**線性系統導論 期末報告**

**電機4A 00453048 翁偉恆**

目錄：

基本程式說明

選擇系統簡介

斜坡運動

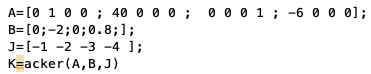
重直圓周運動

斜拋平衡運動

結論

1. 基本程式說明

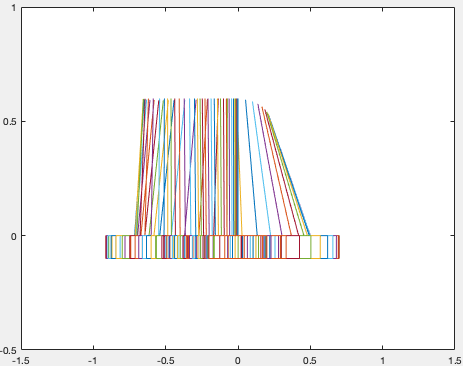
討論K值



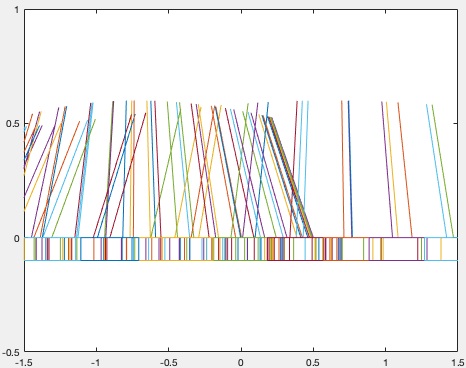
不同極點下車體運動的結果**(ack.m、IP\_orinial.m，手動測)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 極點 | K值 | 結果 |
| 極點在左半平面 | -1,-2,-3,-4 | -37.98 ,-6,-1.2,-2.5 | 車子擺盪後停住。 |
| 極點在虛軸 | 1j,2j,-1j,-2j | -22.58,0,-0.2,0 | 車子來回擺盪不停。 |
| 極點在右半平面 | 1,2,3,4 | -37.98,6,-1.2,2.5 | 車子跑出視窗  ，不再回來。 |

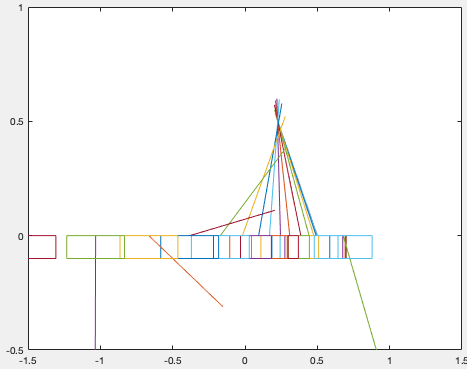
極點=-1,-2,-3,-4



極點=1j,2j,-1j,-2j



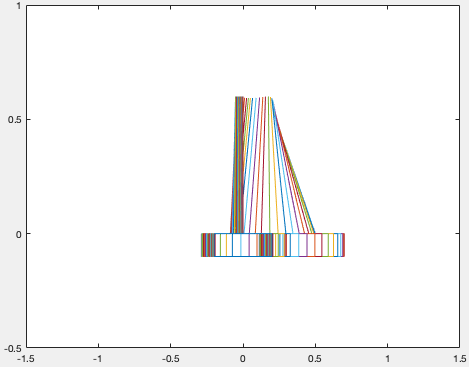
極點=1,2,3,4



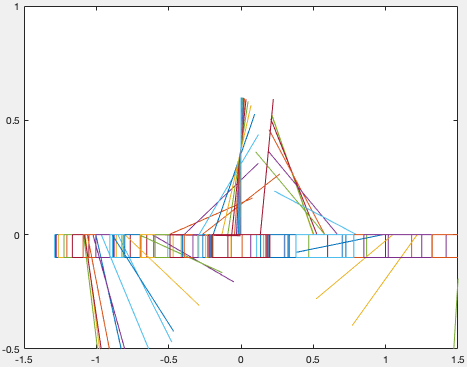
極點遠離原點及極點有虛部對車體運動的影響**(ack.m、IP\_orinial.m，手動測)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 極點 | K值 | 結果 |
| 比較極點實部大小對車體的影響 | -3,-4,-5,-6 | -86.7 ,-15.84 ,  -18 ,-17.1 | 實部數值較大的車體車體較快停住。 |
| -30,-40,-50,-60 | -77970 ,-6930 , -180000 ,-17100 |
| 比較極點虛部大小對車體的影響 | 3j,4j,-3j,-4j | -35.3800 ,0  -7.2000 ,0 | 虛部數值較大的車體車體較快停住。 |
| 30j,40j,-30j,-40j | -30070 ,0  -72000 ,0 |
| 比較極點虛部比例對車體的影響 | -2+1j,-2+2j,-2-1j,-2-2j | -35.3000 ,-5.0400, -2.0000 ,-2.6000 | 虛部比例較大的車體超出視窗較多且車體停住花費的時間也較長。 |
| -2+10j,-2+20j,-2-10j,-2-20j | -1122.3 ,-44.6 ,  -2100.8,-101.6 |

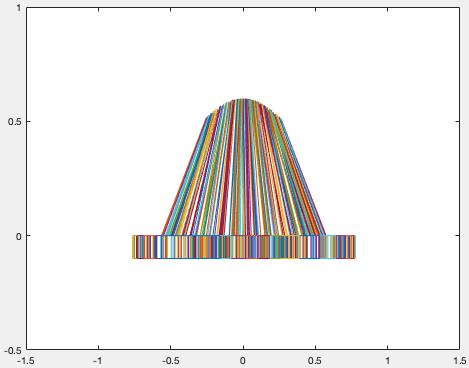
極點=-3,-4,-5,-6



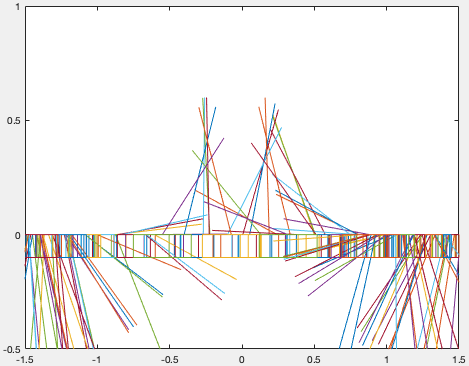
極點=-30,-40,-50,-60



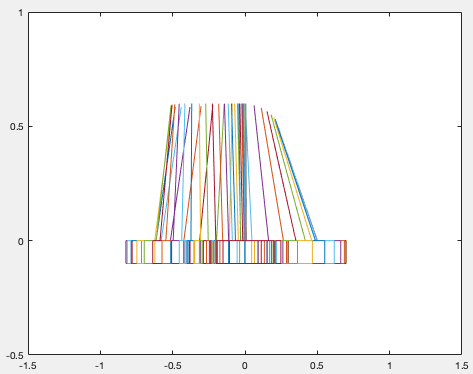
極點=3j,4j,-3j,-4j



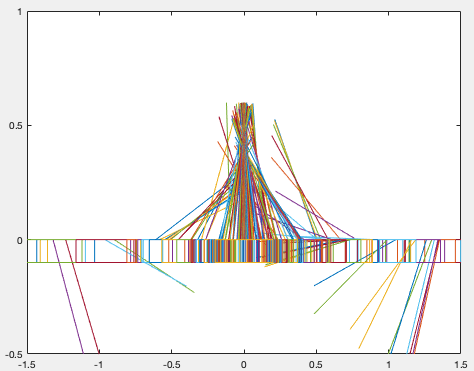
極點=30j,40j,-30j,-40j



極點=-2+1j,-2+2j,-2-1j,-2-2j



極點=-2+10j,-2+20j,-2-10j,-2-20j



當極點在實軸上且遠離原點時，會使車體較快停下，這是因極點遠離原點會使輸入的波形較快達到穩態(反應時間短)，而使車體較快停住，但車體移動範圍較大。

當極點在虛軸上的結果與在實軸上雷同。

因此極點越遠離原點的車體，越快停住。

當極點虛部比例較大的車體，停住所花費的時間較長，這是因極點遠離實軸，使輸入波形的阻尼比降低，因此使波形的overshoot變大，而需要達到穩態的時間就較長，因此車體停下就需花費較長時間。

