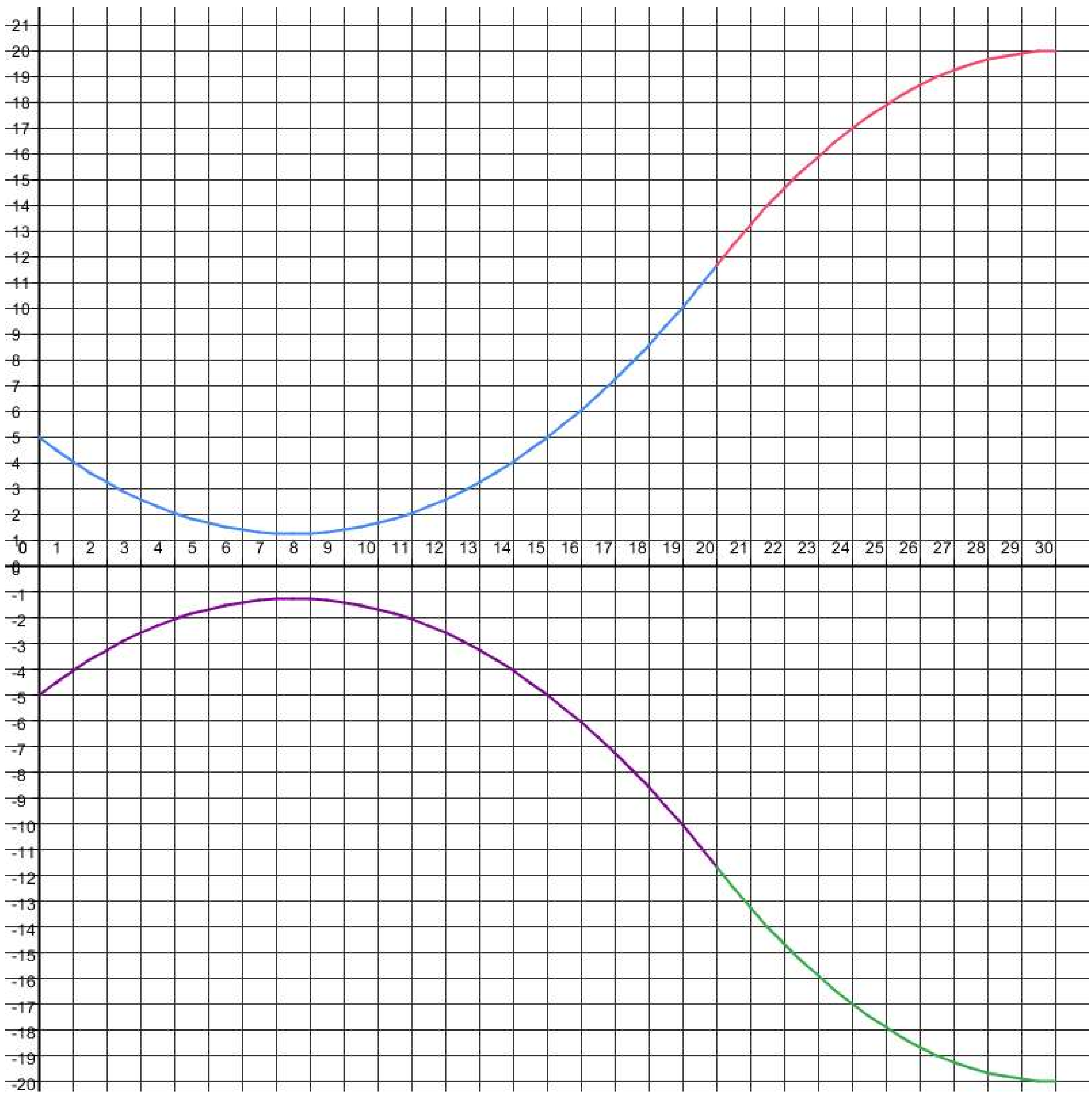
Differentiering

Navn: Oleksandr

Klasse: 2d1

Studieretning: Mat/Prog

Opgaveformulering: Hvorledes er det muligt at benytte numeriske metoder til numerisk integration af differentialligninger?



