**האוניברסיטה הפתוחה**

**המחלקה למתמטיקה ולמדעי המחשב**

סמינר באנליזה נומרית

מספר קורס 20369

שימוש בטכניקות אנליזה מרוכבת למטרת קירוב ערכי נגזרות של פונקציות ממשיות

**העבודה הוכנה על-ידי:** יניב שדה, ת"ז 302545546

**בהדרכתה של** **ד"ר** מיריי אביגל

**תאריך ההגשה:** י"ז בכסלו התשע"ה, 28/11/2015

תוכן עניינים

[1. הקדמה 3](#_Toc436510863)

[2. סקירת עבודות קודמות 4](#_Toc436510864)

[3. רקע מתמטי 5](#_Toc436510865)

[3.1. טורי טיילור ולורן 5](#_Toc436510866)

[3.2. אינטגרלי קושי 6](#_Toc436510867)

[4. חישוב נגזרות בשיטת טורי טיילור 7](#_Toc436510868)

[4.1. נגזרת ראשונה 7](#_Toc436510869)

[4.2. נגזרת שנייה 8](#_Toc436510870)

[4.3. נגזרות גבוהות 9](#_Toc436510871)

[5. חישוב נגזרות באמצעות אינטגרלי קושי 11](#_Toc436510872)

[5.1. תיאור השיטה והערכת השגיאה 11](#_Toc436510873)

[5.2. יישום השיטה 12](#_Toc436510874)

[5.3. שימוש נוסף: חישוב ערכי סינגולריות סליקה 13](#_Toc436510875)

[6. השוואת ביצועים נומריים 14](#_Toc436510876)

[6.1. נגזרת ראשונה 14](#_Toc436510877)

[6.2. חישוב ערך בסינגולריות סליקה ("נגזרת אפס") 15](#_Toc436510878)

[6.3. נגזרת שנייה 17](#_Toc436510879)

[6.4. נגזרת חמישית 19](#_Toc436510880)

[6.5. נגזרות גבוהות יותר 22](#_Toc436510881)

[7. דיון ומסקנות 24](#_Toc436510882)

[8. כיוונים להמשך מחקר 25](#_Toc436510883)

[9. ביבליוגרפיה 26](#_Toc436510884)

[10. נספח: קוד התוכנית 27](#_Toc436510885)

[10.1. seminar.h 27](#_Toc436510886)

[10.2. seminar.cpp 29](#_Toc436510887)

1. הקדמה

אחת הבעיות באנליזה נומרית היא חישוב ערכי נגזרות של פונקציה בנקודה. על-אף שכיום ישנן יותר ויותר שפות ושיטות לביצוע חישובים סימבוליים, יישומי גזירה-נומרית עדיין רלוונטיים: בין אם בביטוי מורכב, כי החישוב כבד, או שהכלים אינם מאפשרים טיפול סימבולי.

בעיה נוספת קיימת בבואנו לחשב ערכי פונקציות בנקודות אי-הגדרה סליקות. מכיוון שלרוב מדובר על גבול של "0/0" או "", שגיאת החישוב עלולה להתעצם.

בעבודה זו נעשה שימוש באנליזה מרוכבת לטובת שיפור טכניקות ממשיות קיימות, ונעשה שימוש בטכניקה מרוכבת שאינה נובעת מהכללה של המקרה הממשי. נעיר, שהטכניקות דורשות הרחבה אנליטית של הפונקציה הממשית שברשותנו, אך תחת "הנחות סבירות", למשל כשמדובר בפונקציות הנובעות מבעיות פיסיקליות, נרשה לעצמנו להניח שכך הדבר.

עבודה זו אינה מציגה טכניקה חדשנית, אך סוקרת כמה מהשיטות הקיימות, משווה את ביצועיהן ובוחנת את ישימותן. כמו-כן, ניתוח זמני הריצה של השיטות הוא צד שלא קיבל דגש בספרות העוסקת בנושא, המופיע בעבודה זו.

1. סקירת עבודות קודמות

הרעיון להשתמש באנליזה מרוכבת אינו חדש כלל, וכבר בשנות השישים פרסמו Lyness and Moler את [3] הדן באנליזה מרוכבת של אינטגרלי קושי לחישוב ערכי נגזרות של פונקציות ממשיות הניתנות להרחבה אנליטית. במאמר מציגים כמה שיטות המבוססות על אינטגרלי קושי, מוכיחים התכנסות מעריכית של הקירוב, ודנים בהערכת השגיאה. הם אף מראים שהנוסחה מדויקת עבור פולינומים ממעלה מוגבלת, ומתייחסים לשגיאות עיגול הנובעות מייצוג המספרים במחשב.

המאמר [1] מציג שיטה המתבססת על [3] ומאמרים נוספים, המיישמת חישובים אלה לטובת הערכה בדיוק גבוה של מקדמי פיתוח טורי טיילור של פונקציה, כך שמשם ניתן לקבל את הנגזרות המבוקשות על-ידי גזירת הפולינום המתקבל והצבה. מאמר זה מתאר אלגוריתם ותוכנית FORTRAN שנכתבה לצורך כך, ומלבד תיאוריה נוגע יותר בסוגיות פרקטיות של איזון נכון בין שגיאות קיטוע לשגיאות עיגול, וקריטריונים לזיהוי הגעה לדיוק מספק. הלב המתמטי מאחורי השיטה גם הוא מבוסס על אינטגרלי קושי, והשיטה מבצעת טרנספורם פורייה על נקודות במעגל סביב הנקודה המבוקשת (לקבלת המקדמים המבוקשים מתוך ערכי הפונקציה).

ב-2014, פרסמו Trefethen and Weideman את [5] המכיל סקירה מקיפה מאוד של שימוש בכלל הטרפז באנליזה מרוכבת עבור הערכת אינטגרלים של פונקציות מרוכבות בכמה תצורות בסיסיות אך עיקריות (מעגל במישור המרוכב, קטע מחזורי והישר הממשי). במאמרם הם גם דנים בבחירת גודל צעד וניתוח התכנסות, ומשליכים על יישומים אפשריים של השיטה ובפרט מתייחסים לאינטגרלי קושי והערכת נגזרות.

שיטה אחרת לחישוב נגזרות הוצגה בסוף שנות התשעים על-ידי Squire and Trapp [4], שזהה בבסיסה לטכניקה הממשית לקירוב נגזרות. כלומר, בניגוד למאמרים הקודמים הטכניקה אינה מבוססת על רקע מרוכב עשיר אלא מציעה הרחבה של שיטות ההערכה הממשיות. בניגוד לקודמיו, [4] אינו מציג שיטה לחישוב נגזרת כללית, אך חידושו בטכניקה המבטלת לחלוטין את שגיאות העיגול הקיימות בשיטות הקירוב האחרות שנובעות מהפרשים[[1]](#footnote-1). [2] מציג הרחבה של שיטה זו עבור הערכת נגזרות מסדר שני, ומעיר על השימוש באקסטרפולציית ריצ'רדסון לטובת שיפור הדיוק.

בעבודות המוזכרות, מלבד [1] שבו מוצג גרף של זמן הריצה כתלות ברדיוס האינטגרציה, התמקדות הניתוח הייתה במתמטיקה ובדיון התיאורטי של השיטות בלבד, ללא התייחסות לזמן הריצה שגם הוא גורם משמעותי כאשר מדובר על חישובים רבים, כגון מקרה עם נתונים רבים או סימולציה כבדה.

1. רקע מתמטי
   1. טורי טיילור ולורן

אחת הדרכים להציג פונקציות היא בעזרת טורי חזקות. סוג מסויים של טור חזקות מכונה טור-טיילור, והוא מאפשר להגדיר פונקציות בסביבת נקודה נתונה על-סמך נגזרותיה שם, בתנאי שערכה ונגזרותיה בנקודה מוגדרות. בהינתן פונקציה מתאימה גזירה n+1 פעמים בנקודה x=a, ניתן לפתח את טור הטיילור[[2]](#footnote-2) שלה. פיתוח סביב הנקודה מקיים:

כאשר מתקיים [[3]](#footnote-3). הטור נקרא "פולינום טיילור של f מסדר N", והמחובר השני הוא השארית בהצגת לגרנז'. בהנחה שהנגזרת ה-n+1 חסומה, נוכל לרשום את השארית גם בצורה כאשר M החסם.

האפשרות לפתח פונקציה לטור חזקות בסביבת נקודה קיימת הן במקרה הממשי, והן במקרה המרוכב. מכיוון שנדרשות לנו הנגזרות בנקודה, במקרה המרוכב אנליטיות הפונקציה בנקודה a מבטיחה את קיום הפיתוח. המקרה הממשי מורכב יותר, אך בהינתן גזירות n פעמים, ניתן לקבל קירוב לפולינום טיילור מסדר n-1.

לא מובטח שטור החזקות של פיתוח טיילור המתקבל מהפונקציה אכן מתכנס לפונקציה בכל התחום. ישנם מקרים בהם ההתכנסות מתקיימת בכל התחום, אחרים שבהם זה חלקי, וכאלה שבהם ההתכנסות מתקיימת בנקודת הפיתוח בלבד[[4]](#footnote-4). במסגרת העבודה לא נעסוק בניתוח רדיוס ההתכנסות, ונניח שהפונקציות איתן אנו עובדים בעלות רדיוס התכנסות חיובי.

באנליזה של פונקציות מרוכבות, קיימת הכללה של טורי טיילור לטורי לורן. טורים אלה מאפשרים להגדיר טור חזקות סביב נקודה גם כאשר הפונקציה בעלת סינגולריות בה (ובפרט אינה מוגדרת בה). זוהי הכללה לטור טיילור מפני שבמקרה שבו הפונקציה מוגדרת, הטורים מתלכדים. טור לורן של פונקציה הוא מהצורה הבאה:

כאשר האיברים מכונים מקדמי הטור, וטור איברי החזקות השליליות מכונה "החלק הסינגולרי" של הפונקציה. באמצעותו ניתן לסווג סינגולריות לשלושה סוגים: קטבים, עיקריות, וסליקות אם מספר האיברים בטור הסינגולרי סופי, אינסופי, או אפס בהתאמה[[5]](#footnote-5).

דיון מעמיק בטורי טיילור (ולורן) ניתן למצוא בקורסי האוניברסיטה הפתוחה "חשבון אינפיניטסימלי 2" (20212) ביחידות 5 ו-8, ו"פונקציות מרוכבות" (20243) ביחידות 6-7.

* 1. אינטגרלי קושי

אינטגרלי קושי הם אחת התוצאות מאנליזה מרוכבת, לפיה עבור פונקציה f אנליטית בתחום D, מתקיים לכל נקודה השיוויון הבא:

כאשר C קונטור סגור פשוט המוכל ב-D בעל כיוון חיובי ("נגד-השעון"), ו-a נקודה בפנים שלו. נוסחה זו בתוקף גם עבור n=0, ואז אנו מקבלים שיטת הצגה נוספת לפונקציה, בצורת אינטגרל.

במסגרת עבודה זו נניח שהפונקציות להן נרצה לחשב נגזרות הן אנליטיות. הנחה זו אמנם פוגעת בכלליות הדיון כי לא כל פונקציה ממשית ניתנת להרחבה לפונקציה מרוכבת אנליטית (למשל: ערך מוחלט), אך זו אינה פגיעה חזקה כי לרוב ניתקל בפונקציות שהן הרכבות ומנות של פונקציות "טובות" (אקספוננט, טריגונומטריה, פולינומים) – כך שהן אכן אנליטיות וניתנות להרחבה מהממשיים למישור המרוכב, או שניתן להרחיב אנליטית את הפונקציה בסביבה מצומצמת בה נתעניין (למשל: פונקציית הערך המוחלט עבור חיוביים שקולה לפונקציית הזהות, ושם ניתן להרחיבה בקלות).

במקרים שבהם קיימות נקודות סינגולריות לפונקציה, נדאג שהקונטור המקיף את הנקודה a יהיה קרוב מספיק אליה כך שהפונקציה אכן תהיה אנליטית כנדרש. ניתן לבנות מקרים שלא ניתן לעשות זאת כך שתמיד יש סינגולריות בסביבת הנקודה המבוקשת, אך אז פועל יוצא הוא שהנגזרת בנקודה אינה מוגדרת, ולכן אין סיבה לחשבה.

דיון נוסף באינטגרלי קושי ניתן למצוא בקורס "פונקציות מרוכבות" (20243) של האוניברסיטה הפתוחה, ביחידות 5 ו-8.

1. חישוב נגזרות בשיטת טורי טיילור
   1. נגזרת ראשונה

תהי נתונה פונקציה f(x), ונניח שניתן לפתחה לטור טיילור עם רדיוס התכנסות חיובי. עבור N=2 ו-h קטן מספיק מתקיים בסביבת הנקודה x=a:

לאחר סידור המשוואה ובהנחה שיש ל- חסם ב- שנסמנו [[6]](#footnote-6), נקבל:

נוכל גם לסדר את הנוסחה כך תהיה סימטרית סביב a:

כלומר:

תוצאה זו היא התוצאה השגרתית והמוכרת, אך אם נניח שהפונקציה f ניתנת להרחבה למישור המרוכב, נוכל לשפר את הערך המחושב או את שגיאת הקיטוע.

