Peer-review of assignment 4 for *INF3331-ReviewedRepoName UiO-INF3331/INF3331-sindrech.git*

Reviewer 1, williada, williada@ulrik.uio.no
Reviewer 2, marcusrg, marcusrg@matnat.uio.no
Reviewer 3, pederbh, pederbh@student.matnat.uio.no

October 13, 2017

General feedback

The tree structure of your code is a bit confusing. You have empty __init__.py lying around, and you don't need too have a copy of all scripts in two different directories. You can just copy the methods into a single __init__.py. We are not sure, but this might be why you need to have copies of .py files in many different directories (?) It would also be nice with some comments regarding the code i.e assumptions and limitations. We found errors in many of your functions, but they are usually due to a for-loop running too often.

Assignment 4.1

- The error margin in constant function is not entirely correct. It is supposed to have an error margin of 1/N.
- The name of the test are good and easy to understand
- constant function also returns the wrong answer for other ranges, but since we were not expected to test for other than 0 to 1 that is fine. That being said, the function returns an answer above the true answer, not below. We think this is because you iterate one to many times.

```
#Do this instead
for i in range(0, n-1):
```

Assignment 4.2

- We would like to see the image generation to happen in its own method. For testing and general tidy and clean code.
- Agiain, the loop runs too many times, and your answer is therefore incorrect. It gets closer to the true value 1/3, but from above. See 4.1 for fix.
- The quadratic_error.png is completely wrong. You were supposed to make a graph that showed how close your function (for different N values) is to the true answer. That means how close computed and expected answer is for a range of N. Your graph shows the areal of rectangles under the graph for the points in the x-axis.

Assignment 4.3

- Implementation seems to be correct, and vectorizing is used.
- Would like to see a wider range of N in time comparison.

Assignment 4.4

- Again, the result is not entirely correct. This is also due to the loop running too many times. Try this instead

```
for i in range(0, n):
```

- We would also recommend to put the for-loop into its own function, and @jit it. This will speed up the computation, and will be faster than numpy for more advanced functions and hight N's.

Assignment 4.5

Reviewing a INF3331 student.

Assignment 4.6

- midpoint_Integrate has the same problem as in 4.1, 4.2 and 4.4 you run the loop one time to many.
- midpoint Integrate with numpy is not correct. You are thinking in the right direction, but you are supposed to adjust the points half an interval to the left, not the right. So do this instead:

```
res=func(x-intervall/2)
```

- Same goes for midpoint_integrate in numba, but here the adjustment is not implemented.
- The report and test file are also missing the calculation for the required N such that you get to within 10e-10 of the actual answer.

Assignment 4.7

The module works fine, but there are several empty __init__.py files whereas you could paste the methods in a single one. But that is a matter of preference.

Assignment 4.8

Looks good!