
Rammeanalyse med matrisemetoden



Institutt for Marin Teknikk

TMR4167 Marin teknikk 2

5. november 2017

Forfattere:

10024, 10048, 10076, 10011

NTNU: Norges teknisk-naturvitenskapelige universitet

5. november 2017

Forord

Denne rapporten er skrevet for å presentere resultatene fra karaktergivende prosjektoppgave i TMR4167 Marin teknikk 2 - Konstruksjoner. Oppgaven er gjennomført og skrevet i gruppe, der alle har lagt inn like mye tid- og arbeidsmengde. Gruppen ble bestående av 4 personer isteden for 3, da medlemmer fra to grupper ble slått sammen til en. Grunnen til dette var frafall fra faget i begynnelsen av prosjekteringsfasen. Sammenslåingen ble godkjent av faglærer.

Rapporten analyserer rammekonstruksjoner ved hjelp av dataprogrammene MATLAB og Nauticus 3D Beam. I tillegg har noen beregninger og skisser av diagrammer, blitt utført for hånd. Beregninger av ulike tverrsnittsdimensjoner som ellers har blitt utført for hånd, har denne gangen blitt innført i dataprogrammer. Dette har gitt et nytt perspektiv på analyse av rammekonstruksjoner. På denne måten har det også vært mulig å sammenligne resultater fra skriftlige beregninger med resultater fra dataprogrammene som ble brukt.

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

Ligninger og eksempler fra forelesningene, kompendiet i TMR4167 og Irgens formelsamling, brukes for å løse deloppgavene i prosjektet. I tillegg har studentassistentene vært til god hjelp i faget.

Sammendrag

Hensikten med prosjektet var å få trening i å utføre beregningsoppgaver med MATLAB og få innsikt i oppbygningen av dataprogrammer for å analysere rammekonstruksjoner.

Prosjektoppgaven gikk ut på å lage et MATLAB-program som kunne analysere en vilkårlig todimensjonal rammekonstruksjon med matrisemetoden. Programmet skulle brukes til å analysere et utsnitt av et plattformdekk, og finne passende tverrsnittsdimensjoner. Det ble gitt en idealisert regnemodell av en stålramme i en modul på dekket av en offshore plattform. Rammen er antatt til å være uforskyelig. Øverste dekk av rammen er et lagerområde, hvor lasten modelleres som jevnt fordelt (q_3). Dekket under skal bære vekten av annet utstyr (P_1, P_2, q_4). Samtidig er rammen utsatt for sterke vindkrefter (lineært fordelt last over høyden med intensitet fra q_1 til q_2), vinden påvirker også en mast øverst til venstre slik at det oppstår et vindmoment (M_1). Konstruksjonen skulle bestå av sirkulære rørtverrsnitt i de vertikale søylene, og I-profiler i de horisontale bjelkene.

Enhetslastmetoden og matrisemetoden ble benyttet for å sammenligne resultatene fra beregningene gjort for hånd og beregningene gjort i MATLAB-programmet som ble laget. Til slutt skulle moment-, aksial- og skjærkraftdiagram tegnes med utgangspunkt i valgte tverrsnittsdimensjoner.

Når MATLAB-programmet skal sjekkes oppdages det feil. Vi får ikke programmet til å regne like resultater som det vi gjør for hånd. Programmet gir ut fysisk logiske resultater både for en enkel testfil og for den fulle rammekonstruksjonen, men rotasjonene og midtmomentene er feil i forhold til håndberegningene. Med fysisk logiske menes at det er store og små moment der det fysisk vil være det og likevekt der det skal være det. På grunn av tidsbegrensninger ble ikke feilen(e) funnet før prosjektet skulle ferdigstilles, men vi vet at feilen må ligge i utregningene og ikke i logikken, i forbindelse med rotasjoner og midtmomenter.

