行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※

※　　　　　　　　　　　　　　　　　　　　　　　※

※　　　　　　高階飛行動態模擬器之研製　　　　　※

※　　　　　　　　　　　　　　　　　　　　　　　※

※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：□個別型計畫　　■整合型計畫

計畫編號：NSC-93-2622-E-009-016-

執行期間： 93 年 01 月 01 日至 94 年 07 月 31 日

計畫主持人：林進燈　　教授　　交通大學

共同主持人：楊谷洋　　教授　　交通大學

陳甘霖　　副教授 和春技術學院

本成果報告包括以下應繳交之附件：

□赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

□出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學電機與控制工程研究所

中　華　民　國　94 年　10 月　28 日

摘要

本產學計畫「高階飛行動態模擬器研製」是延續前三年前所提之產學案「即時動態模擬系統研製」之成果，主要目標發展一符合美國飛航單位所制訂之飛行模擬訓練機標準的即時動態飛行模擬系統。在本產學計畫三年的執行過程中，我們基於目前的技術核心，進一步擴展原系統之機構、控制、動態模擬、力搖桿等各項技術，達成一飛行模擬訓練機之水準。其中包括五個研究子題協力共同進行，詳細的內容請見各子題的成果報告。

（一）六軸動作平台架構改進、分析與控制

六軸動作平台為整合系統之動作展示核心，我們為了符合一飛行模擬器之水準及產業需求，在此方面花費很多心力於機構的探討、分析與控制。由於電動平台之動態方程式十分複雜，不易使用傳統的控制方法來達成良好的控制目的，既使花費長時間去嘗試找出一組合適的控制參數，也往往因為機構上之負載變化而有控制不佳情狀發生，有鑑於此，本子題研究利用適應性控制理論與小腦模型控制器設計方法，設計了一架構簡單、具快速學習且不需要事先知道受控系統之動態模型的控制方法。

（二）動態模擬單元與虛擬實境場景之建構

本子題負責動態模擬單元與虛擬實境場景的建構，由於此兩部份對擬真的效果上佔著極重要的地位，且工作上可將這兩部份放在同一台電腦或程式執行，故我們由同一子題負責此兩部份的進行。此外由於本產學案目的為發展一符合飛行模擬訓練機，因此本子題將以「飛行模擬」為研究核心，目標進度包括動態模擬單元部份的各種飛行狀態的模擬，如最常見的起飛、降落、碰撞、亂流、爬升…等，都要能適切地模擬出真實的飛行感受；以及虛擬實境場景部份的周圍環境、飛機駕駛艙的繪製、碰撞測試、音效播放…等功能，要盡可能在畫面上符合真實情況。我們完成了資料收集與研讀，並針對國軍二代戰機F-16作次音速的風洞數據模擬，運用數值式的方法建立飛行動態模，並且完成多螢幕的環繞場景增加虛擬實境的擬真度。

（三）飛機的力資訊處理與操控器研發

本子題的重點在於提供操作者在使用飛機模擬訓練系統時，能感受到彼此互動間接觸力的感覺，並發展能處理力訊號的控制策略以及操控器，以利使用者進行更具真實感、更有效的操縱。由於現代飛行技術的提升，隨著因應各種不同用途而被生產出來的飛機也越來越多，駕駛員的實機訓練越來越不容易，且實機飛行訓練有一定的危險性存在，因此虛擬實境飛行模擬提供了訓練飛行駕駛員的良

好解決方案。虛擬實境飛行模擬其實就是利用一套電腦所建立的虛擬飛行場景，利用栩栩如生的風景與擬真的飛機動態、聲光震撼，讓人有身歷其境的飛行感受。藉由地面的飛行模擬，不但可以節省實機飛行的花費，也可以保障人機的安全，並且達成訓練飛行駕駛員的目的。

（四）六軸動作平台控制卡之研發設計

基於產業上實際的需求，在這部份我們以上個產學案所開發出的 DSP 控制卡為基礎，進一步改版設計，增加 CAN Bus 的網路功能以及加上六軸動作平台的驅動電路，並希望能縮小硬體電路的總體積。為了提高六軸運動平台的穩定度與安全性，控制系統對外界訊息的反應必須要更為迅速，以確保在系統出現問題時能立即做出適當的處理。因此，在六軸運動平台控制的改善上，為了讓系統呈現出更完善的模擬效果，我們以 StrongARM 之嵌入式系統，並以 Linux 為作業系統設計週邊電路及為其撰寫驅動程式，在既有的控制系統中，加入一個即時作業系統，並分析與設計整個即時控制環境的架構以及實現了整個六軸平台即時控制系統。

（五）高階分散式網路架構建立與應用

模擬器是個整合電腦圖學、機械控制、動力學運算及虛擬環境的系統。為讓使用者有『身歷其境』感覺，系統需能即時地處理圖形、影像及聲音等運算。隨著虛擬場景複雜度的增加，及使用者要求更逼真、更即時的虛擬環境模擬系統，這些需求需要高效率運算能力的電腦來支援才能達到，故早期之研究多侷限於使用大型工作站級以上或是特殊用途的電腦，但是這樣的設備都很昂貴，不是一般使用者所能負擔。然而隨著科技的日新月異，現今個人電腦和工作站的運算速度越來越快、價格也越來越低廉，因此將多台個人電腦或工作站透過區域網路的結合，形成一大型的分散式運算環境，也同樣可達高效率的運算能力。分析一個模擬器電腦系統的工作，包括圖學顯像、輸入介面、音效、虛擬場景資料庫管理、真實模組、預測模組等。而我們打算採用的設計理念係以美國國防部高階模擬架構(High-Level Architecture, HLA)之執行架構(Run-Time Infrastructure, RTI)觀念為底層架構，並實際以數台個人電腦構成模擬器電腦系統，以平行運算方式使整個模擬器電腦系統能達到即時的要求(顯像頻率為每秒 20~30 個畫面)，進而符合美國 FAA 標準。並且將發展的環境加以標準化及模組化，使得軟體可使用率提高，在發展六軸平台的相關軟體上可逐漸減少開支。

除了上述研究進度與成果皆達到原先計畫進度之規劃外，本研究團隊跟配合廠商亦有密切的配合與技術交流，如六軸電動平台之架設工作與高階分散式網路架構之商業運用等範圍討論。本研究團隊同時也積極參與國內外之學術研究工作，將本產學計畫所開發之技術文件投稿至相關的期刊與研討會，目前已有三篇

研討會論文被接受刊登，另有數篇論文還在審查階段。另外，本研究團隊於 2002 年也有參加教育部所舉辦的「九十一年度精密機電設計科技教育改進計畫-全國學生專題實作競賽」，亦獲得不錯的佳績，相關部分資料可在補充資料中找到。

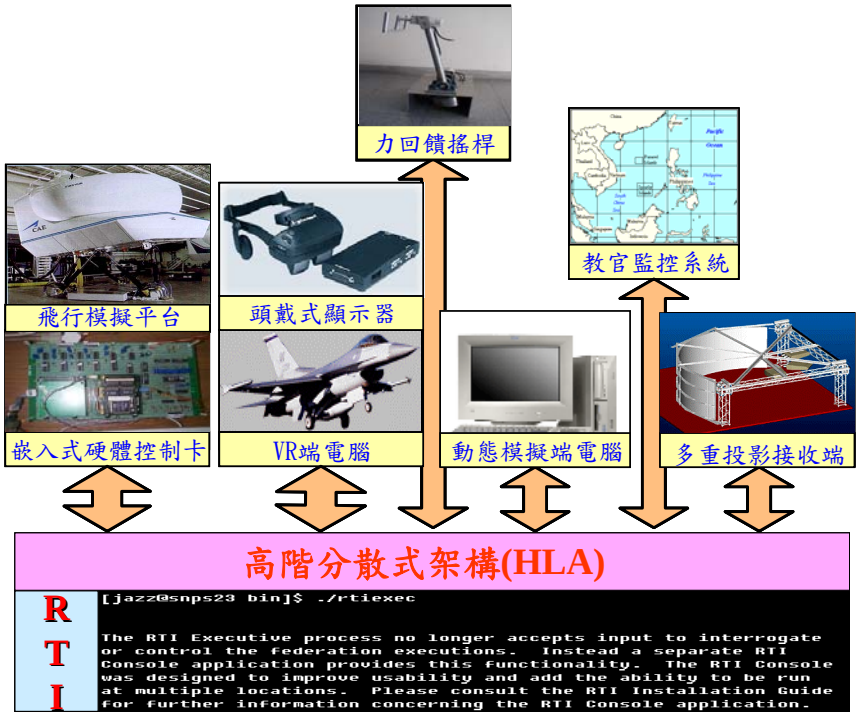


圖 1　飛行動態模擬系統完整架構圖

子題一：六軸動作平台架構改進、分析與控制

相較於本研究團隊現有的油壓運動平台，本子題所開發的電動運動平台，具有低噪音、低污染(無油污漏油的問題)與易維修等優點，甚至只需家用 110V 電源即可驅動馬達的轉動使整個電動平台正常運作。基於以上這些優點，目前國外製作運動平台的公司，也都已經陸續開發出電動平台以取代油壓平台，而國內外幾個相關的研究單位對於運動平台的需求，也都將注意力轉移至電動平台開發上。本子題為了加速讓國內業者具備電動平台開發的能力，我們與本計畫之合作廠商一同協力開發電動平台，合作廠商主要負責平台的機構設計與製造，而本子題主要負責分析與控制研究部分。

由於電動平台之動態方程式十分複雜，不易使用傳統的控制方法來達成良好的控制目的，既使花費長時間去嘗試找出一組合適的控制參數，也往往因為機構上之負載變化而有控制不佳情狀發生，有鑑於此，本子題研究利用適應性控制理論與小腦模型控制器設計方法，設計了一架構簡單、具快速學習且不需要事先知道受控系統之動態模型的控制方法，本子題命名為適應性小腦模型控制器，其包含一小腦模型類神經網路控制器與一補償控制器，小腦模型類神經網路控制器主要用來近似一理想控制器，而補償控制器用來補償小腦模型類神經網路控制器之學習誤差，最後經由實驗結果驗證，我們利用此完全 model-free 的控制設計方法可以用來解決電動平台的控制問題。

1.1 電動六軸運動平台設計與開發

在電動六軸平台的開發上，本計畫之合作廠商已完成電動運動平台的機構設計、加工及驅動系統架設工作，整個電動六軸平台如圖 1-1-1 所示，但因六軸平台在分析與設計上十分複雜，而且整個六軸平台開發工作有些延遲，所以本子題先使用單軸的電動缸進行測試驗證所開發設計之控制效能，由於本子題所設計之控制方法因不需事先獲知受控系統之數學模式，所以預計將依舊可以很精確地控制整個平台動作。以下文章將探討各項硬體設備、角色功能，再以電動平台位置控制系統架構圖來說明各元件於整個平台控制中所扮演之角色，最後則描述我們為了修改運動平台之控制參數利用 Borland C++ Builder 5.0 所發展之人機界面，透過這個人機界面可以讓使用者更容易修改控制參數、儲存資料及察看結果。

圖 1-1-1　電動六軸運動平台

1.1.1 馬達本體

設定所欲建構的電動平台載重為 1000 公斤，最後再乘以 2 倍之安全係數得單軸所需載重為 470 公斤，經由計算結果得知所需負載轉矩為 4.25 *Nm* 與慣性矩為 18.6 $Kg \cdot cm^2$。最後經查閱三菱 MELSERVO-J2_SUPER 規格書選定對應之馬達類型為 HC_SFS_153B (2000rpm/min) ，如圖 1-1-2 所示，而此馬達規格如下所列

額定輸出-1.5KW
額定轉速-2000rpm
額定電流-9A
額定轉矩-7.16N-m
速度/ 位置檢出-絕對值，增量型共用 17 位元編碼器
編碼器分解能-131072 P/rev
慣性矩-20 $kg.m^2$
機體構造：全閉自冷式、重量：9(kg)

本研究之所以採用此款交流伺服馬達主因為其具有如下之優點

(1) 加、減速容易控制，反應佳且高轉速下旋轉角度準確。

(2) 速度易於控制且安定。

(3) 可連續旋轉並能適應頻繁的加、減速。

(4) 能輸出大扭力。

(5) 可靠性高、壽命長、振動少、保養容易。

圖 1-1-2　伺服馬達外形圖

1.1.2 驅動器

由於控制系統經介面放出來的小信號，不足驅動大功率與扭力大的馬達，所以需由一驅動器（放大器）來驅動馬達，交流伺服馬達亦是如此。本伺服馬達系統設定在速度運轉模式。三相交流伺服馬達的驅動器主要為變流器，目前工業界最普遍使用的變流器為正弦式 PWM 變流器,其主要的控制方式為適切的切換功率晶體，使功率晶體在適當的時間導通或截止，以便提供交流電源給伺服馬達。驅動器大至可分為電壓型及電流型。電壓型驅動器是控制馬達輸入端的電壓，主要是用以控制馬達的轉速，常使用在可變速的馬達。電流型驅動器是控制馬達的相電流，主要是控制馬達的力矩，常使用在控制力矩輸出的馬達驅動器上。而本研究主要是控制電動缸之速度而非其力矩，所以我們採用電壓型的馬達驅動器。普遍的電壓型驅動器是PWM 中的三相六步變頻器(Three-Phase Six-Step Inverter)其外形，如圖 1-3 所示，其伺服驅動器規格為

電源系統：額定電壓與頻率：三相 AC200.230V /50~60Hz，單相 AC200~230V / 50~60Hz

控制方式：正弦波 PWM 控制，電流控制方式.

頻率響應：550Hz 以上

保護功能：具過電流切斷、回生過電壓切斷、過負荷切斷等多重保護

控制模式：位置、速度、轉矩控制模式

機體構造：開放強制冷卻式

圖 1-1-3　伺服馬達驅動器外形圖

1.1.3 編碼器(Encoder)

角度編碼器常以每轉多少輸出脈衝訊號為基準，同時也要注意所使用的電壓與電流大小，例如 12V 15mA 500 脈衝/轉；角度解碼器通常有五組接點，為電源、地線、A 相、B 相及 C 相，A、B 及 C 相三組為輸出訊號接點，其中 A 相與 B 相訊號相差 90 度相角，如此便可以判別出正轉或是反轉，C 相則為每轉一圈輸出一個脈衝訊號。

角度編碼器在使用上相當方便，由於其輸出訊號是一數位式的訊號，因此在實務上多利用一界面卡將訊號蒐集，並將訊號傳送至微電腦之中央處理單元運算。在硬體的安裝上就如同使用脈波計數器一樣，先將編碼器資料蒐集卡與編碼器連線，同時以驅動軟體驅動介面卡，便完成編碼器與介面卡的安裝，圖 1-1-4 為編碼器安裝配置圖。

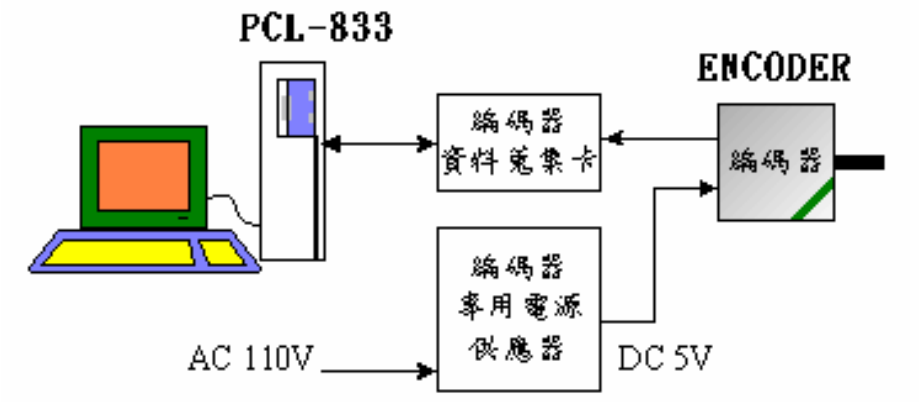


圖 1-1-4　編碼器安裝配置圖

1.2 控制端硬體 I/O 介面

控制端電腦之間需安裝一塊 PCL-726(D/A)卡(見圖 1-2-1)，用以輸出±10V 類比電壓至伺服馬達 DRIVER，然後驅動馬達在±7500RPM 運轉。此外電動缸之極限安全開關亦經由 PCL-726 之 Digital I/O 輸入控制端 PC 作為電動平台之上下極限安全保護。另外在控制端的電腦還安裝一塊 PCL-833 脈波計數卡(見圖 1-2-2)，其功用在於讀入伺服馬達編碼器迴授訊號，控制馬達作位置控制之用。

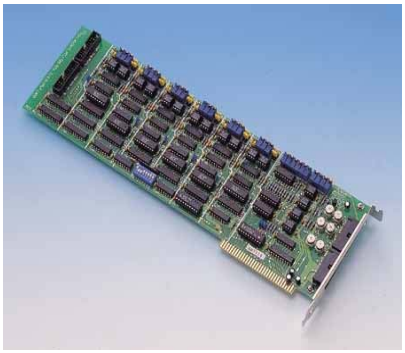


圖 1-2-1　PCL-726(D/A)卡

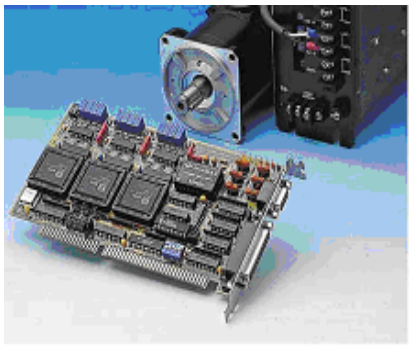


圖 1-2-2　PCL-833 脈波計數卡

1.2.1 D/A 轉換器

此卡主要為 Digital to Analog 的使用，含有 6 個獨立的輸出端，其類比輸出為 12 bits，有正負 10 伏的電壓範圍，目前實驗上只有用到單軸輸出，所以只先使用 1 個輸出端做控制。

1.2.2 伺服馬達之編碼器計數器

PCL-833 是專為計數編碼器(encoder)及光學尺(磁性尺)等元件之高速脈波而設計的卡片，具有三個訊號通道(channel)。本卡可接受 3 只編碼器/光學尺的輸入(即訊號通道)做為脈波計算的回授控制、量測或顯示之用，每個通道可接受 A, B, Z 的三相訊號，及配合 Home 的極性開關輸入。若輸入的 A、B 兩相 Quadrature 訊號可以經程式規劃分割為 X1, X2, X4 倍，以提高解析度，而配合 Z 相及 Home 的輸入可規劃成具自動清除計數的功能。

1.2.3 極限保護近接開關

極限保護近接開關提供致動器之上、下極限保護，其特性為無接觸方式檢出，適合高速，高頻的位置感測器，且採用專用 IC 因而外型很小。其電路是由振盪電路、檢波電路、史密特電路、輸出電路、電源電路等構成參閱，而這些電路全部包含在一個專用 IC 內而達到小型化，高性能化，低消耗電力的要求。圖 1-2-3 為本實驗所採用之 OMRON E2E-X1R5E1 2670 型近接感測器外觀構造，當有導磁或抗磁金屬接近線圈前端平面時，會使感應線圈內之磁渦流產生變化而改變內部控制電路之輸出接點狀態(導通)。因其必須透過電磁效應感測，故此型感測器所能感

測的物件必須為具有導磁性金屬物體，並且因為震盪線圈周圍的磁力線範圍會受到周遭物體所影響，故若在檢測物體附近有其他磁性物體或其他近接開關時容易產生感測誤差，於配置時應注意並避免此類情況發生。

圖 1-2-3　E2E-X1R5E1 2670 感測器

1.3 電動缸載重測試平台

目前本子題與計畫合作廠商一同協力開發製做了一組線性滑軌往復式載重測試平台，如圖 1-3-1 所示，用以測試伺服馬達以及電動缸之載重能力。它包含 AC 伺服馬達，滾珠螺桿(電動缸)、直線導引機構等三大部份，詳細分述如下

AC 伺服馬達：運動平台用的 AC 伺服馬達要求異於一般用途的馬達，它必須具有獨特的轉矩與優異的加減速特性，以便於控制工作台的各種定位及模擬運動。

滾珠導螺桿：載重測試平台的運動，是利用滾珠導螺桿將 AC 伺服馬達的旋轉運動轉換成直線運動。由於滾珠導螺桿的被利用，提高了運動平台的定位精度，縮短了定位時間，並能耐重(本研究所設計開發之載重測試平台最高可承載 750KG 之重量)，同時將傳統的滑動摩擦轉換成滾動摩擦，使機件間之摩擦阻力減少。滾珠導螺桿由螺桿、螺帽、回流管、鋼珠等所組成。

直線導引機構：由於載重測試平台需承受巨大重量與高速度的位移運動，過去傳統機構所採用的掙壓滑動接觸，因摩擦係數太大，易產生附著滑動現象，為了改善此項缺失，目前導引機構採用了柱或滾珠的直線運動(linear motioni)承座見。由於運動方式由滑動改變成滾動，因此大幅降低了摩擦係數，附著滑動現象也獲得改善。

本載重測試平台之機構組合規格如下所示，採用的電動缸最大伸長量為 300mm，所以電動缸最大與最小的長度限制約為 780mm 到 1080mm，其機構組合規格如下

平台座：黑鐵材質、將交流伺服馬達及導螺桿(電動缸)和線性滑軌組合於上、其間以齒規皮帶作為連結

尺寸：120mmL×60mmW×160mmH

AC 伺服馬達控制系統：三菱 MELSERVO-J2_SUPER 及其對應之馬達 HC_SFS_153B (2000rpm/min)

直線導引機構：HIWIN AG15.

滾珠導螺桿行程：300mm max.

滾珠導螺桿每轉移動距：5mm.

進一步地，本計畫之合作廠商承續之前油壓平台製作的經驗與最近合作開發之線性滑軌往復式載重測試平台心得，已完成電動運動平台的機構設計、加工及驅動系統架設等工作，整個電動六軸平台如圖 1-3-2 所示。

圖 1-3-1　線性滑軌往復式載重測試平台

圖 1-3-2　電動六軸運動平台

1.4 平台控制架構

目前我們所使用的電動平台位置控制系統架構如圖 1-4-1 所示,其中主要構成方塊有三種：

(1) 電動平台方塊包含馬達致動器與極限開關為實際硬體部分,其中馬達致動器主要為本研究所設計開發之電動缸，負責提供可動平台運動時所需的支撐力，而極限開則提供致動器之上、下極限保護。

(2) 控制端 PC 方塊以電腦為基礎,其中包含一塊插在電腦擴充槽上具有多通道的 D/A、Digital I/O 卡和一塊編碼介面電路的伺服控制卡，如此電腦控制系統架構可以隨時將所開發設計的控制器加以實驗驗證其實現之可能性，增加我們所設計開發的彈性與空間。

(3)伺服馬達系統方塊包含驅動器、馬達本體及脈波編碼器，負責提供平台運動時所需的動力及實際腳長資料之回授。

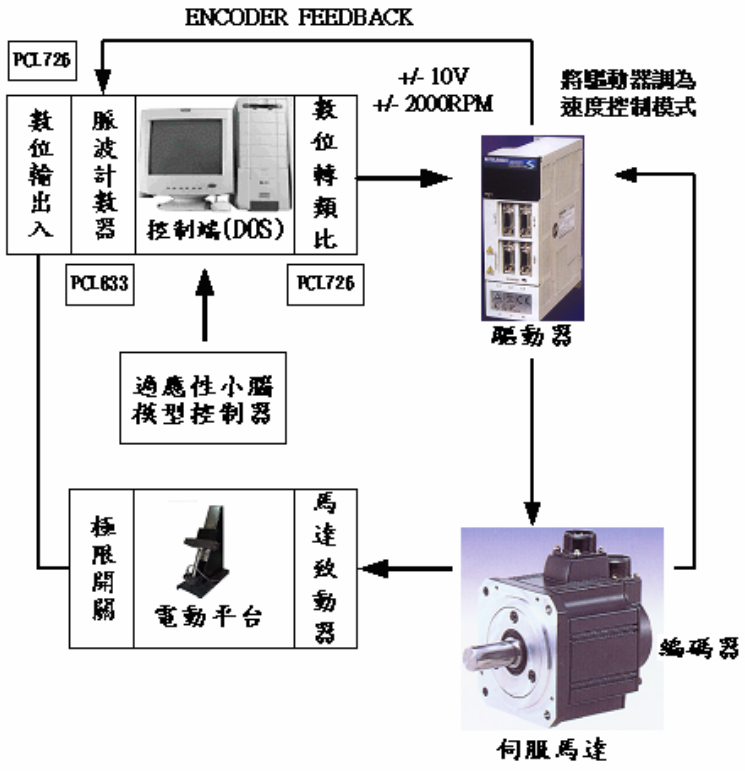


圖 1-4-1　電動平台控制系統架構圖

1.5 人機界面(Human Machine Interface)

近幾年來，個人電腦的發展相當快速，在個人電腦上發展應用軟体以達分散式處理的目的，似乎是未來各管理監控系統的發展重心，而運動平台在動感電影院的應用上，在與影片的配合上，當設計師發現動作平台的動作與影片無法配合

時，勢必需對動作平台的控制參數做修改，所以發展一個配合運動平台修改控制參數的人機界面軟体是有其必要的。這裡我們利用 Windows 98 結合 Borland C++ Builder 5.0(BCB5.0)來發展適合的人機界面，透過這個人機界面可以讓使用者更容易修改控制參數、儲存資料及察看結果。

BCB 是設計 Windows 應用軟体的利器，它使得 Windows 應用程式的開發變得相當容易，而且 BCB 是物件導向的程式語言(Object-Oriented Programmed, OOP)，物件導向的程式製作方法是目前最新、最進步的方法之一，此種方法將程式碼與資料組成封裝的物件，在一個程式內製作的物件，可以很容易的在另外一個應用程式中使用，因此可以節省程式開發的時間。一般 Windows 軟体內，下拉式的選單、連續按兩下就可以執行程式，視窗放大縮小、開檔存檔等等，均可容易的利用 BCB 設計出來，而其它如多執行序功能、DLL 的撰寫、網路應用程式及與各種資料庫的連結等，也都相當容易的可以在 BCB 中加以完成。另外，我們知道視窗程式的撰寫較麻煩的地方在於事件的處理程序，但是 BCB 也已經將所有的元件應有的事件處理程序設定好了，我們只需在對應的事件處理程序裡頭撰寫程式碼即可，可說非常方便。這個人機界面主要的功能有

(1) 緊急停止鈕，當平台控制發生意外時可立即中斷平台之動作。
(2) 可以直接以滑鼠托曳滑動鈕來修改PID控制器之參數資料。
(3) 可以動態觀看上/下極限保護開關是否作動及腳長、命令、電壓輸出等資料。
(4) 可以在自動模式下以滑鼠托曳滑動鈕來修改腳長命令資料，並即時觀察實際腳長位置之變化。
(5) 在自動模式下選擇追蹤命令，總共有Sine Wave、 rectangle Wave、 triangle Wave 三種控制訊號可供選擇。
(6) 手動調整平台位置。
(7) 直接對硬体做存取，將修改後之腳長命令直接送到D/A卡再送到平台實際執行。

自從 PC 的作業系統由 16 bits 的系統(Windows 3.x/DOS)轉換到 32 bits (Windows 95/98)的系統後，撰寫程式就無法直接作硬体 I/O 的存取與中斷。這是因為傳統的 DOS 作業系統是一個非多工的系統，使用者對系統週邊硬体擁有完整的存取權力，因此我們可以充分的利用系統，也可以直接對系統的各個介面作充分的利用及存取。但是，Windows 95/98 是一個粹純的 32 位元多工系統，在作業系統中實際執行的除了我們的程式外，還有許多 windows 本身的系統功能，保護軟体等，為了確保作業系統的穩定性，只好犧牲讓使用者直接對作業系統硬体的存取權。此外，多工作業系統利用執行序來安排 CPU 的處理時間，當分配時間已到，系統便切換到較高執行權限的執行序，即使上一個執行序尚未完全執行結束也會進行強制性切換，切換的過程有可能是在兩個指令中間，因此直接對硬体做存取所帶來的問題不小。所以一些常用的 C 語言指令如 outport、outportb、inport、inportb 等 I/O 指令及 getvect、setvect 等中斷指令均無法使用，這對硬体控制來說，實在

是相當麻煩的。本人機界面的開發作業系統是 Windows 95/98，在 95/98 中為了與舊有應用軟体的相容性，理所當然它必須接受硬体的存取，而在 BCB 中，要對硬体做 I/O 上的存取，主要的方法有：

(1) 自己寫驅動程式

(2) 用別人寫好的元件

(3) 用 Windows API

(4) 利用組合語言

而本研究所利用的方法是第四個，利用 BCB 可以在程式內加入組合語言的程式，來達到我們的目的。另外一點需注意的是，這個方法僅能適用於 Windows 95/98 的作業系統，如果是 Windows NT 的話，是行不通的，因為 Windows NT 作業系統是一個嚴謹的作業系統，本身並不允許應用程式與硬体直接溝通，更何況，NT 是一個跨平台的作業系統，不同的處理器有不同的硬体架構及存取方式，彼此間並不相容，所以若要在 Windows NT 下與硬体溝通的解決辦法只能利用 Windows API 與自行建立週邊的驅動程式(Device Drivers)，圖 1-5-1 為本研究利用 BCB 軟體開發之人機界面的操作介面，如此一來，我們可以有一簡便的 PID 控制介面，可以直接方便調整 PID 控制器之控制增益參數來獲得不錯的系統響應，同時也可以隨時監控整個系統之控制狀態。

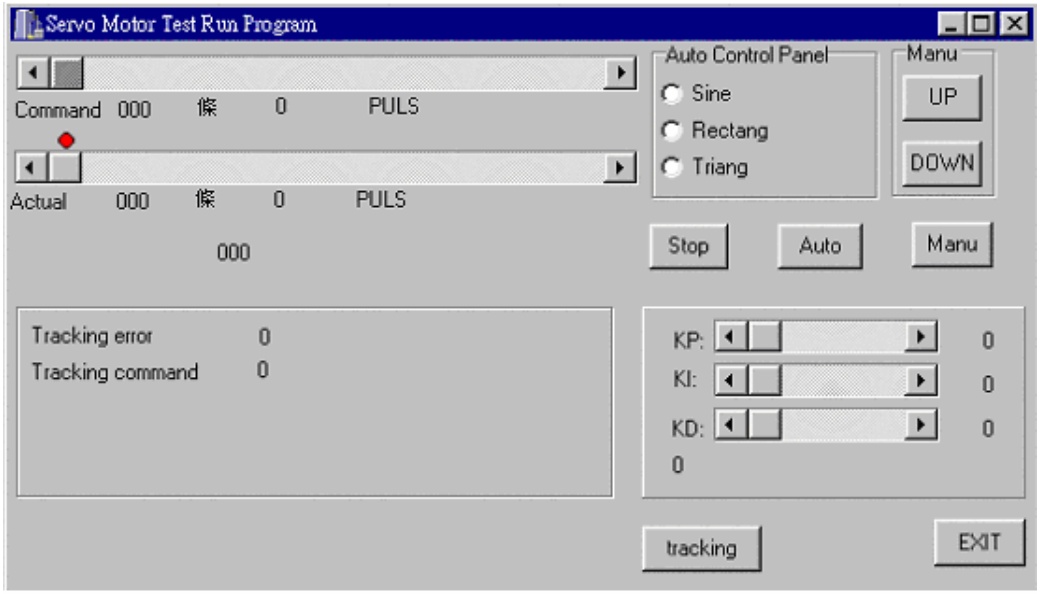


圖 1-5-1　人機界面的操作介面

1.6 適應性小腦模型控制器設計

基於本實驗室過去開發油壓運動平台的經驗，本研究已成功完成電動運動平台的機構設計、加工及驅動系統研究，由於整個電動運動平台之運動是透過 AC 伺服馬達經由皮帶帶動滾珠導螺桿使其由圓周運動轉成直線運動，所以其動態數學模式十分複雜而不好求得，所以不易使用傳統的控制理論來達成控制的目的。

為解決此問題，本研究提出一架構簡單、具快速學習且不需要受控系統動態模型的適應性小腦模型控制器來解決此控制問題，經由實驗結果發現本研究所提出之適應性小腦模型控制器可以有效地準確控制電動運動平台之腳長長度，以下各小節將逐一介紹其使用理論背景與設計步驟。

1.6.1 小腦模型控制器模型

小腦模型控制器(CMAC)理論首先由 Albus 於 1975 年提出，其主要依據為 Marr 於 1969 年所提出的人類小腦皮質模型架構為基礎，如圖 1-6-1。小腦模型控制器為模仿人類小腦皮質層的訊習儲存模式【1-1 ~ 1-4】，在此模式下，首先將輸入的狀態予以離散化(discretization)，其過程可視為將類比信號轉換為數位信號的一個取樣程序，如圖 1-6-2 所示，為 CMAC 對目標函數做離散化的情形，$S \in \{s_1, s_2, ...s_n\}$；再將每一量化的結果，將其對應到索引指標記憶體(S->C)，從索引指標記憶體，將我們設定好的類化指標，也就是致能記憶的個數，記為 ρ，一般而言，ρ 之值愈大，則類化程度愈好，至於輸出 y 也就是在類化指標致能記憶體後，將記憶體的內含值相加取總合的結果。

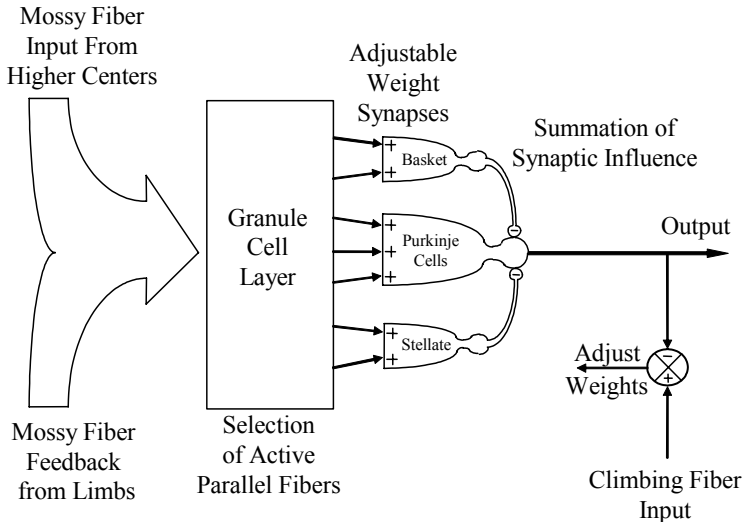


圖 1-6-1　小腦皮質模型

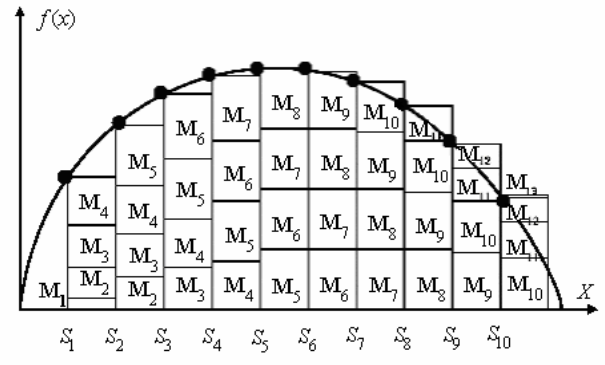


圖 1-6-2　CMAC 對目標函數做離散取樣的情形(ρ=4)

1.6.2 泛化型小腦模型控制器之數學模型

在原始小腦模型控制器中，每一個超立方體皆為一常數。只要輸入向量在此超立方體中，不論位置在何處，所得的值皆相同。CMAC 的輸出則為數個超立方體之值的和，輸入向量只被當作是記憶體的位址。如此一來輸出對輸入並無任何關係，所以也沒有輸出對輸入微分的訊息。此問題可藉一「非常數型態、可微分之基底函數」取代原常數基底來解決。所以選一泛化基底函數(General Basis Function，GBF)取代每一超立方體中的常數如圖 1-6-3。此修正後之小腦模型控制器可視為泛化型之小腦模型控制器。若選用之泛化函數為可微分則輸出對輸入之微亦可得【1-4】。

基本上泛化型小腦模型控制器之位址對映技術與原始小腦模型控制器是相同的。不同點在於泛化型採用泛化基底函數(General Basis Function，GBF)取代超立方體中的常數。函數$b_i(\cdot)$為第 i 個超立方體之基底函數，在超立方體之範圍內所有之基底函數都是有邊界的。如此小腦模型控制器之輸出則被對映到超立方體基底函數之線性組合。對於要儲存值（輸出）為 y_s，輸入為 $\mathbf{x_s}$ 之數學式為

$$y_s = \mathbf{a_s}^{\mathrm{T}}\mathbf{w}(\mathbf{x_s}) = \begin{bmatrix} a_{s,1} & a_{s,2} & \cdots & a_{s,N_h} \end{bmatrix} \begin{bmatrix} w_1(\mathbf{x_s}) \\ w_2(\mathbf{x_s}) \\ \vdots \\ w_{N_h}(\mathbf{x_s}) \end{bmatrix} = \sum_{j=1}^{N_h} a_{s,j} w_j(\mathbf{x_s}) \qquad (1.1)$$

其中$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_{N_h} \end{bmatrix}$是超立方體選擇向量，其中共有 N_e 個 1。$\mathbf{w}(\mathbf{x_s})$是記憶體內容值向量，其第 i 個向量元素為$w_i(\mathbf{x_s}) \equiv v_i b_i(\mathbf{x_s})$。$v_i$ 為一比重因子可經由學習獲得。則$\mathbf{w}(\mathbf{x_s})$及輸出 y_s 可表示如下

$$\mathbf{w}(\mathbf{x_s}) = \begin{bmatrix} v_1 b_1(\mathbf{x_s}) \\ v_2 b_2(\mathbf{x_s}) \\ \vdots \\ v_{N_h} b_{N_h}(\mathbf{x_s}) \end{bmatrix} = \begin{bmatrix} b_1(\mathbf{x_s}) & 0 & \cdots & 0 \\ 0 & b_2(\mathbf{x_s}) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & b_{N_h}(\mathbf{x_s}) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N_h} \end{bmatrix} \equiv \mathbf{B}(\mathbf{x_s})\mathbf{v} \qquad (1.2)$$

將(1.2)式代入(1.1)式，我們可得整個小腦模型控制器輸出為

$$y_s = \mathbf{a_s}^{\mathrm{T}}\mathbf{B}(\mathbf{x_s})\mathbf{v} \qquad (1.3)$$

從上列式子可知原始小腦模型控制器實為泛化型小腦模型控制器之特殊情況，即當所有$b_i(\mathbf{x_s})$為 1 時，泛化型小腦模型控制器即為原始小腦模型控制器。

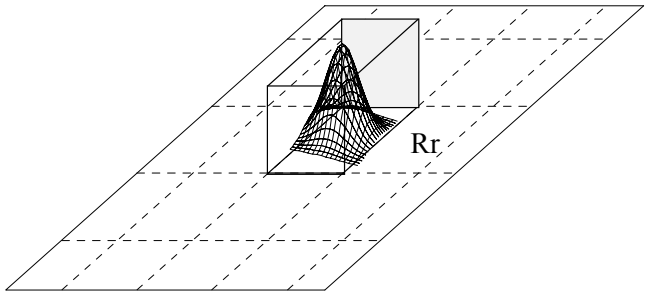


圖1-6-4　泛化型小腦模型控制器中以GBF取代原常數值

1.6.3 小腦模型控制器重要特性

小腦模型控制器之重要特性整理如下【1-1】

(1) CMAC 之學習，收斂至最小平方根誤差。

(2) 擁有良好的一般化能力。在變數量化與小腦模型控制器對映中，每一離散狀態都與其相鄰狀態共用超立方體，因此擁有好的一般化能力。

(3) 較低的計算負載。對一輸入狀態計算其輸出時，唯有涵蓋此狀態之超立方體會被使用，況且其計算只用到數學加法。與其他類神經網路比較，例如多層前向式網路，其計算負載顯然較低。

(4) 快速的學習速度。在每次重複的學習中，目標值與輸出值之誤差被直接用來修正與目前狀態相關之記憶體。而區域性一般化特性更加速了學習速度。

(5) 無論是在軟體撰寫或硬體的製作皆相當容易。在此技術中，其計算和結構都很簡單。

1.6.4 類神經網路近似定理

綜合上述介紹的原始型小腦模型控制器實與泛化型小腦模型控制器之類神經網路架構的輸出式為都寫成下列之形式【1-5】

$$y = \boldsymbol{\alpha}^T \boldsymbol{\Phi} \tag{1.4}$$

其中 $\boldsymbol{\alpha}^T = [a_1, a_2, ..., a_m] \in R^m$ 為類神經網路之參數向量，$\Phi = [w_1, w_2, ..., w_m] \in R^m$ 是一組泛化基底函數的向量，而(1.4)式之數學式已被證明且廣泛被用來近似任何非線性甚至時變之函式 Ω。經由近似理論，我們可以得知存在一組理想之類神經網路系統 y^*，如下列式子【1-5, 1-6】

$$\Omega = y^* + \varepsilon(t) = \boldsymbol{\alpha}^* \boldsymbol{\Phi} + \varepsilon(t) \tag{1.5}$$

其中 $\boldsymbol{\alpha}^*$ 為 $\boldsymbol{\alpha}$ 之理想向量，而 $\varepsilon(t)$ 則為小腦模型類神經網路之學習近似誤差，一般而言此近似誤差的值會隨著類神經網路神經元的增加而減少，事實上要最佳化的去近似一組非線性函式 Ω，所須之理想向量 $\boldsymbol{\alpha}^*$ 是很難獲得的，而且往往甚至為非唯

一解，因此一個估測類神經網路 $\hat{y}$ 被設計成

$$\hat{y} = \hat{\boldsymbol{\alpha}}^T \boldsymbol{\Phi} \tag{1.6}$$

其中 $\hat{\boldsymbol{\alpha}}$ 為 $\boldsymbol{\alpha}$ 的估測值，而小腦模型類神經網路近似誤差 $\tilde{y}$ 的定義為

$$\tilde{y} = \Omega - \hat{y} = \boldsymbol{\alpha}^{*T} \boldsymbol{\Phi} + \varepsilon - \hat{\boldsymbol{\alpha}}^T \boldsymbol{\Phi} = \tilde{\boldsymbol{\alpha}}^T \boldsymbol{\Phi} + \varepsilon \tag{1.7}$$

其中 $\tilde{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^* - \hat{\boldsymbol{\alpha}}$，並且假設 $|\varepsilon(t)| \le E$，即小腦模型類神經網路之學習近似誤差可被一正常數 E 限制其邊界值大小。

1.6.5 伺服馬達與電動缸系統簡化動態方程式

基本上，感應伺服馬達的系統動態數學模式包含下列三個方程式【1-7 ~ 1-9】

(1)電壓方程式

電壓方程式為

$$\boldsymbol{v} = \boldsymbol{R}\boldsymbol{i} + \frac{d}{dt}\boldsymbol{\lambda} \tag{1.8}$$

式中的矩陣表示定子及轉子之每一相的分量。其中 $\boldsymbol{v}$ 為電壓、$\boldsymbol{i}$ 為電流、$\boldsymbol{R}$ 為電阻以及 $\boldsymbol{\lambda}$ 為磁通。

(2)轉矩方程式

轉矩方程式為

$$T_e = \frac{m}{2} n_p (\lambda_m \times i_r) \tag{1.9}$$

電動機所產生的電磁轉矩可表示為氣隙磁通和轉子電流向量的乘積。上式 T_e 為電磁轉矩、m 為相數、n_p 為極對數目、λ_m 為氣隙磁通、i_r 為轉子電流以及 $\times$ 為外積符號。

(3)轉動方程式

轉動方程式為

$$J\ddot{\theta}_r + B\dot{\theta}_r + T_L = T_e \tag{1.10}$$

其中 J 是轉動慣量(Moment of Inertia)、B 是黏性摩擦係數(Damping Coefficient)、θ_r 是轉子角度、T_e 定義為電磁轉矩以及 T_L 為外部的

負載轉矩。根據磁場導向控制方法，則可將電磁轉矩簡化成

$$T_e = K_t i_{qs}^* \tag{1.11}$$

$$K_t = (3n_p/2)(L_m^2/L_r) i_{ds}^* \tag{1.12}$$

其中 K_t 是轉矩常數、i_{qs}^* 是轉矩命令電流、i_{ds}^* 為磁通命令電流、L_m 為每相的磁化電感以及 L_r 為每相的轉子電感。結合方程式(1.10)和方程式(1.11)，則磁場導向感應伺服馬達驅動系統動態可表示為

$$\begin{aligned}\ddot{\theta}_r(t) &= -\frac{B}{J}\dot{\theta}(t) + \frac{K_t}{J} i_{qs}^*(t) - \frac{1}{J} T_L \\ &\triangleq A_p \dot{\theta}(t) + B_p u(t) + D_p T_l\end{aligned} \tag{1.13}$$

其中 A_p、B_p 以及 D_p 代表感應伺服馬達系統參數，且分別定義為 $A_p = -B/J$、$B_p = K_t/J$ 以及 $D_p = -1/J$；$u(t) = i_{qs}^*$ 代表控制輸入。根據磁場導向理論，感應伺服馬達的動態特性近似於它激式的直流馬達；然而其控制特性依然會受到系統之不確定量所影響，包含機械參數變動、外來負載干擾、非理想磁場導向暫態響應以及實際應用時之不可模式化動態。

方程式(1.13)中之 T_L 即為研究室所開發製做之線性滑軌往復式載重測試平台，它包含負重平台，滾珠螺桿(電動缸)、直線導引機構等三大部份。其中負重平台可承載 375K 之重物，滾珠導螺桿(電動缸)則是將 AC 伺服馬達的旋轉運動轉換成直線運動，並將滑動摩擦轉換成滾動摩擦，使機件間之摩擦阻力減少，直線直線導引機構採用了滾珠式直線運動承座。由於運動方式由滑動改變成滾動，因此大幅降低了摩擦係數，附著滑動現象也獲得改善。而整個線性滑軌往復式載重測試平台之動態方程式十分複雜不易求得，所以本研究直接將整個線性滑軌往復式載重測試平台視為一伺服馬達之外來負載，整個電動缸系統方塊圖如圖 1-6-4 所示。

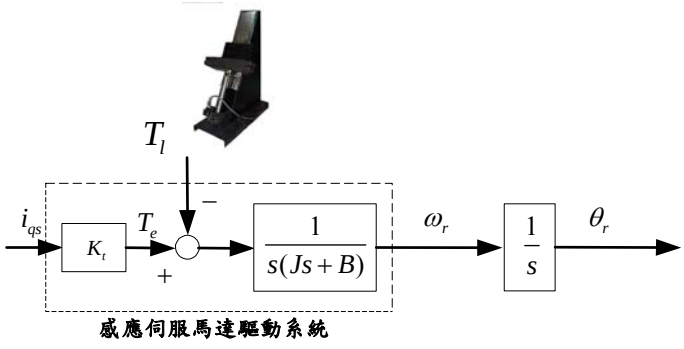


圖 1-6-4　電動缸系統動態簡化方塊圖

1.6.6 適應性小腦模型類神經網路控制系統

如果系統參數已知，而且外部負載及干擾是可量測的則一個理想控制器可獲得如下【1-10】

$$u^*(t) = B_p^{-1}[-A_p\dot{\theta}(t) - D_pT_l + \ddot{\theta}_d(t) + k_1\dot{e}(t) + k_2e(t)] \tag{1.14}$$

將(1.14)式代入(1.13)式可得到

$$\ddot{e}(t) + k_1\dot{e}(t) + k_2e(t) = 0 \tag{1.15}$$

此時我們假如適當選擇k_1及k_2讓(4.20)式為一赫維茲多項式，即使得多項式的根落在複數平面的左半平面，如此(4.21)暗示$\lim_{t\to\infty} e(t) = 0$，即達到控制目的，即利用理想控制器$u^*$可使系統達到控制目的。但是不幸地，由於系統參數、外部負載及干擾等參數通常為不確定項或無法求得，所以理想控制器並無法獲得及實現，尤其針對本子題所將控制之整個六軸電動平台系統將更難以實現。

為了解決上述之問題，在此提出設計一適應性小腦模型類神經網路控制系統，如圖 1-6-5 所示，其中包含適應性小腦模型類神經網路控制器以及具學習法則的線上學習機制，其中包含一小腦模型類神經網路控制器與一補償控制器。而且，本子題因為泛化型小腦模型控制器之類神經網路架構其激發函數不像一般型小腦模型控制器一般死板激發，泛化型小腦模型控制器在激發時包含了模糊漸層的觀念，更加適合人類的學習方式，所以本子題之小腦模型類神經網路控制器在此選用泛化型小腦模型控制器之類神經網路架構。

整個控制系統設計目的為利用適應性小腦模型類神經網路架構來實現線上訓練之適應性小腦模型類神經網路控制器，並定義追蹤誤差$e = \theta_d - \theta$，其中θ代表感應伺服馬達實際的轉子位置，θ_d代表參考模式命令訊號，且將e和$\dot{e}$分別作為適應性小腦模型類神經網路的輸入訊號，適應性小腦模型類神經網路控制系統的輸出表示成u，以作為感應伺服馬達驅動系統的輸入訊號。由圖中可以清楚知道整個控制法則為

$$u(t) = \hat{u}_{CMAC}(\mathbf{e}(t), \hat{\boldsymbol{\alpha}}) + u_{cp} = \boldsymbol{\alpha}^T\hat{\boldsymbol{\Phi}}(\mathbf{e}(t)) + u_{cp} \tag{1.16}$$

其中小腦模型控制器$\hat{u}_{CMAC}$為主要追蹤控制器用來近似理想控制器$u^*(t)$；而補償控制器u_{cp}則被設計來消除理想控制器與小腦模型控制器之間的誤差。首先，在此定義滑動表面函式為

$$s(t) = \dot{e} + k_1 e + k_2 \int e(\tau) d\tau \tag{1.17}$$

並由之前介紹的近似理論可知一個理想之小腦模型控制器 u^*_{CMAC} 能夠被設計來近似理想控制器，即【1-6】

$$u^*(t) = u^*_{CMAC} + \varepsilon = \boldsymbol{\alpha}^{*T}\boldsymbol{\Phi} + \varepsilon \tag{1.18}$$

其中 ε 為近似誤差，並且假設 $|\varepsilon(t)| \le E$。經由一些簡單計算可得以下之誤差方程式

$$\dot{s}(t) = \ddot{\theta}_e(t) + k_1 \dot{e}(t) + k_2 e(t) = B_p[u^*(t) - \hat{u}_{fz} - u_{cp}] \tag{1.19}$$

為了解決近似誤差之邊界值 E 問題，本研究引進一具近似誤差觀測能力之適應性小腦模型控制系統，以近似誤差線上學習法則來修正不確定量邊界值，我們將使用一個極限值估測器來監測不確定量邊界值 E

$$\tilde{E}(t) = E - \hat{E}(t) \tag{1.20}$$

其中 $\hat{E}(t)$ 為我們要估測的不確定量邊界值 E。

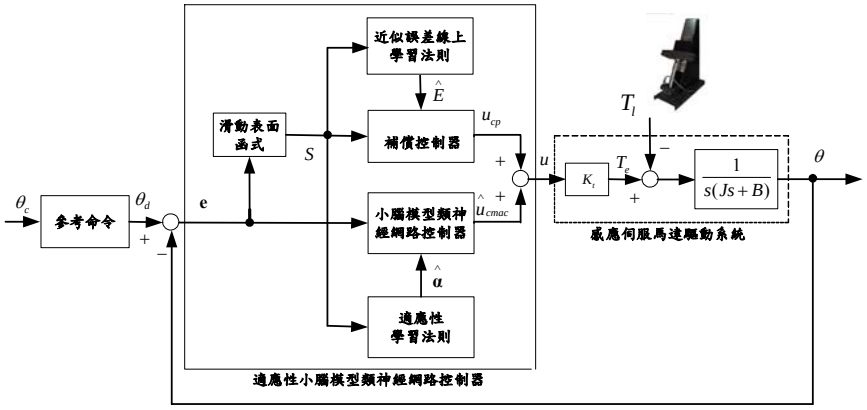


圖 1-6-5 適應性小腦模型類神經網路控制系統方塊圖

經過先前一些推導，我們可以將可各控制系統先整理成一個定理如下，並附上其詳細數學推導過程如下

定理：一個由(1.13)式所代表的伺服感應馬達系統，如果我們將適應性小腦模型控制系統設計成(1.16)式，則適應性小腦模型控制器之學習法則將被設計為式子(1.21)

與(1.22)，而補償控制器將設計成為(1.23)式

$$\dot{\hat{\boldsymbol{\alpha}}} = \eta_1 s(t)\Phi \tag{1.21}$$

$$\dot{\hat{E}} = \eta_2 \left|s(t)\right| \tag{1.22}$$

$$u_{cp} = \hat{E}(t)\,\mathrm{sgn}(s(t)) \tag{1.23}$$

其中η_1與η_2是學習速率，其值為一正的常數值，sgn(·)為符號函式，如此則保證適應性小腦模型控制系統之追蹤誤差收斂。

證明：

定義一李亞普諾函數為

$$V_2(s(t),\tilde{E}(t),\tilde{\boldsymbol{\alpha}}) = \frac{1}{2}s^2(t) + B_p[\frac{\tilde{E}^2(t)}{2\eta_2} + \frac{\tilde{\boldsymbol{\alpha}}^T\tilde{\boldsymbol{\alpha}}}{2\eta_1}] \tag{1.24}$$

將(1.24)式對時間微分，及利用(1.19), (1.21)~(1.23)可得到

$$\begin{aligned}
\dot{V}_2 &= s\dot{s} + \frac{B_p}{\eta_1}\tilde{\boldsymbol{\alpha}}^T\dot{\tilde{\boldsymbol{\alpha}}} + \frac{B_p}{\eta_2}\tilde{E}\dot{\tilde{E}} \\
&= s(t)[B_p(\tilde{\boldsymbol{\alpha}}^T\hat{\boldsymbol{\Phi}} + \varepsilon - u_{cp})] + B_p\frac{\tilde{\boldsymbol{\alpha}}^T\dot{\tilde{\boldsymbol{\alpha}}}}{\eta_1} + \frac{B_p}{\eta_2}\tilde{E}\dot{\tilde{E}} \\
&= \varepsilon B_p s(t) - EB_p\left|s(t)\right| \\
&\le -(E - \left|\varepsilon\right|)B_p\left|s(t)\right| \quad \le 0
\end{aligned} \tag{1.25}$$

依據里亞普諾穩定理論，$V > 0$和$\dot{V} \le 0$，因此當時間趨近於無窮大時，存在於參考模式與受控體之間的追蹤誤差，必然會收斂至零，如此則適應性小腦模型控制系統之穩定性可得到保證。

綜合以上之控制器設計推導，我們可以將整個與設計步驟與想法簡單描述如下：

步驟一：

定義追蹤誤差$e = \theta_d - \theta$，其中θ代表伺服馬達實際的轉子位置，θ_d代表參考模式命令訊號，並定義一滑動表面$s = \dot{e} + k_1 e + k_2\int e d\tau$。

步驟二：

利用一小腦模型控制類神經網路線上學習近似一理想控制器。

步驟三：

依據最佳近似定理我們可得知存在一近似誤差，為了克服此誤差往往使用一切換控制器補償之，但卻因而造成控制力有嚴重的顫抖現象。

步驟四：

使用一個極限值估測器來監測不確定量邊界值，在此定義不確定量邊界值估測誤差為 $\tilde{E}(t) = E - \hat{E}(t)$ 。

步驟五：

適應性小腦模型控制系統設計成 $u(t) = \hat{u}_{CMAC} + u_{cp}$ ，其中 $\hat{u}_{CMAC}$ 為主要追蹤控制器用來近似理想控制器;而補償控制器 u_{cp} 則被設計來消除理想控制器與小腦模型控制器之間的誤差。

步驟六：

依據李亞普諾夫穩定法則推論而得之線上學習法則

$$\dot{\hat{\boldsymbol{\alpha}}} = \eta_1 s(t)\Phi$$

$$u_{cp} = \hat{E}(t)\,\mathrm{sgn}(s(t))$$

$$\dot{\hat{E}} = \eta_2 \left| s(t) \right|$$

其中 η_1 、 η_2 是學習速率，其值為一正的常數值，sgn(·) 為符號函式。

1.6.7 實驗結果

為了驗証本研究所設計之智慧型控制器對於機械參數變化與外來負載干擾的系統強健性，本研究設計兩個實驗條件驗證控制器之可行性，兩種實驗條件分別為

空載條件(條件一): 於實作中未加入任何負載，只有馬達透過齒規皮帶直接帶動整個平台運動。

負載條件(條件二)：於實作中加入 375 kg 之砝碼於負載平台上，當作系統參數變化。

接下來我們利用週期性正弦波及方波位置命令來驗證控制器追隨控制角度變化的軌跡來觀測平台的動作是否正確，以驗証我們所發展之小腦模型控制器設計是否正確，而當位置命令為方波時，為避免在命令轉換瞬間產生過大之控制電壓而傷害實驗機構,因此皆設置一前置率波器當作參考模型，用來規範週期性步階命令之響應特性,其前置率波器如下

$$\frac{w_n^{\,2}}{s^2 + 2\zeta w_n s + w_n^{\,2}} = \frac{64}{s^2 + 16s + 64} \tag{1.26}$$

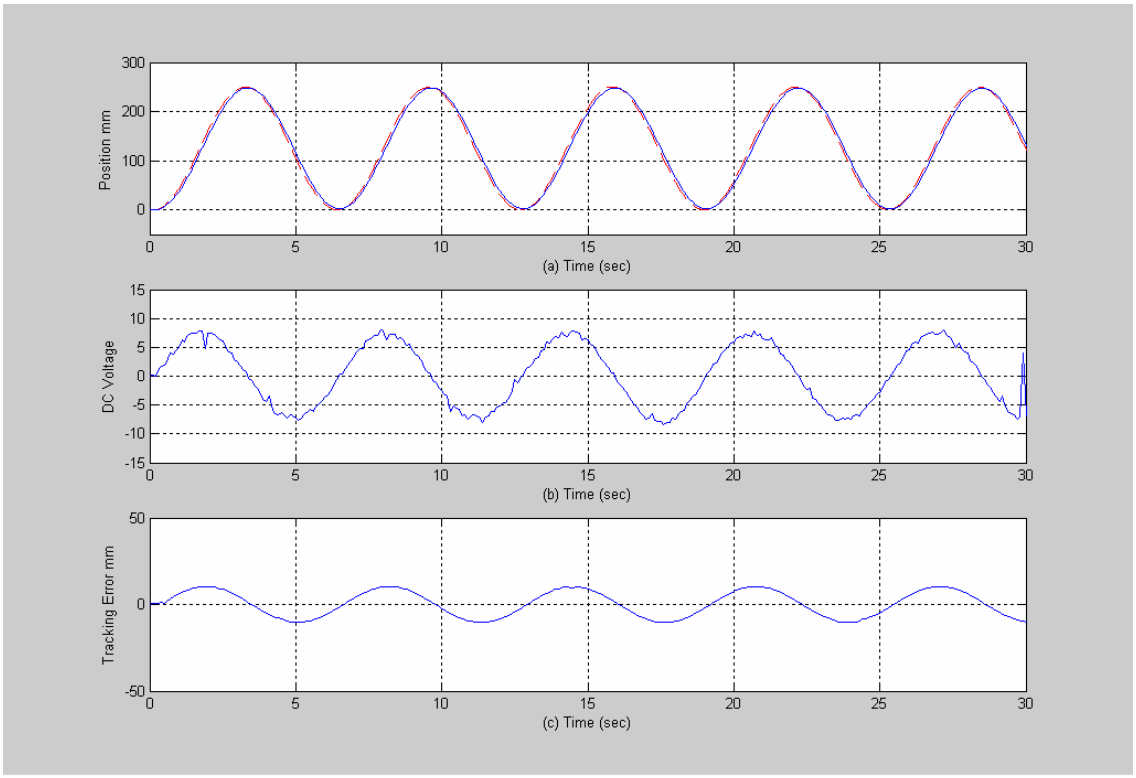
其中 ζ 表示阻尼比，並設臨界阻尼比為一， w_n 表示欠阻尼自然頻率。設定臨界阻尼是因為臨界阻尼的二階系統極點為兩個實重根，其系統暫態響應不呈現震盪且穩定於固定終值上，其上升時間也較過阻尼系統短。另一方面當軌跡命令為週期性弦波時，則使參考模型與實際弦波相同，亦即參考模型設定為一，控制執行週期為 0.002 秒。

首先以一般 PID 控制器來控制實驗平台，在 PID 控制系統中所選擇的參數，不但必須於實驗中達到最佳的暫態控制響應，且亦需考慮系統穩定性及控制效能

的需求，在此以試誤法得到 PID 控制系統的參數值 P=35、I=1.5、D=5，其實驗結果如圖 1-6-6 及圖 1-6-7，其中各圖中的(a), (b)與(c)為實驗測試條件一(未加入任何負載)之實驗結果，各圖中的(d), (e)與(f)為實驗測試條件二(加入 375 kg 之砝碼)之實驗結果，其中(a)與(d)為控制響應圖，(b)與(e)為控制訊號圖，(c)與(f)為控制誤差圖，觀察實驗結果發現雖經過長時間的調整控制參數，但其系統響應依舊有些許不盡理想之處，尤其在於測試條件二下，當受控系統有很大參數變化時，此時良好的系統控制響應即無法獲得，如圖 1-6-6(d)與圖 1-6-7(d)可明白看出系統追蹤響應已沒有很好。為了克服此缺點，接著我們將所設計之適應性小腦模型控制器於整個實驗平台，其實驗結果如圖 1-6-8 及圖 1-6-9，其中各圖中的(a), (b)與(c)為實驗測試條件一(未加入任何負載)之實驗結果，各圖中的(d), (e)與(f)為實驗測試條件二(加入 375 kg 之砝碼)之實驗結果，其中(a)與(d)為控制響應圖，(b)與(e)為控制訊號圖，(c)與(f)為控制誤差圖，由實驗結果中可以明白發現不管任何實驗測試條件下均可獲得良好的控制性能。綜合以上所言，我們可以簡單建立一 PID 控制器與適應性小腦模型控制器之比較表格，如表 1-6-1 所示。

表 1-6-1　PID 控制器與適應性小腦模型控制器之性能比較表

控制法則	設計複雜度	系統強健性	是否具有學習能力	是否需要專家經驗
PID 控制	易	差	否	是
適應性小腦模型控制器	難	佳	是	否



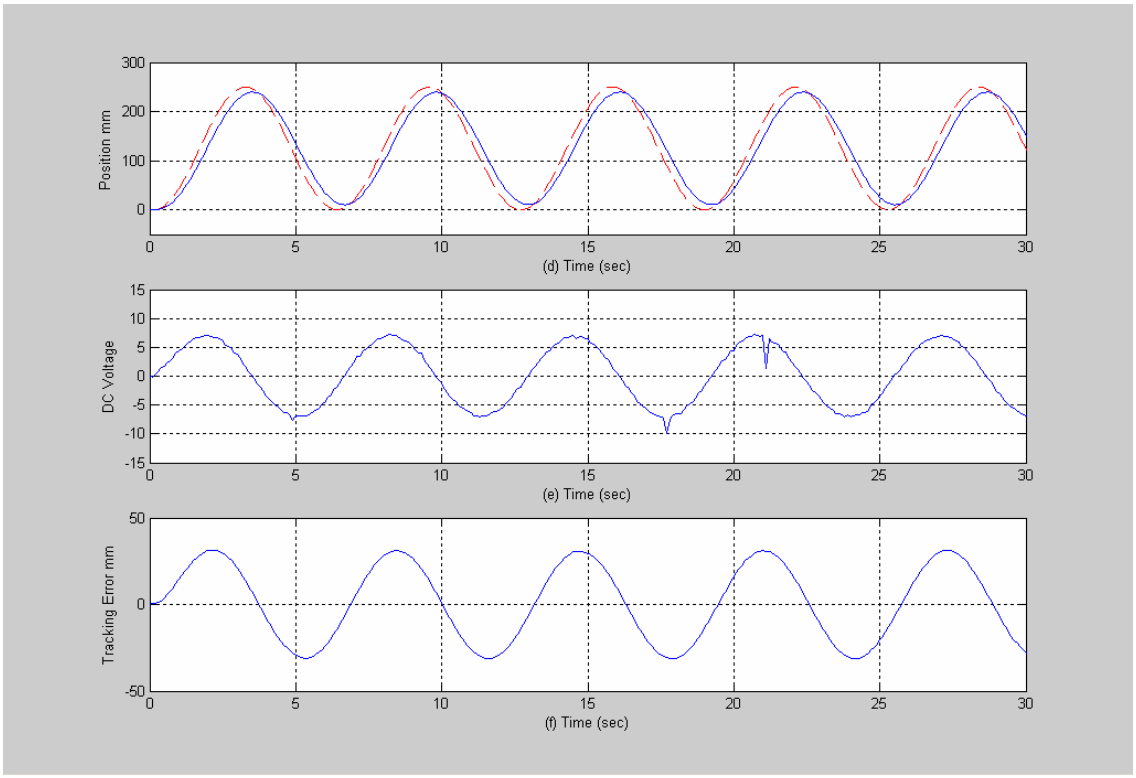
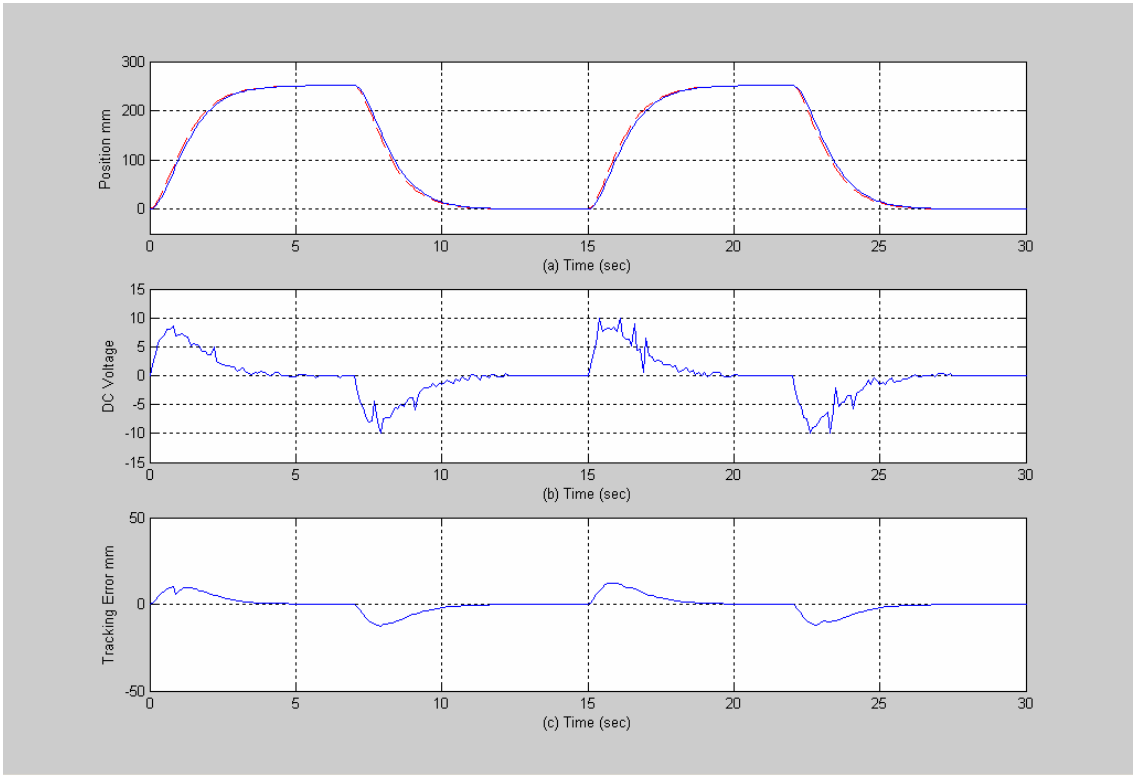


圖 1-6-6　利用 PID 控制器之正弦波實驗結果
(a)與(b)為無負載之實驗結果;(c)與(d)為載重 375kg 之實驗結果



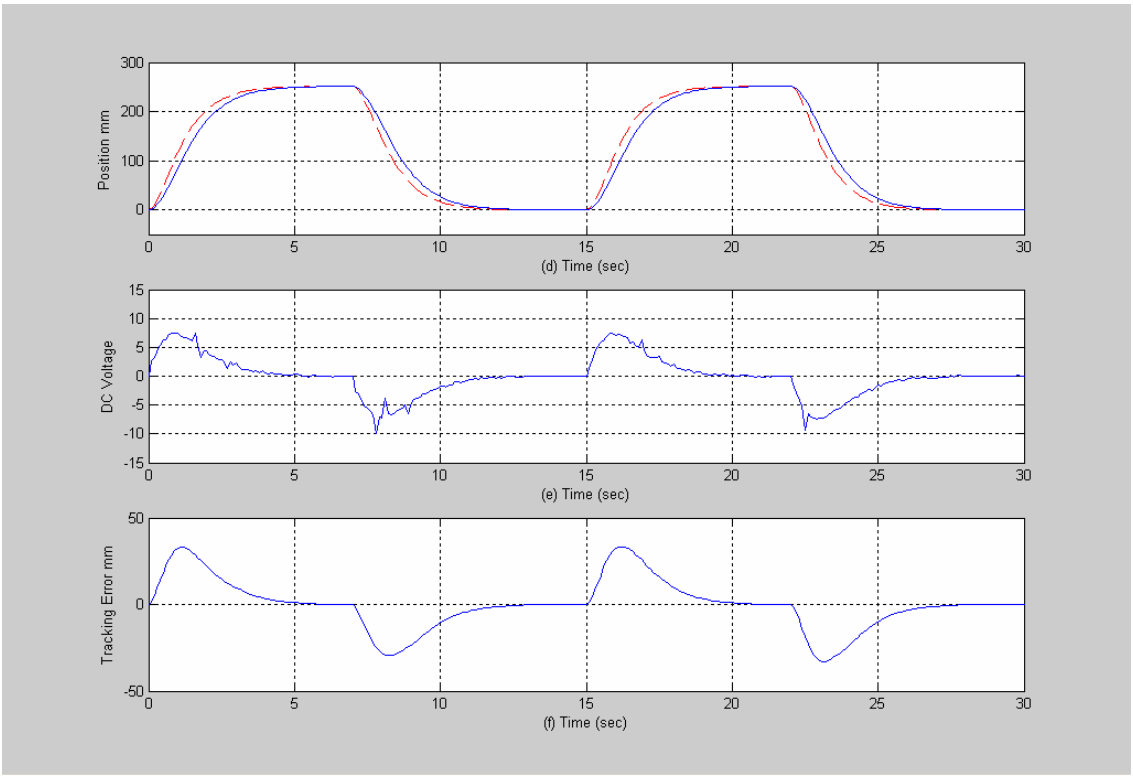
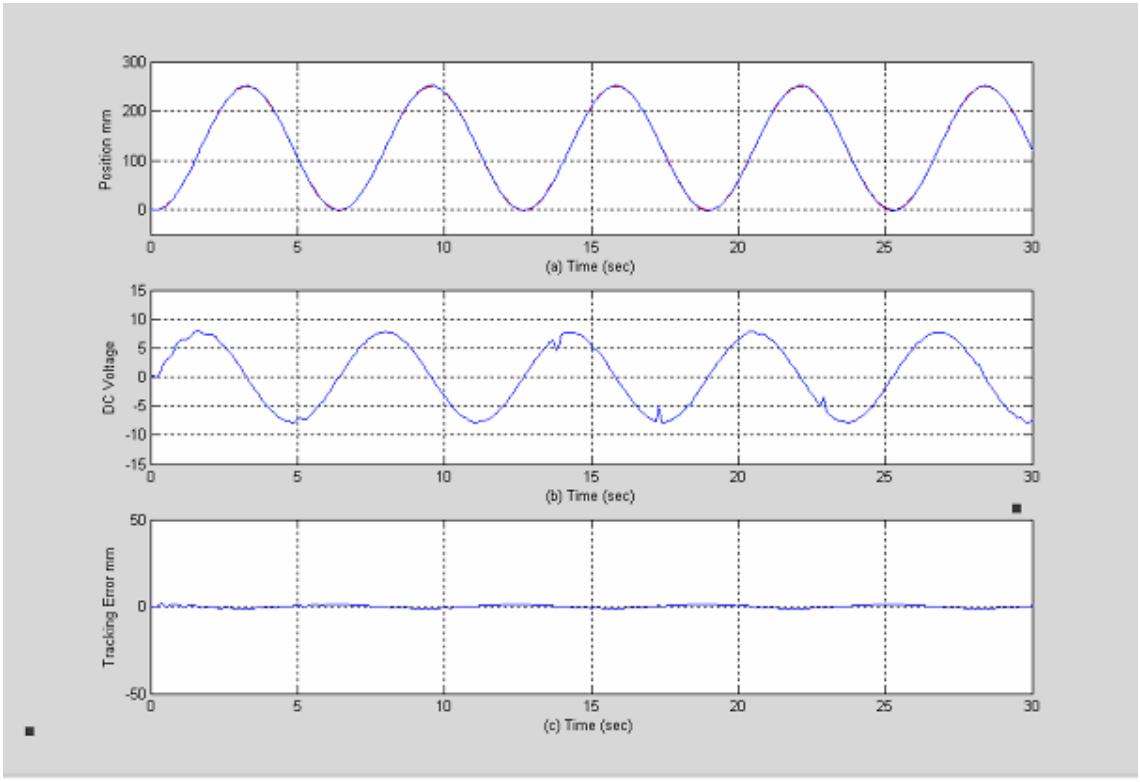


