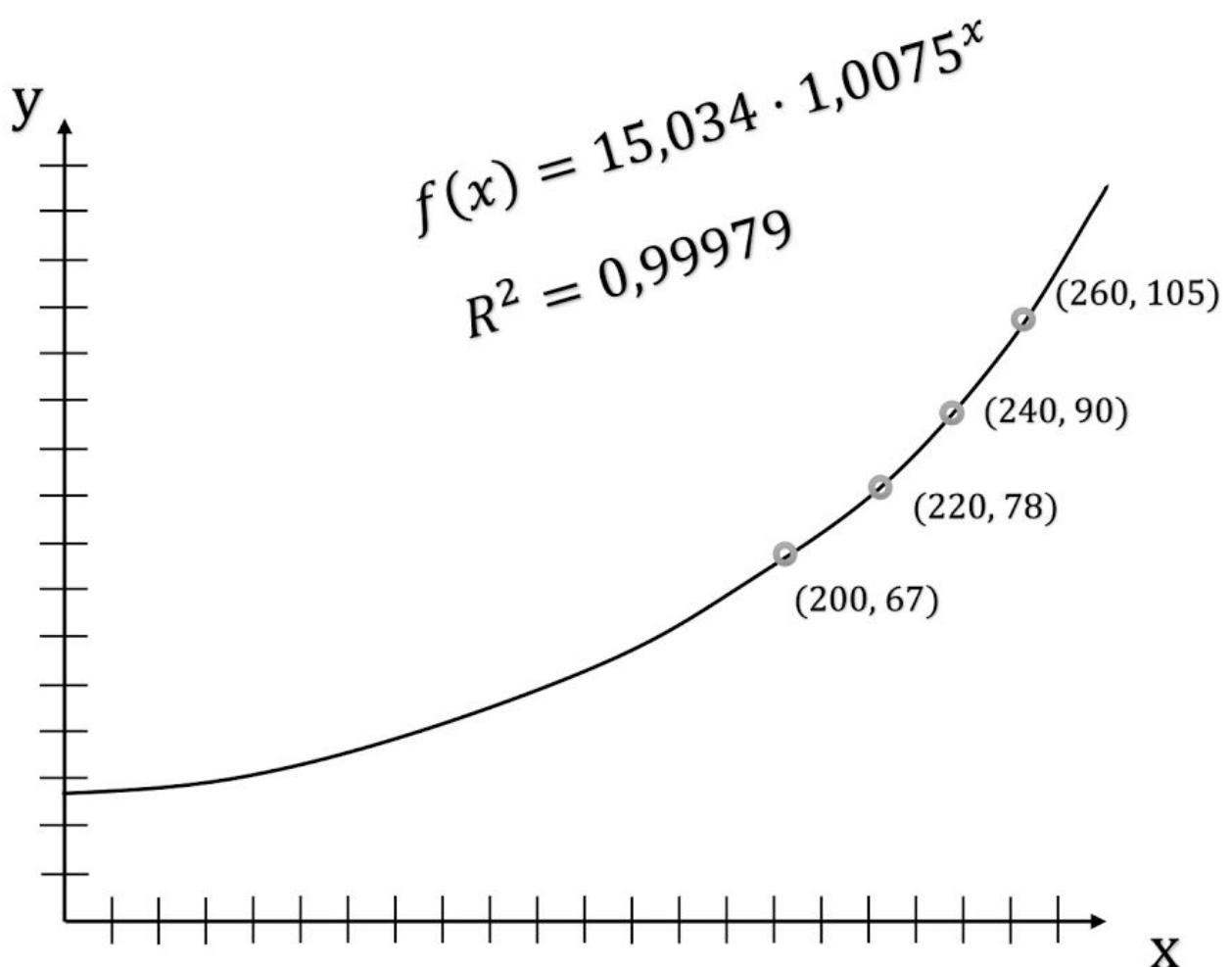


SO-projekt | Eksponentielle modeller

Matematik A - Informationsteknologi B



Vejledere:

- Karl G. Bjarnason (gkb)
- Jørn Christian Bendtsen (jcb)

Forfattere:

- Ferhat Akdeniz - 2.4
- Malthe Reuber - 2.4
- Mikael Zhang - 2.4

Aflevering:

- 27. november 2017. kl. 08:10

Tværfagligt projekt - Eksponentielle modeller

Tværfagligt projekt - Eksponentielle modeller

HTX → Roskilde

Matematik A & Informationsteknologi B - Efterår 2017

1. udgave

Denne rapport blev fremstillet i sammenhæng med et tværfagligt forløb i fagene matematik og informationsteknologi på det tekniske gymnasium HTX, og omhandler eksponentielle modeller. Rapporten danner ramme og dokumentation for elevernes evne til at bearbejde samt analysere et givet sæt forsøgsresultat. Eleverne skal vise at de kan udføre matematisk analyse, beregninger, dokumentere forløbet, og dertil udarbejde et IT-produkt. Studieområdet spiller en central rolle i forløbets arbejde. Eleverne vil vise deres evne til at anvende fagligt relevante studieteknikker og arbejdsmetoder, heriblandt at organisere og indgå i gruppearbejde, og vise deres fortroligheden med den naturvidenskabelige metode. Eleverne vil vise deres evne til at arbejde problemorienteret, samt producere viden om praktisk-teoretiske problemstillinger i samspillet mellem fag. Arbejdet formidles skriftligt, og der anvendes relevante IT-værktøjer under hele arbejdet.

Aflevering: 27. november 2017, kl. 08:10

Vejledere: Karl G. Bjarnason (gkb)
Jørn Christian Bendtsen (jcb)

Forfattere: 2.4 - Ferhat Akdeniz
2.4 - Malthe Reuber
2.4 - Mikael Zhang



Pulsen 4

DK - 4000 Roskilde

Tlf.: 61 94 85 46

<http://www.htxroskilde.dk>

Indholdsfortegnelse

1. Introduktion	4
2. Planlægning	4
2.1. Belbin roller (Vedhæft som Bilag)	4
2.2. Tidsplan	5
3. Opgaveanalyse	5
3.1. Opgavebeskrivelse	5
3.1.1. Krav fra opgaveanalyse	5
4. Matematisk Løsningsmodel	6
4.1. Opgaveanalyse	6
4.1.1. Krav	7
4.2. Løsningsforslag	7
4.3. Vurdering af løsningsforslag	8
4.4. Dokumentation af beregninger	8
4.4.1. 1. inkredsning	8
4.4.2. 2. indkredsning	11
4.4.3. 3. indkredsning	13
4.5. Vurdering af resultater	14
4.6. Delkonklusion	16
5. Dokumentation af IT-program	17
5.1. Indledende kravfangst	17
5.1.1. User stories	17
5.1.2. Lasswell's Kommunikationsmodel	17
5.2. IT-værktøjer og redskaber	18
5.2.1. Beskrivelse og begrundelse	18
5.3. Overordnet design	19
5.4. 1. Iteration af systemudvikling - Spike Solution	20
5.4.1. Planlægning	20
5.4.2. Kontrol af krav og testprocedurer	21
5.4.3. Design	21
5.4.4. Implementering	21

5.4.5. Test	24
5.5. 2. Iteration af systemudvikling	25
5.5.1. Planlægning	25
5.5.2. Kontrol af krav og testprocedurer	25
5.5.3. Design	25
5.5.4. Implementering	26
5.5.5. Test	31
5.6. Resultatopgørelse	33
6. Vurdering	33
6.1. Opfyldning af krav	33
6.2. Delkonklusion	34
7. Konklusion	34
8. Litteraturliste	35
9. Bilag	35
9.1. Tidsplan	35
9.2. Faglige mål	35
9.2.1. Matematik	36
9.2.2. Informationsteknologi	36
9.2.3. Overfaglige mål (studieområdet)	36

1. Introduktion

Dette er et tværfagligt projekt, som sammenspiller mellem informationsteknologi og matematik. Projektet omhandler analysering af gærcellers eksponentielle vækst. Heriblandt skal der undersøges løsningsmodeller, og udvikles et program der skal give mulighed for at hjælpe med fremtidige analyser. Der er anvendt relevant gymnasie niveaus matematik til udarbejdelsen af den matematiske løsningsmodel. Heriblandt regneregler og formler for eksponentielle funktioner. Der er sågar også anvendt relevante IT-værktøjer til udarbejdelse af IT-produktet. Heriblandt programmeringssproget Python, cross platform widget-toolkit wxPython, matplotlib plotting widgets fra WxMPlot og Python redux implementeringen pydux.

2. Planlægning

2.1. Belbin roller (Vedhæft som Bilag)

X	Ferhat	Malthe	Mikael
Idemand		x	
Opstarter			x
Koordinater		x	
Analysator	x		
Organisator		x	
Kontaktskaber			x
Specialist			x
Afslutter	x		
Formidler	x		

Som der kan ses i ovenstående tabel har vi en fordeling, som viser hvem og hvor man tilhører i forhold til Belbins gruppefordeling roller. Desuden ser vi, at der både er tale om en vis naturlig fordeling, samt en variation mellem fordelingerne, hvilket ses som en kæmpe fordel, da alle Belbin rollerne minimum tilhøre en person.

2.2. Tidsplan

Følgende tidsplan er taget fra opgavebeskrivelsen, og beskriver projektarbejdets mål for de individuelle perioder (uger). Tidsplanen er ikke en konkret plan, men en løs arbejdsfordeling plan.

Uge 44:

- Projektoplægget udleveres onsdag i uge 43.
- Bearbejdning af data indledes.
- Introduktion til IT-værktøjer.

Uge 45:

- Opstillingen af matematiske modeller.
- Analyse, design og implementering af brugergrænsefladen til IT-produktet.
- Fokus på den naturvidenskabelige metode.

Uge 46-47:

- Delaflevering af de matematiske beregninger og foreløbige modeller mandag i uge 46 senest kl. 8.10.
- Vurdering af de matematiske modeller og færdiggørelse af IT-produktet (*herunder test og fejlretning*).

3. Opgaveanalyse

3.1. Opgavebeskrivelse

Følgende opgave omhandler gærceller, som i et bestemt interval udvikler sig eksponentielt. Teorien siger, at når gærcellernes vækst for alvor er kommet i gang, vil der være en fase, hvor gærcellerne som sagt øges eksponentielt. Desuden vil der fornævnte fase være en fase, hvor gærcellerne akklimatiserer sig i opløsningen, og hvor antallet således ikke vokser. På et tidspunkt efter den eksponentielle udvikling vil øgningen i antallet af gærceller flade ud på grund af mangel af næringsstoffer og ophobning af affaldsstoffer, her vil den såkaldte stationære fase starte. Vores opgave er, at bearbejde, samt analyserer forsøgsresultaterne fra en matematisk synsvinkel, hvor der tilhørende skal opstilles flere matematiske modeller, der skal være med til at forklare data. Dette skal endvidere visualiseres ved hjælp af vores IT-produkt.

3.1.1. Krav fra opgaveanalyse

Desuden i forbindelse med analyserne af de empiriske data, samt formidlingen skal der blandt andet fokuseres på følgende aspekter:

Fokus aspekter	
1	indkredsning af en eventuel eksponentiel fase,
2	udarbejdelse af vækstkurver i forskellige typer koordinatsystemer,
3	bestemmelse af flere forskellige funktionsudtryk (matematiske modeller) til forklaring af væksten (herunder regressionsmodeller),
4	nærmere analyse af de forskellige funktionsudtryk, herunder beregning af fordoblingskonstanter,
5	vurderinger af de forskellige matematiske modellers evne til at afspejle væksten (data),
6	analyse af målgruppe og design af brugergrænseflade til IT-produktet,
7	giv brugeren mulighed for at ændre på noget af det visuelle, f.eks. farve, tekster på akserne, intervaller, punkter og/eller grafen
8	implementering af to eller flere prototyper, samt dokumentation og test af IT- produktet.
9	publicering af IT-produktet på StudieWeb, beskrivelse af forløbet,
10	arbejdet planlægges, gennemføres, dokumenteres og eventuelt præsenteres ved hjælp af en systemudviklingsmetode (som f.eks. MedieLabs metode).

4. Matematisk Løsningsmodel

4.1. Opgaveanalyse

Ud fra et matematisk synspunkt omhandler dette projektoplæg som sagt om eksponentielle udviklinger ud fra en given tabel over gærcellers udvikling. Herved skal vi analysere og opstille matematiske modeller. De givne forsøgsresultater over gærcellernes udvikling kan ses på tabel 4.1.

På tabellen ses der et tidsinterval fra 0-320 minutter, hvor antallet af gærceller er stigende med en start mængde på 15 gærceller og en slut mængde på 125 gærceller. Der er altså sket en ændring i antallet af gærceller over de 320 sekunder. Vi kan vurdere om denne ændring i gærcellernes antal under tidsintervallede er eksponentiel i et selvvalgt interval.

Tid i minutter	Antal gærceller
0	15
20	16
40	18
60	19
80	22
100	24
140	31
160	42
180	49
200	67
220	78
240	90
260	105
280	109
300	122
320	125

Tabel 4.1.

4.1.1. Krav

Kravene for denne opgave ud fra et matematisk synspunkt er at vælge et interval i det givne forsøgsresultat, hvor vi skal vurdere og analysere om der sker en eksponentiel udvikling. De overordnede krav kan ses nedenfor.

- *Udarbejdning af vækstkurve forskellige typer koordinatsystemer.*
- *Opstilling af funktionsudtryk, samt bestemmelsen og vurdering af regressionsmodeller.*
- *Analysere funktionsudtryk, herunder funktionsudtrykkets fordoblingskonstant.*
- *Vurdere matematiske modellers evne til at afspejle væksten som data.*

4.2. Løsningsforslag

Ud fra analysen og kravene for opgaven skal vi tage udgangspunkt i et selvvalgt interval som vi formoder er en eksponentiel udvikling. Dette er muligt ved at beregne fremskrivningsfaktoren for hver spring i forsøgsresultaterne. Herved har vi opstillet ‘Tid i minutter’ som x-værdierne og ‘Antal gærceller’ som y-værdierne. Vi skal som sagt finde fremskrivningsfaktoren. En fremskrivningsfaktor er en procentvis ændring i en sekvens, som bestemmer om den eksponentielle udvikling er stigende eller aftagende. Udtrykket for fremskrivningsfaktoren lyder som følgende:

$$a = \sqrt[x_2 - x_1]{\frac{y_2}{y_1}} = \left(\frac{y_2}{y_1}\right)^{\frac{1}{x_2 - x_1}}$$

Som der kan ses ud fra ovenstående udtryk for fremskrivningsfaktoren (a), skal vi kende to punkter (x_1, y_1) og (x_2, y_2) , dette gør vi selvfølgelig da vi har en tabeloversigt over forsøgsresultaterne. Dette gentages med alle værdierne i tabellen. Nu kan man se hvor gærcellerne opfører sig som en eksponentiel udvikling ud fra om deres fremskrivningsfaktorer er tætliggende. Vi vil som sagt gerne opstille et funktionsudtryk over intervallet. Funktionsudtrykket for en eksponentiel model bliver defineret som:

$$f(x) = b * a^x$$

Hvor på b er grundværdien, a er den gennemsnitlige fremskrivningsfaktorer for intervallet. Vi skal altså finde den gennemsnit fremskrivningsfaktor for vores interval. For at bestemme den gennemsnitlige fremskrivningsfaktorer i et interval, benytter vi følgende udtryk.

$$a_{gennemsnit} = \sqrt[n]{a_1 * a_2 * a_3 * \dots}$$

Hvor på antallet af fremskrivningsfaktorer er defineret som variablen n , og de fremskrivningsfaktorer man vil tage gennemsnittet af er a_1, a_2, a_3 , osv.

