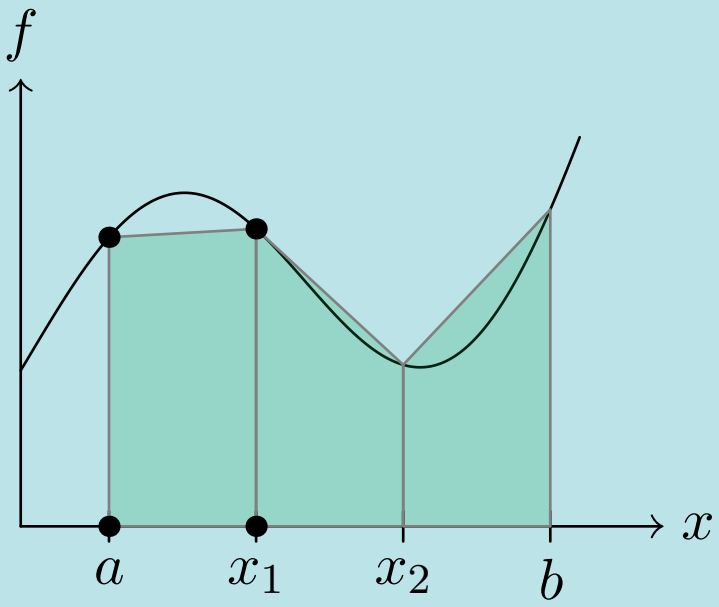
Anvendt matematikk 2
1T, R1 og R2



Sindre Sogge Heggen

Innhold

Kapittel 1

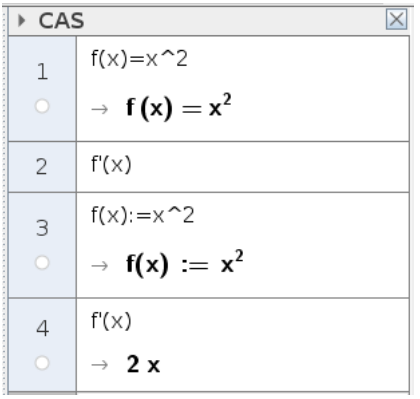
Digitale verktøy

1.1 GeoGebra

1.1.1 CAS

Definere variabler

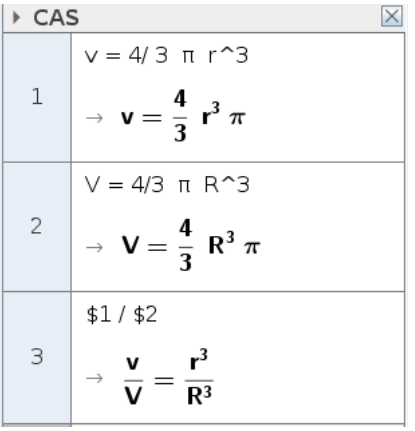
Hvis vi ønsker å definere variabler som vi skal bruke i andre celler, må vi skrive `:=` . I figuren under er forskjellen mellom `=` og `:=` demonstrert med et forsøk på å finne $f'(x)$ til funksjonen $f(x) = x^2$:



Av figuren legger vi også merke til at celle 3 og celle 4 er markert med en hvit runding. Dette indikerer at størrelsen vil vises i *Grafikkfelt* eller *Grafikkfelt 3D* hvis man trykker på markøren (den skal da bli blå).

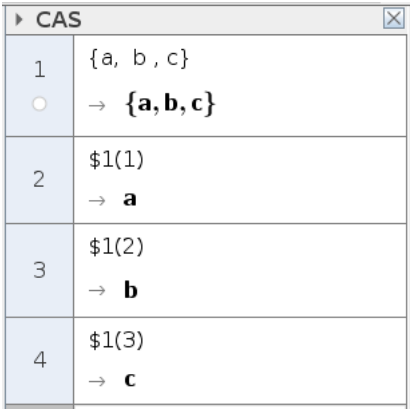
Celle-referanser

Ofte kommer vi ut for situasjoner der vi ønsker å bruke uttrykket vi har funnet i tidligere celler. Som eksempel har vi i celle 1 skrevet inn volumet v av en kule med radius r, mens i celle 2 har vi volumet V av en kule med radius R. Ønsker vi å finne forholdet mellom disse, kan vi bruke cellereferanser som hjelpemiddel. For å referere til celle 1 skriver vi `$1` og for celle 2 skriver vi `$2`. Forholdet $\frac{v}{V}$ kan vi da skriver som `$1/$2`:

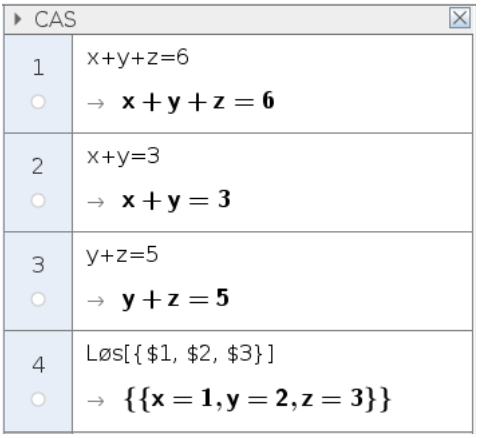


Lister

Når et uttrykk står inni sløyfeparanteser {}, betyr det at det er laget en liste. En liste inneholder flere elementer som vi kan hente ut. Dette gjør vi ved å skrive paranteser bak listen, hvor vi angir nummeret til elementet i listen.

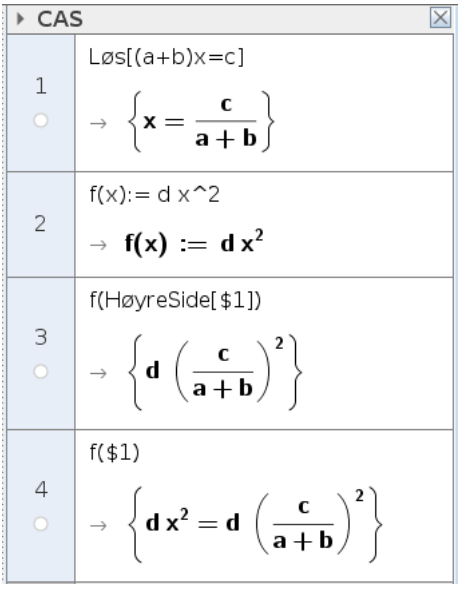


Lister bruker vi også når vi skal løse liginger med flere ukjente:



Høyre- og venstresiden

De fleste uttrykkene vi jobber med i CAS inneholder et $=$ tegn. Disse uttrykkene er en ligning med en venstre- og høyreside. Ofte ønsker vi å bruke uttrykket på bare én av disse sidene, og oftest høyresiden. Som eksempel har vi løst ligningen $(a + b)x = c$ og definert funksjonen $f(x) = dx^2$. Vi ønsker så å sette løsningen av ligningen inn i funksjonen. Dette gjør vi ved hjelp av `HøyreSide`-kommandoen (resulatet uten bruken av denne er vist i celle 4).

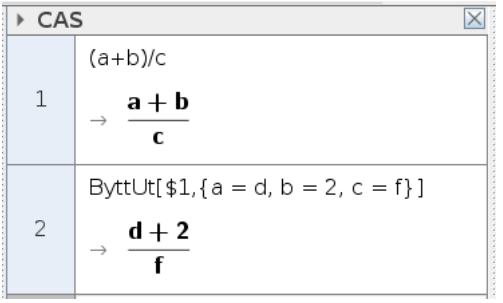


ByttUt

Noen ganger ønsker vi å endre en variabel i et utrykk. For å gjøre dette kan vi anvende `ByttUt( <Uttrykk>, <Liste med forandringer> )`. La oss se på uttrykket

$$\frac{a+b}{c}$$

Vi ønsker nå å sette $a = d$, $b = 2$ og $c = f$. Dette kan vi gjør ved å skrive følgende:



1.1.2 Knapper og kommandoer

Grafikkfelt

Knappene velges fra rullemenyer på verktøylinjen. Nummereringen av menyene er fra venstre.

