Prosjektoppgave i FYS2130, vår 2017

Modellering av bølger på streng

Kandidatnummer: 15043

9. mai 2017

I dette prosjektet er målet å simulere en vibrerende streng.

Jeg har jobbet sammen med kandidatnummer 15032. Dette kan hovedsaklig sees på grafene.

Oppgave 1

Vi bruker her ligningene (O.1), (O.2) og (O.3) fra oppgaveteksten til prosjektet.

Vi ønsker å utlede bevegelsesligningen for massepunktet

Først tar vi utgangspunkt i likning (O.2) og løser for y_i^+.

$$y_i^+ = \ddot{y}_i(\Delta t)^2 + 2y_i^0 - y_i^-$$

Fra ligning (O.3) har vi et uttrykk for kraften, dvs. akselerasjonen. Dette setter vi inn i uttrykket over slik at

$$\ddot{y} = \frac{F_i}{m_i} \;\Rightarrow\; y_i^+ = \frac{F_i}{m_i}\Delta t^2 + 2y_y^0 - y_i^- \tag{1}$$

Setter vi utrykket for F_i (ligning O.3) inn i likning 1 får vi bevegelseslikningen vi er ute etter.

$$y_i^+ = -(k_{i-1} + k_i)y_i^0 + k_{i-1}y_{i-1}^0 + k_i y_{i+1}\frac{(\Delta t)^2}{m_i} + 2y_y^0 - y_i^- \tag{2}$$

Videre må vi se på hva slags endepunkter vi jobber med. Vi kan enten ha åpne eller reflekterende ender.

Ved åpne ender setter vi kun inn kraften fra høyre i startpunkter og kraften fra venstre i endpunktet. Endepunktene vil dermed ha følgende form

$$y_0^+ = -k_i(y_i - y_{i+1})\frac{(\Delta t)^2}{m_i} + 2y_y^0 - y_i^-$$

$$y_{N-1}^{+} = -k_{i-1}(y_i - y_{i-1})\frac{(\Delta t)^2}{m_i} + 2y_y^0 - y_i^-$$

Reflekterende ender får vi når verdien på svingning i endepunktet holdes på 0. Dette er det samme som en fast vegg, og vi kan gi endemassepunktene en mye høyere masse sammenlignet med resten for å oppnå dette.

Oppgave 2

Nå ønsker vi å utlede bølgeligningen for en vibrerende streng. Vi introduserer en massetetthet,

$$\mu = m/\Delta x, \tag{3}$$

samt den konstante fjærstivheten

$$\kappa = k\Delta x \tag{4}$$

Vi har at

$$F_i = m_i\frac{d^2y}{dt^2} = -(k_{i-1} + k_i)y_i^0 + k_{i-1}y_{i-1}^0 + k_i y_{i+1}$$

Ved å sette inn for konstantene ender vi med

$$\frac{\kappa}{\mu}\left(\frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2}\right)$$

Inne i parentesen har vi den deriverte, $\frac{d^2y}{dx^2}$. Videre får vi

$$\frac{\kappa}{\mu}\frac{d^2y}{dx^2}$$

.

og får til slutt

$$\frac{d^2y}{dt^2} = v_B^2\frac{d^y}{dx^2}$$

Oppgave 3

Vi har følgende betingelse på den numeriske hastigheten

$$\frac{\Delta x}{\Delta t} > v_B \tag{5}$$

Fra oppgave 2 har vi et utrykk for utbredelseshastigheten gitt ved

$$v_B = \sqrt{\frac{\kappa}{\mu}} \tag{6}$$

Innsatt i ligningen for den numeriske hastigheten får vi

$$\frac{\Delta x}{\Delta t} > \sqrt{\frac{\kappa}{\mu}} = \sqrt{\frac{k\Delta x^2}{m}} \tag{7}$$

Dermed blir en betingelse for tidssteget at

$$\Delta t < \sqrt{\frac{m}{k}} \tag{8}$$

Her ser vi at posisjonssteget Δx er uavhengig av m og k. Det er ingen forflytting i x-aksen, og det eneste Δx gir er lengden på strengen, gitt antall massepunkter ($L = N\Delta x$). Vi vil derfor bruke at $\Delta x = 1$.

