**Rapport for MATLAB-Prosjektet i Marin 2**

**Høsten 2014**

**10041, 10050 og 10028**

# Forord

I dette prosjektet er oppgaven å lage et program i MATLAB som vil kunne utføre beregninger på en ramme i en konstruksjon. Av dette prosjektet forventer vi å få et utbytte i form av at vi får inn matrisemetoden, samt ser hvor mye lettere beregninger kan gjøres i et dataprogram sammenlignet med å gjøre utregningene manuelt. Gjennom dette vil vi også få en god forståelse for hvordan slike program fungerer, og få en viss innsikt i hvordan dette kan brukes videre i utdanningen og arbeidslivet.

Matrisemetoden har vi tidligere kunnskap om gjennom forelesninger med Jørgen Amdahl samt øvingsoppgaver om temaet, og dette prosjektet vil gjøre oss enda mer rustet til å bruke denne metoden.

Prosjektet har vært et samarbeide mellom tre studenter med kandidatnummer 10041, 10050 og 10028.

# Sammendrag

I dette prosjektet ble det laget et MATLABprogram som skal kunne analysere en konstruksjon påkjent av forskjellige lasttyper. Programmet består av flere mindre funksjoner som sammen gir ut den informasjonen man trenger om konstruksjonen og om den vil tåle påkjenningene den blir utsatt for. Programmet skriver til slutt ut informasjon om rotasjon i knutepunktene og om momentene de ulike knutepunktene blir påkjent med.

Det ble tatt utgangspunkt i en fagverkskran, samt at programmet ble kontrollkjørt på en enklere rammekonstruksjon for å sikre at den fungerte optimalt og som ønsket. På modellen er fagverkskranen beskrevet som en konstruksjon med 20 bjelker som forbinder 16 knutepunkt. De horisontale bjelkene er laget av såkalte ”boksprofil”, mens de vertikale av ”rør-profil.” Som forutsetning for at krana skal være sikker har man funnet ut at materialets flytspenning ligger på 320MPa, men for sikkerhets skyld tolereres bare et bøyespenningsnivå på mellom 30 – 70% av dette. En del av oppgaven var å dimensjonere konstruksjonen når den ble påkjent av åtte ulike laster. Det er både en lineært fordelt last langs hele høyden på kranen, jevnt fordelte laster over et og flere elementer, samt påsatt moment i knutepunkt, påsatt last i punkt og last som virker i en gitt vinkel ut i fra konstruksjonen. Informasjonen om profilene og lastene ble brukt til å dimensjonere profilene til bjelkene på en slik måte at bøyespenningen befant seg innenfor det tillatte området av flytspenningen. Dette ble gjort ved at MATLABscriptet ble iterert med ulike verdier helt til man fant noen som kunne bære konstruksjonen. Det er også lagt inn en sikkerhetsfunksjon som opplyser brukeren dersom bøyespenningen kommer over det tillatte nivået, i tilfelle programmet skal gjenbrukes på en annen konstruksjon i fremtiden. Tilslutt endte konstruksjonen opp med et massivt boksprofil med høyde 200mm og bredde 100mm, mens rørprofilet hadde en diameter på 200mm og en tykkelse på 100mm.

For å få et bedre overblikk over hva dette faktisk vil si for konstruksjonen vil man i tillegg til programfilene vedlagt også finne moment, skjær og normalkraftdiagram for fagverkskranen, samt kontroller av diagrammene laget i modeleringsprogrammet Nauticus 3D – Beam. I tillegg til kommentarene i programfilene er også hver av underfunksjonene i scriptet forklart i rapporten.

Alt i alt vil vi påstå at programmet vårt fungerer rimelig bra, sett bort i fra midtmomentene for bjelkene som er påkjent av den lineært fordelte lasten.

# Innholdsfortegnelse

Innhold

[Forord i](#_Toc402803685)

[Sammendrag ii](#_Toc402803686)

[Innholdsfortegnelse iii](#_Toc402803687)

[Figuroversikt iv](#_Toc402803688)

[Tabelloversikt iv](#_Toc402803689)

[Vedleggsoversikt v](#_Toc402803690)

[1. Innledning 1](#_Toc402803691)

[2 Matrisemetoden 4](#_Toc402803692)

[3 Fremgangsmåte ved rammeanalyse 5](#_Toc402803693)

[3.1 Inputfilen 5](#_Toc402803694)

[3.2 lesinput.m 6](#_Toc402803695)

[3.3 lengder.m 6](#_Toc402803696)

[3.4 annet\_arealmoment.m 6](#_Toc402803697)

[3.5 boyestivhet.m 6](#_Toc402803698)

[3.6 fastmoment.m 6](#_Toc402803699)

[3.7 lastvektor.m 6](#_Toc402803700)

[3.8 Systemstivhetsmatrise.m 7](#_Toc402803701)

[3.9 bc.m 7](#_Toc402803702)

[3.10 rot 7](#_Toc402803703)

[3.11 endeM. 7](#_Toc402803704)

[3.12 midtmoment.m 8](#_Toc402803705)

[3.13 boyespenning.m 8](#_Toc402803706)

[3.14 sjekkspenning.m 8](#_Toc402803707)

[3.15 skjearkrefter.m 8](#_Toc402803708)

[4 Resultater 9](#_Toc402803709)

[4.1 Oppgave c: Bøyemoment ved enhetslastmetoden 9](#_Toc402803710)

[4.2 Dimensjonering 11](#_Toc402803711)

[4.3 Diagrammer 13](#_Toc402803712)

[4.3.1 Momentdiagram 13](#_Toc402803713)

[4.3.2 Skjærkraftdiagram 15](#_Toc402803714)

[4.3.3 Aksialkraftdiagram 16](#_Toc402803715)

[5 Feilkilder og konklusjon 17](#_Toc402803716)

[6 Kilder 18](#_Toc402803717)

[Vedlegg II](#_Toc402803718)

# Figuroversikt

[Figur 1: Viser hele plattformen 1](#_Toc402803826)

[Figur 2: Konstruksjonen som skal analyseres 1](#_Toc402803827)

[Figur 3: Viser konstruksjonen med laster og lengder 2](#_Toc402803828)

[Figur 4: Viser konstruksjon i oppgave c 9](file:///C:\Users\Maria\Documents\NTNU\Rapport%20for%20MATLABprosjekt.docx#_Toc402803829)

[Figur 5: Momentdiagram 14](#_Toc402803830)

[Figur 6: Skjærkraftdiagram 15](#_Toc402803831)

[Figur 7: Normalkraftdiagram 16](#_Toc402803832)

# Tabelloversikt

[Tabell 1: Viser verdier til laster og lengder 2](#_Toc402793982)

[Tabell 2: Verdier brukt i utregning: 9](#_Toc402793983)

[Tabell 3: Viser resultater ved enhetslastmetoden 9](#_Toc402793984)

[Tabell 4: Viser momentene til konstruksjonen ved matlab 9](#_Toc402793985)

[Tabell 5: Viser bøyespenninger i 105 kPa 10](#_Toc402793986)

[Tabell 6: Viser dimensjoner for boksprofil 10](#_Toc402793987)

[Tabell 7: Viser dimensjoner for rørprofil 11](#_Toc402793988)

[Tabell 8: Viser momentfordeling for elementene 11](#_Toc402793989)

# Vedleggsoversikt

[Vedlegg 1: Diskretisering av konstruksjonen II](#_Toc402804656)

[Vedlegg 2: Tabell ved diskretisering III](#_Toc402804657)

[Vedlegg 3: input.txt (Inputfilen) IV](#_Toc402804658)

[Vedlegg 4: Rammeanalyse VI](#_Toc402804659)

[Vedlegg 5: lesinput.m VIII](#_Toc402804660)

[Vedlegg 6: lengder.m X](#_Toc402804661)

[Vedlegg 7: annet\_arealmoment.m XI](#_Toc402804662)

[Vedlegg 8: fastmoment.m XII](#_Toc402804663)

[Vedlegg 9: lastvektor.m XIII](#_Toc402804664)

[Vedlegg 10: Systemstivhetsmatrise.m XIV](#_Toc402804665)

[Vedlegg 11: bc.m XV](#_Toc402804666)

[Vedlegg 12: endeM.m XVI](#_Toc402804667)

[Vedlegg 13: midtmoment.m XVII](#_Toc402804668)

[Vedlegg 14: boyespenning.m XIX](#_Toc402804669)

[Vedlegg 15: sjekkspenning.m XX](#_Toc402804670)

[Vedlegg 16: skjearkrefter XXI](#_Toc402804671)

[Vedlegg 17: boyestivhet.m XXIII](#_Toc402804672)

[Vedlegg 18: output.m XXIV](#_Toc402804673)

[Vedlegg 19: Oppgave c del 1 XXV](#_Toc402804674)

[Vedlegg 20: Oppgave c del 2 XXVI](#_Toc402804675)

[Vedlegg 21: Inputfil til oppgave C XXVII](#_Toc402804676)

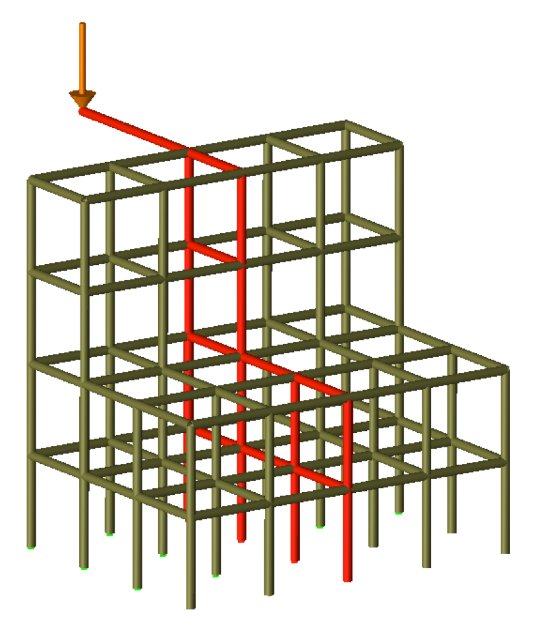
[Vedlegg 22: Momentdiagram fra Nauticus 3D Beam XXVIII](#_Toc402804677)

[Vedlegg 23: Skjærkraftdiagram fra Nauticus 3D Beam XXIX](#_Toc402804678)

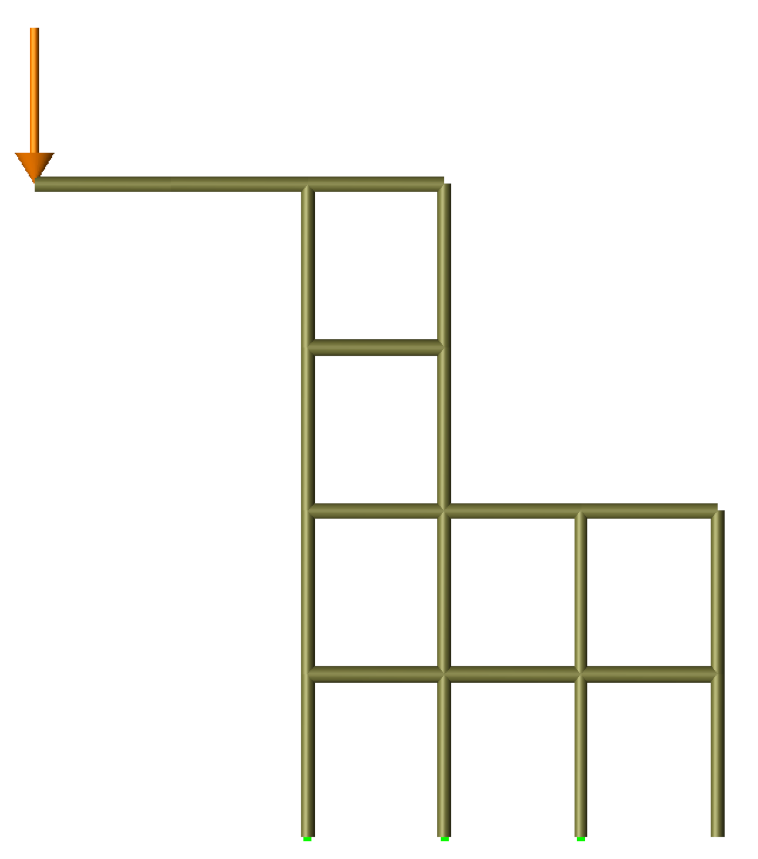
# 1. Innledning

I dette prosjektet er oppgaven å lage et program i MATLAB som kan analysere en del av en plattform, som vist i figur 1. Det skal også fungere for et uvisst antall elementer og knutepunkter, altså holder det ikke å bare kunne analysere gitte ramme, men også andre todimensjonale konstruksjoner. Det er matrisemetoden som skal bli brukt i dette arbeidet, og i det ønskede resultatet vil vi finne endemomentene på elementene, samt midtmoment. Hvis det er en punktlast, ønsker vi også å finne momentet direkte under den.

