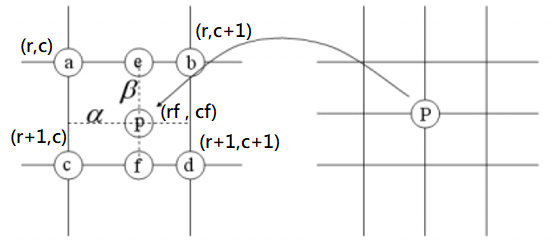
數位影像處理

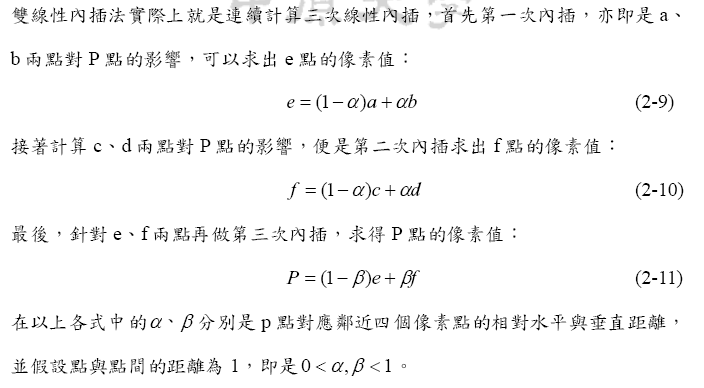
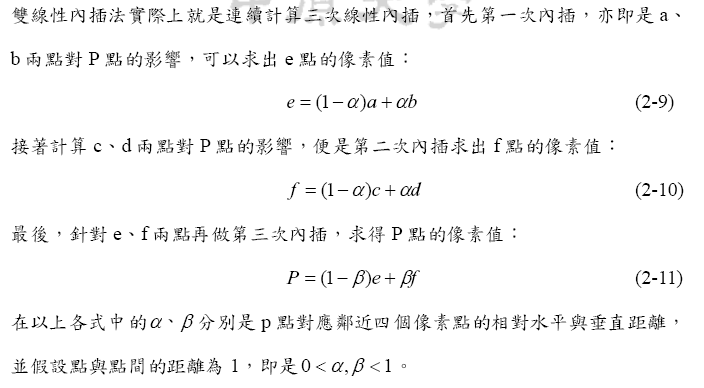
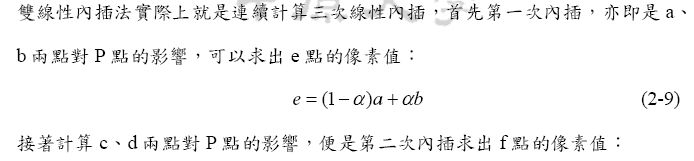
105323031蔣圳嵐

