SOP - Mindste Kvadraters Metode

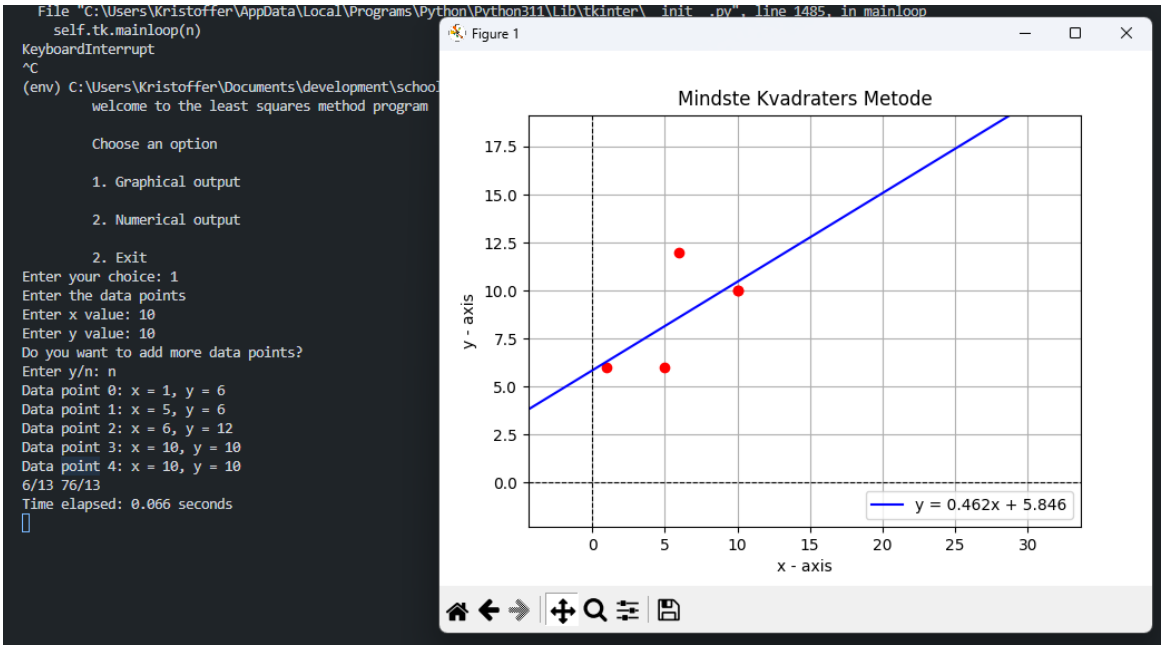
Rybners HTX - Vejleder: Carsten Finn Sørensen og Vicki Jacob

Programering B og Matematik A

Periode for opgave: 05/12-24 til 19/12-24

Af Kristoffer Sørensen

19. december 2024



Figur 1: Grafisk afbildning af user interfacet

Kristoffer Sørensen
Tegn: 43594

Resume

Denne opgave undersøger anvendelsen af Mindste Kvadraters Metode til at modellere data og ved hjælp af det, finde den bedst passende funktion. Metoden er baseret på matematikken bag funktioner af to variable og partiel differentiation. Partiel differentiation bruges her til at finde et minimumspunkt hvilket gør det muligt at finde det sted hvor summen af kvadraternes areal mellem datapunkter og en lineær linje er mindst. Den teoretiske forståelse blev omsat til praksis ved anvendelse på et datasæt og implementering i Python. I programmeringskonteksten blev metoden implementeret med fokus på modularitet og brugervenlighed, en pseudokode blev lavet for at strukturere processen. Fordelene ved programmering inkluderer hurtigere beregninger, automatisering og muligheden for at håndtere store datasæt, men udfordringer som fejl i input og beregnings hastighed ved store datasæt bliver også belyst. En diskussionen om kodningsmetoder fremhævede forskellene mellem procedural, objektorienterede og funktionel programmerings tilgange, hvor valget af metode afhænger af projektets kompleksitet og skalerbarhed. Der blev desuden også diskuteret hvilke kontrolstrukture der er bedst i hvilke senarier. Samlet set demonstrerer opgaven, hvordan Mindste Kvadraters Metode i kombination med programmering kan omsætte komplekse matematiske beregninger til effektive og anvendelige løsninger i dataanalyse.

Kristoffer Sørensen
Tegn: 43594

Indhold

Kristoffer Sørensen
Tegn: 43594

1 Indledning

I en verden, hvori data spiller en central rolle, er evnen til at analysere og modellere datasæt afgørende for videnskabelig forskning, og genrelt tolkning af datasæt. For at kunne analysere data er det nødvendigt at kunne finde en model, der beskriver sammenhængen mellem de forskellige variable. En af de metoder der kan anvendes til at beskrive data er Mindste Kvardraters Metode. Denne metode gør det muligt, at finde den best mulige funktion, ved at funktionstilpasse. Mindste Kvardraters Metode er især nyttig når man arbejder med lineære regression hvor målet er at finde en ret linje, der bedst beskriver sammenhængen mellem to variable. Ved at anvende denne metode kan man nemt finde og forudsige tendenser i værdier baseret på allerede eksisernede data. For at forstå anvendelsen af Mindste Kvardraters Metode er det nødvendigt at dykke ned i de del elementer der ingår. Herunder funktioner af to variable og det at partiellt aflede noget. Dette giver en god teoretisk baggrund for at implementere metoden effektivt i en programerings kontekst.

2 Opgaveformulering

Hvordan kan Mindste Kvadraters Metode anvendes til at modellere data for at finde den bedst mulige funktionstilpasning?

- Redegør for matematikken bag mindste kvadraters metode, herunder skal der redegøres for funktioner med to variabler og de partielle afledede.
- Anvend mindste kvadraters metode på et selvvalgt datasæt.
- Producer et eksempel på implementering af mindste kvadraters metode.
- Vurder hvilke fordele og begrænsninger der følger med denne metode i en programmeringskontekst.
- Diskuter forskellige kodningsmetoder til løsning af mindste kvadraters metode.

Kristoffer Sørensen
Tegn: 43594

3 Redegørelse for Mindste Kvadraters Metode

Antag at du har en række data. Disse data beskriver en sammenhæng mellem tid og distance for en bil. Bilen kører med en konstant hastighed. Du har nu brug for at finde ud af hvor langt bilen har kørt efter 10 sekunder. Dette datapunkt ingår dog ikke i datasættet. For at finde denne information kunne man fx. opstille en funktion der beskriver denne sammenhæng ved hjælp af funktionstilpasning. Mindste Kvadraters Metode kunne her anvendes som regressions værktøj. Mindste Kvadraters Metode anvendes ofte til at beskrive en ret linje ($y = ax + b$). En ret linje kan skrives på mange former. Den kan deusden også beskrvies med forskellige hældninger m.v Dog vil der kun være en linje der beskriver et datasætt bedst. For at finde frem til det kræver det, at der fordybes i funktioner af to variable, samt hvordan de afledes.

