

Numerical Analysis

Final task

Submission date: **TBD** (strict).

This task is individual. No collaboration is allowed. Plagiarism 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 (unless explicitly stated otherwise in the instructions of the particular assignment). This restriction 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 deduction of points. Failure to announce the use of any restricted functions will result in disqualification of the assignment.

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

(not studied in class) numpy.linalg.cholesky, torch.cholesky, linalg.qr, torch.qr (10% 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)

TBD – restriction of numeric differentiation functions

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. BUT! existing unit tests are provided for demonstration and to encourage you to write additional tests as you go. You can add any number of additional unittests to ensure correctness of your implementation. Passing only the existing unittests does not ensure that your code will not fail in all cases. It is your responsibility to test your code and ensure that it is stable. You should add 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 five assignment files and this document with answers to the theoretical questions. **TBD specific submission instructions.**

Assignments will be graded according to **error** of the numerical solutions and **running time**. Some assignments have required specific error bounds – they will be graded according to running time. Some

assignments limit the running time – they will be graded according to error. For all executions there is 2 minutes running time cap after which the execution will be halted.

Every assignment will be AUTOMATICALLY tested on a number of different functions and different parameters. It may be executed multiple times on the same function with the same parameters. Every execution will start with a clean memory. Any exception thrown during an execution will render the execution invalid and nullify its contribution to the grade. **Test your code!!!**

Any disqualification of an assignment (e.g. due to unannounced use of restricted functions) or an execution (e.g. due to exception) will not contribute to the grade regardless the effort put in the development.

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 – add additional unittests with implementations of these functions. At least half of the test functions will be polynomials. Functions 3,8,10,11 will account for at most 5% of the test cases. All test functions are continuous in the given range. If no range is given the function is continuous in $[-\infty, +\infty]$.

1. $f_1(x) = 5$
2. $f_2(x) = x^2 - 3x + 5$
3. $f_3(x) = \sin(x^2)$
4. $f_4(x) = e^{-2x^2}$
5. $f_5(x) = \arctan(x)$
6. $f_6(x) = \frac{\sin(x)}{x}$
7. $f_7(x) = \frac{1}{\ln(x)}$
8. $f_8(x) = e^{e^x}$
9. $f_9(x) = \ln(\ln(x))$
10. $f_{10}(x) = \sin(\ln(x))$
11. $f_{11}(x) = 2^{\frac{1}{x^2}} * \sin\left(\frac{1}{x}\right)$
12. For Assignment 4 see sampleFunction.*

Assignment 1 (15pt):

(10pt) Implement the function `Assignment1.interpolate(..)` following the pydoc instructions.

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%

Running time complexity will be measured empirically as a function of n .

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 $n \in \{1, 10, 20, 50, 100, 200, 500, 1000\}$ on variety of functions at least half of which are polynomials of various degrees with coefficients ranging in $[-1, 1]$.

(5pt) Question 1.1: Explain the key points in your implementation.

השתמשתי ב- Cubic Bezier Curves ובנוסף בשביל ליעל את זמן הריצה השתמשתי גם ב-Thomas algorithm. בין כל שתי נקודות ידועות ביצעתי אינטרפולציה היוצרת פונקציה התלויה ב t הנמצא בטווח $[0, 1]$ והפלט של אותה פונקציה הוא (x, y) . מצאתי מבין כל הפונקציות את הפונקציה המתאימה לערך הקלט x , נרמלתי את x לערך בין $[0, 1]$ והחזרתי את המבוקש.

Assignment 2 (15pt):

(10pt) Implement the function **Assignment2.intersections(..)** following the pydoc instructions.

The function will receive 2 functions- f_1, f_2 , and a float $maxerr$.

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

$$\forall x \in X, |f_1(x) - f_2(x)| < maxerr$$

$$\forall x_i, x_j \in X, |x_i - x_j| > maxerr$$

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

(5pt) **Question 2.1:** Explain the key points in your implementation in particular explain how did you address the problem of finding multiple roots.

על מנת למצוא את החיתוכים של שתי הפונקציות, מצאתי במקום את השורשים של פונקציית ההפרש בין הפונקציות. חילקתי את הטווח למרווחים שווים כתלות בגודל הטווח הנתון. ועבור כל מרווח בדקתי האם פוטנציאלי להכיל שורש, אם אכן פוטנציאלי אני מחפש באמצעות שיטות נומריות שלמדנו בכיתה את השורש במרווח. עבור כל שורש שמצאתי אני בודק האם תקין, ואם אכן תקין מכניס למערך המכיל את הפתרון המבוקש.

Assignment 3 (35pt):

Implement a function **Assignment3.integrate(...)** and **Assignment3.areabetween(..)** following the pydoc instructions and answer two theoretical questions.

(5pt) Assignment3.integrate(...) receives a function f , a range, and a number of points n .

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 2 minutes for $n=100$.

(5pt) Question 3.1: Explain the key points in your implementation of **Assignment3.integrate(...)**.

מימשי את Composite Simpson's rule, אותו אלגוריתם שלמדנו בכיתה, האלגוריתם משתמש בח נקודות כנדרש, וכאשר n אי זוגי האלגוריתם ישתמש ב $n-1$ נקודות.

(10pt) Assignment3.areabetween(..) receives two functions f_1, f_2 .

It must return the area between f_1, f_2 .

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 $x \in [1, 100]$.

Note: there is no such thing as negative "area".

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

(5pt) Question 3.2: Explain the key points in your implementation of **Assignment3. areabetween (...)**.

