# Numerisk Integration

Søren Fritzbøger 3F
HTX Hillerød - Erhversskolen Nordsjælland
Vejleder: Stig og CHR

2. Februar 2015

**Abstract**

This article demonstrates a basic set of LaTeX formatting commands. Compare the typeset output side-by-side with the input document.smøøre

# Indhold

# 1 Indledning

# 2 Forklaring af numerisk integration

# 3 Trapezmetoden

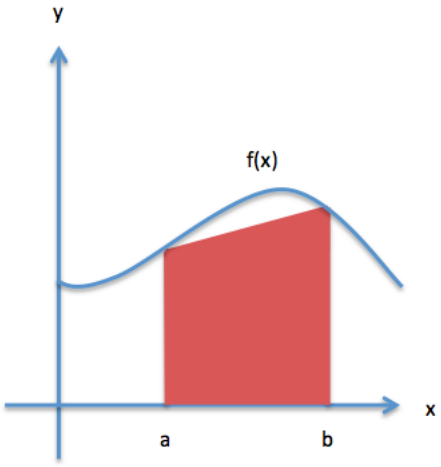
Trapezmetoden er en af flere metoder til at finde areal under en funktion. Det er en af de mere præcise metoder, da den, i modsætning til højre-, venstre- og middelsum, laver en trapez og ikke en firkant.

## 3.1 Bevis for trapezmetoden

Ved at bruge trapezmetoden kan vi tilnærmelsesvist finde det bestemte integral, selv for funktioner, hvor man ikke direkte kan bestemme det bestemte integral. Et eksempel på dette er funktionen $f(x) = e^{-x^2}$. For at bestemme arealet under en funktion vil vi gerne nærme os det bestemte integral.

$$\int_a^b f(x)dx$$

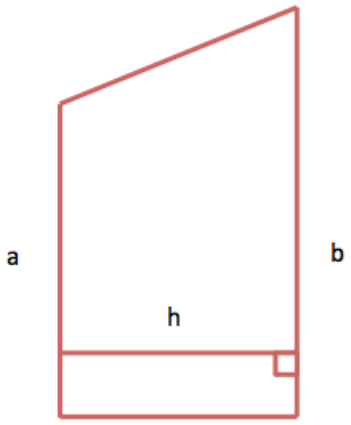
Når $a < b$ og $f$ er kontinuerlig og differentiabel i intervallet $[a; b]$, kan vi bruge trapezmetoden til at udregne det tilnærmelsesvist bestemte integral, som er arealet under funktionen.
Som navnet trapezmetoden så fint hentyder til, opdeler vi vores funktion i en trapez, som vi udregner arealet af. Dette er illustreret på figur 1




Figur 1: Trapezmetoden

På figur 1 kan man se, hvordan trapezmetoden opdeler en parabel i en trapez. Denne trapez har intervallet $[a; b]$. Længden $a = f(a)$ og længden $b = f(b)$. Formlen for arealet af en trapez ses i formel (3.1)

$$A = \frac{1}{2}h \cdot (a + b) \tag{3.1}$$




Figur 2: Trapez

Som man kan se på figur 2, er $a$ og $b$ er sidelænger og $h$ er højden. I et trapez er højden defineret som afstanden mellem de to parallelle linjestykker, i dette tilfælde $a$ og $b$, som man kan se illustreret på figur 2

Som man kan se på figur 2 svarer højden $h$ til afstanden mellem $a$ og $b$, som svarer til $h = b - a$. Når man indsætter henholdsvis $a$ og $b$ værdien ind i vores $f(x)$ funktion, får vi vores $y$ værdi, som svarer til længden af henholdsvis $a$ og $b$. Hvis vi indsætter dette i arealet for en trapez, får vi derfor:

$$\begin{aligned} A &= \frac{1}{2}h \cdot (a+b) && \Rightarrow \\ &= \frac{1}{2}(b-a) \cdot (a+b) && \Rightarrow \\ &= \frac{b-a}{2}(a+b) && \Rightarrow \end{aligned}$$

i stedet for $a$ og $b$ indsættes højderne på trapezens sider, $f(a)$ og $f(b)$

$$= \frac{b-a}{2}(f(a)+f(b)) \tag{3.2}$$

Arealet af denne trapez svarer tilnærmelsesvist til arealet under funktionen. Så derfor kan vi sige at arealet af trapezen tilnærmelsesvist svarer til værdien af det bestemte integral:

