# KARAKTERGIVENDE OPPGAVE

TMA4167 MARINE KONSTRUKSJONER

Kandidater: 10110, 10129, 10021

03.11.2014

NTNU, Trondheim

# Forord

I faget TMR4167: Marine konstruksjoner fikk vi et prosjekt hvor hensikten var å få trening i å utføre beregningsoppgaver med MATLAB. I tillegg skulle vi få innsikt i oppbygningen av dataprogram for analyse av rammekonstruksjoner. Vi har fått svært god øving av matrisemetoden, samt den praktiske utnyttelsen av MATLAB.

Det er viktig å merke seg at den metoden vi har brukt, ikke blir brukt til dimensjonering av virkelige konstruksjoner ettersom vi kun tar hensyn til rotasjonene. Dette er kun en dimensjonering vi kan bruke tidlig i en prosjekteringsfase for å få en pekepinn på størrelsene.

Prosjektet har vært en gruppeoppgave som er gjennomført av de tre studentene med kandidatnumrene: 10110, 10129 og 10021. Vi har samarbeidet om alle hovedelementene i prosjektet, altså kodingen, utregningen og rapportskrivingen.

Vi vil gjerne takke studentassistentene for hjelpen med prosjektet.

# Sammendrag

Denne prosjektoppgaven gikk ut på å lage et program i MATLAB som kunne analysere en vilkårlig todimensjonal rammekonstruksjon med matrisemetoden. Vi skulle bruke programmet til å analysere et utsnitt av et plattformdekk, og finne passende tverrsnittsdimensjoner. Plattformdekket skulle bære tunge laster (P1, P2, q1, q2, q3), tåle sterke vindkrefter (lineært fordelt last fra q4 til q5) og ha en dekkskran (modelert med knutepunktsmomentet M1 og P3). Konstruksjonen skulle bestå av rør- og boksprofiler, men i kampens hete ble det i vedlagt MATLAB-program rør- og I-profiler. Dette er en beklagelig feil som vi håper ikke har betydelig konsekvens for prosjektoppgaven. Grunnet overseelse fra vår side, fant vi ikke ut at vi hadde feil profil før helt i slutten av prosjekteringstiden.

Vi programmerte et fungerende MATLAB-program som støtter alle kriteriene oppgaven ga. Programmet tar en inputfil med informasjon om rammekonstruksjonen, gjør beregninger i en rekke underfunksjoner, og leverer resultatene i en egen fil.

Ved hjelp av iterasjoner fikk vi de vertikale rørprofiler med tverrsnittsdimensjonene $d$=450mm og $t$=20mm, og horisontale I-profiler med dimensjonene tilsvarende med IPE450. Elementet med størst påkjenning skulle ha en bøyespenning i område 30-70% av oppgitt flytespenning. Maksimal bøyespenning virket på element 3 med 162,37MPa. Dette tilsvarer 50,74% av flytespenningen gitt av oppgaven på 320MPa.

# Innholdsfortegnelse

# Figurer



# Tabeller



# Formelliste

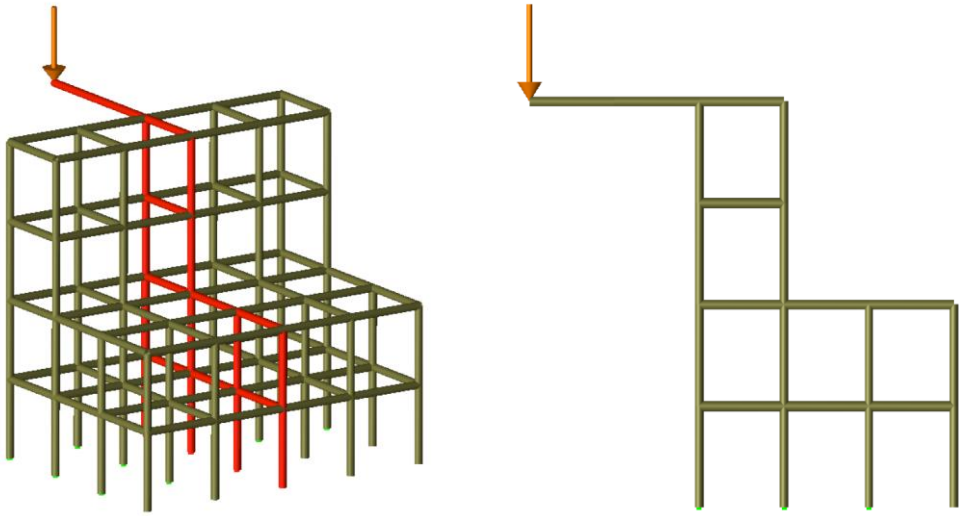
# Vedlegg

# 1 Innledning

## 1.1 Oppgaven

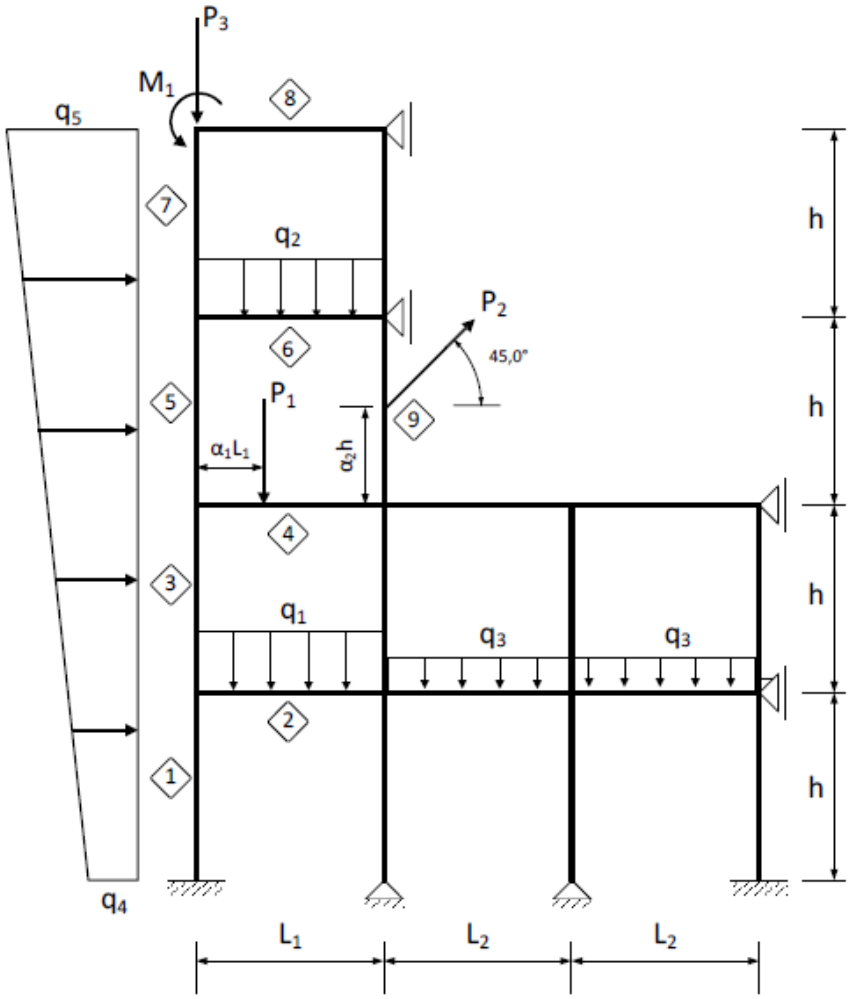
Oppgaven vår går ut på å lage et program i MATLAB som kan analysere en hvilken som helst todimensjonal konstruksjon ved hjelp av matrisemetoden. Elementvis skal programmet kunne finne endemomentene, samt midtmomentet på bjelken. Om det er elementer som er påvirket av punktlast, skal programmet også kunne oppgi momentet under punktlasten.

Vi har fått utgitt en figur av den lastbærende strukturen i en modul på et plattformdekk. Øverst på denne modulen er det en dekkskran som påvirker konstruksjonen med betydelige krefter (se Figur 1). Vår oppgave er å utføre en todimensjonal rammeanalyse for det planet som overfører høyst andel av kranlasten. Ut i fra beregningene våre skal vi kunne fastsette dimensjonene til bjelkene og søylene i konstruksjonen.



*Figur 1. Lastbærende konstruksjon til venstre. To-dimensjonalt utsnitt.*

## 1.2 Konstruksjonen




*Figur 2. Rammekonstruksjon med laster og lengder.*[3]

En idealisert regnemodell av stålrammen som skal dimensjoneres er vist på Figur 2. Rammen skal antas å være uforskyvelig slik at deformasjoner er fullstendig beskrevet av knutepunktrotasjoner. De vertikale søylene i rammen består av rørtverrsnitt, og de horisontale bjelkene består av I-profiler. I dette prosjektet antas det at kapasitetskravet er oppfylt uavhengig av valgte tverrsnittsdimensjoner. Størrelsen på lastene og lengdene fikk vi utlevert. Noen av verdiene er like for alle, og noen velges ut i fra et kanditatnummer. Vi har brukt kanditatnummer 10110, og får dermed følgende verdier:

*Tabell 1. Fordelte laster, punktlaster og moment.*

| *Laster* | $q_1$ [kN/m] | $q_2$ [kN/m] | $q_3$ [kN/m] | $q_4$ [kN/m] | $q_5$ [kN/m] | $P_1$ [kN] | $P_2$ [kN] | $P_3$ [kN] | $M_1$ [kN/m] |
|---|---|---|---|---|---|---|---|---|---|
| *Verdi* | 7 | 8 | 6 | 10 | 16 | 80 | 40 | 60 | 340 |

*Tabell 2. Lengder og høyder.*

| *Lengder* | $L_1$ [m] | $L_2$ [m] | h [m] | $\alpha_1$ [-] | $\alpha_2$ [-] |
|---|---|---|---|---|---|
| *Verdi* | 18 | 20 | 14 | 0,4 | 0,5 |

*Tabell 3. E-modul og flytspenning.*

| *E-modul* [GPa] | 210 |
|---|---|
| *Flytspenning* [MPa] | 320 |

### 1.3 Rapporten

Rapporten er skrevet hovedsakelig etter at det meste av programmeringen var ferdig. Her tar vi først for oss fremgangsmåten for prosjektarbeidet og teorien bak matrisemetoden. Alle formler i denne delen er hentet fra *TMR4167 Marin Teknikk 2 Del 1: Konstruksjonsanalyse*[1] og Irgens *Formelsamling mekanikk*[2]. Til en viss grad blir det beskrevet både metode for å regne ut resultater for hånd (momenter, rotasjoner, bøyespenninger osv.), og metode for programmeringen i MATLAB. I resultatdelen blir de endelige resultatene beskrevet og diskutert. Denne delen inneholder en analyse av hovedkonstruksjonen (Figur 2) gjort i programmet. Den inneholder også en sammenligning av den digitale analysen og håndberegninger gjort på den enkle konstruksjonen gitt i oppgave C.

## 2 Fremgangsmåte og teori

### 2.1 Matrisemetoden

Matrisemetoden bygger på deformasjonsmetoden. Hovedforskjellen mellom disse er at matrisemetoden beregner på systemet som en helhet, mens deformasjonsmetoden tar for seg et og et element. I dette prosjektet vil kun knutepunktsrotasjonene være ukjente da vi neglisjerer aksial- og skjærkraftdeformasjoner. Det vi hovedsakelig skal finne når vi bruker matrisemetoden, er stivhetsmatrise K og lastvektor R, som begge er komponenter i systemrelasjonen (ligning 1) som vi skal løse i MATLAB.

$$\mathbb{K}\boldsymbol{r} = \mathbb{R}$$

*Formel 1. Systemrelasjonsligningen.*

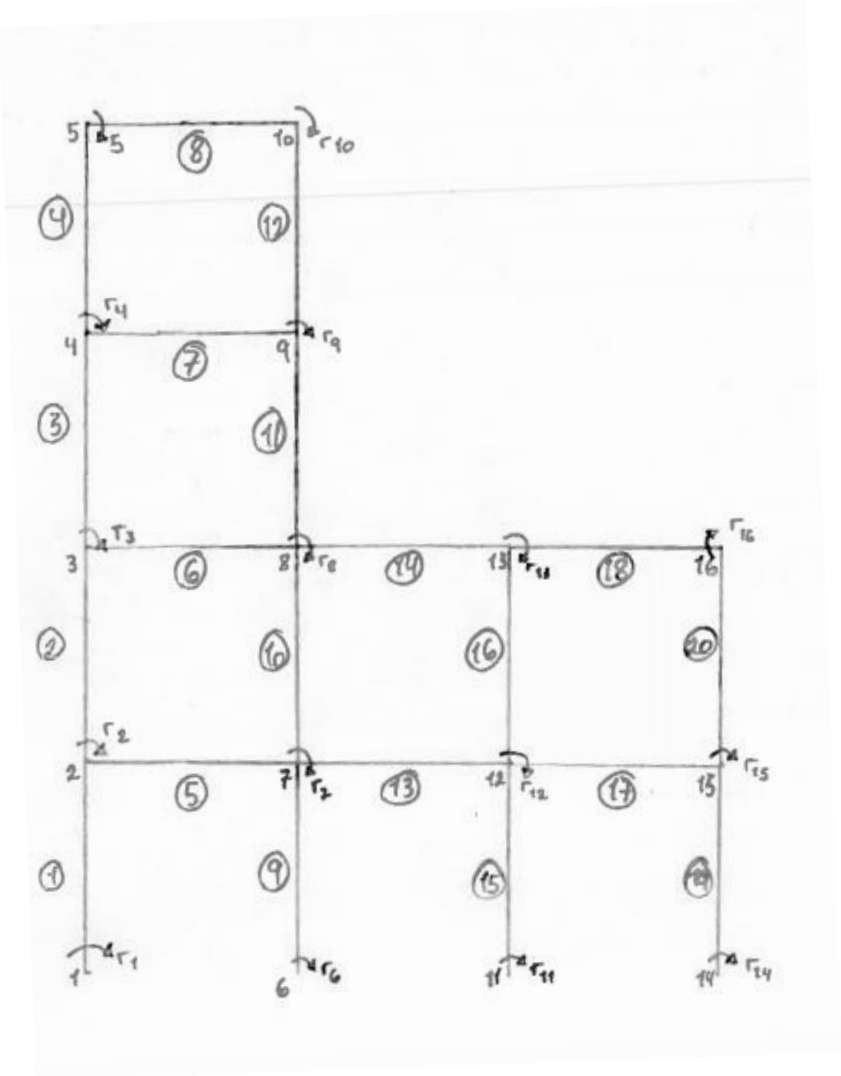
Der $\mathbb{K}$ er stivhetsmatrisen, $\boldsymbol{r}$ = rotasjonsvektor og $\mathbb{R}$ = er lastvektor. Alle disse elementene vil nærmere beskrevet senere i rapporten.

