# Лекция 9

Python + CUDA:

- **CUDA Python**:
  https://developer.nvidia.com/how-to-cuda-python
- [**PyCUDA:** https://mathema.tician.de/software/pycuda/ ]

## Глоссарий

**Anaconda** — (*Free Open Source Software*) дистрибутив python для научных вычислений.

**NumPy** — библиотека python для поддержки численных расчетов.

**Numba** — оптимизирующий компилятор python для CPU и GPU.

**Matplotlib** — графическая библиотека python.

**PyLab** — процедурный интерфейс Matplotlib.

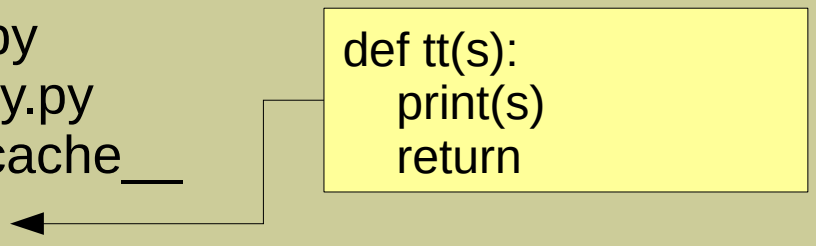
**Module** — файл .py

**Package** — коллекция модулей со структурированным пространством имён.

# Пакеты и модули

```
...ws/numsch> ls -l
total 16
-rw-r--r-- 1 malkov users    2 Apr  2 13:07 __init__.py
drwxr-xr-x 3 malkov users 4096 Apr  2 13:15 interpol
drwxr-xr-x 2 malkov users 4096 Apr  2 13:09 __pycache__
drwxr-xr-x 3 malkov users 4096 Apr  2 13:09 weno
```

```
 ...ws/numsch> ls -l  interpol
 total 16
 -rw-r--r-- 1 malkov users    2 Apr  2 13:08 __init__.py
 -rw-r--r-- 1 malkov users 2458 Apr  2 13:14 interpoly.py
 drwxr-xr-x 2 malkov users 4096 Apr  2 13:18 __pycache__
 -rw-r--r-- 1 malkov users  150 Apr  2 13:18 test.py
```




```
 ...ws> python
 Python 3.5.2 |Anaconda custom (64-bit)| (default, Jul  2 2016, 17:53:06)
 [GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux
 Type "help", "copyright", "credits" or "license" for more information.
 >>> import numsch.interpol.test as t
 >>> t.tt(34.8)
 34.8
 >>>
```