$$\int_a^b f(x) \approx \frac{b-a}{2}(f(a)+f(b)) \tag{3.3}$$

### 3.1.1 Opdeling i flere delintervaller

Som man kan se på figur 1 er trapezmetoden meget upræcis i forhold til det bestemte integral. Dette kan man forbedre ved at dele intervallet $[a;b]$ op i flere delintervaller, $n$. Dvs. at vi deler vores funktion op i mindre intervaller, som vi så finder arealet af med trapezmetoden. Disse delintervaller kalder vi for $x_0, x_1, \cdots x_n$. Vi skal bruge højden på trapezen, som er længden af delintervallet $x_0 - x_1$. I dette tilfælde svarer det til:

$$h = x_1 - x_0 = x_2 - x_1 = \cdots = x_n - x_{n-1} \tag{3.4}$$

Ud fra dette kan vi bestemme at højden, $h$, må svare til intervallet $[a;b]$ opdelt i $n$ intervaller:

$$h = \frac{b-a}{n} \tag{3.5}$$

Det samlede areal under funktionen kan betegnes som summen af alle delintervallerne:

$$A = \lim_{n\to\infty} \sum_{i=1}^{n} \frac{h}{2}(f(x_i) + f(x_{i+1})) \approx \int_a^b f(x) \qquad \text{hvor } h = \frac{b-a}{n} \tag{3.6}$$

Når n går mod uendelig bliver vores delinterval infinitesimalt lille, hvilket vil sige at summen af alle trapezarealerne nærmer sig det bestemte integral. Det bestemte integral svarer faktisk til at vi deler arealet op i uendelige små og uendelig mange led.

For at udlede dette til formlen for trapezmetoden bliver vi nødt til at opskrive dette på en lidt anden måde.

$$\begin{aligned} A &= \frac{h}{2}(f(x_0) + f(x_1)) + \frac{h}{2}(f(x_1) + f(x_2)) + \cdots + \frac{h}{2}(f(x_{n-1}) + f(x_n)) \\ &= h(\frac{1}{2}f(x_0) + \frac{1}{2}f(x_1)) + h(\frac{1}{2}f(x_1) + \frac{1}{2}f(x_2)) + \cdots + h(\frac{1}{2}f(x_{n-1}) + \frac{1}{2}f(x_n)) \\ &= h \cdot \frac{1}{2}f(x_0) + h \cdot \frac{1}{2}f(x_1) + h \cdot \frac{1}{2}f(x_1) + h \cdot \frac{1}{2}f(x_2) + \cdots + h \cdot \frac{1}{2}f(x_{n-1}) + h \cdot \frac{1}{2}f(x_n) \end{aligned}$$

Dette bliver reduceret til formlen for trapezmetoden (3.7)

$$\boxed{\int_a^b f(x) \approx \frac{h}{2}(f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n))} \tag{3.7}$$

$$\text{Hvor } h = \frac{b-a}{n}$$

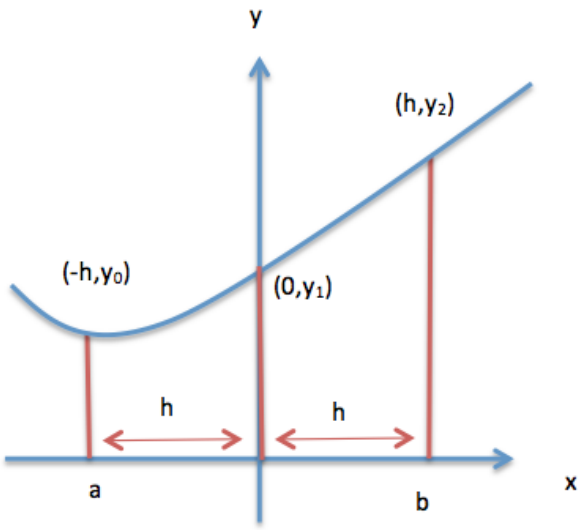
# 4 Simpsons metode

En anden metode til tilnærmelsesvist at udregne arealet under en funktion er simpsons-metoden, som virker ved at dele funktionen op i mindre parabler, man så finder arealet under. Denne metode er mere præcis end trapezmetoden[1], og giver derfor et resultat som oftest har en meget lille afvigelse fra det korrekte resultat, det bestemte integral.

