迭代圖形及其動畫

林煜傑　李香儀

January 22, 2022

1 甚麼是 Julia Set?

Introduce

Julia set 是一個在複數平面上形成碎形的點的集合。以法國數學家加斯頓·朱利亞 (Gaston Julia) 的名字命名。

Definition

朱利亞集合可以由下式進行反覆迭代得到：

$$f_c(z) = z^2 + c$$

對於固定的複數 c，取某一 z 值（如 $z = z_0$），可以得到序列

$$z_0, f_c(z_0), f_c(f_c(z_0)), f_c(f_c(f_c(z_0))), \ldots.$$

這一序列可能發散於無窮大或始終處於某一範圍之內並收斂於某一值。我們將使其不擴散的 z 值的集合稱為朱利亞集合。

2 Julia Set 圖形

根據設定不同的 c 值，所對應的 Julia Set 圖形也會不同，這裡我們設定 c 值為 $-0.701176 + -0.38421i$，後續可以進行調整。但是在選擇 c 值時，需要特別注意的是：$|c| < 2$，否則一定會發散。

Method 1

首先，創造出大量 x,y 都從-2 到 2 的格子點，接下來將這些點依次代入 $f_c(z)$，代入後數列收斂，則將其收集於一集合中；否則不收集。最後，用收集了所有符合條件的點的集合畫出圖形。

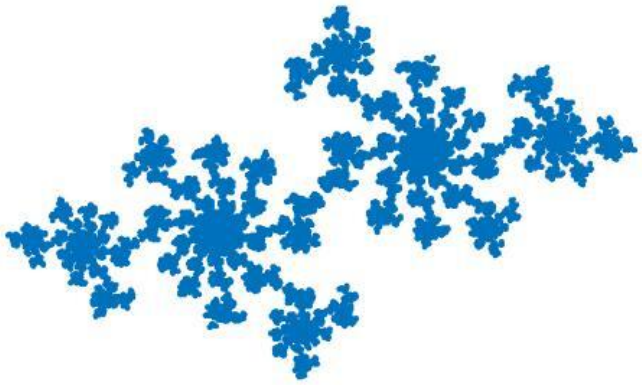
用 Matlab 執行的程式碼如下。

Code

```
function julia(c)
for i=1:2000   %產生格子點
    for j=1:2000
        A(i,j)=complex(-5+i*0.005,-5+j*0.005);
    end
end
B=reshape(A,[1,4000000]);
C=[];
for i=1:4000000   %迭代並收集
    k=B(i)^2+c;
        for j=1:30
            k=k^2+c;
        end
        if abs(k)<=2
            C=[C,B(i)];
        else

        end
end
X=[];
Y=[];
for k=1:length (C)
    X=[X,real(C(k))];
    Y=[Y,imag(C(k))];
end
s=scatter(X,Y,10,'filled');
axis equal ; axis off ;
```

輸入 $julia(-0.70176-0.3842*1i)$ 後，output 為下圖



但是這時，相對於網上的圖，這張圖還有些不足：這裡沒有根據每一數列何時開始發散畫出不同顏色，而這種方法如果要上色，計算量會相當大，如後續。所以我們需要找到一個新的方法來作圖，而這個方法就是矩陣。

