無人機系統識別及控制實驗報告

SYSTEM IDENTIFICATION AND CONTROL ON TELLO DRONE

B08901161 林邑恆

指導教授: 劉浩澧

一、系統簡介

此實驗目的為透過無人機作為系統，驗證課堂所學之控制理論和實驗課所學之馬達系統鑑別能否套用在較高階新穎的系統上。藉由相機作為感測器量測無人機於空間中所處的位置，可以實行系統鑑別和回授控制。此系統架構包含作為控制本體的無人機，作為感測器的相機，及實現回授控制的軟體三大部分。

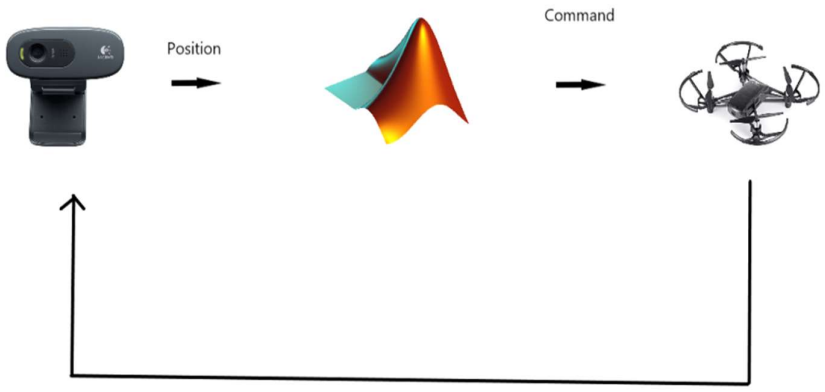


fig.1

二、硬體簡介

2.1 無人機簡介

此實驗使用的無人機為 DJI 公司發售的 Tello EDU 無人機，此無人機搭載前鏡頭及下視鏡頭，並可以透過無線網路進行指令下達。

fig.2

2.1. 1 指令下達

此無人機可以透過 matlab 的環境進行指令下達，包含起降，移動等(常用指令詳見 fig.3)，並且 tello 也提供了官方的 SDK 供使用者新增一些指令。

為了實現無人機的控制，需安裝 matlab 提供的 tello support package，並加入 SDK 的指令。安裝方法如 fig4 ~ fig.7

Functions collapse all

Basic Flight

takeoff	Initiate Ryze drone takeoff
land	Land Ryze drone
abort	End flight of Ryze drone

Advanced Flight

flip	Flip Ryze drone in specified direction
move	Move Ryze drone in all three axes
moveback	Move Ryze drone backwards
movedown	Move Ryze drone down
moveforward	Move Ryze drone forward
moveleft	Move Ryze drone left
moveright	Move Ryze drone right
moveup	Move Ryze drone upwards
turn	Turn Ryze drone by a specified angle

fig.3

Step 1. 到 [Ryze Tello Drone Support from MATLAB - Hardware Support - MATLAB & Simulink (mathworks.com)](), 按 get support package 下載所需檔案

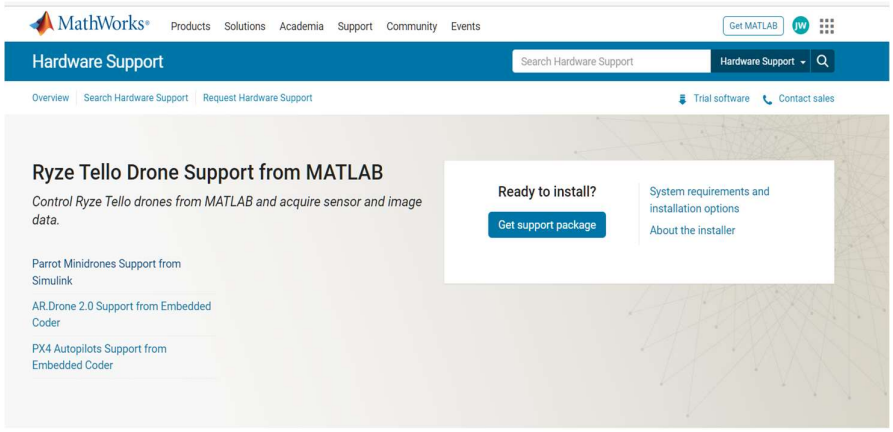


fig.4

Step .2 在 matlab command line 中輸入 ***open('ryze.m')*** 開啟原始碼，並找到 function ***move***

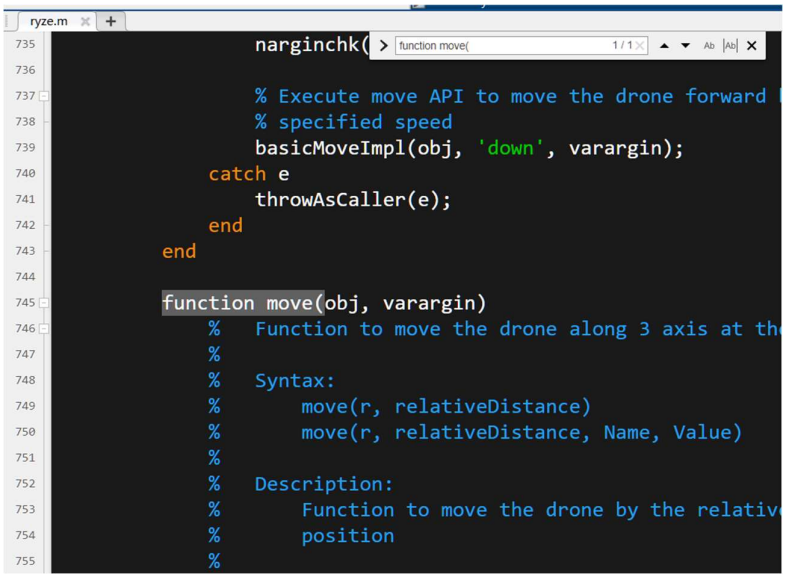


fig.5

Step .3 在 function ***move*** 後加入以下程式碼

```
%-------------------------------------------------------------
        function rc(obj, a,b,c,d)
            try
                % Set speed and then send move command

                command = char(sprintf("rc %0.4f %0.4f %0.4f %0.4f", a,b,c,d));
                write(obj.CommandConnection, uint8(command));

            catch e
                throwAsCaller(e);
            end
        end
```

fig .6

註: 不同版本中下達指令的變數名稱不同(此處為 R2021b)，可以參考 ***move*** 中 ***write*** 裡面放的變數名(詳見 fig.6 中反白處)。