[1] [4, side 15]

## 4.1 Bevis for Simpsons metode

Simpsons metode er som nævnt, en anden metode til tilnærmelsesvist at finde det bestemte integral, selv for funktioner, hvor man ikke direkte kan bestemme det bestemte integral. Med trapezmetoden, se afsnit 3, brugte man en ret linje fra $f(a)$ til $f(b)$ i intervallet $[a;b]$, hvilket selvfølgelig ikke er helt præcis, da en funktion ikke nødvendigvis er ret. Med Simpsons metode opdeler man intervallet $[a;b]$ i en parabel ud fra 3 punkter, $a, b, c$. Disse tre punkter kan bruges til at opstille en simplere parabel, som vi så kan regne arealet under. Parablens ligning ser således ud: $y = ax^2 + bx + c$




Figur 3: Simpsons metode

Figur 3 viser hvilke punkter vores parabel går gennem når vi har intervallet $[a;b]$. Ligesom med trapezmetoden kan man opdele funktionen i $n$ antal dele. Til at starte med går vi ud fra at $n = 2$, for at gøre det nemt.*I afsnit 4.1.1 bliver der viderebygget på Simpsons metode med flere intervaller.* Som man også kan se på figuren bliver $[a;b]$ delt op i to, hvor $\triangle x = h$. Dette viser sig også i punkterne som henholdsvis er $(-h, y_0)$, $(0, y_1)$ og $(h, y_2)$. Intervallet $[a;b]$ opdeles i $n$ antal delintervaller, som for dette tilfælde er 2, hvilket giver

afstanden h mellem vores punkter.

$$\begin{aligned} h &= \frac{b-a}{n} \\ &= \frac{b-a}{2} \qquad \text{Når } n = 2 \end{aligned} \tag{4.1}$$

Vi ved at vores parabel går fra $-h$ til $h$ og at den har forskriften $y = ax^2 + bx + c$. Dette simple udtryk for vores parabel kan vi bestemme det bestemte integral af, da vi ved at $f(x) = y$. Vi finder dermed det bestemte areal under en parabel med forskriften $y = ax^2 + bx + c$ fra $-h$ til $h$

$$\begin{aligned} A &= \int_{-h}^{h} f(x)dx \\ &= \left[\frac{ax^3}{3} + \frac{bx^2}{2} + cx\right]_{-h}^{h} \\ &= \left(\frac{ah^3}{3} + \frac{bh^2}{2} + ch\right) - \left(-\frac{ah^3}{3} + \frac{bh^2}{2} - ch\right) \\ &= \frac{ah^3}{3} + \frac{ah^3}{3} + \frac{bh^2}{2} - \frac{bh^2}{2} + ch + ch \\ &= \frac{2ah^3}{3} + 2ch \\ &= \frac{h}{3}(2ah^2 + 6c) \end{aligned} \tag{4.2}$$

Nu skal vi finde vores 2 ubekendte, som er $2ah^2$ og $c$. Dette kan vi gøre ved at opstille tre ligninger for parablen, $y = ax^2 + bx + c$, med vores tre punkter $(-h, y_0)$, $(0, y_1)$, $(h, y_2)$ fra figur 3, hvor vi så har 3 ubekendte. Vi går ud fra at $y$ svarer til punkternes y-koordinat og $x$ til vores x-koordinat.

$$y_0 = ah^2 - bh + c \qquad \text{Hvor } (x, y) = (-h, y_0) \tag{4.3}$$

$$y_1 = a \cdot 0^2 + b \cdot 0 + c \Rightarrow$$

$$y_1 = c \qquad \text{Hvor } (x, y) = (0, y_1) \tag{4.4}$$

$$y_2 = ah^2 + bh + c \qquad (x, y) = (h, y_2) \tag{4.5}$$

Som man kan se på ligningen for $y_1$ (4.3) har vi allerede fået isoleret vores $c$ værdi, som svarer til $c = y_1$ Vi vil gerne have isoleret $2ah^2$, og dette kan vi

gøre ved at substituere $c$ ind i de to ligninger for $y_0$ og $y_2$ og derefter ligge dem sammen.

