Control System Design Report

Lab #3

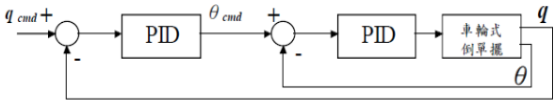
Date: 2017.05.15
ECE106　0210749　賈恩宇
ECE106　0210825　蕭宇杰

A. Objective

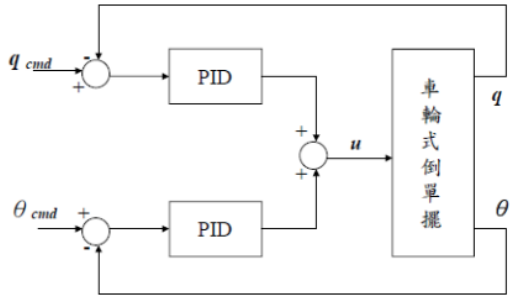
本次實驗主要為設計控制器以達成對車體位置達成控制的目的。實驗中嘗試以兩種不同方式實作，其一為在已使用一 PID 控制器控制車子傾角的情況下，再使用另一 PID 控制器對其位置進行控制；另一種方式則為使用 State-feedback 控制，直接利用可量測到的系統 State 並對於 State 其中的位置進行控制。

B. Principle & Derivation

首先是嘗試使用 PID 控制位置的部分。在本次實驗中嘗試 PID 控制器時，有兩種配置方式，如圖(1)和圖(2)。



圖(1)



圖(2)

而其中平衡車系統使用的模型則是參考論文[5] [3][4][2]使用的動態方程式，其線性化後的方程式如式(1)與式(2)所示。

$$\psi'' + \rho_1 x' + \rho_2 \psi + \rho_3 x'' = \rho_4 v_a \tag{1}$$

$$x'' + \sigma_1 x' + \sigma_2 \psi'' = \sigma_4 v_a \tag{2}$$

透過上述兩式，經過整理並帶入$\rho_1 \sim \rho_4, \sigma_1 \sim \sigma_2, \sigma_3$的實際數字後，可以得到輸入電壓$v_a$與車傾角$\psi$以及平衡車位置$x$的兩個轉移函式，如式(3)與式(4)。

$$\frac{\Psi}{V_s} = \frac{-5.456s}{s^3+8.971s^2+136.9s+1307.8} \tag{3}$$

$$\frac{X}{V_s} = \frac{4.196s^2+581.5}{s(s^3+8.971s^2+136.9s+1307.8)} \tag{4}$$

首先針對圖(1)的配置方式，可以將對於車傾角的 PID 控制器以及式(4)這兩部分整合為一 inner loop，因此可以得到新的轉移函式作為新的 plant，如式(5)。

$$\frac{X}{V_s} = \frac{1.625s^4+61.25s^3+677.4s^2+8488s+6.267\cdot10^4}{1.625s^5+62.25s^4+686.3s^3+8625s^2+1308s} \tag{5}$$

接下來即可對於此新的 plant 進行控制。

而對於圖(2)的配置方式，則是直接以式(4)作為 plant 進行控制。

另外一種方式則是直接使用 State-feedback 的方式控制車子位置。同樣經過整理後，可以將式(1)與式(2)以式(6)與式(7)矩陣的形式表示。

$$\begin{bmatrix} x' \\ x'' \\ \psi' \\ \psi'' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & A_1 & A_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_3 & A_4 & 0 \end{bmatrix} \begin{bmatrix} x \\ x' \\ \psi \\ \psi'' \end{bmatrix} + \begin{bmatrix} 0 \\ B_1 \\ 0 \\ B_2 \end{bmatrix} v_s \tag{6}$$

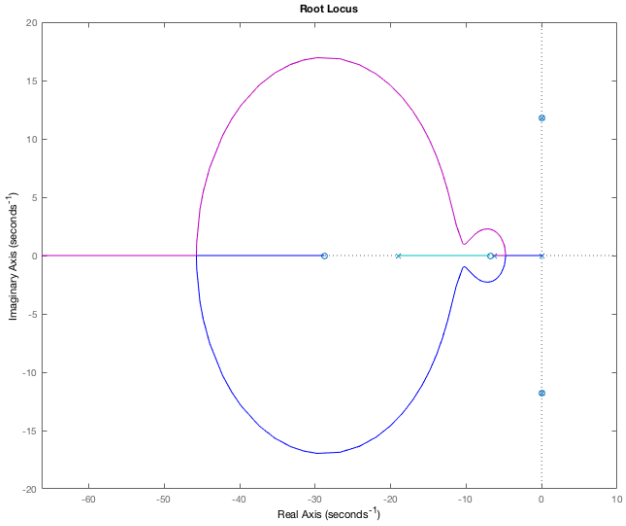
$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ x' \\ \psi \\ \psi' \end{bmatrix} \tag{7}$$