 Lager et nytt punkt. (Meny nr. 1)

 Lager linje mellom to punkt. (Meny nr. 2)

 Finner topp- og bunnpunkt til en funksjon. (Meny nr. 2)

 Finner nullpunktene til en funksjon. (Meny nr. 2)

 Finner skjæringspunkt mellom to objekt. (Meny nr. 3)

 Lager vektoren mellom to punkt (Meny nr. 3)

 Lager en tekstboks. (Meny nr. 10)

 Flytter grafikkfeltet. Endrer verdiavstanden hvis man peker på aksene. (Meny nr. 10)

CAS

 Gjengir uttrykket som er inntastet, ofte i forkortet form.

 Gjengir uttrykket som er inntastet.

 Gir tilnærmet verdi av et uttrykk (som desimaltall).

 Gir eksaktløsningen av en ligning.

 Gir tilnærmet løsning av en ligning som desimaltall.

Hurtigtaster

	Beskrivelse	PC	Mac
$\sqrt{}$	kvadratrot	`alt + r`	`alt + r`
π	pi	`alt + p`	`alt + p`
∞	uendelig	`alt + u`	`alt + ,`
$\otimes$	kryssprodukt	`alt + shift + 8`	`ctrl + shift + 8`
e	eulers tall	`alt + e`	`alt + e`
$^\circ$	gradtegnet ($\frac{\pi}{180}$)	`alt + o`	`alt + o`

Kommandoliste

`abs( <x> )`
Finner lengden til et objekt x.

`Asymptote( <Funksjon> )`
Finner asymptotene til en funksjon.

`Avstand( <Punkt>, <Objekt> )`
Gir avstanden fra et punkt til et objekt.

`ByttUt( <Uttrykk>, <Liste med forandringer> )` (CAS)
Viser et gitt uttrykk etter endring av variabler, gitt i en liste.

`Deriverte( <Funksjon> )`
Gir den deriverte av en funksjon.
Merk: For en definert funksjon $f(x)$, kan man like gjerne skrive `f'(x)`.

`Ekstremalpunkt( <Funksjon>, <Start>, <Slutt> )`
Finner lokale ekstremalpunkt og ekstremalverdier for en funksjon f på et gitt intervall.

`Ekstremalpunkt( Polynom )`
Finner lokale ekstremalpunkt og ekstremalverdier til et polynom.

`Funksjon( <Funksjon>, <Start>, <Slutt> )`
Tegner en funksjon på et gitt intervall.

`Høyde( <Objekt> )`
Gir avstanden fra toppunkt til grunnflate i et objekt. *Merk*: Avstanden har retning, og derfor kan den noen ganger være negativ. Tallverdien er den geometriske høyden.

`HøyreSide( <Likning> )` (CAS)
Gir høyresiden til en likning.

`HøyreSide( <Liste med likninger> )` (CAS)
Gir en liste med høyresidene i en liste med ligninger.

`Integral( <Funksjon> )`
Gir uttrykket til det ubestemte integralet av en funksjon. (Merk: Hvis kommandoen skrives i inntastingsfeltet, blir konstantleddet utelatt).

`Integral( <Funksjon>, <Start>, <Slutt> )`
Gir det bestemte integralet av en funksjon på et intervall.

`Integral( <Variabel> )` (CAS)
Gir uttrykket til det ubestemte integralet til en funksjon av gitt variabel. (Brukes dersom man ønsker å integrere funksjoner avhengig av en annen variabel enn x).

`Kule( <Punkt>, <Radius> )`
Viser en kule i Grafikkfelt 3D med sentrum i et gitt punkt og med en gitt radius.

`Kurve( <Uttrykk>, <Uttrykk>, <Uttrykk>, <Parametervariabel>, <Start>, <Slutt> )`
Viser parameteriseringen av en kurve i Grafikkfelt 3D på et gitt intervall. Uttrykkene er henholdsvis uttrykkene for x, y og z-koordinatene, bestemt av en gitt parametervariabel.
Merk: Med mindre et bestemt intervall av kurven er ønsket, er det bedre å skrive parameteriseringen direkte inn i inntastingsfeltet som `A+t*u`, hvor A er et punkt på linja og u er en retningsvektor.