במקום לבחור צעד-קטן ממשי, נבחר צעד בכיוון המדומה (h ממשי) ונקבל:

כעת נטיל את המשוואה על הרכיב המדומה בלבד, ולאחר סידור המשוואה נקבל:

כלומר קיבלנו ביטוי לנגזרת הראשונה[[7]](#footnote-7) שאינו כולל חיסור, כך שאין חשש מאיבוד ספרות משמעותיות. חשוב לשים לב שיהיו מעט יותר חישובים כי אריתמטיקה מרוכבת כוללת שני רכיבים לכל מספר, אבל בסך-הכל התוצאה שהתקבלה משפרת את הביצועים. בנוסף, קיבלנו "בחינם" גם שיפור בשגיאת הקיטוע, מכיוון שגורם הנגזרת השנייה התבטל בלקיחת החלק המדומה.

נבחין שבניתוח יכולנו לבחור כל כיוון |z|=1 במישור המרוכב, וממנו לקחת את הרכיב המדומה, אבל אם הכיוון אינו הציר המדומה, נפסיד בשגיאת הקיטוע כי המקדם של הנגזרת השנייה יהיה מרוכב, ולמעשה אנו יוצרים שילוב בין השיטה שזה-עתה הצגנו (גזירה בכיוון הציר המדומה) לשיטה הממשית (גזירה בכיוון הציר הממשי). כמובן, ניתן גם לגזור בכיוון -i:

* 1. נגזרת שנייה

באופן דומה לחישוב הנגזרת הראשונה, נוכל לפתח טור טיילור ולחשב את הנגזרת השנייה:

*נסכם את שני הפיתוחים לקיזוז הביטויים שמחליפים סימן, ונקבל לאחר סידור המשוואה[[8]](#footnote-8):*

את השיפור המרוכב ניישם בדומה לנגזרת הראשונה, הפעם בכיוון j המקיים [[9]](#footnote-9). פעם נוספת נקבל שיפור לשגיאת הקיטוע יחד עם חיסכון בפעולות הפרש, אך לא נחסוך את כולן:

נסכם את שני הפיתוחים, וניקח את הרכיב המדומה. נקבל:

לסיכום, נקבל שיפור בשני סדרי גודל לשגיאת קיטוע, ופחות פעולת חשבון:

לו ניסינו לגזור כאן בכיוון המדומה, היינו מרוויחים, אך פחות:

פונקציה ממשית וגם a, לכן , ומכאן נקבל:

אך החיסרון כאן כפול: גם שגיאת הקיטוע הפסידה שני סדרי גודל, וגם במקום סכום בביטוי יש לנו הפרש. מכיוון שאנו מחשבים ערכים בנקודות קרובות, סביר יותר שיהיו דומים ולכן הפרשים יכניסו בסיכון גבוה שגיאות-הפרשים, בעוד שסכום יסכן אותנו רק כאשר הערך המבוקש קטן וקרוב לאפס, ואז הערכים עלולים להיות מנוגדי סימן.

* 1. נגזרות גבוהות

נגזרות גבוהות יותר (שלישית ואילך) כבר קשה לקבל באופן ישיר (יחסית) מנוסחת טיילור, בין אם בפונקציות ממשיות או מרוכבות. הטכניקה המשמשת במקרה זה מערבת מקרה פרטי של אקסטרפולציית ריצ'רדסון, המבטל ביטויים מסדרים נמוכים לקבלת דיוק גבוה יותר, ובמקרה שלנו את הנגזרת המבוקשת.

נדגים עבור הנגזרת השלישית, ראשית את המקרה הממשי:

ומכאן:

באופן דומה עבור 2h:

כך שנקבל:

וכעת נשווה לגישה המרוכבת. נבחר צעד של jh. לאחר תהליך דומה נקבל:

כאשר מקדם מספרי כלשהו שניתן לחשב, אך אינו רלוונטי. לקיחת החלק הממשי תבטל את הגורם של (כזכור ) ונקבל שיפור בשגיאת הקיטוע:

*פתרון אחר שאינו משפר את שגיאת הקיטוע אבל חוסך חישובים יהיה לעצור כבר בשלב מוקדם יותר, כבר בשלב שבו חישבנו את :*

עתה לקיחת החלק הממשי תבטל את הגורם הנוסף ותשאיר רק תוצאה ושגיאת קיטוע:

למרות ששגיאת הקיטוע בשיטה זו פחות טובה מהקודמת, בזכות החיסכון בהפרשים שיטה זו תתגלה במוצלחת יותר. נדגים זאת בהמשך, בפרק הניתוח הנומרי.

המקרה של נגזרת אי-זוגית גבוהה יותר מנגזרת שלישית יהיה דומה, כאשר בצעד האחרון נוכל לבחור אם לשפר את שגיאת הקיטוע או לחסוך בחישובים.

במקרה של נגזרות גבוהות זוגיות, נחבר את הביטויים (כלומר: ) כדי לאפס את האיברים האי-זוגיים בפיתוח בטור ונבצע אקסטרפולציה דומה. בכל מקרה, עדיין נבחר את הצעד להיות (במישור המרוכב), ונבחר בסוף את הרכיב הממשי.

1. חישוב נגזרות באמצעות אינטגרלי קושי
   1. תיאור השיטה והערכת השגיאה

בניגוד לפרק של טורי-טיילור, בו היה ברור מיד מה החישוב ומה השגיאה, כאן נדרש תחילה להגדיר כיצד לחשב את ערך האינטגרל ומשם להעריך את השגיאה בתוצאה. הטכניקה הפשוטה ביותר שניתן לחשוב עליה היא חישוב האינטגרל לאורך קונטור מעגלי ברדיוס r (בתנאי שהעיגול מוכל בסביבה אנליטית), כאשר את חישוב האינטגרל נבצע באמצעות כלל-הטרפז, על-ידי בחירת N נקודות בריווח אחיד לאורך המעגל[[10]](#footnote-10). כלומר:

*כאשר הוא שורש היחידה ה-k מסדר N ואנו מניחים כרגע פיתוח סביב 0. אם נניח שהפונקציה הנתונה אמנם אנליטית ב-0, אז קיים לה פיתוח טיילור עם המקדמים ומתקיים:*

*החלפת סדר הסכימה מותרת כי הטור מתכנס בהחלט ברדיוס התכנסותו. עתה, נבחין שאם m אינו כפולה של N (כולל 0), אז סכום החזקות ה-mיות של שורשי היחידה מתאפס, אחרת כל חזקה היא 1 והסכום הוא N. לכן נקבל:*

כמה קרובה התוצאה שקיבלנו לערך האינטגרל האמיתי? לפי חישוב מדויק ונוסחת קושי, מתקיים:

כלומר, יחד עם אי-שיוויון המשולש ואי-שיוויון קושי למקדמי טיילור (), נקבל:

כאשר ו-R רדיוס עיגול שבו הפונקציה אנליטית (ומכאן שכן כל החישוב שלנו על הקונטור ברדיוס r הניח אנליטיות). לכן תיאורטית, מומלץ לנו לחשב את האינטגרל ברדיוס r קטן ככל האפשר כדי להגדיל את המנה , ולשפר את הדיוק.

עם-זאת, בפיתוח שבוצע עד כאן הנחנו אנליטיות של הפונקציה בנקודה z=0, אבל בהמשך כשנחשב נגזרות מפאת צורת-קושי ניתקל בסינגולריות (קוטב) שם. לכן עלינו לקחת זאת בחשבון בניתוח שעשינו, ולתקן כנדרש עבור פונקציה אנליטית בטבעת . גם במקרה זה יידרש כמובן R>r. נחליף את טור טיילור בטור לורן:

גם עבור חזקות שליליות נקבל שהטור מתאפס כאשר m אינו כפולה של N, כך שנקבל:

את החסם על החלק הלא-סינגולרי כבר חסמנו וחישבנו קודם, ודבר לא השתנה. החלק הסינגולרי מוסיף תרומה זהה, על-סמך השיקול הבא: אם נבצע החלפת משתנים , החלקים הסינגולרי והלא-סינגולרי יחליפו תפקידים. לכן נוכל להעריך את הביטוי הסינגולרי באופן דומה כאשר:

יש לשים לב שמכיוון שלאו דווקא , ייתכן שהטרנספורמציה הזו תניב M שונה עבור החלק הסינגולרי (כי המעגל שלפיו מחושב M יהיה עבור כעת). לכן נגדיר את M בתור הגדול מבין שני החסמים, ואז אכן נוכל להכליל את אי-השיוויון לכל המקדמים, ומכאן כבר ברור שהחסם שיתקבל יהיה החסם שכבר מצאנו, פעמיים:

במאמר [5] אף מעירים שהקבוע הדוק ואינו ניתן לשיפור ומראים זאת על-ידי דוגמה של הפונקציה , בגבול .

* 1. יישום השיטה

בתיאור השיטה התייחסנו לפונקציה כלשהי, סביב הנקודה 0. כעת נתאר כיצד ליישם את השיטה לנגזרת של פונקציה בנקודה כללית[[11]](#footnote-11).

תהי נתונה הפונקציה f(z), וברצוננו להעריך את הנגזרת ה-n בנקודה z=a. נגדיר את הפונקציה:

*שהיא פשוט הזזה של f על-פני המישור המרוכב, ומתקיים . כלומר הנגזרת ה-n של f בנקודה a, שווה לנגזרת ה-n של h בנקודה 0:*

נבחין, שבגלל המקדם כלל לא ברור שעלינו לבחור את r קטן ככל האפשר, כי זה יגדיל את השגיאה. הסיבה להבדל בין מקרה זה לניתוח שעשינו היא שהנחנו שם אנליטיות, וכאן הנקודה z=0 היא סינגולריות של h(z), ולכן צריך להיזהר בחישובים בקרבתה. נסכם, קיבלנו שהקירוב המבוקש הוא:

כלומר:

נזכור להשתמש בחסם עבור אנליטיות בטבעת בגלל הסיכוי לקוטב, ונקבל את הערכת השגיאה*[[12]](#footnote-12)*:

באופן לא מפתיע, השגיאה גדלה ככל שהנגזרת המבוקשת מסדר גבוה יותר. בנוסף, ככלל-אצבע נרצה לבחור N>n כך שהביטוי יהיה קטן, בפרט כאשר סביר שנעבוד עם ערכים r<1. כמו-כן נדרוש שבכל מקרה R>r, כדי שהחסם יהיה מוגדר (ואף נעדיף יחס גדול, להקטנת החסם).

נבחין שיתרון נוסף שהתקבל בעקיפין מעצם השימוש בשיטה, הוא שהחישוב אחיד עבור כל נגזרת מסדר כלשהו, ואין צורך לפתח אנליטית ביטוי חדש לכל נגזרת כפי שעשינו בשיטת טיילור.

* 1. שימוש נוסף: חישוב ערכי סינגולריות סליקה

יתרון נוסף לשימוש באינטגרלי קושי הוא היכולת לחשב בצורה מדויקת יותר ערכי פונקציה בנקודות סינגולריות-סליקה[[13]](#footnote-13). הבעיה בחישוב ערכים בסביבת סינגולריות סליקה היא פוטנציאל שגיאה גדול עקב חלוקה במספר קרוב לאפס. כדי לפתור זאת, נוכל לחשב את "הנגזרת האפס" של הפונקציה בנקודה זו ובכך לקבל את ערכה שם. כמובן, גם במקרה זה כמו בחישובי נגזרות, עלינו להיזהר בבחירת הקונטור המעגלי סביב הנקודה, כדי שיכיל אך ורק את הסינגולריות המבוקשת ולא סינגולריות נוספות.

1. השוואת ביצועים נומריים

בפרק זה בעבודה נשווה את הטכניקות השונות שסקרנו לפי מספר פרמטרים. נתחיל עם ניתוח של שיטת טיילור המרוכבת לשיפור בחישוב הנגזרת הראשונה, ואת השימוש בשיטת אינטגרלי קושי לחישוב ערך פונקציה בסינגולריות סליקה.

לאחר מכן נעבר להשוואה מלאה בין כל השיטות בחישוב הנגזרת השנייה של מספר פונקציות בכמה נקודות. אחר-כך נבחן את ביצועי השיטות עבור הנגזרת החמישית, כמייצגת לנגזרות מסדרים גבוהים. את התוצאות נשווה ברמת הדיוק, ובזמן הריצה[[14]](#footnote-14). לבסוף נבדוק את ישימות שיטת קושי לנגזרות גבוהות ככל האפשר.