Indholdsfortegnelse

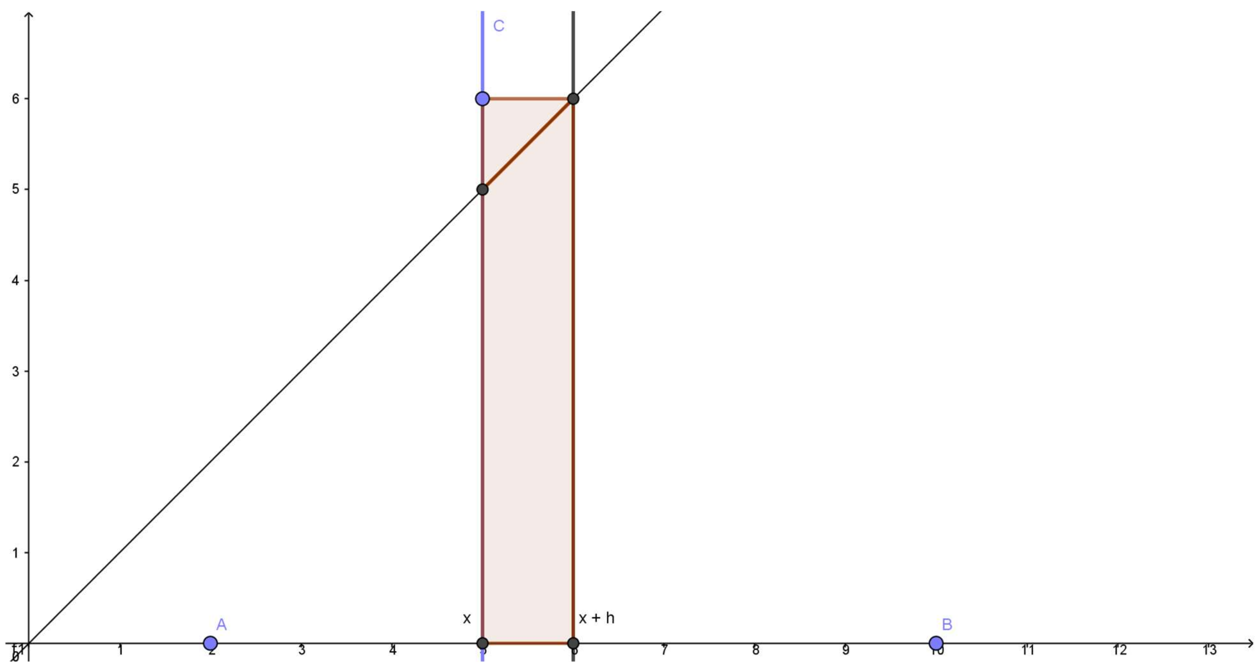
Indledning

Problemformulering
Hvordan kan man numerisk integrere en funktion og hvad er sammenhænget mellem præcision og hastighed i disse metoder.

Teoretiske del
Matematik
Ofte i matematik er der brug for at vide hvad arealet under en funktion er, fx hvis man skal beregne mellem acceleration til position hvis man har kurven for acceleration til et objekt. Udregningen af dette areal kalder man for integration og den typiske måde at integrere noget på er analytisk.

Når integrering sker analytisk, bliver det gjort ved at tage summen arealer af små stykker af kurven vår at forskellen mellem x - dx går imod 0. Og ifølge Infinitesimalregningens hovedsætning er $\int_0^x f'(x) = f(x)$ med andre ord at integrering er inverst fra differentiering.

Infinitesimalregningens hovedsætning bliver bevist herunder (for en positivt stigende kurve):



Figur 1.