`Linje( <Punkt>, <Punkt> )`
Gir uttrykket til en linje mellom to punkt. Hvis punktene har tre koordinater besår uttrykket av et punkt på linja og en fri variabel }*lambda* mulitplisert med en retningsvektor.

`Løs( <Likning med x> )` (CAS)
Løser en likning med x som ukjent.

`Løs( <Liste med likninger>, <Liste med variabler> )` (CAS)
Finner alle løsninger av en liste med ligninger med gitte variabel som ukjente.

`Løs( <Likning>, <Variabel> )` (CAS)
Finner alle løsninger av en gitt likning med en gitt variabel som ukjent.

`Maks( <Funksjon>, <Start x-verdi>, <Slutt x-verdi> )`
Finner absolutt maksimum og maskimalpunkt for en funksjon f på et gitt intervall.

`Min( <Funksjon>, <Start x-verdi>, <Slutt x-verdi> )`
Finner absolutt minimum og minimumspunkt for en funksjon f på et gitt intervall.

`Nullpunkt( <Polynom> )`
Finner alle nullpunkter til et polynom.

`NullpunktIntervall( <Funksjon>, <Start>, <Slutt> )`
Finner alle nullpunkter på et gitt intervall til en hvilken som helst funksjon.

`Plan( <Punkt>, <Punkt>, <Punkt> )`
Viser et plan i Grafikkfelt 3D, utspent av to av vektorene mellom tre gitte punkt.

`Prisme( <Punkt>, <Punkt>, ... )`
Framstiller en prisme i Grafikkfelt 3D. `Prisme[A,B,C,D]` lager en prisme med grunnflate ABC og tak DEF, `Prisme[A,B,C,D,E]` har grunnflate $ABCD$ og tak EFG. F, G og eventelt E blir konstruert av GeoGebra slik at hver sideflate er et parallellogram. Under kategorien *Prisme* i algebrafaltet finner man en konstant som oppgir volumet til pyramiden.

`Punkt( <Liste> )`
Lager et punkt med koordinater gitt som liste.
Merk: For å lage punktet (x, y), kan man liksågodt skrive `(x,y)` i inntastingsfeltet. Skriver man `(x,y)` i CAS lager man vektoren $[x, y]$.

`Pyramide( <Punkt>, <Punkt>, ... )`
Framstiller en pyramide i Grafikkfelt 3D. `Pyramide[A,B,C,D]` lager en pyramide med grunnflate A, B, C og toppunkt D, mens `Pyramide[A,B,C,D,E]` har grunnflate A, B, C, D og toppunkt E. Under kategorien *Pyramide* i algebrafaltet finner man en konstant som oppgir volumet til pyramiden.

`RegLin( <Liste> )`
Bruker regresjon med en rett linje for å tilpasse punkt gitt i en liste.

`RegEksp( <Liste> )`
Bruker regresjon med en eksponentialfunksjon for å tilpasse punkt gitt i en liste.

`RegPoly( <Liste>, <Grad> )`
Bruker regresjon med et polynom av gitt grad for å tilpasse punkt gitt i en liste.

`RegPot( <Liste> )`

Bruker regresjon med en potensfunksjon for å tilpasse punkt gitt i en liste.

`RegSin( <Liste> )`
Bruker regresjon med en sinusfunksjon for å tilpasse punkt gitt i en liste.

`Skalarprodukt( <Vektor>, <Vektor> )`
Finner skalarproduktet av to vektorer.
Merk: For to vektorer u og v kan man like gjerne skrive `u*v`.

`Skjæring( <Objekt>, <Objekt> )`
Finner skjæringspunktene mellom to objekter.

`Skjæring( <Funksjon>, <Funksjon>, <Start>, <Slutt> )`
Finner skjæringspunktene mellom to funksjoner på et gitt intervall.

`Sum( <Uttrykk>, <Variabel>, <Start>, <Slutt> )` (CAS)
Finner summen av en rekke med en løpende variabel på et intervall.

`TrigKombiner( <Funksjon> )`
Skriver om et uttrykk på formen $a\sin(kx) + b\cos(kx)$ til et kombinert uttrykk på formen $r\cos(kx - c)$.