Innhold

| | |
|---|----|
| Forord | i |
| Sammendrag | ii |
| Figurliste | vi |
| Tabelliste | vi |
| 1 Introduksjon | 1 |
| 1.1 Beskrivelse av oppgaven | 1 |
| 1.2 Teori | 1 |
| 1.2.1 Matrisemetoden | 1 |
| 1.2.2 Diskretisering | 2 |
| 1.2.3 Elementanalyse | 2 |
| 1.2.4 Systemstivhetsmatrise | 3 |
| 1.2.5 Elementkrefter | 3 |
| 1.2.6 Annet arealmoment | 4 |
| 2 Tverrsnittsdimensjoner og data | 5 |
| 2.1 Valgte tverrsnittsdimensjoner | 5 |
| 2.2 Data for konstruksjonen | 5 |
| 3 Oppgaver | 7 |
| 3.1 Oppgave a) | 7 |
| 3.2 Oppgave b) | 7 |
| 3.3 Oppgave c) | 8 |
| 3.3.1 Implementasjon | 8 |
| 3.3.2 Resultater | 9 |
| 3.4 Oppgave d) | 10 |
| 3.4.1 Nauticus 3D Beam | 10 |
| 3.4.2 Fremgangsmåte | 11 |
| 3.5 Oppgave e) | 11 |
| 4 Oppsummering og videre arbeid | 13 |
| 5 Appendix | A |
| A Oppgave C: Håndberegninger | A |
| B Oppgave E: MATLAB-resultat | B |
| C Oppgave E: Håndberegninger | C |
| D Oppgave E: Håndberegninger | D |
| E Oppgave E: Håndberegninger | E |
| F Oppgave E: Håndberegninger | F |
| G Resultater | F |
| G.1 Resultater oppgave c | G |
| G.2 Resultater Hovedoppgaven | H |
| H Kode | I |
| H.1 Main | I |
| H.2 lesinput | K |
| H.3 2.Arealmoment | M |
| H.4 lengder | O |
| H.5 Elemtstivhet | P |

| | | |
|------|---------------------------------------|---|
| H.6 | Amplituder ved fordelt last | Q |
| H.7 | moment | R |
| H.8 | lastvektor | T |
| H.9 | stivhet | U |
| H.10 | bc | V |
| H.11 | endeM | W |
| H.12 | Bøyemoment pga fordelt last | X |
| H.13 | Bøyemoment pga punktlast | Y |
| H.14 | Spennin | Z |
| H.15 | Spennin test | |
| H.16 | Ytre moment | |
| H.17 | input | |
| H.18 | input oppgave c | |

Tabeller

| | | |
|---|--|---|
| 1 | Konnektivitetsmatrise | 2 |
| 2 | Tverrsnittsdimensjoner, rør-profil | 5 |
| 3 | Tverrsnittsdimensjoner, I-profil | 5 |
| 4 | Data for konstruksjonen | 6 |

Figurer

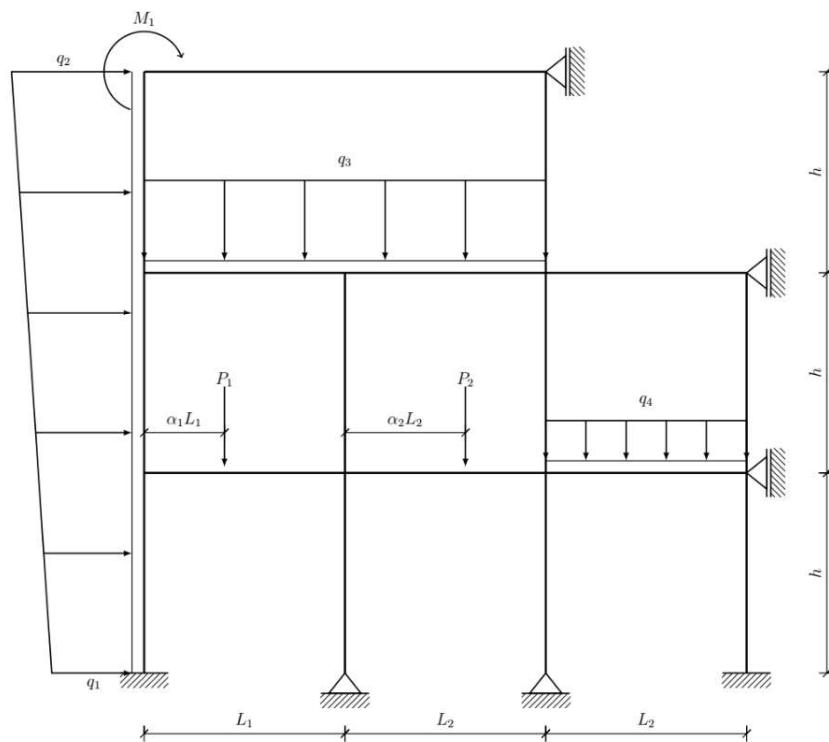
| | | |
|---|--|---|
| 1 | Rammekonstruksjon | 1 |
| 2 | Diskretisering | 2 |
| 3 | Profiler med tilhørende parametere | 5 |
| 4 | Diskretisering av skipsramme | 9 |

1 Introduksjon

1.1 Beskrivelse av oppgaven

Oppgaven går ut på å lage et enkelt MATLAB-program, som gjør at man kan analysere en konstruksjon med matrisemetoden. Programmet skal være dimensjonsløst.