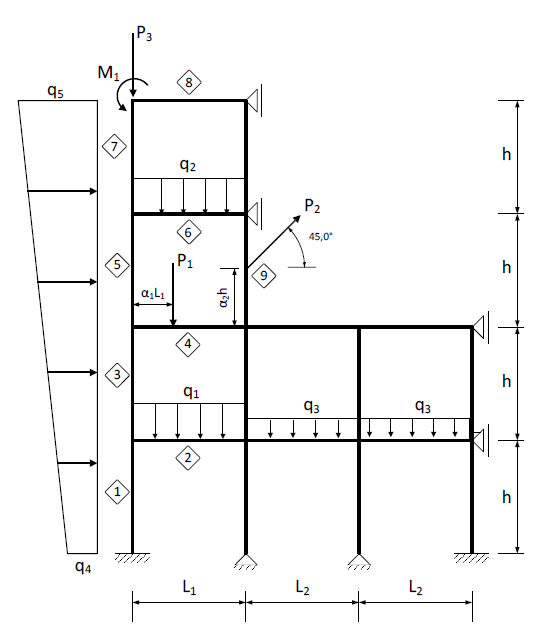
Oppgaven vår baserer seg på en analyse av den rammen vi finner i figur 2, som da er den delen av plattformen i figur 1 som får størst belastning.

****

Figur 1: Viser hele plattformen

****

Figur 2: Konstruksjonen som skal analyseres



Figur 3: Viser konstruksjonen med laster og lengder

Vi skal bruke figur 3 i dimensjoneringsprosessen, hvor de viste lastene og lengdene gis ut i fra tabell 1. Enkelte av tallene fra denne tabellen er gitt ut i fra de to siste sifrene i kandidatnummer 10050. I tillegg til dette vet vi at E-modulen for konstruksjonen er 210 GPa og at flytspenningen er 320 GPa.

#### Tabell 1: Viser verdier til laster og lengder

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **P1 [kN]** | **P2 [kN]** | **P3 [kN]** | **L1 [m]** | **L2[m]** | **h[m]** | **M1[kNm]** |
| 80 | 40 | 60 | 18 | 20 | 14 | 340 |
| **q1[kN/m]** | **q2[kN/m]** | **q3[kN/m]** | **q4[kN/m]** | **q5[kN/m]** | **alpha1** | **alpha2** |
| 7 | 8 | 6 | 10 | 16 | 0,4 | 0,5 |

Vår diskretisering av konstruksjonen kan sees i vedlegg 1. Der finner man elementnummer og knutepunktsnummer.

Ved å se på vedlegg 2 kan man se sammenhengen mellom elementenes lokale ender og hvilke knutepunkter de samsvarer med i en tabell.

Ved bruk av denne metoden er det viktig å merke seg at det er mange deformasjoner som ikke tas hensyn til, da matrisemetoden kun tenker på rotasjonene.

# 2 Matrisemetoden

## 2.1 Diskretisering:

For å kunne ta i bruk matrisemetoden, må vi dele konstruksjonen opp i elementer og knutepunkt. Dette gjør også at vi har bedre oversikt over enkeltkomponentene og kreftene som virker på dem. Inndelingen er som vist i vedlegg 2. Her ser vi kun på bøyedeformasjoner i konstruksjonen. Aksial- og skjærkrefter tas kun hensyn til ved rand. Knutepunktenes forskyvninger og rotasjoner blir en del av systemets frihetsgrader.

## 2.2 Elementanalyse

## 2.3 Systemstivhetsmatrise

## 2.4 Lastvektor

## 2.5 Ligningsløsning

## 2.6 Lokale forskyvninger

## 2.7 Elementkrefter

# 3 Fremgangsmåte ved rammeanalyse

Under presenteres alle funksjonene vi har brukt ved rammeanalysen. Programmet vårt består av en rekke forskjellige underfunksjoner, 13 i alt. I tillegg til disse har vi også laget en funksjon som sjekker at bøyespenningsnivået er under 70% av flytspenningen og en funksjon som regner ut skjærkrefter, men disse er ikke med i selve rammeanalyseprogrammet. Alle disse funksjonene er å finne som vedlegg, og den totale rammeanalysen vil være i å finne i vedlegg 4.

## 3.1 Inputfilen

Inputfilen er den filen som har all informasjonen man trenger for å analysere konstruksjonen. Den inputfilen vi har brukt ved rammeanalysen finner man i vedlegg 3. Den inneholder altså informasjon om diskretiseringen av konstruksjonen, alle laster og tverrsnittsdimensjoner. Den er firedelt, med en del om knutepunkt, en om elementer, en om tverrsnittstyper og en om laster som virker.

Det første tallet i inputfilen vil være antall knutepunkt. Etter dette følger en matrise som er npunktx3 stor. Her vil kolonne 1 være x-koordinat, kolonne 2 være y-koordinat og kolonne 3 vil fortelle om det er fast innspent eller ikke, hvor da 1=fast innspent og 0=rotasjonsfri.

Deretter skriver man inn hvor mange bjelkeelement det er. Disse puttes inn i inputfilen som en nelemX4 matrise. Kolonne 1 vil være knutepunktnummer for lokal ende 1, kolonne 2 vil være knutepunktnummer for lokal ende 2, kolonne 3 gir oss E-modulen til elementet og kolonne 4 forteller oss hvilken tverrsnittstype det er. Programmet er laget for to typer tverrsnitt, så her er 1=boksprofil og 2=rørprofil.

Det neste som tas inn er antallet profiler. Programmet er som sagt laget for bare to. Informasjonen om disse skrives som en nprofilx5 matrise. For begge profiltypene vil kolonne 1 fortelle om det er boks eller rørprofil det er snakk om. For boksprofil har vi tatt hensyn til at vi kan ha en hul boks, så da vil kolonne 2 være høyde indre, kolonne 3 være høyde ytre, kolonne 4 være bredde indre og kolonne 5 være bredde ytre. Om det er rørprofil vil kolonne 2 være diameter og kolonne 3 være tykkelse, mens resterende kolonner vil være null.

Til sist tas det inn informasjon om lasten. Da skriver man lasttype i første kolonne, hvor 1=punktlast, 2=jevnt fordelt last, 3=lineært fordelt last og 4=påsatt moment.

Ellers er det viktig å merke seg at i inputfilen skrives alle krefter i kilonewton og lengder i meter, samt E-modul i kilopascal.

## 3.2 lesinput.m

Først og fremst blant funksjonen har vi den som leser inputfilen. Denne funksjonen fikk vi utdelt halvferdig, og vi har selv valgt hvilke verdier den vil ta inn. Som beskrevet vil den ta inn det som vi har i inputfilen og skriver dette ut som åtte forskjellige matriser. Dette gjør den enkelt og greit ved å åpne filen, lese seg gjennom den og deretter lukke den. Funksjonen kan sees i sin helhet i vedlegg 5.

## 3.3 lengder.m

Denne funksjonen (Vedlegg 6) finner lengdene til elementene basert på hvilke knutepunkt det aktuelle elementet har sin lokale ender i. Ved bruk av Pytagoras regnes så lengden til elementet ut, og presenteres i listeform hvor lengdens plassering i listen tilsier hvilket element den hører til.

## 3.4 annet\_arealmoment.m

Her regnes annet arealmoment ut for våre to profiltyper og plasseres på rett plass. Vi har ikke laget denne funksjonen så den fungerer for mer enn to profiltyper, da det er dette oppgaven har bedt om. Funksjonen tar inn elementinformasjonen, antallet elementer, profilinformasjonen og antallet profiler. Den sjekker så for hvert element hvilket profil det har, og kjører en if-setning som regner ut aktuelt annet arealmoment. Vi får output i listeform, med radnummer lik elementnummer.

I denne koden har vi to for-løkker som plasserer annet arealmoment på rett plass, en under boksprofilutregningen og en under rørprofilutregningen, som ikke strengt tatt er nødvendig, da vi kunne gjort dette med én for-løkke på slutten av funksjonen. Funksjonen er å finne i vedlegg 7.

## 3.5 boyestivhet.m

Funksjonen boyestivhet.m (Vedlegg 17) regner ut EI/L for alle elementene, som vi kommer til å trenge videre til systemstivhetsmatrisen. Den henter inn lengdene, antall elementer, elementinformasjon og annet arealmoment og bruker dette til å enkelt regne ut EI/L ved å bruke en for-løkke som går gjennom alle elementene. Også her vil plassering tilsvare hvilket element tallet tilhører.

## 3.6 fastmoment.m

Her regnes fastinnspenningsmoment ut i henhold til tabell 8.3 fra kompendiet til TMR4167. Det blir da sjekket hva slags type last det er, og lastens informasjon hentes så ut.

## 

## 3.7 lastvektor.m

Lastvektoren er en nx1-matrise der hver rad tilsvarer fastinnspenningsmomentet til et knutepunkt der ytre momentet som virker i punktet er lagt til, og fastinnspenningsmomentet til knutepunktet ved siden av, er trukket fra.

## 3.8 Systemstivhetsmatrise.m

Systemstivhetsmatrisen (Vedlegg 10) lagrer all informasjonen om styrken i elementene og knutepunktene for konstruksjonen. Den tar utgangspunkt i formel 8.183 fra kompendiet og bruker den sammen med informasjon fra inputfilen og funksjonen ”boyestivhet.m” til å beregne elementstivhetsmatrisen for hvert av elementene. I samme loopen bruker den informasjonen om hvilke globale knutepunkt det aktuelle elementet er knyttet til, til å finne ut hvor informasjonen for det lokale elementet hører hjemme i den store systemstivhetsmatrisen for hele systemet (Matrix of Nodal Point Correspondence). Etter å ha funnet dette blir verdien til hvert element i elementstivhetsmatrisen addert sammen med de allerede eksisterende verdiene i den korresponderende plassen i systemstivhetsmatrisen.