2019/03/17

# 原理:

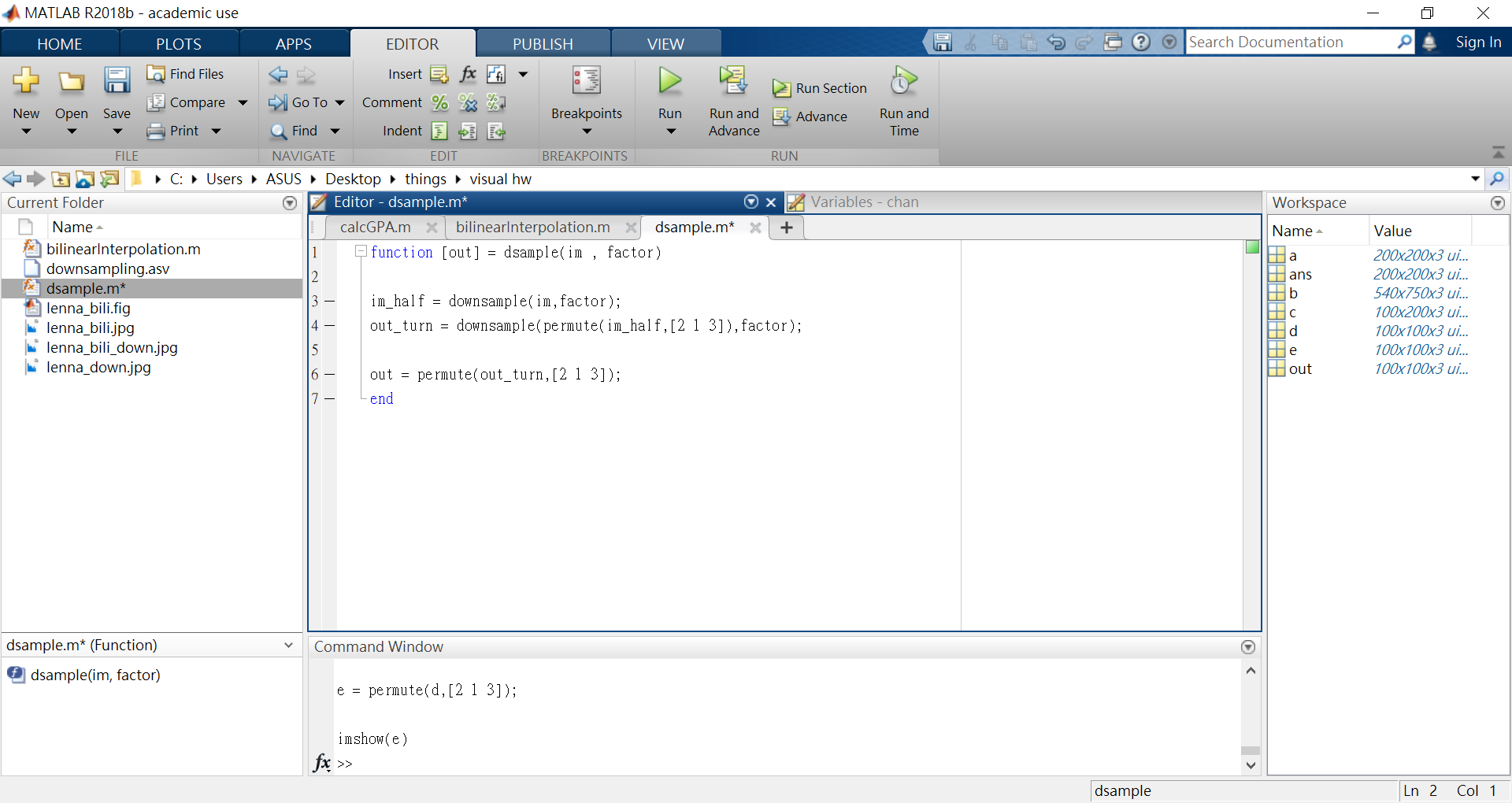
1. Downsample:使用Matlab內建的downsample函數實現。
2. Upsample:利用上課時老師所介紹的Bilinear interpolation之方法來實現。Bilinear interpolation即是假設要求得p點(rf,cf)的數值，並且令點a、b之間的距離為1，就要取得鄰近四點(a,b,c,d)的數值再乘上x軸之權重(α)得到式(2-9)和式(2-10)也就是e點和f點的數值，最後再利用e點和f點乘上y軸之權重(β)即可得到P點的數值。



▲原理示意圖

# 程式說明:

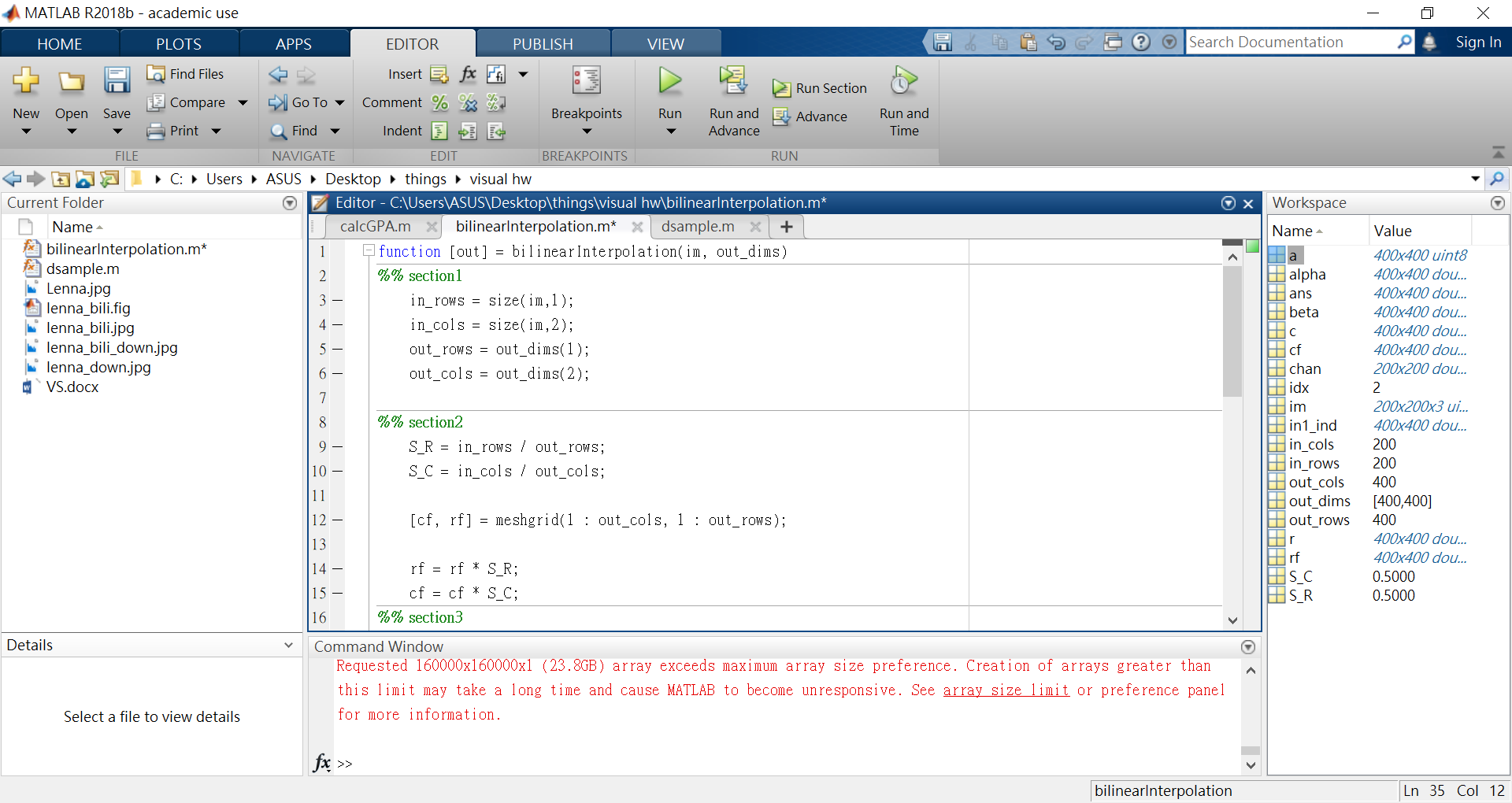
1. Downsample:



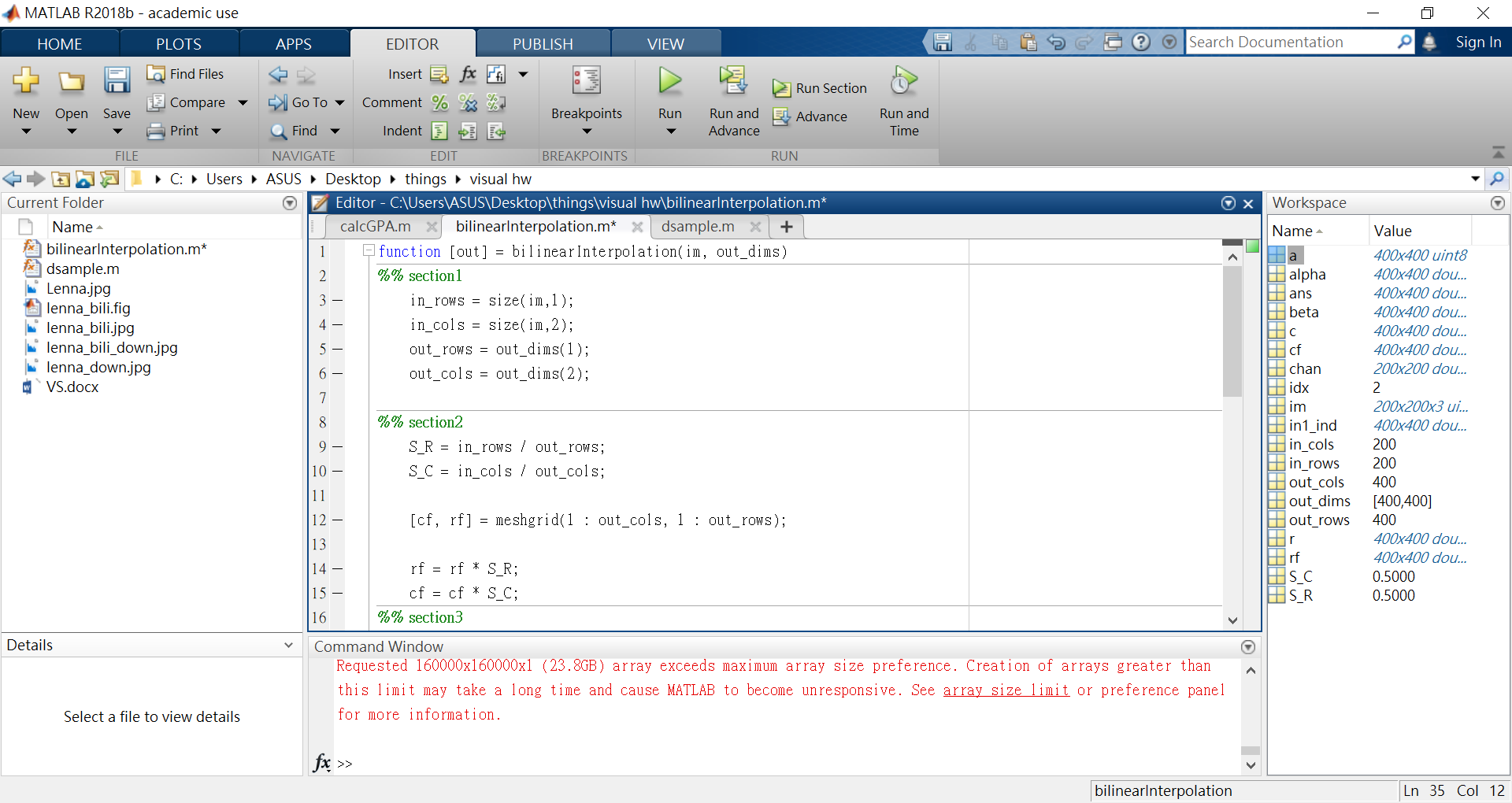
先將函式取名為dsample輸入為im降低倍數為factor輸出為out，注意取名不得與Matlab內建的函式名稱重複。當我使用downsample函式時發現程式只會處理columns的值，因此將im的columns做downsample取得新矩陣im\_half，之後把im\_half的row和column對調再進行一次downsample，最後要將row和column對調回來即可完成downsample。