For at beregne b indsætter man den første værdi i intervallet som y og isolerer b .

$$y = b * a^x$$

⇓

$$b = \frac{y}{x}$$

Nu har vi de nødvendige informationer til at lave en eksponentiel funktion. Da vi har funktionsudtrykket kan vi også opstille vores regressionsmodel, samt analysere grafen, ved hjælp af et kartesisk og enkelt logaritmisk koordinatsystem. Derudover kan vi også beregne og aflæse funktionens fordoblingskonstant, altså gærcellernes fordoblingshastighed. For at beregne fordoblingskonstanten kan man benytte udtrykket som lyder:

$$T_{fordobling} = \frac{\log(2)}{\log(a)}$$

Det er altså tydeligt at vi kun har behov for at benytte fremskrivningsfaktoren, som selvfølgelig kan aflæses fra funktionsudtrykket til at beregne fordoblingskonstanten.

Heresfter kan vi vurdere grafen ved hjælp af enten vores IT-program eller på en anden programmel måde. Hvorefter kan vi indkredse vores interval endnu mere, således at vi får en mere præcis eksponentiel funktion i forhold til punkterne, det vil altså sige en korrelationskoefficient nær 1.

4.3. Vurdering af løsningsforslag

Som sagt har vi fået gennemgået ét løsningsforslag, som kan benyttes til at fuldføre problemstillingen på en meget passende måde. Det skal dog understreges, at der findes en masse andre måder, at løse opgaven på, men dog har vi taget udgangspunkt i følgende løsningsforslag, hvor vi kan konkludere at det er muligt at benytte følgende løsningsforslag til at afkræfte/bekræfte empirien om gærcellernes eksponentielle udvikling :D

4.4. Dokumentation af beregninger

4.4.1. 1. inkredsning

Som gennemgået i løsningsforslaget (punkt 4.2.), har vi tænkt os at opstille en funktionsforskrift og analysere og vurdere data. Vi starter ligesom i løsningsforslaget med at beregne fremskrivningsfaktoren mellem punkterne. Vi regner fremskrivningsfaktoren af 0-20 min og 15-16 gærceller, herved:

$$a = \sqrt[x_2-x_1]{\frac{y_2}{y_1}} = \left(\frac{y_2}{y_1}\right)^{\frac{1}{x_2-x_1}}$$

(Indsættelse af værdier)

Tid i minutter	Antal gærceller	Fremskrivningsfaktor
0	15	1,0032
20	16	1,0059
40	18	1,0027
60	19	1,0074
80	22	1,0044
100	24	1,0054
140	31	1,0153
160	42	1,0077
180	49	1,0158
200	67	1,0076
220	78	1,0072
240	90	1,0077
260	105	1,0019
280	109	1,0056
300	122	1,0012
320	125	-

Tabel 4.4.

$$a = \sqrt[20-0]{\frac{16}{15}}$$

⇓

$$a = \sqrt[20]{\frac{16}{15}}$$

⇓

$$a = 1,003232 \approx 1,0032$$

Dette gentager vi som sagt med alle punkter. Herved kan vi lave en tabel over fremskrivningsfaktoren, se tabel 4.4.

Vi har derudover som gennemgået i løsningsforslaget og kravene for oplægget tænkt os at opstille en funktionsforskrift for den eksponentielle udvikling. Men for at opstille funktionsforskriften skal vi først finde den gennemsnitlige fremskrivningsfaktor af vores interval. Herved har vi tænkt os at vælge intervallet 60-240 minutter for gærcellerne.

Da vi ved at der ikke er tale om en eksponentiel udvikling i starten af intervallet samt i faldet til sidst, da der er så stor forskel på fremskrivningsfaktorerne.

Vi finder den gennemsnitlige fremskrivningsfaktor for vores interval på følgende måde:

$$a_{gennemsnit} = \sqrt[n]{a_1 * a_2 * a_3 * \dots * a_n}$$

⇓

$$a_{gennemsnit} = \sqrt[8]{1,0074 * 1,0044 * 1,0054 * 1,0153 * 1,0077 * 1,0158 * 1,0076 * 1,0072} = 1,008842$$

Nu har vi fremskrivningsfaktoren som er 1,0088 og en vækstrate på:

$$r = a - 1$$

⇓

$$r = 1,0088 - 1$$

⇓

$$r = 0,0088\%$$

En vækstrate betyder procentvis ændring, i vores tilfælde med vores interval har vi en ændring på 0,0088 % og altså en positiv vækstrate som også beskriver at den igen er en voksende eksponentiel funktion.

Vi kan opstille en funktionsforskrift, således.

$$f(x) = b * a^x$$

⇓ (indsættelse af fremskrivningsfaktor)

$$f(x) = b * 1,0088^x$$

⇓ (indsættelse af første værdi fra interval som y)

$$19 = b * 1,0088^x$$

⇓ (isolering af b)

$$b = \frac{19}{1,0088^x}$$

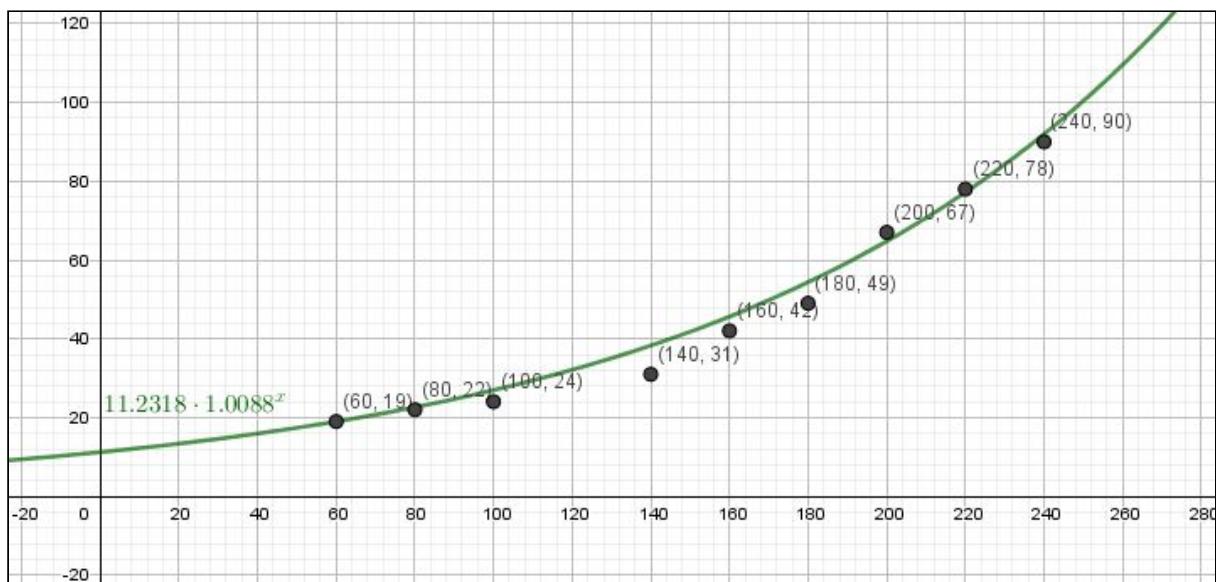
∅ (da der stadig er tale om første værdi fra intervallet,
må x være lig med 60)

$$b = \frac{19}{1,0088^{60}}$$

$$b = 11,2318$$

Funktionsforskriften for vores interval ser derfor sådan ud:

$$f(x) = 11,2318 * 1,0088^x$$



Vi kan ud fra denne funktionsforskrift aflæse at, hvis $x = 0$ vil funktionen skærer y -aksen i 11,232. Desuden ser vi, at vores nye interval viser mere eksponentielt fremgang.

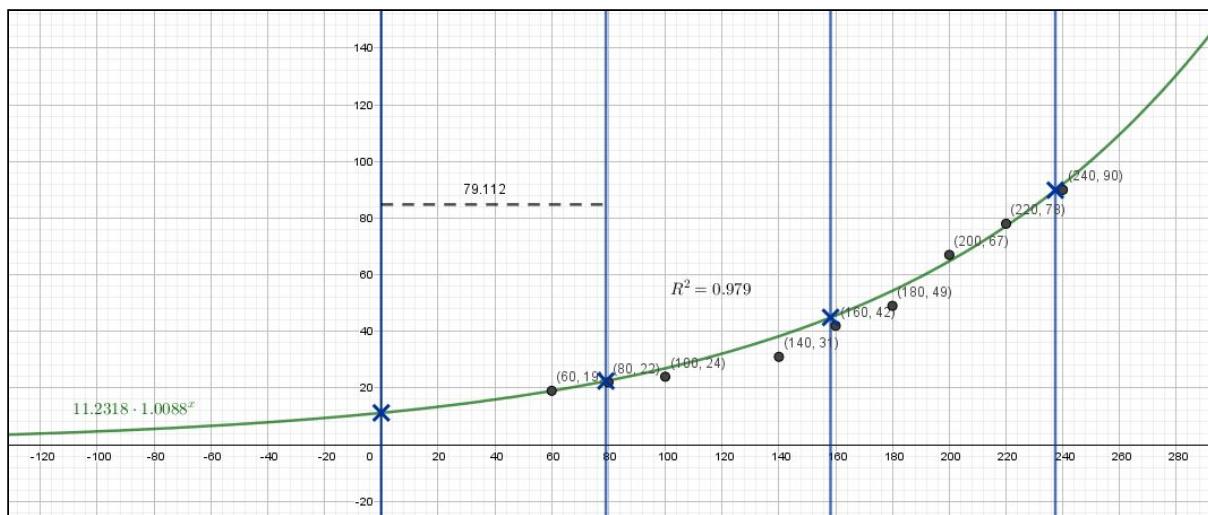
Derudover kan vi også bestemme fordoblingskonstanten af vores funktionsforskrift ved hjælp af følgende udtryk.

$$T_{fordobling} = \frac{\log(2)}{\log(a)}$$

Vi har som sagt kun behov for at kende fremskrivningsfaktoren for at bestemme fordoblingskonstanten.

$$T_{fordobling} = \frac{\log(2)}{\log(1,0088)} = 79,11279 \approx 79,112$$

Fordoblingskonstanten beskriver hvornår vores y -værdi bliver fordoblet. I dette tilfælde er fordoblingskonstanten 79,112 som så siger at når vi går 79,112 hen ad x -aksen, uanset hvor vi starter vil y -værdien være fordoblet. Dette betyder konkret at hver gang der er gået 79,112 minutter vil antallet af gærceller være fordoblet. Dette kan ses på nedenstående graf.



Vi kan derudover som sagt også finde R^2 af vores funktionsudtryk, igen ved hjælp af programmel. Regressionen kan findes på ovenstående graf, hvorpå korrelationskoefficienten er lig med 0,97902, hvilket vil sige, at der er tale om en meget god model, da korrelationskoefficienten er meget tæt på 1. Men for at få den bedst mulige interval, hvor vi kan bekræfte/afkræfte hypotesen/empirien skrumper vi intervallet lidt mere.

4.4.2. 2. indkredsning

Nu indkredser vi vores interval endnu mere således at vi kan forvente at få en bedre tilnærmelse værdi i forhold til det maksimale korrelationskoefficient, 1. Herved vælger vi følgende interval som kan ses til højre på tabel 4.4.1.

Vi har altså valgt intervallet således at vores intervaller er som følgende:

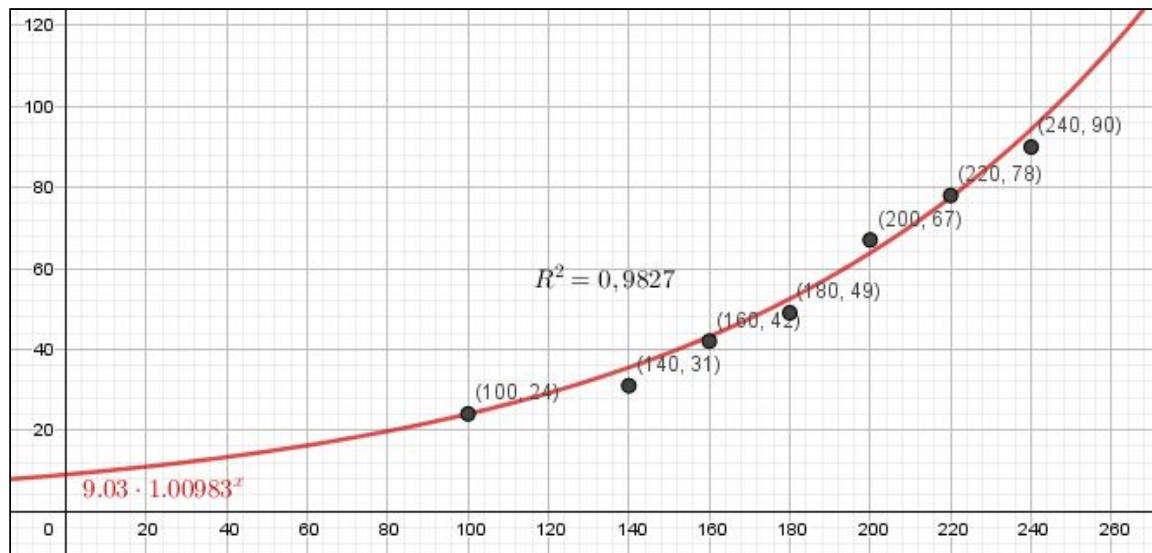
$$\begin{aligned} \text{gærceller} &: [24 : 90] \\ \text{tidsinterval} &: [100 : 240] \end{aligned}$$

Vi har gentaget alle beregner fra ovenstående og fra løsningsforslaget til at finde frem til at funktionsudtrykket for vores nuværende interval er som følgende:

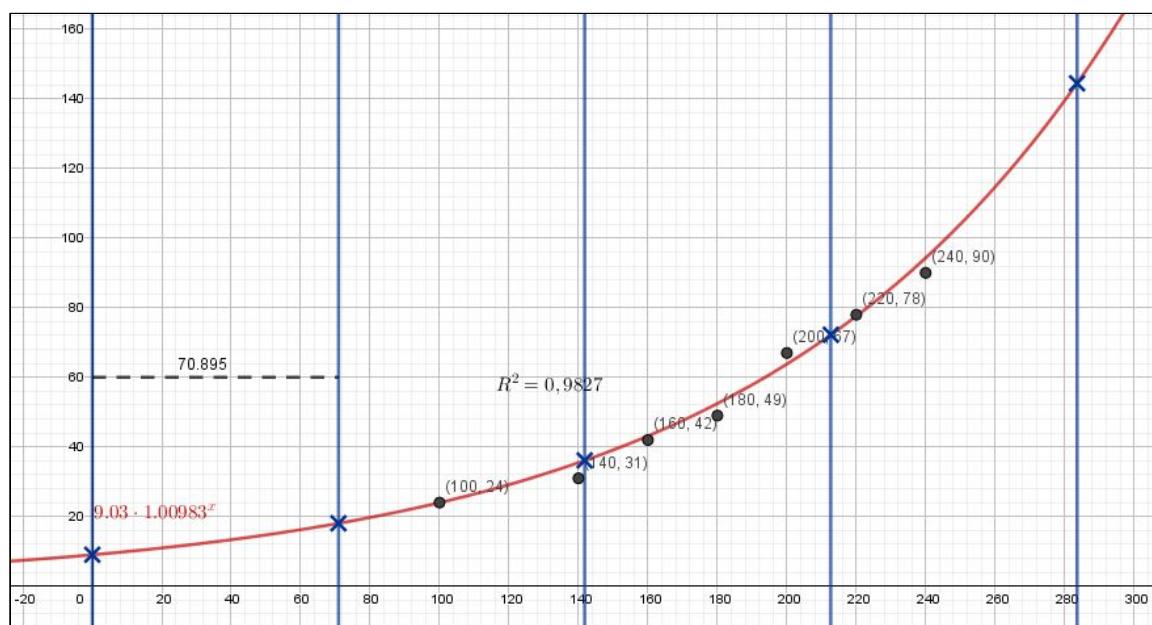
$$g(x) = 9,028 * 1,0098^x$$

Tid i minutter	Antal gærceller	Fremskrivningsfaktor
100	24	1,0054
140	31	1,0153
160	42	1,0077
180	49	1,0158
200	67	1,0076
220	78	1,0072
240	90	-

Tabel 4.4.1.



