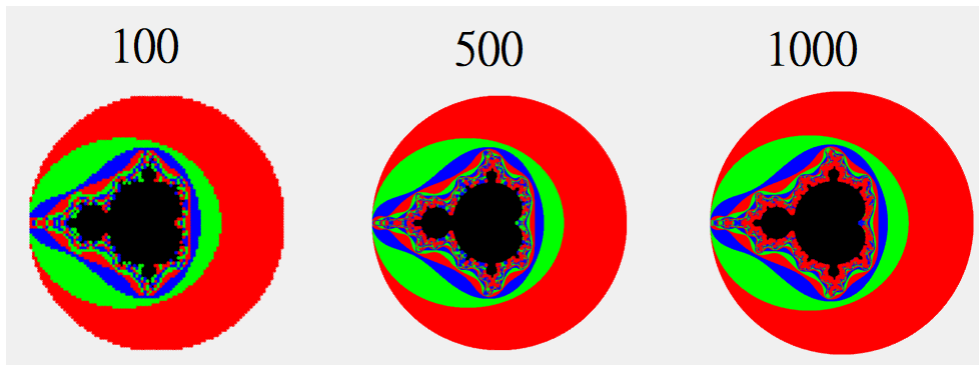


- Assignment 要求內容

1. 比較 100、500、1000 個等分點的繪圖差異。
  - a. 細節部分，1000 的圖像邊緣較圓滑，100 可以看出是一格一格的。



- b. 不能放到很大，100 會先像素化，再來是 500、1000。  
請觀看（附件一），code 為 RGB\_compare.m。
2. 步驟重覆 20 次，利用 Matlab 製作動畫影片。  
請觀看（附件二），code 為 RGB\_20.m。
  3. 影像放大裁切平移 + 影像著色 + 音樂配置。

請至 [https://youtu.be/tQVHO\\_8k8w0](https://youtu.be/tQVHO_8k8w0) 觀看，code 為 Final.m。

- 方法描述 - 演算法原理與實作方式描述

Mandelbrot 實作有兩種方法：

第一種，較差的方法、

1. 在平面上面取  $N$  個等分點，有  $N*N$  個平均分布的點。
2. 每運算一次就把落到圓外的所有點記錄起來，並 plot 出所有記錄到的點。
3. 將剩下的點繼續做運算，再把落到圓外的所有點記錄起來，plot 出來。
4. 依此延續變可以得出 Mandelbrot set。

第二種，較佳的方法

1. 在平面上面取  $N$  個等分點，有  $N*N$  個平均分布的點。
2. 開一個  $N*N$  陣列，用以紀錄每個點在圓內的次數。
3. 每運算一次便查看每個點是否在圓內，在圓內則把陣列中那點的值加一。
4. 最後可得出每點在圓內的次數，依照次數 plot 出來。

註：有一個函式 `imagesec()` 是專門繪出這種資料，因此我在 Final.mp4 中是用這個函式繪圖。

將放大倍率放大至很高的方法：

程式分階段執行，每個階段一開始先延續上一個階段的邊界，並在邊界內取樣、運算，再把畫面放大一定倍率，交給下一個階段，如（附件三）。

以此模式延續執行，便可以快速的產生出連續放大的圖形。

- 執行方式 - 執行的函數名稱、參數設定等

參數設定：

等分點個數	1000
Iteration times	800
放大參數	0.994
放大焦點 x	-0.72922175749625
放大焦點 y	-0.22322837539578
影片幀數	80*60 = 4800
影格速率	24