I oppgaven tas det utgangspunkt i en modell av en stålramme i en modul på dekket av en offshore plattform, som vi ser i figur 1. Rammen antas å være uforskyelig slik at deformasjonen er fullstendig beskrevet av knutepunktsrotasjonene. Øverste dekk av rammen er et lagerområde hvor lasten modelleres som jevnt fordelt last q_3 . Dekket under skal bære vekten av annet utstyr, modellert som P_1 , P_2 og q_4 . Samtidig er rammen utsatt for sterke vindkrefter, som kan antas lineært fordelt over høyden med intensitet på q_1 og q_2 ved henholdsvis bunnen og toppen av rammen. Vindmomentet fra en mast i det øvre venstre hjørnet modelleres som M_1 . De vertikale søylene i rammen består av sirkulære rørtverrsnitt, mens de horisontale bjelkene består av I-profil. Knutepunktene skal antas å være stive.



Figur 1: Rammekonstruksjon

1.2 Teori

1.2.1 Matrisemetoden

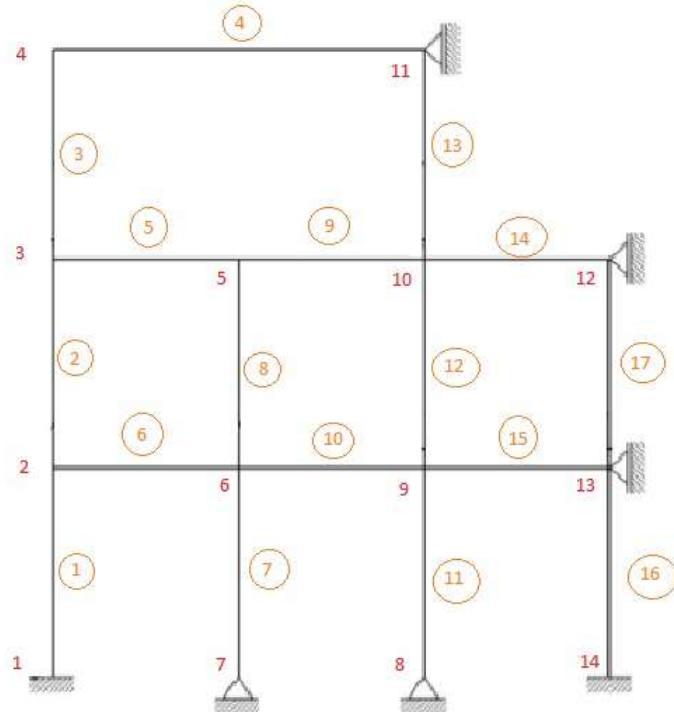
Matrisemetoden benyttes for å løse statisk ubestemte systemer og tar utgangspunkt i deformasjonsmetoden. Ved bruk av deformasjonsmetoden betraktes endepunktsforskyvningene og rotasjonene som ukjente, og ligninger stilles opp slik at krav til likevekt tilfredsstilles. Begge disse metodene er eksempler på datamaskinbaserte metoder. Forskjellen er at deformasjonsmetoden er mest egnet til å løse problemer der konstruksjoner kan deles inn i enkle elementer, mens matrisemetoden ser på systemet som helhet. Ved bruk av matrisemetoden går man gjennom enkelte trinn.