Det gør vi således:

$$\begin{aligned} y_0 &= ah^2 - bh + c && \Rightarrow && \text{Hvor } c = y_1 \\ y_0 &= ah^2 - bh + y_1 && \Rightarrow \\ y_0 - y_1 &= ah^2 - bh && && (4.6) \end{aligned}$$

$$\begin{aligned} y_2 &= ah^2 + bh + c && \Rightarrow && \text{Hvor } c = y_1 \\ y_2 &= ah^2 + bh + y_1 && \Rightarrow \\ y_2 - y_1 &= ah^2 + bh && && (4.7) \end{aligned}$$

Ligning (4.6) og (4.7) ligges nu sammen, så vi kan isolere den sidste og manglende ubekendte, $2ah^2$.

$$\begin{aligned} y_2 - y_1 &= ah^2 - bh && + \\ y_0 - y_1 &= ah^2 + bh \\ & && = \\ ah^2 - bh + bh &= y_2 + y_0 - y_1 - y_1 \Rightarrow \\ ah^2 &= y_2 + y_0 - 2y_1 && && (4.8) \end{aligned}$$

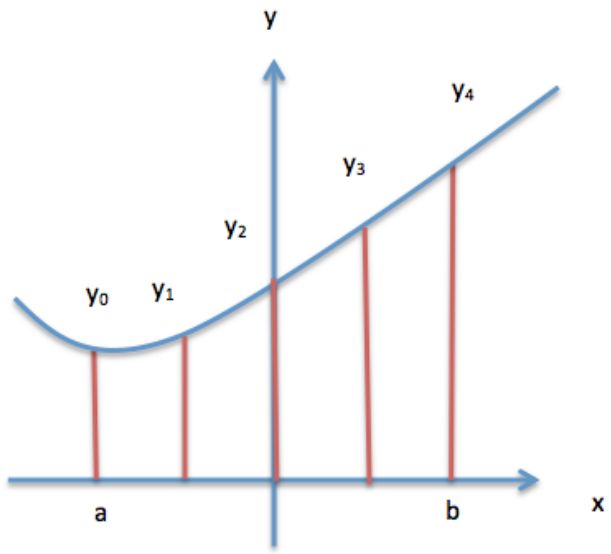
$ah^2$ og $c$ substitueres nu ind i formel (4.2) som ser således ud: $A = \frac{h}{3}(2ah^2 + 6c)$.

$$\begin{aligned} A &= \frac{h}{3}(2ah^2 + 6c) \\ &= \frac{h}{3}\left((y_2 + y_0 - 2y_1) + 6 \cdot y_1\right) \\ &= \frac{h}{3}(y_0 + 4y_1 + y_2) && (4.9) \end{aligned}$$

### 4.1.1 Opdeling i flere delintervaller

Når vi deler vores interval op i flere delintervaller, dvs. at $n$ bliver større, kan vi finde summen af $n$ antal delintervaller mellem $[a; b]$. På figur 4 kan man

se et eksempel på en opdeling med $n = 2$, hvor vi faktisk har 2 parabler. En fra $y_0$ til $y_2$ gennem $y_1$ og en fra $y_2$ til $y_4$ gennem $y_3$.




Figur 4: Opdeling i flere delintervaller

Ved hjælp af formel (4.9) kan vi udregne arealet af en parabel fra $y_0$ til $y_2$. Men hvis vi udvider denne som man kan se på figur 4, kan vi fortsætte simpsonsreglen fra $y_2$ til $y_4$, og dermed få det samlede areal for $[a; b]$ som går fra $y_0$ til $y_4$, som man kan se på figur 4. Summen af arealet fra $[a; b]$ når opdelingen $n = 4$

$$\begin{aligned} A_{[a;b]} &= \frac{h}{3}(y_0 + 4y_1 + y_2) + \frac{h}{3}(y_2 + 4y_3 + y_4) \\ &= \frac{h}{3}(y_0 + 4y_1 + y_2 + y_2 + 4y_3 + y_4) \\ &= \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + y_4) \qquad \text{Hvor } y = f(x) \end{aligned} \tag{4.10}$$

Det samme sker hvis vi har en opdeling på $n = 6$

$$A_{[a;b]} = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + 4y_5 + y_6) \qquad \text{Når } n = 6 \tag{4.11}$$

Ud fra dette kan vi opstille en ligning der siger at summen af alle delarealerne i intervallet $[a;b]$ opdelt i $n$ delintervaller tilnærmelsesvist må svare til det bestemte integral.

$$A = \lim_{n\to\infty} \sum_{i=1}^{n/2} \frac{h}{3}(y_{i-2} + 4y_{i-1} + y_i) \approx \int_a^b f(x) \qquad \text{hvor } h = \frac{b-a}{n} \tag{4.12}$$

Når n går mod uendelig bliver vores delinterval infinitesimalt lille, hvilket vil sige at summen af alle vores delintervaller, hvor vi har udregnet arealet under parablen. På den måde nærmer det, vha. Simpsons metode, udregnede areal tilnærmelsesvist det bestemte integral.

