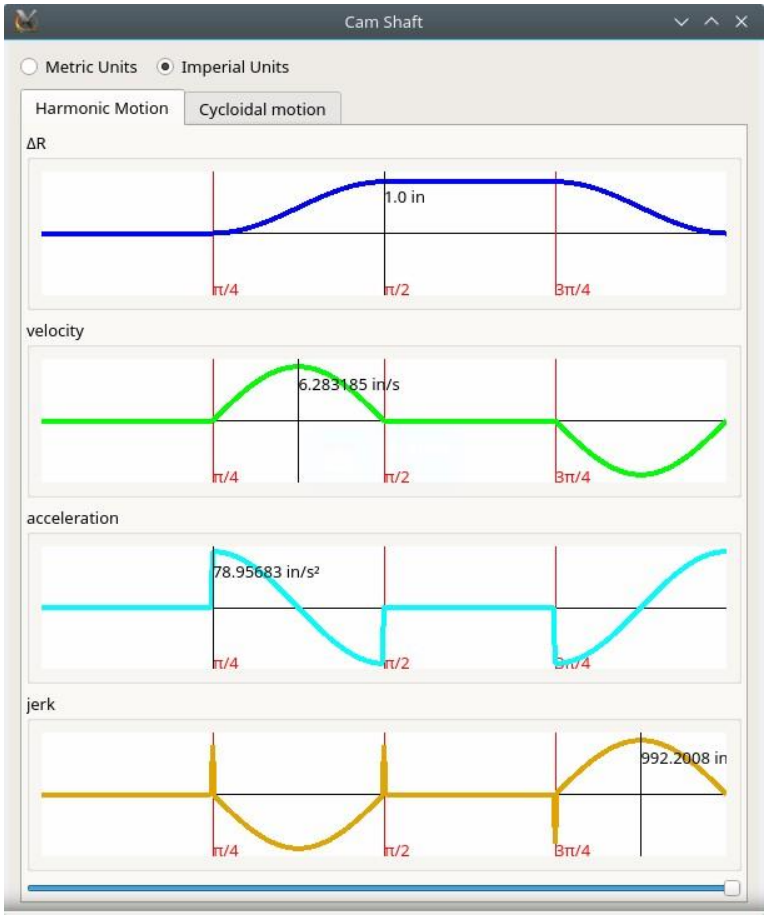
機構學報告

簡諧運動與擺線運動

班級：四設計三乙

學號：40323230

姓名：張元



程式倉儲：https://github.com/40323230/camCal

繪製工具：**Python 3.5、PyQt 5.7** QPainter

授權：AGPL-3.0或以上（繼承自 PyQt）

*附上 demo.avi 影片紀錄此程式的功能。

*由於編譯環境在 Linux 作業系統，無法在 Windows 平台運行。

*若要在 Windows 平台編譯，必須安裝相關工具（Python3、PyQt、PyInstaller），下載原始碼後製作，硬碟暫無空間所以不方便做出 Windows 平台版本。

