python　色鮮やか

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()