補充：因為im為一個三維的矩陣因此無法使用’(也就是Matlab內建的算術運算transpose)來對調column和row，所以要使用permute函式來解決permute(欲對調矩陣 , [依次擺放維度的順序，1為row依此類推])。

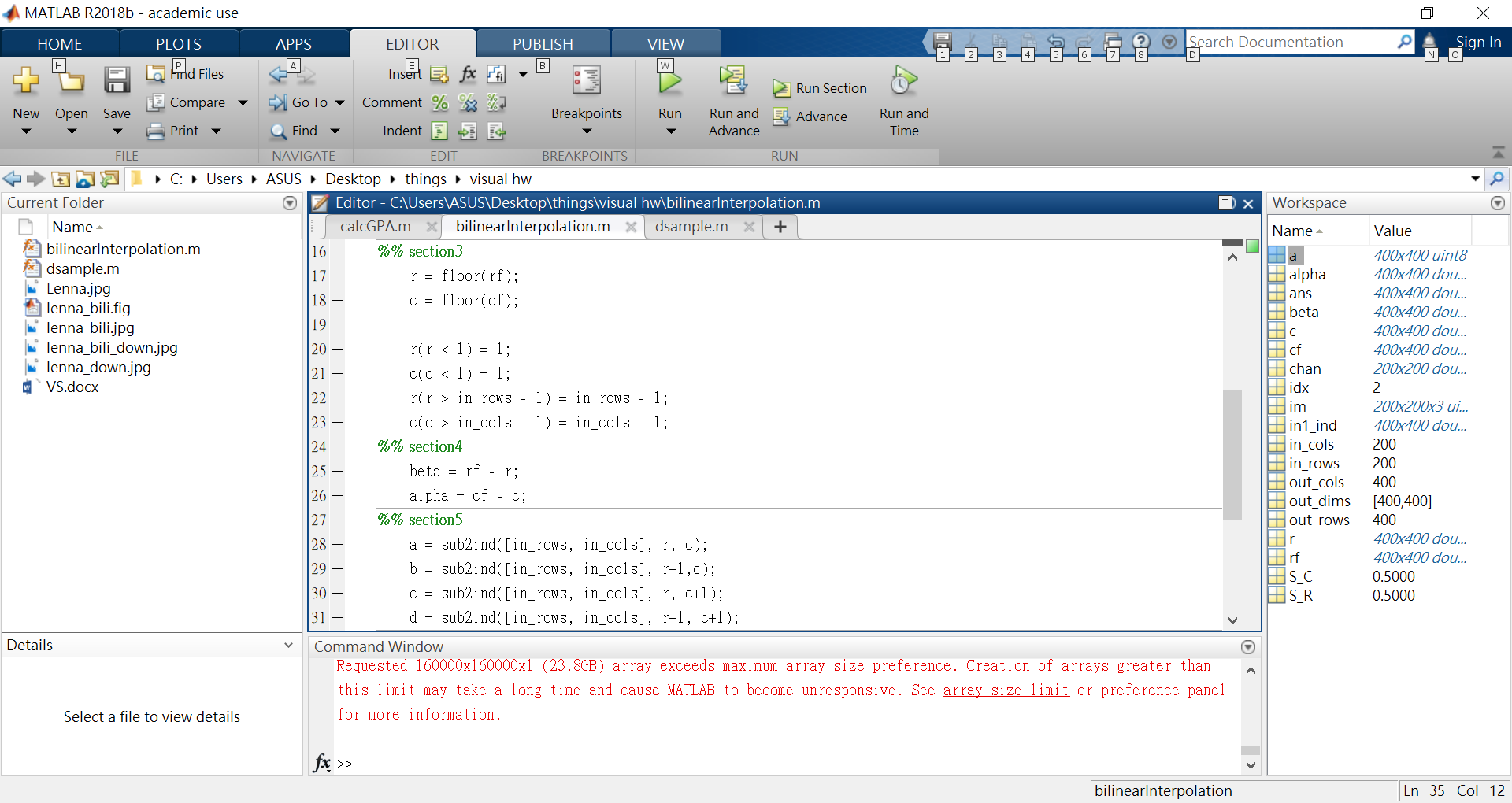
1. Upsample:



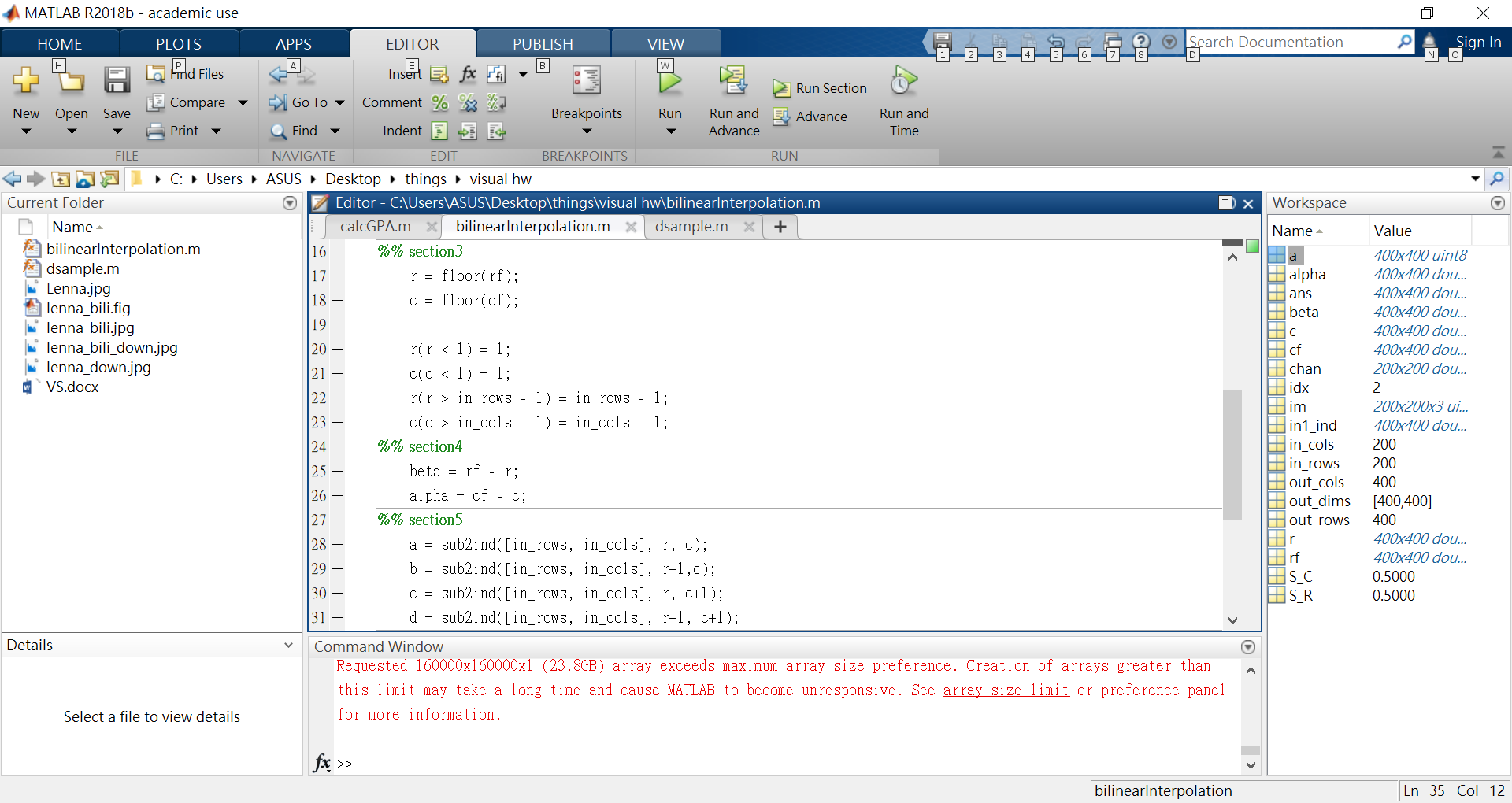
首先先設定函式bilinearInterpolation的輸入圖片為im欲輸出像素out\_dims輸出為out。之後在section1裡面設定一些基本數值，輸入和輸出個別column和row的數量。