功能：拉動滑桿可以計算當前最大值，並做公英制單位轉換。

使用單位：預設公制，可切換。

簡諧運動

ΔR 公式：

$$\Delta R = \frac{H}{2}(1 - \cos(2\varphi_i))$$

velocity 公式：

$$v = 2\pi H \sin(2\varphi_i)$$

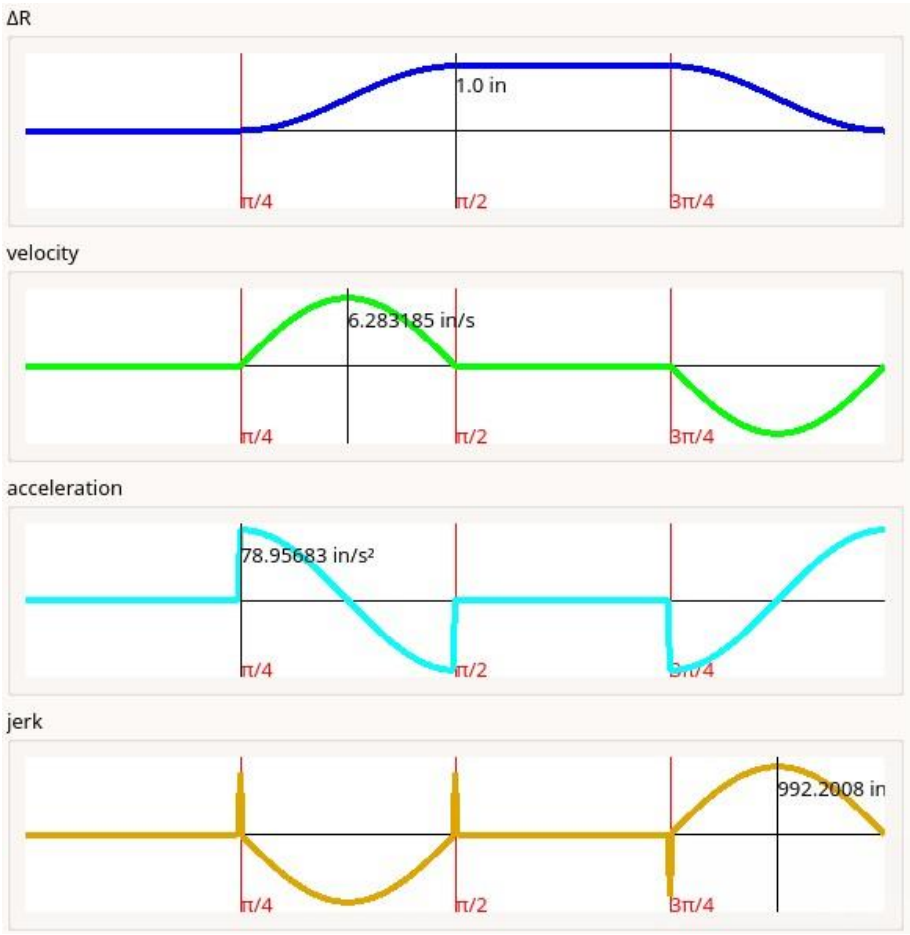
acceleration 公式：

$$a = 8\pi^2 \mathrm{H}\cos(2\varphi_i)$$

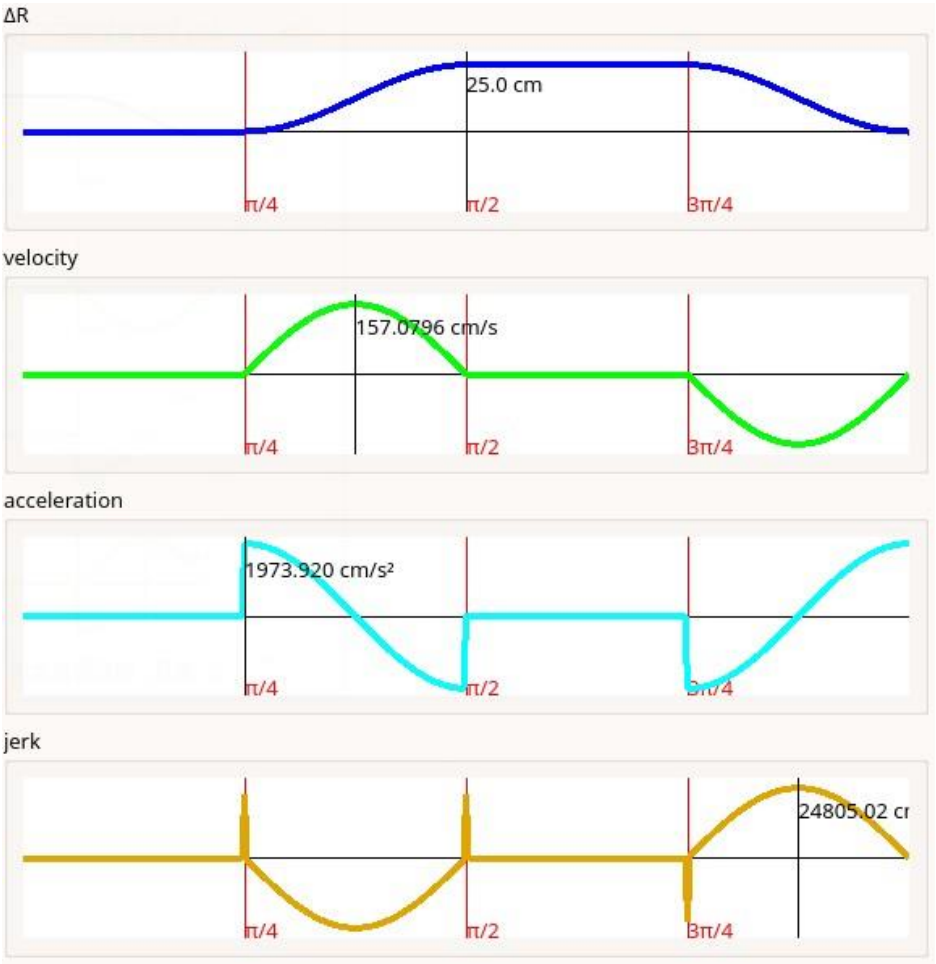
jerk 公式：

$$j = -32\pi^3 H sin(2\varphi_i)$$

單位：英制（in、in/s、in/s²、in/s³）



單位：公制（cm、cm/s、cm/s²、cm/s³）



擺線運動

ΔR 公式：

$$\Delta R = -\frac{H}{2\pi}\sin(4\varphi_i) + \frac{2H}{\pi}$$

velocity 公式：

$$v = 4\mathrm{H}(1 - \cos(4\varphi_i))$$

acceleration 公式：

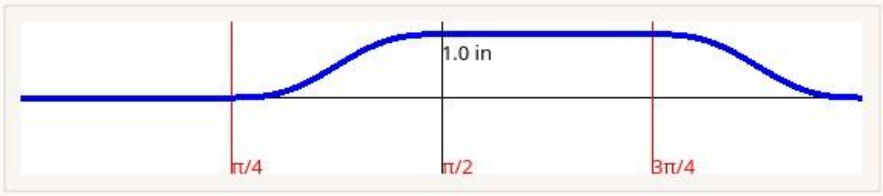
$$a = 8\pi^2 H sin(4\varphi_i)$$

jerk 公式：

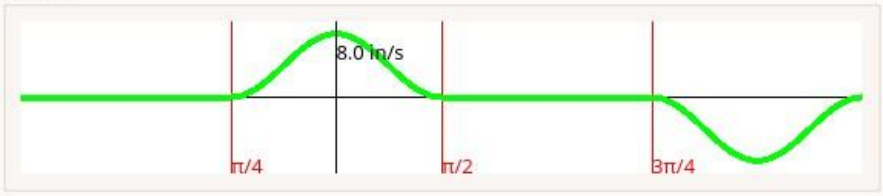
$$j = 256\pi^2 H cos(4\varphi_i)$$

單位：英制（in、in/s、in/s^2、in/s^3）

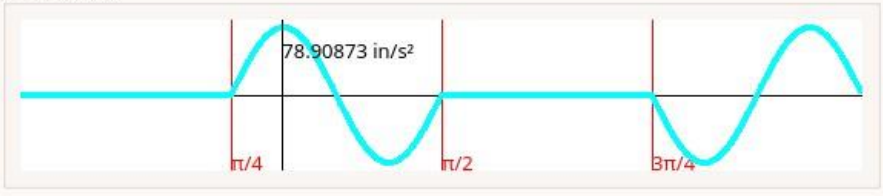
