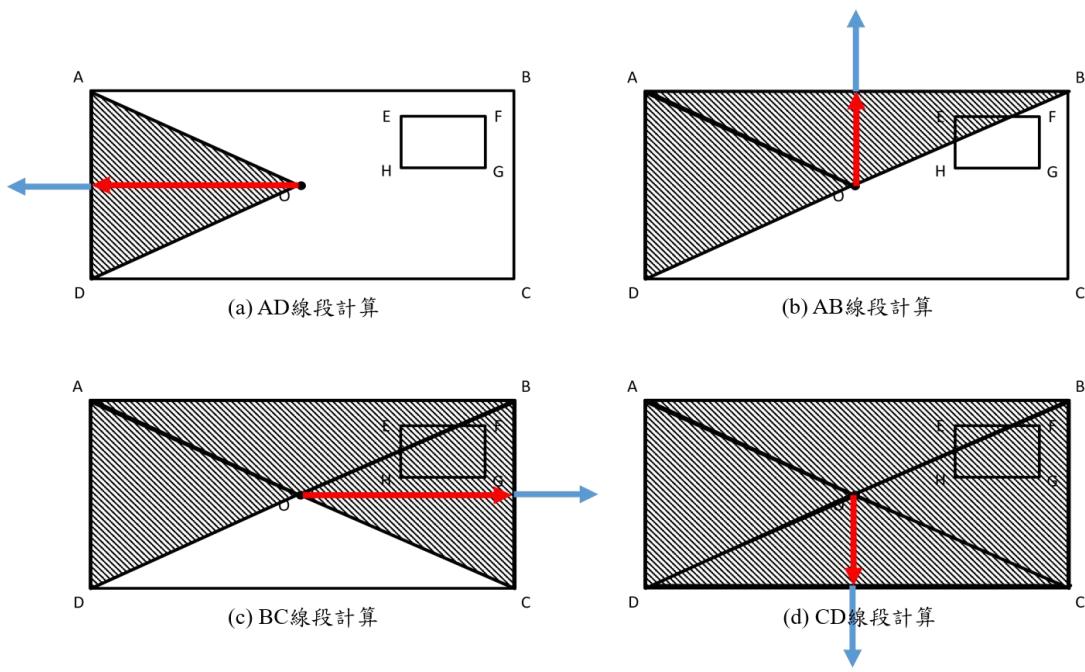


Figure 4.7 範例 Figure 4.6 的截面線段 ABCD 分析

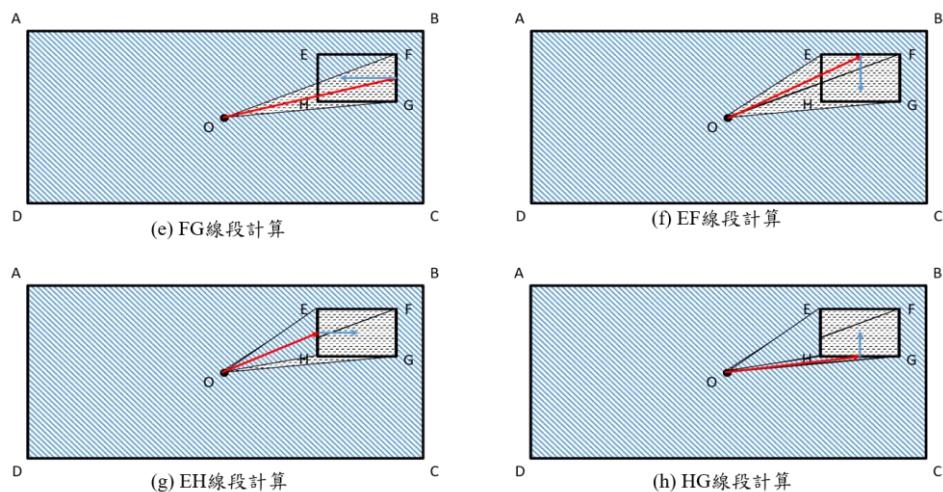


Figure 4.8 範例 Figure 4.6 的截面線段 EFGH 分析

Table 3 範例 Figure 4.6 之每一線段計算與總截面積關聯

目標線段	對應圖	中點向量與法向量內積	三角形面積	總截面積
AD	(a)	+	$Area(OA, OD)$	$+Area(OA, OD)$
AB	(b)	+	$Area(OA, OB)$	$+Area(OA, OB)$
BC	(c)	+	$Area(OB, OC)$	$+Area(OB, OC)$
CD	(d)	+	$Area(OC, OD)$	$+Area(OC, OD)$
FG	(e)	-	$Area(OF, OG)$	$-Area(OF, OG)$
EF	(f)	-	$Area(OE, OF)$	$-Area(OE, OF)$
EH	(g)	+	$Area(OE, OH)$	$+Area(OE, OH)$
HG	(h)	+	$Area(OH, OG)$	$+Area(OH, OG)$

#三角形面積計算為兩項量行列式值後之絕對值後除以二

$$Area(X, Y) = \frac{|\det(|X|)|}{2}$$

而在 Figure 4.7 ~ Figure 4.8 的計算之後，及可以得到如 Figure 4.9 的截面積樣貌。而最終即可整合完整的截面積計算流程(如 Figure 4.10 所示)，程式碼可參考附錄-附件二。

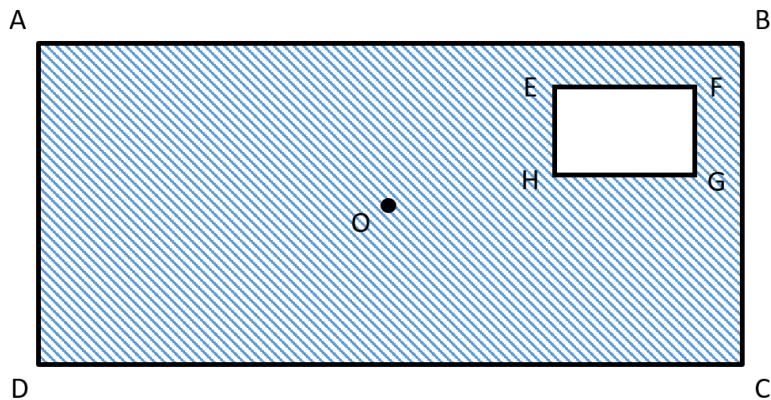


Figure 4.9 範例 Figure 4.6 最終截面積

STL截面積計算演算法

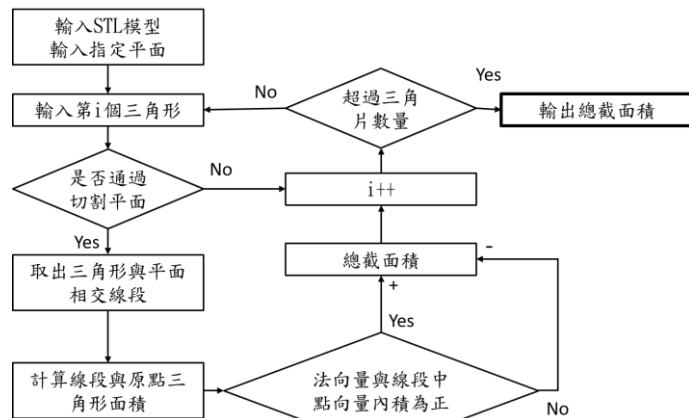


Figure 4.10 截面積計算演算法

4.4 粒子群演算法

在最佳化演算法的設計中無非是希望有夠好的模型、夠強的收斂度以及高速的運算速度。然而本研究所面對的切割模型是無固定模型且尺寸範圍大小變化很大，故也造成切割上不少變數，因此也降低收斂度以及運算速度。故在切割模擬建模中，需產生更多與傳統的粒子群演算法不同的地方。

4.4.1 切割刀數

完整的切割流程有不同的方向以及距離，理論上來說每一刀的距離以及方向都會是最佳化演算法當中的變數，在小型物件上或許還能夠成功模擬出較佳的方案。故在此定義切割刀數為決定後端演算法的變數數量(對應 4.2 中的 N_x 、 N_y 、 N_z)，而這些變數量值則量質則表示這每一刀的位置(對應 4.2 中的 x_i 、 y_j 、 z_k)。

在切割上將切割方向分三個不同方向來進行演算法，首先在 Y 軸方向進行演算法後切割，之後便改為 Z 軸方向跑演算法後切割，最後為 X 軸方向跑演算法後切割，而每次演算法運行的變數量將會由尺寸大小以及長度限制共同決定。

4.4.2 切割長度限制

本研究使用的容器目前皆是以長方體為主，故切割尺寸限制會以此長方容器的長寬高為參考(對應到 4.2 中 Equation(6)(7)(8)的 Container Size)。單一方向中變數的數量由輸入容器的尺寸而定，且在切割的種種情況中較長的切割長度限制可以讓模型有機會花更少的切割次數完成工程。而在單一方向切割前會先將此模型的 XYZ 方向的長度讀出，且在包裝容器的長寬高長度皆不同的情況下，切割長度限制的決定方法分下列幾種情形，而在這些不同的情況終將設計

較好的方法設定切割長度限制。(假設目前切割方向為 Y 方向)

1. 切割模型 XYZ 方向尺寸皆超過長方形容器長寬高

若三方向皆超過長寬高最大尺寸，則 Y 方向切割長度限制為長、寬、高中最大長度

包裝容器		
長	寬	高
1690	1680	1630

切割物件			
方向	X	Y	Z
尺寸	4560	18000	4560
最小容納尺寸	NA	NA	NA

Max Cut Length = 1690
單位: mm

Figure 4.11 工件 XYZ 方向尺寸皆超過容器尺寸時的最大切割長度決定

2. XYZ 方向有兩方向超過長方形長寬高

假設未超過容器長寬高的邊長為 a，保留長方形長、寬、高中最小可容納 a 的長度，於剩餘兩者中挑選較大一邊為最大切割長度限制

包裝容器		
長	寬	高
1690	1680	1630

切割物件			
方向	X	Y	Z
尺寸	3200	800	3200
最小容納尺寸	NA	1630	NA

Max Cut Length = 1690

方向	X	Y	Z
尺寸	5300	1685	4500
最小容納尺寸	NA	1690	NA

Max Cut Length = 1680
單位: mm

Figure 4.12 工件 XYZ 方向尺寸有兩方向超過容器尺寸時的最大切割長度決定

3. XYZ 方向有僅有一方向超過長方形長寬高

假設未超過容器長寬高的邊長為 a、b，保留長方形長、寬、高中最小可容納 a、b 兩組合後，於剩餘長度為最大切割長度限制

包裝容器

長	寬	高
1690	1680	1630

切割物件

方向	X	Y	Z
尺寸	1640	9000	1640
最小容納尺寸	1690	NA	1680

方向	X	Y	Z
尺寸	800	2000	800
最小容納尺寸	1680	NA	1630

Max Cut Length = 1630

Max Cut Length = 1690

單位: mm

Figure 4.13 工件 XYZ 方向尺寸僅有一方向超過容器尺寸時的最大切割長度決定

4. XYZ 方向皆不超過長方形長寬高

若三方向皆小於切割容器長度，則此模型已不需要進行切割，可輸出檔案。

包裝容器

長	寬	高
1690	1680	1630

切割物件

方向	X	Y	Z
尺寸	1530	1600	1685
最小容納尺寸	1630	1680	1690

方向	X	Y	Z
尺寸	800	1200	800
最小容納尺寸	1630	1690	1680

Cutting is not necessary

Cutting is not necessary

單位: mm

Figure 4.14 工件 XYZ 方向尺寸皆不超過容器尺寸時的最大切割長度決定

4.4.3 切割精確度

在先前的 PSO 演算法當中是以向量的形式找尋最佳解，以 Figure 3.12 為例，在尋找最小值時最佳解的精準度會隨著迭代的次數逐漸增加。然而因為在 STL 截面積計算當中存有誤差，故切割的位置的精確度再準也沒有意義。故在開始執行粒子群演算法前先將切割問題離散化以簡化模型複雜度(如 Figure 4.15 所示)，而離散化的程度可以藉由使用者參數調整，而本研究稱此參數為切割「精準度」。

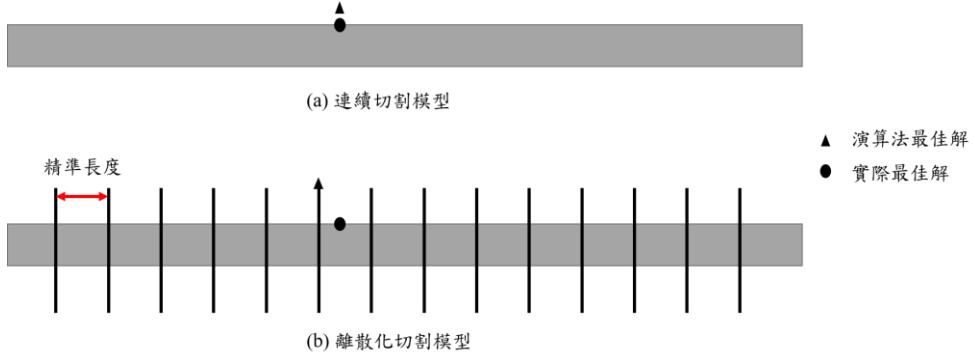


Figure 4.15 精準度概念圖

從 Figure 4.15 中可以觀察到離散化的模型確實簡化模型的複雜度，但同時精準度的引入會導致最終的解答出現誤差，故在設定此參數時需要考量整體工程的誤差容忍度。而在本研究中調整精準長度為 10 mm，對於切割工程足夠的精確度。

