Numerical Analysis

Final task

Submission date: 15/2/2023 23:59

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.

Comments within the Python templates of the assignment code are an integral part of the assignment instructions.

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 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 $[-\infty, +\infty]$.

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 (14pt):

Check comments in Assignment1.py.

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.

<u>Grading policy (10pt):</u>

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

Question 1.1: Explain the key points in your implementation (4pt).

קוד זה יוצר עקומת Bezier שמתקרבת לפונקציה נתונה f(x).
הקוד יוצר תחילה קבוצה של n נקודות המדגימות את הפונקציה f על פני המרווח [a, b].
לאחר מכן, הקוד מחשב את המקדמים A ו-B של עקומות Bezier המחברים כל זוג נקודות עוקבות. הדבר נעשה על ידי פתרון מערכת משוואות ליניאריות באמצעות אלגוריתם תומס, אשר מיושם בפונקציית TDMAsolver.
הקוד יוצר קבוצה של עקומות Bezier ומכניס אותם למילון שהkey שלו מוגדר כנקודה (x, y) וה-value שלו מוגדר כעקומת Bezier מותאמת אליו.
הפונקציה g מוגדרת להחזיר את ערך ה-y של עקומת Bezier בערך x נתון על ידי נרמול x ומציאת עקומת ה Bezier המתאימה והערכתה באמצעות אינטרפולציה לינארית.

Assignment 2 (14pt):

Check comments in Assignment2.py.

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

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:

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

<u>Grading policy (10pt):</u> 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 (4pt).

הפונקציה intersections מקבלת את f1 וf2 וצריכה למצוא את נקודות החיתוך ביניהם. אותה פעולה שווה למציאת השורשים של פונקציית החיתוך שלהם (g). תחילה הקוד מנסה להשתמש בשיטת ב--newton raphson כדי למציאת השורשים של g, אבל אם השיטה לא מתכנסת תוך 15 איטרציות או שמגיעה לנגזרת השווה לאפס כלומר מגיעה לחישוב אותו אלגוריתם לא יכול לחשב, היא עוברת לשיטת הbisection בניסיון למצוא את השורש. במידה והתקבלו שגיאות או שאותם אלגוריתמים לא הצליחו למצוא שורש יחזור None. לבסוף, הקוד מסנן תוצאות זהות ומחזיר את רשימת התוצאות המכילה את השורשים של g שנמצאו, כלומר את נקודות החיתוך של f1 וf2.

Assignment 3 (36pt):

Check comments in Assignment3.py.

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.

<u>Grading policy (10pt):</u> 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(…). (4pt)

הפונקציה משתמשת בשלוש שיטות אינטגרציה מספריות: mid-point rule, trapezoidal rule and Simpson's rule. n הוא מספר מרווחי המשנה שאליהם יש לחלק את המרווח. כאשר n=1, נוכל לחשב את התוצאה על ידי נקודת האמצע של הקטע בעזרת mid-point rule, כאשר n=2, נוכל לחשב שטח טרפז בין הנקודות של התחום בעזרת trapezoidal rule ובשאר המקרים נשתמש ב-Simpson's role . הפונקציה תחזיר את השטח ע"י אחת מהשיטות.

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".

<u>Grading policy (10pt):</u> 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 (…) (4pt).

תחילה נייבא רשימה של נקודות החיתוך בין 2 הפונקציות (ממשימה 2), ממוינות בסדר עולה. הקוד בודק אם אורך רשימה זו קטן מ-2. אם זה המקרה, הפונקציה מחזירה None. אחרת, מוגדרת פונקציית g, פונקציית החיסור של f1 ו-f2 בערך מוחלט. נעבור על פני רשימת נקודות החיתוך. עבור כל איטרציה, המשתנים x1 ו-x2 מוגדרים כאיבר הנוכחי והבא ברשימה, בהתאמה. לאחר מכן, אנו קוראים לפונקציה מסעיף א' אשר מחשבת את האינטגרל של הפונקציה g על פני המרווח [x1, x2] בעזרת simpson. התוצאה מתווספת בכל איטרציה למשתנה סכימה בשם result. לבסוף, הפונקציה מחזירה את השטח בין שתי הפונקציות f1 וf2.

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? (4pt)

פונקציה זו קשה לחישוב ע"י חלוקה לנקודות מרווחות שוות כיוון שיש לה סינגולריות ב-x = 0. זה אומר שהפונקציה לא מוגדרת או שהערך שלה הופך לאינסוף בנקודות אלו. בנוסף, בדרך כלל קל יותר להשתמש בנקודות מרווח שוות בחישוב אינטגרציה מספרית, אם אין לפונקציה שינויים חדים בערכה. עם זאת, אם הפונקציה אינה רציפה אחידה, ייתכן שהנקודות המרווחות באופן שווה לא יספקו תוצאות מדויקות עבור האינטגרציה.

