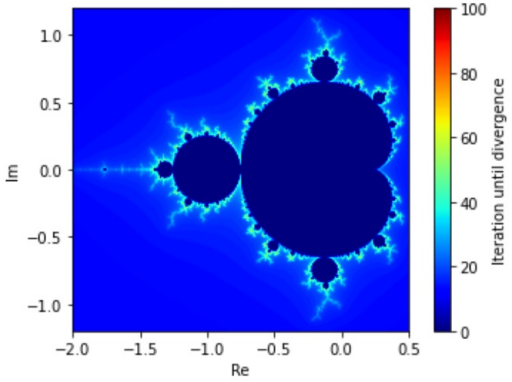
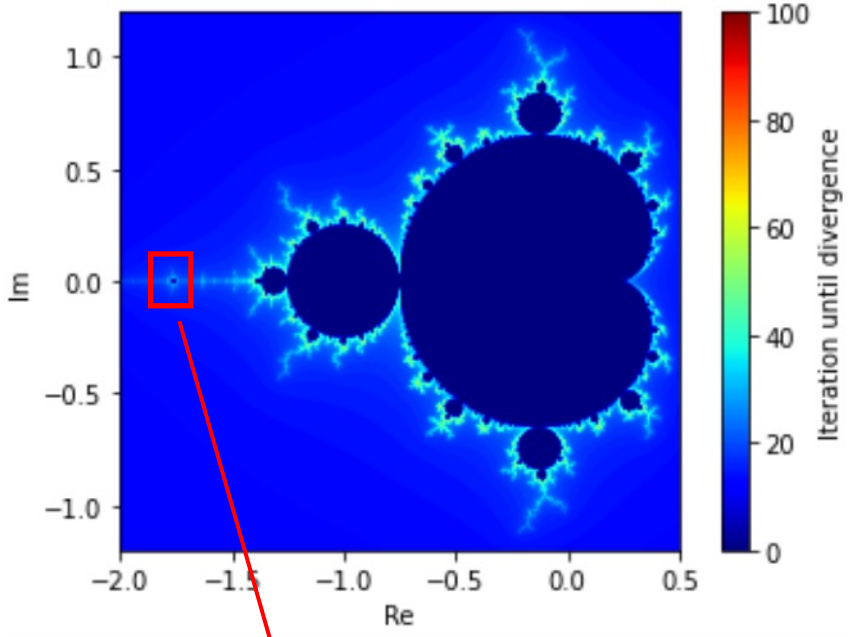
```
pip install --upgrade pip
python -m pip install numba
```