Som der kan aflæses fra funktionsforskriften er skæringen i y -aksen lig med 9,028. Samt er fordoblingskonstanten lig med 70,895. Som der kan ses på nedenstående graf, er der ligesom før visuelt vist hvordan fordoblingskonstanten ser ud i forhold til start i $x = 0$.



Derudover er korrelationskoefficienten også kommet nærmere til 1, som vil sige at værdierne i intervallet er mere sammenhængende. $R^2 = 0,9827$.

4.4.3. 3. indkredsning

Vi indkredser intervallet endnu en gang, intervallet kan ses på tabel 4.4.2. Bemærk at der faktisk ikke er sket en indkredsning fra 2. indkredsning, men derimod en indkredsning fra den 1. indkredsning. Dette kan ses på at vi har 260 minutter med, som ikke var med i forrige indkredsning.

Man kan selvfølgelig også beskrive intervallet som vi har gjort tidligere, derfor:

$$\text{gærceller} : [67 : 105]$$

$$\text{tidsinterval: } [200 : 260]$$

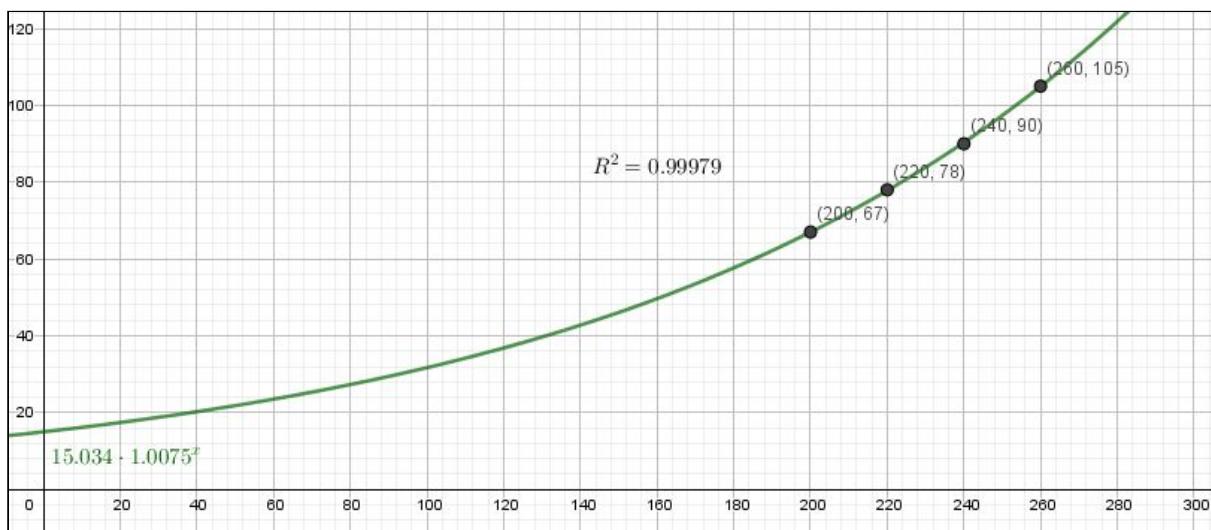
Som vi gjorde med tidligere interval er dette blot endnu en gentagelse af beregninger, derfor kommer vores funktionsforskrift til at se således ud.

$$h(x) = 15,034 * 1,0075^x$$

Tid i minutter	Antal gærceller	Fremskrivningsfaktor
200	67	1,0076
220	78	1,0072
240	90	1,0077
260	105	-

Tabel 4.4.2.

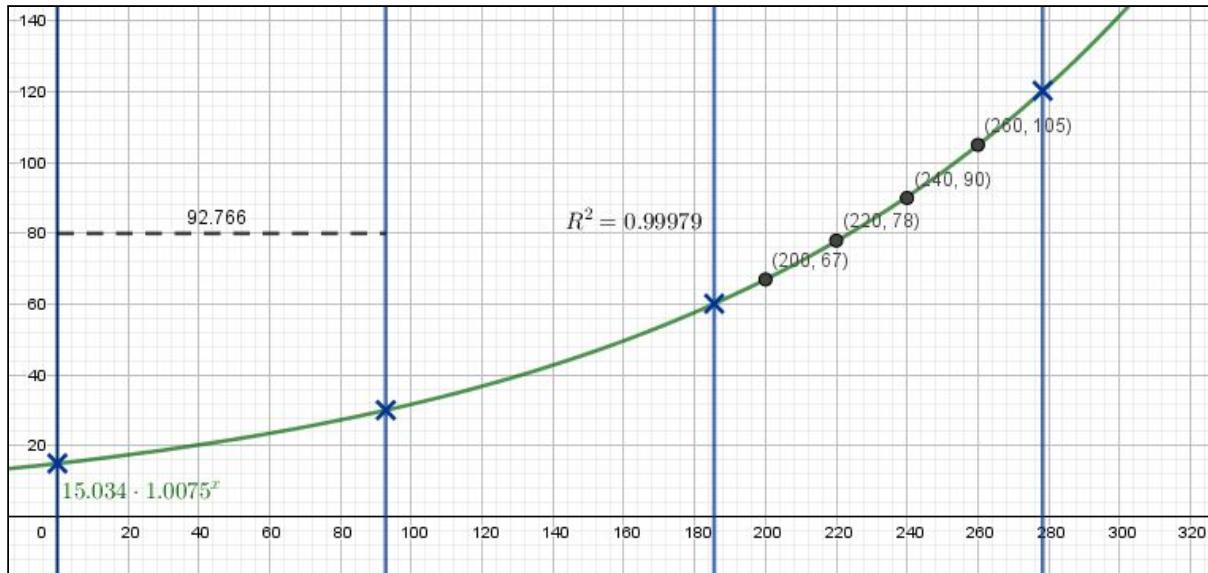
En graf for funktionsforskriften, punkterne og korrelationskoefficienten kan ses nedenfor:



Som man kan se er korrelationskoefficienten rigtigt tæt på 1, korrelationskoefficienten er nemlig $R^2 = 0,99979$, hvilket vil sige at der er en sammenhæng mellem punkterne, og ikke mindst vores funktionsudtryk. Derudover kan vi også finde fordoblingskonstanten for funktionsudtrykket, herved:

$$T_{fordobling} = \frac{\log(2)}{\log(1,0075)} = 92,766$$

Dette betyder som sagt at fordoblingskonstanten er 92,766. Nedenfor ses en vores funktion og fordoblingskonstanten med start i $x = 0$.



4.5. Vurdering af resultater

Vi kan hermed konkludere, at vi både har regnet os frem til hver enkelt beregning, hvor vi dernæst har vurderet vores resultater via programmel, herunder har vi brugt Geogebra som vurderingsværktøj. Ser vi nærmere på beregninger ser vi at de stemmer overens med vurderingerne af programmel. Derudover kan vi også vurdere at vores 3. Indkredsning er den bedste i forhold til korrelationskoefficienten. En tabel over de forskellige korrelationskoefficienter kan ses på tabel 4.5. Derudover kan vi også vurdere afvigelsen fra funktionsforskrift til punkt og finde den gennemsnitlige afvigelse i hver graf. For dette benytter vi blot følgende udtryk:

Indkredsning	Regressionsværdi
1.	0,97902
2.	0,9827
3.	0,9998

Tabel 4.5.

$$P_{afvigelse} = \frac{(målt værdi - tabelværdi)}{tabelværdi} * 100 \%$$

For at kunne benytte udtrykket skal vi først kende funktionsudtrykkets værdi for det ønskede punkts x -værdi. Vi tager udgangspunkt i 1. Indkredsning, det vil sige $f(x) = 11,2318 * 1,0088^x$ og punkt $(60; 19)$. Vi indsætter blot x -værdien på x 's plads.

$$f(60) = 11,2318 * 1,0088^{60}$$

$$f(60) = 19$$

Og nu kan vi finde afvigelsen, derfor:

$$P_{afvigelse} = \frac{19-19}{19} * 100 \% = 0 \%$$

Dette gentager vi med alle punkterne og alle 3. Indkredsninger. Dette kan ses på nedenstående tabeller 4.5.1., 4.5.2. og 4.5.3.

1. Indkredsning med $f(x)=11,2318*1,0088^x$		
Punkt	Punkt i forhold til funktionsudtryk	Afvigelse
(60; 19)	(60; 19)	0 %
(80; 22)	(80; 22,64)	2,9 %
(100; 24)	(100; 26,97)	12,38 %
(140; 31)	(140; 38,3)	23,55 %
(160; 42)	(160; 45,63)	8,64 %
(180; 49)	(180; 54,37)	10,96 %
(200; 67)	(200; 64,78)	-3,31 %
(220; 78)	(220; 77,19)	-1,04 %
(240; 90)	(240; 91,97)	2,19 %
Gennemsnit afvigelse:	7,22 % afvigelse til punkterne	

Tabel 4.5.1

Afvigelsen er en måde ligesom korrelationskoefficienten med til at vurdere punkterne. Det vil sige at desto mindre afvigelse til punkterne, desto mere præcis funktionsudtryk har man i forhold til punkterne. Som der også blev vurderet før er 3. indkredsning igen den bedste, da grafen har den laveste afvigelse til punkterne.

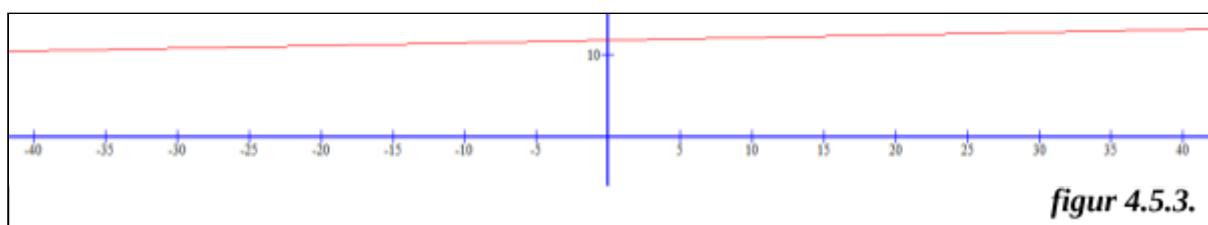
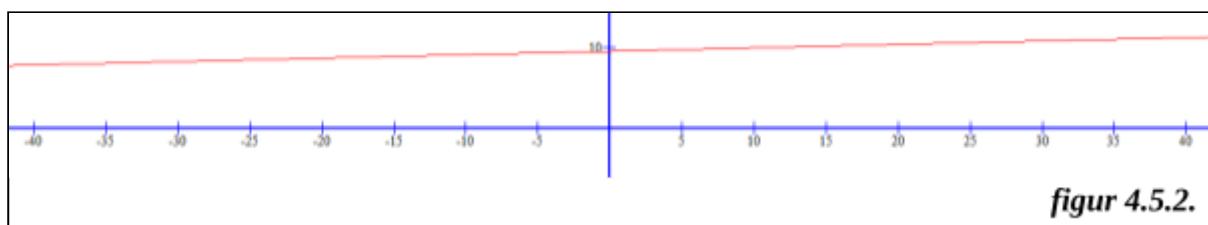
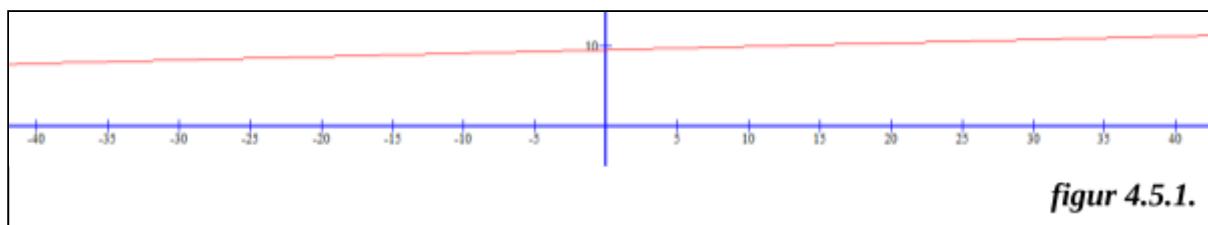
Vi kan derudover også vurdere graferne i et enkeltlogaritmisk koordinatsystem i stedet for et kartesisk koordinatsystem. Se figur 4.5.1., 4.5.2. Og 4.5.3, hvorpå figur 4.5.1. Visualiserer funktionsudtrykket fra 1. indkredsning, altså $f(x)=11,2318*1,0088^x$ i et enkeltlogaritmisk koordinatsystem og 4.5.2. Er for 2. indkredsning, $g(x)=9,028*1,0098^x$ og 4.5.3. Er selvfølgelig for 3. indkredsning, $h(x)=15,034*1,0075^x$.

2. Indkredsning med $g(x)=9,028*1,0098^x$		
Punkt	Punkt i forhold til funktionsudtryk	Afvigelse
(100; 24)	(100; 23,94)	-0,25 %
(140; 31)	(140; 35,36)	14,06 %
(160; 42)	(160; 43,98)	4,71 %
(180; 49)	(180; 52,23)	6,59 %
(200; 67)	(200; 63,48)	-5,25 %
(220; 78)	(220; 77,16)	-1,08 %
(240; 90)	(240; 93,77)	4,19 %
Gennemsnit afvigelse:	5,16 % afvigelse til punkterne	

Tabel 4.5.2

3. Indkredsning med $h(x)=15,034*1,0075^x$		
Punkt	Punkt i forhold til funktionsudtryk	Afvigelse
(200; 67)	(200; 67)	0 %
(220; 78)	(220; 77,8)	-0,25 %
(240; 90)	(240; 90,34)	0,38 %
(260; 105)	(260; 104,9)	-0,1 %
Gennemsnit afvigelse:	0,1825 % afvigelse til punkterne	

Tabel 4.5.3.



Man kan altså se at alle graferne er nogenlunde lineære i et enkeltlogaritmisk koordinatsystem, hvilket altså betyder at de er eksponentielle.

4.6. Delkonklusion

Alt i alt kan vi konkludere, at vi har fået afgrænset vores datasæt 3 gange. Første intervals grænse havde en afvigelse på 7,22% fra punkterne, hvilket gav en korrelationskoefficient på 0,97902, hvorimod anden indkredsning havde en afvigelse på 5,16% og en korrelationskoefficient på 0,9827. Vores endelige afgrænset interval havde en lille afvigelse på 0,1825%, og en korrelationskoefficient på 0,99979, hvilket er ekstremt højt og i den grad tæt på 1.

Vi havde en hypotese/empiri om, at der var tale om en eksponentiel fremgang i datasættene. Dette skulle vi enten afkræfte eller bekræfte ved at analysere på dataene. For at gøre bekræftelsen mest mulig sikker har vi indkredset datasættene 3 gange. Den 3. indkredsning har altså en funktionsforskrift som følgende, $h(x) = 15,034 * 1,0075^x$, en fordoblingskonstant på $T_{fordobling} = 92,766$ og selvfølgelig en korrelationskoefficient på $R^2 = 0,99979$ og en gennemsnitlig afvigelse på 0,1825 %. Vi kan hermed konkludere, at vi kan bekræfte hypotesen.

5. Dokumentation af IT-program

5.1. Indledende kravfangst

5.1.1. User stories

User Story 1

1. (Hvem): Jeg er en elev på HTX-Roskilde.
2. (Hvad): Som gerne vil lære, om hvordan systemudviklingen fungere på en simpel men informativ måde.
3. (Hvorfor/fordi): Fordi jeg gerne vil bruge det i min fremtid.