## 3.9 bc.m

Her etableres randbetingelsene og brukes til å avgjøre hvilke deler av systemstivhetsmatrisen som skal nulles ut. Denne sjekker om elementet er fast innspent, og om det er det så vil den nuller ut korresponderende rader og kolonner, og sette diagonalelementet lik et vilkårlig tall, som vi har valgt til å være 1.

## 3.10 rot

Dette er selve løsningen av ligningssystemet. Her brukes systemstivhetsmatrisen og lastvektoren til å beregne rotasjonen som vil forekomme i det aktuelle punktet på konstruksjonen. Den tar utgangspunkt i likning 8.190 fra kompendiet. Kr = R, eller på vårt MATLABspråk, Kn\*rot = Bn. Der ”Kn (K)” er systemstivhetsmatrisen, ”rot (R)” er rotasjonen i punktet og ”r (Bn)” er lastvektoren. Det er rotasjonen vi er ute etter å bestemme og får å gjøre dette må vi invertere systemstivhetsmatrisen og matrisemultiplisere med lastvektoren Bn for å bestemme rotasjonen. I MATLAB er dette ekvivalent med å bruke ”matrisedivisjon”, dvs. kommandoen Kn \ Bn. Dette blir akkurat det samme som å invertere osv, så vi har istede brukt dette i vårt MATLABscript.

NB: Rot er ikke en egen funksjon, men en kommando i scriptet ”rammeanalyse” som vi likevel mente trengte en egen kommentar.

## 3.11 endeM.

Endemomentfunksjonen tar inn elementinformasjon, stivhet, rotasjoner og fastinnspenningsmomentene. Den lager så en standard k-matrise og bruker denne ved å gangen den med stivheten og rotasjonen, hvorpå fastinnspenningsmomentene adderes til dette. Den vil så gi oss en matrise som inneholder endemomentene for alle elementene sine lokale ender.

## 3.12 midtmoment.m

Denne funksjonen (Vedlegg 13) tar inn endemomentene, lastinformasjonen, antallet laster, antallet elementer og lengden på elementene. Den bruker så en for-løkke til å gå gjennom alle lastene og sjekke hva slags type last det er, hvorpå den bruker formler til å regne ut midtmomentent, samt plusse på endemomentbidraget.

Her er det brukt formler fra *Formelsamling mekanikk* av Irgens til å finne momentene, som vi finner på midten for jevnt fordelt last, under lasten for punktlast og på L\*sqrt(3) for lineært fordelt last. Sånn sett er funksjonens navn noe misvisende, men de momentene vi har finner er de som er viktigst uavhengig om alle er midtmomenter eller ikke.

## 3.13 boyespenning.m

Vi besluttet at en funksjon som regnet ut bøyespenningen var nødvendig, så denne regner ut bøyespenning for hvert element på det mest belastede stedene. Den tar inn midtmomentene, endemomentene, elementinformasjon, annet arealmoment og elementlengder. Så går den gjennom alle elementene og renger ut bøyespenningen for hver ende og midten ved bruk av formelen

Dette skrives så ut i en matrise med tre kolonner og like mange rader som det er elementer.

## 3.14 sjekkspenning.m

Sjekkspenning.m tar inn spenning, flytspenning og antallet elementer. Flytspenningen tas inn i kPa og må skrives inn manuelt, da denne ikke ligger i inputfilen. Deretter sjekker funksjonen enkelt og greit om spenningen overstiger 70% av flytspenningen for noen av elementene. Gjør det det, skriver «Ikke okei» ut, mens om den ikke gjør det skrives «WOOHOO» ut. Denne funksjonen gjorde at arbeidet gikk fort i dimensjoneringsprosessen.

## 

## 3.15 skjearkrefter.m

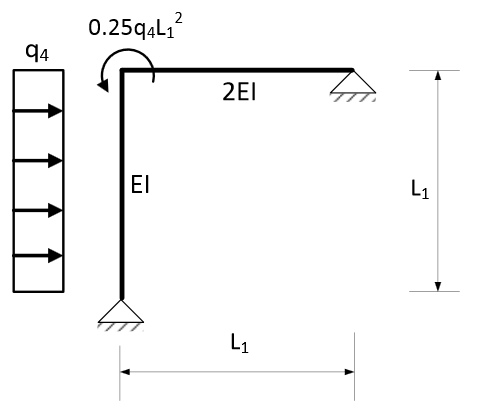
Ved tegning av skjærkraftdiagram fant vi ut at det kunne være nyttig med en egen funksjon som kunne gjøre disse beregningene for oss.   
Funksjonen går gjennom alle elementene og finner lengdene, hvorpå den regner ut skjærkreftene basert på endemomentene. Deretter sjekker den hvilke laster som virker på elementet. Når den avgjør hvilken last det er, går den om opplagerkreftene for elementet og regner derfra ut skjærkreftene og summerer dette sammen med skjærbidraget fra endemomentene.

# 4 Resultater

## 4.1 Oppgave c: Bøyemoment ved enhetslastmetoden

I oppgave C skulle man analysere en konstruksjon, som vist i figur 4, ved hjelp av MATLABprogrammet og ved hjelp av enhetslastmetoden. Poenget var å se om disse resultatene stemte overens.

Som en del av oppgaven skal vi regne på en enkel konstruksjon (se figur 4) ved hjelp av enhetslastmetoden, og finne bøyemomenter ved endene ved punktlast og utbøying på midten der det er jevnt fordelt last. Utregningen er vist i vedlegg 18 og verdiene brukt i utregningen i tabell 2.



Figur 4: Viser konstruksjon i oppgave c

#### Tabell 2: Verdier brukt i utregning:

|  |  |  |  |
| --- | --- | --- | --- |
| **q4 [kN/m]** | **L1 [m]** | **E [kPa]** | **I [m4]** |
| 10 | 18 | 1 | 1 |

I tabell 3 presenteres da resultatene våre med enhetslastmetoden.

#### Tabell 3: Viser resultater ved enhetslastmetoden

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Lokal ende 1 [kNm]** | **Lokal ende 2 [kNm]** | **Midten [kNm]** |
| **1** | 0 | 0 | 405 |
| **2** | 810 | 0 | 0 |

Ved sammenligning i med resultatene vi får ved å gjøre samme utregning i MATLAB (inputfil til MATLABprogram i vedlegg 21), som kan sees i tabell 4. Vi får da samme resultat for bøyemoment og utbøying ved midten, bortsett bra at fortegnet til endemomentet er annerledes i MATLAB. Dette er bare fordi vi har definert positivt moment på en annen måte i MATLAB, da vi i MATLAB har sagt at moment på oversiden er negativt og moment på undersiden er positivt, med andre ord stemmer resultatene overens.

#### Tabell 4: Viser momentene til konstruksjonen ved MATLAB

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Lokal ende 1 [kNm]** | **Lokal ende 2 [kNm]** | **Midtmoment [kNm]** |
| **1** | 0 | 0 | 450 |
| **2** | -810 | 0 | 0 |

Ut i fra dette har vi konkludert med at programmet vårt fungerer godt, da det for en konstruksjon med jevnt fordelt last, punktlast og påsatt moment gir oss rette verdier.

## 4.2 Dimensjonering

Etter å ha fått bekreftet at programmet kjørte og ga ut fornuftige verdier, var det på tide å finne ut hva slags tverrsnittsdimensjoner konstruksjonen vil ha bruk for. Oppgitt i oppgaven var det at de vertikale bjelkene skulle ha boksprofil og de horisontale skulle ha rørprofil.

Vi fant bøyespenningene ved hjelp av funksjonen boyespenning.m (Se vedlegg 14), disse kan sees i tabell 5. Vi brukte deretter sjekkspenning.m (Se vedlegg 15) for å sjekke at spenningene ikke var for høye.

Dimensjoneringsprosessen besto av mange iterasjoner, og til slutt fant vi noen tverrsnittsdimensjoner som sørger for at ingen av bøyspenningene overstiger 70% av flytspenningen vår, som var oppgitt som 320 MPa.

#### Tabell 5: Viser bøyespenninger i 105 kPa

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Spenning i ende 1** | **Spenning i ende 2** | **Spenning på midten** |
| **1** | -0.3108 | 0,6675 | 1,0214 |
| **2** | -1,78 | 1,95 | 0,97 |
| **3** | -0,23 | 0,67 | 1,20 |
| **4** | -2,05 | 1,39 | 1,67 |
| **5** | -0,16 | 1,24 | 1,93 |
| **6** | -1,95 | 2,14 | 1,20 |
| **7** | -0,76 | -0,67 | 0,86 |
| **8** | -0,66 | -0,25 | 0 |
| **9** | 0,06 | -0,27 | 0 |
| **10** | -0,27 | -0,33 | -0,21 |
| **11** | -0,01 | -0,01 | 0 |
| **12** | -0,01 | 0 | 0 |
| **13** | -2,00 | 2,01 | 1,00 |
| **14** | 0,01 | 0 | 0 |
| **15** | 0 | 0,01 | 0 |
| **16** | -2,05 | 1,90 | 1,03 |
| **17** | 0 | 0 | 0 |
| **18** | 0.02 | 0,05 | 0 |
| **19** | -0,01 | -0,20 | 0 |
| **20** | -0,27 | -0,13 | 0 |

Vår høyeste bøyespenning befinner seg på element nummer 6. Bjelke 6 er en horisontal bjelke og dermed et boksprofil. Spenningen i lokal ende 2 er på 214 kPa, noe som tilsvarer 66,6% av flytspenningen. Dette tilfredsstiller altså ønsket om at ingen av bøyespenningene skal overstige 70% av flytspenningen.

#### Tabell 6: Viser dimensjoner for boksprofil

|  |  |  |  |
| --- | --- | --- | --- |
| Ytre høyde | Indre høyde | Ytre bredde | Indre bredde |
| 200 [mm] | 0 [mm] | 150 [mm] | 0 [mm] |

Her ser man altså at vi har endt opp med et solid boksprofil, da vi fant ut at hulrom i midten ikke var nødvendig. Vi vet at dette i virkeligheten ville blitt veldig tungt, og egenvekten fra bjelken ville begynne å belaste konstruksjonen også. Vi har ikke tatt hensyn til egenvekt i disse beregningene, men i ettertid skal det sies at et solid boksprofil neppe vil være noen god løsning selv om våre utregninger nå sier at det vil gå bra. Dette hadde man måtte undersøke nærmere dersom konstruksjonen faktisk skulle bygges.

#### Tabell 7: Viser dimensjoner for rørprofil

|  |  |
| --- | --- |
| **Diameter** | **Tykkelse** |
| 350 [mm] | 100 [mm] |

Rørprofilet vårt er hult og vi syns dimensjonene vi endte opp med her er relativt gode.

