TILLEGG D

Symbolske beregninger

Dette kapitlet viser hvordan vi kan bruke biblioteket Sympy til å utføre symbolske beregninger. Dette tilsvarer operasjonene man gjør i CAS i GeoGebra. Biblioteket må være installert for at programmene skal fungere. Se side 361 dersom du trenger hjelp med installering.

D.1 Trekke sammen bokstavuttrykk

```python
import sympy as sp

# Definer symbolske variabler
a, b = sp.symbols('a b')
uttrykk = 3*a - 4*b + 7*a + 8*b
forenklet_uttrykk = sp.simplify(uttrykk)

print(forenklet_uttrykk)
```

```
10*a + 4*b
```

Programmet trekker sammen uttrykket $3a-4b+7a+8b$ til $10a+4b$.

D.2 Faktorisering av bokstavuttrykk

```python
import sympy as sp

# Definer symbolske variabler x og a
x, a = sp.symbols('x a')
uttrykk = 2*x**3 - 4*x**2 + 6*a*x
faktorisert_uttrykk = sp.factor(uttrykk)

print(faktorisert_uttrykk)
```

```
2*x*(x**2 - 2*x + 3*a)
```

Programmet faktoriserer uttrykket $2x^3-4x^2+6ax$ til $2x(x^2-2x+3a)$.

©Jørgensen & Dahl Forlag AS

D.3 Forkorting av brøk med flerleddet teller og nevner

```
import sympy as sp

x = sp.symbols('x')
brok = (x**2 - 1)/(x - 1)
forkortet_brok = sp.simplify(brok)

print(forkortet_brok)
```

```
x + 1
```

Programmet forkorter brøken $\frac{x^2-1}{x-1}$ til $x+1$.

D.4 Multiplikasjon av flerleddede uttrykk

```
import sympy as sp

a, b = sp.symbols('a b')
uttrykk1 = a**2 - b
uttrykk2 = a-b
produkt = uttrykk1 * uttrykk2
resultat = sp.expand(uttrykk1 * uttrykk2)

print(f"{produkt} = {resultat}")
```

```
(a - b)*(a**2 - b) = a**3 - a**2*b - a*b + b**2
```

Programmet ganger sammen uttrykkene $a^2 - b$ og $a - b$, hvilket gir $a^3 - a^2 \cdot b - ab + b^2$.

D.5 Sammentrekning av brøker

```
import sympy as sp

brok1 = sp.Rational(1, 3)
brok2 = sp.Rational(4, 5) * sp.Rational(7, 2)

sum_brok = brok1 + brok2
print(sum_brok)
```


©Jørgensen & Dahl Forlag AS

```
47/15
```

Programmet regner ut $\frac{1}{3} + \frac{4}{5} \times \frac{7}{2}$ og gir svaret som brøk.

D.6 Brøker med symboler og fellesnevner

```python
import sympy as sp

a, b = sp.symbols('a b')

uttrykk = a / 3 + (4 / b) * (7*a / 2)
# Kombiner til én brøk
en_brok = sp.together(uttrykk)
print(uttrykk)
print(en_brok)
```

```
a/3 + 14*a/b
a*(b + 42)/(3*b)
```

D.7 Polynomdivisjon

```python
import sympy as sp

x = sp.symbols('x')
P = x**3 - 3*x**2 + 3*x - 1
Q = x - 1

kvotient, rest = sp.div(P, Q)

print("Kvotient:", kvotient)
print("Rest:", rest)
```

```
Kvotient: x**2 - 2*x + 1
Rest: 0
```

Programmet utfører polynomdivisjon av $x^3 - 3x^2 + 3x - 1$ med $x - 1$. Resultatet er kvotienten $x^2 - 2x + 1$ og resten 0.

©Jørgensen & Dahl Forlag AS

D.8 Løse førstegradslikning

```
import sympy as sp

x = sp.symbols('x')
likning = sp.Eq(3*x - 6, 8*x + 5)
losning = sp.solve(likning, x)

print("Løsning:", losning[0])
```

```
Løsning: -11/5
```

Programmet løser 1.gradslikningen $3x-6=8x+5$ og finner at løsningen er $x=-\frac{11}{5}$.