畫車子

**function printCar(initral\_x,initral\_y,stick\_long,stick\_th,car\_angle)**

**xp=[initral\_x initral\_x+stick\_long\*sin(stick\_th-car\_angle)]; % 桿子**

**yp=[initral\_y initral\_y+stick\_long\*cos(stick\_th-car\_angle)];**

**plot(xp,yp,'Color',[1,0,0])**

**hold on**

**the=0:0.01:2\*pi;% 桿子頂端球**

**x=initral\_x+stick\_long\*sin(stick\_th-car\_angle)+(1/10)\*stick\_long\*cos(the);**

**y=initral\_y+stick\_long\*cos(stick\_th-car\_angle)+(1/10)\*stick\_long\*sin(the);**

**plot(x,y,'Color',[1,0,0])**

**xb=[initral\_x-(3/5)\*stick\_long\*cos(car\_angle)+(3/5)\*stick\_long\*sin(car\_angle) initral\_x+(3/5)\*stick\_long\*cos(car\_angle)+(3/5)\*stick\_long\*sin(car\_angle) initral\_x+(3/5)\*stick\_long\*cos(car\_angle) initral\_x-(3/5)\*stick\_long\*cos(car\_angle) initral\_x-(3/5)\*stick\_long\*cos(car\_angle)+(3/5)\*stick\_long\*sin(car\_angle)];% 車身方塊**

**yb=[initral\_y-(3/5)\*stick\_long\*sin(car\_angle)-(3/5)\*stick\_long\*cos(car\_angle) initral\_y+(3/5)\*stick\_long\*sin(car\_angle)-(3/5)\*stick\_long\*cos(car\_angle) initral\_y+(3/5)\*stick\_long\*sin(car\_angle) initral\_y-(3/5)\*stick\_long\*sin(car\_angle) initral\_y-(3/5)\*stick\_long\*sin(car\_angle)-(3/5)\*stick\_long\*cos(car\_angle)];**

**plot(xb,yb,'Color',[0,0,1])**

**plot(initral\_x+(7/10)\*stick\_long\*sin(car\_angle)+((2/5)\*stick\_long\*cos(car\_angle))+(1/10)\*stick\_long\*cos(the),initral\_y+((2/5)\*stick\_long\*sin(car\_angle))-(7/10)\*stick\_long\*cos(car\_angle)+(1/10)\*stick\_long\*sin(the),'Color',[0,0,1])**

**plot(initral\_x+(7/10)\*stick\_long\*sin(car\_angle)-((2/5)\*stick\_long\*cos(car\_angle))+(1/10)\*stick\_long\*cos(the),initral\_y-((2/5)\*stick\_long\*sin(car\_angle))-(7/10)\*stick\_long\*cos(car\_angle)+(1/10)\*stick\_long\*sin(the),'Color',[0,0,1])**

**fill(x,y,'b')**

**fill(xb,yb,'g')**

**fill(initral\_x+(7/10)\*stick\_long\*sin(car\_angle)+((2/5)\*stick\_long\*cos(car\_angle))+(1/10)\*stick\_long\*cos(the),initral\_y+((2/5)\*stick\_long\*sin(car\_angle))-(7/10)\*stick\_long\*cos(car\_angle)+(1/10)\*stick\_long\*sin(the),'r')**

**fill(initral\_x+(7/10)\*stick\_long\*sin(car\_angle)-((2/5)\*stick\_long\*cos(car\_angle))+(1/10)\*stick\_long\*cos(the),initral\_y-((2/5)\*stick\_long\*sin(car\_angle))-(7/10)\*stick\_long\*cos(car\_angle)+(1/10)\*stick\_long\*sin(the),[1 0 0])**

**axis([-2\*stick\_long 2\*stick\_long -2\*stick\_long 2\*stick\_long])**

**axis('square')**

**grid on;**

**return**

以上程式碼為畫出各個角度倒單擺的程式，最終狀態為垂直於自走車水平方向，參數中initral\_x設為倒單擺的線與自走車連接的x位置。

initral\_y設為倒單擺的線與自走車連接的y位置。

stick\_long為倒單擺的線長度。

stick\_th為當下倒單擺與垂直於地面的中垂線的夾角。

car\_angle為自走車水平方向與地面的夾角。

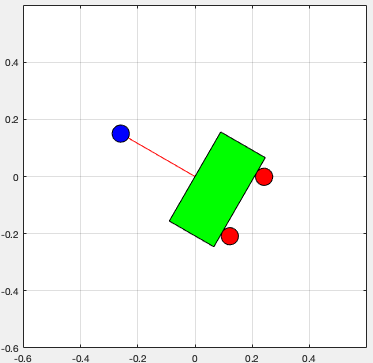
程式碼中

xp和yp為倒單擺線的x、y位置範圍。

x、y為倒單擺上的球函式。

xb和yb為自走車的頂點集合

畫出來的圖如下：以自走車水平方向與地面夾角為60度為例**(printCar.m)**



**function printStraightCar(initral\_x,initral\_y,stick\_long,stick\_th,car\_angle)**

**xp=[initral\_x initral\_x+stick\_long\*sin(stick\_th)]; % 桿子**

**yp=[initral\_y initral\_y+stick\_long\*cos(stick\_th)];**

**plot(xp,yp,'Color',[1,0,0])**

**hold on**

**the=0:0.01:2\*pi;% 桿子頂端球**

**x=initral\_x+stick\_long\*sin(stick\_th)+(1/10)\*stick\_long\*cos(the);**

**y=initral\_y+stick\_long\*cos(stick\_th)+(1/10)\*stick\_long\*sin(the);**

**plot(x,y,'Color',[1,0,0])**

**xb=[initral\_x-(3/5)\*stick\_long\*cos(car\_angle)+(3/5)\*stick\_long\*sin(car\_angle) initral\_x+(3/5)\*stick\_long\*cos(car\_angle)+(3/5)\*stick\_long\*sin(car\_angle) initral\_x+(3/5)\*stick\_long\*cos(car\_angle) initral\_x-(3/5)\*stick\_long\*cos(car\_angle) initral\_x-(3/5)\*stick\_long\*cos(car\_angle)+(3/5)\*stick\_long\*sin(car\_angle)];% 車身方塊**

**yb=[initral\_y-(3/5)\*stick\_long\*sin(car\_angle)-(3/5)\*stick\_long\*cos(car\_angle) initral\_y+(3/5)\*stick\_long\*sin(car\_angle)-(3/5)\*stick\_long\*cos(car\_angle) initral\_y+(3/5)\*stick\_long\*sin(car\_angle) initral\_y-(3/5)\*stick\_long\*sin(car\_angle) initral\_y-(3/5)\*stick\_long\*sin(car\_angle)-(3/5)\*stick\_long\*cos(car\_angle)];**