## 4.3 Diagrammer

### 4.3.1 Momentdiagram

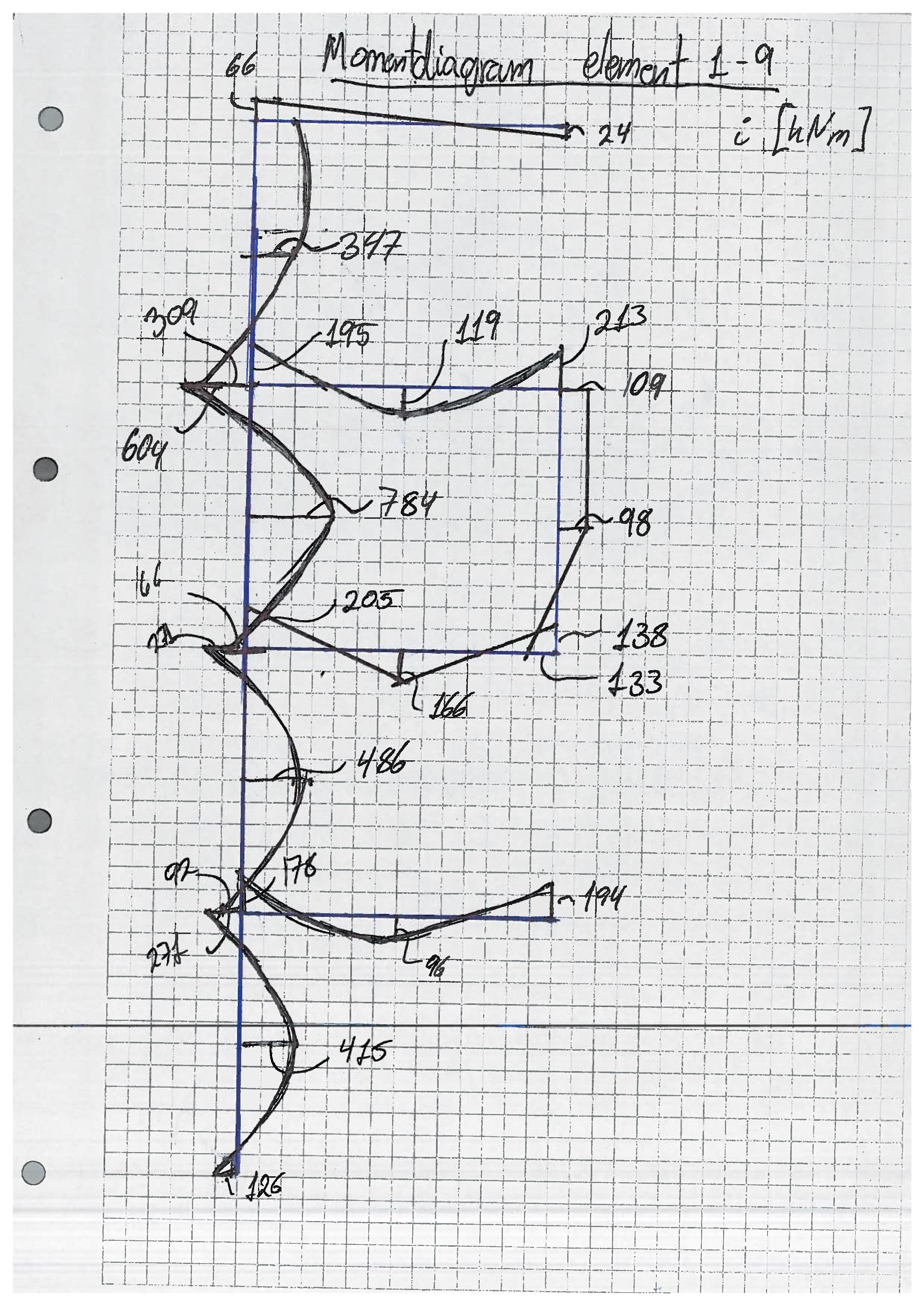
Ved tegning av momentdiagram brukte vi verdiene vi fikk fra MATLABprogrammet, som er vist i tabell 8.

#### Tabell 8: Viser momentfordeling for elementene

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Endemoment 1 [kNm]** | **Endemoment 2**  **[kNm]** | **Midtmoment**  **[kNm]** |
| **1** | -126 | 271 | 415 |
| **2** | -179 | 194 | 97 |
| **3** | -93 | 272 | 487 |
| **4** | -205 | 138 | 167 |
| **5** | -66 | 505 | 784 |
| **6** | -195 | 213 | 119 |
| **7** | -309 | -273 | 348 |
| **8** | -66 | -24 | 0 |
| **9** | 25 | -104 | 0 |
| **10** | -109 | -133 | -86 |
| **11** | -0,8 | -0,6 | 0 |
| **12** | -4 | 1 | 0 |
| **13** | -199 | 201 | 100 |
| **14** | 3 | 0 | 0 |
| **15** | 0 | 3 | 0 |
| **16** | -205 | 190 | 103 |
| **17** | 2 | -1 | 0 |
| **18** | 2 | 4 | 0 |
| **19** | -5 | -83 | 0 |
| **20** | -107 | -54 | 0 |

Ut i fra dette ser vi at det elementet med den største momentpåkjenningen er nummer 5, med et midtmoment på 784 kN. Dette er mistenkelig høyt, noe alle midtmomentene for elementene som er påkjent av lineært fordelt last er. Vi tror derfor at vi har en feil i midtmomentfunksjonen for dette lasttilfellet, men vi har dessverre ikke vært i stand til å lokalisere feilen. Momentene har dog korrekt fortegn, da positivt moment vil si at det er på det som vil betraktes som bjelkens underside, mens negativt moment vil være på bjelkens overside for midtmomentene. Underside og overside defineres ut i fra lokale ender, når man har lokal ende 1 til venstre og lokal ende 2 til høyre.  
Endemomentene er definert annerledes på grunn av fastinnspenningsmomentene, så der vil negativ verdi i lokal ende 1 være på oversiden og positiv verdi i lokal ende 2 være på oversiden.

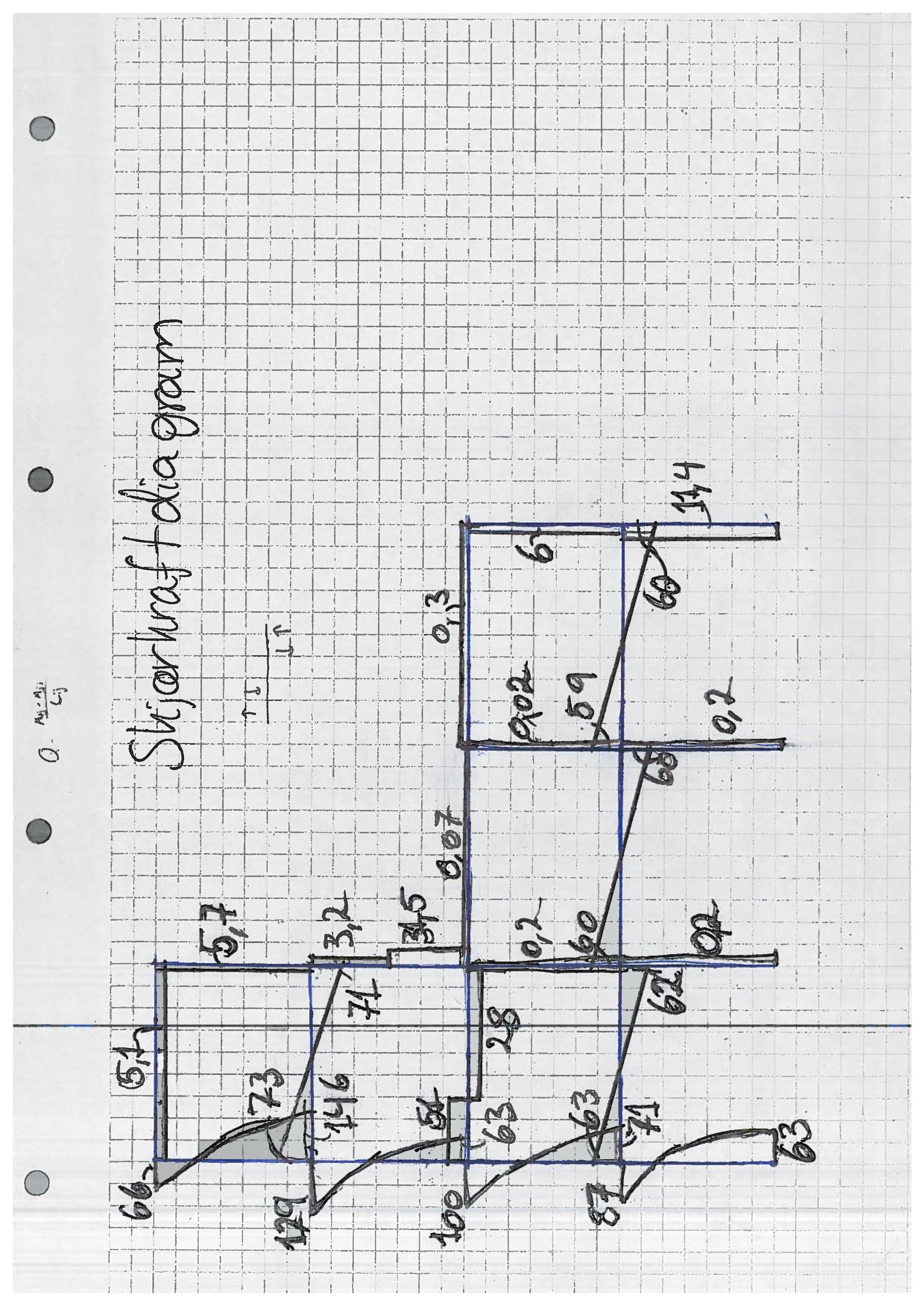
Momentdiagrammet vårt ble ut i fra dette slik man kan se i figur 5 på neste side.



Figur 5: Momentdiagram

### 4.3.2 Skjærkraftdiagram

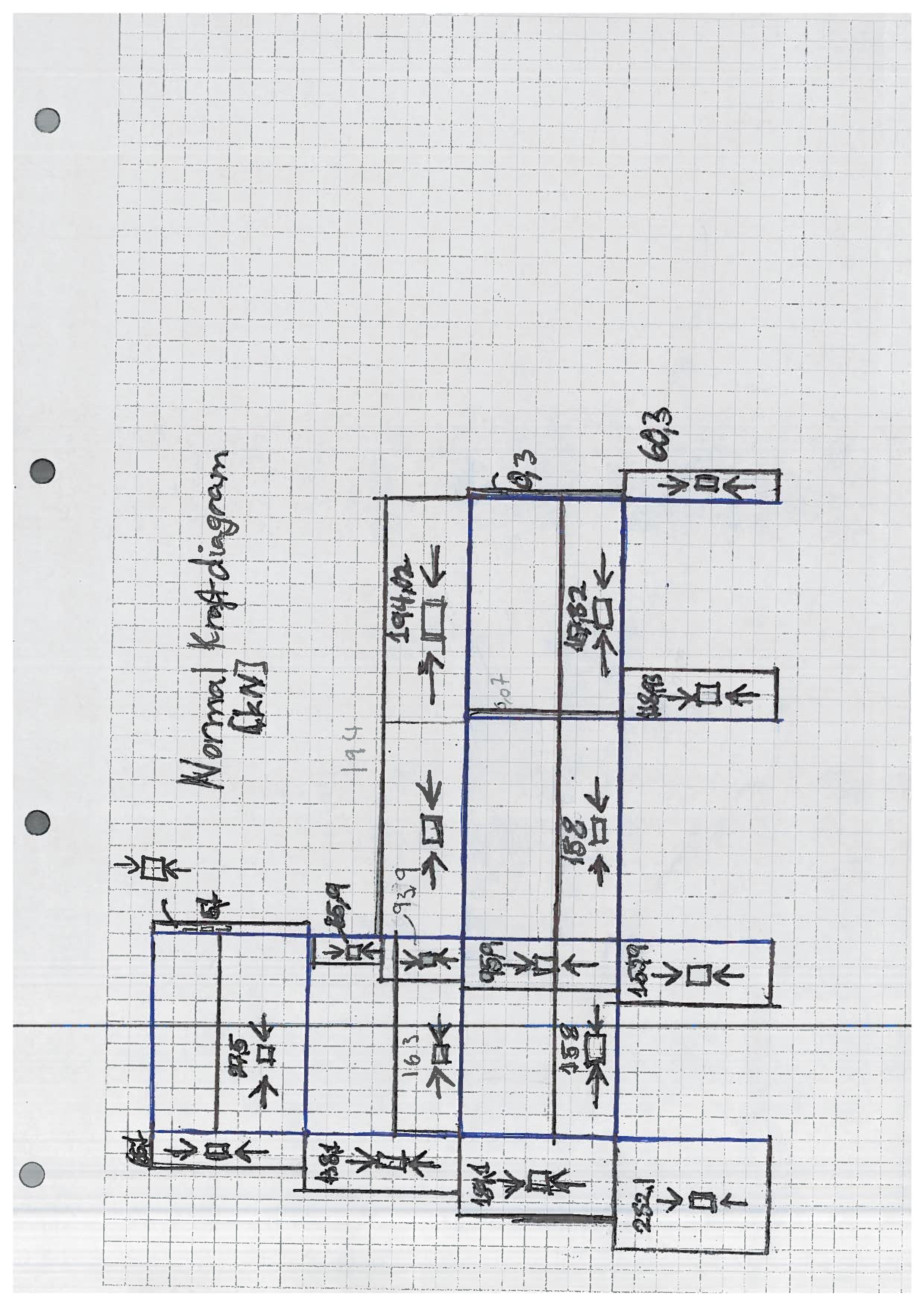
Skjærkraftdiagrammet vårt er tegnet ut i fra verdiene som funksjonen skjearkrefter.m (Se vedlegg 16). Funksjonen er basert på verdiene for momentene, ergo vil verdiene i skjærkraftdiagrammet ha samme feilmargin som momentdiagrammet.