1.2.2 Diskretisering

Diskretisering vil si å dele opp konstruksjonen i et endelig antall elementer. Det er kjent at en konstruksjon kan bestå av knutepunkter og elementer. Knutepunktenes forskyvninger og rotasjoner defineres som systemets frihetsgrader. En måte å diskretisere konstruksjonen på er å nummerere de forskjellige knutepunktene og elementene. Dette er gjort som vist i figur 2. Ut i fra den diskretiserte modellen kan en konnektivitetsmatrise settes opp. En slik matrise viser sammenhengen mellom knutepunkt- og elementnummer. Konnektivitetsmatrisen er vist i tabell 1.

| Elementnummer | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---------------|---|---|---|----|---|---|---|---|----|----|----|----|----|----|----|----|----|
| Lokal node 1 | 1 | 2 | 3 | 4 | 3 | 2 | 7 | 6 | 5 | 6 | 8 | 9 | 10 | 10 | 9 | 14 | 13 |
| Lokal node 2 | 2 | 3 | 4 | 11 | 5 | 6 | 6 | 5 | 10 | 9 | 9 | 10 | 11 | 12 | 13 | 13 | 12 |

Tabell 1: Konnektivitetsmatrise



Figur 2: Diskretisering

1.2.3 Elementanalyse

Deretter må man etablere elementenes stivhetsmatriser:

$$\mathbf{S} = \mathbf{k}_i \mathbf{v}_i + \bar{\mathbf{S}} \quad (1)$$

\mathbf{S} = vektor av knutepunktskrefter og moment

\mathbf{k}_i = stivhetsmatrise for element i

\mathbf{v}_i = vektor av lokale knutepunktsforskyvninger

$\bar{\mathbf{S}}$ = vektor med ubalanserte knutepunktskrefter og moment

I vårt tilfelle vil de lokale knutepunktsforskyvningene ikke være forskyvelige i x- eller z-retning, og dermed vil v_i matrisen ikke bestå av u- eller w-forskyvninger i beregningene. Bjelkestivheten for alle elementene har det blitt dannet en kode for som vi kan finne i appendiks H.5.

1.2.4 Systemstivhetsmatrise

Elementstivhetsmatrisen er en matrise som oversiktlig viser stivheten i de ulike elementene. Denne etableres ved å beregne koeffisientene gitt ved

$$\frac{4EI_i}{L_i} = \frac{4EI}{L} = \text{konst for } i=1,2\dots \quad (2)$$

Systemstivhetmastrisen inkluderer stivheten til alle delene i systemet. Det er denne matrisen som hovedsakelig skiller deformasjonsmetoden og matrisemetoden. Fremgangsmåten består i å sette opp frihetsgrader med to indekser: én for å indikrere lokal bjelkeende og én for elementnummer. Likevekten er $\sum M = 0$ og kinematisk kompatibilitet gjelder. Et bidrag fra en bjelke med fire stivhetsledd kan da se ut som følger

$$k_i = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \quad (3)$$

k_{11} = stivhetsledd for ende 1 for forskyvninger i samme ende

k_{22} = stivhetsledd for ende 2 for forskyvninger i samme ende

k_{21} = koblingsledd mellom moment i ende 2 og rotasjon i ende 1

k_{22} = koblingsledd mellom moment i ende 1 og rotasjon i ende 2

Videre kan sammenhengen mellom lokale og globale frihetsgrader bli samlet i en konnektivitetsmatrise.

1.2.5 Elementkrefter

For å finne spenningsresultantene i hvert element, vil lokale stivhetsmatriser og lokale knutepunktsforskyvninger bli brukt. Dette krever at man regner på lokale forskyvninger og bruker lastvektoren til å finne systemrelasjonen. Lastvektoren inneholder konsentrert knutepunktskrefter og momenter, og knutepunktskrefter og moment som skyldes last i felt. Kode for lastvektoren, \mathbf{R} , er oppgitt i appendiks H.8. Systemrelasjonen vil dermed bli:

$$\mathbf{Kr} = \mathbf{R} \quad (4)$$

\mathbf{K} = stivhetsmatrise

\mathbf{r} = randbetingelser

\mathbf{R} = lastvektor

Det neste skrittet blir å innføre randbetingelser for systemet. I denne oppgaven er det laget en egen funksjon i MATLAB-programmet som innfører randbetingelsene for knutepunktsrotasjon. Videre blir det enklere å bestemme lokale forskyvningene for hvert bjelkeelement ettersom de kan finnes rett fra de globale frihetsgradene. Deretter benyttes formel 1 for å beregne lokale krefter fra forskyvninger og lokale fastinnspenningskrefter. Der $\bar{\mathbf{S}}$ betegner fastinnspenningskrefter.