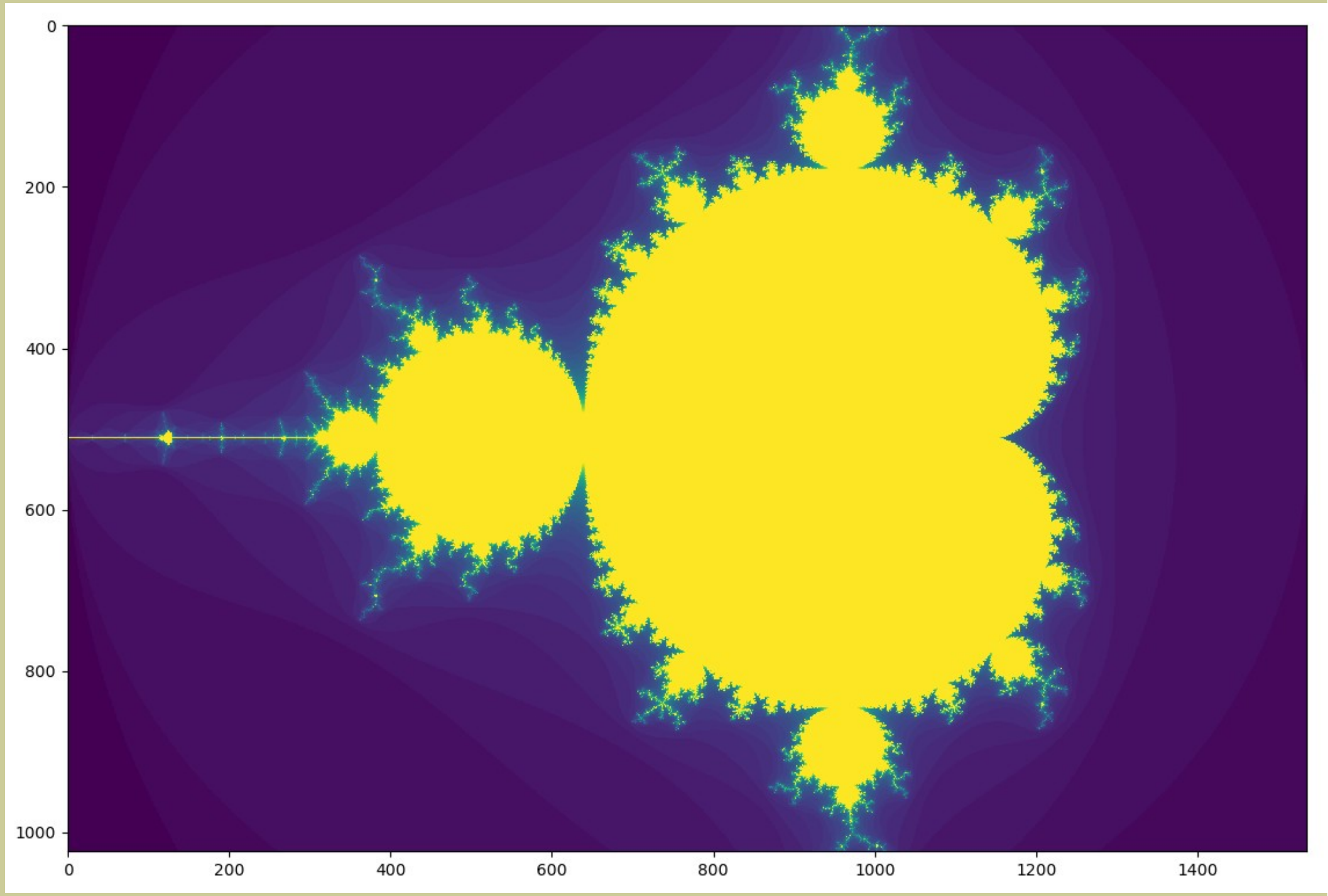
# Сравненние производительности кодов, генерируемых компилятором Numba