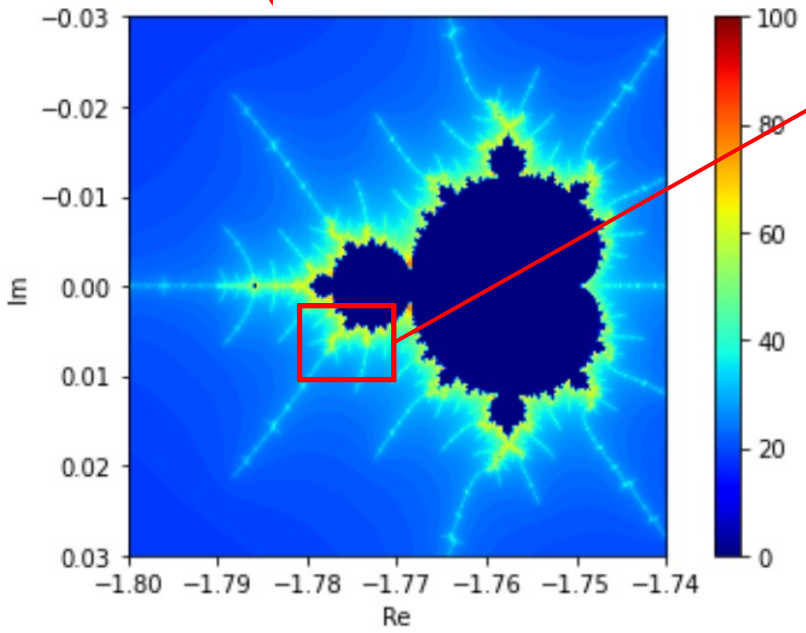
```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.colors import Normalize          # カラーマップを自在に操作するために必要
from numba import jit                            # これが無いとおそろしく計算時間がかかる
import time                                      # 計算時間を見るために必要

t0 = time.time()

@jit                                             # NumbaによるJust In Time Compileを実行
def mandelbrot(c_real, c_imag, n_max):
    Re, Im = np.meshgrid(c_real, c_imag)         # ReとImの組み合わせを計算
    n_grid = len(Re.ravel())                     # 組み合わせの総数
    z = np.zeros(n_grid)                         # マンデルブロ集合のデータ格納用空配列

    # zにマンデルブロ集合に属するか否かのデータを格納していくループ
    for i in range(n_grid):
        c = complex(Re.ravel()[i], Im.ravel()[i])   # 複素数cを定義

        # イタレーション回数nと複素数z0を初期化
        n = 0
        z0 = complex(0, 0)

        # z0が無限大になるか、最大イタレーション数になるまでループする
        while np.abs(z0) < np.inf and not n == n_max:
            z0 = z0 ** 2 + c                     # 漸化式を計算
            n += 1                               # イタレーション数を増分

        # z0が無限大に発散する場合はn, 収束する場合は0を格納
        if n == n_max:
            z[i] = 0
        else:
            z[i] = n

        # 計算の進捗度をモニター(毎ループだと計算が遅くなるため)
        if i % 100000 == 0:
            print(i, '/',n_grid, (i/n_grid)*100)

    z = np.reshape(z, Re.shape)                  # 2次元配列(画像表示用)にリシェイプ
    z = z[::-1]                                  # imshow()で上下逆になるので予め上下反転
    return z

# 水平方向h(実部Re)と垂直方向v(虚部Im)の範囲を決める
h1 = -2
h2 = 0.5
v1 = -1.2
v2 = 1.2

# 分解能を設定
resolution = 4000

# 実部と虚部の軸データ配列、最大イタレーション数を設定
c_real = np.linspace(h1, h2, resolution)
c_imag = np.linspace(v1, v2, resolution)
n_max = 100

# 関数を実行し画像を得る
z = mandelbrot(c_real, c_imag, n_max)

t1 = time.time()
print('Calculation time=', float(t1 - t0), '[s]')


# ここからグラフ表示----------------------------------------
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.set_xlabel('Re')
ax1.set_ylabel('Im')

mappable = ax1.imshow(z, cmap='jet',
                      norm=Normalize(vmin=0, vmax=n_max),
                      extent=[h1, h2, v1, v2])

mappable.set_clim(0, n_max)
cbar = plt.colorbar(mappable=mappable, ax=ax1)
plt.tight_layout()
plt.show()
plt.close()
```