MATLAB-programmet skal blant annet kunne hente informasjon om bjelkenes dimensjoner og materialer, for deretter å behandle denne informasjonen slik at vi får en stivhetsvektor. Ut i fra informasjonen om ytre laster skal programmet også opprette en lastvektor. Når vi

har fått både stivhetsmatrise og lastvektor, har vi det vi trenger for å beregne rotasjonsvektoren, som skal brukes videre i programmet. Denne skal blant annet brukes til å beregne endemomenter, midtmomenter, bøyespenning, skjærkrefter og normalkrefter.

## 2.2 Diskretisering

Det første man gjør når man skal begynne å analysere, er å foreta en diskretisering; en oppdeling av konstruksjonen i et endelig antall elementer og knutepunkter. Vi nummererer elementene og knutepunktene hver for seg. Dette må gjøres på en geometrisk fornuftig måte for at resultatene skal bli oversiktlige og korrekte. Vi valgte å diskretisere fra nede til venstre, til oppe til høyre. Dermed får vi enkelt en oversikt over hvor på konstruksjonen enkelte krefter virker når vi begynner med beregningene.



*Figur 3. Diskretisering.*

I Figur 3 er en oversikt over utført diskretisering. Ut ifra denne kan vi lage en konnektivitetsmatrise som viser sammenhengen mellom lokale og globale frihetsgrader. Konnektivitetsmatrisa er vist under, i Tabell 4.

*Tabell 4. Konnektivitetsmatrise. DOF = Degree of freedom/frihetsgrad.*

| *Elementnr.* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *DOF 1* | 1 | 2 | 3 | 4 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 7 | 8 | 11 | 12 | 12 | 13 | 14 | 15 |
| *DOF 2* | 2 | 3 | 4 | 5 | 7 | 8 | 9 | 10 | 7 | 8 | 9 | 10 | 12 | 13 | 12 | 13 | 15 | 16 | 15 | 16 |

## 2.3 Inputfilen

Lesinput-filen er grunnlaget for hele programmet. Denne tar inn informasjon om konstruksjonen fra en tekstfil (inputfil), og behandler denne. Formålet med lesinput-filen er at informasjon om konstruksjonen skal gjøres brukbar i MATLAB-programmet. Lesinput.m tar inn informasjon om element og knutepunkt (som ble bestemt gjennom diskretiseringen), laster, dimensjoner, E-modul og stivhet. Alle disse blir organisert og definert gjennom lesinput-filen, slik at programmet lettere kan bruke opplysningene videre.

## 2.4 Element, knutepunkt, koordinater og E-modul

Gjennom diskretiseringen har vi nummerert alle knutepunktene og elementene. Dette blir gjort på en geometrisk systematisk måte fra nederst til venstre, til øverst til høyre. På denne måten får vi koordinater til konstruksjonens knutepunkt der origo er i nederste venstre hjørne. Enheten i x- og y-retning er meter. Både elementenes nummer og plassering, knutepunktenes nummer og plassering, og E-modul er informasjon som blir lest av lesinput-filen.

## 2.5 Laster og moment

Konstruksjonen er påvirket av lineært fordelte laster, punktlaster og ytre moment. I lesinput-filen blir de forskjellige type lastene og momentene analysert hver for seg. Dette gjør det enklere for resten av programmet å lese informasjonen.

For lineært fordelte laster er informasjonen som trengs: antall laster og hvilke element de går over, samt endeamplitudene. Vi vil ikke at det skal være nødvendig å gjøre beregninger for hånd, og programmet er derfor skrevet slik at man ikke trenger å splitte opp lineært fordelte laster som går over flere element. For punktlaster blir det tatt inn informasjon om hvor stor kraften er, hvor den virker, hvilke element den virker på og med hvilken vinkel den

virker. Noenlunde samme informasjon blir tatt inn om ytre moment, nemlig størrelse på momentet, hvilke knutepunkt det virker på, og hvilken retning det går i. Positiv retning er med klokken, og fortegnet viser om momentet er positivt eller negativt.

## 2.6 Lastvektor

Lastvektoren lages ved å summere alle fastinnspenningsmoment og eventuelle ytre moment i hvert knutepunktet. Fastinnspenningsmomentene summeres med negativt fortegn. Bakgrunnen for dette er at bjelkeendemomentet pluss fastholdingsmomentet i et knutepunkt, skal være lik det ytre momentet i knutepunktet. Lastvektoren inngår i systemrelasjonsligningen, og er dermed viktig for beregningen av rotasjonene i knutepunktene.

## 2.7 Fastinnspenningsmomenter

Når man skal beregne fastinnspenningsmomentet, ser man for seg at bjelken er fast innspent, og ser på det momentet man da får av ytre last på bjelken. I vårt tilfelle har vi bare to typer ytre laster; punktlast og lineært fordelt last. De formlene vi har benyttet er derfor

$$m_{ende1} = -\frac{Pab^2}{L^2}$$

*Formel 2. Fastinnspenningsmomentet for punktlaster fra ende 1.*

$$m_{ende2} = \frac{Pa^2b}{L^2}$$

*Formel 3. Fastinnspenningsmomentet for punktlast fra ende 2.*

for punktlaster, der *P* er punktlasten, *a* er avstand mellom ende 1 og P, *b* er avstand mellom ende 2 og P, og *L* er lengder på elementet. De jevnt fordelte lastene deler vi opp i to trekanter istedenfor en firkant, og får dermed formlene

$$m_{ende1} = -\frac{q_1 L^2}{20} - \frac{q_2 L^2}{30}$$

*Formel 4. Fastinnspenningsmomentet for trapeslast fra ende 1.*

$$m_{ende2} = \frac{q_2 L^2}{20} + \frac{q_1 L^2}{30}$$

*Formel 5. Fastinnspenningsmomentet for trapeslast fra ende 2.*

der $q_1$ er lastintensiteten i ende 1, $q_2$ er lastintensiteten i ende 2, og *L* er lengden på elementet.

## 2.8 Lokal stivhetsmatrise

Et av de viktige elementene som trengs for å beregne rotasjonene i endepunktene, er den globale stivhetsmatrisen. For å komme frem til denne trenger man først de lokale stivhetsmatrisene. Dette vil si en stivhetsmatrise for hvert element. Den generelle formelen for denne matrisen er

$$k_i = \frac{E_i I_i}{L_i}\begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}$$

*Formel 6. Lokal stivhetsmatrise.*

der *k* blir stivhetsleddet for element nummer i, $E_i$ er E-modul for element i, $I_i$ er 2. arealmoment for element i og $L_i$ er lengden på element i. Fra formelen ser man at det eneste som skiller de forskjellige stivhetsmatrisene, er verdiene for E-modul, annet arealmoment og lengden på elementet.

## 2.9 Global stivhetsmatrise

Når man har de lokale stivhetsmatrisene kan man sette disse inn i en global stivhetsmatrise. Dette gir en global sammenheng mellom alle de forskjellige elementene. Vi gjør dette ved hjelp av konnektivitetsmatrisen (Tabell 4) siden det er denne som gir sammenhengen mellom elementene og knutepunktene som ble bestemt ut ifra diskretiseringen. Fra konnektivitetsmatrisen kan man lese av plasseringen for hvert stivhetsledd i den globale stivhetsmatrisen.

## 2.10 Løsning av ligningssystem og beregning av endemoment

Vi har nå beskrevet alle delene som inngår i systemrelasjonsligningen (ligning 1). Ved hjelp av den globale stivhetsmatrisen og lastvektoren kan man løse denne ligningen med hensyn på knutepunktsrotasjonen **r**. Det er denne vi trenger for å videre beregne endemomentene for alle elementene i konstruksjonen.

## 2.11 Randbetingelser

For å løse systemrelasjonsligningen er det nødvendig å innføre randbetingelser. For knutepunkt med fast innspenning vil det ikke bli rotasjon, dermed kan man neglisjere rader og kolonner der dette gjelder. Dette medfører også at momenter i lastvektoren for det aktuelle knutepunktet kan neglisjeres.

## 2.12 Beregning av endemomenter

For å løse systemrelasjonsligningen må man gjøre om på ligning 1 slik at man får rotasjonen på den ene siden. Dette gjøres ved å ta den inverse av stivhetsmatrisen og gange denne med lastvektoren slik

$$\boldsymbol{r} = \mathbb{K}^{-1}\mathbb{R}$$

Man kan da løse ligningen med hensyn på **r**, og får da en vektor med alle rotasjonene i alle knutepunkt. Disse settes deretter inn i de lokale stivhetsmatrisene for å få ut alle de forskjellige endemomentene. Likningen for dette blir da seende slik ut:

$$\begin{bmatrix} M_i \\ M_j \end{bmatrix} = \frac{EI}{L}\begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} \theta_i \\ \theta_j \end{bmatrix}$$

*Formel 7. Lokal stivhetsmatrise med rotasjonsledd.*

der *M* er bjelkens endemoment, *i* er knutepunkt 1, *j* er knutepunkt 2 og $\vartheta$ er rotasjonen.

## 2.13 Midtmoment

For å beregne midtmomentet på hver bjelke blir superposisjonsprinsippet brukt. Dette går ut på at momentdiagrammet blir splittet opp i to, og deretter summert sammen igjen. De delene vi har valgt å splitte er momentdiagram skapt av ytre last og momentdiagram skapt av bjelkens endemomenter. Ved hjelp av dette kan det midtre momentet på bjelkene beregnes.

## 2.14 Moment på grunn av bjelkens endemoment

Vi har to typer laster i vår konstruksjon; lineært fordelte laster og punktlaster. For lineært fordelte laster er det fordelaktig å finne momentet midt på bjelken. Dette gjøres ved hjelp av endemomentene som ble beregnet ved hjelp av ligning 7. Vi får da at

$$M_{midt} = \frac{-M_1 + M_2}{2}$$

*Formel 8. Midtmoment på grunn av endemomenter for bjelker med jevnt fordelt last.*

der $M_1$ er momentet i punkt 1, og $M_2$ er momentet i punkt 2 på bjelken. Når bjelken er påvirket av en punktlast, er det derimot interessant å finne momentet på det punktet der punktlasten virker. Man kan gjøre dette ved hjelp av en variasjon av ligning 8. Den nye ligningen blir da som følger

$$M_{midt} = \frac{-M_1(L-a) + M_2 a}{L}$$

*Formel 9. Midtmoment på grunn av endemomenter for bjelker med punktlast.*

der $M_1$ er momentet i punkt 1 og $M_2$ er momentet i ende 2 av bjelken slik som i ligning 8. $a$ er da avstanden mellom punktlasten og ende 1, mens $L$ er hele lengden på bjelken. Forholdet mellom disse to ligningene er at man får ligning 8 om man setter inn $a = \frac{1}{2}$L i ligning 9.

## 2.15 Moment på grunn av ytre last på en fritt opplagt bjelke

Også her må det skilles mellom punktlaster og lineært fordelte laster. For punktlaster kan midtmomentet beregnes ved hjelp av formelen

$$M_{midt} = -\frac{Pab}{L}$$

*Formel 10. Midtmomenter for bjelker med jevnt fordelt last.*

der $P$ er størrelsen på punktlasten, $a$ er avstanden fra ende 1, $b$ er avstanden fra ende 2 og $L$ er lengden på bjelken. Når det gjelder lineært fordelte laster må man differensiere enda litt til. Om begge endeamplitudene er like store kan man bruke formelen

$$M_{midt} = -\frac{qL^2}{8}$$

*Formel 11. Midtmomenter for bjelker med punktlast.*

der $q$ er lastintensiteten og $L$ er lengden på bjelken.

Om endeamplitudene er ulike, må man regne ut midtmomentet på en annen måte. Her sier oppgaveteksten at man skal prøve å unngå å bruke snittmetoden, selv om denne sikkert hadde vært første alternativ om beregningen skulle gjøres for hånd på en mindre konstruksjon. På grunn av dette brukes her superposisjonsprinsippet i stedet. For trapesformede fordelte laster vil dette si at vi deler de opp i to trekanter som vi beregner midtmomentet for, for deretter å summere de sammen igjen. Formelen vi bruker er dermed

$$M_{midt} = -\frac{\sqrt{3}}{27} q_1 L^2 - \frac{\sqrt{3}}{27} q_1 L^2$$

*Formel 12. Midtmomenter bjelker med trapeslast.*

der $q_1$ er lastintensiteten i ende 1, $q_2$ er lastintensiteten i ende to og $L$ er elementets lengde. Dette vil imidlertid gi et overslag i forhold til den virkelige verdien fordi de to trekantlastene vil ha sine maksimale momenter på hver sin side av midten.

## 2.16 Momentdiagram

Man har nå all informasjon som er nødvendig for å tegne momentdiagrammet for konstruksjonen, nemlig endemomenter og midtmomenter. Formen på kurvene vil variere med lasttypen elementet er påvirket av, og i vårt tilfelle vil dette være lineære, 2.- eller 3.-gradspolymon.

## 2.17 Skjærkraftdiagram

For å kunne lage skjærkraftdiagram må skjærkreftene som virker på konstruksjonen bestemmes. Dette gjøres ved hjelp av formelen

$$Q = Q_0 - \frac{M_i + M_j}{L}$$

*Formel 13. Skjærkraft.*

der $Q_0$ er skjærkraft skapt av ytre laster, $M_i$ og $M_j$ er endemomentene i henholdsvis ende 1 og 2 , og $L$ er lengen på elementet. Hele det siste leddet er skjærkraft skapt på grunn av endemomentene til bjelken på grunn av rotasjon. Her er det altså $Q_0$ som skaper variasjon på skjærkraften over en bjelke da det siste leddet vil gi en lineær skjærkraft over hele elementet. Ved trapeslast vil vi dermed få et 2.-gradspolynom, ved jevnt fordelt last vil vi få et lineært skjærkraftdiagram, og ved punktlast vil vi også få et lineært skjærkraftdiagram, men med hopp der punktlasten virker.

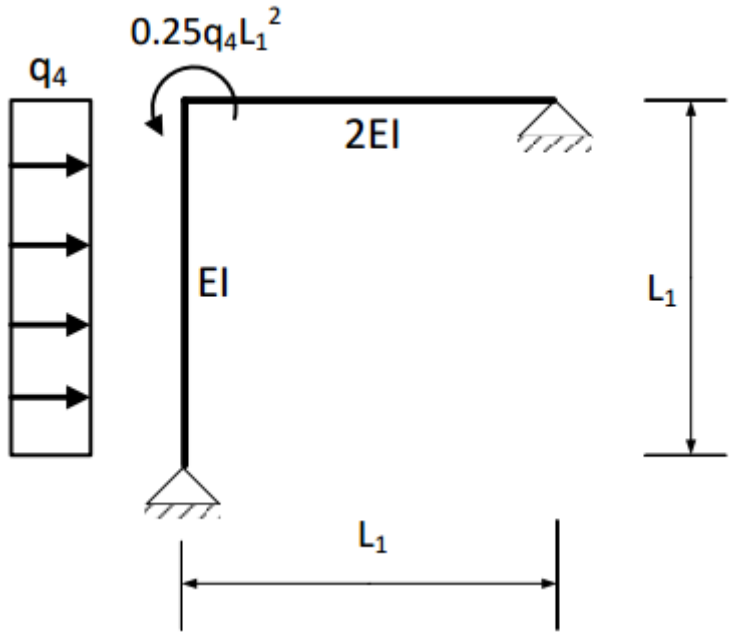
## 2.18 Normalkraftdiagram

Normalkraftdiagram kan enkelt regnes og tegnes ut i fra skjærkraftdiagrammet, men enda enklere tas ut i fra Nauticus. I og med at resultatene gitt ved programmeringen korresponderte med det vi så i Nauticus, benyttet vi oss av dette og brukte aksialdiagrammet vi fikk fram.