I dette prosjektet vil vi fremover bruke at

$$\Delta t = 0.1 \cdot \sqrt{\frac{m}{k}} \tag{9}$$

Oppgave 4

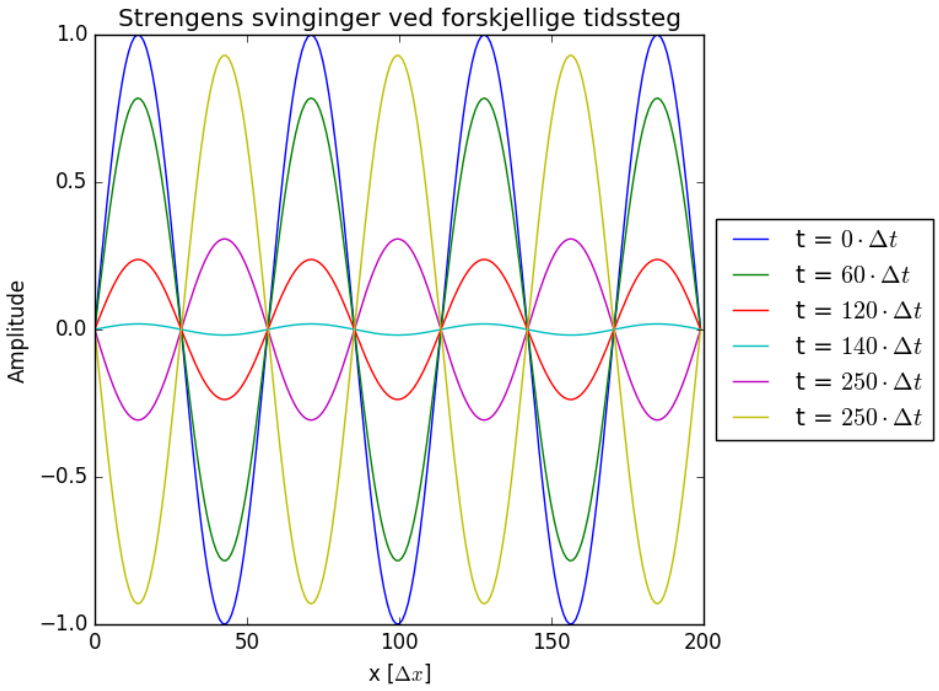
Vi har nå alt verktøyet for å kunne regne på svingebevegelsen. Startbetingelsen vår er

$$y_i^0 = \sin\left(7\pi \frac{i}{N-1}\right), \qquad 0 \leq i \leq N-1 \tag{10}$$

Vi setter også denne verdien for det siste tidssteget, $y_i(t_{\text{max}})$. Grunnen til dette er at vi trenger en god posisjon *før* vi begynner å telle. Siden Python regner element -1 som det siste, slipper vi å legge inn en ekstra if-test for startbetingelsene. Denne verdien blir uansett overskrevet til slutt, og virker kun som en «proxy» for $y_i(-\Delta t)$

Hadde vi valgt $y_i(-\Delta t) = 0$ ville det vært som om strengen plutselig hoppet fra ro til en sinusbølge, som gir et annet resultat enn det vi er interessert i.

Resultatet kan sees i figur . Vi ser her utbredelsen av en stående bølge. Strengen vil altså bevege seg opp og ned på samme sted.



Figur 1:

Oppgave 5

Målet vårt i denne oppgaven er å finne frekvensen til strengens svingning ved midtpunkt ($y_{99}(t)$) ved å bruke tre forskjellige metoder: en analytisk (forventet) verdi, en numerisk verdi som vi «teller» oss frem til. og en verdi fra frekvensspektret gitt ved Fourier-transformasjonen. Selve bevegelsen kan sees i figur

Vi begynner med den «analytiske» løsningen, og bruker denne til å finne antall tidssteg som trengs for ti perioder.

En bølge har egenskapen at utbredelseshastigheten er gitt ved

$$v_B = f\lambda \tag{11}$$

Bølgelengden λ er gitt ved bølgetallet n. Vi kan se på n som antall fysiske topper strengen har når den vibrerer. Fra initialbetingelsen $\sin\left(7\pi\frac{i}{N-1}\right)$ ser vi at vi må ha bølgetallet $n = \frac{7\pi}{2\pi} = 3.5$. Siden strengen er N punkter lang, kan vi finne bølgelengden ved

$$\lambda = \frac{N}{n} = \frac{200}{3.5} \tag{12}$$

i vårt tilfelle.

Nå kan vi regne ut den analytiske frekvensen

$$f_{analytisk} = \frac{v_B}{\lambda} = 0.391311896062 \text{ Hz} \tag{13}$$

Denne frekvensen gir oss mulighet til å regne ut antall tidssteg for ti perioder.

$$\#\text{tidssteg} = \frac{10}{\Delta t \cdot f} \sim 5715 \tag{14}$$

når vi runder opp.

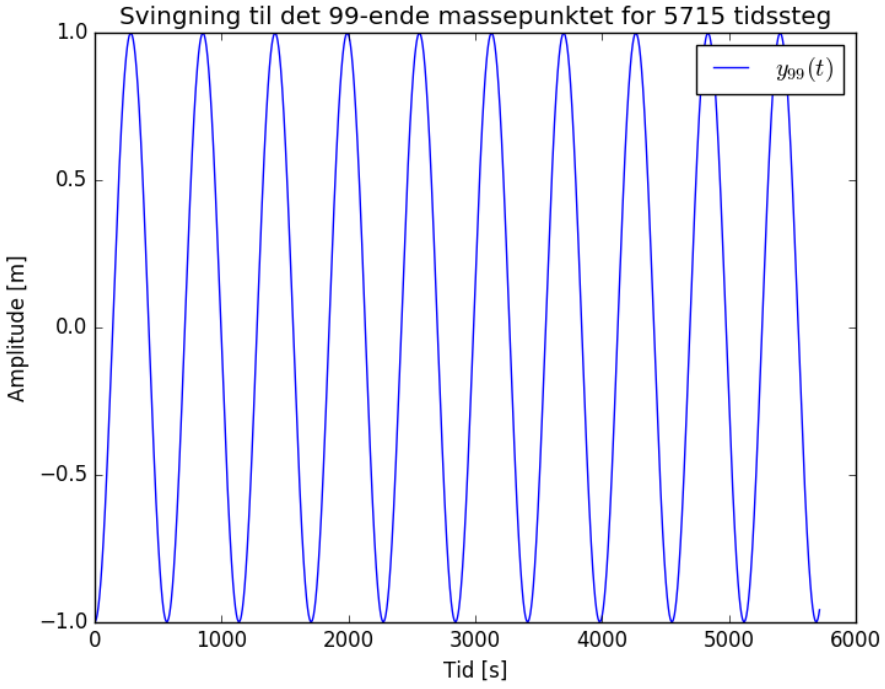
Videre vil vi regne frekvensen numerisk ved å telle hvor mange perioder vi har i et tidsrom. Jeg gjorde dette ved å velge startpunktet til funskjonen, $y_{99}(t_0 = 0) = -1$, og sjekke hvor mange ganger den oppnådde denne verdien. Deretter regnet jeg ut frekvensen ved