4.4.4 初代切割最大化

PSO 演算法通常在初代會使用隨機的方式撒點，而後續這些粒子會漸漸地趨近於最佳解。然而隨機撒點並不適用於切割工程的規劃，因為隨機撒點可能導致單一方向的切割變數量增加(如 Figure 4.16 (a)所示)，雖然就最佳化演算法最終可能還是會到最佳解，但相對所需要的迭帶次數與時間就會必較多。故在設計群體粒子演算法時，初始化的切割狀態為所有切割距離皆為最大切割長度(如 Figure 4.16(b)所示)，以便在執行演算法時可以減少部分的變數量。本研究的最大切割長度為 4.4.2 的方法所獲得的最大長度限制。

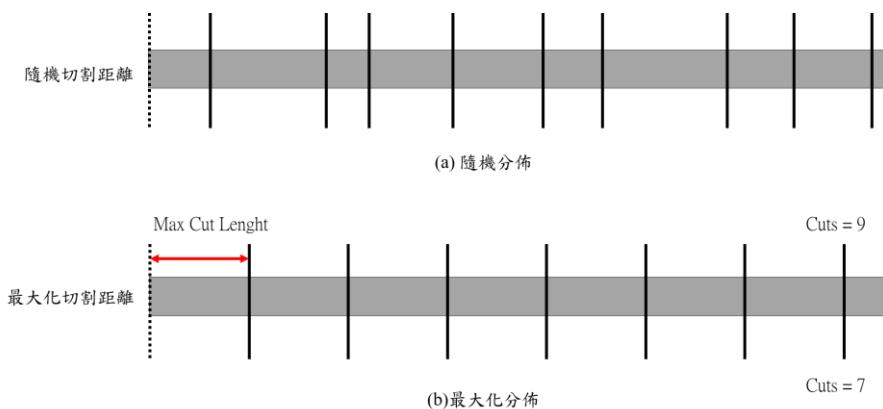


Figure 4.16 初始化最大化切割長度(a)隨機分佈(b)最大化分佈

4.4.5 變異度

在章節 3.5.1 中有提到最佳化演算法需要藉由變異因子來擴大尋找的切割範圍，但在 4.4.3 當中也提到本研究處理的內容為離散數據，向量變異的方式就相對不利於本次所處理的工程，原因為 PSO 的向量是 0~1 的隨機浮點數，屬於連續的數據移動。故傳統的粒子群演算法所使用的向量法已不適用於本次所針對的離散化切割問題，因此在本研究中使用一個新的變異數據參數稱之為「變異度」，目的是讓 PSO 在面對不連續的數據點時能夠保留原本的變異性，而變異度在切割數據的迭帶中為每一粒子可以隨機移動的範圍。

$$\text{Variability} = \left[\frac{\text{Max Cut Length}}{\text{精準度}} \right] \div 4 \quad (8)$$

Equation 8 中除以 4 是經由不斷的嘗試後發現有最好的收斂效果，故粒子在迭帶的方式會如 Figure 4.13 所示。範例 Figure 4.17 ~ Figure 4.21 皆假設變異度 = 2、最大切割長度 = $6 \times \text{精準長度}$ 。

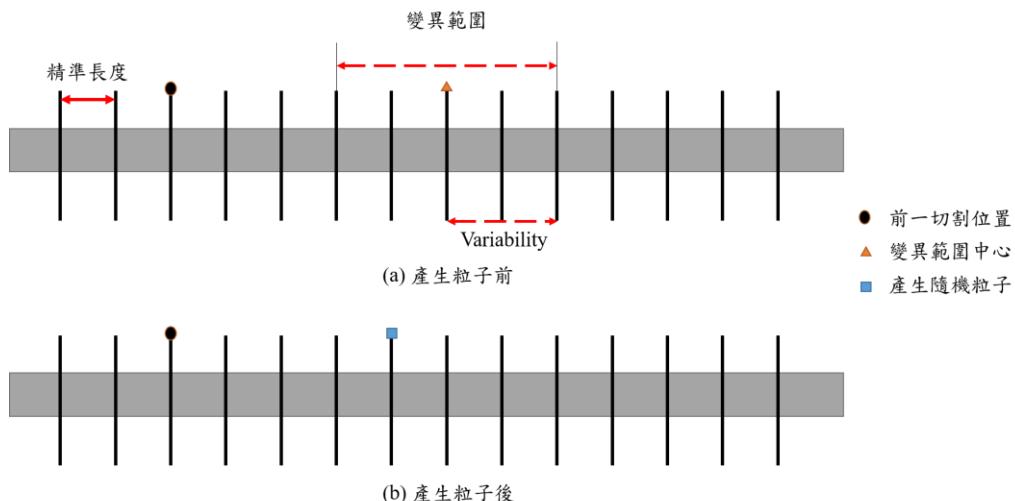


Figure 4.17 變動度對粒子影響(a)產生粒子前(b)產生粒子後

然而這變異度可能會造成在迭帶的過程中兩個切割位置的距離超過切割長度限制而違反 4.2 中的 Equation (6)(7)(8) 的限制條件，故若變動的位置與前一粒子的距離超過切割長度限制，則此切割位置會更新為最大切割長度(如 Figure 4.18 所示)。

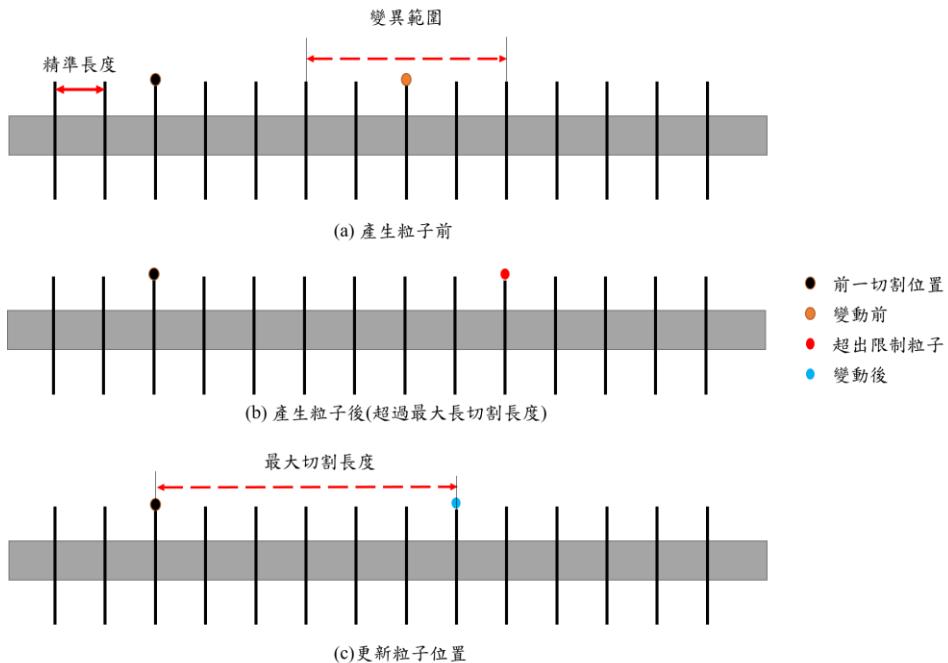


Figure 4.18 粒子超出最大長度限制則更新為最大長度

4.4.6 Global Best

本研究中仍然有使用到 Local Best 以及 Global Best 的概念，然而原先 Local Best 是用來找尋不同區域的區塊最佳解、Global Best 全體中最佳的解答，兩者都是一種「最終解答」。但在本研究中 Local Best 的主要目的為輔助找尋 Global Best 的功能，利用 Local Best 在不同區段中找尋該區段中的最小截面積位置，再利用變異度找尋這些最小截面積位置附近所構成的切割組合找出更加的 Global Best。

在演算法初始化時 Global Best 為多變數的陣列(double[n] , N=切割刀數)，且其中每一刀為最大化切割下的數據，而其中每一切割數據為該區段附近的 Local Best(double)(如 Figure 4.19)。而 Local Best 的更新條件為粒子在那該區域中所找尋的新切割位置小於原先的截面積則更新(如 Figure 4.20)。而 Global Best 更新的條件為當整體的切割截面積小於原先的截面積總和則更新，且同時所有 Local Best 也進行更新。(如 Figure 4.21)

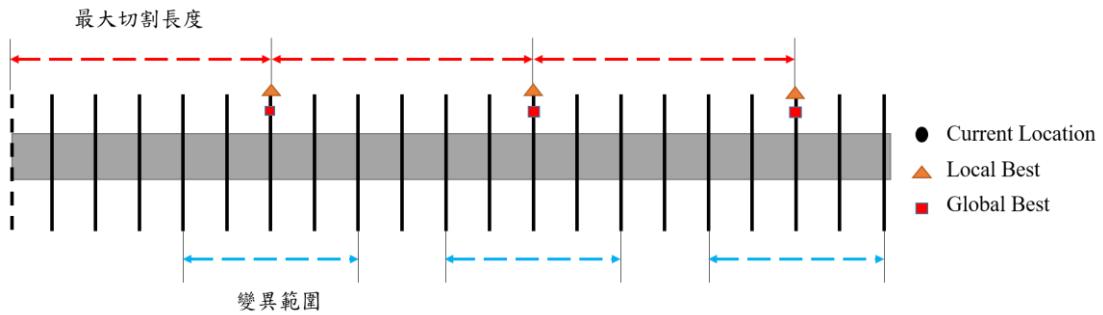


Figure 4.19 PSO 初始條件設定

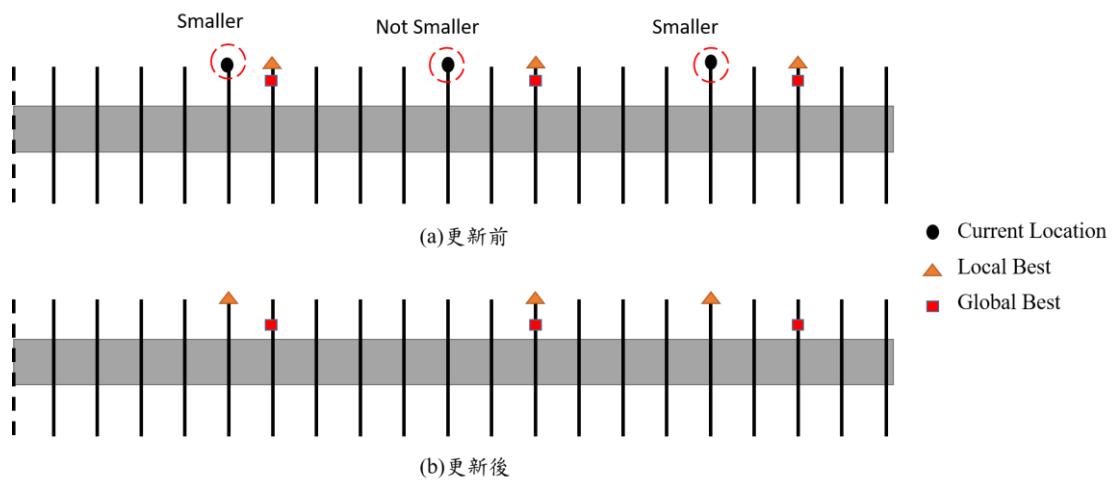


Figure 4.20 Local Best 更新方法

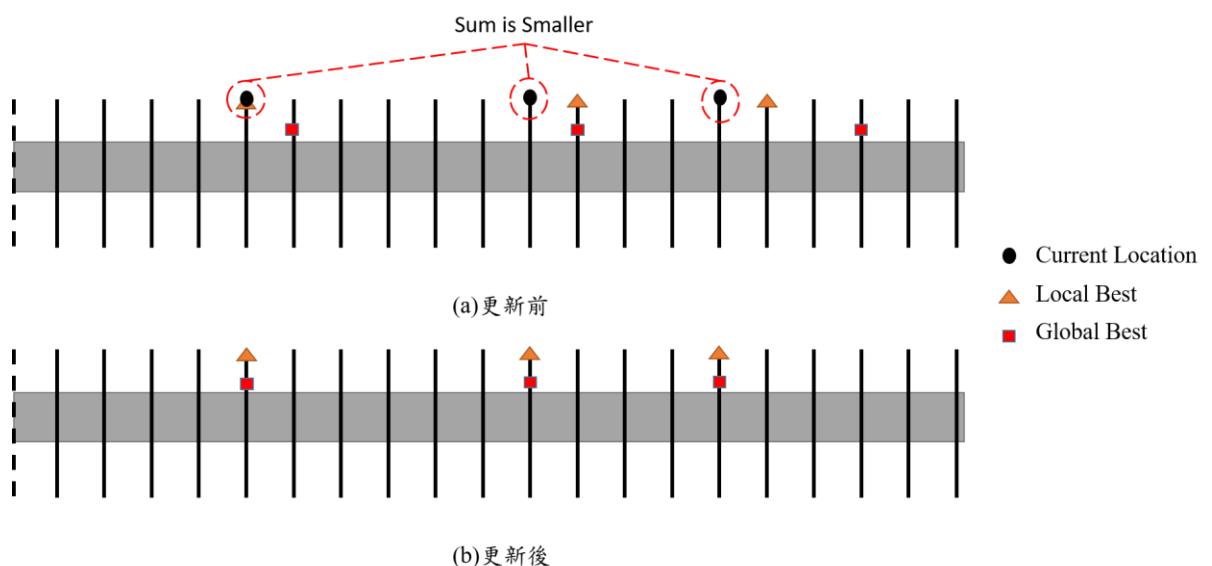


Figure 4.21 Global Best 更新方法

Table 4 演算法最佳化變數比較

	Global_Best	Local_Best
資料型態	Double Array	Double Variable
資料數量	1	N(切割刀數)
更新條件	總切割截面積降低	找到附近較小截面積