完成上述步驟後，***rc*** 此 SDK function 就可以在 matlab 中使用，進行 input 的傳

輸。

```
function move(obj, varargin)


    try
        narginchk(2,6);

        % Validate if the input parameters are within range
        [coordinate, speed, blocking] = ryzeio.internal.Utility.validateMoveParams(varargi

        % Convert units to cm
        coordinate = coordinate.*100;
        speed = speed * 100;

        % Set speed and then send move command
        obj.CommandAck = false;
        command = char(sprintf("speed %d", round(speed)));
        write(obj.CommandConnection, uint8(command));
        waitForCommandAck(obj);

        obj.CommandAck = false;
        obj.CommandError = false;
        if(strcmpi(obj.State, char(ryzeio.internal.DroneStateEnum.Hovering)))
            obj.State = ryzeio.internal.DroneStateEnum.Flying;
```

fig. 7

2.1.2 無人機飛行

Tello 無人機提供使用者控制前後，左右，上下和旋轉的移動，但無法控制如 roll 或 pitch 的角度。 並且無人機的控制上有諸多限制，無人機最低的飛行高度為 20 公分，飛行的最大速度為 1 公尺/秒。

2.1.3 RC 指令

RC 指令為實現自動控制的重要指令，有別於如 move 指令等直接指定無人機的飛行距離及速度，RC 指令為一無單位，代表無人機控制端(ex app 內的搖桿)輸入的大小，範圍為-100 ~ 100 ，且只接受整數輸入。
其詳細的說明如 fig.8 所示，參數 a,b,c,d 分別代表控制左/右，前/後，上/下，及 yaw 角度的搖桿輸入。

rc a b c d	Set remote controller control via four channels. "a" = left/right (-100-100) "b" = forward/backward (-100-100) "c" = up/down (-100-100) "d" = yaw (-100-100)

fig.8

2.2 相機簡介

2.2.1 相機硬體簡介

此系統中使用的相機為羅技 C270 HD 網路攝影機，最高 frame rate 為 30 frame/second ，在支援此 frame rate 下最高的解析度為 640*480。

2.2.2 影像定位系統簡介

此影像定位系統利用相機辨識無人機目前在前方平面所在的位置，為了加

快運算速度達到較高的 frame rate，無人機的辨識僅使用灰階處理，若背景並非全白辨識效果會大受影響。

2.2.3 影像辨識程式簡介

影像辨識首先需要定義無人機的初始位置，多利用無人機未起飛時進行無人機目前在影像中所在相素作為無人機的參考原點，後續則找尋無人機在影像中所在相素和原點相素的差值，利用相機焦距做換算找出無人機現在位置和原始位置的差值作為無人機所在位置的依據。這樣的定位方式會使得在影像邊緣的位置較不準確，但可以大大提升影像的辨識速率及解析度要求。

三、軟體簡介

3.1 軟體簡介

此系統的軟體架構主要負責生成用於系統辨識的輸入方波和實現 closed loop PID control，使用者可以透過 GUI 改變參數進行實驗。具體使用方式如下所述。