* 1. נגזרת ראשונה

ראשית נבחן את הפונקציה[[15]](#footnote-15) , בנקודה x=0 [[16]](#footnote-16):

1. נגזרת ראשונה

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| step-size | f'(0)=1 | | error | | time (ms) | |
|  | Real | Complex | real | complex | real | complex |
| 1 | 2.223509184784455 | 0.245979395288907 | 1.22E+0 | 7.54E-1 | 0.284 | 4.120 |
| 10^-1 | 1.006527322092925 | 0.993199297261026 | 6.53E-3 | 6.80E-3 | 0.297 | 4.250 |
| 10^-2 | 1.000066652997333 | 0.999933319669321 | 6.67E-5 | 6.67E-5 | 0.287 | 4.070 |
| 10^-3 | 1.000000666665335 | 0.999999333331967 | 6.67E-7 | 6.67E-7 | 0.304 | 4.190 |
| 10^-4 | 1.000000006667334 | 0.999999993333333 | 6.67E-9 | 6.67E-9 | 0.295 | 4.080 |
| 10^-5 | 1.000000000062062 | 0.999999999933333 | **6.21E-11** | 6.67E-11 | 0.287 | 4.090 |
| 10^-6 | 0.999999999973244 | 0.999999999999333 | **2.68E-11** | 6.67E-13 | 0.282 | 4.080 |
| 10^-7 | 1.000000000028755 | 0.999999999999994 | **2.88E-11** | 6.33E-15 | 0.282 | 4.080 |
| 10^-8 | 0.999999993922529 | 1.000000000000000 | 6.08E-9 | 2.22E-16 | 0.277 | 4.080 |
| 10^-9 | 1.000000027229220 | 1.000000000000000 | 2.72E-8 | 0 | 0.294 | 4.030 |
| 10^-10 | 1.000000082740371 | 1.000000000000000 | 8.27E-8 | 0 | 0.275 | 4.030 |

ניתן להבחין שהשגיאה במקרה הממשי מינימלית כשגודל הצעד הוא כ- , שם הטריידאוף שבין שגיאת הקיטוע ושגיאות-ההפרשים מתחיל להתהפך. לעומתו, בשיטה המרוכבת הדיוק רק הולך וגדל ככל שגודל הצעד קטן, עד לרמת הדיוק "מושלמת" במסגרת רזולוציית הטיפוס double. נבחין, שמחיר הדיוק המשופר הוא פקטור 15 בזמן החישוב.

עתה נחזור על הניתוח ההשוואתי עבור , ונקבל תוצאות דומות:

1. נגזרת ראשונה

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| step-size |  | | error | | time (ms) | |
|  | real | complex | real | complex | real | complex |
| 1 | 2.792294140944899 | -0.959854655665624 | 3.09E-1 | 4.06E+0 | 0.286 | 4.450 |
| 10^-1 | 3.061511866568116 | 3.144276040634560 | 4.03E-2 | 4.25E-2 | 0.286 | 4.370 |
| 10^-2 | 3.101352937655877 | 3.102180075411271 | 4.13E-4 | 4.14E-4 | 0.311 | 4.350 |
| 10^-3 | 3.101762258158169 | 3.101770529535847 | 4.14E-6 | 4.14E-6 | 0.289 | 4.340 |
| 10^-4 | 3.101766352480162 | 3.101766435192940 | 4.14E-8 | 4.14E-8 | 0.287 | 4.350 |
| 10^-5 | 3.101766393398541 | 3.101766394249621 | **4.38E-10** | 4.14E-10 | 0.318 | 4.340 |
| 10^-6 | 3.101766393509564 | 3.101766393840188 | **3.26E-10** | 4.14E-12 | 0.289 | 4.340 |
| 10^-7 | 3.101766399282723 | 3.101766393836095 | 5.45E-9 | 4.35E-14 | 0.285 | 4.350 |
| 10^-8 | 3.101766399282723 | 3.101766393836054 | 5.45E-9 | 2.66E-15 | 0.295 | 4.340 |
| 10^-9 | 3.101766399282723 | 3.101766393836053 | 5.45E-9 | 1.78E-15 | 0.287 | 4.250 |
| 10^-10 | 3.101758849766156 | 3.101766393836053 | 7.54E-6 | 1.33E-15 | 0.286 | 4.250 |

כעת נבחן מקרה נוסף שבו שיטתנו מסתכנת בסינגולריות: . הפונקציה רציפה ואנליטית ברצועה הכוללת את הישר הממשי, אך בעלת סינגולריות ב- . נשווה ביצועים עבור :

1. נגזרת ראשונה

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| step-size |  | | error | | time (ms) | |
|  | real | complex | real | complex | real | complex |
| 1 | -0.004999999987500 | -49.99875003124922 | 1.50E-2 | 5.00E+1 | 0.129 | 1.550 |
| 10^-1 | -0.019602116052349 | -0.020401875948413 | 3.94E-4 | 4.06E-4 | 0.119 | 1.550 |
| 10^-2 | -0.019992003198721 | -0.019999999200000 | 4.00E-6 | 4.00E-6 | 0.115 | 1.810 |
| 10^-3 | -0.019995960620023 | -0.019996040579986 | 4.00E-8 | 4.00E-8 | 0.118 | 1.610 |
| 10^-4 | -0.019996000199529 | -0.019996000999720 | 4.00E-10 | 4.00E-10 | 0.119 | 1.550 |
| 10^-5 | -0.019996000583111 | -0.019996000603918 | **1.68E-11** | 4.00E-12 | 0.118 | 1.550 |
| 10^-6 | -0.019996000577560 | -0.019996000599960 | **2.24E-11** | 4.00E-14 | 0.124 | 1.550 |
| 10^-7 | -0.019995999855915 | -0.019996000599920 | 7.44E-10 | 3.99E-16 | 0.161 | 1.580 |
| 10^-8 | -0.019995993749689 | -0.019996000599920 | 6.85E-9 | 3.47E-18 | 0.131 | 1.590 |
| 10^-9 | -0.019996004851919 | -0.019996000599920 | 4.25E-9 | 3.47E-18 | 0.118 | 1.560 |
| 10^-10 | -0.019995671785011 | 0.019996000599920- | 3.29E-7 | 0 | 0.118 | 1.550 |

נבחין שהרגישות לסינגולריות באה לידי ביטוי כאשר גודל הצעד גדול ומכיל אותה בפנים הקונטור, אך כבר ב- 10^-1, אין זכר לסינגולריות והביצועים מצוינים כמו בשיטה הממשית. ניתן לראות שההתנהגות דומה גם כאשר x=0, אך לא נצרף טבלה לשם כך. כמובן שבמקרה x=0 אי אפשר לבחור גודל-צעד של 1, כי אז ממש נדרוך על הסינגולריות.

* 1. חישוב ערך בסינגולריות סליקה ("נגזרת אפס")

נבחן עתה את הפונקציה[[17]](#footnote-17) , אשר מניתוח אנליטי ידוע שהיא בעלת סינגולריות סליקה ב- x=0, וערכה שם חצי. נשווה בין הדיוק של שיטת המיצוע:

לעומת שיטת אינטגרלי קושי עבור "הנגזרת האפס". נקבל:

1. סינגולריות סליקה

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| step-size |  | | error | | time (ms) | |
|  | Real | Cauchy  N=4 | Real | Cauchy | Real | Cauchy |
| 1 | 0.543080634815244 | 0.501389164473552 | 4.31E-2 | 1.39E-3 | 0.322 | 13.600 |
| 10^-1 | 0.500416805580361 | 0.500000138888889 | 4.17E-4 | 1.39E-7 | 0.352 | 13.500 |
| 10^-2 | 0.500004166680279 | 0.500000000013820 | 4.17E-6 | **1.38E-11** | 0.327 | 13.500 |
| 10^-3 | 0.500000041703252 | 0.500000000014347 | 4.17E-8 | **1.43E-11** | 0.325 | 13.500 |
| 10^-4 | 0.500000002512380 | 0.500000002512073 | **2.51E-9** | 2.51E-9 | 0.337 | 13.500 |
| 10^-5 | 0.499999486258673 | 0.500000318922880 | 5.14E-7 | 3.19E-7 | 0.363 | 13.600 |
| 10^-6 | 0.499988939139939 | 0.500044450260554 | 1.11E-5 | 4.45E-5 | 0.328 | 13.700 |
| 10^-7 | 0.499600361081320 | 0.499600360775158 | 4.00E-4 | 4.00E-4 | 0.322 | 14.100 |
| 10^-8 | 0.000000000000000 | 0.277555753094672 | 5.00E-1 | 2.22E-1 | 0.349 | 17.300 |

ניתן להבחין ששיטת קושי, כבר עבור 4 נקודות בלבד, משמעותית טובה יותר מהשיטה הממשית כאשר גודל הצעד אינו קטן מדי (אז נגרמת לשגיאת דיוק). על-אף שנראה שעבור 10^-4 השיטות מניבות ביצועים דומים, נשים לב שאם איננו מתעקשים על גודל צעד קטן, עבור 10^-2 , 10^-3 שיטת אינטגרלי קושי טובה בשני סדרי גודל מהדיוק הגבוה ביותר בשיטת המיצוע הממשית.

כמו-כן ניתן לשפר עוד יותר אם נבחר בשיטת קושי N גדול יותר:

1. סינגולריות סליקה

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| step-size | Cauchy error | | | | time (ms) | | | |
|  | N=4 | N=8 | N=16 | N=32 | N=4 | N=8 | N=16 | N=32 |
| 10^-1 | 1.39E-7 | **1.33E-15** | **1.55E-15** | **1.44E-15** | 13.70 | 26.30 | 50.40 | 111.00 |
| 10^-2 | 1.38E-11 | 3.62E-14 | 1.05E-13 | 2.20E-13 | 13.80 | 26.20 | 52.00 | 107.00 |
| 10^-3 | 1.43E-11 | 7.11E-12 | 3.62E-11 | 2.16E-12 | 14.00 | 27.30 | 59.40 | 104.00 |
| 10^-4 | 2.51E-9 | 1.26E-9 | 1.86E-10 | 3.86E-10 | 14.60 | 28.10 | 54.50 | 141.00 |
| 10^-5 | 3.19E-7 | 1.59E-7 | 1.45E-7 | 3.93E-9 | 13.60 | 26.80 | 50.40 | 105.00 |
| 10^-6 | 4.45E-5 | 2.22E-5 | 2.85E-7 | 3.97E-6 | 14.10 | 25.80 | 62.40 | 105.00 |

על-סמך הטבלה, ניתן להבחין בפרטים הבאים:

1. השיטה רגישה מאוד לגודל הצעד. כאשר אנו בוחרים גודל צעד קטן מדי, איננו מקבלים שיפור טוב ולכן עדיף להישאר עם צעד "בינוני", כל עוד ניתן לסמוך על-כך שהקונטור שלנו לא יכיל נקודות בעייתיות נוספות (סינגולריות או נקודות הסתעפות[[18]](#footnote-18)). יותר מכך, מלבד המקרה של N=4, בשאר המקרים הדיוק רק מדרדר ככל שהצעד קטן – מה שאומר שעלינו לבחור צעד גדול באופן יחסי.
2. הגדלת N מאריכה את זמן החישוב ביחס ההגדלה, כלומר זמן החישוב הוא O(N). זה הגיוני, שכן N הוא מספר הנקודות להערכת הפונקציה, וזמן הריצה ליניארי במספרן.
3. למרות הגידול הליניארי ב-N בזמן הריצה, ההגדלה משפרת את הדיוק בכמה סדרי גודל. שיפור זה מהיר מאוד, וכבר N=8 מרבע את הדיוק (עבור 10^-1). כפי שראינו בניתוח האנליטי שלנו, ונידון בהרחבה במאמר [5], תלות השגיאה ב-N אכן מעריכית. עם-זאת נבחין שהקפיצה יחסית מיידית, וכפי שמתואר במאמר [1] לצרכים מעשיים לאו דווקא יהיה הבדל ניכר כל-עוד נבחר את N להיות "מספק" ונשקיע בבחירת גודל-הצעד.
4. כאשר גודל הצעד קטן יותר איננו מצליחים לקבל דיוק משופר בסדרי-גודל רבים כאשר מגדילים את N, משום רגישות השיטה לגודל הצעד, כפי שהערנו בהערה הראשונה.
   1. נגזרת שנייה

לאחר שהשווינו מקרים של נגזרת ראשונה וסינגולריות סליקה ("נגזרת אפס"), נעבור להשוות השוואה רחבה יותר בין שלוש השיטות ביחס לנגזרת השנייה: טיילור ממשי, ושני השיפורים המוצעים של טיילור מרוכב, ואינטגרלי קושי.