$$f_{numerisk} = \frac{\#\text{tellinger}}{\Delta t(t_1 - t_0)} = 0.393120249209 \text{ Hz} \tag{15}$$

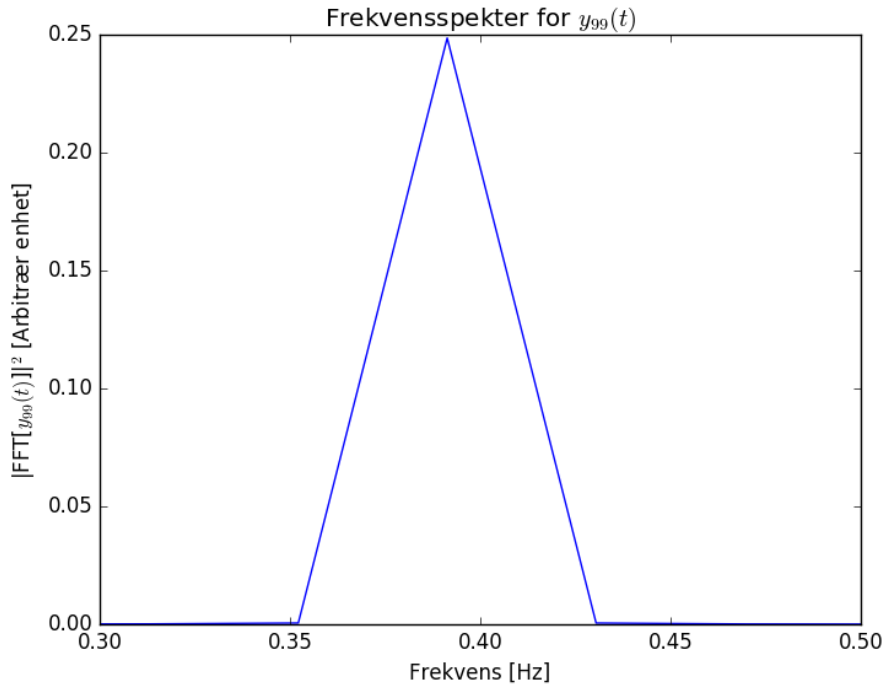
En annen måte for å regne ut den numeriske frekvensen hadde vært å sjekke hvor ofte funksjonen byttet fortegn. Denne ville vært mer generell, da jeg har som utgangspunkt at den første y-verdien er -1.

Til slutt ser vi på frekvensspektret til funksjonen gjennom å Fourier-transformere signalet. Resultatet kan sees i figur . Verdien i toppunktet er

$$f_{fourier} = 0.391331462636 \text{ H Hz} \tag{16}$$



Figur 2:



Figur 3:

Oppgave 6

For å regne ut energien til hele strengen ved ett tidssteg, må vi summere kinetisk og potensiell energi over alle i-verdier.

For en fjær er kinetisk energi gitt ved

$$E_k = \frac{1}{2}\sum_i m_i v_i^2 \tag{17}$$

og potensiell energi ved

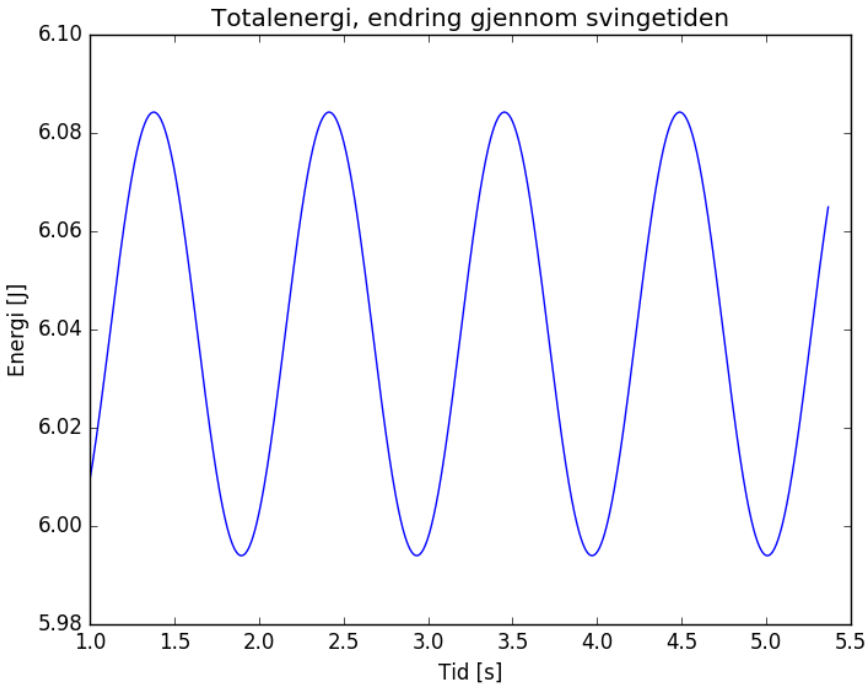
$$E_p = \frac{1}{2}\sum_i k_i \Delta y^2 \tag{18}$$