`TrigKombiner( <Funksjon>, sin(x) )`
Skriver om en funksjon på formen $a\sin(kx) + b\cos(kx)$ til et kombinert uttrykk på formen $r\sin(kx + c)$.

`Vektor( <Punkt> )`
Lager vektoren fra origo til et gitt punkt.
Merk: I CAS kan man lage vektoren $[x, y]$ ved å skrive `(x,y)`, dette anbefales.

`Vektorprodukt( <Vektor>, <Vektor> )` (CAS)
Finner vektorproduktet av to vektorer.
Merk: Merk: For to vektorer u og v kan man like gjerne skrive `u⊗v`.

`Vendepunkt( <Polynom> )`
Finner vendepunktene til et polynom.

`VenstreSide( <Likning> )` (CAS)
Gir venstresiden til en likning.

`VenstreSide( <Liste med likninger> )` (CAS)

Gir en liste med venstresidene i en liste med ligninger.

`Vinkel( <Vektor>, <Vektor> )`
Gir vinkelen mellom to vektorer. Kan også brukes for vinkel mellom plan/linjer, plan/plan og linje/linje

1.2 Python

Merk: Denne teksten bygger på teksten om Python i AM1.

1.2.1 NumPy

I Python kan man importere det som kalles **bibliotek** for å få tilgang til enda flere typer objekter, funksjoner og liknende. NumPy er et bibliotek som inneholder typen `numpy.ndarray`. Denne typen har mange fellestrekk med en liste, men skiller seg ut ved at den inneholder et bestemt antall elemener. Dette gjør blant annet at prossesser som bruker NumPy-arrays istedenfor lister går raskere, og at NumPy-arrays egner seg bedre til regneoperasjoner.

For å lage NumPy-arrays må vi importere NumPy-biblioteket:

```
1  import numpy as np
2
3  a = np.array([1, 2]) # lager array fra en liste
4  b = np.arange(4) # samme som å skrive np.array(range
       (4))
5  c = np.zeros(3) # array med 3 elementer lik 0
6  d = np.linspace(2,11,4) # array med 4 elementer.
                           d[0] = 2 og d[3] = 11.
                         Naboelement har lik differanse
7
8  print(a)
9  print(b)
10 print(c)
11 print(d)
```

Utdata

```
[1 2]
[0 1 2 3]
[0. 0. 0.]
[ 2. 5. 8. 11.]
```

Merk

Til forskjell fra lister, er elementene skilt bare med mellomrom når de printes.

Klassisek regnearter

Regneoperasjoner mellom NumPy-arrays blir utført elementvis:

```
1 import numpy as np
2
3 a = np.array([10, 20])
4 b = np.array([2, 4])
5
6 print(a+b) # [10+2 20+4]
7 print(a*b) # [10*2 20*4]
```

Utdata

```
[12 24]
[20 80]
```

Vektoroperasjoner

NumPy-arrays fungerer ypperlig til å representere vektorer, og har innebygde metoder for å finne skalarprodukt, kryssprodukt og determinanter:

```
1 import numpy as np
2
3 a = np.array([2, -7])
4 b = np.array([1, 5])
5 c = np.array([2, -7, 1])
6 d = np.array([1, 5, 0])
7
8 print(a.dot(b)) # skalarprodukt av a og b
9 print(np.cross(c, d)) # kryssproduktet av c og d
10
11 ab = np.array([a, b])
12 print(np.linalg.det(ab)) # det(a,b)
```

