**Numerical Analysis**

**Final task**

**Name: Etay arie Lorberboym**

**ID: 314977596**

Submission date: 19/2/2021 15:30am

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.

|  |
| --- |
| את פונקציית האינטרפולציה ביצעתי על ידי מימוש של עקומות ביזייר בעזרת הנקודות הנתונות שקיבלתי מהדגימות.  יצרתי n דגימות מהפונקציה f שקיבלתי בנתונים ,וחילקתי את המקטע בין a ל- b לn נקודות שוות במרחקן, שעליהן אבצע את האינטרפולציה.  בניתי בעזרת הנוסחה שלמדנו בשיעור ועם הפונקציה הנתונה של bezier3 עקום ביזייר מדרגה 3 בין כל 2 נקודות והוספתי לווקטור שמכיל את כל הפונקציות, בנוסף בניתי וקטור שנותן לי את המקטע של כל עקום ביזייר.  השתמשתי בנוסחה אשר לפיה אנחנו צריכים לעקום הראשון 4 נקודות ואחרי זה לכל עקום עוד 2 נקודות מכיוון שיש את אילוצי הנגזרות בנקודות החיבור.  לאחר מכן החזרתי פונקציה שכשהיא מקבלת אינפוט היא רצה בלולאה על הווקטור של המקטעים ובודקת בין איזה מקטע נמצא האינפוט איקס שקיבלה הפונקציה g וכשהיא מוצאת היא הולכת לקטור הפונקציות ומחזירה את הערך t המומר מהערך איקס שהתקבל לפונקציה של המקטע שנמצא.  בנוסף, מימשתי גם piecewise Bezier interpolation שבבדיקות שביצעתי נתן בבדיקות של פולינומים ממעלות גבוהות תוצאות פחות טובות מהשיטה הראשונה. כאשר בחרתי פולינומים ממעלות נמוכות השיטה השנייה פעלה יותר טוב, אך לא הצלחתי למצוא threshold מתאים לכל המצבים האפשריים ולכן בחרתי ללכת עם הדרך הראשונה. לפונקציה השנייה קראתי interpolation2. |

**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.

|  |
| --- |
| חיפשתי שיטה שתעבוד בטוח ומהיר, אז מצאתי שילוב של השיטות למציאת שורשים ניוטון רפסון ושיטת החציה.  חילקתי לשני מצבים, הראשון כאשר נתונות לי 2 פונקציות פולינומים של נאמפיי והשני כאשר נתונות לי שתי פונקציות למבדה.  בשיטה הראשונה, יצרתי פונקציה חדשה שהיא פונקציה 1 פחות פונקציה 2 והשתמשתי בשיטה שבניתי על מנת למצוא את השורשים של הפונקציה.  ראשית, אני בודק אם הinput שלa ו-b תקין.  לאחר מכן, אני מחלק את המקטע בין a ל – b לאורך המקטע כפול 100 בעזרת הפונקציה linspace בכדי לקבל חלוקות שוות של נקודות בקטע ואני בודק בין כל 2 נקודות האם יש שורש על ידי שימוש בדגימת הפונקציה והנגזרת בשני החלקים.  אם קיבלתי שהתנאים לכך שיש שורש בין 2 הנקודות מתקיימים אז אני מריץ את השיטה של newt-safe.  שיטת newt-safe עובדת כך שמתחילים בצעד של שיטת bisection ואז מתקדמים בצעדים של שיטת Newton -Raphson. כאשר לא ניתן לבצע צעד בשיטת Newton-Raphson אז אני מבצע צעד בשיטת bisection.  לאחר כל צעד, אני בודק אם התקדמתי מספיק לשורש הפונקציה בעזרת בדיקה מול הmaxerr.  אם התקדמתי מספיק אז אני מוסיף את השורש לרשימה של שורשים.  בחרתי לבצע 10 איטרציות של השימוש בשיטה למציאת השורש, שכן השיטה של ניוטון רפסון מתכנסת מהר מאוד (בערך 3-5 איטרציות) ובבדיקה פרמטר זה נתן תוצאה טובה.  אם לא נמצא שורש ב10 איטרציות שביצעתי אז אני ממשיך לקטע הבא.  השיטה השנייה של הלמבדות עובדות אותו הדבר פשוט חישבתי בעזרת פונקציה את הנגזרת בנקודה וביצעתי את החיסור בין הפונקציות בצורה שונה על מנת שיפעל (כי לא ניתן לחסר פונקציות למבדה).  כל השאר עובד אותו הדבר. |

**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(…).

|  |
| --- |
| בפונקציה integrate ראשית כל מימשתי את שתי השיטות שלמדנו בכיתה לחישוב אינטגרלים, שיטת traperzoid ושיטת simpson עם n=2.  למדנו בכיתה שברוב הפעמים שיטת סימפסון תחזיר ערך מדויק יותר מכיוון שהיא מקרבת פולינומים של לגראנז ממעלה שניה בשונה משיטת טרפזויד שמקרבת בעזרת פולינומים ממעלה ראשונה לפונקציה שעושים לה את האינטגרל, אך ניתן להשתמש בה רק אם מספר הקטעים שאנחנו משתמשים בהן הוא זוגי.  לכן, אם n אי זוגי(מספר הנקודות) אני אשתמש בשיטת סימפסון ואם זוגי אשתמש בשיטת טרפזויד.  בנוסף, קראתי על שיטות של סימפסון ממעלות גבוהות יותר ומימשתי אחת מהן על מנת לפתור את הhard-case אך היא לא צלחה – simpson\_rule2. |