Til start kan man sige at $f(x) \cdot h \leq A(x)$ som er arealet ved x til x + h. Derudover kan man sige at $f(x+h) \cdot h \geq A(x)$. Samlet kan man sige at $f(x) \cdot h \leq A(x) \leq f(x+h) \cdot h$

Derefter kan man dividere alle tre udtryk med h

$$f(x) \leq \frac{A(x)}{h} \leq f(x+h)$$

Man forudsætter at h vil tilnærme sig 0, sådan. Og A(x)/h er det samme som A’(x):

$$f(x) \leq A'(x) \leq f(x+0)$$

Man kan nu sige at $A'(x) = f(x)$

Nu skal man også bevise at det kan ske den omvendte vej dvs. at arealet inden for mellem a og b af grafen vil være det samme som integralet mellem a og b. også på figur 1.

Til start kan man sige

$A(x) + k = F(x)$ (1)

$A(a) + k = F(a)$ (2)

Vi siger at ved A(a) er arealet 0, da vi måler areal fra a til b.

$0 + k = F(a)$ (3)

Derudover kan man sige at

$A(b) = F(b)$ (4)

Ud fra (4) (3) er det sandt at

$$F(b) - F(a) = A(b) + k - k$$

Som kan blive omskrevet til

$$A(b) = F(b) - F(a)$$

Med andre ord $\int_b^a f(x)\,dx = F(b) - F(a)$

Det er nu bevist at man arealet er det samme som integralet og at integral er inverst fra differentiering.

Derfor kan det være nødigt at vide hvordan man differentier for at kunne integrere. Fordi så hvis man ved en funktion kan man finde stamfunktionen ved at gøre det omvendte. Fx

Hvis man har en funktion $f(x) = 3 * x + \ 5 * x^2$ så kan man funktionen op i to da integralet af to funktioner lagt sammen er det samme som integralet af en funktion som er summen af de to funktioner. Til start kan man sige $x^2 = 2x$ men da der står 3x og ikke 2x, bliver det til $\frac{3}{2}x^2$for den anden del kan man gøre det samme $x^3 = 3x^2$ og det bliver til $\frac{3}{5}x^2$. Til sidst kan man sige at $F(x) = \frac{3}{2}x^2 + \frac{3}{5}x^2$.

Men det er ikke altid nødvendigt at bruge analytisk integrering, i stedet for at ens dx tilnærmer sig 0 kan man stoppe på et tal og udregne summen for alle disse firkanter, da på grund af computere kan man gøre det meget hurtigt og præcist, da man i anvendt matematik ikke har brug for at have perfekt præcision, og kan nøjes med et par decimaler. Der findes flere måder at udregne integraler numerisk, den mest simple er ved hjælp af rektangler, disse rektangler kan være højre eller venstre rektangler, deres højde er baseret på f(x) eller f(x + h). Desuden er der trapez-metoden, som har rektangler ligesom rektangelmetoden, men der er også tilført en trekant som har arealet $\frac{(f(x)-f(x+h))\cdot dx}{2}$, læg mærke til hvordan hvis hældningen er under 0 så vil trekantens areal være negativ og den vil fjerne fra den samlede sum af spalten.

Desuden er der ”Simpsons rule” som er en måde at tilnærme sig en kurve ved at bruge arealet af en parabel som en tilnærmelse givet 3 med et centrum og de to punkter skal være lige langt fra centrum. Dette bliver gjort med denne formel $\frac{\Delta x}{3}\left(1\cdot f(x_0)+4\cdot f(x_1)+1\cdot f(x_2)\right)$ også kaldet ”141 rule” dette kan også blive gjort flere gange i træk for højere præcision så længe at antallet af punkter er lige og over 3.

$$\frac{\Delta x}{3}\left(1\cdot f(x_0)+4\cdot f(x_1)+1\cdot f(x_2)+1\cdot f(x_2)+4\cdot f(x_3)+1\cdot f(x_4)\right)$$

$$\Updownarrow$$

$$\frac{\Delta x}{3}\left(1\cdot f(x_0)+4\cdot f(x_1)+2\cdot f(x_2)+4\cdot f(x_3)+1\cdot f(x_4)\right)$$

Dette vil lave et mønster 1 (42)n 4 1 hvor n repræsentere et antal gange som Simpsons regel skal gentages over en gang.