D.9 Andregradslikning

```
import sympy as sp

x = sp.symbols('x')
likning = -x**2 + 20/7*x + 3/7

losninger = sp.solve(likning, x)

print("Løsning 1:", losninger[0])
print("Løsning 2:", losninger[1])
```

```
Løsning 1: -0.142857142857143
Løsning 2: 3.00000000000000
```

Programmet løser andregradslikningen $-x^2+\frac{20}{7}x+\frac{3}{7}=0$ og gir to løsninger for x.

©Jørgensen & Dahl Forlag AS

D.10 Andregradslikning: løsninger som brøk

```python
import sympy as sp

x = sp.symbols('x')
likning = -x**2 + sp.Rational(20, 7)*x + sp.Rational(3, 7)

losninger = sp.solve(likning, x)

print("Løsning 1:", losninger[0])
print("Løsning 2:", losninger[1])
```

```
Løsning 1: -1/7
Løsning 2: 3
```

Programmet løser andregradslikningen $-x^2 + \frac{20}{7}x + \frac{3}{7} = 0$ og gir to løsninger for x, begge representert som brøk (med mindre løsningen er et helt tall).

D.11 Likning med t og rotuttrykk

```python
import sympy as sp

t = sp.symbols('t')
likning = sp.Eq(t**2, 2)

losninger = sp.solve(likning, t)

print("Løsning 1:", losninger[0])
print("Løsning 2:", losninger[1])
```

```
Løsning 1: -\sqrt{2}
Løsning 2: \sqrt{2}
```

Programmet løser likningen $t^2 = 2$ og gir to løsninger for t, begge representert med rotuttrykk.

©Jørgensen & Dahl Forlag AS

D.12 Løsning av eksponentiallikning med prøve

```
import sympy as sp

x = sp.symbols('x')
likning = sp.Eq(sp.exp(x**2 - 4), 2)

x1, x2 = sp.solve(likning, x)

print("x1 =", x1)
print("x2 =", x2)

# Setter prøve på løsningen x1
vs = sp.exp(x**2 - 4)
vs_x1 = vs.subs(x, x1)
hs = 2
if vs_x1 == hs:
    print(f"x1 = {x1} løser likningen")
```

```
x1 = -sqrt(log(2) + 4)
x2 = sqrt(log(2) + 4)
x1 = -sqrt(log(2) + 4) løser likningen
```

Programmet løser eksponentiallikningen $e^{x^2-4} = 2$, og setter prøve på den ene løsningen.

D.13 Løsning av likninger med ln og lg

```
import sympy as sp

x = sp.symbols('x')

# Likning med ln
likning_ln = sp.Eq(sp.ln(x), 2)
losning_ln = sp.solve(likning_ln, x)

# Likning med lg (logaritme med base 10)
likning_lg = sp.Eq(sp.log(x, 10), 3)
losning_lg = sp.solve(likning_lg, x)
# Prøv uten * og se forskjellen
print(*losning_ln)  # Eller print(losning_ln[0])
print(*losning_lg)
```


©Jørgensen & Dahl Forlag AS

```
exp(2)
1000
```

Programmet løser likningene $\ln x = 2$ og $\lg x = 3$, og finner at løsningene er henholdsvis $x = e^2$ og $x = 1000$.

D.14 Omgjøring av formel

```python
import sympy as sp
# Bokstaven O,ikke null
l, b, O = sp.symbols('l b O')
formel_for_O = 2*l + 2*b
likning = sp.Eq(O, formel_for_O)

# Løs for b i formelen
formel_for_b = sp.solve(likning, b)

print("Formel for b:", formel_for_b[0])
```

```
Formel for b: O/2 - l
```

Programmet omformer formelen $O = 2l + 2b$ for å uttrykke b ved O og l. Resultatet er $b = \frac{O}{2} - l$.

D.15 Beregning av den deriverte

```python
import sympy as sp

x = sp.symbols('x')
f = 3*x**4 - 5*x**2 + 10

fd = sp.diff(f, x)

print(f"Den deriverte av f(x) er: {fd}")
```

```
Den deriverte av f(x) er: 12*x**3 - 10*x
```

Programmet beregner den deriverte av funksjonen $f(x) = 3x^4 - 5x^2 + 10$. Resultatet viser at den deriverte, $f'(x)$, er $12x^3 - 10x$.