其中$A_1 = -\frac{\sigma_1 - \sigma_2\rho_1}{1-\sigma_2\rho_3}, A_2 = \frac{\sigma_2\rho_2}{1-\sigma_2\rho_3}, A_3 = -\frac{\rho_1 - \rho_3\sigma_1}{1-\sigma_2\rho_3}, A_4 = -\frac{\rho_2}{1-\sigma_2\rho_3}, B_1 = \frac{\sigma_4 - \sigma_2\rho_4}{1-\sigma_2\rho_3}, B_2 = \frac{\rho_4 - \rho_3\sigma_4}{1-\sigma_2\rho_3}$

由於上式中的 state $\mathbf{x} = [x; x'; \psi; \psi']$皆可直接或間接透過平衡車程式的撰寫取得，再加上當控制於某一位置時，除了車子位置x外，對於其餘 state 皆要控制至零，所以可以重新設計 state $\bar{\mathbf{x}} = [x - x_0; x'; \psi; \psi']$，其中的$x_0$即是目標位置。藉由這樣的設計，可以在不改變原先 state feedback 設計的情況下，改變平衡的位置。接下來只要令輸入為$v_s = -\mathbf{K}\bar{\mathbf{x}}$，並選擇好 Pole 的位置，透過計算$|sI - (A - BK)|$即可得設計好的$\mathbf{K}$矩陣。

C. Data, Chart and Analysis

首先是針對使用圖(1)的 PID 控制方式進行設計。圖(3)即是利用式(5)繪製的 Root locus 圖。

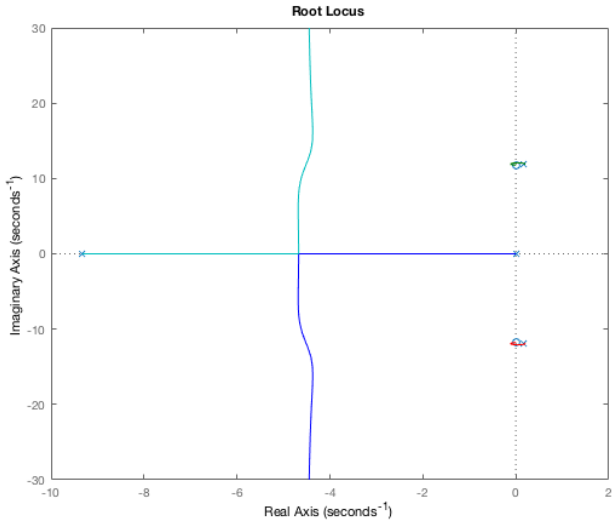


圖(3)

根據上圖，因為加入 I 與 D 控制器可能會更不穩定，因此可以設計出一組「P」控制器，可以勉強使其 stable，但放到實際的平衡車上卻無法平衡，因此無法採用。

接下來則嘗試使用圖(2)的控制方式設計。對於圖(2)由於對於位置的轉移函數以及對於車傾角的轉移函數直接共用同一輸入，所以只要使用式(4)作為 plant 控制即可，所以可以利用以式(4)繪製出的 Root locus 來進行設計，如圖(4)。