Utdata

```
-33
[-5 1 17]
17.0
```

Kapittel 2

Numeriske metoder

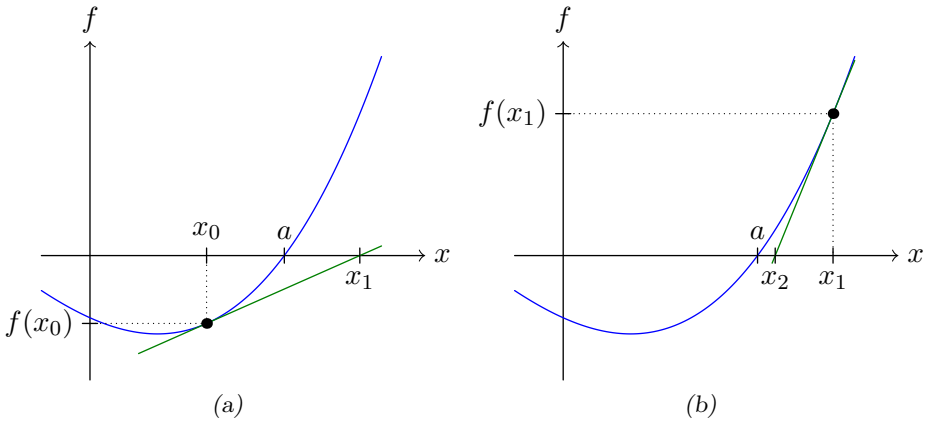
2.1 Newtons metode

Gitt en funskjon $f(x)$ si at vi ønsker å finne et tall a slik at $f(a) = 0$. Ved **Newtons metode** gjør vi denne antakelsen for å en tilnærming a:

> La x_1 være skjæringspunktet mellom x-aksen og tangenten til f i x_0. Vi antar da at $|x_1 - a| < |x_0 - a|$. Sagt med ord antar vi at x_1 gir en bedre tilnærming for a enn det x_0 gjør.

Siden x_1 er skjæringspunktet mellom x-aksen og tangenten til f i x_0, har vi at[1]

$$\begin{aligned} f'(x_0)(x_1 - x_0) + f(x_0) &= 0 \\ f'(x_0)x_1 &= f'(x_0)x_0 - f(x_0) \\ x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \end{aligned}$$



(a) *(b)*

La x_2 være skjæringspunktet mellom x-aksen og tangenten til f i x_1. Ved å gjenta prosedyren vi brukte for å finne x_1, kan vi finne x_2, som vi antar er en enda bedre tilnærming for a enn x_1. Prosedyren kan vi gjenta fram til vi har funnet en x-verdi som gir en tilstrekkelig[2] tilnærming til a.

[1] Se oppgave??

[2] Hva som er en *tilstrekkelig tilnærming* er det opp til oss selv å bestemme.

R 2.1 Newtons metode

Gitt en funskjon $f(x)$ si at vi ønsker å finne et tall a slik at $f(a) = 0$. Gitt x-verdiene x_n og x_{n+1} for $n \in \mathbb{N}$. Ved å bruke formelen

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

antas det at x_{n+1} gir en bedre tilnærming for a enn x_n.

Språkboksen

Newtons metode kalles også **Newton-Rhapsos metode**.

Når er tilmærmingen god nok?

Newtons metode beskriver en iterasjonsprossess som man håper at nærmer seg en verdi. Hvis meotden lykkes, vil x_{n+1} og x_n etterhvert være veldig like, og slik kan en grense for hvor liten $|x_{n+1} - x_n|$ kan være fungere som et godt mål for når iterasjonsprossessen skal stoppe.

2.2 Trapesmetoden

Gitt en funksjone $f(x)$. Integralet $\int_a^b f\,dx$ kan vi tilnærme ved å

1. Dele intervallet $[a, b]$ inn i mindre intervall. Disse kaller vi **delintervall**.
2. Finne en tilnærmet verdi for integralet av f på hvert delintervall.
3. Summere verdiene fra punkt 2.

I *figur 2.1a* har vi 3 like store delintervaller. Hvis vi setter $a = x_0$ og $\Delta x = \frac{b-a}{3}$, betyr dette at

$$x_1 = x_0 + \Delta x \qquad x_2 = x_0 + 2\Delta x \qquad x_3 = x_3 + 3\Delta x = b$$

En tilnæret verdi for $\int_a^{x_1} f\,dx$ får vi ved å finne arealet til trapeset med hjørner (husk at $x_0 = a$)