©Jørgensen & Dahl Forlag AS

D.16 Løsning av likningssett med to ukjente

```
import sympy as sp

x, y = sp.symbols('x y')
likning1 = sp.Eq(2*x + 3*y, 7)
likning2 = sp.Eq(5*x - y, 6)

losning = sp.solve((likning1, likning2), (x, y))
print("x =", losning[x])
print("y =", losning[y])
```

```
x = 25/17
y = 23/17
```

Programmet løser likningssettet $2x+3y=7 \wedge 5x-y=6$, og finner at løsningene er $x = {}^{25}/_{17}$ og $y = {}^{23}/_{17}$.

D.17 Løsning av likningssett med tre ukjente

```
import sympy as sp

x, y, z = sp.symbols('x y z')
likning1 = sp.Eq(x + y + z, 6)
likning2 = sp.Eq(2*y - z, 3)
likning3 = sp.Eq(x + 2*z, 7)

losning = sp.solve((likning1, likning2, likning3), (x, y, z))

print("x =", losning[x])
print("y =", losning[y])
print("z =", losning[z])
```

```
x = -3
y = 4
z = 5
```

Programmet løser likningssettet

$$\begin{aligned} x+y+z &= 6 \\ 2y-z &= 3 \\ x+2z &= 7 \end{aligned}$$

og finner at løsningene er $x=-3$, $y=4$, og $z=5$.

©Jørgensen & Dahl Forlag AS

D.18 Løsning av en lineær ulikhet

```python
import sympy as sp

x = sp.symbols('x')
ulikhet = 3*x - 5 <= 7

losning = sp.solve_univariate_inequality(ulikhet, x)

print(losning)
```

```
(-oo < x) & (x <= 4)
```

Programmet løser den lineære ulikheten $3x - 5 \leq 7$ og finner at løsningen er $x \leq 4$. Merk at løsningen blir presentert som $-\infty < x \leq 4$, som er ekvivalent med $x \leq 4$.

D.19 Løsning av en kvadratisk ulikhet

```python
import sympy as sp
from sympy.solvers.solveset import solveset_real

x = sp.symbols('x')
ulikhet = -3*x**2 + 10 < x

losning = solveset_real(ulikhet, x)
print(losning)
```

```
Union(Interval.open(-oo, -2), Interval.open(5/3, oo))
```

Programmet løser den kvadratiske ulikheten $-3x^2 + 10 < x$ og finner at løsningen er $L = \langle -\infty, -2 \rangle \bigcup \langle 5/3, \infty \rangle$.

Brukere av Jupyter Notebook eller tilsvarende plattform kan ha glede av kommandoen `pprint` i stedet for `print`. Det gir muligheter for å skrive ut løsninger med vakker matematisk notasjon og symboler. Kommandoen må importeres riktig. Søk på nett!

©Jørgensen & Dahl Forlag AS

D.20 Bestemme andregradsuttrykk gitt to punkter

```
import sympy as sp
# Definer variabler
x, a, b, c = sp.symbols('x a b c')

# Gitt punktene
P = (0, -3)
Q = (1, -4)
R = (5, 12)

# Opprett likninger basert på punktene
likning1 = sp.Eq(a*P[0]**2 + b*P[0] + c, P[1])
likning2 = sp.Eq(a*Q[0]**2 + b*Q[0] + c, Q[1])
likning3 = sp.Eq(a*R[0]**2 + b*R[0] + c, R[1])

# Løs systemet av likninger
L = sp.solve((likning1, likning2, likning3), (a, b, c))
a, b, c = L.values()  # L er en ordbok (dict)
print(f"{a = }, {b = }, {c = }")
# Sett inn løsningene for a, b og c i andregradsuttrykket
andregradsuttrykk = a*x**2 + b*x + c

print("Andregradsuttrykket er:", andregradsuttrykk)
```

D.21 Eksakt beregning av sinusverdi for en vinkel i grader

```
import sympy as sp

vinkel_grader = 45
vinkel_radianer = sp.rad(vinkel_grader)

# Beregn eksakt sinusverdi
sin_verdi_eksakt = sp.sin(vinkel_radianer)

print(f"Sinus av {vinkel_grader}° er {sin_verdi_eksakt}")
print(f"Avrundet til 3 desimaler: {sin_verdi_eksakt:.3f}")
```

```
Sinus av 45° er \sqrt{2}/2
Avrundet til 3 desimaler: 0.707
```

Programmet beregner $\sin 45^\circ$ eksakt, og viser også verdien avrundet til tre desimaler.