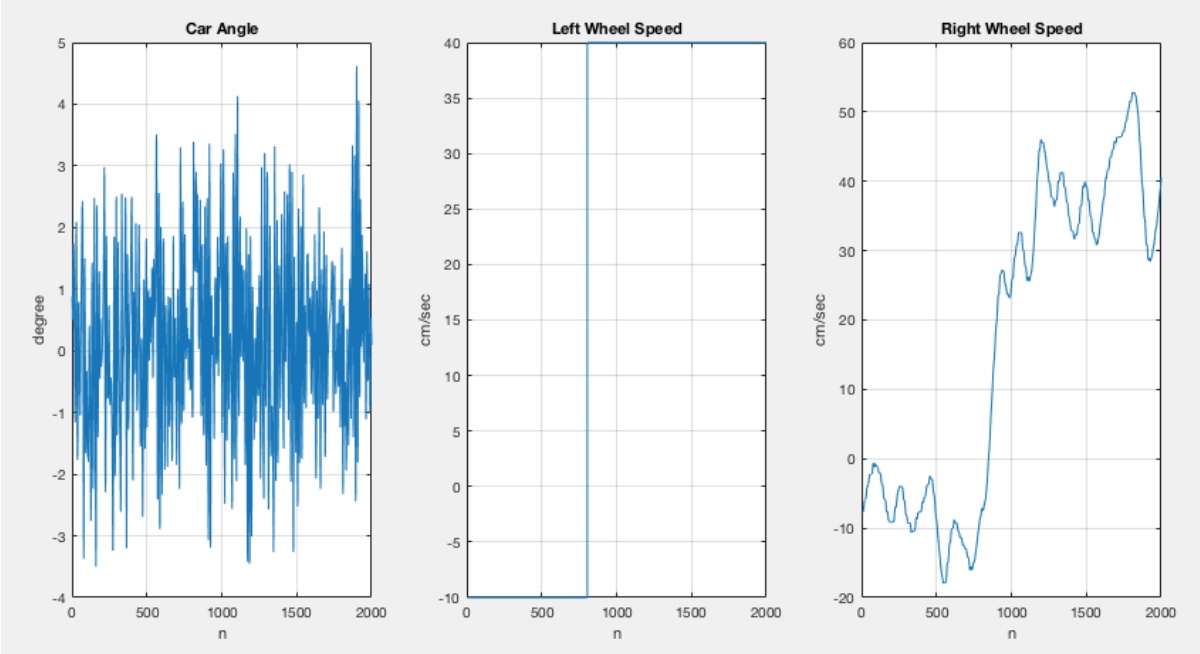
根據圖(4)同樣可以設計出一 PID 控制器，然而同樣遇到將設計好的控制器放到實際的平衡車系統時會無法平衡，因此也無法採用。



圖(4)

所以最後只能嘗試使用 State-feedback。利用試誤法，對於 4 個 State 設計新的極點的位置在$\{-1,-3,-4,-16\}$，由此求出的增益依序為$\{0.330,3.177,506.8,11.64\}$。值得注意的是，由於後兩項增益針對的是單位為 rad 的車傾角以及單位為 rad/s 的車傾角旋轉速度，而實際的平衡車中使用的單位則分別是 degree 以及 degree/s，所以可以預先乘以$\pi/180$，最後可得實際系統中使用的增益$\{0.3302, 3.1772, 8.8459, 0.2031\}$。

將上述參數帶入系統中，可得實際表現圖。圖(5)由左而右依序為車傾角、目標位置以及實際的位置。



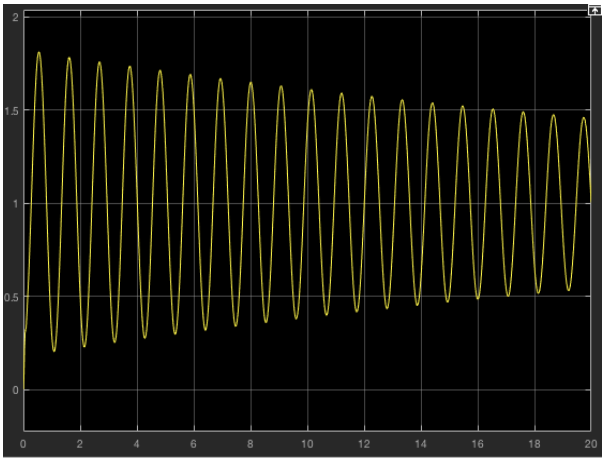
圖(5)

由上圖可見，其車傾角在平衡車體時仍可保持平衡，且可以跑到預定的位置，然而仍會在大約正負 10 公分的範圍內震盪。我想可能和其到達該位置後仍需要做車傾角控制，而選擇的增益無法讓其在很小的範圍內即能做到平衡，才會有此現象。