User Story 2

1. (Hvem): Jeg er lærer på HTX-Roskilde.
2. (Hvad): Som gerne vil lære mine elever om systemudvikling.
3. (Hvorfor/fordi): Fordi at jeg gerne vil bruge det både i min undervisning samt kan de bruge det i deres karriere.

User Story 3

1. (Hvem): Jeg er en elev på Roskilde Gymnasium.
2. (Hvad): Som gerne vil lære om eksponentielle udviklinger.
3. (Hvorfor/fordi): Fordi jeg gerne vil programmerer ét spil.

5.1.2. Lasswell's Kommunikationsmodel



5.1.2.1. Hvem siger

Afsenderen er os som en gruppe som består af Ferhat, Malthe og Mikael. Vi er tre gymnasieelever, som er i gang med et SO-tværfagligtprojekt sammen med Matematik & IT.

5.1.2.2. Hvad

Det vi som afsender har til formål er, at afkræfte en matematisk hypotese som omhandler eksponentiel udvikling. I store træk i hvilke intervaller, der er tale om eksponentiel udvikling og hvorfor. Desuden vil vi på baggrund af følgende udvikle et IT-produkt, som har formål at lette sværhedsgraden for gymnasieelever til, at finde og analyse eksponentiel funktioner.

5.1.2.3. Gennem hvilken kanal

Vi har tænkt os at benytte vores studieweb som et medie og måske Facebook til at ramme så mange gymnasieelever som muligt.

5.1.2.4. *Til hvem*

Modtageren er gymnasieelever og personer med en interesse for matematik og/eller programmering.

5.1.2.5. *Med hvilken effekt*

Formålet eller retter sagt hensigten er, at hjælpe gymnasieelever i færd med læring om eksponentielle udviklinger, i den forstand programmet skal hjælpe gymnasieleverne huske matematiske begreber under emnet.

5.2. IT-værktøjer og redskaber

Værktøj	URL
Python	https://www.python.org/
wxPython	https://www.wxpython.org/
WXMPlot	https://newville.github.io/wxmplot/
pydux	https://github.com/usrlocalben/pydux

5.2.1. Beskrivelse og begrundelse

Værktøj	Beskrivelse	Begrundelse
Python	<i>Programmeringssprog</i>	<i>Vi vil bruge dette sprog til at programmere vores program i.</i>
wxPython	<i>Widget Toolkit - wxWidgets til Python</i>	<i>Vi vil bruge dette til at lave grafisk brugergrænseflade.</i>
WXMPlot	<i>Matplotlib plotting widgets til wxPython</i>	<i>Vi vil bruge dette til at tegne grafer.</i>
Pydux	<i>Redux implementering i Python</i>	<i>Vi vil bruge dette til at holde styr på vores programs stadie, for at holde styr på alt.</i>

Python er et programmeringssproget vi har tænkt os at bruge. Det nemt og godt. Da det er noget vi alle kender, giver det den bedste mulighed for vores IT-produkt.

Det er ikke altid nemt at lave brugergrænseflade. Med wxPython er der indbygget widgets fra wxWidgets, der kan styre i Python. Disse widgets er cross platform, og giver et native look på hvert operativsystem, da de bruger hvert systems egne widgets, heriblandt GTK+ for Linux og Cocoa for macOS.

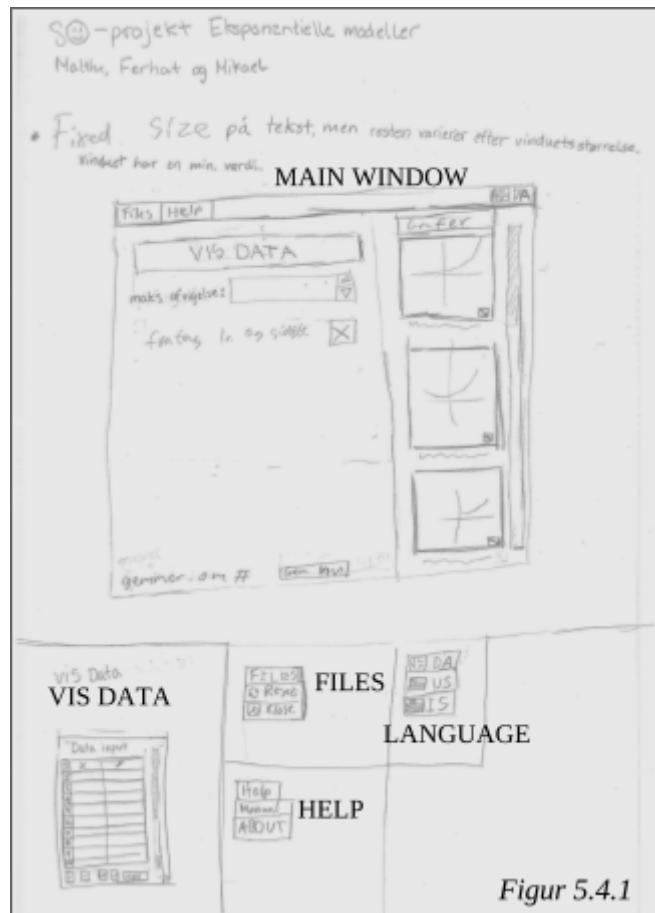
Når det gælder plotting og grafer er matplotlib en af de klart mest populære valg. De har også deres egne widgets til wxPython, men WXMPlot tilføjer nogle widgets der giver nogle flere muligheder, og mindre forarbejde. Disse gør det nemt at lave grafer.

Redux er hen af de seneste par år blevet MEGET populært i webudviklingsbranchen. Redux er en vidreudvikling af Facebooks "Flux" arkitektur til brugergrænseflader. Nu er Redux også blevet implementeret i Python, af flere endda. Vi har valgt at bruge den af usrlocalben på Github. Redux giver mulighed for have en state container hvor stort set hele programmet udfolder sig fra, hvilket gør det meget nemmere at håndtere programmet, og især effektivt til brugergrænseflader.

5.3. Overordnet design

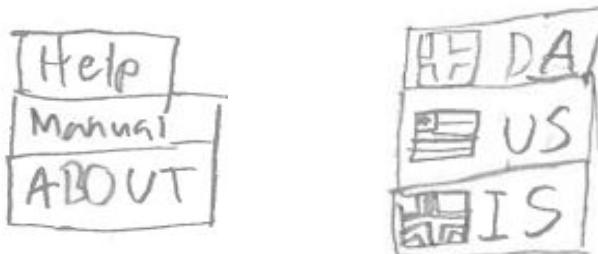
Som der kan ses på figur 5.4.1 har vi lavet en arbejdstegning, som visualisere vores tankegang omkring vores ideér før arbejdet med det endelige IT produkt.

Der ses yderligere, at vi har et par pop-up vinduer. Herunder selve start vinduet (*main vindue*), statistik af funktionerne, tabel over data, en menu bar som indeholder nogle overordnede kontekstmenuer (*files, help & language*). Vores files kontekstmenu indeholder følgende sektioner, reset og close, hvor at reset giver brugeren mulighed for at nulstille alle brugerdefinerede data og starter på ny, og til sidst har vi close kontekstmenuen som giver brugeren mulighed for at lukke hele programmet.



Figur 5.4.1

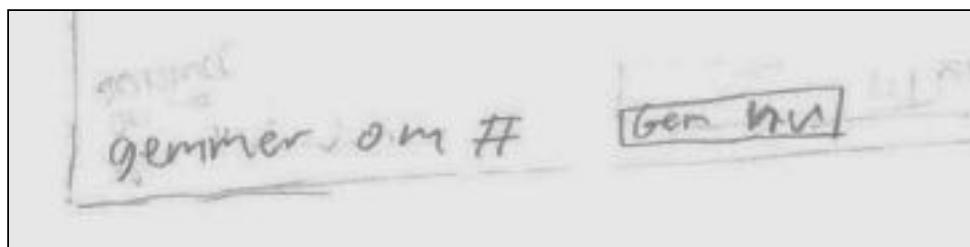
Desuden har vi en help sektion, som indeholder en manual og about kontekstmenuer, hvor at manual sektionen giver brugeren muligheden for at få en kort og konkret beskrivelse af hvordan programmet fungere, samt hvordan brugeren kan bruges. I sidste ende har vi about menuen, som kort fortæller om hvem vi er.



Dernæst har language kontekstmenuen, som giver brugeren mulighed for at skifte mellem diverse sprogmuligheder, herunder dansk (DA), engelsk (US) og islandsk (IS).

Derudover har vi et popup statistik vindue, som viser diverse oplysninger omkring vores funktionsgraf, herunder definitionsmængden (Dmf), værdimængden (Vm), skæring i y-aksen samt x-aksen, fremskrivningsfaktorer, grundværdi og selve funktionsforskriften. Derudover vil der forekomme en visualisering af funktionsforskriften i venstre del af vinduet.

Vi har også planlagt at gøre således at programmet automatisk gemmer brugerens input hvert x sekund, men samtidig skal der være mulighed for brugeren at gemme manuelt. Dette kan ses på skitsen nedenfor.



5.4. 1. Iteration af systemudvikling - Spike Solution

5.4.1. Planlægning

Vi har tænkt os lave små prøver med de værktøjer, vi har tænkt os at bruge. Dette sikre at vi ved, at vi kan gøre, hvad vi vil, og forhindre mulige fremtidige problemer.

5.4.2. Kontrol af krav og testprocedurer

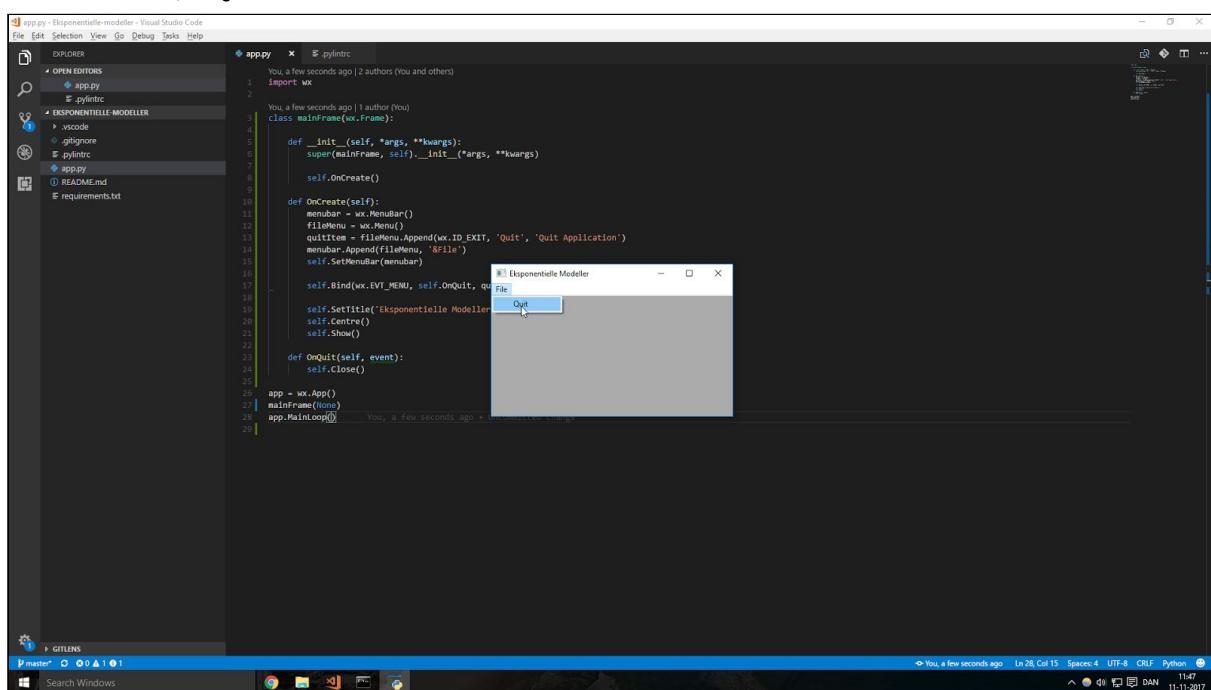
Kan programmet gøre de basiske ting vi har brug for?

5.4.3. Design

Programmet skal indeholde alle de elementer af programmet vi tænkt os bruge. Disse kan findes på vores tegning. Til vores spike solution, skal der være en simpel menubar, måske med en luk knap. Der skal en knap der skal starte programmet. Denne knap skal dispatche en action vidre gennem en pydux store. Der skal så være et panel der tegner en eksponentiel graf med wxmplot. Der skal dertil også være en anden knap der åbner et pop up vindue med en tabel.

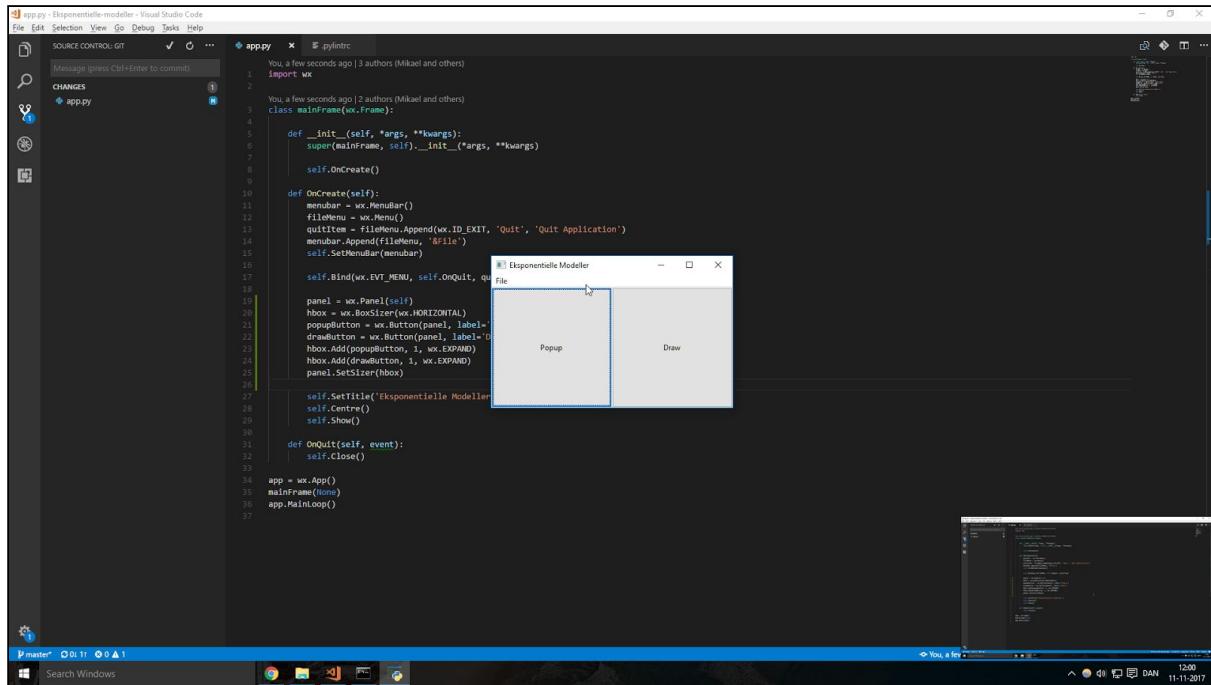
5.4.4. Implementering

Vi startede med kun wxPython, og lavede vinduet (Frame). Vi lavede et tomt vindue med en menubar. Denne menu bar indeholder en menu item, "File". File indeholder kun en knap i context menu, "Quit".