4.4.7 演算法整合

將上述的 PSO 敘述整合後可以得到 Figure 4.18 以及 Table 5 的 pseudo code

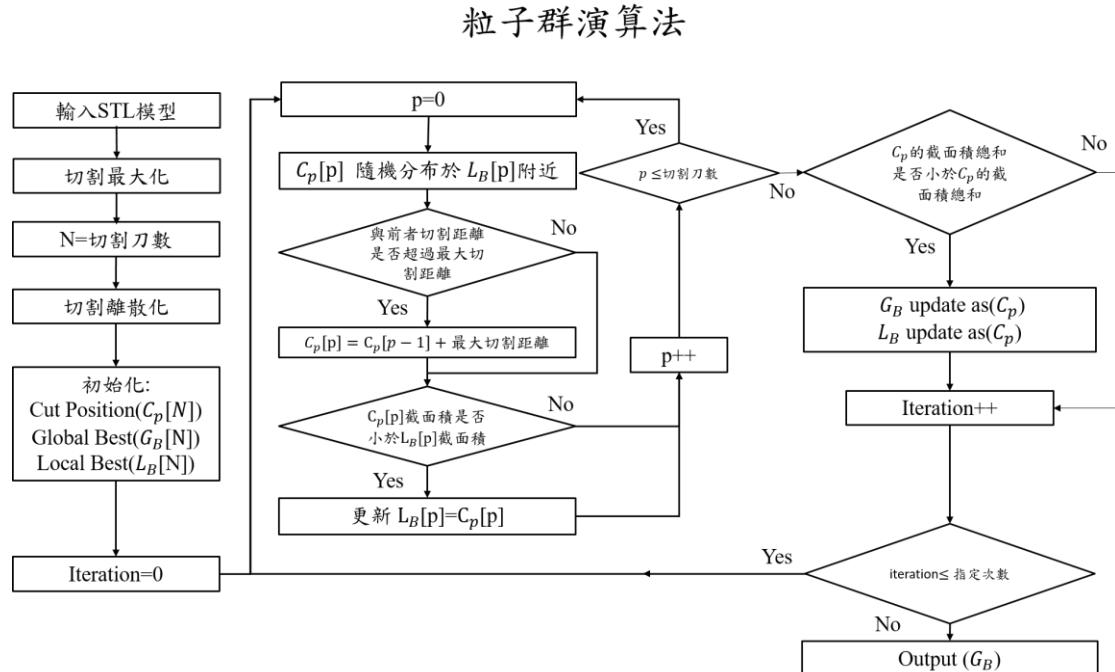


Figure 4.22 PSO 切割演算法設計流程圖

Table 5 PSO 演算法 pseudo code

Step	Command																
STL 建模	STL Model = Convert to STL (SolidWorks Model)																
計算最大切割長度	Max Cut len = Get Max Cut Length()																
使用者輸入演算法參數	Iterations = User Input Number() Data Accuracy = User Input Number()																
離散化模型	Model Discretize(Data Accuracy)																
演算法初始化	<table border="1"> <tr> <td>最大化切割模型</td> <td> $N = \text{int}(\text{Model Len} / \text{Max Cut Len})$ For $p = 1$ to N do: $\text{Cut Position}[p] = \text{Max Cut Len} * p$ </td> </tr> <tr> <td>定義 Global Best 定義 Local Best</td> <td> $\text{Cut Position} = \text{new double}[N]$ $\text{Global best} = \text{new double}[N]$ $\text{Local best} = \text{new double}[N]$ For $p = 1$ to N do: $\text{Cut Position}[p] = \text{Max Cut Len} * p$ $\text{Local best}[p] = \text{Cut Position}[p]$ $\text{Global best}[p] = \text{Cut Position}[p]$ End For </td> </tr> </table>	最大化切割模型	$N = \text{int}(\text{Model Len} / \text{Max Cut Len})$ For $p = 1$ to N do: $\text{Cut Position}[p] = \text{Max Cut Len} * p$	定義 Global Best 定義 Local Best	$\text{Cut Position} = \text{new double}[N]$ $\text{Global best} = \text{new double}[N]$ $\text{Local best} = \text{new double}[N]$ For $p = 1$ to N do: $\text{Cut Position}[p] = \text{Max Cut Len} * p$ $\text{Local best}[p] = \text{Cut Position}[p]$ $\text{Global best}[p] = \text{Cut Position}[p]$ End For												
最大化切割模型	$N = \text{int}(\text{Model Len} / \text{Max Cut Len})$ For $p = 1$ to N do: $\text{Cut Position}[p] = \text{Max Cut Len} * p$																
定義 Global Best 定義 Local Best	$\text{Cut Position} = \text{new double}[N]$ $\text{Global best} = \text{new double}[N]$ $\text{Local best} = \text{new double}[N]$ For $p = 1$ to N do: $\text{Cut Position}[p] = \text{Max Cut Len} * p$ $\text{Local best}[p] = \text{Cut Position}[p]$ $\text{Global best}[p] = \text{Cut Position}[p]$ End For																
演算法最佳化	<table border="1"> <tr> <td>產生 For 代數迴圈</td> <td>For i = 1 to Iterations do</td> </tr> <tr> <td>處理粒子</td> <td>For $p = 1$ to N do:</td> </tr> <tr> <td>移動粒子位置</td> <td>$\text{Cut Position}[p] = \text{Move Random Around}(\text{Local Best}[p])$</td> </tr> <tr> <td>限制粒子長度限制</td> <td>If $\text{Cut Position}[p] - \text{Cut Position}[p-1] > \text{Max Cut Len}$ $\text{Cut Position}[p] = \text{Cut Position}[p-1] + \text{Max Cut Len}$ End If </td> </tr> <tr> <td>更新 Local Best</td> <td>If $\text{Area}(\text{Cut Position}[p]) < \text{Area}(\text{Local Best}[p])$ $\text{Local Best}[p] = \text{Cut Position}[p]$ End If </td> </tr> <tr> <td>粒子處理完畢</td> <td>End For</td> </tr> <tr> <td>更新 Global Best</td> <td>If $\text{Sum of Area}(\text{Cut Position}) < \text{Sum of Area}(\text{Global Best})$: $\text{Global Best update all as}(\text{Cut Position})$ $\text{Local Best update all as}(\text{Cut Position})$ End if </td> </tr> <tr> <td>結束迴圈</td> <td>End For Return Global Best</td> </tr> </table>	產生 For 代數迴圈	For i = 1 to Iterations do	處理粒子	For $p = 1$ to N do:	移動粒子位置	$\text{Cut Position}[p] = \text{Move Random Around}(\text{Local Best}[p])$	限制粒子長度限制	If $\text{Cut Position}[p] - \text{Cut Position}[p-1] > \text{Max Cut Len}$ $\text{Cut Position}[p] = \text{Cut Position}[p-1] + \text{Max Cut Len}$ End If	更新 Local Best	If $\text{Area}(\text{Cut Position}[p]) < \text{Area}(\text{Local Best}[p])$ $\text{Local Best}[p] = \text{Cut Position}[p]$ End If	粒子處理完畢	End For	更新 Global Best	If $\text{Sum of Area}(\text{Cut Position}) < \text{Sum of Area}(\text{Global Best})$: $\text{Global Best update all as}(\text{Cut Position})$ $\text{Local Best update all as}(\text{Cut Position})$ End if	結束迴圈	End For Return Global Best
產生 For 代數迴圈	For i = 1 to Iterations do																
處理粒子	For $p = 1$ to N do:																
移動粒子位置	$\text{Cut Position}[p] = \text{Move Random Around}(\text{Local Best}[p])$																
限制粒子長度限制	If $\text{Cut Position}[p] - \text{Cut Position}[p-1] > \text{Max Cut Len}$ $\text{Cut Position}[p] = \text{Cut Position}[p-1] + \text{Max Cut Len}$ End If																
更新 Local Best	If $\text{Area}(\text{Cut Position}[p]) < \text{Area}(\text{Local Best}[p])$ $\text{Local Best}[p] = \text{Cut Position}[p]$ End If																
粒子處理完畢	End For																
更新 Global Best	If $\text{Sum of Area}(\text{Cut Position}) < \text{Sum of Area}(\text{Global Best})$: $\text{Global Best update all as}(\text{Cut Position})$ $\text{Local Best update all as}(\text{Cut Position})$ End if																
結束迴圈	End For Return Global Best																

第5章

SolidWorks 切割模擬

在上一節的群體粒子演算法中得到切割的位置數據後，最後是藉由 SolidWorks 實際模擬加工後的結果。故必須再次使用 SolidWorks API 將目標檔案分割成切割零件。

5.1 切割流程簡介

在前章節中說明利用 STL 檔以及演算法成功得到單一方向的切割數據，然而當工件的三維方向都超過包裝容器時，需要有三方向的切割工程才能完成整項工程，故在 API 切割工程模擬中必須要安排特定的切割流程順序才能得到實體的切割結果。故在 SolidWorks API 設計當中以 Y、X、Z 軸方向為順序切割，且每方向的演算法計算完畢後會先進行模型切割，之後再將切割後各個零件的 STL 檔進行尺寸驗證，再依章節 4.3.2 之方法決定下一方向的切割尺寸。

SolidWorks 切割模擬流程

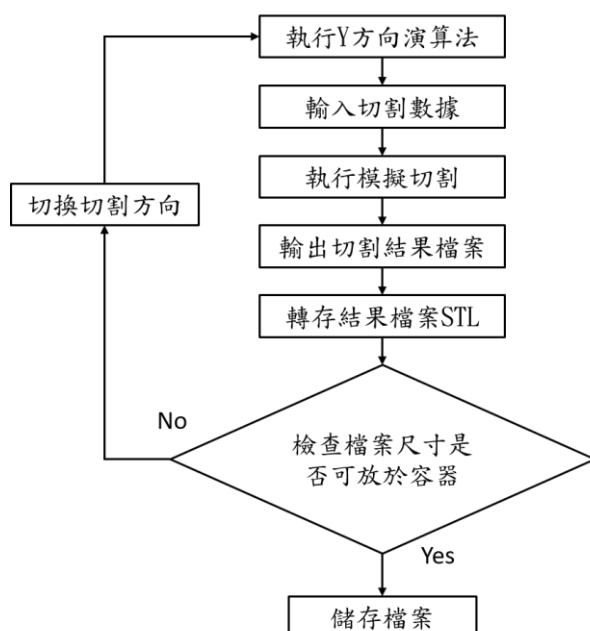


Figure 5.1 SolidWorks 切割模擬流程

5.2 切割模型方法

模擬切割的方法有需多種類，但若使用繁瑣的切割方式會導致在執行切割時耗費更多的時間。故本研究使用 SolidWorks 內建特徵「分割」為主要的切割方式，其運作的原理為再模型當中選出指定的切割平面(可複選)後，SolidWorks 會自動分割此模型並將所有分割零件儲存(如 Figure 5.2 所示)，如此一來可以省去許多草圖繪製以及除料特徵的程式運行時間。然而在分割的特徵中無法指定分割面的切割除料特徵，因此在輸出的檔案當中並非真實切割後的樣貌，但若模型的尺寸遠大於刀具切割厚度，除了可以忽略不計之外，在零件放入防輻射容器時可以保留些許的寬度以便置入。