3.1 Funktioner med to variable

Du har måske stiftet bekendtskab med funktioner af en variabel. Disse funktioner skrives ofte som $f(x) = x^2$, hvor man indsætter en værdi for x og beregner den tilsvarende værdi af $f(x)$. Her giver man altså en værdi og får en værdi tilbage. Men hvad nu hvis du skal beskrive en sammmenhæng, der er afhængig af to variabler? Her kommer funktioner med to variabler ind i billedet. Når der arbejdes med funktioner af en variabel, vil den grafiske afbildning forgå i et koordinatsystem med en x-akse og en y-akse. Altså et todimensionelt koordinatsystem. For at kunne afbillede funktioner af to variable, arbejdes der ikke længere i et koordinatsystem, med todimensioner, men i et tredimensionelt koordinatsystem. Her er der en x-akse, en y-akse og en z-akse. Den nye akse (z-aksen) står vinkelret på de to andre akser (Grøn m.fl., 2019, s. 246–248). For bedre at forstå, hvordan det tredimensionelle koordinatsystem hænger sammen, kan man tænke på det som en kasse, hvor x-aksen er længden, y-aksen er bredden, og z-aksen er højden. Når der arbejdes med funktioner af to variable, vil den typiske notation se således ud: $f(x, y)$. Her er både x og y variable. Disse to variable kan bruges til at beskrive et punkt i det tredimensionelle koordinatsystem. Dette skyldes, at $z = f(x, y)$. Et eksempel kunne være et vejrkort, hvor temperaturen (z) afhænger af geografiske koordinater (x, y). Her angiver x og y en lokation, og z angiver temperaturen ved dette punkt. På den måde afhænger temperaturen af to variable, nemlig det geografiske punkt, som beskrives ved (x, y).
I forbindelse med Mindste Kvadraters Metode bruges funktioner af to variable til at beskrive summen af afvigelserne mellem punkt og en linje. I denne sammenhæng opstilles funktionen $f(a, b)$, hvor a og b er de parametre, der bestemmer linjens hældning og skæring. Denne funktion kan opfattes som en flade i et tredimensionelt koordinatsystem. For at finde den linje, der bedst passer til datasættet, skal værdierne af a og b findes. Dette kræver, at der arbejdes med metoder til at finde minimumspunkter i funktioner af to variable. Grunden til hvorfor minimumspunktet skal findes forklares i afsnit 3.2. Hvordan minimumspunktet findes forklares dog i næste afsnit.

3.1.1 Partielt differentiation

For at finde hældningen af tangenten til en funktion af to variable, skal man gå igennem en lidt længere proces end ved funktioner af en variabel. For at finde hældningen af tangenten til funktionen af en variabel, diffrencere man med hensyn til en variabel som ofte vil være x. Et eksempel kunne være $f(x) = x^2$ her er $f'(x) = 2x$. Dette skyldes følgende regneregl: $a \cdot x^n$ bliver til $a \cdot n \cdot x^{n-1}$.
Ved funktioner af to variable krærver dette at man benytte sig af en metode kaldt: *Partial*

differentiation. Ved partial differentiation, tages der fat i en variabel ad gangen. Partial betyder delvis. Grunden til at dette ord anvendes er da man differentieret mht. x og y i hver deres led (Larsen, 2016, s. 4). Her undersøges, hvordan funktionen ændrer sig med hensyn til en variabel, mens den anden betragtes som konstant. Opgaven vil nu være at finde den partial afledte for en funktion. For at finde den partielle afledte for funktionen $f(x, y)$. Måden hvorpå dette gøres er at der startes med at antage at en af variablerne til en konstant. Når der diffrenceres med en funktion af en variabel, agives dette sådan:

$$\frac{df(x)}{dx}$$

Sådan er det dog ikke ved partial differentiation. Her angives det sådan:

$$\frac{\partial f(x, y)}{\partial x} \tag{1}$$

Det bløde d (∂) angiver at der differentieres partielt. For at illustrere hvordan partial diffrencering fungeret tages der udgangspunkt i funktionen $f(x, y) = x^2 + 3y^2$. Målet er at finde de partielle afledede af $f(x, y)$. Først diffrenceres der med hensyn til x, mens y dermed betragtes som en konstant. Regnereglerne foreskriver at en konstant bliver til nul når den diffrenceres. Dernest anvendes regnereglen, der foreskriver at $a \cdot x^n$ bliver til $a \cdot n \cdot x^{n-1}$ (Pihl m.fl., 2019, s. 14). Dermed må den partial afledte mht. x være:

$$\frac{\partial f(x, y)}{\partial x} = 2x$$

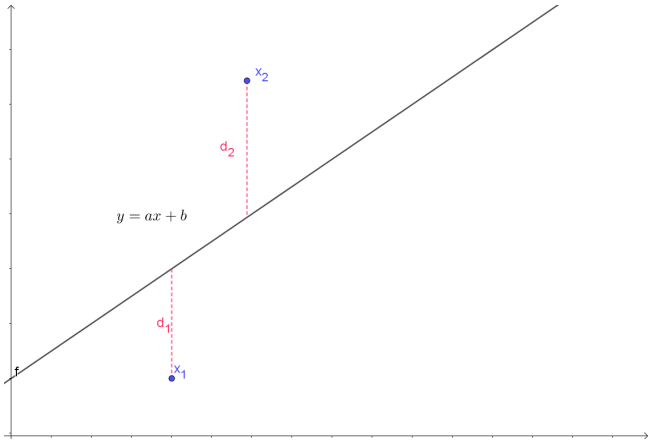
På samme måde findes den partielle afledede med hensyn til y ved at betragte x som konstant. Dette giver:

$$\frac{\partial f(x, y)}{\partial y} = 6y$$

Ved Mindste Kvadraters Metode anvendes denne teknik til at finde ekstremumspunkter for funktioner af to variable. Ekstremumspunkterne repræsenterer de værdier for parametrene a og b, der minimerer summen af kvadrerede afvigelser mellem datapunkterne og modellen. Dette uddybes i det følgende afsnit.

3.2 Sådan finder du frem til den bedste funktion

I de foregående afsnit er der blevet redegjort for funktioner af to variable og, hvordan de differentieres. Der er også blevet forklaret, hvorfor disse metoder er relevante, og hvordan de kan bruges i forbindelse med Mindste Kvadraters Metode. For at finde den bedste funktion skal der udvikles et udtryk der beskriver hvor godt funktionen passer til et datasættet. Den nemmeste måde at beskrive hvor godt en linje passer på et datasæt, vil være at beskrive afstanden fra punkt til linje. Da det er med til at beskrive hvor god linjen i virkeligheden er. Dette kan gøres på flere forskellige måder. Afstanden kunne beskrives som værende den vinkelrette afstand fra linjen ned til punktet. Dette er dog ret svært at regne med, eftersom det kræver at det vides hvor på linjen at afstanden er vinkelret på linjen. I stedet for det kan den lodrette afstand fra punkt til linje findes. Dette gøres ved at finde forskellen i y-værdierne. Se figur 2 for grafisk ilustration. Dette skaber dog en ny udfordring som man skal være opmærksom på. Nemlig at der kan være punkter over og under linjen. En distance kan altså derfor både være positiv og negativ, og de kan dermed udligne hinanden. For at undgå at dette bliver et problem opløftes afstanden i anden. ("Forberedelsessæt matematik A HTX maj 2013", 2013, s. 2).