©Jørgensen & Dahl Forlag AS

D.22 Beregning av grenseverdier

```
import sympy as sp

x = sp.symbols('x')
f = sp.sin(x) / x

# Beregn grenseverdien for f når x går mot 0
grense_f = sp.limit(f, x, 0)

g = (4*x**2 - 3) / (5*x - x**2)
# Beregn grenseverdien for g når x går mot uendelig
grense_g = sp.limit(g, x, sp.oo)   # To o-er, ikke null

print(f"Grenseverdien til f når x går mot 0: {grense_f}")
print(f"Grenseverdien til g når x går mot uendelig: {grense_g}")

```

```
Grenseverdien til sin(x)/x når x går mot 0 er: 1
Grenseverdien til (4x^2-3)/(5x-x^2) når x går mot uendelig er: -4
```

Programmet beregner grenseverdiene $\lim_{x\to 0} \frac{\sin(x)}{x}$ og $\lim_{x\to\infty} \frac{4x^2-3}{5x-x^2}$.

D.23 Omvendt funksjon

```
import sympy as sp
x, y = sp.symbols('x y')
h = sp.sqrt(x**2 - 1)

# Definer likningen x = h(y)
likning = sp.Eq(x, h.subs(x, y))

# Løs likningen for y
losninger = sp.solve(likning, y)
print(losninger)   # Løsningene må vurderes
print("Omvendt funksjon h_invers =", losninger[1])
```

```
[-sqrt(x**2 + 1), sqrt(x**2 + 1)]
Omvendt funksjon h_invers = sqrt(x**2 + 1)
```

Programmet bestemmer uttrykket til $h^{-1}(x)$ når $h(x) = \sqrt{x^2-1}$.

©Jørgensen & Dahl Forlag AS

D.24 Bestemme parameterframstilling gitt to punkter

```python
import sympy as sp

# Definer variabler
t = sp.symbols('t')
A = sp.Matrix([-3, 5])
B = sp.Matrix([1, -2])

# Parameterframstilling
P = A + t*(B - A)

print(f"x(t) = {P[0]}")
print(f"y(t) = {P[1]}")
```

```
x(t) = 4*t - 3
y(t) = 5 - 7*t
```

Programmet bestemmer en parameterframstilling til linja gjennom punktene $A = (-3, 5)$ og $B = (1, -2)$.

D.25 Andre- og tredjederiverte

```python
import sympy as sp

x, k = sp.symbols('x k')

# Derivasjon av cos(x)
f = sp.cos(k*x)
fd1 = sp.diff(f, x)   # Den deriverte
fd2 = sp.diff(f, x, 2)   # Den andrederiverte
fd3 = sp.diff(f, x, 3)   # Den tredjederiverte
print(f"{fd1 = }, {fd2 = } og {fd3 = }")
```

```
fd1 = -k*sin(k*x), fd2 = -k**2*cos(k*x) og fd3 = k**3*sin(k*x)
```

Programmet bestemmer første-, andre- og tredjederiverte til funksjonen $f(x) = \cos kx$, der $k \in \mathbb{R}$.

©Jørgensen & Dahl Forlag AS

D.26 Eksakt bestemt integral

```python
import sympy as sp

x = sp.symbols('x')
a, b = 1, 3  # Integralgrenser
# Definer funksjonen
f = sp.cos(x / sp.pi)
# Finn det bestemte integralet
F = sp.integrate(f, (x, a, b))

print("Eksakt verdi av integralet:", F)
print(f"Avrundet til 4 desimaler: {F:.4f}")
```

```
Eksakt verdi av integralet: -pi*sin(1/pi) + pi*sin(3/pi)
Avrundet til 4 desimaler: 1.5812
```

Programmet finner det eksakte uttrykket til integralet

$$\int_1^3 f(x)\,\mathrm{d}x = \int_1^3 \cos\left(\frac{x}{\pi}\right)\,\mathrm{d}x$$

Programmet skriver også ut integralet med 4 desimaler.

D.27 Ubestemt integral

```python
import sympy as sp

# Definer x som et symbol
x = sp.symbols('x')

# Funksjonen
f = sp.sin(x)**2
# Finn den antideriverte (ubestemt integral)
F = sp.integrate(f, x)

print("Antideriverte av sin^2(x) er:", F)
```

```
Antideriverte av sin^2(x) er: x/2 - sin(x)*cos(x)/2
```

Programmet løser det ubestemte integralet $\int f(x)\,\mathrm{d}x = \int \sin^2(x)\,\mathrm{d}x$. Merk at svaret også kan uttrykkes $\frac{x}{2} - \frac{1}{4}\sin 2x$.