執行的函數名稱：

數學函式、

linspace()                      取等分點

meshgrid()                    擴展座標

Figure 函式、

fig.Position = [x y width height]              設定 figure 的位置大小

title()                          設定 figure 的標題

繪圖函式、

imagesc(x, y, value)                      依照每點的 value 繪畫出顏色

colormap([bone();0 0 0])                    可設定 imagesc 繪畫的顏色表

- 實驗結果 - 每一個階段的圖片、數據結果

請觀看 Final.mp4，code 為 Final.m。

或至 [https://youtu.be/tQVHO\\_8k8w0](https://youtu.be/tQVHO_8k8w0) 觀看。

- 結果討論 - 對於實驗結果的一些解釋和討論

$$f(x_{n+1}) = x_n^2 + c$$

Mandelbrot 看起來雖然極為複雜，但它是由一個非常簡短的公式所產生的，將不會發散的區域保留下來就是所謂的 Mandelbrot Set。

● 問題討論 - 作業撰寫中遭遇的演算法問題與實作的困難

問題 1: 圖形放大很多次以後，就可以很容易的看出畫面是由點構成的，如（附件四）。

解決方法: 經過測試以後，我發現當畫面中一直行或一橫列的點，總和少於約 150 個的時候，就會發生這個情況。

因此可以計算出： $\frac{\text{linspace 的取樣點數}}{150} = \text{最高可放大的總倍率}$

只要放大倍率不要超過計算出來的值就可以免除這個問題。

問題 2: 影片的每一幀都會放大一些，如何決定每一幀的放大倍率或決定總共幀數才不會超過最高可放大的總倍率？

解決方法: 假設: 影片要做的幀數 = frames  
每一幀的放大倍率 = r  
linspace 的取樣點數 = n

1. 求影片要做的幀數 frames:


$$r^{\text{frames}} \leq \frac{n}{150} \rightarrow \log r^{\text{frames}} \leq \log \frac{n}{150} \\ \rightarrow \text{frames} \leq \frac{\log \frac{n}{150}}{\log r} = \log_r \frac{n}{150}$$

2. 求每一幀的放大倍率 r:

$$r^{\text{frames}} \leq \frac{n}{150} \rightarrow r \leq \sqrt[\text{frames}]{\frac{n}{150}}$$

問題 3: 程式跑得太慢。

解決方法: 發現原因是出在將兩個值從一個陣列賦值到另一個陣列。  
使執行時間差了將近 5 倍。



```
for j = 1 : dotAmount
    if dots(1,j)^2 + dots(2,j)^2 > 4
        nextdotAmount = nextdotAmount - 1;
        out(:,counter(2)) = dots(:,j);
        counter(2) = counter(2) + 1;
    else
        in(:,counter(1)) = dots(:,j);
        counter(1) = counter(1) + 1;
    end
end
Elapsed time is 0.711329 seconds.
```

```
for j = 1 : dotAmount
    if dots(1,j)^2 + dots(2,j)^2 > 4
        nextdotAmount = nextdotAmount - 1;
        out(1,counter(2)) = dots(1,j);
        out(2,counter(2)) = dots(2,j);
        counter(2) = counter(2) + 1;
    else
        in(1,counter(1)) = dots(1,j);
        in(2,counter(1)) = dots(2,j);
        counter(1) = counter(1) + 1;
    end
end
Elapsed time is 0.152800 seconds.
```

問題 4: getframe 產生影片，由於長寬是利用 axis([xmin xmax ymin ymax])，每次畫面的大小有些微差距，使影片不能寫入。

解決方法: 使用 imresize，設定 cdata。

```
F = getframe;
F.cdata = imresize(F.cdata, [1000 1000]); % 設置 行列
```

問題 5: 延續上題，每次畫面的大小有些微差距，使影片的周圍有時會出現細細的白色線條，如（[附件四](#)）。

解決方法: `getframe` 時直接指定擷取 `gcf` 和擷取的畫面大小，同時解決了白色線條和上一個問題。

```
F = getframe(fig,[0 0 1538 810]);
```

問題 6: 延續上題，使用上一題的 `getframe(fig,[x,y,width,height])`時，已經設定好 `figure` 大小和擷取大小，卻出現錯誤訊息

```
fig.Position = [0 0 1600 810];  
F = getframe(fig,[0 0 1600 810]);
```

Error using `getframe` (line 113)

The specified rectangle is not fully contained within the figure. MATLAB no longer supports this capability.

解決方法: 經過數次的失敗後，發現原因出在 `figure` 大小跟原先設定的不同，可能是觸碰到螢幕的邊界了，所以只要不要超過限制的大小就行了。

```
>> fig
```

```
fig =
```

[Figure](#) (1) with properties:

```
Number: 1  
Name: ''  
Color: [0.1000 0.1000 0.1000]  
Position: [0 0 1538 810]  
Units: 'pixels'
```

Show [all properties](#)