# 3 Resultater

## 3.1 Oppgave C

MATLAB-programmet vi har programmert skal kunne analysere en hvilken som helst konstruksjon, ikke bare den som er gitt i oppgaven. For å teste dette er en del av prosjektet å analysere en enklere konstruksjon gjennom programmet. Denne konstruksjonen blir i tillegg analysert manuelt ved hjelp av enhetslastmetoden. Konstruksjonen er vist i Figur 4.




*Figur 4. Enkel konstruksjon med laster og lengder.*[3]

Her er det brukt de samme verdiene på fordelt last og lengde som er å finne i Tabell 1 og Tabell 2. Det vil si at $q_4$ = 10 kN/m og $L_1$ = 18 m.

Ved å bruke enhetslastmetoden ble endemomentene for begge element, samt midtmomentet for elementet med jevnt fordelt last bestemt. Disse resultatene er presentert i Tabell 5. Se Vedlegg A. 5 for utregning.

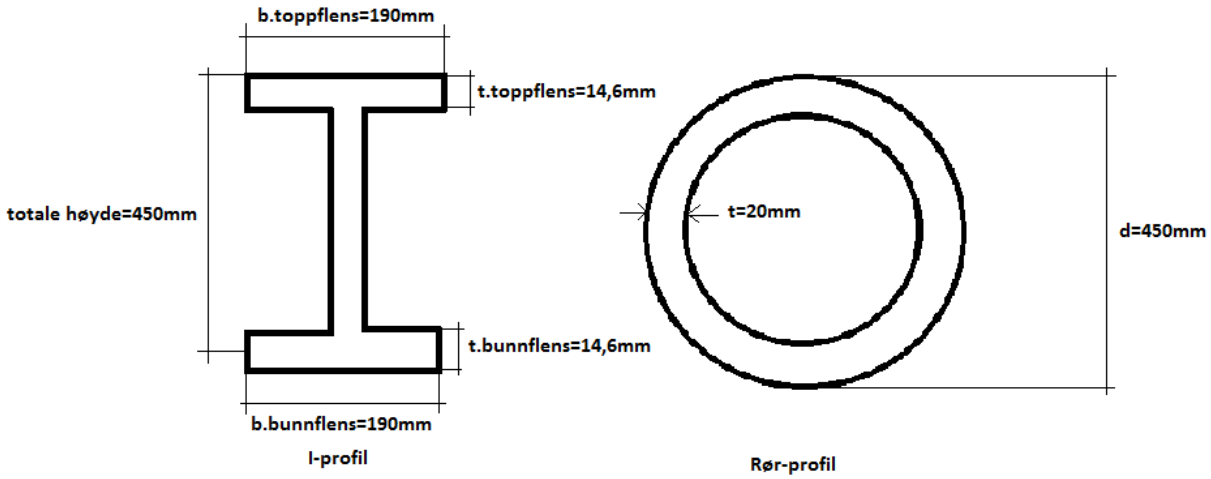
*Tabell 5. Momenter på enkel konstruksjon beregnet manuelt.*

| Elementnummer | Moment ende 1 [kNm] | Moment ende 2 [kNm] | Midtmoment [kNm] |
|---|---|---|---|
| 1 | 0 | 0 | 405 |
| 2 | 810 | 0 | - |

Informasjonen om den enkle konstruksjonen ble deretter lagt inn i MATLAB-programmet, for å teste om dette ville gi samme resultat. Det viste seg at programmet gav ut nøyaktig de samme verdiene som håndberegningen. Resultatet fra programmet er dermed identiske med verdiene som er presentert i Tabell 5.

## 3.2 Bøyespenning og dimensjonering av profiler

Bøyespenningen og tversnittsdimensjonene avhenger av hverandre, og er avgjørende for flytespenningen. Valget av dimensjoner var en itereringsprosess, det mest utsatte elementet skulle ha en maksimal bøyespenning i området 30-70% av den oppgitte flytespenningen på 320MPa. Dette tok vi høyde for i selve programmeringen, ved å legge inn noen ekstra linjer i funksjonen rammeanalyse.m (se Vedlegg B. 19) som henter ut maksimal bøyespenning og hvilket element det befinner seg på. Element 3 er det kritiske elementet med en bøyespenning på 162,37MPa - 50,74% av flytespenningen (se resultater i Vedlegg B. 20). Dette tilsvarer en IPE450 (I-profil), og rørprofil med radius 450mm og tykkelse 20mm (se Figur 5).




*Figur 5. Dimensjonering av profiler.*

## 3.3 Nauticus 3D Beam

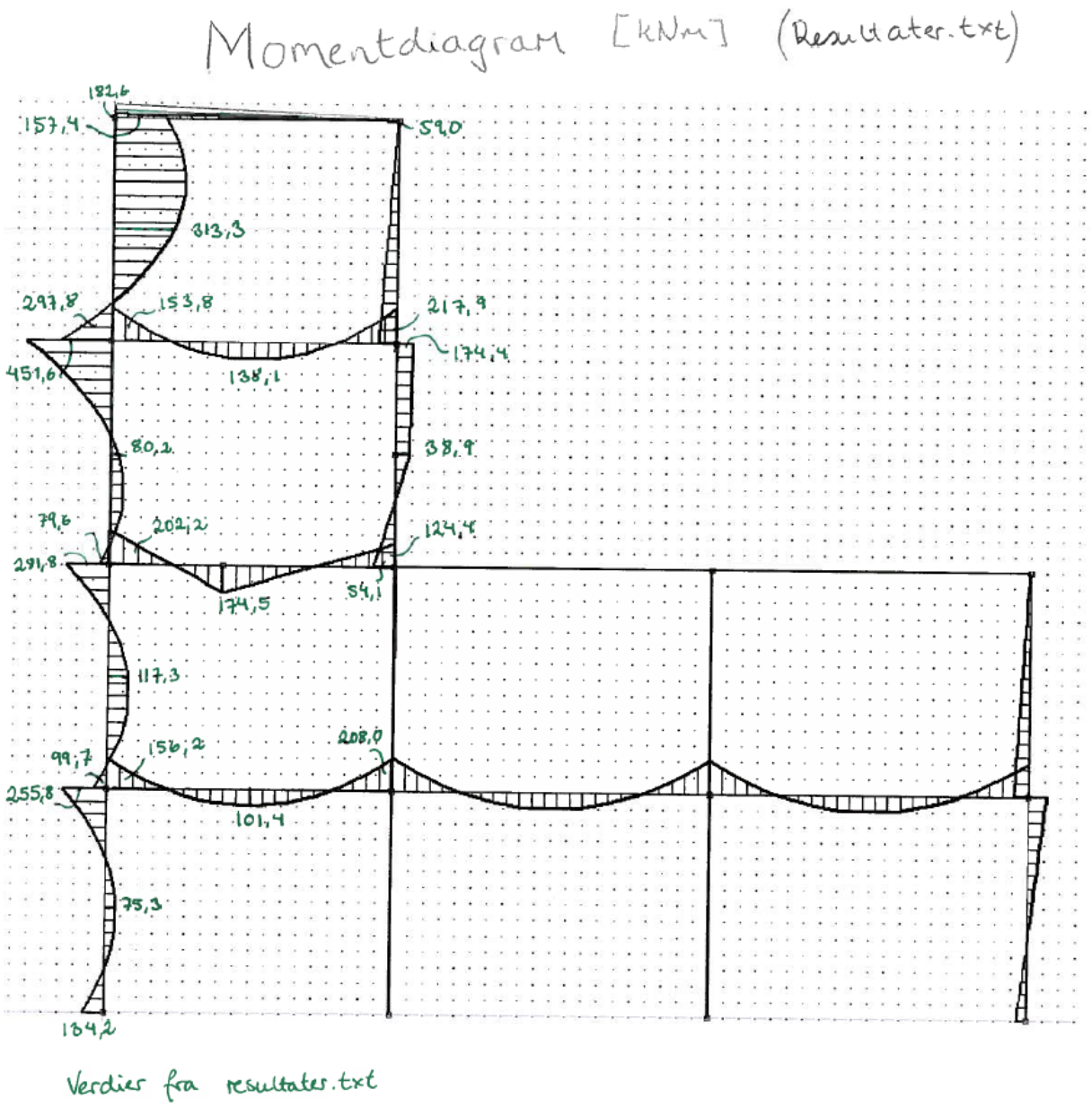
Da vi fikk resultatene fra MATLAB-programmet, kontrollerte vi disse med Nauticus 3D Beam (heretter referert til som Nauticus). I Nauticus tegnet vi opp rammen vår med tilhørende fordelte laster, punktlaster, momenter, samt valgte dimensjoner på rør- og i-profil. Da dette var gjort regnet Nauticus ut momenter, skjærkrefter og aksialkrefter, og tegnet opp tilhørende diagrammer.

Momentdiagrammet vi får fra Nauticus stemmer godt overens i form med momentdiagrammet vi får fra resultatene i MATLAB-programmet. Likevel har verdiene på momentene varierende avvik. Disse er på 1-76 kN, og er størst for elementer påvirket av punktlast. Vi har mange momenter som har en imponerende likhet mellom programmene, men også noen som har et overraskende stort avvik. Det typiske avviket er på ca. 20kN som

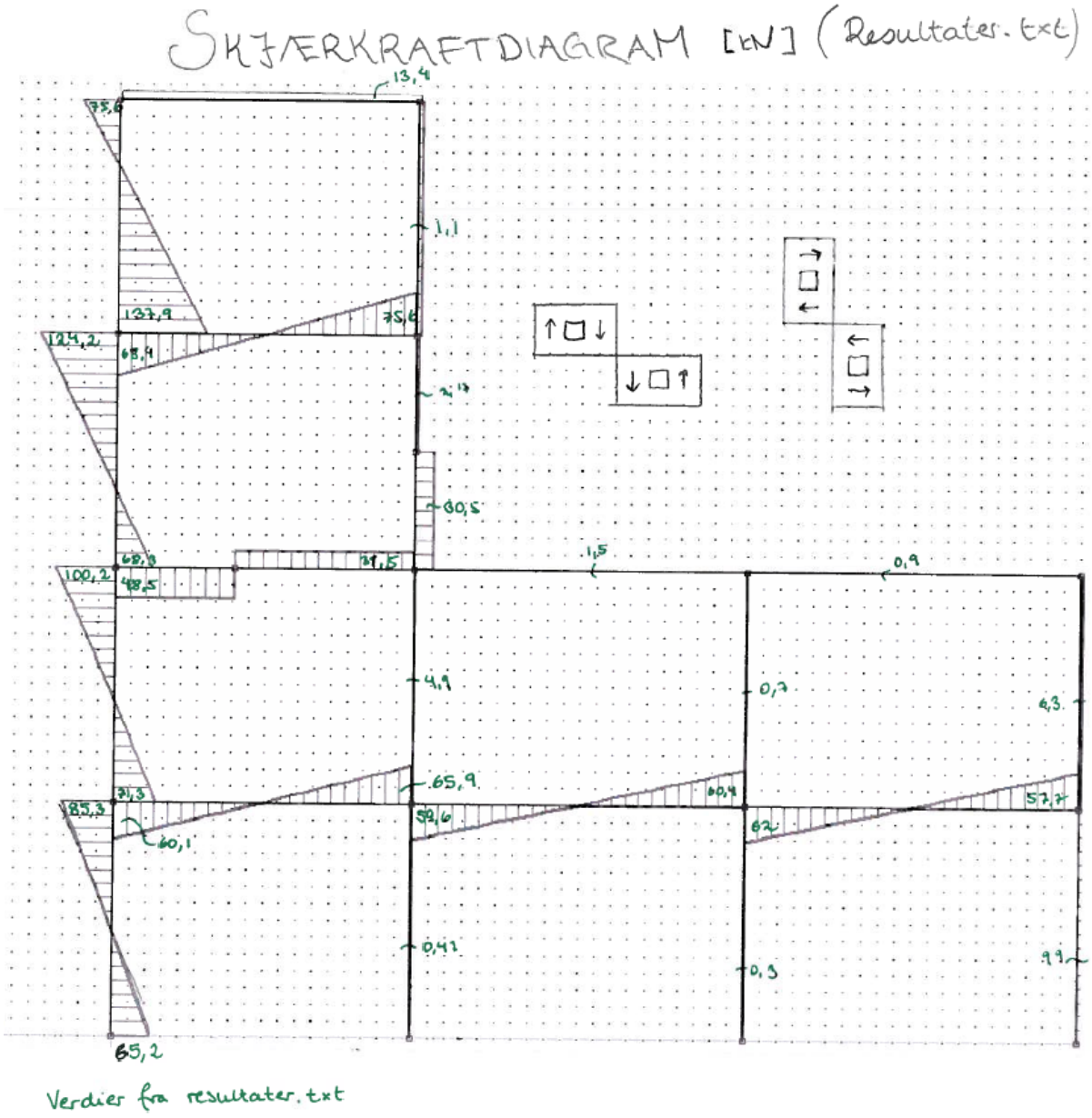
er såpass utslagsgivende, at en feil må ha skjedd. Etter mye feilsøking finner vi ut at Nauticus baserer seg på en bjelkeformulering som inkluderer bidrag fra skjærdeformasjoner, noe vi ikke skal ha i dette prosjektet. Vi antar at det er dette som har vært utslagsgivende ved de punktene med størst avvik.

Nauticus oppgir ikke midtmomenter, men om vi måler ut i fra momentdiagrammet ser vi at også disse stemmer relativt godt overens med MATLAB-resultatene. Skjærkrefter og aksialkrefter regnes ut fra momentene, så disse stemmer også forholdsvis godt. De har dermed lik nøyaktighet som endemomentene.