圖 1-6-7　利用 PID 控制器之方波實驗結果

(a)與(b)為無負載之實驗結果;(c)與(d)為載重 375kg 之實驗結果



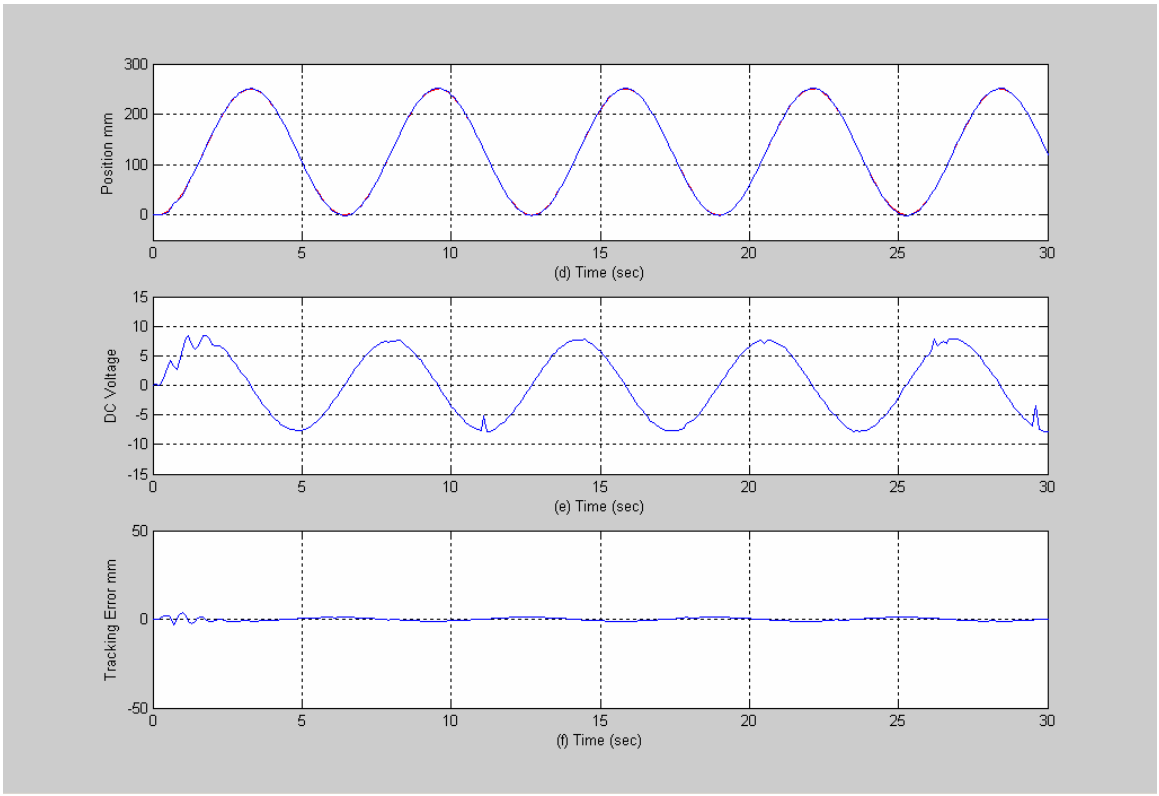
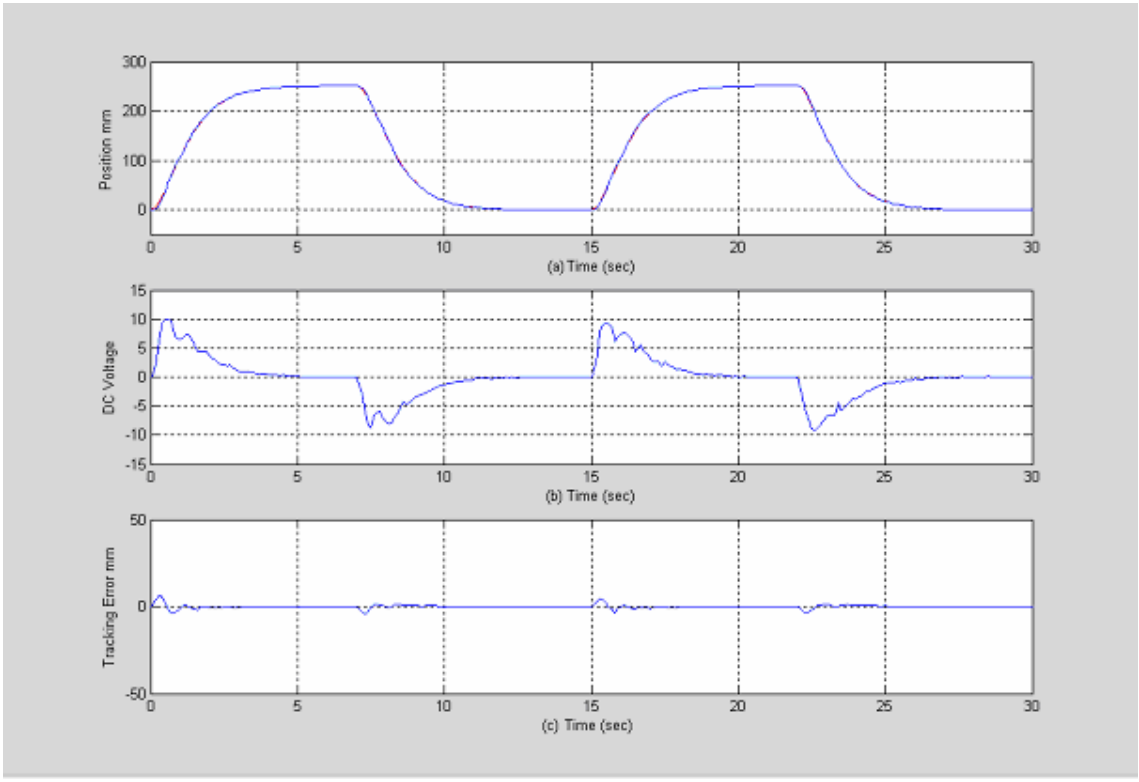


圖 1-6-8　利用適應性小腦模型控制器之正弦波實驗結果
(a)與(b)為無負載之實驗結果;(c)與(d)為載重 375kg 之實驗結果



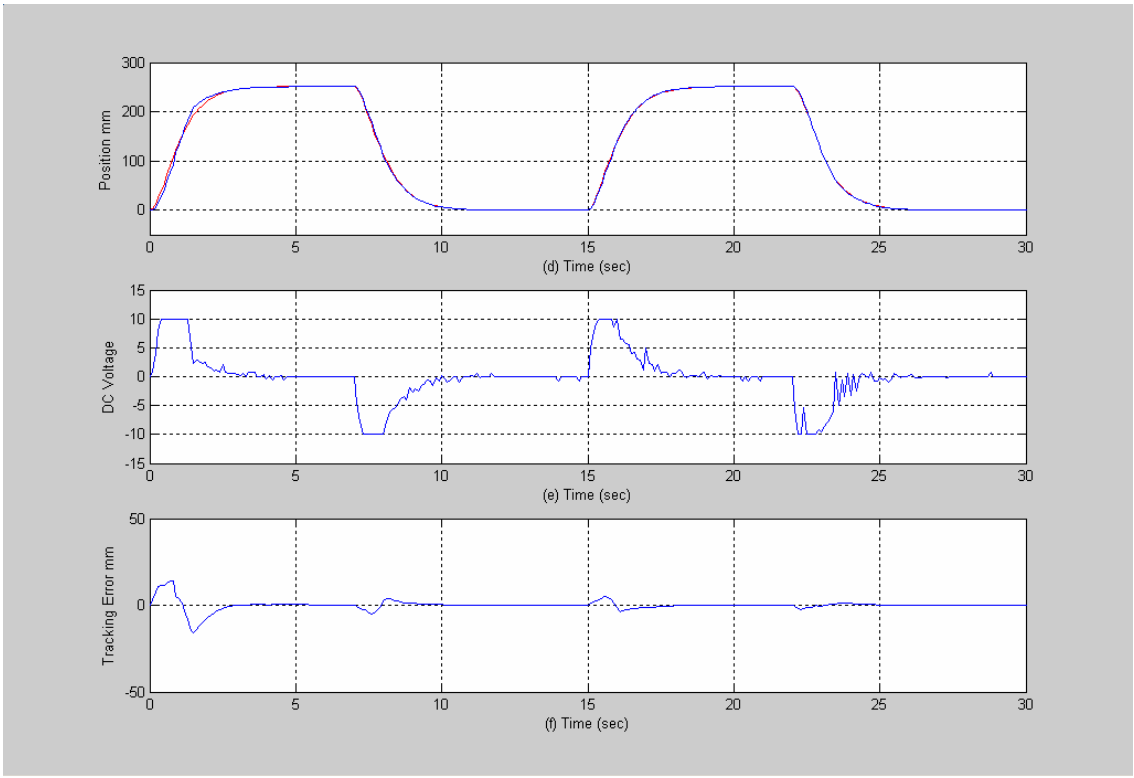


圖 1-6-9　利用適應性小腦模型控制器之方波實驗結果
(a)與(b)為無負載之實驗結果;(c)與(d)為載重 375kg 之實驗結果

1.7 陀螺儀加速規

物體的運動的分析將使用歐拉角(Euler angles) ϕ, θ, ψ (phi, theta, psi)。為了展示如何將這些角用來定義物體的位置，參考圖 1-22 (a)所示的陀螺。此陀螺附在 O 點上，在某些瞬間其相對於固定 X, Y, Z 軸的方位如圖 1-22 (d)所示。為了定義此位置，須使用第二組座標 x, y, z 軸。為了方便討論，假設此座標固定在陀螺上。一開始 X, Y, Z 與 x, y, z 軸重合，如圖 1-7-1(a)，陀螺的最終位置使用下列三個步驟來決定

(1) 將陀螺繞 Z(或 z 軸)轉動 ϕ 角($0 \le \phi < 2\pi$)，圖 1-7-1(b)。

(2) 將陀螺繞 x 軸轉動 θ 角($0 \le \theta \le \pi$)，圖 1-7-1(c)。

(3) 將陀螺繞 z 軸轉動 ψ 角 ($0 \le \psi < 2\pi$)獲得最終位置，圖 1-7-1 (d)。

這三個角 ϕ, θ，而後 ψ 的順序不可改變，因有限旋轉並不是向量。雖然如此，但無限小旋轉 $d\phi, d\theta$ 及 $d\psi$ 為向量，故陀螺的角速度 ω 可以歐拉角的時間導數來表示。其角速度分量 $\dot{\phi}, \dot{\theta}, \dot{\psi}$ 分別稱為進動(precession)，俯動(nutation)及自轉(spin)。其正方向如上圖所示。您可發現這三個向量並不完全相互垂直，但陀螺的 ω 仍可以這三個分量來表示。

AHRS400 是由加速規 (accelerometers)和陀螺儀 (gyro)所組成；加速規主要功能在於量測系統所受的加速度，陀螺儀則是量測系統的角速度，其他平台資訊包括位移、速度、姿態角、角速度可由上面所量測的數值加以微分或積分而得。角速度感應器(rate gyroscope =角速度陀螺儀)是應用 Coriolis 效果的電子儀器，其採用硅素超精密環狀傳感片閉迴路設計,而產生一耐震耐衝擊的高精確度類比輸出電壓，利用 MEMS 製程中體型微機械加工(Bulk- Micromachining)微感測器,陀螺儀所量得的是角速度資料,經積分可得旋轉角度,而經微分可得角加速度資料. 慣性系統最大的一個缺點,就是它的誤差會隨著工作時間而累積,這樣的誤差是無法接受的。需要以各種濾波和定位方法,去修正位置的誤差。(數位濾波一例:卡爾曼濾波器 Kalman Filter, 和移動平均)使用者可在陀螺儀及 ADC 間加插 1 階線性抗混淆濾波器(anti-alias filter of first-order)以避免高頻信號混淆於所關注的頻帶內。目前使用的取樣率: 200Hz.以 AD 後加 digital 低通濾波器 0-10Hz/20Hz 和移動平均。

目前本子題已購進一組陀螺儀加速規如圖 1-7-3 所示，未來準備將利用 AHRS4000 所量測到之運動平台的 X,Y,Z 方向線性速度、加速度及角速度等機構之動態資料，於控制器設計中同步驗證其系統動態之正確性，並進一步加入平台加速度之控制。目前本子題已完成安裝 AHRS4000 的工作，整合到原有的油壓平台上已有初步之成果，整個量測成果如圖 1-7-4 所示，未來即利用這些訊號代入控制系統設計中使整個平台模擬程度更加真實。

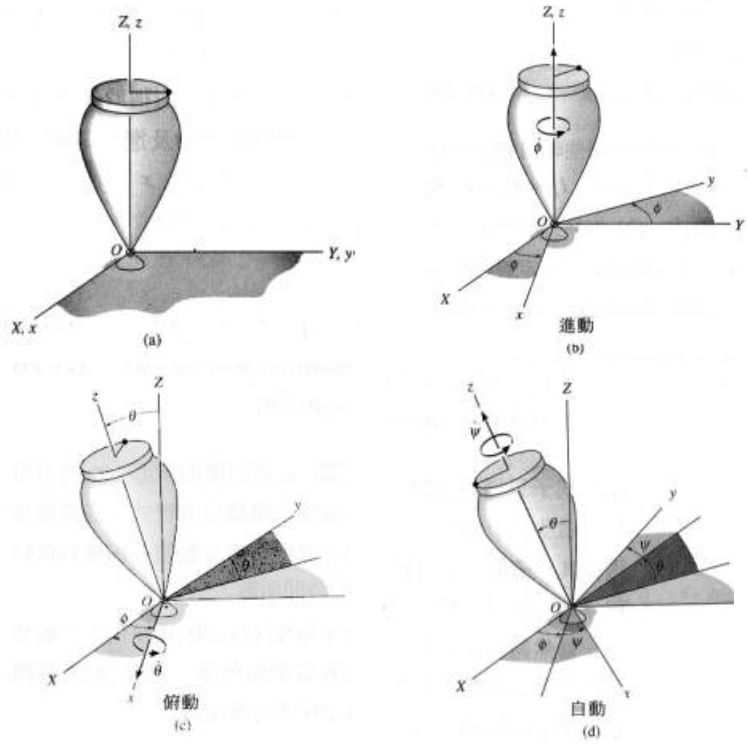


圖 1-7-1　迴轉儀運動分析

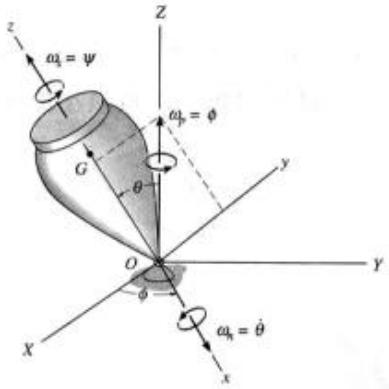


圖 1-7-2　迴轉儀運動分析

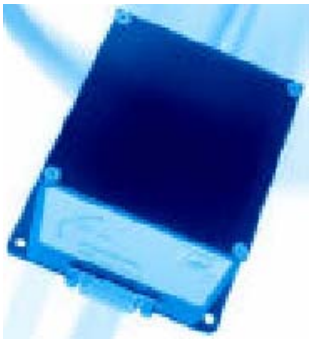


圖 1-7-3　AHRS4000

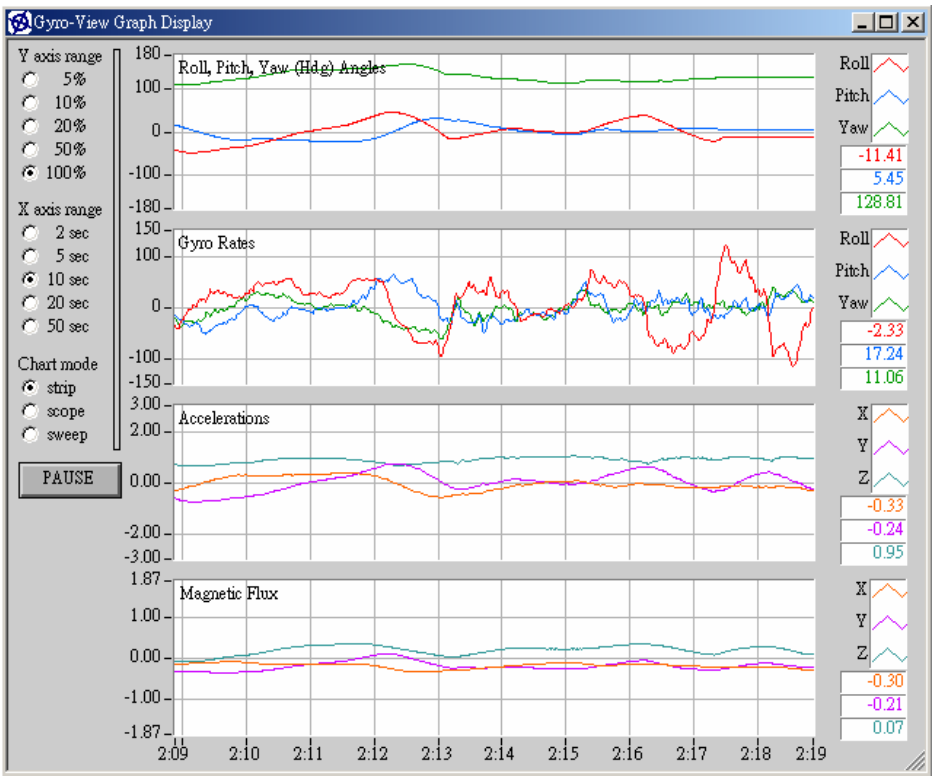


圖 1-7-4 加速規量測運動平台之線性速度、加速度及角速度等機構之動態資料

子題二： 動態模擬單元與虛擬場景之建構

本子題負責動態模擬單元與虛擬實境場景建構兩部份，由於此兩部份對擬真的效果佔著極重要的地位，且在工作上可將這兩部份放在同一台電腦或程式執行，因此本子題一起負責此兩部份的進行工作。此外由於本產學案目的為發展一符合美國飛航單位(Federal Aviation Administration, FAA)法規之飛行模擬訓練機，因此本子題以「飛行模擬」為研究核心，目標進度包括動態模擬單元部份的各種飛行狀態的模擬，如常見的起飛、降落、碰撞、亂流、爬升等等現象，都要能適切地模擬出真實的飛行感受；以及虛擬實境場景部份的周圍環境、飛機駕駛艙的繪製、碰撞測試、音效播放等功能也盡可能符合真實情況。

基於過去執行成效，本子題繼續研究探討如何綜合利用物理運動定律，輸出、輸入行為，以及人為修正訊息等，來精確地架構出真實系統模式，進一步將所發展的即時動態模擬系統由現有的技術層面提升至符合 FAA 所制訂之飛行模擬訓練機的標準。在我們的研究中已經模擬出 F16 戰機動態模擬系統，並加入系統識別判斷戰機動作是否正確，另外也用數值式的方法建構出飛行的動態模型。而在場景製作方面，本子題增加了聲效、場景視點之變化，利用多台電腦控制多台投影機以及網路串連的方式，成功地建構出多螢幕環繞場景，使得模擬駕駛更有深入其境之感覺。

2.1 F-16 戰機動態模擬

飛行模擬器性能表現主要取決於是否能充分反應出真實飛機的動態特性，而飛機的動態特性取決於空氣動力數學模式是否精確。這裡將根據飛行力學建立飛機的數學模式，亦即所謂的六自由度飛行動態運動方程式。而飛行模擬器即是利用此組運動方程式，配合事先建立的龐大氣動力資料表，經由數值查表及積分運算模擬出飛機的各種動態。

2.1.1 座標系統定義

在推導動態方程式之前，我們需先建立相關的座標軸系統，以正確表達出系，在此所採用的座標系統包括：機體座標（Aircraft-Body Axis）如圖 2-1-1、圖 2-2-1(左)，地球地理座標 (Local Axis)如圖 2-1-2(右)、2-1-3 及尤拉角座標(Euler Axis)如圖 2-1-4【2-1～2-3】。

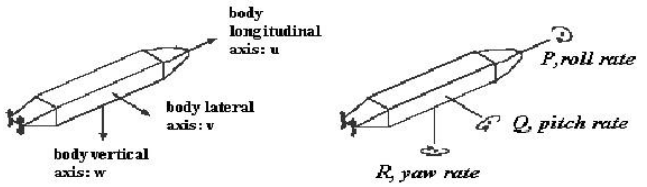


圖 2-1-1　機體座標示意圖

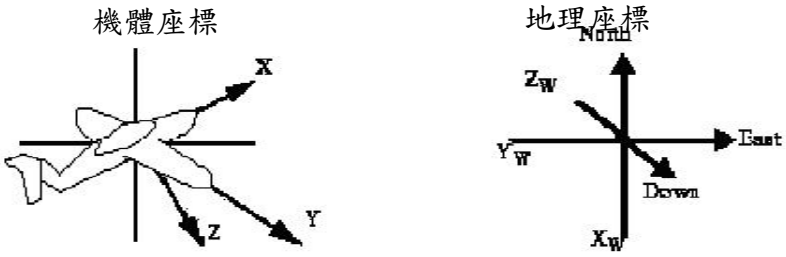


圖 2-1-2 機體座標（左）與 地理座標（右）示意圖

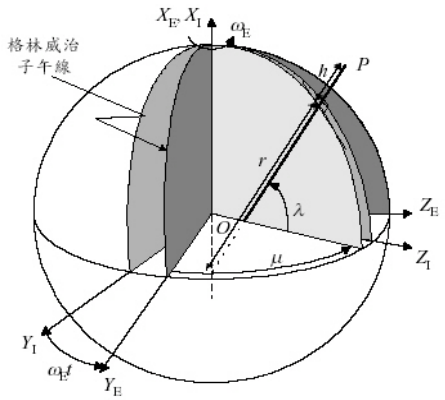


圖 2-1-3 地球經緯與座標高度之關係圖

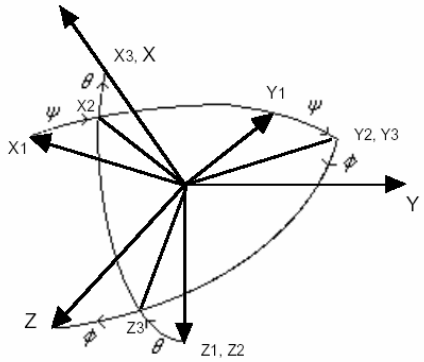


圖 2-1-4 尤拉角座標示意圖

地球地理座標系與機體座標之間因受力定義不同，故推導轉換座標提供快速參考，如圖 2-1-2 所示，可對各軸運算轉換得以下各公式【2-1】

針對ψ轉動

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{2.1}$$

針對θ轉動

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} \tag{2.2}$$

針對ϕ轉動，

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} \tag{2.3}$$

合併上述三個矩陣可得

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ -\sin\psi\cos\theta+\cos\psi\sin\theta\sin\phi & \cos\psi\cos\phi+\sin\psi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\psi\sin\theta+\cos\psi\sin\theta\sin\phi & -\cos\psi\sin\phi+\sin\psi\sin\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{2.4}$$

並以$[LtoB]$代表地球地理座標轉換至機體座標，而機體座標轉換至地球地理座標則以$[LtoB]^{-1}$表示之。

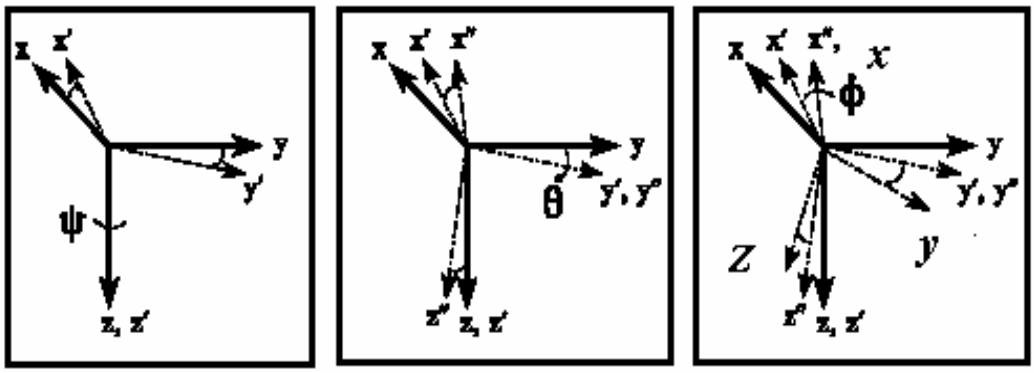


圖 2-1-5　尤拉角座標轉換示意圖

2.1.2 F-16 戰機簡介

自波斯灣戰爭後，中共深刻領悟到高科技武器裝備在現代戰爭中所扮演的重要性，近幾年來一直積極執行軍事武器現代化，再加上其從未放棄武力犯台之作法，對我國國防安全之威脅性益形加重。在歷經多年的努力之後，我國軍方期盼多年的「二代戰機」─IDF、F-16 與幻象兩千等新型戰機終於得以翱翔在台灣領空，所以本子題將朝向 F-16 戰機飛行模擬研究。

飛機運動行為的影響表現在其機體的操縱面上，其俯仰、側滾及偏航運動是分別受到水平尾翼、副翼及方向舵所控制以完成飛行動作。為了要正確地模擬一飛行器，我們必須先針對欲要模擬之系統有些初步且正確的認知，本子題今年依舊接續去年的成果繼續探討 F-16 戰機，眾所皆知，F-16 戰機共有一對副翼(Aileron)、一對水平尾翼(Elevator)及一片方向舵(Rudder)五個控制面，如圖 2-1-6 所示。而且在其基本規格如下列所示與外型規格如圖 2-1-7 所示。

F-16 戰機基本規格

Primary Function: Multirole fighter

Builder: Lockheed Martin Corp.

Power Plant: F-16C/D: one Pratt and Whitney F100-PW-200/220/229 or General Electric F110-GE-100/129

Thrust: F-16C/D, 27,000 pounds

Length: 49 feet, 5 inches (14.8 meters)

Height: 16 feet (4.8 meters)

Wingspan: 32 feet, 8 inches (9.8 meters)

Speed: 1,500 mph (Mach 2 at altitude)

Ceiling: Above 50,000 feet (15 kilometers)

Maximum Takeoff Weight: 37,500 pounds (16,875 kilograms)

Range: More than 2,000 miles ferry range (1,740 nautical miles)

圖 2-1-6　F-16 戰機控制面

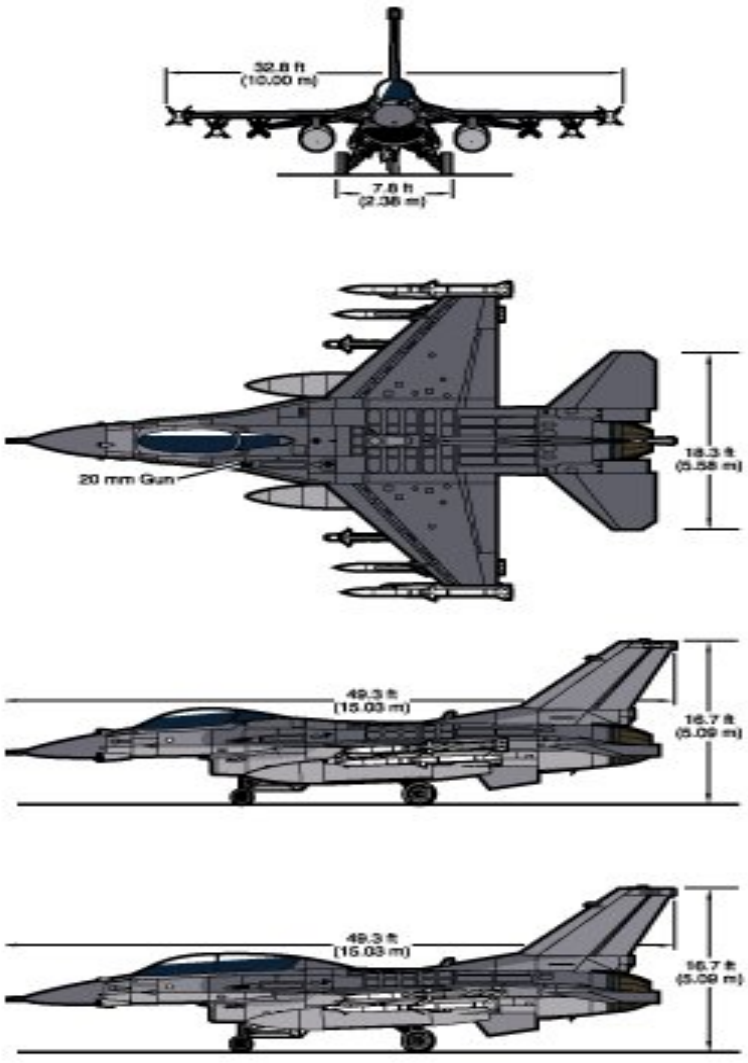


圖 2-1-7　F-16 戰機外形規格圖

2.1.3 F-16 戰機飛行動態系統建立

飛機視為剛體，就運動而包含有六個自由度（Degree of freedom），故其運動可用六個變數來陳述；互為垂直的三個重心分速 u,v,w；及對穿過重心三垂直軸的旋轉，以三個角速度 p,q,r。而角位移則以尤拉角 φ,θ,ψ 表示。物體與空氣作相對運動時作用在物體上的力，簡稱氣動力，而航空器基本受力如圖 2-1-8 所示。它由兩個分布力系組成：一是沿物體表面法線方向的法向分佈力系，另一是在表面切平面上的切向分佈力系。空氣動力通常就是指這兩個力系的合力。將空氣動力分解為三個方向上的分量。x 軸平行於氣流方向且正向與氣流方向相反，z 軸在飛行器對稱面內與軸垂直且正向指向航空器上方，y 軸指向右翼，則合力在 x,y,z 三個軸上的分量分別稱為阻力、側向力，升力。若空氣動力作用點與飛行器重心不重合，則飛行器還受到一個合力矩的作用，它在 x,y,z 三個軸上的分量分別稱為滾轉力矩、偏航力矩和俯仰力矩，因此，航空器所受總力可由三個部分所構成如圖 2-1-9 所示【2-1】。

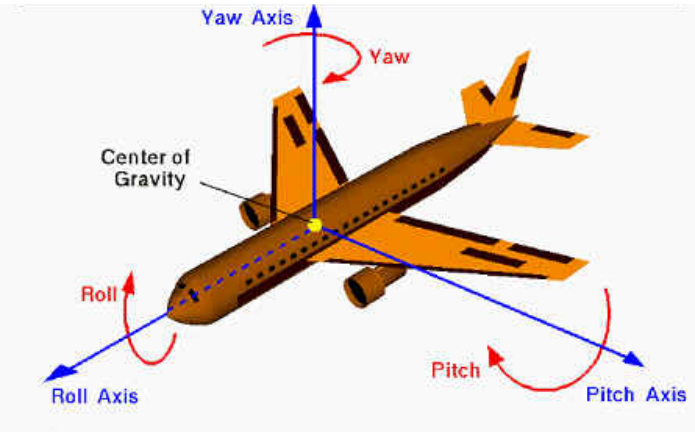


圖 2-1-8 航空器座標定義圖

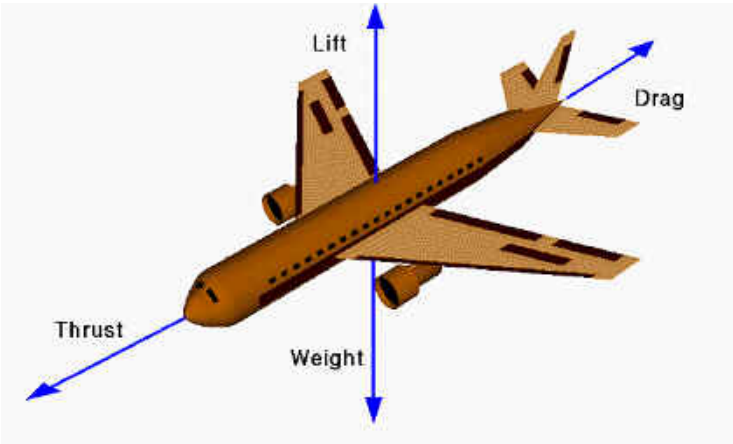


圖 2-1-9　航空器基本受力圖

空氣動力含引擎推力部分其總力表示為

$$Fx = \bar{q}Cx - mg\sin\theta + Thrust \tag{2.5}$$
$$Fy = \bar{q}sCy + mg\sin\phi\cos\theta \tag{2.6}$$
$$Fz = \bar{q}sCz + mg\cos\phi\cos\theta \tag{2.7}$$

產生的空氣動力力矩為

$$L = \bar{q}sbC_l \tag{2.8}$$
$$M = \bar{q}scC_m \tag{2.9}$$
$$N = \bar{q}sbC_n \tag{2.10}$$

式中 $\bar{q}$ 為動壓（Dynamic Pressure）、s 為機翼面積為、b 翼長、c 為機翼參考弦長與 C_l，C_m，C_n 分別為滾轉軸、俯仰與偏航軸力矩之係數

(1) 升　力(Lift Force)

$$F_{Lift} = \left[C_L + C_{L_{\delta E}} \times \delta_E + \left(C_{L_\alpha} \times \dot{\alpha} + C_{L_Q} \times Q\right) \times {}^{C}\!/_{2V} + C_{L_0}\right] \times Q_{DP} \times S_{ref}$$

(2) 阻　力(Drag Force)

$$F_{Drag} = \left[C_D + C_{D_{MDD}} + C_{D_{\delta E}} \times \left|\delta_E\right| + C_{D_{\delta R}} \times \left|\delta_R\right| + C_{D_{\delta A}} \times \left|\delta_A\right| + C_{D_0}\right] \times Q_{DP} \times S_{ref}$$

(3) 側向力(Side Force)

$$F_{Side} = \left[C_{Y_\beta} \times \beta + C_{Y_{\delta R}} \times \delta R + \left(C_{Y_P} \times P + C_{Y_R} \times R\right) \times {}^{B}\!/_{2V}\right] \times Q_{DP} \times S_{ref}$$

(3) Pitching Moment

$$\begin{aligned} M_{Pitch} = {} & \left[C_M + C_{M_{\delta E}} \times \delta E + \left(C_{M_\alpha} \times \dot{\alpha} + C_{M_Q} \times Q\right) \times {}^{C}\!/_{2V} + C_{M_0}\right] \times Q_{DP} \times C \times S_{ref} \\ & + \left(F_{Lift} \times \cos\alpha + F_{Drag} \times \sin\alpha\right) \times \left(X_{CG} - X_{ref}\right) / 12 \\ & - \left(F_{Drag} \times \cos\alpha - F_{Lift} \times \sin\alpha\right) \times \left(Z_{CG} - Z_{ref}\right) / 12 \end{aligned}$$

(4) Rolling Moment

$$\begin{aligned} M_{Roll} = {} & \left[C_{l_\beta} \times \beta + C_{l_{\delta A}} \times \delta A + \left(C_{l_P} \times P + C_{l_R} \times R\right) \times {}^{B}\!/_{2V} + C_{l_{\delta R}} \times \delta R\right] \times Q_{DP} \times B \times S_{ref} \\ & + \left(F_{Lift} \times \cos\alpha + F_{Drag} \times \sin\alpha\right) \times \left(Y_{CG} - Y_{ref}\right) / 12 \; - \; F_{Side} \times \left(Z_{CG} - Z_{ref}\right) / 12 \end{aligned}$$

(6) Yawing Moment

$$\begin{aligned} M_{Yaw} = {} & \left[C_{n_\beta} \times \beta + C_{n_{\delta R}} \times \delta R + \left(C_{n_P} \times P + C_{n_R} \times R\right) \times {}^{B}\!/_{2V}\right] \times Q_{DP} \times B \times S_{ref} \\ & - \left(F_{Drag} \times \cos\alpha - F_{Lift} \times \sin\alpha\right) \times \left(Y_{CG} - Y_{ref}\right) / 12 \; + \; F_{Side} \times \left(X_{CG} - X_{ref}\right) / 12 \end{aligned}$$

2.1.4 大氣環境模型

飛機於空中飛行與大氣環境交互作用產生浮力、阻力等等。在大氣環境中不同的高度擁有不同的空氣密度、靜壓力及動壓，同時也影響飛機推力輸出。本子題引用美國太空總署 NASA 所提供的 1976 年標準大氣環境模型作為飛行動態系統的大氣模擬環境，詳細資料如表 2-1-1 所示。

表2-1-1 1976 Standard Atmosphere Characteristics

Altitude（Feet）	Density（Ratio）	Pressure（Ratio）	Speed Sound（KTAS）	Temp（F）	Airspeed（TAS/IAS）	**Mach 0.8**（KIAS）
0	1	1	59	622	1	530
2500	0.9289	0.9129	50	656	1.0376	506
5000	0.8617	0.8321	41	650	1.0773	483
7500	0.7983	0.7527	32	645	1.1192	461
10000	0.7386	0.6878	23	639	1.1636	439
15000	0.6295	0.5646	5	627	1.2604	398
20000	0.5332	0.4599	-12	615	1.3695	359
25000	0.4486	0.3716	-30	602	1.4930	323
30000	0.3747	0.2975	-48	590	1.6336	289
35000	0.3106	0.2360	-66	577	1.7943	257
40000	0.2471	0.1858	-67	574	2.0117	228
45000	0.1945	0.1462	-67	574	2.2675	202
50000	0.1535	0.1151	-67	574	2.5557	180
55000	0.1206	0.0906	-67	574	2.8796	159
60000	0.0914	0.0687	-67	574	3.3084	139
65000	0.0747	0.0562	-67	574	3.6576	126

2.1.5 F-16 戰機系統識別

整體飛行器數學模型中包含了該飛行器動態的表現，如引擎、起落架、飛行控制面的狀態以及外在環境所造成的影響等，為了研究其非線性系統的整體行為就必須建立非線性數學模型。系統識別一般可稱為逆向工程，利用系統輸入與系統響應來推導系統模型，但其非線性數學模型複雜的程度即使在高速電腦迅速發展的今日應用傳統的數值方法亦相當費時。

基本上，對系統的運動方式有足夠的知識，可以利用牛頓運動定律推導出數學動態模型，但數學模型往往是以非線性微分方程或非線性差分方程形式表示，對這類數學模型的辨識可以採用線性化展開成特殊函數等方法。但是如果對系統瞭解得不夠充分，則在推導其數學模型往往就顯得困難重重。為了解決上述問題，系統辨識就是不管系統本身的真實結構如何，而著力去找出能達到要求精度的系統輸入輸出關係的近似模型。一般常見且有效的近似方法有兩種，一種是利用泛

函級數展開方法，另一種是用多項式逼近方法。不論使用哪一種方式描述系統的行為，模型的選擇皆取決於模型的可辨識性、參數估計的難易程度和模型適用性檢驗等。

針對 F-16 戰機的非線性空氣動力係數作系統識別，初期依據 F-16 戰機的次音速風洞實驗數據，利用一般數值解法作參數識別與狀態估測來找出飛機的最佳數學動態模式之系統參數。但在建立其動態模式過程中，發現其運算由於數據量過於龐大及數值解法中往往需要複雜的積分運算，並且整體流程需採疊代方式不停地修正誤差，使得電腦處理速度非常緩慢，雖然採用數值解法可求得準確的結果，但耗費的時間卻相當龐大。故必須研究其他有效且適合的計算機程式運算法，例如代數及多項式逼近來尋求近似解等，來解決需要較大的電腦記憶空間及耗費長時間計算的問題。

為了解決大量的運算時間與記憶空間，並獲得更加的系統鑑別效果，本子題應用生物控制論中的演化方法以多項式來近似地表示非線性系統的輸入輸出關係。這裡提出以基因程式(Genetic Programming)為基礎的啟發式自組織建立模型(Self-Organization Modeling)來建立多項式函數所組成的級數，以其具有可近似任意函數的的性質，進而完成現代飛行器的空氣動力的非線性數學模型。圖 2-1-10 為傳統常見的系統鑑別方式，而圖 2-1-11 為本子題所設計的系統鑑別方式。

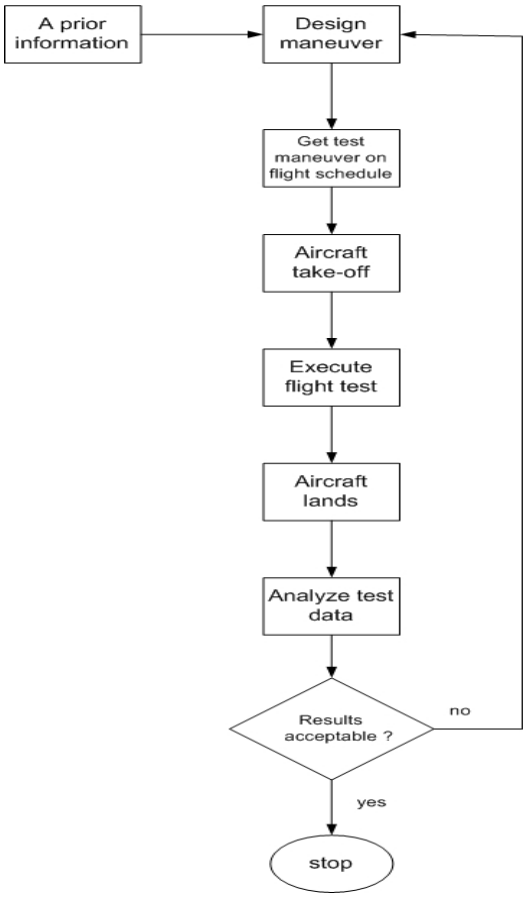


圖 2-1-10　慣性系統的飛行辨識流程

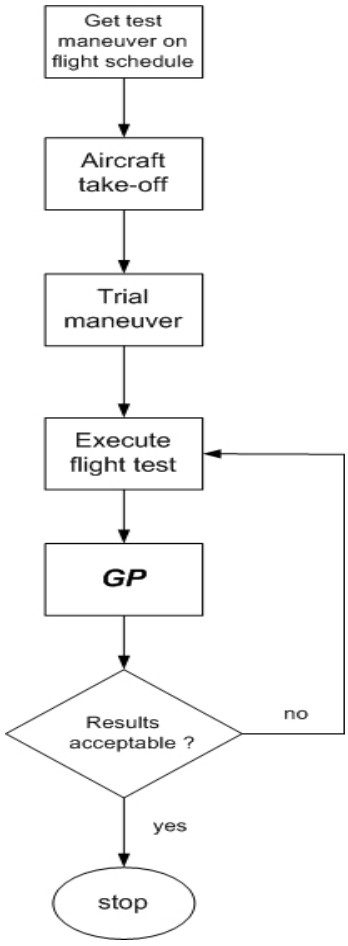


圖 2-1-11　飛行中戰機系統辨識

2.1.6 基因程式

基因程式（GP）其原理大都來自於自然界生物演化的機制－就是生物學家達爾文提出的「物競天擇，適者生存」的演化論。演化論的大意是說，在有限資源的環境下，生態群體(population)中的各類生物個體(individual)，必須為了生存而互相競爭。在競爭中，失敗者被選擇淘汰，而勝利者除了生存外，也增加它繁殖子代的機會。個體所繁殖的子代會經由各種遺傳機制，而和個體本身類似。在遺傳過程，子代可能會有和父代不同的變異，而導致新物種的產生。因此隨著每一世代(generation)中，新生個體的繁殖與失敗個體的淘汰，群體中的個體也就演化得愈來愈適應環境。美國史丹福大學的教授柯薩 (Koza) 【2-4】，針對電腦研究上的一項重點問題，“如何使電腦能在不提供它詳盡程式指令下，自發學習解決問題？”而創造了「基因程式」。基本上他是根據遺傳演算法的機制，將原本以位元串構成的單元，推廣到以程式構成的單元。而每個單元的適應度是由其程式對環境的執行結果而定。所以「基因程式」，就是將傳統「遺傳演算法」的搜尋位元串狀態空間，推廣到搜尋程式行為空間。而隨著其單元結構由位元串修改成程式，所有相應的遺傳機制也必須作調整，以對應各單元的程式結構。為了讓單元程式可以演化，所以其中的程式必須被表示為剖析樹結構。因此柯薩就採用，利於樹狀結構操作的 LISP 語言，作為「基因程式」的程式語言。「基因程式」的機制中，與「遺

傳演算法」的不同點大致上可有以下幾點重要不同之處

1. 在初始時必須依照環境，決定演化單元程式所需的函式集(function set)，如算術運算子、數學函數、邏輯運算子等與元素集(terminal set)，通常是一般的常數或是變數，以隨機組合函式集與元素集來產生初始群體的各單元程式。
2. 重複執行下列子步驟直到符合目標的衡量標準。
   (1) 執行每一代演化的族群所產生的程式，並根據對其系統解的近似程度定義適應值(fitness value)。
   (2) 進行演化下一代族群程式；其機制依據族群個體中適應值(fitness value)的高低，可以排定每個個體適應度的順序，使得適應值高個體有較高機率被選擇來交配(crossover)而產生下一代族群。有二種方法進行演化下一代族群
      i. 複製：如圖 2-1-12，先從母代中選二個欲做基因交換的樹，分別在二顆樹上隨機選取二個交換節點，將自這個節點以下的二顆子樹交換，若是二個被選取的交換節點都是根節點的話，交換基因後自然還是原來的那二顆樹。
      ii. 交配：如圖 2-1-13，若隨機選取選取交換節點根節點外之任意節點的話，一旦做基因交換後，就產生兩顆新的子樹。
3. 根據基因程式的控制因素，如族群的大小，和想要跑幾個世代而產生最後的程式可能為系統的最佳解或是近似解。

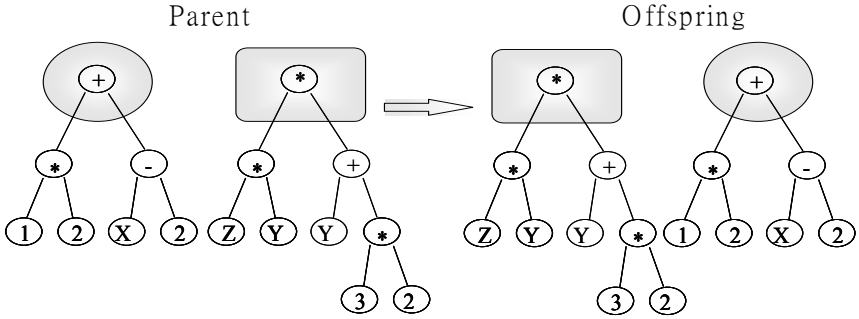


圖 2-1-12　基因程式中個體複製過程

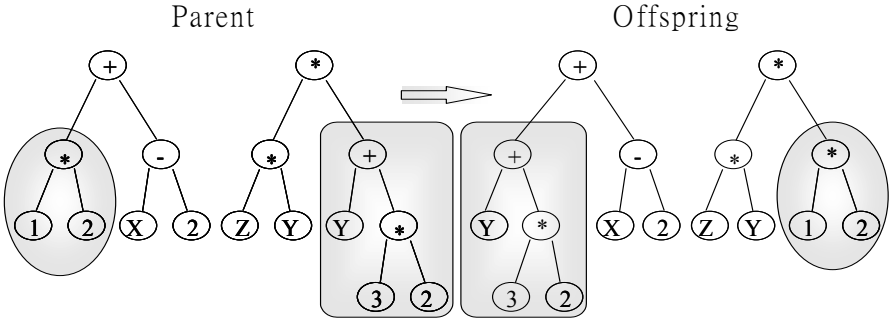


圖 2-1-13　基因程式中個體交配及其子代

2.1.7 啟發式自組織建立模式

啟發式自組織建立模型（Self-Organization Modeling）【2-5】是對無法以經驗等方式建立數學模式的未知系統，應用測試未知系統所收集的數據資料為根據區分為實驗組與檢驗組，以基因程式的模式作為學習機制，將輸入變數作交配（crossover）後所產生下一代族群中的多項式帶入實驗組數據資料，並以其輸出結果以適應函數比對檢驗組數據資料近似的程度定義適應值來選擇或淘汰部分個體，作為下一子代的輸入。最後，如基因程式的控制因素，設定結束條件所產生結果的常態多項式可能為系統的最佳解或是近似解。以下我們將針對這種方法以例子說明此一做法。

(*Step I*)

在初始時必須依照環境也就是測試未知系統所收集的數據資料，決定演化單元程式所需的輸入變數及初始的多項式。如圖 2-1-14 之 Step *I*；在一般動態或靜態系統模擬中可採用二次多項式來建立，亦即

$$y_k = f_k(v_i, v_j) = a_0 + a_1 v_i + a_2 v_j + a_3 v_i v_j + a_4 v_i^2 + a_5 v_j^2 \qquad (2.11)$$

並觀察未知系統中之獨立變數作為輸入，如 $v_1 ..... v_5$，並以二次多項式 $f_k(v_i, v_j) = a_0 + a_1 v_i + a_2 v_j + a_3 v_i v_j + a_4 v_i^2 + a_5 v_j^2$ 作為所要產生的族群函數及其輸出 y_k。

(*Step II*)

如圖 2-1-13 之 Step *II*；將獨立變數 $v_1 ..... v_5$ 根據族群函數以不重複的方式兩兩作交配，帶入二次多項式中產生第一子代。而每一子代的個數為若前一代數量為 m 則每一子代可產生 m(m-1)/2 的個體。以圖 2-1-13 之 Step *II* 為例，第一代族群中共可產生 10 個個體，且第一子代族群中所產生如（2.11）式的多項式，其次方數及變數個數小於或等於 2。

(*Step III*)

如圖 2-1-15，將輸入變數作交配（crossover）後所產生下一代族群中的多項式帶入未知系統的實驗組數據資料，將輸出的結果配合適應函數評估（fitness criterion）出族群中常態多項式對檢驗組數據資料的近似程度定義出個別適應值來選擇或淘汰部分個體，作為下一子代的輸入。常用的適應函數如最小誤差平方法作為評估方法。而在本章在所採用的適應函數稱為預測誤差法（Predict Square Error）將於下一節中說明。在此一步驟中，我們將適存的子代所產生的輸出以代數的方法取代並代表其常態多項式，亦即 $y_k = w_k$，作為下一子代交配的輸入。

(*Step IV*)

如圖 2-1-15 Step *IV*，其中黑方格代表族群中所選擇適應值高的個體，將最佳之子代個體重覆圖 2-1-13 之 Step *II*，而得到下一世代的群體，新群體子代中所產生的多項式中由於包含了前一父代第二次多項式，故其展開的多項式中其次方數及變數個數小於等於 4。

(*Step V*)

如圖 2-1-16，重複圖 2-1-14 Step *III*，淘汰部份個體。在此一步驟中，我們將適存的子代所產生的輸出以代數的方法取代並代表其常態多項式，亦即 $y_k = Z_k$，作為下一子代交配的輸入。

(*Step VI*)

如圖 2-1-16，經過三代的演化後顯示出選擇各子代的結果，並將 w_k、z_k 所代表的常態多項式於最後的二次多項式中展開，所產生的常態多項式中，由於包含了祖父代及父代的二次多項式，故其展開的多項式中其次方數及變數個數小於或等於 8，且其結果為近似於未知系統測試數據的特性。

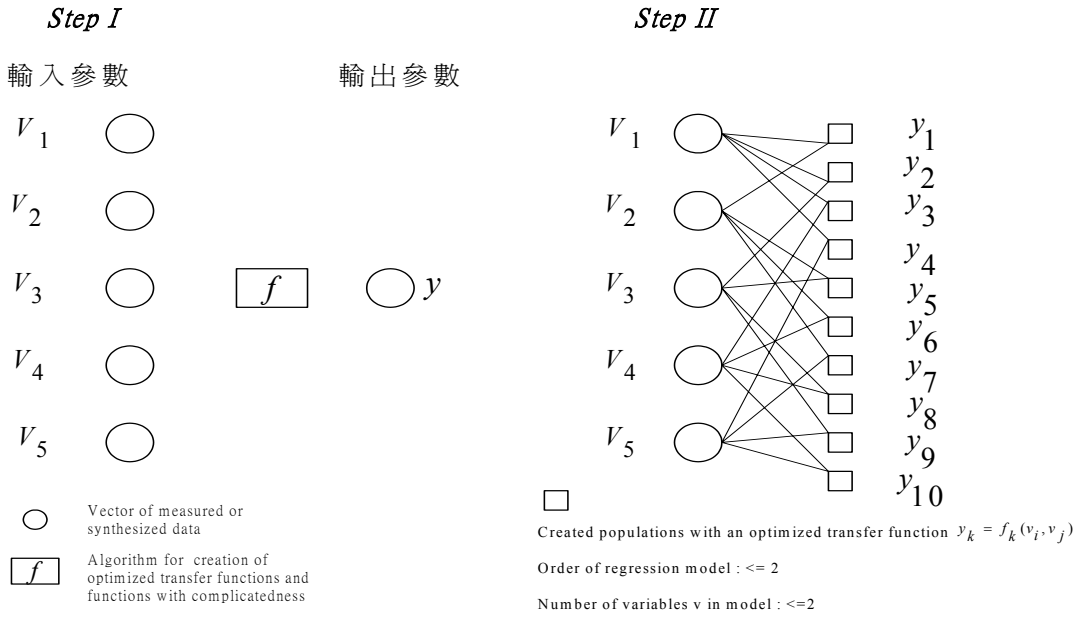


圖 2-1-14　啟發式自組織建立模式流程一

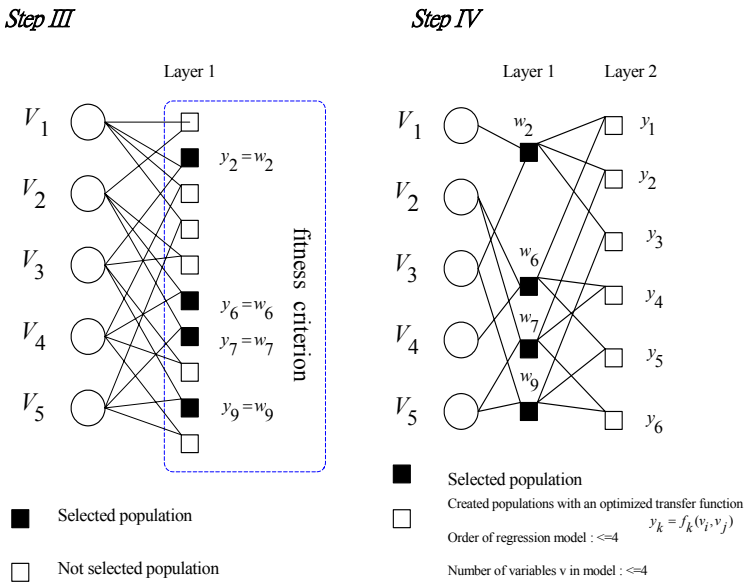


圖 2-1-15　啟發式自組織建立模式流程二

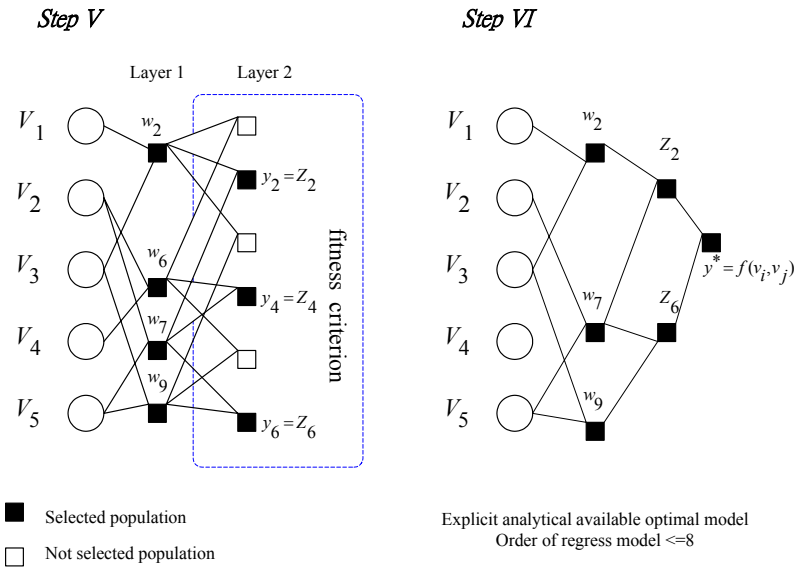


圖 2-1-16　啟發式自組織建立模式流程三

由數值分析的角度來看，由於演化過程中的系統會趨向複雜。因此演化可視為適應性複雜系統的變化過程，尤其在大部份複雜非線性的數學問題上，因為缺乏解析決定的求解方法，啟發式自組織建立模式便成為最有效的數值分析工具。另一方面啟發式自組織建立模式可視為類似搜尋的最佳化演算法。因為其本質是非決定性的，所以能搜尋問題空間內所有的可能解答。而在啟發式自組織建立模式的群體中，大量個體追隨環境利基，聚集而形成各式族群。不同族群也就代表了、用以解決問題的不同途徑與其資訊，這就是啟發式自組織建立模式含有多樣性以解決問題的能力。

2.1.8 適應函數

在產生新一代的多項式後，利用適應函數（Fitness function）來進行選擇適存度較高的個體生存，淘汰部份個體流程如圖 2-1-17 所示，其中適應函數亦稱為價值函數（cost function），在以風洞實驗數據為標準進行演化式計算時我們採用最小平方價值函數（Least-Squares Cost Function）為基礎的預測誤差法（Predict Error Method）來作為適應函數稱為預測最小平方誤差（Predict Squares Error）。

設非線性系統的模型為

$$y=f(x,\theta) \tag{2.12}$$

式中 y 是系統的輸出，x是輸入，θ 是參數(它們可以是向量)。在估計參數時模型的形式 f 是已知的，經過輸入 N 筆實驗數據使得 $\{(y_i, x_{ip}): i=1,.....N \quad p=1,...m\}$，$N>m$。因方程式的個數大於未知數的個數時通常無一解可滿足，但可求得最小平方解係數解 $\hat{\theta}$，帶入原方程式 $y=f(x,\theta)$ 求

得預測的輸出 $\hat{y}$，及目標函數最小誤差平方和 $J=\sum_{i=1}^{N}[y_i-f(x,\theta)]^2$ 求得最小平方誤差 $\hat{J}$，由於考慮數據分佈的特性，應用機率統計的方法利用平均數與變異數可進一步預測出每一代生存的最佳標準，亦即 $PSE=\frac{\hat{J}}{N}+\sigma^2\frac{m}{N}$ 【2-4】。其中 $\sigma^2=\frac{1}{N}\sum_{i=1}^{N}(Y-\bar{y})^2$，亦即表示檢驗數據的變異性。其中 Y 為風洞實驗數據 $\bar{y}$ 為 N 次利用風洞實驗數所建立的常態多項式所求得預測 $\hat{y}$ 的平均值，m 為多項式項數。所以 PSE 的值即是用來作為篩選族群個體的標準。

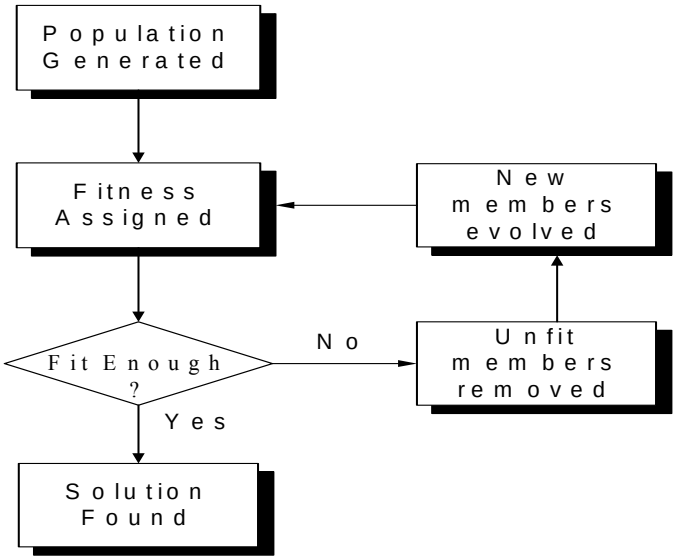


圖 2-1-17　適應函數選擇流程

2.1.9　F-16 戰機非線性空氣動力系統識別

在此章節，我們將上述之利用基因程式為基底的系統識別方法應用於 F-16 戰機模型，啟發式自組織建立模式乃藉由風洞測試數據的特性建立常態多項式來近似模擬，在本節針對原報告書中提到的縱向飛行系統 C_x、C_z、C_m 橫向飛行系統 C_y、C_l、C_n 及最後之最佳多項式結果與風洞測試作一比較，其結果誤差程度可達 5%的接受程度，證明此一方法的可靠度。因此對於未知系統的模型建立，啟發式自組織建立模式提供了一種快速而有效的方法。其縱向飛行系統 C_x、C_z、C_m 之比較差異如圖 2-1-18 ~ 2-1-20，而橫向飛行系統 C_y, C_l, C_n 比較如圖 2-1-21 ~ 2-1-23，觀察以上之系統識別結果，我們可以很發現整個識別效果達到我們使用之要求範圍，接著我們將整個 F-16 戰機系統識別之結果放入整個飛行動態模擬系統之中，希望藉由此系統識別之結果可以更加真實地模擬出飛機飛行動態。

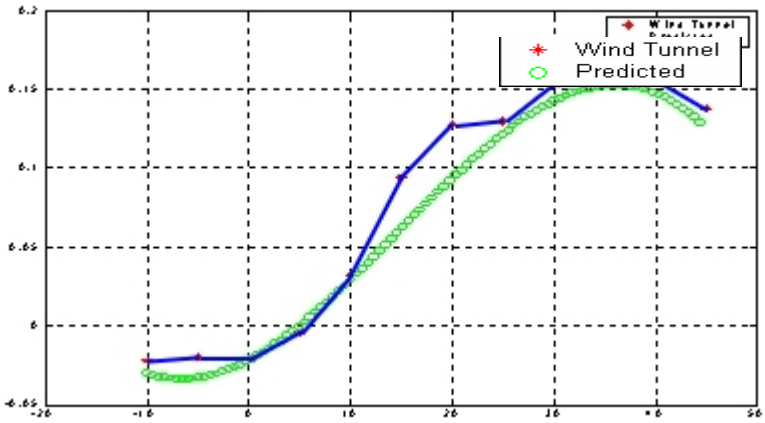


圖 2-1-18　實際與系統識別之C_x係數

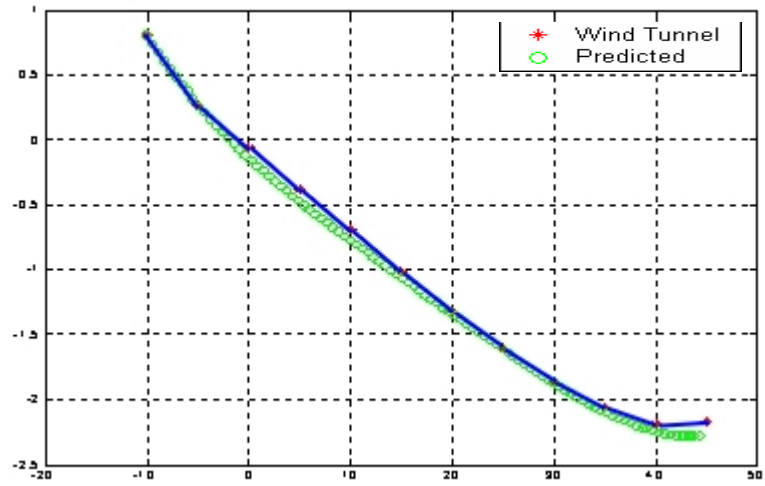


圖 2-1-19　實際與系統識別之C_z係數

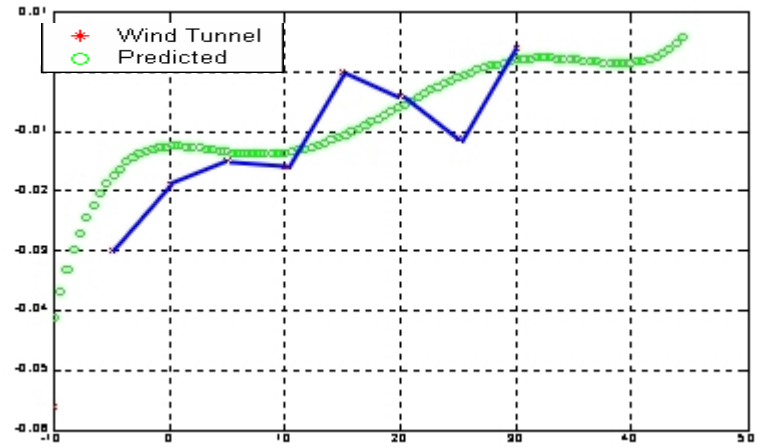


圖 2-1-20　實際與系統識別之C_m係數

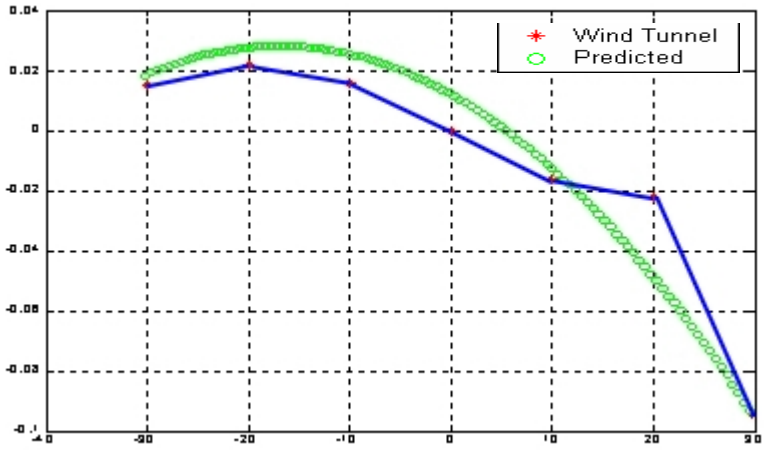


圖 2-1-21　實際與系統識別之C_l係數

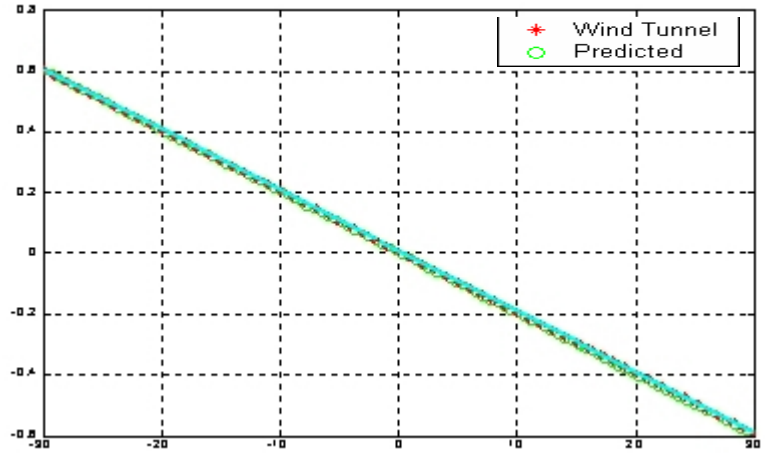


圖 2-1-22　實際與系統識別之C_y係數

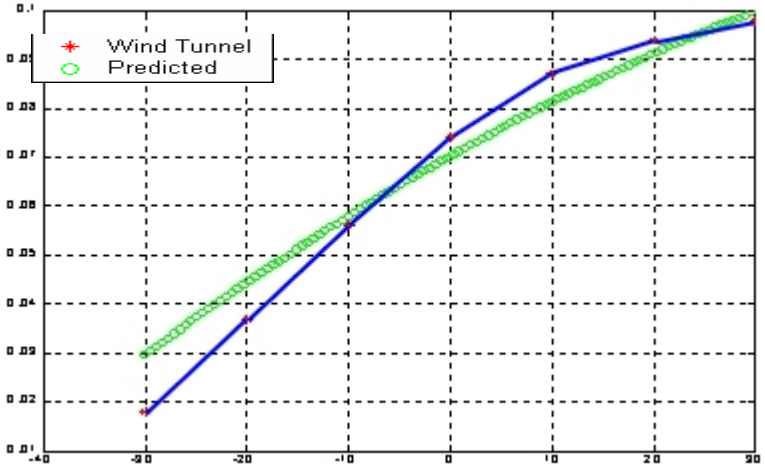


圖 2-1-23　實際與系統識別之C_n係數

2.1.10 數值動態模式模擬載台

近年來，隨著科技的進步，電腦的計算能力及其可能的應用技術亦日新月異，電腦的應用已經不再侷限於一般的計算功能及簡單的模擬。在本計劃的研究中，我們將利用電腦建立數值式直昇機的模擬系統，並搭配 3D 動態場景系統、飛彈模擬系統，來建立一空戰電腦模擬系統。此一系統可提供直昇機駕駛員在訓練時的一重要工具。