Figur 6: Skjærkraftdiagram

### 4.3.3 Aksialkraftdiagram

Normalkraftdiagrammet vårt er basert på skjærkraftverdiene og tegnet ut i fra dem.



Figur 7: Normalkraftdiagram

# 5 Feilkilder og konklusjon

Programmet vårt er ikke perfekt. Resultatene vi får for midtmoment på de bjelkene som er påkjent av lineært fordelt last er ikke helt som ønsket, noe som påvirker bøyespenningene og dermed også tverrsnittsvalg.  
  
Grunnen til at disse midtmomentene har blitt litt uventet store tror vi ligger i at funksjonen vår ikke regner midtmomentet for lineært fordelte laster nøye nok. Dette kunne vært fikset ved å for eksempel sørge for at moment faktisk blir regnet på midten, ikke litt over den slik vi har det nå. Vi kunne også ha fått sjekket ting nøyere i Nauticus 3D Beam, da dette ville hjulpet oss til å finne feilene. Vi slet dessverre litt med å få programmet til å fungere slik vi hadde tenkt, samt at det tar hensyn til for eksempel egenvekt, og dermed fikk vi ikke entydige svar på momentverdiene. Vi fikk dog korrekt form på momentdiagrammet vårt med tanke på Nauticus 3D Beam sitt diagram (Se vedlegg 22), noe vi ser på som svært positivt. Vi har ut i fra det konkludert med at momentverdiene ikke er helt optimale, mens selve diagrammets form tyder på at programmet vårt gjør mye rett.

Vårt boksprofil er som nevnt ikke hult, noe det nok burde vært i den virkelige verden. Programmet vårt mente dette skulle gå fint likevel, noe som er på grunn av at vi ikke tar hensyn til konstruksjonens egenvekt. Hadde vi gjort det, ville nok programmet ment at dette boksprofilet blir for tungt til at konstruksjonen vil holde seg opp.

Sett bort i fra de tingene vi har utpekt som usikre eller feil, så er vi godt fornøyde med hvordan programmet kjører og de verdiene vi får ut. Vi føler vi har fått en god innsikt i hvordan rammeanalyse ved hjelp av dataprogrammer fungerer, og det har blitt tydelig for oss hvor mye lettere det er enn å gjøre rammeanalysen for hånd.

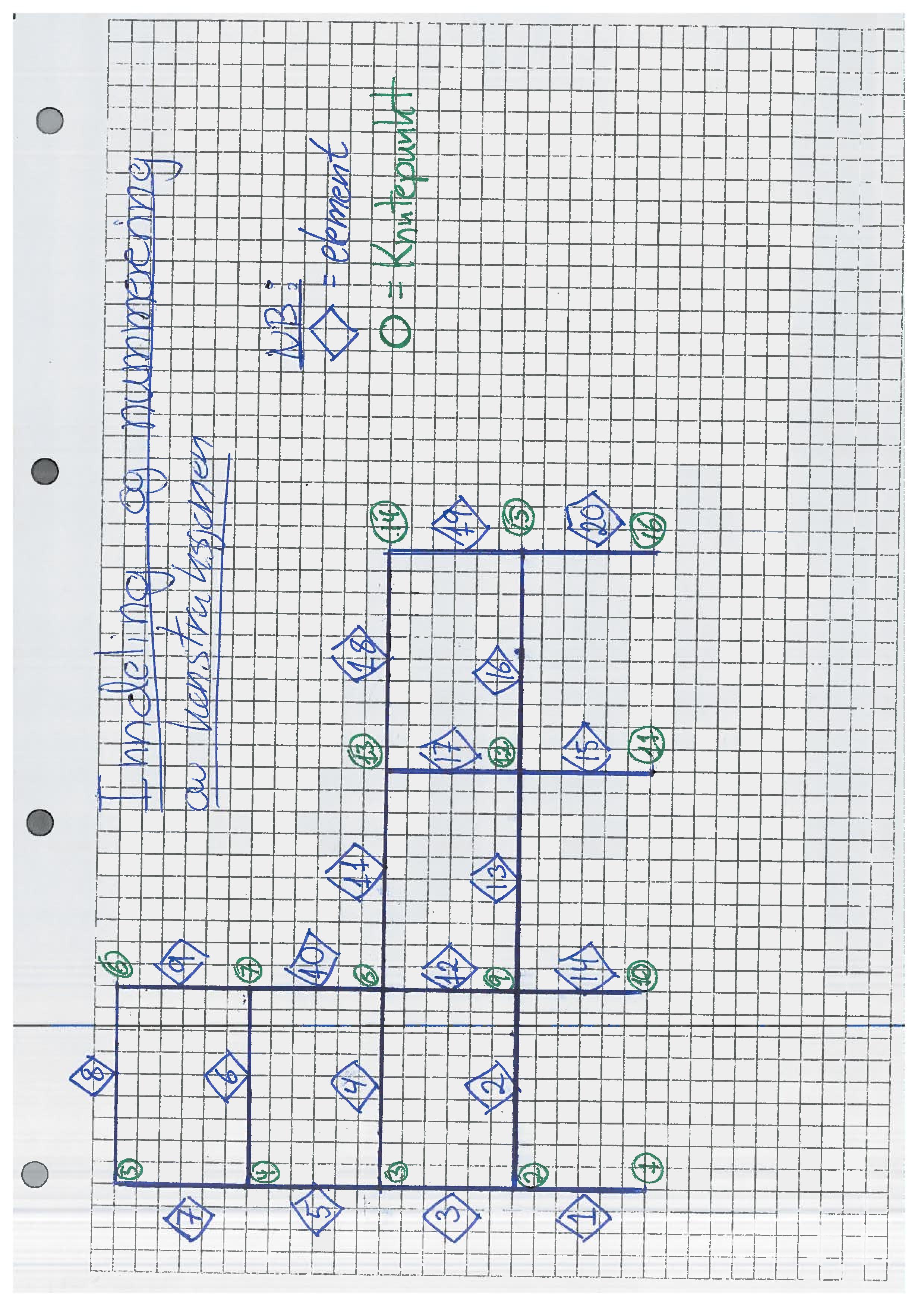
# 6 Kilder

Amdahl, Jørgen: *Kompendium i TMR4167 Marin Teknikk 2, Del 1: Konstruksjonsanalyse.* Trondheim, institutt for marin teknikk; 2010

Irgens, F: *Formelsamling mekanikk*   
Tapir akademisk forlag; 3 utgave 1999, 5 opplag 2010

# Vedlegg

##### Vedlegg 1: Diskretisering av konstruksjonen



##### Vedlegg 2: Tabell ved diskretisering

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| El.nr | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Kp.1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 6 | 7 | 8 | 8 | 9 | 9 | 11 | 12 | 12 | 13 | 14 | 15 |
| Kp.2 | 2 | 9 | 3 | 8 | 4 | 7 | 5 | 6 | 7 | 8 | 13 | 9 | 12 | 10 | 12 | 15 | 13 | 14 | 15 | 16 |

##### Vedlegg 3: input.txt (Inputfilen)

16

0 0 1

0 14 0

0 28 0

0 42 0

0 56 0

18 56 0

18 42 0

18 28 0

18 14 0

18 0 0

38 0 0

38 14 0

38 28 0

58 28 0

58 14 0

58 0 1

20

1 2 210e6 2

2 9 210e6 1

2 3 210e6 2

3 8 210e6 1

3 4 210e6 2

4 7 210e6 1

4 5 210e6 2

5 6 210e6 1

6 7 210e6 2

7 8 210e6 2

8 13 210e6 1

8 9 210e6 2

9 12 210e6 1

9 10 210e6 2

11 12 210e6 2

12 15 210e6 1

12 13 210e6 2

13 14 210e6 1

14 15 210e6 2

15 16 210e6 2

2

1 0 0.2 0 0.15

2 0.35 0.1 0 0

16

1 4 0.4 80 0

1 10 0.5 -40 0.78534

1 8 0 60 0

2 2 1 7 0

2 13 1 6 0

2 16 1 6 0

2 6 1 8 0

3 1 1 10 0

3 1 2 11.5 0

3 3 1 11.5 0

3 3 2 13 0

3 5 1 13 0

3 5 2 14.5 0

3 7 1 14.5 0

3 7 2 16 0

4 5 0 -340 0

##### Vedlegg 4: Rammeanalyse

%{

RAMMEANALYSE: Analyserer en konstruksjon med gitte profiler og laster ved

hjelp av en rekke underfunksjoner. Skriver ut rotasjonene i knutepunkene

og endemomentene i knutepunkene til konstruksjonen

%}

% Sletter alle variabler

clear all

% -----Leser input-data-----

[npunkt, punkt, nelem, elem, nprofil, profil, nlast, last] = lesinput();

% -----Regner lengder til elementene i [m]-----

[elemlengder] = lengder(punkt,elem,nelem);

% -----Regner ut annet areamlmoment for hvert element i [m^4] -----

[I\_elem] = annet\_arealmoment(nelem, elem, profil,nprofil );

% -----Regner stivhet til alle elementene i [kN \* m]-----

[stivhet] = boyestivhet(nelem, elemlengder, elem, I\_elem);

% ------Fastinnspenningsmomentene i [kNm]------

[fim] = fastmoment(nelem,nlast,last,elemlengder);

% ------Setter opp lastvektor i [kNm]-------

[b] = lastvektor(fim,npunkt,nelem,elem,nlast,last);

% ------Setter opp systemstivhetsmatrisen i [kNm]-------

[K] = Systemstivhetsmatrise(nelem,elem,stivhet,npunkt);

% ------Innfoerer randbetingelser-------

[Kn, Bn] = bc(npunkt, punkt, K, b);

% -----Løser ligningssytemet-------

rot = Kn \ Bn ;

% -----Finner endemoment for hvert element [kNm] -------

[endemoment] = endeM(nelem,elem,rot,fim,stivhet);

% -----Finner midtmoment/største moment for hvert element [kNm] -----

[midtmom] = midtmoment(endemoment,nlast,last,nelem,elemlengder);

% -----Finner bøyespenningpå utsatte steder [kN/m^2]=[kPa]-----

[spenning] = boyespenning(I\_elem,elem,nelem,profil,endemoment,midtmom );

%-----Skriver ut rotasjene i de forskjellige punktene og endemomentene for

%elementene -----

output(rot,npunkt,endemoment,nelem)

##### Vedlegg 5: lesinput.m

function [npunkt, punkt, nelem, elem, nprofil, profil, nlast, last]=lesinput()

%{

LESINPUT: Henter inn informasjon fra inputfilen og lagrer det i variable

MATLAB kan bruke og som videre kan brukes til utregnigene for rammen.