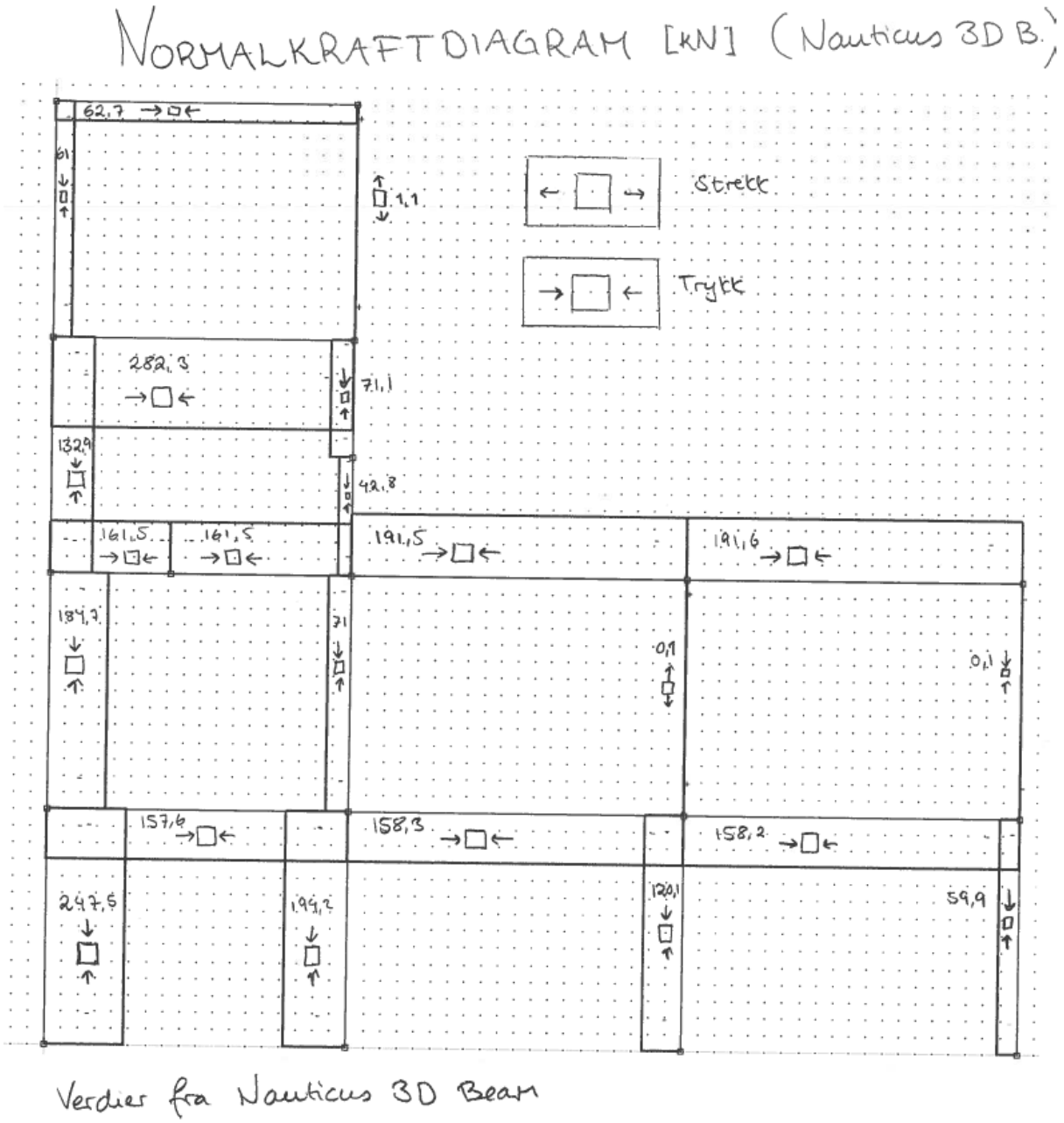
## 3.4 Diagrammer




*Figur 6. Momentdiagram basert på MATLAB-resultater.*




*Figur 7. Skjærkraftdiagram basert på MATLAB-resultater.*




*Figur 8. Normalkraftdiagram basert på resultater fra Nauticus 3D Beam.*

# 4 Diskusjon

Rammekonstruksjonen i denne oppgaven skulle bestå av rør- og boksprofiler. Grunnet noe uklarhet i oppgaveteksten, og misforståelse fra vår side, endte vi opp med å bruke I-profiler istedenfor boksprofiler. Dette tenkte vi ikke over før vi skulle velge profil i Nauticus. På grunn av tidsmangel valgte vi å beholde funksjonene slik de var. Dette mener vi ikke har noe å si for hverken utfordring eller utbytte av prosjektet. I teorien vil det ikke være et problem å implementere valg av boks-profil inn i det allerede eksisterende programmet.

Da vi skulle sammenligne resultater gitt av programmeringen med verdiene gitt av Nauticus, fant vi dessverre relativt store avvik i endemomentene på elementer utsatt for punktlast. Vi tror dette skyldes feil i rammebetingelsene vi oppga i Nauticus, da programmet vårt analyserer den enkle konstruksjonen gitt i oppgave C, uten feil.

# 5 Konklusjon

Vi har laget et MATLAB-program som kan analysere en todimensjonal rammekonstruksjon ved å beregne endemomenter, midtmomenter, bøyemomenter og skjærkraft. Vi har brukt dette programmet til å analysere rammekonstruksjonen gitt i oppgaven. Hensikten med å analysere konstruksjonen var at vi skulle finne dimensjoner på rør- og i-profilene, slik at maksimal bøyespenning var mellom 30-70% av flytspenningen på 320MPa. Dette var en itereringsprosess som til slutt gav oss IPE 450, og rør med diameter på 450mm og tykkelse på 20 mm. Element nummer 3 er det kritiske elementet, og fikk en maksimal bøyespenningen på 50,7% av flytspenningen. Vi kontrollerte resultatene med Nauticus, og resultatene stemmer relativt godt overens, med unntak av elementer påvirket av punktlast. Vi har også kontrollert at programmet fungerer på andre todimensjonale rammekonstruksjoner. Dette gjorde vi ved å kjøre programmet med en mindre, todimensjonal ramme, og deretter sammenligne resultatene mot beregninger gjort for hånd. Disse var identiske, og vi har derfor klart å kode et fungerende program.

Programmet vårt har dessverre en del begrensninger – matrisemetoden neglisjerer tverr- og aksialdeformasjoner, programmet kan ikke regne midtmomenter for konstruksjoner der elementer blir påvirket av både fordelt last og punktlast samtidig. Likevel er vi godt fornøyd med det endelige resultatet, og selve MATLAB-programmet.

# 6 Gruppeevaluering

Dette har vært et stort prosjekt som har krevd mye tid og energi. Dårlige forkunnskaper i MATLAB gjorde at vi måtte bruke veldig mye tid på å sette oss inn i selve programmeringen før vi kunne begynne å kode. Prosjektoppgaven ble utdelt 05.10.14 med leveringsfrist 03.11.14. Vi begynte å samles allerede den første uken og fordelte oppgaver vi kunne gjøre

individuelt. Dette viste seg derimot å ikke være like effektivt som å jobbe sammen i gruppe, noe vi opererte med den siste tiden.

For å holde orden på kodingen i MATLAB var det best å gjøre dette i fellesskap. Vi var alltid to som kodet, mens tredjemann skrev rapporten parallelt. Dette gjorde at det var enklere å holde styr på selve oppsettet og diverse variabler. Vi byttet på hvem som gjorde hva slik at alle fikk innsikt i de forskjellige arbeidsoppgavene.

Til tross for advarsler fra eldre studenter om å sette av god tid til rapporten, undervurderte vi tidsmengden en rapport av dette omfanget tar. Vi ble derfor nødt til å avse flere timer enn planlagt fram mot innleveringsfristen. Et skippertak en Mannhulitt verdig.

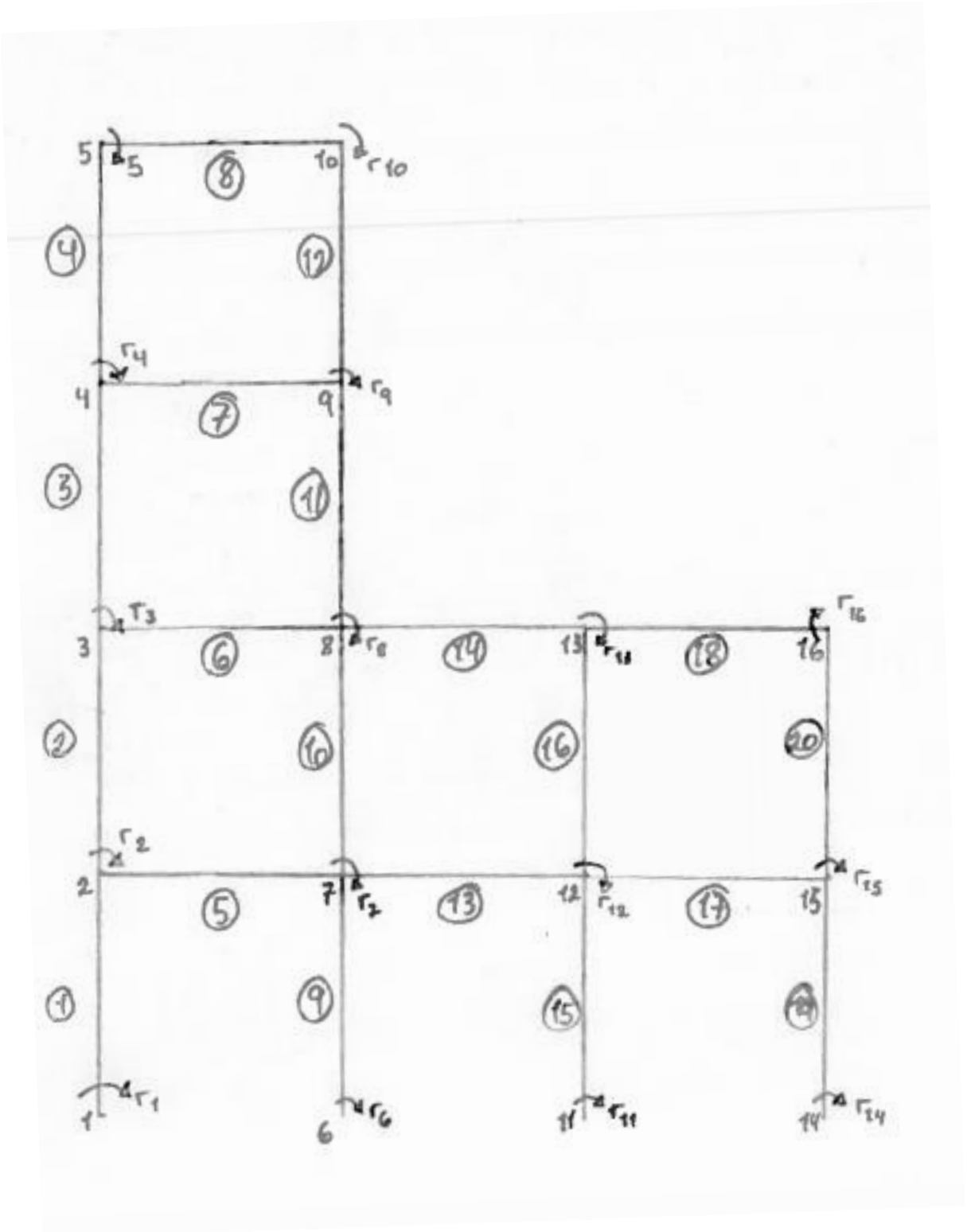
Vi har utnyttet hverandres, og andres, kunnskaper, samtidig som vi har brukt mye tid på å sette oss inn i fagstoffet dette prosjektet baserer seg på. Vi føler denne kombinasjonen, bestående av faglig forståelse og godt samarbeid, har gitt oss et størst mulig læringsutbytte.

# 7 Referanser

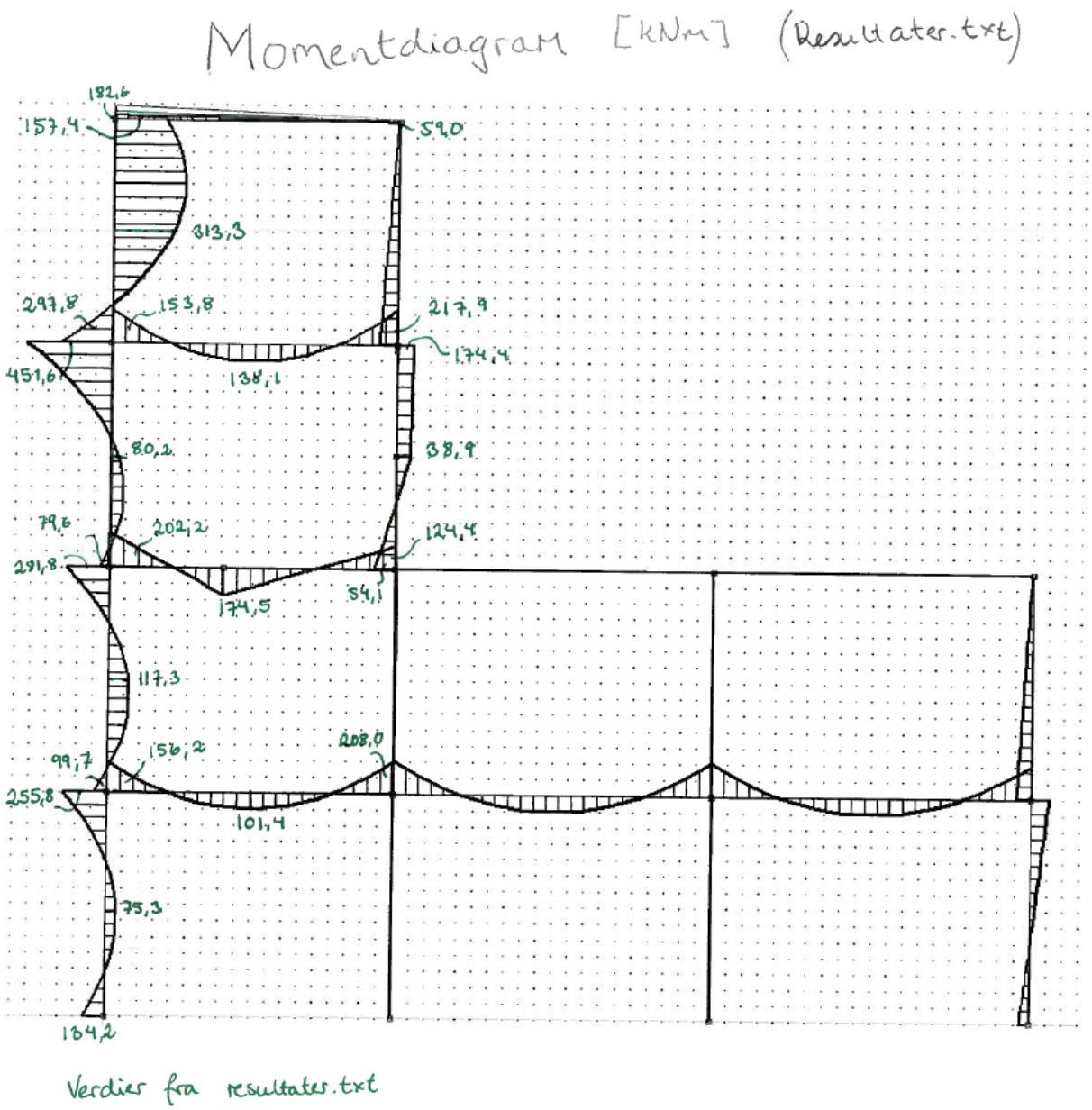

1. Amdahl, J., *TMR4167 Marin Teknikk 2 Del 1: Konstruksjonsanalyse*, Institutt for Marin Teknikk, 2010
2. Irgens, F., *Formelsamling mekanikk*, 5. Opplag, Tapir Akademiske Forlag, 3. Utgave, 2010.
3. Amdahl, J., *Karaktergivende Prosjektoppgave TMR4167 Marin Teknikk 2*, Oktober 2014