ΔR



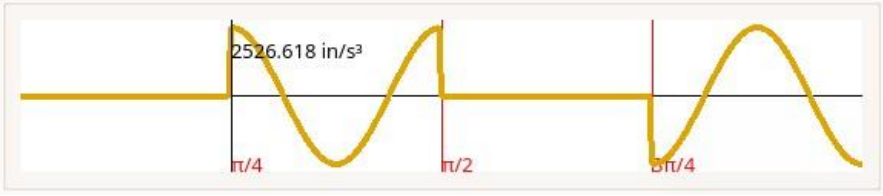
velocity



acceleration

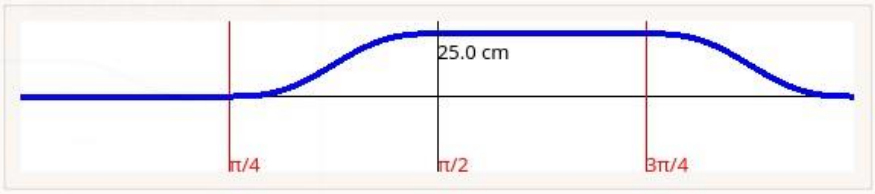


jerk

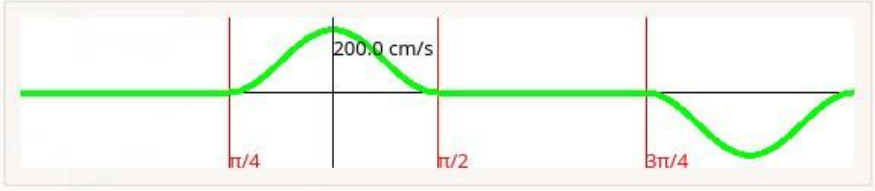


單位：公制（cm、cm/s、cm/s^2、cm/s^3）

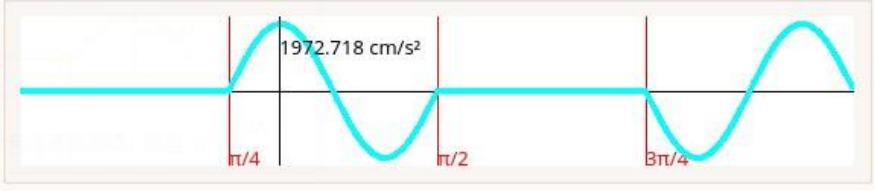
ΔR



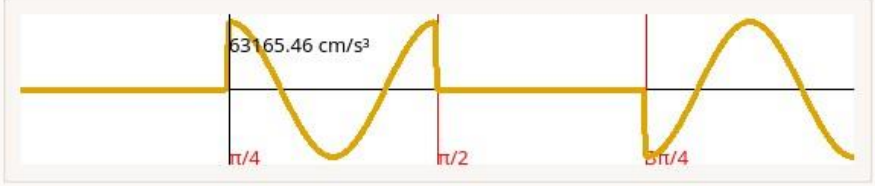
velocity



acceleration



jerk



程式碼簡述

簡單介紹程式碼中，公式運作的部份。
由於一部分的程式是用來產生圖形界面的，所以詳細內容不便繁述。

launch_cam.py

這個腳本專門呼叫整個程式運作，使用 Python 直譯器可以馬上啟動 core 資料夾的程序。

main.py

主程式，操控界面所有選項。繼承了8個圖形並插入界面中。

tabs.py

設立兩大 class（簡諧運動和擺線運動），各包含4個圖形畫布。這兩個 class 有附屬一個 function 用來切換公英制單位。

canvas.py

自創一個名為 chart 的 class，繼承自 QWidget class，相當一個空白區域，當為 QWidget 定義 paintEvent（繪製事件）後，就能使用 QPainter 在區域中著色。這個事件的作為是**動態**的，因此用內部或外部 function 更改變數時，paintEvent 可以畫出不同的圖形。
所有的圖表都是使用這個畫布畫出的，藉由更改算出的結果來改變每個畫布的內容；由於是視窗元件，x 軸朝右為正，y 軸朝下為正。