NB: i = heltall (integer) %f : desimaltall (flyt-tall)

OUTPUT:

\* npunkt = integer, antall knutepunkter

\* punkt = npunkt x 3 - matrise, lagrer knutepunktenes koordinater og

kode for grensebetingelsene

\* nelem = integer, antall elementer

\* elem = nelem x 4 - matrise, lagrer nummeret til endepunkene til elementene,

samt e-modul og kode for tverrsnittstype.

\* nprofil = integer, antall ulike profiler

\* profil = nprofil x 5 -matrise, all info for hver profiltype, høyde, bredde

osv.

\* nlast = integer, antall laster som virker

\* last = nlast x 5 - matrise, lagrer type last, hvilket element eventuelt knutepunkt

det virker på, med hvilken intensitet og eventuelt med hvilken vinkel i

forhold til elemtet/lokale akser

%}

% Åpner inputfila

filid = fopen('input3.txt','r');

% Leser hvor mange punkt det er

npunkt = fscanf(filid,'%i',[1 1]);

% LESER INN XY-KOORDINATER TIL KNUTEPUNKTENE

% Nodenummer tilsvarer radnummer i "Node-variabel"

% x-koordinat lagres i første kolonne, y-koordinat i 2.kolonne

% Grensebetingelse lagres i kolonne 3, fast innspent=1 og fri rotasjon=0

punkt = fscanf(filid,'%f %f %i',[3 npunkt])';

%Leser hvor mange element det er

nelem = fscanf(filid,'%i',[1 1]);

% Leser konnektivitet: sammenheng elementender og knutepunktnummer. Og EI for elementene

% Elementnummer tilsvarer radnummer i "Elem-variabel"

% Knutepunktnummer for lokal ende 1 lagres i kolonne 1

% Knutepunktnummer for lokal ende 2 lagres i kolonne 2

% E-modul for materiale lagres i kolonne 3

% Tverrsnittstype lagres i kolonne 4, boksprofil=1 og rørprofil=2

elem = fscanf(filid,'%i %i %f %i',[4 nelem])';

% Leser hvor mange profiler det er(Merk at programmet er laget for to)

nprofil = fscanf(filid, '%i',[1,1]);

% Leser tverrsnittsdata. Kolonne 1 = profiltype (1=boks og 2=rør)

% Kolonne 2 er høyde indre/diameter rør

% Kolonne 3 er tykkelse rør/høyde ytre

% Kolonne 4 er bredde indre (vil være 0 for rør)

% Kolonne 5 er bredde ytre (vil være 0 for rør)

profil= fscanf(filid, '%f %f %f %f %f',[5 nprofil])';

% Leser hvor mange laster som virker.

nlast = fscanf(filid,'%i',[1 1]);

% Leser lastdata

% Kolonne 1 representerer hvilken type last det er. 1=punktlast 2=Jevnt

% fordelt, 3=lineært fordelt og 4=påsatt moment.

% Kolonne 2: Hvilket bjelkeelement den virker på(For

% moment=knutepunkt)

% Kolonne 3: Lastens avstand til lokal ende 1 på bjelkeelementet, alpha.

% (Hvis jevn, oppgi 1. Hvis lineær, oppgi 1 hvis max i LE 1 og 2 hvis max i LE 2)

% (For moment=0)

% Kolonne 4: Lastens størrelse i Newton(/M eller M)

% Kolonne 5: Vinkel i grader(=0 for alt annet enn punktlast)

last = fscanf(filid,'%i %f %f %f %f',[5 nlast])';

% LUKKER INPUT-FILEN

fclose(filid);

end

##### Vedlegg 6: lengder.m

function [elemlengder] = lengder(punkt,elem,nelem)

%{

LENGDER: Denne funksjonen finner lengden på elementene basert på

koordinatene til knutepunktene

%}

elemlengder=zeros(nelem,1);

% Beregner elementlengder

for i=1:nelem

dx = punkt(elem(i,1),1) - punkt(elem(i,2),1);

dy = punkt(elem(i,1),2) - punkt(elem(i,2),2);

% Beregner total elementlengde i [m]

elemlengder(i) = (sqrt(dx\*dx + dy\*dy));

end

end

##### Vedlegg 7: annet\_arealmoment.m

function [I\_elem] = annet\_arealmoment(nelem, elem, profil,nprofil )

%{

ANNNET\_AREALMOMENT: Finner annet arealmoment for elementdelen, hvis den er

boks-profil eller rør-profil

Tallene som hentes fra inputfilen vil være meter

OUTPUT: Vektor med et tall per element nedover

%}

% Danner en matrise i ønsket størrelse

I\_elem = zeros(nelem,1);

% Sjekker antall profiler og finner ut om det er boks eller rør.

for i = 1:nprofil

if profil(i,1) == 1 % Hvis boks-profil

Iindre = ((profil(i,2))^3) \* profil(i,4) \* (1/12); % Regner ut det annet arealmoment for indre dimensjoner

Iytre = profil(i,5) \* ((profil(i,3))^3) \* (1/12); % Regner ut for ytre

I\_boks = Iytre - Iindre; % Finner total I ved å ta ytre I minus indre I

for j=1:nelem % Går gjennom alle elementene

if elem(j,4) == 1 % Sjekker om elementet har boksprofil

I\_elem(j) = I\_boks; % Hvis det har det setter I til det for boks

end

end

% Hvis ikke boks-profil, så rørprofil

else % Regner annet arealmoment for rørprofil(da vi bare har to typer profil)

Dytre = profil(i,2); % Finner ytre diameter

Dindre = Dytre-2\*profil(i,3); % Finner indre diameter

I\_ror = ( pi/64)\*(Dytre^4-Dindre^4);% Regner ut I i henhold til formel

for j=1:nelem % Går gjennom alle elementene

if elem(j,4) == 2 % Sjekker om elementet har rørprofil

I\_elem(j) = I\_ror; % Setter i så fall inn I for rør på rett sted

end

end

end

end

end

##### Vedlegg 8: fastmoment.m

function [fim] = fastmoment(nelem, nlast,last, lengder)

%{

FASTMOMENT:

OUTPUT: nelem x 2 - matrise med verdien til fastinnspenningesmomentet til

hvert element i hver ende. Element 3 fastinnspenningsmoment i ende 2 finner

man f.eks i (3,2)

%}

fim = zeros(nelem,2); %Danner matrise med nuller med nelem rader og 2 kolonner(da vi har to ender)

for i = 1:nlast

if last(i,1) == 1 %Vil si at det er punktlast

storrelse = last(i,4) \* cos(last(i,5)); %Finner lastens størrelse

%i [kN], vinkel i radianer

L = lengder(last(i,2)); %Finner lengden på elementet i [m]

Avstand1 = last(i,3) \* L ; %Finner avstand fra lokal ende 1 i [m]

Avstand2 = L - Avstand1 ; %Finner avstand fra lokal ende 2 i [m]

M1 = -(storrelse \* Avstand1 \* (Avstand2)^2)/(L^2); %Bruker formel hentet fra kompendiet

M2 = (storrelse \* Avstand2 \* (Avstand1)^2)/(L^2);

fim(last(i,2),1) = fim(last(i,2),1) + M1; %Plasserer momentene på rett plass

fim(last(i,2),2) = fim(last(i,2),2) + M2;

elseif last(i,1) == 2 %Vil si at det er jevnt fordelt last

storrelse = last(i,4); %Finner lastens størrelse i [kN/m]

L = lengder(last(i,2)); %Finner elementets lengde

M1 = -((1/12)\*storrelse\*(L^2)); %Her er formel fra tabell 8.3 i kompendiet brukt

M2 = -M1;

fim(last(i,2),1) = fim(last(i,2),1)+M1; %Plasserer momentene på rett plass

fim(last(i,2),2) = fim(last(i,2),2)+M2;

elseif last(i,1) == 3 %Vil si at det er lineært fordelt last

storrelse = last(i,4); %Lasten størrelse i [kN/m]

L = lengder(last(i,2));

M1 = -(1/30) \* (L^2) \* storrelse;

M2 = (1/20) \* (L^2) \* storrelse;

if last(i,3) == 1

fim(last(i,2),1) = fim(last(i,2),1) - M2;

fim(last(i,2),2) = fim(last(i,2),2) - M1;

else fim(last(i,2),1) = fim(last(i,2),1) + M1;

fim(last(i,2),2) = fim(last(i,2),2) + M2;

end

end

end

end

##### Vedlegg 9: lastvektor.m

function [b] = lastvektor(fim,npunkt,nelem,elem,nlast,last)

%{

LASTVEKTOR: Beregner momentene fra lastene og fastinnspenningsmomentet i

hver av knutepunkte.

OUTPUT: vektor, med verdier momentene pga. lasten i hvert knutepunkt

%}

b = zeros(npunkt,1);

for j = 1:nelem ; %Går gjennom alle elementene

b(elem(j,1),1) = b(elem(j,1),1)-fim(j,1); %Legger sammen fastinnspenningsmomentetene i knutepunktet

b(elem(j,2),1) = b(elem(j,2),1)-fim(j,2);

end

for k = 1:nlast % Går igjennom lastmatrisen

if last(k,1) == 4 % Finner hvilke knutepunkter hvor det virker moment

knp = last(k,2); % Knutepunkt

strl = last(k,4); % Størrelse moment i knutepunkt i [kNm]

b(knp,1) = b(knp,1)+strl; % Legger til ev. ytre moment som virker i knutepunkter

end

end

end

##### Vedlegg 10: Systemstivhetsmatrise.m

function [K] = Systemstivhetsmatrise(nelem, elem,stivhet, npunkt)

%{

SYSTEMSTIVHETSMATRISE: Regner ut systemstivhetsmatrisen til konstruksjonen

Først blir den lokalse elemtstivhetsmatrisen til hvert element regnet ut,

før den blir plassert på rett plass i den store systemstivhetsmatrisen K.