# APPENDIX A – Diagrammer og beregninger

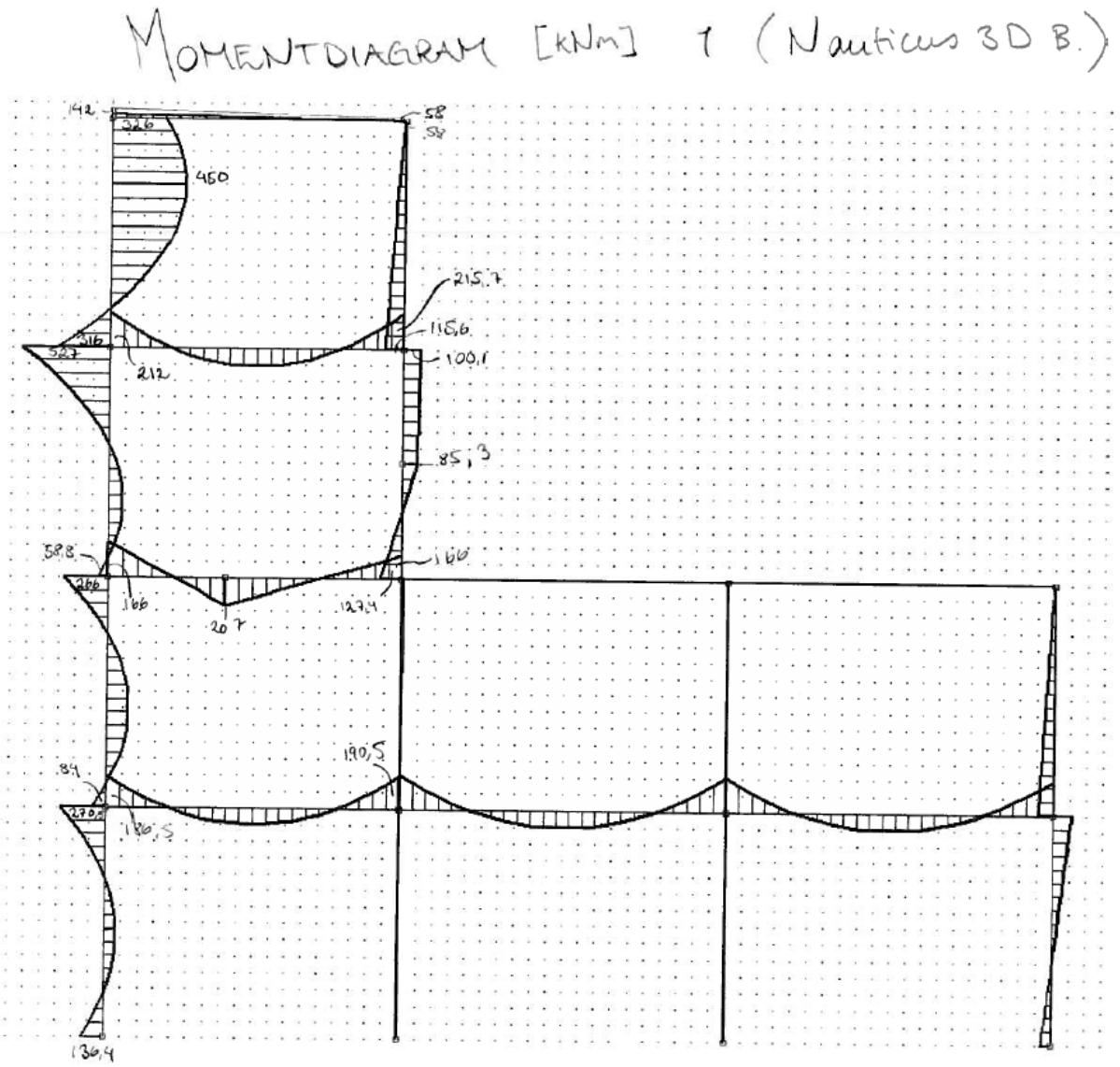
## A. 1 Diskretisering



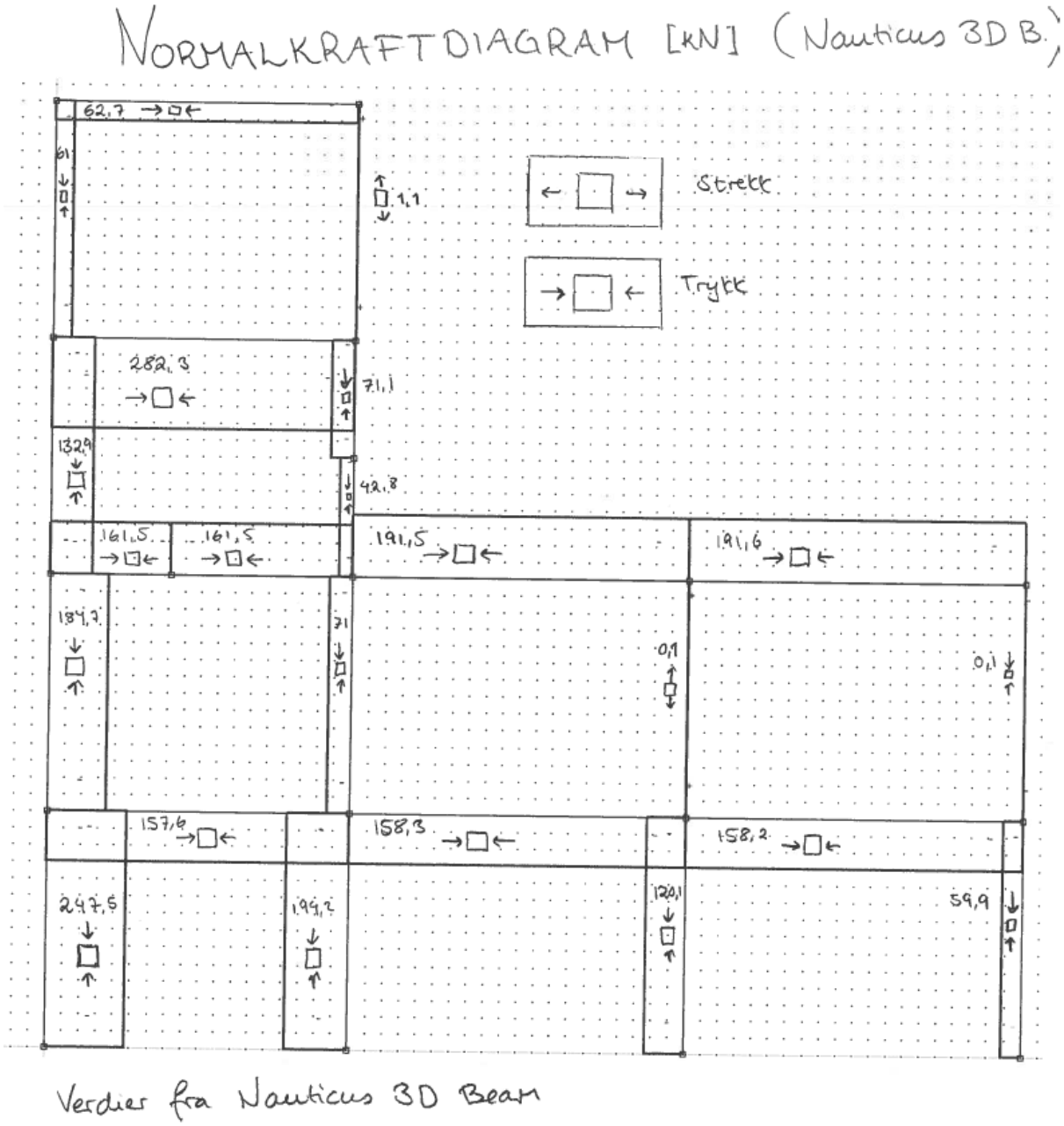
## A. 2 Momentdiagram basert på resultater.txt



## A. 3 Momentdiagram basert på Nauticus 3D Beam

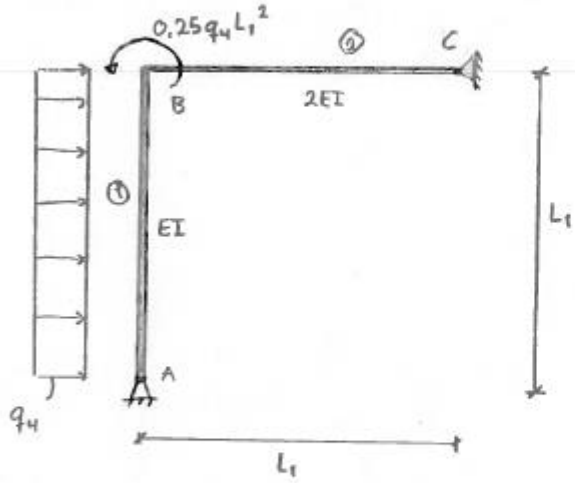


## A. 4 Momentdiagram basert på Nauticus 3D Beam



## A. 5 Håndberegninger av oppgave C

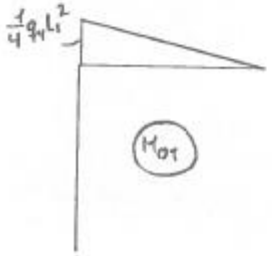
Skal berekne bøgemoment ved alle bjelkeendar og midt på bjelken med fordelt last for ramma under.



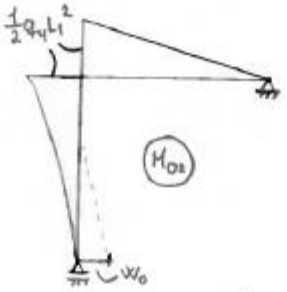

$q_4 = 10\ \frac{kN}{m}$

$L_1 = 18m$

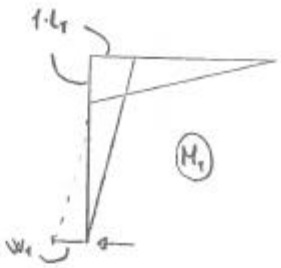
Brukar enhetslastmetoden:




Momentbidrag frå moment i C.




Momentbidrag frå fordelt last. Tek bort $A_x$




Setter på ei enheitslast i A.

$$w_{01} = \int \frac{M_{01} M_1}{EI} dx = -\frac{1}{3} \cdot \frac{\frac{1}{4} q_4 L_1^2 \cdot L_1 \cdot L_1}{2EI} = -\frac{1}{24} \frac{q_4 L_1^4}{EI}$$

$$w_{02} = \int \frac{M_{02} M_1}{EI} dx = -\frac{1}{4} \cdot \frac{\frac{1}{2} q_4 L_1^2 \cdot L_1 \cdot L_1}{EI} - \frac{1}{3} \cdot \frac{\frac{1}{2} q_4 L_1^2 \cdot L_1 \cdot L_1}{2EI} = -\frac{5}{24} \frac{q_4 L_1^4}{EI}$$

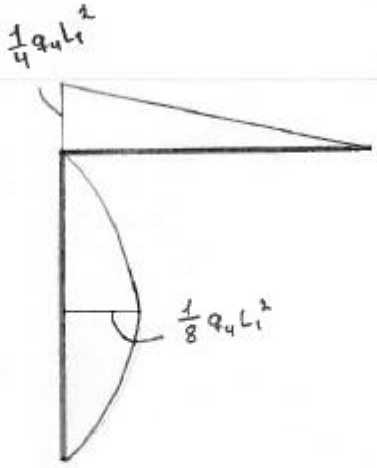
$$w_0 = w_{01} + w_{02} = \left(-\frac{1}{24} - \frac{5}{24}\right) \frac{q_4 L_1^2}{EI} = -\frac{1}{4} \frac{q_4 L_1^4}{EI}$$

$$w_1 = \int \frac{M_1^2}{EI} dx = \frac{1}{3} \cdot \frac{L_1 \cdot L_1 \cdot L_1}{EI} + \frac{1}{3} \cdot \frac{L_1 \cdot L_1 \cdot L_1}{2EI} = \frac{1}{2} \frac{L_1^3}{EI}$$

Kompabilitet: $w_0 + w_1 \cdot A_x = 0 \Rightarrow A_x = -\frac{w_0}{w_1}$

$$A_x = -\frac{-\frac{1}{4} \frac{q_4 L_1^4}{EI}}{\frac{1}{2} \frac{L_1^3}{EI}} = \frac{1}{2} q_4 L_1$$

Momentdiagram:




Vi har at:

$q_4 = 10 \frac{kN}{m}$

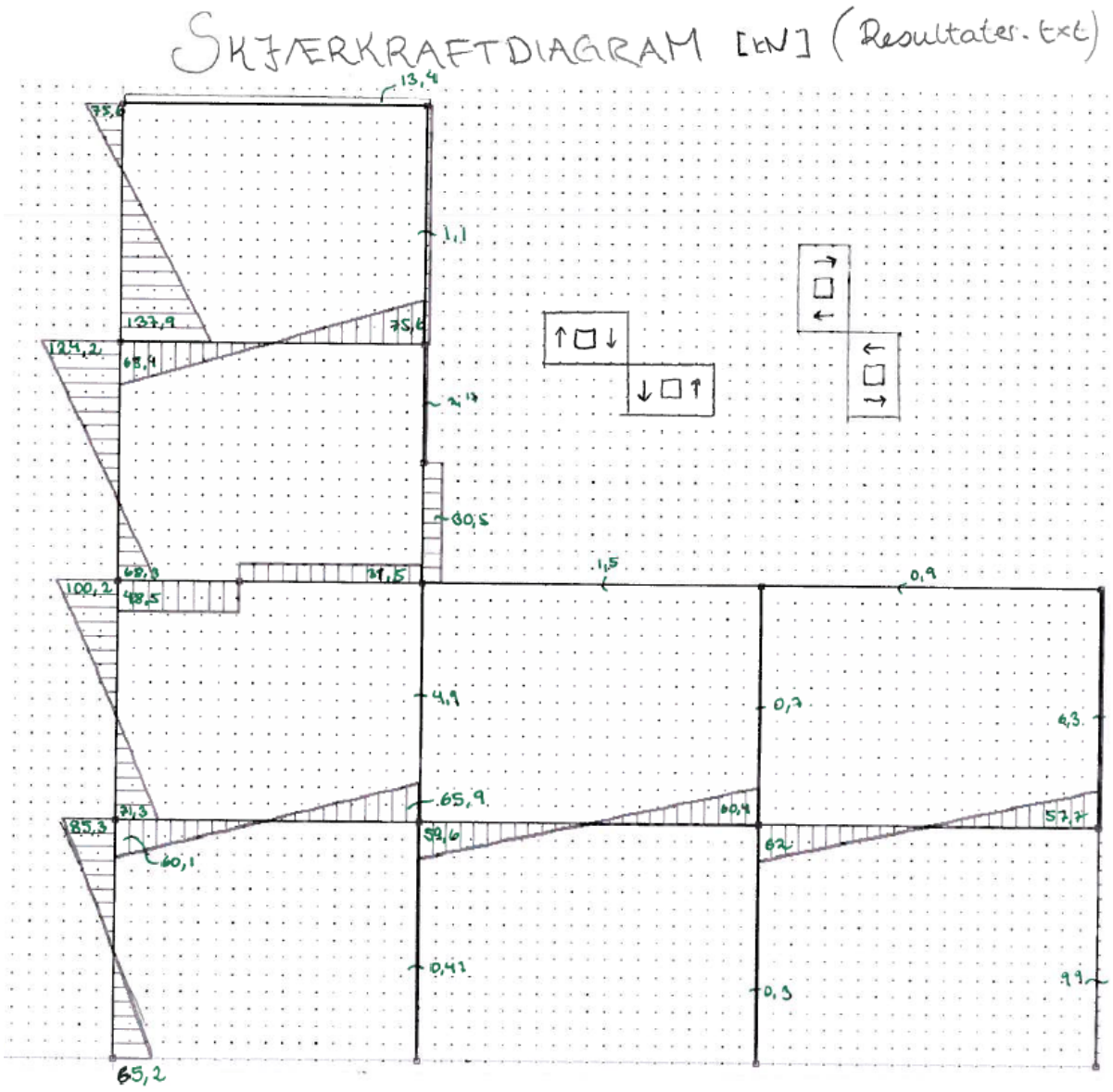
$L_1 = 18 m$

$\Rightarrow$

$\frac{1}{4} q_4 L_1^2 = \frac{10}{4} \frac{kN}{m} \cdot (18m)^2 = 810 kNm$

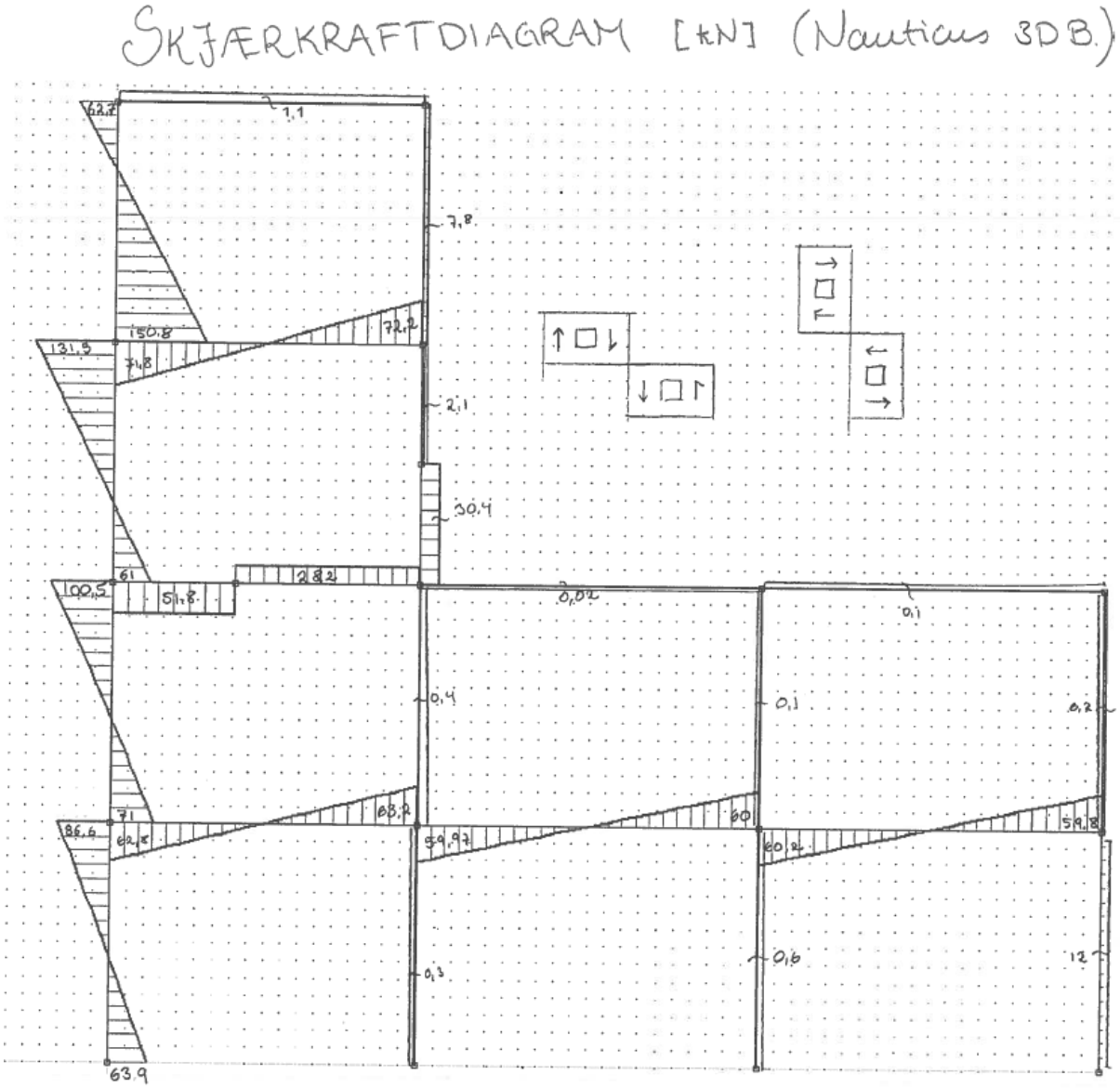
$\frac{1}{8} q_4 L_1^2 = \frac{10}{8} \frac{kN}{m} \cdot (18m)^2 = 405 kNm$

## A. 6 Skjærkraftdiagram basert på resultater.txt




Verdier fra resultater.txt

## A. 7 Skjærkraftdiagram basert på Nauticus 3D Beam



# APPENDIX B – MATLAB-funksjoner

## B. 1 annetarealmoment_iprofil.m

```
function [I_iprofil, y_global] = annetarealmoment_iprofil(h, t_bunnflens,
t_toppflens, t_steg, b_bunnflens, b_toppflens)

%Her regner vi ut annet arealmoment (IPE-profiler)

%Finner arealene til toppflens, bunnflens og steg, samt lengde til lokale
%arealsentre

a_toppflens = b_toppflens * t_toppflens;
a_bunnflens = b_bunnflens * t_bunnflens;
a_steg = (h - t_toppflens - t_bunnflens) * t_steg;
y_toppflens = h - t_toppflens/2;
y_bunnflens = t_bunnflens/2;
y_steg = h/2;
a_tot = a_toppflens + a_bunnflens + a_steg;

%Hvis arealet på toppflensen er ulik arealet på bunnflensen, må vi regne ut
%plasseringen til arealsenteret

    if ((t_bunnflens ~= t_toppflens)||(b_bunnflens ~= b_toppflens))
        yc = (a_toppflens * y_toppflens + a_bunnflens * y_bunnflens + y_steg *
a_steg)/a_tot;


        if yc > (h-yc)

            y_global = yc;

        else
            y_global = h-yc;

        end

    else
        yc = h/2;


        y_global = yc;

    end

    %Regner ut annet arealmoment og legger til Steiners for hver del.
    %Vi får totalt annet arealmoment ved å summere disse.

    am1 = ((t_toppflens^3)*b_toppflens)/12 + (h-yc-t_toppflens/2)^2 * a_toppflens;
    am2 = ((t_bunnflens^3)*b_bunnflens)/12 + (h-yc-t_bunnflens/2)^2 * a_bunnflens;
    am3 = (((h-t_toppflens-t_bunnflens)^3)*t_steg)/12 + ((y_steg-yc)^2 * a_steg);

    I_iprofil = (am1 + am2 + am3) * 10^(-12);


end
```

*Published with MATLAB® R2014b*

## B. 2 annetarealmoment_ror.m

```
function [I_ror, y] = annetarealmoment_ror(d,t)

%Her regner vi ut annet arealmoment for et rør med diameter (d) og tykkelse
%(t)
%Begynner med å finne radius

r = d/2;

%Så regner vi ut annet arealmoment

I_ror = ((pi*((r^4)-(r-t)^4))/4)*10^(-12);

%Finner y, da denne skal brukes senere til å beregne bøyespenning
y = r;


end
```

*Published with MATLAB® R2014b*

## B. 3 beregn_midtmoment.m

```
function midtmoment = beregn_midtmoment(endemoment, nelem, n_fordelt_last,
fordelt_last, amplituder_q, elementlengder, n_punkt_last, punkt_last)

%Her legger vi sammen midtmoment fra fordelt-, og punktlast elementvis i
%en vektor. Dette kan vi gjøre hvis lastene (hhv. punkt og fordelt) ikke er på
samme bjelke

midt_punkt = midtmoment_punktlast(n_punkt_last, elementlengder, nelem, punkt_last,
endemoment);

midt_fordelt = midtmoment_fordelt(endemoment, nelem, n_fordelt_last, fordelt_last,
amplituder_q, elementlengder);

midtmoment = midt_punkt + midt_fordelt;

%Hvis vi får positivt fortegn vil strekksiden være mot venstre eller
%oppover


end
```

*Published with MATLAB® R2014b*

## B. 4 boyespenning.m

```
function spenning = boyespenning(endemoment, nelem, I, midt_moment, Y)

spenning = zeros(nelem,3);

    for i = 1:nelem

        %Begynner med å hente maksimal y
        y = Y(i) * 10^(-3);

        %Deretter regner vi ut spenning og legger denne elementvis inn i en matrise
        spenning(i,1) = endemoment(i,1) * y/I(i);
        spenning(i,2) = endemoment(i,2) * y/I(i);
        spenning(i,3) = midt_moment(i,1) * y/I(i);

    end

end
```

*Published with MATLAB® R2014b*

## B. 5 endeM.m

```
function [endemoment, rotasjonsmoment] = endeM(nelem, elem, bjelkestivhet,
rotasjon, fim)

%Totalt endemoment
endemoment = zeros(nelem,2);

%Bjelkeendemoment fra rotasjon
rotasjonsmoment = zeros(nelem,2);

%Dette bruker vi senere i beregningen av skjærkraft

    for i = 1:nelem

        %Først henter vi ut knutepunktnummer for elementet
        punkt1 = elem(i,1);
        punkt2 = elem(i,2);

        %Derettet henter vi ut rotasjonen
        vinkel1 = rotasjon(punkt1);
        vinkel2 = rotasjon(punkt2);

        %Så beregner vi bjelkeendemoment fra rotasjoner
        rotasjonsmoment(i,:) = bjelkestivhet(i) * [4*vinkel1 + 2*vinkel2, 2*vinkel1
+ 4*vinkel2];

        %Til slutt setter vi bjelkeendemomentene fra rotasjoner i en matrise,
sammen med
        %faastinnspenningsmomentene
        endemoment(i,:) = fim(i,:) + rotasjonsmoment(i,:);

    end

end
```

*Published with MATLAB® R2014b*

## B. 6 fast_innspent.m

```
function fim = fast_innspent (fordelt_last, n_fordelt_last, nelem, elementlengder,
n_punkt_last, punkt_last, amplituder_q)

%Først henter vi fastinnspenningsmomentene fra punktlastene
punktlast = moment_punktlaster (nelem, n_punkt_last, punkt_last, elementlengder);

%Deretter henter vi de fra fordelte laster
fordeltlast = moment_fordelt (fordelt_last, n_fordelt_last, elementlengder, nelem,
amplituder_q);

%Til slutt legger vi sammen de forskjellige lastene
fim = punktlast + fordeltlast;

end
```

*Published with MATLAB® R2014b*

## B. 7 input.txt

```
16
0 0 1
0 14 0
0 28 0
0 42 0
0 56 0
18 0 0
18 14 0
18 28 0
18 42 0
18 56 0
38 0 0
38 14 0
38 28 0
58 0 1
58 14 0
58 28 0
2
1 450 14.6 14.6 9.4 190 190
0 450 20 0 0 0 0
20
1 2 210 2
2 3 210 2
3 4 210 2
4 5 210 2
2 7 210 1
3 8 210 1
4 9 210 1
5 10 210 1
6 7 210 2
7 8 210 2
8 9 210 2
9 10 210 2
7 12 210 1
8 13 210 1
11 12 210 2
12 13 210 2
12 15 210 1
13 16 210 1
14 15 210 2
15 16 210 2
4
4
4 10000 16000 1 2 3 4
1 7000 7000 5 0 0 0
2 6000 6000 13 17 0 0
1 8000 8000 7 0 0 0
3
6 80000 7.2 0
11 -40000 7 45
8 60000 0 0
1
5 -340000
```

## B. 8 inputC.txt

```
3
0 0 0
0 18 0
18 18 0
1
0 500 20 0 0 0 0
21 2 210 1
2 3 420 1
1
1
1 10000 10000 1
0
1
2 -810000
```

## B. 9 lastvektor.m

```
function b = lastvektor (fim, ytre_moment, nelem, npunkt, elem)

b = zeros (npunkt, 1);

    for i = 1:nelem

        %Her henter  vi knutepunktsnummer for det aktuelle elementet
        punkt1 = elem (i,1);
        punkt2 = elem (i,2);

        %Vi endrer fortegnet på fastinnspenningsmomentene og setter det inn
        %i en lastvektor
        b(punkt1, 1) = b(punkt1,1) - fim(i,1);
        b(punkt2, 1) = b(punkt2,1) - fim(i,2);

    end

%Her legger vi til ytre moment
b = b + ytre_moment;

end
```

*Published with MATLAB® R2014b*

## B. 10 lengder.m

```
function svar=lengder(punkt,elem,nelem)


svar = 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);

   %Benytter pytagoras
   svar(i) = sqrt(dx*dx + dy*dy);

    end


end
```

