# Numerical Scientific Computing – Mini Project Part 2

## Optimization of data types

**Size:** 10000

**File:** mandelbrot\_datatypes.py

|  |  |  |
| --- | --- | --- |
|  | Complex64 | Complex128 |
| Float16 | 155.1 | 193.18 |
| Float32 | 157.76 | 199.54 |
| Float64 | 199.08 | 223.37 |

## Execution time between NUMPY and DASK version

**Size:** 8000

**File:** mandelbrot\_dask.py

## Local DASK execution

**Size:** 1000

**File:** mandelbrot\_dask.py

## Distributed DASK execution

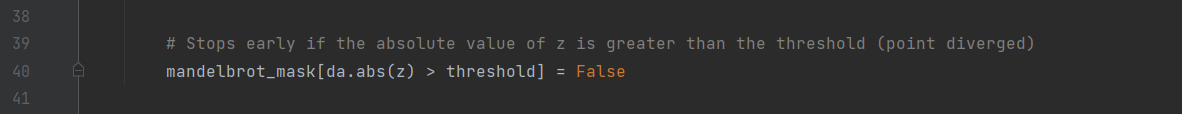
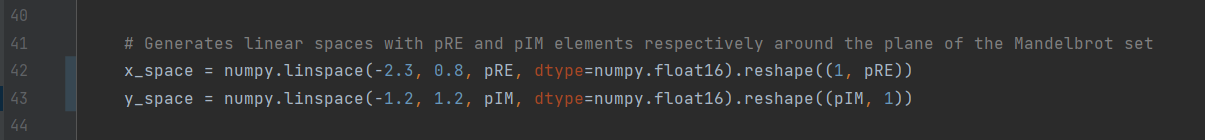
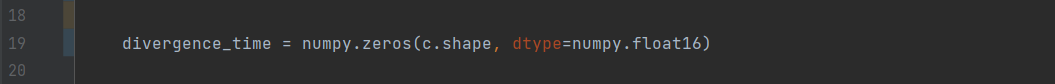
**Size:** 1000

**File:** mandelbrot\_dask.py

## Chunk Size Performance

In the previous two tests, it was found for both local and distributed that a chunk size of 500 was the best performing.

## Improvements/optimizations

1. Stops early if a point is already diverged. See the end of the Mandelbrot function in mandelbrot\_dask.py, mandelbrot\_vectorized.py and mandelbrot\_datatypes.py:  
   
2. The NumPy and Dask version of the Mandelbrot is optimized to run on the most optimal datatype based on the first computation test. See mandelbrot\_vectorized.py and mandelbrot\_dask.py:
3. Using dask.abs() is more optimal than using numpy.abs(), since Dask uses lazy evaluation.  
   