בשאלה זו אני משתמש ב intersections (שאלה 2) על מנת למצוא את החיתוכים של שני הפונקציות בטווח הנתון. לאחר מכן אני בונה פונקציה חדשה של הפרש בערך מוחלט של שתי הפונקציות הנתונות, ומשתמש בפונקציה integrate כדי לחשב את השטח בטווח שבין שתי נקודות החיתוך הקיצוניות ביותר. משום שמדובר בשטח בין שתי פונקציות התוצאה תמיד תהיה מספר חיובי. אם לא נמצאו לפחות 2 חיתוכים בין הפונקציות, אני מחזיר np.nan כנדרש.

(5pt) Question 3.3: Explain why is the function $2^{\frac{1}{x^2}} * \sin\left(\frac{1}{x}\right)$ is difficult for numeric integration with equally spaced points?

הקושי נובע מהתנודות הגבוהות של הפונקציה, בפרט בטווח המבוקש, המרחק בין כל תנודה אינו קבוע, ולכן נוצרת בעיה בחישוב שטח. על מנת לחשב את השטח נצטרך לחלק את הגרף לחלקים מאוד קטנים, ובחישוב של שטחים אלו וגם לאחר מכן, בחיבורם על מנת לחשב את האינטגרל תיווצר שגיאה נגררת משמעותית בעקבות יכולות הדיוק של המחשב.

(5pt) Question 3.4: What is the maximal integration error of the $2^{\frac{1}{x^2}} * \sin\left(\frac{1}{x}\right)$ in the range $[0.1, 10]$? Explain.

ע"פ השגיאה ב Composite Simpson's rule $-\frac{h^4}{180}(b-a)f^4(\xi)$, כלומר $h = \frac{b-a}{n}$ $Error = -\frac{(b-a)^4}{n^4 * 180} f^4(\xi)$ ניתן לראות שככל ש n גדל השגיאה הולכת וקטנה, לכן נרצה n כמה שיותר קטן. כלומר $n=2$. זהו ה- n המינימלי לחישוב אינטגרל.

$$\frac{(b-a)^4}{n^4 * 180} * \max_{\xi \in [a,b]} |f^4(\xi)|$$

--

Assignment 4 (15pt)

(10pt) Implement the function **Assignment4.fit(...)** following the pydoc instructions.

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

Assignment4.fit should return a function g fitting the data sampled from the noisy function. Use least squares fitting such that g will exactly match the clean (not noisy) version of the given function.

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

You have no constraints on the number of invocations of the noisy function but the maximal running time is limited. Invocation of f may take some time but will never take longer than 0.5 sec.

Additional parameter to **Assignment4.fit** is `maxtime` representing the maximum allowed runtime of the function, if the function will execute more than the given amount of time, the execution will not contribute to the grade causing significant deduction. You should consider the risk of failure vs gains in accuracy when you get close to the time limit.

Grading policy: the grade is affected by the error between g (that you return) and the clean (not noisy) version of the given function, much like in Assignment1. 60% of the test cases for grading will be polynomials with degree up to 3, with the correct degree specified by d . 30% will be polynomials of degrees 4-12, with the correct degree specified by d . 10% will be non-polynomials with random $d \in [1..12]$.

(5pt) **Question 4.1:** Explain the key points in your implementation.

בשאלה זו אני משתמש בשיטה Least Squares שנלמדה בכיתה, תחילה אני דוגם נקודות במרווחים שווים על הפונקציה. מכיוון שהפונקציה שאנו מקבלים רועשת (רעש המתפלג נורמלי), אני מבצע מיצוע של ערכי ה- y השונים המתקבלים מהפונקציה הנתונה, ובחישוב לאחר מכן אני משתמש בוקטור ה- y אשר התקבל מפעולת המיצוע. שיטה זו מתבססת גם על מעלת הפונקציה שאותה אנו רוצים לקרב, לכן אני נעזר בפעולה `numpy.diff` על מנת לחזות את המעלה הנכונה אשר באמצעותה אקרב את הפונקציה. לפתרון המשוואה של Least Squares אני משתמש בשיטת ה-PLU Decomposition שנלמדה בכיתה להיפוך מטריצה. כל פעולה בפונקציה מתחשבת ב-`maxtime` הנתון על מנת לא לחרוג ממנו.

Assignment 5 (30pt).

(10pt) Implement the function **Assignment5.area(...)** following the pydoc instructions.

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 n . You can assume that 10,000 points are sufficient to precisely compute the shape area. Your challenge is stopping earlier than that according to the desired error in order to save running time.

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

(5pt) Question 5.1: Explain the key points in your implementation.

השתמשתי ב Shoelace theorem, נוסחה לחישוב שטח של פוליגון על בסיס קודקודיו. מספר הנקודות נקבע לפי \maxerr הנתון בשאלה.

(10pt) Implement the function **Assignment5.fit_shape(...)** and the class **MyShape** following the pydoc instructions.

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 should 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). When calling the function **AbstractShape.area()**, the return value should be the area of the shape. You may use your solution to **Assignment5.area** to implement the area function.

Additional parameter to **Assignment5.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 execution will be halted.

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.

(5pt) Question 4B.2: Explain the key points in your implementation.

תחילה אני דוגם 10000 נקודות, מכיוון שהנקודות רועשות אני משתמש באלגוריתם DBScan אשר מתחשב בצפיפות הנקודות ו"מנקה" לי מספר משמעותי של נקודות רועשות. לאחר מכן כאשר הנקודות שלי פחות רועשות אני מבצע את האלגוריתם KMeans ולוקח את מרכזי הקלסטרים שאותו KMeans מייצר לי. בשלב זה אני משתמש ב ConvexHull כדי ליצור את הצורה המבוקשת, ולחשוב השטח אני משתמש בפונקציה ConvexHull.volume אשר מחזירה את נפח הצורה, בצורה דו-מימדית מחזירה שטח.