而為了使直昇機模擬系統能有效的呈現真實的狀況，以使模擬的結果能真正用於相關任務的執行，我們必須使直昇機的模擬是擬真的。由於直昇機的動態方程式及其飛行品質等有相當大的變化，因此本計畫首先針對直昇機的飛行特質及其相關的動態行為做了解與分析。在文獻上，直昇機動態運動的數學表示式為

力方程式：
$$\dot{u} = rv - qw + \frac{X}{M_a} - g\sin\theta$$
$$\dot{v} = pw - ru + \frac{Y}{M_a} + g\sin\phi\cos\theta$$
$$\dot{w} = qu - pv + \frac{Z}{M_a} + g\cos\phi\cos\theta$$

力矩方程式：
$$\dot{p} = (c_1 r + c_2 p)q + c_3 L + c_4 N$$
$$\dot{q} = c_5 pr - c_6(p^2 - r^2) + c_7 M$$
$$\dot{r} = (c_8 p - c_2 r)q + c_4 L + c_9 N$$

姿態方程式：
$$\dot{\phi} = p + \tan\theta(q\sin\phi + r\cos\phi)$$
$$\dot{\theta} = q\cos\phi - r\sin\phi$$
$$\dot{\psi} = \frac{q\sin\phi + r\cos\phi}{\cos\theta}$$

其中狀態 $x = [u \;\; v \;\; w \;\; p \;\; q \;\; r \;\; \psi \;\; \theta \;\; \phi \;\; \cdots]$，$[u \;\; v \;\; w]$及$[p \;\; q \;\; r]$分別為機體座標系統下三個軸向的速度及角速度，$[\theta \;\; \psi \;\; \phi]$為機體姿態角(Euler angle)。此外，也可以包括在慣性座標下的位置、葉片的擺動角度等。力矩方程式的係數如下：$c_1 = \frac{(I_{yy} - I_{zz})I_{zz} - I_{xz}^2}{\Gamma}$，$c_2 = \frac{(I_{xx} - I_{yy} + I_{zz})I_{xz}}{\Gamma}$，$c_3 = \frac{I_{zz}}{\Gamma}$，$c_4 = \frac{I_{xz}}{\Gamma}$，$c_5 = \frac{I_{zz} - I_{xx}}{I_{yy}}$，$c_6 = \frac{I_{xz}}{I_{yy}}$，$c_7 = \frac{1}{I_{yy}}$，$c_8 = \frac{I_{xx}(I_{xx} - I_{yy}) + I_{xz}^2}{\Gamma}$，$c_9 = \frac{I_{xx}}{\Gamma}$，$\Gamma = I_{xx}I_{zz} - I_{xz}^2$，力方程式中的外力為重力及氣動力，力矩方程式中的外力矩只有氣動力矩。氣動力 X、Y、Z 及氣動力矩 L、M、N 為五個次系統氣動力及氣動力矩的總和，可表示如下：

$$X = X_R + X_T + X_F + X_{tp} + X_{fn} \qquad L = L_R + L_T + L_F + L_{tp} + L_{fn}$$
$$Y = Y_R + Y_T + Y_F + Y_{tp} + Y_{fn} \qquad M = M_R + M_T + M_F + M_{tp} + M_{fn}$$
$$Z = Z_R + Z_T + Z_F + Z_{tp} + Z_{fn} \qquad N = N_R + N_T + N_F + N_{tp} + N_{fn}$$

其中 R, T, F, tp, fn 分別代表主旋翼，尾翼，機身，水平尾翼平面(horizontal tail plane) 及垂直安定翼(vertical fin)。

一般在探討系統的穩定性及設計控制律時，皆是針對某飛行狀態的線性模式來作分析的工作。整個線性化的過程必需針對配平點展問，否則會有殘餘的力和力矩存在，得到的線性模式也就不能反映出非線性模式的特性。因此線性模式的求得必須先指定某一配平狀況(trim condition)，這也稱作飛行狀況。求得此飛行狀況的配平點，再針對此配平點，將非線性模式利用泰勒級數展開，最後得到直昇機在此飛行狀況下的線性模式。首先說明如何指定一般的飛行狀況。如[2-12] 中，對直升機的簡介所談到的，對於飛行品質及飛行動力學的認知，有下列四個參考因素；任務及操作模式，操作環境，直昇機的結構、動力學及飛行動做範圍，以及飛行員及飛行操空介面。在一次的飛行中，可以有幾種不同的任務所組成。而每一任務實際也是分解為不同的任務單元(mission-task-element－MTE)。一個 MTE 是由幾個操控動作所結合而成，其必需也要滿足一些事先定義的性能要求。而操控動作即是一簡單的飛行動作，一般是指某一軸的改變。而一般而言，客觀的飛行品質是定義於操控動作上，而主觀的飛行評價則是以 MTE 來定義的。然而，在現行美國陸軍操作品質需求 ADS-33C[2-13]，則是直接定義於 MTE 上。這是 ADS-33C 和其前身 Mil Spec 8501A[11]極大的不同的地方。 Mil Spec 8501A 在定義參數時，飛機的重量及大小是一非常重要的因素。而以 MTE 為基礎的 ADS-33C 也和固定翼飛行需求 MIL-F-8785C[2-14]也不太一樣。後者是以飛行狀態為任務單元。因此，在非終端飛行狀態的 A 類(以快速運動及精確追隨為主要特質)，則包含了空中交戰、空中加油、及地表低飛。在漸進運動狀態，包含了爬升、空中加油(加油機)及緊急減速。而終端飛行狀態 (以精確飛行路徑控制及漸進運動為主要特質)，則包含了起飛及降落。一般而言，一個 MTE 是無法再分割的行為。而此一行為一般是需要多軸的運動 方能執行的，如降落、側飛(side step)等。而針對 MTE 內容的定義，則是一些限制及性能的要求。如圖 2-1-24 及圖 2-1-25，即分別表示階層式解析民用及軍用任務中的某一 MTE。而這些 MTE 的行為則決定了直昇機飛行行為，也決定了直昇機控制設計的要求，而詳細的定義討論可見[2-12]中的第七章。

在操作環境因素中，主要考慮環境的條件，如溫度，密度標高，風強度及可見度。一般是說明直昇機的可操作環境。而一重要的因素則是外視(outside visual cues, OVC)的品質，也就是偵測直昇機的姿態和環境關係的能力。而此一能力，在直昇機執行貼地飛行(nap of the-earth, NOE)時尤其重要。一般的對偵測直昇機的姿態和環境關係的能力，是以外視條件來定義。而在 ADS-33 則是以可用條件環境(usable cue environment, UCE)來定義。此一描述除了包含 OVC 之外，也包括了其他可用之訊息來源，如影像、儀測等。實際上 MTE 及 UCE 提供了對飛行品質描述的二個重要參考座標。

第三個因素則是直昇機的結構、動力學及飛行動作範圍，也就是在飛行動作範圍內的行為探討，如姿態、穩定度、及控制反應，也是一般控制所探討的範圍。而在控制行為上有四個控制點，三個在主旋轉翼（如圖 2-1-26），一個在

尾旋轉翼。而控制反應上也都有 cross-coupling 的現象，因此在行為控制探討上都必須加以處理，一般都以 on-axis 及 off-axis 來分析，對於這些課題也將是本計畫探討的重點。而最後一個因素則是飛行員及飛行操控界面，這部分則是模擬系統的人機介面所需要處理的，我們也將就可能的範圍內收集必要之資料。

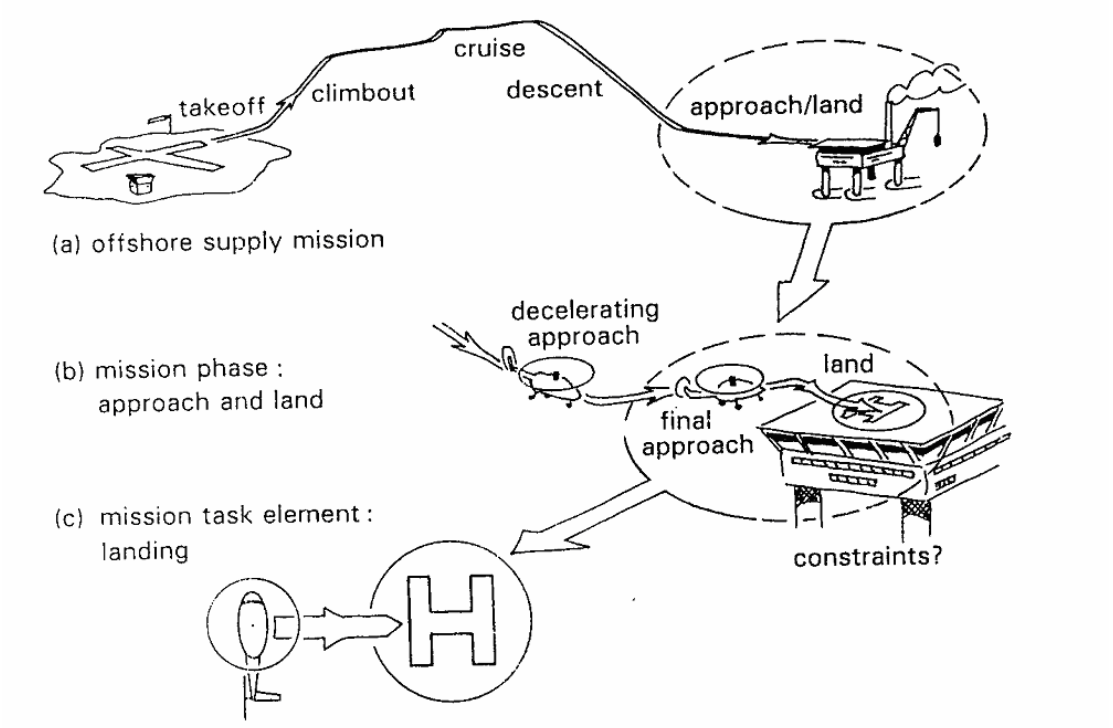


圖 2-1-24 Elements of civil mission – offshore supply

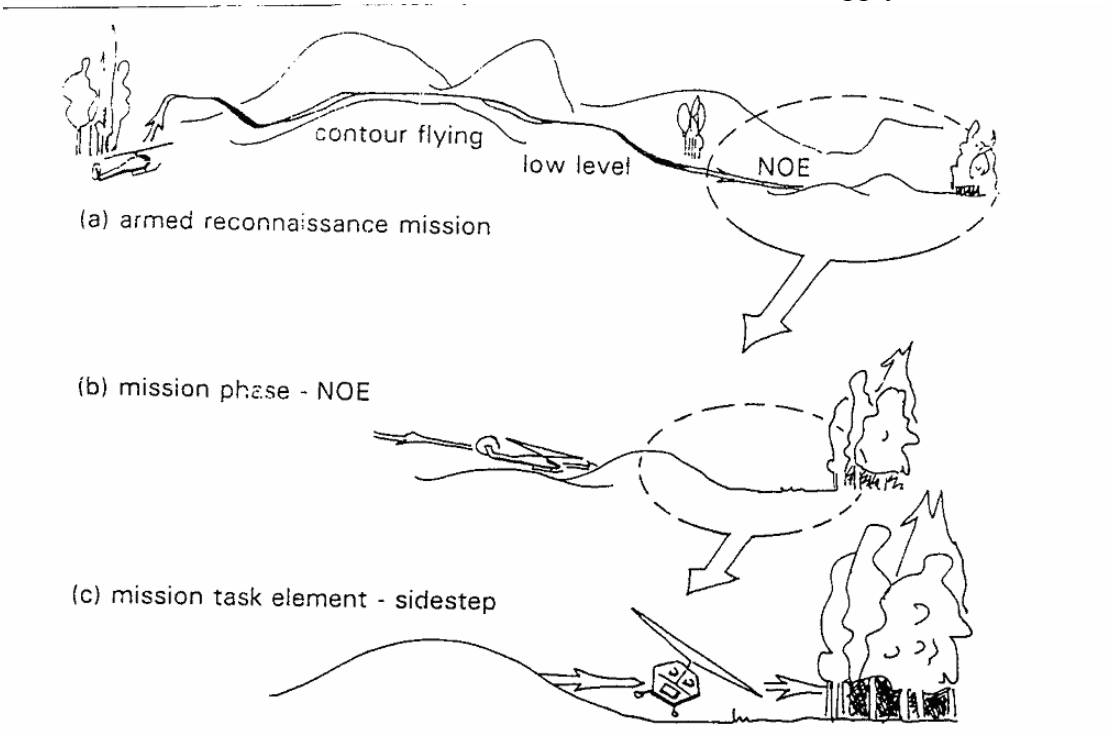
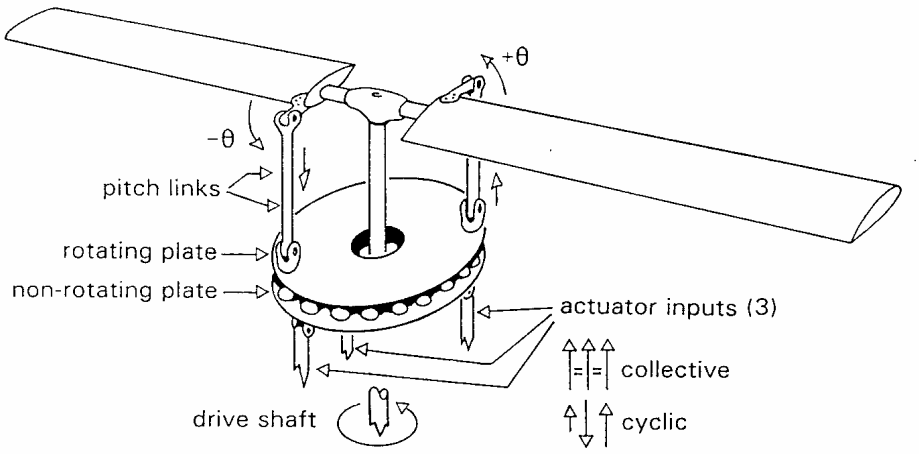


圖 2-1-25 Elements of a military mission – armed reconnaissance



Rotor control through a swash plate

圖 2-1-26.1 直昇機控制

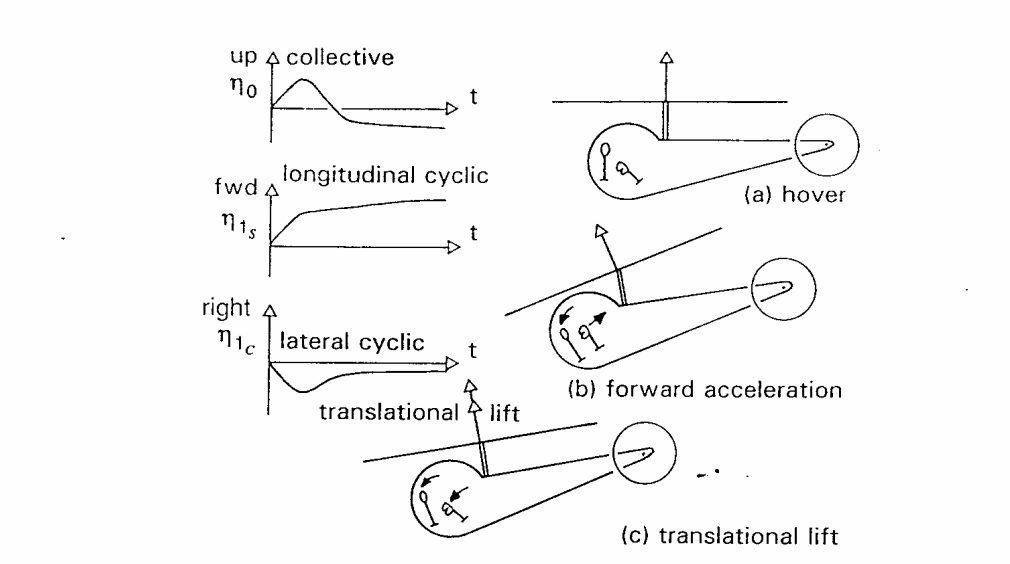


圖 2-1-26.2 直昇機控制

而在我們的研究中，由於在動力系統之分析及係數取得有限的情形下，將以 Level 1 來探討，同時如果能夠取得飛行的遙測資料（也就是每一 Sampling time 時的控制命令、姿態角及其他相關參數），我們也將利用我們已建立之混合式模式建立（Hybrid Modeling）[2-15]來達到更精準的模式建立。所謂混合式模式建立即是利用已知的動態方程式當知識，而加於 TSK 模糊法則[2-17、18]中，再利用遙測資料對模糊法則進行參數調適，以使系統能更精準的表現系統的反應。而針對 Level 1 的模式建立，在[2-12]中有詳細的介紹，我們將這些理論加以實現在我們的程式中，若中科院有真正的規格時，我們可加以模擬之。

而在文獻中[2-16]也利用類神經網路的學習來達到對直昇機系統的線性化，而我們也將引入如是的設計在階層式控制器中，但會用類神經模糊系統來處理。這是因為在類神經網路的系統中，其外插行為是非常難控制的，也就是

當輸入點並非在訓練資料所涵蓋的範圍時，其網路所產生的輸出常常都不是我們想要的，因此當訓練資料之取得不夠完整時，其所學到之系統在面對不尋常之輸入點時，將會有外插現象而造成不可預測之後果。這也是類神經網路在實際運用中常被詬病的問題。而由於直昇機是一動態系統，因此當有一不恰當控制輸入時，其後續之行為皆受影響，而常造成不可收拾之後果。而當利用類神經模糊系統時由於有法則庫的外在形態，因此系統不會有外插現象而達到可預測之效果。同時，當利用法則庫形態時，也可以加入邊界設定規則來解決不可知的輸入點問題。最後在直昇機遙控方面，由於是遙控，所以訊號可能會不持續，所以利用階層式智慧控制之概念，也就是在控制輸入部分可以建立安全外插的效能，應可解決此一問題。

在低階控制為利用導航信息來維持機身在位置，方向及速度上的要求，控制系統包含四個控制環路，roll，pride，yaw，及 collector/throttle 每一環路是以 PID 來控制，並且有調 trim 的功能。而在外環路方面則是去產生位置方向及速度命令給內環路，而其產生的方式主要是依據不同導航模式來產生的。在該系統中，其導航模式有 Ground mode, Runup mode, takeoff mode, waypoint hover mode, waypoint through mode, track hover made, track through mode, waypoint land mode, 及 Pilot assist mode。

Ground mode: 指的是當直昇機是再地面不動時,因此在此模式下，其命令為一非常低速(或稱惰速)的 throttle 命令。

Runup mode: 指的事引擎逐漸上升到足以起飛的狀態，此時之翼葉片的 Pitch 角度仍維持 0。

Takeoff mode: Throttle 設定值仍維持 Runup mode 下的值，而主翼 Pitch 則事逐漸增加，以產生足夠的升力已達到起飛的高度。而任何導航模式中當直昇機接近地面時，嘗試去修正水平位置誤差,將造成直昇機的翻滾。

Waypoint hover mode: 此模式將直昇機帶到一指定的位置及指定的高度，並盤旋指定的時間，其導航的時間是直接產生一路徑從現在位置到指定位置。

Waypoint through mode: 指的是其到達指定位置及高度後，並不停留盤旋而是直接去利用下一定位點來產生下一路徑。

Waypoint land mode: 此一模式分為幾個子模式；首先，系統會要求直昇機到降落地點上空盤旋，而後下降到著地高度(3 feet)，並盤旋至垂直及水平速度夠小(~5 feet/sec)，而後以每秒 0.5feet 的下降速度著陸，而當在 1.5feet 高時,高速度逐漸調降從 hover mode 降到 ground mode 的速度。

Track hover(through) mode : 則和 Waypoint hover(through) mode 類似。現在是要求到兩 Waypoint 間的直線，也就是在前往 Waypoint 的過程中，若追隨漂離設定直線，則會要求修正此一誤差。

Pilot assist mode: 則是由此地面控制站直接下達位置/高度/方向的命令。

配平的意義在數學上即是求得直昇機動態方程式的平衡點，使得在機體座標系統下，物理量的微分為零。要注意的是機體座標系統下，物理量的微分為零，並不代表在慣性座標系統下，物理量的微分為零。因此在配平點上直昇機的狀態微分如加速度、角加速度雖然為零，事實上若以慣性座標軸來觀察，直昇機是可以有加速度存在的，也就是直昇機仍然可以做轉彎的動作。旋轉模式 ，要指定某一配平狀況，只要設定 4 個配平參數：飛行速度 V_{fe}、飛行路徑角 Υ_{fe}、轉彎率 $\Omega_{ac}=\Psi$、側滑角 β_e。藉由這些參數的指定，可以描述直昇機不同的飛行狀況，如水平直飛，可以指定不同的飛行速度，其它參數則設為零。既然要使得非線性模式中的狀態微分為零，配平演算法的推導必須和非線性模式中所使用的公式相同。配平的流程可分為縱向配平、橫向/偏航配平及主旋翼轉速配平。配平前除了指定 4 個配平變數，設定某一配平狀況之外，對於在求解的過程中，數值未定卻先用到的變數，必須先給定初始猜測值，這些猜測值在配平的過程中將不斷疊代，並與原先的初始猜測值比較，使其誤差在容許範圍內。這些初始猜測值包括：主旋翼整體擺動角度 β_0、縱向擺動角度 β_{1C}，橫向擺動角度 β_{1S}，尤拉俯仰角 θ、滾轉角 Φ，尾旋翼整體控制角度 θ_{0T}，主旋翼轉速 Ω。整個疊代的過程在[2-12]有流程圖表示。線性化的方法是利用微擾理論，將非線性模式，針對配平點做泰勒展開，並保留至前面線性項，整理求得直昇機的線性模式。一個基本的假設是氣動力和氣動力矩是狀態、狀態微分及控制角度的解析函數。

目前中科院有成大航太系已執行直昇機動態模型建立研究，本研究團隊與蘇順豐老師的合作下將利用其程式建立一直昇機動態模擬之程式。我們參考一架現有成大航太系的直昇機動態模型的氣動力資料，以數值模擬的方式，利用前述方法建立其數學模式，隨後以水平直飛為例，改變速度將其劃分為不同的飛行狀況，求得各飛行狀況的配平點。並在各個配平點上求直升機的氣動力偏導數及線性化模式，然後針對線性化模式進行簡單制律設計。最後利用剛體運動方程式，進行受控直升機閉迴路非線性模式的動態模擬。在進行數值模擬的過程中，所需的直升機資料稱為組態資料(configuration data)。主要的依據是參考資料[2-19]Westland 公司出產的山貓型(Lynx)直升機詹氏年鑑[2-21]中的發動機資料。由於[2-19]中提供的資料不足數值模擬所需,資料的取得也不容易，因此採用其它估算的組態資料[2-20]，以作為數值模擬的基礎。整個非線性模式以 Matlab 程式加以實現(implement)。首先針對水平飛行(level fight)的飛行狀況，進行配平分析。飛行速度由 0m/s~70m/s，每隔 5m/s 分成不同的飛行狀況，再求得各飛行狀況下的配平點。其它的配平參數則設定為:飛行路徑角=0、轉彎率=0、側滑角=0。理論上，在配平點卜狀態微分值應為零，可是由於配平演算法的收斂程度及程式計算的誤差，計算出來的配平點與真實配平點之間仍然有些微差距存在。我們可以從狀態的微分值接近於 0 的程度，判斷出配平結果的好壞。往後所設計的迴授控制器，"其強健特性可以將這些線性模式的不確定性

克服。以停懸為例，針對此配平點線性化，可得系統矩陣的特徵值如下

-10.47 -2.12 -0.30 -0.24 -0.0745+0.41i 0.0543+0.44i

其中有一對共軛複數根之實部為正。顯示在此配平點，系統是不穩定的。之後利用配平的結果，計算配平點上的氣動力導數。吾人採用模式逼近法，先求得氣動力及氣動力矩隨狀態及控制角度的變化曲線，然後以三階的多項式逼近(最小平方誤差法)此曲線，對此多項式函數微分，以求得在配平點上的微分值。爾後給定任一款直升機的幾何組態資料，吾人所建立之動態模擬程式即可算出該款直升機的每一個氣動力偏導數。我們針對線性模式設計簡單的狀態迴授控制器

$$u=-\boldsymbol{K}\boldsymbol{x}+r,$$

其中 r 為參考命令利用極點指定的方法，設計閉迴路系統特徵值為

-10.47 -2.12 -2 -3 -2±2i -3±3j

其中迴授增益矩陣為

$$K=\begin{bmatrix} 0.0019 & 0.0034 & -0.0306 & 0.0029 & -0.0005 & 0.0015 & 0.0185 & 0.0030 \\ -0.2847 & -0.0509 & -0.0181 & -0.0400 & 0.3182 & 0.0122 & -0.2263 & 1.6203 \\ 0.0388 & -0.0185 & -0.0006 & 0.0213 & -0.0237 & -0.0123 & -0.1657 & -0.2409 \\ 0.0757 & 0.0965 & -0.0287 & 0.0960 & -0.1110 & -0.1011 & 0.5543 & -0.4042 \end{bmatrix}$$

不過系統的穩態值卻稍微偏離了原來的配平點，而達到了新的平衡位置。這是由於在配平分析求得配平點時，仍然有殘餘的力和力矩存在，這些殘餘的力量在線性模式分析時是沒有考慮到的，然而在帶入非線性模式時，這些力量便出現了。這些殘餘力量是狀態的非線性函數，雖然控制的加入可以克服這些殘餘的力量(若指定的極點愈遠離虛軸，控制的力量愈大，則偏離愈小)，可是其穩態值也偏離了，事實上這些殘餘的力量也可以視為一種隱性的參考輸入。

關於線性模式階數的決定，若線性模式的階數只有 6 階(狀態為 u、v、w、p、q、r)，根據此線性模式所設計的狀態迴授控制律，由於沒有過授姿態角的結果，會使得姿態角發散。重力在機體座標系統三個軸向的分量也會改變，使得飛行狀況漸漸遠離線性模式適用的範圍。這個問題在加入狀態θ 、ϕ迴授之後，即可解決。偏航角度Ψ並不會影響重力的分量，對於系統的穩定也就沒有影響，不過沒有過授偏航角度的結果，會造成偏航角速度，為一不等於零的值。基本上，線性模式的階數為 8 階即可以使用。將來若進一步迴授位置，則位置可以追隨指定的軌跡，而達到導航的功能。

文獻中[2-19]有類似的推導，可是其中的氣動力參數也不易取得，不過在本計劃的研究中，成大航太所楊憲東教授也將嘗試求取可能之相關參數值以供模擬驗證用。一般在進行一架飛行器飛行模擬或製作其控制器時，所需要的是這架飛行器在各狀態及控制角度下的氣動力參數，也就是分析時的數學模型，氣動力參 數的取得可來自兩方面 : 一為風洞資料，另一方面為理論推導求得。固定翼飛機方面，目前已經有很多公開的風洞資料或是理論推導的軟體，但是

直昇機氣動力參數的取得卻不容易，直昇機的風洞實驗並不像定翼機般普遍，風洞資料更是缺乏。因此想到 Neural – Fuzzy 具有 model – free 及 nonlinear 的特性，可利用實體直昇機在飛行中的六個自由度的資料及相關可得的參數來將其 model 學出來，在依照此 model 設計其控制器來控制整架直昇機飛行動作。

系統模型，但其非線性數學模型複雜在直昇機模式建立方面，目前大部的研究都是以氣動力分析為主並利用文獻中的動態方程式來架構，可是如是的建模常需要在氣動力係數上尋求，其常常是耗時費力，而很難取得時而且這些係數是隨環境而會改變的。整體飛行器數學模型中包含了該飛行器動態的表現，如引擎、起落架、飛行控制面的狀態以及外在環境所造成的影響等，為了研究其非線性系統的整體行為就必須建立非線性數學模型。系統識別一般可稱為逆向工程，利用系統輸入與系統響應來推導的程度即使在高速電腦迅速發展的今日應用傳統的數值方法亦相當費時。以一般數值解法作參數識別與狀態估測來找出飛機的最佳動態模式流程。在建立其動態模式運算過程由於數據量的龐大及數值解法中積分與偏微分運算使並且整體流程需採疊代方式修正誤差，使得電腦處理能力非常緩慢。因此雖然採用適當的數值解法可求得準確的結果，但其耗費的時間相當龐大。故必須研究特殊有效且適合計算機程式地方法如代數及多項式逼近來尋求近似解等等，來解決需要較大的電腦記憶空間及耗費長時間計算的問題，因此我們應用生物控制論中的演化方法以多項式來近似地表示非線性系統的輸入輸出關係。藉由我們設計飛行模擬器時所使用的風洞測試數據的特性建立常態多項式來近似模擬，針對原縱向飛行系統C_x、C_z、C_m橫向飛行系統C_y、C_l、C_n及最後之最佳多項式結果與風洞測試作一比較，其結果誤差程度可達 5%的接受程度，證明此一方法的可靠度。

飛機於空中飛行與大氣環境交互作用產生浮力、阻力等等。在大氣環境中不同的高度擁有不同的空氣密度、靜壓力及動壓，同時也影響飛機推力輸出。本子題引用美國太空總署 NASA 所提供的 1976 年標準大氣環境模型作為飛行動態系統的大氣模擬環境，詳細資料如表 2-1-1 所示。

表 2-1-1 — 1976 Standard Atmosphere Characteristics

Altitude (Feet)	Density (Ratio)	Pressure (Ratio)	Speed Sound (KTAS)	Temp (F)	Airspeed (TAS/IAS)	**Mach 0.8** (KIAS)
0	1	1	59	622	1	530
2500	0.9289	0.9129	50	656	1.0376	506
5000	0.8617	0.8321	41	650	1.0773	483
7500	0.7983	0.7527	32	645	1.1192	461
10000	0.7386	0.6878	23	639	1.1636	439
15000	0.6295	0.5646	5	627	1.2604	398
20000	0.5332	0.4599	-12	615	1.3695	359
25000	0.4486	0.3716	-30	602	1.4930	323
30000	0.3747	0.2975	-48	590	1.6336	289
35000	0.3106	0.2360	-66	577	1.7943	257
40000	0.2471	0.1858	-67	574	2.0117	228
45000	0.1945	0.1462	-67	574	2.2675	202
50000	0.1535	0.1151	-67	574	2.5557	180
55000	0.1206	0.0906	-67	574	2.8796	159
60000	0.0914	0.0687	-67	574	3.3084	139
65000	0.0747	0.0562	-67	574	3.6576	126

根據上述的參考資料，本團隊與蘇順豐老師的合作下沿用之前所設計的直升機飛行控制器，希望藉由控制器參數的調整使直升機能達到基本的飛行動作，進而說明直升機飛行控制器參數該如何應對自動導航系統所下達的飛行命令。以下將對直升機的操作、基本的直升機飛行動作、飛行自動導航系統與直升機飛行控制器、直升機飛行模擬作詳細說明。

(1). 直升機的操作

首先以駕駛員的角度來簡單說明直升機的操作，直升機駕駛員座艙主要的操縱機構是：循環槳距操作桿、尾旋翼槳距踏板、整體槳距與油門操作桿等。循環槳距操作桿位於駕駛員座椅前面，與主旋翼的自動傾斜器連接，以循環槳距操作桿偏離中立位置表示：向前—直升機低頭並向前運動；向後—直升機擡頭並向後退；向左—直升機向左傾斜並向左側運動；向右—直升機向右傾斜並向右側運動。尾旋翼槳距踏板位於座椅前下部，對於單尾旋翼的直升機來說，駕駛員以尾旋翼槳距踏板操縱尾旋翼整體槳矩改變尾旋翼推(拉)力。整體槳距與油門操作桿操縱主旋翼整體槳距和發動機油門，實現主旋翼總距和油門聯合操縱。

由機械與物理的角度來簡單說明直升機的操作，直升機是藉由改變主旋翼整體槳矩控制角度θ_O、主旋翼縱向循環槳矩控制角度θ_{ls}、主旋翼橫向循環槳矩控制角度θ_{lc}、尾旋翼整體槳矩控制角度θ_{OT}，來改變主旋翼的升力、擺動方向及尾旋

翼的推力，來達成機身運動方向的控制。直升機機身的姿態，可視為施於機身上的各力量平衡後得到的結果。在不考慮發動機的情況下，共有九個狀態u、v、w、p、q、r、φ、θ、ϕ將被考慮，其中u、v、w與p、q、r分別為機體座標系統下的三個軸向速度及角速度如圖2-1-27。而φ、θ、ϕ為機體姿態角，當機體座標系統下的三個軸向速度及角速度達到平衡(力矩平衡)時，機體姿態角也就固定，所以只有討論u、v、w與p、q、r的控制，其中角速度p、q是間接受所控制，因此直升機實際上需要控制的狀態為u、v、w、r，接下來由力矩平衡的角度說明直升機飛行動作是如何完成的。

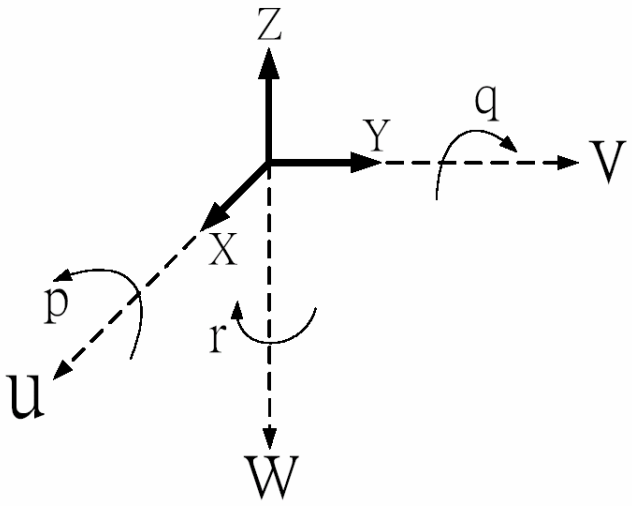


圖2-1-27 機體座標系統與三個軸向的速度及角速度示意圖

(2). 基本的直升機飛行動作

本節將以施加於直升機上的各力量平衡之情形，來說明直升機基本飛行動作如何達成。注意文中使用的符號其正負號(表示方向)、文字等皆配合圖 2-1-27 所設定。

懸停

懸停是直升機在一定高度上保持航向和對地標位置不變的飛行狀態。懸停時，單旋翼式直升機力矩的平衡如圖 2-1-28 所示。主旋翼拉力在鉛垂面的升力分量與飛行重力平衡；尾旋翼推力 Fw－Fg Fr 平衡主旋翼反扭力矩 Fr－ 。即鉛垂方向： －Fw＝Fg，水平側向：Fr＝－Fr 。

垂直上升

直升機垂直上升飛行速度以 V_{up} 表示，通常直升機的垂直上升速度都不大，機體阻力與飛行重力 Fg 比較起來則爲一個小量，可以忽略不計，因此直升機垂直上升時力矩的組成與懸停時基本上相同。即鉛垂方向： －Fw＞Fg，水平側向：Fr＝－Fr 。

垂直下降

垂直上升相反，利用它可以使直升機在狹小場地著陸。由於這時主旋翼的旋轉方向與其運動的相對氣流方向相反，氣流流經主旋翼的兩側時，由於方向相反的氣流使主旋翼流場變得更加複雜。隨著下降速度的增加，當兩股氣流的速度數值十分接近時，會有亂流產生，這物理現象在這裡並不討論之。因此，垂直下降與懸停及垂直上升時力矩的平衡基本上是一樣，即鉛垂方向： $-Fw < Fg$，水平側向：$Fr = -Fr$ 。

側飛

側飛是直升機特有的飛行狀態，一般側飛是在懸停基礎上實施的飛行狀態，其特點是要多注意側向力的變化和平衡。由於直升機機體的側向投影面積很大，造成機體在側飛時其空氣動力阻力特別大，因此直升機側飛速度通常很小。由於單旋翼帶尾槳直升機的側向受力是不對稱的，因此左側飛和右側飛受力各不相同，所以左右側飛行特性也不會相同，如圖 2-1-29。保持直升機等速側飛的力矩平衡條件爲：即鉛垂方向： $-Fw = Fg$，水平側向：$Fv > -Fx$ 。

平飛

相對於圖 2-1-27 中速度軸平飛時，作用在直升機上的力主要有懸空拉力 F(u, −w)，飛行重力 Fg，機體阻力 Fx 及尾旋翼推力 Fr 尾。若採取前飛時的速度軸系選取原則： X 軸指向飛行速度 u 方向； Z 軸垂直於 X 軸向上爲正，Y 軸按右手法則確定，如圖 2-1-30。保持直升機等速直線平飛的力矩平衡條件爲：

X 軸： $Fx > Fu$。

Z 軸： $-Fw = Fg$。

Y 軸： $Fr = -Fr$。

起飛

直升機利用主旋翼拉力離開地面、並增速上升至一定高度的運動過程叫做起飛。正常的起飛動作有三種，其一為“直接垂直起飛”，這種起飛方式是在場地周圍有一定高度與場地大小的限制時採用，保持直升機直接垂直起飛的力矩平衡條件爲：鉛垂方向：$-Fw > Fg$，水平側面：$Fr = -Fr$。

其二為“標準起飛”，是當場地淨空條件較好，直升機先垂直離地約 0.15—0.25 個旋翼直徑的高度，進行短暫懸停，檢查一下發動機情況，然後以較小的爬升角進行增速爬升，到達一定飛行高度的過程。在這個過程中直升機旋翼的需用功率變化很大，即直升機的受力狀態變化很大，對操縱動作的協調性要求很高，因此保持直升機標準起飛的力矩平衡條件在這先不討論。

其三為“滑行起飛”，當直升機的載重量過大或者起飛場地標高及其他氣象條件使直升機無法以其他起飛方式起飛時採用，它像固定翼飛機那樣採用滑跑方式起飛。直升機的滑跑起飛，省去了垂直離地和近地面懸停這兩個階段，而分成地面滑行增速和空中增速兩個階段進行。直升機增速至一定值後，由於主旋翼需

用功率的減小，就有足夠的功率來增加主旋翼的拉力，克服重力升空。隨著飛行速度進一步增加，旋翼需用功率進一步下降，這時直升機就有部分剩餘功率用來爬升和增速，完成整個起飛過程。由於整體操縱動作含有大量外部影響因素，因此這起飛狀況的力矩平衡條件在這並不討論。

著陸

直升機從一定高度下降，減速、降落到地面直到運動停止的過程稱爲著陸，是起飛的逆過程。一般可分為四種著陸方式，其一為”超越障礙物垂直著陸”， 當著陸場地面積狹小，周圍又有一定高度的障礙物，直升機在接近著陸場地的空間不允許做慢速貼地飛行，此時就採用超越障礙物垂直著陸方式著陸。它與正常垂直著陸不同的是作減速和接地前短暫懸停高度不同，由於懸停時不能利用場地淨空條件，所以這種方式消耗功率較大。同時著陸點附近又有障礙物，直升機縱橫向不允許較大的位移，操縱難度大一些。使直升機進行超越障礙物垂直著陸的力矩平衡條件爲：鉛垂方向：$-Fw < Fg$，水平側面：$Fr = -Fr$，直到直升機著陸。

其二為”正常垂直著陸”， 對於預定著陸地點場地淨空條件良好，以這種方式著陸的做法是：以一定的下滑角大致向預定點下降，並逐漸減速，在接近著陸預定點前，直升機作小速度貼地飛行，由於充分利用了場地淨空條件，所以消耗的功率減小。或者在到達預定點的上空 3 − 4m 高度上做短時間懸停，再以 0.2 − 0.1m/s 的下降率垂直下降直至接觸地面，這種著陸方式對著陸場地表面質量要求少，場地面積相對來說比較小。由於整體操縱動作含有大量外部影響因素，所以直升機進行正常垂直著陸的力矩平衡條件在此並不討論。

其三為”滑行著陸”， 直升機在高原、高溫地區，或載重量較大時，可用功率不足以允許用垂直著陸方式著陸，可以像固定翼飛機一樣進行滑行著陸。滑行著陸與垂直著陸不問，直升機在接觸地面瞬間，不但具有垂直速度，同時還有水平速度。直升機在接觸地面後有一個滑行過程，可進一步利用主旋翼產生一個減速的水平分力，使直升機持續減速直到運動停止。由於整體操縱動作含有大量外部影響因素，因此直升機進行滑行著陸的力矩平衡條件在此並不討論。

其四為”旋翼自轉狀態的下滑著陸”，當如發動機功率不足時(故障)，將發動機關閉，這時只利用主旋翼的自轉做減速下降，此時直升機猶如竹蜻蜓一般。在這種降落狀態，完全依靠直升機下降時重力位能作功提供給主旋翼來產生拉力來抵消大部份的重力，直升機在接觸地面後機身與人員必須承受剩餘的重力衝擊。這是使用在緊急狀況下的著陸動作，所以使直升機進行旋翼自轉狀態的下滑著陸的力矩平衡條件在此並不討論。

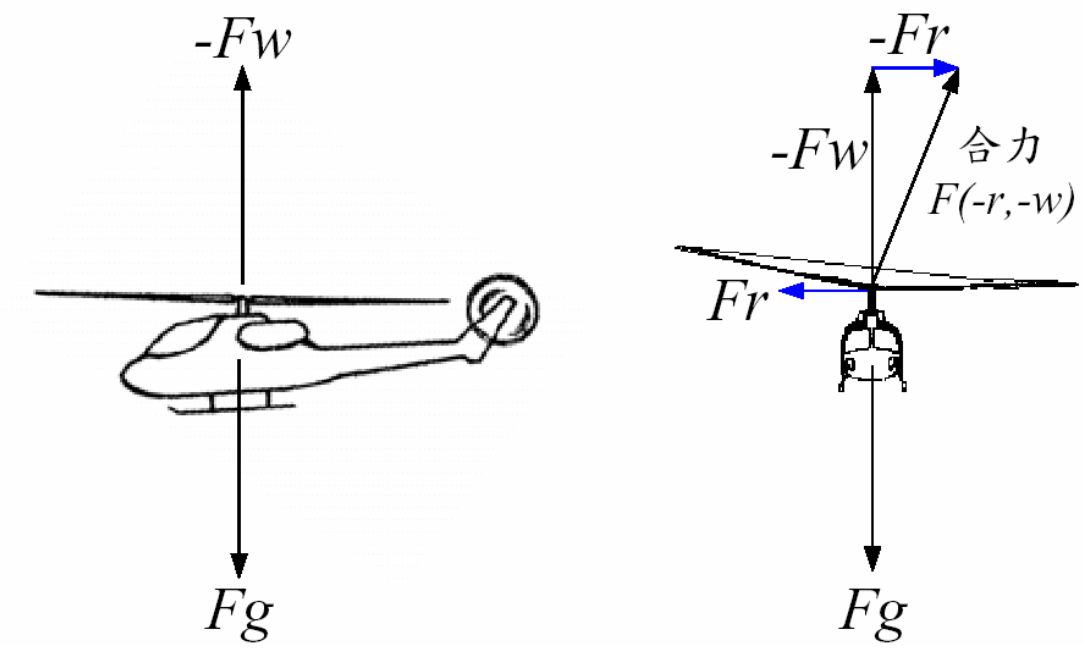


圖 2-1-28 直升機懸停時，機身上各力量與平衡關係

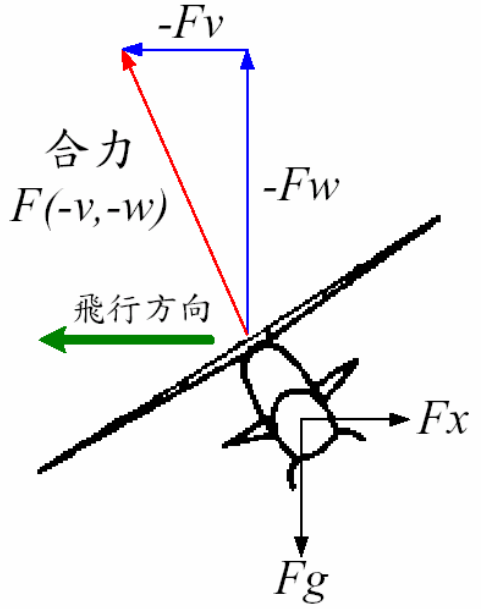


圖 2-1-29 直升機側飛的各力量與平衡關係

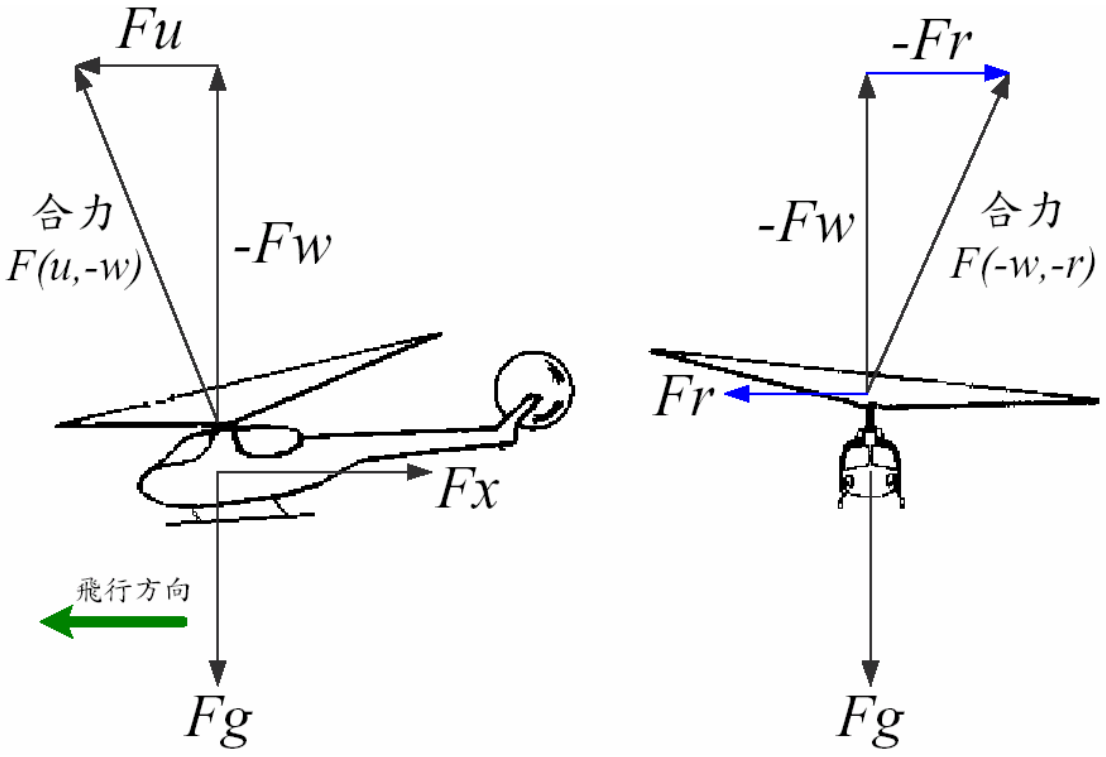


圖 2-1-30 直升機平飛時，機身上各力量與平衡關係

(3). 飛行自動導航系統與直升機飛行控制器

飛行自動導航系統對本控制器而言，猶如飛行動作的命令者一般，自動導航系統藉由其他的訊息判斷出目前實際飛行狀態與期望飛行狀態的誤差，依這誤差值對直升機的飛行狀態進行修正，這誤差與修正的問題隱藏著許多須要研究克服的問題，所以在本文中將不被說明。接下來，本文中將直接探討如何將飛行自動導航系統所下達飛行動作指示，轉換成直升機飛行控制器參數的設定。由於本研究使用的直升機模型尚未考慮空氣阻力、天氣、控制器響應、直升機慣性等因素，所以在這裡便不深入探討飛行速度的精確求法。令 u_m、v_m、w_m 及 r_m 為三個軸向速度 u、v、w 與一個角速度 r 的目標值。

[i] 飛行方向與速度的估算

情況一：直升機機鼻方向固定(直升機自座標不變，令 $r_m = 0$)

本節的內容僅從向量的觀點來說明飛行方向與速度的估算方式，估算的結果會與實際結果會有一定的誤差。由上一節的內容可知道，直升機的各種基本飛行動作，是藉由施於直升機上的各項力矩的組合而達成，而各項力矩(Fu,Fv,Fw)的大小是正比於三個軸向速度(u,v,w)，所以可藉由調整這三個軸向速度(u,v,w)，進而調整 Fu,Fv,Fw 的大小，來改變得到的力矩平衡關係，理想狀態下合力矩可由向量公式估算。圖 2-1-28~30 所式示的二維空間(平面)中合力大小可由畢式定理

$$F_{total} = k_F \sqrt{F_1^2 + F_2^2}$$

估算出近似值，其中 F_1、F_2 為兩個不同的力，F_{total} 為得到的合力， 為一正負值表示合力方向，由力矩的座標系決定正負值，而合力方向即為飛行方向，該方向的

飛行速度也是可依畢式定理

$V_{total} = k_V \sqrt{V_1^2 + V_2^2}$

估算出近似值，V_1、V_2 為兩個不同的速度，V_{total} 為得到的合成速度，k_V 為一正負值表示合成速度方向，由速度的座標系決定正負值。

參考圖 2-1-31 在三維空間中的飛行速度與角度的近似求法，其中 u_1i、v_2j、w_3k 表示在座標 i、j、k 上的速度，將向量 a 的大小 $\|a\|$ 視為合成速度，即可求出圖 2-1-31 中的合成速度與相對於各軸的角度 α、β、γ， 圖 2-1-31 中向量 a=u_1i+v_2j+w_3k，求出的結果整理如下：

$\|a\| = \sqrt{u_1^2 + v_2^2 + w_3^2}$

$\alpha = \cos^{-1}\left(\frac{u_1}{\|a\|}\right)$

$\beta = \cos^{-1}\left(\frac{v_2}{\|a\|}\right)$

$\gamma = \cos^{-1}\left(\frac{w_3}{\|a\|}\right)$

請注意，這三維空間中的飛行方向與速度估算方式僅供參考，因為直升機完成三維空間的飛行動作是需要實際飛行經驗才能了解的，本文不以旁觀者的角度來解釋飛行經驗。

例一：理想狀況下，希望直升機水平向前且右偏 45 度以時速飛行，可經由設定 $u_m = \frac{20(1000)}{\sqrt{2}(3600)} = 3.928m/\sec$ 、 $v_m = -3.928m/\sec$ 、 $w_m = 0$ 、 $r_m = 0$ 所滿足，根據 $V_{total} = \sqrt{3.928^2 + (-3.928)^2} = 5.5m/\sec = 19.9km/h$ ，若改為左偏 45 度，則只須令 $v_m = 3.928m/\sec$ 即可，本例題的正負值請參考圖 2-1-27 的座標系，與圖 2-1-32 的飛行軌跡示意圖。

例二： 令圖 2-1-31 中的 $a = 2i + 5j + 4k$ 則合成速度 $\|a\| = \sqrt{2^2 + 5^2 + 4^2} = 3\sqrt{5}$ ，

$\alpha = \cos^{-1}\left(\frac{2}{3\sqrt{5}}\right) = 1.27(rad) or 72.27°$ ， $\beta = \cos^{-1}\left(\frac{5}{3\sqrt{5}}\right) = 0.73(rad) or 41.8°$ ，

$\gamma = \cos^{-1}\left(\frac{4}{3\sqrt{5}}\right) = 0.93(rad) or 53.4°$ ，本例題未參考圖 2-1-27 的座標系統。

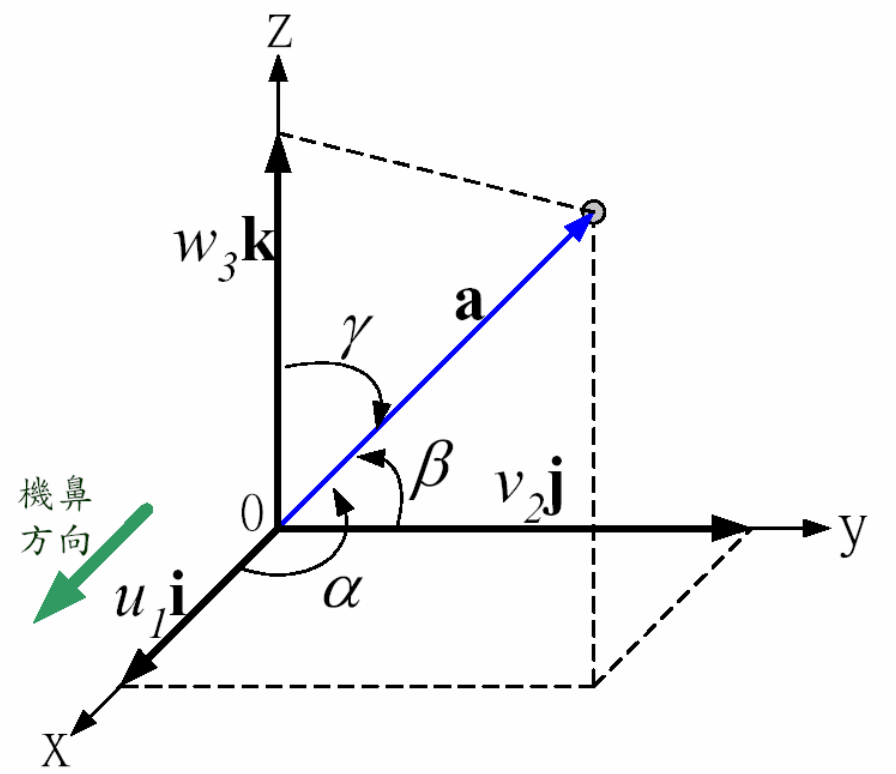


圖 2-1-31 三維飛行空間示意圖，其中原點為直升機機身重心

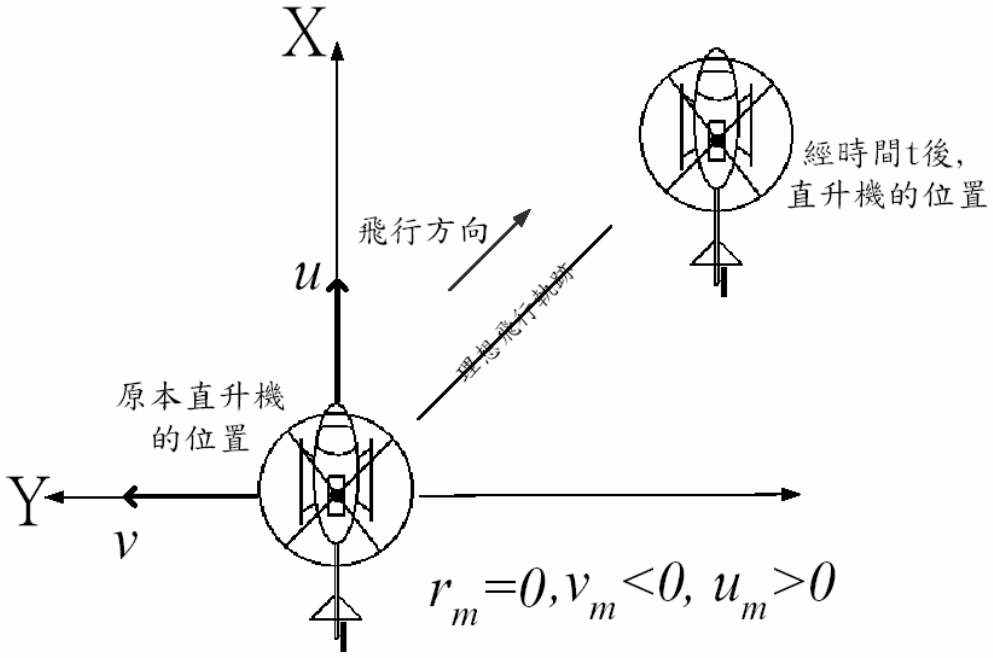


圖 2-1-32 直升機機鼻方向固定的二維飛行空間示意圖，控制器參數可參考例一

情況二： 直升機機鼻方向隨飛行方向改變(直升機自座標改變，令 $r_m \neq 0$)

在這種情況下，配合控制尾旋翼槳距控制角度 θ_{OT} 來改變尾旋翼的推力，進而改變機身擺動方向，達成機身運動方向的控制。令 $r_m > 0$ 則機鼻向右擺動，令 $r_m < 0$ 則機鼻向左擺動，而 $r_m = 0$ 則擺動停止，擺動的角度可由 r 與時間的變化估算出。加入控制尾旋翼槳距的飛行方式則可在不考慮速度 v 的情況下，改變飛行方向，但也改變了直升機自座標，若沒有配合其他的定位系統，則直升機會迷失方

位。之前提到與改變飛行方向的飛行動作，皆可以在同時改變直升機機鼻方向的情況下完成。但這些飛行動作的難度也增加，在沒有飛行經驗的前提下，所以本文便不深入說明。

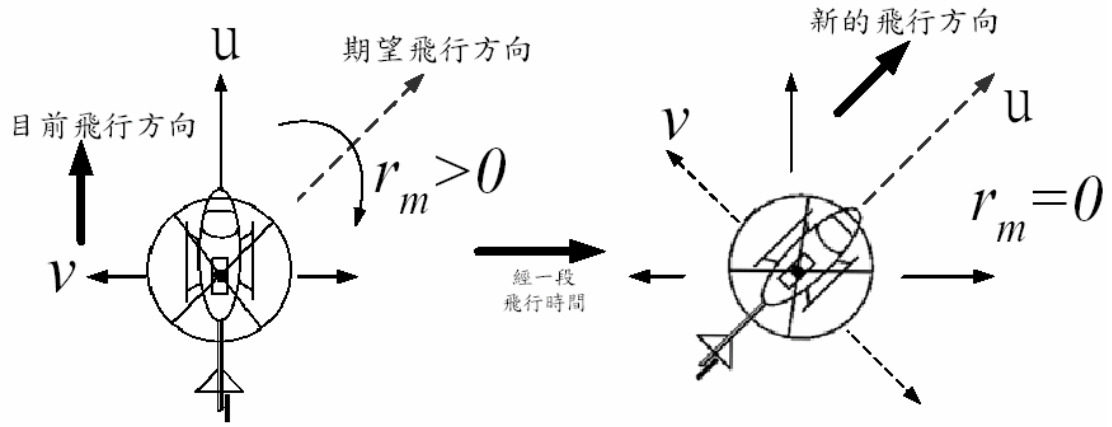


圖 2-1-33 直升機機鼻方向隨飛行方向改變的二維飛行空間示意圖

[ii] 自動導航系統的飛行動作指示與直升機飛行控制器參數的轉換

現在考慮之前設計的控制器，控制器允許使用者依要求設定三個軸向速度 u、v、w 與一個角速度 r ，令 u_m、v_m、w_m 及 r_m 為各速度的目標值，而使用者便可依 u_m、v_m、w_m、r_m 的設定來決定直升機的飛行方向與速度，本文將基本飛行動作的 u_m、v_m、w_m、r_m 設定值整理於表 2-1-2、3 中，表中只將各速度的方向列出，而各速度的值取決於自動導航系統對飛行動作的要求。飛行自動導航系統要能依各種飛行狀況，對控制器下達合理的命令(如表 2-1-2、3 所示)以滿足飛行要求，如何預測、判斷飛行狀況與下達何種命令，這將是一大挑戰。

另外若是在虛擬的飛行空間(動畫)中，只要利用直升機的三個軸向速度和三個軸向角速度 u、v、w、p、q、r 並配合對應的時間，即可求出飛行距離、機身姿態、飛行高度、機鼻方向等參數，利用這些參數便可以進行航道確認和修正、動畫展示等工作，但在實際的情況下則需要更為客觀嚴謹的監督如全球定位系統、雷達等。

表 2-1-2 直升機機鼻方向固定時，基本飛行動作與控制參數設定對照表

	垂直上升	垂直下降	向前爬升	向前下降	向前平飛	倒退平飛	向右側平飛
u_m	等於零	等於零	大於零	大於零	大於零	小於零	等於零
v_m	等於零	等於零	等於零	等於零	等於零	等於零	小於零
w_m	小於零	大於零	小於零	大於零	等於零	等於零	等於零
r_m	等於零	等於零	等於零	等於零	等於零	等於零	等於零

續表 2-1-2

	向左側平飛	偏右向前平飛	偏左向前平飛	懸停直接垂直	起飛	超越障礙物垂直著陸
u_m	等於零	大於零	大於零	等於零	等於零	等於零
v_m	大於零	小於零	大於零	等於零	等於零	等於零
w_m	等於零	等於零	等於零	等於零	小於零	大於零
r_m	等於零	等於零	等於零	等於零	等於零	等於零

表 2-1-3 直升機機鼻方向隨飛行方向改變的基本飛行動作與控制參數設定對照表

	懸停情況下機鼻向右擺動變化	懸停情況下機鼻向左擺動變化	平飛狀態下，機鼻向右擺動變化	平飛狀態下，機鼻向左擺動變化
u_m	等於零	等於零	大於零	大於零
v_m	等於零	等於零	小於或等於零	大於或等於零
w_m	等於零	等於零	等於零	等於零
r_m	大於零	小於零	大於零	小於零

註：r_m 值不為零時，尾旋翼會提供一個推力使機身向左或向右迴轉，迴轉的角度可由實際的角速度 r 乘上時間即可求出。

(4). 直升機飛行模擬

本節共有三個模擬結果，模擬一為直升機機鼻方向固定情況下的基本飛行動作測試。模擬二為直升機機鼻方向隨飛行方向改變情況下，直升機進行定點旋轉的機身穩定模擬。模擬三為參考例一的參數設計進行飛行方向、速度模擬。

模擬一：直升機機鼻方向固定時，共計 90 秒的基本飛行動作測試。

本模擬的目的是驗證表 2-1-2 的控制參數設定是否能滿足飛行動作的要求，並且提供將來自動導航系統的飛行動作指示與直升機飛行控制器參數的轉換範例。

表 2-1-4 直升機機鼻方向固定情況下，模擬飛行動作與控制參數設定對照表

時 間 (sec)	動作編號	要求動作	u_m 設定值 (m/sec)	v_m 設定值 (m/sec)	w_m 設 定 值(m/sec)	r_m 設定值 (rad/sec)
0~5	1	直接垂直起飛	0	0	-6	0
5~10	2	向前平飛	20	0	0	0
10~15	3	懸停	0	0	0	0
15~25	4	向左側平飛	0	1	0	0
25~35	5	向右側平飛	0	-1	0	0
35~45	6	偏左向前平飛	5	1	0	0
45~55	7	偏右向前平飛	5	-1	0	0
55~60	8	向前下降	10	0	1	0
60~65	9	向前爬升	10	0	-1	0
65~75	10	倒退平飛	-10	0	0	0
75~90	11	垂直下降	0	0	1	0

模擬結果：請參考圖 2-1-34~42。

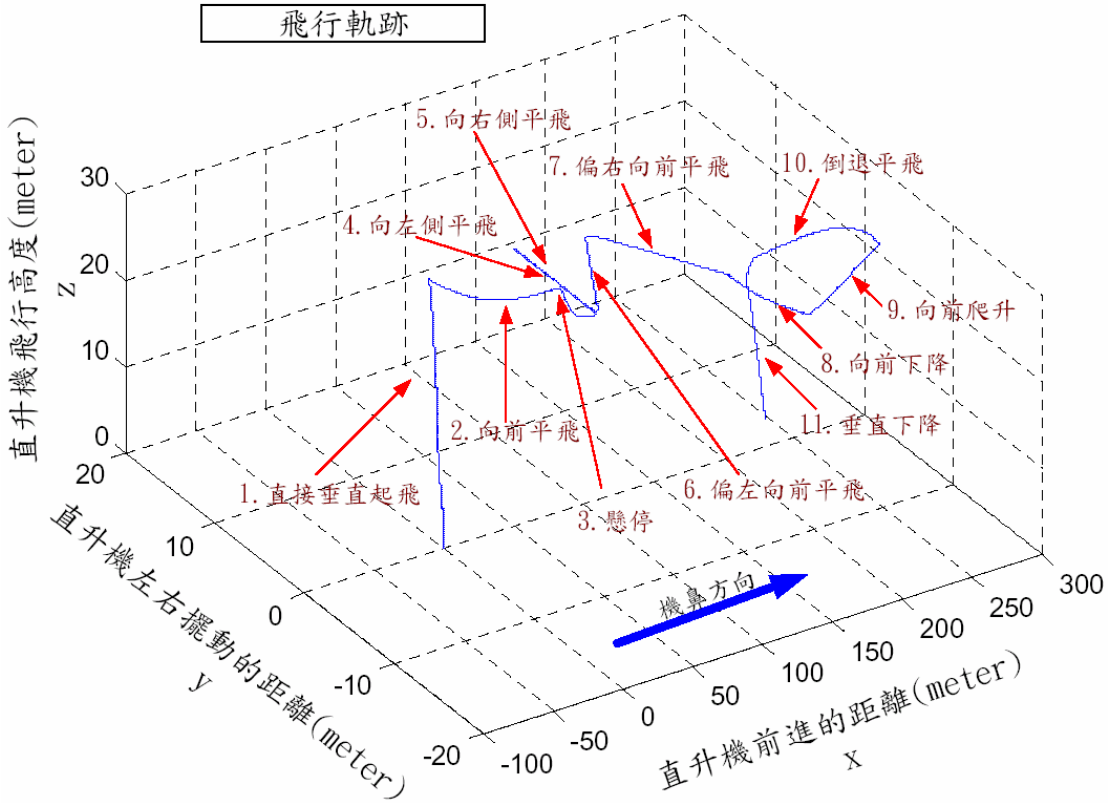


圖 2-1-34 直升機飛行軌跡，模擬飛行動作與控制參數設定請參考表 2-1-4，相關模擬數據參考圖 2-1-41~42

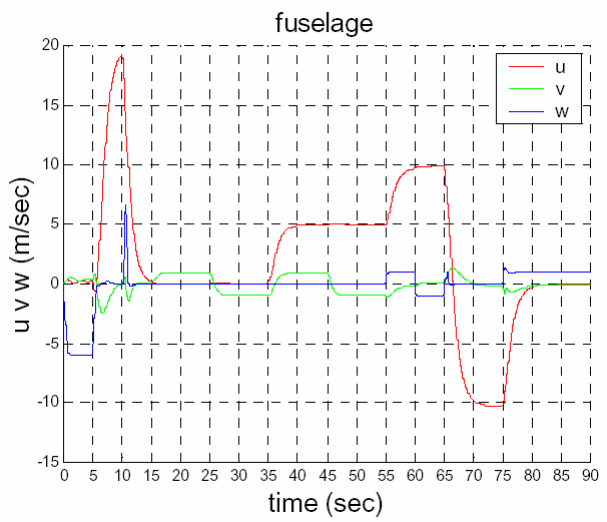


圖 2-1-35 直升機各速度的變化

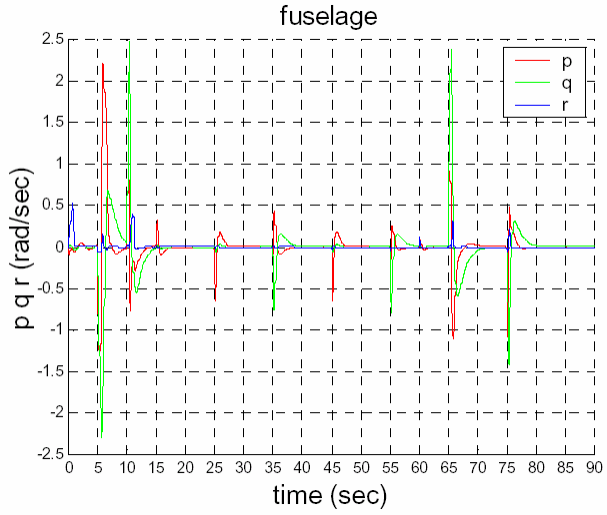


圖 2-1-36 直升機各角速度的變化

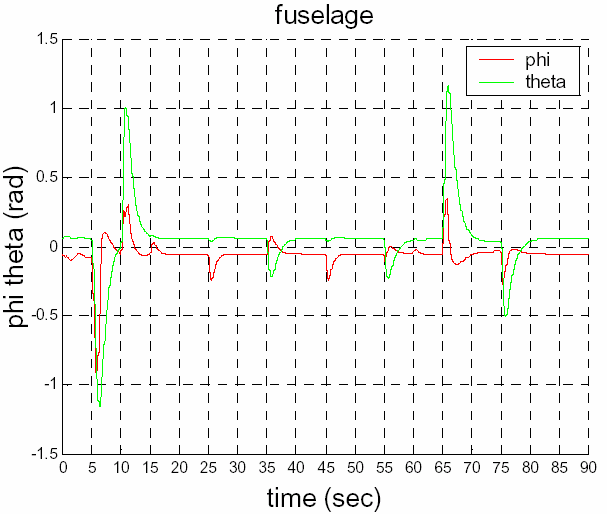


圖 2-1-37 直升機機身姿態角度變化

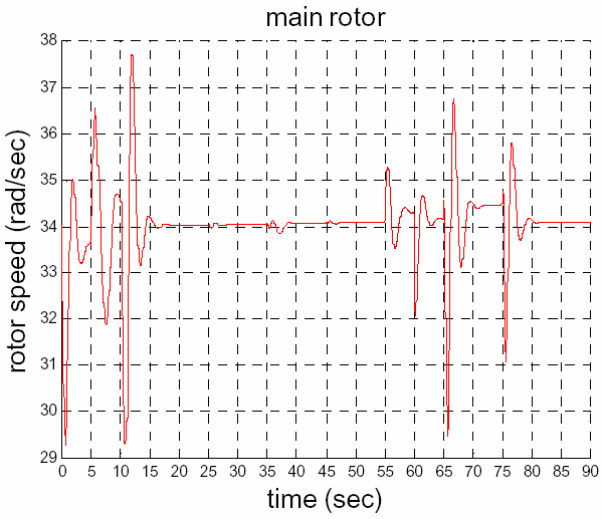


圖 2-1-38 直升機主旋翼轉速變化

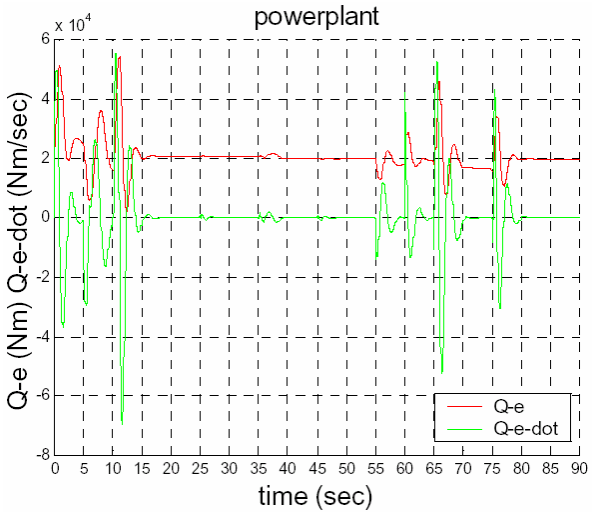


圖 2-1-39 直升機發動機輸出功率的變化

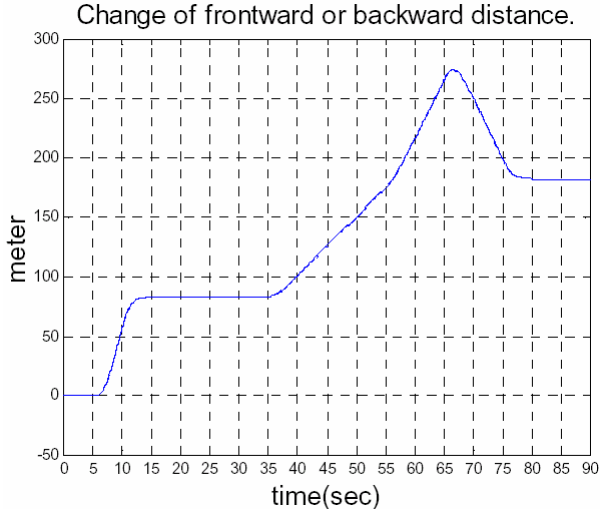


圖 2-1-40 直升機飛行距離的變化(x 座標方向)

