**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 should 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.

The use of the following methods in the submitted code must be clearly announced in the beginning of the explanation of each assignment where it is used and will result in reduction of points:

numpy.linalg.solve (15% of the assignment score)

(not studied in class) numpy.linalg.cholesky, torch.cholesky, linalg.qr, torch.qr (1% of the assignment score)

numpy.\*.polyfit, numpy.\*.\*fit (40% of the assignment score)

numpy.\*.interpolate, torch.\*.interpolate (60% of the assignment score)

numpy.\*.roots (30% of the assignment 2 score and 15% of the assignment 3 score)

All numeric differentiation functions are allowed (including gradients, and the gradient descent algorithm).

Additional functions and penalties may be allowed according to requests in the task forum.

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. At least half of the test functions will be polynomials. Functions 3,8,10,11 will account for at most 4% of the test cases. All test functions are continuous in the given range. If no range is given the function is continuous in .

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): 20-80%

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.

Solutions will be tested with on variety of functions at least half of which are polynomials of various degrees with coefficients ranging in .

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

|  |
| --- |
| The algorithm is piecewise hermite interpolation. It has three parts. The first two are for the cases where . Otherwise, the algorithm need at least n bigger or equal to 4, since hirmate require for each . The run time complexity to generate the required function is and for the returned function itself is . |

**Assignment 2 (15pt):**

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.

|  |
| --- |
| The algorithm runs over the interval with some constant step size*.* At each step if the current value is less or equal to the then the current value of x is returned. Otherwise, the algorithm check if the multiplication of current y and the next y is smaller than 0, then it run the bisection method on the current x and the next one. The algorithm stops when the current x is greater than b. |

**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(…).

|  |
| --- |
| The algorithm implements the as taught in class. In order to minimalize the computation error, the values were sorted and then added accordingly. |

**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. You may ignore all intersection points outside the range .

Note: there is no such thing as negative “area”.

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 (…).

|  |
| --- |
| The algorithm from question 2 were used to find the intersection points. Afterwards the area between each two neighboring points is calculated using the integrate method and added up to get the required area. |

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

|  |
| --- |
| This function has large oscillations near zero and its limit in is zero. Therefore, equality spaced pointswill not be accurate enough near zero since numeric integration used points to estimate the area, but those large oscillations require concentrated points to be accurate. |

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

|  |
| --- |
| Composite Simpson rule for a=0.1 and b=10 with n = 2. The absolute error is ~-2.9since this algorithm is less accurate when we use is it with small number of points. |

**Assignment 4A (20pt)**

Implement the function **Assignment4A.fit(…)**

The function will receive an input function that returns noisy results. The noise is normally distributed.

Assignment4A.fit should return a function fitting the data sampled from the noisy function. Use least squares fitting such that will exactly match the clean (not noisy) version of the given function.

To aid in the fitting process the arguments and signify the range of the sampling. The argument is the expected degree of a polynomial that would match the clean (not noisy) version of the given function.

You have no constrains on the number of invocation of the noisy function but the maximal running time is limited. Additional parameter to **Assignment4A.fit** 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.

Grading policy: the grade is affected by the error between (that you return) and the clean (not noisy) version of the given function, much like in Assignment1. 65% of the test cases for grading will be polynomials with degree up to 3, with the correct degree specified by . 30% will be polynomials of degrees 4-12, with the correct degree specified by . 5% will be non-polynomials

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

|  |
| --- |
| I implemented polynomial regression using the method of least squares. The algorithm builds a polynomial function in degree d. The main target is to find the coefficients. The coeffects can be calculated using matrix multiplication. |

**Assignment 4B (10pt + bonus 20pt).**

Implement the function **Assignment4.area(…)**

The function will receive a shape contour and should return the approximate area of the shape. Contour can be sampled by calling with the desired number of points on the contour as an argument. The points are roughly equally spaced.

Naturally, the more points you request from the contour the more accurately you can compute the area. Your error will converge to zero for large . You can assume that 10,000 points are sufficient to precisely compute the shape area. Your challenge is stopping earlier than according to the desired error in order to save running time.

Grading policy: the grade is affected by your running time.

**Question 4B.1:** Explain the key points in your implementation.

|  |
| --- |
| The algorithm implements the idea that was presented at class. Its splits the contour points intro small groups. For each group the algorithm calculates the area under it using composite trapezoidal rule. If the x of the first member in the group is bigger than the last x value of the last member, then it multiply the area by -1. Its add up all of those areas and return the sum of them. |

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.

In this assignment only, you may use any numeric optimization libraries and tools. 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 4B.2:** Explain the key points in your implementation.

|  |
| --- |
|  |