Programmering

Programmets klassestruktur er meget simpel, den består af programmet som en klasse og så har den 3 klasser: Function, Axes og SegFunc. Function er en klasse som indeholder en liste af alle dens koefficienter til hvert grad af polynomier, så at indekset repræsentere graden, feks. Hvis man har en liste på {0, 1, 3, 1/6} så vil funktionen være:

$$f(x)=0x^0+1x^1+3x^2+(1/6)x^3$$

Med andre ord kan man sige at på indeks 0, er det konstanten.

$$f(x)=k+p_1x^1+p_2x^2+p_3x^3$$

Derudover har den en fLDraw(); funktion som givet en Axes instans kan den tegne funktionen på aksen. Derudover er der en ”overloaded” version af funktionen der udover ”Axes” instansen tager en start og en stopværdi.

Generelt med alle disse metoder er der en afvejning mellem hastighed af beregning og dens præcision. Og man kan antage at alle disse metoders præcision vil stige forskelligt givet forskelligt mængde tid.

Derfor giver det mening at se på de forskellige metoders køretider. Typisk når man skal måle køretider så kan der være meget variation fra gang til gang da din computer ikke altid kan køre et helt program færdigt hver gang, hvis der for eksempel sker ”interrupts”. Derfor er det vigtigt at man køre programmet mange gange, for at køretiden bliver udlignet. Derudover understøtter Processing ikke et tids interval under et millisekund, men til gengæld understøtter millis(), derudover tager det langt under 1ms for at køre intdx() (rektangulær integration) i de fleste tilfælde. Derfor vælger man at køre det et stort antal gange fx 1000 gange. Og derfor hvis køretiden for et program et tusind ganger er 3ms. Så vil det tage 3 µs for programmet at køre en gang.

Ud over ”Function” er der ”Axes” klassen, som har x-/yPos værdi som betegner axernes origo, der findes x-/yMarks som er antallet af markeringer dvs den numeriske længde og højde af funktionen. Den har farver til forskellige dele af aksen, som man kan bestemme og nogle ”flags” der kan bestemme hvad skal tegnes. Derudover er der en draw() funktion som tegner den aksen og en x-/yPos funktion som giver positionen på vinduet funktionens værdi.

Derudover er der SegFunc klassen som er en form for wrapper, til hvis man har flere funktioner der er sammensat. Man kan bruge Segfunc.addFunc() til at added funktioner og dens grænser, og man kan derefter tegne dem eller integrere dem.

Generelt i programmet blev der brugt ”Double” til at have stor præcision og for at gøre det nemmere i analyse-delen at se hvilken metode der er nemmest, da man har flere decimaler at arbejde med. Og for at det kunne lade sig gøre blev man nødt til at lave sin egne pow(double, int) og abs(double) funktion.

Derudover i FunkyStuff.pde blev der implimenteret kontrollering af akserne ved hjælp af wasd og ←→↑↓ taster, ”f” og ”1”. Wasd ændrer på skaleringen af akserne, piletasterne skifter på positionen af aksen, f saver et billed i projekt mappen og 1 starter test sekvensen.

