Peer-review of assignment 4 for *INF3331-oyvinssc*

Reviewer 1, syhd, syhd@ifi.uio.no
Reviewer 2, mwaandah, mwaandah@student.matnat.uio.no
Reviewer 3, steinavp, steinavp@student.matnat.uio.no

October 16, 2016

1 Review

System used for review:
OS: Windows 10
Python 3.5 (Anaconda distribution)

General feedback

- Your code is well documented with docstring, which is great for people who will use the code. However, it might be good to also write short comments in the code as well (especially in the pure python and the numpy implementation) to make it easier for people to understand what you are trying to do.
- Good usage of README-files!
- If you try to plot the complex plane for the region $x \in [-2, 1], y \in [-0.5, 1]$, you will see that the image plotted actually appears up-side-down. This error can easily be fixed with the `origin` parameter in the `imshow` function:
  ```
  imshow(matrix, origin='lower', extent=[xmin,xmax,ymin,ymax], interpolation='none')
  ```
- Python is an object oriented language, thus it makes sense to use classes in a Python program. However, for this assignment, using classes is not really necessary, and does not make much sense from an objected-oriented point of view. The computation of the Mandelbrot function is essentially an algorithm, not an object or a concept that exists in the real world. Hence, your code would make more sense if you have implemented the three Mandelbrot implementations as simple Python functions.
- Not sure what operative system you use for testing the code, but when we tried to compile and run your code on a Windows computer or on an IFI-Linux machine, we got some errors. Maybe you can make the following changes in order to make your code more "cross-platform":
  1. Compiling with your setup.py gives the error: numpy/arrayobject.h (no such file or directory) [1]. This can be avoided by adding the line: `include_dirs=[numpy.get_include()]` into the `setup` function.
  2. When running the scripts, the `import` function gives error that it can't import from relative path. So it might be better to use absolute path in this case, by replacing the [dot] (denoting the current folder) with the `mandelbrot` package.
  3. You've used latex to render text on your image by toggle the `usetex` option to `true`, which is great. However, on a machine that does not have latex installed, this might lead to a complete crash of Python.
- Other than the points mentioned above, it looks like you have put a lot of efforts and care into this submission. Your code might be a little hard to follow, but you have done an amazing job to optimize the code to make your implementations faster. I am particularly astonished with your cython implementation, which is full of low-level codes with explicit memory allocation and deallocation, but I believe that that is the key to your implementation's huge performance boost. Great job!

Assignment 4.1: Python implementation

- Everything works as expected.
- Generally, the code is Pythonic, with the usage of list comprehension over traditional loops, as well as the usage of a class as a function (through `__call__`). However, the triple for-loop could easily be replaced by a function similar to `numpy.linspace`. This would make the code easier to read and more Pythonic.
- The code seems a bit too low-level with complicated code for the manipulation of complex numbers. For any calculations with complex numbers, it is much better to simply use Python's inbuilt functionality (e.g. `z=z**2+c`) rather than working on the real and the imaginary parts separately.

[1] `http://stackoverflow.com/questions/14657375/cython-fatal-error-numpy-arrayobject-h-no-such-file-or-directory`

Assignment 4.2: numpy implementation

- Everything works as expected.
- Vectorization is used efficiently with the help of `numpy` package. However, you can optimize the code even more. In your implementation, for every iteration of the for-loop, you re-allocate and then re-initialize the entire Boolean matrix `indices`, which means that your program must go through the entire `z` matrix in every iteration of the for-loop. But this wastes a lot of CPU time since you only want to check the complex numbers that have not escaped 2 yet during the previous iteration. This can be optimized by initializing the `indices` matrix beforehand, and then modify that matrix as you go along inside the for-loop. If you try this code below, the computation time actually reduces almost by half:

```
#...
indices = numpy.ones(c.shape, dtype=bool) #-- initialize all to true
for i in range(1, self.max_escape_time + 1):
    indices[indices] = (z[indices].real**2 + z[indices].imag**2) <= div
    z[indices] = z[indices]**2 + c[indices]
    divergence_steps[indices] = i
return divergence_steps
```

- Maybe you could have used `complex128` as the datatype for the complex numbers instead of just `complex64`. This would add more precision to the computation, which might be necessary if you try to zoom into a very small region on the complex plane.
- report.txt contains the info requested for report2.txt.

Assignment 4.3: Integrated C implementation

- Everything works as expected.
- Cython is used efficiently. Time measurement is amazing! Good usage of functions and datatypes in the C-library (especially with malloc and free). They all seem unnecessary at first, but they are essential for the great performance boost that you have. Great job!
- You've made a tiny little mistake in your code inside the first for-loop where you assign to the variable `c_imag`. Instead of `c_imag = ymin + j*dx`, it should be `c_imag = ymin + j*dy`.
- report.txt contains the info requested for report3.txt

Assignment 4.4: An alternative integrated C implementation

Not implemented.

Assignment 4.5: User interface

- Excellent! Everything works as expected. Thanks to the use of the `argparse` package, your code is short but can do so many things. Invalid inputs are handled in a very good manner.
- Minor mistake: it seems that you've forgotten to call the `main()` function in the user interface.

Assignment 4.6: Packaging and unit tests

Both the packaging and the unit tests work as expected. See general feedback for more comments on setup.py.

Assignment 4.7: More color scales + art contest

- It seems that you haven't provided more color scales other than the default scale used by `matplotlib`.
- You contest image looks alright. You may wanna zoom in a little bit to see more of the Mandelbrot's magnificence.

Assignment 4.8: Self replication

The code works as expected. It's short and simple.