**Numerical Analysis**

**Final task**

Submission date: 12/2/2021 8:00am

This task is individual. No collaboration is allowed. Plagiarism will be checked and will not be tolerated.

The programming language for this task is Python 3.7. You can use standard libraries coming with Anaconda distribution. In particular limited use of numpy and pytorch is allowed and highly encouraged.

**You must not use those parts of the libraries that implement numerical methods taught in this course.** This includes, for example, finding roots and intersections of functions, interpolation, integration, matrix decomposition, eigenvectors, solving linear systems, etc. You must not use reflection (self-modifying code).

Attached are mockups of for 4 assignments where you need to add your code implementing the relevant functions. You can add classes and auxiliary methods as needed. Unittests found within the assignment files must pass before submission. You can add any number of additional unittests to ensure correctness of your implementation.

In addition, attached are two supplementary python modules. You can use them but you cannot change them.

Upon the completion of the final task, you should submit the four assignment files and this document with answers to the theoretical questions archived together in a file named <your ID>.zip

All assignments will be graded according to **accuracy** of the numerical solutions and **running time**.

Expect that the assignment will be tested on various combinations of the arguments including function, ranges, target errors, and target time. We advise to use the functions listed below as test cases and benchmarks:

1. For Assignment 4 see sampleFunction.\*

**Assignment 1 (30pt):**

Implement the function **Assignment1.interpolate(..)**.

The function will receive a function f, a range, and a number of points to use.

The function will return another “interpolated” function g. During testing, g will be called with various floats x to test for the interpolation errors.

Grading policy:

Running time complexity > O(n^2): 0-20%

Running time complexity = O(n^2): 20-60%

Running time complexity = O(n): 50-100%

The grade within the above ranges is a function of the average relative error of the interpolation function at random test points. Correctly implemented linear splines will give you 50% of the assignment value.

**Question 1.1:** Explain the key points in your implementation.

|  |
| --- |
|  |

**Assignment 2 (10pt):**

Implement the function **Assignment2.intersections(..)**.

The function will receive 2 functions- , , and a float maxerr.

The function will return an iterable of approximate intersection Xs, such that:

Grading policy: The grade will be affected by the number of correct/incorrect intersection points found and the running time of **Assignment2.intersections(..)**.

**Question 2.1:** Explain the key points in your implementation.

|  |
| --- |
|  |

**Assignment 3 (25pt):**

Implement a function **Assignment3.integrate(…)** and **Assignment3.areabetween(..)** and answer two theoretical questions.

**Assignment3.integrate(…)** receives a function f, a range, and several points to use.

It must return approximation to the integral of the function f in the given range.

You may call f at most n times.

Grading policy: The grade is affected by the integration error only, provided reasonable running time e.g., no more than 5 minutes for n=100.

**Question 3.1:** Explain the key points in your implementation of Assignment3.integrate(…).

|  |
| --- |
|  |

**Assignment3.areabetween(..)** receives two functions .

It must return the area between .

In order to correctly solve this assignment you will have to find all intersection points between the two functions.

Grading policy: The assignment will be graded according to the integration error and running time.

**Question 3.2:** Explain the key points in your implementation of Assignment3.areabetween (…).

|  |
| --- |
|  |

**Question 3.3:** Explain why is the function is difficult for numeric integration with equally spaced points?

|  |
| --- |
|  |

**Question 3.4:** What is the maximal integration error of the in the range [0.1, 10]? Explain.

|  |
| --- |
|  |

**Assignment 4 (35pt).**

Implement the function **Assignment4.fit\_shape(…)** and the class **MyShape**

The function will receive a generator (a function that when called), will return a point (tuple) (x,y), a that is close to the shape contour.

Assume the sampling method might be noisy- meaning there might be errors in the sampling.

The function will return an object which extends **AbstractShape**  
When calling the function **AbstractShape.contour(n)**, the return value should be array of n equally spaced points (tuples of x,y).

Additional parameter to **Assignment4.fit\_shape** is maxtime representing the maximum allowed runtime of the function, if the function will execute more than the given amount of time, the grade will be significantly reduced.

You may use any numeric optimization libraries and tools for solving this assignment. Reflection is not allowed.

Grading policy: the grade is affected by the error of the area function of the shape returned by Assignment4.fit\_shape.

**Question 4.1:** Explain the key points in your implementation.

|  |
| --- |
|  |