Figur 2: Afstand fra punkt til linje

Da der nu er styr på hvordan afstanden fra punkt til linje findes. Skal der nu findes en udtryk hvorpå det er muligt at beskrive afstanden ift linjen. Hvis der startes med at tage udgangspunkt i et punkt. Punktet kunne hede hvad som helst. Linjen hedder dog altid: $y = a \cdot x + b$. Så må afstanden i y-værdien kunne beskrives som:

$$d_n = (a \cdot x_n + b - y_n)^2 \tag{2}$$

Grunden til at y_n fratrækkes, er at y-værdien for linjen ved x_n ville være $y = a \cdot x + b$. Isoleret set så er afstanden y-værdien for funktionen minus y-værdien for punktet. På den måde kan afstanden altså bestemmes. Afstanden har på den måde nu dannet en ligning med to ubekendt. Til sammen beskriver de arealet af en kvadrat. Grunden til at de beskriver arealet af en kvadrat er da afstanden som nævnt før, opløftes i anden. Det må derfor være det samme som at regne arealet af en kvadrat (Bentzen, 2014). Nu begynder navnte Mindste Kvadraters Metode lige så stille at give mening. Da målet er at finde den linje hvor Kvadraternes areal er mindst muligt. For at regne distance fra alle punkter til linjne anvendes den formlen som blev introduceret før (Formel: 2). Deres fælles areal kan derfor beskrives som følgende:

$$A = \sum_{n=1}^{n} (a \cdot x_n + b - y_n)^2 \tag{3}$$

Resultat af denne udregning kunne omskrives til en funktion af to variable, hvor a og b er de ubekendte. Der hvor arealerne er mindst må de to ubekendte danne den best mulige linje. Dette ville kunne ses som et ekstrema, på funktionen af variablerne a og b. Dette ekstrema findes ved at lave partial diffrencering (Se afsnit 3.1.1) af funktionen med hensyn til a og b og til sidst sætte differentialkvotienterne til 0. Dette skyldes at når hældningen af tangenten er

nul så vil der være enten et toppunkt eller minimumspunkt. I tilfældet her vil der kun være et minimumspunkt. Da slut funktionen danner en parabel ligende figur. Der vil nu være dannet to ligninger med med to ubekendte, vil være hhv a og b Disse to ligninger kan løses og dermed findes den bedste linje. ("Mindste kvadraters metode", u.d.)

4 Praktisk anvendelse af Mindste Kvadraters Metode

Med baggrund i afsnit 3 om redegørsel for Mindste Kvadraters Metode kan metoden nu anvendes på datasættet om bilen, der kører med konstant hastighed. Datasættet viser sammenhængen mellem tid og den tilbagelagte distance for bilen:

$Tid[s]$	$Distance[m]$
1	6
5	6
6	12
10	10

Tabel 1: Sammenhæng mellem tid og distance.

For at finde frem til den funktion, der bedst beskriver datasættet, opstilles først den ligning som blev forklaret for oven (Formel: 3), der beskriver afstanden mellem punkterne og linjen:

$$A = (a \cdot 1 + b - 6)^2 + (a \cdot 5 + b - 6)^2 + (a \cdot 6 + b - 12)^2 + (a \cdot 10 + b - 10)^2$$

Dette udtryk skal udvides, så alle ledene med a, b og ab er samlet et sted.Første skridt er at omskrive hvert kvadrat så det kommer på denne form:

$$(a \cdot x_n + b - y_n)^2 = (a \cdot x_n + b - y_n) \cdot (a \cdot x_n + b - y_n)$$

Dette gør det muligt at udvide udtrykkene ved at gange parenteserne ud. Alle led tages nu hvert for sig og udvides:
Punktet $(\mathbf{1}, \mathbf{6})$ udvidet:

$$(a \cdot 1 + b - 6) \cdot (a \cdot 1 + b - 6) = a^2 + b^2 + 2ab - 12a - 12b + 36$$

Punktet $(\mathbf{5}, \mathbf{6})$ udvidet:

$$(a \cdot 5 + b - 6) \cdot (a \cdot 5 + b - 6) = b^2 + 10ab - 12b + 25a^2 - 60a + 36$$

Punktet $(\mathbf{6}, \mathbf{12})$ udvidet:

$$(a \cdot 6 + b - 12) \cdot (a \cdot 6 + b - 12) = b^2 + 12ab - 24b + 36a^2 - 144a + 144$$

Punktet $(\mathbf{10}, \mathbf{10})$ udvidet:

$$(a \cdot 10 + b - 10) \cdot (a \cdot 10 + b - 10) = b^2 + 20ab - 20b + 100a^2 - 200a + 100$$

Alle led samles nu til et samlet udtryk.

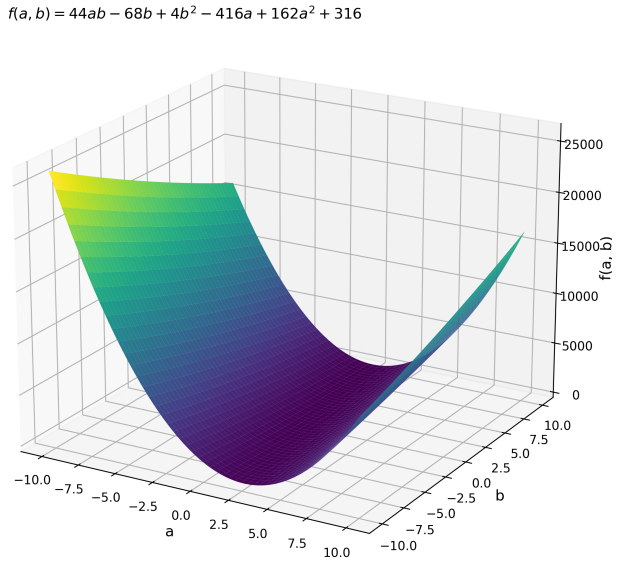
$$A = a^2 + b^2 + 2ab - 12a - 12b + 36 + b^2 + 10ab - 12b + 25a^2 - 60a + 36 +$$
$$b^2 + 12ab - 24b + 36a^2 - 144a + 144 + b^2 + 20ab - 20b + 100a^2 - 200a + 100$$

Dette udvidet udtrykket kan nu reduceres til:

$$A = 44ab - 68b + 4b^2 - 416a + 162a^2 + 316$$

Dette omskrives til en funktion af to variabele, hvor a og b er de ubekendte.

$$f(a,b) = 44ab - 68b + 4b^2 - 416a + 162a^2 + 316$$



Figur 3: Grafisk afbildning af funktionen $f(a,b)$

For at finde den funktion der passer bedst til datasættet, skal der findes det sted hvor funktionen $f(a,b)$ har et minimumspunkt (Se evt figur 3) Dette gøres ved at tage partielt afledede af funktionen $f(a,b)$ med hensyn til a og b.

$$\frac{\partial f(a,b)}{\partial a} = 44b + 324a - 416$$

$$\frac{\partial f(a,b)}{\partial b} = 8b + 44a - 68$$

Disse to difransialkvotienter sættes nu til 0. Dette giver følgende ligningsystem:

$$44b + 324a - 416 = 0$$

$$8b + 44a - 68 = 0$$

Dette ligningsystem kan nu løses med hensyn til a og b. Dette giver følgende resultat:

$$a = 0,5122$$

$$b = 5,68293$$

Resultatet af beregningerne af dette datasæt viser at denne metode godt kan anvendes til at finde en linje, der passer bedst til et datasæt. Det kan være svært at konkludere præcist hvor god den er, men resultatet er sat op mod GeoGebra og her stemmer resultaterne overens, helt ned til 5 decimal.

