Numerical Analysis

Final task – Shahar Oded, 208388918

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).

במטלה זו בחרתי לממש שיטה מעט שונה של אינטרפולציה בחלקים על מנת לשפר את הדיוק שמתקבל ע"י אינטרפולציה בחלקים בשיטת בזייר. בשיטה שלי, דגמתי את f כמספר הפעמים המותר, וחילקתי את רשימת הנקודות לקבוצות של 7 נקודות, כאשר בין כל קבוצה לקבוצה יש חפיפה של 2 הנקודות הקיצוניות בצדדים, לדוגמה: [[p1,p2,p3,p4,p5,p6,p7][p3,p4,p5,p6,p7,p8,p9]...] . כל אחת מתתי הקבוצות האלו ייצגה לי spline עבורו יצרתי פונקציית אינטרפולציה נפרדת, אשר נשמרה במילון מתאים. פונקציית האינטרפולציה לכל מקטע נוצרה בשיטת lagrange, כאשר עבור מספר קבוע של נקודות (7), הוא פועל בזמן ריצה לינארי, ולכן אני עומד במגבלת ה(o(n. הסיבה שיצרתי את החפיפה בין תתי הקבוצות היא שידוע לנו שלגראנז' זו שיטת אינטרפולציה מדויקת יותר במרכז הקטע, ומתבדרת בקצוות (בהרצאה למדנו צ'בישב כדי לטפל בזה, אך במספר קבוע וקטן של נקודות זה לא רלוונטי, ותכף אסביר גם למה לא מתאים בשיטה שלי). מכיוון שכך מתנהגת אינטרפולציית לגראנז', הפונקציה שאחזיר עבור קלט x תהיה הפונקציה שבניתי במילון עבורה מתקיים $p3 < x < p5$ ז"א – אני משתמש אך ורק במרכז הקטע, המדויק יותר (אם הייתי מפעיל צ'בישב, הייתי מדייק את הקצוות על חשבון המרכז). יש לציין שטיפלתי בנפרד באינטרפולציה לקצוות על מנת שלא יהיה תחום לא מטופל של x. לבסוף, לאחר בניית המילון, הגדרתי פונקציה המקבלת x כקלט (לשים לב, לא היה פה צורך בתרגום x לt יחסי), מוצאת את הspline אליו x שייך באמצעות חיפוש בינארי, ומחזירה את ערך הy המתאים, תחת ההנחה שבדיוק בקטע המבוקש הדיוק מאוד מאוד גבוה, וביתר התחום יש התבדרות משמעותית (שלא ממש מעניינת אותי).
פונקציית interpolate שלי מחזירה בreturn את הפונקציה שהוגדרה באמצעות המילון.
בוצעו בדיקות של מספר הנקודות האופטימאלי עבור הדיוק באינטרפולציה, כרגע נבחר המספר 7 אך נבדקו המספרים 5,7,9,11 למול הרצות הleaderboard.

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).

ראשית, על מנת למצוא חיתוכים בין 2 הפונקציות יצרתי פונקציית חיתוך בינהן (חיסור שלהן אחת מהשנייה).
במשימה זו ניסיתי לעשות את הפתרון שיחזיר הכי הרבה שורשים בזמן הקצר ביותר. לכן, הגדרתי $\mathrm{h} = \frac{1}{50}$
בתור גודל הצעד שלי. בכל איטרציה אני מתקדם על ציר הx את המרחק h, ובתחום זה בודק האם אני מצליח
למצוא שורש (הכלוא בתוך התחום, כאשר אם התבדרתי החוצה – אצא מהאיטרציה). הבדיקה שלי מנסה קודם
כל להריץ ניוטון רפסון, ולאחר 9 איטרציות בהן לא נמצא שורש עושה שימוש במשפט ערך הביניים , ובמידת
ומצאה שקיים שורש כלוא – נכנסת עם שיטת הbisection. על מנת למנוע שגיאות תחום הגדרה עבור ניוטון
מימשתי try-except.
בכניסה לכל מקטע אני בודק האם במקרה קצוות המקטע מקיימים לי קרבה מספקת לשורש. לאחר בדיקה זו
על המקטע אני נכנס למקטע עם הפונקציה המתוארת. אם לא מצאתי שורש – עובר לאיטרציה הבאה (דילוג
של h על ציר הx). אם כן מצאתי, אני מכניס אותו לרשימה X. במימושים קודמים גם עשיתי בדיקה האם השורש
הזה כבר נמצא ברשימה עד כדי maxerr,אך לאחר מספר leaderboards ראיתי שזה רק פוגע לי בזמן הריצה
ולא משפיע על הניקוד, ולכן הורדתי את הבדיקה.
חשוב לומר שאם מצאתי שורש בתחום $\text{left} \leq root \leq right$, הצעד הבא שלי יהיה $root + h$ במקום
$\mathrm{right} + h$