3.2 軟體使用

3.2.1 System Identification

此處的 GUI 提供使用者取得無人機用作系統識別資料的方法。詳細 GUI 如 fig.9 所示。

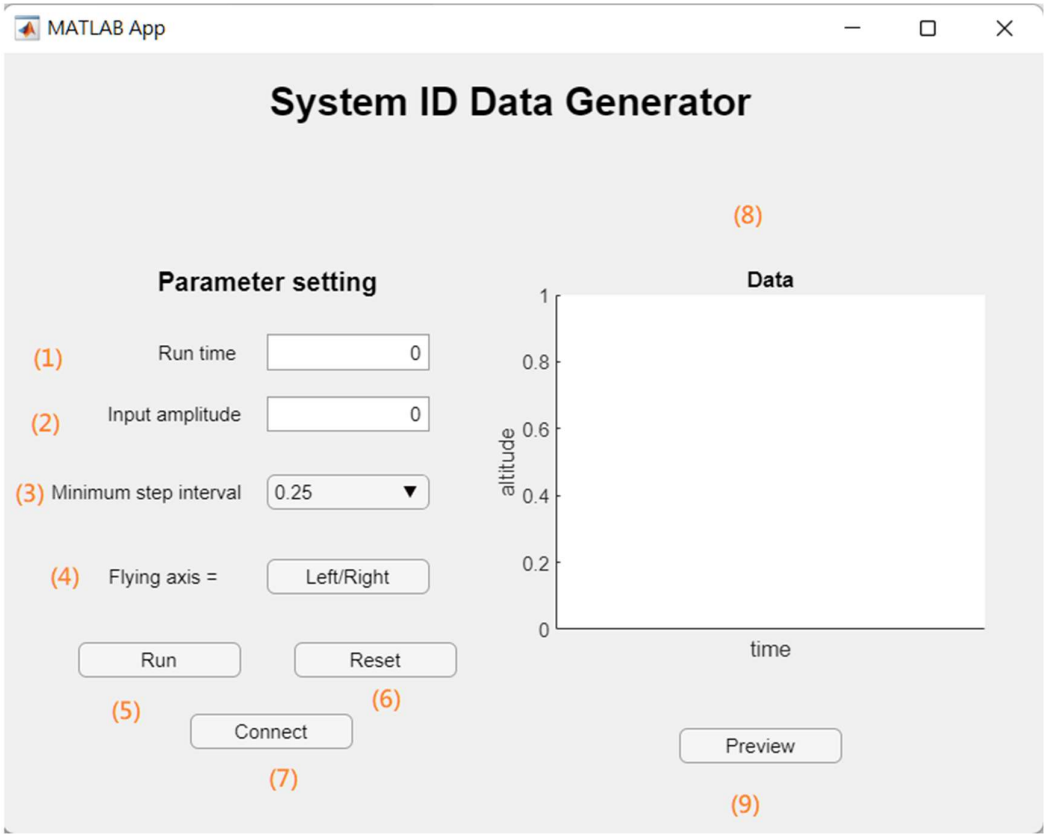


fig.9

(1) 設定飛行總時長
(2) 設定 input 方波的振幅 (0~100)
(3) 設定產生出來的 random 方波的最小時間間隔
(4) 設定無人機飛行的方向 (上下/左右)
(5) 執行飛行
(6) 重設所有參數，並清空圖表
(7) 連結無人機
(8) 無人機的位置 - 時間圖
(9) 檢查相機鏡頭是否合適

在執行飛行(run) 前，需先確保電腦跟無人機已透過 wifi 連線，並執行(7)使 matlab 跟無人機進行連線。 若網路未確實連結，會跳出如 fig. 10 的警告視窗。

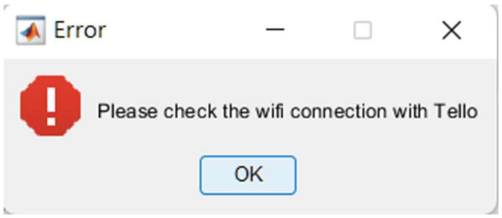


fig.10

透過執行此程式，使用者可以取得無人機對於隨機產生的方波的響應(無人機位置)，程式會回傳輸入的隨機方波跟位置響應數據到 matlab workspace，供使用者進行後續系統鑑別使用。

3.2.2 PID control

此 GUI 提供使用者透過設定 P , I , D 及欲移動的距離，實現閉迴路回授控制，詳細 GUI 內容可見 fig.11 。

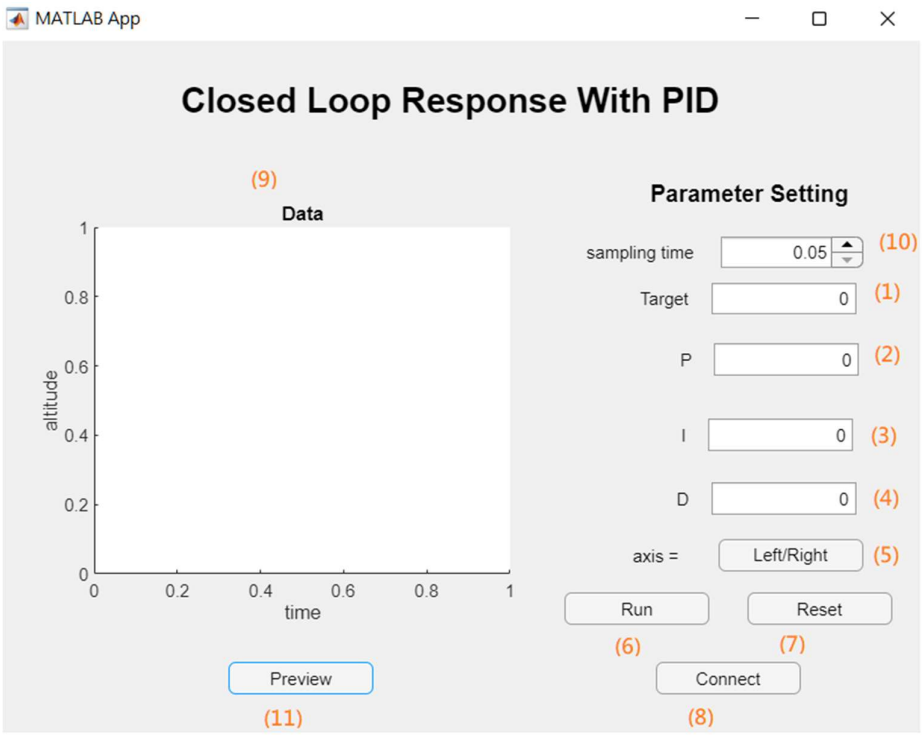


fig.11

(1) 設定無人機目標移動距離
(2) 設定 Kp 值
(3) 設定 Ki 值
(4) 設定 Kd 值
(5) 設定無人機飛行的方向
(6) 執行飛行
(7) 重設所有參數，並清空圖表
(8) 連結無人機
(9) 無人機的位置 - 時間圖
(10) 設定取樣時間(sampling time)
(11) 檢查相機鏡頭是否合適

透過此 GUI，使用者可以實現無人機的閉迴路回授控制，並且程式會回傳無人機的位置資料，用於後續分析。

3.3 使用程式簡介

3.3.1 無人機控制

3.3.1.1 起飛

經多次觀察，無人機內建的起飛指令每次所達到的高度並不相同，為了固定一開始起飛所在的高度，首先利用前述(2.1.3) 提及的 rc 指令將無人機降至最小飛行高度(20cm)，再利用 moveup 指令移動無人機向上飛行 80 公分，使起飛後的高度大約處於離地 100 公分的位

