1 Background

The Mandelbrot set is formally defined as the set of all complex numbers c for which the sequence $z_{n+1} = z_n^2 + c$ does not diverge. This is equivalent to find all points (x, y) for which the sequence

$$\begin{aligned} x_{n+1} &= x_n^2 - y_n^2 + c_x \\ y_{n+1} &= 2x_n y_n + c_y, \end{aligned} \tag{1}$$

remains bound. In order to visualize this set we will follow the next steps

2 Problems

(P1) Create two arrays of length $N = 64$ for c_x and c_y

```
N = 64
Cx = np.linspace(-2.5, 1.0, num = N, endpoint = True)
Cy = np.linspace(-1.0, 1.0, num = N, endpoint = True)
```

(P2) Create a square grid of size $w = n \times n$. We will assume that the entry `w[i, j]` will store 1 if the sequence with $c_x =$ `cx[j]` and $c_y =$ `cy[i]` converges and 0 otherwise.

(P3) Create a function that receives a pair of numbers c_x and c_y and decides if the sequence in Eq. (1) diverges after $n = 1000$ iterations. The function returns 1 if it converges and 0 otherwise.

One way of knowing if the sequence diverges is to check for the value of $x_n^2 + y_n^2$. When this number is larger that 2^2 the sequence diverges.

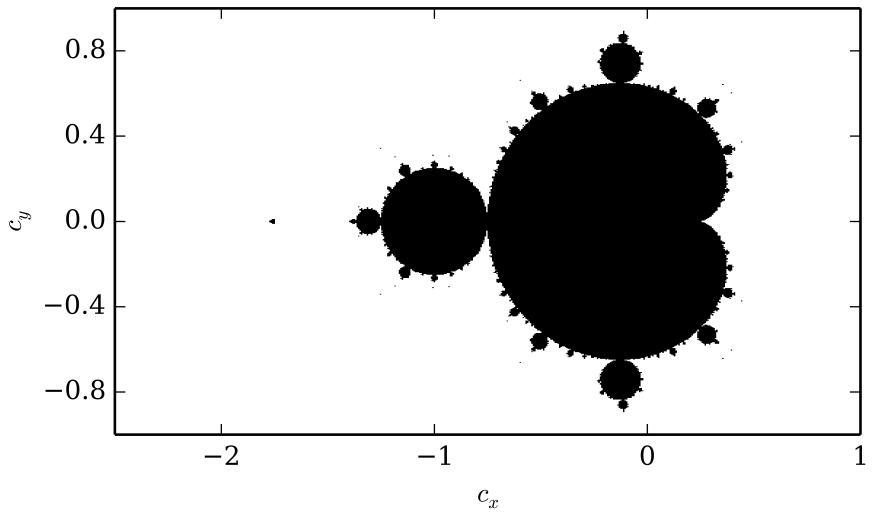
Always use $x_0 = y_0 = 0$

```
def iterate(cx, cy)
```

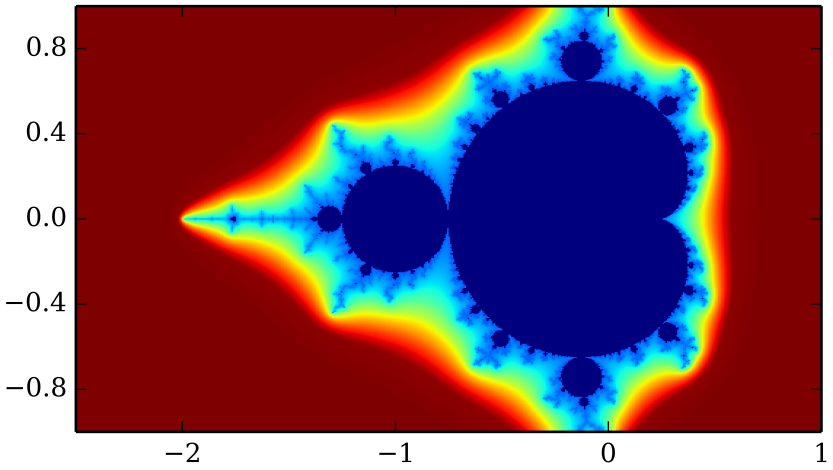
(P4) Call the function defined in the previous step to assess the converge in each pixel of the matrix `w`

(P5) Plot the matrix `w`

(P6) Repeat the same experiment for $N = 128$, $N = 256$ and $N = 512$



(P7) **Advanced Problem:** Encode in each pixel the escape time



3 Solutions

(P1)
```
N = 64
Cx = np.linspace(-2.5, 1.0, num = N, endpoint = True)
Cy = np.linspace(-1.0, 1.0, num = N, endpoint = True)
```

(P2)
```
w = np.zeros([N, N])
```

(P3)
```
def iterate(cx, cy):

    n = 1000
    x = 0
    y = 0
    for i in range(n):
        x1 = x**2 - y**2 + cx
        y1 = 2 * x * y + cy

        x = x1
        y = y1

        if x**2 + y**2 > 4:
            return 0

    return 1
```

(P4)
```
for i in range(N):
    for j in range(N):

        cx = Cx[j]
        cy = Cy[i]
        w[i, j] = iterate(cx, cy)
```

(P5)
```
plt.imshow(w, interpolation='none', cmap=plt.cm.Greys,
    extent=(Cx.min(), Cx.max(), Cy.min(), Cy.max()))
plt.show()
```