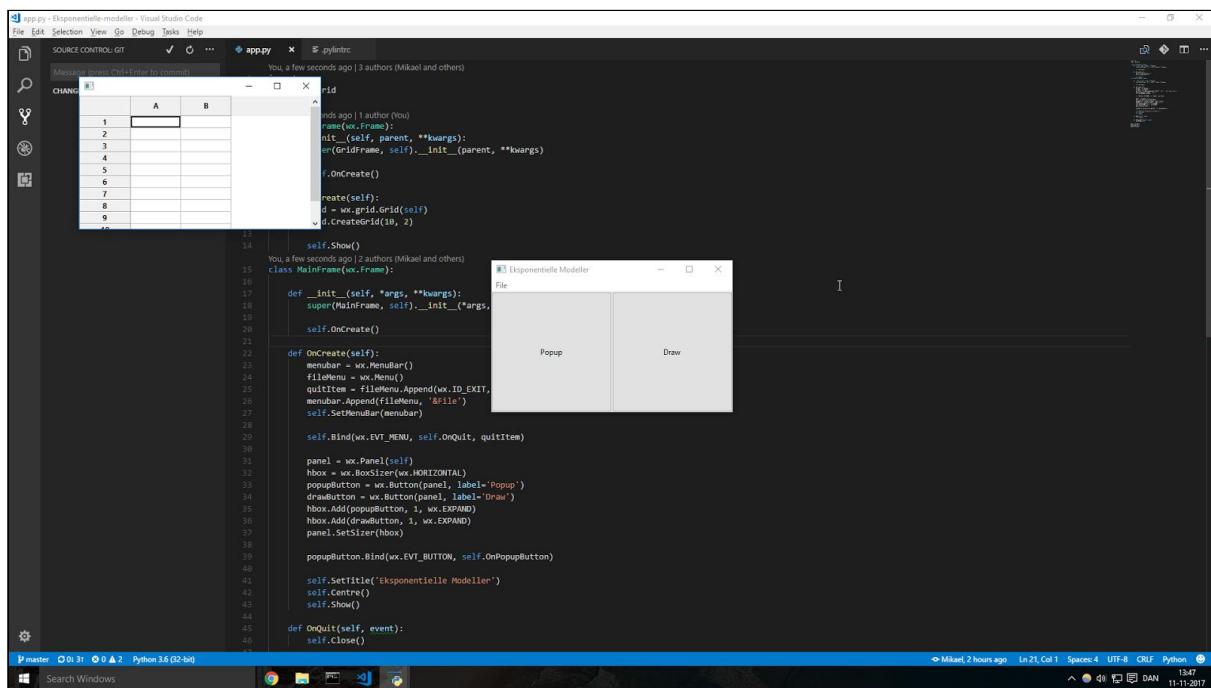


Hherefter indsatte vi to knapper, der hver fylder halvdelen. Dette blev gjort med en horizontal box sizer, der placerer knapperne horizontalt. Vi satte dem til "wx.EXPAND" for at de fylder så meget som muligt.

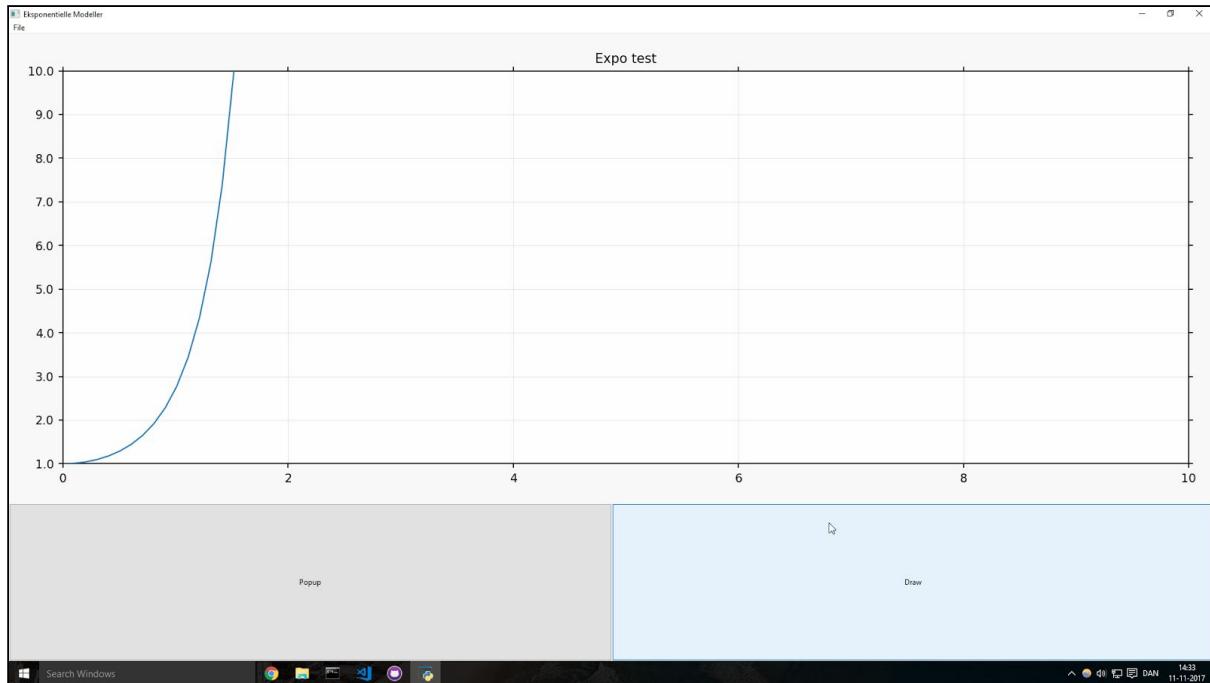
SO-projekt - Eksponentielle modeller



Herefter lavede vi en anden frame, der huser en 2×10 tabel. Denne frame er sat til at vise, når der blev trykket på "Popup"-knappen.



Den næste del gik ud på at tegne en eksponentiel graf. Her brugte vi PlotPanel fra WXMplot, de tillader os at tegne grafer. Vi satte denne til at tegne den simple funktion x^2 , når der blev trykket på "Draw"-knappen.



Nu hvor vi havde hele brugerfladen på plads, var det på tide at lege med Pydux. Her satte vi en store op, som bliver passet ned til vores MainFrame. Her har vi bare lavet en simpel reducer, der skifter state mellem "True" og "False". Der er to funktioner der er subscriptet til storen, en print lambda, og en funktion til at grafen. Eventhandleren til "Draw"-knappen er sat til at dispatche en tom action.

```

app.py - Eksponentielle-modeller - Visual Studio Code
File C:\Windows\system32\cmd.exe - python app.py
D:\053X,Y= 6.20, 4.967X,Y= 6.27, 4.888X,Y= 6.39, 4.823X,Y= 6.38, 4.765X,Y= 6.42, 4.707X,Y= 6.46, 4.635X,Y= 6.50, 4.578X,Y= 6.54, 4.521X,Y= 6.58, 4.464X,Y= 6.62, 4.407X,Y= 6.66, 4.350X,Y= 6.70, 4.293X,Y= 6.74, 4.236X,Y= 6.78, 4.179X,Y= 6.82, 4.120X,Y= 6.86, 4.062X,Y= 6.90, 4.004X,Y= 6.94, 3.946X,Y= 6.98, 3.888X,Y= 6.10, 3.830X,Y= 6.14, 3.772X,Y= 6.18, 3.714X,Y= 6.22, 3.656X,Y= 6.26, 3.598X,Y= 6.30, 3.540X,Y= 6.34, 3.482X,Y= 6.38, 3.424X,Y= 6.42, 3.366X,Y= 6.46, 3.308X,Y= 6.50, 3.250X,Y= 6.54, 3.192X,Y= 6.58, 3.134X,Y= 6.62, 3.076X,Y= 6.66, 3.018X,Y= 6.70, 2.960X,Y= 6.74, 2.902X,Y= 6.78, 2.844X,Y= 6.82, 2.786X,Y= 6.86, 2.728X,Y= 6.90, 2.670X,Y= 6.94, 2.612X,Y= 6.98, 2.554X,Y= 6.10, 2.496X,Y= 6.14, 2.438X,Y= 6.18, 2.380X,Y= 6.22, 2.322X,Y= 6.26, 2.264X,Y= 6.30, 2.206X,Y= 6.34, 2.148X,Y= 6.38, 2.090X,Y= 6.42, 2.032X,Y= 6.46, 1.974X,Y= 6.50, 1.916X,Y= 6.54, 1.858X,Y= 6.58, 1.800X,Y= 6.62, 1.742X,Y= 6.66, 1.684X,Y= 6.70, 1.626X,Y= 6.74, 1.568X,Y= 6.78, 1.510X,Y= 6.82, 1.452X,Y= 6.86, 1.394X,Y= 6.90, 1.336X,Y= 6.94, 1.278X,Y= 6.98, 1.220X,Y= 6.10, 1.162X,Y= 6.14, 1.104X,Y= 6.18, 1.046X,Y= 6.22, 9.868X,Y= 6.26, 9.390X,Y= 6.30, 8.912X,Y= 6.34, 8.434X,Y= 6.38, 7.956X,Y= 6.42, 7.478X,Y= 6.46, 6.990X,Y= 6.50, 6.512X,Y= 6.54, 6.034X,Y= 6.58, 5.556X,Y= 6.62, 5.078X,Y= 6.66, 4.600X,Y= 6.70, 4.122X,Y= 6.74, 3.644X,Y= 6.78, 3.166X,Y= 6.82, 2.688X,Y= 6.86, 2.210X,Y= 6.90, 1.732X,Y= 6.94, 1.254X,Y= 6.98, 6.690X,Y= 6.17, 9.958
C:\Users\multimikaeligt\projects\Eksponentielle-modeller>python app.py
x,Y= 0.894, 0.869X,Y= 0.894, 0.835X,Y= 0.894, 0.225X,Y= 0.880, 0.010X,Y= 0.573, 0.153X,Y= 0.565, 0.320X,Y= 0.554, 0.559X
y,Y= 0.516, 0.988X,Y= 0.565, 0.917X,Y= 0.629, 0.272X,Y= 0.656, 0.010X,Y= False

4.0
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76

    os.getcwd() = D:\Eksponentielle-modeller
    app = wx.App()
    hbox = wx.BoxSizer(wx.VERTICAL)
    hbox.Add(drawPanel, 1, wx.EXPAND)
    hbox.Add(buttonPanel, 1, wx.EXPAND)
    buttonPanel.SetSizer(hbox)
    self.SetTitle('Eksponentielle Modelle')
    self.Centre()
    self.Show()

def OnQuit(self, event):
    self.Close()

def OnPopupButton(self, event):
    GridFrame(self)

def OnDrawButton(self, event):
    self.store.dispatch({'type': ''})

def OnStoreChange(self):
    x = np.linspace(0.0, 10.0, 100.0)
    self.plotPanel.plot(x, np.exp(x**2.0), title='Expo test', xmax=10.0, ymax=10.0)

store = pydux.create_store(lambda state, action: True if state is None else not state)
store.subscribe(lambda: print(store.get_state()))
app = wx.App()
frame = MainFrame(None, store)
app.MainLoop()

```

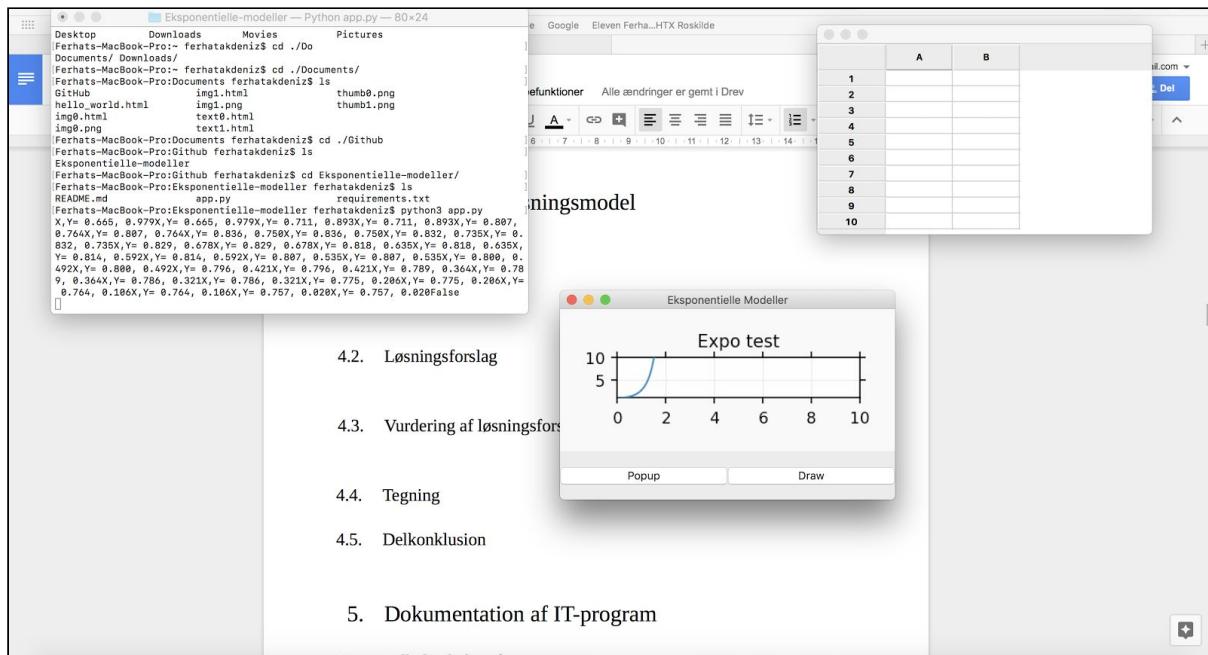
The figure shows a screenshot of a Windows desktop. On the left is a terminal window showing command-line output for running the application. On the right is a code editor showing the Python code for the application, including the main loop and event handlers. The status bar at the bottom shows the date and time as 11-11-2017 16:14.

Som det ses ovenfor, virker programmet stadig helt fint. Vi kan også se at der bliver printet "False" ud til sidst. Dette er den lambda funktion vi satte til at printe state, hvilket bekræfter at vores reducer virker.

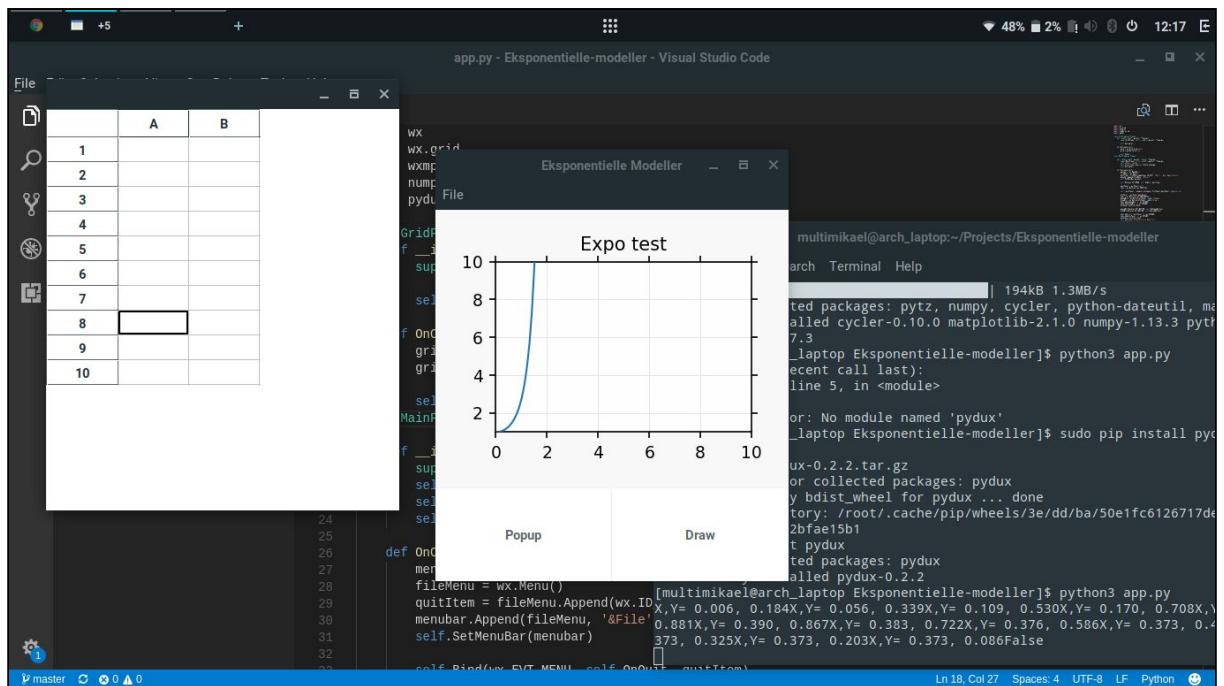
5.4.5. Test

Her har vi testet vores program design på tre diverse operativsystemer macOS, Windows og Linux (GTK 3).