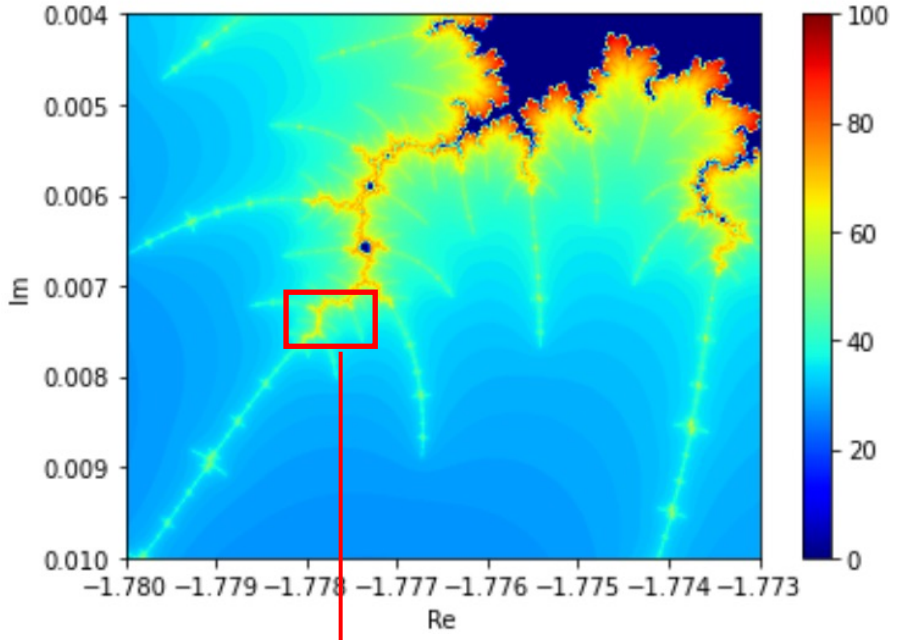
h1 = −2
h2 = 0.5
v1 = −1.2
v2 = 1.2



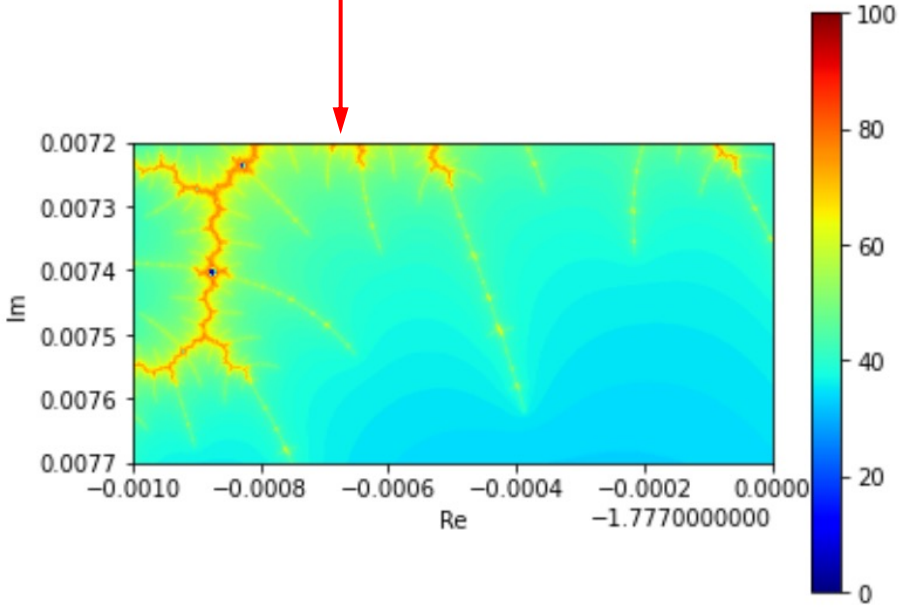
h1 = −1.8
h2 = −1.74
v1 = 0.03
v2 = −0.03



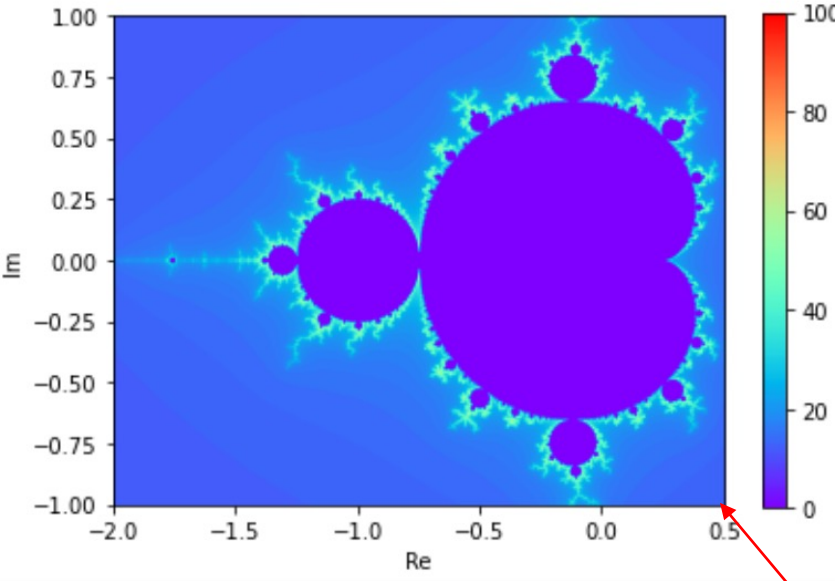
h1 = −1.78
h2 = −1.773
v1 = 0.01
v2 = 0.004