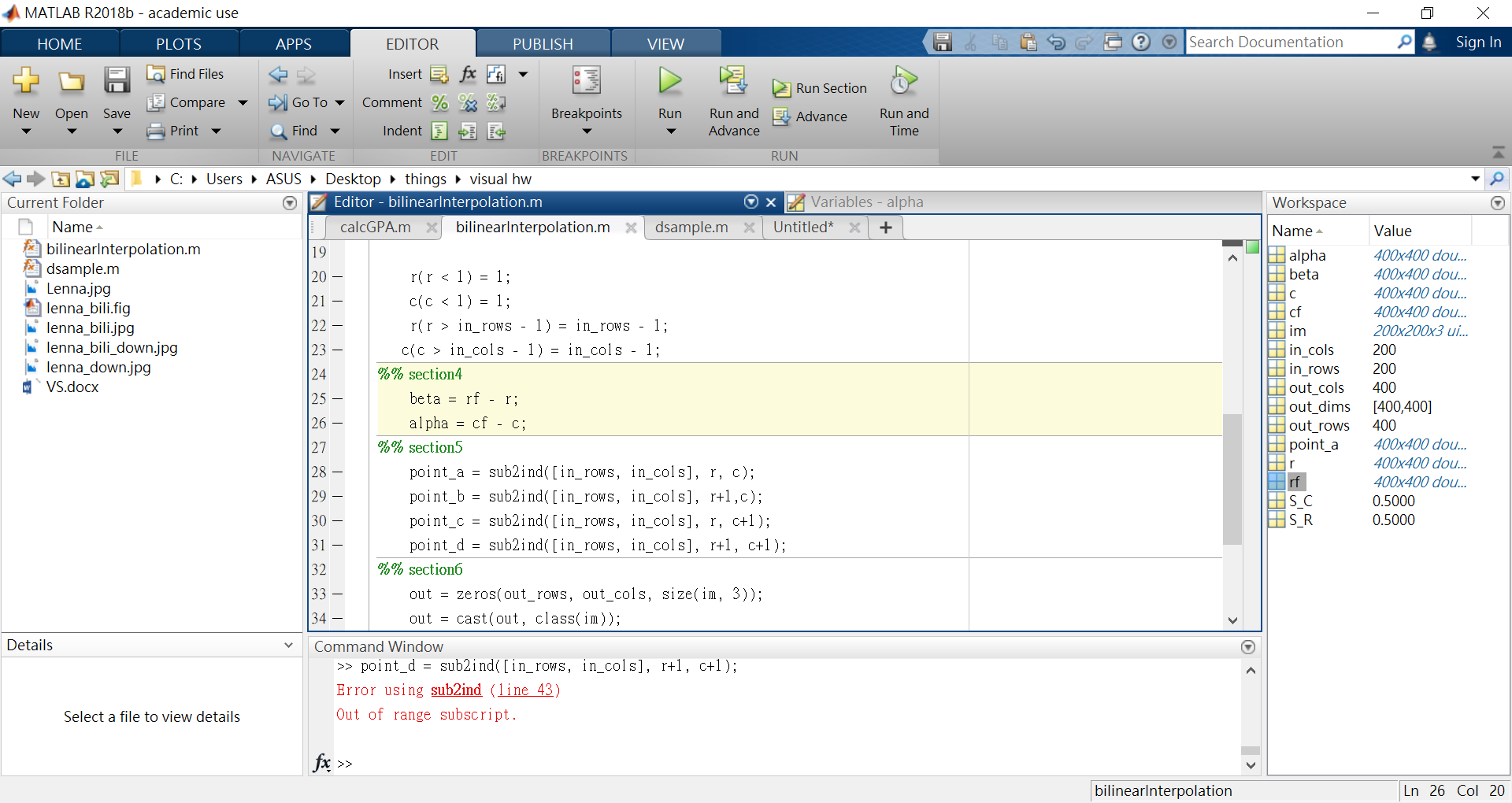
在section2中要計算出rf和cf的值，因此先利用輸入和輸出各自rows和columns的比值去計算出S\_R和S\_C，之後創造一個矩陣rf代表row的值從1到輸出要求row的像素值，最後把rf和S\_R相乘即可得到我們真正要的rf。而cf也是同樣道理，即可求得真正的cf。