For at udlede dette til formlen for Simpsons metode, skal vi kigge nærmere på formel (4.9) og omskrive denne til at $n$ går mod uendelig.

$$\begin{aligned} A &= \frac{h}{3}(f(x)_0 + 4f(x)_1 + f(x)_2) && \text{Hvor } f(x) = y \text{ og når } n = 2 \\ &= \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2) + f(x_2) + 4f(x_3) + f(x_4) + \cdots + f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)) \end{aligned}$$

Dette bliver reduceret til Simpsons metode (4.13)

$$\boxed{\int_a^b \approx \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n))} \tag{4.13}$$

Hvor $h = \dfrac{b-a}{h}$ og $n$ er lige

# 5 Numerisk integration og computere

Når man skal omsætte en matematisk formel, til noget computeren kan forstå og regne med, skal man lave en såkaldt *algoritme*. En algoritme er en forskrift til løsning af et matematisk eller logisk problem, hvor man kender antallet af beregningsskridt[2]. I dette tilfælde er forskriften de 2 udledte formler (3.7), trapezmetoden, og (4.13), Simpsons metode, fra afsnit 3 og 4, der tilnærmelsesvist bestemmer det bestemte integral. Samtidig kender vi også antallet af beregningsskridt, da dette er antallet af delintervaller $n$. Derfor kan vi med sikkerhed sige, at vi kan lave en algoritme der tilnærmelsesvist udregner arealet under en vilkårlig matematisk funktion. Når man skal skrive en algoritme til en computer, skal dette gøres i et programmeringssprog[3], som f.eks. JavaScript, PHP, C++, java osv. Hvert sprog kan bruges til forskellige ting og på forskellige enheder. Nogle programmeringssprog bruges til regulære computerprogrammer mens andre er udviklet til at fungere i en webbrowser som Chrome, Internet Explorer og lignende.
Når man arbejder med algoritmer, er der nogle ting, som er meget vigtige at have styr på. Det ene er selvfølgelig at algoritmen giver et korrekt resultat, altså at den er nøjagtig, og det andet er hvor lang tid algoritmen bruger på udregningerne. Der skal være en fin balance mellem nøjagtigheden og effektiviteten. Nogle gange er det meget vigtigt at algoritmen er 100% nøjatig, mens at andre algoritmer er så avancerede, at en 100% nøjagtighed tager for lang tid. Nogle algoritmer kan være så komplekse og kræve så store beregningsmæssige ressourcer, at de ikke kan løses med de computere, vi har adgang til i dag.

Jeg har brugt JavaScript til at opbygge mine algoritmer for Trapezmetoden og Simpsons metode. Fordelen ved JavaScript er at det kan køre i browseren, og det er derfor forholdsvist nemt for alle at køre. Samtidig er JavaScript et af de sprog, der har en mere simpel syntaks, så det er nemt at forstå, og det er derfor også meget tilgængeligt.

## 5.1 Effektivitetsoptimering i JavaScript

Når man skal skrive en algoritme, er det næsten umuligt ikke at have et såkaldt *loop*. Et loop, løkke på dansk, er en kode, der kører så længe at en

[2][5][Version2.dk]
[3][6][Den store danske]

defineret erklæring passer. Den løkke jeg skal bruge til mine algoritmer, skal køre $n$ antal gange, altså den skal køre det antal gange, jeg har delt min funktion op i. I teorien kan n bevæge sig mod uendelig, men dette vil så også tage op mod uendelig tid. For at finde en 100% præcis værdi af vores numeriske integration, kan vi godt risikere at vi kommer op på et delinterval på flere millioner. Derfor at det vigtigt at vi optimere løkken mest muligt, da det er en kode, der potentielt skal køre mange millioner gange. En måde at optimere løkken på er ved at vælge en while-løkke i stedet for en for-løkke. Et hurtigt eksperiment[4] viser, at while-løkken i gennemsnit er en smule hurtige[5]. Samtidig udførte eksperimentet kun en meget simpel beregning, hvor den algoritme vi skal udregne, er meget mere avanceret. Derfor kan der være en stor besparing i tid og beregningskraft, hvis man bruger en while-løkke i stedet for en for-løkke.