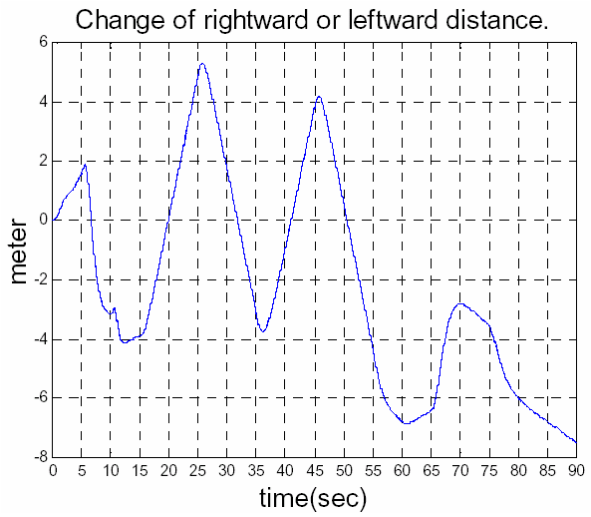


圖 2-1-41 直升機飛行距離的變化(y 座標方向)

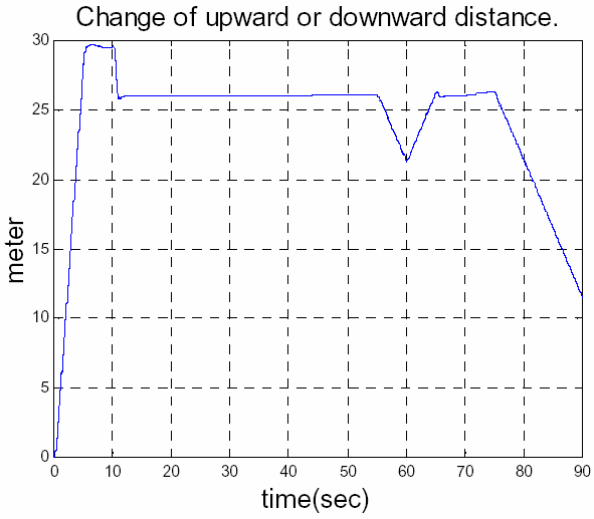


圖 2-1-42 直升機飛行距離的變化(z 座標方向)

模擬二：直升機機鼻方向隨飛行方向改變時，共計 35.94 秒的基本飛行動作測試。

模擬目的是希望能知道直升機的尾旋翼控制器性能是否能符合要求，但由於無法配合動畫，所以只能以機身各方向的距離變化來表現，如圖 2-1-49~52。設 r_m=±3.0 是令直升機機鼻能在約 10.49 秒完成向左或向右旋轉一周，由圖 2-1-52 的模擬結果可知有一點誤差存在，由圖 2-1-49~51 可觀察到直升機重心的偏移程度，以懸空高度約 30 公尺而言，這重心的偏移程度是可接受的。至於其他的合理飛行動作測試，因為難以現有的資源表現出直升機動態，所以便不進行模擬，不過由模擬一得到的結果可推測，其他相關的合理飛行動作應可以達成。

表 2-1-5 直升機機鼻方向隨飛行方向改變時，模擬飛行動作與控制參數設定對照表

時間(sec)	動作編號	要求動作	u_m 設定值 (m/sec)	v_m 設定值 (m/sec)	w_m 設定值 (m/sec)	r_m 設定值 (rad/sec)
0~5	1	直接垂直起飛	0	0	-6	0
5~15.47	2	機鼻向左轉 360 度	0	0	0	-0.3
15.47~25.94	3	機鼻向右轉 360 度	0	0	0	0.3
25.94~35.94	4	懸停	0	0	0	0

模擬結果：

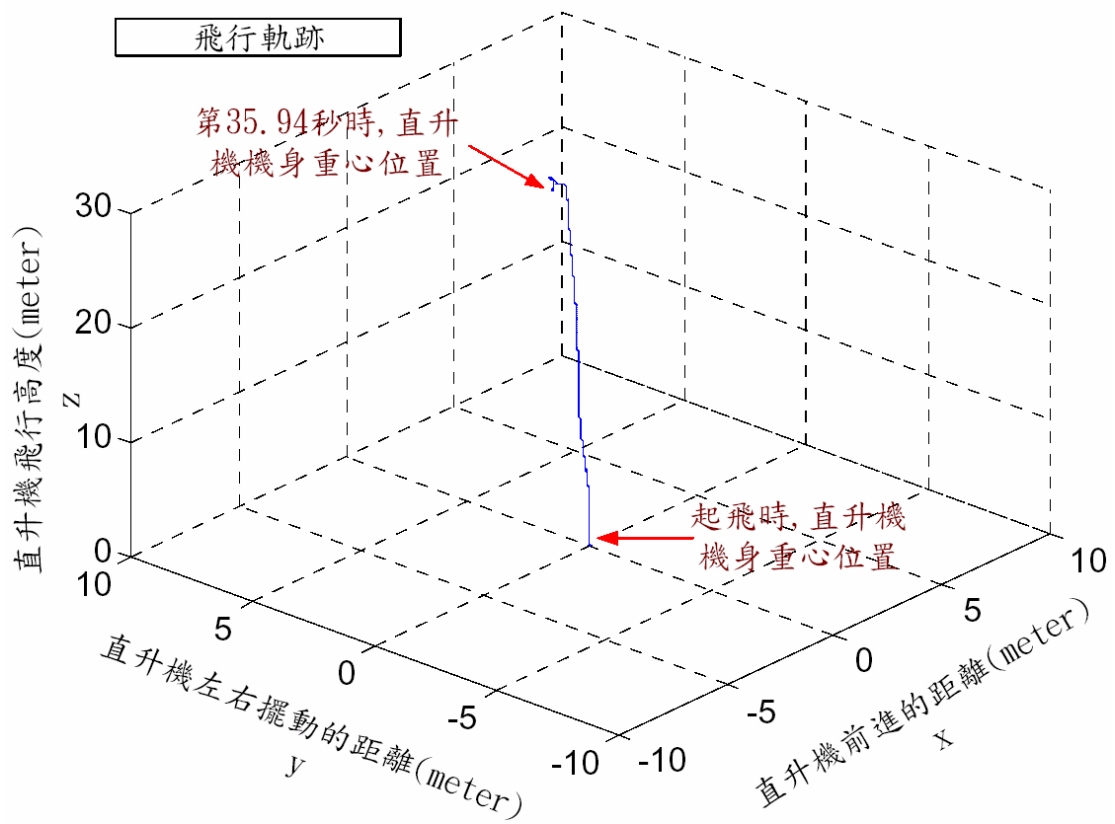


圖 2-1-43 直升機飛行軌跡，模擬飛行動作與控制參數設定請參考表 2-1-5，相關模擬數據參考圖 2-1-49~51

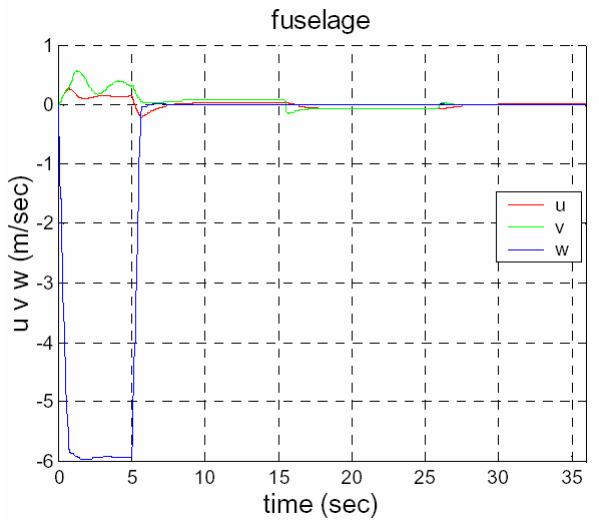


圖 2-1-44 直升機各速度的變化

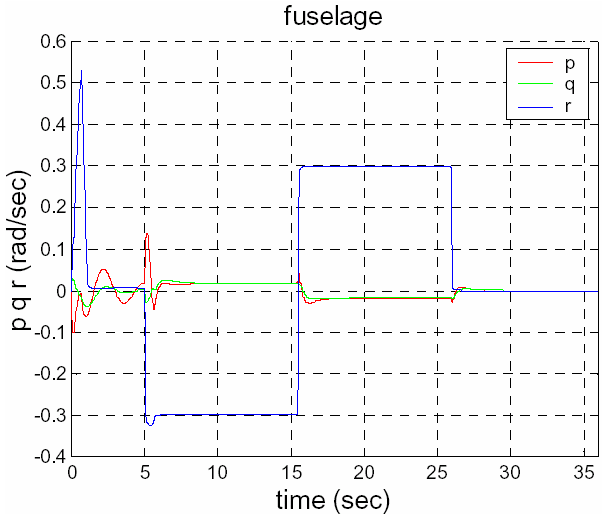


圖 2-1-45 直升機各角速度的變化

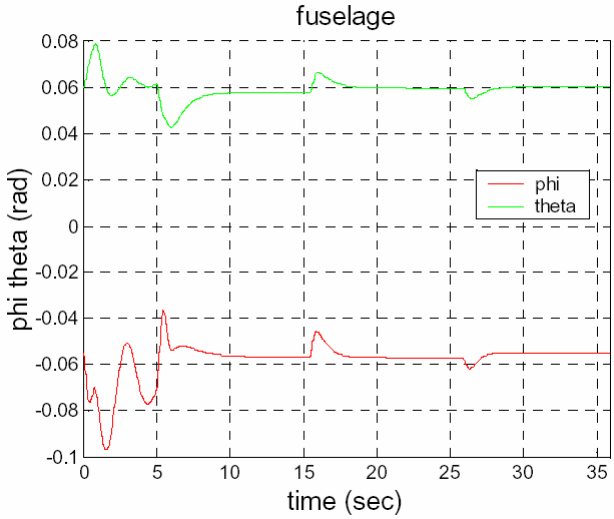


圖 2-1-46 直升機機身姿態角度變化

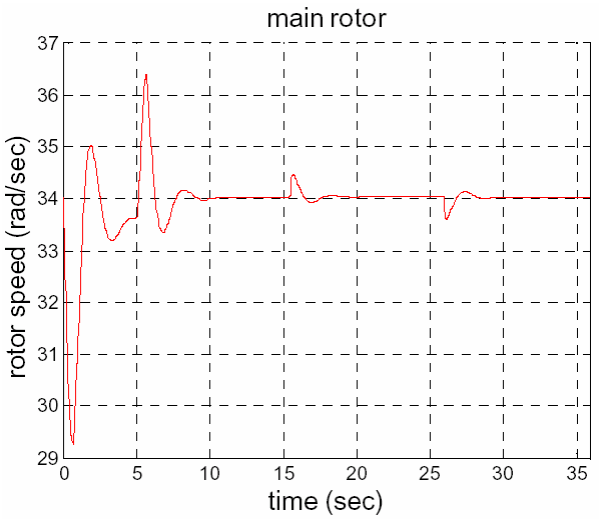


圖 2-1-47 直升機主旋翼轉速變化

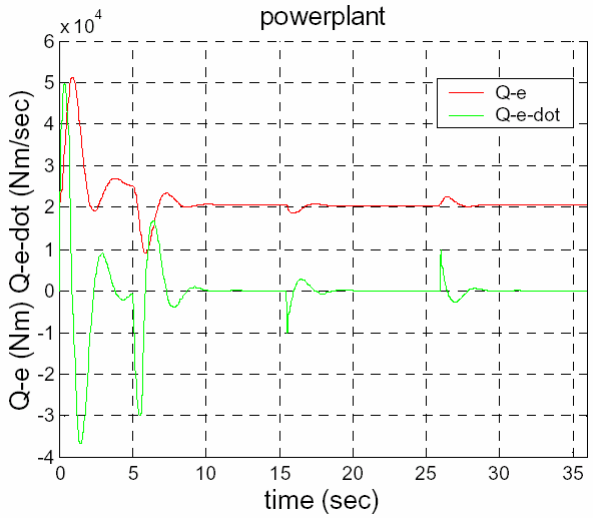


圖 2-1-48 直升機發動機輸出功率的變化

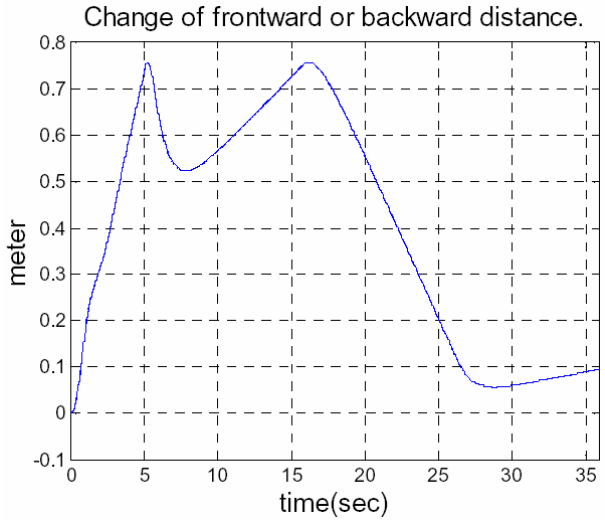


圖 2-1-49 直升機飛行距離的變化(x 座標方向)

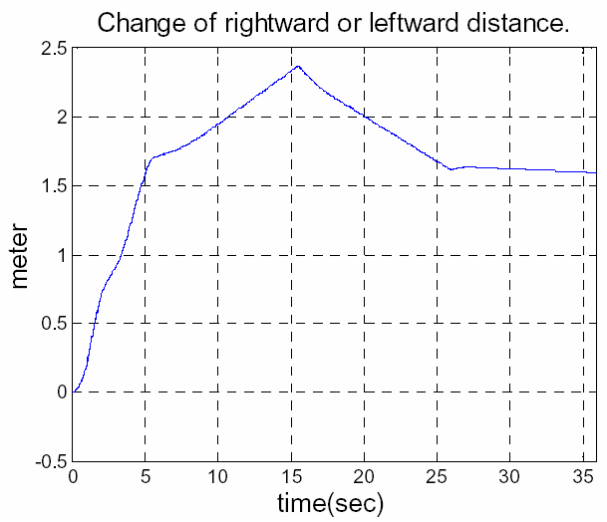


圖 2-1-50 直升機飛行距離的變化(y 座標方向)

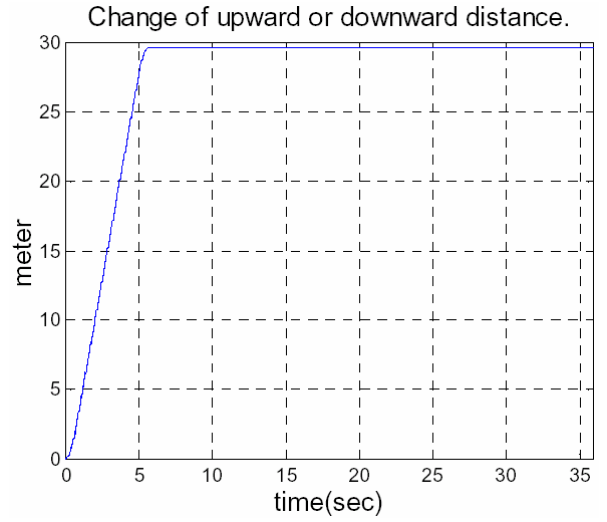


圖 2-1-51 直升機飛行距離的變化(z 座標方向)

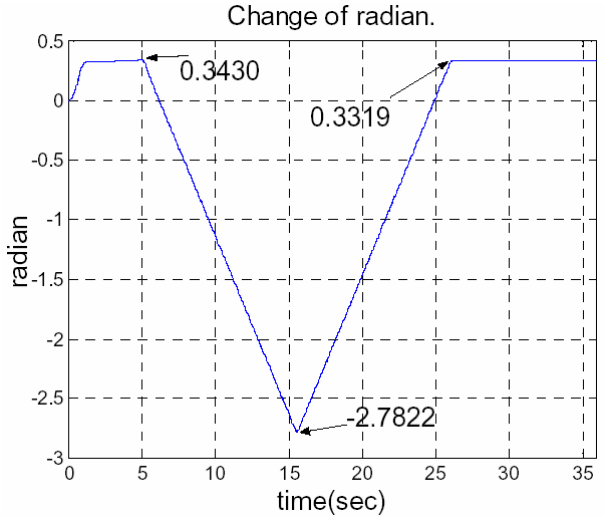


圖 2-1-52 直升機機鼻旋轉的弧度變化，由圖可知 5~15.47 秒機鼻向正方向旋轉約(3.1252(rad)≈360°)，15.47~35.49 秒機鼻向負方向旋轉約(3.1141(rad)≈360°)

模擬三：參考例題一設計的參數進行飛行方向、速度模擬。

本文將直接以模擬結果驗證 (3).i 節中的情況一所述的估算方法，模擬使用的參數設定請參考例一，令 $v_m=u_m=3.928$(m/sec)時，預估得到的合成飛行速度為 5.5(m/sec)、飛行方向為朝左方 45°。由圖 2-1-54 可知，在第 15 秒至第 20 秒時，模擬得到的飛行方向為朝左方，由這時間區間可知飛行軌跡的斜率 $m=\frac{23.4101}{22.7214}=1.03032\cong 1$，可知模擬得到的飛行方向為朝左方約 45°，與預估的十分相近。接下來先參考圖 2-1-58，第 15 秒時控制器實際可達到的速度，所以預估的合成飛行速度為 $\sqrt{v^2+u^2}=5.405(m/\sec)$，但參考圖 30，在第 15 秒至第 20 秒時的實際的飛行速度約為 $(51.3931-31.9082)/(20-15)=3.896(m/\sec)$。可知預估的合成飛行速度是與實際的飛行速度有誤差存在。

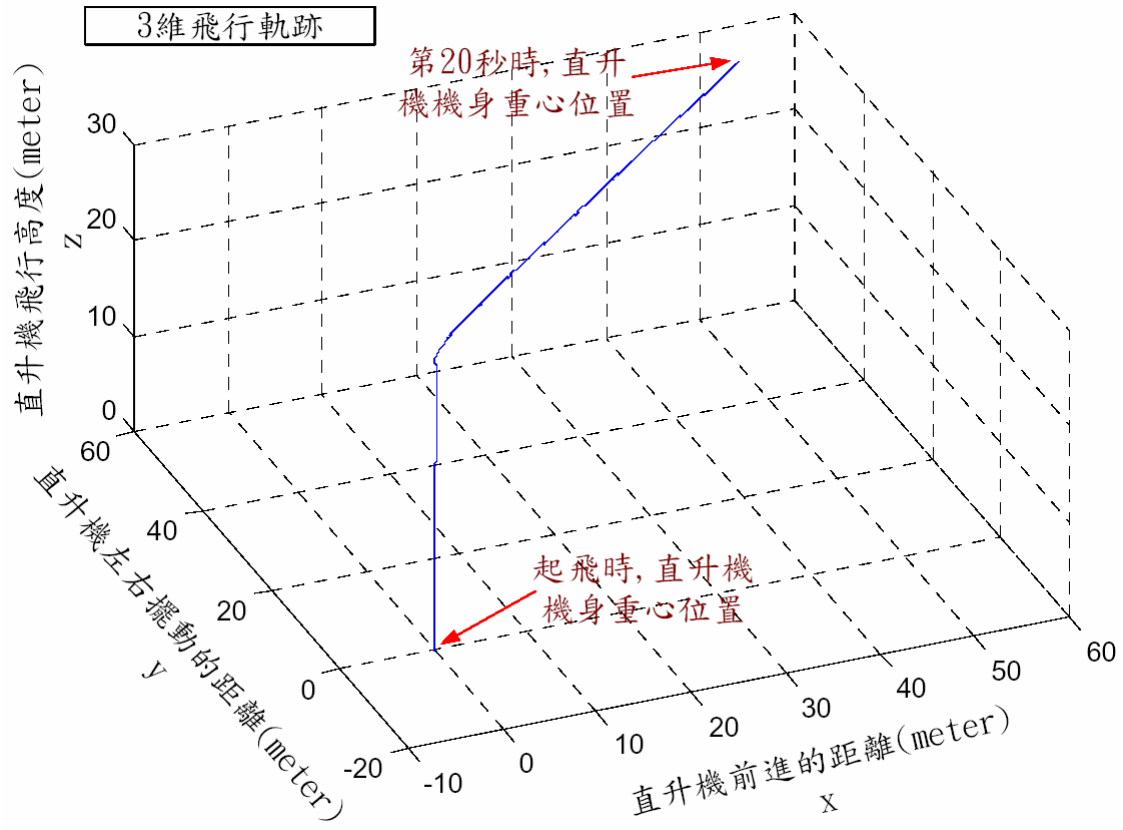


圖 2-1-53 模擬三得到的直升機三維飛行軌跡

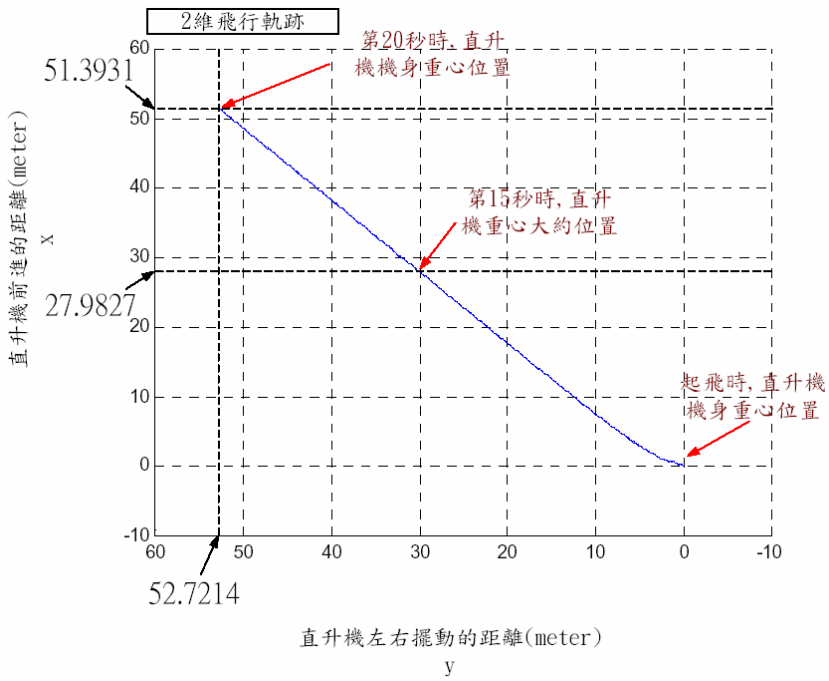


圖 2-1-54 模擬三得到的直升機二維飛行軌跡

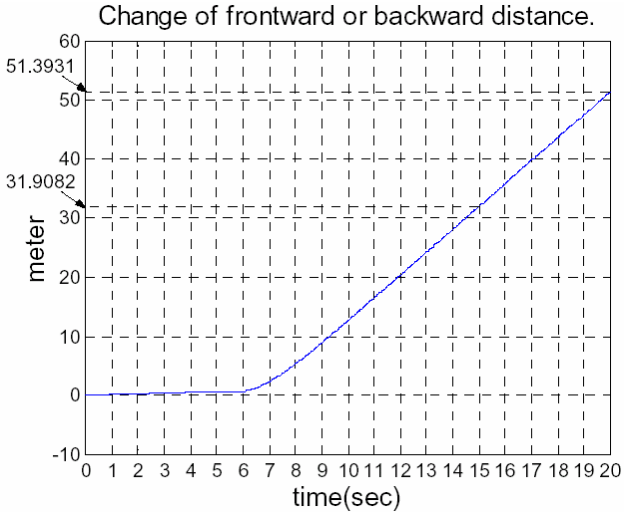


圖 2-1-55 直升機飛行距離的變化(x 座標方向)

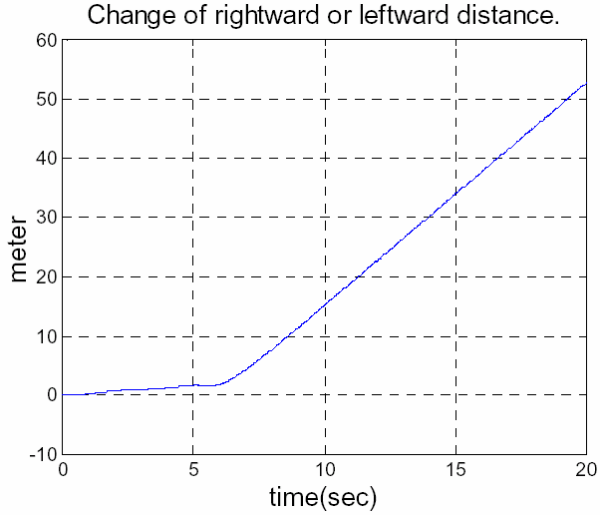


圖 2-1-56 直升機飛行距離的變化(y 座標方向)

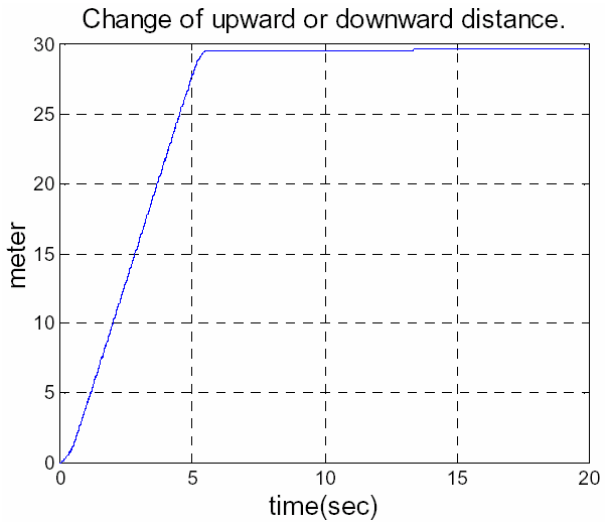


圖 2-1-57 直升機飛行距離的變化(Z 座標方向)

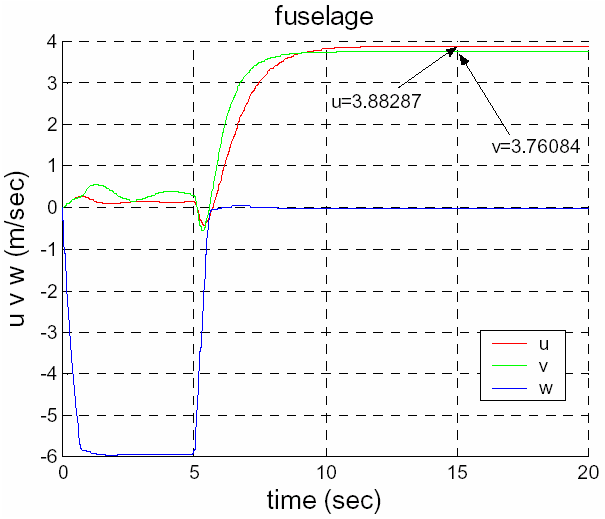


圖 2-1-58 直升機各速度的變化

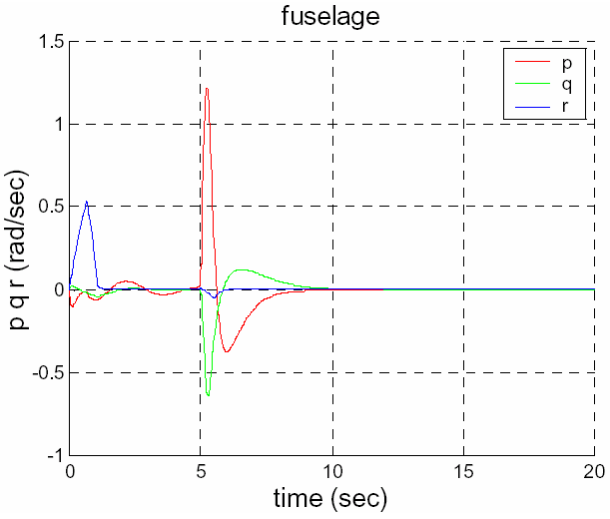


圖 2-1-59 直升機各加速度的變化

由模擬一得到的結果可知，表 2-1-2 所列的基本飛行動作與控制參數設定的關係是成立的。由模擬二與模擬一得到的結果，表 2-1-3 所列的基本飛行動作與控制參數設定的關係是應該可以成立的。由模擬三得到的結果可知，(3).i 節中的情況

一所述的估算方法大致上是可接受的，因為飛行速度只要調整控制器參數即可彌補，不過日後仍需要再重新檢討估算方法的設計。由直升機的各項輸出響應與響應之間的關係可知，雖然有穩態誤差存在，不過考慮控制的對象是速度以及本控制器結構的精簡程度，因此本控制器的性能大致上是可接受的。

2.2 虛擬場景之發展

本子題暨上年度利用 FlightGear 之免費軟體建立開發整個飛行模擬場景，本年度本子題更進一步的探討場景的改善以及功能上的加強。整個飛行模擬場景如圖 2-2-1 之示意圖，在上年度場景在畫面上已經初步完成，在本年度中除了接續改善畫面外，更加入了聲效與多種飛機外型以及觀看之角度變化。為了要達到真實的感覺，特別在視覺、聽覺、觸覺等方面上要予以滿足，所以本子題在場景中，加入環繞喇叭，產生立體音效，使得整體更有臨場之感覺。除此之外，在實驗室中，加上其他週邊系統六軸的史都華平台，以及輸入介面，場景配合平台動作更能表現出我們飛行模擬之目的。另一方面，我們也加強了視點變化的研究，如此能讓操作者在視覺上有更多變化性與選擇性，為了能是視野更加廣闊與感受，並且著手研究多螢幕場景之開發讓整個場景更加廣闊，同時等待硬體設備隨著更新架設完成，之後將可讓操作者更有深入其境之感受。

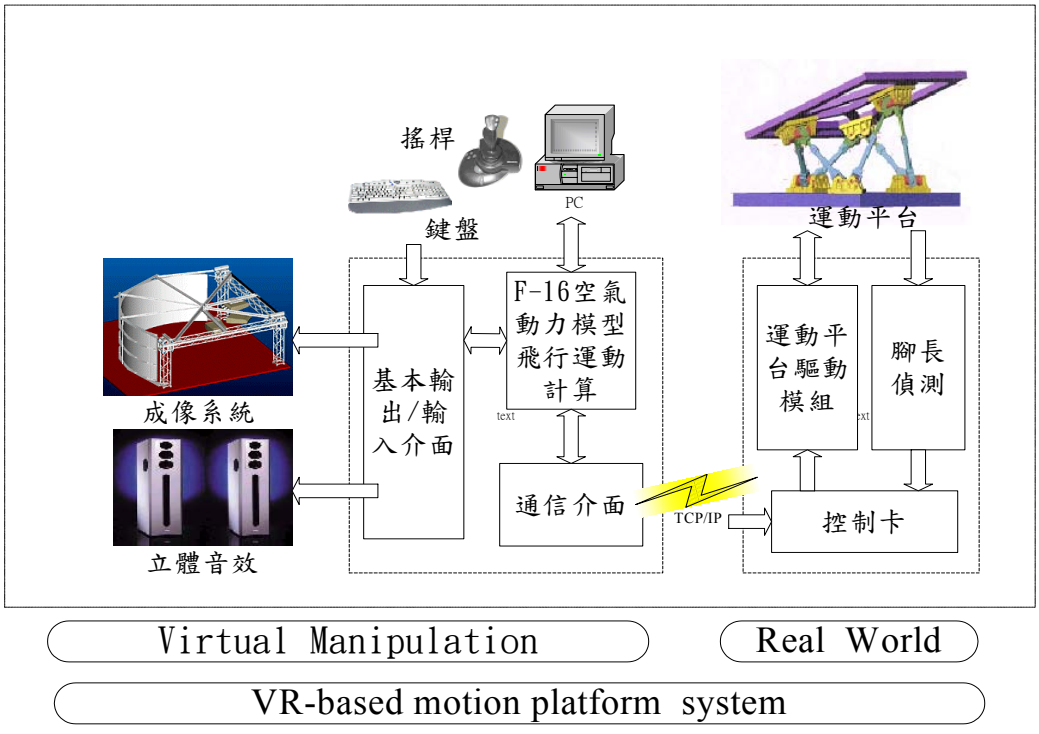


圖 2-2-1　虛擬場景系統整合示意圖

2.2.1 FlightGear 虛擬飛行場景之架構

整個利用 FlightGear 飛行系統之架構可以由圖 2-2-2 來說明，其中主體 FlightGear 套件是一種免費且以公開源碼之方式散佈的軟體【2-6】，可以自由的供人下載修改出自己需要的飛行軟體。在 FlightGear 發展，主要是基於 OpenGL 的函式庫所建構【2-7】，FlightGear 中 PLIB 提供了精簡的繪圖以及聲音之函式庫，透過呼叫 SimGear 函式庫可用來讀取圖形物件的位置及姿態與使用 I/O 及網路溝通等功能。

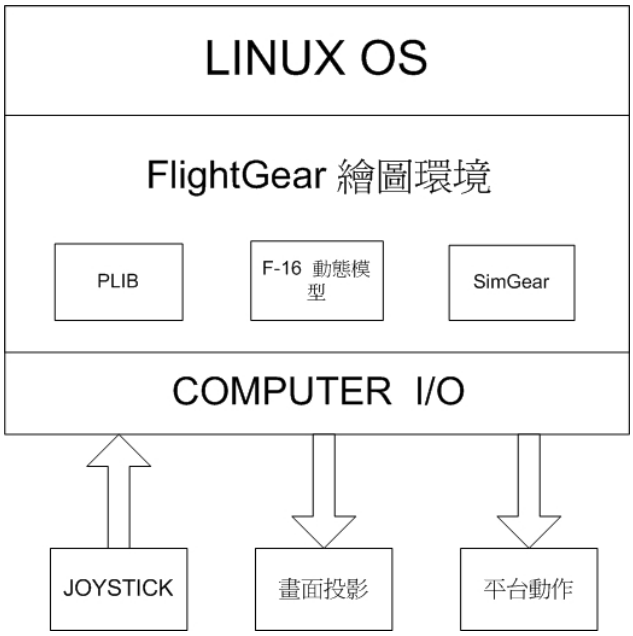


圖 2-2-2 FlightGear 組成架構圖

2.2.1 虛擬場景之音效建立

在第一年的計畫中，我們主要完成的是初步的戰機動態模擬的部分，以及場景初步規畫，但場景中，只有畫面而無音效會使得場景整體之真實感銳減，故在第二年中，加入了場景音效之部分，FlightGear 在增加音效之後，在駕駛上更加有臨場感。因為聲音為人體中重要的感覺之一，重要度不下於視覺，利用聲音可以產生有如現場的立體感受，亦即可以假造聲音定位之感覺，所以在做飛行模擬時會覺得更加真實。再加上環繞音效，使得人會有定位感、空間感，層次感。

FlightGear 音效之建立主要是使用架構 FlightGear 中 PLIB 的函式庫，PLIB 中的 SL 所提供之 Library 是針對於一般遊戲中的音效，不是用來播放音樂使用，對於建立 FlightGear 音效相當之幫助，不但能正確的展示出正確且高品質之音效，對於 CPU 資源之使用也相當少，這對於資源使用度高的繪圖部分，有更多的資源可以使用。

2.2.2 虛擬場景視點之切換

在 FlightGear 上可以擁有多種不同之視角來飛行，使用者可以依照需要而自行

切換所需之視角。除此之外，FlightGear 亦可以自由地變換地圖場景，並在 FlightGear 開放原始碼的網站中，提供了各種場地地圖供人自由下載。圖 2-2-3～2-2-6 為本子題直接利用 FlightGear 所建立之虛擬場景之視點切換結果圖，圖 2-2-7 和 2-2-8 為顯示出不同的場景地圖。

圖 2-2-3　虛擬場景之視點一

圖 2-2-4　虛擬場景之視點二

圖 2-2-5　虛擬場景之視點三

圖 2-2-6　虛擬場景之視點四

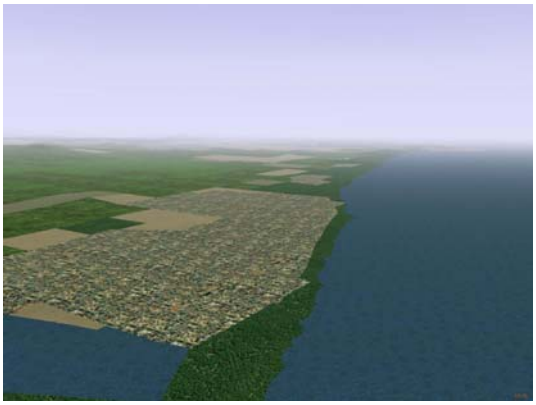


圖 2-2-7　虛擬場景之場景地圖一

圖 2-2-8　虛擬場景之場景地圖二

2.2.3 3D 立體環場顯示

在 3D 立體環場顯示的建構上，本實驗室運用 7 台電腦連線如圖 2-2-9 所示，加上 7 台投影機投射而成一個 3D 立體環繞虛擬空間，中間主要的視覺範圍為利用上下二台 G5 投影機加上偏光鏡頭及 3D 投影屏幕所構成之 3D 立體場景，如圖 2-2-10 所示。其它五台投影機分別投向左前、右前、左、右、後的方向如圖 2-2-11～13，在後方向投影的部分由於場地限制，使用一面鏡子反射投影畫面以加長投影距離，投出的畫面才能符合實際應用大小。每一台投影設備分別由一台電腦控制，電腦之間由網路串連，每個部分投射出的場景，分別是實際量測平台中心到投影屏幕的距離、方位及可視角度，在開發場景中呈現出此參數會投影出的場景，再同步投影建構出此 3D 立體環繞虛擬空間。

圖 2-2-14 為整個 3D 立體環繞虛擬空間示意圖，由電腦連接出來的線分為三種：黑色為網路線，所有電腦的網路相連其中一台為伺服器，大部分的運算皆由此計算而得，再傳輸直昇機的動態及場景的變化與現在的視點到其他電腦，其他電腦把場景的改變表現出來，在根據視點與此電腦相連投影機之位置，轉換視點角度投影出來。紅色線為畫面的一個輸出端，所有電腦皆連到螢幕切換器統一監控。藍色線為畫面另一個輸出端，每個電腦各自連到所屬的投影機。圖 2-2-15、16 為本研究團隊開發之虛擬空間全圖。

圖 2-2-9 7 台電腦連線投影 3D 立體環繞場景

圖 2-2-10 中間立體投影

圖 2-2-11 左前與左方環繞場景投影

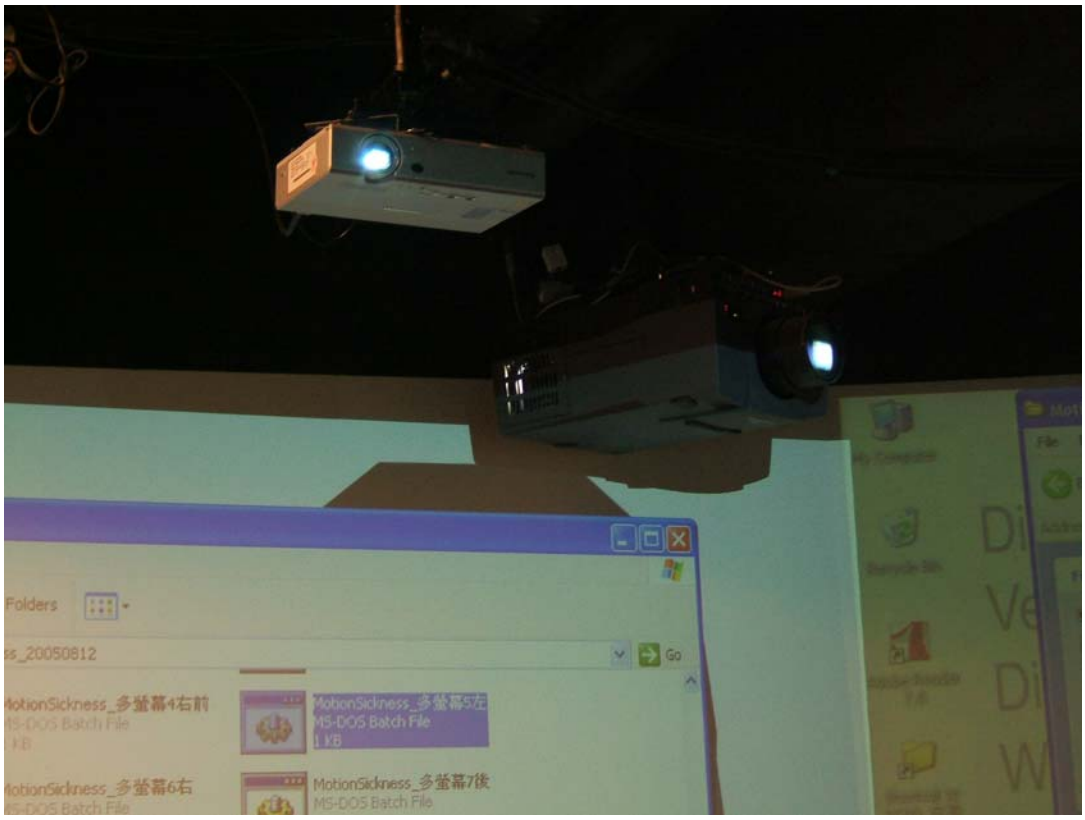


圖 2-2-12 右前與右方環繞場景投影

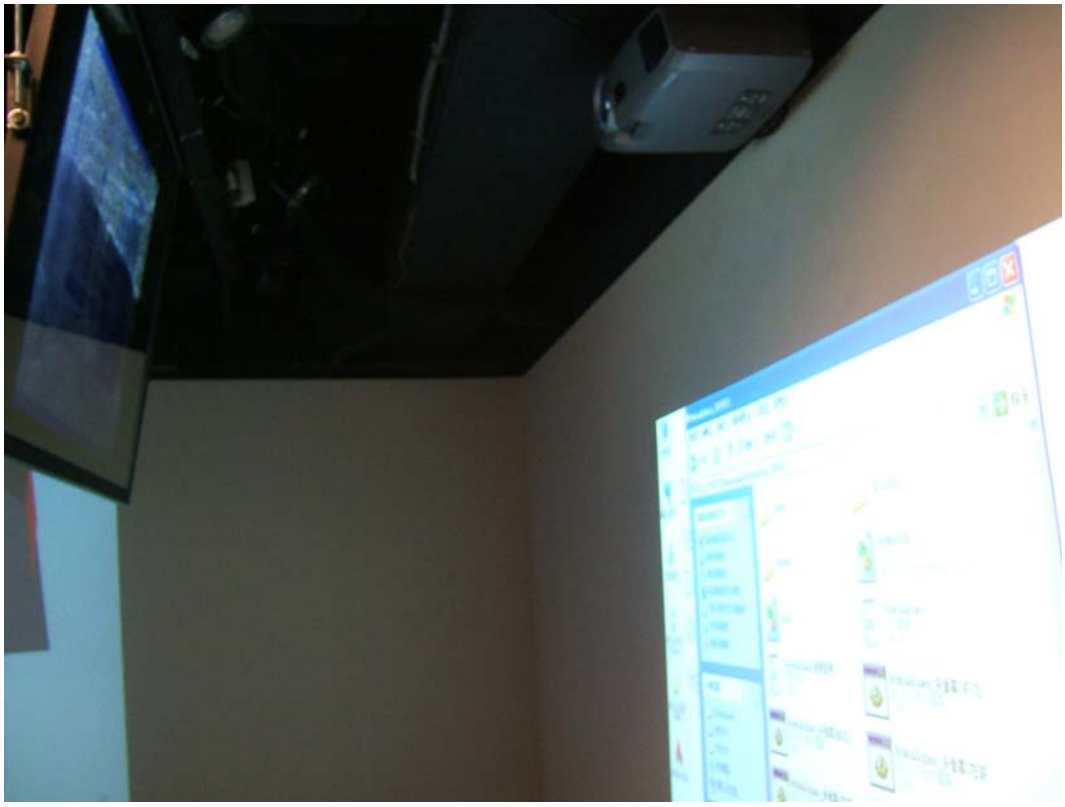


圖 2-2-13 後方反射式投影

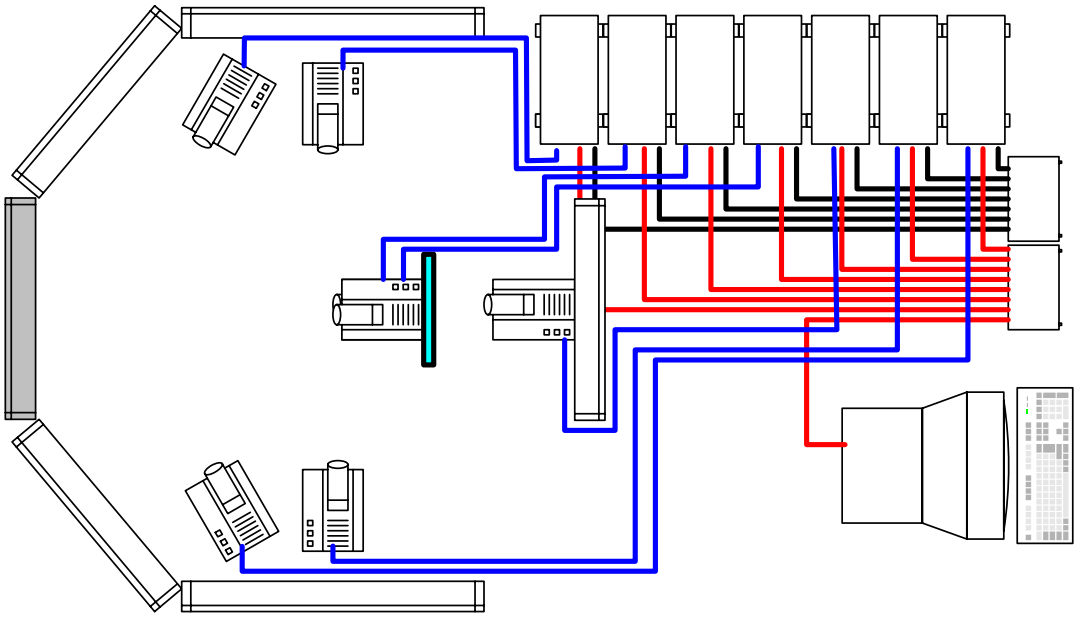


圖 2-2-14 3D 立體環繞虛擬空間示意圖

圖 2-2-15 直昇機 3D 立體環場顯示

圖 2-2-16 模擬車輛駕駛 3D 立體環場顯示圖

子題三：飛機的力資訊處理與操控器研發

本子題的重點在於提供操作者在使用飛機模擬訓練系統時，能感受到彼此互動間接觸力的感覺，並發展能處理力訊號的控制策略以及操控器，以利使用者進行更具真實感、更有效的操縱。由於現代飛行技術的提升，隨著因應各種不同用途而被生產出來的飛機也越來越多，駕駛員的實機訓練越來越不容易，且實機飛行訓練有一定的危險性存在，因此虛擬實境飛行模擬提供了訓練飛行駕駛員的良好解決方案。虛擬實境飛行模擬其實就是利用一套電腦所建立的虛擬飛行場景，利用栩栩如生的風景與擬真的飛機動態、聲光震撼，讓人有身歷其境的飛行感受。藉由地面的飛行模擬，不但可以節省實機飛行的花費，也可以保障人機的安全，並且達成訓練飛行駕駛員的目的。

我們著力於飛行力回饋搖桿的研發以及利用虛擬實境動態模擬系統來模擬飛機的不穩定狀態，例如 PIO 現象。並進一步加以改善搖桿的不穩定現象。然後接著討論 Pilot-Induced Oscillation (PIO)問題，首先簡單介紹 PIO 發生的可能原因與近幾年來各種偵測 PIO 的方法。為了克服 PIO 發生的危險性，本子題建立一套模糊偵測系統來偵測 PIO，並試著用一套相位補償的方式來降低 PIO 傾向。最後利用飛行模擬系統做些初步實驗，並將所得之實驗數據對 PIO 偵測器加以模擬驗證，由實驗與模擬結果發現利用相位補償的方式確實可以降低 PIO 傾向，達到安全飛行的目的。

3.1 力回饋搖桿

力回饋搖桿提供使用者和虛擬場景間互動的一個良好介面。透過力回饋搖桿，使用者可以控制虛擬場景中的物件，並可因場景的變化或碰撞等事件的發生而產生力回饋，讓使用者有身歷其境的真實的感受。一般專業力回饋搖桿的售價約在數十萬元，而本子題自製的力回饋搖桿卻能以較為合理的價格，提供虛擬實境飛行模擬用的新選擇。且因其具有泛用性，因此可透過改變增益比例，切換成操作從戰鬥機至客機等各種大小不同的飛機的搖桿。本子題自製之力回饋搖桿其系統與運作流程如圖 3-1-1 所示，由搖桿下達控制命令至 PC 的虛擬場景，接著由 PC 送出位置訊號，經由馬達控制卡轉換成轉矩命令送至驅動器，形成力回饋感受。本節首先介紹本子題自製力回饋搖桿的機構部分，接著在說明其控制部分，最後指出其在飛行模擬中所佔的位置。

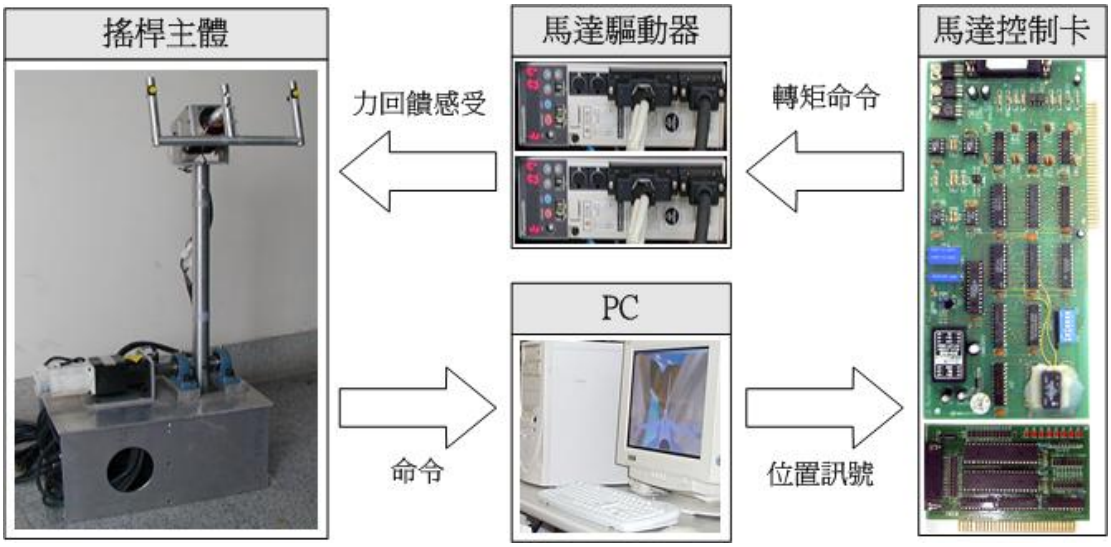


圖 3-1-1　力回饋搖桿系統及其運作流程

3.1.1 機構部份

圖 3-1-2 所示為自製方向舵力回饋搖桿的機構圖及其 X 軸、Y 軸的活動情形，此搖桿有 X 軸、Y 軸兩個自由度，X 軸可像汽車方向盤般 360 度自由轉動，Y 軸較類似一般搖桿的操作感受，只能作前後約 40 度的活動，兩軸的活動方向近乎垂直且各自擁有獨立的活動空間。X 軸與 Y 軸的力臂分別為 15 公分及 54 公分，外型上與實際飛行搖桿十分類似。而一般力回饋搖桿只需手和手腕即可操作，最大輸出力量約在 10N 左右，Immersion 公司的 Impulse Engine 2000 專業力回饋搖桿只有 8.9N。而本子題所自行設計開發之力回饋搖桿為了避免搖桿上下重量分配不均勻，所以在 Y 軸下方掛上鐵塊以保持平衡，造成操作除了手和手腕外，還需要手肘及肩部的施力，且在兩軸與馬達間有齒輪連接，藉以提高輸出力量至 43N 及 60N，比起一般搖桿更有能有實際操作飛機的感受。

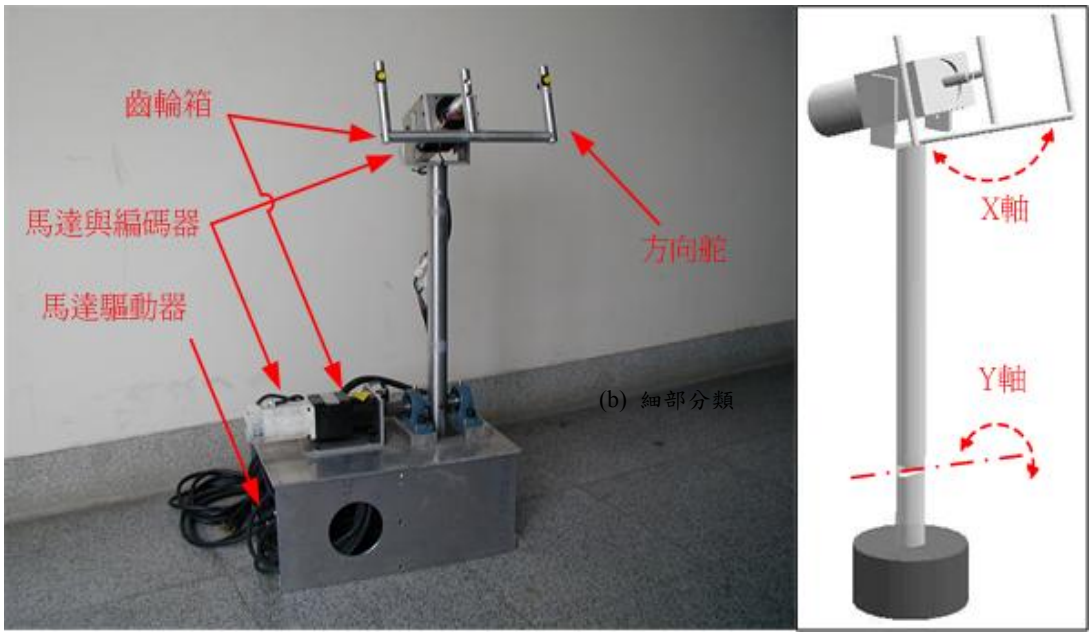


圖 3-1-2　力回饋搖桿的機構與其 X 軸、Y 軸的活動情形

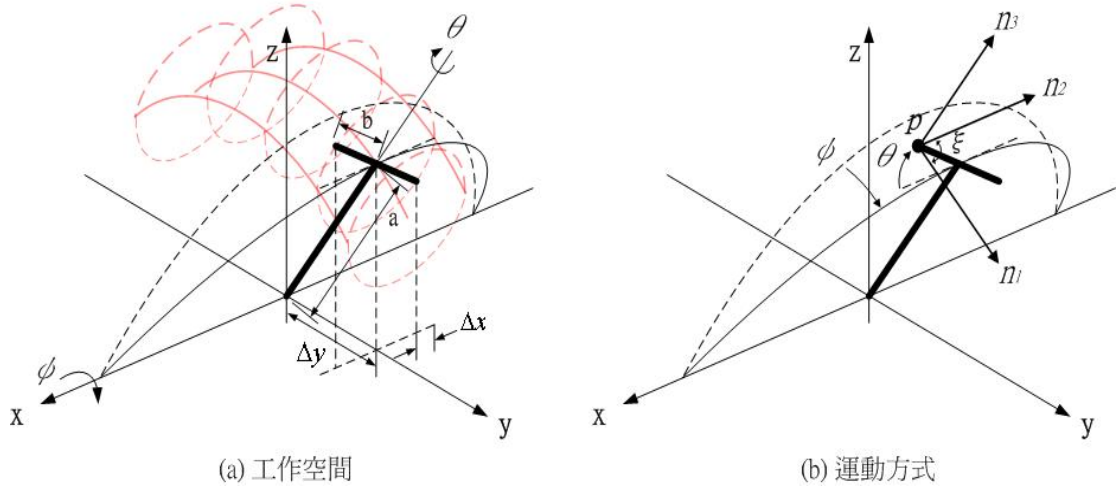


圖 3-1-3　自製力回饋搖桿的工作空間與運動方式

圖 3-1-3(a)顯示力回饋搖桿的工作空間，其中空間座標的原點定位於搖桿 Y 軸的旋轉軸心處，原點到搖桿 Y 軸末端的距離定義為 a，搖桿 X 軸端點至中心的距離定義為 b，而 ϕ 與 θ 分別為 X 軸與 Y 軸的馬達旋轉量。圖 3-1-3(b)則顯示搖桿的運動方式，ϕ 的變化導致方向舵上 ***p*** 點的移動，其方向為 n_1 且形成軌跡 T_x，同理 θ 造成方向為 n_2 且軌跡 T_y 的運動，T_1 與 T_2 的交點位於 ***p*** 點，其方向為 n_3，可由 n_1 與 n_2 的外積(Cross Product)求得【3-1】，若令 $\psi = 90° + \theta$ ，則可以得到如下的結果

$$n_3 = n_1 \times n_2 = \frac{\cos\phi\sin\psi}{\sqrt{1-\sin^2\phi\sin^2\psi}}\vec{i} + \frac{\sin\phi\cos\psi}{\sqrt{1-\sin^2\phi\sin^2\psi}}\vec{j} + \frac{\cos\phi\cos\psi}{\sqrt{1-\sin^2\phi\sin^2\psi}}\vec{k} \quad (3.1)$$

其中 $\phi = [\phi_{\min}, \phi_{\max}]$ 與 $\psi = [\psi_{\min}, \psi_{\max}]$ 對應於搖桿機構的限制，ϕ 為 $\pm 40°$ 之間，ψ 則

沒有角度限制。(3.1)式的分母項係由於 n_1 與 n_2 在操作過程中不是完全地相互垂直，而是有一角度存在，令兩者間的不垂直性(nonorthogonality)為$|90°-\xi|$，其中

$$\xi = \cos^{-1}(\sin\phi\cos\psi) \tag{3.2}$$

若$|\phi|$、$|\psi| \le 25°$時

$$0.98 < \sqrt{1-\sin^2\phi\cos^2\psi} \le 1 \tag{3.3}$$

表示兩軌跡平面 T_x 與 T_y 近似近似於相互垂直。若我們將圖 3-1-3(a)與圖 3-1-3(b)互相對應，則於工作空間中方向舵上的 $\boldsymbol{p}$ 點可以表示為

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \frac{1}{\sqrt{1-\sin^2\phi\sin^2\psi}} \cdot \begin{bmatrix} \cos\phi(a+b\sin\psi) \\ \sin\phi(a+b\cos\psi) \\ \cos\phi(a+b\cos\psi) \end{bmatrix} \tag{3.4}$$

當兩旋轉量ϕ與ψ很小時，方向舵上的 $\boldsymbol{p}$ 點可以表示為

$$p \approx a \cdot \begin{bmatrix} \psi \\ \phi \\ 1+\dfrac{b}{a} \end{bmatrix}_{small\ \phi,\psi} \tag{3.5}$$

此時工作空間近似為一個平面，這樣的特性賦予使用者更直覺性的控制過程，不會因為原柱狀的工作空間造成操作上的困擾。且經由操控形成的位移，也可以直接解離成ϕ與ψ兩分量，如此一來能更簡單地分析控制策略。此外我們定義一變數 $q = [\phi, \psi, (a+b)]$與式(3.3)成為可逆轉換(Invertible Mapping)，即 $p = L(q)$，由此可導出其 Jocobian 矩陣

$$J = \frac{\partial p}{\partial q} = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} \tag{3.6}$$

接著我們討論馬達輸出力矩與手的操作力矩之間的關係。首先定義 $\rho = a + b\cos\psi$ 為工作空間中的原點至方向舵上手施力的等效作用點的距離，馬達輸出力矩為τ_ϕ與τ_ψ，則傳縱至等效作用點的力量 F 可表示為【3-1】

$$F = J^{-T}P \tag{3.7}$$

其中 J 為 Jocobian 矩陣，$P=\left[\tau_\phi,\tau_\psi,F_\rho\right]^T$ 為對應於 q 的力量成分，第三成分 F_ρ 為順著搖桿操控方向的力量，利用(3.6)式可將 F 展開為

$$F=\begin{bmatrix}F_x\\F_y\\F_z\end{bmatrix}=\begin{bmatrix}\left(\dfrac{\cos\psi\sqrt{1-\sin^2\phi\sin^2\psi}}{\rho\cos\phi}\right)\tau_\psi\\\left(\dfrac{\cos\phi\sqrt{1-\sin^2\phi\sin^2\psi}}{\rho\cos\psi}\right)\tau_\phi\\\left(\dfrac{\sin\phi\sqrt{1-\sin^2\phi\sin^2\psi}}{\rho\cos\psi}\right)\tau_\phi-\left(\dfrac{\sin\psi\sqrt{1-\sin^2\phi\sin^2\psi}}{\rho\cos\phi}\right)\tau_\psi\end{bmatrix} \tag{3.8}$$

當兩旋轉量 ϕ 與 ψ 很小時，傳送至等效作用點的力量 F 可近似為：

$$F\approx\frac{1}{\rho}\cdot\begin{bmatrix}\tau_\psi\\\tau_\phi\\\phi\cdot\tau_\phi-\psi\cdot\tau_\psi\end{bmatrix}_{small\ \phi,\psi} \tag{3.9}$$

顯然 F_Z 較小於其他力量成分。由(3.8)式可知在 X 軸與 Y 軸方向上 τ_ϕ 與 τ_ψ 都有可能產生解耦合(decoupling)的現象，唯一會有耦合(coupling)發生的只有在 Z 軸上。

3.1.2 控制部份

為了正確讀取搖桿的位置，及輸出適當的力回饋給使用者，我們選用一整組可提供力矩控制的馬達、編碼器、馬達驅動器、以及馬達控制卡(健昇 NCC9322)，而為了讀取馬達驅動器上所提供的力矩資料，另外需要用到一張 Multi-I/O 卡，如圖 3-1-4 所示。當使用者透過搖桿下達命令，編碼器會將其命令轉換成位置信號，透過馬達控制卡與電腦中的虛擬場景相結合。而當虛擬場景中發生的事件欲傳回一力回饋訊號時，首先會先透過馬達控制卡，將力資訊依照馬達驅動器的額定電壓與額定轉矩等比例換算成適當的電壓，接著馬達驅動器便會依照此電壓對馬達下達命令，最後馬達靠著齒輪箱增大輸出力量，產生力回饋的感受。

接著我們對搖桿作一個簡單的性能測試，測試力量命令與輸出力量是否一致，用以鑑定力回饋搖桿的輸出性能。測試方法為用手固定住操控器方向盤使其不動，分別對 X 及 Y 軸輸入 0.5Hz 的正弦波力量命令，f_{xc} 及 f_{yc}，如 (3.10) 及 (3.11) 式所示

$$f_{xc}=20\cdot\sin\left(2\pi\cdot0.5t\right) \tag{3.10}$$
$$f_{yc}=30\cdot\sin\left(2\pi\cdot0.5t\right) \tag{3.11}$$

同時由量測馬達的力矩，計算其在 X 及 Y 軸的輸出力量，f_{xo} 及 f_{yo}，整理測試結果如圖 3-1-5(a)及圖 3-1-5(b)所示，將 X 軸的力量命令及其測量輸出力量繪成圖 3-1-5(c)，Y 軸部分繪成圖 3-1-5(d)，由此二圖可以看到，力量命令及其輸出力量幾乎成斜率為 1 的直線，故此系統可以達成正確地控制輸出力量這項要求。

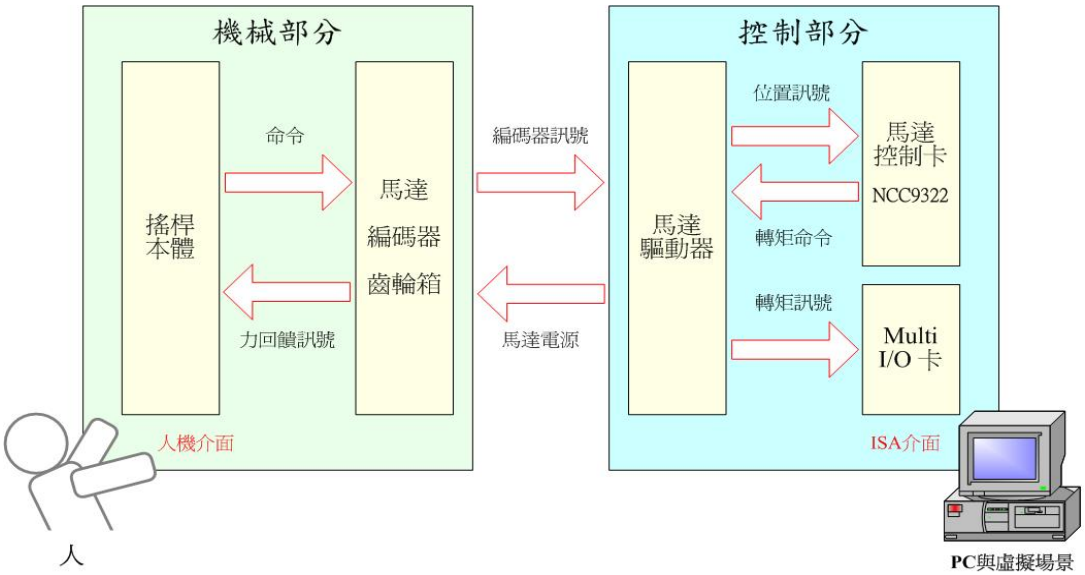
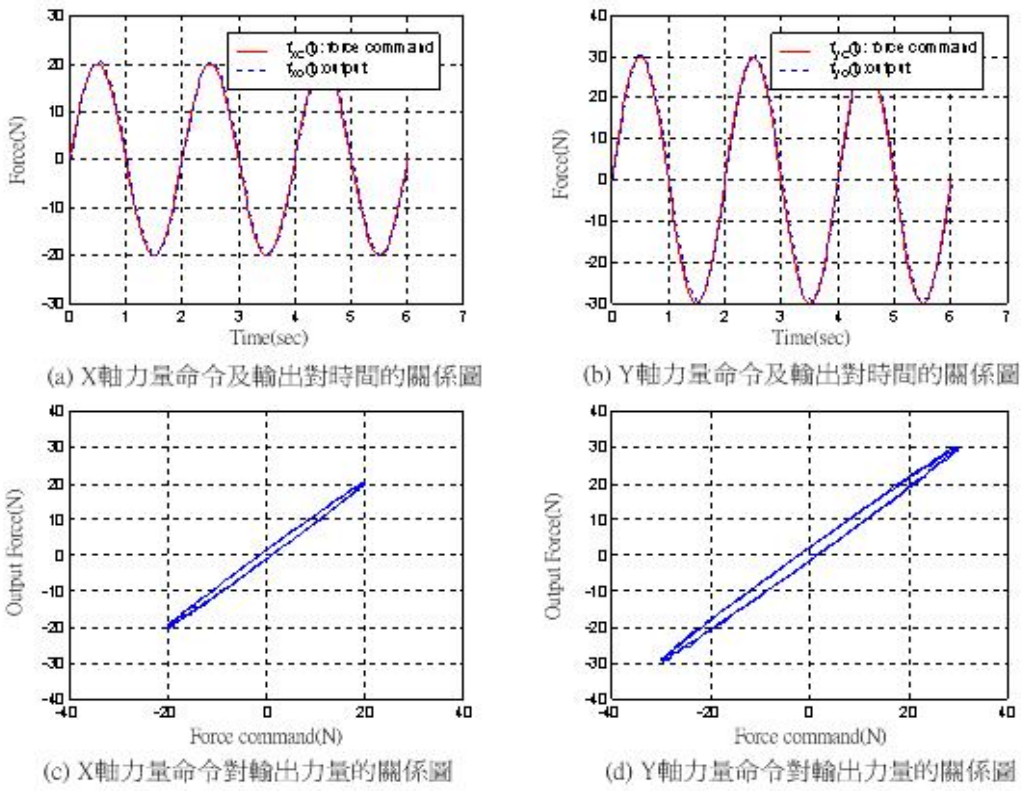


圖 3-1-4　人、搖桿與電腦間的互動



(a) X軸力量命令及輸出對時間的關係圖　(b) Y軸力量命令及輸出對時間的關係圖

(c) X軸力量命令對輸出力量的關係圖　(d) Y軸力量命令對輸出力量的關係圖

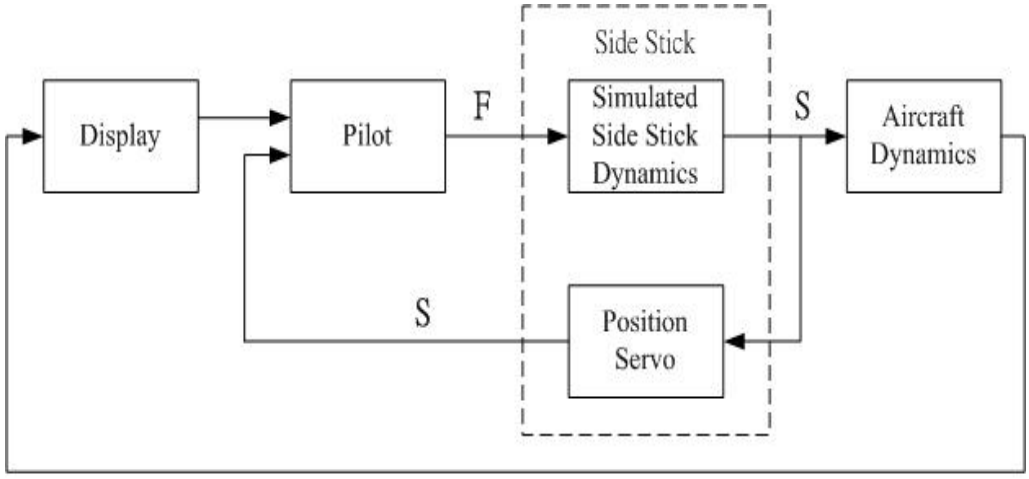
圖 3-1-5　力量命令與輸出力量關係測試結果圖

至於飛行系統中的駕駛員是如何利用搖桿去控制飛機的飛行動態方式，大致上可分為主動式與被動式兩種控制方式【3-2】，以下簡單說明兩種不同控制方式之差異性如下

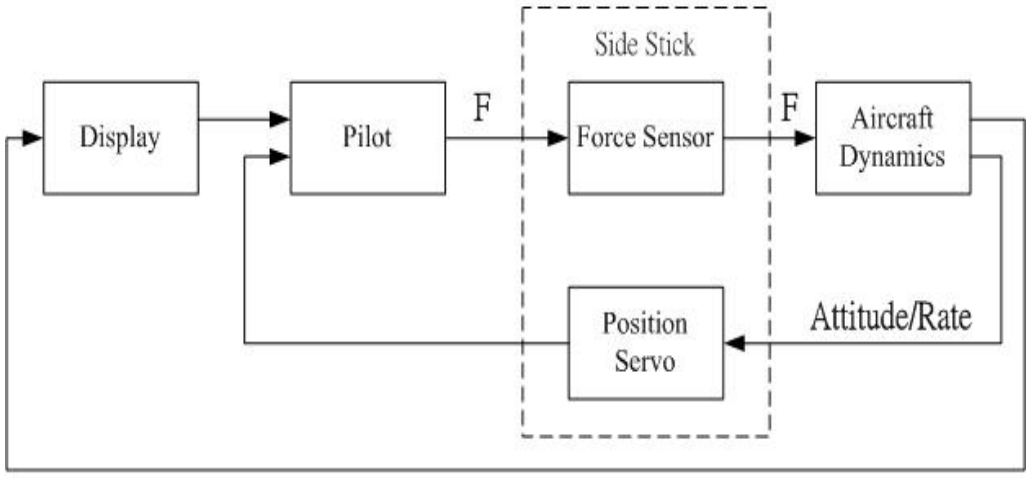
被動式搖桿是經由使用者對搖桿下達力量命令，經由模擬的搖桿動態轉換成位置訊號後，一方面送至飛機動態模型中，另一方面由位置控制伺服裝置迴授至使用者端，如圖 3-1-6(a)所示。