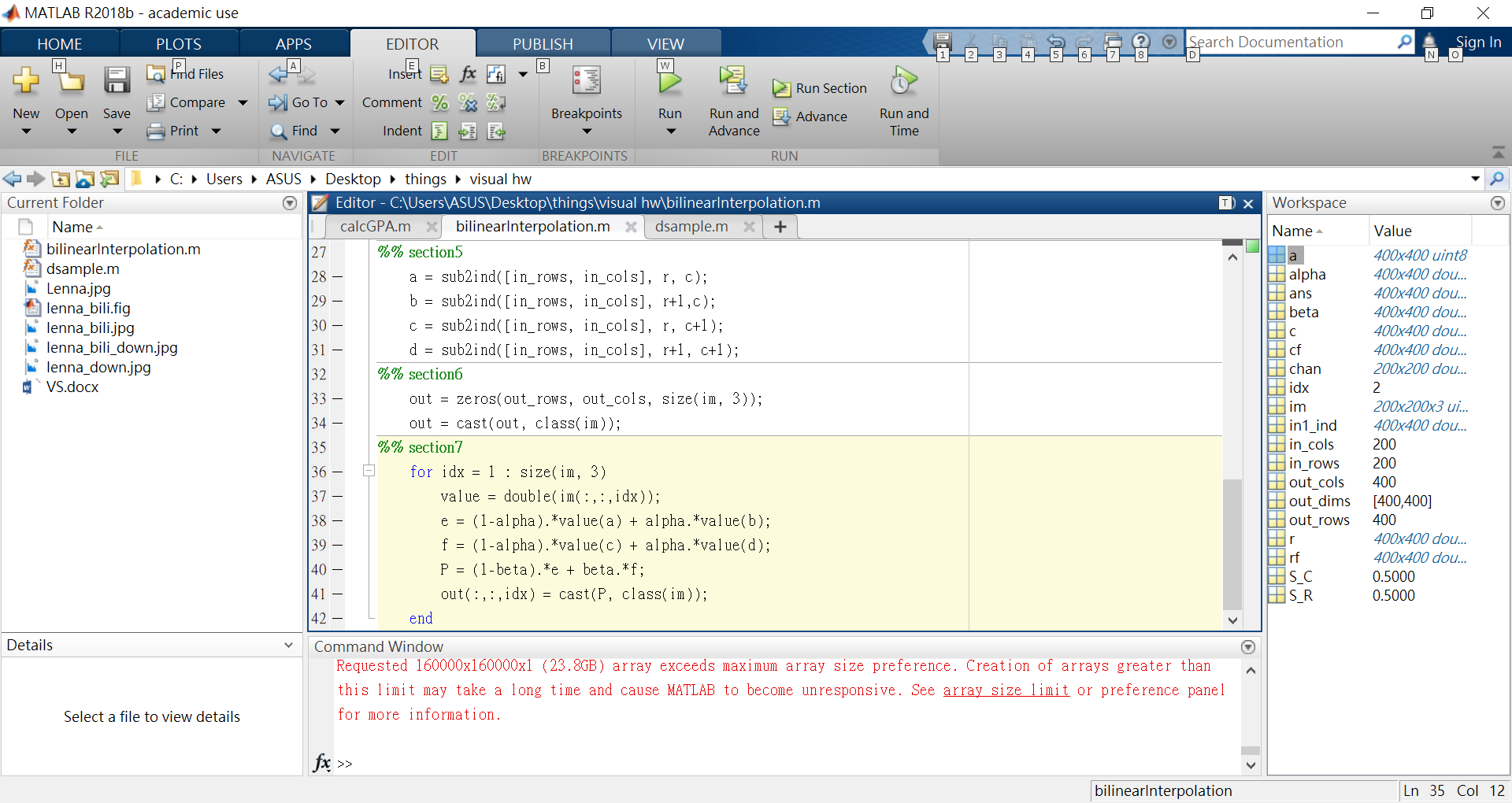
在section3裡面，首先先取得rf鄰近的整數值取名為r。為了之後取值的時候不要出現錯誤因此下面加了兩個判斷式分別式當r小於1的時候值為1和r大於輸入row像素-1值為輸入row像素-1。c也是使用同樣道理。