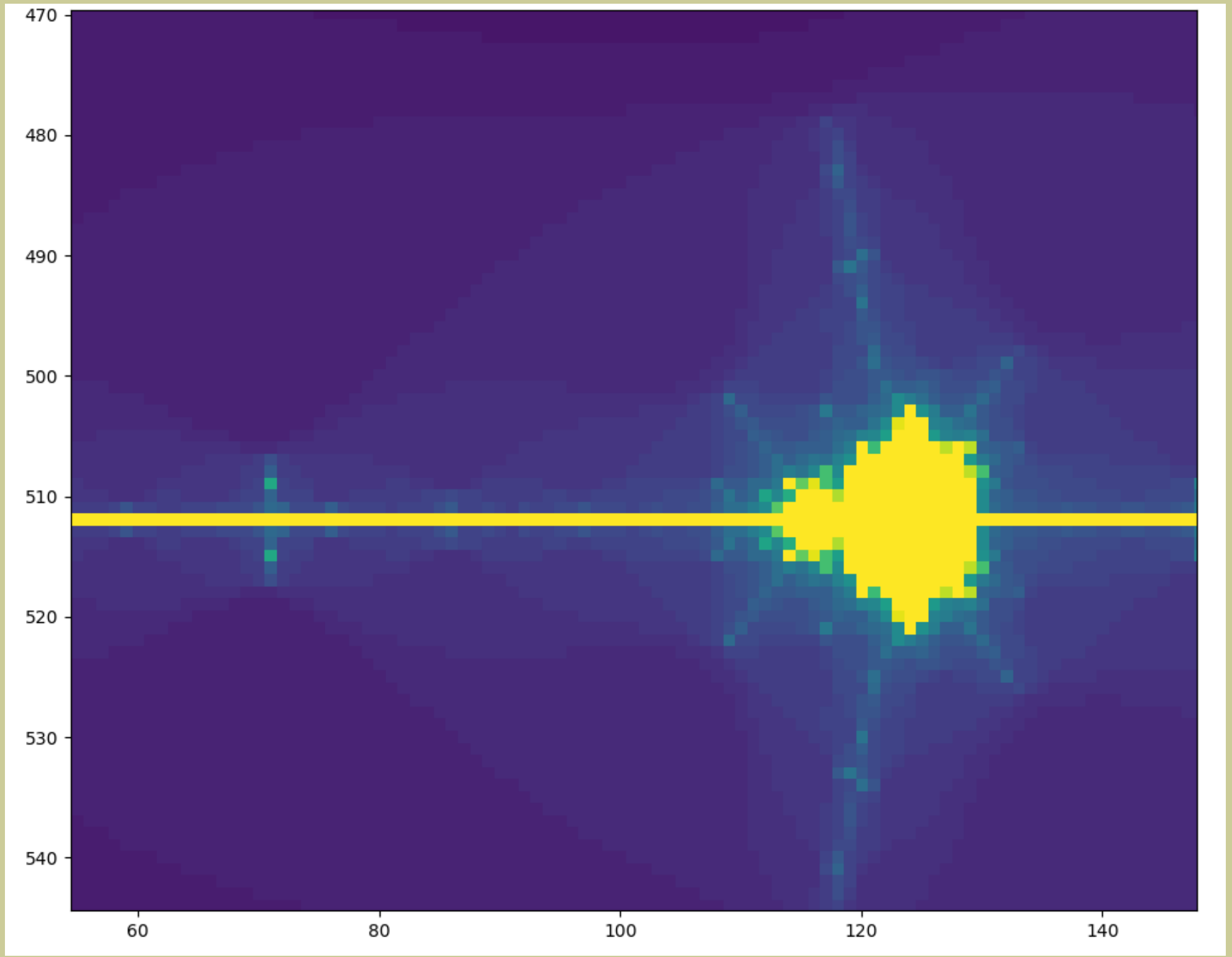
http://nbviewer.jupyter.org/gist/harrism/f5707335f40af9463c43