ננתח את הפונקציות שכבר שימשו אותנו בסעיפים הקודמים[[19]](#footnote-19):

נתחיל מ-f ונבחן את ערכי נגזרתה השנייה[[20]](#footnote-20) ב- x=0, בשיטות השונות. כפי שכבר שמנו לב, שיטת קושי רגישה לגודל הצעד ולכן נבחר גודל-צעד שונה בין השיטות, לפי מה שיתאים להן:

1. נגזרת שנייה

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| step-size |  | | error | | time (ms) | |
|  | Real | Complex | real | complex | real | complex |
| 1 | 0.767566420814179 | 3.046024339315498 | 3.23E+0 | 9.54E-1 | 0.363 | 8.230 |
| 10^-1 | 4.023353532173734 | 3.999982298429450 | 2.34E-2 | 1.77E-5 | 0.371 | 8.210 |
| 10^-2 | 4.000233335110792 | 3.999999998222223 | 2.33E-4 | 1.78E-9 | 0.367 | 8.230 |
| 10^-3 | 4.000002333359731 | 3.999999999999664 | 2.33E-6 | **3.36E-13** | 0.393 | 8.220 |
| 10^-4 | 4.000000020099037 | 3.999999999999284 | **2.01E-8** | **7.16E-13** | 0.356 | 8.260 |
| 10^-5 | 4.000000330961483 | 4.000000000017240 | 3.31E-7 | 1.72E-11 | 0.360 | 8.290 |
| 10^-6 | 4.000355602329362 | 3.999999999898655 | 3.56E-4 | 1.01E-10 | 0.360 | 8.210 |
| 10^-7 | 3.996802888650562 | 3.999999999951594 | 3.20E-3 | 4.84E-11 | 0.355 | 8.230 |
| 10^-8 | 0.000000000000000 | 3.999999984731471 | 4.00E+0 | 1.53E-8 | 0.341 | 8.040 |

עתה יחס זמני הריצה הוא 22.5 (טיילור המרוכב איטי יותר), אך עדיין אין ספק באשר לדיוק המשופר של שיטה זו. נוסיף טבלה נפרדת לבחינת שיטת קושי עם מספר ערכי N:

1. נגזרת שנייה

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| step-size | Couchy | | | | error | | | |
|  | N=4 (21ms) | N=8 (40ms) | N=16 (78ms) | N=32 (154ms) | N=4 | N=8 | N=16 | N=32 |
| 1 | 1.345391162160405 | 2.195707750737950 | 2.145228628223059 | 2.256398306010956 | 2.65  E+0 | 1.80  E+0 | 1.85  E+0 | 1.74  E+0 |
| 10^-1 | 4.000017854015708 | 4.000000076222547 | 4.000000000000030 | 3.999999999999991 | 1.79  E-5 | 7.62  E-8 | **3.02**  **E-14** | **8.88**  **E-15** |
| 10^-2 | 4.000000001772586 | 3.999999999995298 | 4.000000000003029 | 4.000000000001294 | 1.77  E-9 | **4.70**  **E-12** | 3.03  E-12 | 1.29  E-12 |
| 10^-3 | 3.999999999726445 | 3.999999999626482 | 3.999999999643178 | 3.999999999768078 | **2.74**  **E-10** | 3.74  E-10 | 3.57  E-10 | 2.32  E-10 |
| 10^-4 | 3.999999986792346 | 3.999999969322800 | 4.000000014547922 | 3.999999997200687 | 1.32  E-8 | 3.07  E-8 | 1.45  E-8 | 2.80  E-9 |
| 10^-5 | 4.000000330961483 | 3.999997931395494 | 4.000000192183605 | 3.999998873793764 | 3.31  E-7 | 2.07  E-6 | 1.92  E-7 | 1.13  E-6 |

והדגשנו את החישוב המוצלח ביותר לכל N שנבדק.

שוב ניתן לראות שהזמן גדל ליניארית ב-N, ושהדיוק גדל במהירות גדולה יותר עם הגידול ב-N, אך מאידך הרגישות לגודל הצעד גדלה ולכן הדיוק הטוב ביותר עבור N=16,32 הוא עבור צעד 0.1, בעוד שעבור N=4,8 הצעד האופטימלי מהערכים שבדקנו קטן יותר (10^-2 , 10^-3).  
  
נבחן עוד את הפונקציה, במספר נקודות שונות, תחת הפרמטרים ה"מועדפים" שמצאנו. את שיטת קושי נבחן עבור N=32 ועבור N=8, מכיוון ש- N=32 מבכר דיוק, והבדיקה עם N=8 היא לתרחיש שבו N=32 יתברר כרגיש מדי לשגיאות דיוק.

1. נגזרת שנייה

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| f(x)=value | method (step) | value | error | time (ms) |
|  | Taylor R[[21]](#footnote-21) (1E-4) | -0.172855672830874 | 2.24E-10 | 0.407 |
| Taylor C (1E-3) | -0.172855673055036 | 5.30E-14 | 8.340 |
| **Cauchy32 (1E-1)** | **-0.172855673055089** | **2.22E-16** | **171.0** |
| Cauchy 8 (1E-2) | -0.172855673054861 | 2.28E-13 | 43.70 |
|  | Taylor R (1E-4) | 19.361834846520765 | 2.29E-6 | 0.410 |
| Taylor C (1E-3) | 19.361832557282089 | 2.79E-9 | 8.380 |
| **Cauchy32 (1E-1)** | **19.361832560070550** | **5.33E-14** | **155.0** |
| Cauchy 8 (1E-2) | 19.361832560108642 | 3.81E-11 | 40.10 |
|  | Taylor R (1E-4) | -6.203532532111922 | 2.56E-7 | 0.417 |
| Taylor C (1E-3) | -6.203532787630185 | 4.19E-11 | 8.210 |
| **Cauchy32 (1E-1)** | **-6.203532787672127** | **2.40E-14** | **155.0** |
| Cauchy 8 (1E-2) | -6.203532787678427 | 6.32E-12 | 40.00 |
|  | Taylor R (1E-4) | 19.241909932077306 | 4.08E-7 | 0.403 |
| Taylor C (1E-3) | 19.241909523809703 | 5.17E-11 | 8.180 |
| **Cauchy32 (1E-1)** | **19.241909523861395** | **7.11E-15** | **154.0** |
| Cauchy 8 (1E-2) | 19.241909523839556 | 2.18E-11 | 39.90 |

נבחין שבכל אחת מהנקודות קיבלנו שהשיטות המרוכבות משפרות בכמה סדרי גודל (לפחות 3-4) את דיוק התוצאה. ביצועי שיטת קושי עבור N=8 דומים יחסית לשיטת טיילור המרוכבת אך איטיים פי חמישה. עם-זאת, שיטת קושי עבור N=32 מנצחת באופן מובהק (במחיר האטה פי עשרים משיטת טיילור המרוכבת).

נערוך השוואה דומה לפונקציה h(x) במספר נקודות:

1. נגזרת שנייה

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| h(x)=value | method (step) | value | error | time (ms) |
|  | Taylor R (1E-4) | 0.015522322316386 | 1.03E-9 | 0.581 |
| Taylor C (1E-3) | 0.015522321286448 | 2.87E-14 | 5.970 |
| **Cauchy32 (1E-1)** | **0.015522321286419** | **6.97E-16** | **129.0** |
| Cauchy 8 (1E-2) | 0.015522321286163 | 2.56E-13 | 43.60 |
|  | Taylor R (1E-4) | 0.062047078586147 | 3.48E-9 | 0.685 |
| Taylor C (1E-3) | 0.062047082068930 | 1.32E-12 | 7.180 |
| **Cauchy32 (1E-1)** | **0.062047082067595** | **1.27E-14** | **144.0** |
| Cauchy 8 (1E-2) | 0.062047082068454 | 8.46E-13 | 32.90 |
|  | Taylor R (1E-4) | 0.135128352951597 | 7.52E-8 | 0.865 |
| Taylor C (1E-3) | 0.135128277718483 | 3.68E-13 | 7.050 |
| **Cauchy32 (1E-1)** | **0.135128277718839** | **1.18E-14** | **124.0** |
| Cauchy 8 (1E-2) | 0.135128277720185 | 1.33E-12 | 45.50 |
|  | Taylor R (1E-4) | 0.224295826356524 | 4.46E-8 | 0.799 |
| Taylor C (1E-3) | 0.224295781797875 | 9.09E-14 | 8.650 |
| **Cauchy32 (1E-1)** | **0.224295781797792** | **8.41E-15** | **178.0** |
| Cauchy 8 (1E-2) | 0.224295781794809 | 2.98E-12 | 42.60 |

וגם במקרה זה קיבלנו תוצאות דומות: שיפור משמעותי במספר סדרי גודל של השיטות המרוכבות, וזמני ריצה דומים. עם-זאת נבחין שבמקרה זה עליונות שיטת קושי עם N=32 פחות מובהקת, ושיטת טיילור המרוכבת קרובה מעט יותר בביצועיה (הבדל בסדר-גודל או שניים "בלבד").

* 1. נגזרת חמישית

ראשית נחזור על כל מה שעשינו בהשוואה עבור הנגזרת השנייה, גם בפרק זה על הנגזרת החמישית. נתחיל בטבלה שתזהה (בקירוב) את הפרמטרים המועדפים לכל שיטה, ולאחר מכן נשווה עבור מספר ערכים.

תחילה, טבלה עבור h(x), עבור x=1. חישוב אנליטי[[22]](#footnote-22) נותן:

*הבדיקה מראה שכל השיטות (הממשית והמרוכבות) מתחילות לצבור שגיאה החל מגודל צעד :*

1. נגזרת חמישית

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| method | step-size | value | error | time (ms) |
| Real Taylor | 1 | -1.#IND | -1.#JE+0 | 2.040 |
| 10^-1 | 0.051070897177004 | 1.56E-4 | 1.200 |
| **10^-2** | **0.050916770799603** | **1.76E-6** | **1.000** |
| 10^-3 | -0.397829917157348 | 4.49E-1 | 1.030 |
| 10^-4 | 18503.7170770859410 | 1.85E+4 | 1.024 |
| Complex Taylor | 1 | 0.050796861810353 | 1.18E-4 | 11.50 |
| **10^-1** | **0.050914994503632** | **1.17E-8** | **11.50** |
| 10^-2 | 0.050887334436002 | 2.77E-5 | 11.40 |
| 10^-3 | -0.130023281734750 | 1.81E-1 | 11.50 |
| 10^-4 | 50286.2742169021000 | 5.03E+4 | 11.50 |
| 10^-5 | -28669520305.145706 | 2.87E+10 | 11.50 |
| Cauchy N=4 | 1 | 48.857102544503313 | 4.88E+1 | 14.70 |
| 10^-1 | 338061.856764150490 | 3.38E+5 | 14.60 |
| Cauchy N=8 | 1 | 7.550915006375869 | 7.50E+0 | 26.80 |
| **10^-1** | **0.050915006211127** | **5.61E-11** | **26.80** |
| 10^-2 | 0.051175730320097 | 2.61E-4 | 26.70 |
| 10^-3 | 0.000000000000000 | 5.09E-2 | 26.70 |
| Cauchy N=16 | 1 | 3.800915006155032 | 3.75E+0 | 51.10 |
| **10^-1** | **0.050915005170293** | **9.85E-10** | **50.90** |
| 10^-2 | 0.051054993566169 | 1.40E-4 | 50.90 |
| 10^-3 | 6.661338147750938 | 6.61E+0 | 50.90 |
| Cauchy N=32 | 1 | 1311213694078071.0 | 1.31E+15 | 100.0 |
| **10^-1** | **0.050915004129459** | **2.03E-9** | **100.0** |
| 10^-2 | 0.050886378444304 | 2.86E-5 | 100.0 |
| 10^-3 | -0.832667268468867 | 8.84E-1 | 99.40 |

*בכל שיטה הדגשנו את גודל הצעד האופטימלי מבין שנבדקו (צעדים קטנים יותר ידרדרו עוד את הדיוק). בענייני שגיאות וזהירות, נשים לב לשני דברים:*

1. *במקרה הממשי, מכיוון שלפונקציה h יש סינגולריות ב-0, אז החישוב עם צעד של 1 גרם לביטוי לא מוגדר בחישוב של קירוב טיילור ממשי.*
2. *מעניין לראות ששיטת קושי עם N=4 מניבה שגיאות גבוהות בכל מקרה. ככל הנראה הדבר נובע מכך שבמקרה זה N<n (n=5), וכפי שראינו בניתוח האנליטי זה מתרגם צעד קטן לשגיאה גדולה בגלל הגורם הכפלי שבביטוי. אכן, אם נבדוק את העניין נגלה שעבור* N=5 *עדיין יש שגיאה גדולה, אך כאשר* N=6 *הקירוב כבר סביר.*

עתה נמשיך עם הפרמטרים מהשורות המודגשות, שנתנו את הביצועים הטובים ביותר. אמנם לא מובטח לנו שבחירת נקודה אחרת לא תשנה את הפרמטרים האופטימליים, אך הם קרובים מספיק ונעשה שימוש בהם.

1. נגזרת חמישית

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | method (step) | value | error | time (ms) |
|  | Taylor R (1E-2) | 0.002527075770864 | 9.04E-7 | 1.120 |
| Taylor C (1E-1) | 0.002527979345185 | **6.55E-10** | 12.10 |
| Cauchy 8 (1E-1) | 0.002527980075273 | **7.53E-11** | 26.70 |
|  | Taylor R (1E-2) | 0.016414138566863 | 6.74E-6 | 1.023 |
| Taylor C (1E-1) | 0.016407411020196 | **1.10E-8** | 11.50 |
| Cauchy 8 (1E-1) | 0.016407414260566 | **1.43E-8** | 29.30 |
|  | Taylor R (1E-2) | 0.043176480909087 | 4.22E-6 | 1.010 |
| Taylor C (1E-1) | 0.043180642172579 | **5.78E-8** | 11.50 |
| Cauchy 8 (1E-1) | 0.043180653219732 | **4.68E-8** | 26.70 |
|  | Taylor R (1E-2) | 0.001915643569698 | 2.19E-2 | 1.011 |
| Taylor C (1E-1) | 0.023809519673729 | **4.14E-9** | 11.50 |
| Cauchy 8 (1E-1) | 0.023809539506647 | **1.57E-8** | 27.00 |

וניתן לראות ששתי השיטות המרוכבות מניבות דיוק דומה עבור הפרמטרים שבחרנו.