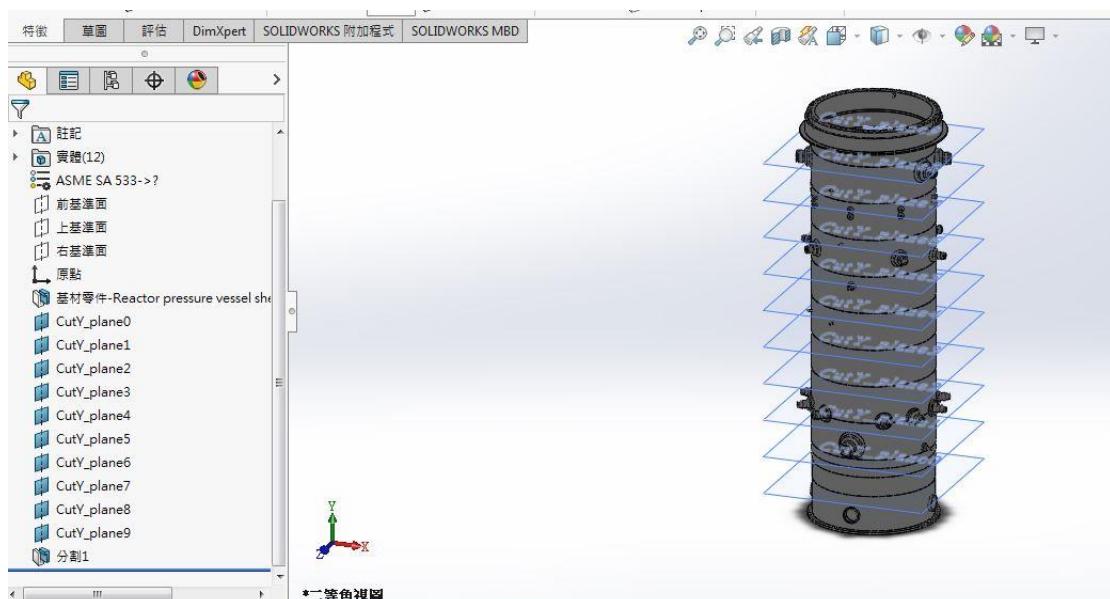


Figure 5.2 核反應爐 Y 方向切割

在有所有切割的平面標的後可以開啟分割功能，並將所有的切割平面標的以複選的方式選取，最後時可勾選編輯完畢的案件已完成自動分割(相關程式碼可參考附錄-附件三)。

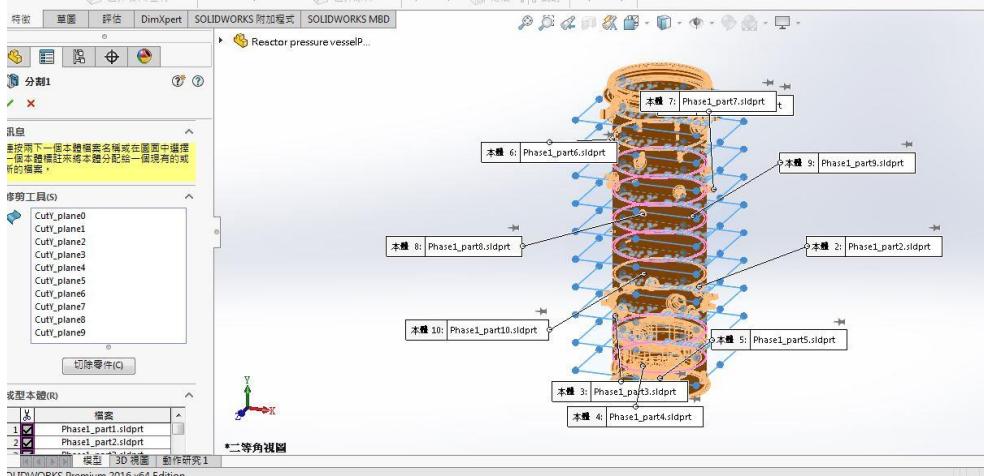


Figure 5.3 分割功能操作介面

6.1 人機介面設計

利用 Visual Studio C# 設計出本研究的人機介面，而其中所有的功能包括模型檔案的選擇、切割資料的輸入及切割資料的輸出與存檔，詳細的介面如 Figure 6.1 ~ 6.10 所示。

而在開始執行演算之前，必須先選取目標的切割檔案 CAD 檔，其中選去的檔案必須為.SLDPRT(SolidWorks 零件檔)，不然無法執行下一步。

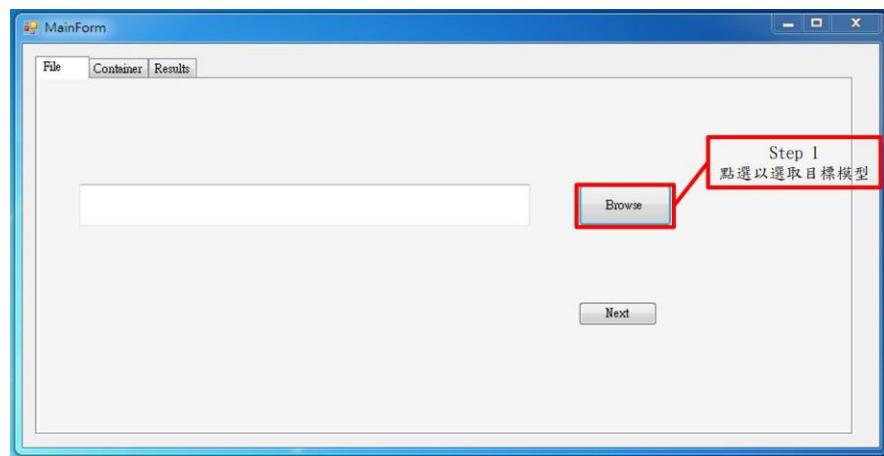


Figure 6.1 瀏覽模型檔案

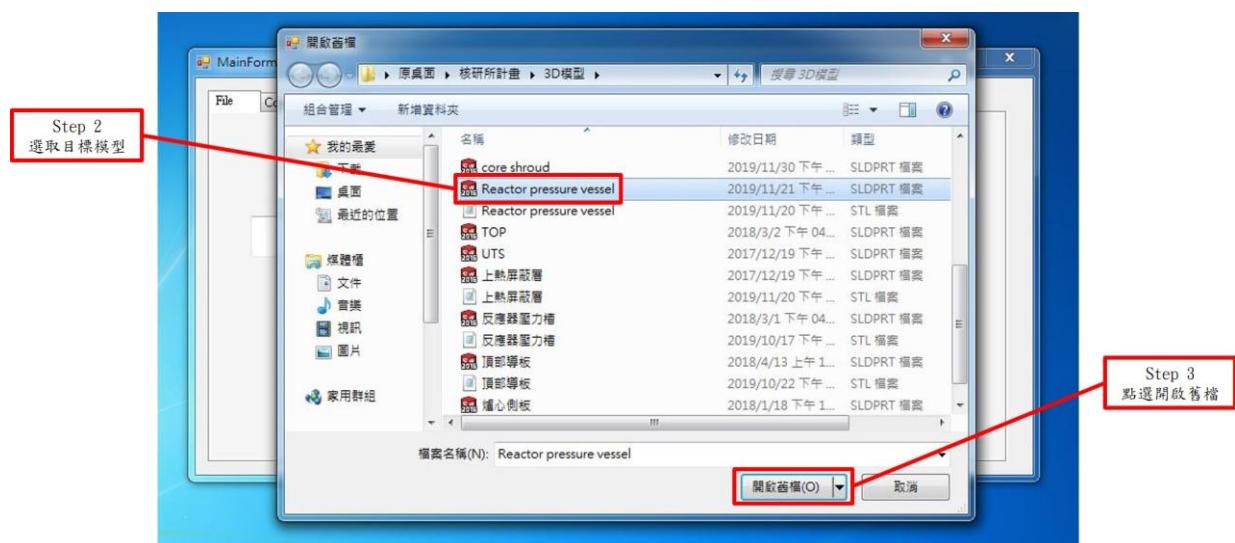


Figure 6.2 選擇目標切割檔案

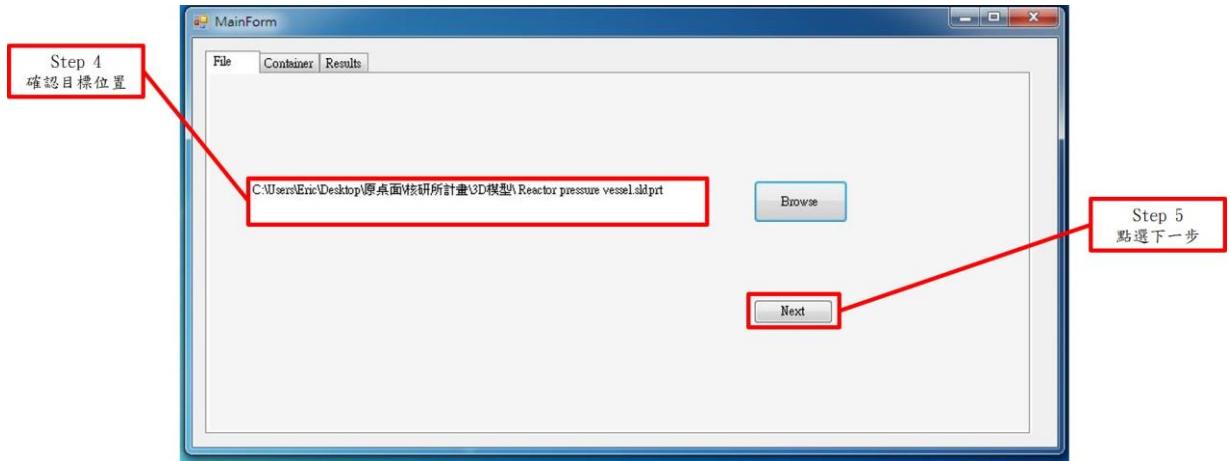


Figure 6.3 確認模型位置後點選下一步

而輸入切割資料會要求使用者輸入長方體容器尺寸，以及刀具厚度才能開始進行試算，而 PSO 演算法的執行迭帶次數以及切割精準度可以自行決定，較大的迭帶次數會讓運行時間增加，但結果輸出時的答案會較佳。而切割精密度則是可以依照不同的切割刀具以及容器尺寸大小來決定適合的切割精密度。

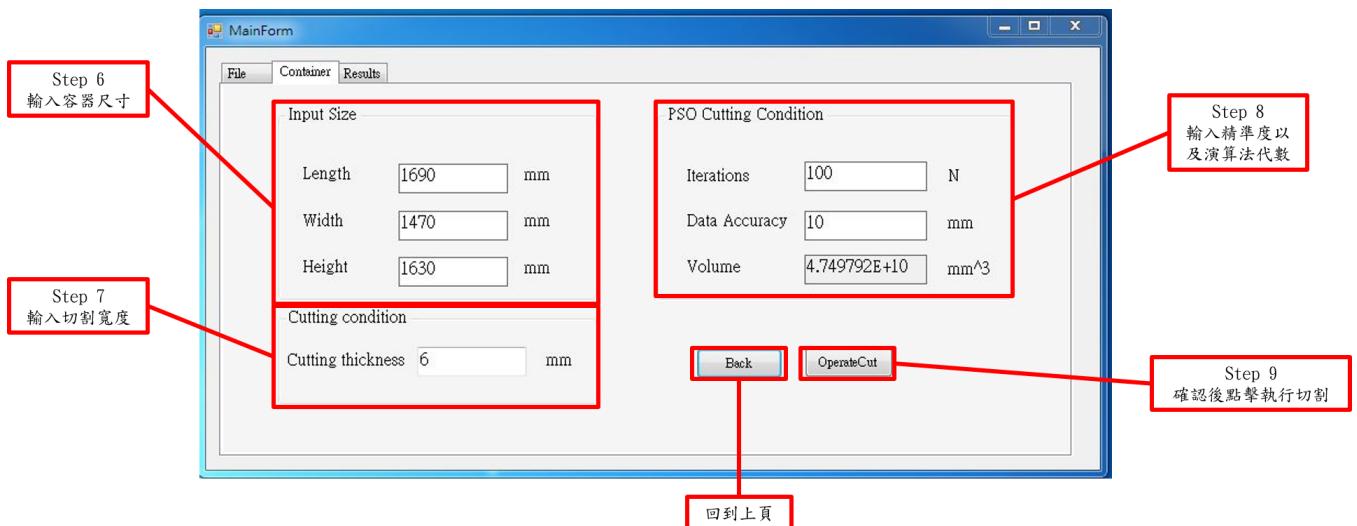


Figure 6.4 切割資料輸入介面

再輸入切割資料以及模型可以執行 OperateCut，會出現如 Figure 6.5 的運行視窗，而運行視窗中會顯示目前的切割方向以及處理中的檔案，已讓使用者知道目前的進度。