**plot(xb,yb,'Color',[0,0,1])**

**plot(initral\_x+(7/10)\*stick\_long\*sin(car\_angle)+((2/5)\*stick\_long\*cos(car\_angle))+(1/10)\*stick\_long\*cos(the),initral\_y+((2/5)\*stick\_long\*sin(car\_angle))-(7/10)\*stick\_long\*cos(car\_angle)+(1/10)\*stick\_long\*sin(the),'Color',[0,0,1])**

**plot(initral\_x+(7/10)\*stick\_long\*sin(car\_angle)-((2/5)\*stick\_long\*cos(car\_angle))+(1/10)\*stick\_long\*cos(the),initral\_y-((2/5)\*stick\_long\*sin(car\_angle))-(7/10)\*stick\_long\*cos(car\_angle)+(1/10)\*stick\_long\*sin(the),'Color',[0,0,1])**

**fill(x,y,'b')**

**fill(xb,yb,'g')**

**fill(initral\_x+(7/10)\*stick\_long\*sin(car\_angle)+((2/5)\*stick\_long\*cos(car\_angle))+(1/10)\*stick\_long\*cos(the),initral\_y+((2/5)\*stick\_long\*sin(car\_angle))-(7/10)\*stick\_long\*cos(car\_angle)+(1/10)\*stick\_long\*sin(the),'r')**

**fill(initral\_x+(7/10)\*stick\_long\*sin(car\_angle)-((2/5)\*stick\_long\*cos(car\_angle))+(1/10)\*stick\_long\*cos(the),initral\_y-((2/5)\*stick\_long\*sin(car\_angle))-(7/10)\*stick\_long\*cos(car\_angle)+(1/10)\*stick\_long\*sin(the),[1 0 0])**

**axis([-4\*stick\_long 4\*stick\_long -2\*stick\_long 6\*stick\_long])**

**axis('square')**

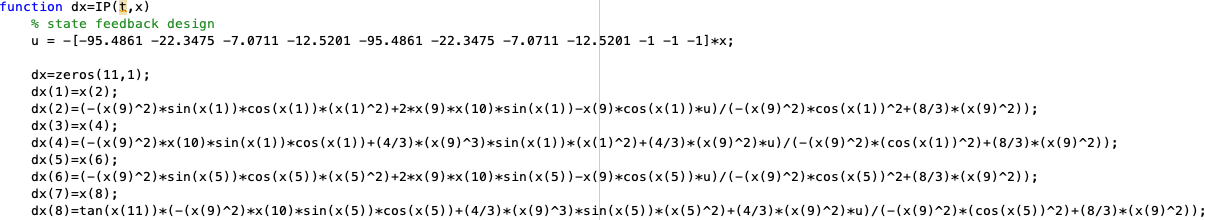
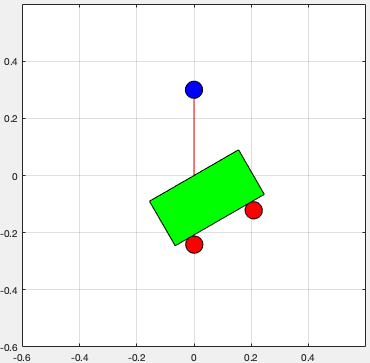
**grid on;**

**return**

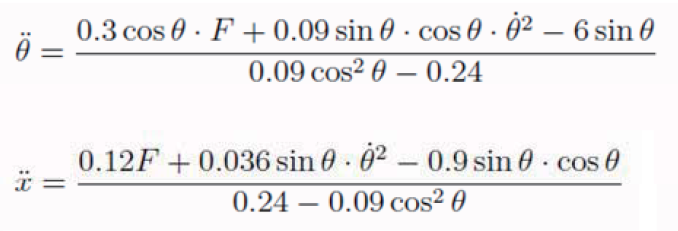
以上程式碼為畫出各個角度倒單擺的程式，最終狀態為垂直於地面，參數與

程式碼中的變數與垂直於自走車水平方向幾乎一模一樣，只是倒單擺的線連接球的頂點由initral\_x+stick\_long\*sin(stick\_th-car\_angle)與initral\_y+stick\_long\*cos(stick\_th-car\_angle)改成initral\_x+stick\_long\*sin(stick\_th)與initral\_y+stick\_long\*cos(stick\_th)

畫出的結果如下圖：以自走車水平方向與地面夾角為60度為例**(printStraightCar.m)**



以上程式碼為這次作業最重要的部分，這邊描述了本次模擬系統的狀態方程式**(IP.m)**，其中x(1)~x(8)為微分方程式的狀態變數，x(9)、x(10)、x(11)純粹當成微分方程式需要用到的常數(因為我無法用其他方法代入進去ode23()這個函式)，上段程式碼寫成微分方程式的形式為



以及

其中y的二次微分與x的二次微分呈現tanα的關係，其中α為自走車水平方向與地面的夾角，而θ與φ的方程式一樣，又為何要重複寫2次呢？是因為要加快程式的收斂速度，如果沒有φ的話，dx(0)~dx(4)與dx(5)~dx(8)就無法完全分隔開來，可能會有線性相依的情狀，不只會使收斂速度變慢，且不好設計。

程式碼雖然有x(9)、x(10)、x(11)，但哪只是傳l(倒單擺線長度)、g(重力加速度)、tanα這些參數用的，所以以上程式碼可以等效於一個8\*8的矩陣。

我猜測上面微分方程式寫成矩陣形式會如下：

dx=

如果不多兩個φ的方程式，則矩陣會變成

dx=

θ的兩次為方同時被θ、x、y所影響，比較難設計是否收斂。

初始x(0)=[ -1 0 0 0 -1 0 0 0]

且每1個row之中，只有一個column有非0值，不難看出此系統可完全控制

另外我個人將範例的運動方程式從頭推倒了幾遍，將原本的範例程式碼IP\_NL.m改得更為靈活，原本的程式碼有假設

倒單擺車質量=1(kg)

倒單擺質量=1(kg)

倒單擺線長度=0.3(m)

重力加速度=10()

轉動慣量=0.03()

而我改寫過後的微分方程式將倒單擺線長度、重力加速度設為可改變輸入的參數，假設

倒單擺車質量=1(kg)

倒單擺質量=1(kg)

由於轉動慣量=1/3\*(倒單擺質量\*)，假設倒單擺質量=1(kg)之後，

轉動慣量=1/3\*()，而倒單擺線長度為可改變的參數，所以在程式中可直接用倒單擺線長度的關係表示轉動慣量。

不過老師原本程式的K值是假設倒單擺線長度=0.3(m)時設計的，所以如果需要改倒單擺線長度的話，需要先轉換成矩陣模式送到ack.m，用acker()函式算出K值，也就是u需要用x乘以一堆係數來獲得的係數。