1. נגזרת חמישית

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | method & parameters | value | error | time (ms) |
|  | Taylor R (1E-2) | 7.116028299021989 | 2.89E-2 | 0.883 |
| Taylor C (1E-1) | 7.066317900635627 | 2.08E-2 | 19.80 |
| **Cauchy 8 (1E-1)** | **7.087082906479660** | **3.11E-7** | **40.30** |
|  | Taylor R (1E-2) | -48968.07138554671 | 1.26E+3 | 0.901 |
| Taylor C (1E-1) | -37866.49057823190 | 9.84E+3 | 19.40 |
| **Cauchy 8 (1E-1)** | **-47722.18686826032** | **1.09E+1** | **41.60** |
|  | Taylor R (1E-2) | 1346.0476192920605 | 6.32E+0 | 0.864 |
| Taylor C (1E-1) | 1340.5506599733499 | 1.18E+1 | 19.50 |
| **Cauchy 8 (1E-1)** | **1352.3705203635209** | **3.73E-4** | **40.30** |
|  | Taylor R (1E-2) | -164.6698177815154 | 6.70E-1 | 0.860 |
| Taylor C (1E-1) | -163.5194905839312 | 4.81E-1 | 18.70 |
| **Cauchy 8 (1E-1)** | **-164.0000072035130** | **7.20E-6** | **40.00** |

התוצאות הנ"ל מעניינות – אמנם בפונקציה h שיטת טיילור טובה כמו שיטת קושי, אך בפונקציה f מתגלה כוחה של שיטת קושי. הסיבה לכך היא שערכי הפונקציה שאנו מחשבים כאן גדולים יותר, ו"רועשים" יותר עקב נקודות הסינגולריות שיש ל-f בנקודות המקיימות , למשל שקרובה מאוד ל- . לכן, כאשר אנו מסכמים וממצעים ערכים של מספר רב של נקודות, אנו מקטינים את אפקט השגיאה שתורמת נקודה סוררת בודדת. במקרה זה נוכל לראות שיפור נוסף אם נגדיל את N:

1. נגזרת חמישית

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Method  (step 1E-1) | value | error | time (ms) |
|  | Cauchy (N=8) | 7.087082906479660 | 3.11E-7 | 45.700 |
| **Cauchy (N=32)** | **7.087082595293687** | **3.50E-11** | **154.00** |
|  | Cauchy (N=8) | -47722.1868682603240 | 1.09E+1 | 44.700 |
| **Cauchy (N=32)** | **-47711.3410295032230** | **4.10E-2** | **154.00** |
|  | Cauchy (N=8) | 1352.370520363520900 | 3.73E-4 | 43.800 |
| **Cauchy (N=32)** | **1352.370147709113600** | **3.40E-9** | **154.00** |
|  | Cauchy (N=8) | -164.000007203512950 | 7.20E-6 | 40.600 |
| **Cauchy (N=32)** | **-164.000000000399840** | **4.00E-10** | **154.00** |

אך עבור N גדול יותר (למשל, 64 או 128) כבר אין שיפור דיוק משמעותי.

לסיום סעיף זה, נותר לנו חוב מפרק הניתוח האנליטי בו ציינו שאפשר לשפר את שגיאת הקיטוע במקום לחסוך בפעולות חשבון. נשווה כעת עבור הנגזרת החמישית בין שתי השיטות, וניווכח שההבדל זניח.

לאור זאת, מכיוון שהשיטה שחוסכת פעולות חשבון גם מהירה יותר, אכן מוצדק שבחנו רק אותה לאורך הניתוח מול השיטות האחרות (שיטת קושי, ושיטת טיילור הממשית). למתעניינים, לא נפרט כאן את הקירוב האנליטי לנגזרת החמישית בכל אחת מהשיטות, אך הוא מופיע בקוד שצורף שבנספח. להלן השוואה:

1. נגזרת חמישית

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | value | | error | | time (ms) | |
| step | value | Taylor C1  (less calculations) | Taylor C2  (better theoretical truncation error) | Taylor C1 | Taylor C2 | Taylor C1 | Taylor C2 |
| 0.1 |  | 0.002527979345185 | 0.002527971471989 | 8.37E-1 | 8.37E-1 | 11.80 | 17.20 |
|  | 0.016407411020196 | 0.016407343589305 | 7.36E-1 | 7.36E-1 | 11.50 | 17.20 |
|  | 0.043180642172579 | 0.043180444838779 | 6.80E-1 | 6.80E-1 | 11.60 | 17.30 |
|  | 0.023809519673729 | 0.023809416215103 | 7.14E-1 | 7.14E-1 | 11.50 | 20.70 |
|  | 7.066317900635627 | 6.676228419999505 | 4.19E+1 | 3.96E+1 | 18.50 | 27.50 |
|  | **-37866.49057823190** | **2091.8039525704539** | **1.96E+3** | **1.07E+2** | **20.90** | **27.40** |
|  | 1340.5506599733499 | 1132.5713184322217 | 2.17E+2 | 1.84E+2 | 20.50 | 29.40 |
|  | -163.5194905839312 | -154.4925498135009 | 4.19E+1 | 3.96E+1 | 18.50 | 27.60 |
| 0.01 |  | 0.002527097544707 | 0.002528238315009 | 8.37E-1 | 8.37E-1 | 11.50 | 17.20 |
|  | 0.016428680840959 | 0.016400288335664 | 7.35E-1 | 7.36E-1 | 11.40 | 17.20 |
|  | 0.043185613870349 | 0.043178499388896 | 6.80E-1 | 6.80E-1 | 11.40 | 19.30 |
|  | 0.027109608914749 | 0.023057895371901 | 6.75E-1 | 7.23E-1 | 12.70 | 17.10 |
|  | 7.087080889803938 | 7.087038699702286 | 4.20E+1 | 4.20E+1 | 21.10 | 27.40 |
|  | -47710.11808881979 | -47685.66860031953 | 2.47E+3 | 2.46E+3 | 18.40 | 27.40 |
|  | 1352.3689915494463 | 1352.3451025926875 | 2.19E+2 | 2.19E+2 | 18.40 | 27.30 |
|  | -163.9999472183231 | -163.9989885133258 | 4.20E+1 | 4.20E+1 | 18.30 | 27.50 |

השורה המודגשת היא היחידה שבה קיבלנו הבדל בסדר-גודל של ביצועי השיטות. אמנם בדקנו רק מדגם של ארבע נקודות, לשתי פונקציות ועם שני גדלי צעד בלבד, אך ניתן להשתכנע שאכן אין לשיטה השנייה (שאיטית בכ-50%) עדיפות על-פני הראשונה.

* 1. נגזרות גבוהות יותר

לסיום, ננתח את דיוק שיטת קושי לנגזרות גבוהות במיוחד. לא נשווה שיטה זו לשיטה הממשית משום שאין ביטוי כללי ממשי לנגזרת שרירותית. בתת-פרק זה נבחן את גידול השגיאה בשיטת קושי לעומת הגידול בנגזרת.

כדי לעשות זאת, נשתמש במספר פונקציות שנגזרותיהן ידועות:

נבצע ניתוח אנליטי נקבל:

ועתה לטבלאות. בכל המקרים מדובר על שיטת קושי עם צעד 0.1, ו- N=16:

1. נגזרת גבוהות

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | (52ms) | absolute err | relative err |  |
| 1 | 0.166666666666648 | 1.87E-14 | 1.12E-13 | 1.87E-15 |
| 2 | 0.083333333333598 | 2.65E-13 | 3.18E-12 | 1.32E-15 |
| 3 | 0.050000000027910 | 2.79E-11 | 5.58E-10 | 4.65E-15 |
| 4 | 0.033333333540860 | 2.08E-10 | 6.23E-9 | 8.65E-16 |
| 5 | 0.023809524123120 | 3.14E-10 | 1.32E-8 | 2.61E-17 |
| 6 | 0.017858756973244 | 1.61E-6 | 9.04E-5 | 2.24E-15 |
| 7 | 0.013898932005318 | 1.00E-5 | 7.23E-4 | 1.99E-16 |
| 8 | 0.014156675831600 | 3.05E-3 | 2.74E-1 | 7.55E-16 |
| 9 | 1.554856243757284 | 1.55E+0 | 1.70E+2 | 4.26E-15 |

ניתן להבחין שהדיוק יורד עם העלייה בסדר הנגזרת, הן בשגיאה האבסולוטית והן בשגיאה היחסית. עם-זאת, כצפוי על-סמך הניתוח התיאורטי, נרמול השגיאה בגורם נותן גודל קבוע יחסית.

1. נגזרות גבוהות

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | (34 ms) | absolute err | relative err |  |
| 1 | 0.500000000000000 | 2.22E-16 | 4.44E-16 | 2.22E-17 |
| 2 | -0.866025403784427 | 1.12E-14 | 1.29E-14 | 5.61E-17 |
| 3 | -0.499999999999757 | 2.43E-13 | 4.85E-13 | 4.04E-17 |
| 4 | 0.866025403808441 | 2.40E-11 | 2.77E-11 | 1.00E-16 |
| 5 | 0.500000000944189 | 9.44E-10 | 1.89E-9 | 7.87E-17 |
| 6 | -0.866025326451413 | 7.73E-8 | 8.93E-8 | 1.07E-16 |
| 7 | -0.499997843128596 | 2.16E-6 | 4.31E-6 | 4.28E-17 |
| 8 | 0.866103189167688 | 7.78E-5 | 8.98E-5 | 1.93E-17 |
| 9 | 0.460791405032523 | 3.92E-2 | 7.84E-2 | 1.08E-16 |
| 10 | -3.525180147789794 | 2.66E+0 | 3.07E+0 | 7.33E-17 |

גם במקרה זה רואים שהדיוק (האבסולוטי והיחסי) יורד עם העלייה בסדר הנגזרת, והשגיאה המנורמלת בגורם היא גודל קבוע יחסית. במקרה של פונקציה זו, מכיוון שערכי הנגזרת אינם דועכים לאפס, הצלחנו "לגרד" עוד נגזרת אחת (n=9) לפני שהתקבלה שגיאה מסדר-הגודל של הנגזרת עצמה.

טבלה אחרונה לסיום, וגם בה המסקנות דומות:

1. נגזרות גבוהות

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | (39 ms) | absolute err | relative err |  |
| 1 | 1.000000000000000 | 0 | 0 | 0 |
| 2 | -0.999999999999994 | 6.44E-15 | 6.44E-15 | 3.22E-17 |
| 3 | 2.000000000000261 | 2.61E-13 | 1.30E-13 | 4.34E-17 |
| 4 | -5.999999999998400 | 1.60E-12 | 2.67E-13 | 6.67E-18 |
| 5 | 24.000000000367464 | 3.67E-10 | 1.53E-11 | 3.06E-17 |
| 6 | -119.999999976180760 | 2.38E-8 | 1.98E-10 | 3.31E-17 |
| 7 | 719.999998397002400 | 1.60E-6 | 2.23E-9 | 3.18E-17 |
| 8 | -5040.000007838770200 | 7.84E-6 | 1.56E-9 | 1.94E-18 |
| 9 | 40319.983192205167000 | 1.68E-2 | 4.17E-7 | 4.63E-17 |
| 10 | -362879.2431542567200 | 7.57E-1 | 2.09E-6 | 2.09E-17 |
| 11 | 3628864.1941384333000 | 6.42E+1 | 1.77E-5 | 1.61E-17 |
| 12 | -39930181.36139164900 | 1.34E+4 | 3.35E-4 | 2.79E-17 |
| 13 | 479642872.41963822000 | 6.41E+5 | 1.34E-3 | 1.03E-17 |
| 14 | -6525584352.878856700 | 2.99E+8 | 4.79E-2 | 3.42E-17 |
| 15 | -12476493838.06501800 | 9.97E+10 | 1.14E+0 | 7.62E-17 |

נזכיר שהשגיאה בשיטת קושי מוערכת לפי , ולכן ציפינו שעמודת ה- תהיה יחסית קבועה בשלוש הטבלאות כי הגורם השני במכפלה אינו תלוי בסדר הנגזרת המבוקש. יובהר שאיננו יכולים בהכרח להסיק את ערכו מהערכים שבעמודה זו, משום שזהו חסם וייתכן שאינו הדוק.

נוסף על האבחנה שלעיל, נשים לב שבמקרה של , מכיוון שערך הנגזרת גדל גם הוא בצורת עצרת, הדיוק אמנם מדרדר, אך השגיאה היחסית מדרדרת לאט יותר מאשר שקורה בשתי הפונקציות האחרות (), כך שהצלחנו לחשב עד הנגזרת ה-14 תוך קבלת שגיאה-יחסית קטנה.

