**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.

|  |
| --- |
| Lagrange I know its n^2 but I had problems with calculating c in newton method  Before I do Lagrange I extract n points from the function given to me with equal range between each other  note that il line 59 I write the algorithm in comment because for some reason it goes into a infinite loop with the fallowing code-  arr = [lambda x:x]\*10  for i in range(1,10):  arr[i] = lambda x: arr[i-1](x)+1  print(arr[7](3))  So I fix it by doing recursion |

**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\_secant\_method  we do g(x) = f1(x) – f2(x) then I look at n points in the range(a,b) if we flip sign I take note  then for each time we flip sign I do the secant method  n is random number I give at start I give it 100 as I can never find everything  for example if your function has if(number <x < number I don’t hit) then  f(x)= f(x)\*sin(x^a very big number) il never see the infinite number of intersections there |
|  |

**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(…).

|  |
| --- |
| We just create equal distant points and call it dx then we calculate the area in the trapezoid dx\*(b1+b2) /2 |

**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 (…).

|  |
| --- |
| Here I don’t use trapezoid but just for each xi point I do (f1(xi)-f2(xi)) \*dx  Also I use the assignment2 intersections to and use the start and end of the area as my starting points |

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

|  |
| --- |
| Like said in questions 2 when you have an erraticfunction when we take a sample with dx distance we can totally miss that it my have gone very much down then up again do to something like 2^(a big number)  The we will get will depend very much on where we take the sample as a very small diffrint in dx makes a very big different in dy for 10% of the area we are looking at |

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

|  |
| --- |
| **Error <= (b-a)^5 /(180\*n^4) \* max of (d^4/dx^4 ( f(x))) assuming n = 100**  **Which is something about**  **Error =(10-0.1)^5/(180\*100^4) \*** max{d^4/(dx^4)(2^(1/x^2) sin(1/x))|0.1<x<10} = 426924748403407756374269031102808064 at x≈0.110472  **so as we get closer to 0 the number keeps getting bigger and bigger** |

**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.

|  |
| --- |
| Least square fitting  We create a liner equation for a coefficient then solve it using then determinants of the matix using Cramer’s rule  matrix is( k\*k \* creating sum )-> I make it so it takes O(k\*n) to create the matrix and not k\*k\*n. Then I use np.det for each matrix.  the problem become the time it takes to do all this and the time it takes to take a sample  I place a hard cap of 1500 points  But before I start the I look how much time it takes to calcite the matrix for x^k when k is 1..12  and I tell it to use 3\* (maxtime/time\_for\_poly\_deggree\_k) also I make sure that we don’t take more then maxtime/4 for the sampling of the given function |

**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.

|  |
| --- |
| Il create 2 point array f1 and f2 which represent half of the shape each(up and down side)  No like assignment 3 I have 2 functions of points and I just need to calc the area between the 2.  Now how do we pick how many points to take  Lets look for a sec at the a graph where the x cords are n and the y cords are the error from the real area of the shape.  Lets now pick 2 random number n lets say n1 = 21 and n2 = 33 we can now draw a line of the liner function f that pass throw n1 and n2 we can find N where the value where f(N) < maxerror  Obviously the realy function f is not liner and acts more like 1/n but for simplistic sake I just took 2 points and created a liner function to find N  if I had a bit more time I would create about 10 points n1…n10 and find a polynom that acts like 1/n and try to find N that way |

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.

|  |
| --- |
| Il explain what I wanted to do hopping to get some points (I did not understand what needs to be the return value so I stop mid way)  I start by sorting N points using a type of quicksort algorithm that works for x and y axis – the full algorithm is a bit more complex and I’m not going to fully explain it- this is already implemented and it draw the points sorted if you remove the comment at line 283 and 284 in the assignment4B.py  The idea was to take every 4 fallowing points and create a bizarre line something that should have been really easy for me for me but considering I created a 3D bizarre in the past in a different course  Link –  Then just apply it to my shape but I don’t get what is myshape as for how many point it will it ..  It will spend about 1/8 of the time taking points (like I did 4A) as sort algorithms are not the most effective  And creating a bizarre should take almost no time compared to sorting as it is O(n) where a sort is O(log(n)\*n) |