以下為時間(t，橫軸)與各個狀態()的關係圖**(print\_IP.m)**

其中

紅線代表與t的關係 或 與t的關係

綠線代表與t的關係 或 與t的關係

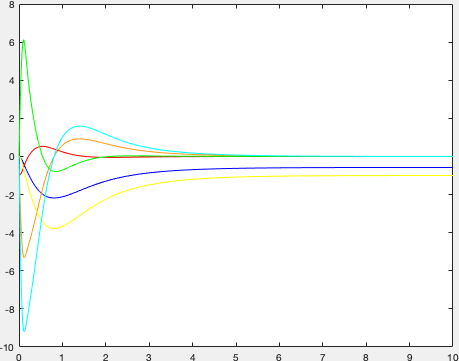
藍線代表與t的關係

橘線代表與t的關係

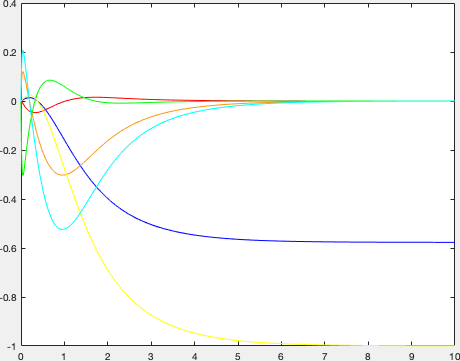
黃線代表與t的關係

黃線代表與t的關係

初始狀態x=[-1 0 0 0 -1 0 0 0]

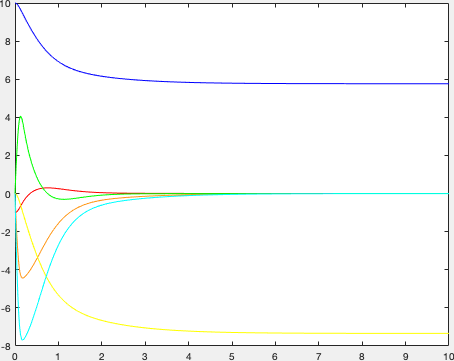


初始狀態x=[0 0 0 0 0 0 0 0]

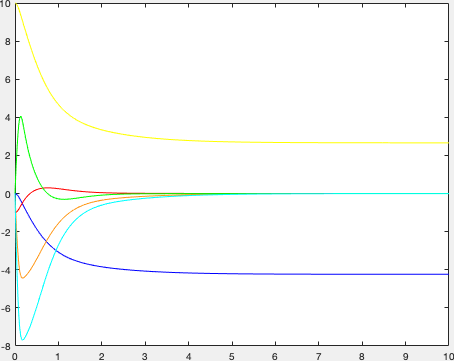


由上面兩張圖可以看出改動的初始值時，發生變動，進而導致產生劇烈變動，會嚴重降低穩定性。

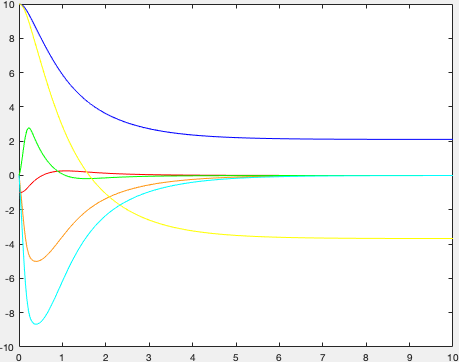
初始狀態x=[-1 0 10 0 -1 0 0 0]



初始狀態x=[-1 0 0 0 -1 0 10 0]

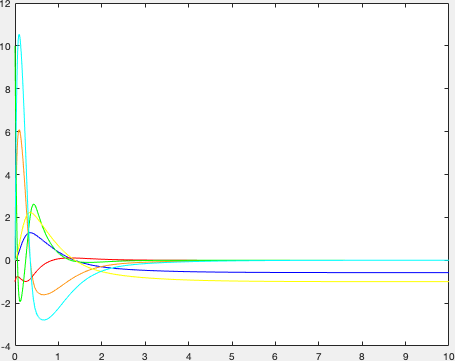


初始狀態x=[-1 0 10 0 -1 0 10 0]



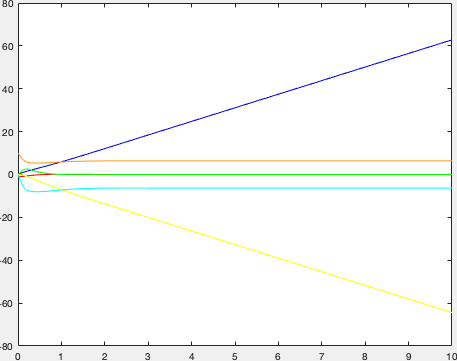
由上面三張圖可以看出改動的初始值時，幾乎不太會有變動，即使有變動幅度也算是極小，雖然會導致產生變動，但除了穩定時的位置改變之外，只要改動幅度不大，基本上不引響穩定。

初始狀態x=[-1 10 0 0 -1 10 0 0]

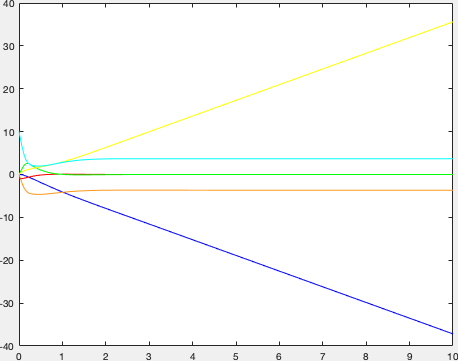


由上面一張圖與初始狀態x=[-1 0 0 0 -1 0 0 0]的圖對比後可以看出，增加的初始值時，亦會增加，發生變動的幅度需要看的幅度來決定，在初始狀態x=[-1 10 0 0 -1 10 0 0]的情況下，雖然有變動，但不影響的穩定性。

初始狀態x=[-1 0 0 10 -1 0 0 0]



初始狀態x=[-1 0 0 0 -1 0 0 10]



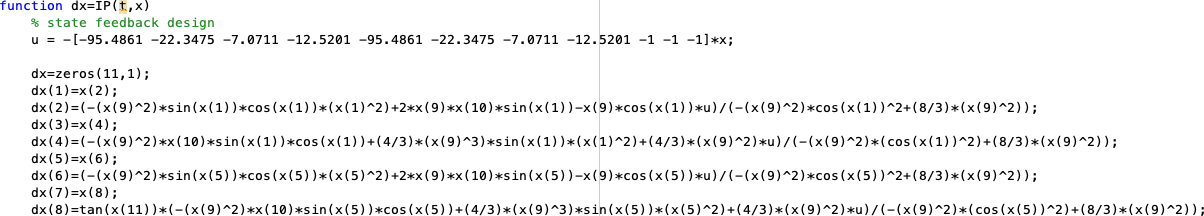
由上面兩張圖與初始狀態x=[-1 0 0 0 -1 0 0 0]的圖對比後可以看出，增加或增加的初始值時，的影響幅度，需要看的增加幅度來決定，但一定會大幅提高的變化量，嚴重降低穩定性，如圖，雖然數值上好像沒有發散(因為程式有跑出來)，但畫出來的倒單擺車一定是暴衝的。

(二)選擇系統簡介：

我修改了老師的IP\_NL.m，使l(倒單擺線長度)、g(重力加速度)，為變數。

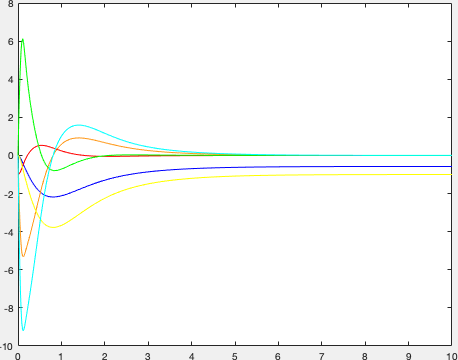
本次作業中，我選用兩種系統：

1. l(倒單擺線長度)=0.3(m)、g(重力加速度)=9.8()