Analyse del

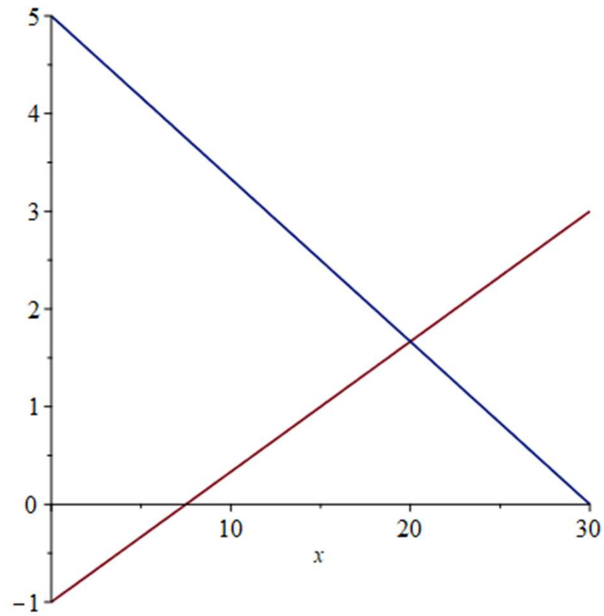
For at have noget at sammenligne blev der lavet en sammensat funktion, der skal forstille sig at være et vinglas. Det blev gjort ved at lave to lineære funktioner der skar hinanden ved $x = 20$

Den første funktion skulle have en negativ starthældning og skulle bevæge sig mod uendelig ved $x = \infty$ for at lave bunden af vinglasset og for at den bevægede sig op igen. Den anden funktion havde til gengæld en positiv starthældning og en negativ krumning for at krumme sig ind ved toppen af glasset.

$ff(x) := -1 + \frac{2\,x}{15} = x \rightarrow -1 + \frac{2}{15}\,x$

$gg(x) := -\frac{x}{6} + 5 = x \rightarrow -\frac{1}{6}\,x + 5$

$plot([\,ff(x),\, gg(x)\ ],\, x = 0\,..30)$



De to funktioner blev derefter integreret

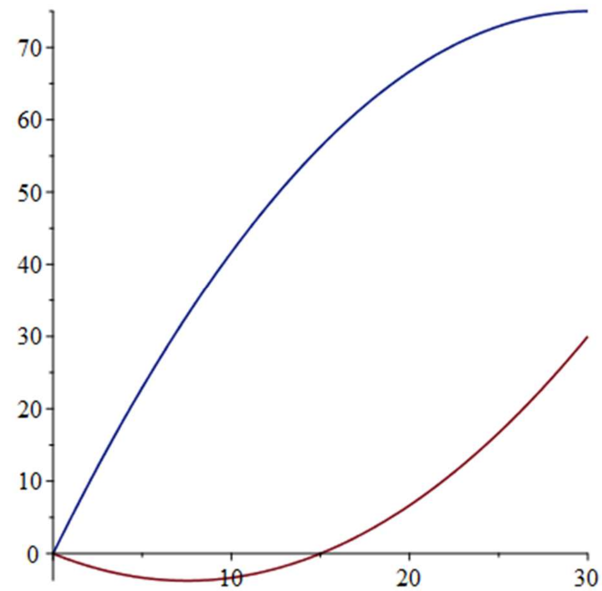
$f(x) := -x + \frac{1}{15}x^2$

$$f := x \mapsto -x + \frac{1}{15}x^2$$

$g(x) := -\frac{1}{12}x^2 + 5x$

$$g := x \mapsto -\frac{1}{12}x^2 + 5x$$

$plot([f(x), g(x)], x = 0 ..30)$



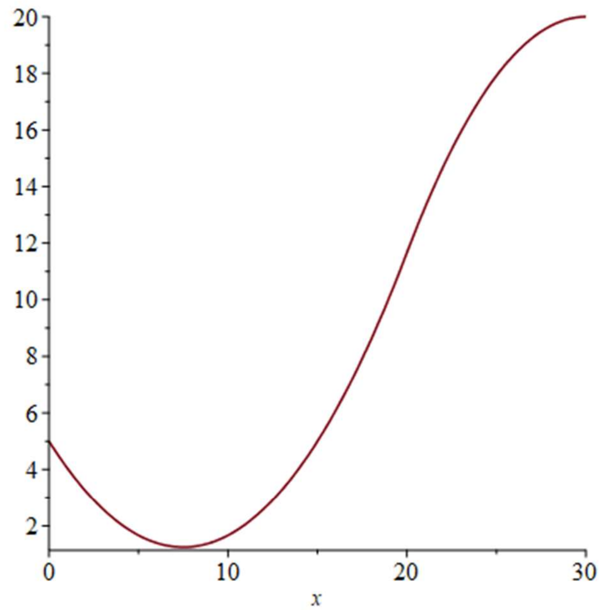
Men som man kan se selvom de to funktioner har samme hædlning ved $x = 20$ så har de forskellig y position, derudover er glassets stilk -tykkelse, som er for tyndt. Derfor blev er der først addert 5 til f(x) og derefter fandt man forskellen mellem f(20) og g(20) og det blev adderet til g(x).