Kristoffer Sørensen
Tegn: 43594

5 Programmering og Matematik i samspil

Matematik og programmering hænger på mange måder naturligt godt sammen. Matematik bruges til at beskrive og forklare komplekse problemstillinger, mens programmering gør det muligt at omsætte disse problemstillinger til konkrete løsninger. Med programmering følger også en række praktiske fordele, så som at man kan gøre selv de sværeste håndberegninger på sekunder. Dette gælder især, når man arbejder med store datasæt. Programmering giver os værktøjerne til at bearbejde data hurtigt og præcist, hvilket er noget mennesker aldrig ville kunne gøre manuelt lige så hurtigt. ("Decoding the Relationship Between Mathematics and Programming", 2024) Forestil dig for eksempel, at du har ti tusinde datapunkter fra en virksomheds salgsdata. At finde et mønster manuelt ville være en næsten umulig opgave. Her kommer computeren og programmeringen ind i billedet. Ved hjælp af simple algoritmer og lidt programmeringserfaring kan man omsætte dataene til resultater på få sekunder. Det samme gælder, når der arbejdes med Mindste Kvadraters Metode. Hvor det manuelt ville kræve mange trin og lang tid at finde den bedst passende linje, kan en computer beregne dette på ingen tid. Se bare hvor mange skridt der er blevet taget i afsnit 4, for at finde den bedst mulige linje. Forestil dig bare hvor lang tid det ville tage hvis der var mere end fire punkter. Programmering er ikke bare hurtigere. Det åbner også op for at eksperimentere og tilpasse løsninger. Har man for eksempel data, hvor nogle observationer afviger markant fra de andre resultater, kan man justere modellen eller filtrere data for at få en bedre beskrivelse. Med Mindste Kvadraters Metode kan det også hurtigt testes, hvordan små ændringer i datasættet påvirker slut resultatet.Denne fleksibilitet er især hvis det data der kommer ud af et forsøg ikke opføre sig som forventet. Muligheden for at ændre beregninger undervejs, er dog ikke den eneste fordel, som kommer da programmering anvendes.

En anden fordel ved at kombinere matematik og programmering er muligheden for at visualisere. Når en model er blevet beregnet, kan man hurtigt lave grafer, der viser, hvordan modellen passer til det konkrete datasæt. For eksempel kan man plotte en lineær regression sammen med de oprindelige datapunkter for at få et visuelt overblik over, hvordan sammenhængen ser ud. Det gør resultaterne nemmere at forstå og formidle. Det er netop i kombinationen af matematik og programmering, at den virkelige styrke ligger. Matematikken giver præcision og struktur, mens programmeringen giver hastighed, fleksibilitet og mulighed for at arbejde med data i stor skala. Dette gør ikke bare processen hurtigere, men det gør det også muligt at håndtere komplekse problemstillinger, som ellers ikke har været til at løse. Når man arbejder med metoder som Mindste Kvadraters Metode, bliver denne kombination særlig tydelig. Det er her teori og praktisk erfaring mødes. ("Handling Large Datasets in Python", 2024).

6 Implementering af Mindste Kvadraters Metode

Som tidligere nævnt er en af styrkerne ved at anvende programmering dens evne til at automatisere og effektivisere beregninger. Dette er en stærkt fordel, når der arbejdes med Mindste Kvadraters Metode. Hvor det manuelt ville tage mange trin at finde den bedst passende linje, gør programmering det muligt at omsætte de matematiske beregninger til kode, der hurtigt og præcist kan håndtere selv store datasæt. Dette åbner op for en række anvendelser, lige fra praktisk dataanalyse til visualisering og optimering. I dette afsnit vil der blive fokuseret på, hvordan

Kristoffer Sørensen
Tegn: 43594

Mindste Kvadraters metode kan implementeres i programmering ("Least Square Method | Definition Graph and Formula", 2024). Ved at anvende en struktureret tilgang kan de matematiske formler opdeles i klare og logiske trin, som nemt kan oversættes til kode. Pseudokode fungerer som et effektivt værktøj til at beskrive denne proces på en enkel og overskuelig måde.

6.1 Pseudokode for Mindste Kvadraters Metode

For at implementere Mindste Kvadraters Metode er det vigtigt at det hele tiden anvendes en struktureret tilgang som beskrevet for oven. En metode til at holde overblik samt holde styr på strukturen kunne være anvendelsen af pseudokode. Følgende er en pseudokode, der beskriver, hvordan Mindste Kvadraters Metode kan implementeres i programmering:

Require: Datasæt $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
Start
Opret:
    sum_a2 = 0
    sum_b2 = 0
    sum_ab = 0
    sum_a = 0
    sum_b = 0
    konstant_sum = 0
For hvert datapunkt $(\mathbf{x_i}, \mathbf{y_i})$ i datasættet:
Indset i formlen $(a \cdot x_i + b - y_i)^2$
    Udvid $(a \cdot x_i + b - y_i)^2$
    Beregn bidragene til a^2, b^2, ab, og konstanter
    Tilføj disse værdier i sum_a2, sum_b2, sum_ab, sum_a, sum_b, og konstant_sum
Kombiner alle bidrag i den samlede funktion $\mathbf{f(a, b)}$:
    $f(a, b) = suma2 + sumb2 + sumab + suma + sumb + konstantsum$
Find den partielt afledede for funktionen $f(a, b)$:
    $\frac{\partial f}{\partial a}$
    $\frac{\partial f}{\partial b}$
Find frem til $aogb$ ved at sætte differentialkvotienterne til 0:
    $\frac{\partial f}{\partial a} = 0$
    $\frac{\partial f}{\partial b} = 0$
return a, b

Denne pseudokode viser, hvordan Mindste Kvadraters Metode kan implementeres trin for trin i et programmerings øjemed. De matematiske beregninger, som blev beskrevet tidligere, er blivet til en proces, der hurtigt og præcist kan håndtere data ("Linear Regression Method Pseudocode", u.d.). At anvende Pseudokode virker i sig selv, måske lidt irelevandt. Men reelt kan det faktisk gøre selve programmerings proceen nemere. Pseudokoden fortæller nemlig hvad man skal gøre skridt for skidt. Altså først opprettes variablerne så køres alle resultaterne igennem en løkke. Det giver altså en muligehed for let at holde styr på det samt at skrive koden. Det hjælper samtidigt med hvordan matematikken bag Mindste Kvadraters Metode kan implementeres i programmering, og hvordan matematik og programmering kan arbejde sammen om at løse konkrete problemstillinger.

6.2 Program Design

Afsnittet om pseudokode har givet et godt overblik over, hvordan Mindste Kvadraters Metode kan implementeres i programmering. Der er dog flere overvejelser, der skal gøres. Det at skrive et program, er ikke bare at skrive et program. Det kræver en grundig planlægning og overvejelse af, hvordan programmet skal designes, hvilke sprog og biblioteker der skal anvendes, og hvordan det hele skal hænge sammen. Det bedste sted at starte ved et hvert program er: hvad er formålet med programmet? Formålet med programmet her, er at implementere Mindste Kvadraters Metode for at finde den bedst passende linje til et givet datasæt.

Programmet skal være baseret på nøje overvejelser om skalerbarhed, fleksibilitet, og muligheden for nemt ændre i forskellige dele af programmet. En vigtig overvejelse i designet har været, hvilket paradigme der skulle anvendes. Skal programmet være objektorienteret, funktionelt, eller noget helt tredje? Alt sammen kommer med deres fordele og ulemper. Mere om dette i afsnittet om Kodningsmetoder (Afsnit: 8). Valget af paradigme endte dog på funktionel programmering. Grunden til at valget af paradigme endte på funktionelt programmering, er at det giver en række klare fordele når det kommer til modularitet (Shikha, u.d.). En anden af de helt store årsager til at funktionelt programmering er valgt over objektorienteret er at funktionel programmering i højere grad fremmer ”immutability”(Datastruktur eller et objekt ikke kan ændres efter, det er blevet oprettet).
Alt dette gør til sammen at det bliver lettere at tilføje eller ændre funktionalitet løbende uden at påvirke andre dele af programmet. Dette kan især være vigtigt i et tilfælde, hvor man ønsker at håndtere data, der ændres dynamisk, eller hvor man skal beregne nye resultater baseret på forskellige datasæt. Funktionel programmering tilbyder en række værktøjer og principper, såsom højere-ordens funktioner og pure functions, der gør det lettere at arbejde med komplekse matematiske operationer som Mindste Kvadraters Metode (“Functional Programming Paradigm”, 2024). Hvis man derimod skulle implementere Mindste Kvadraters Metode i et objektorienteret paradigme, ville man typisk skulle bruge flere ressourcer på at administrere tilstande og objekter. For eksempel kan det kræve flere klasser og metoder til at definere og opdatere de forskellige parametre, som a og b, samt opbevare datasættet.
Når det kommer til sprogvalg, så faldt beslutningen på Python. Dette valg blev truffet på baggrund af sprogets forcer. Python er et af de mest populære programmeringssprog i verden (Paul, 2024), hvilket betyder, at der er masser af dokumentation og biblioteker til rådighed. Dette er dog ikke det eneste der gør Python til et godt valg. Python er desuden kendt for sin enkelhed og læsbarhed, hvilket gør det nemt at skrive og vedligeholde kode. Dette er en stor fordel, især i projekter, hvor klarhed og hurtig implementering er vigtig.