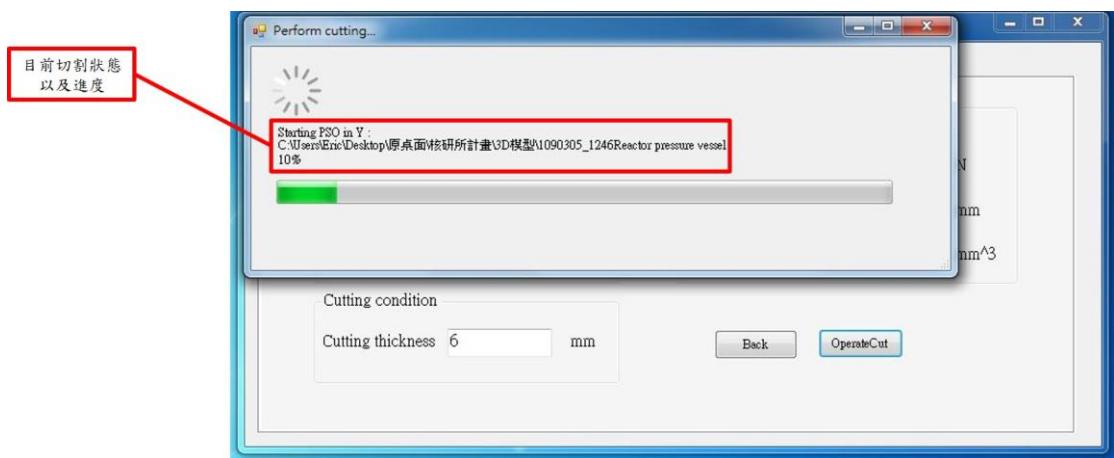


Figure 6.5 自動化切割執行狀態

運行完畢後即可以按下 Result 則會在下方 Data Frame 中產生所有零件的切割尺寸以及體積，並會在左方呈現切割後的零件數量以及最終所切割的廢料體積。而在 Result 的資料呈現完畢後會問使用者是否要打開切割檔案的資料夾，若回答是，則軟體會自動打開切割資料夾，以供使用者觀看切割 Excel 檔以及零件檔。若回答否則是不會，但可以藉由一旁的 Open File 再次開啟切割資料夾。

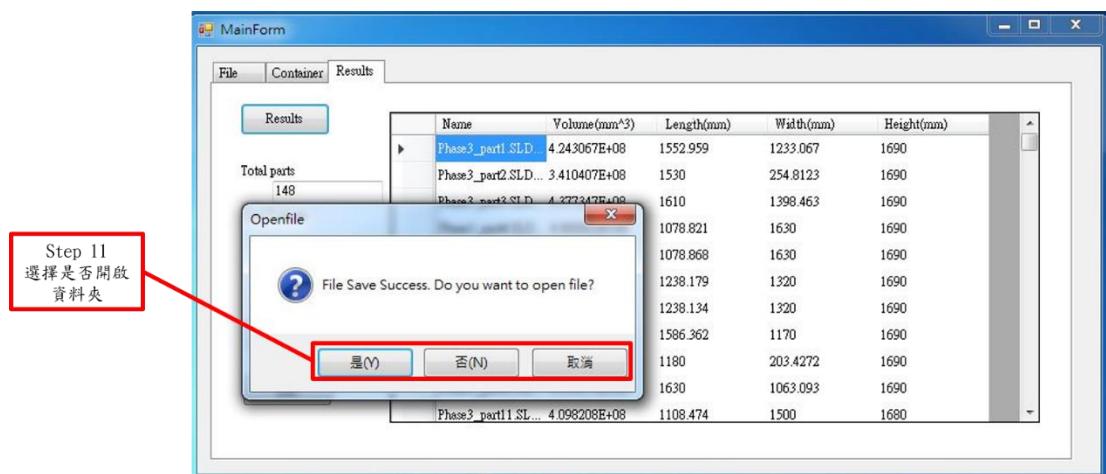


Figure 6.6 選擇是否開啟結果資料夾

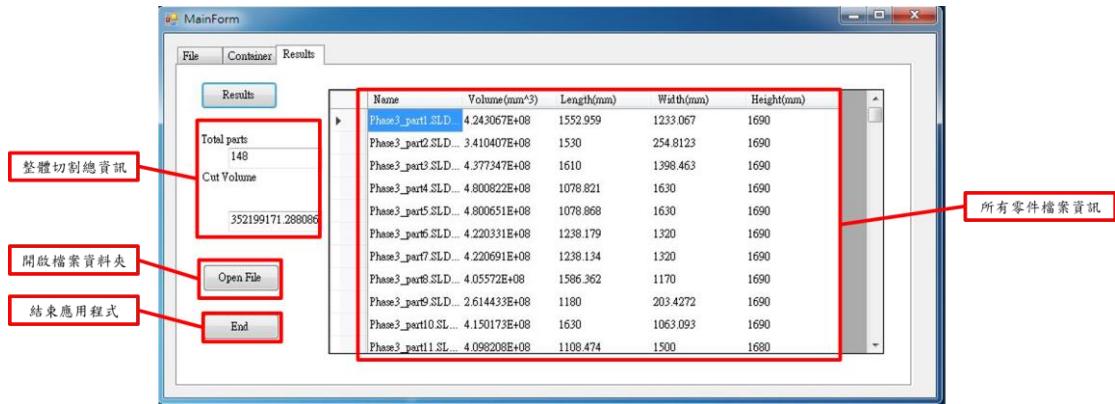


Figure 6.7 最終切割資料介面

在切割資料夾中會有四個不同的檔案，其中 Phase1 為在 y 軸方向的切割完畢後的檔案，而 Phase2 則是 Phase1 中每個檔案在 Z 軸方向切割的檔案，Phase3 則是最終 X 軸方向所剩的檔案。而 Cut_Data.xlsx 則是本次的切割模擬中每一個方向的切割收斂數據以及收斂圖，可以觀察此圖以確認演算法是否正常運作以及運作效率，而另外也會附上人機介面中的零件尺寸以及體積數據。

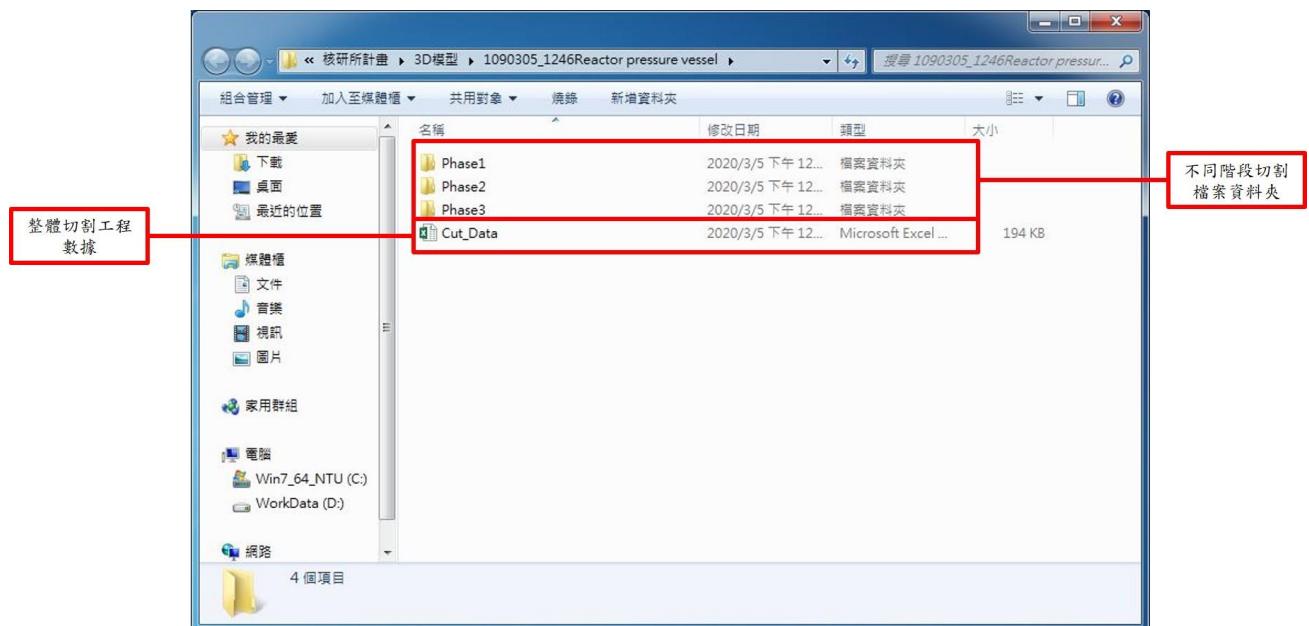


Figure 6.8 切割資料夾內容

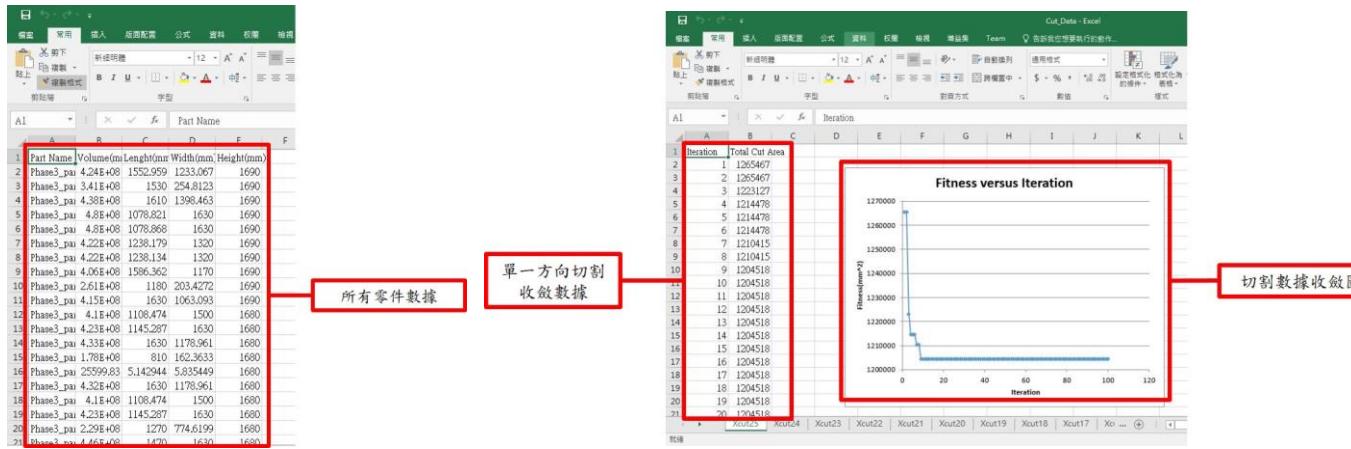


Figure 6.9 切割數據 Cut_data 內容

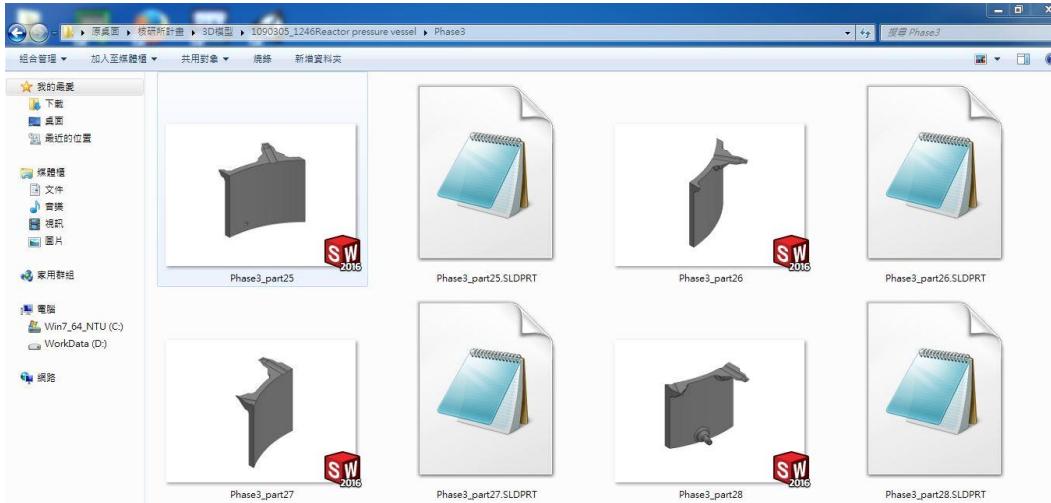


Figure 6.10 最終切割結果零件

6.2 模型切割結果展示

6.2.1 頂部導板切割結果

頂部導板的切割容器使用的是 1,000 mm x 1,000 mm x 1,200 mm 的容器，而使用的切割刀具厚度為 6 mm。而在演算法的參數設計中使用 100 次的迭帶次數，且切割精確度設為 10 mm。演算法執行的切割時間約為 35 sec，且切下的零件數量為 18，切削廢料的體積為 2,709,059 mm³。