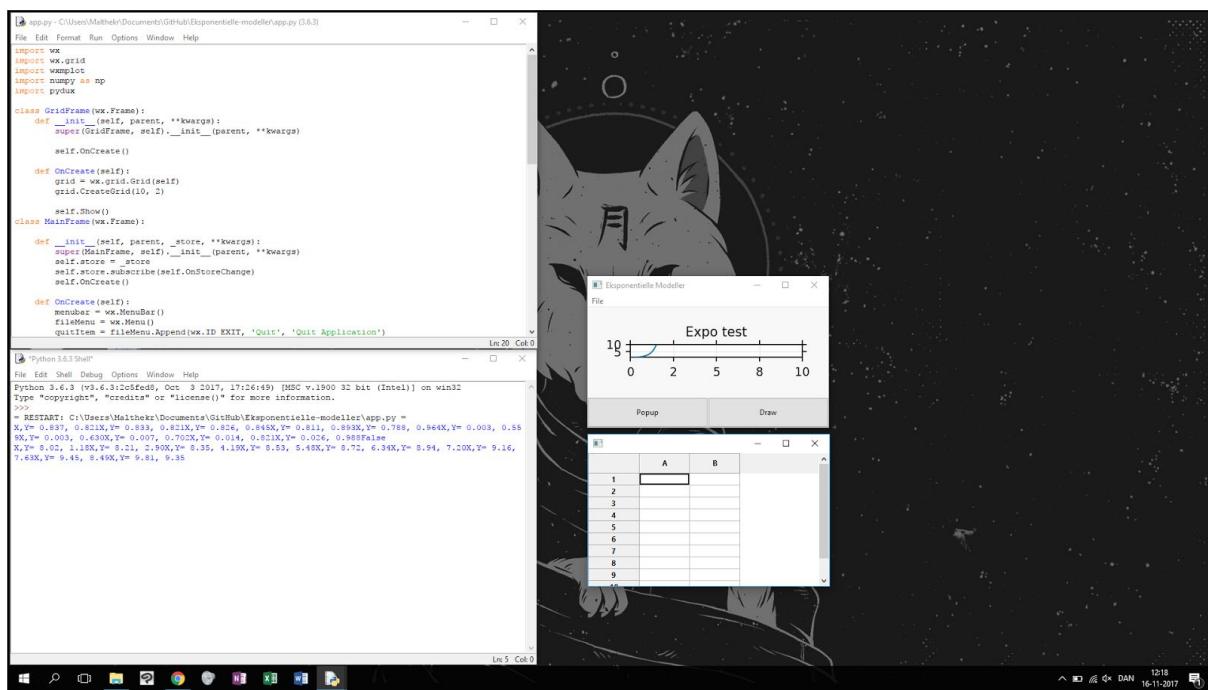
Nedenfor ses programmet på **macOS Sierra**:



Her ses programmet på **Linux (GTK 3)**:



Her ses programmet på **Windows 10**:



5.5. 2. Iteration af systemudvikling

5.5.1. Planlægning

Det er her vi har tænkt os at implementere vores løsningsforslag. Vi har tænkt os implementere vores matematiske løsningsforslag i koden og det meste af designet.

5.5.2. Kontrol af krav og testprocedurer

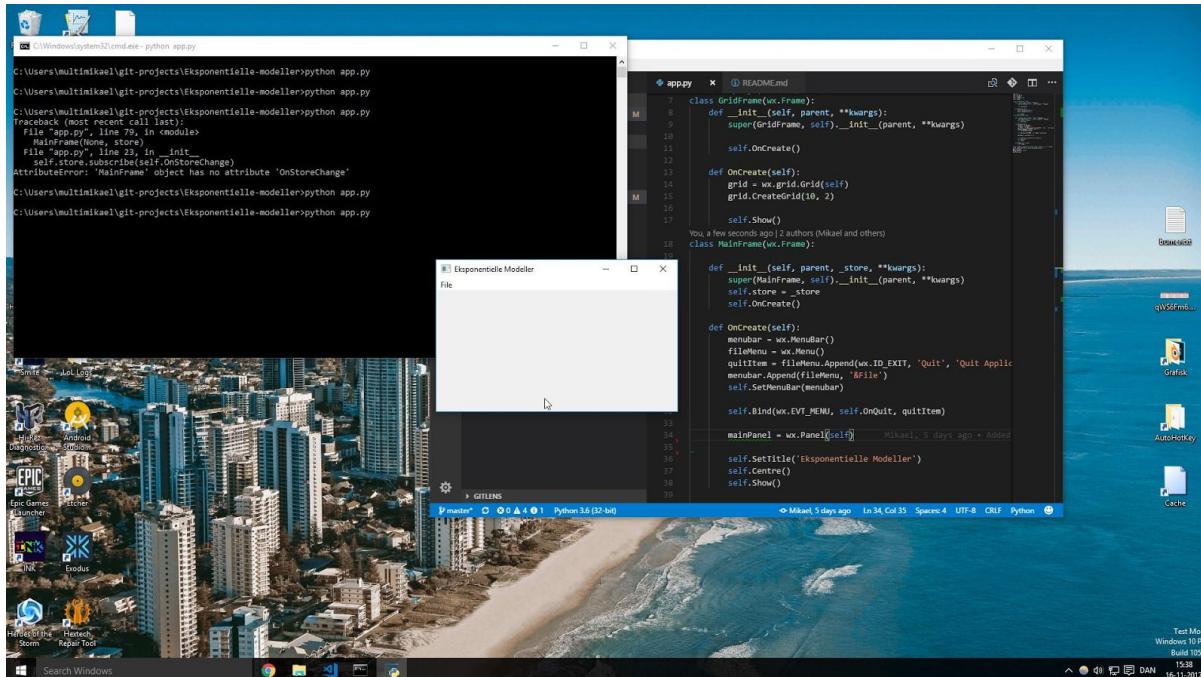
- Kan programmet tage data fra tabellen og andre inputs?
- Vil programmet opdatere efter ændring i input?
- Kan programmet finde evt. eksponentielle kurver?
- Kan programmet finde funktionsforskriften for den eksponentielle vækst?

5.5.3. Design

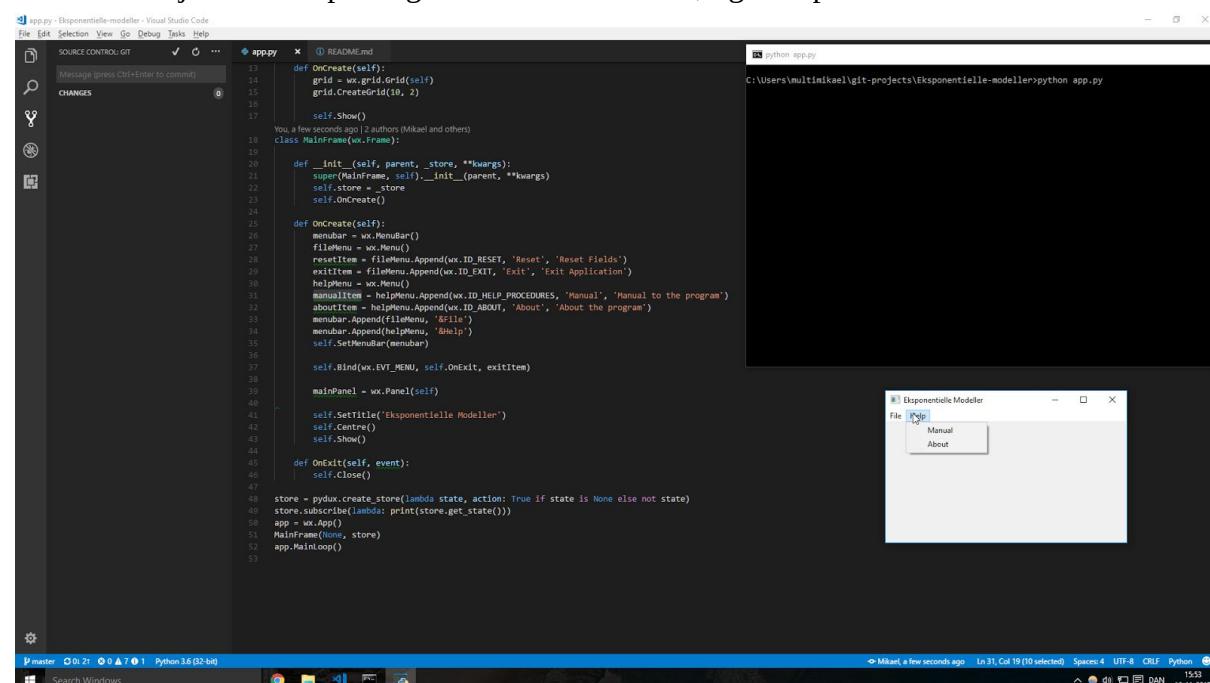
Som planlagt skal designet indeholde to kontekstmenuer ud over den i 1. Iteration. Det vil altså sige at designet kommer til at indeholde både sprog og help kontekstmenuer. Dette er med til at give vores program mere brugervenlighed og mere fuldstændighed. Derudover skal grafen også kunne ændre sit udseende, i den forstand at grafen ændrer sig efter brugerinput i tabellen, checkbox og spin control. Dertil skal det være muligt at se statistikker for grafen.

5.5.4. Implementering

Vi startede med at slette næsten alt i MainFrame. Det vil sige alt bortset fra vores menu bar. Dette gav os mulighed for nemmere at implementere vores design.



Herefter tilføjede vi den planlagte "Reset" menu item, og "Help" menuen.



Vi tilføjede derefter vores input felter. En knap til visning af data, en spin control til indtastning af afvigelse og en checkbox til at filtrere første og sidste data set.

SO-projekt - Eksponentielle modeller

The screenshot shows the Visual Studio Code interface with the app.py file open. The code implements a wxPython application with a menu bar, a main panel containing input fields, and a graph panel. A terminal window to the right shows the application running and printing a message. The taskbar at the bottom indicates the application is running in Python 3.6 (32-bit).

```

app.py - Eksponentielle-modeller - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
SOURCE CONTROL: GIT  CHANGES  app.py  README.md
Message (press Ctrl+Enter to commit)
30     helpMenu = wx.Menu()
31     mainItem = helpMenu.Append(wx.ID_HELP_PROCEDURES, 'Manual', 'Manual to the program')
32     aboutItem = helpMenu.Append(wx.ID_ABOUT, 'About', 'About the program')
33     menuBar = menuBar.Append(helpMenu, '&Help')
34     self.SetMenuBar(menuBar)
35
36     self.Bind(wx.EVT_MENU, self.OnExit, exitItem)
37
38     mainPanel = wx.Panel(self)
39     hboxInputGraph = wx.BoxSizer(wx.HORIZONTAL)
40     graphVBox = wx.BoxSizer(wx.VERTICAL)
41     inputVBox = wx.BoxSizer(wx.VERTICAL)
42
43     dataBtn = wx.Button(mainPanel, label='View Data')
44     deviationBox = wx.BoxSizer(wx.HORIZONTAL)
45     deviationText = wx.StaticText(mainPanel, label='Maximum Deviation: ')
46     deviationSpinCtrl = wx.SpinCtrl(mainPanel)
47     deviationSpinCtrl.SetRange(0, 100)
48     deviationMBox = wx.BoxSizer(wx.HORIZONTAL)
49     deviationMBox.Add(deviationText, 0, wx.ALIGN_CENTER)
50     deviationMBox.Add(deviationSpinCtrl, 1, wx.EXPAND)
51
52     filterCheckBox = wx.CheckBox(mainPanel, label='Filter out first and last')
53
54     inputVBox.Add(dataBtn, 0, wx.EXPAND|wx.ALL)
55     inputVBox.Add(deviationMBox, 0, wx.EXPAND|wx.ALL)
56     inputVBox.Add(filterCheckBox, 0, wx.ALIGN_CENTER)
57
58     hboxInputGraph.Add(inputVBox, 3, wx.EXPAND|wx.ALL)
59     hboxInputGraph.Add(graphVBox, 1, wx.EXPAND|wx.ALL)
60     mainPanel.SetSizer(hboxInputGraph)
61
62     self.SetTitle('Eksponentielle Modeller')
63     self.Centre()
64     self.Show()
65
66 def OnExit(self, event):
67     self.Close()
68
69 store = pydux.create_store(lambda state, action: True if state is None else not state)
70 store.subscribe(lambda: print(store.get_state()))
71 app = wx.App()
72 MainFrame(None, store)
73 app.MainLoop()
74
75
    
```

Herefter tilføjede vi noget tekst til vores grafer og et panel til et “scrollle” på, til brug senere hen. Graferne fylder $\frac{1}{4}$ horisontalt, mens inputs fylder $\frac{3}{4}$ horisontalt.