©Jørgensen & Dahl Forlag AS

D.28 Forenkling av trigonometrisk uttrykk

```python
import sympy as sp

x = sp.symbols('x')

# Trigonometrisk uttrykk
uttrykk = sp.sin(x)*sp.cos(x) + 2*sp.sin(2*x)
forenklet_uttrykk = sp.simplify(uttrykk)

print(forenklet_uttrykk)
```

```
5*sin(2*x)/2
```

Programmet forenkler det trigonometriske uttrykket $\sin x \cdot \cos x + 2\sin 2x$ og finner at

$$\sin x \cdot \cos x + 2\sin 2x = \frac{5}{2}\sin 2x$$

D.29 Utvidelse av trigonometrisk uttrykk

```python
import sympy as sp

x, y = sp.symbols('x y')

# Trigonometrisk uttrykk
uttrykk = sp.sin(x + y)
utvidet_uttrykk = sp.expand_trig(uttrykk)

print(utvidet_uttrykk)
```

```
sin(x)*cos(y) + sin(y)*cos(x)
```

Programmet utvider det trigonometriske uttrykket $\sin(x+y)$ ved hjelp av trigonometriske identiteter og gir den utvidede formen som output.

©Jørgensen & Dahl Forlag AS

D.30 Generell løsning av differensiallikning

```
import sympy as sp
t = sp.symbols('t')
y = sp.Function('y')(t)

yd = y.diff(t)   # y'(t)
ydd = yd.diff(t)   # y''(t)
diff_likn = sp.Eq(ydd + 2*yd + y, 0)
losning = sp.dsolve(diff_likn, y)
print(losning)
print(f"y(t) = {losning.rhs}")   # rhs = right hand side
```

```
Eq(y(t), (C1 + C2*t)*exp(-t))
y(t) = (C1 + C2*t)*exp(-t)
```

Programmet gir den generelle løsningen til differensiallikningen $y'' + 2y' + y = 0$.

D.31 Differensiallikning med en initialbetingelse

```
import sympy as sp

x, C1 = sp.symbols('x C1')
y = sp.Function('y')(x)

# Definer differensiallikningen
diff_likning = sp.Eq(y.diff(x), y)   # y' = y

# Løs differensiallikningen
generell_losning = sp.dsolve(diff_likning, y)
print(generell_losning)
# Sett inn initialbetingelsen y(1) = 3e
init = {y.subs(x, 1): 3*sp.exp(1)}
spesiell_losning = sp.dsolve(diff_likning, y, ics= init)
print(spesiell_losning)
print(f"y(x) = {spesiell_losning.rhs}")
```


©Jørgensen & Dahl Forlag AS

D.32 Løsning av andreordens differensiallikning

```
import sympy as sp

# Definer variabler og funksjonen
x, C1, C2 = sp.symbols('x C1 C2')
y = sp.Function('y')(x)

# Definer differensiallikningen y'' - 4y' + 9y = 0
diff_likning = sp.Eq(y.diff(x, x) - 4*y.diff(x) + 9*y, 0)

# Løs differensiallikningen
generell_losning = sp.dsolve(diff_likning, y)
print(generell_losning)

# Sett inn initialbetingelsene
init = {y.subs(x, 0): 0, y.diff(x).subs(x, 0): -8}

spesiell_losning = sp.dsolve(diff_likning, y, ics=init)

print(f"y(x) = {spesiell_losning.rhs}")
```

```
Eq(y(x), (C1*sin(sqrt(5)*x) + C2*cos(sqrt(5)*x))*exp(2*x))
y(x) = -8*sqrt(5)*exp(2*x)*sin(sqrt(5)*x)/5
```

Programmet løser differensiallikningen

$$y'' - 4y' + 9y = 0$$

med initialbetingelser $y(0) = 0$ og $y'(0) = -8$. Programmet gir først den generelle løsningen

$$y(x) = C_1 \cdot \sin(\sqrt{5} \cdot x) + C_2 \cdot \cos(\sqrt{5} \cdot x) \cdot e^{2x}$$

, deretter settes initialbetingelsene inn og vi får den spesielle løsningen

$$y(x) = \frac{-8\sqrt{5}}{5} \cdot e^{2x} \cdot \sin(\sqrt{5} \cdot x)$$

.

©Jørgensen & Dahl Forlag AS