En anden væsentlig styrke ved Python er dets evne til at håndtere store datasæt effektivt, hvilket er centralt i dette projekt. Med biblioteker som NumPy og sumPy kan komplekse matematiske beregninger, som dem der kræves til Mindste Kvadraters Metode, udføres hurtigt og præcist. Selv om andre sprog som C og Java også kunne løse opgaven, blev Python valgt, fordi det tillader hurtigere udvikling og lettere forståelse af koden. Disse egenskaber gør Python til et oplagt valg til dette projekt (John, 2024).

I programdesign spiller ikke kun valget af sprog og paradigme en rolle, men også hvordan koden struktureres og dokumenteres. Et godt dokumenteret program gør det muligt for andre (eller en selv) at forstå og ændre koden i fremtiden. Kommentarer og klare funktionsbeskrivelser

er en stor del af designet og sikrer, at koden er så selvforklarende som muligt. Der er dog en anden ting som også ingår, som en del af program design, og det er valg af hvilken case der anvendes. Dette er normalt ikke noget en programmør skal tage stilling til, men hvis python anvendes er det lidt op til den individuelle programør. Mange anvender snake-case og andre anvender camelcase. Samtidig understøtter Python brugen af docstrings, som giver en struktureret måde at dokumentere hver funktion og dens formål på. Alt i alt er programmets design med til at sikre, at det er nemt at vedligeholde, forstå og udvide. Ved at bygge programmet med baggrund i de før nævnte principper sikres det, at det ikke kun opfylder dets nuværende formål: at implementere Mindste Kvadraters Metode. Men også forbliver robust og fleksibelt nok til at håndtere fremtidige behov og anvendelser. Dette gør programmet til en bæredygtig løsning.

6.3 Program redegørelse

Programmet kan som nævnt i det foregående afsnit, skrives på forskellige måder, afhængigt af hvilket paradigme, biblioteker og sprog der anvendes. Programmet er som nævnt skrevet i Python og anvender sumPy, NumPy og mathplotlib. Programmet er ikke optimeret på en måde hvor det fylder så lidt som muligt, samt så det kører så hurtigt som muligt. Det er derimod skrevet på en måde hvor det er nemt at forstå. Det er skrevet på en sådan måde at det nærmest er en til en kontra pseudokoden beskrevet i afsnit 6.1. Programmet består af tre funktioner. En funktion der hedder `leastSquaresMethod(dataPoints)`. Den funktion beregner a og b. Den anden funktion hedder `plotter(dataPoints)` den funktion har til opgave at plottet beregningerne og punkterne ind i et koordinatsystem. Til sidst har programmet en funktion der hedder `main()`. Funktionen `main()` styrer hele programmet. Det er herfra de andre funktioner kaldes.

Når programmet køres møder slutbrugeren et user interface (Se evt. figur 1). Dette interface spørger ind til tre ting. Den første mulighed for få en grafisk afbildning, af datasættet samt den lineære linje. Den anden mulighed er et numerisk resultat for den lineære linje. Den sidste valgmulighed er at afslutte programmet. Det er en nødvendighed at have en exit funktion da funktionen `main()` er rekursiv og derfor kalder sig selv, i tilfælde at input fejl, samt efter afsluttet beregning. Efter denne overordnede introduktion til programmets opbygning, vil der nu blive set nærmere på, hvordan den helt centrale funkton arbejder, for at implementere Mindste Kvadraters Metode. Denne funktion spiller en afgørende rolle i processen, og det er derfor relevant at gennemgå dens specifikke opgave og sammenhæng.

6.3.1 Gemmengang af koden for Mindste Kvadraters Metode

I dette afsnit vil det blive forklaret hvordan Mindste Kvadraters Metode er programmeret. Denne forklaring vil være bid for bid, sådan at man til sidst ved hvordan a og b findes. De andre funktioner ville ikke blive gemmengået, da de ikke direkte har noget med Mindste Kvadraters Metode at gøre. Koden har dog kommentar i sig, sådan at det er nemt at forstå hvad der sker. Koden kan ses i sin helhed i bilag 10.1.

```
1    from sympy import symbols, solve, Eq, diff
2    import matplotlib.pyplot as plt
3    import numpy as np
```

Kristoffer Sørensen
Tegn: 43594

Dette er den allerførste del af koden. Her importeres de nødvendige biblioteker. `sympy` er et bibliotek der bruges til at løse ligninger. `matplotlib.pyplot` er et bibliotek der bruges til at plotte grafer ind i kordinatsystener. `numpy` er et bibliotek der bruges til at håndtere matematiske operationer, på arrays og matricer m.v (Et array er som en liste med nummerede bokse, hvor du kan gemme forskellige ting, f.eks. tal, ord eller andre data).

```
1     def leastSquaresMethod(data_points):
2         a, b = symbols('a b')
3         sumA2 = 0
4         sumB2 = 0
5         sumAB = 0
6         sumA = 0
7         sumB = 0
8         sumConstant = 0
9         n = len(data_points)
```

Der sker en del, i dette kode udrag. Først og fremmest defineres funktionen `leastSquaresMethod(dataPoints)`. Funktionen kræver, at den får et datasæt som input, bestående af punkter. Kravet for den måde punkterne skal være formateret på, kommer i det næste kode udrag. I funktionen defineres en række variable. Disse variable bruges til at holde styr på summen af de forskellige led i ligningen. `n` er antallet af punkter i datasættet. Grund til at programmet ønsker at finde antallet af punkter er for at kunne løbe igennem alle punkterne i datasættet, og vide hvornår det er færdigt, så den ikke prøver at finde et punkt der ikke er der. Der dog en del der mangler at blive forklaret. Nemlig linje to, i dette kode udrag. I linje to kan det ses at der defineres to variable `a` og `b`. Disse variable er symbolske variable. Normalt vil python ikke kunne forstå hvad symboler i denne form er. Python er programmeret til at forstå hvad tal og bogstaver er. Python understøtter altså ikke symbolsk matematik. Dette kan dog håndteres med biblioteket SymPy, som giver mulighed for at arbejde med symboler. SymPy opretter et objekt for hver symbolsk variabel i dette tilfælde a og b. Disse variable repræsenterer en del af et matematisk udtryk. Disse objekter gemmes som instanser i Python og kan manipuleres, analyseres og bruges til forskellige matematiske operationer som differentiering, og løsning af ligninger ("What is Symbolic Computation in SymPy?", 2022).

```
1     for i in range(n):
2         x = dataPoints[i][0]
3         y = dataPoints[i][1]
4         print(f"Data point {i}: x = {x}, y = {y}")
5         sumA2 += a**2 * x**2
6         sumB2 += b**2
7         sumAB += 2*a*b*x
8         sumA += 2*a*x*y
9         sumB += 2*b*y
10        sumConstant += 2*y
```

I denne kodebid sker selve beregningen, der ligger til grund for Mindste Kvadraters Metode. Koden starter med en for-løkke, der gennemgår alle de datapunkter, som programmet modtager. For-løkken stopper først, når den har behandlet alle punkterne i datasættet. Koden her er meget afhængig af at få punkterne i et specefikt format. nemlig en liste af tuples, f.eks.