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)

לאחר בדיקות רבות של כל שיטות האיטגרציה שהוזכרו עבור קטעים פתוחים וסגורים, מצאתי כי השיטה המדויקת ביותר שהצלחתי לממש היא לפי אלגוריתם Gauss Legendre quadrature.האלגוריתם, שלא נלמד בכיתה ואת צורת המימוש שלו למדתי באינטרנט, מחזיר ערכי x,w אופטימאליים בכל מקטע, כאשר הגדרתי לו (מטעמי זמן ריצה וטריידאוף בין זמן לדיוק שיחזיר 2 נקודות בכל תת קטע). לאחר מכן, מחשב השטח לכל תת קטע, באמצעות הנקודות האופטימאליות שחושבו באופן רקורסיבי, וסוכם את שטחי כל תתי הקטעים. הגדרתי לאלגוריתם שיחלק השטח למספר תתי הקטעים המירבי שהוא יכול, תוך שבכל קטע הוא משתמש ב2 נקודות, ועדיין לא חורג ממגבלת n הנקודות בתרגיל, אז במקרה הזה מספר הקטעים בתחום האינטגרציה הוא $\frac{n}{2}$.
תוצאת האינטגרציה עוברת המרה לfloat32.

חשוב לציין שבמקביל מימשתי אלגוריתם סימפסון, באופן המפורט מטה, על מנת להשתמש בו בareabetween. הסיבה לכך היא שהאלגוריתם של גאוס צורך זמן ריצה משמעותית יותר גבוה, ובעוד שבintegrate זמן הריצה לא משפיע לנועל הציון הסופי, בareabetween הוא משמעותי.

את סימפסון מימשתי באופן הבא:
אני שולף n נקודות מהקטע הנתון לי בפיזור שווה, ומחלק את התחום למקטעים כך שבכל קטע יהיו בדיוק 3 נקודות. תוך שימוש במערכים מבצע בכל תא את אלגוריתם סימפסון, ולבסוף סוכם את תוצאת האינטגרציה. התוצאה מוכפלת בh/3 בהתאם לאלגוריתם. הערך המוחזר עובר המרה לfloat32 בהתאם לדרישה.

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).

על מנת לייעל ככל הניתן את הפונקציה הזו, מימשתי ראשית פונקציית מציאת שורשים מותאמת (מהירה יותר, מעט פחות מדויקת בשורשים מכיוון שרצה על פחות תתי מקטעים, וללא תפיסת שגיאות. שגיאות תחום הגדרה נתפסו ע"י יציאה מתחום הקטע לפני הפעלת הפונקציה) ופונקציית אינטגרציה לפי סימפסון. לאחר מכן נכנסים לפונקציה: הפונקציה מקבלת את 2 הפונקציות כקלט, מוצאת את כל החיתוכים בינהן באמצעות פונקציית השורשים, ומעבירה את פונקציית החיתוך בתחום שבין כל שני שורשים לפונקציית האינטגרציה. חשוב לציין, פונקציית השורשים שלי עלולה לשכפל שורשים (במטרה לחסוך זמן ריצה) ולכן הכנסתי בדירה נוספת שלא לחשב שטח עבור 2 שורשים קרובים מידי עד כדי פעמיים השגיאה המקסימאלית המוגדרת בתרגיל 2. שגיאה זו היא התחום הגדול ביותר בו עלולים להיות 2 שורשים שהם בעצם זהים ביחס לשורש אמיתי כלשהו. כך בין כל 2 שורשים מתקבל השטח הכלוא בין הפונקציות. השטח נסכם בכל איטרציה (בערכו המוחלט, אין כזה דבר שטח שלילי) ולבסוף מוחזר לאחר שעבר המרה ל float32

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)

בתחום שבין [-0.35,0.35] לפונקציה יש (אני מאמין) אינסוף נקודות קיצון, ובאופן כללי השיפוע שלה בתחום [-1,1] הוא תלול מאוד מאוד, והפונקציה שואפת לקיצון בערכים אינסופיים בכל פעם. בפועל, תוך שימוש בנקודות המפוזרות על מקטע מסוים באופן שווה אנו מקצים מעט נקודות לכל איזור, והערכים שנקבל מ f יהיו שונים באופן קיצוני זה מזה. לכן, האינטגרציה ממעלה 2 שעושה סימפסון לפונקציה בכל תחום היא דיי בסיסית ומאוד לא מדויקת (ולא נשענת אפילו על ערכים מקורבים של מקסימום הפונקציה). לכן, בהיתקלות עם פונקציות בעלות שיפועים גדולים באופן משמעותי, וזו בפרט, אין יכולת להעריך באופן מדויק את שטח הפונקציה, מכיוון שניסיון האינטרפולציה עבורה בכל מקטע "מקולקל". אם נרצה להתחיל ולנסות להתמודד עם זה, נצטרך לקחת הרבה יותר נקודות בתוך התחום המתבדר, אבל קשה להעריך עד כמה יותר, יכול להיות עד כדי $n \to \infty$ כדי לקבל תוצאה מקורבת היטב.
חשוב לציין ששגיאת סימפסון היא ביטוי מוגדר וחסום כפי שנלמד בכיתה.