1.2.6 Annet arealmoment

Annet arealmoment også kalt treghetsmoment, gir hvor utfordrende det er å sette et legeme i bevegelse. Konstruksjonen består av to forskjellige elementtyper, rør og I-profil. Annet arealmoment for et I-profil blir beregnet ved formel 5. Der hvor det andre ledet er Steiners sats.

$$I = \frac{bh^3}{12} + bhy^2 \quad (5)$$

b = profilbredde

h = profilhøyde

y = avstanden mellom globalt og lokalt arealsenter i I-profilen.

Annet arealmoment for et rørprofil blir beregnet ved formelen 6.

$$I = \frac{\pi}{4}(r^4 - (r-t)^4) \quad (6)$$

D = ytre diameter til røret

t = tykkelse til røret

Disse formlene er anvendt i MATLAB-programmet som vist i appendix. Før man finner treghetsmomentet finner koden hvilken elementtype man har - enten sirkulært tverrsnitt eller I-profil.

2 Tverrsnittsdimensjoner og data

2.1 Valgte tverrsnittsdimensjoner

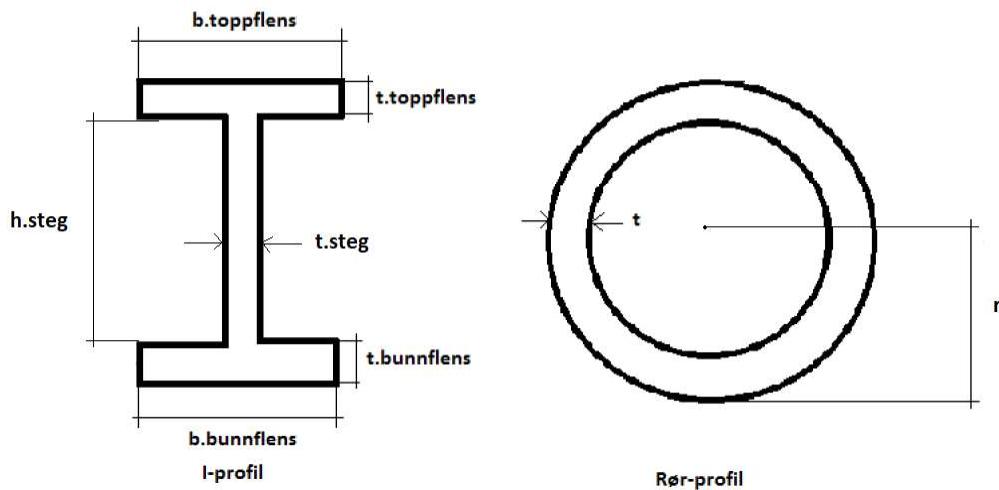
Ut i fra interasjon har tverrsnittsdimensjonene for konstruksjonens profiler blitt valgt. I tabell 2 og 3 er dimensjonene oppgitt. Navnforklaring ses fra figur 3.

| | |
|---|--------|
| r | 500 mm |
| t | 10 mm |

Tabell 2: Tverrsnittsdimensjoner, rør-profil

| | |
|-------------|--------|
| b.toppflens | 180 mm |
| t.toppflens | 26 mm |
| b.bunnflens | 180 mm |
| t.bunnflens | 26 mm |
| h.steg | 248 mm |
| b.steg | 14 mm |

Tabell 3: Tverrsnittsdimensjoner, I-profil



Figur 3: Profiler med tilhørende parametere

2.2 Data for konstruksjonen

Ut i fra de to siste sifrene i studentnummeret (XXXAB) til den ene på gruppa, blir data for konstruksjonen definert. Studentnummeret som er brukt sluttet på 41 (A=4 og B=1), og dataverdiene er som vist i tabell 4. Kreftene benevnes i [N], intensiteter i [N/mm] og lengder i [mm]. Navnforklaring ses fra figur 1.

| L_1 | L_2 | h | α_1 | α_2 | P_1 | P_2 | q_1 | q_2 | q_3 | q_4 | M_1 |
|-----------------|-----------------|-----------------|------------|------------|-----------------|-----------------|-------|-------|-------|-------|------------------|
| $18 \cdot 10^3$ | $20 \cdot 10^3$ | $12 \cdot 10^3$ | 0.6 | 0.4 | $90 \cdot 10^3$ | $60 \cdot 10^3$ | 10 | 16 | 15 | 12 | $200 \cdot 10^6$ |

Tabell 4: Data for konstruksjonen

3 Oppgaver

3.1 Oppgave a)

I den første oppgaven lages en funksjon som leser inn konstruksjonsdata som knutepunktskoordinater, elementenes tilknytning til knutepunktene, fordelte og konsentrerte laster til elementene, tverrsnittsdata for rør- og I-profil, grensebetingelser og materialdata. Det lages også en funksjon som beregner bøyestivheten for elementene på bakgrunn av tverrsnittsdata.