נוכל לשפר מעט את המצב בשתי הפונקציות הראשונות אם נבחר step=1, ככה ש- לא יתרום לגידול בשגיאה, אלא רק הגורם n! . לא נדגים זאת על משום שהנקודה x=0 אינה מוגדרת בה ואינה ניתנת להשלמה אנליטית, כך שלא ניתן לבחור לה step=1:

1. נגזרות גבוהות

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | error of h | |  | error of f2 | |
| n | (53ms) | absolute | relative | (34 ms) | absolute | relative |
| 1 | 0.166666666666667 | 2.78E-17 | 1.67E-16 | 0.500000000000001 | 1.44E-15 | 2.89E-15 |
| 3 | 0.050000000000000 | 1.94E-16 | 3.89E-15 | -0.500000000000000 | 0 | 0 |
| 5 | 0.023809523809528 | 4.08E-15 | 1.71E-13 | 0.500000000000007 | 6.55E-15 | 1.31E-14 |
| 7 | 0.013888888889025 | 1.36E-13 | 9.79E-12 | -0.500000000000033 | 3.32E-14 | 6.64E-14 |
| 9 | 0.009090909059020 | 3.19E-11 | 3.51E-9 | 0.499999999960830 | 3.92E-11 | 7.83E-11 |
| 11 | 0.006410255239020 | 1.17E-9 | 1.83E-7 | -0.500000005724499 | 5.72E-9 | 1.14E-8 |
| 12 | 0.005494417301732 | 8.82E-8 | 1.61E-5 | 0.866025252151648 | 1.52E-7 | 1.75E-7 |
| 13 | 0.004759506944163 | 2.40E-6 | 5.04E-4 | 0.499996905643307 | 3.09E-6 | 6.19E-6 |
| 14 | 0.004117999117703 | 4.87E-5 | 1.17E-2 | -0.866100357071531 | 7.50E-5 | 8.65E-5 |
| 15 | 0.002150493846997 | 1.53E-3 | 4.15E-1 | -0.502380769336952 | 2.38E-3 | 4.76E-3 |
| 16 | 10461394944000.00 | 1.05E+13 | 3.20E+15 | 18119667561053.0 | 1.81E+13 | 2.09E+13 |

כאשר עצרנו את החישובים כשהשגיאה-היחסית הגיעה (ואף עברה) את סדר-הגודל של הערך המחושב. והשיפור משמעותי בהחלט: אם קודם הגענו עד הנגזרת השביעית/שמינית, עתה חישבנו בהצלחה עם שגיאה-יחסית דומה (10^-5) את הנגזרת ה-14.

1. דיון ומסקנות

לאורך כל העבודה **ראינו שבאופן מובהק ניתן לשפר את ההערכות הנומריות לנגזרות על-ידי הרחבת הפונקציה הממשית למרוכבת** ושימוש בשיטת טיילור עם צעד בכיוון מרוכב, או שימוש בשיטות אינטגרלי קושי וקירוב שלהם.

לכל אורך הניתוח, **מצאנו ששיטת אינטגרלי קושי חזקה ומשפרת מאוד את הקירוב בשיטות הממשיות הרגילות**. חשוב לציין שלשיטה שני חסרונות עיקריים:

1. מהירות ריצה: שיטה זו מכבידה מאוד, פי כמה עשרות. יחס זה משתנה בנסיבות, אך ראינו בהשוואות של הנגזרת השנייה שהיחס הוא כ-50-100 (כאשר N=8), ובנגזרת החמישית ההבדל קטן מעט כי נדרשים יותר חישובים בשיטות טיילור. נזכור שזמן החישוב ליניארי ב-N, כך שעבור N=32 (למשל) יחס זמני הריצה גדל עוד, פי 4.
2. רגישות לגודל-צעד: כפי שראינו, שיטה זו תלויה מאוד בגודל הצעד ובפרט בנגזרות גבוהות, וגודל הצעד האופטימלי ששימש אותנו לאורך רוב החישובים בשיטה זו היה 0.1. הבעיה היא, שייתכן שאם לא נוכל להקטין את גודל הצעד "נתפוס" בקונטור שלנו סינגולריות ושאר בעיות שלא נוכל להימנע מהן.

למרות שתי מגרעות אלה, אין ספק ש**אם בכוונתנו להעדיף דיוק על-פני מהירות בכל מחיר, שיטה זו עדיפה ביותר, ותניב תוצאות טובות שעלותן אינה גבוהה אם נבחר N=16** מה שיספיק לנו לכל נגזרת מעשית שנרצה לחשב (כנראה שגם N=8).

**יתרון נוסף חשוב בשיטת קושי היא כלליותה. באמצעות נוסחה אחת פשוטה, ניתן לכתוב פונקציה תכנותית בודדת שמקבלת את סדר הנגזרת כפרמטר**, ואין צורך לפתח ביטוי אנליטי ולכתוב פונקציה לקירוב המבוקש לכל סדר.

למרות האמירות לעיל, **אל לנו לזלזל בשיטת טיילור המרוכבת בדיון על הנגזרת הראשונה**. כפי שראינו, **שיטה זו נותנת ביטוי לנגזרת ברמת הדיוק של המכונה**. אמנם לא השווינו ישירות בין שיטת קושי לשיטת טיילור בנגזרת הראשונה, אך אין צורך – גם אילו שיטת קושי תניב תוצאות כה מדויקות (מוטל בספק, כי היא כוללת הרבה אריתמטיקה), היא איטית בהרבה ואינה משתלמת כאן.

במסגרת העבודה הקדשנו פרק קצר לבחינה נומרית של **שיטת קושי על נגזרות גבוהות. באופן עקרוני, היא מאפשרת לנו לחשב נגזרות מסדר גבוה ככל שנרצה**, אפילו שהשגיאה מתבדרת בצורה די חזקה כתלות בסדר הנגזרת. **בפועל, אין לנו צורך מעשי** והניתוח הנומרי היה תיאורטי בעיקרו, כדי לבחון את החסם שמצאנו בניתוח האנליטי.

מרבית היישומים הפיסיקליים דורשים נגזרות ראשונות ושניות כגון מהירות ותאוצה במכאניקה או זרם והשראות בחשמל, ומעבר לכך כבר מדובר על מקרים נדירים יותר, ולא הצלחתי למצוא התייחסויות לצורך ממשי בנגזרת גבוהה מארבע, עם חריג אחד של נגזרות המקום לפי זמן[[23]](#footnote-23).

גם במשוואות דיפרנציאליות, שהן עולם שונה מעט אך נניח שהתרחיש הוא שיש בידינו פתרון שאנו רוצים לבדוק את נכונותו נומרית, מרבית המשוואות אינן מערבות נגזרות מסדר גבוה כך ששוב אין צורך מעשי בחישוב נומרי לנגזרת גבוהה מאוד[[24]](#footnote-24).

אם בכל-זאת נתעקש לחשב נגזרות גבוהות, עקב התבדרות השגיאה (לפי עצרת), כנראה שעדיף יהיה למצוא את המקדמים כמתואר במאמר [1], ומהם לחשב את ערך הנגזרת במקום המבוקש.

1. כיוונים להמשך מחקר

ניתן להמשיך ולפתח את המחקר שבוצע בעבודה. להלן כמה כיווני המשך:

* שיפור שיטת קושי להתמודדות עם גדלי צעד קטנים: כדי לאפשר הקטנת רדיוס הקונטור, כדי לשפר את הביטחון שלא נכלול בתוכו בטעות סינגולריות שיפגעו בחישוב שלנו. כיוון למחשבה הוא ביצוע מתיחה של המישור , כך שגודל צעד גדול למעשה יבטא צעד קטן. נבחין שמתיחה משפיעה על הנגזרת ( ), ועל-כן ייתכן שכיוון זה יתברר כפחות מועיל מהנראה לעין.
* פיתוח טכניקות לזיהוי שגיאות-הערכה: כדי שהשימוש בקירובים הללו יהיה ישים בצורה יחסית אוטומטית, ולא ידרוש בקרת מתכנת צמודה, נרצה למצוא שיטה שמבלי לדעת את הערך המדויק של הנגזרת תדע לוודא שהקירוב שחישבנו אינו שגוי מהותית (כגון בנגזרת החמישית כשערכי קושי עבור N=4 היו שגויים מאוד). בעיה זו רלוונטית בכל-מקרה גם בבעיה הממשית המקורית, אך בהרחבה למישור המרוכב אנו מכניסים תוספת-סיכון (למשל בפונקציה שהצגנו עם הקוטב ב- ).
* העמקת ההשוואה בין השיטות שהצגתי בעבודה זו: המשך זה לא יכיל חידוש, אך בחינת יותר פונקציות ביותר נקודות, ועם מגוון פרמטרים עשוי לחדד את ההשוואה בין השיטות ולהציף מקרים שלא נתקלנו בהם, שישפכו אור שונה/נוסף על היחס האיכותי בין השיטות.
* ניתוח מתמטי תיאורטי סדור יותר: ניתן לפתח את הניתוח למגוון כיוונים, ביניהם:
  + ניסיון לוותר על הצורך באנליטיות פונקציית ההרחבה המרוכבת, בתנאי שמובטחת הגזירות הממשית, אותה אנו מחפשים.
  + שיפור חסמי ההערכה שנתנו לשיטת קושי, משום שהנחנו בניתוח שמדובר על פונקציה מרוכבת כללית אך בפועל מדובר בפונקציה ממשית שהורחבה אז ייתכן שתכונותיה "טובות" יותר.
  + חיפוש שיטה שלישית שאינה מתבססת על טורי טיילור או אינטגרלי קושי, והשוואתה לאחרות.
  + הכללת השימוש בשיטות מרוכבות לנגזרות חלקיות של פונקציות מרובות משתנים.

1. ביבליוגרפיה
2. Fornberg, B. (1981). **Numerical Differentiation of Analytic Functions**. ACM Transactions on Mathematical Software, 7(4), 512-526.
3. Homescu, C. (2011). **Generic Computing Alternatives for Better Greeks**. Social Science Research Network, abstract\_id=1921085.[[25]](#footnote-25)
4. Lyness, J. N., and Moler, C. B. (1967). **Numerical Differentiation of Analytic Functions**. SIAM Journal on Numerical Analysis, 4(2), 202-210.
5. Squire, W., and Trapp, G. (1998). **Using Complex Variables to Estimate Derivatives of Real Functions**. SIAM Review, 40(1), 110-112.
6. Trefethen, L. N., and Weideman, J. A. C. (2014). **The Exponentially Convergent Trapezoidal Rule**. SIAM Review, 56(3), 385-458.
7. נספח: קוד התוכנית
   1. seminar.h

#include <complex>

using std::complex;

typedef complex<double> DBL\_COMPLEX;

enum Method {TAYLOR\_REAL,TAYLOR\_COMPLEX1,TAYLOR\_COMPLEX2,CAUCHY};

//basic mathematics that <cmath> doesn't include

double factorial(int n); //factorial returns double because 'int' will overflow too quickly (factorial(12) is fine, factorial(13) overflows).

//methods for evaluating derivations

double cauchy\_approx(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h, int n, int N\_samples); ////any derivative, CAUCHY's type integral.

double taylorR\_prime0(double(\*func)(double), double x, double h); //removable-singulatiry

double taylorR\_prime1(double(\*func)(double), double x, double h); //first derivative, taylor-series, real.

double taylorC\_prime1\_ver1(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h); //first derivative, taylor-series, complex.

double taylorR\_prime2(double(\*func)(double), double x, double h); //second derivative, taylor-series, real.

double taylorC\_prime2\_ver1(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real); //second derivative, taylor-series, complex. version1 is quicker version.

double taylorC\_prime2\_ver2(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real); //second derivative, taylor-series, complex. version2 should slightly improve the truncation-error.

double taylorR\_prime3(double(\*func)(double), double x, double h); //fifth derivative, taylor-series, real.

double taylorC\_prime3\_ver1(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real); //third derivative, taylor-series, complex. version1 is quicker version.

double taylorC\_prime3\_ver2(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real); //third derivative, taylor-series, complex. version2 should slightly improve the truncation-error.

double taylorR\_prime5(double(\*func)(double), double x, double h); //fifth derivative, taylor-series, real.

double taylorC\_prime5\_ver1(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real); //fifth derivative, taylor-series, complex. version1 is quicker version.

double taylorC\_prime5\_ver2(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real); //fifth derivative, taylor-series, complex. version2 should slightly improve the truncation-error.

//function used for applying the calculations and reporting it to screen

void single\_case(double x, double h, int deg, enum Method method, int N\_CAUCHY, double (\*f\_real)(double), DBL\_COMPLEX (\*f\_complex)(DBL\_COMPLEX), double exact\_value); //single test. output to screen and temp-file.

//########################################################################################

// ### Following are function templates, to allow easy real/complex "code-duplication" ###

// ### note: the names correlate to the names used in the project's paper. ###

//########################################################################################

template <class T> T F\_FUNC(T x){

T c = cos(x);

T s = sin(x);

return exp(x)/(c\*c\*c+s\*s\*s);

}

template <class T> T F\_prime2(T x){

T c = cos(x);

T s = sin(x);

T s2 = 2\*c\*s; //sin(2\*x);

T a\_inv = 1./(c\*c\*c+s\*s\*s);

T e = exp(x);

T temp;

temp = 4.5 \* s2 \* (1-s2) \* a\_inv;

temp = (-6 \* s + temp) \* s2 \* a\_inv;

return e \* a\_inv \* (4 + temp);

}

template <class T> T G\_FUNC(T x){

return 1./(x\*x+1.);

}

template <class T> T G\_prime(T x){

T temp = x\*x + 1.;

return -2.\*x/(temp\*temp);

}

template <class T> T H\_FUNC(T x){

return (exp(x)-1.-x)/pow(x,2);

}

template <class T> T H\_prime2(T x){

T x2 = x\*x;

if (abs(x) < 0.000001){ //account for small-x: approximate by taylor 2nd order

return (5. + x \* (3. \* +x) )/60.;

} else { //apply "naive" formula

return (exp(x)\*(x2-4.\*x+6.)-(2.\*x+6.))/(x2\*x2);

}

}

template <class T> T F2\_FUNC(T x){

return sin(x);

}

template <class T> T F3\_FUNC(T x){

return log(x);

}

* 1. seminar.cpp

#define \_USE\_MATH\_DEFINES //for some reasonm M\_PI is undefined if this defintion and include come after "seminar.h"

#include <cmath>

#include "seminar.h"

#include <iostream>

using std::cout;

using std::endl;

#include <sys/timeb.h>

#define M\_SQ\_3 1.7320508075688772 //not originally defined in the math constants

const DBL\_COMPLEX I(0,1); //the square root of -1

const DBL\_COMPLEX J(M\_SQRT1\_2,M\_SQRT1\_2); //the square root of I

const DBL\_COMPLEX TwoPiI = 2.\*M\_PI\*I; //2\*pi\*i, another useful constant

FILE\* output = fopen("c:\\temp\\output.txt","wb+"); //temporary file to output the strings (to copy into my document)

void single\_case(double x, double h, int deg, enum Method method, int N\_cauchy, double (\*f\_real)(double), DBL\_COMPLEX (\*f\_complex)(DBL\_COMPLEX), double exact\_value){

//this functions applies a single test-case of approximation given its parameters: value, step size, method, and function for evaluations.