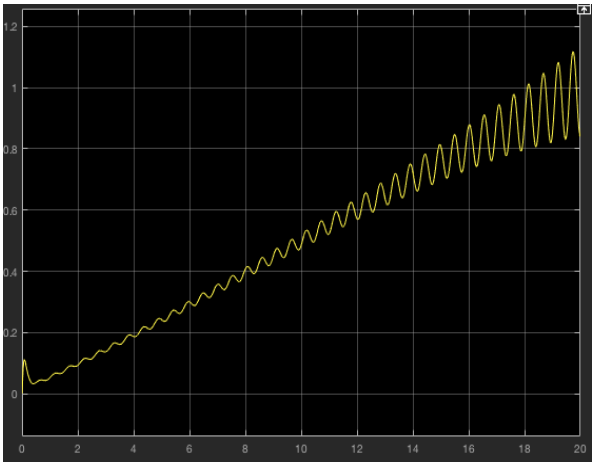
除此之外，車子移動到預定位置也需要較長的時間，如上述移動 50 公分需時大約 10 秒，若調整的速度過快，則因為馬達的輸出有限，無法同時保持平衡車的車傾角維持平衡。

D. Simulation

接下來來使用模擬驗證。下圖(6)與圖(7)為對應使用圖(1)與圖(2)兩種控制方式其模擬的結果。其中圖(1)使用的是 P 控制器，$K_p = 25.8$，而圖(2)使用的則是 PID，$K_p, K_i, K_d = \{1.96, 0.197, 4.87\}$。



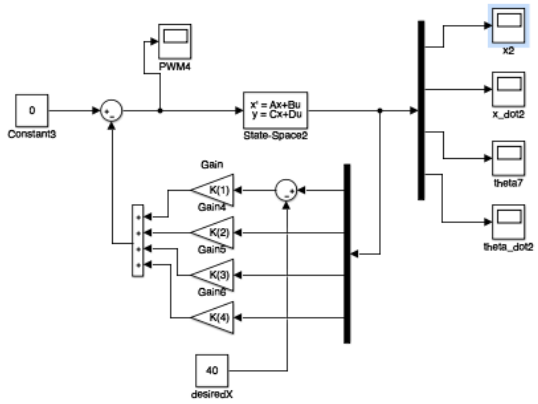
圖(6)



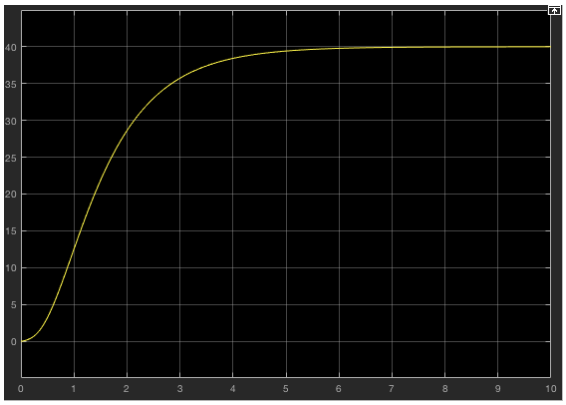
圖(7)

由上圖可見，無論採用何種控制方式，皆會震盪，甚至看起來會不穩定，而實際將參數帶入到平衡車系統時，情況更糟，因此我們才會改採用 State-feedback 來控制。

下圖(8)則為模擬 State-feedback 的方塊圖。而圖(9)則是將模擬車子位置的圖。

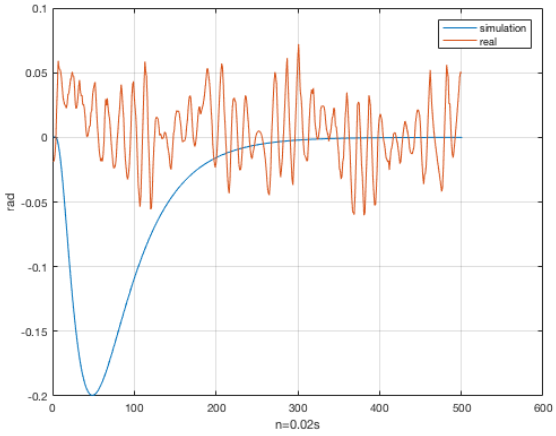


圖(8)

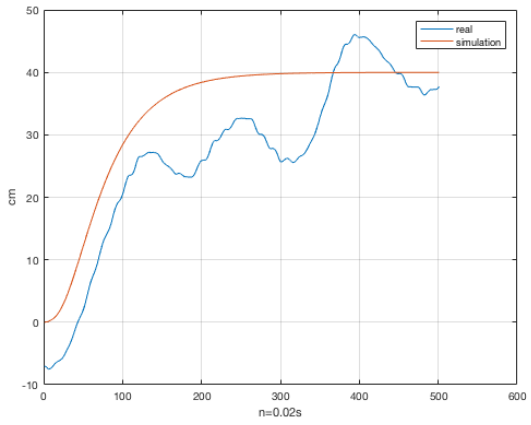


圖(9)