`[(x_1, y_1), (x_2, y_2)]`. Hvis inputtet ikke følger dette format, vil programlogikken ikke fungere korrekt. Programmet har ikke inprogrameret nogen form for fejlhåndtering, på dette. Det skyldes at programmet som det er nu, selv formaterer det når punkterne indsættes. Af den grund burde der derfor ikke opstå fejl.

For hver omgang løkken køres beregnes bidragene fra det aktuelle punkt (`dataPoints[i]`) til de summer, der bruges til at opbygge slutfunktionen $f(a, b)$. Disse summer (`sumA2`, `sumB2`, `sumAB`, `sumA`, `sumB` og `sumConstant`) repræsenterer de matematiske led, der kræves for at finde den linje, der bedst beskriver datasættet. Hvert datapunkt bidrager med specifikke værdier til følgende summer: $a^2 \cdot x^2$, b^2, $2 \cdot a \cdot b \cdot x$, $2 \cdot a \cdot x \cdot y$, $2by$, og en konstant y^2. Disse summer opdateres løbende gennem løkken. Undervejs ville det være muligt at se hvordan linjen tilpasse i takt med at kvadratsummen for flere punkter beregnes. Måden at se det på at ved at indsætte de beregninger der forklares i det følgende kode eksempel i løkken. I slut programmet er det dog valgt ikke at have det inde i løkken da, det er meget ineffektivt

```
1      f = sumA2 + sumB2 + sumAB - sumA - sumB + sumConstant
2
3      diffA = diff(f, a)
4      diffB = diff(f, b)
5      solutions = solve([diffA, diffB], (a, b))
6      return solutions[a], solutions[b]
```

Når alle punkterne er behandlet, samles de udregnede summer til en samlet funktion i programmet kaldt f (`f = sumA2 + sumB2 + sumAB - sumA - sumB + sumConstant`). Denne funktion repræsenterer summen af kvadraternes arealer, som skal minimeres for at finde den bedst passende linje. For at finde det sted hvor kvadraternes areal er mindst, skal der som sagt tages det partielt afledte af funktionen $f(a, b)$ med hensyn til a og b. Differentiation er dog ikke en indbygget funktion i Python. Derfor bruges biblioteket SymPy til at beregne den partielt afledte funktion. SymPy beregner $\frac{\partial f}{\partial a}$ og $\frac{\partial f}{\partial b}$, hvilket resulterer i to ligninger. Disse ligninger løses derefter med hensyn til a og b ved hjælp af SymPy's `solve`-funktion. Til sidst returnerer funktionen løsningerne for a og b som en tuple, f.eks. `[a, b]`. Disse værdier repræsenterer hældningen og skæringen for den linje, der bedst passer til datasættet.

7 Fordele og Begrænsninger

Mindste Kvadraters Metode er en kraftfuld teknik til at finde den bedst passende linje til et datasæt. Metoden ligger klart op til at blive indsat i et programeringsmilijø. At implementere det i et programeringsmilijø åbner op for en række muligheder, men også visse udfordringer. En af de absolut største fordele ved at implementere Mindste Kvadraters Metode i programmering er muligeheden for automatisering og hastigheden. I programmeringssprog som Python, hvor biblioteket SymPy kan anvendes, kan komplekse matematiske beregninger udføres hurtigt og effektivt. Dette er særligt vigtigt, når man arbejder med store datasæt, hvor man manuelt ville skulle bruge adskillige timer på beregninger, mens et program kan håndtere det på få sekunder. Når programmet er skrevet, giver det også mulighed for, at brugere uden forudgående kendskab

til metoden kan anvende det. Der er som nævnt mange fordele. På trods af styrkerne er der også nogle klare begrænsninger ved Mindste Kvadraters Metode i en programmeringskontekst. En af de mest betydningsfulde steder hvor det kan gå galt er i gennemsigtigheden. Det der menes med mangel på gennemsigtighed er at en evt fejl ved programmeringen kan være svær at finde, en fejl kunne blandet andet være at punkterne ikke er i det rigtige format. Eller at matematikken er kodet forkeret. Der er altså mange steder hvor det kan gå galt, og det kan være svært at finde fejlene. Men der vil stadig være nogle andre begrænsninger. Hvis løsning kodes på samme måde som den er blevet beskrevet i dette projekt (Se koden i bilag 10.1), vil programmet tage ret lang tid om at beregne. Det skyldes flere ting. Blandet andet diffrencering og løsning af ligninger er meget tidskrævende operationer. Derudover er der også en del overhead i programmet. Programmet er skrevet på en måde hvor det er nemt at forstå og ikke så meget så det køre hurtigt. I det beskrevne program tager det 0,08 sekunder at beregne den bedste linje for 4 datapunkter. Med den kode vil 200 sekunder for at beregne den bedste linje for 10000 datapunkter. Det kan dog optimeres. Men igen denne metode er ret langsom, det vil uanset hvad selvfølgeligt være hurtigere end at gøre det manuelt. Så skal det ses i et større perspektiv, er det en simpel og solid løsning. Som vil være udemærket til små og mellemstore datasæt, i hvert fald i programmeringskontekst.

8 Diskussion af Kodningsmetoder

Når man skal implementere Mindste Kvadraters Metode, står man overfor flere valg om, hvordan koden best kan struktureres og udføres som beskrevet i afsnit 6.2. Der er mange kodningsmetoder og teknikker der kan anvendes, og hver metode har sine fodele og ulemper. Denne diskussion vil forkusere på to hovedeområder: Valg af programmeringsparadigme, samt valg af kontrolstrukture som loops og rekursive funktioner.
Den første overvejelse må være hvilket paradigme der skal anvendes. De mest oplagte valg ville være objektorienteret programmering, funktionel programmering eller Procedural programmering. Hver af disse paradigmer har sine fordele og ulemper. De forskellige fordele og ulemper vil nu blive beskrevet i punkt form, og tilsidst vil der blive givet en konklusion på hvilket paradigme der er bedst egnet til Mindste Kvadraters Metode.

- **Procedural programmering**
  Procedural programmering indebære at programmet opdeles i funktioner, der hver især læser en specifik del af problemet. For Mindste Kvadraters Metode betyder det, at man kan skrive sepearte funktioner til at beregne hældning og skæringen samt til at plotte resultaterne. Procedural programmering er kendt for at være enkelt og let at forstå. Den kommer dog også med ulemper blandet andet hvis den skal skaleres eller genavendes i komplekse senairer ("The procedural paradigm", u.d.).

- **Objektorienteret programmering**
  OOP eller objektorienteret programmering er et paradigme, hvorman strukturerer programmet i objekter, som kombinerer data og funktionalitet. For at implementere Mindste Kvadraters i et objektorienterede paradigme kunne en klasse med navnet regression skabes. Den klasse kunne indeholde datasættet som attributter samt metoder til at beregne hældningen og skæringen. Dette giver en stærk struktur, og mulighed for genbrug, da klassen kan genanvendes med forskellige datasæt uden at ændre logikken. OOP gør det desuden lettere at vedligeholde og udvide koden, da nye funktioner kan tilføjes ved blot

at skabe nye metoder eller klasser, der arver fra eksisterende klasser. Dog kan OOP tilføje unødig kompleksitet, især i mindre projekter som dette, hvor strukturen kan føles overdimensioneret. En anden ulempe ved OOP er, at det kan kræve flere ressourcer at designe og implementere korrekt, især hvis programmets skala ikke retfærdiggør denne indsats ("Introduction of Object Oriented Programming", 2023).