For strengen tilsvarer det at $\delta y = y_{i+1} - y_i$. Hastigheten v_i har vi ikke et uttrykk for, men siden vi har diskretisert strengen, kan vi regne ut denne numerisk ved

$$v_i^0 = \frac{y_i^+ - y_i^0}{\Delta t} \tag{19}$$

Vi har plottet resultatet i figur .

Her ser vi at vi har en svingning i energi, men den er relativt liten. Det vil si at energien er tilnærmet konservert, men siden vi regner diskretisert er det naturlig å forvente små svinginger.



Figur 4:

Oppgave 7

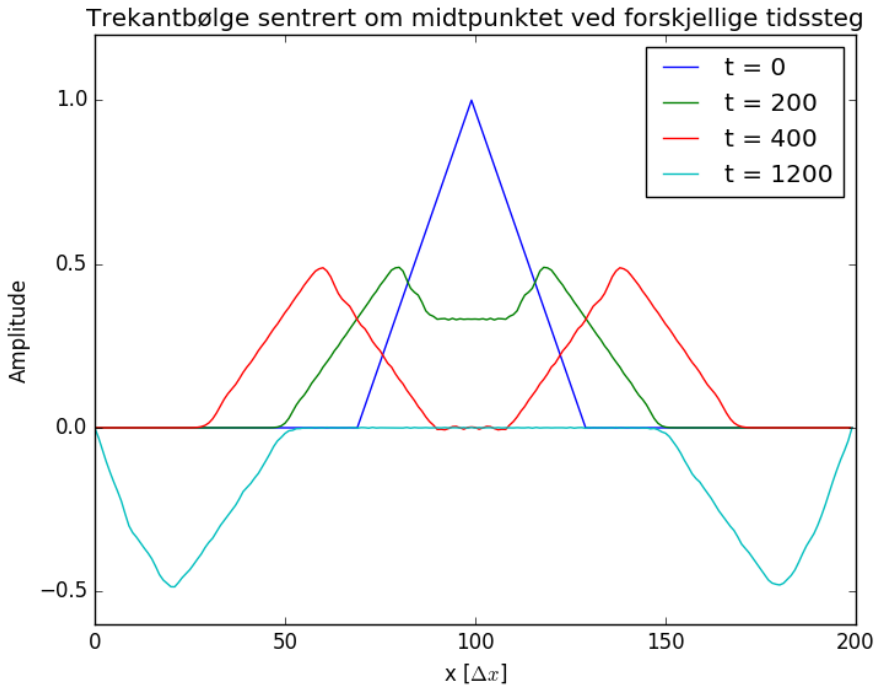
Vi vil nå se på en bølge formet som en trekant. Den vil ha følgende initialbetingelser

$$y_i^0 = y_i^- = \begin{cases} (i-69)/30 & 70 \leq i \leq 99 \\ (129-i)/30 & 100 \leq i \leq 128 \\ 0 & \text{else} \end{cases} \tag{20}$$

Ved å bruke programmet vi utviklet i oppgave 4 og endre initialbetingelsene til det ovenfor, får vi bevegelsen i figur .

Her ser vi at bølgen deler seg i to og beveger seg i hver sin retning, hver med halvparten av amplituden til den original bølgen. Når bølgene treffer sine respektive vegger vil reflekteres ned på undersiden av strengen. Deretter møtes de igjen og blir en stor bølge med den original amplituden.

Når vi setter initialbetingelsene til å være slik at $y_i^0 = y_i^-$, altså at startbetingelsen og posisjonen før er like, vil det være det samme som at vi har en superposisjon av to mindre bølger som beveger seg i hver sin retning. Dermed vil bølgen dele seg i to i de neste tidsstegene og gå i hver sin retning.