置。

3.3.1.2 輸入訊號輸入

在無人機的飛行指令下達上，我們使用 rc 指令，將輸入作為 rc 指令中的參數傳達給無人機。

3.3.2 系統識別

3.3.2.1 輸入訊號產生

在輸入訊號產生上，我們透過給定的最小 step interval 和總共的執行時間，產生相對應的輸入訊號，具體作法為先利用 matlab 的 rand 函式產生一串長度為(總共執行時間/最小方波時間)的隨機向量，在利用給定的取樣時間拓展到我們需要的輸入訊號。

3.3.2.2 系統識別飛行

在實際的飛行中，我們利用一個 while 迴圈控制指令的下達，為了要控制固定的取樣時間，故測量每一個迴圈執行的時間，在尾端加上 pause 指令補足不足的時間達到對取樣時間的固定。

3.3.3 閉迴路控制

和前述(3.3.2.2)相同，此處採用 while 迴圈控制時間，並且實時的計算目標和現在位置的差值作為回授信號，進行 PID 控制。

四、系統識別

4.1 系統識別簡介

無人機的系統識別在許多文獻中都有被探討，而在()特別針對如本實驗使用的低成本無人機進行系統識別。本部分旨在探討在控制實驗中所學的方法能否套用在無人機系統上，獲得無人機移動的系統函數，幫助無人機更順利的執行任務。

4.2 資料取得

透過使用前述(3.2.1)中所示的 GUI，設定參數並進行無人機飛行取得參數，過程中需注意如果無人機中途遇到飛行至最低飛行高度(20cm)處造成無法繼續下飛的情況，會使得無人機無法正確地反映出響應，並進行降落及在 matlab 工作區報錯，具體的避免方法為設定較小的輸入震幅或合適的運行時間。
(經實驗若輸入震幅小於 20 時，無人機容易受到較大的外界環境干擾，識別效果較差) 。

4.3 tool box 分析

利用前述 (3.2.1) 中的 system ID GUI 取得數據後，利用 matlab 的 system Identification tool box (fig.12) 進行系統識別的分析。

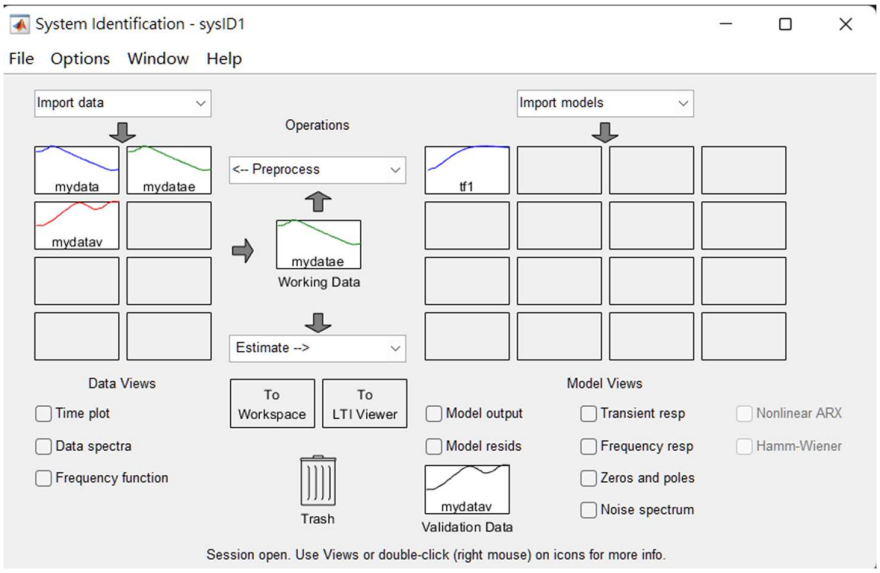


fig.12

透過使用不同頻率生成的輸入信號作為系統識別中的輸入信號，並使用無人機的位置作為輸出信號，進行無人機的系統識別。

通過觀察無人機的步階響應，發現其大約可以用三階或四階模型進行近似，且參考教科書(ref. 2) 中提及的無人機系統函數，決定選取四階模型作為系統模型進行識別。

$$\frac{x_m(s)}{T_{lon}(s)} = -13\frac{1}{s(s-1.6\pm 2.8j)(s+3.4)(s+20)}$$

fig.13　參考書中提供的無人機系統模型

4.4 實驗結果

4.4.1 上下移動

4.4.1.1 極點分布

在上下方向的飛行中，通過五次獨立的實驗獲得如以下的結果。

fig.14 中是五次實驗得到的極點分布

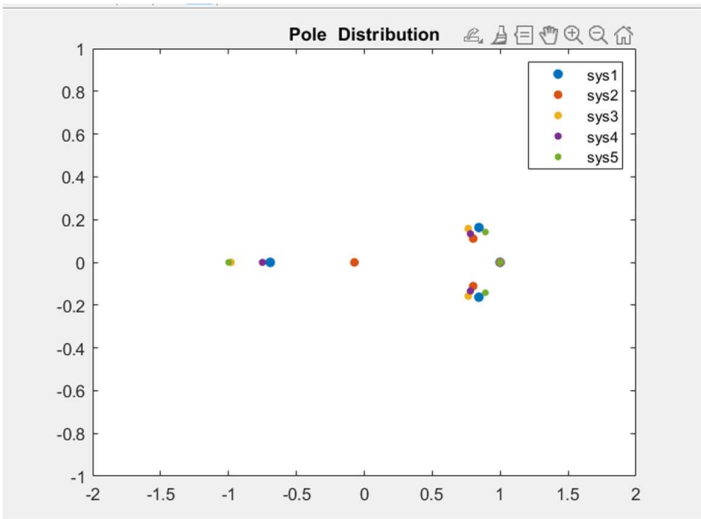


fig.14

如圖中所示，在右半平面的極點分布都很一致，故可推測在右半平面的極點識別結果不錯。在左半平面部分，除了 sys2 的極點很明顯是離群值，各有兩兩相近的極點(sys1 和 sys4 , sys5 和 sys3)，故須進行閉迴路驗證。並且此處因為無人機的移動 dynamic 比較偏向一不穩定的系統，而在(1,0)的 pole 就很好的描繪了這樣的特性。