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)

הפונקציה "מתפוצצת" עד אינסוף בתחום זה, כ3 פעמים (יש לציין שיש לה אינסוף נקודות קיצון ככל ששואפים לx=0, מדובר אך ורק על התחום המבוקש). מכיוון שלא ניתן להגדיר $f(x) = \infty$ כחלק מהפונקציה עבור האינטגרציה, סימפסון מוגבל לערכים סופיים בלבד, ולכן אני מאמין שטעות האינטגרציה המקסימאלית שואפת לאינסוף.

חשוב לציין שלפי נוסחת סימפסון בקטע סגור מדרגה שנייה, הטעות נמדדת ע"י $-\frac{h^5}{90} f^{(4)}(\xi)$, ולכן עלינו לחשב נגזרת מסדר רביעי של הפונקציה עבור נקודה כלשהי בתחום $0.1 \le \xi \le 10$, בפועל, נקבל שגיאה מקסימאלית המוערכת ע"י החסם

$$\left|E_S^{total}\right| \le \frac{h^4 (b-a) f^{(4)}(\tilde{\xi})}{180}$$

השגיאה הכוללת היא מסדר 4

לאחר ניסיון בדיקה במספר מחשבונים, ChatGPT מעריך שהשגיאה היא בסדר גודל של לכל הפחות 10^{10}.

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)

המימוש שלי נשען על שיטת ה*fitting* שנלמדה בכיתה באמצעות בזייר. ראשית דגמתי הנקודות מהפונקציה, כתלות בזמן הריצה המקסימאלי המוקצה לי. יצרתי מהדגימות את ווקטור הנקודות C, המכיל את $|d+1|$ הנקודות האופטימאליות ביותר ליצירת עקומת הבזייר (מדרגה d). את הנקודות האלו מצאתי תוך מימוש המשוואה:

$$C = M^{-1}\left(\mathbf{T}^{\mathrm{T}}\mathbf{T}\right)^{-1}\mathbf{T}^{\mathrm{T}}\mathrm{P}$$

המוצאת את הנקודות האופטימאליות ביותר תוך צמצום השגיאות היחסיות בין תוצאת הקירוב לנקודה הנדגמת (*least square errors*). עם קבלת וקטור הנקודות הפעלתי פונקציית בזייר המייצרת עקומה (יחידה) מדרגה דינאמית (בכיתה למדנו לייצר בזייר מדרגה 3 בלבד, הייתי צריך להתאים את העקומה שלי לדרגות גבוהות יותר. חשוב לציין, בפועל צמצמתי את העקומות שלי לדרגה 10 לכל היותר, לאחר טסטים על ה*grader* שהראו לי שדרגה זו (ומטה), ביחס גם לסיבוכיות ולזמן הריצה שלה, היא האופטימאלית לי (מכיוון שמונעת התאמת יתר של התוצאות לנקודות הרועשות). לאחר יציאת העקומה יצרתי פונקציית מודל המתרגמת ערך *x* מסוים בטווח לערך ה*t* הרלטיבי שלו, ובאמצעותו החזרתי מהעקומה את ערך ה*y* לכל קריאה. פונקציית *fit* שלי החזירה פונקציה המבצעת את התרגום הזה ומחזירה את ערך ה*y*.
אחד האתגרים המשמעותיים במשימה היה מציאת כמות הדגימות האופטימאלית, תחת ההבנה כי נדרשת כמות קטנה ביותר של נקודות, אחרת הטסטים נופלים על זמן ריצה. בסופו של דבר מימשתי דגימה התלויה ב*maxtime* המוקצה לי, כך שאני דוגם סדר גודל של $maxtime * 11$ נקודות בכל בניית מודל.

חשוב לציין, את היפוך המטריצה מימשתי עם המתודה $np.linalg.inv$ כפי שאושר בפורום המטלה. הכנסתי בדיקה נוספת, המבררת האם המטריצה (משיקולים של $floating\ point\ errors$) כעת היא מטריצה שאינה הפיכה (או קרובה מאוד לכך). אם גיליתי שכן, הפעלתי את המתודה $np.linalg.pinv$ המייצרת קירוב להיפוך מטריצה שאינה הפיכה תוך שימוש באלגוריתם $Moore-Penrose\ pseudoinverse$ באופן זה ניסיתי להימנע משגיאות מתמטיות בחלק מהטסטים.