Output: Matrise K, npunkt x npunkt stor

%}

%lager en tom stivhetsmatrise K med like mange rader og kolonner som

%konstrusjonen har knutepunkt

K = zeros(npunkt);

% Lager den faste matrisen vi må multiplisere med EI/l per element for å

% finne elementstivhetsmatrisene for hvert enkelt element

mat1 = [4 2; 2 4];

%looper gjennom hvert element i konstruksjonen

for n = 1: nelem

% henter frem endepunkene der de lokale frihetsgradene til bjelke n er

lok\_fri\_1 = elem(n,1);

lok\_fri\_2 = elem(n,2);

% Lager tom elementstivhetsmatrise for hvert element

k\_lokal = zeros(2);

% looper gjennom ogfyller inn elementstivhetsmatrisen til hvert element med

% riktig verdi (mest for test)

for i = 1:2;

for j = 1:2;

k\_lokal(i,j) = (stivhet(n)) \* mat1(i,j);

end

end

% fyller inn i den store stivhetsmatrisen K

K(lok\_fri\_1, lok\_fri\_1)= K(lok\_fri\_1, lok\_fri\_1) + k\_lokal(1,1);

K(lok\_fri\_1, lok\_fri\_2)= K(lok\_fri\_1, lok\_fri\_2) + k\_lokal(1,2);

K(lok\_fri\_2, lok\_fri\_1)= K(lok\_fri\_2, lok\_fri\_1) + k\_lokal(2,1);

K(lok\_fri\_2, lok\_fri\_2)= K(lok\_fri\_2, lok\_fri\_2) + k\_lokal(2,2);

end

end

##### Vedlegg 11: bc.m

function [Kn, Bn] = bc\_final (npunkt, punkt, K, b)

%{

BC:Innfører grensebetingelser for knutepunktsrotasjonene. Tar utgangspunkt

i systemstivhetsmatrisen og lastvektoren, og "nuller ut" det som ikke skal

med pga. grensebetingelser

%}

Kn = K; % Setter Kn lik systemstivhetsmatrisen

Bn = b; % Setter Bn lik lastvektoren

randbetingelse = (punkt(:,3)); % Henter ut informasjon om fast innspent eller rotasjonsfri

for i = 1:npunkt

if randbetingelse(i) == 1 % Hvis det er fast innspent

Kn(:,i) = 0; % Nuller ut radene, slik spesifisert i oppgave

Kn(i,:) = 0; % Nuller også ut kolonnene

Kn(i,i) = 1; % Setter diagonalelementet lik et vilkårlig tall

Bn(i) = 0; % Lastvektoren må også nulles ut her, for dette knutepuntket

end

end

end

##### Vedlegg 12: endeM.m

function endemoment = endeM(nelem,elem,rot,fim,stivhet)

%{

ENDEM: Regner ut endemomentet

%}

k = [4 2; 2 4];

endemoment = zeros(nelem,2); % Danner matrise med nuller

for i = 1:nelem

R = [rot(elem(i,1));rot(elem(i,2))];

% Danner en rotasjonsmatrise ut i fra rotasjonene til hvert knutepunkt

K1 = (stivhet(i)) \* k; % Gir oss stivhetsmatrise

mom = [fim(i,1);fim(i,2)]; % Gir oss fastinnspentmomentmatrise

EM = (K1\*R) + mom; % Ganger stivhetsmatrise med rotasjonsmatrisen og plusser på tall fra FIMmatrise

% Dette gir oss da summen av endemomentene i den lokale enden på elementet

endemoment(i,1) = EM(1); % Her plasseres endemomentene på rett plass i endemomentmatrisen

endemoment(i,2) = EM(2);

end

end

##### Vedlegg 13: midtmoment.m

function [midtmom] = midtmoment(endemoment,nlast,last,nelem,elemlengder)

%{

MIDTMOMENT: Her finner vi momentet under punktlastene, momentet midt på bjelker med

jevnt fordelt last, samt største moment på bjelker med lineært fordelt

last

%}

midtmom = zeros(nelem,1); % Danner en matrise med nuller i rett størrelse

for i = 1:nlast

if last(i,1) == 1 % Sjekker om det er punktlast

if last(i,3) > 0

element = last(i,2); % Henter ut hvilket element den virker på

L = elemlengder(element); % Finner elementets lengde

avstand1 = last(i,3) \* L; % Finner avstanden fra lokal ende 1

avstand2 = L-avstand1; % Finner avstanden fra lokal ende 2

F = (cos(last(i,5))) \* last(i,4); % Finner kraften til lasten

midt1 = (F\*avstand1\*avstand2) / L; % Gir oss momentet direkte under den

midt2 = ((endemoment(element,1)\*avstand2) - (endemoment(element,2)\*avstand1)) / L;

% midt2 henter ut momentet under punktlasten ut i fra

% endemomentene

midtmom(element,1) = midt1 + midt2 + midtmom(element,1);

end

% Finner det totale momentet under lasten og setter det på rett sted

% Merk: Dette er ikke midtmomentet, men momentet under der

% punktlasten virker, som jo er det relevante å se på

elseif last(i,1) == 2 % Sjekker om det er jevnt fordelt last

element=last(i,2); % Finner elementnummer

L=elemlengder(element); % Finner lengde på elementet

F=last(i,4); % Finner kraften til lasten

midt1= ((F\*(L^2)) / 8); % Regner ut midtmoment i henhold til Irgens formler

midt2= ((endemoment(element,1))-(endemoment(element,2))) / 2;

% Finner midtmoment ut i fra endemomentene

midtmom(element,1) = midt1 + midt2 + midtmom(element,1);

% Finner totatlt midtmoment og plasserer det på rett sted

elseif last(i,1) == 3 % Sjekker om det er lineært fordelt last

element=last(i,2); % Finner elementnummer

L=elemlengder(element); % Finner lengde på elementet

F=last(i,4); % Finner lastens kraft

midt1=(F\*(L^2)) / (9\*sqrt(3));

% Merk at dette ikke helt er på midten, men da vi er noe begrenset i

% våre midler tok vi formelen ut fra Irgens formelsamling og den

% oppgir bare momentet på L/sqrt(3), som jo er NESTEN på midten

midt2= -(abs(endemoment(element,1))-abs(endemoment(element,2)))/2;

% Finner midtmomentet ut i fra endemomentene

midtmom(element,1)=midt1 + midt2 + midtmom(element,1);

% Finner totalt moment og plasserer det på rett sted

end

end

end

##### Vedlegg 14: boyespenning.m

function [spenning] = boyespenning(I\_elem,elem,nelem,profil,endemoment,midtmom )

%{

BOYESPENNING: Regner ut bøyespenningene for elementene,

gitt i [kN/m^2] = [kPa].

%}

spenning=zeros(nelem,3);

for i=1:nelem

if elem(i,4)==1

I=I\_elem(i);

Y=(profil(1,3))/2;

M1=endemoment(i,1); % Finner endemomentet i ende 1

M2=endemoment(i,2); % Finner endemomentet i ende 2

M3=midtmom(i,1); % Finner midtmomentet

% Regner ut bøyespenningene

Spenning1=(M1\*Y)/I;

Spenning2=(M2\*Y)/I;

Spenning3=(M3\*Y)/I;

% Plasserer bøyespenningene på rett sted

spenning(i,1)=Spenning1;

spenning(i,2)=Spenning2;

spenning(i,3)=Spenning3;

else

I=I\_elem(i);

Y=(profil(2,2))/2;

M1=endemoment(i,1); % Finner endemomentet i ende 1

M2=endemoment(i,2); % Finner endemomentet i ende 2

M3=midtmom(i,1); % Finner midtmoment

% Regner ut bøyespenningene

Spenning1=(M1\*Y)/I;

Spenning2=(M2\*Y)/I;

Spenning3=(M3\*Y)/I;

% Plasserer bøyespenningene på rett sted

spenning(i,1)=Spenning1;

spenning(i,2)=Spenning2;

spenning(i,3)=Spenning3;

end

end

end

##### Vedlegg 15: sjekkspenning.m

function sjekkspenning(spenning,flyt,nelem)

%{

SKJEKKSPENNING: Sjekker om bøyespenningene overskrider 70% av flytspenningen

%}

for i=1:nelem

if abs(spenning(i,1)) > (flyt\*70)/100

disp ('Ikke okei')

elseif abs(spenning(i,2)) > (flyt\*70)/100

disp ('Ikke okei')

elseif abs(spenning(i,3)) > (flyt\*70)/100

disp ('Ikke okei')

else disp('WOOOHOOO')

end

end

##### Vedlegg 16: skjearkrefter

function [V] = skjearkrefter(nelem,elemlengder, nlast, last,endemoment)

%{

SKJEARKREFTER: Regner ut skjærkreftene i alle elementene og skriver ut

verdiene

%}

V = zeros(nelem, 2); %Danner matrise med nuller

for i = 1:nelem %Går gjennom alle elementene

M1 = endemoment(i,1); %Finner endemomentet i lokal ende 1

M2 = endemoment(i,2); %Finner endemomentet i lokal ende 2

L = elemlengder(i); %Finner lengden til bjelken

skjearende = (M1 + M2) / L; %Finner endemomentene sitt bidrag til skjærkraften

V1 = skjearende; %Setter skjær i ende 1 lik endebidraget

V2 = skjearende; %Setter skjær i ende 2 lik endebidraget

for j = 1:nlast %Går gjennom lastene

if last(j,2) == i %Sjekker om lasten virker på det aktuelle elementet

if last(j,1) == 1 %Sjekekr om laster en punktlast

if last(j,3) > 0 %Tar ikke med laster som virker i knutepunkt

F = last(j,4) \* cos(last(j,5)); %Finner lastens kraft

avstand1 = last(j,3) \* L; %Finner avstand til lokal ende 1

avstand2 = L - avstand1; %Finner avstand til lokal ende 2

V1 = -V1 + (F \* avstand2) / L; %Legger til skjærkraften

V2 = -V2 - (F \* avstand1) / L; %Legger til skjærkraften

end

elseif last(j,1) == 2 %Sjekker om det er jevnt fordelt last

A = (last(j,4) \* L) / 2; %Finner opplager 1

B = A; %Finner opplager 2

skjear1 = -A; %Finner skjærkraft 1

skjear2 = B; %Finner skjærkraft 2

V1 = V1 - skjear1; %Legger kreftene til på rett sted

V2 = V2 - skjear2;

elseif last(j,1) == 3 %Sjekker om det er lineært fordelt last

if last(j,3) == 1 %Sjekker om den virker i lokal ende 1

A = (last(j,4) \* L) / 3; %Regner ut opplager 1

B = ((last(j,4) \* L) / 2) - A; %Regner ut opplager 2

skjear1 = - (((last(j,4) \* L) / 2) - B); %Finner skjærkraft1

skjear2 = B; %Finner skjærkraft2

elseif last(j,3) == 2 %Sjekker om den virker i lokal ende 2

B = (last(j,4) \* L) / 3; %Finner opplager 2

A = ((last(j,4) \* L) / 2) - B; %Finner opplager 1

skjear2 = (((last(j,4) \* L) / 2) - A); %Finner skjærkraft 2

skjear1 = -A; %Finner skjærkraft 1

end

V1 = V1 + skjear1; %Legger til kreftene på rett sted

V2 = V2 + skjear2;

end

end

V(i,1) = -V1; %Plasserer kreftene på rett plass i matrisen

V(i,2) = -V2;

end

end

##### Vedlegg 17: boyestivhet.m

function [stivhet] = boyestivhet(nelem, lengder, elem, I\_elem)

%{

BOYESTIVHET: Regner ut bøyestivheten til elementene

OUTPUT: Vektor med et tall for bøyestivheten til hvert element nedover),

f.eks til element 4 lagret i rad 4.