- **Funktionel programmering**
  Funktionel programmering er et paradigme, der fokuserer på brugen af rene funktioner (Pure functions), som hver især kun opererer på deres input og ikke ændrer eksterne data. Dette gør funktionerne forudsigelige og uafhængige af konteksten, hvilket sikrer, at de altid returnerer det samme output for det samme input. I forbindelse med Mindste Kvadraters Metode kan funktionel programmering tilbyde en elegant tilgang, hvor hver beregning for eksempel summen af kvadraterne eller differentiering kunne have været hver deres funktion. Dette gør koden mere modulær og lettere at teste. En funktionel tilgang gør også brug af immutability. Hvilket betyder at data ikke ændres efter oprettelse. Dette reducerer risikoen for fejl og gør programmet mere pålideligt. Funktionel programmering kan dog være mindre intuitivt for udviklere, der er vant til procedurale eller objektorienterede tilgange, og kan kræve flere beregninger, da data konstant kopieres i stedet for at blive ændret direkte ("Functional Programming Paradigm", 24).

Konklusionen om hvilket paradigme der er bedst afhænger altså at den konkrete løsning. Hvert paradigme har sine fordele og ulemper, som gør det mere eller mindre velegnet i forskellige projekter. Skal der dog siges noget generelt, må det dog siges at valget afhænger af, om projektet er simpelt, komplekst eller kræver muligehed for skalerbarhed. Hvis projektet er lille og ikke forventes at skulle skaleres. Så vil procedural programmering ofte være det beste valg. Den simple struktur gør denne tilgang ideel, til hurtigt at udvilke en løsning, der er nem at forstå. Denne løsning er ikke god, hvis den skal indgå som en del af et større projekt. Da koden ofte vil være langsom, og svær at vedligeholde. Hvis projektet er stort og komplekst, vil objektorienteret programmering ofte være det bedste valg. Dette skyldes, at klasserne kan opprettes dynamisk, genbruges i flere dele af programmet. Klasser oprettes (instansieres) kun, når de anvendes, hvilket gør hukommelsesforbruget fleksibelt. Klasserne vil dog leve i hukommelsen lidt inden de tilsidst vil blive fjernet af sprogets garbage collecter. Dog kan der være en kortvarig stigning i ressourceforbruget, når en klasse oprettes eller initialiseres, især hvis der udføres tunge beregninger eller dataindlæsning i konstruktøren. Samlet set giver OOP en god balance mellem struktur, fleksibilitet og genbrugelighed, når det anvendes korrekt i store projekter. Den sidste valgmulighed er funktionel programmering. Det ligger lidt et sted midt imellem de to. Den tilbyder en modulær tilgang, hvor beregningerne kan opdeles i mindre funktioner, der hver især udfører en specifik opgave. Dette gør koden mere læsbar og lettere at vedligeholde. Funktionel programmering er også kendt for at være mere sikker, da den understøtter immutability, hvilket reducerer risikoen for fejl. I praksis afhænger valget ofte af personlige præferencer og erfaringer. Hvis udvikleren har stor erfaring med funktionel programmering, kan det være en oplagt løsning på trods af det mindre projekt. Omvendt kan procedural programmering være et bedre valg, hvis udvikleren søger en hurtig og letforståelig tilgang.

Det er nu klarlagt hvilket paradigme der er mest oplagt, men hvilke kontrolstrukturer er bedst? Dette vil altid være et godt sprøgsmål det afhænger igen af hvor stort projekt er, hvordan det kodes og meget mere. Om der anvendes While loops, for loops, eller rekursive funktioner til

beregningerne kommer med hver deres forcer og svagheder. For loops er gode til at gennemgå en liste af elementer, mens While loops er gode til at køre en kodeblok, indtil en betingelse er opfyldt. Rekursive funktioner er gode til at løse komplekse problemer. De kan dog være svære at forstå og kan føre til uendelige løkker, hvis de ikke er korrekt implementeret. I forbindelse med Mindste Kvadraters Metode vil et for loop være det mest oplagte valg, da det er nødvendigt at gennemgå hvert datapunkt i datasættet. En for loop er også lettere at forstå og implementere, end en while loop eller rekursiv funktion. Et while loop kan dog være nyttig, hvis der er behov for at sætte en betingelse for, hvornår løkken skal stoppes. Ved for loops køre den indtil et givet stop, og ikke at 1+1 nu giver to. Rekursive funktioner er mindre velegnede til Mindste Kvadraters Metode, da det ikke er nødvendigt at løse problemet på denne måde. Samlet set afhænger valget af kontrolstrukturer af den konkrete opgave og udviklerens præferencer, i dette tilfælde vil for-loops dog være at foretrække. Da loopet skal køre et bestemt antal gange, og ikke stoppe før en betingelse er opfyldt.

9 Konklusion

Denne opgave har skulle undersøge, hvordan Mindste Kvadraters metode kan anvendes til at modellere data og finde den bedst mulige funktionstilpasning. Gemmen en teoretisk redegørsel, praktisk anvendelse og implementering i en programmeringskontekst, er metoden blevet belyst i både de matematiske perspektiver samt i programmeringsmæssige. Matematikken bag Mindste Kvadraters Metode er baseret på konceptet om at minimere summen af kvadraternes areal mellem datapunkterne og en foreslået funktion. Dette kræver forståelse for funktioner af to variable, partielle afledede, og løsning af ligninger. De partielt afledede anvendes til at finde det minimumspunkt, der repræsenterer den optimale hældning og skæring for en linje. Den teoretiske redegørelse blev yderligere underbygget med konkrete udregninger, der demonstrerer metodens anvendelighed på et datasæt. Implementering i programmering blev udført ved hjælp af Python, hvilket illustrerede metodens styrke i at blive automatiseret og effektiviseret. Her blev pseudokode brugt til at strukturere processen, og programmet blev designet med fokus på modularitet og brugervenlighed. Gennem praktisk implementering blev det klart, hvordan programmering gør det muligt at håndtere store datasæt og visualisere resultater hurtigt og præcist. Metoden har mange fordele i en programmeringskontekst, herunder automatisering, hastighed og tilgængelighed for bruger uden matematiske egenskaber.D og er der også begrænsninger, såsom risikoen for fejl i inputdata eller implementering samt hastigheden for beregning af store datasæt. Disse udfordringer kan dog afhjælpes gennem optimering af programmet. I diskussion af Kodningsmetoder blev det belyst, hvordan forskellige paradigmer, som proceduralt, objektorienteret og funktionelt programmering, hver tilbyder unikke fordele og udfordringer i forhold til implementeringen. Her blev det konkluderet at: For mindre projekter er procedurale løsninger ofte tilstrækkelige, mens objektorienteret eller funktionel programmering kan være mere velegnet til større og komplekse opgaver. Til sidst kan det konkluderes, at Mindste Kvadraters Metode er en simpel og kraftfuld teknik til dataanalyse, der kan anvendes i mange sammenhænge. Metoden er dog ikke super velegnet til at blive implementeret i et programmering da der skal laves en del funktioner til diffrencering og løsning af ligninger. Generelt set så er Mindste Kvadraters Metode en god metode til funktionstilpasning.