Figur 5:

Oppgave 8

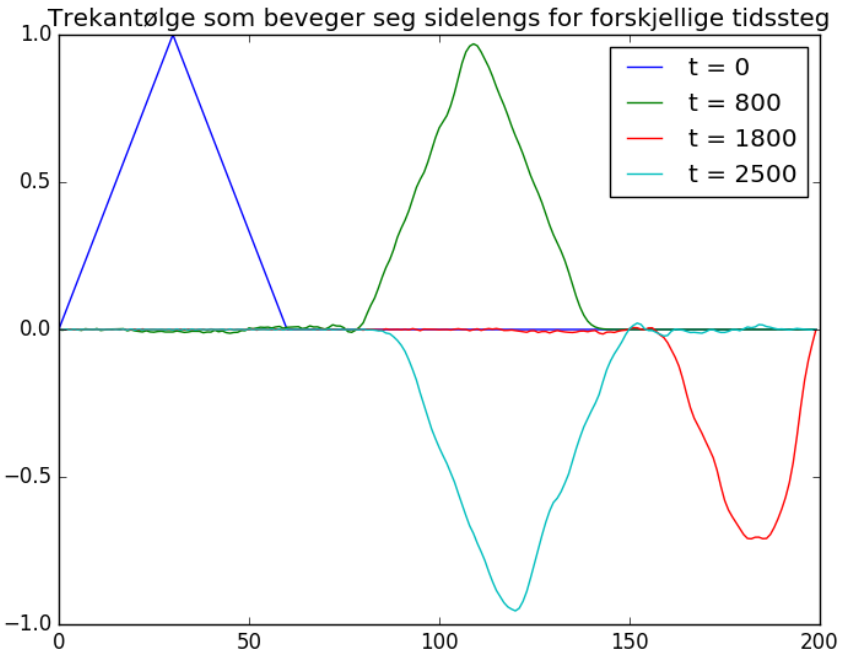
Vi sentrerer nå strengen rundt indeks $i = 30$, slik

$$y_i^0 = \begin{cases} \frac{i}{30} & 1 \leq i \leq 30 \\ \frac{60-i}{30} & 31 \leq i \leq 60 \\ 0 & \text{else} \end{cases} \tag{21}$$

Deretter vil vi at trekanten skal bevege seg mot høyre. Det gjør vi det å legge til pre-initialbetingelsen

$$y_i^- = \begin{cases} y_i^0 + \Delta t \cdot v_B/30 & 0 \leq i \leq 29 \\ y_i^0 - \Delta t \cdot v_B/30 & 30 \leq i \leq 59 \\ 0 & \text{else} \end{cases} \tag{22}$$

Dette er for å gi verdien i det forrige tidssteget som endringen av det som ville vært før om bølgen hadde kommet fra venstre. Se figur .



Figur 6:

Oppgave 9

Nå lager vi tråden 200 punkter lenger og gjør disse massepunktene tre ganger så tunge. Vi sender fortsatt bølgen til høyre, men nå vil deler av den bli reflektert tilbake, mens resten vil gå over i den tykkere enden av tråden. Se figur .

Vi får at transmittert bølge har amplitude

$$T \sim 0.693 \tag{23}$$

og reflektert bølge har amplitude

$$R \sim -0.259 \tag{24}$$

Vi kan regne ut forholdet i impedans som

$$\frac{z_1}{z_0} = \sqrt{\frac{3mk}{mk}} = \sqrt{3} \sim 1.732 \tag{25}$$

De følgende relasjonene gjelder for reflesjons- og transmisjonsamplitudene

$$R = \frac{z_0 - z_1}{z_0 + z_1} \tag{26}$$

$$T = \frac{2z_0}{z_0 + z_1} \tag{27}$$

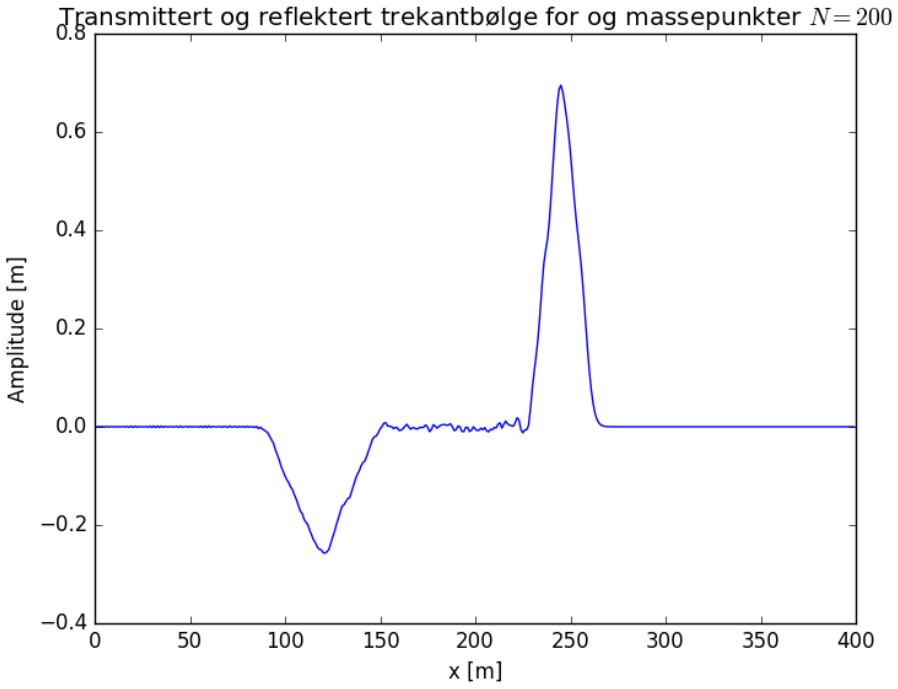
Det gir oss følgende forhold for den reflekterte bølgen

$$\frac{z_1}{z_0} = \frac{1-R}{1+R} \sim 1.699 \tag{28}$$

og følgende for den transmitterte

$$\frac{z_1}{z_0} = \frac{2}{T} - 1 \sim 1.886 \tag{29}$$

Som vi ser av dette ligger verdiene våre ganske nærme opptil den teoretiske verdien. Fordi vi gjør disse utregningene diskret og numerisk vil det alltid være litt støy. Som vi ser av figur beveger ikke bølgen seg som en perfekt trekant. Dermed kan vi ikke forvente at den teoretiske verdien vil stemme perfekt overens med våre.