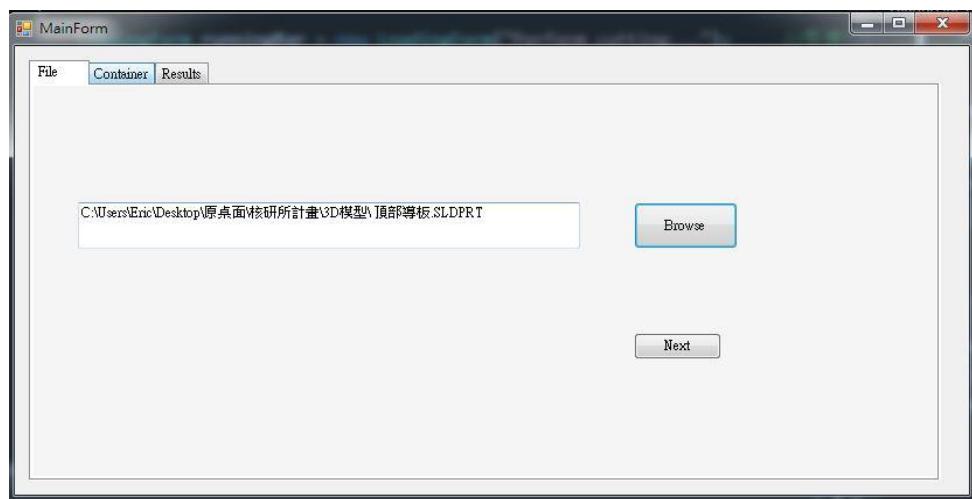


Figure 6.11 頂部導板檔案輸入

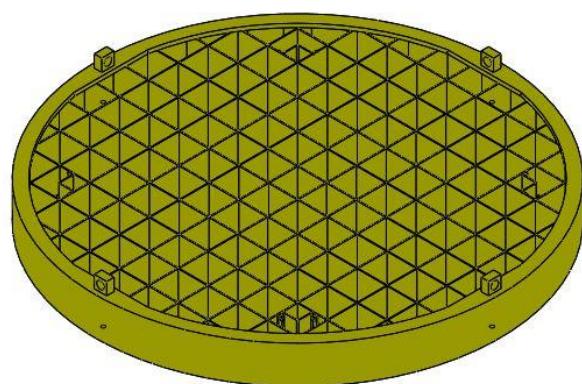


Figure 6.12 頂部導板 SolidWorks 模型

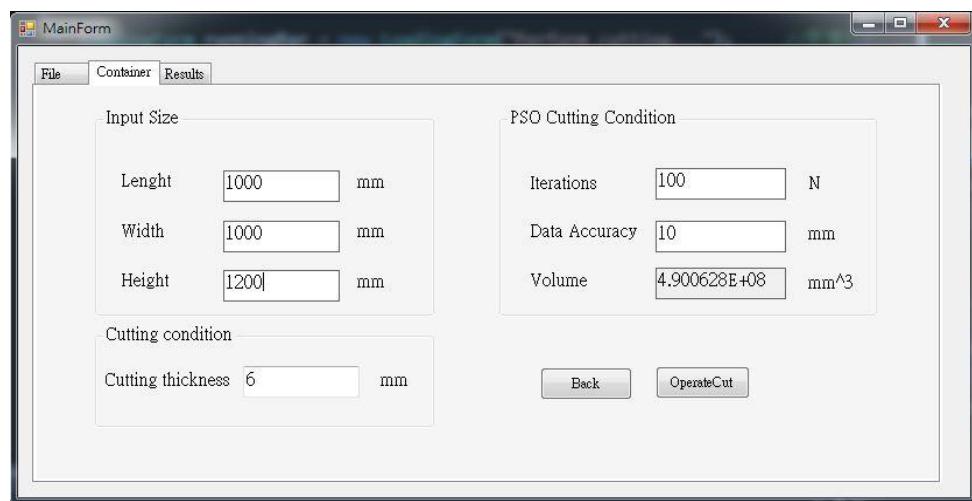


Figure 6.13 頂部導板切割參數

MainForm

File Container Results

Results

Total parts 18

Cut Volume 2709059.54882813

Open File End

Name	Volume(mm ³)	Length(mm)	Width(mm)	Height(mm)
Phase3_part1 SLD...	4.514183E+07	1000	1200	409.448
Phase3_part2 SLD...	9551474	500	1200	330.2
Phase3_part3 SLD...	2.1408E+07	900	1200	330.2
Phase3_part4 SLD...	1.59251E+07	690	1200	330.2
Phase3_part5 SLD...	81910.59	826001	50.4762	384.048
Phase3_part6 SLD...	4.431274E+07	980	1200	409.448
Phase3_part7 SLD...	1.711019E+07	850	1170	330.2
Phase3_part8 SLD...	1.944531E+07	760	1170	330.2
Phase3_part9 SLD...	2.516221E+07	950	1170	330.2
Phase3_part10 SLD...	3.491164E+07	684.3127	1170	536.448
Phase3_part11 SLD...	4.070518E+07	880.1118	1170	536.448

Figure 6.14 頂部導板切割結果

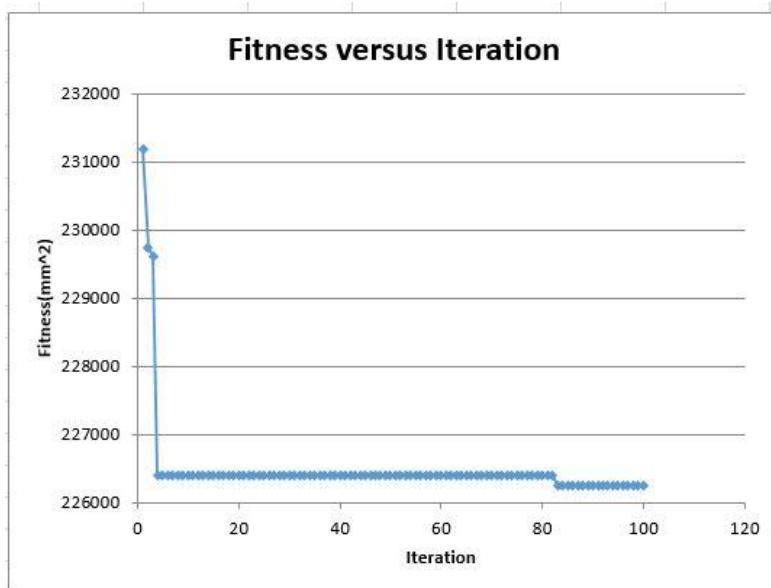


Figure 6.15 頂部導板切割截面積收斂圖

6.2.2 反應器壓力槽切割結果

反應器壓力槽的切割容器使用的是 1,690mm x 1,470mm x 1,630mm 的容器，而使用的切割刀具厚度為 6mm。而在演算法的參數設計中使用 100 次的迭帶次數，且切割精確度設為 10mm。演算法執行的切割時間約為 220sec，且切下的零件數量為 152，切削廢料的體積為 350,029,042mm³。



Figure 6.16 反應器壓力槽檔案輸入

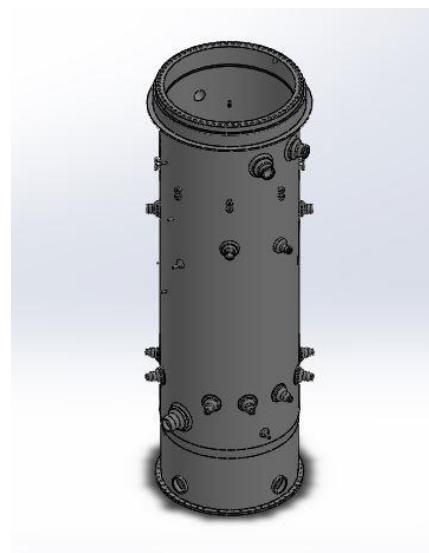


Figure 6.17 反應器壓力槽 SolidWorks 模型

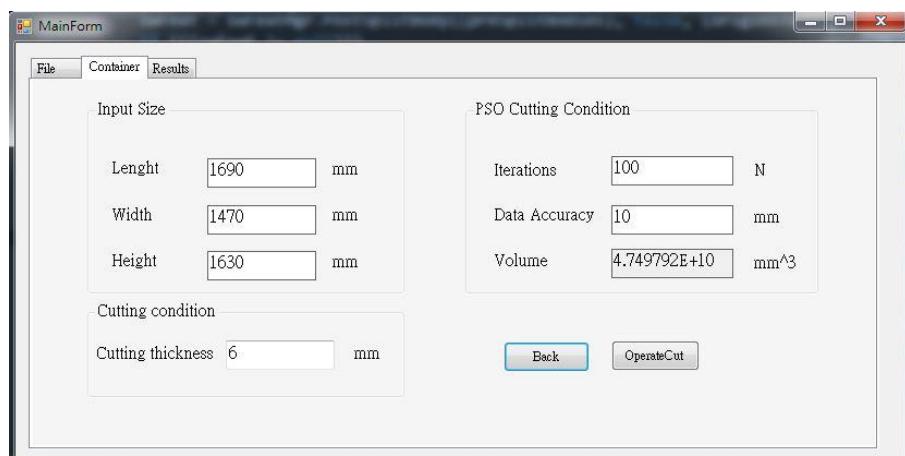


Figure 6.18 反應器壓力槽切割參數

	Name	Volume(mm³)	Length(mm)	Width(mm)	Height(mm)
Total parts	152				
Cut Volume					
	Phase3_part144 S...	6.553782E+08	1249.473	1630	1652
	Phase3_part145 S...	6.553823E+08	1249.473	1630	1652
	Phase3_part146 S...	6.510664E+08	1215.836	1630	1652
	Phase3_part147 S...	6.510417E+08	1215.836	1630	1652
	Phase3_part148 S...	5.016254E+08	1470	1389.221	1652
	Phase3_part149 S...	5.647937E+08	1470	932.1047	1652
	Phase3_part150 S...	4.511517E+08	1130	1069.105	1652
	Phase3_part151 S...	950243.8	146.6265	230.7911	54.60156
	Phase3_part152 S...	3.936884E+08	1360	1275.825	1652

Figure 6.19 反應器壓力槽切割結果

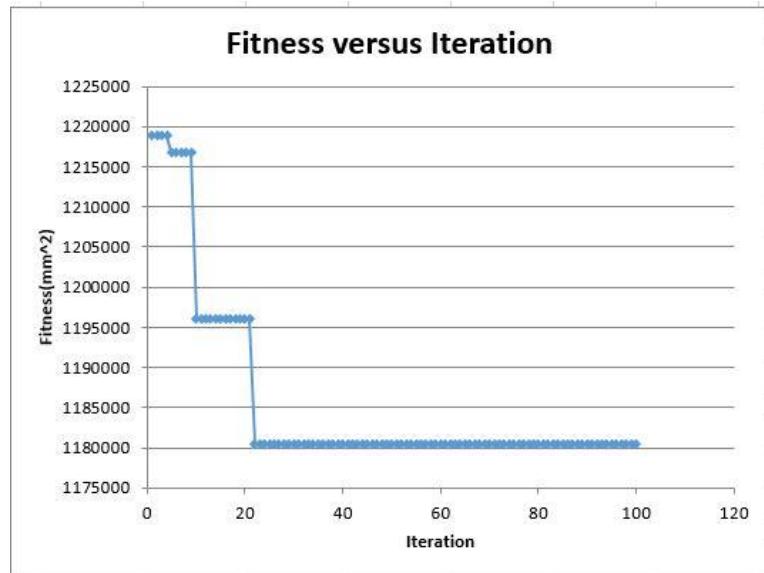


Figure 6.20 反應器壓力槽切割截面積收斂圖

6.2.3 爐心側板切割結果

爐心側板的切割容器使用的是 1,000mm x 1,000mm x 1,200mm 的容器，而使用的切割刀具厚度為 6mm。而在演算法的參數設計中使用 100 次的迭帶次數，且切割精確度設為 10mm。演算法執行的切割時間約為 170sec，且切下的零件數量為 96，切削廢料的體積為 44,561,995mm³。



Figure 6.21 爐心側板檔案輸入

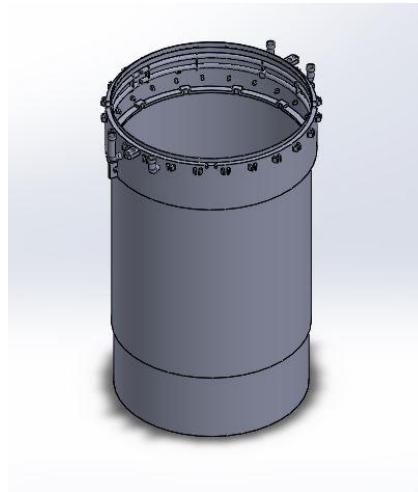


Figure 6.22 爐心側板 SolidWorks 模型

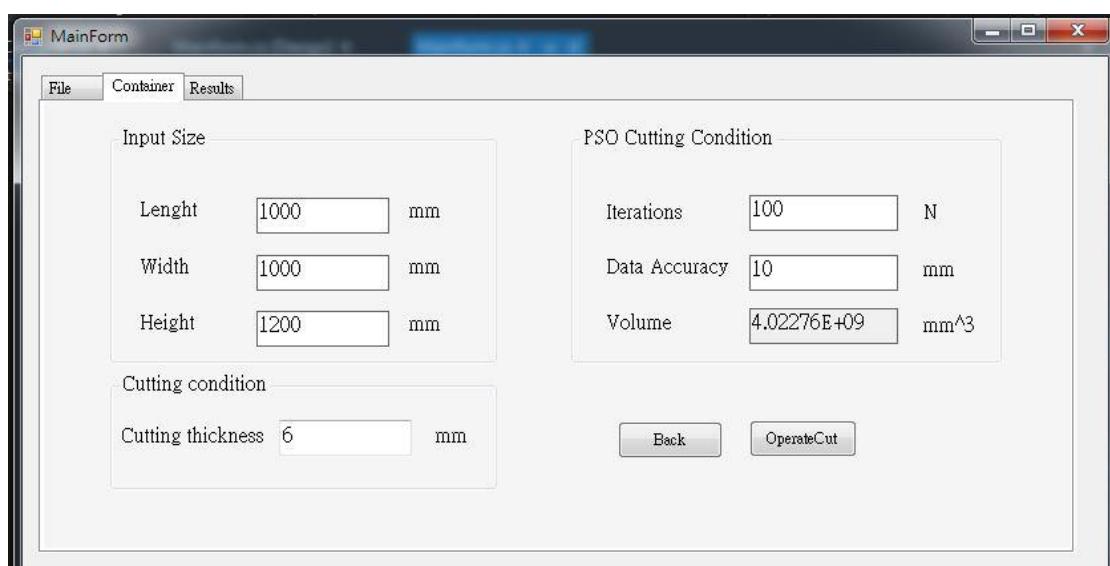


Figure 6.23 爐心側板切割參數

Name	Volume(mm³)	Lenght(mm)	Width(mm)	Height(mm)
Phase3_part1.SLD...	165376	64.86426	50.7207	72.2749
Phase3_part2.SLD...	4.179642E+07	740	360.504	1200
Phase3_part3.SLD...	6.008892E+07	999.8069	817.6748	1200
Phase3_part4.SLD...	5.365098E+07	940	379.3462	1200
Phase3_part5.SLD...	6.009963E+07	1000	817.7539	1200
Phase3_part6.SLD...	3.966983E+07	515.2577	640	1200
Phase3_part7.SLD...	3.966991E+07	515.2577	640	1200
Phase3_part8.SLD...	5.569634E+07	268.3916	1000	1200
Phase3_part9.SLD...	5.569678E+07	268.3916	1000	1200
Phase3_part10.SLD...	5.463706E+07	613.1575	890	1200
Phase3_part11.SLD...	5.463711E+07	613.1575	890	1200