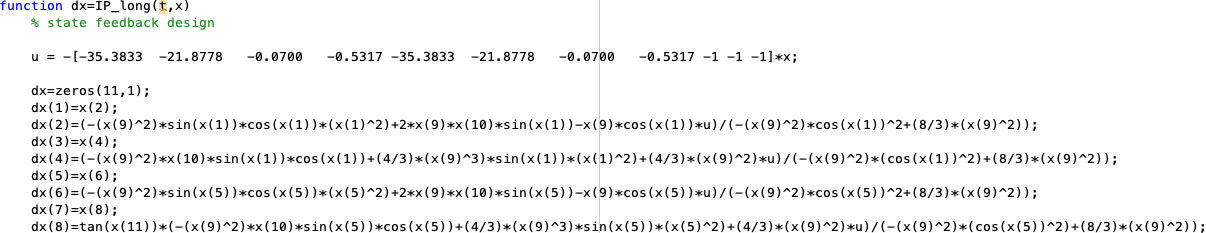


極點為老師預設

選擇t=0~10(s)之間的[t,x]關係圖如下：**(IP.m)**

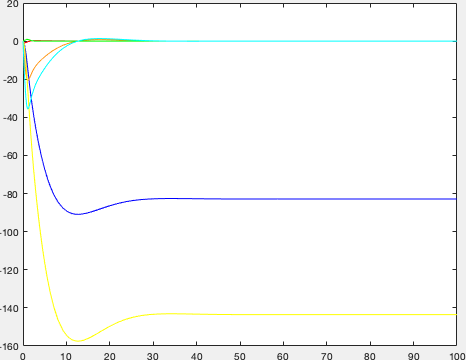


1. l(倒單擺線長度)=5(m)、g(重力加速度)=9.8()

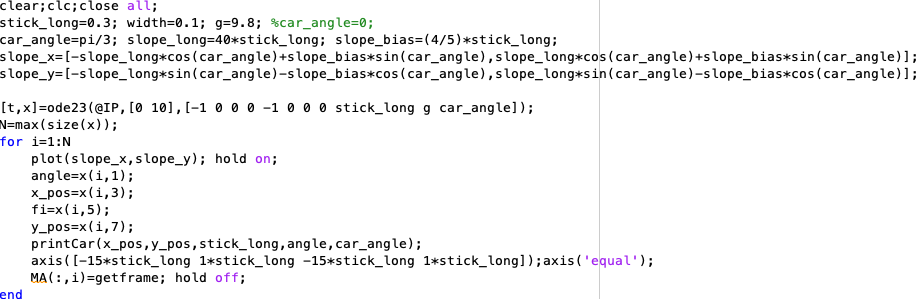


極點選擇=[-0.4 -0.5 -0.6 -0.7]

選擇t=0~10(s)之間的[t,x]關係圖如下：**(IP\_long.m)**



(三)斜坡運動：



以上為模擬自走車在斜坡上保持倒單擺垂直於自走車水平方向的程式碼，[t,x]=ode23(@IP,[0 10],[-1 0 0 0 -1 0 0 0 stick\_long g car\_angle]);是在解微分方程式，其中stick\_long、g、car\_angle在這段程式碼中，純粹最當參數用，完全不影響微分方程式。

這段程式碼的意思是從t=0時，x=[-1 0 0 0 -1 0 0 0 stick\_long g car\_angle]為初始條件下，代入微分方程IP，每單位時間解出當下的狀態變數x，直到t=10。

微分方程解出之後，x矩陣內存了從t=0到收斂時，所有t當下的

[]

亦即

x=

[

……

]

其中，N為收斂時，經過多少個單位時間。

然後跑迴圈1:N

變數angle會取每單位時間的θ值、

變數x\_pos會取每單位時間的x值、

變數fi會取每單位時間的φ值、

變數y\_pos會取每單位時間的y值，

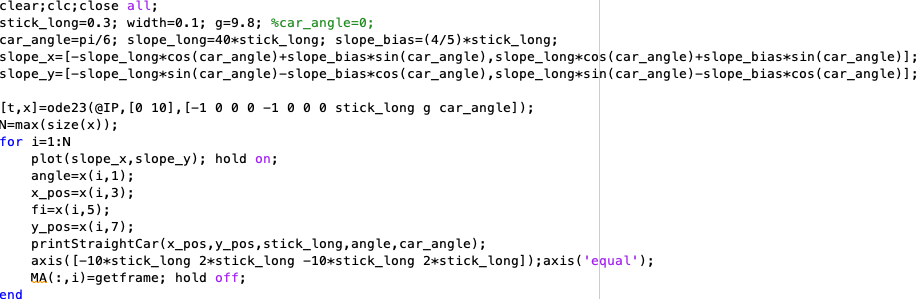
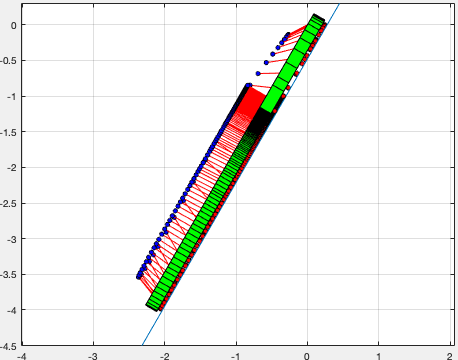
取完值後，把變數x\_pos、y\_pos、angle，以及常數stick\_long、car\_angle代入自創函式printCar()畫圖出來，其中，stick\_long為倒單擺的線長，car\_angle為自走車水平方向與地面的夾角。

最終狀態為倒單擺線與自走車水平方向垂直。

以系統參數l(倒單擺線長度)=0.3(m)、g(重力加速度)=9.8()、選擇時間區間t=0~10(s)、

極點為老師預設，

畫出來的圖如下：以自走車水平方向與地面夾角為60度為例**(slopeCar.m)**



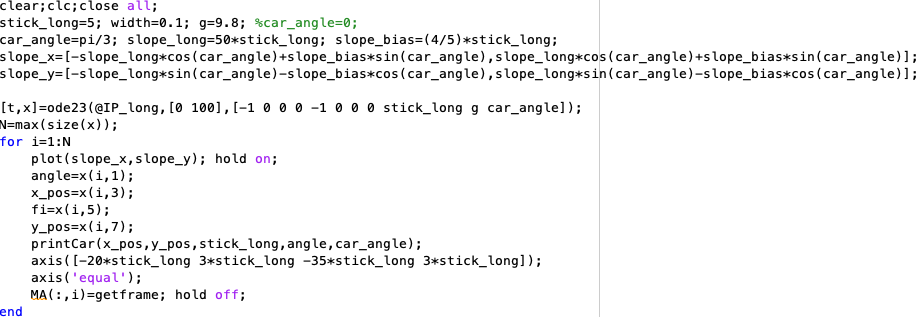
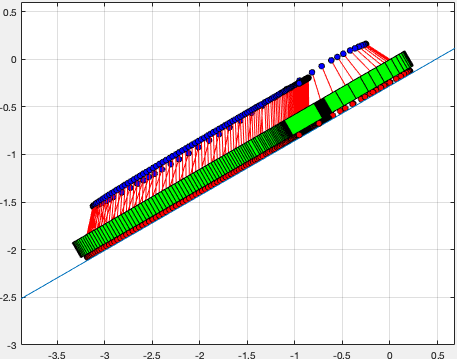
以上為模擬自走車在斜坡上保持倒單擺垂直於地面的程式碼，