## 5.2 Opbygning af algoritmerne

Når man skal lave en algoritme til en matematisk funktion, er det vigtigt at have analyseret formlen først, så man kan skabe en præcis og effektiv algoritme. Jeg har udarbejdet en fungerende algoritme, som jeg har prøvet at effektivisere så meget som muligt. Derefter har jeg analyseret nøjagtigheden og effektiviteten af både metoden og algoritmen.

### 5.2.1 Trapezmetoden

Formlen for Trapezmetoden (3.7), som også kan ses på side 6, ser således ud:

$$\int_a^b f(x) \approx \frac{h}{2}(f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)) \quad \text{Hvor } h = \frac{b-a}{n}$$

Når man analyserer Trapezmetoden i forhold til at skulle lave en algoritme, er der et par ting man skal have fokus på. For det første er $\frac{h}{2}$ uden for parentesen hvilket gør, at vi til sidst i algoritmen kan udregne summen af delarealerne ganget med $\frac{h}{2}$. En anden ting er, at første og sidste punkt, henholdsvis $x_0$ og $x_n$ ikke skal ganges med 2 ligesom resten af delarealerne. Derfor kan vi sætte disse to uden for vores hoved-løkke, hvilket gør at vi ikke skal bekymre os

[4] [7]

[5] Det skal nævnes at det ikke er et 100% korrekt eksperiment, da andre processer fra computeren, kan have haft en rolle i beregningstiden. Hvis man dog fjerne de unormale resultater fra det samlede billede, er while-løkken stadig gennemsnitlig hurtigst.

om dem i løkken. Ud fra denne korte analyse har jeg skrevet min algoritme for Trapezmetoden i intervallet $[a; b]$ og med antallet af delintervaller $n$.

Algoritme 1: "udregnArealTrapezmetoden()"

```
1  function udregnArealTrapezmetoden(a,b,n) {
2    var h = (b-a)/n;
3    var funktion = document.getElementById("funktion").value;
4
5    var w = 1;
6    var x = a;
7
8    var areal = eval(funktion);
9    while(w < n) {
10     x = x + h;
11     areal += 2*eval(funktion);
12     w++;
13   }
14
15   x = b;
16   areal += eval(funktion);
17   areal = (h/2)*areal;
18
19   return areal;
20 }
```

Min algoritme er opbygget som en funktion med variabler $a, b, n$. $a$ er startintervallet, $b$ er slutintervallet og $n$ er antal delintervaller, præcis som i formlen for Trapezmetoden. $h$ defineres præcis som i Trapezmetoden, dvs $h = \frac{b-a}{n}$. Variablen $funktion$ tager værdien fra et input der har id'et "funktion", som indeholder vores funktion formateret som JavaScript. Et eksempel på dette ses her for $f(x) = e^{x^2}$

```
1  Math.exp(Math.pow(x,2))
```

Variablen $w$ er vores tæller som starter på 1. $x$ bliver sat til a, da vi skal substituere $x$ ind i vores funktion. På linje 8 udregnes arealet af $x_1$, som sker før vores løkke, som nævnt i analysen. På linje 9 starter vores løkke, som kører så længe w er mindre end n. Dvs. at løkken kører lige ind til vi når sidste punkt, $x_{n-1}$, så vi kan regne $x_n$ særskilt. Inde i løkken bliver x sat til længden af delintervallet, $h$ lagt sammen med sig selv, når vi finder den næste x-værdi i intervallet. Så udregnet arealet af delintervallet og ligges til variablen $areal$. Derefter lægger vi 1 til w, så den tæller op mod n. Efter løkken har kørt, sætter vi $x$ til at være vores sidste værdi, $b$, og regner arealet

for det sidste delinterval. Til sidst ganges hele det udregnede delareal med $\frac{h}{2}$ for at finde det samlede areal. Funktionen returnerer arealet til sidst.