4.4.1.2 閉迴路驗證

為了驗證左半平面極點的正確性，此處進行閉迴路的驗證。首先採用 sys3 透過 matlab 的 siso tool 進行 PID tuning ，獲得參數 P=2.9, I=0 ,D=0 , 將此參數設定於 3.2.2 中的 GUI 並設定 target=100，此操作使無人機在閉迴路 PID 控制下向上移動 100 公分，而結果如 fig.15,fig.16 所示，

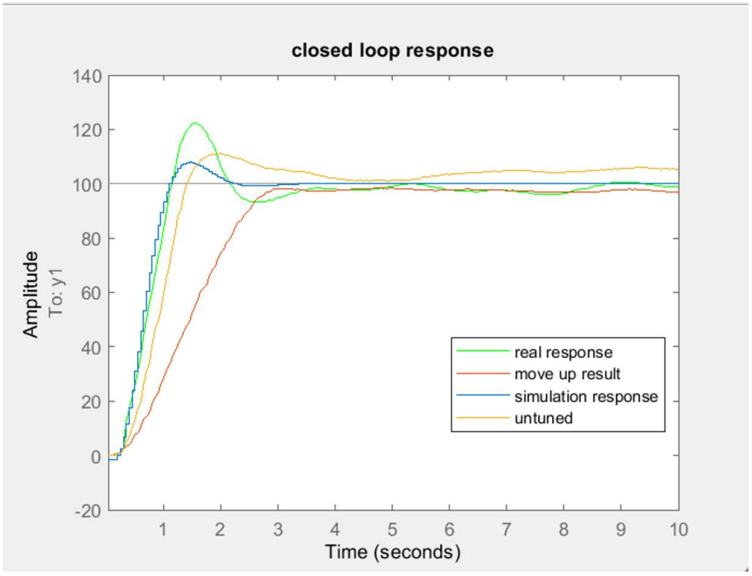


fig.15 利用 sys3 進行比較

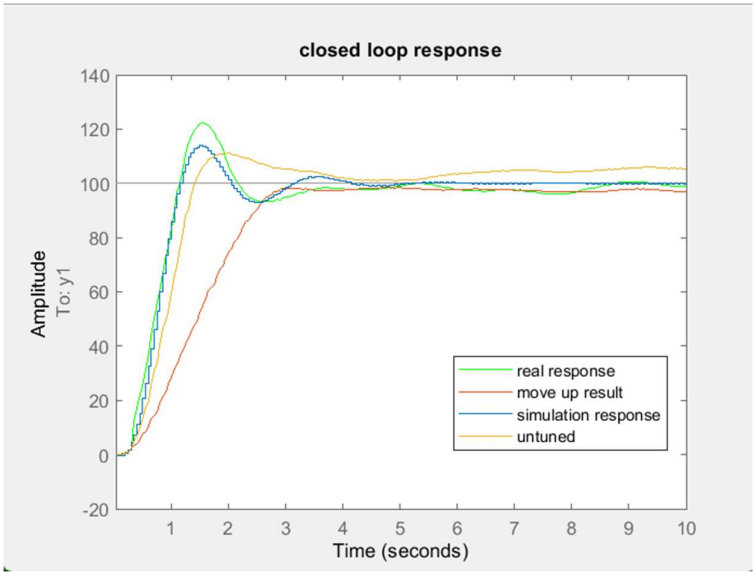


fig.16 利用 sys4 進行比較

透過比較實際的實驗結果和利用 sys3, sys4 分別進行的模擬結果，可以發現 sys4 較符合實際結果，故 sys1, sys4 應該較符合真實情況。

4.4.2 左右移動

4.4.2.1 左右移動分析

在左右移動中，無人機內部有一控制器保持 roll angle 維持在 0，故左右移動的響應會受到此一因素影響而造成在改變方向時有明顯的左右晃動，此一 dynamic 無法很好的只用線性的系統和控制左右方向的輸入來進行模擬，因此採取低頻濾波器濾除。產生的極點分布如 fig.17 所示