最終狀態從原本的倒單擺線與自走車水平方向垂直，改進最終狀態為倒單擺線與地面垂直。

以系統參數l(倒單擺線長度)=0.3(m)、g(重力加速度)=9.8()、選擇時間區間t=0~10(s)、

極點為老師預設，

畫出來的圖如下：以自走車水平方向與地面夾角為30度為例**(slopeStraightCar.m)**



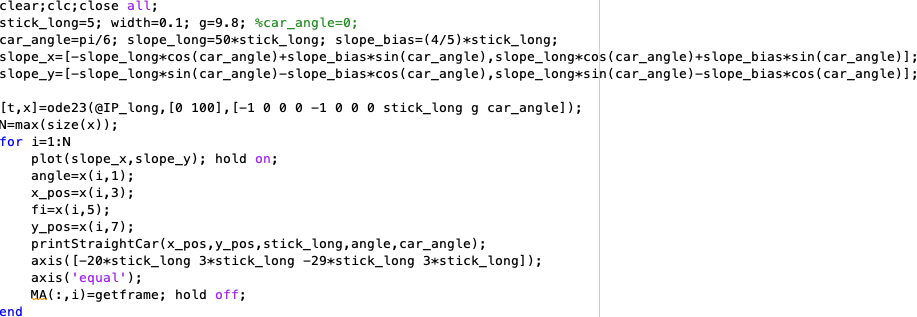
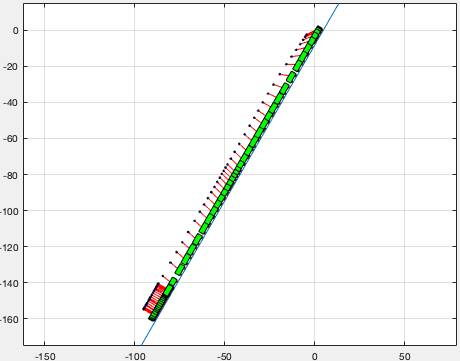
以上為模擬自走車在斜坡上保持倒單擺垂直於自走車水平方向的程式碼，

最終狀態為倒單擺線與自走車水平方向垂直。

以系統參數l(倒單擺線長度)=5(m)、g(重力加速度)=9.8()、選擇時間區間t=0~100(s)、

極點選擇=[-0.4 -0.5 -0.6 -0.7]

畫出來的圖如下：以自走車水平方向與地面夾角為60度為例**(longSlopeCar.m)**



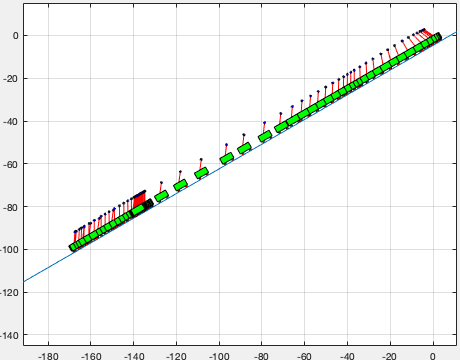
以上為模擬自走車在斜坡上保持倒單擺垂直於自走車水平方向的程式碼，

最終狀態從原本的倒單擺線與自走車水平方向垂直，改進最終狀態為倒單擺線與地面垂直。

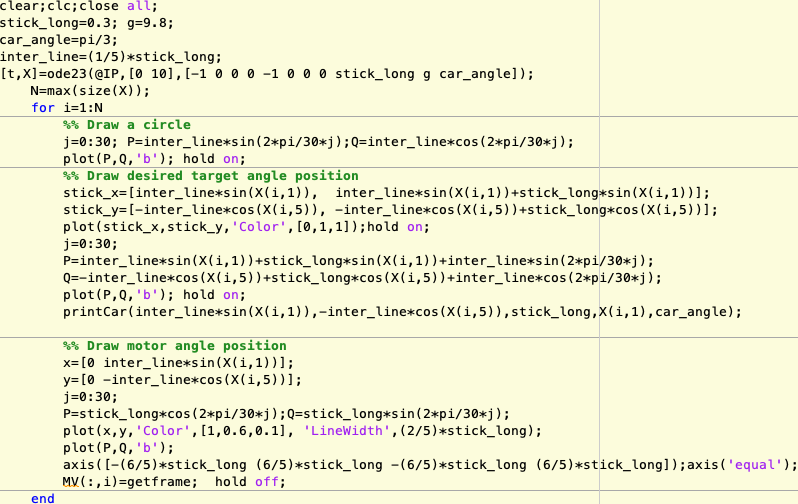
以系統參數l(倒單擺線長度)=5(m)、g(重力加速度)=9.8()、選擇時間區間t=0~100(s)、

極點選擇=[-0.4 -0.5 -0.6 -0.7]

畫出來的圖如下：以自走車水平方向與地面夾角為30度為例**(longSlopeStraightCar.m)**



(四)垂直圓周運動：



以上為模擬自走車在垂直圓面上上保持倒單擺垂直於自走車水平方向的程式碼，ode23()在解微分方程式，微分方程解出之後，

第8行計算垂直圓面、

第9行畫出垂直圓面、

第11、12行在計算倒單擺線當下的形狀、

第13行畫出倒單擺線、

第14行計算倒單擺頂點的球、

第15行畫出倒單擺頂點的球、

第13~15行畫出的倒單擺垂直於地面，而第16行畫出的倒單擺車的倒單擺，垂直於自走車水平方向。可對兩者進行比較且看出關係。

第16之後畫的是自走車原點(倒單擺與自走車相連之處)，所預期會遵守的路徑，結果會是一個圓。

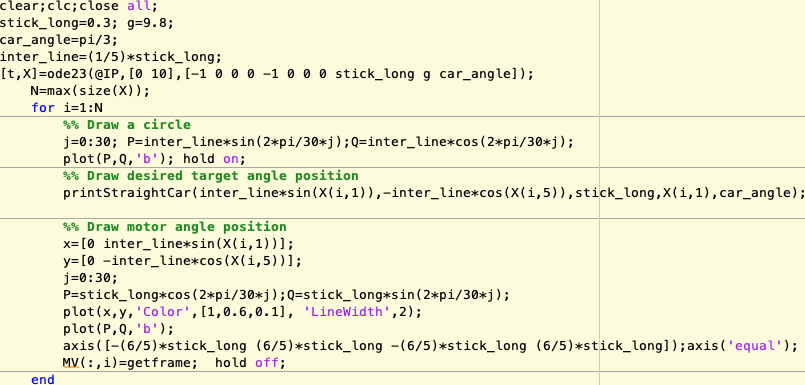
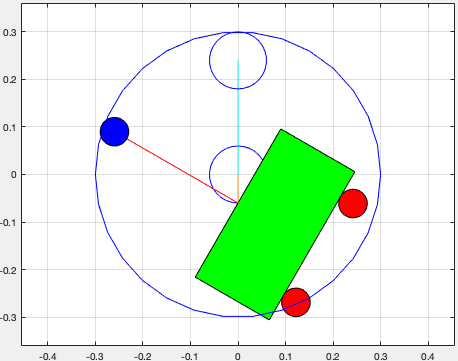
以上程式有參考老師的motor\_mv.m