For at MATLAB-funksjonen H.2 lesinput skal kunne få tilgang til konstruksjonsdataen er det laget en input-fil H.17. Inputfila er strukturert slik at første del består av 14 knutepunkt. Knutepunktene er identifisert med x-koordinat i første kolonne og y-koordinat i andre. Origo i koordinatsystemet er satt til knutepunkt 1 nederst i venstre hjørne. I tredje kolonne er "1definert som fast innspent og "0er fritt opplagret og har mulighet for rotasjoner. Neste del består av 17 elementer. Elementene er identifisert med tilhørende knutepunkt 1 i første kolonne og tilhørende knutepunkt 2 i andre kolonne. I tredje kolone står E-modulen. Selv om den er konstant for konstruksjonen trenger vi den som input i oppgave c. Samt at det gjør MATLAB-programmet mer universalt ved at man kan andre materialer. For å bestemme om elementet består av I-profil eller sirkulære rørtverrsnitt har vi brukt verdiene 1 og 2 i fjerde kolonne. 1 representerer de horisontale I-profilene og 2 representerer de vertikale rørprofilene.

I neste del tar vi inn lastene. Først er det 3 jevnt fordelte laster. Første kolonne er antall elementer lasten går over. Andre kolonne består av lasten over knutepunktet som er nederst og til høyre, deretter er det lasten som er øverst og til venstre. De neste fire kolonnene er elementene lasten går over. Deretter tar vi inn punktlastene. Første kollonne er elementet lasten virker på, andre kolonne er størrelsen på lasten og tredje kolonne er verdien til alpha. Alpha påvirker avstanden fra elementets tilhørende knutepunkt 1. Laster som medvirker til positivt moment er definert som positive. Dette gjelder for alle typer laster.

I siste del i input-fila leser vi inn det ene påsatte momentet. Første kolonne er knutepunktet momenter er satt rundt og den andre kolonnen er størrelsen på momentet. Positivt moment defineres i retningen med klokka.

Lesinput-funksjonen tar inn informasjon om konstruksjonen fra input-fila, og behandler denne. Formålet med lesinput-filen er at informasjon om konstruksjonen skal gjøres brukbar i MATLAB-programmet. All informasjonen fra input-fila blir organisert og definert gjennom lesinput-funksjonen, slik at programmet lettere kan bruke opplysningene videre.

På bakgrunn av tverrsnittsdata skal bøyestivheten for elementene regnes ut. Krav fra oppgaven er at det skal være mulig å benytte ulik tykkelse på flensene og stegene til I-profilet. Elementene skal også kunne kobles til riktig geometri og materiale via geometri-og materialnummer i inputfila. Programmet leser et vilkårlig antall knutepunkt, elementer, laster, tverrsnittsdata og materialer.

Funksjonen H.5 elemStivhet beregner elementstivhetene. Den går gjennom alle elementene fra input-fila og identifiserer elementprofilene. Deretter regner den ut stivhetene for rør- og I-profilene, eller skriver ut error om det er et ukjent profil og stivheten blir 0.

3.2 Oppgave b)

I oppgave b) er de nødvendige funksjonene laget for å beregne elementstivhetsmatriser, fastinnspenningsmoment, lastvektor, oppbygging av systemstivhetsmatrise, beregning av bøyemoment. Funksjonene er implementert i hovedprogrammet som ble utdelt sammen med prosjektet. Nedenfor er funksjonene kort forklart og grensebetingelsene for knutepunktsrotasjon er gjort rede for.

Funksjonen H.9 stivhet beregner elementene i systemstivhetsmatrisen. Den tar elementstivhetene opprettet i elemStivhet, identifiserer ende 1 og ende 2 for hvert element og setter dem inn i en lokal elementstivhetsmatrise for det tilhørende elementet. Til slutt settes alle elementstivhetsmatrisene inn på riktig plass i den globale systemstivhetsmatrisen.