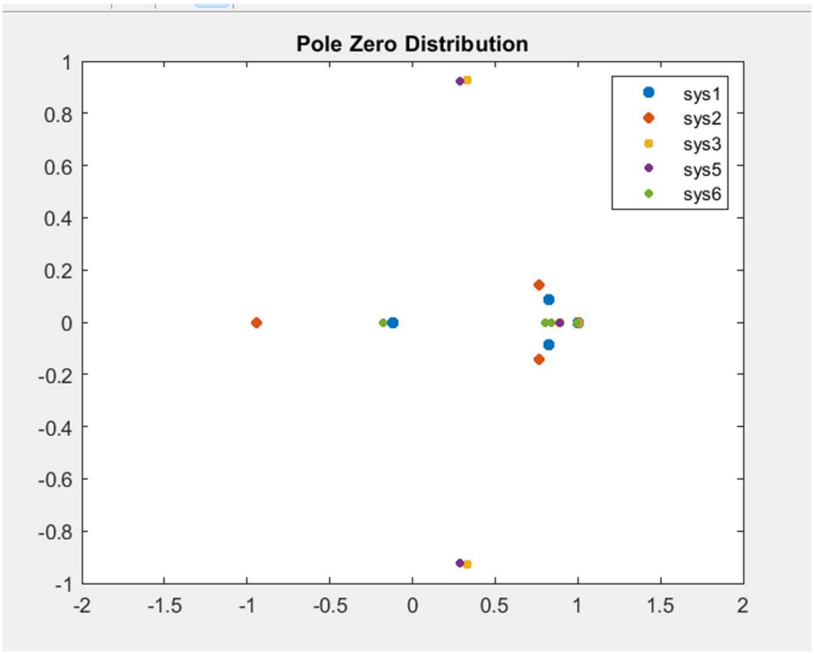
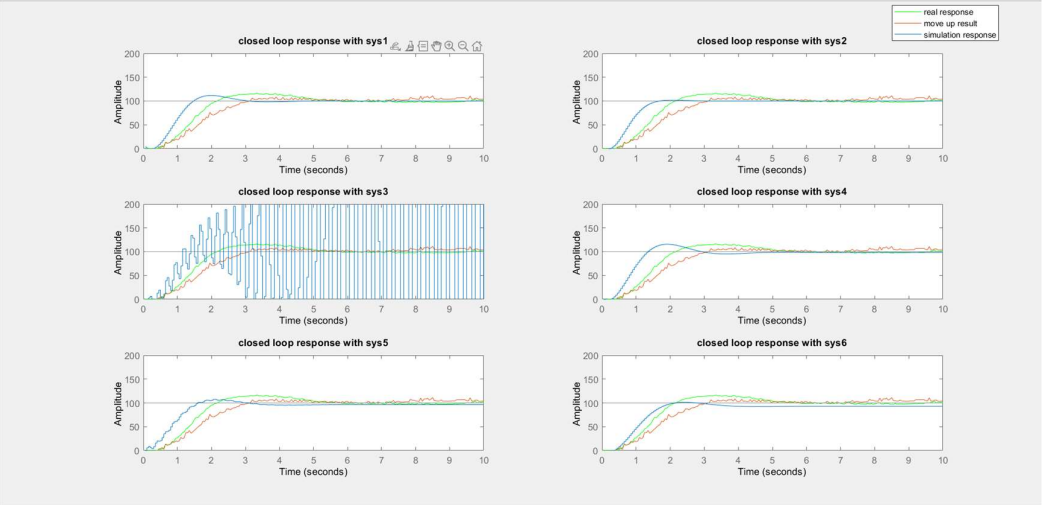


fig.17

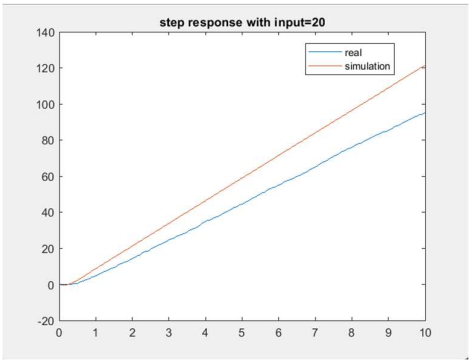
發現有極點分布較混亂的狀況，因此對每個系統模型進行閉迴路的比較，結果如 fig.18 所示

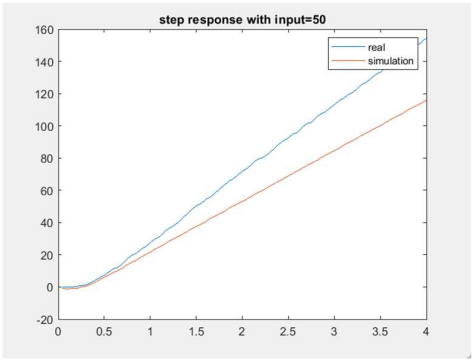


經觀察發現無論是哪個模型模擬的效果都沒有很好，推測是因為上述無人機內部控制器控制 roll angle 造成的影響，但還是能大致上近似實際的結果，可以作為 PID 參數調變的參考。

4.5 系統識別問題 – 非線性現象

上述實驗採用的輸入方波震幅為 30，但觀察如 fig , fig ，用震幅為 30 的方波識別出來的系統無法很好的擬合輸入震幅為 20 和 50 的步階響應。透過實際測試利用輸入方波震幅為 50，發現此時可以分別的在擬合相對應的步階響應上有較好的表現，如 fig.19 所示，因此推測輸入震幅和系統響應中有一非線性關係，可能會影響閉迴路的實際表現。





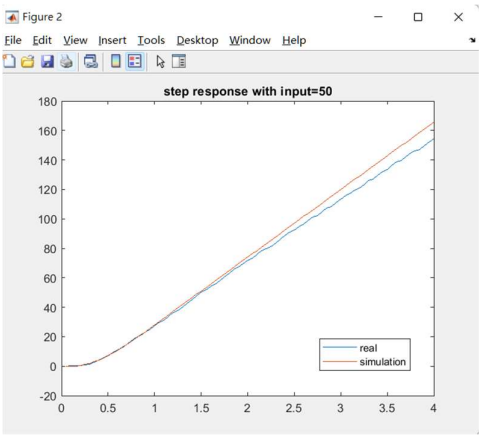


fig.19 步階響應比較

五、回授控制

5.1 回授控制目標

有鑑於左右方向的系統識別不太理想，此處針對上下方向的移動進行回授控制，回授控制的目標為使 settling time 小於無人機內建的 moveup 指令並同時保持合理的 overshoot。利用 moveup 指令飛行的軌跡如 fig.20 所示