### 5.2.2 Simpsons metode

Formlen for Simpsons metode (4.13), som også kan ses på side 11, ser således ud:

$$\int_a^b \approx \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n))$$

Hvor $h = \frac{b-a}{h}$ og $n$ er lige

Simpsons metode ser en del anderledes ud end Trapezmetoden, så derfor skal den analyseres anderledes, og algoritmen kommer formentlig også til at blive mere avanceret.
Igen har vi h uden for parentesen, dog bare $\frac{h}{3}$ i stedet. Samtidig skal punkt $x_0$ og $x_n$ ikke ganges med noget, så de kan også beregnes uden for løkken. En ting der skal lægges meget mærke til ved Simpsons metode, er at den skifter mellem at multiplicere med 4 og 2. En måde at beskrive det er, at alle ulige skal ganges med 4 og lige med 2. Derfor skal løkken indeholde en *if/else* sætning, der tjekker om det er et lige eller ulige antal vi beregner på. Ud fra denne analyse, har jeg skrevet min algoritme for Simpsons metode i intervallet $[a; b]$ og med antallet af delintervaller $n$

Algoritme 2: "udregnArealSimpsonsMetode()"

```
1  function udregnArealSimpsonsMetode(a,b,n) {
2  var h = (b-a)/n;
3  var funktion = document.getElementById("funktion").value;
4
5  var w = 1;
6  var x = a;
7
8  var areal = eval(funktion);
9  while(w < n) {
10   x = x +h;
11   if(w%2 == 0)
12     areal += 2*eval(funktion);
13   else
14     areal += 4*eval(funktion);
15
16   w++;
```

```
17 }
18
19 x = b;
20 areal += eval(funktion);
21 areal = (h/3)*areal;
22
23 return areal;
24 }
```

Igen er algoritmen opbygget som en funktion med intervallet $[a; b]$ og antal delintervaller $n$. De første 8 linjer er præcis det samme som i Algoritme 1: "udregnArealSimpsonsMetode()", hvor der defineres forskellige variabler, og arealet for $x_1$ udregnes. Det næste er while-løkken på linje 9 som kører så længe w er mindre end antal delintervaller $n$. Først sættes x til at være næste delinterval, og dernæst tjekkes der om tælleren, $w$, er et lige eller ulige tal. Dette gør den vha. modulus funktionen, (*skrives % i JavaScript*), som er en matematisk funktion. Modulus svarer til det positive heltal, der er i rest, når man dividerer. Så når man tager modulus 2 af et tal, giver det enten en rest på 1, når tallet af ulige, og ellers en rest på 0, da alle lige tal går op i 2. Arealet af delintervallet regnes så ud fra denne *if/else*-sætning, hvor lige ganges med 2 og ulige med 4.
Resten af koden afviger kun fra algoritmen til Trapezmetoden ved at gange det samlede udregnede areal med $\frac{h}{3}$ som det står i Simpsons metode (4.13).

## 5.3 Vurdering af effektiviteten og nøjagtigheden

Helt grundlæggende vil jeg Når jeg skal vurdere nøjagtigheden, vil jeg både vurdere selve metoden og algoritmen. Hvis algoritmen er skrevet korrekt, burde der ikke være forskel på nøjagtigheden i forhold til metoden. Jeg vil undersøge nøjagtigheden ud fra resultatet af algoritmen/metoden opdelt i forskellige delintervaller, $n$, i forhold til det bestemte integral.
Effektiviteten kan være svær at vurdere, da jeg ikke har noget at stille mine data op imod. Jeg har undersøgt hvor lang tid det tager, at køre algoritmen med forskellige delintervaller. Denne tid kan sammenlignes med nøjagtigheden af algoritmen, og ud fra det vil jeg vise en sammenhæng mellem præcision og hastighed.
Jeg har udvalgt 2 funktioner som minder om hinanden, dog er den ene en del mere avanceret end den anden. Den ene er $e^x$ og den anden $e^{x^2}$. Den anden funktion, $e^{x^2}$, er en af de funktioner man ikke kan bestemme det bestemte integral af, mens den første er simpel at bestemme. Til at bestemme area-

let der er så tæt på det bestemte integral som muligt for $e^{x^2}$, har jeg brugt mathcad, som har givet den værdi der står i tabel 3 og 4

