# Peer-review of assignment 4 for *INF3331-aschowdh*

**daniejak, daniejak@student.matnat.uio.no**
oliveref, oliveref@uio.no
zuitaom, zuitaom@ifi.uio.no

October 13, 2017

## 1 Introduction

### 1.1 Goal

The review should provide feedback on the solution to the student. The main goal is to *give constructive feedback and advice* on how to improve the solution.

## 2 Review

**System:** Ubuntu 16.10, Python 3.5.2 :: Anaconda 4.2.0 (64-bit)

### General feedback

The main thing i noticed is that you will benefit from making more generalized functions. Especially in longer functions, splitting them up and allowing them to take different data (ex the integration functions) as input allows you to save a lot of code while making things more readable. Calls to function names (ex integrate.__name__) could also save some space in printing.

### Assignment 4.1

The tests are good and readable. For testing the pure python integrate method everything looks perfect here. One thing to note is that you could easily have implemented them generally to take integration functions as input. That way testing for any type (cython, numba etc) will be as easy as copy-pasting a call with that integration function.

The assignment didn't explicitly require tests for anything but integrate and numpy integrate, but getting a better result with less work is never wrong.

### Assignment 4.2

The integration method is nice, also a good thing you included some error handling (never easy to know the right amount of these in python, it's more a question of what can't go wrong than what can).

It seems you've misunderstood the plotting part however. The task wanted the error-margin as a function of integration points N.

### Assignment 4.3

Looks good, pretty standard implementation i'm guessing 90% of us handed in. It might be more logical to put the function f in whatever file you call the integration from.

The report is decent, you observe an improvement with Numpy. If you had gone a bit deeper(higher N, more measurements) you would probably have gotten some data to support the oblig's assertion that integrate() scales linearly with N, while numpy_integrate should not.

### Assignment 4.4

Same as 4.3, pretty standard implementation that looks totally ok. The only question: what is numpy_integrate doing in this file?
The report is here again fairly extensive which is good. You make an observation as to why Numba outperforms Numpy for larger points N, which if i understand correctly (Numpy is memory gated to a larger extent), makes good sense.

## Assignment 4.5

I have no experience with Cython, so fairly hard to give any constructive feedback here. The report is extensive which is good. In it you point out the function might not be properly cythonised, so that could have been something to improve.

## Assignment 4.6

The general comment i made about splitting functions mainly applies to this section. I think you make this a bit more complicated than it has to be. In error() you already have something good going with the list and check lists, expanding that combined with some splitting of the function would have looked real nice. Same goes for performance().

The midpoint integration functions looks good, it might have made more sense to put them in their respective integration files (ex midpoint_integrate goes in integrate.py). That is very minor however, and to me it seems the oblig suggests your approach anyway.

## Assignment 4.7

The setup.py script works, the package it generates looks valid (both integrate and test folders with respective files included).

## Assignment 4.8

Again an extensive report, the code also looks fine. Nothing much to say here.