圖(10)則是比較模擬以及實際車傾角的部分，較為特別的是，其實際的車傾角並非如模擬的趨勢變化，且實際的抖動極為明顯，但相對其變化範圍較小。



圖(10)



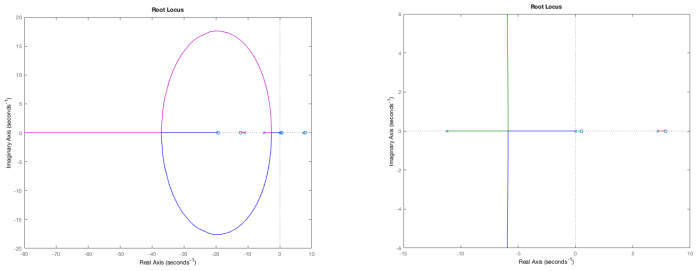
圖(11)

圖(11)則是比較模擬以及實際位置的部分，由圖可見兩者的趨勢相當接近，然而會受制於實際車傾角抖動的情況，而無法一致的往目標位置移動。

E. Questions and Discussions

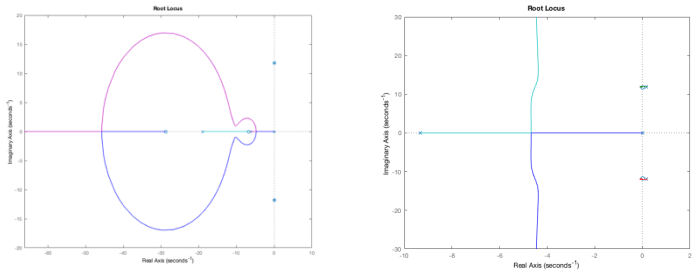
1. 為何第二層ＰＩＤ設計失敗？

i. 講義動態方程式：

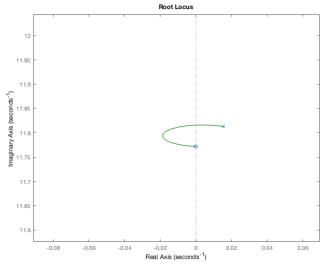


以上兩張分別是在 B 部分的圖(1),(2)的 Root locus 圖，一張圖有包含對ψ的 PID 設計一張沒有，如圖可以看到兩種方式都會產生右半平面的 zero,pole，pole 直接往 zero 跑不到左半平面，導致不論哪種對θ的 PID 控制器配置方式都還是會在 unstable 的狀態，因此做到這裡我們改用 paper 的動態方程式

ii. paper 動態方程式：



一樣，以上兩張分別是對 paper 裡x(位置)的 Root locus 圖，一張圖有包含對ψ的 PID 設計一張沒有，當中二張在虛軸表現類似，有一個虛軸上的 zero 跟很靠近的右半平面 pole，放大該區域圖如下



雖說 pole 可以被拉到左半平面，但是由於虛軸部份過大且實軸過度靠近虛軸，導致震盪幅度太大，overshoot 約略 100%，最終第二層 PID 控制器設計也宣告失敗

2. 為何 state feedback 可以控制，PID 卻不行?

state feedback 是直接可以 assign pole 的位置，直接設計 pole 皆在左半平面，且同時去控制θ, θ', x, x'，然而 PID 設計難以同時對兩者一起設計會互相影響，並且 PID 設

計參數設計限制較多，須滿足一定的規格條件，多一個 zero 可能方程式就無解

3. 為何位置控制不能一次控制在遠處

車子馬達速度有限，當前進的時候馬 pwm 以 255 全力衝刺，當抵達目的地的時候，需要更快的速度來平衡，但極限就是 pwm=255 不可能再大，因此會失去控制

F. Conclusion

本次實驗嘗試過兩種 paper 的 pid 設計，但由於經歷了重重失敗，經過分析覺得有點難達成目標，因此先往 state feedback 去做設計，並且增加可以指定他要在離出發距離多遠處平衡的功能，以及指定速度移動。