Nu kan man lave det til en piecewise funktion og integrere den.

$pp(x) := piecewise\left(0 \le x \le 20, -x + \frac{1}{15}x^2 + 5, 20 < x \le 30, -\frac{1}{12}x^2 + 5x - 55\right)$

$$pp := x \mapsto \begin{cases} -x + \frac{1}{15}x^2 + 5 & 0 \le x \le 20 \\ -\frac{1}{12}x^2 + 5x - 55 & 20 < x \le 30 \end{cases}$$

$plot(pp(x), x = 0..30)$



$int(pp(x), x = 0..30)$

$$250$$

Man er kommet frem til at integralet er lige akkurat **250**. Og man har nu noget at sammenligne til ved numerisk udregning.

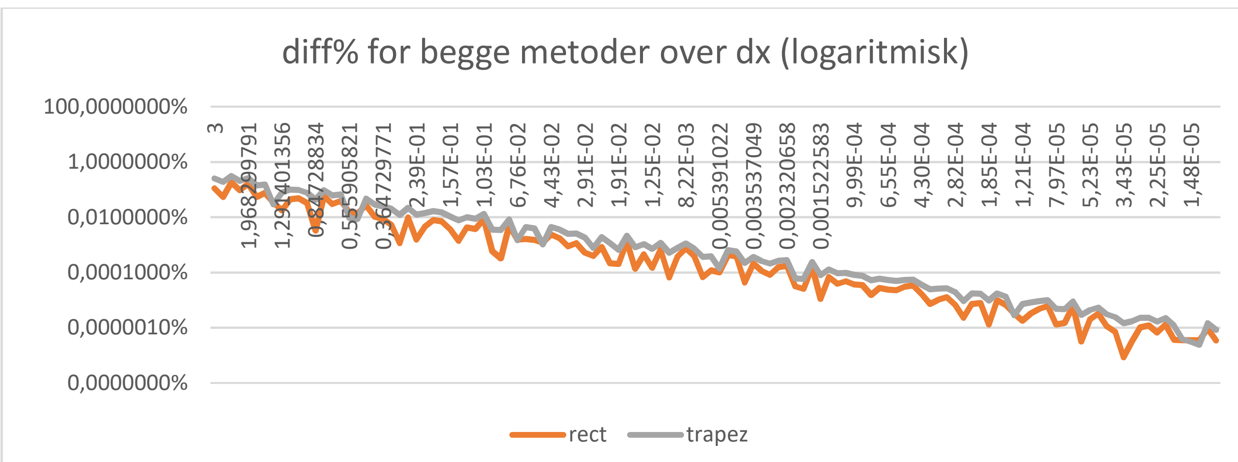
Først bliver funktionerne sat ind i den globale functions arraylist som indeholder alle funktioner. Og derefter bliver de to funktioner adderet til segFunc som er en insatns af "SegFunc" som så bliver adderet til segFuncs, som indeholder alle segmenterede funktioner.

```
functions.add(new Function(5, -1, 0.066666666666666));
functions.add(new Function(-55, 5, -0.083333333333333));
segFunc.addFunc(functions.get(0), 0, 20);
segFunc.addFunc(functions.get(1), 20, 30);
```