//It requires also 'N' if the method is Cauchy's, and exact\_value to evaluate the error.

//useful variables

int time\_diff, calc\_batch, n\_digits;

double approx,val\_diff;

struct timeb tmb1,tmb2; //timers to measure time

//interface function names

double (\*taylorR)( double (\*)(double), double,double);

double (\*taylorC1)( DBL\_COMPLEX (\*)(DBL\_COMPLEX), double,double);

double (\*taylorC2)( DBL\_COMPLEX (\*)(DBL\_COMPLEX), double,double);

//first, set the interface function names to the correct function, based on the chosen case

switch (deg) {

case 0:

if (method == TAYLOR\_REAL || method == CAUCHY){ taylorR = taylorR\_prime0; }

break;

case 1:

taylorR = taylorR\_prime1;

taylorC1 = taylorC\_prime1\_ver1;

taylorC2 = taylorC\_prime1\_ver1; //no second version in the case of first derivative

break;

case 2:

taylorR = taylorR\_prime2;

taylorC1 = taylorC\_prime2\_ver1;

taylorC2 = taylorC\_prime2\_ver2;

break;

case 3:

taylorR = taylorR\_prime3;

taylorC1 = taylorC\_prime3\_ver1;

taylorC2 = taylorC\_prime3\_ver2;

break;

case 5:

taylorR = taylorR\_prime5;

taylorC1 = taylorC\_prime5\_ver1;

taylorC2 = taylorC\_prime5\_ver2;

break;

default:

if (method != CAUCHY) {

printf("usupported %d-derivative in taylor methods.\n",deg);

fprintf(output,"usupported %d-derivative in taylor methods.\n",deg);

return; //unsupported number for specific-derivative calculation not using cauchy's method

}

}

//then, calculate the approximation and measure its time:

time\_diff = 0; calc\_batch = 1;

while (time\_diff < 100){ //re-calculate until we get a noticable time, to have a correct estimation of the calculation-time

calc\_batch \*= 10; //increase exponentially the batch until we get a non-negligible time to measure

switch (method){

case TAYLOR\_REAL:

if (calc\_batch == 10) { printf("real taylor: "); } //print only on the first time

ftime(&tmb1);

for(int i=0;i<calc\_batch;i++) {

approx = taylorR(f\_real, x, h);

}

ftime(&tmb2);

break;

case TAYLOR\_COMPLEX1:

if (calc\_batch == 10) { printf("complex qck: "); } //print only on the first time

ftime(&tmb1);

for(int i=0;i<calc\_batch;i++) {

approx = taylorC1(f\_complex, x, h);

}

ftime(&tmb2);

break;

case TAYLOR\_COMPLEX2:

if (calc\_batch == 10) { printf("complex acc: "); } //print only on the first time

ftime(&tmb1);

for(int i=0;i<calc\_batch;i++) {

approx = taylorC2(f\_complex, x, h);

}

ftime(&tmb2);

break;

case CAUCHY:

if (calc\_batch == 10) { printf("cauchy N=%2d: ",N\_cauchy); } //print only on the first time

ftime(&tmb1);

for(int i=0;i<calc\_batch;i++) {

approx = cauchy\_approx(f\_complex, x, h, deg, N\_cauchy);

}

ftime(&tmb2);

break;

}

time\_diff = (tmb2.time - tmb1.time)\*1000 + (tmb2.millitm-tmb1.millitm);

}

val\_diff = abs(approx-exact\_value);

//following are two extra modes to evaluate the error, which I used a little as well"

val\_diff = (exact\_value==0 ? abs(approx-exact\_value) : abs(approx-exact\_value)/abs(exact\_value)); //calculates the relative error, or the absolute error if the exact value is 0.

//val\_diff \*= (pow(h,deg) / factorial(deg)); //calculates "err \* h^n / n!", which should be about constant per n, where n in the number of the derivative

//lastly, report the results: value, error, time:

if (val\_diff == 0){

fprintf(output,"%3.15lf\t0\t%3.3lf\n",approx, 1000.\*time\_diff/calc\_batch);

printf("val = %3.15lf , %3.3lf(ms/single) , err = 0\n",approx, 1000.\*time\_diff/calc\_batch);

} else {

n\_digits = 0;

if (val\_diff < 10) {

while (val\_diff < 1) {

val\_diff \*= 10;

n\_digits --;

}

} else {

while (val\_diff >=10) {

val\_diff /= 10;

n\_digits ++;

}

}

if (n\_digits < 0){

fprintf(output,"%3.15lf\t%1.2lfE%d\t%3.3lf\n",approx, val\_diff, n\_digits, 1000.\*time\_diff/calc\_batch);

printf("val = %3.15lf , %3.3lf(ms/single) , err = %1.2lfE%d\n",approx, 1000.\*time\_diff/calc\_batch,val\_diff,n\_digits);

} else {

fprintf(output,"%3.15lf\t%1.2lfE+%d\t%3.3lf\n",approx, val\_diff, n\_digits, 1000.\*time\_diff/calc\_batch);

printf("val = %3.15lf , %3.3lf(ms/single) , err = %1.2lfE+%d\n",approx, 1000.\*time\_diff/calc\_batch,val\_diff,n\_digits);

}

}

}

int main(){

//input order to 'single\_case' function: x, h, deg, method, N\_cauchy, dbl->dbl, complex->complex, exact-value

double x,y,exact\_value,h,N;

int deg,MAX\_DEG;

double x\_array1[] = {-M\_PI, -0.5, M\_PI/4., M\_PI/2.};

double x\_array2[] = {-M\_PI, -0.5, M\_PI/4., 0};

double exact\_array2\_h[] = {0.00252798, 0.0164074, 0.0431807, 0.0238095238095};

double exact\_array2\_f[] = {7.087082595258652, -47711.3, 1352.3701477125185, -164.0000000};

struct timeb time\_start,time\_end;

ftime(&time\_start);

//################################

//### chapter FIRST DERIVATIVE ###

//################################

//Table1 in chapter "compare=>first derivative"

for(double h=1.0; h>=1E-10; h/=10){

single\_case(0.0, h, 1, TAYLOR\_REAL, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, 1.0);

} for(double h=1.0; h>=1E-10; h/=10){

single\_case(0.0, h, 1, TAYLOR\_COMPLEX1, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, 1.0);

} printf("\n");

//Table2 in chapter "compare=>first derivative"

for(double h=1.0; h>=1E-10; h/=10){

single\_case(M\_PI/4., h, 1, TAYLOR\_REAL, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, 3.1017663938360513); //M\_SQRT2 \* exp(0.25\*M\_PI)

} for(double h=1.0; h>=1E-10; h/=10){

single\_case(M\_PI/4., h, 1, TAYLOR\_COMPLEX1, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, 3.1017663938360513); //M\_SQRT2 \* exp(0.25\*M\_PI)

} printf("\n");

//Table3 in chapter "compare=>first derivative"

x = 0.01;

for(double h=1.0; h>=1E-10; h/=10){

single\_case(x, h, 1, TAYLOR\_REAL, 0, G\_FUNC<double>, G\_FUNC<DBL\_COMPLEX>, G\_prime<double>(x));

} for(double h=1.0; h>=1E-10; h/=10){

single\_case(x, h, 1, TAYLOR\_COMPLEX1, 0, G\_FUNC<double>, G\_FUNC<DBL\_COMPLEX>, G\_prime<double>(x));

} printf("\n");

//this is not a table, but mentioned in words right after table5:

x = 0.0;

for(double h=1.0; h>=1E-10; h/=100){

single\_case(x, h, 1, TAYLOR\_REAL, 0, G\_FUNC<double>, G\_FUNC<DBL\_COMPLEX>, G\_prime<double>(x));

} for(double h=1.0; h>=1E-10; h/=10){

single\_case(x, h, 1, TAYLOR\_COMPLEX1, 0, G\_FUNC<double>, G\_FUNC<DBL\_COMPLEX>, G\_prime<double>(x));

} printf("\n");

//######################################

//### chapter REMOVEABLE SINGULARITY ###

//######################################

//Table1 in chapter "compare=>removable singularity"

for(double h=0.01; h>=1E-8; h/=10){

single\_case(0.0, h, 0, TAYLOR\_REAL, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 0.5);

} for(double h=0.01; h>=1E-8; h/=10){

single\_case(0.0, h, 0, CAUCHY, 4, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 0.5);

} printf("\n");

//Table2 in chapter "compare=>removable singularity"

for(int N=4; N<=32;N\*=2){

for(double h=0.1; h>=1E-6; h/=10){

single\_case(0.0, h, 0, CAUCHY, N, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 0.5);

}

} printf("\n");

//#################################

//### chapter SECOND DERIVATIVE ###

//#################################

//Table1 in chapter "compare=>second derivative"

for(double h=1.0; h>=1E-8; h/=10){

single\_case(0.0, h, 2, TAYLOR\_REAL, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, 4.0);

} for(double h=1.0; h>=1E-8; h/=10){

single\_case(0.0, h, 2, TAYLOR\_COMPLEX1, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, 4.0);

} printf("\n");

//Table2 in chapter "compare=>second derivative"

for (int n=4;n<=32;n\*=2){

for(double h=1.0; h>=1E-5; h/=10){

single\_case(0.0, h, 2, CAUCHY, n, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, 4);

}

} printf("\n");

//Table3 in chapter "compare=>second derivative"

for(int i=0;i<4;i++){

x = x\_array1[i];

exact\_value = F\_prime2<double>(x); fprintf(output,"%3.16lf\n",exact\_value);

single\_case(x, 1E-4, 2, TAYLOR\_REAL, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, exact\_value);

single\_case(x, 1E-3, 2, TAYLOR\_COMPLEX1, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, exact\_value);

single\_case(x, 1E-1, 2, CAUCHY, 32, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, exact\_value);

single\_case(x, 1E-2, 2, CAUCHY, 8, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, exact\_value);

} printf("\n");

//Table4 in chapter "compare=>second derivative"

for(int i=0;i<4;i++){

x = x\_array1[i];

exact\_value = H\_prime2<double>(x); fprintf(output,"%3.16lf\n",exact\_value);

single\_case(x, 1E-4, 2, TAYLOR\_REAL, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, exact\_value);

single\_case(x, 1E-3, 2, TAYLOR\_COMPLEX1, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, exact\_value);

single\_case(x, 1E-1, 2, CAUCHY, 32, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, exact\_value);

single\_case(x, 1E-2, 2, CAUCHY, 8, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, exact\_value);

} printf("\n");

//################################

//### chapter FIFTH DERIVATIVE ###

//################################

//Table1 in chapter "compare=>fifth derivative"

x = 1.0;

for(double h=1.0; h>=1E-5; h/=10){

single\_case(x, h, 5, TAYLOR\_REAL, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 0.0509150061550372); //840 - 109\*e

} for(double h=1.0; h>=1E-5; h/=10){

single\_case(x, h, 5, TAYLOR\_COMPLEX1, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 0.0509150061550372); //840 - 109\*e

} for (int n=4;n<=32;n\*=2){

for(double h=1.0; h>=1E-5; h/=10){

single\_case(x, h, 5, CAUCHY, n, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 0.0509150061550372); //840 - 109\*e

}

} printf("\n");

//To support the comment after Table1 regarding that 'N' (number of points) must be greater than 'n' (the degree of the derivative)

for (int n=4;n<=8;n++){

for(double h=0.1; h>=1E-3; h/=10){

single\_case(1.0, h, 5, CAUCHY, n, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 0.0509150061550372); //840 - 109\*e

}

} printf("\n");

//Table2 in chapter "compare=>fifth derivative"

for(int i=0;i<4;i++){

x = x\_array2[i]; y = exact\_array2\_h[i];

single\_case(x, 1E-2, 5, TAYLOR\_REAL, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, y);

single\_case(x, 1E-1, 5, TAYLOR\_COMPLEX1, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, y);

single\_case(x, 1E-1, 5, CAUCHY, 8, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, y);

} printf("\n");

//Table3 in chapter "compare=>fifth derivative"

for(int i=0;i<4;i++){

x = x\_array2[i]; y = exact\_array2\_f[i];

single\_case(x, 1E-2, 5, TAYLOR\_REAL, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, y);

single\_case(x, 1E-1, 5, TAYLOR\_COMPLEX1, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, y);

single\_case(x, 1E-1, 5, CAUCHY, 8, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, y);