Figure 6.24 爐心側板切割結果

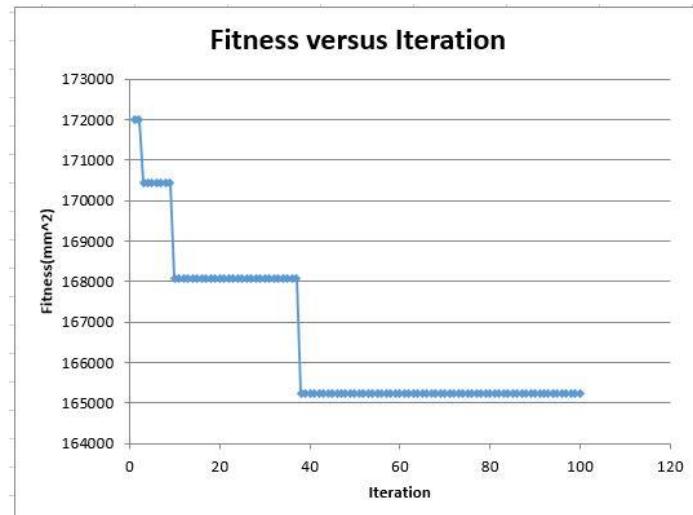


Figure 6.25 爐心側板切割截面積收斂圖

6.2.4 上熱屏蔽層切割結果

上熱屏蔽層的切割容器使用的是 1,690mm x 1,470mm x 1,630mm 的容器，而使用的切割刀具厚度為 6mm。而在演算法的參數設計中使用 100 次的迭帶次數，且切割精確度設為 10mm。演算法執行的切割時間約為 220sec，且切下的零件數量為 152，切削廢料的體積為 26,693,327mm³。

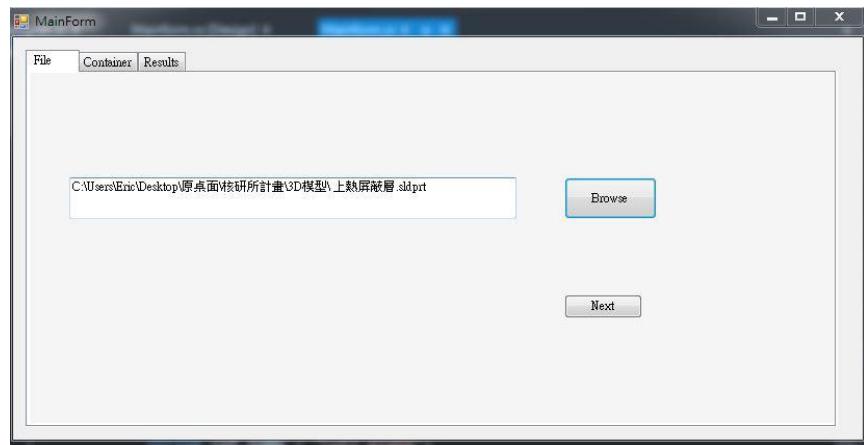


Figure 6.26 上熱屏蔽層檔案輸入

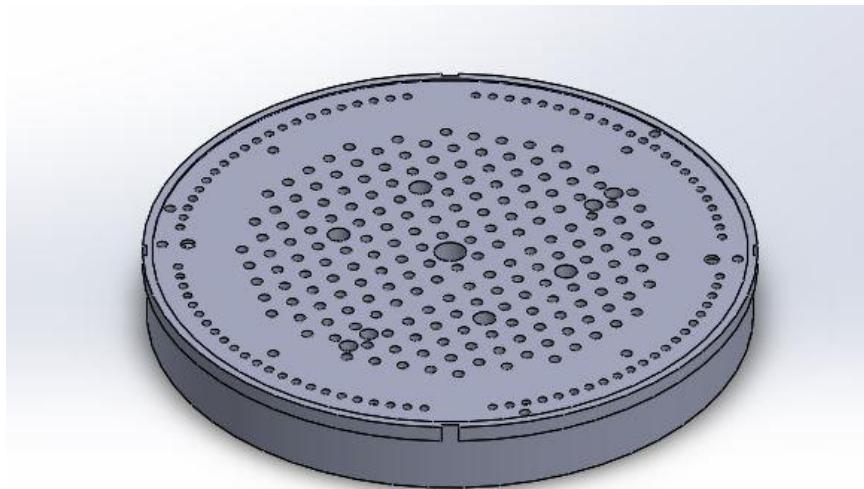


Figure 6.27 上熱屏蔽層 SolidWorks 模型

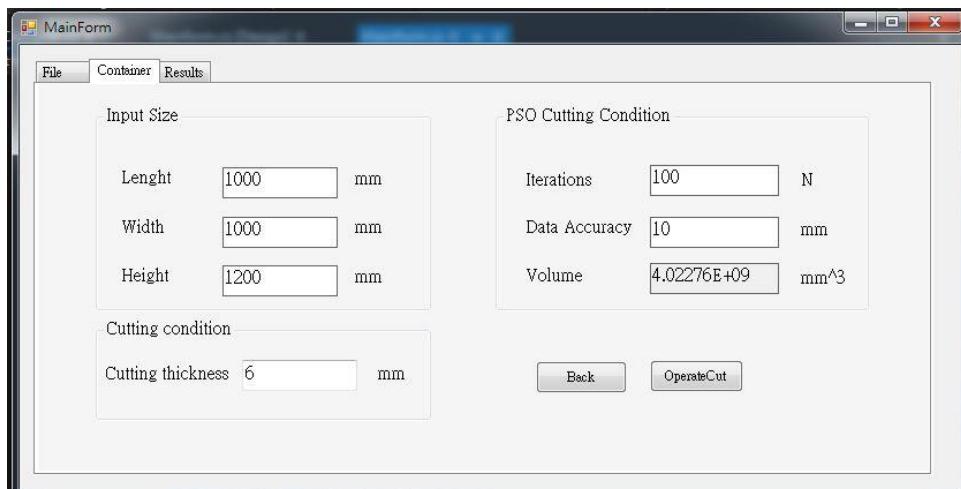


Figure 6.28 上熱屏蔽層切割參數

MainForm

	Name	Volume(mm ³)	Length(mm)	Width(mm)	Height(mm)
Total parts	Phase3_part1.SLD...	5.825661E+07	1000	243.4984	1280
46	Phase3_part2.SLD...	1.262686E+08	1630	1556.154	1280
Cut Volume	Phase3_part3.SLD...	256.0365	0.3710938	1.316833	404.9749
26693327.6015625	Phase3_part4.SLD...	1.291646E+08	1630	1680.721	1280
	Phase3_part5.SLD...	6.921825E+07	271.397	1190	1280
	Phase3_part6.SLD...	6.921894E+07	271.397	1190	1280
	Phase3_part7.SLD...	1.258429E+08	1565.552	1690	1280
	Phase3_part8.SLD...	6.228985E+07	1070	261.7194	1280
	Phase3_part9.SLD...	1.266343E+08	1630	1564.744	1280
	Phase3_part10.SLD...	1.411124E+08	1466.072	1529.797	1690
	Phase3_part11.SLD...	7.369848E+07	1130	117.9148	1690

Figure 6.29 上熱屏蔽層切割結果

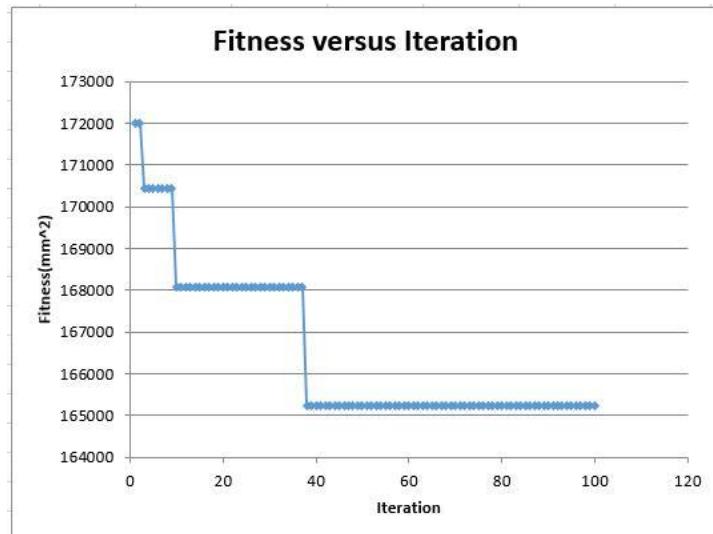


Figure 6.30 上熱屏蔽層切割截面積收斂圖

第7章 結論與未來展望

7.1 結果討論

本研究的宗旨為開發一個用於核反應爐組件切割的設計自動化與最佳化切割軟體，軟體設計過程中的 Visual Studio C# 編譯、STL 截面積計算、PSO 演算法設計、SolidWorks API 應用、以及最終的 Excel 資料輸出都是不可或缺的重要步驟。最終的軟體可以成功的做自動化以及最佳化切割規劃

7.1.1 優勢分析

1. 參數化設計：

參數化的切割設計，讓模擬工程能夠自由的選擇長方體容器尺寸以及切割刀具厚度，也能調整適合的切割精度以及 PSO 的執行迭代數。

2. 截面積讀取

本研究開發軟體的優勢在於讀取截面積不受模型特徵所影響，只要能將檔案存成 STL 模型都能夠進行截面積計算。除此之外 STL 的截面積計算比起利用 SolidWorks API 計算截面積快速許多。

3. 粒子群演算法

本研究所使用的粒子群演算法與傳統的不同，最大化切割首先減少演算法的變數量，之後的精準度以及變異度減少演算法的資料複雜度，對於整體的執行速度幫助不少。而在最佳化演算法 Local Best 的輔助功能增加最佳化演算法的收斂能力。

4. SolidWorks API 設計

SolidWorks API 的應用幫助在切割規劃時能夠自動化完成重複性高的動作，

其中包括產生切割平面、STL 檔案的自動輸出以及切割檔案等步驟，對於自動化設計提供許多便利性。

5. Excel 資料輸出

最終切割資料夾中的 Excel 檔內容有附上所有切割方向的演算法執行的收斂圖，方便使用者觀察演算法的執行是否正常運作。

7.1.2 改良空間

1. 切割容器種類少

本研究僅針對長方體容器做切割規劃，但在業界中所使用的切割容器種類變化多，僅針對長方體容器設計可能無法滿足業界需求。

2. 切割模式不夠彈性化

本研究中 STL 的截面積讀取僅限於 X 軸、Y 軸以及 Z 軸方向的切割，然而面對不同的模型時，卡式座標的切割規劃或許不是最佳的切割方法。以反應器壓力槽為例(Figure 6.17)，此模型的主要形狀為圓柱狀，利用極座標方式(如 Figure 7.1 所示)的切割或許有較佳的切割方式。模型在現實生活中可能透過轉動就能夠放入容器中，但在 STL 的數據中僅會得到模型的 XYZ 方向的最大長寬，故導致有些不需要切割的零件需要多餘的切割才能結束切割指令(如 Figure 7.2 所示)。