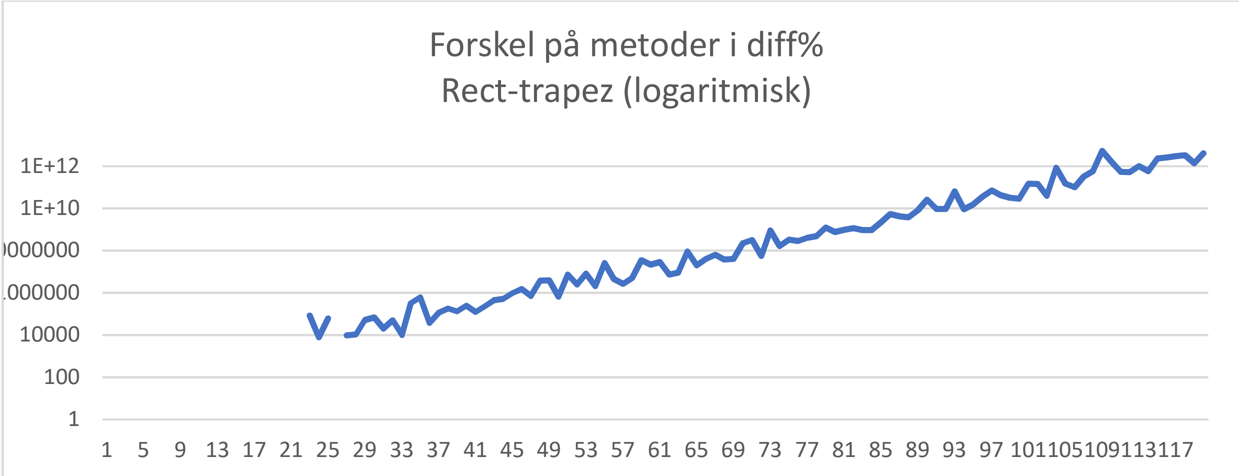
Derefter blev intDX()(rektangulære integral) og trapez()(trpezformet integral) testet ved at blive kørt 1000 (1001) gange hvorefter dens dx, integral og tid blev printet separeret med tabs for at gøre det nemt at sætte i Excel. For hver gang dette bliver gjort bliver dx ganget med 0.9 hvor at dx startede med 3, dette bliver ved med at ske indtil at dx bliver $> \frac{1}{100\,000}$ dvs. 120 gange.

```
println("trapez\t dx\t int\t time");
for (double dx = 3; dx > 1/100000.0; dx *= 0.9) {
  int start = millis();
  for (int i = 0; i < 1000; i ++) {
    segFuncs.get(0).trapez(0.0, 30.0, dx);
  }
  println("\t", dx, "\t", segFuncs.get(0).trapez(0.0, 30, dx), "\t", millis() - start);
}
println("intdx\t dx\t int\t time");
for (double dx = 3; dx > 1/100000.0; dx *= 0.9) {
  int start = millis();
  for (int i = 0; i < 1000; i ++) {
    segFuncs.get(0).intDX(0.0, 30.0, dx);
  }
  println("\t", dx, "\t", segFuncs.get(0).intDX(0.0, 30.0, dx), "\t", millis() - start);
}
println("done");
```