Figur 7:

Appendiks – kodesnutter

Dette er programmene som er brukt for å kjøre koden som lager plottene over. Noen steder vil det være mulig å kommentere inn visse deler for å se animasjoner av bølgebevegelsene.

Oppgave 4

Svingebevegelse

```
import numpy as np
import matplotlib.pyplot as plt
```

```
import matplotlib.animation as an

# Number of mass points
N = 200

# All mass points have the same mass (except endpoints):
m = 0.02*np.ones(N) # kg
m[0] = m[-1] = 1.

# Spring stiffness / constant
k = 10*np.ones(N) # kg/s^2

vB = np.sqrt(k[10]/m[10])

wave_length = N/3.5
frequency = vB/wave_length


print("prop speed: ", vB)
print("Frequency: ", frequency)

# Length of time step
dt = 0.1*np.sqrt(m[10]/k[10]) # s
numberOfTimeSteps = 1200
t_max = numberOfTimeSteps*dt # s
time = dt*range(numberOfTimeSteps)

# Matrix of the fluctuations of the string (for time and x values)
# End points are zero due to reflecting edges.
y = np.zeros([N,numberOfTimeSteps])

# Set the initial conditions, t = 0
for i in range(N):
    y[i,0] = np.sin(7*np.pi*i/(N-1))
    y[i,numberOfTimeSteps-1] = np.sin(7*np.pi*i/(N-1))

for j in range(numberOfTimeSteps-1):
    for i in range(1,N-1):

        F_left = -k[i-1]*(y[i,j] - y[i-1,j])
        F_right = -k[i]*(y[i,j] - y[i+1,j])
        F_i = F_left + F_right

        # y[i,j-1] = 0 since we haven't calculated y[i,-1] yet:
        y[i,j+1] = F_i*dt**2/m[i] + 2*y[i,j] - y[i,j-1]
```

```
# plt.legend(["timestep = 0", "timestep = 60",  "timestep = 120", "timestep = 1


fig = plt.figure()
ax = plt.subplot(111)

ax.plot(y[:,0], label=r"t = $0\cdot\Delta t$")
ax.plot(y[:,60], label=r"t = $60\cdot\Delta t$")
ax.plot(y[:,120], label=r"t = $120\cdot\Delta t$")
ax.plot(y[:,140], label=r"t = $140\cdot\Delta t$")
ax.plot(y[:,170], label=r"t = $250\cdot\Delta t$")
ax.plot(y[:,250], label=r"t = $250\cdot\Delta t$")
plt.title("Strengens svinginger ved forskjellige tidssteg")
ax.set_xlabel(r"x [$\Delta x$]")
ax.set_ylabel("Amplitude")

# Shrink current axis by 20%
box = ax.get_position()
ax.set_position([box.x0, box.y0, box.width * 0.8, box.height])

# Put a legend to the right of the current axis
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.savefig("problem4.png")

plt.show()
```

Oppgave 5

Frekvensspekter

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as an

# Number of mass points
N = 200

# All mass points have the same mass (except endpoints):
m = 0.02*np.ones(N) # kg
m[0] = m[-1] = 1.

# Spring stiffness / constant
k = 10*np.ones(N) # kg/s^2

vB = np.sqrt(k[10]/m[10])

wave_length = N/3.5
frequency = vB/wave_length
period = 1./frequency

t_max = 10*period

print("Propagation speed: ", vB)
print("Frequency: ", frequency)

# Length of time step
dt = 0.1*np.sqrt(m[10]/k[10]) # s
numberOfTimeSteps = int(t_max/dt) + 1
print("t_max: ", t_max)
print("Time steps: ", numberOfTimeSteps)

# Matrix of the fluctuations of the string (for time and x values)
# End points are zero due to reflecting edges.
y = np.zeros([N, numberOfTimeSteps])

# Set the initial conditions, t = 0 and
for i in range(N):
    y[i,0] = np.sin(7*np.pi*i/(N-1))

    # Since we don't have an expression for y(-dt)
    y[i,numberOfTimeSteps-1] = np.sin(7*np.pi*i/(N-1))

t_0 = 0
t_1 = 0
counted_freq = 0
```

```
tol = 0.00005
cnt = 0


for j in range(numberOfTimeSteps-1):
    for i in range(1,N-1):

        F_left = -k[i-1]*(y[i,j] - y[i-1,j])
        F_right = -k[i]*(y[i,j] - y[i+1,j])
        F_i = F_left + F_right

        y[i,j+1] = F_i*dt**2/m[i] + 2*y[i,j] - y[i,j-1]