最終狀態為倒單擺線與自走車水平方向垂直。

以系統參數l(倒單擺線長度)=0.3(m)、g(重力加速度)=9.8()、選擇時間區間t=0~10(s)、

極點為老師預設，

畫出來的圖如下：以自走車水平方向與地面夾角為60度為例**(radCar.m)**



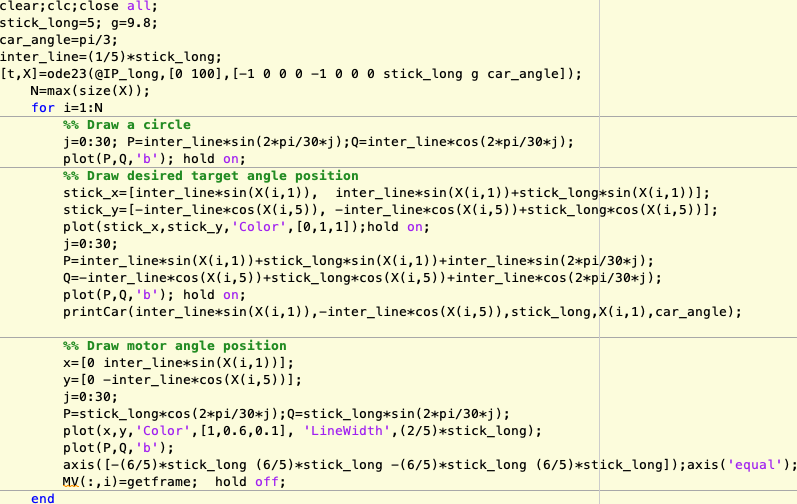
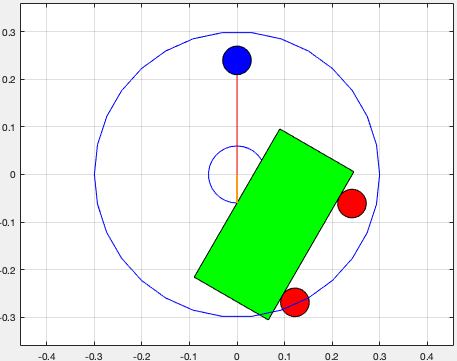
以上為模擬自走車在垂直圓面上上保持倒單擺垂直於地面的程式碼，

最終狀態從原本的倒單擺線與自走車水平方向垂直，改進最終狀態為倒單擺線與地面垂直。

以系統參數l(倒單擺線長度)=0.3(m)、g(重力加速度)=9.8()、選擇時間區間t=0~10(s)、

極點為老師預設，

畫出來的圖如下：以自走車水平方向與地面夾角為60度為例**(radStraightCar.m)**



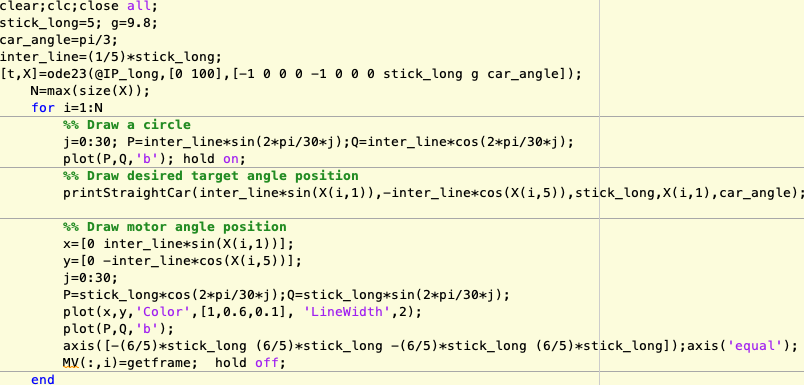
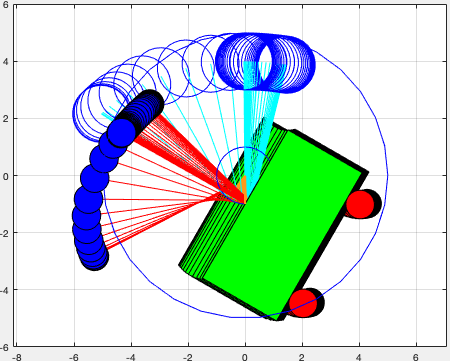
以上為模擬自走車在垂直圓面上上保持倒單擺垂直於地面的程式碼，

最終狀態為倒單擺線與自走車水平方向垂直。

以系統參數l(倒單擺線長度)=5(m)、g(重力加速度)=9.8()、選擇時間區間t=0~100(s)、

極點選擇=[-0.4 -0.5 -0.6 -0.7]

畫出來的圖如下：以自走車水平方向與地面夾角為60度為例**(longRadCar.m)**



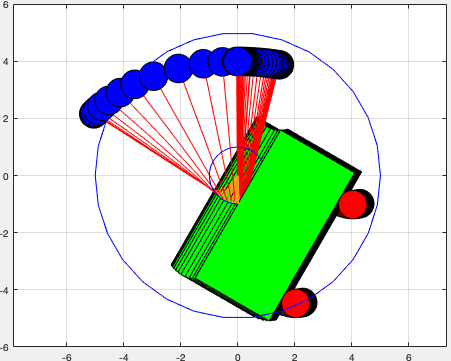
以上為模擬自走車在垂直圓面上上保持倒單擺垂直於地面的程式碼，

最終狀態從原本的倒單擺線與自走車水平方向垂直，改進最終狀態為倒單擺線與地面垂直。

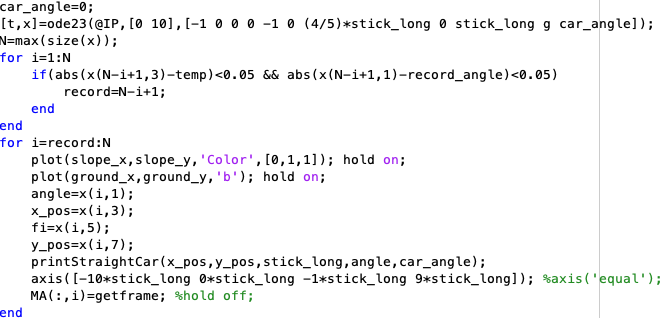
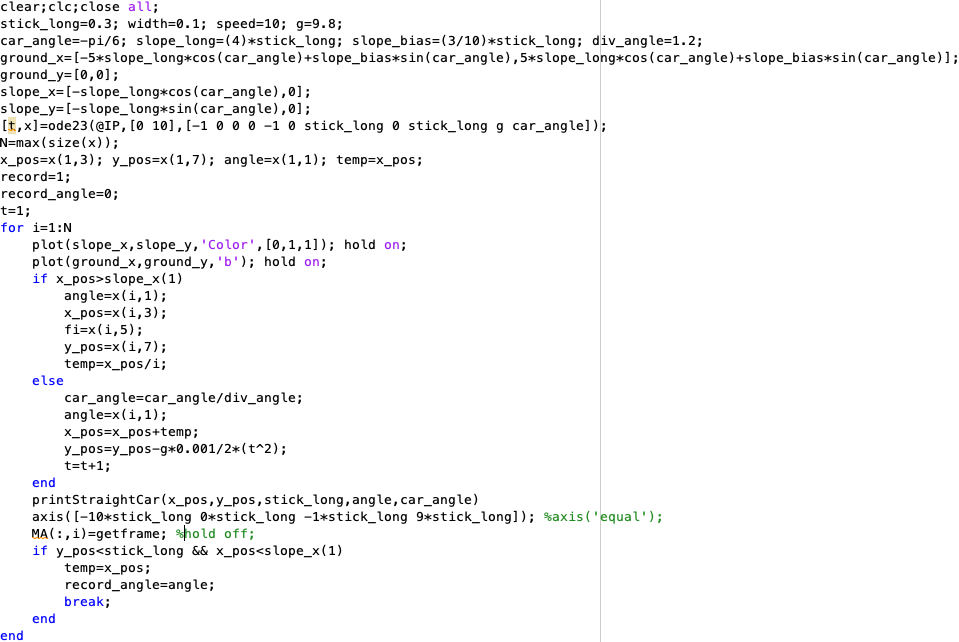
以系統參數l(倒單擺線長度)=5(m)、g(重力加速度)=9.8()、選擇時間區間t=0~100(s)、