*Published with MATLAB® R2014b*

## B. 11 lesinput.m

```
function [npunkt, punkt, nelem, elem, punkt_last, moment_last, fordelt_last,
n_punkt_last, n_fordelt_last, nmoment, nprofil, profil, n_fordelt_lengste] =
lesinput()

%i = heltall
%f = flyttall


%Åpner og leser inputfilen
filid = fopen('input.txt','r');

%Finner antall knutepunkter i rammen
npunkt = fscanf(filid, '%i', [1,1]);

%Leser kordinatene og hvorvidt knutepunktet er fast innspent
  %Kolonne 1: x-koordinater
  %Kolonne 2: y-koordinater
  %Kolonne 3: Grensebetingelse: fri rotasjon = 0, fast innspent = 1
punkt = fscanf(filid,'%f %f %i',[3,npunkt]);
punkt = transpose(punkt);

%Finner antall ulike bjelkeprofiler
nprofil = fscanf(filid, '%i', [1,1]);

%Leser profildimensjoner
  %Kolonne 1: Profiltype:  Rør = 0,       I-profil = 1
  %Kolonne 2: Dimensjon 1: Rør - radius,  I-profil - total høyde
  %Kolonne 3: Dimensjon 2: Rør -tykkelse, I-profil - tykkelse på bunnflens
  %Kolonne 4: Dimensjon 3: Rør = 0,       I-profil - tykkelse på toppflens
  %Kolonne 5: Dimensjon 4: Rør = 0,       I-profil - tykkelse på steg
  %Kolonne 6: Dimensjon 5: Rør = 0,       I-profil - bredde på bunnflens
  %Kolonne 7: Dimensjon 6: Rør = 0,       I-profil - bredde på toppflens
  profil = fscanf(filid, '%i %f %f %f %f %f %f', [7, nprofil]);
  profil = transpose (profil);

%Finner antall elementer i kontruksjonen
nelem = fscanf(filid,'%i',[1 1]);

%Leser inn elementenes konnektivitet, hvilket tilsier sammenhengen mellom
elementer, knutepunktsnummer, E-modul og
%bjelketversnitt. Elementnummer tilsvarer radnummer.
  %Kolonne 1: Globalt knutepunktnummer for lokal ende 1
  %Kolonne 2: Globalt knutepunktnummer for lokal ende 2
  %Kolonne 3: E-modul for materialet
  %Kolonne 4: Tverrsnittstype: rør(sirkulært) = 2, bjelke(rektangulært) = 1
elem = fscanf(filid,'%i %i %f %i',[4, nelem]);
elem = transpose(elem);

%Finner antall lineært fordelte laster
n_fordelt_last = fscanf(filid,'%i',[1, 1]);
```

```
%Leser maksimalt antall elementer den største fordelte lasten går over
%Dette er viktig for å få riktig matrisedimensjon
n_fordelt_lengste = fscanf(filid,'%i',[1, 1]);

%Lager en string med variabelsymboler
nelement_string = '';
    for i = 1:n_fordelt_lengste;
        nelement_string = [nelement_string, ' %i'];
    end

%Leser lastdata for lineære laster
  %Kolonne 1:       Antall elementer lasten går over
  %Kolonne 2:       Laststr. i knutepunktet lengst ned og til venstre
  %Kolonne 3:       Laststr. i knutepunktet lengst opp og til høyre
  %Kolonne 4 og ut: Elementene lasten går over
fordelt_last = fscanf(filid, ['%i %f %f', nelement_string], [n_fordelt_lengste + 3,
n_fordelt_last]);
fordelt_last = transpose(fordelt_last);

%Finner antall punktlaster
n_punkt_last = fscanf(filid, '%i', [1, 1]);

%Leser punktlaster
  %Kolonne 1: Aktuelt element
  %Kolonne 2: Størrelse (definert positiv for høyre og ned)
  %Kolonne 3: Avstand fra last til lokalt knutepunkt 1
  %Kolonne 4: Vinkel i forhold til bjelkenormalen [grader]
punkt_last = fscanf(filid, '%i %f %f %f', [4, n_punkt_last]);
punkt_last = transpose (punkt_last);

%Leser antall ytre momenter
nmoment = fscanf(filid, '%i', [1, 1]);

%Leser ytre momenter
  %Kolonne 1: Knutepunkt
  %Kolonne 2: Størrelse (definert positivt med klokken)
moment_last = fscanf(filid, '%i %f', [nmoment, 2]);

%Lukker inputfilen
fclose(filid);

end
```

*Published with MATLAB® R2014b*

## B. 12 midtmoment_fordelt.m

```
function moment = midtmoment_fordelt(endemoment, nelem, n_fordelt_last,
fordelt_last, amplituder_q, elementlengder)

moment = zeros(nelem,1);

%Vi har valgt å oppgi de lineært fordelte lastene som elementene de går over.
%Dette gjør at vi trenger en nøstet for-løkke

    % i = lastnummer, j = elemetnummer i last
    for i = 1:n_fordelt_last

        %Antall elementer den aktuelle lasten går over
        nelementer_last = fordelt_last(i,1);

        %Henter data om den aktuelle lasten
        for j = 1:nelementer_last

            %Elementnummer
            elemnr = fordelt_last(i,j + 3);

            %Lastintensitet, ende 1
            q1 = amplituder_q(elemnr,1);

            %Lastintensitet, ende 2
            q2 = amplituder_q(elemnr,2);

            %Lengde
            L = elementlengder(elemnr);

            if q1 == q2 %Hvilket betyr rektangulær last

                %m1: Eksakt moment for rektangulær last
                m1 = -q1 * L^2/8;

            else %Trapes

                %Ved trapeslast tilnærmer vi oss problemet ved å splitte
                %lasten i to trekanter. Dette blir ikke helt eksakt,
                %men litt større enn maksimalt moment
                m1 = -q1 * L^2/(9 * sqrt(3)) - q2 * L^2/(9 * sqrt(3));

            end

            %m2: Moment på midten grunnet endemomenter
            m2 = -(endemoment(elemnr,1) - endemoment(elemnr,2))/2;

            %Summerer m1 og m2, som deretter settes elementvis i en vektor
            moment(elemnr,1) = moment(elemnr,1) + m1 + m2;

        end
```

```
    end
end
```

*Published with MATLAB® R2014b*

## B. 13 midtmoment_punktlast.m

```
function moment = midtmoment_punktlast(n_punkt_last, elementlengder, nelem,
punkt_last, endemoment)

moment = zeros(nelem,1);

    %Henter data for aktuell last.
    %Oppererer med samme variabelnavn som Marin 2 kompendiet

    for i = 1:n_punkt_last

        elemnr = punkt_last(i,1);
        L = elementlengder(elemnr);
        a = punkt_last(i,3);
        b = L - a;
        P = cos(punkt_last(i,4) * pi/180) * punkt_last(i,2);

        %Moment fra punktlasten der punktlasten virker
        m1 = -P * a * b/L;

        %Moment fra endemomenter der punktlasten virker
        m2 = (-endemoment(elemnr,1) * b + endemoment(elemnr,2) * a)/L;

        moment(elemnr,1) = m1 + m2 + moment(elemnr,1);

    end

end
```

*Published with MATLAB® R2014b*

## B. 14 moment_fordelt.m

```
function moment = moment_fordelt(fordelt_last, n_fordelt_last, elementlengder,
nelem, amplituder_q)

%Finner fastinnspenningsmomentenefra alle de lineære lastene

moment = zeros (nelem,2);

        %Regner ut momenter fra de fordelte lastene og setter de inn i en matrise

        %For lastnummer i og elemtnummer j i last
        for i = 1:n_fordelt_last

            %Finner antall elementer som lasten går over
            nelementer_last = fordelt_last(i,1);

            for j = 1:nelementer_last

            %Henter elementnummer
            elemnr = fordelt_last(i, 3 + j);

            %Henter laststørrelse
            q1 = amplituder_q(elemnr,1);
            q2 = amplituder_q(elemnr,2);

            %Henter lengden på elementet
            lengde = elementlengder(elemnr,1);

            %Fastinnspeningsmomentene fra lineært fordelte laster regnes ut
            %som to trekantlaster. Henter informasjon fra tabell i
            %kompendiet

            %Moment ende 1
            m12_1 = (-1/20) * q1 * lengde^2;
            m12_2 = (-1/30) * q2 * lengde^2;

            %Moment ende 2
            m21_1 = (1/30) * q1 * lengde^2;
            m21_2 = (1/20) * q2 * lengde^2;

            %Setter disse momentene inn i en matrise
            moment(elemnr,:) = [m12_1 + m12_2, m21_1 + m21_2];

            end

        end

end
```

*Published with MATLAB® R2014b*

## B. 15 moment_punktlaster.m

```
function moment = moment_punktlaster(nelem, n_punkt_last, punkt_last,
elementlengder)

% Regner ut fastinnspenningsmomenter skapt av punktlaster

moment = zeros(nelem,2);

    for i = 1:n_punkt_last
        %Gir størrelse fra punktlastmatrisen en variabel

        %Elementnummer
        elementnr = punkt_last(i,1);

        %Laststørrelse
        last = punkt_last(i,2);

        %Elementlengde
        total_lengde = elementlengder(elementnr);

        %Lastens avstand fra knutepunkt 1
        avstand_a = punkt_last(i,3);

        %Lastens avstand fra knutepunkt 2
        avstand_b = total_lengde - avstand_a;

        %Punktlastens vinkel ifh til normalen [grader]
        last_vinkel = punkt_last(i,4);



        %Regner ut størrelse på moment i ende 1
        %Bruker formelen: -p*a*(b^2)/l^2
        m_a = -last * cos(last_vinkel *pi/180) * avstand_a *
avstand_b^2/total_lengde^2;

        %Regner ut størrelsen på moment i ende 2
        %Bruker formelen: p*a^2*b/l^2
        m_b = last * cos(last_vinkel * pi/180) * avstand_a^2 *
avstand_b/total_lengde^2;

        %Momentene settes inn i vektoren
        moment(elementnr,1) = moment(elementnr,1) + m_a;
        moment(elementnr,2) = moment(elementnr,2) + m_b;

    end

end
```

*Published with MATLAB® R2014b*

## B. 16 moment_ytremoment.m

```
function moment = moment_ytremoment (npunkt, nmoment, moment_last)

%Setter de ytre momentene inn i en vektor, knutepunktvis
%Tar input fra inputfila vår
moment = zeros(npunkt,1);

for i = 1:nmoment
   moment(moment_last(i,1)) = moment(moment_last(i,1)) + moment_last(i,2);
end
end
```

*Published with MATLAB® R2014b*

## B. 17 q_elementer.m

```
function Q = q_elementer(nelem, elementlengder, fordelt_last, n_fordelt_last)

%Regner ut mellomamplituder på de fordelte lastene som går over
%flere elementer

Q = zeros(nelem,2);

    % i = lastnummer, j = elementnummer for last
    for i = 1:n_fordelt_last

        %Avstand fra det i-ende elementet til element nummer 1

        lengde_elem1 = 0;

        %Henter startlast
        q1 = fordelt_last(i,2);

        %Henter ut antall elementer den aktuelle lasten går over
        nelementer_last = fordelt_last(i,1);

        %Regner ut den totale lengden i-ende last går over
        total_lengde = 0;

        for k = 1:nelementer_last

            %Henter elementnummer fra lastvektor
            elemnr = fordelt_last(i,3 + k);

            %Legger sammen lengder
            total_lengde = total_lengde + elementlengder(elemnr);

        end

        %Regner ut amplitude for endepunktene til hvert element

        for j = 1:nelementer_last

            %Henter ut elementnummer
            elemnr = fordelt_last(i, 3 + j);

            %Henter ut lengden til aktuelt element
            lengde = elementlengder(elemnr);

            %Regner ut avstanden til det første elementet
            lengde_elem1 = lengde_elem1 + lengde;

            %Regner ut q2, lastamplituden i motsatt ende av q1, for
            %elementet
            q2 = qfordelt(i, lengde_elem1, fordelt_last, total_lengde);
```

```
            %Setter inn i outputmatrisen
            Q(elemnr,1) = q1;
            Q(elemnr,2) = q2;

            %Setter q1(startamplitude) for det neste elementet
            q1 = q2;

        end

    end
end
```

*Published with MATLAB® R2014b*

## B. 18 qfordelt.m

```
function Q = qfordelt(i, var_lengde, fordelt_last, total_lengde)

%Regner ut amplituden på lasten i en avstand var_lengde fra
%startpunktet

%Henter startamplitude
qstart = fordelt_last(i,2);

%Henter endeamplitude
qslutt = fordelt_last(i,3);

%Regner ut stigningstall for lasten
stigning = (qslutt - qstart)/total_lengde;

%egnes Q-verdi ved en variabel lengde
Q = qstart + stigning * var_lengde;

end
```

*Published with MATLAB® R2014b*

## B. 19 rammeanalyse.m

```
% Sletter alle variabler
clear all


%Leser input-data
[npunkt, punkt, nelem, elem, punkt_last, moment_last, fordelt_last, n_punkt_last,
n_fordelt_last, nmoment, nprofil, profil, n_fordelt_lengste] = lesinput();


%Regner lengder til elementene
elementlengder = lengder(punkt, elem, nelem);

%Regner ut mellomamplituder for de lastene som går over flere elementer
amplituder_q = q_elementer(nelem, elementlengder, fordelt_last, n_fordelt_last);

%Fastinnspenningsmomentene
fim = fast_innspent(fordelt_last, n_fordelt_last, nelem, elementlengder,
n_punkt_last, punkt_last, amplituder_q);

%Sorterer ytre moment knutepunktsvis
ytre_moment = moment_ytremoment (npunkt, nmoment, moment_last);

%Setter opp lastvektor
b = lastvektor(fim, ytre_moment, nelem, npunkt, elem);

%Setter opp en vetor for elementenes bjelkestivhet (EI/L), annet
%arealmoment (I), tversnittets lengste avstand til arealsenteret (Y)
[bjelkestivhet, I, Y] = stivhet_vektor (elem, elementlengder, nelem, npunkt,
profil);


%Setter opp systemstivhetsmatrisen
K = stivhetsmatrise(bjelkestivhet, elem, npunkt, nelem);


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


%Løser ligningssytemet
rotasjon = Kn\Bn;

%Finner endemoment for hvert element
[endemoment, rotasjonsmoment] = endeM(nelem, elem, bjelkestivhet, rotasjon, fim);

%Regner ut momentet på midten av bjelker med fordelt last og punktlaster
midt_moment = beregn_midtmoment(endemoment, nelem, n_fordelt_last, fordelt_last,
amplituder_q, elementlengder, n_punkt_last, punkt_last);

%Regner ut den maksimale bøyespenningen for endemomenter og midtmoment
```

```
spenning = boyespenning(endemoment, nelem, I, midt_moment, Y);

%Henter ut det maksimale momentet og elementet det virker på.
%Denne størrelsen er dimensjonerende
[maksmomentvektor, radplass] = max(abs(spenning));
[maksmoment, id] = max(maksmomentvektor);
maks_elementnummer = radplass(id);

%Regner elementvis ut skjærkraft for hver ende
skjaerkrefter = skjaer(nelem ,rotasjonsmoment, elementlengder, n_punkt_last,
punkt_last, n_fordelt_last, fordelt_last, amplituder_q);

%Skriver ut resultatene til filen resultat.txt
filid1 = fopen('resultat.txt', 'w');

fprintf (filid1, 'RESULTATER\n\n');
fprintf (filid1, 'Maksimal bøyespenning: %6.3f[MPa]\nPå elementnummer: %i\n\n',
maksmoment*10^(-6), maks_elementnummer);

    for i = 1:nelem

        %Moment på midten som uten ytre last er ikke 0, men heller ikke
        %dimensjonerende. Derfor skriver vi de ut som 'ikke dimensjonerende'

        if midt_moment(i,1) == 0
            dimensjon_sp = 'Ikke dimensjonerende';
            dimensjon_M = 'Ikke dimensjonerende';

        else
            dimensjon_sp = num2str(spenning(i,3) * 10^(-6));
            dimensjon_M = num2str(midt_moment(i,1) * 10^(-3));

        end

fprintf(filid1, ('Element %i\n\n Momenter\n Ende 1: %6.3f[kNM]\n Ende 2:
%6.3f[kNm]\n Midten: %s[kNm]\n Bøyespenning\n Ende 1: %6.3f[MPa]\n Ende 2:
%6.3f[MPa]\n Midten: %s[MPa]\n Skjærkraft\n Ende 1: %6.3f[kN]\n Ende 2:
%6.3f[kN]\n\n'), i, endemoment(i,:) * 10^(-3), dimensjon_M,spenning(i,1:2) * 10^(-
6), dimensjon_sp, skjaerkrefter(i,:) * 10^(-3));

    end

disp('Resultater ligger i tekstdokumentet resultater.txt')
```

```
Resultater ligger i tekstdokumentet resultater.txt
```