G. Improvement of experiment process

將原本的 state space

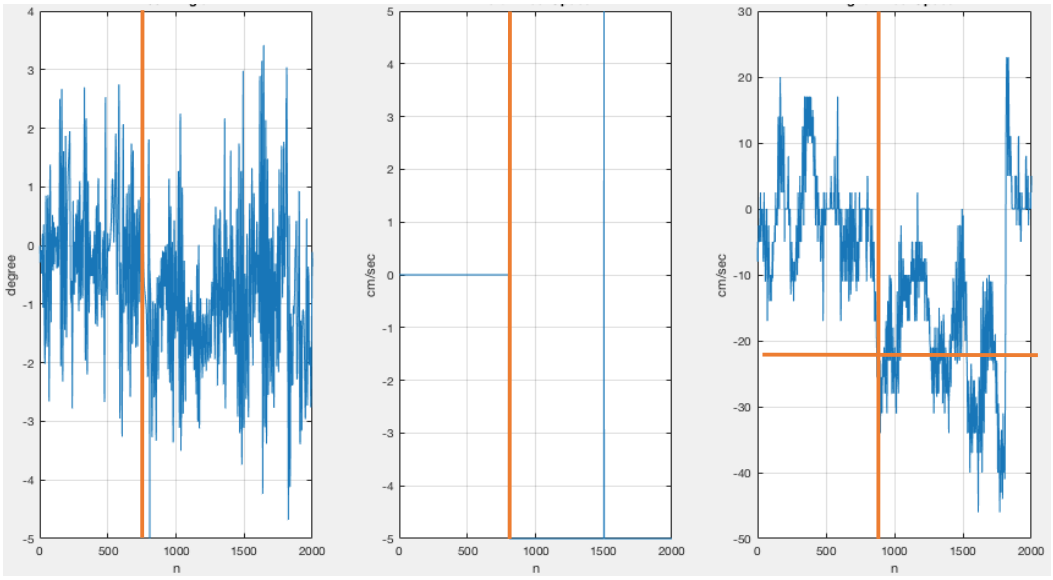
$$\begin{bmatrix} x' \\ x'' \\ \psi' \\ \psi'' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & A_1 & A_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A_3 & A_4 & 0 \end{bmatrix} \begin{bmatrix} x \\ x' \\ \psi \\ \psi'' \end{bmatrix} + \begin{bmatrix} 0 \\ B_1 \\ 0 \\ B_2 \end{bmatrix} v_s$$

將 A 矩陣變成 3*3，去掉位置控制，也就是少去原矩陣第一列與第一行，為了要控制速度所以先不管位置。

$$\begin{bmatrix} x'' \\ \psi' \\ \psi'' \end{bmatrix} = \begin{bmatrix} A_1 & A_2 & 0 \\ 0 & 0 & 1 \\ A_3 & A_4 & 0 \end{bmatrix} \begin{bmatrix} x' \\ \psi \\ \psi'' \end{bmatrix} + \begin{bmatrix} B_1 \\ 0 \\ B_2 \end{bmatrix} v_s$$

跟前面一樣利用一樣的方法去對速度控制，設計 state $\bar{\mathbf{x}} = [x' - v_0; \psi; \psi']$，其中的$v_0$即是目標速度。藉由這樣的設計，可以在不改變原先 state feedback 設計的情況下，改變平衡的位置。接下來只要令輸入為$v_s = -\mathbf{K}\bar{\mathbf{x}}$，並選擇好 Pole 的位置，透過計算$|sI - (A - BK)|$即可得設計好的**K**矩陣。

下圖(12)即為實際測試的圖，由左而右分別是車傾角、目標速度以及實際速度。在 n=800 時設定速度為-5 cm/s。



圖(12)

由圖可見，實際速度並非平衡再-5 cm/s 附近，此原因為我們僅設定目標的速度，卻沒有考慮平衡時的車傾角，因此在計算時同時想保持車傾角為零，所以整體的速度反而

要較大才能使最終結果是可以平衡的。

H. Reference

[1] Modeling and model verification of an intelligent self-balancing two-wheeled vehicle for an autonomous urban transportation system

[2] JOE: A Mobile, Inverted Pendulum

[3] Modeling, Control of a Two-Wheeled Self-Balancing Robot

[4] Design and Control of a Two-Wheel Self-Balancing Robot using the Arduino Microcontroller Board

[5] Balancing a Two-Wheeled Autonomous Robot

[6] Adaptive Neural Network Control of a Self-Balancing Two-Wheeled Scooter