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. (4pt)

על מנת למצוא את השגיאה המינימלית כלומר, השגיאה בעלת הדיוק הגבוה ביותר, נחלק לכמה שיותר חלוקות. לכן על מנת למצוא את השגיאה המקסימלית נחלק לכמה שפחות נקודות. ה-n המגדיר את מספר החלוקות נמצא במכנה, מכן שתחום הגדרתו הוא גדול מ0 (אך גם זוגי מכיוון שסימפסון עובד עם זוגיים) ולכן הn המינימלי שנוכל לקחת הוא n=2.

נשתמש ב-simpson's rule ונציב בנוסחה הבאה: $\frac{(b-a)^5*(\max(f^4(x)))}{180*(n^4)}$ את הערכים הבאים: נגזרת רביעית של פונקציה, a = 0.1, b=10, n=1. ניתן לראות ש-$f^4(x)$ נמצא במונה ולכן על מנת להגדיל את הביטוי נרצה שערכו יהיה מקסימלי כלומר, שערכו יהיה אינסוף. מכאן, נוכל להסיק שהשגיאה המקסימלית שואפת לאינסוף.

Assignment 4 (14pt)

Check comments in Assignment4.py.

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

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

Assignment4A.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 constrains on the number of invocation of the noisy function but the maximal running time is limited. 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 grade will be significantly reduced.

<u>Grading policy (10pt):</u> 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. 65% 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. 5% will be non-polynomials

Question 4.1: Explain the key points in your implementation. (4pt)

תחילה חישבתי את זמן הריצה של קבלת ערך ה-y מהפונקציה המתקבלת בקלט עם ערך x השווה לתחילת התחום (a), על מנת להעריך את זמן הריצה הכולל שייקח בחישוב של כל ערכי y, כאשר הפונקציה מקבלת ערכי x ברווחים שווים של d + 1 במרווח [a, b]. בהתחשבות בזמן שהתקבל (מבלי שיחרוג מהזמן המקסימלי), ביצעתי בכל בדיקה של x מספר קריאות לפונקציה f והכנסתי לרשימה. יתקבלו ערכי y שונים מכיוון שהנקודות רועשות ולכן על מנת לדייק יותר את ערך ה-y אותו נרצה לבחור, לקחתי את הממוצע שלהם והכנסתי למערך. לאחר התהליך התקבל מערך של מערכים (בעלי איבר אחד, ערך הy הממוצע) המוגדר כ-b_list.
יצרתי את מטריצה A, כאשר כל שורה מייצגת את המקדמים של הפולינום המוערכים בערך x ספציפי, עבור כל רשימת האיקסים. נמצא פתרון לינארי בין המטריצה A לווקטור b בעזרת שיטות אלגבריות ע"י transpose ו-invert מספריית numpy. שמרתי במשתנה את המקדמים של הפולינום (הפתרון שהתקבל לאחר החישובים).
החזרתי פונקציה המייצגת את הקירוב הפולינומי של הפונקציה f על פני המרווח [a, b].

Assignment 5 (27pt).

Check comments in Assignment5.py.

Implement the function **Assignment5.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 n. 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.

<u>Grading policy (9pt):</u> the grade is affected by your running time.

Question 4B.1: Explain the key points in your implementation. (4pt)

קוד זה מחשב את השטח של צורה עם קו מתאר נתון. ה-contour הוא פונקציה המחזירה רשימה של נקודות שמגדירות את הצורה. נחשב את שטח הצורה בעזרת Shoelace method. נחזיר את השטח של הפוליגון.

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.

<u>Grading policy (10pt):</u> 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. (4pt)

השיטה מתחילה ביצירת קבוצה של נקודות קווי מתאר, נעצור לאחר 100 איטרציות או עד שהגענו למגבלת הזמן המקסימלית שצוינה. לאחר מכן ניעזר ב-K- mean clustering כדי לזהות את מרכזי המקבץ של נקודות המתאר ונחשב את הזוויות שלהם ביחס לממוצע של כל המרכזים. הזוויות משמשות למיון המרכזים (בסדר נגד כיוון השעון). ניצור אובייקט MyShape כאשר נשלח לו את המדדים הממוינים של המרכזים כקלט. מופע זה מוחזר לאחר מכן כצורה המותאמת.