formula.py

公式存放區，return 結果回 tab.py，後者的兩大類別在初始化時會把結果填入 canvas.py 的畫布中，讓畫面啟動時顯示出各種不同的圖表。
總共有8個函式，一次傳回：一筆360個的 XY 點座標集、y 軸縮放倍率、一筆尋找最大值的數據集。以下是簡諧運動的速度函式：

```
def Har_velocity():
    answer = []
    for t in range(0, 90):
        answer += [{'x':t, 'y':0}]
    for t in range(90, 180):
        y = ((pi*H)/(2*T))*sin(pi*t/T)
        answer += [{'x':t, 'y':y}]
    for t in range(180, 270):
        answer += [{'x':t, 'y':0}]
    for t in range(270, 360):
        y = ((pi*H)/(2*T))*sin(pi*t/T)*(-1)
        answer += [{'x':t, 'y':y}]
    findMax = []
    for i in range(0, 360):
        findMax += [2*pi*sin(2*i/180*pi)]
    return answer, 1, findMax
```

*一些函式的 x 項中由於迴圈中不方便直接帶入對應角度給 y 項，公式有稍作修改。
*函式中，如 H 和 T 之類的固定值是用變數來計算，不過沒有做出使用者介面可調整的選項；若要調整，可以修改 formula.py 紀錄的區域變數。