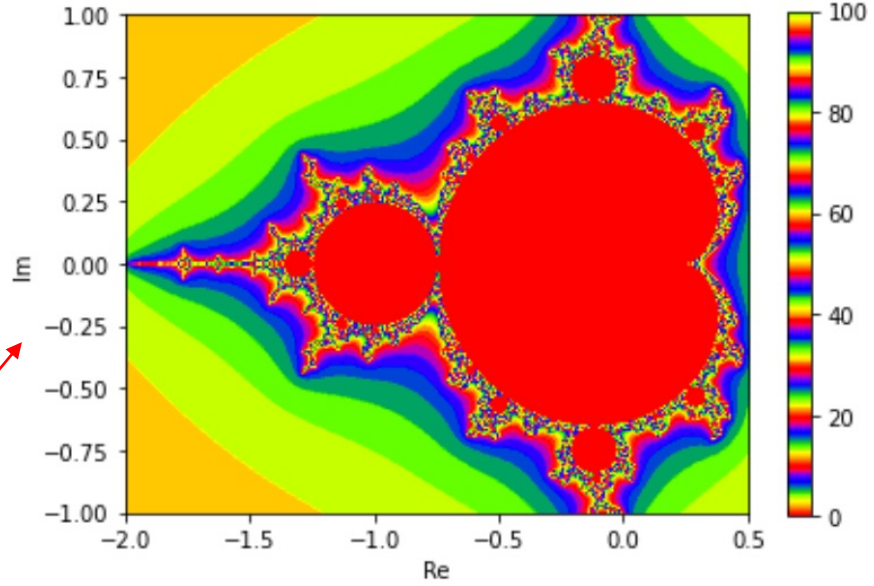
h1 = −1.778
h2 = −1.777
v1 = 0.0077
v2 = 0.0072

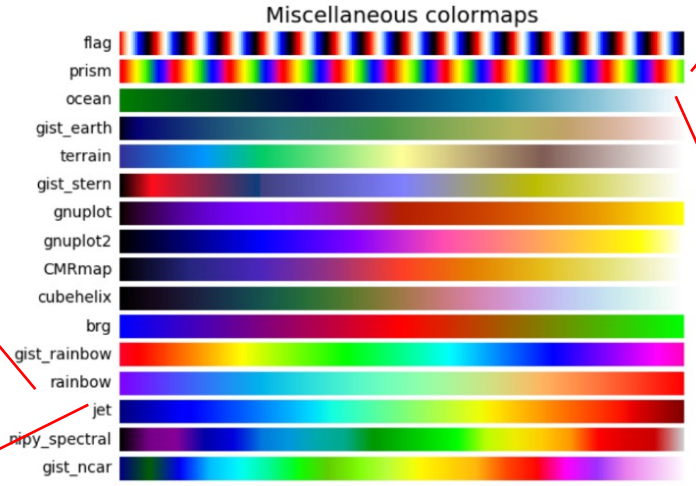


cmap= ‘rainbow’

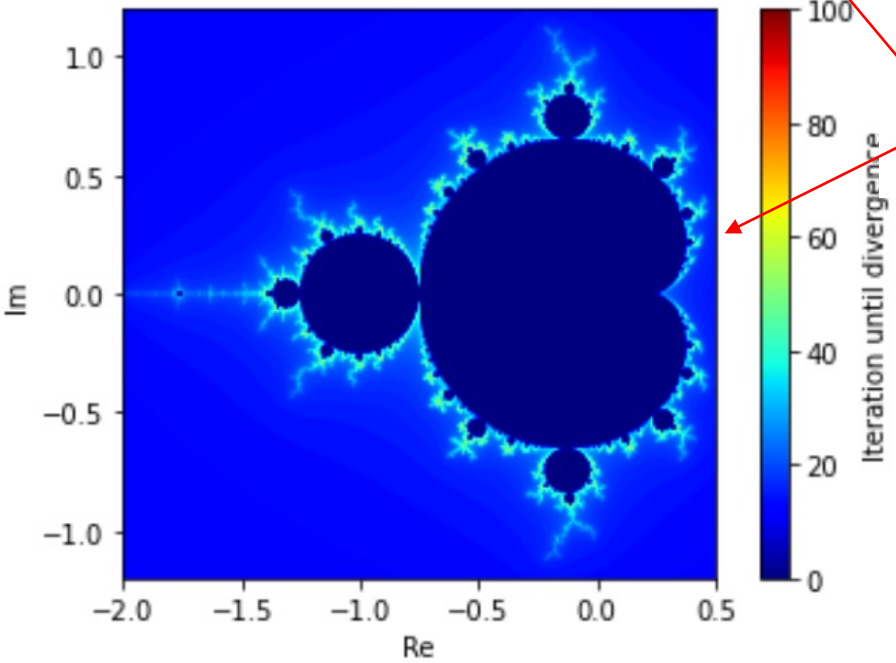


cmap= ‘prism’





cmap=’jet’



cmap= ‘ocean’