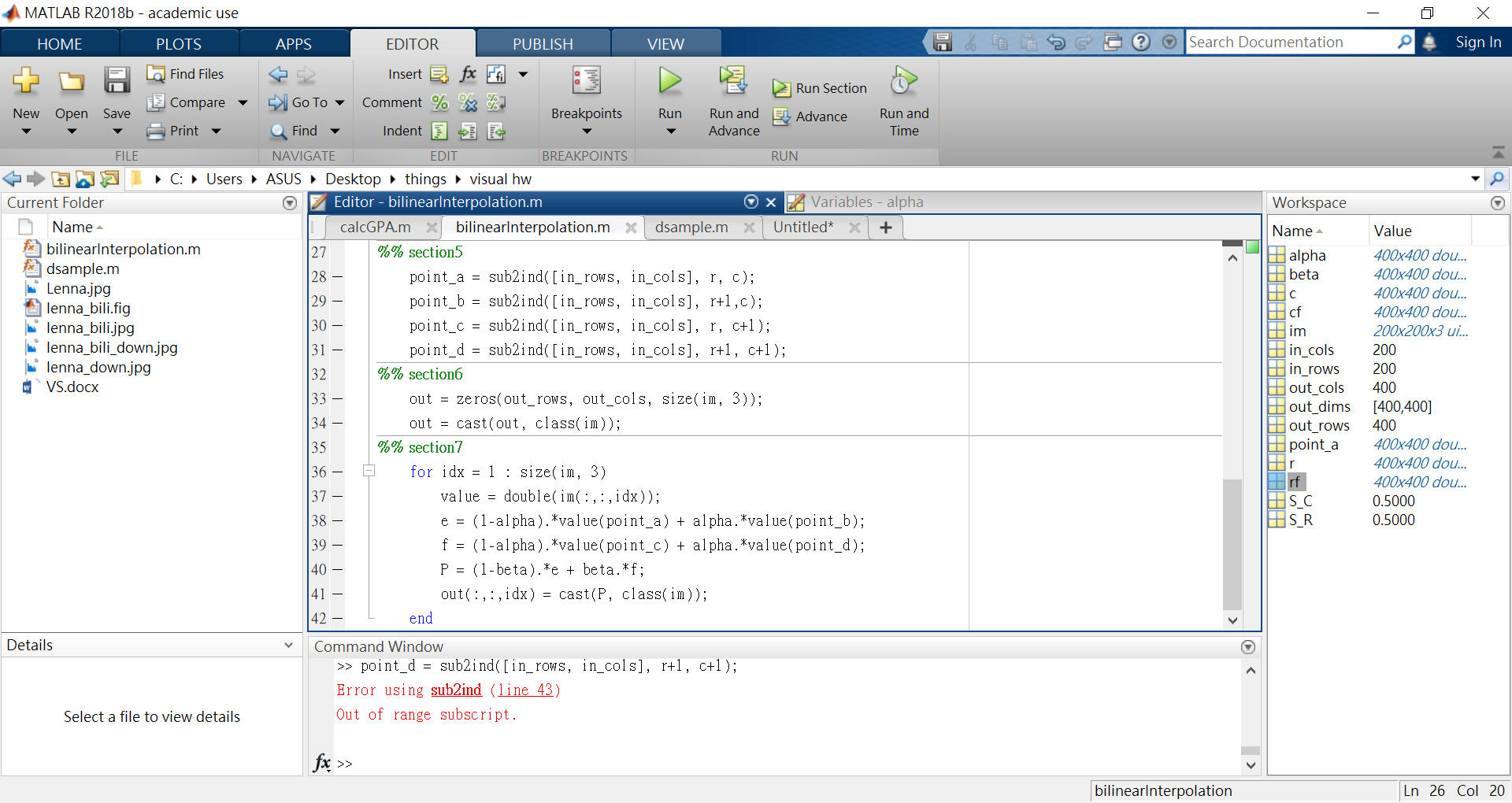
計算出α和β值。



在section5是取得點a,b,c,d在矩陣內的儲存位置。



section6則是設定輸出矩陣out的各項設定，例如：大小、class等等。



最後section7就是使用一個for迴圈，分次取得RGB的值。迴圈跑第一次時事取得R層的值之後丟到value裡面，再帶入從section5中取得的儲存位置即可求得各點R層的數值，再利用式(2-9)、式(2-10)、式(2-11)最後求出P，再將P的值給予矩陣out的R層。之後再跑G、B層就可以得到輸出圖片out了。

# 結論:

這次的程式讓我重視到了class的重要性，因為當兩個不同class的數值是無法做運算的，因此在寫程式的時候要注意。我也藉著這個實驗更了解圖片是如何在電腦內儲存的。然後也觀察到只要經過影像處理之後就會有圖片變模糊的情況發生，因此選擇一個好的處理方法相當重要。

所以當我發現Bilinear interpolation也可以去實現downsample時，我把同樣的一張圖片分別使用兩種方法都downsample 2倍，左圖為Bilinear interpolation，右圖為Matlab內建的downsample。觀察發現右圖帽子頂端的地方都快和背景融為一體，但是左圖的還是可以清晰判斷分界點。鼻子的地方左圖也較沒有顆粒感，因此左圖和右圖相比左圖使用的方法是一個較好的方法。



▲原圖



▲左圖為Bilinear interpolation，右圖為Matlab內建的downsample

再從資料大小上面觀察也能發現左圖的大小比右圖大，這樣也可以驗證我所觀察到的，因此結論是：左圖使用的方法Bilinear interpolation是一個比右圖使用的方法Matlab內建的downsample函式效能還要好。



▲左圖資料大小為5800位元組



▲右圖資料大小5752位元組