---

*Published with MATLAB® R2014b*

## B. 20 resultat.txt

```
Maksimal bøyespenning: 162.369[MPa]
På elementnummer: 3

Element 1

Momenter
 Ende 1: -134.232[kNM]
 Ende 2: 255.835[kNm]
Midten: -75.2944[kNm]
 Bøyespenning
 Ende 1: -48.262[MPa]
 Ende 2: 91.983[MPa]
Midten: -27.0714[MPa]
 Skjærkraft
 Ende 1: 65.164[kN]
 Ende 2: -85.336[kN]

Element 2

Momenter
 Ende 1: -99.653[kNM]
 Ende 2: 281.791[kNm]
Midten: -117.3265[kNm]
 Bøyespenning
 Ende 1: -35.829[MPa]
 Ende 2: 101.315[MPa]
Midten: -42.1837[MPa]
 Skjærkraft
 Ende 1: 71.340[kN]
 Ende 2: -100.160[kN]

Element 3

Momenter
 Ende 1: -79.582[kNM]
 Ende 2: 451.600[kNm]
Midten: -80.1774[kNm]
 Bøyespenning
 Ende 1: -28.613[MPa]
 Ende 2: 162.369[MPa]
Midten: -28.8271[MPa]
 Skjærkraft
 Ende 1: 68.277[kN]
 Ende 2: -124.223[kN]

Element 4

Momenter
 Ende 1: -297.789[kNM]
 Ende 2: -157.361[kNm]
Midten: -313.2747[kNm]
 Bøyespenning
 Ende 1: -107.067[MPa]
 Ende 2: -56.578[MPa]
Midten: -112.6352[MPa]
 Skjærkraft
 Ende 1: 137.861[kN]
```

```
 Ende 2: -75.639[kN]
Element 5Momenter
 Ende 1: -156.182[kNM]
 Ende 2: 208.017[kNm]
Midten: -101.4003[kNm]
 Bøyespenning
 Ende 1: -109.336[MPa]
 Ende 2: 145.623[MPa]
Midten: -70.9856[MPa]
 Skjærkraft
 Ende 1: 60.120[kN]
 Ende 2: -65.880[kN]

Element 6

Momenter
 Ende 1: -202.208[kNM]
 Ende 2: 124.362[kNm]
Midten: -174.5302[kNm]
 Bøyespenning
 Ende 1: -141.557[MPa]
 Ende 2: 87.060[MPa]
Midten: -122.1803[MPa]
 Skjærkraft
 Ende 1: 48.485[kN]
 Ende 2: -31.515[kN]

Element 7

Momenter
 Ende 1: -153.811[kNM]
 Ende 2: 217.910[kNm]
Midten: -138.1394[kNm]
 Bøyespenning
 Ende 1: -107.676[MPa]
 Ende 2: 152.549[MPa]
Midten: -96.7048[MPa]
 Skjærkraft
 Ende 1: 68.439[kN]
 Ende 2: -75.561[kN]

Element 8

Momenter
 Ende 1: -182.639[kNM]
 Ende 2: -59.040[kNm]
Midten: 182.6393[kNm]
 Bøyespenning
 Ende 1: -127.857[MPa]
 Ende 2: -41.331[MPa]
Midten: 127.8572[MPa]
 Skjærkraft
 Ende 1: 13.427[kN]
 Ende 2: 13.427[kN]

Element 9

Momenter
 Ende 1: -0.000[kNM]
 Ende 2: 6.529[kNm]
Midten: Ikke dimensjonerende[kNm]
```

```
 Bøyespenning Ende 1: -0.000[MPa]
 Ende 2: 2.347[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: -0.466[kN]
 Ende 2: -0.466[kN]

Element 10

Momenter
 Ende 1: -18.755[kNM]
 Ende 2: -50.568[kNm]
Midten: Ikke dimensjonerende[kNm]
 Bøyespenning
 Ende 1: -6.743[MPa]
 Ende 2: -18.181[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: 4.952[kN]
 Ende 2: 4.952[kN]

Element 11

Momenter
 Ende 1: -54.130[kNM]
 Ende 2: -174.370[kNm]
Midten: 38.8745[kNm]
 Bøyespenning
 Ende 1: -19.462[MPa]
 Ende 2: -62.693[MPa]
Midten: 13.977[MPa]
 Skjærkraft
 Ende 1: 2.179[kN]
 Ende 2: 30.464[kN]

Element 12

Momenter
 Ende 1: -43.540[kNM]
 Ende 2: 59.040[kNm]
Midten: Ikke dimensjonerende[kNm]
 Bøyespenning
 Ende 1: -15.654[MPa]
 Ende 2: 21.227[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: -1.107[kN]
 Ende 2: -1.107[kN]

Element 13

Momenter
 Ende 1: -195.791[kNM]
 Ende 2: 203.723[kNm]
Midten: -100.2429[kNm]
 Bøyespenning
 Ende 1: -137.064[MPa]
 Ende 2: 142.617[MPa]
Midten: -70.1753[MPa]
 Skjærkraft
 Ende 1: 59.603[kN] Ende 2: -60.397[kN]
```

Element 14
Momenter
 Ende 1: -19.664[kNM]
 Ende 2: -9.711[kNm]
Midten: Ikke dimensjonerende[kNm]
 Bøyespenning
 Ende 1: -13.766[MPa]
 Ende 2: -6.798[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: 1.469[kN]
 Ende 2: 1.469[kN]

Element 15

Momenter
 Ende 1: -0.000[kNM]
 Ende 2: 4.502[kNm]
Midten: Ikke dimensjonerende[kNm]
 Bøyespenning
 Ende 1: -0.000[MPa]
 Ende 2: 1.619[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: -0.322[kN]
 Ende 2: -0.322[kN]

Element 16

Momenter
 Ende 1: 6.228[kNM]
 Ende 2: 3.451[kNm]
Midten: Ikke dimensjonerende[kNm]
 Bøyespenning
 Ende 1: 2.239[MPa]
 Ende 2: 1.241[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: -0.691[kN]
 Ende 2: -0.691[kN]

Element 17

Momenter
 Ende 1: -214.453[kNM]
 Ende 2: 167.856[kNm]
Midten: -108.8452[kNm]
 Bøyespenning
 Ende 1: -150.128[MPa]
 Ende 2: 117.508[MPa]
Midten: -76.1974[MPa]
 Skjærkraft
 Ende 1: 62.330[kN]
 Ende 2: -57.670[kN]

Element 18

Momenter Ende 1: 6.260[kNM]
 Ende 2: 12.278[kNm]
Midten: Ikke dimensjonerende[kNm]

```
 Bøyespenning
 Ende 1: 4.382[MPa]
 Ende 2: 8.595[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: -0.927[kN]
 Ende 2: -0.927[kN]

Element 19

Momenter
 Ende 1: -46.205[kNM]
 Ende 2: -92.410[kNm]
Midten: Ikke dimensjonerende[kNm]
 Bøyespenning
 Ende 1: -16.613[MPa]
 Ende 2: -33.225[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: 9.901[kN]
 Ende 2: 9.901[kN]

Element 20

Momenter
 Ende 1: -75.446[kNM]
 Ende 2: -12.278[kNm]
Midten: Ikke dimensjonerende[kNm]
 Bøyespenning
 Ende 1: -27.126[MPa]
 Ende 2: -4.414[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: 6.266[kN]
 Ende 2: 6.266[kN]
```

## B. 21 resultatC.txt

```
Maksimal bøyespenning: 232.733[MPa]
På elementnummer: 2

Element 1

Momenter
 Ende 1: -0.000[kNM]
 Ende 2: -0.000[kNm]
Midten: -405[kNm]
 Bøyespenning
 Ende 1: -0.000[MPa]
 Ende 2: -0.000[MPa]
Midten: -116.3665[MPa]
 Skjærkraft
 Ende 1: 90.000[kN]
 Ende 2: -90.000[kN]

Element 2

Momenter
 Ende 1: -810.000[kNM]
 Ende 2: 0.000[kNm]
Midten: Ikke dimensjonerende[kNm]
 Bøyespenning
 Ende 1: -232.733[MPa]
 Ende 2: 0.000[MPa]
Midten: Ikke dimensjonerende[MPa]
 Skjærkraft
 Ende 1: 45.000[kN]
 Ende 2: 45.000[kN]
```

## B. 22 skjaer.m

```
function skjaerkrefter = skjaer(nelem, rotasjonsmoment, elementlengder,
n_punkt_last, punkt_last, n_fordelt_last, fordelt_last, amplituder_q)

%Regne elementvis ut skjærkreftene ved endene.
%Skjærretning med urviseren gir positivt fortegn

skjaerkrefter = zeros(nelem,2);

    for i = 1:nelem

        %Regner ut -(Mij+Mji)/L for hvert element
        Q_1 = -(rotasjonsmoment(i,1) + rotasjonsmoment(i,2)) / elementlengder(i,1);

        %Legger til i matrise
        skjaerkrefter(i,:) = skjaerkrefter(i,:) + [Q_1, Q_1];

    end

%Regner ut Q0 for elementer med punktlast

    for i = 1:n_punkt_last

        %Henter laststyrke (P), avstand fra ende 1 (a), elementnummer
        %(elemnr), og lengde på elementer (L)
        P = cos(punkt_last(i,4) * pi/180) * punkt_last(i,2);
        a = punkt_last(i,3);
        elemnr = punkt_last(i,1);
        L = elementlengder(elemnr,1);


        if a > 0 && a < L

            %Bruker momentlilevekt til å regne ut skjærkrefter i endene
            Q_01 = [P*((L-a)/L) , -P*(a/L)];

        %Ønsker ikke laster som virker i knutepunktene, disse blir nullet ut
        else
            Q_01 = zeros(1,2);

        end

        %Legger til skjærkrefter i matrisen
        skjaerkrefter(elemnr,:) = skjaerkrefter(elemnr,:) + Q_01;

    end


    %Regner ut Q0 for elemener med fordelt last
    %i = antall laster, j = elementer per last
    for i = 1:n_fordelt_last
```

```
        for j = 1:fordelt_last(i,1)

            %Henter elementnummer (elemnr), lastintensitet i ende 1
            %(amplitude1), lastintensitet i ende 2 (amplitude2), og lengde
            %på elementet (L)
            elemnr = fordelt_last(i,j+3);
            amplitude1 = amplituder_q(elemnr,1);
            amplitude2 = amplituder_q(elemnr,2);
            L = elementlengder(elemnr,1);

            %Bruker momentlikevekt til å regne ut skjærkrefter i hver ende
            Q_02 = L * [(1/3) * amplitude1 + (1/6)*amplitude2, -(1/3) * amplitude2
- (1/6) * amplitude1];

            %Legger til skjærkrefter i matrisen
            skjaerkrefter(elemnr,:) = skjaerkrefter(elemnr,:) + Q_02;

        end

    end

end
```

*Published with MATLAB® R2014b*

## B. 23 stivhet_vektor.m

```
function [stivhet, I, Y] = stivhet_vektor (elem, elementlengder, nelem, npunkt,
profil)

% Regner ut stivheten (EI/L), annet arealmoment (I), og lengste avstand til
% arealsenteret i et tverrsnitt (Y) for hvert moment

stivhet = zeros (npunkt, 1);
I = zeros (nelem, 1);
Y = zeros (nelem, 1);


    for i = 1:nelem
        %E-modul
        E = elem(i,3)*10^9;
        %Elementlengde
        L = elementlengder(i,1);
        %Tversnittstype
        tverr_type = elem(i,4);


            if profil (tverr_type,1) == 0
                %Rørprofil
                [I(i), Y(i)] = annetarealmoment_ror(profil(tverr_type,2),
profil(tverr_type,3));

            else
                %I-profil
                [I(i), Y(i)] = annetarealmoment_iprofil(profil(tverr_type,2),
profil(tverr_type,3), profil(tverr_type,4), profil(tverr_type,5),
profil(tverr_type,6), profil(tverr_type,7));

            end

       %Regner ut og lagrer stivheten i vektoren
       stivhet(i,1) = E*I(i)/L;

    end

end
```

*Published with MATLAB® R2014b*

## B. 24 stivhetsmatrise.m

```
function K = stivhetsmatrise (stivhetsvektor, elem, npunkt, nelem)

K = zeros (npunkt, npunkt);

%Stivhetsmatrise for hvert enkelt element

k = [4,2;2,4];

    for i = 1:nelem
        %Henter ut knutepunktsnummer for bjelkeender
        punkt1 = elem(i,1);
        punkt2 = elem(i,2);

        %Legger til i k-matrisen for hvert element i den globale
        %stivhetsmatrisen
        K(punkt1,punkt1) = K(punkt1, punkt1) + stivhetsvektor(i) * k(1,1);
        K(punkt2,punkt1) = K(punkt2, punkt1) + stivhetsvektor(i) * k(2,1);
        K(punkt1,punkt2) = K(punkt1, punkt2) + stivhetsvektor(i) * k(1,2);
        K(punkt2,punkt2) = K(punkt2, punkt2) + stivhetsvektor(i) * k(2,2);

    end

end
```

*Published with MATLAB® R2014b*