    # Count how many minimum points we encounter in y_99(t):
    if y[99, j+1] - y[99, t_0] < tol and j+1-t_0 > 1:
        t_1 = j+1

        counted_freq = 1./(dt*(t_1 - t_0))
        t_0 = t_1

        cnt += 1

print("Counted frequency: ", cnt/(dt*t_1))

# The take the Fast Fourier Transform of y_99(t):
y99_abs = abs(np.fft.fft(y[99,:]))**2

freqs = np.linspace(0, 1./dt, numberOfTimeSteps)
print("FFT frequency: ", freqs[10])

plt.plot(range(numberOfTimeSteps)/t_max, y99_abs[:]/numberOfTimeSteps**2)
plt.axis([0.3,0.5,0,0.25])
plt.title(r"Frekvensspekter for $y_{99}(t)$")
plt.xlabel("Frekvens [Hz]")
plt.ylabel(r"|FFT[$y_{99}(t)$]|$^2$ [Arbitr r enhet]")
plt.savefig("problem5-frekvens.png")
plt.show()

plt.plot(y[99, :])
plt.title("Svingning til det 99-ende massepunktet for %d tidssteg" % numberOfTi
plt.xlabel("Tid [s]")
plt.ylabel("Amplitude [m]")
plt.legend([r"$y_{99}(t)$"])
plt.savefig("problem5-svingning.png")
plt.show()
```

Oppgave 6

Energikonservering

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as an

# Number of mass points
N = 200

# All mass points have the same mass (except endpoints):
m = 0.02*np.ones(N) # kg
m[0] = m[-1] = 1.

# Spring stiffness / constant
k = 10*np.ones(N) # kg/s^2

vB = np.sqrt(k[10]/m[10])

# Length of time step
dt = 0.1*np.sqrt(m[10]/k[10]) # s
numberOfTimeSteps = 1200 #int(t_max/dt) + 1
t_max = 1200*dt
print("t_max: ", t_max)
print("Time steps: ", numberOfTimeSteps)

# Matrix of the fluctuations of the string (for time and x values)
# End points are zero due to reflecting edges.
y = np.zeros([N, numberOfTimeSteps])

# Set the initial conditions, t = 0 and
for i in range(N):
    y[i,0] = np.sin(7*np.pi*i/(N-1))

    # Since we don't have an expression for y(-dt)
    y[i,numberOfTimeSteps-1] = np.sin(7*np.pi*i/(N-1))

totalEnergy = np.zeros(numberOfTimeSteps)

for j in range(numberOfTimeSteps-1):
    potentialEnergy = 0
    kineticEnergy = 0
    for i in range(1,N-1):

        F_left = -k[i-1]*(y[i,j] - y[i-1,j])
        F_right = -k[i]*(y[i,j] - y[i+1,j])
        F_i = F_left + F_right
```

```
            y[i,j+1] = F_i*dt**2/m[i] + 2*y[i,j] - y[i,j-1]

            potentialEnergy += 0.5*k[i]*(y[i,j] - y[i-1,j])**2
            kineticEnergy += 0.5*m[i]*((y[i,j+1]-y[i,j])/dt)**2

        totalEnergy[j+1] = potentialEnergy + kineticEnergy

plt.plot(np.linspace(1,t_max,numberOfTimeSteps-1), totalEnergy[1:])
plt.xlabel("Tid [s]")
plt.ylabel("Energi [J]")
plt.title("Totalenergi, endring gjennom svingetiden")
plt.savefig("problem6.png")
plt.show()
```

Oppgave 7

Trekant på streng

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as an

# Number of mass points
N = 200

# All mass points have the same mass (except endpoints):
m = 0.02*np.ones(N) # kg
m[0] = m[-1] = 1.

# Spring stiffness / constant
k = 10*np.ones(N) # kg/s^2

vB = np.sqrt(k[10]/m[10])

wave_length = N/3.5
frequency = vB/wave_length
period = 1./frequency

t_max = 10*period

print("Propagation speed: ", vB)
print("Frequency: ", frequency)

# Length of time step
dt = 0.1*np.sqrt(m[10]/k[10]) # s
numberOfTimeSteps = int(t_max/dt) + 1
print("t_max: ", t_max)
print("Time steps: ", numberOfTimeSteps)

# Matrix of the fluctuations of the string (for time and x values)
# End points are zero due to reflecting edges.
y = np.zeros([N, numberOfTimeSteps])

# Set the initial conditions, t = 0 and
for i in range(N):
    if 70 <= i <= 99:
        y[i,0] = (i - 69.)/30.
        y[i,-1] = y[i,0]
    elif 100 <= i <= 128:
        y[i,0] = (129. - i)/30.
        y[i,-1] = y[i,0]
    else:
        y[i,0] = 0
```

```
        y[i,-1] = y[i,0]


# fig = plt.figure()
# images = []

for j in range(numberOfTimeSteps-1):
    for i in range(1,N-1):

        F_left = -k[i-1]*(y[i,j] - y[i-1,j])
        F_right = -k[i]*(y[i,j] - y[i+1,j])
        F_i = F_left + F_right

        y[i,j+1] = F_i*dt**2/m[i] + 2*y[i,j] - y[i,j-1]