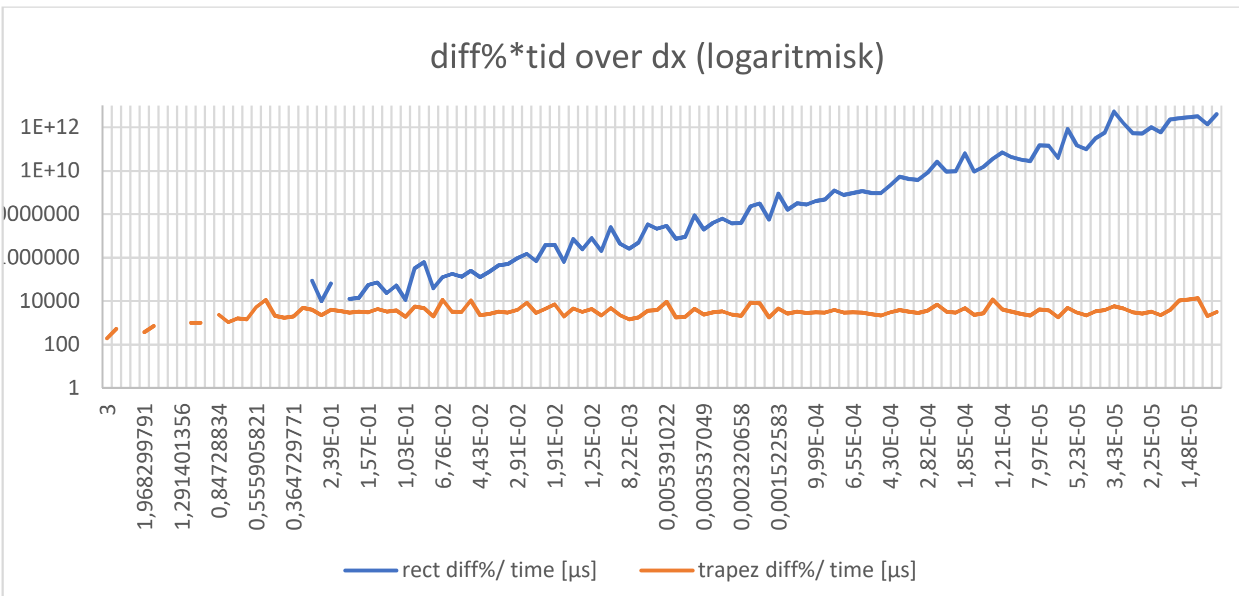
For at analysere hvor godt en funktion klarer skal man tage højde man vil have høj 1/tid dvs. frekvens og man vil have høj 1/diff% hvor diff% er afvigelses procent. Derfor siger vi at en høj værdi for 1/(tid*diff%) er godt.



En graf for diff% givet dx. Man kan generelt se at



Logaritmisk graf for forskel i 1/(tid*diff%) man kan tydeligt se at det er en linære linje på den logaritmiske graf og er derfor eksponentiel i virkeligheden. De manglende prikker er på grund af tid på 0, eller på grund af at rect metoden er bedre.



Trapez ser ud til at være tæt på samme rating, mens rect bliver ved med at stige.

Diskussion/perspektivering

Sammenligningen viste en arbitrer rating af de to metoder over tid, som ikke har et fodfæste i virkeligheden og for forskellige brug har tid/præcision forskellig prioritet. Derudover er trapez() delvist uoptimiseret da den bruger compute(x) to gange og ikke husker værdier fra den sidste iteration.

Generelt kan man sige at numeriske integrationsmetoder er uvigtige da man altid vil kunne lave det til en kurve og integrere det som er meget hurtigere og mere præcist.

Derudover vil det nok være mere rigtigt at plotte for tid i stedet for dx.

Konklusion

I undersøgelsen kom man frem til at man typisk, men ikke altid udregner integralet ved hjælp af søjler. Man kan generelt sige at præcisionen bliver ved med at gå op med en lavere dx men at forskellige metoder har forskellige hastigheder. Og ifølge denne undersøgelse hvis man ser væk fra de mangler den har, så har rekt en højere score, og er derfor bedre i forhold til tid og præcision.

Litteraturliste

Fra MIT's open course ware:
https://ocw.mit.edu/ans7870/18/18.013a/textbook/HTML/chapter20/section04.html

3blue1Brown en person der laver visualiseringer af matematik.
https://www.youtube.com/watch?v=rfG8ce4nNh0&list=PLZHQObOWTQDMsr9K-rj53DwVRMYO3t5Yr&index=8

Gert Friis Nielsen en person der laver et bevis på Infinitesimalregningens hovedsætning.
https://www.youtube.com/watch?v=eOQguKs8EQc

Bilag

I zip