主動式搖桿則是在使用者下達力量命令後，經由力感測器將讀出的力資訊輸入至飛行控制系統中，而飛機的狀態或輸出變數再經由位置控制伺服裝置迴授至使用者端，如圖 3-1-6(b)所示。

簡單來說，被動式搖桿的使用者所感受到的是模擬的搖桿動態(彈簧、阻尼等)，而主動式搖桿的使用者所感受到的卻是飛機的動態。



(a) 被動式搖桿



(b) 主動式搖桿

圖 3-1-6　被動式與主動式搖桿架構圖之比較

3.2 搖桿系統分析

在第一年的計畫中我們致力於力回饋搖桿的製作，以及以此力搖桿來實現操作者與虛擬場景間的雙向互動，為了能夠更逼真的模擬此互動帶給操作者的感受，除了希望力搖桿與虛擬實境能更密切的結合，今年度之計畫我們進一步地對搖桿本身的系統作更佳完整性的分析，藉此找出最適合力搖桿的操作模式並改善搖桿的一些不完美之處。

對每套利用數位取樣的模擬系統來說，由於取樣週期、資料的量化、時間的延遲使得要模擬真實世界連續系統的物體，有其模擬的極限，而一套互動式力回饋設備的性能通常決定於其所能模擬的物體阻抗大小的範圍，而這個範圍通常被系統本身裝置的物理性質所影響。一方面模擬的阻抗物件的精確度受到如馬達的非線性輸出以及感測器的解析度等等的影響，而另一方面系統的取樣時間、感測器解析度還影響了系統模擬的穩定度。【3-3, 3-4, 3-5】

圖 3-2-1 為一個簡化的數位系統阻抗控制方塊圖，我們可以利用控制方塊圖以及前一節所討論的系統物理性質，分別以精確度和穩定度兩大重點討論這些系統性質對於整個搖桿系統性能的影響。

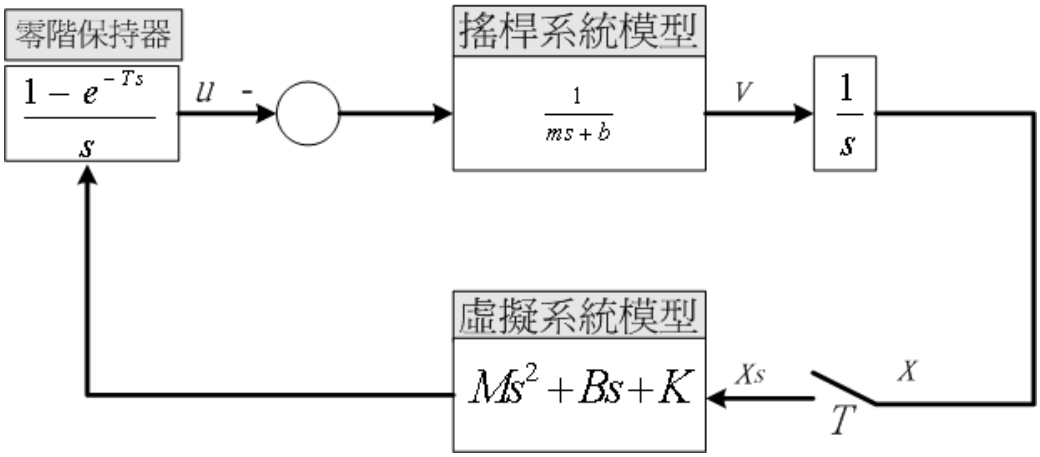


圖 3-2-1　數位阻抗控制系統方塊圖

3.2.1 阻抗模擬精確度

我們要利用數位模擬系統模擬一個虛擬物體，由於數位系統取樣存在的量化誤差，將造成模擬的物體與期望的物體物理性質有一定的差距，例如我們模擬一個簡單的虛擬彈簧，我們先定義期望的虛擬彈性係數為 K_d，真正感覺到的真實彈性係數為 K_r，彈簧模擬誤差為 err_k：

$$err\_k = \frac{K_r - K_d}{K_d} \tag{3.12}$$

利用基本模擬法則，可知我們需要模擬的馬達輸出力 F_m 如下

$$F_m = -K_d X \tag{3.13}$$

上式（3.13）中的 X 為量測到的位移量，若我們假設量化最小單位為 δ，量化誤差為 ε，則搖桿實際的位移量 x 可得如下

$$x = X + \varepsilon(t) \qquad -\delta/2 \le \varepsilon(t) < \delta/2 \tag{3.14}$$

而使用者真正感覺到的彈性 K_r 是彈力 F_h 隨實際位移量影響為

$$F_h = -K_r x = -K_r [X + \varepsilon(t)] \tag{3.15}$$

在靜止狀態，無其他外力下 $F_h = F_m$ ，所以由(3.13)式與(3.15)式可得下列關係式

$$\frac{K_r}{K_d} = \frac{X + \varepsilon(t)}{X} \tag{3.16}$$

$$err\_k = \frac{K_r - K_d}{K_d} = \frac{\varepsilon(t)}{X} \qquad -\delta/2 \le \varepsilon(t) < \delta/2 \tag{3.17}$$

由(3.17)式可知彈簧的模擬誤差正比於量化單位的大小，且工作的位移量越小將使得模擬的誤差更大，我們可以在之後的彈簧模擬實驗中看出這樣的結果。

3.2.2 系統穩定度

為了探討整組搖桿系統之穩定度分析，以下我們針對以下分就「簡單彈簧模擬穩定度問題」、「完整系統穩定度分析」、「摩擦力補償」幾個方向討論與探討，其說明如下。

3.2.2.1 簡單彈簧模擬穩定度問題

一個系統的穩定與否通常可以由這個系統的能量為收斂還是發散來判斷，在真實世界中的基本阻抗原件都是被動性原件，只會儲存或是消耗能量，不會造成能量的增加，但經過數位系統模擬的虛擬阻抗原件卻會因為需要取樣時間間隔的關係，有可能會變成主動原件，我們可以用一個模擬彈簧說明這個現象。圖 3-2-2 即為模擬一個虛擬彈簧，使用者一個簡單的壓縮彈簧後放鬆的動作所得的結果，圖中每一次的位移量為操作速度與取樣週期的積。由於位置編碼器取樣以及零階保持器的關係（zero order hold）每週期系統的輸出位移與輸出力量值都會保持一整個取樣週期，也造成如圖的情況。能量可為施力與位移的積，而當使用者壓縮

彈簧時，彈簧系統會儲存紅色曲線以下的面積大小的能量，但放鬆彈簧確會釋放出藍色曲線以下的面積大小的能量，很顯然的釋放的能量比原本供應儲存的能量要大，也造成虛擬彈簧系統的不穩定因素。但搖桿系統本身的物理阻抗卻可以消耗能量抵銷由於數位系統取樣所造成的能量發散，若我們考慮一個簡單的模擬彈簧系統，整個系統方塊圖如圖 3-2-3 所示。

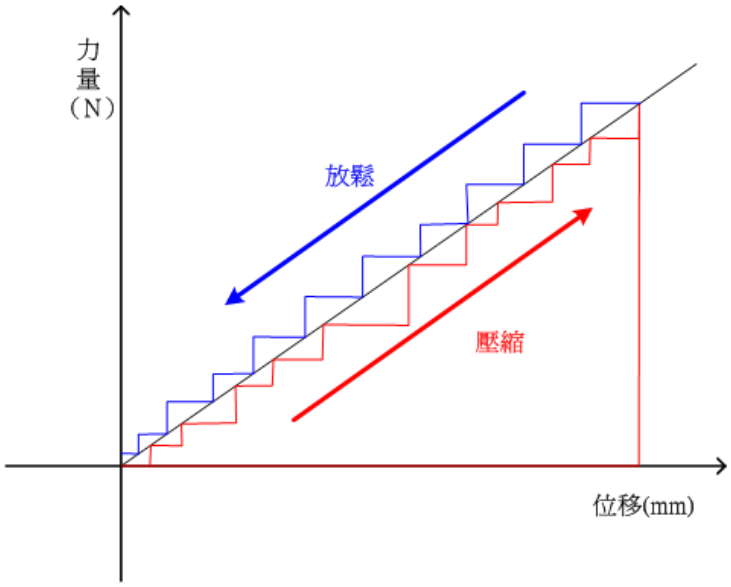


圖 3-2-2　虛擬彈簧能量示意圖

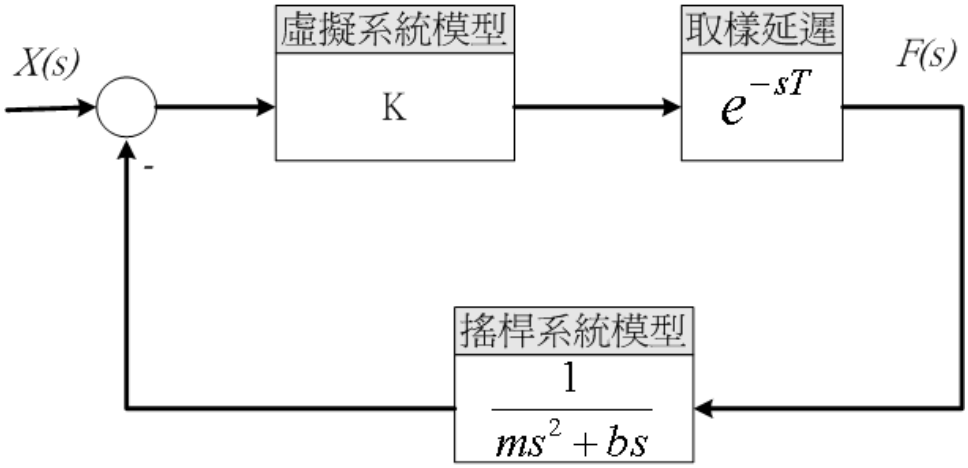


圖 3-2-3　模擬彈簧系統方塊圖

在圖 3-2-3 中，整個系統輸入為位置訊號 $X(s)$與輸出訊號為力量 $F(s)$，則轉移函數可獲得如下

$$G(s)=\frac{F(s)}{X(s)}=\frac{Ke^{-sT}}{\left[1+K(\frac{1}{ms^2+bs})e^{-sT}\right]}=\frac{(ms^2+bs)Ke^{-sT}}{\left[ms^2+bs+Ke^{-sT}\right]} \qquad (3.18)$$

若取樣週期 T 與自然頻率 s 的乘積非常小（約小於 0.1），則我們可以將 e^{-sT} 做二階泰勒展開近似為

$$e^{-sT} = 1 - sT + \frac{1}{2}s^2T^2 \tag{3.19}$$

將(3.19)式代入(3.18)式中，我們可得

$$G(s) = \frac{(ms^2 + bs)Ke^{-sT}}{\left[(m + \frac{1}{2}KT^2)s^2 + (b - KT)s + K\right]} \tag{3.20}$$

為了分析系統之穩定度，由奈式穩定準則(Nyquist stability criterion)可知，假如上述之系統轉移函數要穩定的話，其必須滿足所有的極點全在左半平面之限制條件，也就是分母多項式的解的實數項必須皆小於零。如此，進一步求(3.20)式的極點為

$$\frac{(KT - b) \pm \sqrt{(b - KT^2 - 4(m + \frac{1}{2}KT^2))}}{2m + KT^2} \tag{3.21}$$

所以為了使系統穩定，模擬的彈簧係數 K、取樣週期 T、與系統黏滯係數 b，必須滿足 $KT < b$ 的關係式，由於這是一個近似解，還存在一個常數項 C，使穩定關係式修正為 $KT < cb$，而 Minsky 等人對常數項的建議值為 2【3-3】，因此整個系統穩定關係式變為

$$b > \frac{KT}{2} \tag{3.22}$$

3.2.2.2 完整系統穩定度分析

上述的穩定度討論並沒有加入使用者介入的部分，而通常一個動態系統若加入人為因素時，其穩定度就會變得很難預測，因為人的動態表現變化範圍非常大，難以預估，且隨當時的狀態人的動態傾向穩定系統或是故意造成系統動盪，都會使穩定度的估測結果造成極大的差異。但我們可以把將使用者的動態當成一個未知系統 Z_O (s)，包含入搖桿的動態之中，將真實世界的系統當成 G(s)，模擬的阻抗狀態作為迴授系統 H(z)，考慮被動性的穩定。

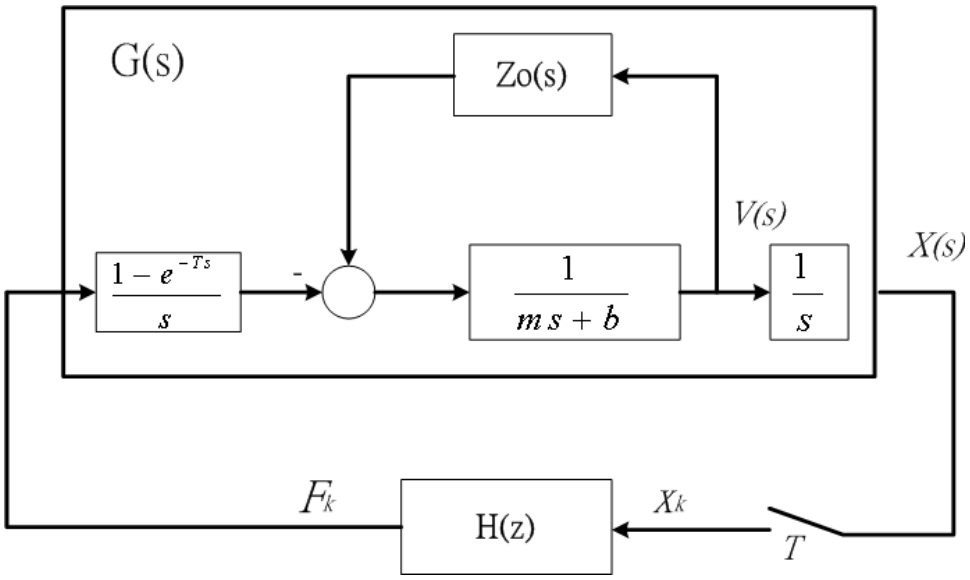


圖 3-2-4　加入使用者動態 Zo 之系統方塊圖

根據 Colgate 所推倒的被動性理論(Passivity criterion)分析，數位取樣系統要能滿足被動性，必須滿足下式【3-4】

$$b > \frac{T}{2}\frac{1}{1-\cos\omega T}\mathrm{Re}\{(1-e^{-j\omega T})H(e^{j\omega T})\} \qquad 0 \le \omega \le \omega_N \tag{3.23}$$

其中 $\omega_N = \pi/T$ 為奈氏頻率(Nyquist frequency)。若我們模擬的物件包含了彈簧與阻尼系統（spring&damper system），虛擬的彈性係數為 K，黏滯係數為 B，則迴授系統 H(z)變成

$$H(z) = K + B\frac{z-1}{Tz} \tag{3.24}$$

將(3.24)式代入(3.23)式之中，則可得

$$b > \frac{T}{2}\frac{1}{1-\cos\omega T}\mathrm{Re}\{(1-e^{-j\omega T})(K + B\frac{e^{j\omega T}-1}{Te^{j\omega T}})\} \qquad 0 \le \omega \le \omega_N \tag{3.25}$$

由線性代數運算，令 $e^{j\theta} = \cos(\theta) + j\sin(\theta)$ ，上式可簡化為

$$b > \frac{T}{2}K - B\cos\omega T \qquad 0 \le \omega \le \omega_N \tag{3.26}$$

以下就分別討論系統是否有黏滯係數之分析

第一種情況：若模擬的系統沒有黏滯係數（$B=0$），則上式結果與 3.2.2.1 的結果類似，從(3.26)式也可以清楚知道，要增加模擬系統的阻抗值且仍然維持系統的被動性，可以增加取樣的頻率或是加裝阻尼器提高搖桿系統的物理黏滯係數。

第二種情況：若系統輸入模擬有黏滯係數($B \neq 0$)，由(3.26)式中可知，當 $0 \le \omega < \omega_N/2$ 時，模擬的黏滯係數 B 會幫助系統穩定，但當 $\omega_N/2 \le \omega < \omega_N$ 時，模擬的黏滯係數 B 反而會破壞系統穩定，而系統的共振頻率(harmonic vibration angular frequency) ω 為

$$\omega = \sqrt{\frac{K}{m}} \tag{3.27}$$

其中 K 為模擬彈簧係數，m 為等效質量。若要使黏滯係數 B 幫助系統穩定，則下式必須成立

$$\sqrt{\frac{K}{m}} > \frac{\pi f}{2} \tag{3.28}$$

上式中的 f 為系統的取樣頻率，我們分別代入系統 X 軸方向與 Y 軸方向等效質量 m_x=0.0345Kg 與 m_y=0.0315Kg，對上式做圖，分別為圖 3-2-5(a)與圖 3-2-5(b)，可以由圖看出在不同的狀況下，黏滯係數 B 對系統的影響。

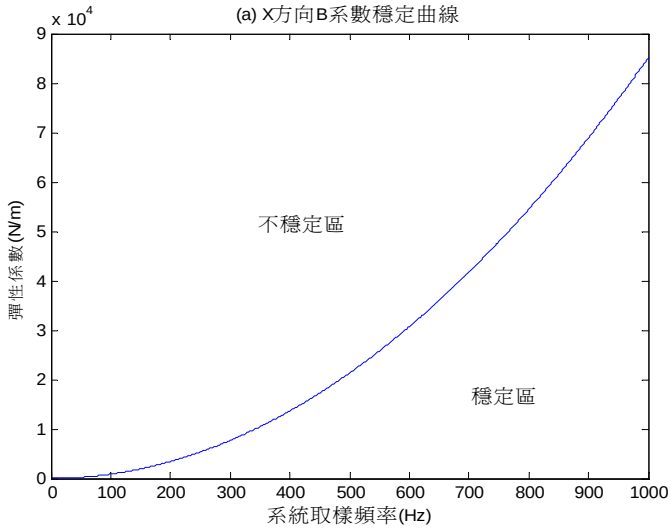


圖 3-2-5(a)　在 X 軸方向之 K-f 變化下黏滯係數對系統穩定關係圖

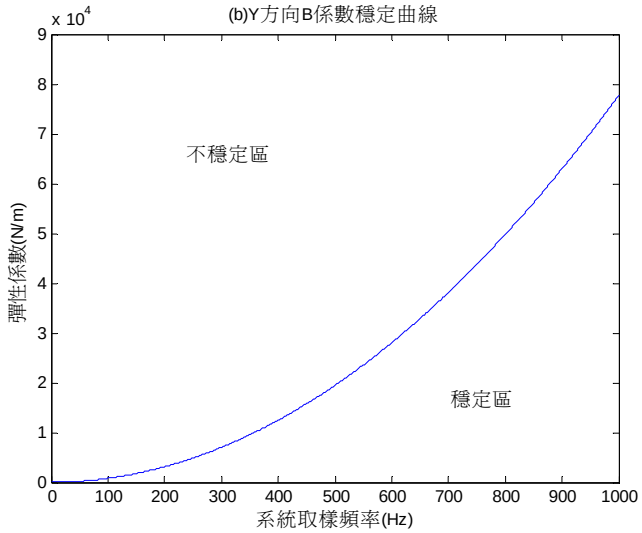


圖 3-2-5(b)　在 Y 軸方向之 K-f 變化下黏滯係數對系統穩定關係圖

3.2.2.3 摩擦力補償

從上一小節中可知實際物理系統的黏滯係數可以增加系統的被動性，這是由於系統本身的黏滯性會提供一個作用力，其大小正比於運動速度，且方向與系統運動方向相反，這作用力會使得系統運動速度漸緩。與這個現象非常相似的是系統的摩擦力，曾有相關文獻提出摩擦力對於力量-位置控制的系統有穩定補償的作用。【3-6】

摩擦力雖不為速度的函數，但也同樣的提供一個與運動方向相反的作用力，摩擦力所提供的穩定力與黏滯係數差了一個運動速度的純量倍數，因此我們可修正(3.22)與(3.26)中的 b 值

$$b_{new} = b + \mathrm{f}_{friction} \frac{1}{\|v\|} \tag{3.29}$$

摩擦力的穩定效應在系統慢速操作下，將會發揮較大的影響。

3.3 PIO 之產生因素

PIO (Pilot-Induced Oscillation) 常見於現今的飛機事故之中，主要是由於相位延遲所導致，除了會造成飛機的損害外，更有可能造成人員的喪生。本節首先先解釋 PIO 的重要性，歸納 PIO 發生的原因，並回顧近期的 PIO 事件以及介紹各種不同的預測 PIO 方式。接著介紹模糊 PIO 偵測器的理論基礎，模糊變數與其歸屬函數的定義與模糊法則，最後提出相位補償的概念，將其與 PIO 模糊偵測器結合，減少發生 PIO 的傾向。安全飛行的示意圖如圖 3-3-1 所示，經由相位補償後的飛行訊號在前置處理後輸入模糊推斷系統中，最後可以得到 PIO 的估測值，藉

此達到安全飛行的目的。

模糊邏輯是以模糊集合為基礎，將二值邏輯推廣為多值邏輯的演算，主要優點在於直接攫取來自於專家的經驗法則即可建構出受控系統的控制法則，而不需要受控體的數學模型，這是跟傳統控制最大的不同點，但仍具有傳統控制之強健控制特性，尤其適用於非線性時變、時延系統的控制。由於不需要了解受控體的數學模型，利用專家等的經驗即可建構出模糊法則，藉此估測 PIO，因此模糊邏輯的方法適用於虛擬實境動態模擬系統中，並可廣泛應用到各種不同的飛機。

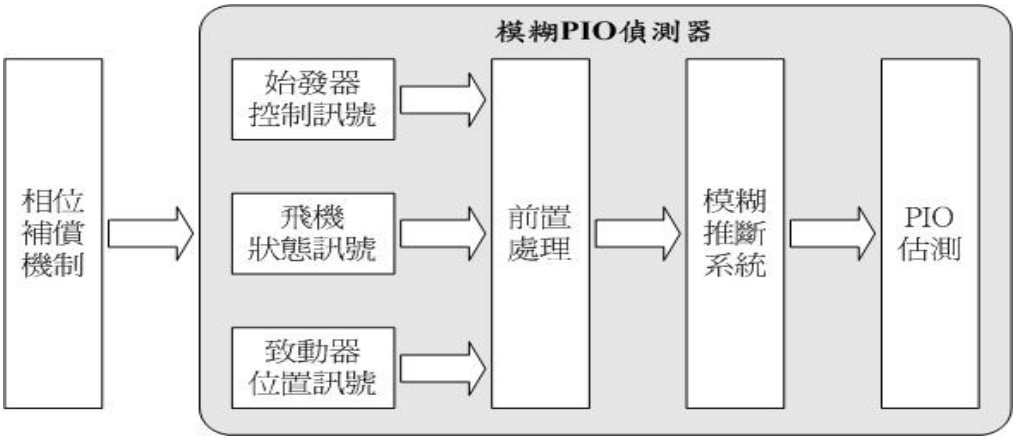


圖 3-3-1　安全飛行的示意圖

3.3.1 PIO

PIO【3-7】是一種 APC (Aircraft-Pilot Coupling)現象，各種類型的飛機都有可能發生。當駕駛員遭遇了機身的一些不穩定狀況，因為本身的反應不夠快或經驗不足，而無法提供適當且精確的補償控制時，或在錯誤的時間裡提供了錯誤方向的補償控制，致使不穩定的振盪無法被有效地消除，甚至會發散成為更大幅度的振盪。通常 PIO 的發生到發散成為極大振盪只需要極短的時間，因此 PIO 大多屬於無法復原的不穩定現象。

PIO 在之前以機械鏈結系統 (Mechanical Linkage System) 為主的飛行系統中，就已經時有所聞。而在提升氣動特性、降低飛機的操縱面積和重量的前提下，取而代之的線傳飛控系統 (Fly-By-Wire) 中，PIO 的案例越來越多，逐漸受到人們的重視。

圖 3-3-2 為一常見的飛行控制系統架構圖，駕駛員藉由搖桿或踏板等輸入命令，經由控制法則的計算後傳達命令至致動器，致動器再帶動整台飛機的動作，感應器的迴授訊號則為控制法則提供了一個補償的機制。其中速率限制器一開始是被設計用來防止駕駛員有過大的輸入以及致動器有超載的現象發生，但卻有可能因為駕駛員的不當操作，造成嚴重的時間或相位的延遲，因而導致 PIO (Pilot-Induced Oscillation，駕駛員誘發振盪)現象的發生。

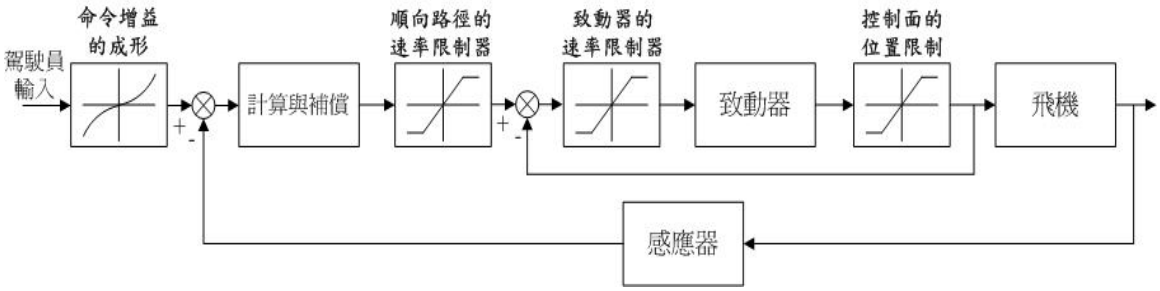


圖 3-3-2　飛行控制系統架構圖

3.3.2 近期發生的 PIO 事件

最早受到注意而引起廣泛討論及研究的 PIO 事件是發生於 1961 年 5 月 18 日，由美國海軍駕駛員於新墨西哥州的 Albuquerque，以低海拔的高度駕駛 F-4H 雙引擎戰鬥機所發生的。該駕駛員原本想要打破最高飛行速度的紀錄，而當他在海拔 200 英呎的高度以趨近於 1.1 馬赫的速度飛行時，發生了劇烈的 PIO 現象，在短短 2 秒鐘之內飛機產生了 3 個振盪並承受了-4g 到+14g 的重力。結果導致了飛機機身的解體與駕駛員的不幸喪生。雖然此次事件已經被這底調查，但是並沒有合理解釋 PIO 的理由，最後歸納原因可能是由於飛機失去了俯仰 (pitch) 方向的阻尼，或是配平 (trim) 系統故障所導致。

SAAB 公司所生產的 JAS 39 Gripen 所使用的是線傳飛控的控制系統，在 1988 年 12 月以來的五次成功的飛行測試之後，在第六次飛行測試中也發生了 PIO 的現象。由於駕駛員之前沒有駕駛過該新機型，因此在遇到亂流時，駕駛員作了過猛的週期控制，嘗試去控制飛機的航線傾角，超過了油液壓操縱面致動器的速率限制，伴隨而來的是振幅極大的 PIO 現象，使得飛機墜毀。官方的解釋是控制系統因為高精確度的輸入被驅動至非線性的速率限制。

PIO 亦常常與採用新設計和新技術有關，如美國的一架 YF-22 試驗機，在 1992 年 4 月 25 日愛德華空軍基地低空飛過跑道時墜毀。飛機從 12 公尺的高度開始，產生四、五次振盪後撞在跑道上。分析表示，當時飛機並沒有發生故障，事故是由於一些意料不到的事件引起的，主要原因是駕駛員在低空時沒有將推力控制切斷，駕駛桿很小的移動產生了意想不到的很大的輸出響應，進而造成 PIO 事故。

其他如 X-15、YF-16、F-18、A-320、MD-11、V-22、B-2 等飛機都曾經在飛行或著陸時發生過 PIO 的事故，有的甚至造成飛機的墜毀與人員的傷亡。而 PIO 現象通常還是以戰鬥機較為常見，因其較常有激烈的操作，客機較為稀少。

3.3.3 PIO 發生可能原因

現代飛機的架構與控制系統日新月異，對駕駛原來說確實是一大挑戰。前面所描述的那些飛機在遭遇 PIO 之前都曾讓許多不同的人駕駛過，然而沒遭遇過 PIO 的卻不一定比那些遭遇過 PIO 者有較熟練的飛行技術，這讓人聯想到 PIO 的發生必定還有其他原因。但是儘管達到之前發生 PIO 過的某種飛行狀態，PIO 也不一定會發生。因此，PIO 的發生必定是由很多種因素同時達成才會發生。

一般來說，引發 PIO 的原因可以通稱為觸發事件 (trigger)。這些觸發事件包括了駕駛員快速地改變飛行控制策略，因改變飛行狀況或完成精確度較高的任務而造成控制增益的改變等，觸發事件可分為下列幾種

(1) 駕駛員的操縱過猛，加上系統的增益太高，或過於靈敏的搖桿，容易導致操縱面偏轉和偏轉速率達到極限，達到速率限制 (Rate Limit)，這種飽和現象會引起過度的相位遲滯或時間延遲，因而造成飛機不穩定。

(2) 在線傳飛控系統中，駕駛員的操縱並不直接與操縱面相連，而是將控制指令以電子信號送到操縱致動器，使駕駛員感受不到運動的速率和操縱面已達到最大偏轉的狀態，這有可能造成飛行員所希望的響應與實際響應不一致，進而產生 PIO 現象。

(3) 操縱模式間的轉換有時也會引發 PIO 事故。線傳飛控系統如果設計得好，飛行操縱系統各模式間的轉換應是平穩的，而且不干擾駕駛員對飛機的操縱，但實際情況並非完全如此，模式間的轉換有時也會引發 PIO 現象。

(4) PIO 通常在駕駛員執行精確度要求很高的任務時發生，如著陸、起飛或空中補給燃料等，這時他要集中精神精確控制飛機的航行。事故往往在使要求很高的任務中斷、或需要更高精度時突然發生，普遍的原因是飛行操縱系統增益校準得不好。如圖 3-3-3 所示，觸發事件與駕駛員和制策略間透過複雜的互動，才有可能產生 PIO 的現象。

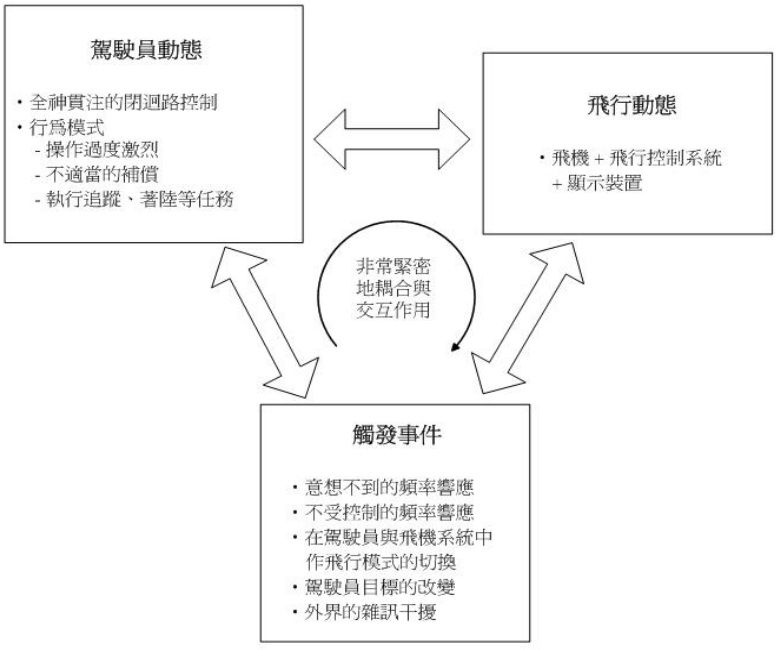


圖 3-3-3　PIO 形成原因間的互動

也有人指出 PIO (或 APC) 這種人與飛機間不穩定的耦合現象的現象並不是駕駛員的過錯，而是在飛行控制性統的設計上有瑕疵，所以有人把 PIO 改名成為 Pilot-Involved Oscillation，或以 APC 來概括 PIO 的現象 (因為有些 APC 並不會

產生振盪)，詳細的 APC 和 PIO 分類圖 3-3-4 所示。

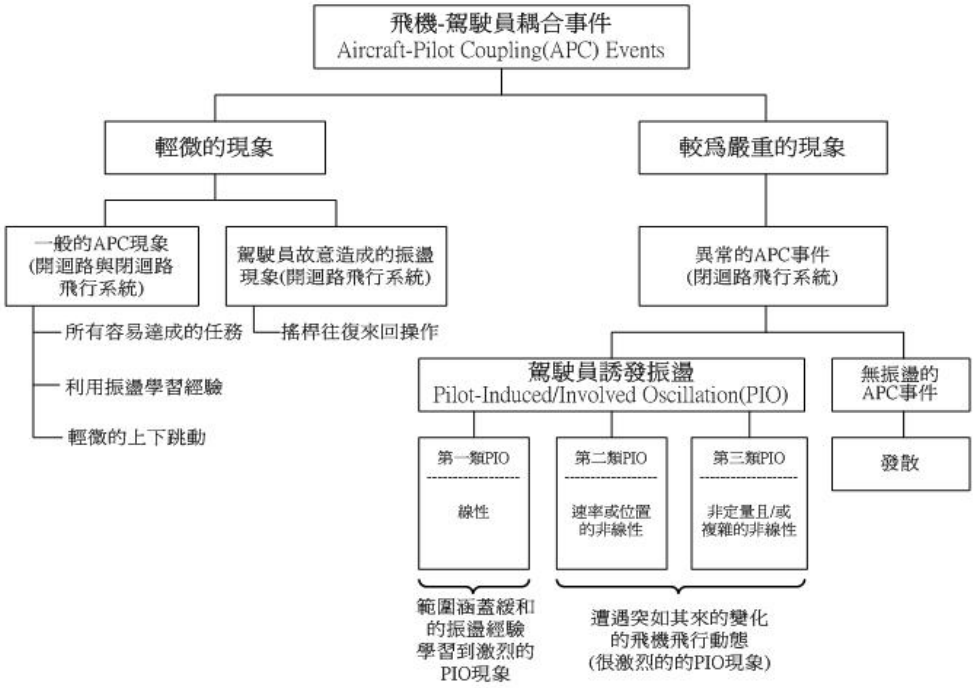


圖 3-3-4　APC 與 PIO 的分類

3.4 模糊 PIO 偵測器

模糊邏輯推論系統具有專家經驗的模糊規則庫，就如同專家推斷 PIO 事件一樣，本子題利用飛行時頻域與時域的變數，與一套完整的模糊法則(Fuzzy Rule)，可提供一個即時的 PIO 偵測裝置。其中頻域的前置處理係使用遞迴有限傅立葉轉換的方法，用以抽取出控制與狀態訊號中的主要頻率與振幅，並決定它們彼此之間相位偏移的情況；而時域的前置處理主要是要抽取出致動器的速率，因為它的飽和與否關係著是否會使得 PIO 現象更為嚴重。以下將對各部分詳細加以說明。

3.4.1 理論基礎

如之前圖 3-3-1 所示，雖然 PIO 的全名為 Pilot-Induced Oscillation，此種現象是由於駕駛員的不當駕駛所引起的，但事實上 APC (Aircraft Pilot Coupling)更能正確地闡述此種閉迴路的不穩定現象，必須是要由駕駛員、飛機與飛行控制系統、及觸發事件的交互作用才有可能導致各種振幅大小不同的發散性振盪現象。而此模糊偵測器所著重的是非預期且振幅較大的 PIO 現象，由於 PIO 的現象都是在一個短暫且不正常的環境下發生，此套模糊偵測系統若與虛擬實境飛行模擬場景結合，則駕駛員可以不受安全顧慮與飛機結構的限制，探索各種飛行狀態下的 PIO 傾向，並可進一步進行補償減輕 PIO 現象。

在這個模糊偵測器中，用來作為模糊推論系統與前置處理的模糊變數(Fuzzy

Variables)共可區分為四種主要特徵，每種特徵都是經由之前所列各種參考資料的結論與案例分析而來，四種特徵詳細描述如下：

(1) 當飛機的姿態讓搖桿位置變化的小訊號產生了將近180° 的相位延遲：這種系統的延遲通常會導致飛機頻率響應的增大，進而使駕駛員對飛機的控制喪失了正確的判斷能力，並且認為這個控制系統已經故障。

(2)駕駛員下達了極大振幅的命令：當駕駛員不能去辨別出某些不同往常的飛機頻率響應特性時，他可能會把一個處於延遲狀態中的命令視為已經無反應而中斷它，而且會透過搖桿等控制方式增強他的命令。這種駕駛員增益的增加通常會使得飛機頻率響應延遲的情況更為嚴重，且駕駛員會嘗試以更大幅度的反向控制去補償它，進而造成發散的 PIO 現象。

(3) PIO 現象的主要耦合頻率通常落在約 0.3 至 1.5Hz 的範圍中：這是在駕駛員操作於同時補償模式下常見的低頻 PIO 範圍，更高頻率的 PIO 也有可能會發生，但是它們通常發生在高頻手控系統與高頻飛行模式耦合的時候，而且這些振盪也會因為太快速而超出人的認知範圍。

(4)控制系統的某個部分(如操縱面的致動器)達到速率飽和：當控制命令的速率超出了致動器的設計範圍時，速率飽和的現象就會導致整個系統的延遲現象。當駕駛員以較積極的高增益控制去執行一個高精確度的追蹤任務時，飛機就有可能會產生不穩定的振盪現象。

就如同專家推斷 PIO 事件一樣，模糊邏輯推論系統使用了上述四個指標鑑別 PIO 事件。

3.4.2 即時傅立葉轉換

利用參考文獻的方法，控制與狀態的訊號以 0.05 秒的的取樣區間作離散化，並以複數頻率因子作比例上的調整。這些序列提供對每個有限傅立葉轉換頻率成分的累加結果，這種遞迴離散的轉換有以下的形式

$$X_i(\omega_k) = X_{i-1}(\omega_k) + x_i e^{-j\omega_k(i\Delta t)} \tag{3.30}$$

離散傅立葉轉換是由 $\omega_k = 0.1$ 計算至 2.5Hz (或 0.63 至 15.7rad/sec)，這包含了常見的 PIO 發生的頻率範圍。遞迴關係式若被修改成只考慮過去不久前的頻率成分，利用加時間窗的方式將窗之前訊號的頻率成分移除可得到如下的關係式

$$X_i(\omega_k) = X_{i-1}(\omega_k) + x_i e^{-j\omega_k(i\Delta t)} + x_{(i-w)} e^{-j\omega_k(i-w)\Delta t} \tag{3.31}$$

即時傅立葉轉換的方塊可以對有興趣的頻率範圍作向量化的離散有限傅立葉轉換。經過前置處理後的模糊變數接著使用類似類神經 PIO 機率偵測器的方法偵測 PIO，唯一的不同點為類神經偵測器是以經傅立葉轉換後兩訊號間具有最小相位延遲的頻率取其平均作為 PIO 的主要頻率，而模糊偵測器則是以具有最大振幅

的控制訊號來當作 PIO 的主要頻率，因其較易判斷之故。相位延遲以餘弦函數取代角度來表示，可以減少因相位平移±360° 時所造成的頻率響應誤差。一旦主要頻率決定選定之後，所對應的控制與狀態的振幅與相位延遲的餘弦函數就會成為如圖 3-4-1 所示模糊推論系統的模糊變數，其中時域的兩個模糊變數分別是致動器的速度與加速度。

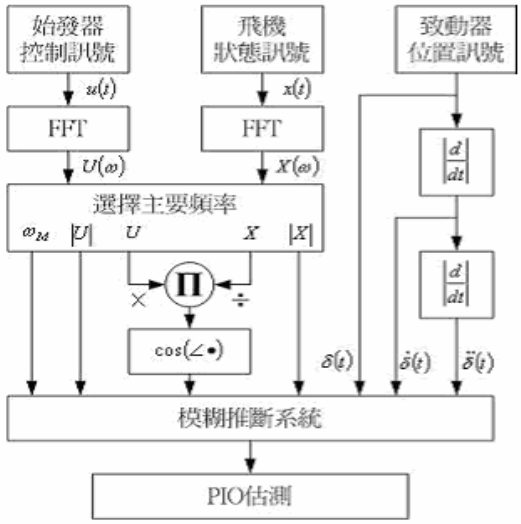


圖 3-4-1　PIO 偵測的前置處理

3.4.3 模糊變數與其規屬函數的定義

對於每個模糊變數，針對它們可能發生 PIO 的範圍，定義模糊集合的歸屬函數(Membership Function)來對它們作模糊化，但是某些歸屬函數最後將被視為是多餘的而不被包含在模糊法則中。以下是此模糊 PIO 偵測器中各個模糊變數及其歸屬函數的定義

主要搖桿模糊變數只有兩個鐘形曲線的歸屬函數，如圖 3-4-2 所示。"Low"的歸屬函數包含了低於始發器動作半徑 30%以內的範圍；而"High"的函數則涵蓋了高於始發器動作半徑 60%以上的範圍。

(1) 主要控制頻率

主要控制頻率模糊變數有三個歸屬函數，如圖 3-4-2 所示。"Nominal"的歸屬函數所使用的是中心點位於 0 Hz 的鐘形曲線，代表著穩態平衡的飛行或較為和緩的操作；"APC Range"函數包含 0.3 至 1.3Hz，代表著飛機會發生耦合振盪現象的頻率；而"Over-Controlling"函數代表超越飛行控制系統的頻率之高頻運動。

(2) 主要搖桿振幅

主要搖桿模糊變數只有兩個鐘形曲線的歸屬函數，如圖 3-4-3 所示。"Low"的歸屬函數包含了低於始發器動作半徑 30%以內的範圍；而"High"的函數則涵蓋了高於始發器動作半徑 60%以上的範圍。

(3) 俯仰方向角度

θ如前面所提到的代表著俯仰方向的尤拉角，而θ角度模糊變數有兩個歸屬函數，如圖 3-4-4 所示，”Low”函數代表著10° 以下、”High”函數代表著 20° 以上的範圍。

(4) 主要控制頻率的餘弦相位延遲

餘弦相位延遲模糊變數包含了三個由-1 至+1 的鐘形曲線的歸屬函數，如圖 3-4-5 所示。”Same Phase”函數包含了+1 附近的值，代表著飛機的狀態和始發器小訊號幾乎同步；”Lag90”函數包含了 0 附近的值，代表著在一般速率控制情況下俯仰方向的相位落後(或領先)始發器小訊號90°；而” Lag180” 包含了 -1 附近的值，代表著駕駛員與飛機間有著180° 的相位差。

(5) 致動器位置

致動器位置模糊變數有兩個歸屬函數，如圖 3-4-6 所示，”Position-Saturated” 函數代表著達到了致動器的位置限制，”Centered”函數則代表其他尚未達到飽和的情形。

(6) 致動器速率

致動器速率模糊變數有三個歸屬函數，如圖 3-4-7 所示。其中所謂的飽和並沒有明確的定義，在此定義標準化的致動器當其速率每秒超過始動器活動半徑的 40%稱之為速率飽和，即”Speed-Saturated”函數；梯形的”Nominal”函數定義在每秒 5%與 40%活動半徑之間；”Zero”函數則涵蓋了接近零的致動器速率。

(7) 致動器加速度

致動器加速度模糊變數有兩個歸屬函數，如圖 3-4-8 所示，”Zero”函數代表著致動器的作等速運動，而”Non-Zero”函數則代表其他所情形。

(8) PIO 估測

PIO 估測是一個模糊輸出變數，且包含了兩個鐘形曲線的歸屬函數，如圖 3-4-9 所示。”Low”代表這並不是一個 PIO 事件；而”High”代表是一個 PIO 事件。它們被明顯地以 0 與 1 的權值區分，目的是為了強調 PIO 與 Non-PIO 事件的不同。

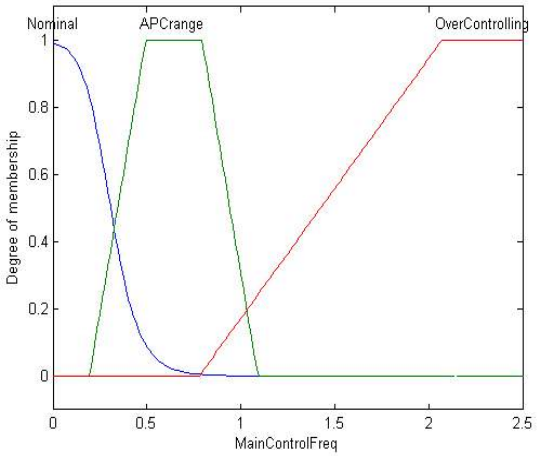


圖 3-4-2　主要控制頻率的歸屬函數

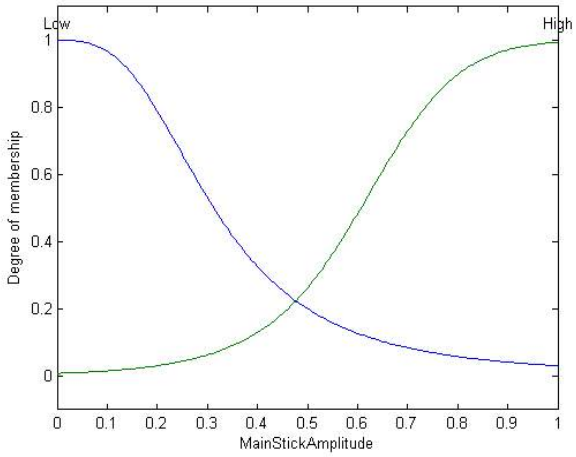


圖 3-4-3　主要搖桿振幅的歸屬函數

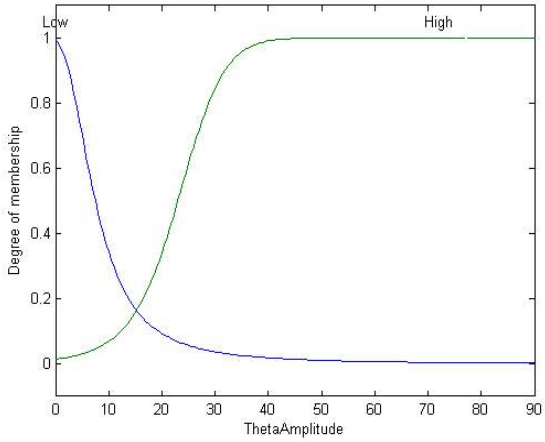


圖 3-4-4　俯仰方向角度的歸屬函數

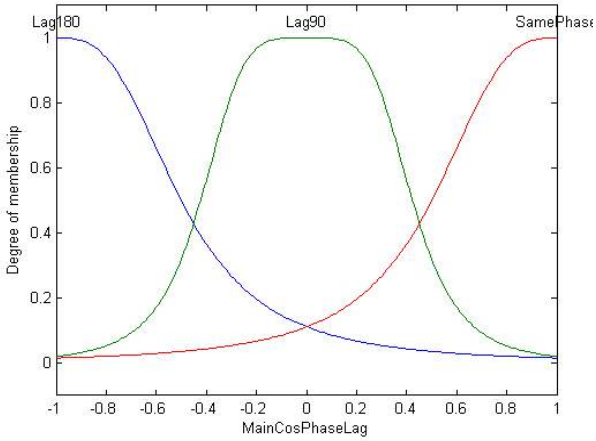


圖 3-4-5　餘弦相位延遲的歸屬函數

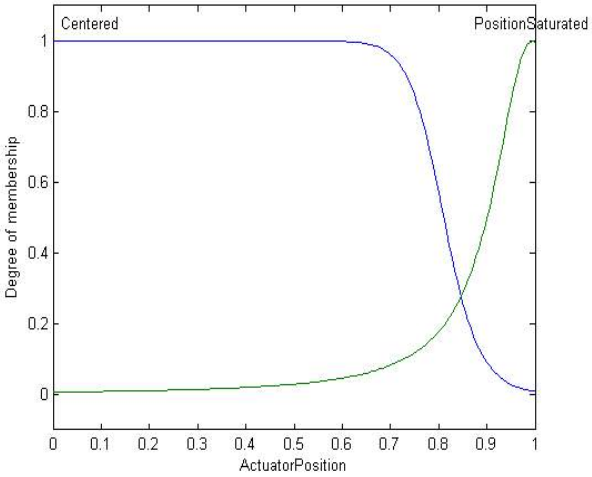


圖 3-4-6　致動器位置的歸屬函數

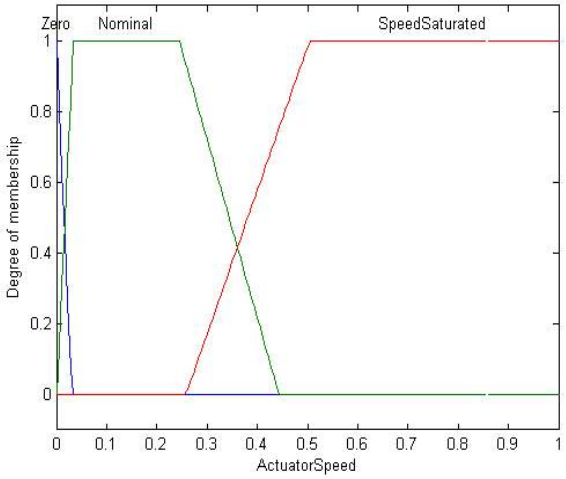


圖 3-4-7　致動器速率的歸屬函數

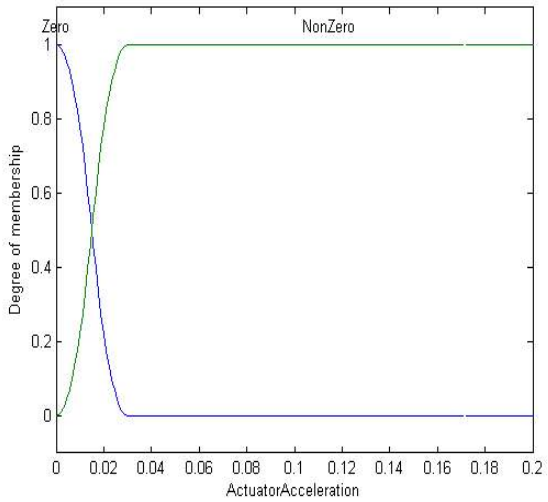


圖 3-4-8　致動器加速度的歸屬函數

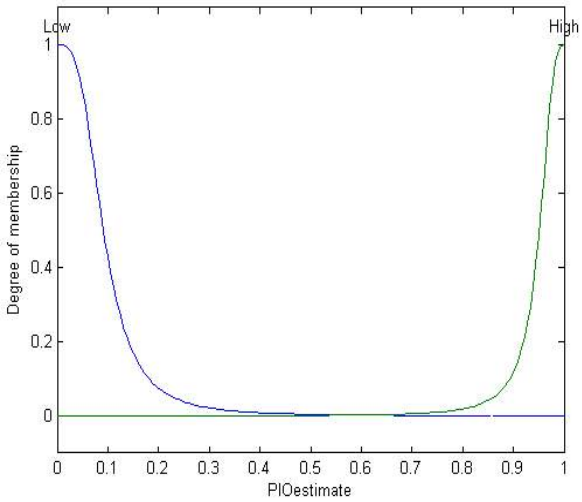


圖 3-4-9　PIO 估測的歸屬函數

3.4.4 模糊法則集合

在這裡所輸出的訊號為 PIO 估測的值，用來估測目前的飛機的飛行狀態是否是一個 PIO 事件，它並不是只有「PIO」與「非 PIO」兩種結果，而是利用連續且由 0 (代表沒有 PIO)至 1 (代表很嚴重的 PIO)的數字表示。這裡的模糊法則集合利用正法則鑑別 PIO 的特徵，而用負法則鑑別一般飛行狀態的特徵。正法則增加了 PIO 估測的值，負法則則區別出普通但是有可能會產生振盪的飛行狀態，並減少 PIO 估測的值。除了相位延遲法則予以兩倍權值外，其他模糊法則都有相同的權值。本子題所提出之模糊 PIO 偵測器所用到的模糊法則主要有下列幾種：

(1) 相位延遲法則

	Fuzzy Variable		Membership Function
IF	MainControlFreq	is	APC Range
AND	MainStickAmp	is	High
AND	MainCosPhaseLag	is	Lag180
THEN	PIO Rating	is	High

相位延遲法則是一個正法則，且滿足於當駕駛員與飛機有著接近180° 的相位差時。因為它被視為是和 PIO 事件最有關係的指標，因此賦予它兩倍的權值。

(2) 額定頻率法則

	Fuzzy Variable		Membership Function
IF	MainControlFreq	is	Nominal
THEN	PIO Rating	is	Low

額定頻率法則是一個負法則，用來定義穩定的飛行狀態。在此飛行狀態下，遞迴傅立葉轉換缺少了能提供高頻相位延遲估測的高頻成分，而低頻的相位延遲估測仍正確。

(3) 同步相位法則

	Fuzzy Variable		Membership Function
IF	MainCosPhaseLag	is	Same Phase
THEN	PIO Rating	is	Low

同步相位法則是一個負法則，用來定義正常的飛行狀態，而不管飛機是處於何種極端的操作之下。此法則是用來抵銷相位延遲法則的。

(4) 致動器飽和法則

	Fuzzy Variable		Membership Function
IF	ActuatorSpeed	is	Speed Saturated
AND	ActuatorAcceleration	is	Zero
THEN	PIO Rating	is	High

致動器飽和法則是一個正法則，適用於當致動器的速率很高且加速度為零時。它

同時也間接指出了速率飽和的狀態，而不需再另外去定義一個飽和的速率值。

(5) 額定致動器法則

	Fuzzy Variable		Membership Function
IF	ActuatorSpeed	is	NOT Speed Saturated
OR	ActuatorAcceleration	is	Non-Zero
THEN	PIO Rating	is	Low

額定致動器法則是一個負法則，用來抵銷致動器飽和法則。它解釋了致動器在某些等速率狀態下的原因亦有可能是駕駛員故意作這樣的飛行控制所導致。

3.4.5 相位補償機制

飛機的液壓操縱面伺服馬達通常都有一定的速率限制，而為了不讓伺服馬達有速率飽和的情形，通常在飛行控制系統的軟體中會有速率限制器存在於伺服馬達的輸入命令端之前。以正弦波形輸入為例，當速率控制器達到飽和狀態時，輸出與輸入間會有振幅與相位的平移發生，且相位的平移情況十分明顯。通常若飛機的飛行控制系統無法再提供45°的相位邊限(Phase Margin)，我們稱它作不穩定的飛行狀態，而一個速率控制器可以很輕易地給予這樣的相位平移。這種相位平移的現象亦會影響到駕駛員的開迴路造成一個相對的時間延遲，這種額外的時間延遲可以用來解釋 PIO 的發生原因，比較像是駕駛員用一個很大的而且快速的搖桿輸入命令所造成的。

有兩種簡單的方法可以用來預防速率限制的發生：(1)降低搖桿的增益；(2)於搖桿輸入訊號後增加低通濾波器。但是這以上所提出之兩種方法不但都會使飛機的增益降到很低，讓人有駕駛 JAS 39 Gripen 好像在駕駛波音 777 的感覺一樣，也會飛機因為加入過多的相位延遲而變得難以控制。就算搖桿的輸入很微小，亦會有相位延遲的情況產生，延遲濾波器只在搖桿有很大的而且快速的輸入命令時才會有保護的作用。

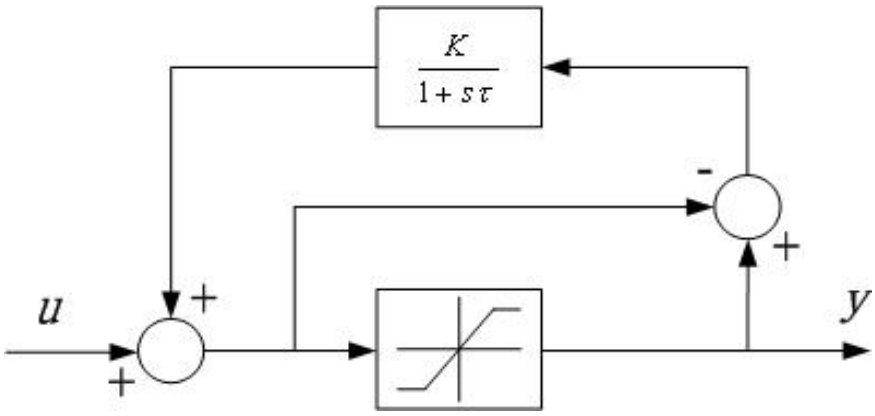


圖 3-4-10　具迴授的速率限制器

取而代之的是一個迴授裝置，如圖 3-4-10 所示。當輸入訊號 u 的速率大於速

率限制器 r 時，迴授的訊號將會賦予負號且減少輸入至速率限制器的訊號，藉此獲得較少的相位平移。當輸入訊號 u 的速率小於 r 時，迴授的訊號將會衰退至零。相較於接下來要介紹的完整的濾波器，這種低通濾波器的相位補償比較少，但是相對地它也比較不會造成偏差。

如圖 3-4-10 所示的迴圈對於表現高頻訊號來說並不是一個良好的解決方法，理由是因為輸入訊號 u 的高頻成分幾乎會阻礙了低頻成分的迴圈動作。對於高頻訊號來說，此迴圈不需要也不能夠提供相位補償，因此相位延遲的問題只有在輸入訊號 u 的低頻成分獲得相位補償後才能解決。另外一種完整的濾波器如圖 3-4-11 所示，輸入訊號 u 的低頻成分被第一個速率限制器作限制(有相位迴授)，且只有輸入訊號 u 的高頻成分被第二個速率限制器作限制(無相位迴授)，而此兩個速率限制器擁有一樣的速率限制，因此低頻與高頻的部分同時獲得補償。在搖桿命令達到致動器的速率限制而產生相位延遲的現象時，利用此完整的濾波器與之前的模糊 PIO 偵測器相結合，可有效地補償相位，減少 PIO 發生的傾向，並可藉由模糊 PIO 偵測器觀察出此結果，達到安全飛行的目的。

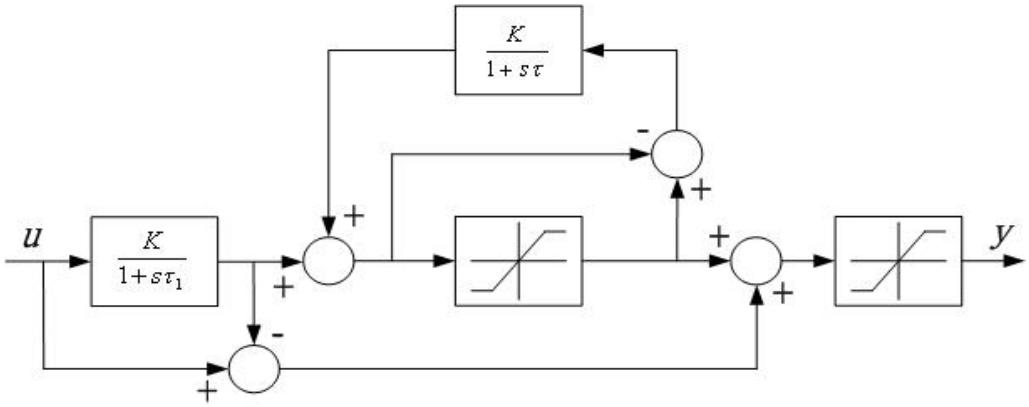


圖 3-4-11　具迴授與旁通迴路的速率限制器

3.4.6 對抗 PIO 的方法

現代飛機的架構與控制系統日新月異，雖飛機上電腦控制系統控制大多數控制面，對駕駛員依舊是一大挑戰，由於駕駛員操作飛機的整體系統非常複雜，使得 PIO 現象很難被準確地預測，目前已有專家學者發表一些預測或評估 PIO 的方法來避免造成飛機的墜毀與駕駛員的傷亡，這幾年來比較有代表性的研究列舉描述如下

(1) Neal-Smith Criteria 【3-8】
(2) Open Loop Onset Point (OLOP) 【3-9】
(3) 功率頻譜密度和 Spectrogram 分析【3-10】
(4) Handling Quality During Tracking (HQDT) 【3-11, 3-12】
(5) Limit Cycle PIO 分析【3-13】
(6) Neural Detector 【3-14~3-16】

3.5 模擬

此節為利用本子題所設計之模糊 PIO 偵測器的模擬結果。若只考慮整個飛行系統的主要部分，則搖桿、虛擬實境飛行場景與模糊 PIO 偵測器的互動情形如圖 3-5-1 所示。搖桿傳達命令至虛擬場景端，而虛擬場景也產生了力回饋訊號回傳至搖桿，並將搖桿的命令訊號δ_C以及飛機的俯仰角θ傳至模糊 PIO 偵測器，作 PIO 傾向的評估。

圖 3-5-2、圖 3-5-3 與圖 3-5-4 顯示此模糊 PIO 偵測器的模擬結果，由上而下依序是(a)搖桿輸入與致動器限制、(b)俯仰角θ、(c)相位延遲(取餘弦)、(d)主要控制頻率、和(e)PIO 估測值。由圖 3-5-2 可看此飛行過程出剛開始有一段故意誘發 PIO 現象的操作，接著是一段平穩的控制，再來又是一段故意的操作，最後是失控的狀態，而由圖上可清楚的看出 PIO 估測值的變化情形。

而在此時若以加入如之前圖 3-4-14 所示的具迴授的速率限制器，即可發現經由相位補償後，PIO 評估值確實有改善的情形，如圖 3-5-3 所示，這代表著飛機比起沒有補償時更不容易發生 PIO。另外若加入先前圖 3-4-15 所示的具迴授與旁通迴路的速率限制器，則此相位補償機制【3-17】將更能有效地降低 PIO 估測值，如圖 3-5-4 所示，其 PIO 估測值均較圖 3-5-3 為低。經由上述之模擬結果我們可以證實相位補償機制確實能減少 PIO 現象的發生，且具有迴授與旁通迴路的速率限制器能有較佳的相位補償效果。

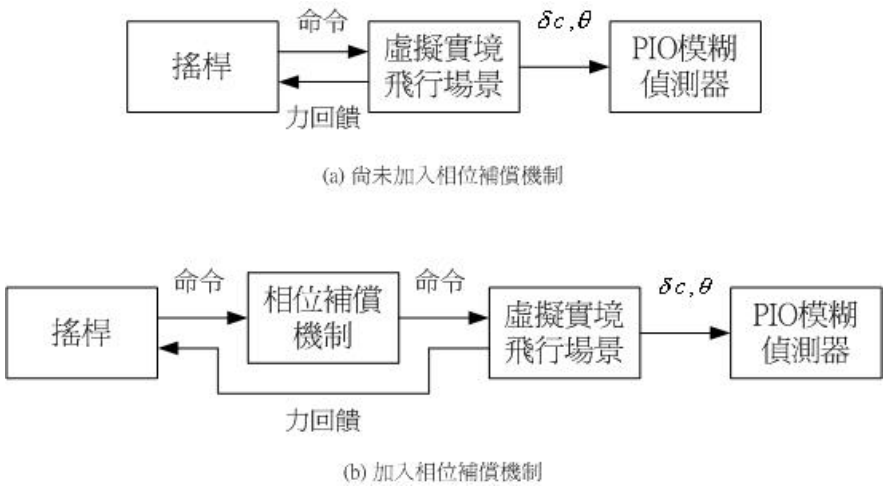


圖 3-5-1　搖桿、虛擬場景與 PIO 偵測器的互動情形

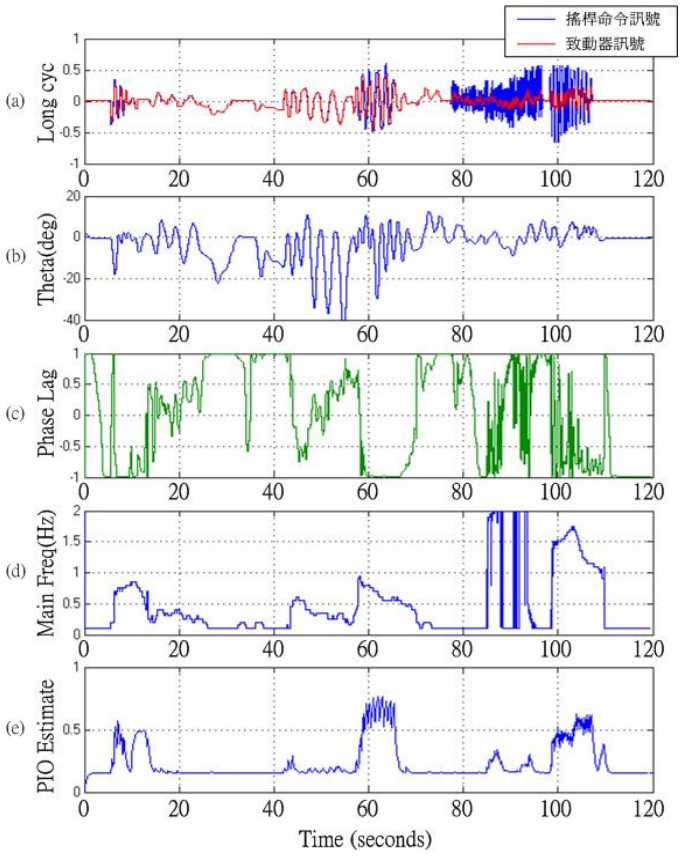


圖 3-5-2　發生 PIO 事件的飛行狀態之 PIO 估測

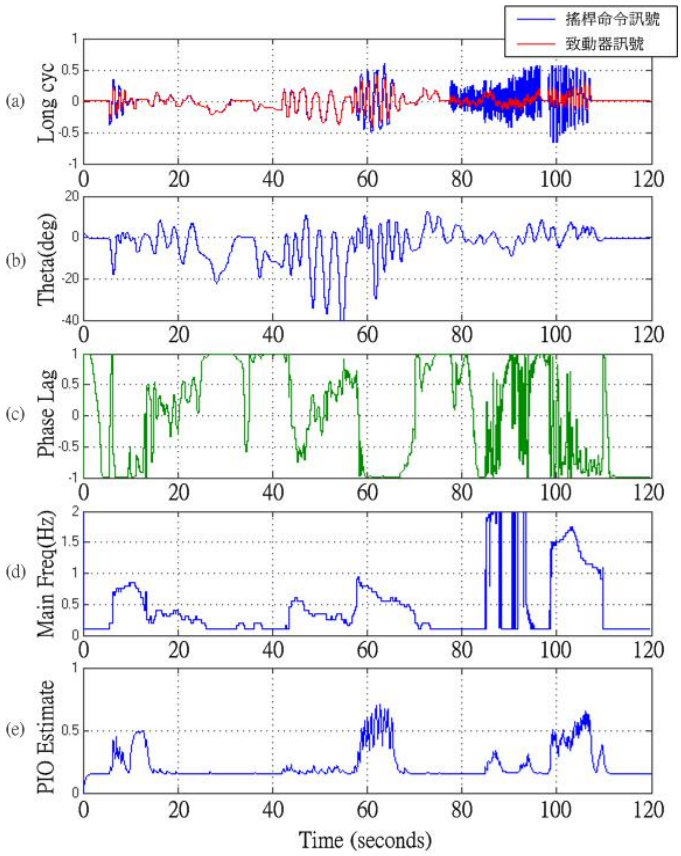


圖 3-5-3　經由迴授的相位補償後所得到的 PIO 估測值

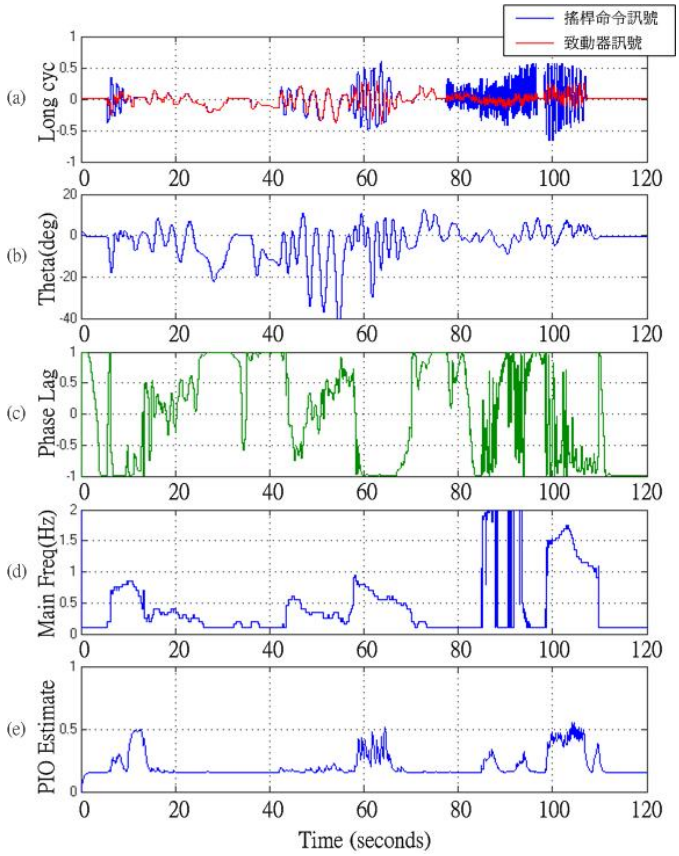


圖 3-5-4　經由迴授與旁通迴路的相位補償後所得到的 PIO 估測值

子題四：六軸運動平台之建立、分析與控制

在本計劃中由於在資料傳輸上受限於 RS-232 串列傳輸，同時以 LART 實驗單板的 Intel StrongARM SA-1100 處理器之 Linux 嵌入式系統來取代舊有的 PC 控制方式，這不僅使控制系統體積大幅減少，而嵌入式系統週邊亦支援 TCP/IP 網路，可是卻沒有 CAN bus 的控制裝置與驅動程式可提供虛擬實境場景的六軸腳長之資料與 LART 實驗單板作為控制資料的連結與傳輸，所以必須另外增加 LART 實驗單板的 interface 與 CAN bus 裝置。並且亦需要建立在 LART 實驗單板上的 CAN bus 驅動程式。然而，在 LART 實驗單板上亦是沒有 A/D 及 D/A 控制器，所以也必須增加 LART 實驗單板的 A/D 及 D/A interface 與 A/D 及 D/A 的控制裝置，並且建立 LART 實驗單板上的 A/D 及 D/A 驅動程式，以達到六軸平台之控制之目的。