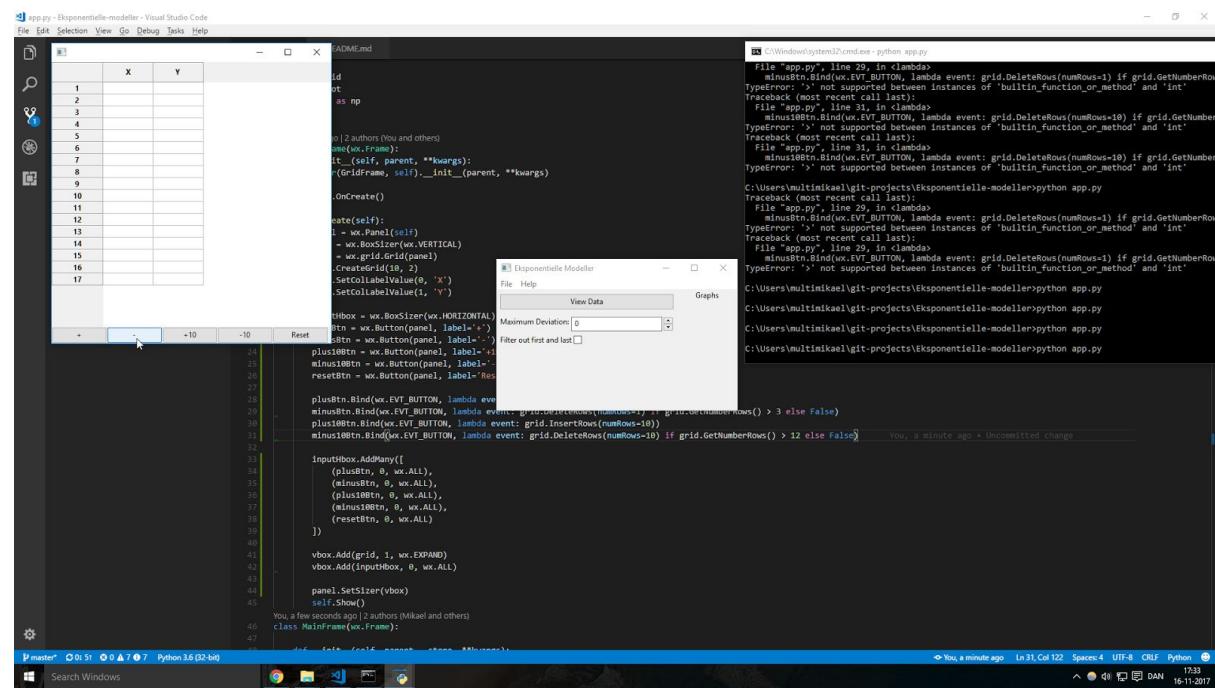
The screenshot shows the Visual Studio Code interface with the app.py file open. The code has been modified to include a 'Graphs' tab in the main panel and a scrollable graph area. A terminal window to the right shows the application running and printing a message. The taskbar at the bottom indicates the application is running in Python 3.6 (32-bit).

```

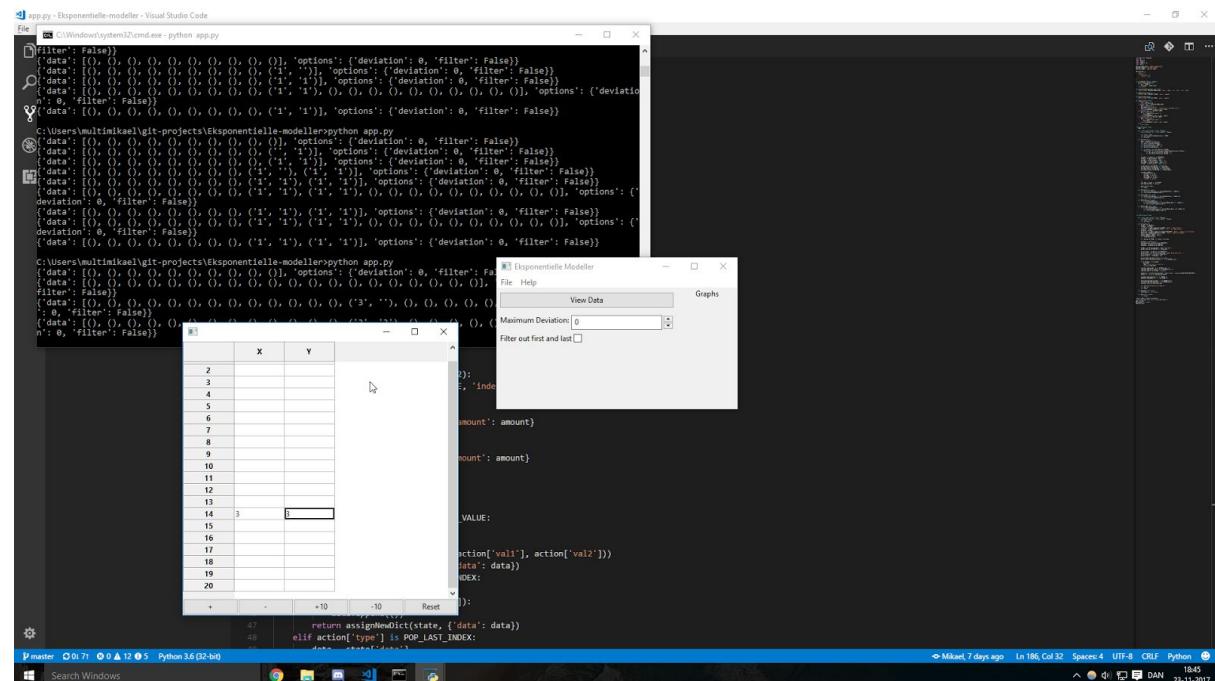
app.py - Eksponentielle-modeller - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
SOURCE CONTROL: GIT  CHANGES  app.py  README.md
Message (press Ctrl+Enter to commit)
30     hboxInputGraph = wx.BoxSizer(wx.HORIZONTAL)
31     graphVBox = wx.BoxSizer(wx.VERTICAL)
32     inputVBox = wx.BoxSizer(wx.VERTICAL)
33
34     dataBtn = wx.Button(mainPanel, label='View Data')
35     deviationBox = wx.BoxSizer(wx.HORIZONTAL)
36     deviationText = wx.StaticText(mainPanel, label='Maximum Deviation: ')
37     deviationSpinCtrl = wx.SpinCtrl(mainPanel)
38     deviationSpinCtrl.SetRange(0, 100)
39
40     deviationMBox = wx.BoxSizer(wx.HORIZONTAL)
41     deviationMBox.Add(deviationText, 0, wx.ALIGN_CENTER)
42     deviationMBox.Add(deviationSpinCtrl, 1, wx.EXPAND)
43
44     filterCheckBox = wx.CheckBox(mainPanel, label='Filter out first and last', style=wx.ALIGN_CENTER)
45
46     inputVBox.Add(dataBtn, 0, wx.EXPAND|wx.ALL, 5)
47     inputVBox.Add(filterCheckBox, 0, wx.EXPAND|wx.ALL, 6)
48     inputVBox.Add(graphsText, 0, wx.EXPAND|wx.ALL, 6)
49
50     graphsText = wx.StaticText(mainPanel, label='Graphs')
51     graphScroll = wx.ScrolledWindow(mainPanel)
52     graphVBox.Add(graphsText, 0, wx.EXPAND|wx.ALL, 5)
53     graphVBox.Add(graphScroll, 1, wx.EXPAND|wx.ALL, 6)
54
55     graphVBox.Add(graphsText, 0, wx.EXPAND|wx.ALL, 5)
56     graphVBox.Add(graphScroll, 1, wx.EXPAND|wx.ALL, 6)
57
58     hboxInputGraph.Add(inputVBox, 3, wx.EXPAND|wx.ALL)
59     hboxInputGraph.Add(graphVBox, 1, wx.EXPAND|wx.ALL)
60     mainPanel.SetSizer(hboxInputGraph)
61
62     self.SetTitle('Eksponentielle Modeller')
63     self.Centre()
64     self.Show()
65
66 def OnExit(self, event):
67     self.Close()
68
69 store = pydux.create_store(lambda state, action: True if state is None else not state)
70 store.subscribe(lambda: print(store.get_state()))
71 app = wx.App()
72 MainFrame(None, store)
73 app.MainLoop()
74
75
    
```

Så tilføjede vi diverse widgets og rettelser til vores tabel. Vi satte en grænse på minimum 3 data set. Knappen “-” kan kun operere hvis der er mindst 4 rækker, mens knappen “-10” kun kan operere hvis der mindst 13 rækker. Reset knappen har ingen funktion endnu. Desuden er tabellens kolonners navne ændret til “X” og “Y” fra “A” og “B”.

SO-projekt - Eksponentielle modeller



Herefter arbejdede vi på pydux implementeringen. Vi lavede en store med en state bestående af vores data set, afgivelse og filtrering. Vi tilføjede tre actions, en til at erstatte en værdi i en liste på et given index (til at modificere data set), en til at tilføje tomme data set, og en til at fjerne data set. Vi bindede modificeringen til hver gang en celle i vores tabel blev modificeret. Vi bindede tilføjelsen af data set til "+" og "+10". Vi bindede fjernelsen af data set til "-" og "-10".



Så tilføjede vi vores spinner control til afgivelse og checkbox til filtrering til pydux. Her fandt vi ud af at der var brug for at håndtere dictionaries inden i dictionaries. Dette løste vi ved at kalde den samme funktion inden i sig selv, hvis det pågældende objekt var dictionary med Pythons *isinstance()*.

SO-projekt - Eksponentielle modeller

The screenshot shows the Visual Studio Code interface with the file `app.py` open. The code implements a state machine for managing data and options. It includes logic for adding data points, removing the last data point, setting doFilter, and filtering data based on deviation. A separate window titled "Eksponentielle Modeler" displays a graph of data points and a fitted exponential curve.

```

app.py - Eksponentielle-modeller - Visual Studio Code
File C:\Windows\system32\cmd.exe - python app.py

data = []
options = {'deviation': 0, 'doFilter': False}
for i in range(action['amount']):
    data.append(options)
return assignNewDict(state, {'data': data})
if action['type'] is POP_LAST_INDEX:
    data = state['data']
    for i in range(action['amount']):
        data.pop()
    return assignNewDict(state, {'data': data})
elif action['type'] is SET_DO_FILTER:
    return assignNewDict(state, {'options': {'doFilter': action['boolean']}})
else:
    return state

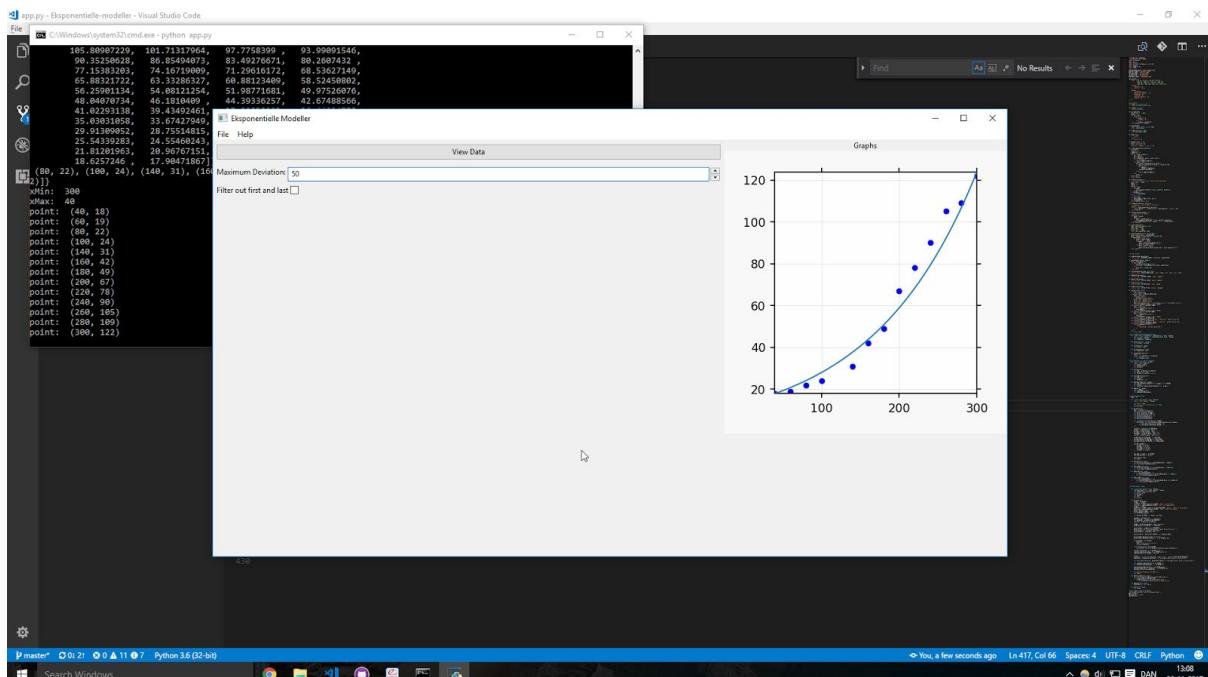
data = state['data']
for i in range(action['amount']):
    data.append(options)
return assignNewDict(state, {'data': data})

def __init__(self, parent, store, **kwargs):
    super(GridFrame, self).__init__(parent, **kwargs)
    self.store = store
    self.store.dispatch(addEmptyIndex(self.ROWS))

Mikael (39 minutes ago) | author (Mikael)
class GridFrame(wx.Frame):
    ROWS = 10
    def _init_(self, parent, store, **kwargs):
        super(GridFrame, self).__init__(parent, **kwargs)
        self.store = store
        self.store.dispatch(addEmptyIndex(self.ROWS))

```

Herefter implementerede vi alt matematikken, der finder en graf for fremskrivningsfaktor inden for vores afvigelse. Vi tilføjede et PlotPanel fra wxmplot til at vise graf og punkter. Da der var nogle små problemer med at udregne for grafer uden punkter, valgte vi at tilføje vores givet punkter ind som en initial state. PlotPanelet fra wxmplot havde nogle problemer hvis vinduet blev for småt, så vi tilføjede bare en minimum size (1280x720) på vinduet. Med de ændring vi lavede duede tabelen ikke længere, da vi ikke havde implementeret synkronisering mellem store og tabel.



Herefter implementerede vi filtreringen. Her fjernes først og sidste datasæt ved udregning. Dette ses på udskriften i terminalen nedenfor, hvor (0, 15) og (320, 125) fjernes.

SO-projekt - Eksponentielle modeller

A screenshot of Visual Studio Code showing a Python script named `app.py`. The code is an implementation of the command-line application from the previous image, handling command-line arguments and displaying results in the console. A data panel shows a scatter plot of points, and a graph panel shows a fitted exponential curve.

```

    161     return results
162
163     def findinspace(points):
164         xMin = points[len(points)-1][0]
165         xMax = points[0][0]
166         return np.linspace(xMin, xMax)
167
168     def manualGraphPanel(parent, manualGraphs):
169
170         graph: dict = {}
171
172         for point in manualGraphs['points']:
173             if 'f' not in graph:
174                 graph['f'] = np.array([1])
175
176             panel = Panel()
177             panel.plot(data, graph['f'][0])
178             panel.scatterplot(np.array([point[0]]), np.array([point[1]]))
179
180             graph['f'] = np.append(graph['f'], np.array([1]))
181
182         panel.show()
183
184         return panel
185
186         if __name__ == '__main__':
187             app = QApplication(sys.argv)
188             myapp = appPy()
189             myapp.show()
190             sys.exit(app.exec_())
    
```

Herefter tilføjede vi grafen hvor x-aksen er rykket til første punkts første koordinat. Det ses dog er der ingen forskel er siden at første koordinaten til at første koordinatsæt er allerede 0.

A screenshot of Visual Studio Code showing the same `app.py` script with modifications to shift the x-axis to the first coordinate. The resulting data points are plotted on a scatter plot, showing a clear linear trend on the log-log scale.

```

    161     return results
162
163     def findinspace(points):
164         xMin = points[len(points)-1][0]
165         xMax = points[0][0]
166         return np.linspace(xMin, xMax)
167
168     def manualGraphPanel(parent, manualGraphs):
169
170         graph: dict = {}
171
172         for point in manualGraphs['points']:
173             if 'f' not in graph:
174                 graph['f'] = np.array([1])
175
176             panel = Panel()
177             panel.plot(data, graph['f'][0])
178             panel.scatterplot(np.array([point[0]]), np.array([point[1]]))
179
180             graph['f'] = np.append(graph['f'], np.array([1]))
181
182         panel.show()
183
184         return panel
185
186         if __name__ == '__main__':
187             app = QApplication(sys.argv)
188             myapp = appPy()
189             myapp.show()
190             sys.exit(app.exec_())
    
```

Herefter tilføjede vi graf for punkterne i dobbeltlogaritmisk koordinatsystem. Her tog vi log af alle de samme punkter, og fandt en lineær linje for log-punkterne.

SO-projekt - Eksponentielle modeller

The screenshot shows a Visual Studio Code window with two tabs: 'app.py - Eksponentielle-modeller - Visual Studio Code' and 'G:\Hjemmesider\htx\Graf\Grafen.py - Python.app'. The code in 'app.py' is a script for a graphical application, while 'Grafen.py' contains the plotting logic. Two scatter plots are displayed in separate windows: one with a linear fit on a large scale (y from 20 to 60) and another with a linear fit on a smaller scale (y from 1.0 to 2.0).