極點選擇=[-0.4 -0.5 -0.6 -0.7]

畫出來的圖如下：以自走車水平方向與地面夾角為60度為例**(longRadStraightCar.m)**



(五)斜拋平衡運動：



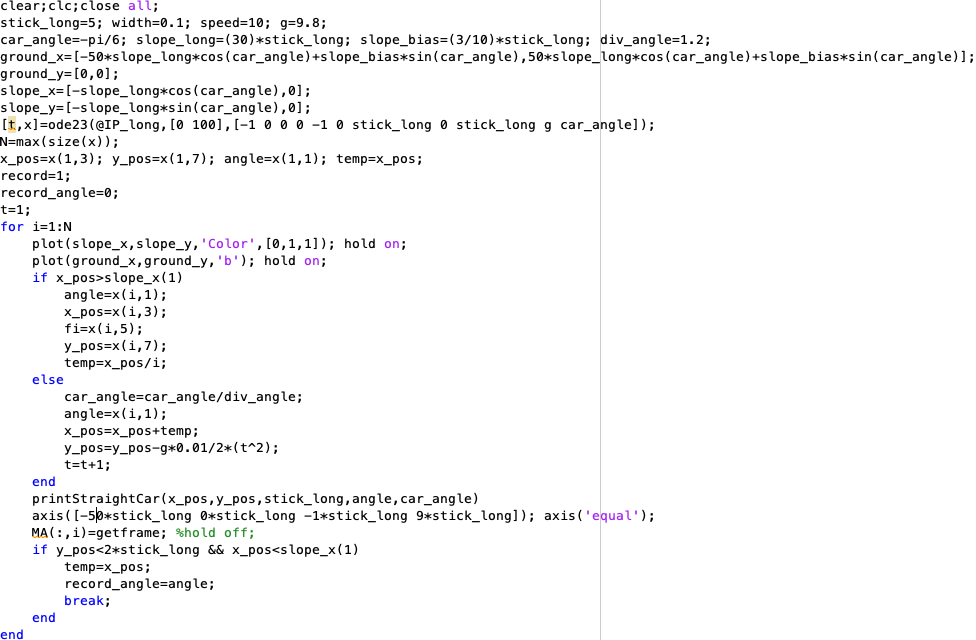
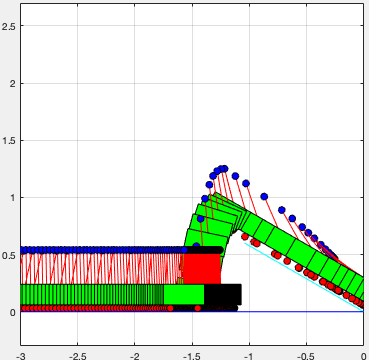
這個運動模擬斜拋的倒單擺平衡，所謂的斜拋，是指經過斜坡運動，滑出斜坡後，y方向進行自由落體，x方向保持原速度等速前進，為了保持平衡，我這個範例著陸後，車子會往原本x前進方向的反方向移動。

本程式分為3部分：

第1部分為斜坡運動如前面所說，並且本次的斜坡經過精心挑選，特別選擇在倒單擺平衡中，會超過斜坡範圍，然後掉下去的設計。

第2部分為拋體運動，x方向保持原速度等速前進，y方向進行自由落體，在程式中重力加速度g=0.0098(現實生活中為9.8，在程式中要一定比例的縮小，不然會一下掉到視窗下面)，temp存取離開斜坡時，每單位時間內，x方向的位移，也就是速度，在拋體運動時，x方向是等速度前進，也就是x\_pos=x\_pos+temp。

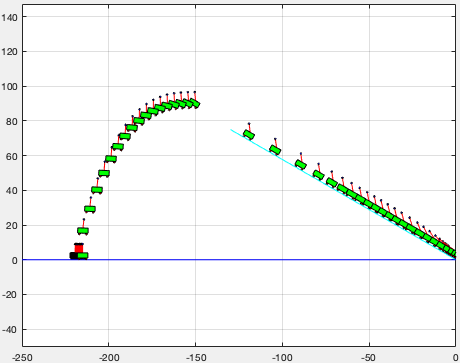
第3部分為著陸後的水平面倒單擺平衡運動，也就是老師最一開始給的範例程式的IP\_car.m，只是我在這利用著陸時的單擺角度對比平面倒單擺平衡時的角度，取一個最接近的時間區段來銜接，初始x位置代入著陸時的x位置。**(throw.m)**



以上程式，系統參數l(倒單擺線長度)=5(m)、g(重力加速度)=9.8()、選擇時間區間t=0~100(s)、

極點選擇=[-0.4 -0.5 -0.6 -0.7]

畫出來的圖如下：以自走車水平方向與地面夾角為30度為例**(longThrow.m)**



(六)結論：

1. 極點需要在複數平面的開放左半平面方能穩定，且極點遠離原點會使輸入的波形較快達到穩態(反應時間短)
2. 以8個狀態變數=[]為例，改動的初始值時，會導致產生劇烈變動。改動的初始值時，要改動幅度不大，基本上不引響穩定。增加的初始值時，發生變動的幅度需要看的幅度來決定，增加或增加的初始值時，會嚴重降低穩定性。

心得：這一次程式作業中，我認真複習了一遍被我拋棄在一邊好久的力學，竟然意外地覺得力學比以前有趣了，尤其是當自走車的行進路徑所產生的幾何美感。我個人在寫這些程式時，最大的問題仍然是在IP.m，因為講義上推出微分方程式後，就直接碰出二次微分的結果，中間省略了很多步，讓我推導的生不如死，最後我選擇了較為保險的狀態方程式，以保證結果收斂。

我個人覺得垂直圓周運動，垂直自走車和垂直地面一起比較的路徑圖非常有趣。感覺就像機械手臂一般，令人覺得有趣。

寫程式方面，我原本一開始是想直接改IP.m的全部函式來控制運動軌跡，但不知是什麼原因，我算出來的微分方程寫到IP.m時，原本可完全控制的方程式變成不可控，車子完全暴衝，或者程式直接發散，一直在IP.m內停不下來，經過幾次一直換微分方程的情況後，最後決定在原本老師的IP\_NL.m下，多加幾個方程式進行更多功能的控制。

順帶一提，我個人在進行數學推倒的過程中，經過每一步仔細了解之後，覺得老師給的IP.m在數學上其實省略了很多東西，我還能接受，畢竟把時間切得很小的情況下，每單位時間的角度變化理應也會很小，在平方的話，可以忽略不計，但，是要在的情況下方能近似，還你又怎麼確定，光範例的初始就為-1，條件就不成立了，而IP\_NL.m所以我個人比起IP.m，更偏向選擇IP\_NL.m，因此我個人狀態方程式的檔案雖然叫IP.m，但其實是用老師的IP\_NL.m來改寫的，而不是IP.m

不過最大的問題應該還是我那個老古董筆電由於記憶體還是硬碟容量不足，無法安裝matlab，讓我花了很多時間在找能跑matlab的電腦上面。