另外，在各種模擬系統中為了擁有更真實的模擬效果，時常需要處理大量的外界訊息，當系統呈現高度負載的情況時，為了依舊保持模擬的完整性，其中有些重要的訊息必須是不能忽略或是要優先處理的。其次，和實際的器具一樣，系統的穩定性、可靠性與安全性都是一個模擬系統所考量的。然而，為了提高六軸運動平台的穩定度與安全性，控制系統對外界訊息的反應必須要更為迅速，以確保在系統出現問題時能立即做出適當的處理。因此，在六軸運動平台控制的改善上，為了讓系統呈現出更完善的模擬效果，於本年度的計劃中，我們在既有的控制系統中，加入一個即時作業系統，並分析與設計整個即時控制環境的架構以及實現了整個六軸平台即時控制系統。

4.1 六軸平台控制卡之改良研發

LART 實驗單板的 Intel StrongARM SA-1100 處理器之 Linux 嵌入式系統來取代舊有的 PC 控制方式，這不僅使控制系統體積大幅減少，也使我們能在 Linux 作業系統上發展新的控制程式，並且可以達到多平台及同步接收資料。而嵌入式系統週邊亦支援 TCP/IP 網路。然而在 LART 實驗單板上並沒有 CAN bus 的控制裝置與驅動程式可提供虛擬實境場景的六軸腳長之資料與 LART 實驗單板作為控制資料的連結與傳輸，所以必須另外增加 LART 實驗單板的 interface 與 CAN bus 裝置。

4.1.1 CAN Bus 裝置之製作

在嵌入式硬體部分，我們所選擇採用的是 LART 實驗單板，選擇這塊板子的主要原因是因為它選擇這塊板子的原因是因為它有豐富的序列傳輸介面，包括 IrDA、RS-232，也有內建 10Base-T 網路，並且支援包括 Linux 嵌入式作業系統，可以發展的嵌入式種類及相關應用程式可謂相當豐富，將十分適合本子題研究之用，整塊單板實體圖片如圖 4-1-1 所示。

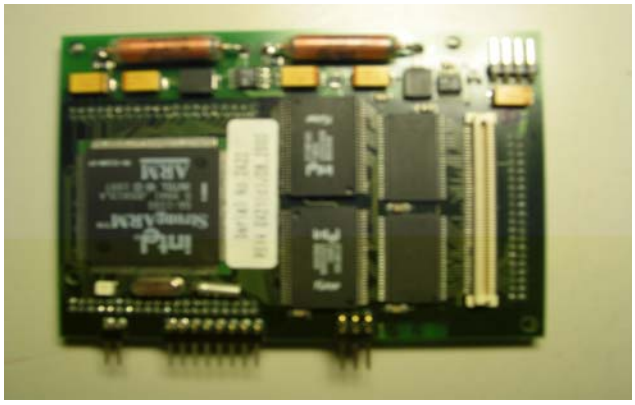


圖 4-1-1　LART 實驗板

除此之外，所使用的微處理器為 SA-1100，而在 LART 的計畫中亦是使用 Strong ARM 微處理器系列。所謂 LART 計畫是由國外的 Delft University of Technology 所主持的研究計畫，主要研究在消耗不到一瓦特功率而可以達到 250MIPS 指令的 Linux 嵌入式系統。它有一套較為完整的 Linux 嵌入式系統文件及 mailing list，更難得的是它亦將所有的軟硬體公開。因此在未來發展 StrongARM SA-1100 嵌入式系統時可以有較為完整的相關文件可以參考。

在 LART 實驗單板上發展嵌入式系統來控制虛擬實境動態模擬器，並沒有 CAN-Bus 的裝置與驅動程式，因此我們必需要發展 CAN-Bus 的裝置與驅動程式，來作為與 LART 實驗單板的傳輸介面。如此才可以與 LART 實驗單板溝通，接收由虛擬實境場景的六軸腳長 data，然後將腳長轉成電壓來控制動態模擬器，其系統方塊圖如圖 4-1-2 所示。

在動態模擬器六軸姿態的傳遞方式上，我們將發展 CAN-Bus 通訊協定，因此

必須在嵌入式系統上發展 CAN-Bus 驅動程式，A/D 及 D/A 驅動程式；除此之外，還需發展 LART 實驗單板之 CAN-Bus 程式，如此才能以 CAN-Bus 傳遞六軸控制姿態至動態模擬器驅動控制盒上。

LART 實驗單板透過 CAN bus 裝置接收來自 VR 場景的六軸姿態，然後經過控制程式，如奇異點迴避及逆向運動學，然後再將六軸腳長傳遞給 D/A 控制卡，來驅動控制動態模擬器。

因為 LART 實驗單板是一個以 StrongAMR SA-1100 微處理器為核心的嵌入式硬體，其發展環境與一般 X86 系統相異，必須先建立 Linux 作業平台（Linux 作業系統版本必須與移植至嵌入式系統相同），並建立 Cross Compiler 環境（在 X86 系統下發展嵌入式系統的環境），如此才能使用 GNU C library 及利用 arm-linux-gcc 來編譯程式及發展驅動程式。值得注意的是 LART 實驗單板 StrongARM SA-1100 微處理器是一顆 RISC 晶片，它以 Memory Mapping 的方式來規劃其暫存器，因此必須在作業系統內規劃好實際記憶體與虛擬記憶體之映射，如此才能設定相關暫存器。圖 4-1-3 為 CAN bus 實際之控制電路板。

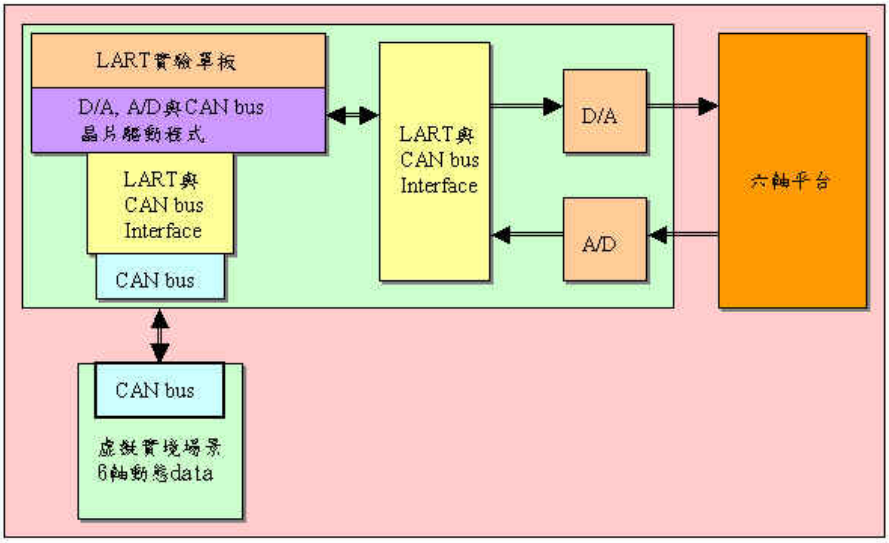


圖 4-1-2　虛擬實境動態模擬器之 CAN Bus 系統控制架構

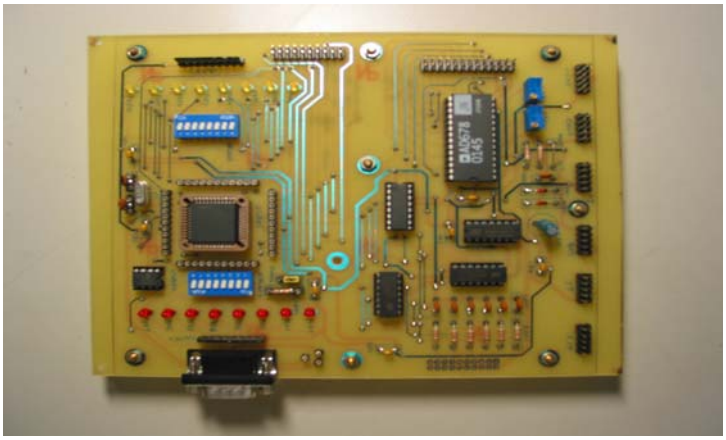


圖 4-1-3　CAN bus 之控制電路板

4.1.2 CAN Bus 原理

CAN（Control Area Network）最早是在 1986 由德國 Robert Bosch 公司依據 Mercedes 與 BMW 的要求所發展出來的傳輸架構，主要的目的是為了應付日漸複雜的汽車控制傳輸系統。與傳統 UART（Universal Asynchronous Receiver/Transmitter）傳輸方式最大的不同在於 UART 只適合用於點對點資料傳輸，而 CAN 則具有單點對多點的傳輸能力。因此在本章節我們將依序介紹 CAN-Bus 基本觀念、資料傳輸框架、訊息過濾與訊息確認、錯誤偵測、字元傳輸時間。

CAN-Bus，是一種支援分散式即時控制的串列傳輸系統。它具有高安全性的序列傳輸通訊協定，傳輸速度最快可達 1 Mbits/s。根據 ISO/OSI 參考模型，CAN-Bus 可以細分為幾個不同層：

Data Link Layer：

Logical Link Control (LLC) Layer：提供資料傳輸和遠端資料要求的服務，並且決定接收哪一個訊號。

Medium Access Control (MAC) Layer：實現傳輸通訊協定，包括控制傳輸資料的架構、執行仲裁、錯誤檢查、錯誤處理及錯誤的限制。

Physical Layer：

定義如何將每個位元逐一送出，並且處理傳輸同步、字元編碼及字元傳輸時間等事件。

4.1.2.1 基本名詞

Bit Rate：CAN-Bus 資料傳輸的速度，最快可達 1 Mbits/s。

Priorities：使用 Bus 的優先順序，由 Identifier 定義。

Bus Off Status：當有不正常的錯誤發生，而次數超過 256 次時，CAN-Bus 會停止一切傳送與接收的動作。

Message Identifier：代表某一訊息的號碼。

Global Mask：又稱為訊息接收過濾器，經過與訊息的 Identifier 比較後，用以決定是否接收某一訊息。

Multimaster：在 CAN-Bus 的網路上，每個單元都有傳送資料的權利。

Multicast：將資料同時傳送給網路上的每一個單元。

Acknowledge：接收端正確接收到資料後，會送出此訊息。

Bus Values：包含兩個互補的邏輯—‘dominant’和’recessive’。當網路上其中一個單元送出’dominant’，而另一個單元同時送出’recessive’時，Bus Value 定義為 ’dominant’。

Remote Data Request：利用送出 REMOTE FRAME，可要求遠端的單元傳輸資料 DATA FRAME。DATA FRAME 和 REMOTE FRAME 具有相同的 Identifier。

Arbitration：當網路上同時有兩個以上的單元要求使用 Bus 時，由 Arbitration 來仲裁哪個單元具有較高的 Bus 使用權。仲裁的方式是比較各個單元的 IDENTIFIER，當其中一個單元的 Identifier 送出的位元是'dominant'，而其他送出的位元是'recessive'時，則送出'dominant'的單元將取得 Bus 的使用權。另外，具有相同 Identifier 的 Data Frame 和 Remote Frame 同時被送出時，DATA FRAME 具有較高的優先順序。

4.1.2.2 資料傳輸框架

Data Frame 主要是經由七個部份所所主成，如圖 4-1-4 與 4-1-5 所示，分別是：

(1) Start Bit
(2) Arbitration Field
(3) Control (identifier) Field
(4) Data Field
(5) CRC Field
(6) Acknowledge Field
(7) End of Frame

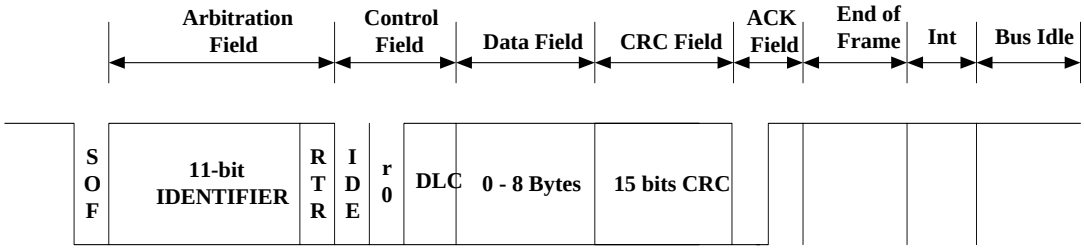


圖 4-1-4　Data Frame－Standard Frames

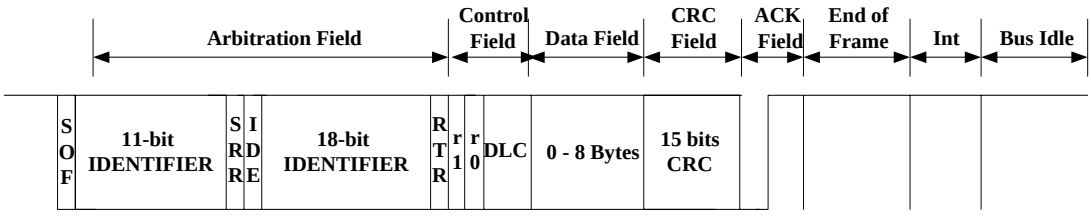


圖 4-5　Data Frame－Extended Frames

其每一個 field 的意義分別介紹如下：

Start of Frame：代表每一個 Data / Remote Frame 的開始。

Arbitration Field：可區分為 Standard Format 和 Extended Format 兩種，這兩種 Format 主要差異在於 Identifier 的長度。Standard Format 中 Identifier 的長度為 11 個位元，Extended Format 中 Identifier 的長度則為 29 個位元。

RTR BIT：Remote Transmission Request Bit，DATE FRAME 中此位元設定為'dominant'，而在 REMOTE FRAME 中此位元設定為'recessive'。

SRR BIT：Substitute Remote Request Bit，為'recessive'位元。

IDE BIT：IDentifier Extended Bit，在 Extended Format 時是'recessive'，屬於 Arbitration Field；在 Standard Format 時是'dominant'，屬於 Control Field。

Control Field：在 Standard Format 和 Extended Format 具有不同的組成。在 Standard Format 中，由 IDE、r0 和 DLC 組成；在 Extended Format 中，由 r0、r1 和 DLC 組成。（r0、r1：保留位元。）

DLC：Data Length Code，由四個位元構成，指出 Data Field 的長度。

Data Field：存放傳輸的資料，一次最多可送 8 個位元組，由每個位元組的 MSB 先傳送。而 Remote Frame 則傳送 0 個位元組。

CRC Field：由 CRC Sequence 和 CRC Delimiter 構成。CRC Sequence 儲存 Cyclic Redundancy Code 計算的結果，用以檢查接收的資料是否正確無誤。CRC Delimiter 為一個'recessive'位元。

ACK Field：包含兩個位元，ACK Slot 和 ACK Delimiter。傳送端送出的 ACK Field 為兩個'recessive'位元，而接收端在收到正確的資料後，在傳送端送出 ACK Slot 的同時送出'dominant'，用以通知傳送端資料已正確地傳送至接收端。

Remote Frame 主要是經由六個部份所所主成，分別是(1) Start Bit, (2) Arbitration Field, (3) Control (identifier) Field, (4) Data Field, (5) CRC Field, (6) Acknowledge Field。RECEIVER 利用送出 REMOTE FRAME，通知 TRANSMITER 傳送相關的資料。REMOTE FRAME 與 DATA FRAME 最大不同點，在於 REMOTE FRAME 不包含 DATA Field，且 Arbitration Field 中的 RTR BIT 為'recessive'。其架構如圖 4-1-6 所示。

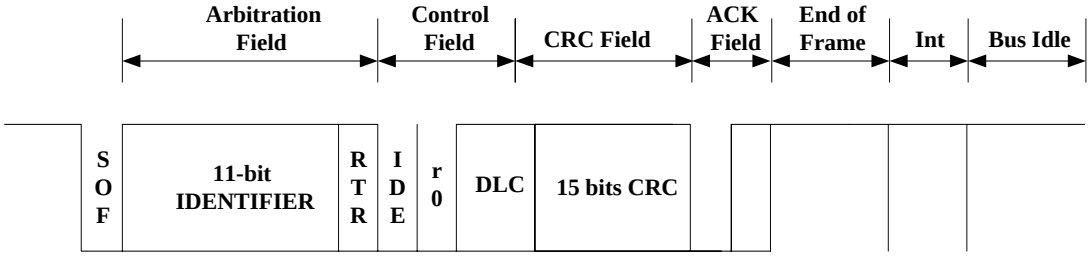


圖 4-1-6　Remote Frame

4.1.2.3 訊息過濾與訊息確認

利用 Mask Registers，使用者可以定義 Identifier 的任意一為'Don't Care'。由於 RECEIVER 在接收資料時，不會去比較 Identifier 中'Don't Care'的位元，利用這個特性，CAN-Bus 可以將一筆資料送給一整個群體，也就是 3.2.2 中提到的 Multicast 的觀念。TRANSMITER 和 RECEIVER 對於一筆資料的正確與否，確認方式不同。其差異點如下：

(1) TRANSMITER：當正確傳送出 END OF FRAME 的最後一個位元後，TRANSMITER 即便確認此次的傳送無誤。而當傳送發生錯誤時，CAN-Bus

的硬體架構會自動地進行再次傳送。

(2) RECEIVER：當正確接收到 END OF FRAME 的第六個位元(倒數第二個位元)後，代表此次接收的資料正確無誤。而 END OF FRAME 的第七個位元則視為'Don't Care'，也就是說，當 END OF FRAME 的第七個位元為'dominant'時，RECEIVER 亦不會將它視為接收錯誤。

在 TRANSMITER 和 RECEIVER 對訊息確認的方式不同的情況下，會有一種錯誤的狀況發生：當 TRANSMITER 在傳送 END OF FRAME 的最後一個位元發生錯誤時，TRANSMITER 會視此次的傳送失敗，自動地再次傳送。在 RECEIVER 端，因為只檢查到 END OF FRAME 的倒數第二個位元，所以不會發現錯誤，而 TRANSMITER 的再次傳送，RECEIVER 會再次接收，造成同一筆資料接收兩次的狀況發生。

4.1.2.4 錯誤偵測

CAN-Bus 定義五種錯誤型態：

BIT ERROR：當 TRANSMITER 送出位元的'Bit Value'與 Bus 上所偵測到的邏輯不同時，即發生 BIT ERROR。不過有個例外的狀況，就是在 ARBITRATION FIELD 和 ACK SLOT 時，若送出的'Bit Value'為'recessive'，但 Bus 上偵測到的邏輯為'dominant'時，不會視為 BIT ERROR。

STUFF ERROR：當 RECEIVER 在 START OF FRAME、ARBITRATION FIELD、CONTROL FIELD、DATA FIELD 和 CRC SEQUENCE 這些區塊，偵測到連續六個相同的'Bit Value'時，代表 TRANSMITER 編碼發生錯誤，發生 STUFF ERROR。

CRC ERROR：RECEIVER 收到的資料，經過 Cyclic Redundancy Code 的計算後，與 CRC SEQUENCE 的值不同時，發生 CRC ERROR。

FORM ERROR：不論是在 DATA FRAME 或是 REMOTE FRAME 中，START OF FRAME、CRC DELIMITER、ACK DELIMITER 和 END OF FRAME 等，皆為固定的'Bit Value'所組成，因此在 RECEIVER 接收到的資料中，這些固定的類型有錯時，發生 FORM ERROR。

ACKNOWLEDGE ERROR：TRANSMITER 偵測出在傳送 ACK SLOT 時，Bus 的'Bit Value'不是'dominant'，代表 RECEIVER 沒有正確地傳回 ACKNOWLEDGEMENT，發生 ACKNOWLEDGE ERROR。

4.1.2.5 錯誤限制

CAN-Bus 定義每個單元具有兩個 COUNT：TRANSMIT ERROR COUNT 和 RECEIVE ERROR COUNT，依照所發生錯誤狀況的不同，累積不同值的 COUNT。當其中一個 COUNT 值超過 256 時，該單元則進入'bus off'的狀態；COUNT 值介於 128~256 間，則該單元進入'error active'的狀態；COUNT 值小於 128 時，該單位屬於'error passive'的狀態。

4.1.2.6 字元傳輸時間

一個 NOMINAL BIT TIME 可區分為四個區塊，如圖 4-1-7 所示，包括 SYNCHRONIZATION SEGMENT (SYNC_SEG)、PROPAGRTION TIME SEGMENT (PROP_SEG)、PHASE BUFFER SEGMENT1 (PHASE_SEG1)、PHASE BUFFER SEGMENT2 (PHASE_SEG2)，其中各功能說明如下：

SYNC_SEG：這個區塊用以處理網路上各個單元之間資料傳輸的同步。
PROP_SEG：這個區塊用以補償物理的延遲時間(physical delay time)。
PHASE_SEG1、PHASE_SEG2：用來補償 edge phase 的誤差。
SAMPLE POINT：在此點定義這個 NOMINAL BIT TIME 所傳輸位元的值。

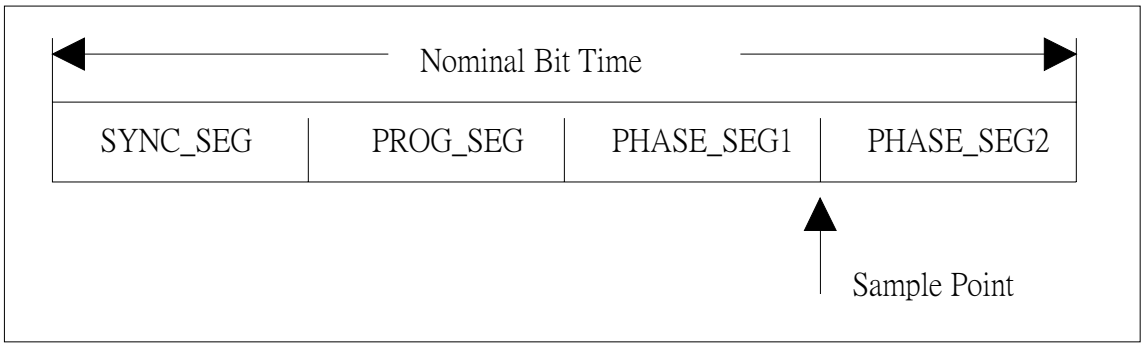


圖 4-1-7　字元傳輸時間

4.1.3 LART 實驗單板與 CAN Bus 間的 interface 電路設計

在 CAN bus 控制晶片中，目前以採用 Intel 公司所出的 i82527 CAN bus 控制晶片與 MicroChip 公司所出的 MCP2551 CAN Transceiver，結合成 CAN bus 控制裝置，因此須要有一 interface 作為 LART 實驗單板控制 CAN bus 控制晶片。LART 實驗單板提供了 StrongAMR SA-1100 微處理器晶片 GPIO，data bus 與 address bus 信號可作為其他額外的控制裝置使用，同時我們利用 address，data bus 與 read/write 的控制信號 pin，並且以 StrongAMR SA-1100 微處理器晶片 memory mapping，mapping 至 static memory 的 bank1 位址，與 ALTERA 公司所提出的 FPGA 晶片，作為 LART 實驗單板與 CAN bus 裝置間位址的 decode，並同時產生 i82527 的控制 waveform 對 i82527 CAN bus 晶片作 read 與 write 的控制。圖 4-1-8 及圖 4-1-9 分別為 StrongAMR SA-1100 微處理器晶片 memory mapping 位址圖和 LART 實驗單板與 CAN bus 間的 interface 電路方塊圖。而實際 interface 電路板如圖 4-1-10 所示。

CAN bus 在 FPGA 的 simulation waveform 如圖 4-1-11 所示。在未來完成部份即是撰寫 CAN bus 的驅動程式與發展 LART 實驗單板之 CAN-Bus 程式，使嵌入式系統與動態模擬器控制系統整合，並可以達到多平台及同步接收資料之功能。

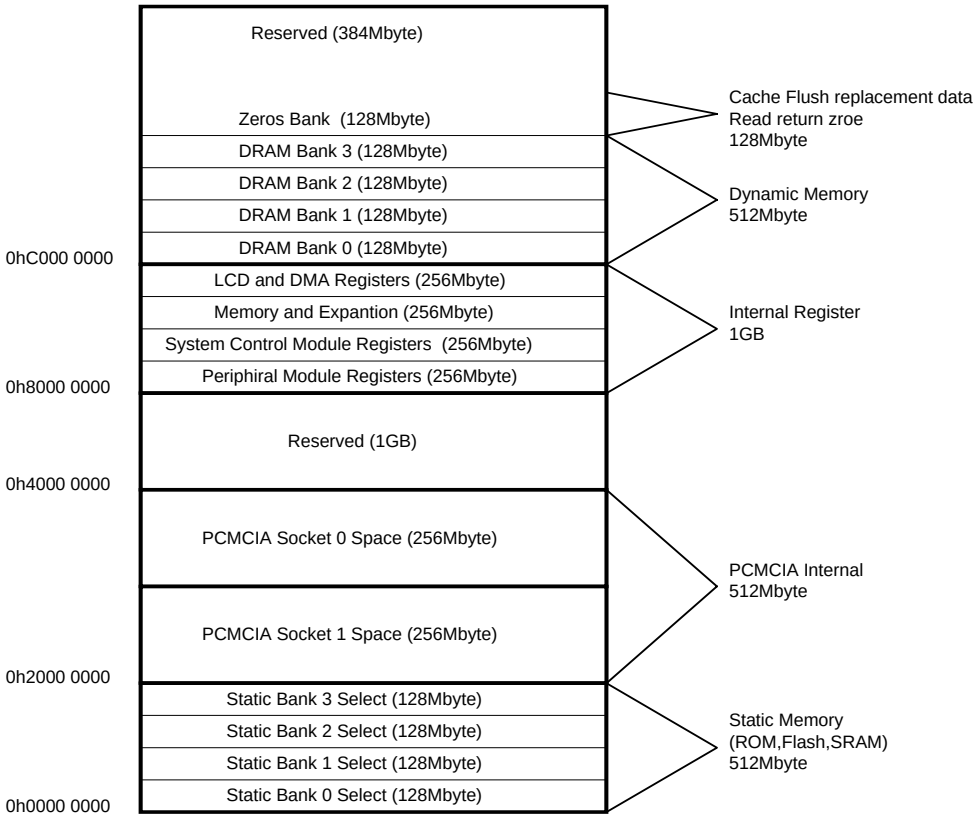


圖 4-1-8　StrongAMR SA-1100 微處理器晶片 memory mapping 位址圖

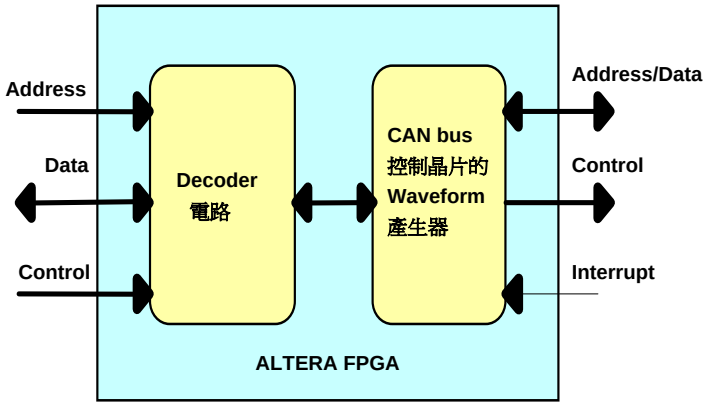


圖 4-1-9　LART 實驗單板與 CAN bus 間的 interface 電路方塊圖

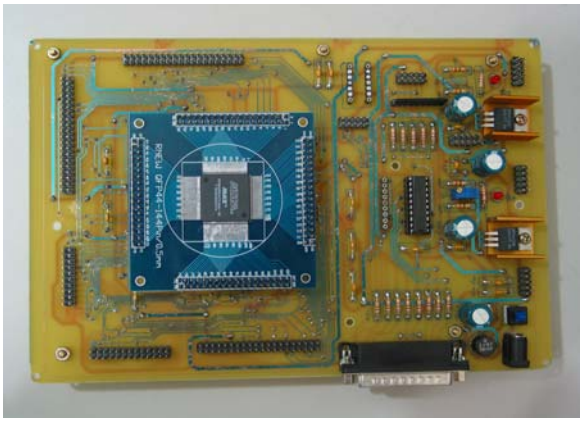


圖 4-1-10　LART 與 CAN bust 之 interface 電路

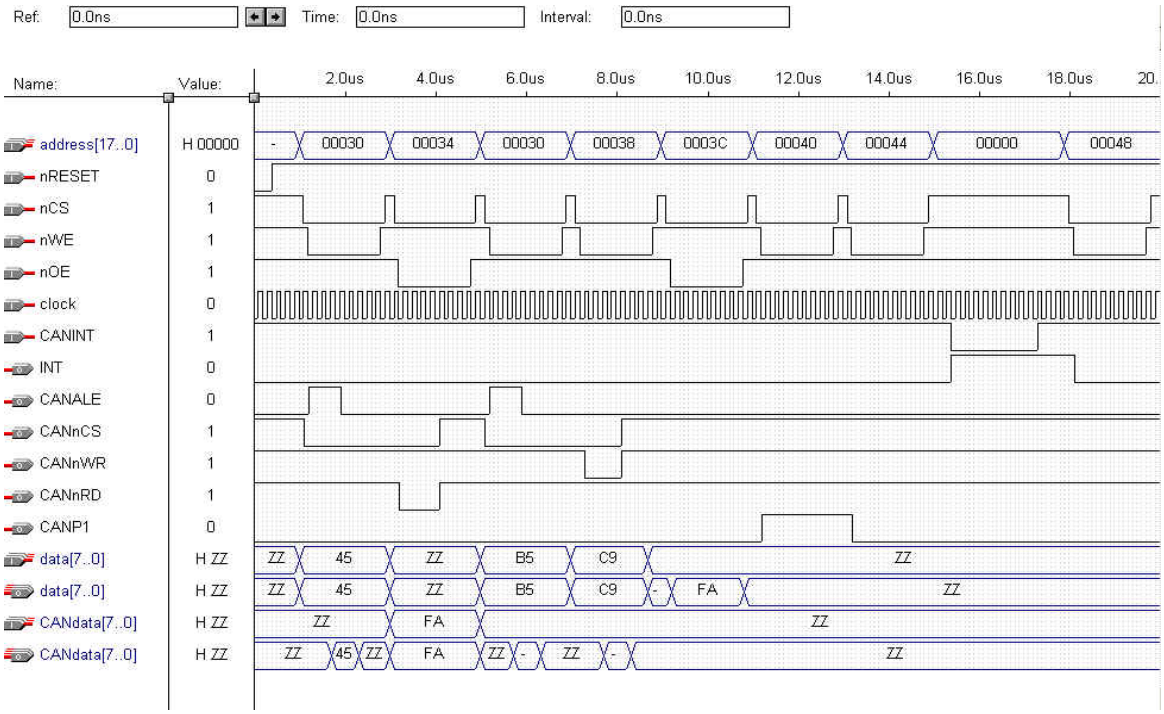


圖 4-1-11　CAN bus 之 FPGA 模擬圖

4.2 六軸平台控制卡之即時控制系統

在各種模擬系統中同時為了擁有更真實的模擬效果與為了提高六軸運動平台的穩定度與安全性，控制系統對外界訊息的反應必須要更為迅速，以確保在系統出現問題時能立即做出適當的處理。因此，在六軸運動平台控制的改善上，為了讓系統呈現出更完善的模擬效果，我們在既有的控制系統中加入一個即時作業系統，採用加強核心方式的 RTAI，來改善原有 Linux 作業系統不具即時性的問題。

4.2.1 即時作業系統

於去年的成果中，我們是以 Linux 作業系統做為控制平台，但是 Linux 本身無法做為硬即時應用，最主要的原因有【4-1】:

(1) 非強取式核心（Non-preemptable kernel）

(2) 時間共用（Time-sharing）排程策略與動態配置優先權

(3) 使用虛擬記憶體

(4) 缺乏高解析度的系統計時器

就第一點而言，Linux 核心在進行系統呼叫（System call）或是在進行核心服務的工作時，系統核心大多是處於將中斷設定為關閉的狀態，這時若進行一個長時間的系統呼叫，那系統將有一段長時間無法處理外部的中斷訊息。而在即時系統裡，當一個優先權較低的工作，對核心進行系統呼叫時，若有一個優先權較高的程式正在等待執行，系統核心必須能被強迫交出控制權給優先權高的程式使用，而目前的 Linux 核心並不具有強奪性（Preemption）的能力。

其次，Linux 是採程式之執行時間共用的排程策略，其目的是為了讓系統裡所

有程式，能公平地共用系統資源（處理器、記憶體、週邊裝置...等），致使系統有最大 CPU 使用率與最大平均產能。在程式排程上，Linux 會將 CPU 傾向（CPU-bound）及 IO 傾向（IO-bound）的工作分開來集中處理，這可能會造成即時性工作超出工作期限的情況。

Linux 在工作優先順序的配置上，則是採用動態安排方式，意思是說，我們隨時可更改程式的優先權大小，而這可能會對排程的既定性造成影響。另外，Linux 利用磁碟空間當作虛擬記憶體使用，而磁碟是屬於機械式裝置，其反應能力無法像電子式的裝置來得即時。因此，在進行虛擬記憶體置換（Swapping）時，將會花費一段未知且無法預測的時間來處理。很明顯地在這段時間內，系統將是無法預測與獲知外界的訊息。

至於最後一個無法採用 Linux 來作即時應用的主要原因，是系統計時器的時間刻度。就 Linux 而言，計時器更新速度為100*Hz*，也就是每10*ms*系統產生一個 Tick，這意謂著系統裡的週期性工作，其最小執行週期為10*ms*。因此，若欲進行一些小於10*ms*的高精確度控制或取樣動作，是無法滿足需求的。此外，系統排程器的反應能力亦和計時器有關，系統計時器的時間刻度越小，排程器越能迅速反應外界訊息並喚醒處理工作。

雖說 Linux 作業系統採用的策略與即時性作業系統的策略方式背道而馳，但 Linux 確提供了豐富的硬體支援、軟體發展工具，這是一般即時作業系統所不能及的。因此，為了利用 Linux 的資源優勢，且讓 Linux 的資源能利用在即時應用上，目前有兩種延伸方法【4-3】：

第一種方法：直接加強核心能力

第二種方法：加入一個即時核心

第一種方法是針對 Linux 核心的強奪性與排程演算做些部份修改，以減少那些不可強取的（Non-preemptible）部份程式碼所耗費的時間。一般而言，採用這種策略方式是為了最小化系統的"中斷延遲"或是即時工作的"排程延遲"，使得作業系統核心具有更好的即時反應能力。採用這種作法，可能遭遇的問題是，無法確切地提供一個延遲保證數據，除非加強型的 Linux 核心中所有可能執行路徑，都做過完整的分析與測試。基本上，這就意謂著在這種環境下，是無法滿足硬即時的要求，但卻能提供軟即時的工作環境。目前採用這類作法的即時作業系統，如：TimeSys、MontaVista 及 RedSonic。

第二種方法除了採用一個獨立的即時排程器之外，較不同於之前修改核心的方式，是在 Linux 核心與硬體間加入了一個硬體處理層，來截取與管理實際的硬體中斷，我們稱為硬體抽象層（Hardware Abstraction Layer，HAL）。系統所有的硬體中斷完全由這個抽象層所控制。若是有必要，抽象層會模擬這些中斷給上層 Linux 核心使用，一般我們稱這樣的系統為一個微核心系統（Micro-kernel system）。而即時排程器部份，會根據系統裡有即時性需求的工作，將它們安排在這個即時核心工作空間中處理，反之，對於非即時性需求的工作，則交由 Linux 核心處理。採用獨立即時排程器與硬體抽象層的好處是，不需要修改大部份的 Linux 核心，也

由於沒有進行大修改，因此，其他的子系統、裝置驅動程式或是使用者程式都能很順利地在上面運行，另外，也不需要額外去分析 Linux 核心程式碼路徑，唯一需要進一步分析的是即時排程器（及相關 IPC 和服務的程式碼）與抽象層。故採用此方法能提供具硬即時的工作環境。目前採用這類作法的即時作業系統，如：RTLinux 及 RTAI。

為了加強六軸運動平台的控制效果，我們採用第二種加強核心方式的 RTAI，來改善原有 Linux 作業系統不具即時性的問題，而 Real-Time Application Interface（RTAI)是由 DIAPM-RTLinux 加以改良的版本。DIAPM 的成員，將原有的 RTLinux 排程器保留，並加入一些新的功能與特性，在改良部份，則是放在即時的計時功能上，除了新加入週期性計時（Periodic timing）功能，並藉由 CPU 的 TSC 來取代 RTLinux 原有的計時方式，大大改善單次計時（One-shot timing）的效能。RTAI 第一個修改核心的正式版本，是在 1999 四月時，由 Paolo Mantegazza 所釋放出來。目前 RTAI 是一個開放源始碼的計劃，早期只支援 x86 的平臺，但目前已提供以下幾種硬體支援平臺：

x86
ARM (StrongARM, ARM7)
PowerPC
MIPS
CRIS

圖 4-2-1 為 RTAI 的運作架構圖。在架構上，RTAI 與 RTLinux 並沒有太大的差異，在設計上，RTAI 在 Linux 上定義了一組即時硬體抽象層（Real-Time Hardware Abstraction Layer，RTHAL）。RTHAL 將 RTAI 需要在 Linux 中修改的部份，集合並定義成一組系統介面，RTAI 可利用這組系統介面和 Linux 溝通。這樣做的好處在於，可以將直接修改 Linux 核心內部的程式碼減至最小。所以，未來在將 RTHAL 移植到新 Linux 核心的工作量，可減至最低。除了加入 RTHAL 外，RTAI 還加入了一個自己的即時核心。事實上，採用這種模組化設計的好處，甚至可不必採用 RTAI 的即時核心，而改用其他即時核心和 RTHAL 進行溝通。其中 RTHAL 主要工作有二項：

(1) 負責收集 Linux 指向硬體相關資料或系統函式的所有指標，至一個結構中（rthal）。
(2) 集合至這個結構的指標，能夠根據系統是否有要載入硬即時延伸，將這些指標以動態方式，指向 Linux 一般系統函式或是 RTAI 的即時系統函式。這讓 Linux 可在執行期間（Run time），進行一般作業系統與硬即時性作業系統的切換。

事實上，這樣的動作，除了輕微降低系統的效能，Linux 本身的運作幾乎不受加入的 RTHAL 所影響。輕微降低系統效能的原因要歸因於進行那些被取代的硬體相關函式呼叫，如：sti 及 cli 相關的功能函式，而這些呼叫並不是直接與硬體連結溝通的。RTAI 系統的即時工作，同樣也是以核心模組的方式存，並且在核心空間

中運作。當 RTAI 載入之後，Linux 核心不再是直接受到硬體中斷所控制的，而即時工作也不再受到它的掌控。至於採用核心模組，有一些好處是即時工作不會被進行分頁快取交換出去的動作，以降低 TLB 的 miss 率。而且即時工作的執行是在處理器的特權模式（Supervisor mode）下運作，因此擁有低階硬體的所有存取權。

因為 RTAI 是一個開放原始碼的計劃，所以早期是以 LGPL 2 的方式授權，但最近有關 RTAI 核心的部份，則是改為 GPL 2 的授權方式，其餘的部份仍是 LGPL 2 的方式。至於 RTAI 的檔雖然不是即時更新的，但算是相當完善。此外，mailing list 每日的訊息量相當地大，在上網詢問問題的時候，往往能很迅速獲得解答與答覆【4-2, 4-7】。

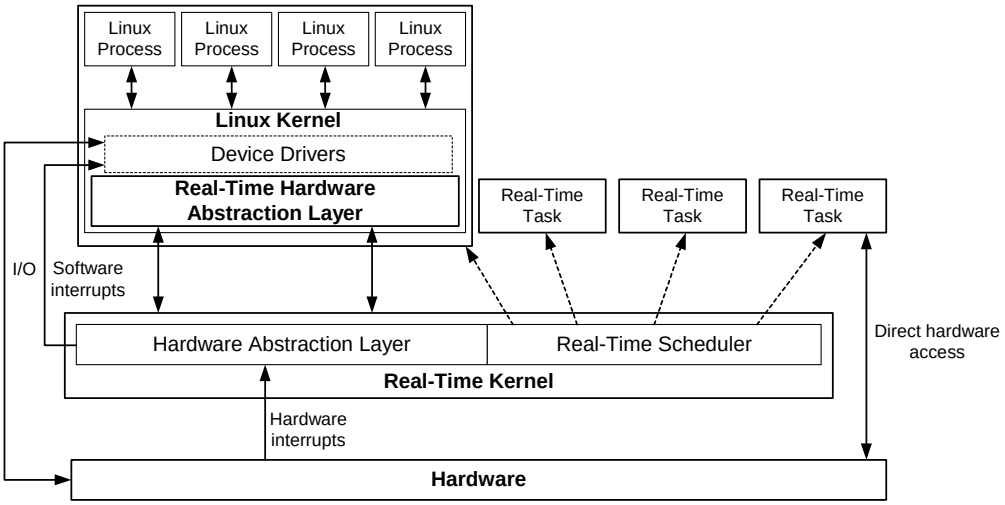


圖 4-2-1　RTAI 系統運作圖

4.2.2 即時控制系統之分析與設計

本年度的成果中我們建構了一個動感平台的嵌入式即時控制系統，其可用來提供一個具可靠性、精確性及穩定性的動態模擬環境，並進行了系統架構分析、系統規格之定義、系統架構設計、排程規劃與設計。

4.2.2.1 六軸平台即時控制系統架構分析

目前既有的控制系統架構如圖 4-2-2 所示，主要包括了主控端（PC or LART）、數位與類比轉換介面（D/A 及 A/D）、Stewart 六軸平台、線性比例尺感應器、油壓致動器、油壓伺服馬達以及硬體控制電路。其中的硬體控制電路，是負責將主控端送出的訊號進行訊號放大，以驅動油壓伺服馬達。除此之外，硬體控制電路亦提供一個 PI 型的硬體位置控制器，可根據主控端的位置訊號與感應器回應的訊號，進行一個閉迴路控制。硬體控制電路亦提供一個使用彈性，讓我們採用軟體實現控制器的方式取代原有的硬體控制器來進行運動平台的迴路控制。

致動器控制架構，如圖 4-2-3 所示。除了外部致動器及致動器感應器部份外，大致可分成兩個大部份：致動器控制與監控部份。致動器控制部份則又可分成五個子系統，分別是平台姿態之輸入處理、逆向運動學轉換、位置控制轉換、控制

器及致動器回授訊號的收集處理。其中逆向運動學轉換的部份是負責進行平台姿態轉換至各致動器長度的運算。由於實際上平台並無法滿足各種平台姿態動作，針對一些超出工作空間的情況，在致動器長度的轉換過程中會加以處理與限制。例如：判斷姿態轉換出來的致動器長度是否超出最大伸長量，而將致動器維持前次處理的長度，以對平台機制進行保護。而位置控制部份是將逆向運動學運算出來的致動器長度轉換至控制電壓訊號，並以固定的頻率輸入至控制器中。在系統監控部份，其主要任務是傳達操作人員的命令以及顯示整個控制系統低階控制訊號的收集與運算結果。

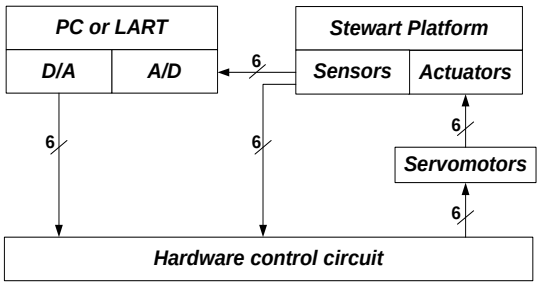


圖 4-2-2　六軸平台控制系統架構圖

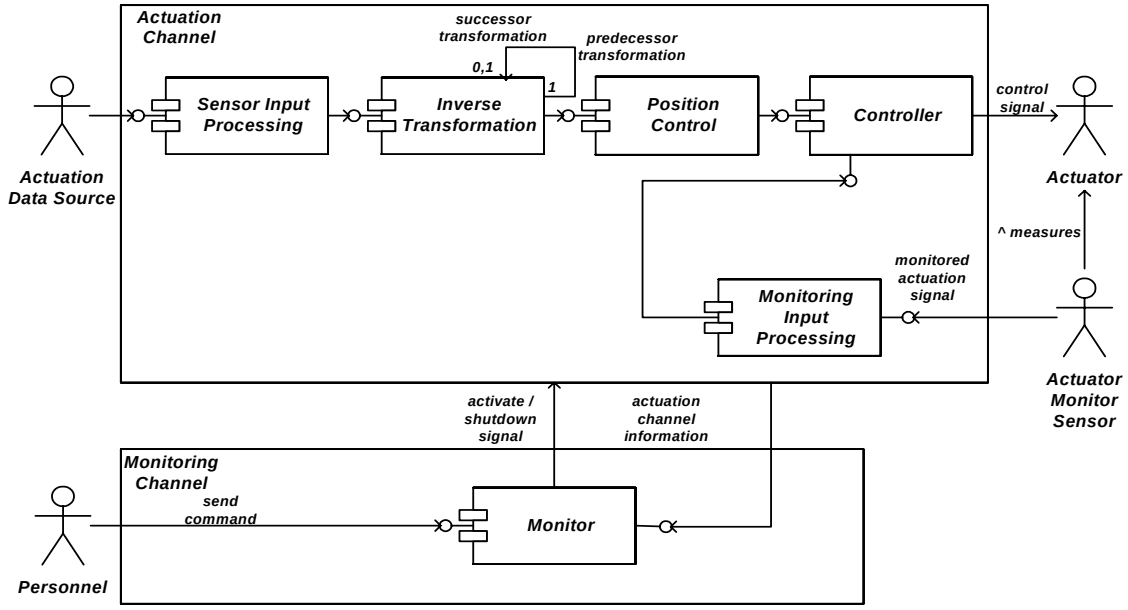


圖 4-2-3　致動器控制架構圖

4.2.2.2 功能規格

根據六軸平台即時控制系統架構分析結果，可將致動器控制系統的流程，歸納成七項子工作：

(1) 平台姿態之輸入處理

(2) 逆向運動學轉換

(3) 位置控制轉換

(4) 致動器控制訊號的輸出處理

(5) 致動器回授訊號的收集處理

(6) 控制資訊和轉換結果之資料收集

(7) 監控操作

在功能與規格的製定上，因項目(4)及項目(5)部份工作，是屬於低階硬體存取部份，為了讓整體模擬有更真實的效果，本論文是以1*ms*週期時間進行硬體控制與資料收集動作。而在考量檢視系統活動及資料處理是否正常，故安排監控探測之工作，並以50*ms*的週期進行系統資料更新和收集動作。至於資料的呈現上，是透過網路的方式在遠端監控的螢幕上展示。整體的系統規劃如圖 4-2-4 所示。其中由場景模擬端負責傳送模擬姿態至嵌入式控制板，並由遠端監視器以圖形的介面方式呈現嵌入式控制板的系統活動與控制數據。

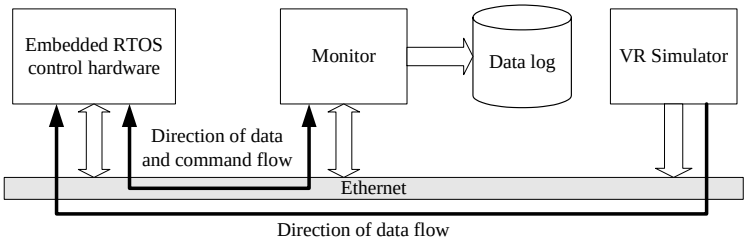


圖 4-2-4　系統整合架構圖

4.2.2.3 系統狀態設計

在整個即時控制系統的狀態設計中，我們根據模擬系統的流程，將系統分成 Testing 與 Simulation 兩種模式。並依據這兩種模式，將模擬系統分成六種系統狀態，分別為 initial, testing, ready, running, shutdown, emergency。圖 4-2-5 為我們的即時控制系統狀態圖，以下分就這兩種模式工作流程進行說明。

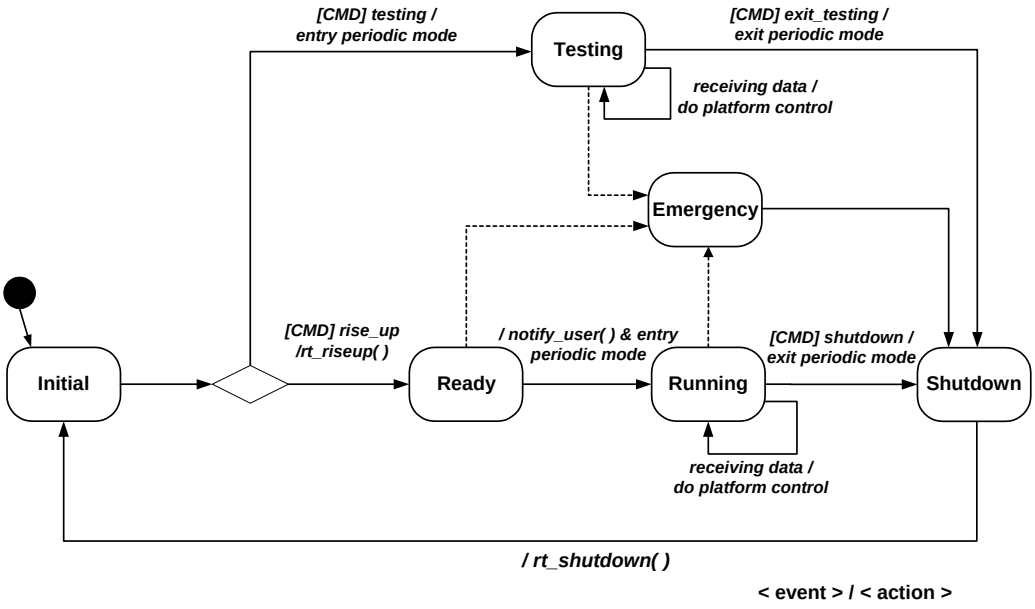


圖 4-2-5　即時控制系統狀態圖

(1) Testing 模式：

在此模式中，其目的是進行平台動作的測試，我們稱作平台軌跡規劃測試。系統將根據我們預先規劃好的平台移動路徑，進行各種姿態的模擬。系統的初始狀態為 initial，一旦使用者下命令要求進入 Testing 模式時，系統會由 initial 狀態進入至 testing 狀態，此時系統會由事先安排的動作路徑，以固定週期，進行即時的資料讀取、處理與控制。待使用者要求離開此模式時或是系統完成軌跡規化測試後，系統會轉移至 shutdown 模式，將平台回歸初始位置，最後系統狀態恢復至 initial 狀態。

(2) Simulation 模式：

平台在未進行任何模擬時，是處於 initial 狀態。當要進行動態模擬時，平台必須提昇至一定的高度，才能開始進行資料的處理與控制。所以，在使用者下達一個”rise_up”的命令後，平台提昇至固定的模擬高度，待完成這項動作後，系統狀態便會由 initial 轉換至 ready 狀態。在進入 ready 狀態後，控制系統接著通知使用者開始進行資料輸入的動作，隨即轉換至 running 的狀態。在此狀態中，模擬系統便不斷地接收控制的資料，進行處理、運算與平台控制。最後，待使用者再下達”shutdown”的命令後，系統會由模擬進行中的 running 狀態，切換至 shutdown 狀態。其中，shutdown 狀態所處理的工作是將模擬平台下降至未模擬的高度，並於工作完成後，回復至 initial 的狀態。

在考量系統安全性上，因為，在進行任何有關硬體平台存取控制的部份，如：ready、running、testing 狀態，皆有可能發生突發性的狀況。因此，在突發性的危急狀況下，我們安排了一個 emergency 的系統狀態來負責將平台動作恢復至初始狀態的動作。一旦進入到此狀態，不論系統此時在進行何種工作，系統會立即進行將平台下降並回復至未模擬狀態的動作。

4.2.2.4 系統架構設計

經由以上的狀態設計後，在整個六軸平台即時控制模擬系統上，除了規格中所列的七項工作外，必須再加入兩項工作，分別負責平台上昇以及平台下降之處理工作。而在進行架構的設計之前，我們將系統之工作切割成兩大類，分別是即時性與非即時性工作。而在即時性與非即時性工作的安排上，其採取的策略是：

■ 週期性工作或是與低階硬體有互動關係且需立即處理與運算的部份，皆被安排成硬即時工作。

■ 網路與檔案存取相關等會造成即時環境不確定性的工作，皆安排在非即時性的作業環境中。

此外，為了達到即時運算、處理的能力，我們將平台的逆向運動學及位置控制部份，安排成硬即時工作。故可完成硬即時工作的分配表，如表 4-2-1 所示。

表 4-2-1　硬即時工作定義表

Task	***Description***
Rise up	平台上昇
Shutdown	平台下降
Inverse kinematic	逆向運動學轉換
Position control	位置控制轉換
Analog output (D/A)	致動器控制訊號的輸出處理
Analog input (A/D)	致動器回授訊號的收集處理
Update	控制資訊和轉換結果之資料收集

而其餘的平台姿態輸入處理與監控操作常駐等工作則是安排在非即時性的使用者空間執行。至於工作及程序間通訊的安排上，則是依照下面兩項原則：

- 非即時工作與即時工作之間的溝通，如使用者空間程式與即時系統之工作的溝通，是採用 FIFO 的通訊機制來達成。
- 即時工作與即時工作之間的溝通，則是採用共享記憶體（Shared memory）的通訊機制來達成。

採用 FIFO 機制的原因是因為可利用其對應的 FIFO handler，適時地喚醒 FIFO 資料處理的工作。使用共享記憶體就不具有這樣的機制，因此無法立即對最新的資料進行處理，雖然我們仍然能利用一個短期週期性工作的輪詢（Polling）機制，達到同樣的目的，但是這並不是非常有效率的作法。特別是在當系統負載大的時候，使用者空間的程序並無法每次都非常迅速地更新共享記憶體資料，導致固定輪詢此記憶體的工作，做了許多不必要的工作而降低系統效能。因此，我們是採用三個 FIFO 佇列來完成所有非即時工作與即時工作間的通訊。其 FIFO 佇列工作定義表，如表 4-2-2 示。

表 4-2-2　FIFO 佇列工作定義表

FIFO-queue	***Description***
Payload information FIFO	平台姿態資訊佇列
Command FIFO	命令佇列
Information FIFO	系統活動與控制資訊佇列

整個平台即時控制系統的設計架構圖，如圖 4-2-6 所示。當平台姿態資訊傳入 Payload information FIFO 時，其對應的 FIFO handler 會喚醒 Inverse kinematic 以及 Position control 的工作，一旦 Inverse kinematic 工作執行並完成動作姿態轉換成各致動器長度之後，會將各致動器長度儲存至共享記憶體區(A)。而 Position control 工作隨即接續執行。Position control 工作將共享記憶體區(A)的各致動器長度轉換成各致動器的控制電壓值並儲存至另一個共享記憶體區(B)。

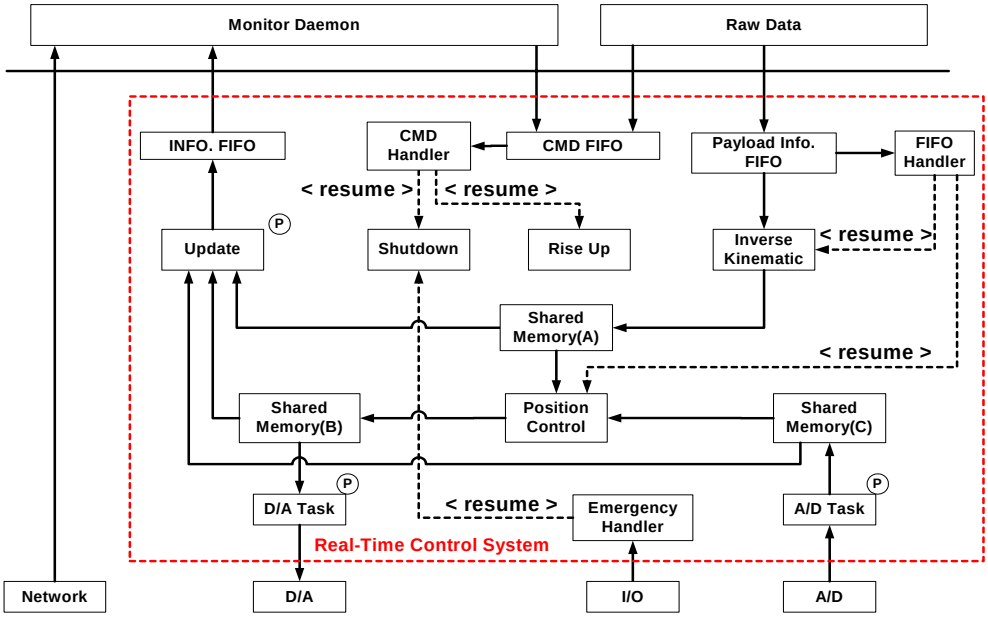


圖 4-2-6　即時控制系統系統架構圖

就週期性的工作而言，一共有三項，分別是：D/A 工作、A/D 工作以及 Update 工作。D/A 的工作是以1*ms*週期讀取存放於共享記憶體(B)中的致動器控制電壓值並以此電壓值控制硬體。而 A/D 的工作亦是以相同頻率讀取感應器之電壓值，並將收集的結果存入共享記憶體(C)中。最後，Update 工作則是以50*ms*的週期速率，收集(A)、(B)、(C)三個共享記憶體內的資訊，並將所有資訊傳入 Information FIFO 中。而 Information FIFO 內的資料，則是由 Monitor daemon 負責收集並透過網路傳送至遠端的監視系統中。

除此之外，系統中的 Command FIFO 是負責接受使用者命令，而其 Command handler 會根據不同命令形式適時地改變工作模式或控制工作，如：Rise up 及 Shutdown 等命令。至於 Emergency 處理方式，則是由外部中斷的 Emergency handler 負責喚醒 Shutdown 工作。

以上已將即時控制系統之各個子工作之關係進行了詳細的劃分。然而，每個即時工作內容有其工作的重要性。因此，我們將根據各個即時工作對時間的需求程度，進一步分析與安排系統工作之優先權順序。

4.2.2.5 工作優先權之分派

為了確保有些關鍵性工作能夠優先執行，如 Shutdown 工作。因此，我們進行了工作優先執行權的指派動作。其指派的原則有幾項：

- 越關鍵的工作或與系統安全性攸關的工作，其優先權越大。例如：逆向運動學的處理，平台的啟動與平台恢復或關閉等工作。
- 較不重要的資訊或是越不需要即時處理的資料，其優先權越低。例如：提供給使用者空間資料的工作，Update。
- 運算處理的時間越短的，其優先權越大。但與原則一、二相違背時，則以先前的原則為準。

利用第一個原則，可決定逆向運動學處理、平台啟動與平台恢復三項工作的優先權。我們假設逆向運動學轉換、平台啟動與平台恢復，其工作優先權為P_1、P_2以及P_3。基於安全性的考量以及平台的下降恢復工作是將目前平台位置至平台初始位置的路徑進行切片（Slice）處理，因此，不需要逆向運動學的轉換，故可以確定的是P_3為最大。反之，平台的啟動上昇動作，是由給定一初始位置姿態，經由逆向運動學轉換至各致動器長度後，再進行路徑切片處理，因此，在平台啟動的過程中，必須喚醒逆向運動學轉換工作，並將工作權交由其處理，故在優先權的安排順序上，為$P_1 > P_2$。由以上分析結果，可得知優先權順序為$P_3 > P_1 > P_2$。

根據第二項原則，由於 Update 的處理是讀取系統的處理結果與狀態，並不涉及任何的硬體控制操作，而所提供的資訊亦是給使用者空間的程式使用，因此，在即時性的需求上，並沒有其他工作來得大，故其執行的優先權為最小。

然而在第三項原則中，我們考量致動器訊號的輸出與收集處理工作兩項週期性工作以及位置控制的優先權順序。假設致動器訊號的輸出T_4與收集處理T_5工作，其工作優先權分別為P_4及P_5，而位置控制工作T_6的優先權為P_6。事實上，考量運算處理的時間越短其優先權越大的原因是為了避免在執行優先權大且處理時間長的工作時，造成進入等待佇列（Ready queue）的週期性工作的長時間延遲，而這種情況特別容易發生在週期性的工作中。因此，在考量P_4及P_5的大小順序上，由於致動器訊號的收集動作，需要進行類比－數位訊號轉換，其處理時間上較數位－類比訊號轉換來得長，故優先權大小為$P_4 > P_5$。而在P_6的安排上，由於T_4使用的控制資料是T_6所提供，在執行順序上應是$T_6 \rightarrow T_4$，則$P_6 > P_4$。但是，這在某些時間點下會造成週期性工作T_4的少許延遲，例如：T_4和T_6在同一時間點發生，由於$P_6 > P_4$，則週期性工作T_4被延後執行。但因在$1ms$控制週期下，其控制資料的差異並不大，因此，T_4被延遲操作並不至於造成太大影響，所以，$P_6 > P_4$的優先順序是可接受的。故由條件：$P_4 > P_5 \cap P_6 > P_4$可得知T_4、T_5及T_6的優先權關係為：$P_6 > P_4 > P_5$。

綜合以上分析，以下是有關工作優先權的分配表如表 4-2-3 所示。其中`1`代表最高，`10`代表最低。

表 4-2-3　工作之優先權分配表

Task	***Priority***	***Comment***
Shutdown platform	1	Command-driven
Inverse kinematics	2	FIFO-driven
Rise up platform	3	Command-driven
Analog output (D/A)	4	Periodic
Position control	5	FIFO-driven
Analog input (A/D)	9	Periodic
Update	10	Periodic

至於整個即時控制系統的工作排程策略是採用 RTAI 預設的先進先出（FirstInFirstOut，FIFO）的排程方式。在這種排程策略下，不同優先權的工作有不同的 FIFO 佇列，而相同優先權的工作會被放置在同一個 FIFO 佇列中，先進入此佇列的工作，會先取得 CPU 的控制權。但這種排程策略有一個缺點，就是相同優先權的工作，當中若是有一個工作時間特別長時，先進入 FIFO 佇列中後，其之後發生的工作必須等待一段較長的時間才能獲得執行權。因此，這種排程策略無法確保最小化在佇列的平均等待時間。然而，就之前的工作優先權分配表中，其分派的大小並無相似的情況發生，因此可避免此種情況的發生。

除此之外，FIFO 排程策略的工作方式為非強取式的，也就是說每一項工作必須完成它的工作內容後，並釋放 CPU 執行權，其後的工作才能取得 CPU 控制權繼續執行。因此，使用此種排程策略能簡化系統的複雜程度且易於進行工作流程的分析，這也是我們採用 FIFO 排程策略最主要的原因。

4.2.3 即時控制系統之效能測試

為了測試 RTAI 在 LART 嵌入式即時系統中的反應能力與排程精確度（Scheduling Precision），我們安排了中斷延遲（Interrupt Latency）、中斷任務延遲（Interrupt Task Latency）、排程延遲（Scheduling Latency）及排程誤差（Scheduling Jitter）等測試樣本，來進行系統即時能力的評估【4-5】。

4.2.3.1 中斷延遲（Interrupt Latency）

中斷延遲是指一個外部中斷發生時，一般是由外部裝置所產生的，一直到系統的中斷處理常式開始執行的這一段時間，如圖 4-2-7。當系統處於閒置的狀態下，這段時間是非常短的，但當有其他外部的中斷也在這個時間內發生時，這時中斷處理常式就可能會被延遲執行。事實上，這段時間的長短差異與硬體架構也有密切相關。中斷延遲是即時系統裡一個重要的指標，因為一個中斷延遲會影響其他的即時性的效能，如：排程的精準性（Scheduling precision）、中斷工作延遲時間（Interrupt task latency）。

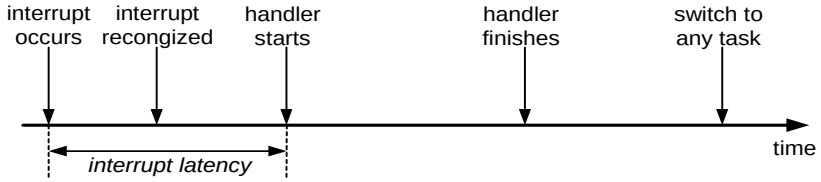


圖 4-2-7　中斷延遲定義

為了量測中斷延遲，我們利用 PC 的並列埠（Parallel port）的 Data Pin 以及 nAck Pin 分別來產生中斷訊號與接受回應訊號，而實驗板上同樣規劃了兩個 GPIO，這裡採未被使用的 GPIO2 及 GPIO3，分別當作中斷源與反應腳位。其中中斷源 GPIO2，被規劃成上緣觸發。量測的接腳圖，如圖 4-2-8 所示，圖中 PC signal 代表由 PC 端送出的訊號； LART signal 代表由 LART 端送出的訊號。

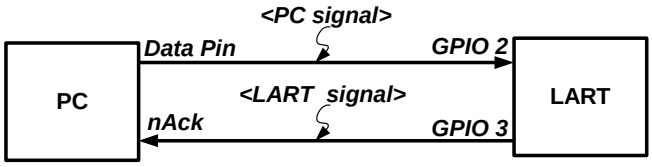


圖 4-2-8　系統量測接腳圖

當 PC signal 由 low 變 high 時，LART 實驗板上的中斷被觸發，作業系統將目前所有中斷，進行中斷處理常式的分派動作，當對應的中斷處理常式被喚醒時，隨即將 LART signal 設成 low，PC 端得知 LART signal 被設成 low 後，同時也將 PC signal 降為 low。最後，中斷處理常式完成工作後，會再將 LART signal 回恢成 high。在 PC 中斷產生至 LART 回應訊號這段時間就是中斷延遲，我們不斷地重覆這樣的程序，並利用邏輯分析儀量測並記錄這段時間。其時序圖，如圖 4-2-9 所示。我們在無負載與負載情況下，分別安排三種不同環境下，進行中斷延遲時間的量測：

(1) RTAI 環境下的 Real-Time interrupt handler

(2) RTAI 環境下的 Linux interrupt handler

(3) 一般 Linux interrupt handler

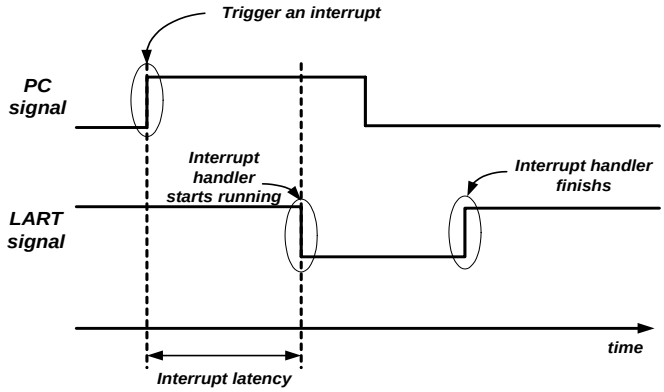


圖 4-2-9　中斷延遲測試時序圖

這裡所呈現的三種不同測試環境之中，RTAI 的 Real-time handler 與 Linux 的 handler 的量測結果，是為了比較在即時作業系統與一般作業系統，其系統的反應能力差異。而 RTAI 下的 Linux handler 量測結果，是為了說明 RTAI 的 RTHAL 對 Linux 核心所造成的一些影響。其量測的結果，如表 4-2-4 及表 4-2-5 所示。

表 4-2-4　無負載情況下中斷延遲統計表

Handler	***Max.***	***Min.***	***Mean***	***Med.***	***Std. Dev.***	***Samples***
RTAI RT-handler	6.976	1.2	1.9944	2.056	0.498	19978
RTAI Linux handler	66.22	3.112	4.6868	4.68	1.231	19945
Plain Linux handler	7.512	1.232	1.7299	1.536	0.60	21397

表 4-2-5　負載情況下中斷延遲統計表

Handler	***Max.***	***Min.***	***Mean***	***Med.***	***Std. Dev.***	***Samples***
RTAI RT-handler	15.56	1.6	3.002	2.744	0.9905	19941
RTAI Linux handler	518.4	4.84	67.151	50.58	64.187	20385
Plain Linux handler	41.63	2.528	5.119	3.76	4.613	22078

首先針對無負載測量的結果來看，我們可以觀察得知，在無負載的情況下 RTAI 的 Real-time handler 與 Linux 的一般 handler，其反應能力的差異並不明顯。但就平均的反應能力而言，RTAI 的1.9944μs略高於 Linux 的1.7299μs，由此可見，在無負載下，Linux 較能提供較佳的平均效能。而造成這種小差距的原因是 RTHAL 在中斷處理時，增加了微小的延遲，此微小的延遲我們可以由圖 4-2-10 及圖 4-2-11 的中斷延遲分佈圖，看出這樣的差異性。其中的中斷延遲分別集中在1.2μs~6μs以及2.2μs~7μs。而在 RTAI 下的 Linux handler，並不是直接接受硬體中斷，而是 RTHAL 先接收硬體中斷後，經過中斷的分派，並執行對應的中斷處理常式後，再產生軟體中斷給 Linux 核心，這時才由以 Linux 來執行中斷處理常式。在 RTHAL 的中斷分派與處理過程中，可能有大量的中斷發生，而即時作業系統必須先行處理必要的中斷常式，因此，這就造成之後在 Linux 核心處理中斷的明顯延遲。

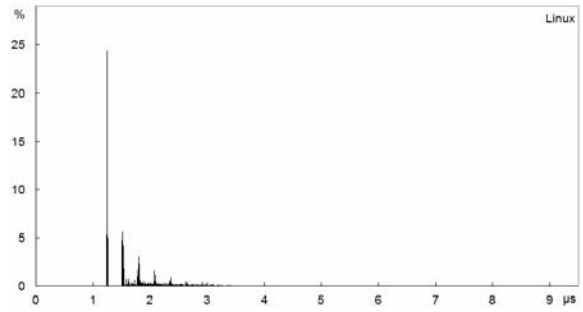


圖 4-2-10　無負載情況下 Linux 中斷延遲分佈

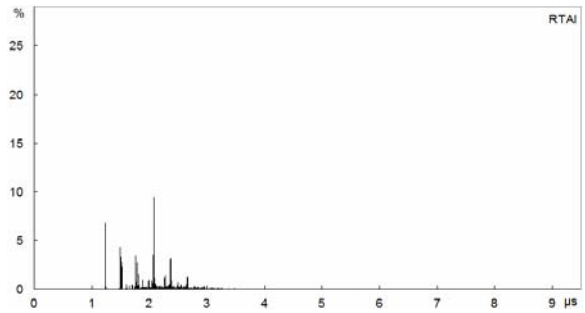


圖 4-2-11　無負載情況下 RTAI 中斷延遲分佈

接著，在負載的情況下，由實驗結果得知 RTAI RT-handler 與 Linux handler 就有較明顯的延遲差異。但此時 Linux 平均延遲5.119μs欲較 RTAI 下的延遲3.002μs大，主要的原因除了加入的外部負載中斷(網路中斷)，確實造成了 RTHAL

與 Linux 核心的負擔外，探視 /proc 下的 Linux 核心資訊，造成 Linux 核心更大的負擔，以致於有較大的延遲。在相同條件下，RTAI 的 RT-handler 不受 Linux 核心負載的影響，而有較好的反應能力。圖 4-2-12 與圖 4-2-13 為負載下中斷延遲分佈圖。