### 5.3.1 Nøjagtighed og effektivitet når $f(x) = e^x$

Tabel 1: Nøjagtighed af Trapezmetoden for $f(x) = e^x$ i intervallet $[-1; 3]$

| $n$ | Beregningstid | Trapezmetoden | Bestemt integral |
|---|---|---|---|
| 2 | 0.05ms | 25.89 | 19.718 |
| 4 | 0.016ms | 21.334 | 19.718 |
| 10 | 0.019ms | 19.98 | 19.718 |
| 20 | 0.022ms | 19.738 | 19.718 |
| 40 | 0.029ms | 19.734 | 19.718 |
| 180 | 0.088ms | 19.718 | 19.718 |

Tabel 2: Nøjagtighed af Simpsons metode for $f(x) = e^x$ i intervallet $[-1; 3]$

| $n$ | Beregningstid | Simpsons metode | Bestemt integral |
|---|---|---|---|
| 2 | 0.0179ms | 20.884 | 19.718 |
| 4 | 0.026ms | 19.815 | 19.718 |
| 10 | 0.040ms | 19.72 | 19.718 |
| 14 | 0.050ms | 19.718 | 19.718 |

### 5.3.2 Nøjagtighed og effektivitet når $f(x) = e^{x^2}$

Tabel 3: Nøjagtighed af Trapezmetoden for $f(x) = e^{x^2}$ i intervallet $[-1; 3]$

| $n$ | Beregningstid | Trapezmetoden | Bestemt integral |
|---|---|---|---|
| 2 | 0.020ms | 8111.239 | 1446.008 |
| 4 | 0.020ms | 4111.218 | 1446.008 |
| 10 | 0.021ms | 2032.925 | 1446.008 |
| 20 | 0.022ms | 1603.752 | 1446.008 |
| 40 | 0.036ms | 1486.247 | 1446.008 |
| 180 | 0.090ms | 1448.008 | 1446.008 |
| 400 | 0.182ms | 1446.413 | 1446.008 |
| 2000 | 1.008ms | 1446.024 | 1446.008 |

Tabel 4: Nøjagtighed af Simpsons metode for $f(x) = e^{x^2}$ i intervallet $[-1; 3]$

| $n$ | Beregningstid | Simpsons metode | Bestemt integral |
|---|---|---|---|
| 2 | 0.018ms | 5411.117 | 1446.008 |
| 4 | 0.020ms | 2777.877 | 1446.008 |
| 10 | 0.039ms | 1593.515 | 1446.008 |
| 20 | 0.025ms | 1460.694 | 1446.008 |
| 40 | 0.037ms | 1447.079 | 1446.008 |
| 180 | 0.098ms | 1446.011 | 1446.008 |
| 252 | 0.129ms | 1446.008 | 1446.008 |

# 6 Perspektivering

# 7 Konklusion

# Litteratur

[1] Christopher R. Johnson, *Denne kilde er et dokument med beviser, beskrivelse og andet af numerisk integration..*
Hamlet Project, Department of Computer Science, University of Utah, `http://www.cs.utah.edu/~zachary/computing/lessons/uces-13/uces-13/contents-node1.html`

[2] @mattmight, *En "tutorial" der viser hvordan man tegner en graf vha. html5 og canvas.*
http://matt.might.net/articles/rendering-mathematical-functions-in-javascript-with-canvas-html/

[3] Aalborg Universitet, *Et foredag fra Aalborg Universitet omhandlende Numerisk Integration.*
http://staff.iha.dk/jse/bioproces/Forelaesningsnoter/BI1MAT1/Numeriske%20metoder%201.pdf

[4] Undervisningsministeriet, *Forberedelse til Matematik A højere teknisk eksamen 2012*, `http://uvm.dk/~/media/UVM/Filer/Udd/Gym/PDF12/Proever%20og%20eksamen/120608%20Mat%20A%20htx%20Forberedelsesmateriale.pdf`

[5] Version2.dk - Klaus Hansen og Casper Thomsen, *Leksikon: Algoritme*, `http://www.version2.dk/leksikon/Algoritme`

[6] Den Store Danske - Jens Clausen og senere af Uffe Rasmussen, *Algoritme* `http://www.denstoredanske.dk/It,_teknik_og_naturvidenskab/Informatik/Software,_programmering,_internet_og_webkommunikation/algoritme`

[7] Stoimen's web log, *JavaScript Performance: for vs. while* `http://www.stoimen.com/blog/2012/01/24/javascript-performance-for-vs-while/`

# 8 Kildekritik

# 9 Bilag

Her skal være regnereglerne for integralregning