**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 (…).

|  |
| --- |
| ראשית השתמשתי בשיטה של מציאת השורשים שמימשתי בassignment2 על מנת למצוא את הנקודות חיתוך בין הפונקציות הנתונות, לאחר מכן אני בודק את השטח בין כל שני חיתוכים על ידי שימוש בפונקציה של האינטגרל שמימשתי בפונקציה integrate.  בין כל 2 נקודות חיתוך דגמתי את הנקודה אשר נמצאת בניהן על מנת לבדוק איזה ערך של פונקציה גבוהה יותר בכדי לחשב את האינטגרל בתחום.  סכמתי את כל האינטגרליים שחישבתי וזוהי התוצאה הסופית. |

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

|  |
| --- |
| יהיה קשה לבצע אינטגרציה נומרית בשיטות שלמדנו לפונקציה הנתונה עם נקודות במרווחים שונים מכיוון, שהשתמשנו בשיטה של Simpson Rule קירבנו את האינטגרל בעזרת פולינומים ממעלה שנייה, ומכיוון שפולינומים ממעלה שנייה יכולים לשנות כיוון עד פעם אחת בלבד יהיה בלתי אפשרי לקרב פונקציה בעלת הרבה שינויי כיוון במרווחים קטנים ולא שווים (strong oscillation). בפונקציה הנתונה יש בתחום מאוד קטן עם המון שינוי כיוון ולכן השיטה שלנו לא מצליחה לקרב אליה פולינום ממעלה שניה. בחקירה בסיסית שביצעתי ניתן לשים לב שבטווח הערכי איקס מינוס חצי לחצי הפונקציה עולה ויורדת הרבה פעמים.  פתרונות אינטגרציה לפונקציה הזו הם או להשתמש בsimpson rule עם דרגת פולינום מספקת לביצוע שינויי הכיוון של הפונקציה או להשתמש בopen-Newton-cote formula אשר בעזרתה אפשר להשתמש ביותר נקודות פנימיות בתוך המקטע וכך להתאים פולינומים ממעלות נמוכות יותר.  לצערי לא הצלחתי לממש את הפתרונות האלה בקוד. |

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

|  |
| --- |
| ניסיתי לחשב את השגיאה על ידי הנוסחה של השגיאה בשימוש בשיטת סימפסון שלמדנו אך חישוב הנגזרת הרביעית והערך המקסימלי שלה מאוד מקשה על הדרך, לכן החלטתי לחשב אינטגרל אחד בעזרת שיטת טרפזויד אשר מחושב בין שתי הקצוות ולחסר מערך האינטגרציה האמיתי שקיבלתי בטסטים.  ערכי f בשתי הקצוות הם:      = 9.9/2\*( |
|  |
|  |

**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.

|  |
| --- |
| ראשית, מימשתי את שיטת least square fitting אשר למדנו בכיתה.  את השיטה מימשתי כך שבהתחלה בניתי פונקציה לקבלת שני וקטורים של נקודות x,y לפי הנתונים שקיבלתי.  לאחר מכן, בניתי שתי שיטות, הראשונה לווקטור הפתרונות של הנוסחה והשנייה למטריצת המקדמים.  פתרתי את המשוואה על ידי מציאת מטריצה הופכית והכפלתה בווקטור הפתרונות משמאל, מה שהחזיר לי את התשובות למקדמים של הפונקציה המקורבת. מווקטור המקדמים שקיבלתי בניתי פונקציית פולינום.  בחרתי בהתחלה לדגום 100 נקודות ואז לרוץ על התהליך של least squere fitting כל עוד זמן הריצה ועוד 0.2 שניות קטן מmaxtime. בכל איטרציה הגדלתי את מספר הדגימות ב200 וביצעתי שוב על מנת לקבל תוצאה מדויקת יותר.  כאשר התקרבתי לmaxtime החזרתי את התשובה האחרונה אותה קיבלתי.  שמתי נקודות עצירה בפונקציה לאחר כל תהליך בשיטת הfit על מנת שלא לגרוע מהmaxtime הנתון. |

**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.

|  |
| --- |
| מכיוון שאנחנו מקבלים נקודות על הצורה מהקונטור בסדר שווה אז אני מבצע אינטגרציה בכיוון הנקודות כך שבין כל 2 נקודות אני סוכם את השטח.  מכיוון שהנקודות הן בסוג של במעגל אז יש שטח שלילי שמתקבל כאשר משנים כיוון ואז הוא מצטמצם עם השטח שהתקבל לפני כן בנקודות.  אני מתחיל בדגימה של 10 נקודות ובודק מהו השטח, לאחר מכן רץ בלולאה ומגדיל את נקודות הדגימה ב2 בכל איטרציה.  כל עוד הrelative error של פערי הדגימות בכל איטרציה גדול מהmaxerr שקיבלתי כנתון אני ממשיך להגדיל את מספר הדגימות עד שאני מגיע לrelative error מספיק קטן.  מכיוון שקיבלתי relative error מספיק קטן בין הדגימות, ואני יודע שזוהי סדרה קטנה אז בין התוצאה שהחזרתי לתוצאה האמיתית הrelative error יהיה קטן יותר מהmaxerr והתוצאה תהיה מספיק טובה. |

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.

|  |
| --- |
| שיטת הפתרון היא לסדר את הנקודות שאנחנו מקבלים מהפונקציה sample על ידי סידור בעזרת מרחק אוקלידי בין נקודות (או לחשב פשוט את הערך איקס הכי קרוב, אבל יכול להיות שיהיו שגיאות) ואז כאשר הצלחתי לסדר את הנקודות בסדר מסוים לבצע least square fit בעזרת השיטה של Bezier שלמדנו. לאחר מכן, לבצע אינטגרציה נומרית לפונקציה שקיבלנו. |