    # images.append(plt.plot(y[:,j+1], "-b"))


plt.plot(y[:,0])
plt.plot(y[:,200])
plt.plot(y[:,400])
plt.plot(y[:,1200])
plt.title("Trekantb lge sentrert om midtpunktet ved forskjellige tidssteg")
plt.xlabel(r"x [$\Delta x$]")
plt.ylabel("Amplitude")
plt.legend(["t = 0", "t = 200", "t = 400", "t = 1200"])
plt.savefig("problem7.png")
plt.show()

# theanimation = an.ArtistAnimation(fig, images, interval=10)
# plt.show()
```

Oppgave 8

Trekant som beveger seg

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as an

# Number of mass points
N = 200

# All mass points have the same mass (except endpoints):
m = 0.02*np.ones(N) # kg
m[0] = m[-1] = 1.

# Spring stiffness / constant
k = 10*np.ones(N) # kg/s^2

vB = np.sqrt(k[10]/m[10])

wave_length = N/3.5
frequency = vB/wave_length
period = 1./frequency

t_max = 10*period

print("Propagation speed: ", vB)
print("Frequency: ", frequency)

# Length of time step
dt = 0.1*np.sqrt(m[10]/k[10]) # s
numberOfTimeSteps = int(t_max/dt) + 1
print("t_max: ", t_max)
print("Time steps: ", numberOfTimeSteps)

# Matrix of the fluctuations of the string (for time and x values)
# End points are zero due to reflecting edges.
y = np.zeros([N, numberOfTimeSteps])

# Set the initial conditions, t = 0 and
for i in range(N):
    if 1 <= i <= 30:
        y[i,0] = i/30.
    elif 31 <= i <= 60:
        y[i,0] = (60. - i)/30.
    else:
        y[i,0] = 0
```

```
for i in range(N):
    if 0 <= i <= 29:
        y[i,-1] = y[i,0] + 1/30.*dt*vB
    elif 30 <= i <= 59:
        y[i,-1] = y[i,0] - 1/30.*dt*vB
    else:
        y[i,-1] = y[i,0]




# fig = plt.figure()
# images = []

for j in range(numberOfTimeSteps-1):
    for i in range(1,N-1):

        F_left = -k[i-1]*(y[i,j] - y[i-1,j])
        F_right = -k[i]*(y[i,j] - y[i+1,j])
        F_i = F_left + F_right

        y[i,j+1] = F_i*dt**2/m[i] + 2*y[i,j] - y[i,j-1]

    # images.append(plt.plot(y[:,j+1], "-b"))

plt.title("Trekant lge som beveger seg sidelengs for forskjellige tidssteg")
plt.plot(y[:,0])
plt.plot(y[:,800])
plt.plot(y[:,1800])
plt.plot(y[:,2500])
plt.legend(["t = 0", "t = 800", "t = 1800", "t = 2500"])
plt.xlabel("")
plt.savefig("problem8.png")
plt.show()

# theanimation = an.ArtistAnimation(fig, images, interval=10)
# plt.show()
```

Oppgave 9

Trekant på tykkere streng

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as an

# Number of mass points
N = 400

# All mass points have the same mass (except endpoints):
m = 0.02*np.ones(N) # kg
m[0] = m[-1] = 1.
# Increase the mass of the last 200 mass points of the string
m[200:-1] *= 3.

# Spring stiffness / constant
k = 10*np.ones(N) # kg/s^2

# Propagatiion speed
vB = np.sqrt(k[10]/m[10])

# Length of time step
dt = 0.1*np.sqrt(m[10]/k[10]) # s
numberOfTimeSteps = 5000

# Matrix of the fluctuations of the string (for time and x values)
# End points are zero due to reflecting edges.
y = np.zeros([N, numberOfTimeSteps])

# Set the initial conditions, t = 0 and
for i in range(N):
    if 1 <= i <= 30:
        y[i,0] = i/30.
    elif 31 <= i <= 60:
        y[i,0] = (60. - i)/30.
    else:
        y[i,0] = 0



for i in range(N):
    if 0 <= i <= 29:
        y[i,-1] = y[i,0] + 1/30.*dt*vB
    elif 30 <= i <= 59:
        y[i,-1] = y[i,0] - 1/30.*dt*vB
    else:
        y[i,-1] = y[i,0]
```

```
fig = plt.figure()
images = []

for j in range(numberOfTimeSteps-1):
    for i in range(1,N-1):

        F_left = -k[i-1]*(y[i,j] - y[i-1,j])
        F_right = -k[i]*(y[i,j] - y[i+1,j])
        F_i = F_left + F_right

        y[i,j+1] = F_i*dt**2/m[i] + 2*y[i,j] - y[i,j-1]

    # images.append(plt.plot(y[:,j+1], "-b"))

print(np.min(y[:,2500]), np.max(y[:,2500]))
print(np.max(y[:,0]))

# theanimation = an.ArtistAnimation(fig, images, interval=10)
plt.plot(y[:,2500])
plt.title(r"Transmittert␣og␣reflektert␣trekantb lge␣for␣og␣massepunkter␣$N=200$
plt.xlabel("x␣[m]")
plt.ylabel("Amplitude␣[m]")
plt.savefig("problem9.png")
plt.show()
```