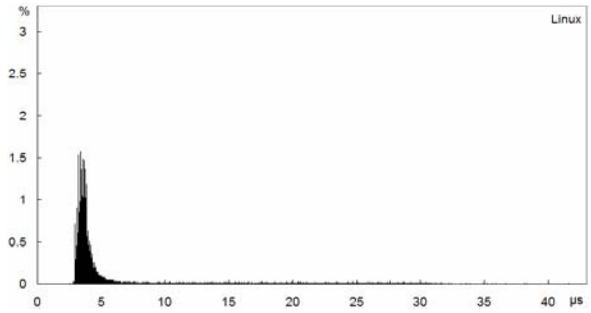


圖 4-2-12　負載情況下 Linux 中斷延遲分佈

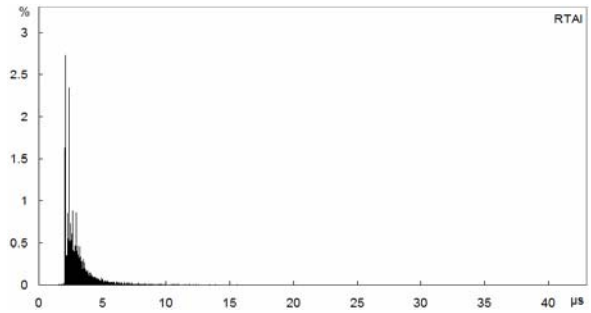


圖 4-2-13　負載情況下 RTAI 中斷延遲分佈

在負載的情形中，其最差情況（Worse-Case）的延遲，如圖 4-2-14 與圖 4-2-15 所示，在 Linux 的環境中，其延遲為 41.63μs ，而 RTAI 的環境中，其延遲皆被局限在16μs 以內，顯然 RTAI 擁有較佳系統反應能力。就其延遲標準差而言，Linux 為 4.613μs ，而 RTAI 的標準差值被控制在1μs 以內，顯視中斷延遲在 RTAI 環境下的變動差異較小，較能提供一個可預測的即時環境。

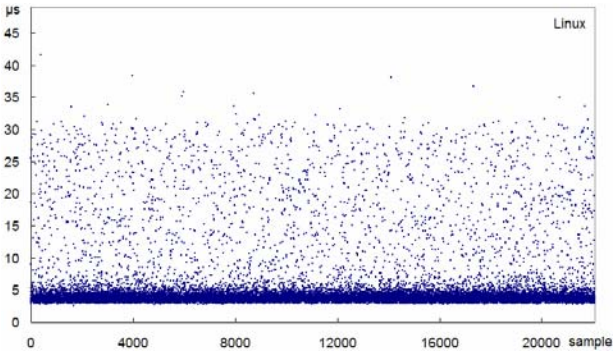


圖 4-2-14　負載情況下 Linux 中斷延遲取樣（22078 samples）

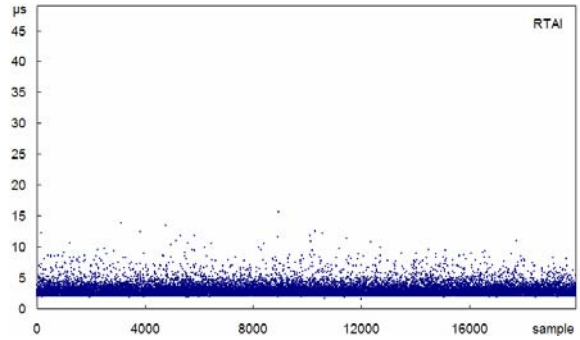


圖 4-2-15　負載情況下 RTAI 中斷延遲取樣（19941 samples）

4.2.3.2 中斷任務延遲（Interrupt Task Latency）

假設一個外部裝置已收集好所有外界資料時，裝置會發送一個硬體中斷訊號給系統，系統接收到中斷後會找尋對應的中斷處理常式並執行。外部裝置中斷發生之後至系統選擇對應的工作這段時間，就是所謂的中斷任務延遲，如圖 4-2-16 所示。為了方便了解系統活動，除了使用原有的 PC signal 當作外部中斷及 LART signal 做為中斷任務的執行狀態外，我們另外安排了一個 GPIO 4 當作中斷處理常式的執行狀況，稱為 Handler signal。

如圖 4-2-17 所示，當 PC signal 設成 high 時，對應的處理常式會被執行，於是 Handler signal 亦被設定在 high，待處理常式結束前，會將優先權高的工作放入等待佇列中，這時 LART signal 被設定成 low，表示工作處於等待狀態。當處理常式結束後，系統會由等待佇列中挑選優先權高的工作執行，一旦對應的工作開始執行時，LART signal 會被恢復成 high 狀態。而 PC signal 為設定成 high 至 LART signal 設定成 high，這段時間就是欲量測的延遲。在這項測試中，分別在無負載與負載情況下，我們安排了 Linux 與 RTAI 兩種環境，進行中斷任務延遲的量測與比較。

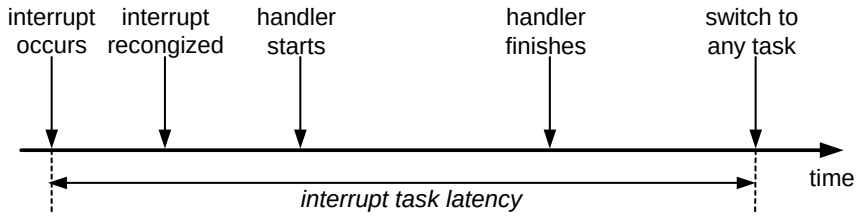


圖 4-4-2-16　中斷任務延遲定義

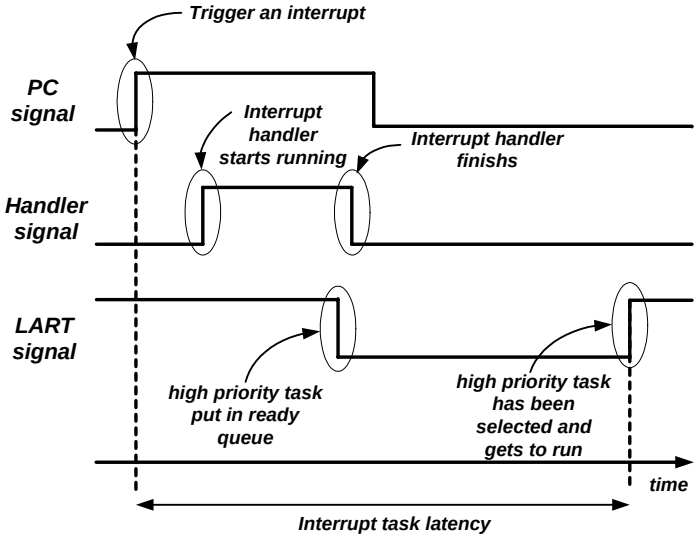


圖 4-2-17　中斷任務延遲測試時序圖

中斷任務延遲是由中斷延遲、處理常式處理時間以及排程延遲所組成。其中，中斷延遲與排程延遲是屬於硬體與系統行為，為了讓中斷任務能快速反應處理，中斷處理常式必須非常迅速地完成工作，這是我們能掌控與關切的。但中斷延遲與排程延遲，則依不同的系統，有不同層度上的表現，也造成了硬即時作業系統（RTAI）與一般作業系統（Linux）的差異。

表 4-2-6 為系統無負載時 RTAI 與 Linux 中斷任務延遲的結果。根據平均延遲與最大延遲來看，Linux 採用較好的平均系統效能處理方式，因此有較佳的平均延遲，而 RTAI 則是將最大延遲進行了最佳化。在系統呈現負載的情況下，如表 4-2-7 所示，Linux 的中斷任務延遲明顯與 RTAI 有非常大的差異（最大值而言，Linux：62.27ms，RTAI：46.93μs；平均值而言，Linux：411.53μs，RTAI：24.94μs），在中斷延遲上的差異則不明顯。而造成這項明顯差距的主要原因，在於 Linux 與 RTAI 排程器在處理上的不同，導致排程延遲的差異。在 Linux 中，當中斷處理常式完成至某一核心執行緒（Thread）被排程器選擇執行的這段時間，所有的中斷皆被重新開啟的，也就是說，在這段時間裡，當外界訊息過度頻繁時，不論訊息的重要性，系統必須先處理這些外界訊息而導致中斷任務明顯的延遲。相對地，RTAI 的排程器在這段時間的處理上，其所有外部中斷仍然是關閉的，以致於排程器能立即安排中斷任務的執行。

圖 4-2-18 及圖 4-2-19 為負載下 Linux 與 RTAI 中斷任務延遲的取樣圖。顯然 RTAI 在中斷任務的處理與反應上，擁有較迅速處理的能力以及較穩定與既定的環境。

表 4-2-6　無負載情況下中斷任務延遲統計表

	Max.	***Min.***	***Mean***	***Med.***	***Std. Dev.***	***Samples***
Plain Linux	65.98	4.904	11.4927	10.22	5.5249	21046
RTAI	25.94	3.12	15.7029	16.59	4.6102	20686

表 4-2-7　負載情況下中斷任務延遲統計表

	Max.	*Min.*	*Mean*	*Med.*	*Std. Dev.*	*Samples*
Plain Linux	62,270	78.69	411.5263	322.5	863.3681	19500
RTAI	46.93	20.49	24.9445	24.28	2.4264	20805

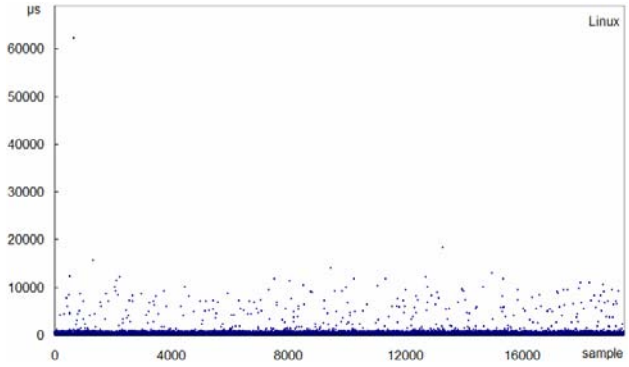


圖 4-2-18　負載情況下 Linux 中斷任務延遲取樣（19500 samples）

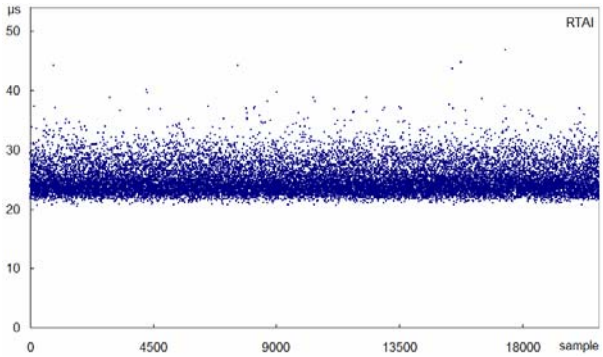


圖 4-2-19　負載情況下 RTAI 中斷任務延遲取樣（20805 samples）

4.2.3.3 排程延遲（Scheduling Latency）

一個即時系統裡，當有一個優先權高的工作已準備好要工作，這時排程器會中斷優先權低的工作，並恢復這個優先權較高的工作的工作環境。這裡所進行的是內文切換（Context Switch）工作，而排程延遲就是排程器花費在工作切換時，這段系統消耗掉的時間。這裡考慮兩種工作切換方式來呈現系統的排程延遲，分別是暫停/恢復（Suspend/Resume）及旗號（Semaphore）機制。在 Suspend/Resume 情況下，一個高優先權的工作先暫停本身的工作，並由低優先權的工作來恢復這個工作的執行。而在 Semaphore 的情況下，則是將 Semaphore 的初始值設定為零，表示當高優先權工作要取得旗號時，必須先被擱置，直到低優先權工作發出一個訊號給高優先權的工作，高優先權的工作才繼續工作。這裡利用 LART signal 當作量測訊號，當低優先權的工作在進行恢復高優先權工作的運作前，會將這個訊號設成 high，一旦高優先權的工作開始執行時，會將這個訊號設成 low，並且再次暫停本身的工作。其時序圖，如圖 4-2-20 所示。為了方便量測，我們另外安排了一

個優先權更低的週期性工作，來負責喚醒低優先權工作，以重覆相同動作。

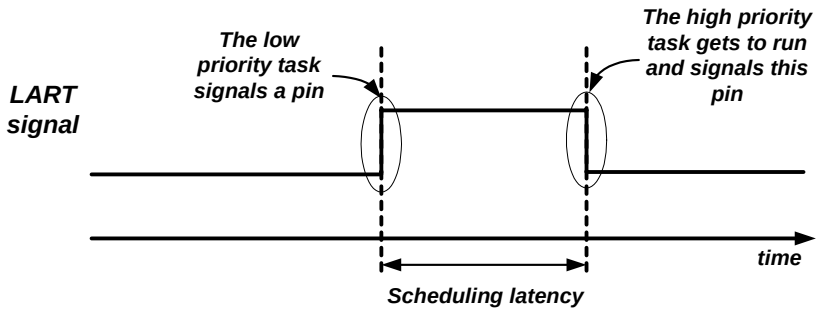


圖 4-2-20　排程延遲時序圖

由表 4-2-8 與表 4-2-9，可以看出使用 Semaphore 的方式較直接 Suspend/Resume 優先權高的工作的方式來得慢，但在做多個工作或執行序的應用中，普遍會使用 Semaphore 機制，因此採用 Semaphore 的測試結果當作系統排程延遲是較貼近實際情況的。

事實上，無負載下的排程延遲最小值1.744μs及1.904μs很可能就是排程器在進行排程的時間，在系統完全處於閒置狀況且無任何中斷發生時，大都是維持這固定的延遲，如圖 4-2-21 的延遲水平所示。

而在進行負載測試過程中，中斷可能在低優先權工作設定訊號為 high 後，且尚未進入排程器中斷關閉區段之前發生，這時系統必須花費額外的中斷處理時間，而使排程延遲增加。就延遲平均值與延遲中間值而言，在無負載時這兩個值之間的差距並不大。而在負載時，依然維持此相對特性，因此在兩種情況下，標準差值並無任何變化。唯一的差異是，外部中斷的影響，造就了一個固定量的偏移延遲量，如圖 4-2-21 與圖 4-2-22 所示。所呈現的是 RTAI 在排程延遲的能力表現。未考慮 Linux 的排程延遲是因為 Linux 在排程器進行排程時，中斷是皆被重新開啟的，不論外部訊息的重要性，系統必須先處理完所有中斷，接著才會完成工作的排程動作。這也是為什麼 Linux 下的 Semaphore 是屬於不精確的同步機制，而 RTAI 下的 Semaphore 卻是精確的同步機制的原因。

表 4-2-8　無負載情況下排程延遲統計表

	Max.	***Min.***	***Mean***	***Med.***	***Std. Dev.***	***Samples***
Suspend/Resume	7.104	1.744	2.1982	1.744	1.0157	10240
Semaphore	7.784	1.904	2.2477	1.912	0.9943	10237

表 4-2-9　負載情況下排程延遲統計表

	Max.	***Min.***	***Mean***	***Med.***	***Std. Dev.***	***Samples***
Suspend/Resume	16.04	5.464	6.6546	6.408	1.1888	10240
Semaphore	18.18	7.112	8.1953	8.0	0.9964	10237

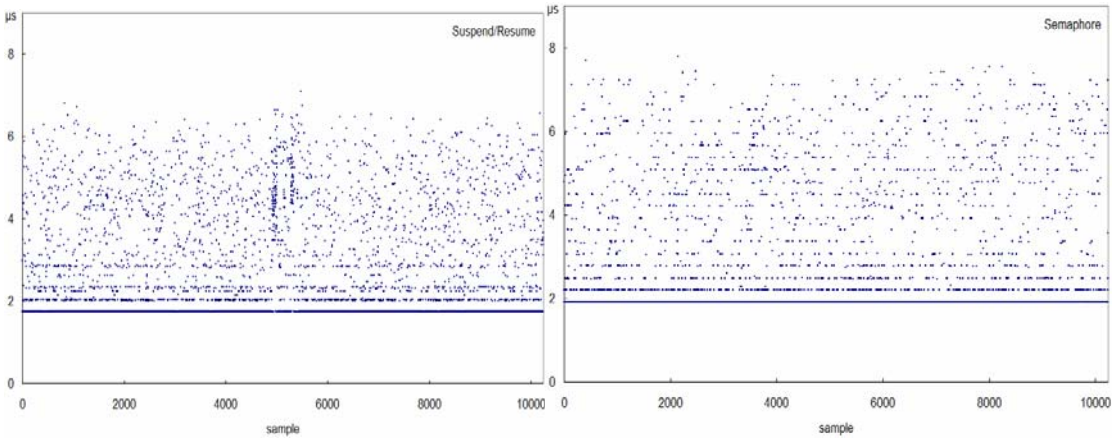


圖 4-2-21　無負載情況下 RTAI 排程延遲取樣

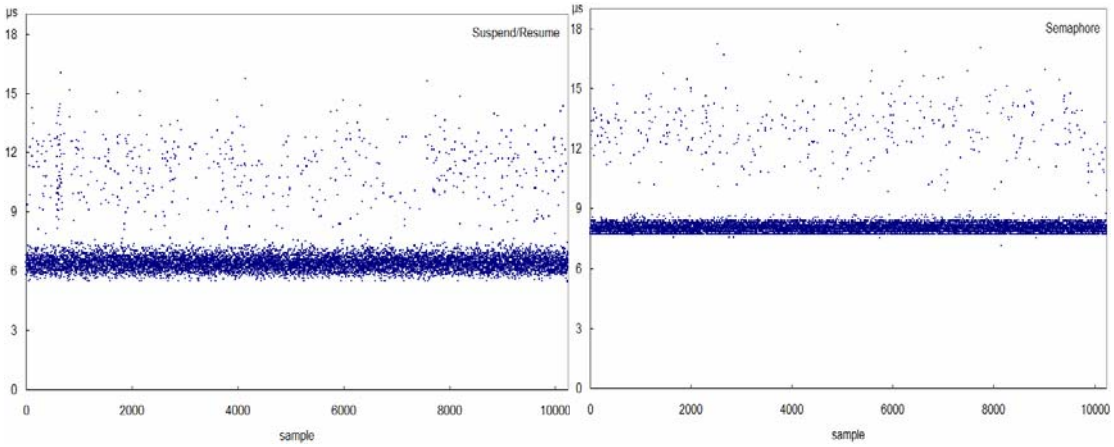


圖 4-2-22　負載情況下 RTAI 排程延遲取樣

4.2.3.4 排程誤差（Scheduling Jitter）

一個週期性工作對於機械裝置資料取樣與控制有密切的關係。而排程的精確度與準確性往往會影響整個控制或是取樣的結果。舉例而言，一個週期為 T 的即時工作，從 $t=0$ 時開始運作，理想化的結果是無視任何優先權低的工作，分別在 $t=T,2T,3T,...$ 的時間點上執行工作。但事實上，並非恰好都能在這些時間點上開始工作，而是如圖 4-2-23 所示，第一次的執行時間可能為 $t=T+\Delta t$，其中 Δt 為一不為零的時間區段。其中與實際的時間點這段偏差區段 Δt，即為排程誤差。實際上，排程誤差的大小與排程器被執行頻率有關，執行的頻率則與即時計時器的解析度有關，此外，額外的中斷處理也會影響系統排程誤差大小。

量測排程誤差的方式，是安排一個週期性的即時工作，並且將 LART signal 規劃成一個開關觸發器，每當工作開始執行時，會切換這個訊號，並等待下個週期來臨，再度切換訊號。不同切換點間的時間，與實際週期的偏差，就是系統的排程誤差。當即時工作被延遲時，那麼這段量測到的工作週期將會是比較長的，若是下次恰好在正確的時間點執行時，那麼下次測量到的週期將會變得非常短。在此測試中，排程器設定為 Periodic mode，分別在無負載環境與負載環境下，進

行不同週期的測試。

表 4-9 及表 4-11 列出了不同的 Timer Tick 與 Task Period，在負載與無負載情況下排程誤差的統計結果。這裡得到的誤差是由取樣到的週期與理想週期相減得到的，因此誤差可為正或負值。而以下統計的數值皆是以誤差之絕對值予以統計。在這項測試中，我們安排了兩種 Timer Tick（100μs 及 500μs）交錯配合兩種 Task Period（500μs 及 1ms），藉以探討排程誤差的影響。這裡所採用的負載測試，並未載入使用讀取 /proc 核心資訊，因為這會造成測試上穩定性的影響，取而代之的方式，是加入兩個低優先權的週期性工作。

由表 4-2-10 與表 4-2-11 所示，可以看出當系統負載時排程誤差值明顯較大。造成這項差異的主要原因是因為 RTAI 系統在接到一個新的中斷時，會將系統所有的中斷關閉一小段時間，若是一個新的中斷在所有中斷關閉的情況下發生，那麼這個中斷不會被系統立即處理，直到所有的中斷再度被開啟後，才會進行這個中斷的處理。而那些由 Timer 中斷啟動的週期性工作，可能也就被這樣的處理延遲一小段時間。

在無負載情況下，當 Tick 週期及工作週期增大時，排程的誤差值有輕微的增加，特別是當工作週期增加至 1ms 有更明顯的趨勢。相信這是即時系統在規劃系統振盪器時所造成的誤差。然而，只要這個偏差值維持一定的值，那麼這個誤差可由其他方式加以補償而不至造成任何問題。

在負載情況下，由於外部大量中斷產生，若系統的 Tick 週期越小，如表 4-11 前兩列，系統受到外部中斷的影響機率越大，導致整體平均誤差值與誤差中間值增大。相同地，如表 4-11 後兩列，工作週期越小，表示受到的中斷影響機率越大，因此，週期性工作其週期越小，工作越頻繁，會有較大的平均誤差值與誤差中間值。然而，大多數誤差源，來自於中斷延遲，因此最差情況下的誤差值與最差情況下的中斷延遲有關。此外，就最大誤差值而言，在相同 Tick 週期且工作週期為 500μs 下，其最大排程誤差值為 38.1μs，故最差情況下約有 7.62% 的偏差比，而就工作週期 1ms，最大排程誤差值 33μs，其最差的偏差比為 3.3%，因此工作週期越大的話，這項誤差值幾乎可忽略。

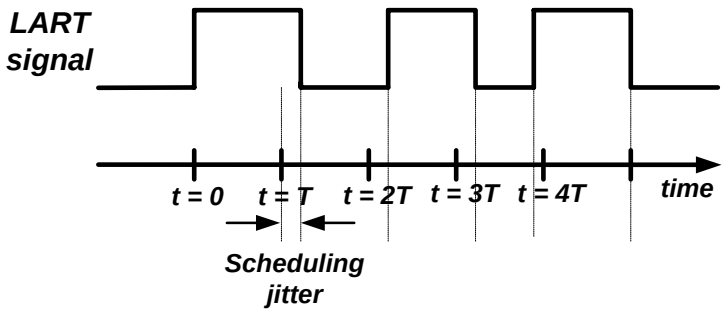


圖 4-2-23 排程誤差時序圖

表 4-2-10　無負載情況下排程誤差統計表

Task Period	*Tick Period*	*Max.*	*Min.*	*Mean*	*Med.*	*Std. Dev.*	*Samples*
500μs	100μs	18.1	0	0.1362	0.1	0.4490	24570
500μs	500μs	21.6	0	0.1404	0.1	0.5522	24570
1*ms*	500μs	22	0	0.3366	0.3	0.6832	24570

表 4-2-11　負載情況下排程誤差統計表

Task Period	*Tick Period*	*Max.*	*Min.*	*Mean*	*Med.*	*Std. Dev.*	*Samples*
500μs	100μs	29.2	0	6.6825	5.8	4.8336	24570
500μs	500μs	38.1	0	5.3783	3.9	5.0989	24552
1*ms*	500μs	33	0	4.8493	3.4	4.6228	24566

4.2.3 研究成果

目前為止，本年度已於 PC 與 LART 平台上完成虛擬動感平台之即時控制系統。圖 4-2-24、圖 4-2-25 及圖 4-2-26 為我們在 PC 平台下利用 LTT 對核心工作追蹤在 Testing 工作模式下的模擬結果。其中，圖 4-2-25 為圖 4-2-24 放大的結果，而圖 4-37 為 FIFO 在讀取平台姿態時，對工作的恢復結果。圖中左欄位之 RT：22、RT：23、RT：24、RT：27 及 RT：28，其分別代表 Inverse kinematic、Update、Position control、Analog input 以及 Analog output 工作。

模擬結果顯示出我們所安排的週期性工作以及工作之優先順序皆符合本論本之即時控制系統架構流程。此外，本論文進行 PC 與 LART 平台這兩種平台之執行效能測試，其分別是採用 x86 及 StrongARM 的處理器硬體架構，在不同硬體架構下其子工作之執行時間的情況，如表 4-2-12 所示。

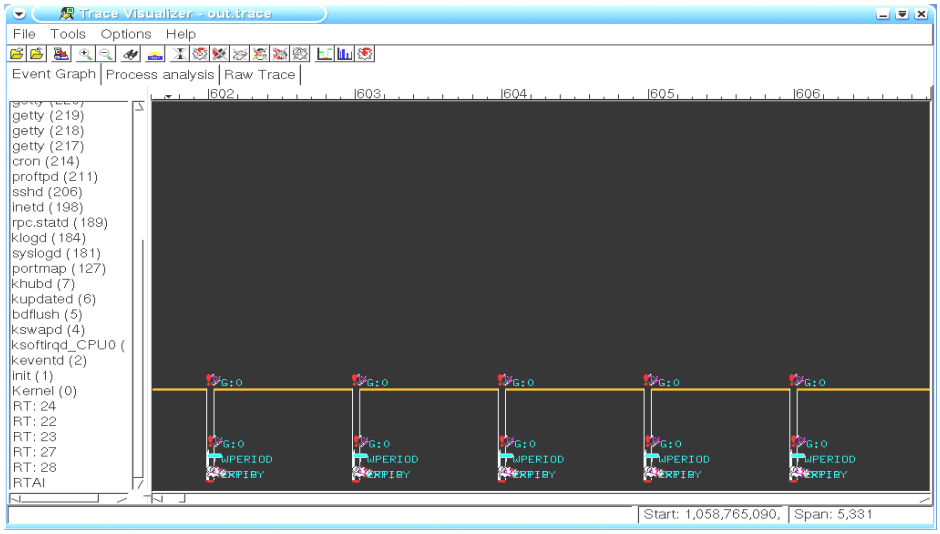


圖 4-2-24　週期性工作模擬結果(1)

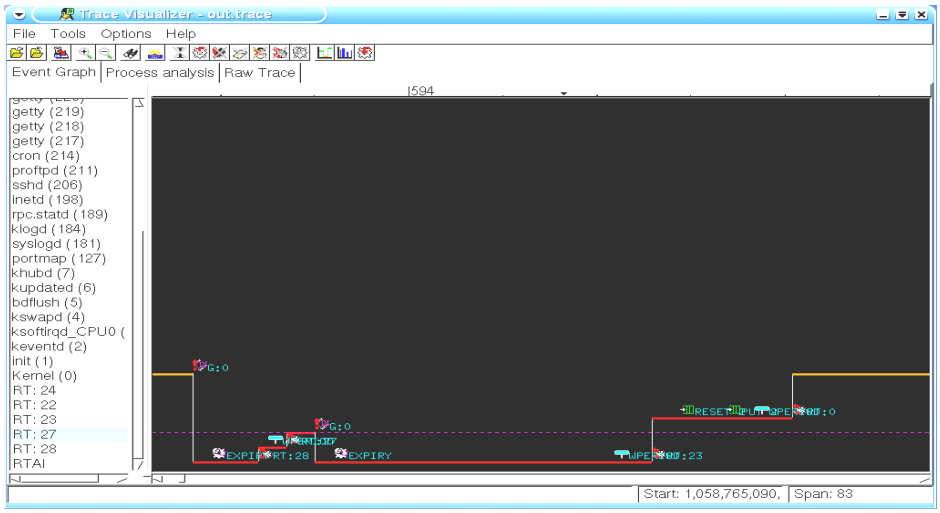


圖 4-2-25 週期性工作模擬結果(2)

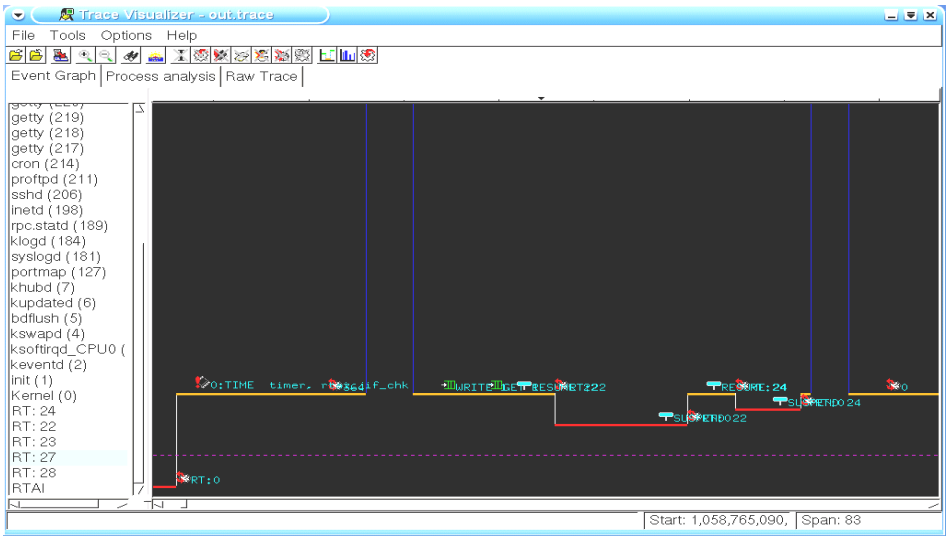


圖 4-2-26 FIFO 對工作恢復之模擬結果

表 4-2-12 x86 與 StrongARM 平台之各子工作執行時間比較

Task	***Priority***	***x86***	***StrongARM***	***Comment***
Shutdown	1	1*s*	1*s*	Command-driven
Inverse kinematics	2	23.8μs	472μs	FIFO-driven
Rise up	3	3*s*	3*s*	Command-driven
Analog output	4	32μs	47.7μs	Periodic
Position control	5	41.4μs	288.4μs	FIFO-driven
Analog input	9	340.8μs	*None*	Periodic
Update	10	42.5μs	263.7μs	Periodic

表 4-2-12 中，在 LART 平台上並未進行外部感應器之取樣動作。由表中的數據，可以明顯看出 StrongARM 在進行數學浮點運算的工作中，皆較 x86 平台要花費更多時間。這是因為 StrongARM 並不具硬體之浮點運算單元，其浮點運算處理

的方式是採用軟體模擬來代替，故大大增加工作的執行時間。其中，就 Inverse kinematic 工作與週期性 Analog output 工作而言，由於前者之執行優先權大於後者，因此當這兩件工作在同一時間點執行時，在 StrongARM 平台上之 Analog output 工作會有一段明顯的 472μs 延遲。而在 x86 平台上的延遲 23.8μs 相對於 1ms 之週期時間，是接近可忽略的。至於整個即時控制系統所採用的計時器之 Tick 週期為 1ms，其代表的是系統每隔 1ms 即會檢視是否有外部中斷發生或著是週期性工作是否即將執行，並依造情況進行中斷分派和排程動作，以喚醒工作執行，並迅速地做出反應。其相較於標準 Linux 而言，Linux 的 Tick 週期會視系統受負載的程度在 1ms ~ 100ms 之間變動。因此，整個即時控制系統不僅提高了控制系統的反應能力，也增加了週期性工作排程的精確度。

整個嵌入式即時控制系統之系統資訊監測介面，如 4-2-27 所示。其監測的內容包括平台的姿態動作資料、致動器的伸長量情況、D/A 控制電壓訊息、A/D 回授電壓訊息、即時工作的執行時間與最大執行時間以及系統目前的 Tick 週期與週期性工作的工作週期資訊。

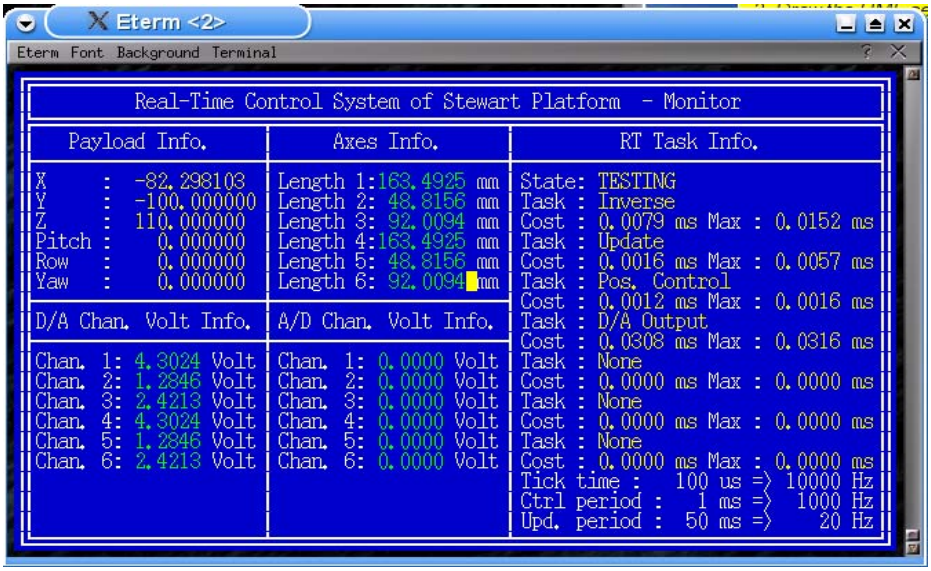


圖 4-2-27　監測系統介面

子題五：高階分散式網路架構建立與應用

由於虛擬實境為了擬真，需要大量的多邊形來模擬物體，需要消耗龐大的計算時間，縱使目前市面上已有多款影像加速卡可用來加快立體圖形的顯像工作，但是模擬迴圈倘若必須處理很多工作，模擬程式的執行效率依舊顯得不足。為解決上述龐大的計算量，本子題主要在探討如何串接整個系統的網路溝通模式，以解決計算負載過重、運算以及傳輸速度延遲的問題，因此本子題採用美國國防部所提出高階分散式網路架構（High-Level Architecture，HLA）之執行架構（Run-Time Infrastructure，RTI）作為底層通訊架構，並且實際以數台個人電腦構成模擬器電腦系統，運用平行運算的方式促使整個模擬器電腦系統達到即時的要求（顯像頻率為每秒 20 至 30 個畫面），進而符合美國飛航單位(Federal Aviation Administration, FAA)標準，本子題以 HLA 技術為探討對象，逐步對規格、內容定義，並提出程序步驟來輔助設計、驗證，以建構高階模擬環境。最後，本子題為了實現多人、多機型混和模擬訓練系統，建立一飛機資料庫系統使得整個虛擬實境可提供多種的飛機選擇彈性，更進一步探討網路效能分析，經由實驗測試結果發現以 HLA 技術為架構之網路傳輸效能遠比傳統常用之網路傳輸效能還要好很多。

5.1 高階分散式網路架構之簡介

高階分散式網路架構（High-Level Architecture，HLA）針對模擬訓練而制訂，提供給開發者一個即時而低流量的模擬環境，以下分就「設計動機」、「基本組成」、「歷史背景」、「開發工具」、「專有名詞」加以說明。

5.1.1 設計動機

制訂HLA的最主要動機是希望能為各種模擬平台提供一套共通的網路架構標準。美國國防部（Department of Defense，DOD）模式與模擬局（DMSO）打從1994年起，分四個時期推廣高階分散式網路架構，以延續以前模擬網路（Networking Simulation，SIMNET）以及分散式互動模擬（Distributed Interactive Simulation，DIS）的發展。關於美國國防部為何要發展HLA的動機曾於"High Level Architecture for Simulation"一文中提及【5-1】

單一模擬不可能滿足所有的應用與不同的使用者

5.1.2 基本組成

HLA由三個部分組成，分別為「HLA介面定義（HLA Interface Specification）」、「HLA物件模型範本（HLA Object Model Template）」、「HLA規則（HLA Rule）」。其中，HLA介面定義（HLA Interface Specification）描述由執行時資訊結構（Runtime Infrastructure，RTI）所提供的服務，並確認每一個 Federate 都有提供 Callback Function。而HLA物件模型範本（HLA Object Model Template）定義了Federation / Simulation / Management Object Model的格式，開發者可運用這些共通的物件模型來記錄資訊。

5.1.3 歷史背景

HLA 算是一個還在發展中的網路架構，是由美國模式與模擬局（DMSO）所發起，其發展歷史年表如下所列：

(1) 1994 年美國先進國防科技研究署（Defense Advanced Research Project Agency，DARPA）與業界簽署三項定義HLA的合約。
(2) 1995 年 3 月 31 日由美國模式與模擬局（DMSO）贊助，Architecture Management Group發表HLA的基本定義草稿。
(3) 1996年三月－與IEEE DIS Workshop合作，成為IEEE標準。
(4) 1996年八月－完成HLA的基本定義標準。
(5) 1996年十月－HLA被美國國防部認可為模式模擬的標準架構。

5.1.4 開發工具

「工欲善其事，必先利其器」，藉著優良的輔助工具，可讓程式分析與設計的難度大幅降低，本子題所採用的HLA開發工具如下所列舉：

(1) RTI 函示庫：

DMSO（Defense Modeling and Simulation Office）網頁上有提供一套免費的 RTI 函示庫，但必須提出申請才能下載，支援 Windows、Linux、IRIX。

(2) OMTD 物件模型開發工具：

DMSO（Defense Modeling and Simulation Office）網頁上有提供一套免費的物件模型開發工具，但必須提出使用申請，僅支援 Windows。其功用是提供一個視窗化的介面，讓開發者可以輕鬆地設計 HLA 所定義的物件模型。

(3) 程式編譯器：

美國國防部所提供的 RTI 函示庫僅支援特定的編譯環境，在 windows 環境下，須使用 Microsoft Visual C，而在 Linux 環境下則使用 GNU C Compiler（GCC）。

(4) Rational Rose 2000：

Rational Rose 公司有一套 UML 物件導向開發工具，可先用圖形的方式設計程式的流程、結構、使用者案例...等。本研究在程式分析初期便是使用此軟體協助設計相關的物件導向程式類別。

5.1.5 專有名詞

由於 HLA 是由美國國防部所制訂的標準，因此很多軍事用語或專有名詞可能不是那麼容易理解，經翻譯之後也很難貼切地表示其原始的意義，故在此簡介幾個接下來會在不斷出現的專有名詞。

(1) Federation

Federation，名詞，中譯為『聯盟』，包含三部分：Federate、Runtime Infrastructure （RTI）、Runtime Interface，是 HLA 所定義的最大模擬單位，指由眾多模擬程式所組成的集合體。其成員介紹如下，其組成如圖 5-1-1 所示。

(2) Federate

Federate 原意若作形容詞中譯為『聯合的』；及物動詞中譯為『使...聯合』。但在 HLA 的用語中，為名詞，代表 Federation 的成員，泛指任何一種形式的模擬程式，倘若依照種類可細分為：電腦模擬（Computer Simulation）、載人模擬器（Manned Simulator）、資料監控程式（Passive Viewer）、資料蒐集程式（Data Collector）、操作介面（An Interface to Live Player）五種。

(3) Runtime Infrastructure（RTI）【5-2】

Runtime Infrastructure，中譯為『執行時資訊結構』，縮寫為 RTI。根據 RTI 官方文件的解釋，RTI 是一種分散式作業系統（Distributed Operating System），而所有 Federate 的資訊傳遞都必須透過 RTI 協助。RTI 定義了許多函示和服務。

(4) Runtime Interface

Runtime Interface 中譯為『執行時介面』，定義 Federate 與 RTI 間的溝通介面。

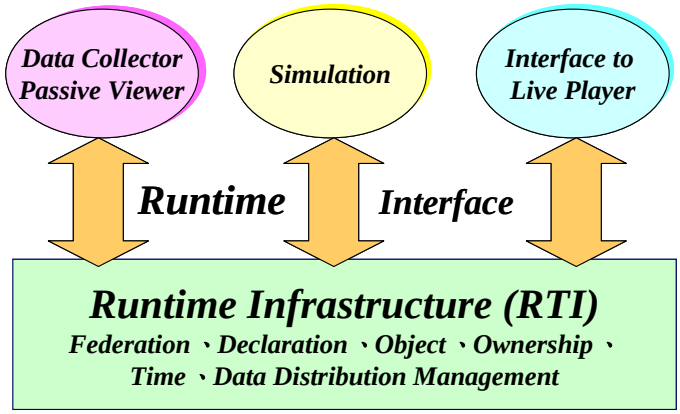


圖 5-1-1　**Federation 的基本組成** 【5-2】

5.2 Runtime Infrastructure（RTI）

RTI 提供給模擬系統一些共通的服務，實現 HLA 介面定義（Interface Specification），是一個鼓勵可攜性與共用性的基礎架構，以下分就基本組成、多人問題及六大管理服務加以說明。

5.2.1 RTI 的基本組成

RTI 主要由「RTI Excutive Process」、「Federation Excutive Process」、「libRTI Library」三大元素組成，逐一說明如下：

(1) RTI Excutive Process（RtiExec）

RtiExec 負責管理 Federation 的生成與消滅，並維護在網路上每一個 Federation 所使用的名稱不重複，每一個執行中的 Federation 都有一個 FedExec 程序協助 RtiExec 進行名稱識別。此外，RtiExec 還必須協助將想要登入之 Federate 導入正確的 FedExec。

(2) Federation Excutive Process（FedExec）

FedExec 負責管理在同一個 Federation 中，多個 Federate 的登入（join）和登出（Resign）情形，並協助 Federate 間的資料交換。每個登入 Federation 的 Federate 都會被 FedExec 指派一個唯一的處理器（Federation Wide Unique Handle）。

FedExec 的生命週期分成六個部分：Federation Management、Declaration Management、Object Management、Data Distribution Management、Time Management、Ownership Management，是 RTI 最重要的部分。

(3) libRTI Library

libRTI 負責為 Federate 提供 HLA 的各種服務，使之能夠與 RtiExec、FedExec 和其他 Federate 溝通。libRTI 中的 RTIambassador 類別包含 RTI 所提供之服務，Federate 的所有請求都必須透過 RTIambassador 的函數來呼叫。而 libRTI 中的抽象類別 FederateAmbassador 用來識別每個 Federate 負責提供的

Callback 函數。由於 FederateAmbassador 是抽象的，所以每個 Federate 都必須重新實作（implement）FederateAmbassador 所定義的每個函數。由於 RTI 對 Federate 的回應是非同步的，Federate 必須預留一個管道供 RTI 呼叫，此管道即為 Callback 函數。如此一來，Federate 不用一直等候 RTI 服務回覆，進而節省等待時間來作更多運算，因此使用 HLA 架構可以讓多人模擬的執行更即時而不受網路環境的優劣影響。

5.2.2 多人模擬的困難

多人模擬最主要的兩件事情便是保持『資料一致性』和『時間一致性』。由於每個參與模擬的程式都有自己的屬性，它必須讓公開部分資訊給其他參與模擬的程式，例如它的座標位置、外觀...等，這樣其他模擬程式才能正確地顯示它，此即『資料一致性』。由於每部電腦的快慢不一，作業系統的執行效能亦不相同，本機的時間也有可能與其他電腦不同，故每一部參與模擬的電腦須互相制訂一個標準的時間作為同步動作的依據，此即『時間一致性』。

就多人模擬實作上會遇到以下幾個問題，RTI 提供了解決的辦法，其對應為：

(1) 管理登入登出 ： Federation Management
(2) 制訂標準時間 ： Time Management
(3) 定義傳輸格式 ： Declaration Management
(4) 交換共用資料 ： Object Management
(5) 有效發送資料 ： Data Distribution Management
(6) 控制權的移轉 ： Ownership Management

以下分六小節介紹這六種管理服務，這六種管理服務在 Federate 與 Federation 之間的互動，扮演很重要的角色，圖 5-2-1 為 Federate 與 Federation 間的互動一覽。

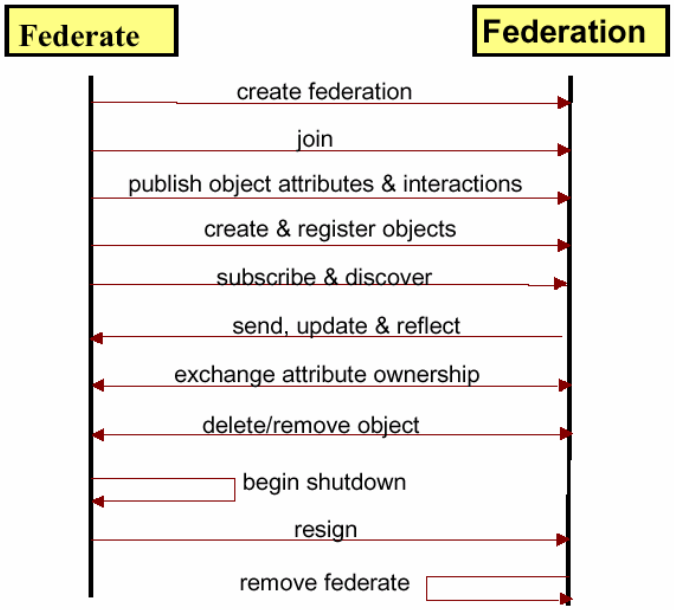


圖 5-2-1 Federate 與 Federation 間的互動 【5-2】

5.2.3 Federation Management

Federation Management 負責管理 Federation 的執行，以及初始化 name space、transportation、ordering defaults、routing spaces 等工作，其功能包括：

(1) 建立 Federation
(2) 將 Federate 登入 Federation
(3) 觀察整個 Federation 的同步點（Synchronization point）
(4) 達成整個 Federation 的儲存（Save）和還原（Restore）
(5) 將 Federate 登出 Federation
(6) 摧毀 Federation

Federation Management 主要是由四個 RTI 函數控制，相關說明如下：

(1) 呼叫 createFederationExecution()函數時，若指定名稱的 Federation 不存在，則由 RtiExec 產生一個新的 FedExec；若指定名稱的 Federation 已存在，則會觸發 FederationExecutionAlreadyExists 的例外（Exception）。
(2) 呼叫 joinFederationExecution()可將 Federate 登入指定名稱的 Federation。
(3) 呼叫 resignFederationExecution()會將 Federate 自 Federation 中登出。
(4) 呼叫 destroyFederationExecution()將終止 Federation 的執行，倘若呼叫此函數的 Federate 不是最後一個登出的，則會產生一個 FederatesCurrentlyJoined 的例外。

5.2.4 Time Management

在介紹 Time Management 之前，必須先瞭解 HLA 所提供的時間管理策略（Policy）。一般來說，Federate 的時間策略可以分為四種：「標準的（regulating）」、「受限的（constrained）」、「既是標準的也是受限的」、「既非標準的也非受限的」。預設值為「既非標準的也非受限的」。宣告自己為「regulating」的 Federate 能夠依照時間戳記（Time Stamp）順序產生事件（Time-stamp-ordered Event，TSO Event）；而未宣告為「regulating」的 Federate 雖然也能產生事件，但與時間無關，稱之「Receive-Ordered Event」。宣告自己為「constrained」的 Federate 有能力接收 TSO Event；而沒有宣告為「constrained」的 Federate 雖然也有能力接收事件，但僅限於 Receive-Ordered Event。Federate 可以隨時變更其狀態為「regulating」或「non-regulating」，「constrained」或「non-constrained」。Federation 可以由任意時間策略組合的 Federate 構成。

RTI 提供最佳化的時間管理服務來協調 Federate 之間的事件（Event）交換，並依循事件的時間戳記協助確保事件的因果關係。RTI 協助 Federate 之間依照不同的策略進行溝通，當有能力傳達 TSO Event 和沒有能力傳達 TSO Event 的子聯盟進行資料交換時，根據最小公因數的原理，事件都是以 Receive Ordered 被傳遞。

Time Management 負責控制每個 Federate 沿著 Federation 時間軸的時間增加量，並配合 Object Management 服務，將訊息依照因果關係，有順序地傳送，其職責有四：

(1) 產生一個事件並加入 federate 的時間當作事件的時間戳記。
(2) 根據 Federate 的時間機制，調整互動、更新屬性、反映或移除物件。
(3) 在 Federation 中支援具因果關係的訊息傳遞行為。
(4) 支援應用不同的時間機制於 Federate 間的互動。

5.2.5 Declaration Management

Declaration Management 負責標明 Federate 將傳送或接收的資料型態，並根據需求，定義哪些資料是需要公開的。在 HLA 的定義之下，所公開的資料又可細分為物件類別（Object Class）和互動類別（Interaction Class）兩種，其中，物件類別由屬性（Attribute）組成，互動類別有參數（Parameter）組成。一個物件的屬性可由多個 Federate 負責提供，每個 Federate 必須宣告可以提供哪些屬性。此外，互動類別的參數必須遵守『一次給足（All or Nothing）』的原則，因此要產生該互動物件的 Federate 必須擁有這些參數資料。

5.2.6 Object Management

Object Management 和 Declaration Management 相互搭配，用來維護『資料一致性』，其職責有五：「新增、修改、移除物件和互動」、「管理物件的認證」、「協助物件的註冊與散佈」、「協調 Federate 之間的屬性更新」、「協助進行不同的傳輸和時間管理機制」。其功能主要為「註冊物件」、「更新屬性」、「傳送互動」、「刪除物件」與「更換傳輸策略」。

5.2.7 Ownership Management

Ownership Management 負責支援物件屬性所有權的移轉，提供「釋放（Push）」和「取得（Pull）」兩種基本移轉機制。其功能包括：「剝奪所有權（Divest）」、「取得所有權（Acquire）」、「詢問所有權（Query）」。在一些限制下，RTI 允許共同分擔更新屬性及刪除物件的權責，但每一個屬性只能有一個 Federate 擁有更新的權責，而且只有一個 Federate 能擁有移除物件的特權。

5.2.8 Data Distribution Management

Data Distribution Management 定義了一個 Routing Space，用來紀錄資料傳遞路線。Routing Space 是由更新區域（Region）所組成，屬性及互動都能定義一個相對應的更新區域，當兩個更新區域相互重疊時才進行資料的傳遞，因此 Data Distribution Management 能夠有效率地傳遞資料，標明資料散佈情形，並回報資料傳遞的狀態。其主要功能包括：「建立 Region」、「更動 Region」、「移除 Region」、「註冊 Region」、「控制更新」。

5.3 HLA Object Model Template

HLA 必須以物件導向的方式設計，因此需要物件模型的範本，不過 HLA 並未規定 Federate 內的物件規劃，只要求開放部分訊息以促進模擬重複使用的可能。物件模型的必備資訊有「Object Class Structure Table」、「Object Interaction Talbe」、「Attribute/Parameter Table」、「FOM/SOM Lexicon」，而額外的資訊可包括「Component Structure Table」、「Association Table」、「Object Model Metadata」。又可細分為「Federation Object Model」、「Simulation Object Model」、「Management Object Model」三種，說明如下：

(1) Federation Object Model

Federation Object Model（FOM）描述在 Federation 有哪些分享的物件，其屬性為何，如何互動。每個 Federation 只有一個 FOM，考量 Federate 間的可能爭議，如資料編碼機制。

(2) Simulation Object Model

Simulation Object Model（SOM）將 Federate 描述成一物件，說明擁有哪些屬性，提供哪些功能，以供未來其他 Federation 重複使用。

(3) Management Object Model

Management Object Model（MOM）定義用來管理 Federation 的物件屬性及互動。

5.4 HLA Rule

HLA 規則（HLA Rule）共包含十條規則，用來定義 Federation 組成子項的責任歸屬與彼此的關連性，又分為 Federation Rule 和 Federate Rule 各五條，說明如下：【5-3, 5-4】

(1) Federation Rule

Federation Rule 說明 Federation 之間的關連性，共有五條，大意如下：

(a) Federation 必須根據 HLA OMT 定義 Federation Object Model（FOM）

(b) 所有 FOM 的物件類別必須定義在 Federate，而非 RTI

(c) 執行過程中，所有 FOM 的資料交換必須透過 RTI

(d) 執行過程中，Federate 與 RTI 的溝通，必須符合 HLA Interface Specification

(e) 執行過程中，特定物件的屬性只能歸屬於一個 Federate

(2) Federate Rule

Federate Rule 旨在說明各個 Federate 的責任歸屬，共有五條，大意如下：

(a) Federation 必須根據 HLA OMT 定義 Simulation Object Model（SOM）

(b) Federate 必須根據 SOM 的定義，更新（update）、反映（reflect）屬性（attribute），傳送（send）、接收（receive）互動（interaction）。

(c) 執行過程中，Federate 必須根據 SOM 的定義，移轉（transfer）/接受（accept）屬性的所有權（Ownership）。

(d) 根據 SOM 的定義，Federate 必須有能力更動其更新物件屬性的條件(Condition)。

(e) Federate 必須有能力管理自己的時間（local time），以利於與其他 Federate 進行資料交換。

5.5 UML 之簡介

UML，全名為 Unified Modeling Language，中譯為「統一化模式語言」，是由三位 OOP 大師 Grady Booch，Ivar Jacobson，James Rumbaugh 所共同制訂。目的是希望能用圖形的方式來表達設計的理念。目前 UML 已經通過 OMG（物件管理組織，Object ManagementGroup）的認證，目前工業界已有許多使用案例，相信將很有可能成為未來標準的模式語言。UML 提供數種能方便表示理念的表示法，以下介紹最常用之一的「循序圖」。

當程式涉及多個類別合作，特別是網路連線程式，便可利用「循序圖」來說明類別甚至物件之間的訊息傳遞。在循序圖中，物件是用一個方格子表示，而在它的下方會有一條垂直的虛線，稱為生命線（Lifeline）。每一個訊息（Message）會連接在兩個物件生命線之間，以帶箭頭的直線表示，而訊息在圖中的上下順序則代表其發生次序。訊息可能包含兩項控制資訊，一是「條件（Condition）」，以中括弧（[...]）表示；另一種是「循環標記（iteration marker）」，代表該訊息會重複送給多個接收物件，以＊表示。而當某個物件完成某項動作，需要「回傳（return）」某個訊息時，則以帶箭頭的虛線表示。圖 5-5-1 為一循序圖的範例。

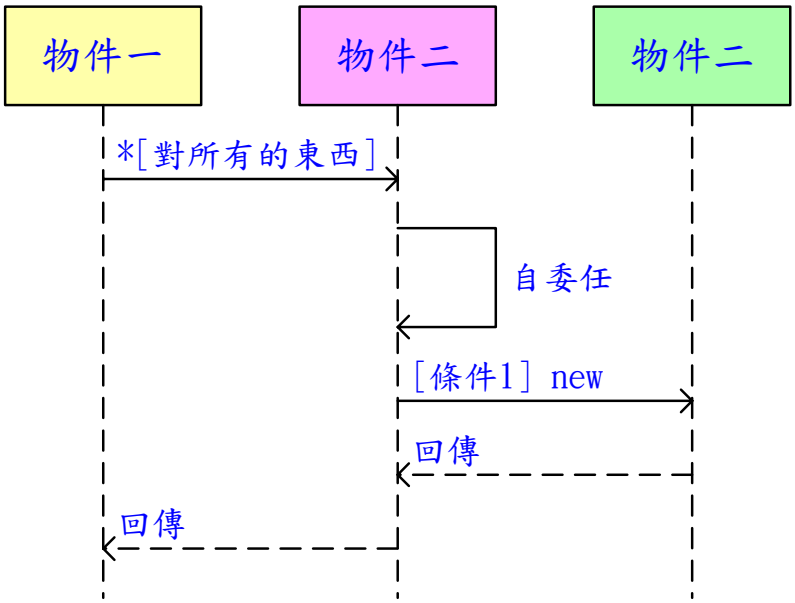


圖 5-5-1　循序圖

5.6 網路效能分析

在 HLA 的架構中，除了一些既有的優點外，在網路傳輸上與一般的通訊協定的差異，也是我們必須要探討的一部分。在網路效能的分析，我們主要探討的是流量的大小，利用 HLA 架構與 UDP 架構的多人模擬飛行來比較。圖 5-6-1 為 HLA 架構雙人飛行模式以十分鐘的時間所得到的效能圖，在圖中所顯示的百分比的部分為利用何種通訊協定傳輸的多寡，在圖右上角的地方為使用何種通訊協定所傳輸的 byte 數，而圖 5-6-2 即為 UDP 架構所得到的效能圖。我們可以很清楚的看到，HLA 的架構中 100%全部是 TCP 的通訊協定，原因在於 HLA 是用 TCP 的協定來模仿 UDP 的廣播功能，而 UDP 的架構有 UDP 及 ICMP 的協定。而在傳輸流量大小的方面，HLA 為 343,288 bytes ，而 UDP 為 4,981,624 bytes，HLA 明顯的小了 14 倍多，可見運用 HLA 的架構很好。以下針對網路流量對不同人數之使用者在飛行模式下做進一步的討論如下

(1) 針對網路流量分析在二人飛行模式下討論，圖 5-6-3 和圖 5-6-4 分別為 HLA 及 UDP 架構二人飛行模式在十分鐘的時間的流量表，以平均值來說，HLA 每秒傳送與接收 6 個封包、4,560 bits，UDP 則為每秒 112 個封包、66,200 bits。以總流量來說，HLA 為 3,523 個封包、343,288 bytes，UDP 為 67,320 個封包、4,981,624 bytes。由分析結果圖可以看得出來 HLA 的效能好很多了。

(2) 針對網路流量分析在三人飛行模式下討論，圖 5-6-5 與圖 5-6-6 分別為 HLA 及 UDP 架構三人飛行模式在十分鐘的時間的流量表，以平均值來說，HLA 每秒傳送與接收 8 個封包、6,200 bits，UDP 則為每秒 114 個封包、103,664 bits。以總流量來說，HLA 為 4,799 個封包、467,940 bytes，UDP 為 68,970 個封包、7,813,532 bytes。UDP 的封包數與雙人模式的差不多，但流量也是變大了。

(3) 針對網路流量分析在四人飛行模式下討論，圖 5-6-7 與圖 5-6-8 分別為 HLA 及 UDP 架構四人飛行模式在十分鐘的時間的流量表，以平均值來說，HLA 每秒傳送與接收 10 個封包、7,760 bits，UDP 則為每秒 124 個封包、172,480 bits。以總流量來說，HLA 為 6,000 個封包、585,000 bytes，UDP 為 74,713 個封包、13,000,449 bytes。

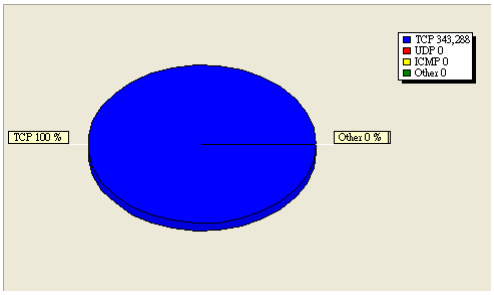


圖 5-6-1　利用 HLA 架構所得之效能圖

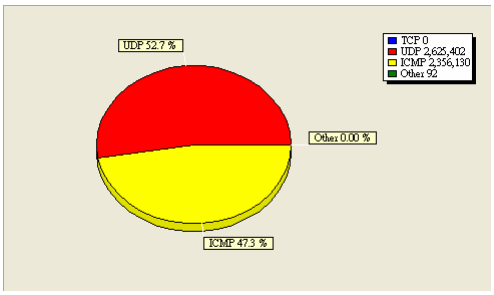


圖 5-6-2　利用 UDP 架構所得之效能圖

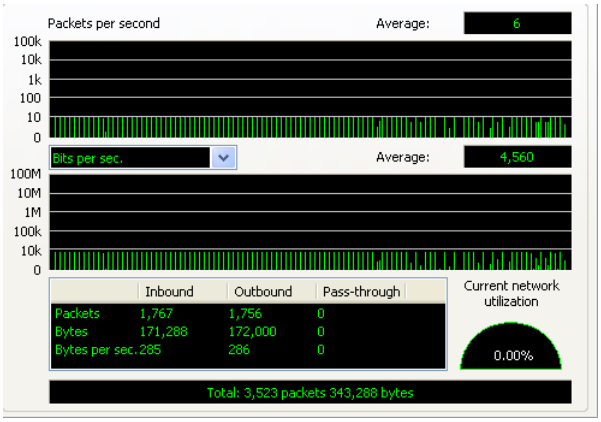


圖 5-6-3　在二人使用情狀下利用 HLA 架構所得之網路流量表

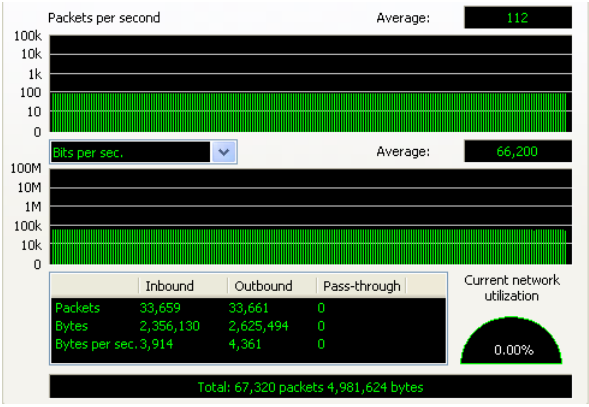


圖 5-6-4　在二人使用情狀下利用 UDP 架構所得之流量表

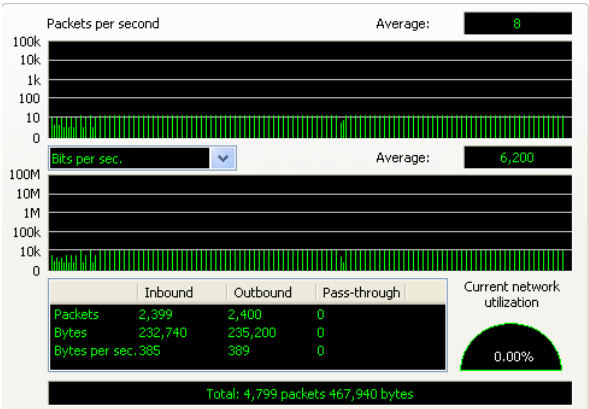


圖 5-6-5　在三人使用情狀下利用 HLA 架構所得之網路流量表

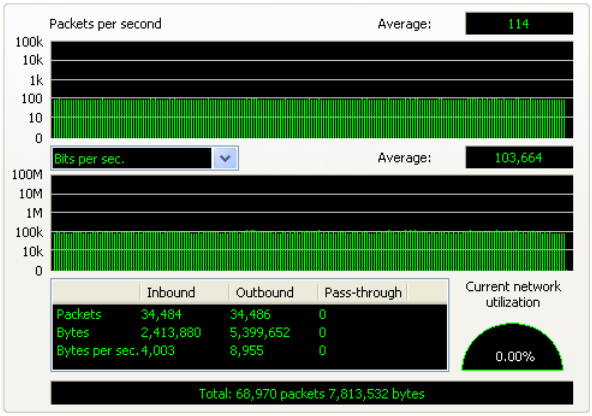


圖 5-6-6　在三人使用情狀下利用 UDP 架構所得之網路流量表

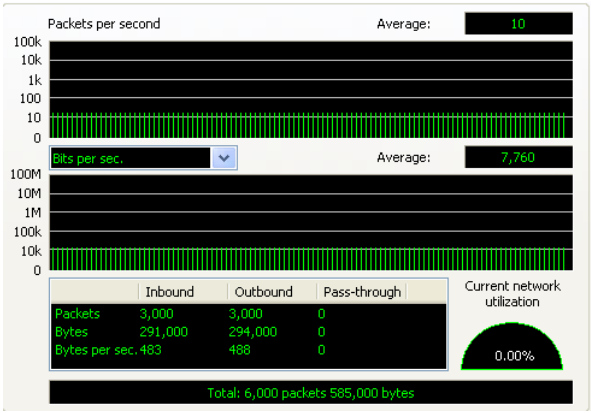


圖 5-6-7　在四人使用情狀下利用 HLA 架構所得之網路流量表

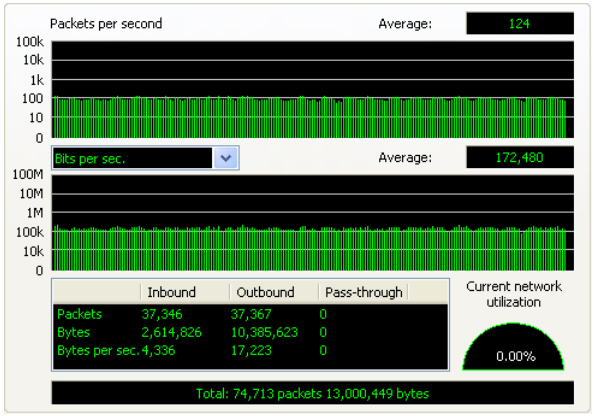


圖 5-6-8　在四人使用情狀下利用 UDP 架構所得之網路流量表

表 5-6-1 為 HLA 以及 UDP 架構分別在二人、三人和四人的飛行模式，測量十分鐘之間利用網路傳輸的封包數量及流量大小。從表中可以看出來，HLA 的架構每增加一人飛行，其封包數量及流量大小會按照一比例而增加；在 UDP 中，每多一人飛行，其封包數量只多了一些，但是在流量方面卻是暴增許多，尤其集中在輸出的部分。HLA 在網路控管方面的優勢，使得其網路效能相較於其他的架構好多了。

網路架構		HLA			UDP		
飛行模式		二人	三人	四人	二人	三人	四人
輸入	Packets	1,767	2,399	3,000	33,659	34,484	37,346
	Bytes	171,288	232,740	291,000	2,356,130	2,413,880	2,614,826
	Bytes per second	285	385	483	3,914	4,003	4,336
輸出	Packets	1,756	2,400	3,000	33,661	34,486	37,367
	Bytes	172,000	235,200	294,000	2,625,494	5,399,652	10,385,623
	Bytes per second	286	389	488	4,361	8,955	17,233
全部	Packets	3,523	4,799	6,000	67,320	68,970	74,713
	Bytes	343,288	467,940	585,000	4,981,624	7,813,532	13,000,449
	Packets per second	6	8	10	112	114	124
	Bits per second	4,560	6,200	7,760	66,200	103,664	172,480

表 5-6-1　HLA 與 UDP 網路架構分析比較表

5.7 飛機資料庫系統

在多人飛行模擬中，僅有單一機型可選擇是不夠的，為了讓整個模擬系統能夠更廣泛地讓使用者來運用，以及為了實現多人、多機型混和模擬訓練情狀，必須建構出一飛機資料庫讓使用者能夠選擇各式各樣的飛行器來參與整個模擬訓練系統。在初步規劃中，我們首先嘗試加入幾款戰機提供選擇，除了先前的完成的訓練模擬器中的飛機外，今年額外新增了國軍 F-16 戰機以及中共的米格-15 可供選擇訓練之用。在圖 5-6-1 中展示了上述幾架飛機的基本選單功能。

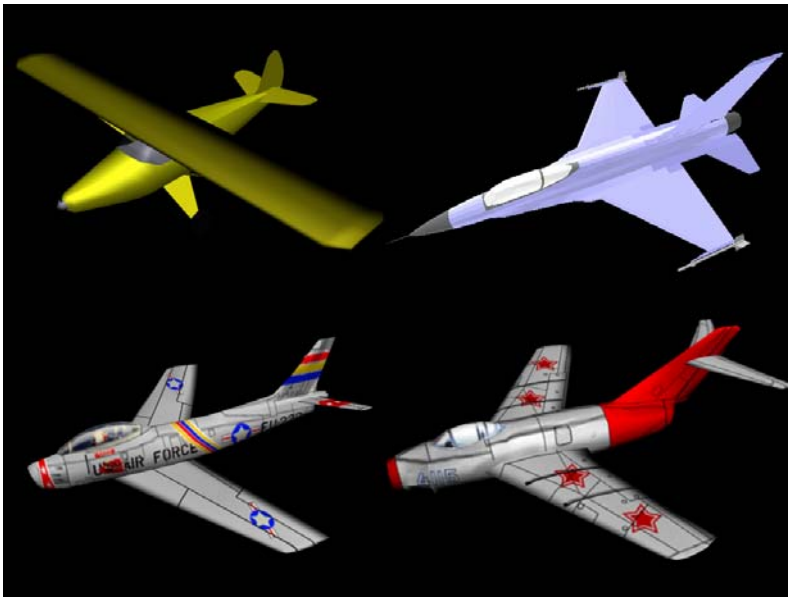


圖 5-7-1　多種飛機選單

5.8 系統整合

本計畫的最終目標是建構一個『多人飛行模擬訓練系統』，用來模擬真實世界中，戰鬥機駕駛訓練所具備的團隊合作與分組對抗機制，並以教官台擔任指揮任務的塔台。以下依系統分析與設計的順序，分就「整合系統」、「多人飛機場景端」、「飛行訓練教官台」及「研究成果」加以說明。

5.8.1 整合系統之分析與設計

最終目標『多人飛行模擬訓練系統』所提到之團隊合作、分組對抗及指揮任務功能，只是一個比較概念性的系統規劃，故進行系統分析前，必須將整合系統的「功能規格」條列清楚，接著以 UML（Unified Modeling Language）表示法的「使用者案例」，開始進行系統分析。本計畫的主軸是整合分散式網路與虛擬實境場景，因此須先確定「程式流程」，再決定「物件模型」，以下逐一說明之。

5.8.1.1 功能規格

本計畫所欲建構之『多人飛行模擬訓練系統』，主要是由三大子系統組成，分別為「多人飛機場景端」、「飛行訓練教官台」及「六軸平台控制端」。飛機場景負責提供駕駛員一個擬真的飛行駕駛訓練環境，讓駕駛員可以透過鍵盤或搖桿操控；而教官台則負責顯示目前參與模擬的飛機位置，並負責發送模擬指示。六軸平台控制端則負責接收飛機場景所送出之姿態，並將姿態呈現在六軸平台上。

本計畫的研究動機源自於整合不同作業系統下的研究成果，因此在挑選分散式網路架構以及虛擬實境場景製作軟體時都有特別考慮過「跨作業系統」的問題。根據實驗室現有的設備，本計畫規劃了如圖 5-8-1 所示之系統架構。其中，「六軸平台控制端」將在 Linux 系統下執行，而其他的「多人飛機場景端」與「飛行訓練教官台」則在 Windows 系統下執行。預估本計畫所設計之整合系統至少會需要四部以上的電腦共同合作，才能達成『多人飛行模擬訓練系統』的基本功能要求。為了證實此系統真的可以多人參與，必須有兩部以上的電腦執行飛機場景，而教官台至少需要一部，若要分組對抗的話，則需要兩部。關於六軸平台控制的部分，由於平台的姿態必須由飛機場景提供，故理論上一部飛機場景就可以搭配一部六軸平台，但實驗室只有一部六軸平台，目前只動用一部電腦來負責六軸平台控制。