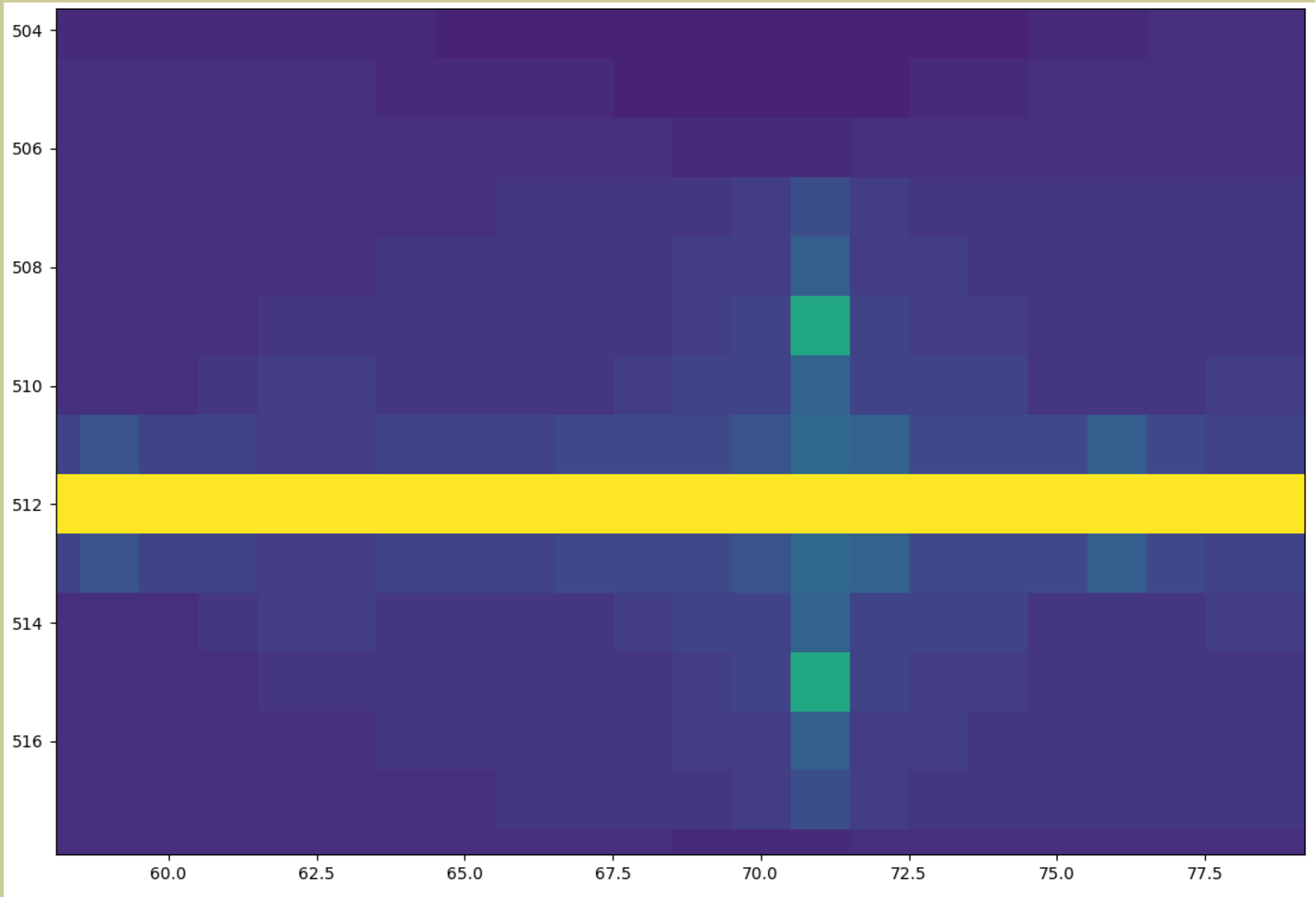
```
import numpy as np
from pylab import imshow, show
from timeit import default_timer as timer

def mandel(x, y, max_iters):
  c = complex(x, y)
  z = 0.0j
  for i in range(max_iters):
    z = z*z + c                                    //отображение Мандельброта
    if (z.real*z.real + z.imag*z.imag) >= 4:
      return i
  return max_iters
```

```
def create_fractal(min_x, max_x, min_y, max_y, image, iters):
  height = image.shape[0]                    #размерности двумерного массива
  width = image.shape[1]

  pixel_size_x = (max_x - min_x) / width          #задание размеров пикселя
  pixel_size_y = (max_y - min_y) / height

  for x in range(width):
    real = min_x + x * pixel_size_x
    for y in range(height):
      imag = min_y + y * pixel_size_y
      color = mandel(real, imag, iters)          #задание цвета пикселя
      image[y, x] = color

image = np.zeros((1024, 1536), dtype = np.uint8)   #инициализация массива
start = timer()
create_fractal(-2.0, 1.0, -1.0, 1.0, image, 20)
dt = timer() - start
print ("Mandelbrot created in %f s" % dt)
imshow(image)
show()
```

Mandelbrot created in **7.139684 s**


0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400


470
480
490
500
510
520
530
540
60
80
100
120
140


504
506
508
510
512
514
516
60.0
62.5
65.0
67.5
70.0
72.5
75.0
77.5

```
…….....
from timeit import default_timer as timer
from numba import autojit

@autojit
def mandel(x, y, max_iters):
…….....
@autojit
def create_fractal(min_x, max_x, min_y, max_y, image, iters):
…….....
```