Figure 7.1 極座標切割反應器壓力槽

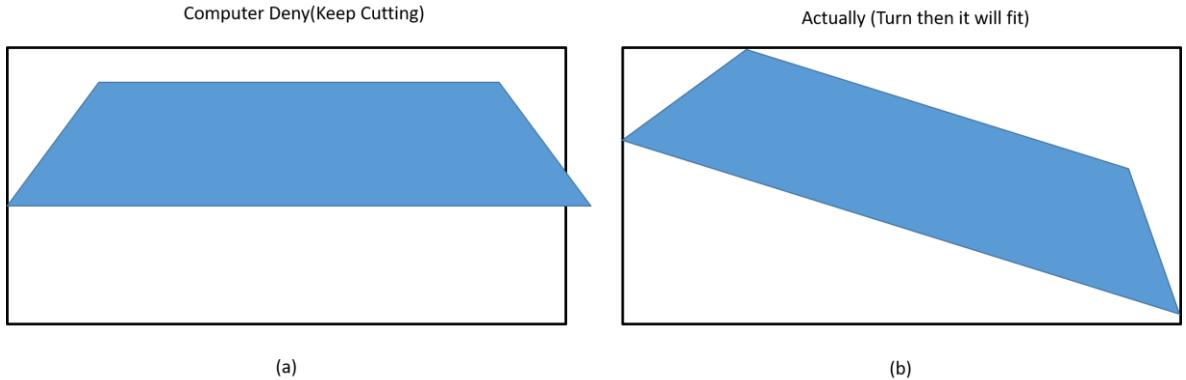


Figure 7.2 電腦卡式座標辨識是否可容於容器錯誤

3. 切割碎屑的產生

由於本研究所設計的演算法並沒有針對切割刀數進行限制，故導致有些邊緣有較窄小的零件會有多餘的切割(如 Figure 7.3 7.4 所示)，雖然可能在數學上這樣的方法的確有較小的總切割截面積，但對於實體工程來將仍需要考慮切割時間以及切割刀具移動成本，若將這些因素考量可能會導致這多餘的一刀成本反而較高。

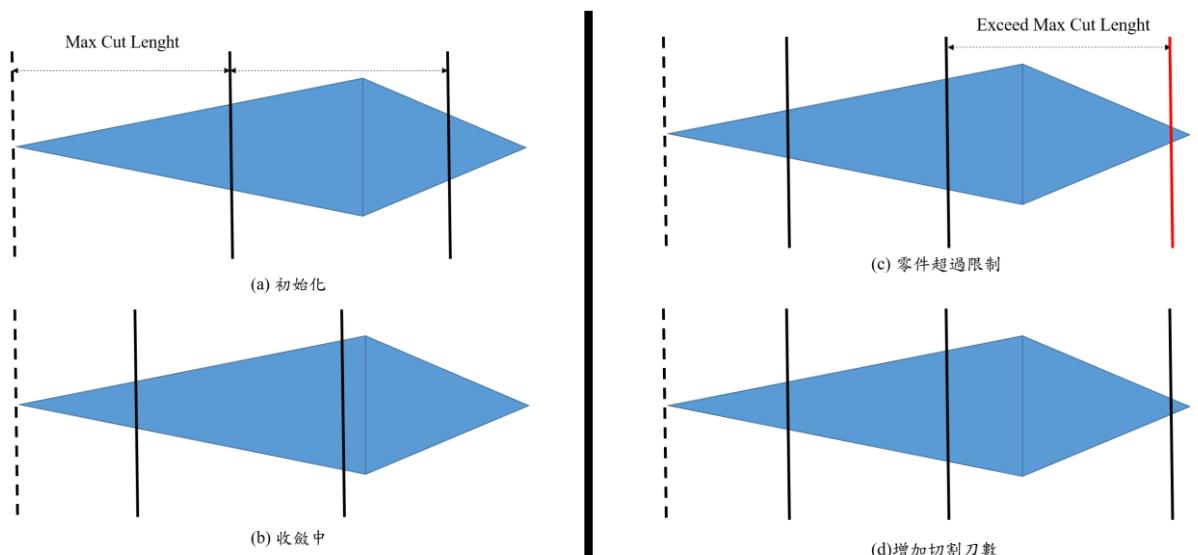


Figure 7.3 演算法碎屑產生示意圖
(a) 初始化 (b) 收斂中 (c) 零件超過限制 (d) 增加切割刀數

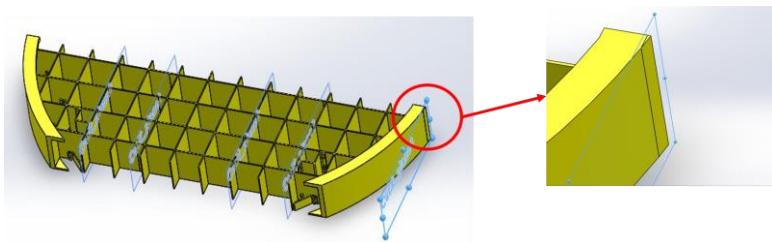


Figure 7.4 演算法切割多餘微小切削現象

7.2 未來展望

本研究在切割模式以及包裝容器上都僅有針對卡式座標以及長方形為主要的規劃方向，而導致最終設計出的軟體面對不同模型時表現不同。故在未來的研究開發方向中希望演算法能有更彈性化多元化的設計方向，主要的開發方向為：

1. 設計不同容器以擴張此切割模擬所能涵蓋的工程適用範圍，如圓柱狀容器利用圓柱直徑以及圓柱高度計算切割長度限制。
2. 將切割規劃多元化，例如設計極座標式的切割方式切割圓柱空心模型，甚至能夠將切割平面的法向量作為切割變數之一，以適應較特殊形狀的切割模型。
3. 針對是否以不需要切割做更深定義，如轉動 STL 座標系統並觀察轉動後是否可以符合容器尺寸，以避免多餘的切割浪費。
4. 將刀具移動納入成本考量或限制最切割刀數，以避免 Figure 6.3 的切割碎削現象發生

參考文獻

- [1] C.-Y. Pan, Ed. *Campbell Biology: Concepts & Connections, 8th Edition, Global Edition.*
- [2] H. K. Hilsdorf, J. Kropp, and H. J. Koch, "The Effects of Nuclear Radiation on the Mechanical Properties of Concrete," *ACI Symposium Publication*, vol. 55, 8/1/1978 1987.
- [3] M. L. Wald, "Dismantling nuclear reactors," *Scientific American*, vol. 288, no. 3, pp. 60-69, 2003. [Online].
- [4] S. Solstad and R. Van Nieuwenhove, "Instrument Capabilities and Developments at the Halden Reactor Project," *Nuclear Technology*, vol. 173, no. 1, pp. 78-85, 2011/01/01 2011.
- [5] D. Hyun *et al.*, "A methodology to simulate the cutting process for a nuclear dismantling simulation based on a digital manufacturing platform," *Annals of Nuclear Energy*, vol. 103, pp. 369-383, 2017/05/01/ 2017.
- [6] K. Jeong *et al.*, "Real-time assessment of exposure dose to workers in radiological environments during decommissioning of nuclear facilities," *Annals of Nuclear Energy*, vol. 73, pp. 441-445, 2014/11/01/ 2014.
- [7] T. Fujiwara, H. Minowa, and Y. Munesawa, "Development of dismantlement support AR system for nuclear power plants using natural features," vol. 43, pp. 1999-2004, 01/01 2015.
- [8] P.-C. Tsai, X. Huang, Y.-H. Hung, C.-H. Huang, and S. Smith, "The computer aided cutting planning of components using genetic algorithms for decommissioning of a nuclear reactor," *Annals of Nuclear Energy*, vol. 130, pp. 200-207, 2019/08/01/ 2019.
- [9] G.-Q. Dun and H.-T. Chen, "SolidWorks API Methods of Modeling for Seed Plate Based on VB," *Soybean Science*, vol. 31, no. 4, pp. 630-634, 639, 2012.
- [10] J. Li, L. Li, Z. Dong, and D. Song, "An Automatic Posture Planning Software of Arc Robot Based on SolidWorks API," *Modern Applied Science*, vol. 3, 06/19 2009.
- [11] S.-N. M. and M. Gy, "Analysis of STL files," *Mathematical and Computer Modelling*, vol. 38, no. 7, pp. 945-960, 2003/10/01/ 2003.
- [12] M. Eragubi, "Slicing 3D CAD Model in STL Format and Laser Path" *International Journal of Innovation, Management and Technology*, vol. 4, 2003.
- [13] X. H. Shi, Y. C. Liang, H. P. Lee, C. Lu, and L. M. Wang, "An improved GA and a novel PSO-GA-based hybrid algorithm," *Information Processing Letters*, vol. 93, no. 5, pp. 255-261, 2005/03/16/ 2005.

附錄

附件一：STL 讀檔程式碼

```
public PSO_GetCutX(BinaryReader read)           // Read STL
{
    read.ReadBytes(80);                         //略過前80個STL識別Bytes
    triagles = read.ReadInt32();                //讀取該檔案的三角形數量
    data = new float[triagles, 3, 3];
    normal = new float[triagles, 3];
    total_volume = 0;
    for (int t = 0; t < triagles; t++)
    {
        for (int dimension = 0; dimension < 3; dimension++)
        {
            float value = read.ReadSingle();          //讀取三角形法向量
            normal[t, dimension] = value;
        }
        read.ReadBytes(2);                         //略過結束Bytes
        for (int point = 0; point < 3; point++)      //讀取三角形的三個點
        {
            for (int dimension = 0; dimension < 3; dimension++)
            {
                float value = read.ReadSingle();
                data[t, point, dimension] = value;
                if (value > Max[dimension]) Max[dimension] = value;
                if (value < Min[dimension]) Min[dimension] = value;
            }
        }
    }
}
```

附件二: STL 截面積計算程式碼

```
public float GetSectionZ(float position) //取得截面積in Z
{
    for (int t = 0; t < triagles; t++)
    {
        float[] Intersect_Z = new float[2];
        float[] Intersect_Y = new float[2];
        int count = 0;
        for (int point = 0; point < 3; point++)
        {
            float x1, x2;
            x1 = data[t, point, 0];
            x2 = data[t, (point + 1) % 3, 0];
            if ((x1 - position) * (x2 - position) < 0) //檢查使否有通過截面
            {
                float z1, y1, z2, y2;
                z1 = data[t, point, 2];
                y1 = data[t, point, 1];
                z2 = data[t, (point + 1) % 3, 2];
                y2 = data[t, (point + 1) % 3, 1];
                //內差法取得交點 Z
                Intersect_Z[count] = (position - x1) / (x2 - x1) * (z2 - z1) + z1;
                //內差法取得交點 Y
                Intersect_Y[count] = (position - x1) / (x2 - x1) * (y2 - y1) + y1;
                count++;
            }
            else if ((x1 - position) * (x2 - position) == 0)
            {
                if ((x1 - position) == 0)
                {
                    Intersect_Z[count] = data[t, point, 2];
                    Intersect_Y[count] = data[t, point, 1];
                    count++;
                }
            }
            if (count == 2) { break; }
        }
    }
}
```

```

        }

float area = 0;
//法向量與中點向量內積

float same_direction = normal[t, 2] * (Intersect_Z[0] + Intersect_Z[1])/2 + normal[t,
1] * (Intersect_Y[0] + Intersect_Y[1])/2;

if (same_direction > 0)    //同向相加
    area += Math.Abs(Intersect_Z[0] * Intersect_Y[1] - Intersect_Z[1] *
Intersect_Y[0]) / 2;
}

else if (same_direction < 0)    //反向相減
{
    area += -Math.Abs(Intersect_Z[0] * Intersect_Y[1] - Intersect_Z[1]
* Intersect_Y[0]) / 2;
}

return area;
}

```

附件三: SolidWorks API C# 特定程式碼

程式用途	程式
開啟程式	<pre> SldWorks swApp; //建立 SolidWorks swApp = new SldWorks(); //開啟 SolidWorks swApp.Visible = false; //是否顯示 swApp.UserControl = false; //電腦操作模式 </pre>
開啟零件	<pre> swModel = swApp.OpenDoc2(fileName, (int)swDocumentTypes_e.swDocPART, false, false, true, ref istatus); //Solidworks 開啟選取檔案 swModelDocExt = swModel.Extension; //指定檔案函式庫 swFeatMgr = swModel.FeatureManager; //指定特徵函式庫 swModel.ClearSelection(); //清除所有選取工作 </pre>
轉存 STL	<pre> string stlFile = \${Path.GetDirectoryName(dlg.FileName)}\\ " + Path.GetFileName(dlg.FileName) + ".STL"; //改變檔案格式 swModel.SaveAs3(stlFile, 0, 0); //另存新檔 </pre>
產生平面	<pre> boolstatus = swModel.Extension.SelectByID2("上基準面", "PLANE", 0, 0, 0, false, 0, null, 0); //選取原點對應方向平面 swModel.CreatePlaneAtOffset3(Math.Abs(Ycut[i] / 1000), false, false); //產生平面 </pre>
分割特徵	<pre> Feature swFeat = null; //新增特徵 SplitBodyFeatureData swSplitBodyFeat = null; //新增分割特徵 swFeat = swFeatMgr.PostSplitBody((preSplitBodies), false, (originsToUse), (vBodyNames)); //產生分割特徵 </pre>