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.

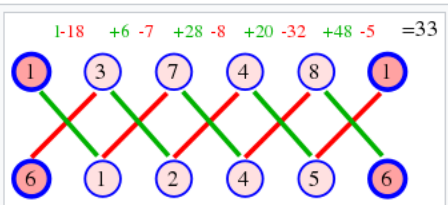
Naturally, the more points you request from the contour the more accurately you can compute the area. Your error will converge to zero for large n. You can assume that 10,000 points are sufficient to precisely compute the shape area. Your challenge is stopping earlier than according to the desired error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

במשימה זו רציתי לחשב שטח פוליגון באופן המדויק ביותר ע"י דגימת נקודות בו. מצאתי כי יש חלוקה ל2 מקרים – פוליגון פשוט ופוליגון מורכב (פוליגון בו הצלעות חותכות זו את זו). את שטח הפוליגון הפשוט מימשתי באופן דיי פשוט - באמצעות שיטת shoelace עליה קראתי. השיטה עושה שימוש בקואורדינטות של כל הנקודות (המסודרות על גבי מעטפת הפוליגון) ובאמצעות הצלבה בין הנקודות (כמו קשירת שרוכים, ומפה השם להבנתי) מחשבת כסכום רץ את השטח בצורה הבאה לכל נקודה מהצורה (x, y):

$$Polygon\ area = \frac{1}{2}\sum_{i=1}^{n-1}(x_i * y_{i+1}) - (y_i * x_{i+1})$$



Horizontal shoelace form for the example.

שיטה זו נמצאה בתור המתאימה ביותר עבור הטסטים עליהם אנו נבחנים, על אף שבמקור מימשתי גם שיטה המסוגלת לזהות ולהחזיר השטח גם לפוליגונים מורכבים (בעלי חיתוכים עצמיים).
ובפונקציית המעטפת קראתי לn נקודות (כ600 בסופו של דבר), והפעלתי עליהן את הפונקציה.

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)

בפונקציה זו המטרה שלי הייתה לדמות באופן מקורב ככל הניתן צורת פוליגון באמצעות נקודותיו, ע"י נרמול הנקודות הרועשות. הנרמול נעשה בכמה שלבים:
ראשית, באמצעות הגנרטור המוגדר באובייקט משכתי נקודות עד להגעה ל0.95 מהזמן המוקצה לי להרצה, אך לא יותר מ$2000 * maxtime$ נקודות. המטרה: למקסם את כמות הנקודות בהן אשתמש אבל גם לא להגזים בזמן הריצה של האלגוריתם.
לאחר מכן מרכזתי את הנקודות סביב נקודת המקור שלהן (ז"א נקודת מרכז המעגל, בהנחה לדוגמה והפוליגון הוא מעגל), ואז חיסרתי מכל נקודה את המרכז על מנת להביא את הנקודות לראשית הצירים.
פעולה זו הייתה חיונית מכיוון שלאחר מכן רציתי להתייחס לכל מקטע בצורה באופן נפרד (ע"י מספר מסוים של נקודות השייכות רק לו). לאחר מרכוז הנקודות מיינתי את המערך לפי הזווית של הנקודות מראשית הצירים (באמצעות תצורה של מספר מרוכב לכל נקודה), וקבעתי מספר קטעים לצורה כתלות במספר הנקודות שיש לי (8 קטעים עבור מעט נקודות, מספר קטעים יחסי עד 500 קטעים כל עוד יש לי מספיק נקודות בכל מקטע). על הנקודות בכל אחד מהקטעים הרכבתי אלגוריתם Boxcar שמטרתו מציאת k נקודות ממוצעות בכל קטע בעל k נקודות (ז"א – לא לאבד נקודות בפעולה הזו, על מנת שפעולת חישוב השטח תהיה מדויקת ככל הניתן).
האלגוריתם משתמש כל פעם בk נקודות, מוצא על פי כל הk ממוצע, ומתקדם ב1. כל הנקודות במקטע מסוים (תחת ההנחה שהוא קטן מאוד) תורמות למציאת ממוצע בינהן ומצמצמות את השגיאה המתפלגת באופן נורמאלי.
את הנקודות שצברתי החזרתי כמערך, אותו שמרתי בתור attribute של myshape.
חוץ מהבנאי של האובייקט היורש שבניתי, הרכבתי גם מתודת area() העושה שימוש במתודה זהה לשל הסעיף הקודם, רק על הנקודות שנשמרו כעת באובייקט שלי.