Mandelbrot created in **0.381171 s**

```
…….....
from timeit import default_timer as timer
from numba import cuda
from numba import  *
@cuda.jit(restype=uint32, argtypes=[f8, f8, uint32], device=True)
def mandel(x, y, max_iters):
…….....
@cuda.jit(argtypes=[f8, f8, f8, f8, uint8[:,:], uint32])
def create_fractal(min_x, max_x, min_y, max_y, image, iters):
  height = image.shape[0]
  width = image.shape[1]
  pixel_size_x = (max_x - min_x) / width
  pixel_size_y = (max_y - min_y) / height

  startX, startY = cuda.grid(2) #threadIdx.x+blockDim.x*blockIdx.x,...
  gridX = cuda.gridDim.x * cuda.blockDim.x;
  gridY = cuda.gridDim.y * cuda.blockDim.y;
  for x in range(startX, width, gridX): #если width>gridX
    real = min_x + x * pixel_size_x
    for y in range(startY, height, gridY):
      imag = min_y + y * pixel_size_y
      image[y, x] = mandel(real, imag, iters)
```

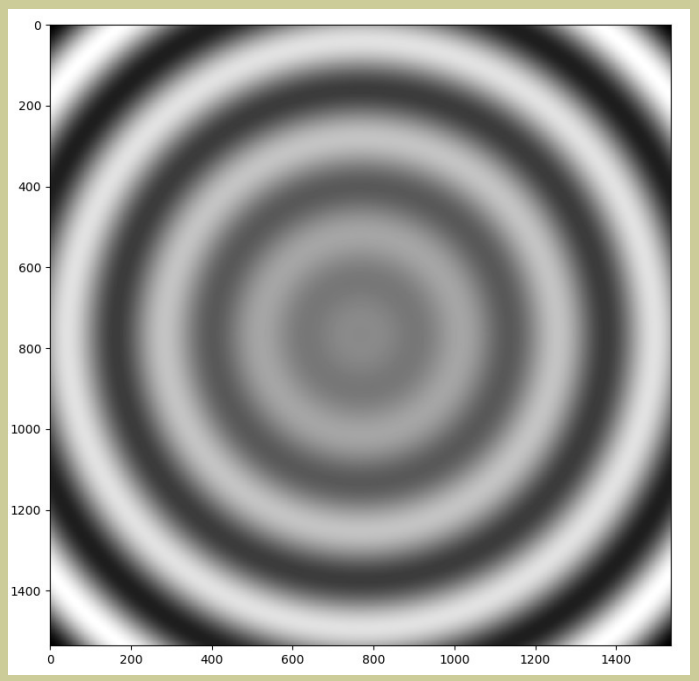
```
image = np.zeros((1024, 1536), dtype = np.uint8)

blockdim = (32, 8)
griddim = (32,16)

d_image = cuda.to_device(image)
create_fractal[griddim, blockdim](-2.0, 1.0, -1.0, 1.0, d_image, 20)
d_image.to_host()
……..
```

