

Лекция 8

Python + CUDA:

- Anaconda Accelerate
(<https://developer.nvidia.com/how-to-cuda-python>).
- PyCUDA (<https://wiki.tiker.net/WelcomePage>).

<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

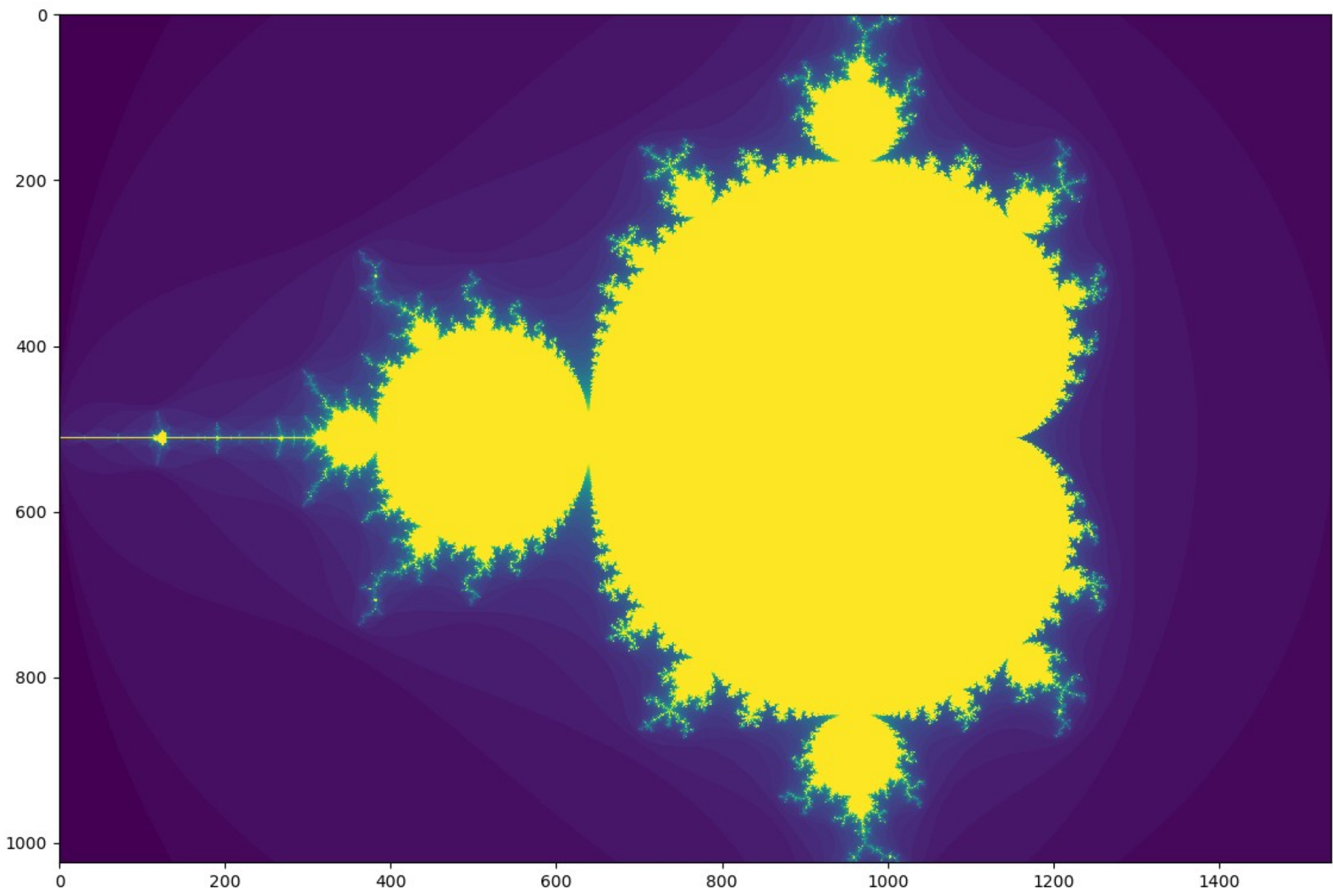
```
        if (z.real*z.real + z.imag*z.imag) >= 4:
```