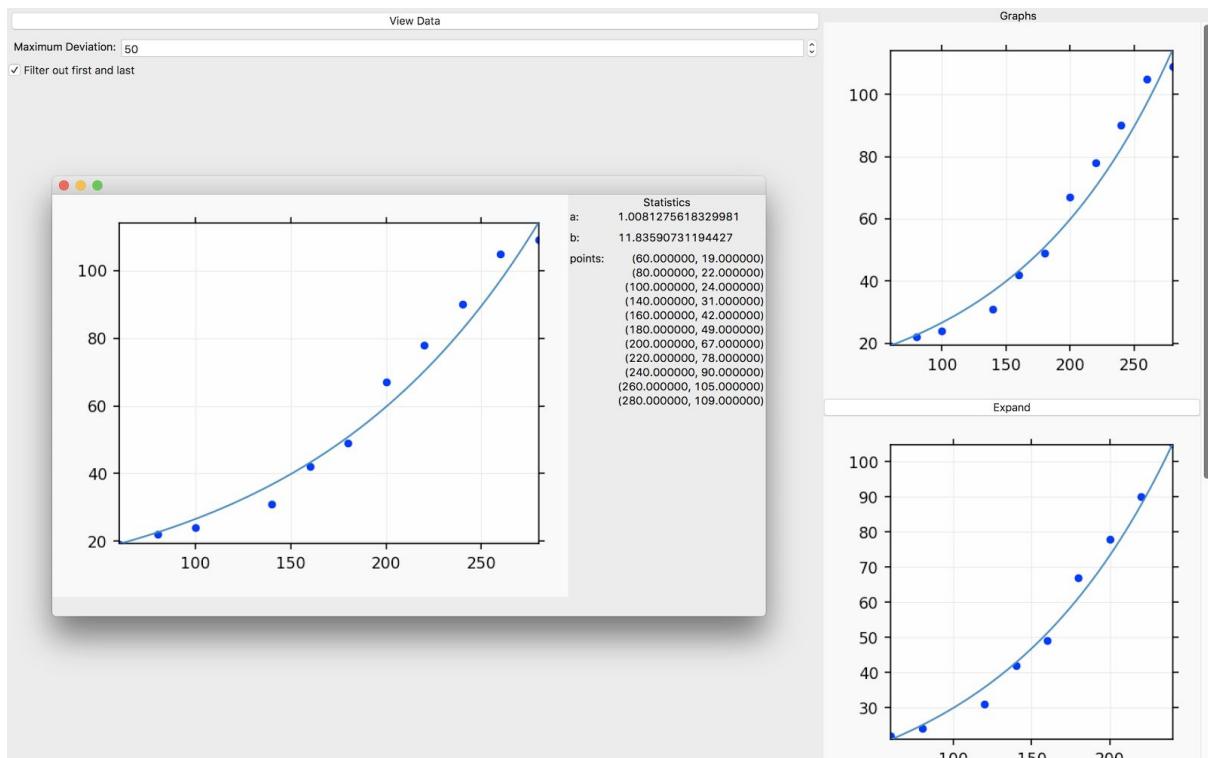
Herefter tilføjede vi et vindue med større graf og statistikker. Vi løb ind i nogle problemer med at få vinduet vist, da PlotPanel fra wxmplot ikke registrerer kliks. Dette løste vi ved at lave en knap med teksten "Expand" under hver graf for at åbne dette vindue.

The screenshot shows the expanded plot window from the previous image, with a larger graph area and additional statistical information like 'Statistics' and 'points' listed on the right. The code in 'Grafen.py' has been modified to include this functionality. The bottom status bar indicates the code is being run on a Windows 10 machine.

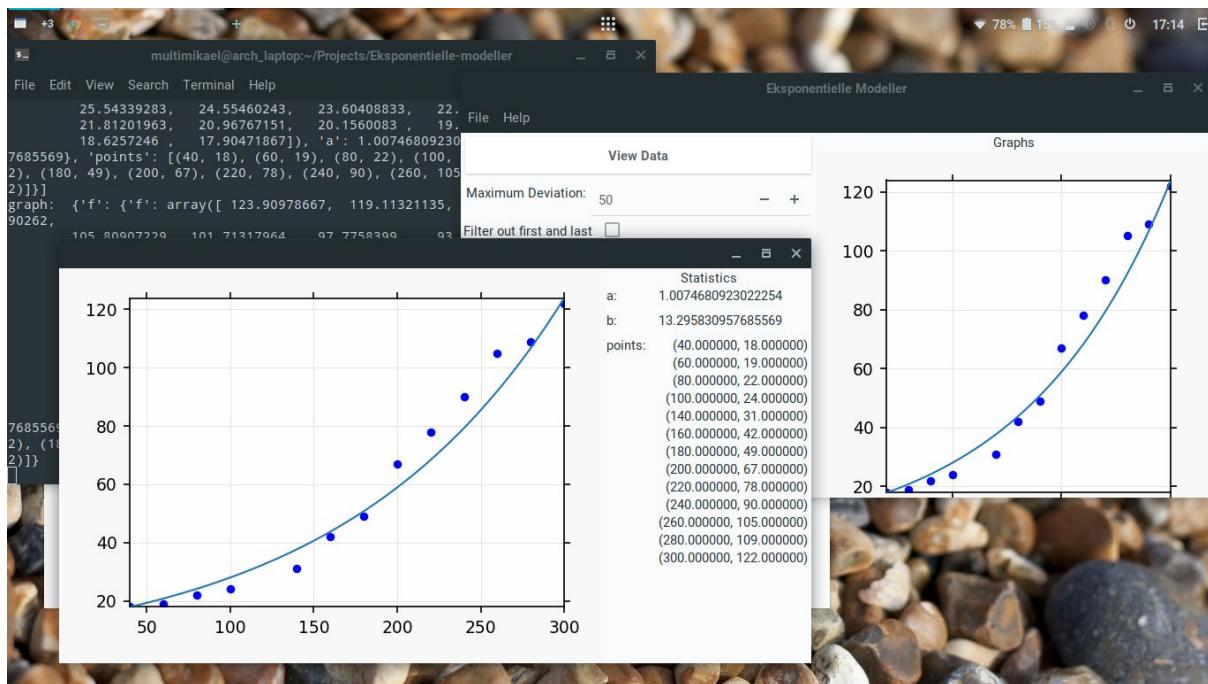
5.5.5. Test

Vi prøvede at åbne programmet på Linux, Windows og Mac som tidligere, med ens resultater på alle platforme. Alt forventet fungerede som det skulle.

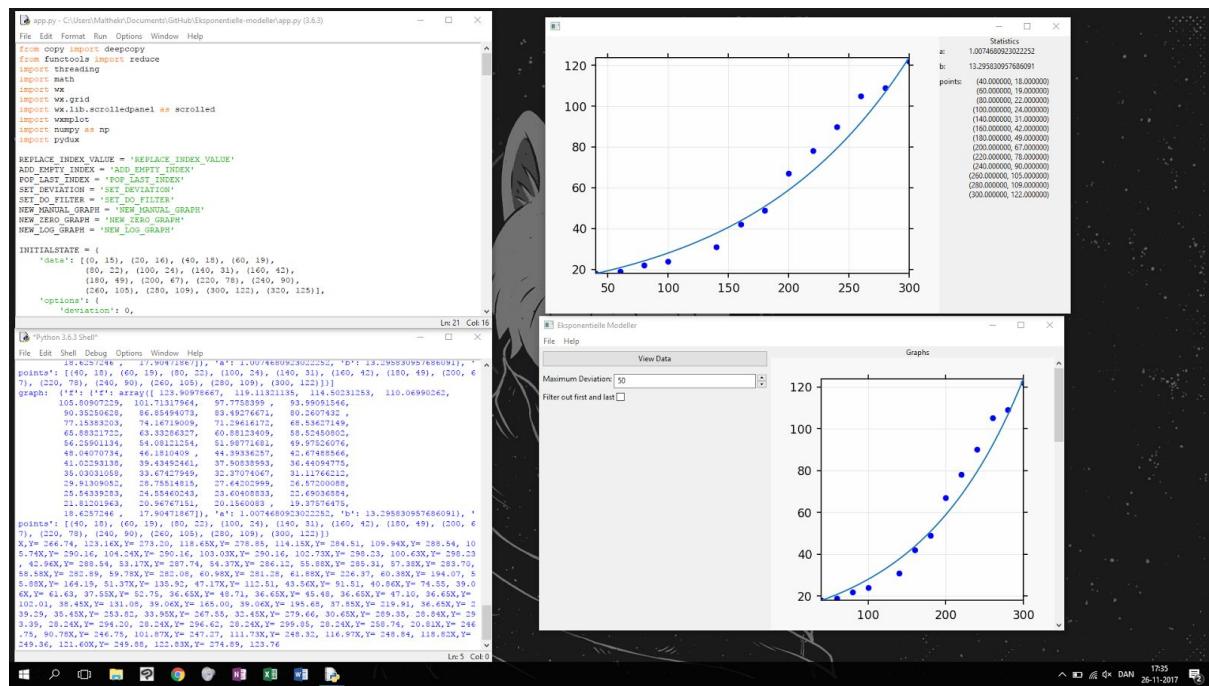
Nedenfor ses programmet på **macOS Sierra**:



Her ses programmet på **Linux (GTK 3)**:



Her ses programmet på **Windows 10**:



5.6. Resultatopgørelse

Resultat blev ikke helt som planlagt. Vi fik implementeret de fleste inputs, spin control til afvigelse og checkbox til filtrering. Data tabellen endte med ikke at virke, da vi ikke fik sat den til at hente data fra pydux storen. Dog lavede vi en hurtig løsning med at hardcode vores given data. Vi fik sat tre forskellige grafer ind, en med punkterne og den udregnet graf, en med y-aksen rykket hen til første punkt, og en i et dobbeltlogaritmisk koordinatsystem. For alle grafer kan man se a , b og punkternes koordinater.

Programmet kan alt den matematik den skal. Brugergrænsefladen er ikke hel optimal og der mangler et par tidligere planlagte features. Reset, Help og About er ikke implementeret i menubaren. Localization med dansk og islandsk er ikke implementeret, og der ikke mulighed for at skifte fra engelsk. Den automatiske gem-funktionen er ikke implementeret, heller er den manuelle. Desuden er brugergrænsefladen ikke non-blocking, da alt foregår på samme tråd. Der er også mulighed for at ændre små detaljer på graferne (som farve) ved at højreklikke på dem. Disse features kommer indbygget fra wxmplot. Programmet kan kun udregne til vores givet data.

6. Vurdering

6.1. Opfyldning af krav

Vi kan altså vurdere at vores program opfylder alle de konkrete krav, heriblandt brugergrænsefladen, mulighed for at ændre på noget visuelt og selvfølgelig dokumentation og

test af program. Vi har som gennemgået testet vores program på 3 forskellige operativsystemer, Windows, MacOS og Linux. Brugeren af vores program har også adgang til at give sit eget input af interval, afvigelsen og selve vinduets dimensioner. Der er mulighed for indkredsning af den eksponentielle fase. Der vækstkurver i flere forskellige koordinatsystemer. Der er ikke bestemt flere forskellige funktionsudtryk, vi har kun regnet med en eksponentiel fase. Lille analyse af funktionsudtryk, dog ingen fordoblingskonstant i programmet (er til stede i den matematisk løsningsmodel). Vi har analyseret målgruppen og designet af brugergrænsefladen til vores IT-produkt. Vi har givet brugeren mulighed for at ændre på noget af det visuelle, heriblandt detaljer på grafer, farver, interval, afvigelse. Vi har implementeret to prototyper heriblandt en spike solution, samt dokumenteret og testet IT-produktet. Der har ikke været mulighed for at publicere IT-produktet på StudieWeb, samt beskrivelse af forløbet. Arbejdet er planlagt, gennemført dokumenteret og præsenteret ved hjælp af dele fra den iterative systemudviklingsmetode fra MediaLab, HTX → Roskilde.

6.2. Delkonklusion

Vi har herved vurderet vores arbejde af matematiske analyse, beregningerne, dokumentationen af forløbet og udarbejdelse af IT-produktet. Vi kan derved konkludere at arbejdet ikke har udfoldet sig helt som vi havde regnet med. Dertil kan vi konkludere at de fleste af de konkrete krav til opgaven udført. De mest essentielle krav for opgaven er løst.

7. Konklusion

Vi har udført matematisk analyse, beregninger og dokumenteret forløbet. Dertil har vi udarbejdet et IT-produkt. IT-produktet er blevet vel dokumenteret ved hjælp af den iterative systemudviklingsmetode fra MediaLab.

Vi har opstillet matematiske løsningsmodeller til at bekræfte vores hypotese om mulig eksponentiel udvikling hos gærceller. Vi har heriblandt anvendt regneregler og formler for eksponentielle funktioner. Vi har benyttet relevant IT-værktøj til at udregne korrelationskoefficient til analysen.

Vi har udviklet et program der kan benyttes til analysen af gærcellers vækst. Programmet kan finde mulige eksponentielle vækst. Udregningen kan dertil tilpasses ved hjælp input fra brugeren. Programmet kan både vise punkter og grafer, der kan vise i flere koordinatsystemer med flere forskellige funktionsudtryk, heriblandt eksponentielle grafer i kartesisk koordinatsystem og lineær grafer i dobbeltlogaritmisk koordinatsystem.

Vi har organiseret og indgået i gruppearbejde, hvoraf vi har gjort brug af Belbins grupperoller og tidsplanlægning. Belbins grupperoller har ikke ændret på hvordan vi arbejder, men har givet os mulighed for at være opmærksom på vores gruppens mulige svagheder. Tidsplanen

har kun hjulpet os med at holde styr på projektets arbejde i de given moduler, men altså ikke hjulpet os yderligere.

8. Litteraturliste

Websites		
Ophav	URL-Adresse	Dato
Python - PSF	https://www.python.org/	Set: 27. oktober, 2017
wxPython	https://www.wxpython.org/	Set: 27. oktober, 2017
WXMPlot newville - Github	https://newville.github.io/wx_mplot/	Set: 27. oktober, 2017
pydux usrlocalben - Github	https://github.com/usrlocalben/pydux	Set: 27. oktober, 2017

9. Bilag

9.1. Tidsplan

Uge 44:

- Projektoplægget udleveres onsdag i uge 43.
- Bearbejdning af data indledes.
- Introduktion til IT-værktøjer.

Uge 45:

- Opstillingen af matematiske modeller.
- Analyse, design og implementering af brugergrænsefladen til IT-produktet.
- Fokus på den naturvidenskabelige metode.

Uge 46-47:

- Delaflevering af de matematiske beregninger og foreløbige modeller mandag i uge 46 senest kl. 8.10.
- Vurdering af de matematiske modeller og færdiggørelse af IT-produktet (*herunder test og fejlretning*).

9.2. Faglige mål

Nedenstående ses både faglige mål for faget matematik samt faget informationsteknologi, hvilket fører frem til de overfaglige mål, herunder faglige mål fra studieområdet.

9.2.1. Matematik

- opøve evnen til at analysere områder udenfor matematikken ved hjælp af matematik,
- opøve evnen til at opstille eksponentielle funktionsudtryk ud fra en ikke-matematisk beskrivelse af problemet,
- opøve evnen til at vurdere matematiske modellers udsagnskraft,
- opøve evnen til at veksle mellem forskellige typer af matematisk repræsentation.

9.2.2. Informationsteknologi

- redegøre for samspillet mellem it-komponenter og bruger,
- anvende it som interaktivt medie til dokumentation og kommunikation,
- realisere prototyper på it-systemer herunder kunne installere, konfigurere og tilpasse relevante it-komponenter,
- søge, finde og vælge værktøj.

9.2.3. Overfaglige mål (studieområdet)

- videreudvikle evnen til at anvende fagligt relevante studieteknikker og arbejdsmetoder,
- videreudvikle evnen til at arbejde problemorienteret,
- videreudvikle evnen til at producere viden om praktisk-teoretiske problemstillinger i samspillet mellem fag,
- videreudvikle evnen til skriftlig formidling og arbejde med relevante IT-værktøjer,
- videreudvikle evnen til at organisere og indgå i gruppearbejde (fortsæt arbejdet med Belbinrollerne og løbende evaluering af arbejdet),
- videreudvikle fortroligheden med den naturvidenskabelige metode.