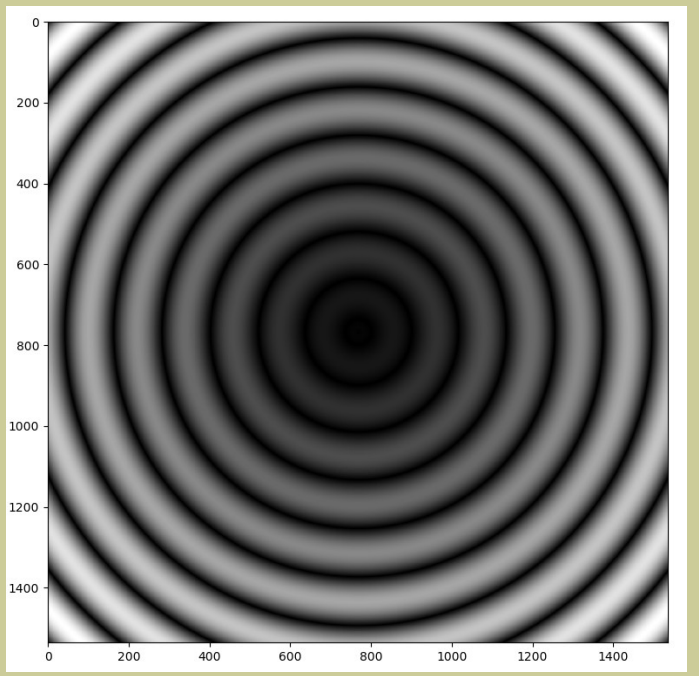
Mandelbrot created in **0.004335 s**

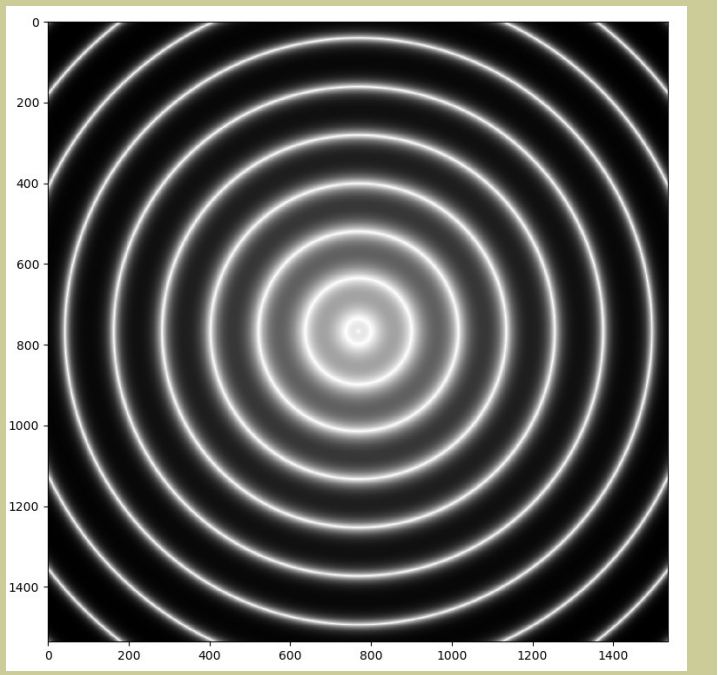
| Технология | Время выполнения *c* | Ускорение |
|---|---|---|
| Python интерпретатор | **7.139684** | 1 |
| Numba jit | **0.381171** | 18.7 |
| CUDA | **0.004335** | 1647.0 |

Спасибо за внимание

# Шлирен метод



```
import numpy as np
from pylab import imshow, show
from timeit import default_timer as timer
from numba import autojit

@autojit
def density(x, y):
    d=(np.cos(np.sqrt(x**2+y**2))*np.sqrt(0.1+x**2+y**2))+20
    return d

@autojit
def create_image(X,Y,min_x, max_x, min_y, max_y, image):
  height = image.shape[0]
  width = image.shape[1]

  pixel_size_x = (max_x - min_x) / width
  pixel_size_y = (max_y - min_y) / height

  for x in range(width):
    X[x] = min_x + x * pixel_size_x
    for y in range(height):
      Y[y] = min_y + y * pixel_size_y
      color = density(X[x], Y[y])
      image[y, x] = color
```

```
image = np.zeros((1536, 1536), dtype = np.float)
X = np.zeros(1536, dtype = np.float)
Y = np.zeros(1536, dtype = np.float)

start = timer()
create_image(X, Y, -20.0, 20.0, -20.0, 20.0, image)
dt = timer() - start

grad_d=np.gradient(image,X,Y)
grad_d_mod=np.sqrt(grad_d[0]**2+grad_d[1]**2)

imshow(image,'gray')
show()

imshow(grad_d_mod,'gray')
show()

imshow(0.8*np.exp(-5.0*grad_d_mod/np.max(grad_d_mod)),'gray')
show()
```