} printf("\n");

//Table4 in chapter "compare=>fifth derivative"

for(int i=0;i<4;i++){

x = x\_array2[i]; y = exact\_array2\_f[i];

single\_case(x, 1E-1, 5, CAUCHY, 8, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, y);

single\_case(x, 1E-1, 5, CAUCHY, 32, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, y);

single\_case(x, 1E-1, 5, CAUCHY, 64, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, y);

} printf("\n");

//Table5 in chapter "compare=>fifth derivative"

for (h=0.1; h>=1E-2; h/=10){

for(int i=0;i<4;i++){

x = x\_array2[i];

exact\_value = H\_prime2<double>(x);

single\_case(x, h, 5, TAYLOR\_COMPLEX1, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, exact\_value);

single\_case(x, h, 5, TAYLOR\_COMPLEX2, 0, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, exact\_value);

} for(int i=0;i<4;i++){

x = x\_array2[i];

exact\_value = F\_prime2<double>(x);

single\_case(x, h, 5, TAYLOR\_COMPLEX1, 0, F\_FUNC<double>, F\_FUNC<DBL\_COMPLEX>, exact\_value);

single\_case(x, h, 5, TAYLOR\_COMPLEX2, 0, F\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, exact\_value);

}

} printf("\n");

//###################################

//### chapter HIGHEST DERIVATIVES ###

//###################################

//Table1 in chapter "compare=>higher-order derivatives"

for (int deg=1;deg<10;deg++){

single\_case(0.0, 0.1, deg, CAUCHY, 16, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 1./((deg+1)\*(deg+2)));

} printf("\n");

//Table2 in chapter "compare=>higher-order derivatives"

for (int deg=1;deg<12;deg++){

exact\_value = 0.5; //assume we check x=PI/3

if (deg%2 == 0) exact\_value \*= M\_SQ\_3; //decide sin(x) or cos(x)

if (deg & 2) exact\_value = -exact\_value; //decide sign of the derivative

single\_case(M\_PI/3., 0.1, deg, CAUCHY, 16, F2\_FUNC<double>, F2\_FUNC<DBL\_COMPLEX>, exact\_value);

} printf("\n");

//Table3 in chapter "compare=>higher-order derivatives"

for (int deg=1;deg<16;deg++){

if (deg == 0) {

exact\_value = 0;

} else {

exact\_value = factorial(deg-1);

if (deg%2 == 0) exact\_value = -exact\_value;

}

single\_case(1.0, 0.1, deg, CAUCHY, 16, F3\_FUNC<double>, F3\_FUNC<DBL\_COMPLEX>, exact\_value);

}

//Table4 in chapter "compare=>higher-order derivatives"

h=1.0; N=16;

deg = 1; while (deg<17){

single\_case(0.0, h, deg, CAUCHY, N, H\_FUNC<double>, H\_FUNC<DBL\_COMPLEX>, 1./((deg+1)\*(deg+2)));

if (deg > 10) deg ++; else deg +=2;

}

deg = 1; while (deg<17){

exact\_value = 0.5; //assume we check x=PI/3

if (deg%2 == 0) exact\_value \*= M\_SQ\_3; //decide sin(x) or cos(x)

if (deg & 2) exact\_value = -exact\_value; //decide sign of the derivative

single\_case(M\_PI/3., h, deg, CAUCHY, N, F2\_FUNC<double>, F2\_FUNC<DBL\_COMPLEX>, exact\_value);

if (deg > 10) deg ++; else deg +=2;

}

/\*\*/

//######################################################

//### finish: print the total time, and wait to exit ###

//######################################################

ftime(&time\_end);

printf("### total run time: %d(s) ###\n",time\_end.time - time\_start.time);

flushall(); getchar();

fclose(output);

return 0;

}

double factorial(int n){ //factorial returns double because 'int' will overflow too quickly (factorial(12) is fine, factorial(13) overflows).

if (n<1) return 1;

double res = 1;

for (int k=1;k<=n;k++){ res \*= k; }

return res;

}

double cauchy\_approx(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h, int n, int N\_samples){ //any derivative, CAUCHY's type integral.

DBL\_COMPLEX w = exp(TwoPiI/(double)N\_samples); //unity-root of degree 'N\_samples'

DBL\_COMPLEX inv\_w\_n = pow(w,-n);

DBL\_COMPLEX z = 1; //initialize to the first unity-root

DBL\_COMPLEX inv\_z\_n = 1;

DBL\_COMPLEX total = 0;

for (int k=0; k<N\_samples; k++){

total += (\*func)(x+h\*z) \* inv\_z\_n; //calculate the function's value nearby (on a near circle) and add it to the total

z \*= w; //go to the next unity-root

inv\_z\_n \*= inv\_w\_n; //update the denominator as well

}

total \*= factorial(n);

total /= ((double)N\_samples \* pow(h,n));

return real(total);

}

//#############################

//### Removable Singularity ###

//#############################

double taylorR\_prime0(double(\*func)(double), double x, double h){

return 0.5\*( (\*func)(x+h) + (\*func)(x-h) );

}

//######################

//### 1st Derivative ###

//######################

double taylorR\_prime1(double(\*func)(double), double x, double h){

return ((\*func)(x+h) - (\*func)(x-h))/(2\*h);

}

double taylorC\_prime1\_ver1(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h){

return imag( (\*func)(x+I\*h) / h );

}

//######################

//### 2nd Derivative ###

//######################

double taylorR\_prime2(double(\*func)(double), double x, double h){

double temp = (\*func)(x+h) + (\*func)(x-h) - 2. \* (\*func)(x);

return temp/(h\*h);

}

double taylorC\_prime2\_ver1(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real){

DBL\_COMPLEX h = J\*h\_real;

DBL\_COMPLEX temp = (\*func)(x+h) + (\*func)(x-h);

return imag(temp)/(h\_real\*h\_real);

}

double taylorC\_prime2\_ver2(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real){

DBL\_COMPLEX temp = (\*func)(x) - (\*func)(x+I\*h\_real);

return 2.\*real(temp)/(h\_real\*h\_real);

}

//######################

//### 3rd Derivative ###

//######################

double taylorR\_prime3(double(\*func)(double), double x, double h){

double A\_h = (\*func)(x+h) - (\*func)(x-h);

double A\_2h = (\*func)(x+2.\*h) - (\*func)(x-2.\*h);

return (A\_2h - 2.\*A\_h)/(2.\*pow(h,3));

}

double taylorC\_prime3\_ver1(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real){

DBL\_COMPLEX h = J\*h\_real;

DBL\_COMPLEX A\_h = (\*func)(x+h) - (\*func)(x-h);

DBL\_COMPLEX temp = 3.\*A\_h/pow(h,3);

return real(temp);

}

double taylorC\_prime3\_ver2(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real){

DBL\_COMPLEX h = J\*h\_real;

DBL\_COMPLEX A\_h = (\*func)(x+h) - (\*func)(x-h);

DBL\_COMPLEX A\_2h = (\*func)(x+2.\*h) - (\*func)(x-2.\*h);

DBL\_COMPLEX temp = (A\_2h - 2.\*A\_h)/(2.\*pow(h,3));

return real(temp);

}

//######################

//### 5th Derivative ###

//######################

double taylorR\_prime5(double(\*func)(double), double x, double h){

double A\_h = (\*func)(x+h) - (\*func)(x-h);

double A\_2h = (\*func)(x+2.\*h) - (\*func)(x-2.\*h);

double A\_4h = (\*func)(x+4.\*h) - (\*func)(x-4.\*h);

return (A\_4h - 10.\*A\_2h + 16.\*A\_h)/(12.\*pow(h,5));

}

double taylorC\_prime5\_ver1(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real){

DBL\_COMPLEX h = J\*h\_real;

DBL\_COMPLEX A\_h = (\*func)(x+h) - (\*func)(x-h);

DBL\_COMPLEX A\_2h = (\*func)(x+2.\*h) - (\*func)(x-2.\*h);

DBL\_COMPLEX temp = (A\_2h - 2.\*A\_h)\*2./pow(h,5);

return real(temp);

}

double taylorC\_prime5\_ver2(DBL\_COMPLEX(\*func)(DBL\_COMPLEX), double x, double h\_real){

DBL\_COMPLEX h = J\*h\_real;

DBL\_COMPLEX A\_h = (\*func)(x+h) - (\*func)(x-h);

DBL\_COMPLEX A\_2h = (\*func)(x+2.\*h) - (\*func)(x-2.\*h);

DBL\_COMPLEX A\_4h = (\*func)(x+4.\*h) - (\*func)(x-4.\*h);

DBL\_COMPLEX temp = (A\_4h - 10.\*A\_2h + 16.\*A\_h)/(12.\*pow(h,5));

return real(temp);

}

1. נבהיר שאיננו מבטל כליל את שגיאות העיגול בחישובים לצורך הערכת ערך הפונקציה בנקודה נתונה. [↑](#footnote-ref-1)
2. טור טיילור סביב הנקודה x=0 מכונה גם טור מקלורן. [↑](#footnote-ref-2)
3. אם h<0 אז מדובר על הקטע (a+h,a). לא נתייחס לדקות זו בהמשך ונניח שהכוונה ברורה. [↑](#footnote-ref-3)
4. לדוגמה, הפונקציה יחד עם ההשלמה f(0)=0 רציפה וגזירה אינסוף פעמים, אך הטור שלה ב-0 הוא פונקציית האפס ואינו מתכנס ל-f(x) אלא בנקודה זו בלבד. [↑](#footnote-ref-4)
5. ניתן לתהות כיצד קיימת סינגולריות אם יש אפס איברים בטור הסינגולרי. התשובה היא שייתכן שהפונקציה המקורית אינה מוגדרת בנקודה, כגון f(x)=x/x. ברור שהיא שקולה ל- f(x)=1, אך בנקודה x=0 הפונקציה המקורית אינה מוגדרת. [↑](#footnote-ref-5)
6. בהמשך העבודה, נסמן ב-Mn את החסם לנגזרת ה-n בקטע הרלוונטי. [↑](#footnote-ref-6)
7. כפי שמופיע גם במאמר [4] [↑](#footnote-ref-7)
8. למעשה, שגיאת הקיטוע צריכה להיות סכום של שני ביטויים O(), אך הכפל בקבוע 2 נבלע בסימון ה-O. וכך גם בפיתוחים שבהמשך. [↑](#footnote-ref-8)
9. שיטה זו דומה לשיטה המוצגת במאמר [2]. [↑](#footnote-ref-9)
10. תת-פרק זה מבוסס על האמור בפרק 2 במאמר [5], בנוגע לאינטגרציה על מעגל במישור המרוכב. [↑](#footnote-ref-10)
11. דיון דומה ניתן למצוא ב-[5], תחילת פרק 12, אך הניתוחים מעט שונים. [↑](#footnote-ref-11)
12. החסם M שכאן מיד נוגע לפונקציה שמופיעה באינטגרנד [↑](#footnote-ref-12)
13. כמודגם ב-[5], פרק 12, עמוד 422. [↑](#footnote-ref-13)
14. מכיוון שחישוב בודד יהיה מהיר מכדי למדוד, נבצע מספר רב של חזרות לקבלת זמן מורגש אותו נמצע. [↑](#footnote-ref-14)
15. הבחירה אינה מקרית, פונקציה זו משמשת לדוגמאות גם במאמרים [1], [3], [4], [5]. [↑](#footnote-ref-15)
16. במאמר [4] ניתן לראות השוואה דומה עבור x=1.5. [↑](#footnote-ref-16)
17. פונקציה זו לקוחה מ-[5], פרק 12, משוואה 12.7. [↑](#footnote-ref-17)
18. לא נעמיק במושג "נקודת הסתעפות", אך נציין שזה סוג בעייתי נוסף של נקודות, ונעיר לדוגמה שבהרחבה האנליטית של ln(x), הנקודה z=0 היא נקודת הסתעפות ואינה סינגולריות "רגילה" (עיקרית/קוטב). [↑](#footnote-ref-18)
19. וכזכור, לקוחות מהמאמרים שבהפניות של עבודה זו. [↑](#footnote-ref-19)
20. ביטוי מפורש עבור f''(x), כאשר [↑](#footnote-ref-20)
21. שם מקוצר לשיטות טיילור: R עבור Real, C עבור Complex [↑](#footnote-ref-21)
22. למשל, ב-wolfram alpha ניתן להכניס את הקלט הבא: "D[(-1 + E^x - x)/x^2, {x, 5}] at x=1" [↑](#footnote-ref-22)
23. אחרי מהירות ותאוצה קיימים גם המושגים נתירה (jerk, נגזרת שלישית), ואחריו יש גם מושגי המשך באנגלית לנגזרות הבאות: jounce (רביעית), snap (חמישית), crackle (שישית), pop (שביעית). [↑](#footnote-ref-23)
24. דוגמאות: סדר ראשון – חוק הקירור של ניוטון, נוסחאות דעיכה רדיואקטיבית, גידול אוכלוסין ; סדר שני – משוואות תנועה הרמונית, משוואת החום, משוואת פואסון ; סדר רביעי – המשוואה הבי-הרמונית. [↑](#footnote-ref-24)
25. קישור למאמר ב-SSRN: <http://ssrn.com/abstract=1921085> [↑](#footnote-ref-25)