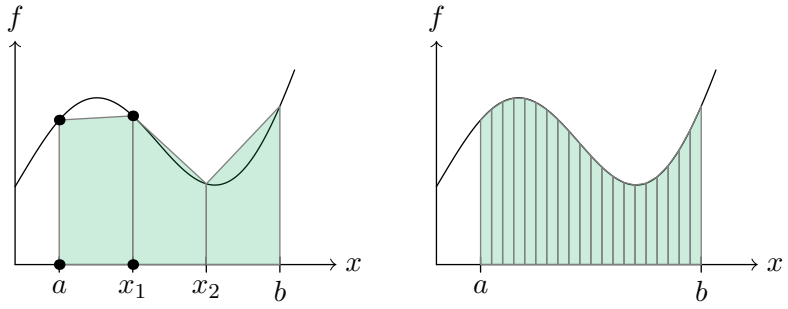
$$(x_0, 0) \qquad (x_1, 0) \qquad (x_1, f(x_1)) \qquad (x_0, f(a))$$

Dette arealet er gitt ved uttrykket

$$\frac{1}{2}(x_1 - x_0)\left[f(x_0) + f(x_1)\right] = \frac{\Delta x}{2}\left[f(x_0) - f(x_1)\right]$$

Ved å tilnærme integralet for hvert delintervall på denne måten, kan vi skrive

$$\int_a^b f\,dx \approx \frac{\Delta x}{2} \sum_{i=0}^{2} \left[f(x_i) + f(x_{i+1})\right]$$



(a) Tilnærming med 3 delintervaller.

(b) Tilnærming med 20 delintervaller

Figur 2.1

R 2.2 Trapesmetoden

Gitt en integrerbar funksjon f. En tilnærmet verdi for $\int_a^b f\,dx$ er da gitt som

$$\int_a^b f\,dx \approx \frac{\Delta x}{2}\sum_{i=0}^{n}[f(x_i)+f(x_{i+1})] \tag{2.1}$$

hvor

$$n \in \hat{\mathbb{N}}$$

$$a = x_0$$

$$b = x_n$$

$$\Delta x = \frac{b-a}{n+1}$$

$$x_{n+1} = x_n + i\Delta x$$

Merk

Slik *regel 2.2* er formulert, vil $[a, b]$ være delt inn i $n + 1$ delintervaller.

Kapittel 3

Blandede oppgaver

Oppgaver for kapittel 3

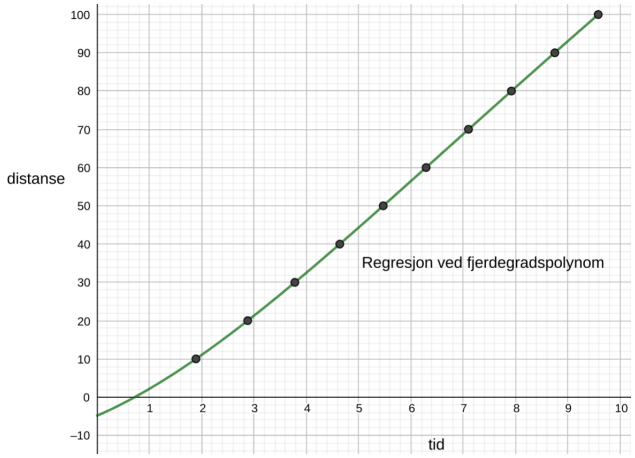
3.1.1

#regresjon #funksjonsdrøfting #omgjøring av enheter

Usain Bolt har verdensrekorden for 100 m sprint. I tabellen under ser du hva tidtakeren viste ved hver 10. meter under dette rekordløpet.

meter	10	20	30	40	50	60	70	80	90	100
sekunder	1.89	2.88	3.78	4.64	5.47	6.29	7.1	7.92	8.75	9.58

a) I figuren under har vi brukt datasettet fra tabellen til å utføre regresjon med et fjerdegradspolynom. Hva er det som er helt feil med denne tilnærmingen?



b) I datasettet kan vi legge til et punkt som vil hjelpe med å korrigere feilen poengtert i a). Hvilket punkt er dette?

c) Bruk regresjon med et fjerdegradspolynom på datasettet fra b).

d) Ut ifra funksjonen du fant i c), hva var toppfarten til Bolt under dette løpet?

e) Bruk datasettet fra b) til å finne gjennomsnittsfarten til Bolt for $t \in [0, 1.89]$ og for $t \in [1.89, 9.58]$. Sammenlikn disse hastighetene med svaret fra oppgave d), og drøft årsaken til ulikhetene/likhetene.