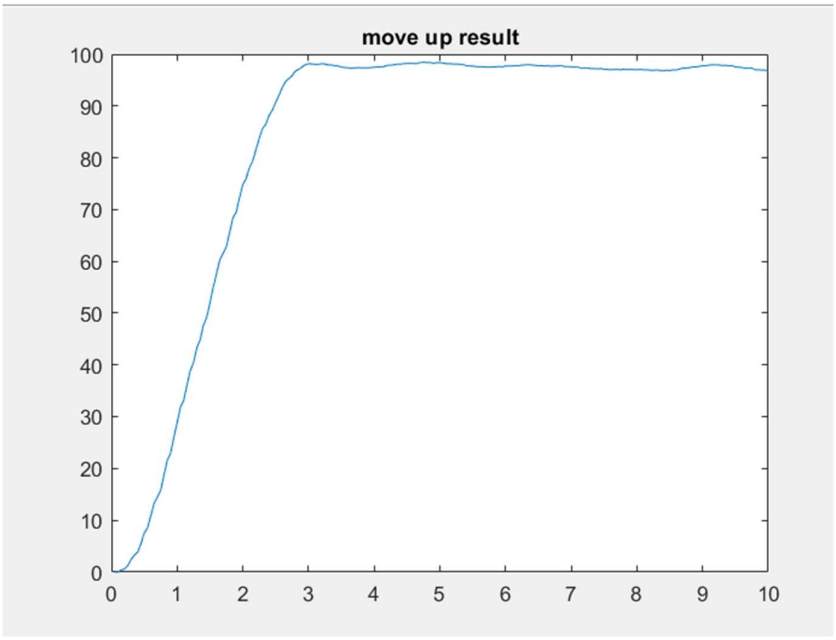


fig.20

可以觀察到 settling time 約為 3 秒。

5.2 回授控制實作

透過 matlab 提供的 sisotool 並取前述較符合實際情形的 sys4 進行操作可以發現如果只有採取單純的閉迴路不加上任何控制器，settling time 大約需 5 秒，遠不及 moveup 指令的 3 秒。

透過 siso tool 進行 tuning 後，發現在控制器 C=3.2 時，可以將 overshoot 控制在 20%內並且達到 settling time 約為兩秒，優於 moveup 的 3 秒。

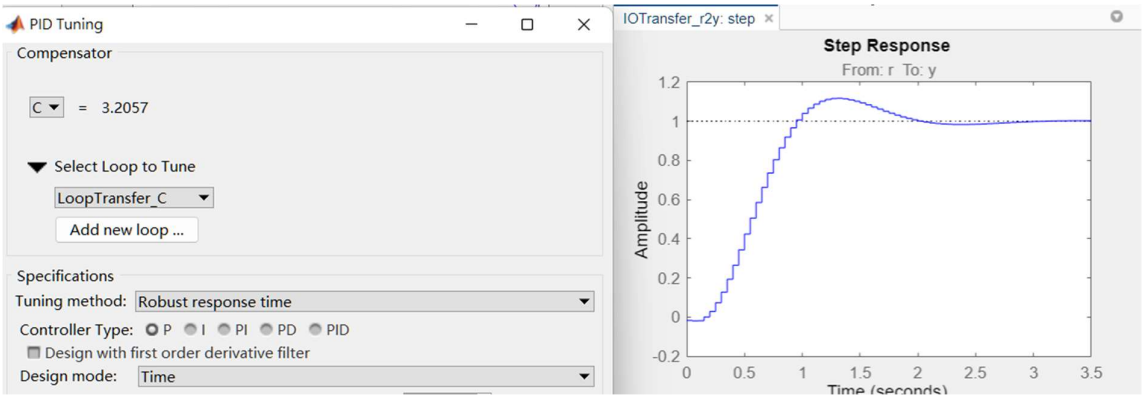


fig.21

實際情況如 fig.22 所示

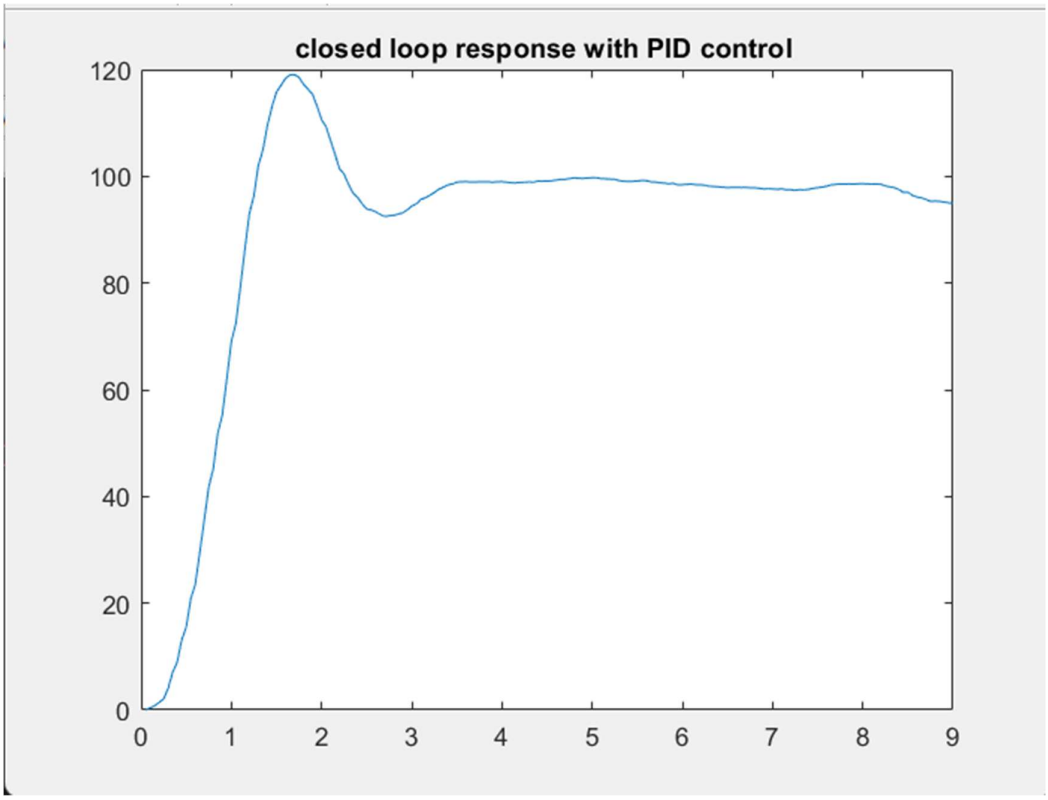


fig.22

可以發現 rise time 雖然很快，但 settling time 並不好。
比較之前系統模型和實際情況的對比，發現實際情況的 overshoot 比模擬得來的大，因此此處應該在 tune overshoot 的時候有所保留。
基於這次經驗，調整 P=2.5，結果如 fig.23 所示