Kristoffer Sørensen
Tegn: 43594

Litteratur

Bentzen, C. (2014 oktober). Mindste Kvadrater. Hentet 10. december 2024, fra https://www.youtube.com/watch?v=VVTcW5cZUv0&t=196s&ab_channel=MrCbentzen

Decoding the Relationship Between Mathematics and Programming. (2024 januar). Hentet 12. december 2024, fra https://www.codewithc.com/decoding-the-relationship-between-mathematics-and-programming/

Forberedelsessæt matematik A HTX maj 2013. (2013).

Functional Programming Paradigm. (24 september). Hentet 16. december 2024, fra https://www.geeksforgeeks.org/functional-programming-paradigm/

Functional Programming Paradigm. (2024 september). Hentet 13. december 2024, fra https://www.geeksforgeeks.org/functional-programming-paradigm/

Grøn, B., Bruun, B., & Lyndrup, O. (2019). Hvad er matematik 3. Lindhardt og Ringhof.

Handling Large Datasets in Python. (2024 april). Hentet 13. december 2024, fra https://www.geeksforgeeks.org/handling-large-datasets-in-python/

Introduction of Object Oriented Programming. (2023 februar). Hentet 16. december 2024, fra https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/

John, T. (2024 oktober). *Why Python is Essential for Data Analysis*. Hentet 13. december 2024, fra https://www.simplilearn.com/why-python-is-essential-for-data-analysis-article

Larsen, M. O. (2016). Funktioner af flere variable. Hentet 8. december 2024, fra https://www.larsen-net.dk/files/Funktioner-af-flere-variable-Ti.pdf

Least Square Method | Definition Graph and Formula. (2024 august). Hentet 13. december 2024, fra https://www.geeksforgeeks.org/least-square-method/

Linear Regression Method Pseudocode. (u.d.). Hentet 13. december 2024, fra https://www.codesansar.com/numerical-methods/linear-regression-method-pseudocode.htm

Mindste kvadraters metode. (u.d.). *webmatematik*. Hentet 10. december 2024, fra https://www.webmatematik.dk/lektioner/matematik-b/regression/mindste-kvadraters-metode

Paul, J. (2024 december). *TIOBE Index*. Hentet 16. december 2024, fra https://www.tiobe.com/tiobe-index/

Pihl, B., Schou, M. H., & Madsen, L. (2019). Formelsamling til delprøve 1 Matematik A HTX.

The procedural paradigm. (u.d.). Hentet 16. december 2024, fra https://adacomputerscience.org/concepts/procprog_paradigm

Shikha, G. (u.d.). *The Importance of Planning and Design in the Software Development Process*. Hentet 13. december 2024, fra https://blog.back4app.com/the-importance-of-planning-and-design-in-the-software-development-process/

What is Symbolic Computation in SymPy? (2022 februar). Hentet 15. december 2024, fra https://www.geeksforgeeks.org/what-is-symbolic-computation-in-sympy/

Kristoffer Sørensen
Tegn: 43594

10 Bilag

10.1 Bilag 1: Koden

```
    from sympy import symbols, solve, Eq, diff
import matplotlib.pyplot as plt
import numpy as np
import time

def leastSquaresMethod(dataPoints):
    startTime = time.time() # Gets the current time
    a, b = symbols('a b')
    sumA2 = 0 # Sum of a^2
    sumB2 = 0 # Sum of b^2
    sumAB = 0 # Sum of 2ab
    sumA = 0 # Sum of 2ax
    sumB = 0 # Sum of 2by
    sumConstant = 0
    n = len(dataPoints) # Gets the number of data points

    for i in range(n): # Loops through all data points and calculates the sums
        x = dataPoints[i][0] # Gets the x value of the data point
        y = dataPoints[i][1] # Gets the y value of the data point
        print(f"Data point {i}: x = {x}, y = {y}") # Prints what data point is being calculated
        sumA2 += a**2 * x**2 # Adds the sum of a^2
        sumB2 += b**2       # Adds the sum of b^2
        sumAB += 2*a*b*x    # Adds the sum of 2ab
        sumA += 2*a*x*y    # Adds the sum of 2ax
        sumB += 2*b*y     # Adds the sum of 2by
        sumConstant += 2*y # Adds the sum of 2c

    # Defines the function of two variabel containing the sum of squared errors
    f = sumA2 + sumB2 + sumAB - sumA - sumB + sumConstant

    # Solve for a and b
    diffA = diff(f, a) # Differentiates f to a
    diffB = diff(f, b) # Differentiates f to b
    solutions = solve([diffA, diffB], (a, b)) # Solves the equations
    print(solutions[a], solutions[b]) # Prints the solutions
```

```
36      timeElapsed = time.time() - startTime # Calculates the time
            elapsed
37      print(f"Time elapsed: {timeElapsed:.3f} seconds") # Prints
            the time elapsed
38      return solutions[a], solutions[b] # Returns the solutions as
            a tuple containing a and b [a, b]
39
40  def plotter(dataPoints):
41      line = leastSquaresMethod(dataPoints) # Calls the
            leastSquaresMethod function and stores the result in line
42      slope = float(line[0])  # Gets the slope of the line
43      intercept = float(line[1])  # Gets the intercept of the line
44
45      for point in dataPoints: # Loops through all data points and
            plots them
46          plt.plot(point[0], point[1], 'ro')
47
48      # Defince x values for the line
49      x = np.linspace(-10, 100, 100)  # Creates a interval of the
            line. The line will be drawn from -10 to 100
50      y = slope * x + intercept # Calculates the y values for the
            line
51
52      # Plot linjen
53      plt.plot(x, y, label=f"y = {slope:.3f}x + {intercept:.3f}",
            color='blue') # Plots the line
54      plt.quiver(x[0], y[0], -1, -slope, angles='xy', scale_units=
            'xy', scale=1, color='blue', width=0.003) # Adds an arrow
            to the start of the line
55      plt.quiver(x[-1], y[-1], 1, slope, angles='xy', scale_units=
            'xy', scale=1, color='blue', width=0.003) # Adds an arrow
            to the end of the line
56
57      # Akse og grid
58      plt.axhline(0, color='black', linewidth=0.8, linestyle='--')
            # Adds a horizontal line at y = 0
59      plt.axvline(0, color='black', linewidth=0.8, linestyle='--')
            # Adds a vertical line at x = 0
60      plt.title('Mindste Kvadraters Metode') # Adds a title to the
            plot
61      plt.xlabel('x - axis') # Adds a label to the x-axis
62      plt.ylabel('y - axis') # Adds a label to the y-axis
63      plt.legend() # Adds a legend to the plot
64      plt.grid() # Adds a grid to the plot
65      plt.show() # Shows the plot
66
```

Kristoffer Sørensen
Tegn: 43594

```
def main():
    print("\t welcome to the least squares method program")
    print("\n \t Choose an option")
    print("\n \t 1. Graphical output")
    print("\n \t 2. Numerical output")
    print("\n \t 2. Exit")
    userInput = input("Enter your choice: ")
    if userInput == "1":
        print("Enter the data points")
        dataPoints = [(1, 6), (5, 6), (6, 12), (10,10)]
        while True:
            x = input("Enter x value: ")
            y = input("Enter y value: ")
            dataPoints.append((int(x), int(y)))
            print("Do you want to add more data points?")
            choice = input("Enter y/n: ")
            if choice == "n":
                break
        plotter(dataPoints)
        main()
    elif userInput == "2":
        while True:
            x = input("Enter x value: ")
            y = input("Enter y value: ")
            dataPoints.append((int(x), int(y)))
            print("Do you want to add more data points?")
            choice = input("Enter y/n: ")
            if choice == "n":
                break
        plotter(dataPoints)
        main()
    elif userInput == "3":
        exit()
    else:
        print("Invalid input")
        main()

main()
```