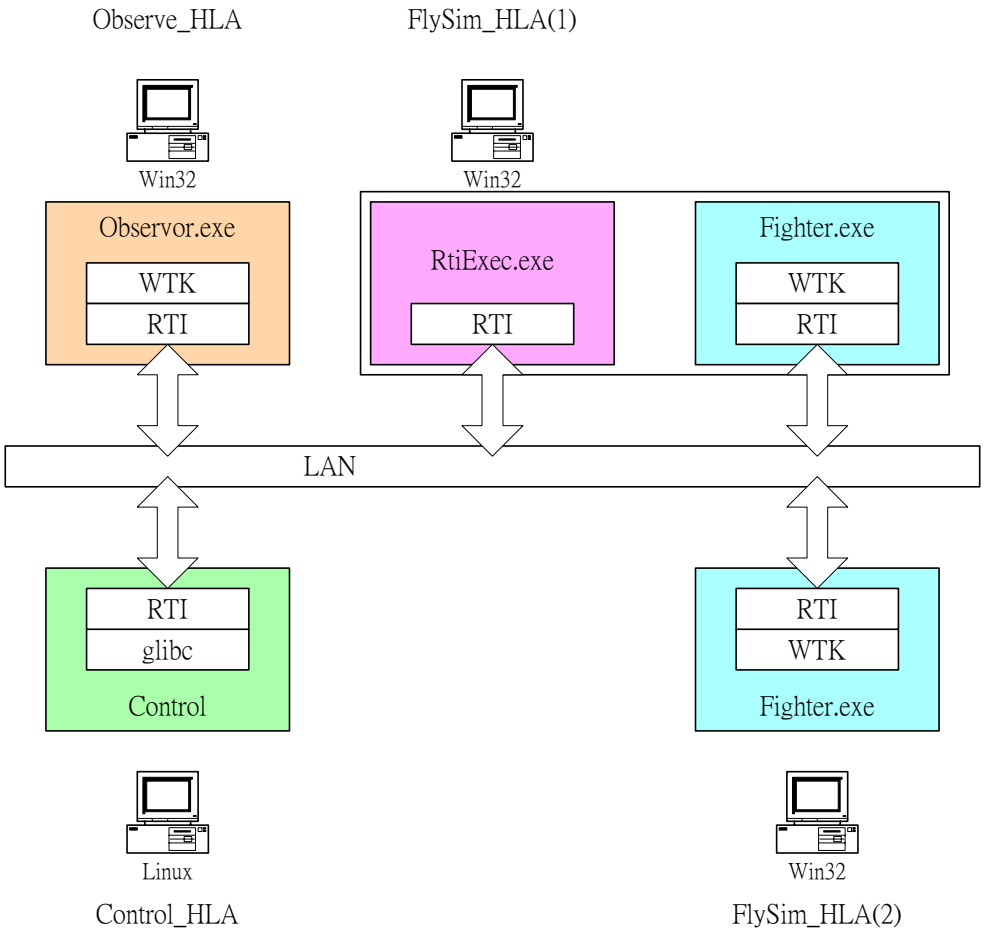


圖 5-8-1　整合系統架構圖

5.8.1.2 使用案例

進一步分析系統規格，可以得到圖 5-8-2 之使用者案例圖。根據使用者案例圖，可以清楚地將三個子系統的細部功能列舉出來：

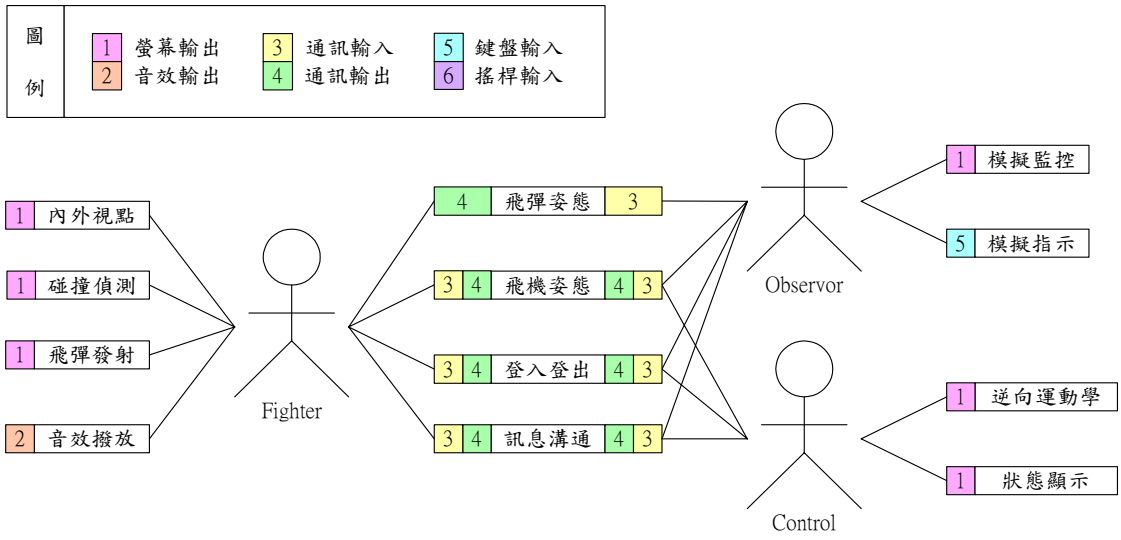


圖 5-8-2　整合系統使用者案例圖

5.8.1.3 程式流程

在決定了系統架構與細部系統功能之後，接著先規劃『多人飛行模擬訓練系統』的程式流程，如圖 5-8-3 所示，數字標號代表執行順序，有星號者（*），代表系統回應。

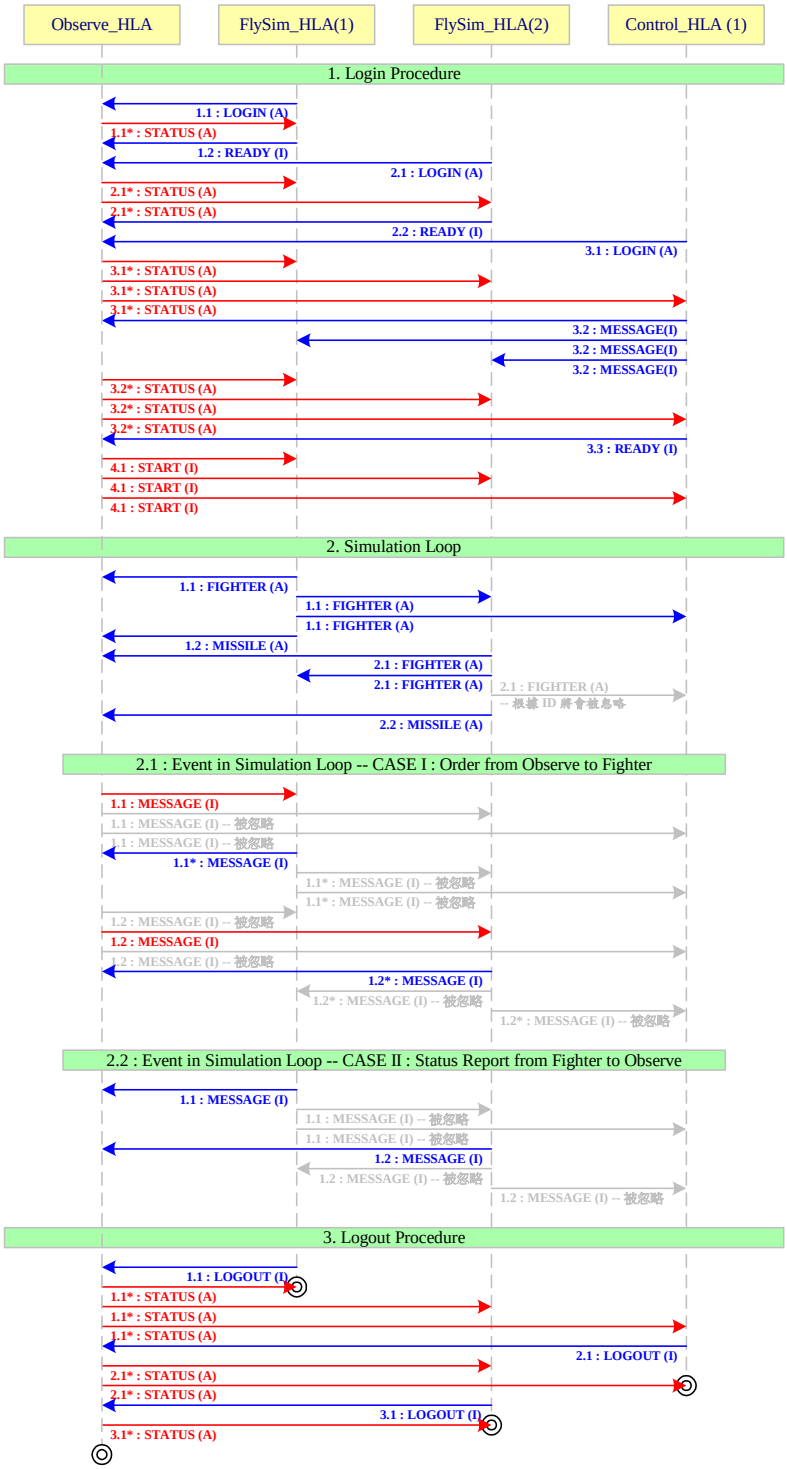


圖 5-8-3　整合系統之模擬流程

5.8.1.4 物件模型

根據程式流程的規劃，接著須確認程式流程中所規範的各種屬性與互動，也就是模擬程式的物件模型【5-5】。如下圖 5-8-4 與 5-8-5 分別為依照 FOM 格式（Federation Object Model 描述在 Federation 有哪些分享的物件，其屬性為何，如何互動。每個 Federation 只有一個 FOM，考量 Federate 間的可能爭議，如資料編碼機制。）定義之互動類別與物件類別。其中，互動類別分別為代表登出的 LOGOUT、代表訊息溝通的 MESSAGE，以及代表完成初始化的 READY。物件類別包含：STATUS 類別用來記錄整合系統的登入情形，LOGIN 物件類別用來紀錄登入的相關資料，而 FIGHTER 物件類別用來紀錄飛機姿態，至於 MISSILE 物件類別則用來記錄飛彈姿態。這都是為了提供『多人飛行模擬訓練系統』能同時由六部飛機連結六部平台進行「團隊合作」與「分組對抗」訓練之用。

Interaction	Parameter
LOGOUT	ID : int
MESSAGE	NUMBER : int
READY	ID : int

圖 5-8-4　FOM 互動類別總覽

STATUS : FOM		
OBS_JOINED : bool		
DYN_JOINED : bool		
FL1_JOINED : bool	FL1_ID : int	FL1_NAME : int
FL2_JOINED : bool	FL2_ID : int	FL2_NAME : int
FL3_JOINED : bool	FL3_ID : int	FL3_NAME : int
FL4_JOINED : bool	FL4_ID : int	FL4_NAME : int
FL5_JOINED : bool	FL5_ID : int	FL5_NAME : int
FL6_JOINED : bool	FL6_ID : int	FL6_NAME : int
CN1_JOINED : bool	CN1_ID : int	FL1_TYPE : int
CN2_JOINED : bool	CN2_ID : int	FL2_TYPE : int
CN3_JOINED : bool	CN3_ID : int	FL3_TYPE : int
CN4_JOINED : bool	CN4_ID : int	FL4_TYPE : int
CN5_JOINED : bool	CN5_ID : int	FL5_TYPE : int
CN6_JOINED : bool	CN6_ID : int	FL6_TYPE : int

LOGIN : FOM
NAME : char*
GROUP : int
TYPE : int
FEDID : int

FIGHTER : FOM
F_ID : int
F_X : float
F_Y : float
F_Z : float
F_PITCH : float
F_ROW : float
F_YAW : float

MISSILE : FOM
M_ID : int
M_X : float
M_Y : float
M_Z : float
M_PITCH : float
M_ROW : float
M_YAW : float

圖 5-8-5 FOM 物件類別總覽

定義完 FOM 之後，接著繼續定義 SOM 模型（Simulation Object Model（SOM）將 Federate 描述成一物件，說明擁有哪些屬性，提供哪些功能，以供未來其他 Federation 重複使用。每個 Federate 只有一個 SOM。），關於登入物件類別的細部定義，如圖 5-8-6 所示，特別定義每個屬性的數值與所代表的意義，是讓教官台能區分該 LOGIN 物件是由哪一種模擬所產生的。

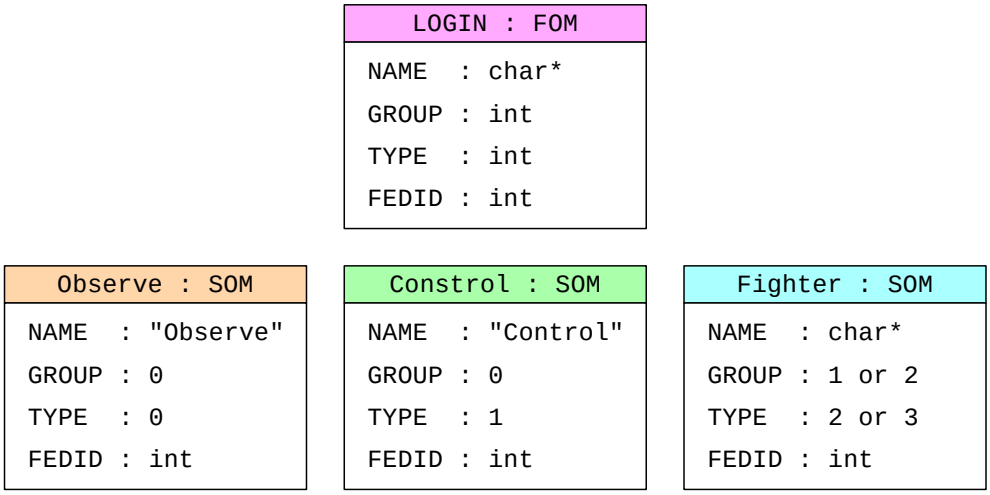


圖 5-8-6 SOM 登入類別欄位定義

最後，還要定義清楚物件屬性與互動的發行與訂閱關係，藉以釐清資料流程是否符合原始定義之程式流程。屬性與互動之發行與訂閱關係如圖 5-8-7 所示，從發行與訂閱的關係，可以找出屬性與互動資料之流向，如圖 5-8-8 所示。

FOM	SOM					
Object Class / Interaction Class	Observe		Control		Fighter	
	發行	訂閱	發行	訂閱	發行	訂閱
FIGHTER	✗	✓	✗	✓	✓	✓
LOGIN	✗	✓	✓	✗	✓	✗
MISSLE	✗	✓	✗	✗	✓	✓
STATUS	✓	✗	✗	✓	✗	✓
LOGOUT	✗	✓	✓	✗	✓	✗
MESSAGE	✓	✓	✓	✓	✓	✓
READY	✗	✓	✓	✗	✓	✗

圖 5-8-7 SOM 登入類別欄位定義

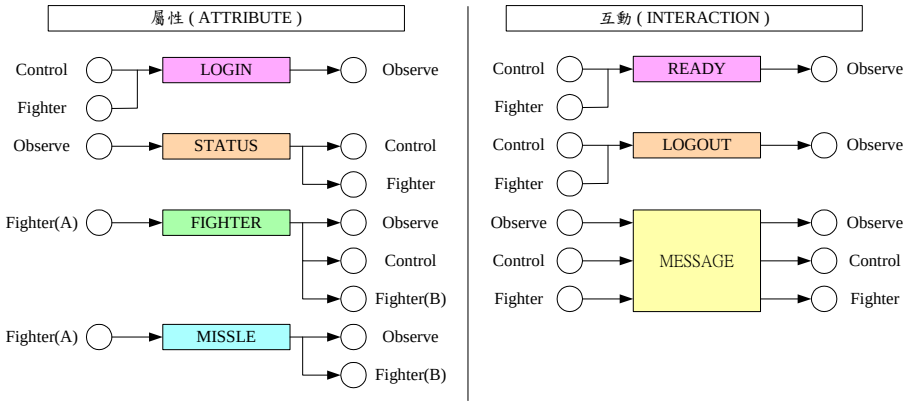


圖 5-8-8 SOM 屬性與互動資料之流向

5.8.2 子系統一：多人飛機場景端

多人飛機場景是『多人飛行模擬訓練系統』中最重要，也是功能最複雜的子系統，因此本節先對多人飛機場景的「場景結構」、「模擬迴圈」及「通訊流程」加以說明。

5.8.2.1 通訊流程

本計畫最重要的工作就是設法將虛擬場景程式與高階分散式網路架構加以整合，為順利完成這項任務，必須先將通訊流程定義清楚，多人飛機場景與其他模擬程式之間的關連性相當高，以下分別就「飛機對飛機」和「飛機對平台」加以說明，至於「飛機對教官台」的通訊流程，將於 5.8.3 節討論。

(1) 飛機場景對飛機場景

飛機場景對飛機場景的流程如圖 5-8-9 與 5-8-10 所示，飛機場景是由三個程式所組成：「FlySim_HLA」、「Fighter」、「FighterAmb」，其中「FlySim_HLA」為主程式，FighterAmb 重新定義 FederateAmbassador，用來提供 Callback 函數（由於 RTI 對 Federate 的回應是非同步的，Federate 必須預留一個管道供 RTI 呼叫，此管道即為 Callback 函數。）的類別，至於「Fighter」類別則是為了讓主程式的流程能比較簡單易懂，並將與 HLA 相關的函數呼叫都寫在「Fighter」類別中，主程式將透過「Fighter」物件呼叫這些函數。

(2) 飛機場景對平台控制

飛機場景對平台控制的通訊流程如圖 5-8-11 所示，平台控制由三個程式組成：「Control_HLA」、「Control」、「ControlAmb」，「Control_HLA」為主程式，ControlAmb 為重新定義 FederateAmbassador，用來提供 Callback 函數的類別，至於「Control」類別則是為了讓主程式的流程能比較簡單易懂，並且將與 HLA 相關的函數呼叫都寫在「Control」類別中，主程式則透過「Control」物件才呼叫這些函數。

圖 5-8-9　Fighter 對 Fighter 的程式流程圖(1)

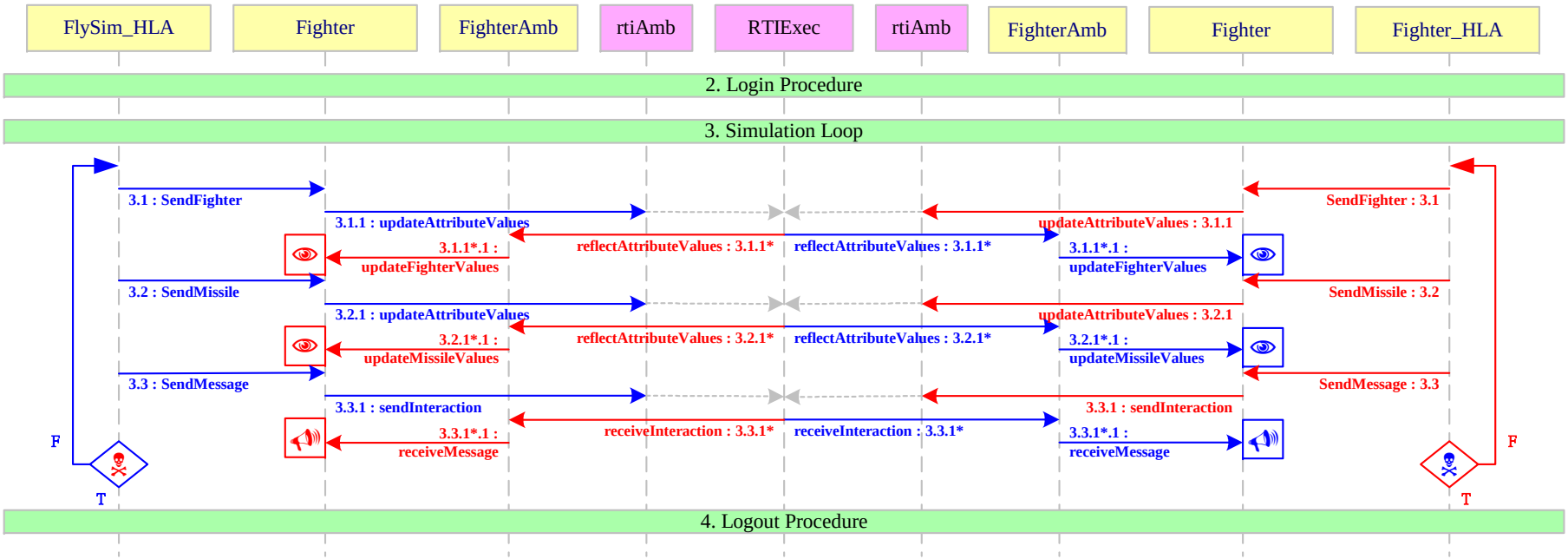


圖 5-8-10 Fighter 對 Fighter 的程式流程圖(2)

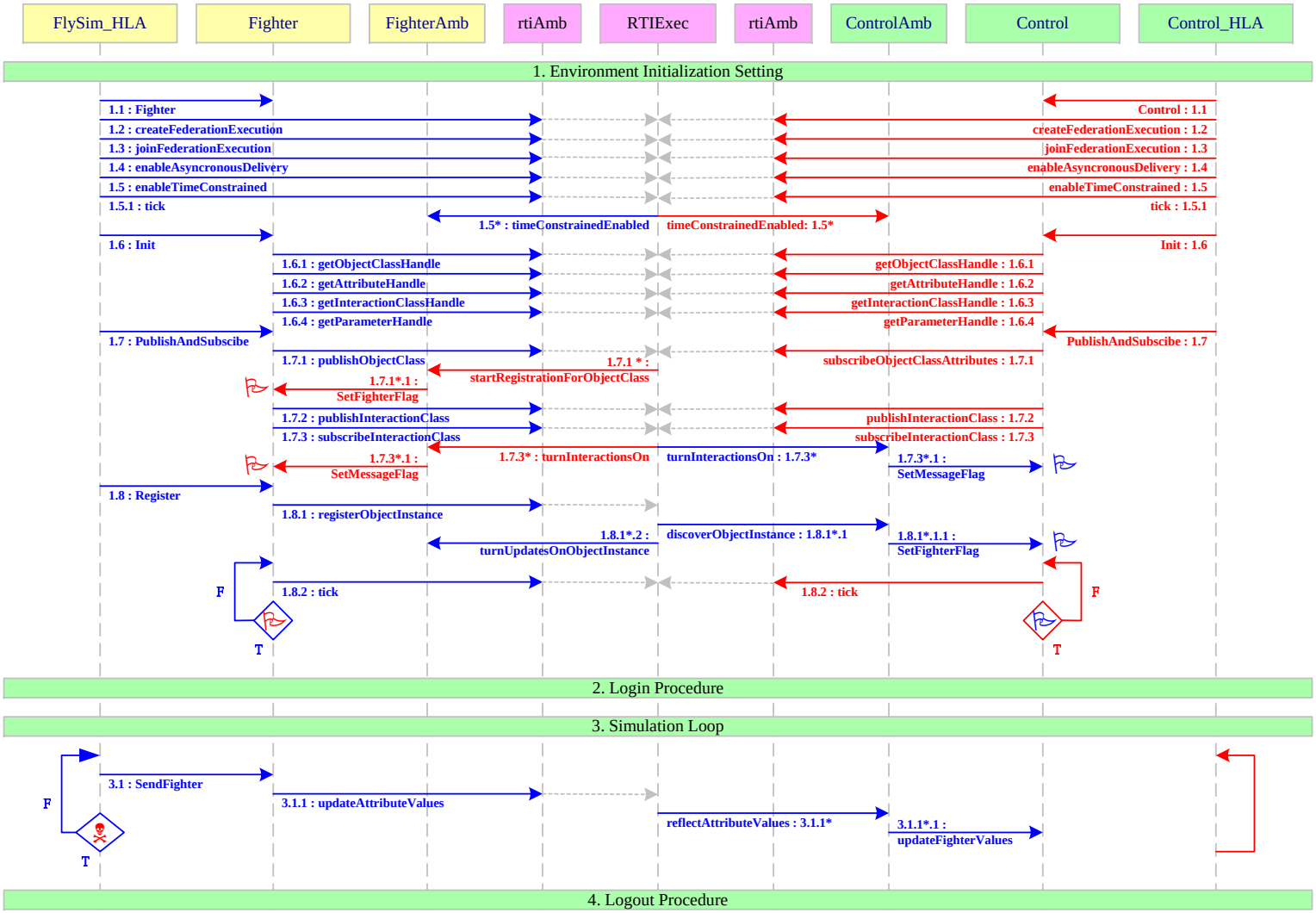


圖 5-8-11 Fighter 對 Control 的程式流程圖

5.8.3 子系統二：飛行訓練教官台

雖然說運用高階分散式網路架構可以讓整個系統比較沒有主從關係，但是為了方便維護目前參與『多人模擬訓練系統』的子系統資訊，本計畫仿照目前飛航模擬訓練器的架構，設計了「飛行訓練教官台」這個子系統，做為管理其他子系統的伺服端（Server），以下分就「場景結構」、「模擬迴圈」及「通訊流程」加以說明。

5.8.3.1 通訊流程

飛行訓練教官台主要負責工作有二：「維護參與模擬的子系統資訊」以及「對飛機場景發送模擬指示」，以下分別就「飛機對教官台」及「教官台對平台」的通訊流程加以說明。

(1) 飛機場景對教官台

飛機場景對教官台的流程如圖 5-8-12、圖 5-8-13 及圖 5-8-14 所示，教官台亦是由三個程式組成：「Observe_HLA」、「Observe」、「ObserveAmb」，其中「Observe_HLA」為主程式，ObserveAmb 為重新定義 FederateAmbassador，用來提供 Callback 函數的類別，至於「Observe」類別則是為了讓主程式的流程能比較簡單易懂，並且將與 HLA 相關的函數呼叫都寫在「Observe」類別中，主程式則透過「Observe」物件來呼叫這些函數。

(2) 教官台對平台控制

教官台對平台控制的流程如圖 5-8-15 與圖 5-8-16 所示。

圖 5-8-12 Fighter 對 Observe 的程式流程圖(1)

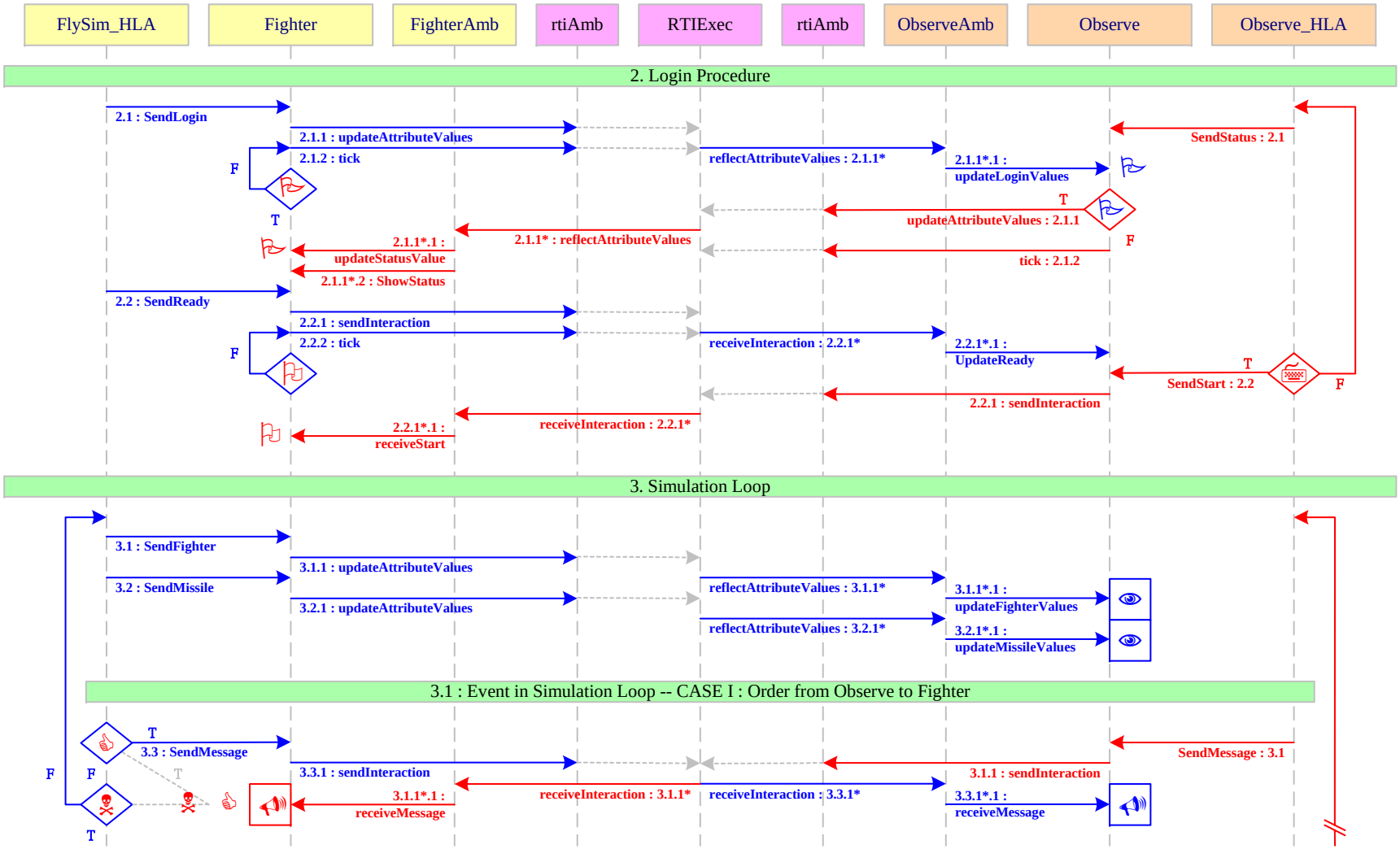


圖 5-8-13 Fighter 對 Observe 的程式流程圖(2)

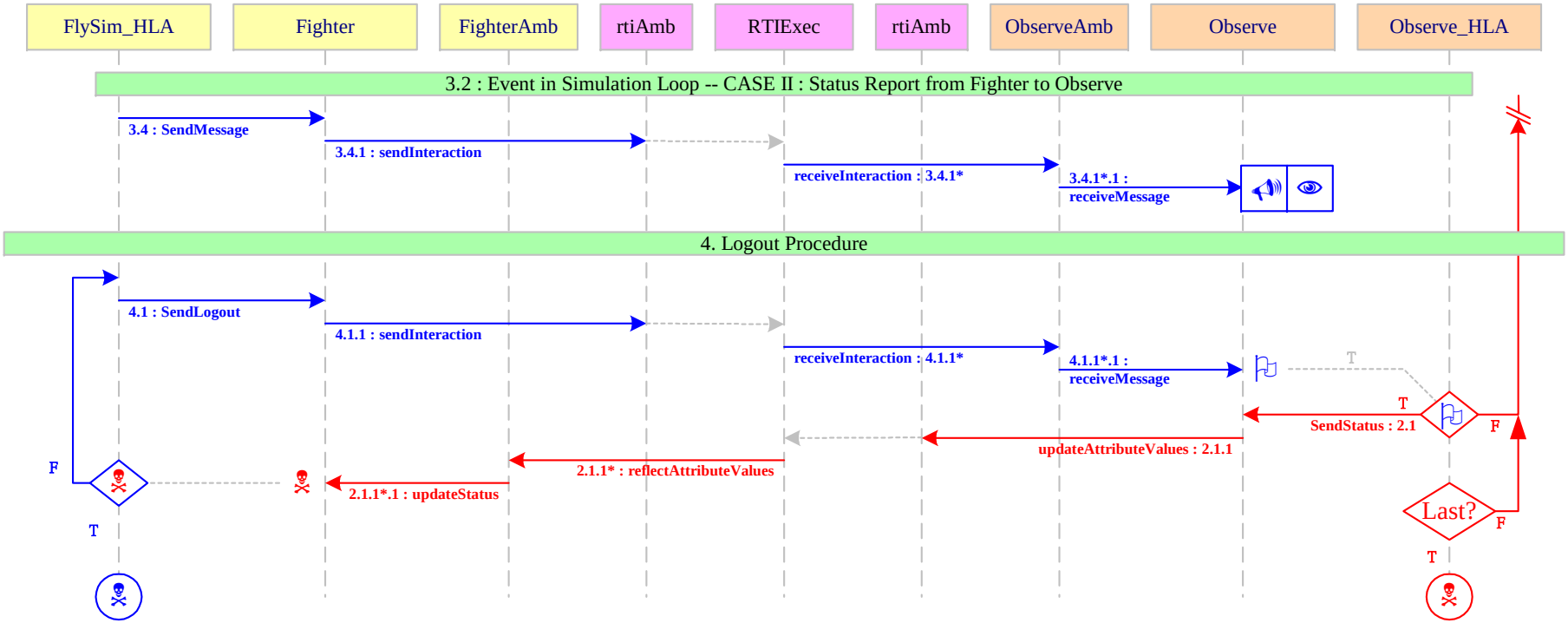


圖 5-8-14 Fighter 對 Observe 的程式流程圖(3)

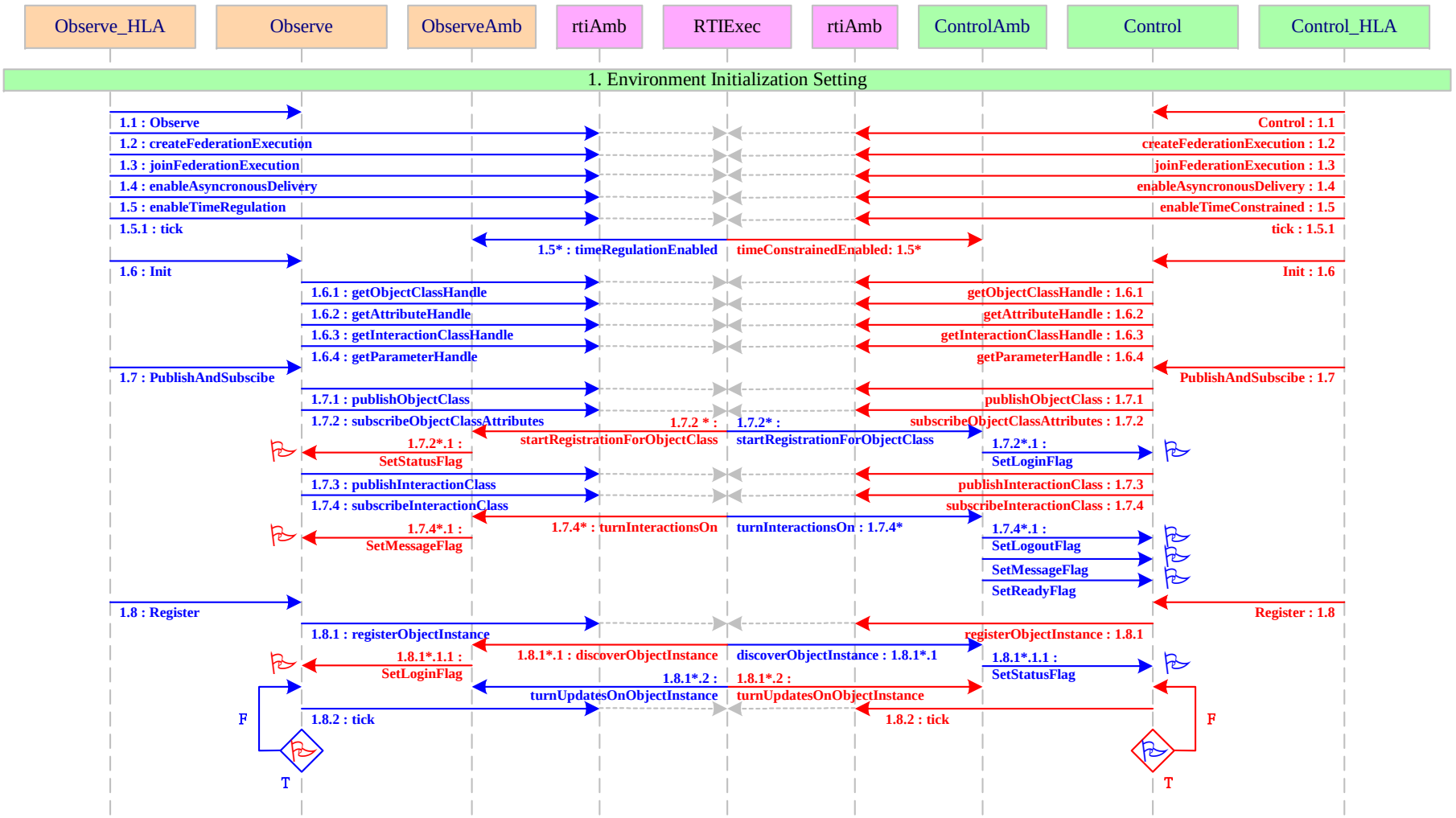


圖 5-8-15 Observe 對 Control 的程式流程圖(1)

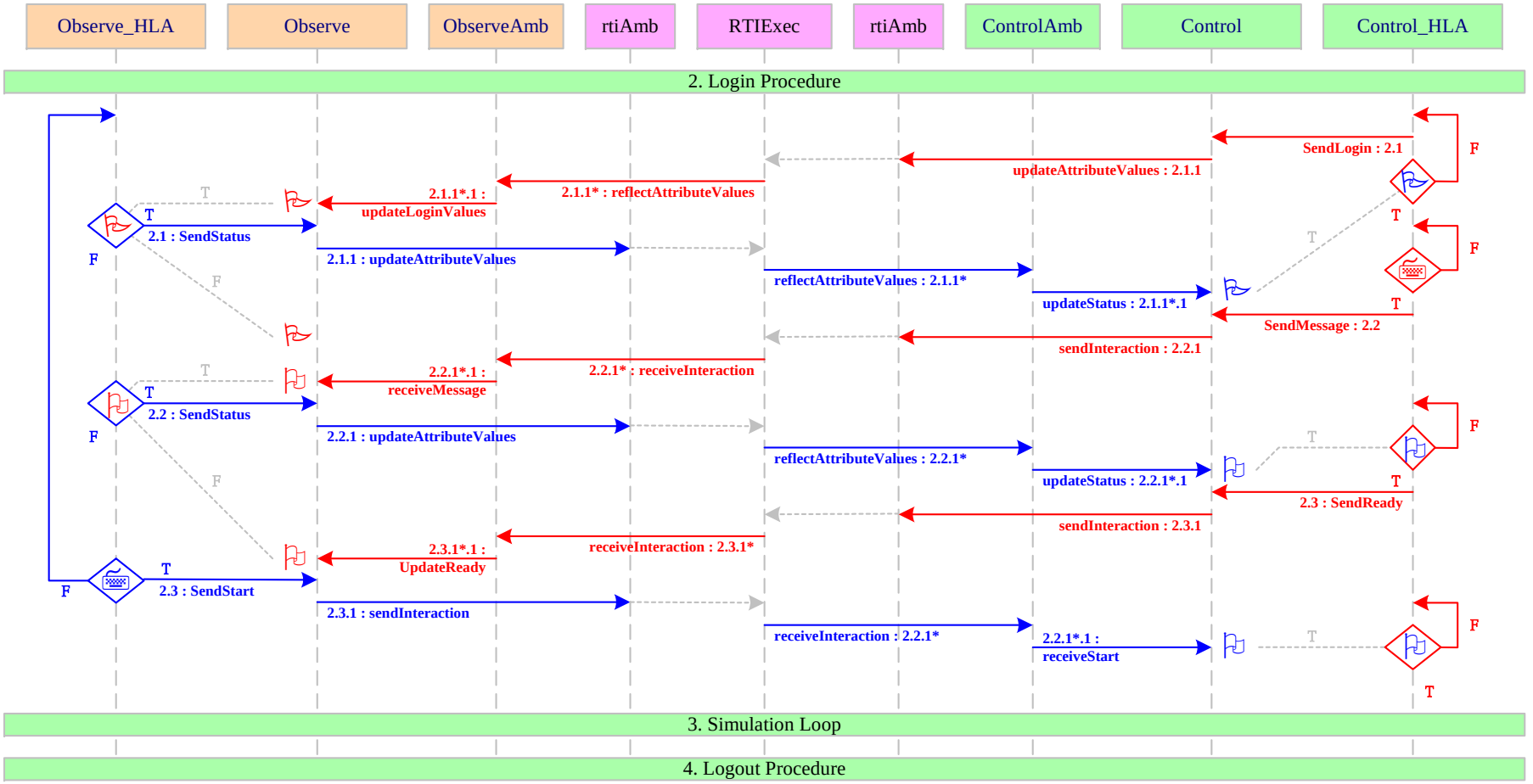


圖 5-8-16 Observe 對 Control 的程式流程圖(2)

5.8.4 研究成果

本子題的研究成果主要包括三個子系統，以下分別就「多人飛機場景端」、「飛行訓練教官台」及「六軸平台控制端」加以說明：

1. 多人飛機場景端

   多人飛行場景端目前已完成之功能如下：

   - 以參數選擇輸入方式、模擬群組、模擬機型、模擬名稱
   - 飛機與地面、飛彈與地面之碰撞偵測
   - 背景音效及飛彈發射音效
   - 簡易版飛機動態程式
   - 飛彈爆炸特效

   其執行畫面如圖 5-8-17 所示。

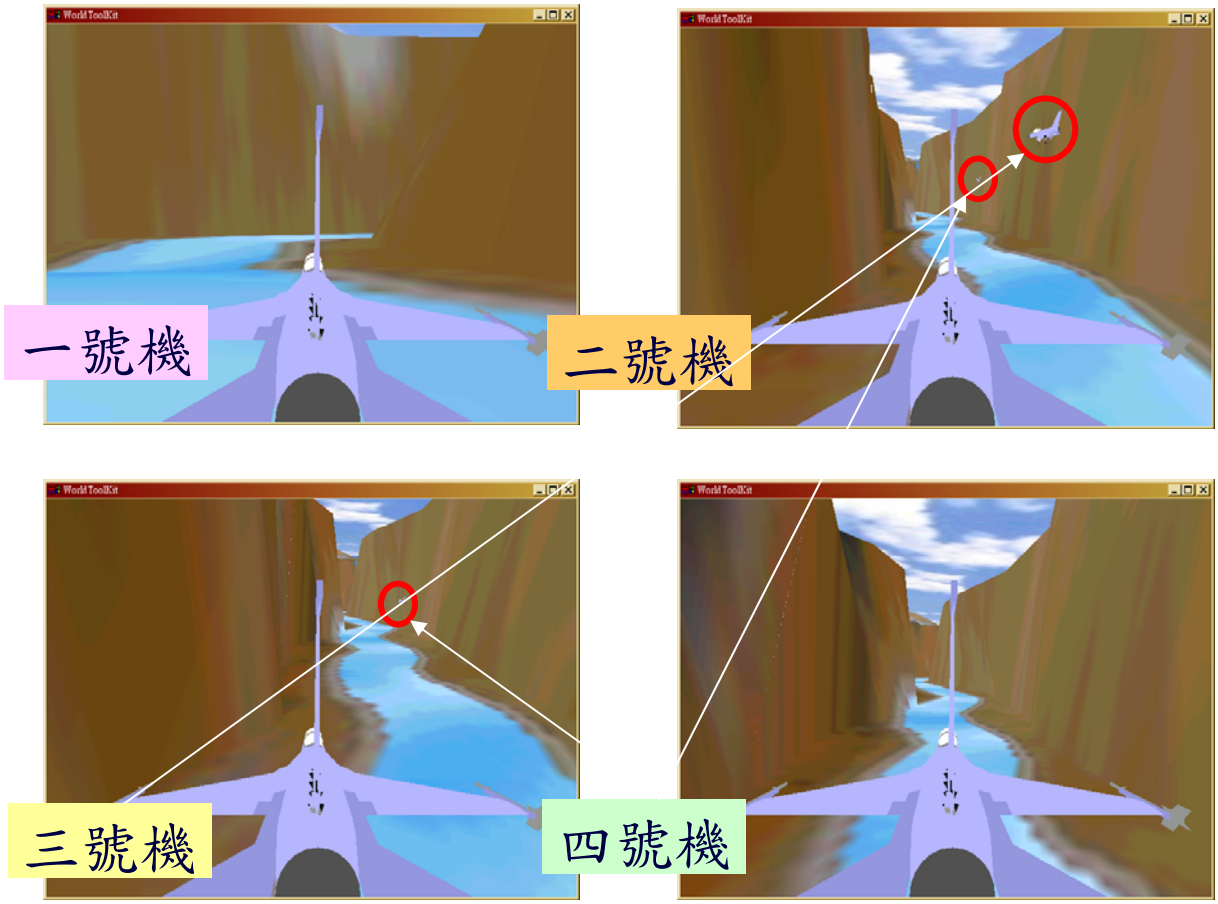


圖 5-8-17 多人飛機場景執行畫面

2. 飛行訓練教官台

飛行訓練教官台目前已完成之功能如下：

- 標示參與模擬的飛機座標位置
- 送出模擬指示

其執行畫面如圖 5-8-18 所示。

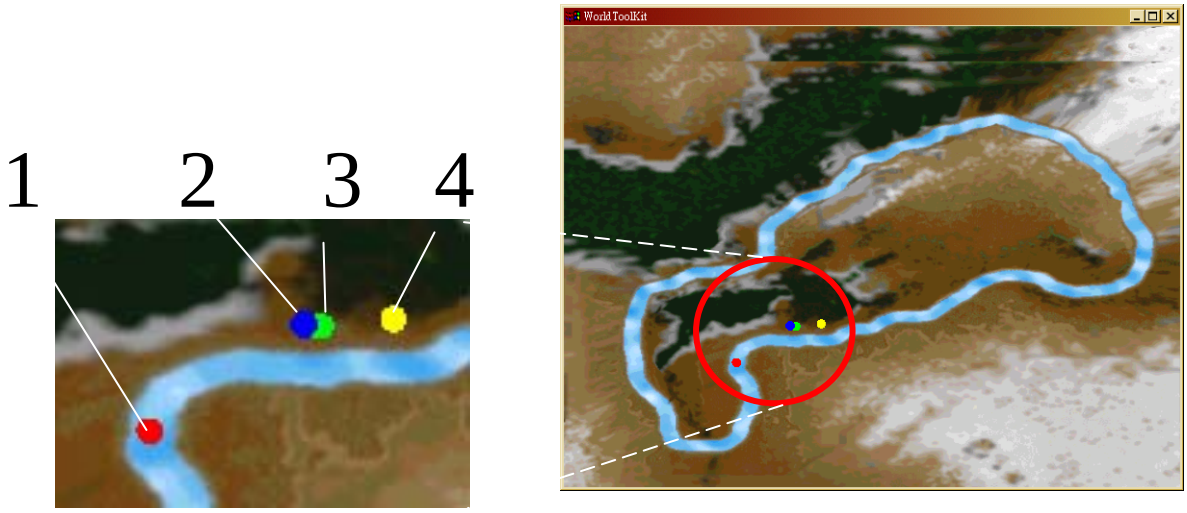


圖 5-8-18 飛行訓練教官台執行畫面

3. 六軸平台控制程式

六軸平台控制程式目前已完成之功能如下：

- 進行逆向運動學運算
- 以硬體計時器（Hardware Timer）與訊號（Signal）模擬中斷服務
- 數位類比轉換卡驅動

其執行畫面如圖 5-8-19 所示。

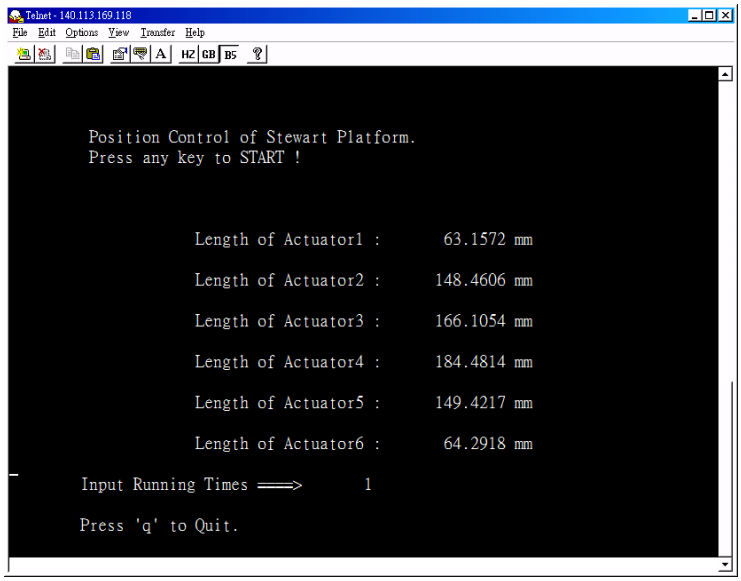


圖 5-8-19 六軸平台控制程式執行畫面

參考文獻

關於子題一部分之參考文獻

【1-1】H. Shiraishi, S. L. Ipri, and D. D. Cho, "CMAC Neural Network Controller for Fuel-injection Systems," *IEEE Trans. Control System Technology*, vol. 3, pp. 32-38, 1995

【1-2】C. T. Chiang, and C. S. Lin, "CMAC with General Basis Functions," *Neural Networks*, vol. 9, pp. 1199-1211, 1996.

【1-3】K. S. Hwang, and C. S. Lin, "Smooth Trajectory Tracking of Three-link Robot: a Self-organizing CMAC Approach," *IEEE Trans. Syst., Man, and Cybern.*, vol. 28, pp. 680-692, 1998.

【1-4】Y. H. Kim, and F. L. Lewis, "Optimal Design of CMAC Neural-network Controller for Robot Manipulators," *IEEE Trans. Syst., Man, and Cybern.*, vol. 30, pp. 22-31, 2000.

【1-5】K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-26, 1990.

【1-6】L. X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1994.

【1-7】B. K. Bose, *Power Electronics and AC Drives*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

【1-8】D. W. Novotny and T. A. Lipo, *Vector Control and Dynamics of AC Drives*, Oxford University Press, New York, 1996.

【1-9】C. M. Liaw and F. J. Lin, "Position Control with Fuzzy Adaptation for Induction Servomotor Drive," *IEE Proc. Electric Power Applications*, vol. 142, pp. 397-404, 1995.

【1-10】J. J. E. Slotine and W. P. Li, *Applied Nonlinear Control*, Englewood Cliffs, NJ: Prentice Hall, 1991.

關於子題二部分之參考文獻

【2-1】R. W. Brumbaugh, "An Aircraft Model for the AIAA Control Challenge," *AIAA* paper 91-2631, August, 1991.

【2-2】B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, Inc., New York, 1992.

【2-3】C. F. Lin, *Modern Navigation, Guidance, and Control Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1991

【2-4】J. R. Koza, *Genetic Programming : On the Programming of Computers by Means of Natural Selection*, MIT Press, U.S.A, 1992.

【2-5】C. T. Lin and C. S. George Lee, *Neural Fuzzy System*, Prentice Hall, Upper Saddle River, New Jersey, USA, 1996.
【2-6】http://www.flightgear.org/
【2-7】OpenGL Architecture Review Board, *OpenGL Programming Guide*, Release 1, Addison-Wesley Publishing Company, 1993.
【2-8】R. Samanta, J. Zheng, T. Funkhouser, K. Li, and J. P. Singh, "Load Balancing for Multi-projector Rendering Systems," *Proceedings of SIGGRAPH /Eurographics Workshop on Graphics Hardware*, pp. 107–116, August 1999.
【2-9】M. Eldridge, H. Igehy, and P. Hanrahan. Pomegranate: A Fully Scalable Graphics Architecture. *Proceedings of SIGGRAPH 2000*, pages 443–454, July 2000.
【2-10】W. Blanke, C. Bajaj, D. Fussel, and X. Zhang, "The Metabuffer: A Scalable Multiresolution Multidisplay 3-D Graphics System Using Commodity Rendering Engines," TR2000-16, University of Texas at Austin, February 2000.
【2-11】G. Humphreys, I. Buck, M. Eldridge, and P. Hanrahan, "Distributed Rendering for Scalable Displays," *IEEE Supercomputing 2000*, October 2000
【2-12】G. D. Padfield, *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling*, Education Series, AIAA, 1995.
【2-13】AVSCOM, *Aeronautical Design Standard (ADC) 33C – Handling Qualities for Military Helicopters*, US Army AVSCOM, 1989.
【2-14】USAF, *Flying Qualities of Piloted Airplanes*, Mil F-8785C, USAF, 1980.
【2-15】Guo-Ying *Chen*, *On the Study of the Learning Performance for Neural Networks and Neural Fuzzy Networks*, Master Thesis, Dept. of Electrical Engineering, NTUST, 1998.
【2-16】A. J. *Calise* and R. T. Rysdyk, "Nonlinear adaptive flight control using neural networks," *IEEE Control Systems*, vol. 18, no. 6, pp.14-25, 1998.
【2-17】T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116-131,1985.
【2-18】M. Sugeno and G. T. Kang, "Fuzzy modeling and control of multilayer incinerator," *Fuzzy Sets and Systems*, vol. 18, pp. 329-346, 1986.
【2-19】G. D. Padfield, *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling*, Education Series, AIAA, 1995.
【2-20】邱岱瑋，"*直昇機*動態模式的建立"，國立成功大學航太所碩士論文，June, 1998.
【2-21】Jane's—All The *World's* Aircraft, Jane's Information Group, 1995.

關於子題三部分之參考文獻

【3-1】B. D. Adelstein and M. J. Rosen, "Design and Implementation of a Force

Reflecting Manipulandum for Manual Control Research," in *Advances in Robotics*, H. Kazerooni, editor, American Society of Mechanical Engineers, New York, pp. 1-12, 1992.

【3-2】R. J. Hosman, B. Benard, and H. Fourquet, "Active and Passive Side-Stick Controllers in Manual Aircraft Control," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 527-529, 1990

【3-3】S. G. Hong, J. J. Lee, and S. Kim, "Generating artificial force for feedback control of teleoperated mobile robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1721–1726, 1999.

【3-4】G. C. Burdea, "Invited review: the synergy between virtual reality and robotics," *IEEE Trans. on Robotics and Automation*, Vol. 15, no. 3, pp. 400-410, 1999

【3-5】J. Beveridge and R. Wiener, *Multithreading Applications in Win32: the Complete Guide to Threads*, Addison-Wesley, 1997.

【3-6】B. D. Adelstein and M. J. Rosen, "Design and implementation of a force reflecting manipulandum for manual control research," in *Advances in Robotics*, H. Kazerooni, editor, American Society of Mechanical Engineers, New York, pp. 1-12, 1992.

【3-7】J. Fourcade, "Longitudinal Stability of Gyroplanes," *French Space Agency,* http://www.asra.org.au/L_Stability.htm (06/01/2003).

【3-8】T. P. Neal and R. E. Smith, "An In-Flight Investigation to Develop Control System Design Criteria for Fighter Airplanes," Report, *AFFDL-TR-70-74*, 1970.

【3-9】D. Holger, "Frequency Domain Analysis of Rate Limiting Elements in Flight Control Systems," Report *No. DLR-FB 94-16*, 1994.

【3-10】A. C. Robbins, "Pilot Variability During Pilot-Induced Oscillation," *MS Thesis, Virginia Polytechnic Institute and State University*, 1999.

【3-11】E. Kevin and J. Andrew, "Handling Qualities Stress Testing," Report, *USAF Test Pilot School,* 2001.

【3-12】D. G. Mitchell and H. R. Hoh, "Development of a Unified Method to Predict PIO," *AIAA Atmospheric Flight Mechanics Conference*, pp. 611-622, 1996.

【3-13】J. E. Lamendola and M. R. Anderson, "Limit Cycle PIO Analysis with Asymmetric Saturation," *AIAA-98-4332*, pp. 379-389, 1998.

【3-14】C. Cox, C. Lewis, and C. Suchomel, "A Neural Network Based, Real-Time Algorithm For Detection and Mitigation of Pilot Induced Oscillations," *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 500-505, 2000.

【3-15】N. Raimbault and P. Fabre, “Probabilistic Neural Detector of Pilot Induced Oscillation (PIO),” *AIAA 2001-4353*, pp. 1-6, 2001.
【3-16】Accurate Automation Corporation, “Neural Network Compensation Strategy For Preventing Pilot-Induced Oscillations,” *Air Force Phase II SBIR F33615-96-C-3608*, 2001.
【3-17】A. Gelb and W. V. Velde, *Multiple-Input Describing Functions and Nonlinear System Design*, Mc Graw Hill, New York, 1968.

關於子題四部分之參考文獻

【4-1】M. P. Andersson and J. H. Lindskov. *Real-Time Linux in an Embedded Environment – A Port and Evaluation of RTAI on the CRIS Architecture*. Master’s thesis, Lund Institute of Technology, January 2003.
【4-2】Rubin and Corbet, *Linux Device Drivers 2nd Edition*, O’Reilly, June 2001.
【4-3】W. Richard Stevens, *Advanced Programming in the UNIX Environment*, June 1992.
【4-4】Herman Bruyninckx, *Real-Time and Embedded Guide Revision 0.04*, Dec. 2002.
【4-5】Tim Bird, *Comparing two approaches to real-time Linux*, Guest column at Linuxdevice.com, Dec. 2000.
【4-6】Kevin Dankwardt, *Fundamentals of Real-Time Linux Software Design*, Dec. 2000.
【4-7】RTLab, Available at http://www.rtlab.org/

關於子題五部分之參考文獻

【5-1】Wayne J. Davis, Gerald Moeller, “The High Level Architecture: Is There A Better Way?”, Proceedings of 1999 Winter Simulation Conference, pp.1595-1601, 1999.
【5-2】*RTI 1.3-Next Generation Programmer’s Guide Version 3.2*, Department of Defense, Defense Modeling and Simulation Office, September 7 2000.
【5-3】*High-Level Architecture Rules, Version 1.3*, Department of Defense, Defense Modeling and Simulation Office, April 20, 1998.
【5-4】*High Level Architecture Federation Development and Execution Process (FEDEP) Model, Version 1.5*, Department of Defense, Defense Modeling and Simulation Office, December 8, 1999.
【5-5】Lars Haendel, *The Function Pointer Tutorials, Introduction to C and C++ Function Pointers, Callbacks and Functors*, January 2002, Dortmund, Germany.

『2002年動感伺服系統之互動式控制應用』
專題實作競賽
<專題實作競賽題目：虛擬實境動態模擬系統>

一、前言

1.1 背景簡介

本實驗室，已有多年的虛擬實境發展經驗，並參與過官、產、學界各項相關的計劃，並有豐碩的成果。目前已初步建構出一由機電系統所組成的動態模擬系統，目前雖然僅侷限於娛樂或簡單操控訓練，但為了讓應用的範圍更加擴大，我們積極地為此動態模擬系統找尋新的應用，像是運動復健輔助、汽車駕駛訓練、飛行駕駛訓練都很可能為我們的模擬系統帶來新的契機。

1.2 重要性

台灣地區地窄人稠，因此虛擬技術在這樣特殊的環境之下更具有其特殊的附加價值，特別是在各項駕駛訓練方面，像是飛行的駕駛訓練，雖然不可或缺的是飛機載具，但連帶也需要徵地建立停機坪、機場、塔台…等週邊建築，用地之廣可以想見；此外，民間有非常多駕駛訓練場，需要花費許多時間、人力和金錢去維護老舊的汽車、場地、號誌，而且更不可能符合形形色色不同的訓練對象與訓練時段的需求。而這些都可以利用虛擬實境來改善。因此，我們所研發的動態模擬系統具備以下三點重要性：

1. 降低架設成本

由於目前台灣在動態模擬系統的市場往往受制於國外大廠，因此在系統架設的成本上便無法降低，若是國內能夠自行提供部分替代方案，那麼整個系統的架設成本便得以降低。

2. 擴展應用範圍

目前虛擬實境的應用仍僅侷限於線上導覽、娛樂、高階駕駛訓練…等高價位市場，且並非民生必需的應用，因此無法融入人們的生活之中。因此，必須走入人們的生活，應用在民生必需的各方面，以擴展其應用範圍。如汽車的駕駛訓練，便符合「行」一項需求，像運動復健輔助訓練則兼具了「醫療」與「娛樂」的雙重功能。

3. 協助國防發展

虛擬實境最早源自於軍事國防，特別是戰鬥機飛行員的駕駛訓練，因此本實驗室正著手發展一個能夠提供軍事國防上更具信服力的模擬系統，這將對協助國防發展有一大貢獻。

1.3 現有成果

本虛擬實境實驗室，在全體研究人員的努力之下，已有一定的成果。在各方面都有累積不少虛擬實境相關之技術。而我們將在第四章系統整合部份，分成下列三大主項，各別介紹：

1. 六軸運動平台機構與運動分析
2. 場景力學資訊處理與力操控器之研發
3. 互動式虛擬實境場景之建構

二、專題研究目的

2.1 研究目的

本專題研究目的，為結合資訊、機電、控制技術，提升企業界現有技術及新一代產品需求，發展一套即時動態模擬系統以供高科技娛樂及模擬訓練用途。而所發展的即時動態模擬系統結合聲光效果及感官刺激，讓使用者進入一個機電系統所模擬的動態虛幻世界，而有身歷其境的感受以達到娛樂或操控訓練目的。在此系統中，操作員將坐在一個六軸運動平台上，手持握著具有力迴饋的操控桿,並沉浸在三度空間之虛擬實境顯示環境裡。根據畫面所出現的場景，操作員將透過操控桿下達指令，以操控 VR 中的一部虛擬載具。透過精密的即時動態模擬，被控載具的實際變化狀況將由六軸運動平台，力操控桿及 VR 顯示器表現出來，而讓操作員有實際操控一部載具的感受。

本專題所要建構的即時動態模擬系統乃是個具有高度整合性的機電系統，意即要求機械技術與電腦技術相互結合，以達成設計上所要求目的。因此我們秉持模組化與系統整合理念，結合機械、油壓、電機、電子、電腦與資訊等專家學者，來共同研發這個虛擬實境動態模擬系統。

2.2 研究主軸與目標

本虛擬實境動態模擬系統主要分成三大主題：

1. 六軸運動平台

   為了使六軸運動模擬器能克服平台工作空間的限制，使模擬動作能更逼真，並實現六軸運動平台的速度控制與加速度控制，我們將利用「動作提示(Motion Cues)」的觀念，以人類身体對力的感受配合眼睛在視覺上的假像，藉由「沖淡演算法」來實現平台的速度與加速度控制，並與自行發展的動態模型結合，完成一個完整的虛擬實境。

2. 力複合操控桿(力操控器)

   在虛擬場景裡受控機構與物件接觸時，其之間的互動、碰撞會造成彼此在位置與力的變化，而為了讓這些虛擬的機構與物件呈現較接近真實的位移與形變，他們均被賦予參照真實物件的物理性質，彼此之間的互動也均遵照實際的物理定律；而虛擬場景裡的這些位置與力變化，一則以影像的方式提供使用者視覺上的臨場感，另一方面，力資訊也經由力回饋搖桿的轉換產生觸感傳送到使用者手上。

3. 虛擬實境互動場景

對於為營造出如臨其境的擬真之虛擬世界而言，VR 場景的使用實佔著一個相當關鍵的地位。因此發展一個多功能的虛擬實境動態模擬系統，並經由虛擬實境（VR）與運動模擬器的結合，逼真地模擬實際場景與設備或載具，而協助達到多種訓練及娛樂目的。我們除了以 SGI 工作站為場景開發的主要平台之外，也試圖在 PC 個人電腦平台下發展虛擬實境場景，以尋求較為廉價且容易普及的解決方案。

三、團隊合作方式

3.1 工作項目

1. 六軸動作平台架構分析與控制
   - ◆角度與加速度感應 Sensors 的加入與分析
   - ◆多種控制法則的開發與系統分析

2. 力資訊處理與操控器研發
   - ◆操控器之順應性控制技術建立
   - ◆模擬系統之力回饋搖桿製作

3. 虛擬實境場景與動態模擬之建構
   - ◆動態模擬技術之探討
   - ◆虛擬實境場景之建立

3.2 團隊成員

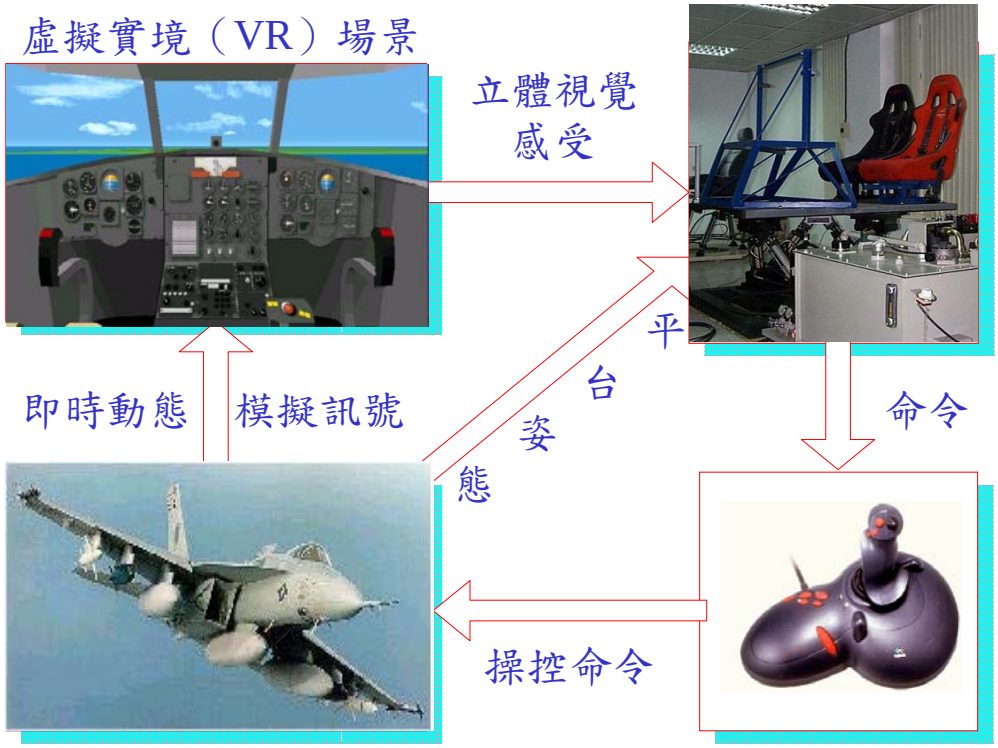
類　別	姓　名	在本競賽內擔任之詳細具體工作性質、項目及範圍
教授	林進燈	指導虛擬實境及六軸平台之相關技術
教授	楊谷洋	指導力回饋搖桿之相關技術
研究生	陳建宏	動態模擬技術之探討
研究生	黃騰毅	虛擬實境場景之開發與建立
研究生	黃冠智	六軸運動平台之控制與分析
研究生	羅仲志	模擬系統之力回饋搖桿控制

四、系統整合

本研究群在交通大學電子資訊大樓 416 室建立了一個多媒體虛擬實境即時動態模擬系統雛型，其功能如下圖所示。在此系統中，操作員坐在一個六軸運動平台上，手握著操控桿，而眼睛看著或頭上戴著 VR 顯示器。根據畫面所出現的場景，操作員透過操控桿下達指令，以操控 VR 中的一部虛擬載具。此操控命令被輸入所

模擬之載具的精確物理模型中，以求得真實情況下系統的反應（包括姿態，速度，加速度，力道等）。這些反應透過行為轉換與控制模組，而由六軸運動平台及 VR 顯示器即時表現出來，以讓操作員獲得身歷其境的感受。簡言之，已發展的多媒體虛擬實境即時動態模擬系統雛型，是以一個六軸運動平台為核心，配合著虛擬實境的影像顯示作互動式的搭配。這套系統可拿來當模擬訓練機使用，亦可用於娛樂用途（如動感電影院）。以下將分項詳述我們所執行的成果。

1. 六軸運動平台
2. 動力回饋搖桿
3. 虛擬實境場景



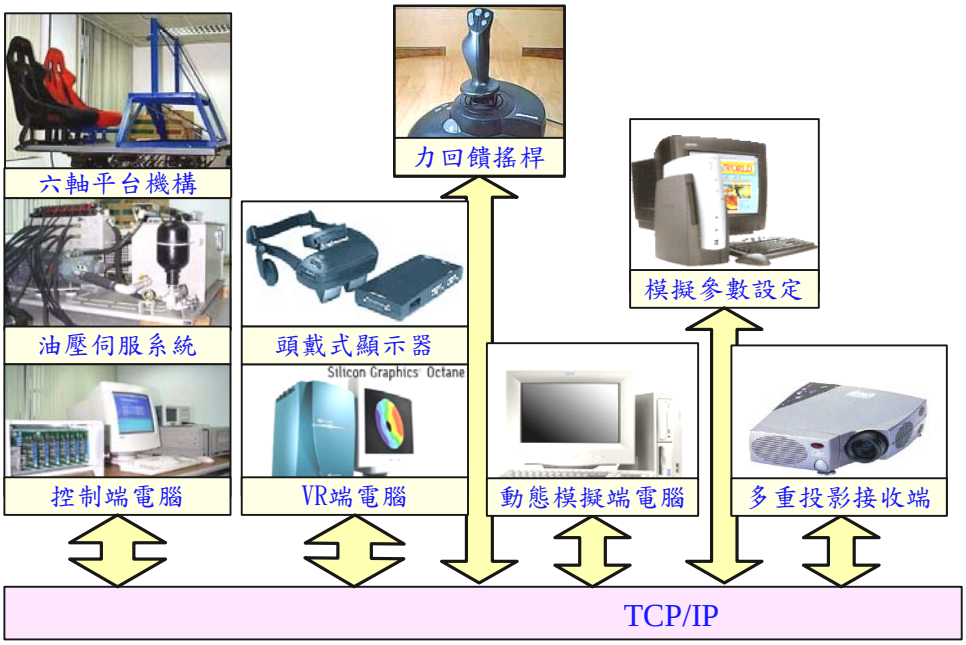
系統整合方塊圖

五、研究成果與結論

5.1 整合測試

虛擬實境動態模擬系統是一個整合電腦圖學、機械控制、動力學運算及虛擬環境的系統。為了讓使用者有『身歷其境』的感覺，系統必須能即時地處理圖形、影像及聲音等運算。隨著虛擬場景複雜度的增加，及使用者要求更逼真、更即時的虛擬環境模擬系統，這些需求需要高效率運算能力的電腦來支援才能達到，因此早期之研究多侷限於使用大型工作站級以上或是特殊用途的電腦，但是這樣的設備都是很昂貴，不是一般使用者所能負擔。然而，隨著科技的日新月異，現今的個人電腦和工作站的運算速度越來越快、價格也越來越低廉，因此，將多台個人電腦或是工作站透過區域網路的結合，形成一個大型的分散式運算環境，也同樣可以達到高效率的運算能力。分析一個模擬器電腦系統的工作，包括圖學顯像(graphics render)、使用者的輸入介面(user interface)、音效(sound effect)、虛擬場景

資料庫管理(virtual world manager)、真實模組(high fidelity models)、預測模組(dead reckoning models)等。



虛擬實境動態模擬系統

隨著網際網路日益盛行，全球通訊業者投注大量資金建構更寬頻的光纖固網，我們的世界也正在劇烈地變化著。未來我們將擁有雙重的身份，一個在實體世界中的身份，而另一個身份則是在由網路所建構，超越國界藩籬的虛擬世界中。而虛擬實境技術在網際網路的發展更具有舉足輕重的角色，它將扮演帶領人們進入虛擬世界的重要工具。目前最明顯的例子是能在網路相關產業一片蕭瑟中異軍突起的「線上遊戲」，發展虛擬實境將間接地帶動整個 3D 立體繪圖顯像技術的提升，對於國內在娛樂市場的發展具有正面的附加價值。