可以看到 settling time 約為兩秒，並且 overshoot 在 20%內

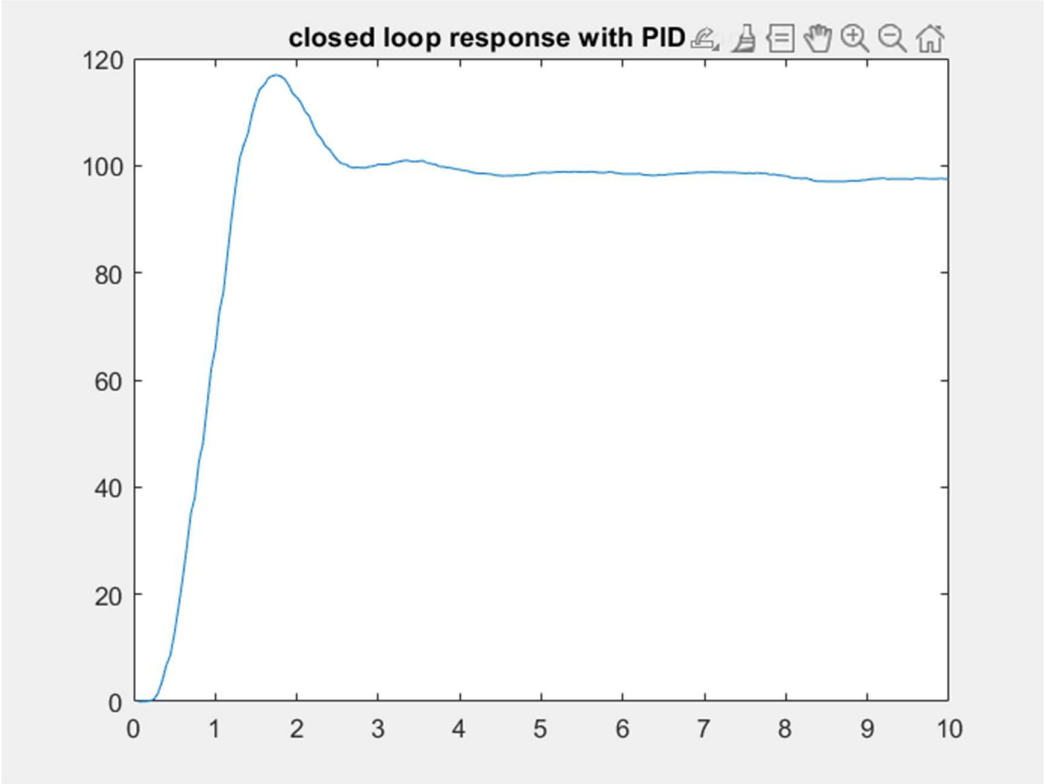


fig.23

fig.24 為和無人機內建指令進行比較，可以看到我們實作控制器的結果比內建的指令在 settling time 上來得更好。

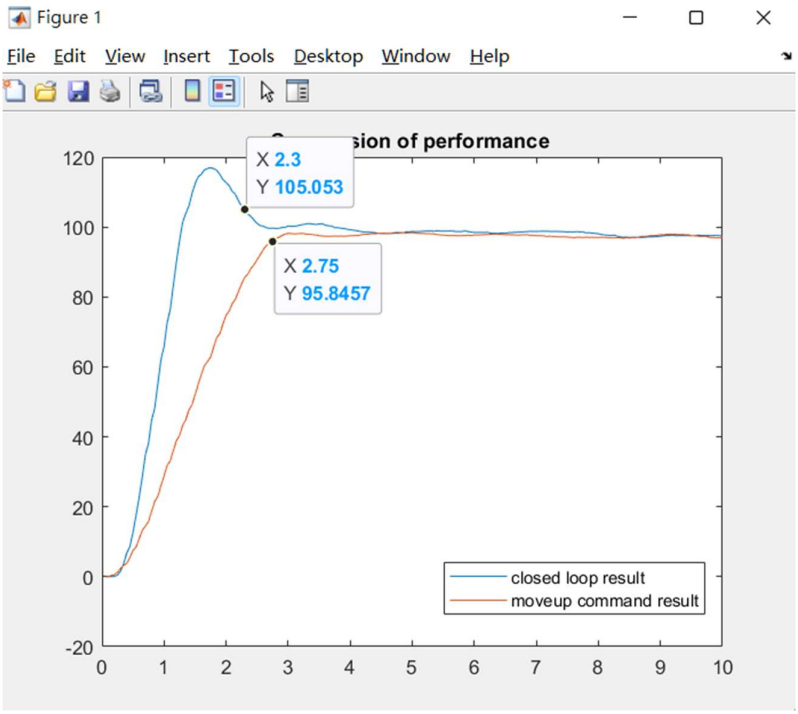


fig. 24

六、注意事項

1. 無人機很容易受外界影響，實驗時若實驗室內電風扇未關閉，會造成極大的影響。
2. 無人機的連線經測試不太穩定，常有經歷過一次起降後無法重新起飛的情形，經測試將無人機重新開機並連線有助於解決。
3. 無人機飛行時扇葉旋轉速度高，若不慎觸碰可能造成傷害。

七、參考資料

[1] A. Hernandez, C. Copot, R. De Keyser, T. Vlas and I. Nascu, "Identification and path following control of an AR.Drone quadrotor," 2013 17th International Conference on System Theory, Control and Computing (ICSTCC), 2013, pp. 583-588

[2] Feedback Control of Dynamic Systems: Global Edition, 8th Edition
Gene F. Franklin, J. Davis Powell, Abbas F. Emami-Naeini