Method 2

首先，創造出 $n*n$ 的兩個零矩陣，再將第一個矩陣第 1 列、和第二個矩陣第 1 行的點，分別替換為-2 到 2 的格子點。

接下來，用這兩個矩陣一個作為實部，另一個作為虛部，生成出另一矩陣，用來後續迭代 $f_c(z) = z^2 + c$。

最後，在迭代時，每迭代一次就判斷一次是否收斂，當不收斂時，將當時跑的迴圈數紀錄在新的收集結果的矩陣中對應的位置，再根據收集結果的矩陣進行顏色設定和畫圖。

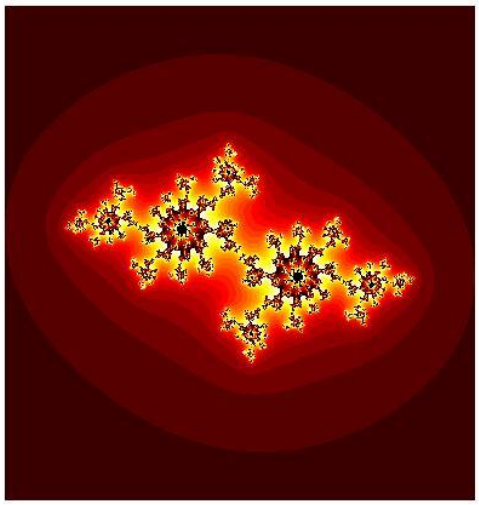
用 Matlab 執行的程式碼如下。

Code

```
function juliaSet(r,w,inc,repeatColors)
    temp=zeros(w,w,2);   %生成初始矩陣
    for i=1:w
        temp(i,:,1)=linspace(-2,2,w);
```

```
        temp(:,i,2)=linspace(-2,2,w);
    end
    values=zeros(w,w);  %用初始矩陣製造出用於迭代的矩陣
    for i=1:w
        for j=1:w
            values(i,j)=temp(i,j,1)+1i*temp(i,j,2);
        end
    end
    output=zeros(w,w);  %生成用於收集結果的矩陣
   for i=1:r    %進行迭代
        for m=1:w
            for k=1:w
                values(m,k)=values(m,k)^2+inc;
                if abs(values(m,k))>10&&output(m,k)==0
                    output(m,k)=i;
                end
            end
        end
    end
    if repeatColors==1
      for m=1:w
          for k=1:w
              output(m,k)=mod(output(m,k),r/2);  %調整顏色，使其更鮮豔
          end
      end
    end
   figure('Position',[200,200,600,600],'Visible','off');
%圖片大小600*600;圖片距離電腦各為200
    imagesc(result);
    colormap('hot');  %用內建的'hot'設定顏色
    axis off
end
```

輸入 $juliaSet(50, 2000, -0.70176 - 0.3842 * 1i, 1)$ 後，output 為下圖



3 動畫

在 Matlab 中將圖片轉為動畫，可以使用指令 pause，將 c 值不斷轉換後的圖產生出對應動畫。我們可以用這種方法對兩種方法做出的圖形分別做出動畫。

Code

```
for k=1:20
    c=-0.70176-0.2*1i+0.01*k*1i
    julia(c) % or juliaSet(50,2000,c,1)
    hold on ;
    pause(0.5)
end
```

但是這時會出現一個問題：pause 的作法是圖片每跑出一張就顯示一張，但是 Julia Set 的圖片需要反覆迭代每一個點，與之對應的是超大的計算量，所以每一張圖都需要跑一些時間，最後輸出的動畫會非常不流暢。(放出來要等到下課才能放完)

為了解決這個問題，我們選擇將圖片先一張一張保存好，再由電腦一次跑出來，這樣就可以得到流暢的動畫了！(動畫在附件中)

Code

```
for p=1:200   %將動畫所需的圖片全部存起來。
```

```
    %中間加入前面第二種方法全部程式
    pic(p)=getframe(gcf);
    C=C+0.005*1i;
    close all;
dt=0.1;  %製成動畫
for i=1:200
    [image,map]=frame2im(pic(i));
    [im,map2]=rgb2ind(image,128);
    if i==1
        imwrite(im,map2,'move_pic.gif','gif',
'writeMode','overwrite','delaytime',dt,'loopcount',inf);
    else
        imwrite(im,map2,'move_pic.gif','gif',
'writeMode','append','delaytime',dt);
    end
end
end
```

4 Mandelbrot set 與其動畫

Mandelbrot set 介紹

與 Julia set 類似，Mandelbrot set，是一種在複數平面上組成碎形的點的集合，以數學家本華·曼德博的名字命名。曼德博集合與朱利亞集合有些相似的地方，例如使用相同的復二次多項式來進行迭代。