%}

%Danner en matrise med nuller i rett størrelse

stivhet = zeros(nelem,1);

for i = 1:nelem %Går gjennomm alle elementene

E\_elem = elem(i,3); %Finner elementets E-modul hentet fra inputfila i MPa

I = I\_elem(i); %Finner annet arealmoment til elementet

L = lengder(i); %Henter ut lengden til elementet

stivhet(i,1) = (E\_elem\*I)/L; %Regner ut bøyestivheten (og deler på L

%fordi det er den verdien vi trenger og plasserer den på rett sted)

end

end

##### Vedlegg 18: output.m

function output(rot,npunkt,endemoment,nelem)

%{

OUTPUT: En funksjon som gjør utskrifen av programmets resultater lettere

å lese. Skriver ut element/knutepunkt og nummmer i tillegg til de

korresponderende verdiene for moment og rotasjon

%}

%Skriver ut rotasjonene knyttet til knutepunkt

fprintf('\nRotasjoner for hvert knutepunkt:\n\n');

for i=1:npunkt

fprintf('Knutepunkt %i:\t\t%f\n',i,rot(i,1));

end

%Skriver ut endemomentetene for hvert element

fprintf('\n\n\nEndemoment for hvert element (lokal ende 1 og lokal ende 2):\n\n');

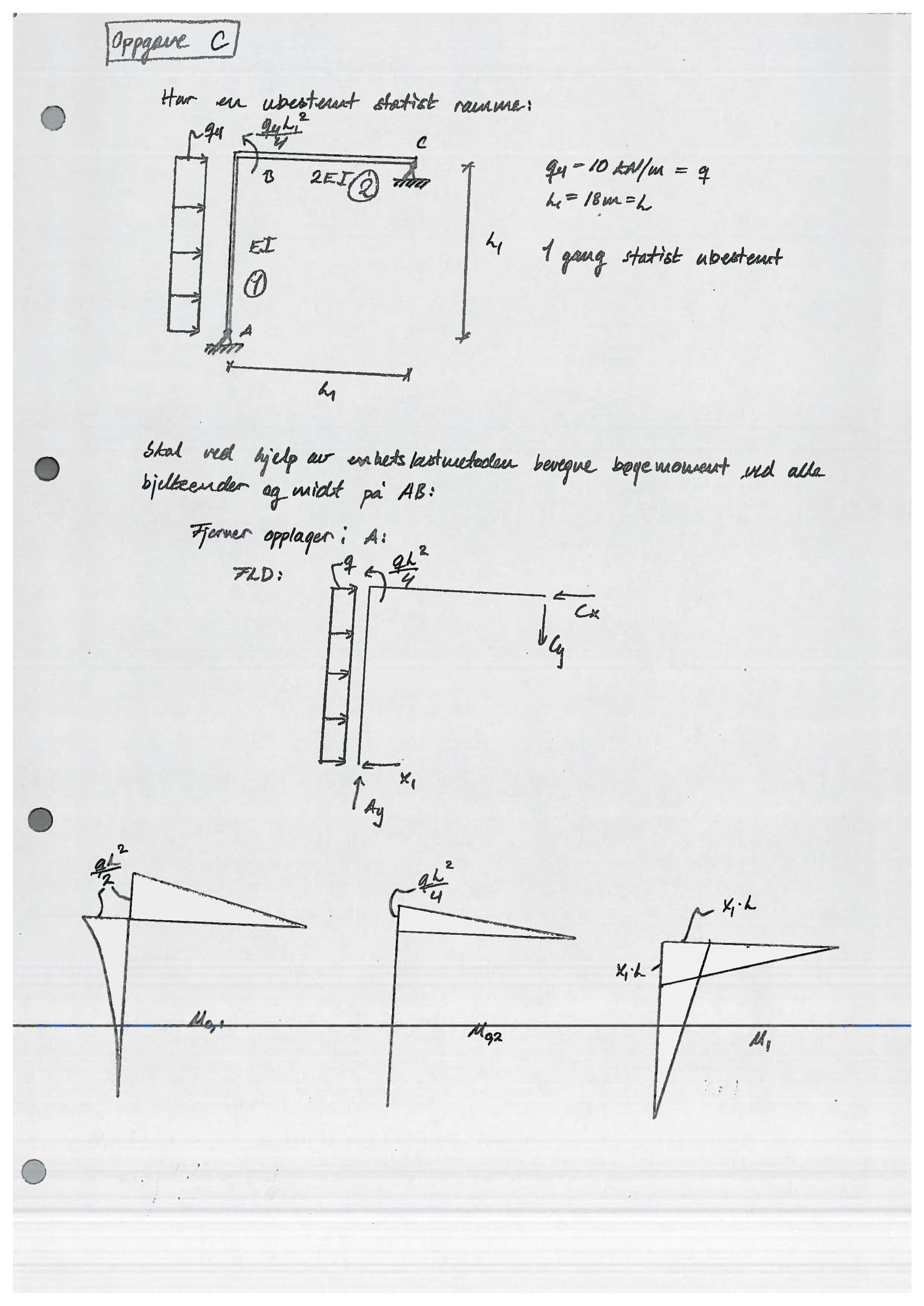
for j=1:nelem

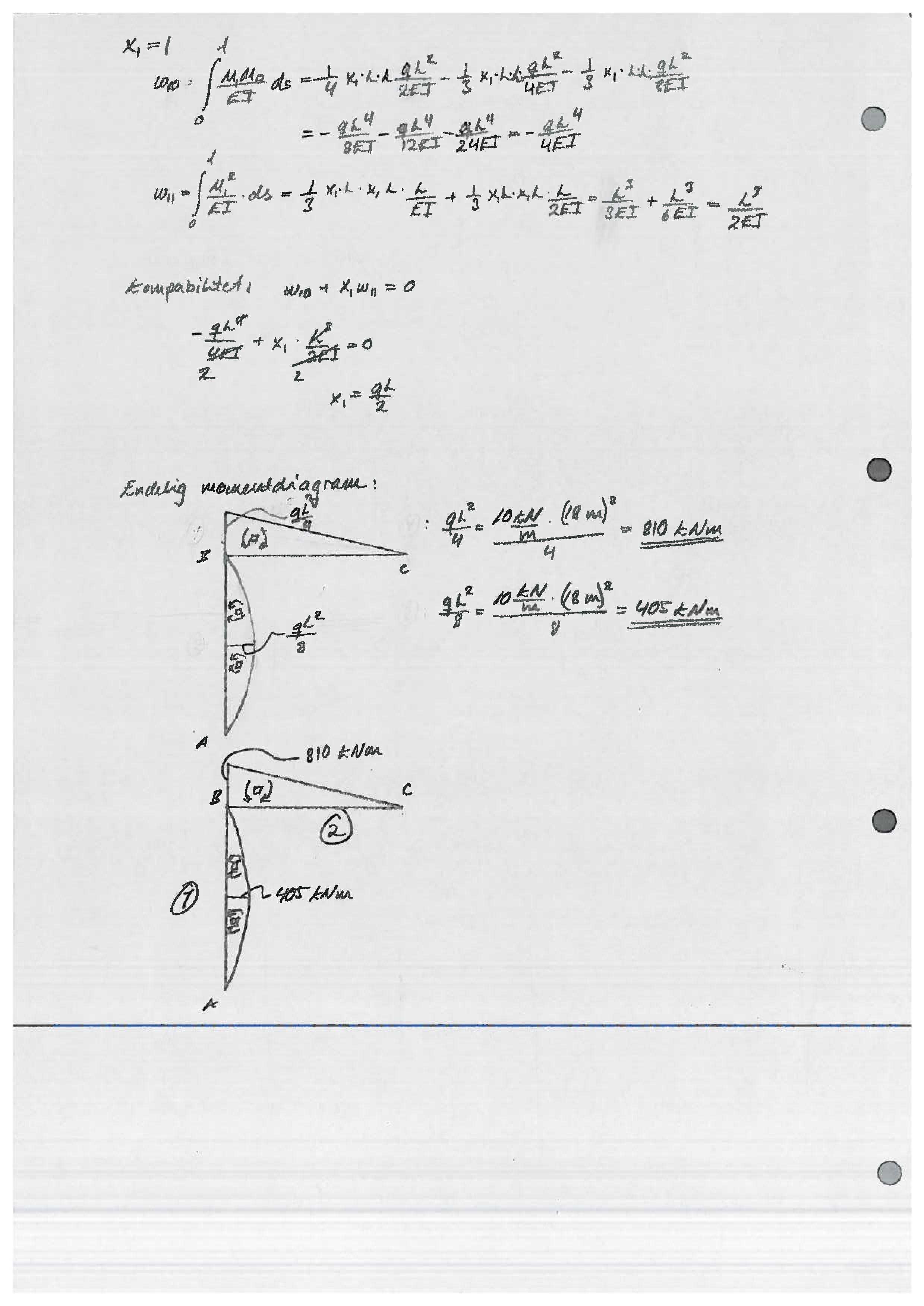
fprintf('Element %i:\t\t%f\t\t%f\n', j, endemoment(j,1), endemoment(j,2));

end

end

##### Vedlegg 19: Oppgave c del 1



Vedlegg 20: Oppgave c del 2   


##### Vedlegg 21: Inputfil til oppgave C

3

0 0 0

0 18 0

18 18 0

2

1 2 210e3 1

2 3 420e3 1

1

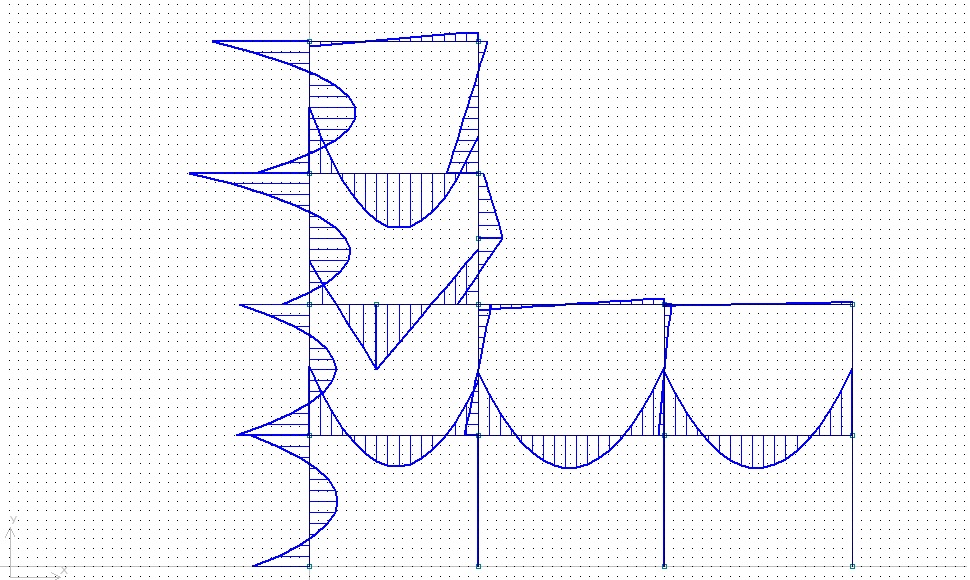
1 0 0.2 0 0.2

2

2 1 1 10 0

4 2 0 -810 0

##### Vedlegg 22: Momentdiagram fra Nauticus 3D Beam



##### Vedlegg 23: Skjærkraftdiagram fra Nauticus 3D Beam