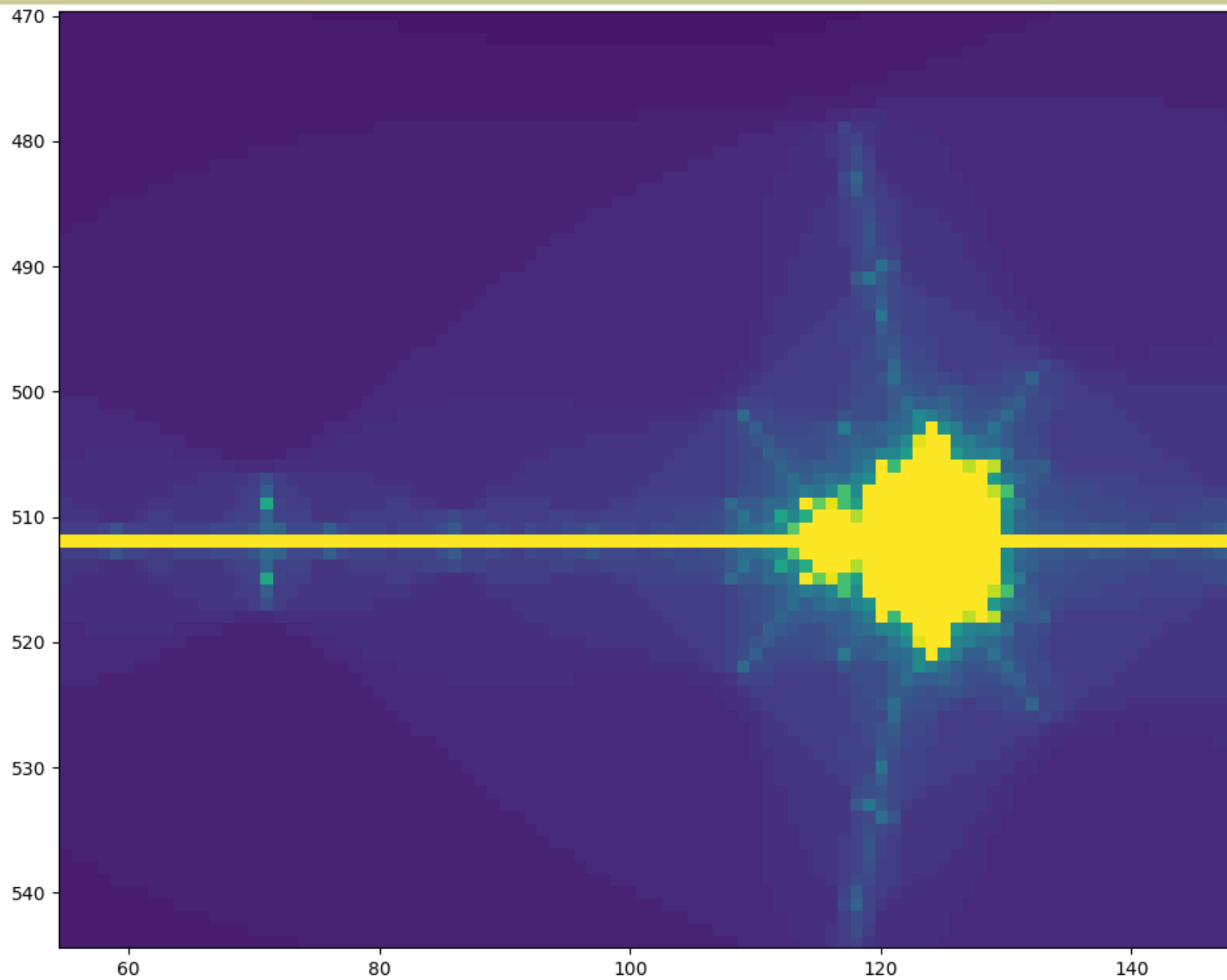
```
            return i
```

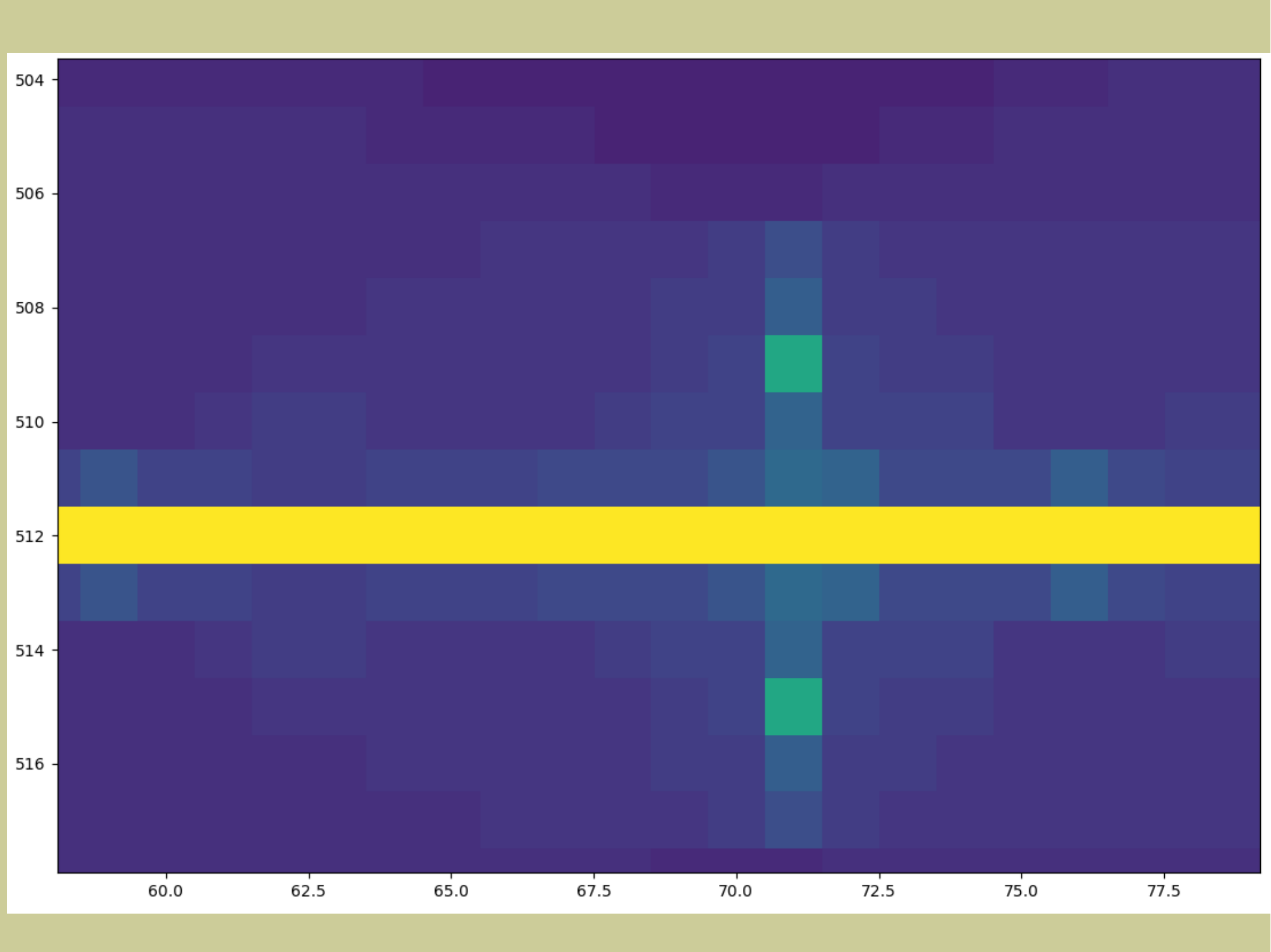
```
    return max_iters
```

```
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**