曼德博集合可以用復二次多項式來定義：$f_c(z) = z^2 + c$ 其中 c 是一個複數參數。從 $z = 0$ 開始對 $f_c(z)$ 進行迭代：$z_{n+1} = z_n^2 + c$,$n = 0, 1, 2, ...$,$z_0 = 0$, $z_1 = z_0^2 + c = c$,$z_2 = z_1^2 + c = c^2 + c$ 每次迭代的值依序如以下序列所示：$(0, f_c(0), f_c(f_c(0)), f_c(f_c(f_c(0))), \ldots)$ 不同的參數 c 可能使序列的絕對值逐漸發散到無限大，也可能收斂在有限的區域內。曼德博集合 M 就是使序列不延伸至無限大的所有複數 c 的集合。

Mandelbrot set 的動畫

做法與 Julia Set 類似，同樣使用迭代的方式，程式碼如下：(動畫在附件中)

Code

```
function madelmove(x,y,s)
%x為圖形細節程度;y為圖像解析度程度;C為複數函數;
%s為是否要顏色對比性增強
for o=1:20
```

```
    A=zeros(y,y,2);%建立2個y*y的0矩陣
    for i=1:y
        A(i,:,1)=linspace(-2,2,y);%填入1號矩陣每個row的數據
        A(:,i,2)=linspace(-2,2,y);%填入2號矩陣每個colume的數據
    end
    initial=zeros(y,y);%initial為一個y*y的0矩陣
    for i=1:y
        for j=1:y
            initial(i,j)=A(i,j,1)+A(i,j,2)*1i;
        end
    end
    result=zeros(y,y);%result為一個y*y的0矩陣
    for j=1:y
        for k=1:y
            p=-2;
            for i=1:x
               p=p^2+initial(j,k);
                if abs(initial(j,k))>2 && result(j,k)==0%判斷是否收斂
                    result(j,k)=i;
%當不收斂時，計算迴圈跑的次數以區分顏色
                end
            end
        end
    end
    if s==1
      for i=1:y
          for j=1:y
             result(i,j)=mod(result(i,j),x/3);
%讓區域顏色對比性更強
          end
      end
    end
    figure('Position',[200,200,600,600],'Visible','off');
%圖片大小600*600;圖片距離電腦各為200
    imagesc(result);
    colormap('hot');
    axis off
    pic(o)=getframe(gcf);
    p=p+0.2;
    close all;
end
dt=0.1;
for i=1:20
    [image,map]=frame2im(pic(i));
    [im,map2]=rgb2ind(image,128);
```

```
    if i==1
        imwrite(im,map2,'move_pic.gif','gif','writeMode',
'overwrite','delaytime',dt,'loopcount',inf);
    else
        imwrite(im,map2,'move_pic.gif','gif','writeMode',
'append','delaytime',dt);
    end
end
end
```

Mandelbort set 的 3D 圖形

另外，我們也在網上找到了 Mandelbort set 的 3D 圖形的呈現方法，並且嘗試做了一些修飾。

```
function mandelbulb
[x,y,z]=ndgrid(linspace(-1.2,1.2,128),
linspace(-2,2,256), linspace(-2,2,256));
orgx=x ;orgy=y ;orgz=z;
val=0.*x;
for i=1:50
    x1=x.^2-y.^2-z.^2+orgx;
    y1=2.*x.*z+orgy;
    z1=2.*x.*y+orgz;
    x = x1; y = y1; z = z1;
    r=sqrt(x.^2+y.^2+z.^2);
    val=val+(r < 2.0);
%r小於2.0時才會加上去
end
val=smooth3(val);
%讓圖片更滑順
p = patch(isosurface(val));
%從三維體數據中提取等值面數據並繪製多邊形
set(gcf, 'color', "white");
%將圖片背景設為白色
set(p, 'FaceColor', [139/256 69/256 19/256], 'EdgeColor', 'none');
%設置表面顏色
axis on     %顯示軸
axis vis3d
%凍結螢幕高寬比，使得一個三維物件的旋轉不會改變座標軸的刻度顯示
view(70,30)
%視角為方位角70;與平面與平面交叉角30度
camlight(0, 0)
%建立一個光源
camzoom(0.8)
```

%缩小0.8倍
light
%打光
rotate3d on
end