```
.....  
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)
gridX = cuda.gridDim.x * cuda.blockDim.x;
gridY = cuda.gridDim.y * cuda.blockDim.y;
for x in range(startX, 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)
```

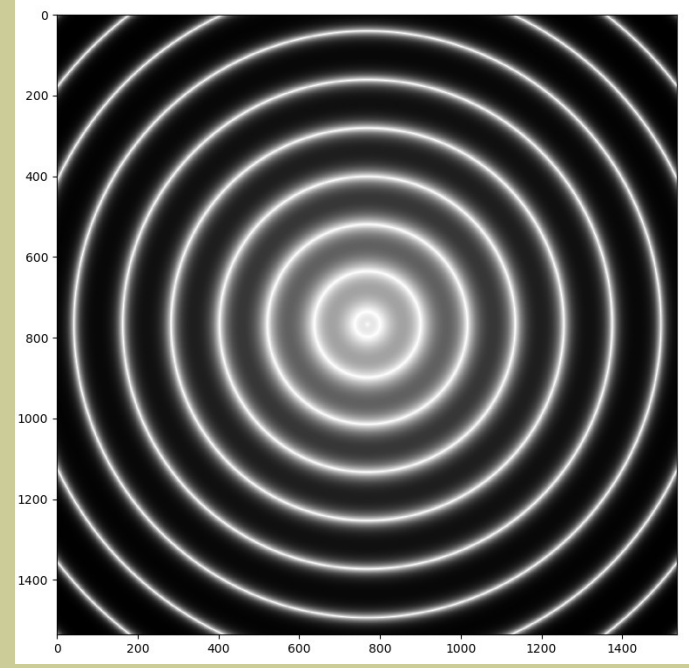
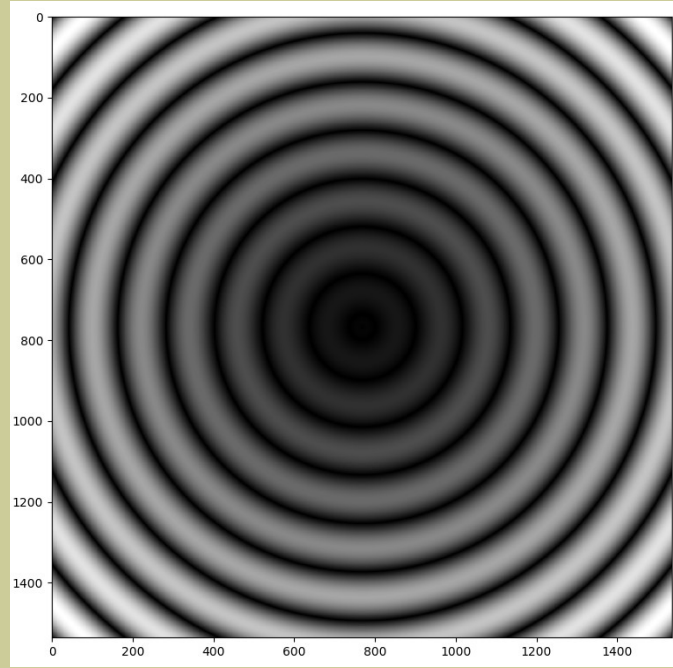
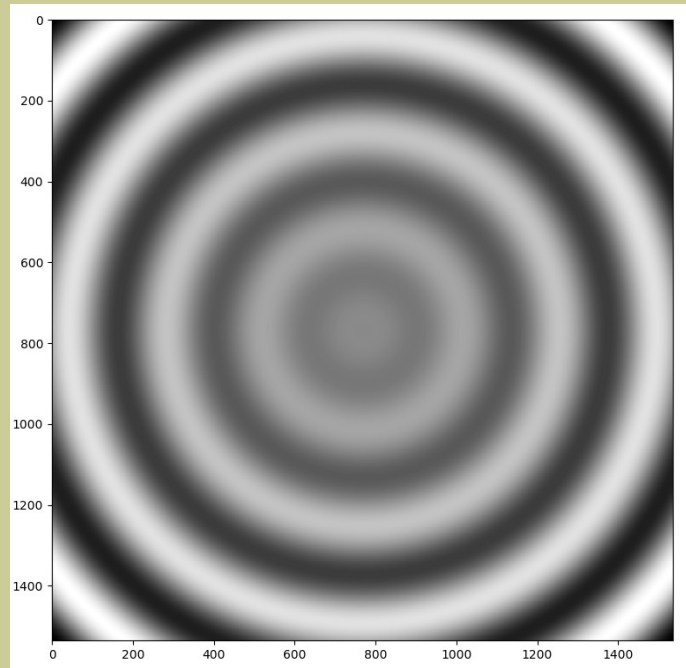
```
d_image.to_host()
```

.....

Mandelbrot created in **0.004335 s**

Технология	Время выполнения с	Ускорение
Python интерпретатор	7.139684	1
Numba jit	0.381171	18.7
CUDA	0.004335	1647.0

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()
```