Fastinnspenningsmomentene skal kunne beregnes generelt for last som er lineært fordelt over elementet og punktlast med vilkårlig plassering fra ende 1. For å finne lasten i knutepunktene 1 og 2 for tilfeller der vi har jevnt fordelt last over et eller flere elementer er funksjonen H.6 q_KPkt laget. I prinsippet splitter den opp de jevnt fordelte lastene i to trekanner for så å finne amplitudene ved hver elementene. Den finner alle de fordelte lastene, antallet elementer lasten går over, hvilke elementer det er, lengden på elementene og totallengden til lasten. Deretter finner den amplituden i hovedende 1 og 2 og stigningstallet til lasten. Fastinnspenningsmomentene i enden av hvert element pga punktlast og fordelte laster er beregna i funksjonen H.7 moment. Funksjonen oppretter en vektor av ytre momenter i knutepunktet de virker i.

Funksjonen H.8 lastvektor setter sammen lastvektoren. Den tar inn fastinnspenningsmomentene og de ytre momentene og omgjør slik at vi har informasjon om hver node i stedet for hver ende av et element.

For beregningene av bøyemoment er det to bidrag - et fra fordelt last og et fra punktlast. Disse beregnes i hendholsvis funksjonene H.12 BoyForL og H.13 BoyPktL. Når bidraget fra de fordelte lastene skal beregnes deles lasten opp i to trekanner før momentet regnes ut på midten av hver trekant. Det totale bøyemomentet blir summen av begge midtmomentene minus summen av endemomentene.

Når bidraget fra punktlastene skal beregnes regner man ut momentet på midten når elementet er fritt opplagt og når elementet er fast innspent og summerer disse. Programmet kan ikke beregne bøyemomenter for tilfeller hvor en bjelke er utsatt for både punktlast og fordelt last.

Når det kommer til knutepunktsrotasjonene må grensebetingelser innføres, fordi knutepunkt med fast innspenning vil ikke ha rotasjon. Grensebetingelsene er valgt å innføres på følgende måte i funksjonen H.10 bc: Når et knutepunkt er fastholdt mot rotasjon skal tilhørende kolonne og linje nulles ut i systemstivhetsmatrisen. Diagonalelementet settes lik et vilkårlig tall. Tilhørende element i lastvektoren nulles også ut.

3.3 Oppgave c)

I denne oppgaven ble det tatt utgangspunkt i en enkel skipsramme som man ønsket å løse både for hånd med enhetslastmetoden og med MATLAB-programmet, slik at man kunne forsikre seg om at MATLAB-programmet fungerte korrekt.

3.3.1 Implementasjon

Diskretiseringen av rammen med knutepunkts- og elementnummering kan sees i figur 4. Vi antar at leseren er kjent med enhetslastmetoden og den er derfor ikke nøyere beskrevet, men beregningene finnes i appendiks A. Håndberegningen er for enkelhets skyld bare gjort for halve rammen. Dette kunne gjøres fordi rammen er symmetrisk. Rotasjonene er definert positive med klokka og lastene er definert positive når de går fra venstre til høyre og ovenfra og ned.

I MATLAB-programmet tas konstruksjonsdataene inn ved hjelp av en egen inputfil H.18, som er satt opp på samme måte som hovedprogrammet, slik det er beskrevet i avsnitt . Merk at elementene har to ulike stivheter, der stivheten i den horisontale bjelken er dobbelt så stor som i den vertikale bjelken. For enkelhets skyld er dette bare blitt tatt inn som to ulike E-moduler slik at den svakeste bjelken har E-module $210 \cdot 10^3$ [MPa] og den sterkeste $420 \cdot 10^3$ [MPa] noe vi anslo som plausible

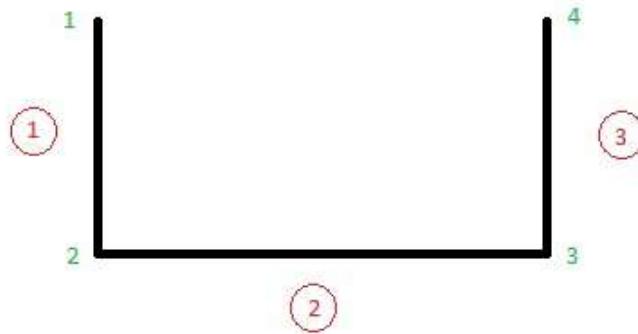
verdier. I hovedprogrammet finnes det en funksjon som beregner 2.arealmoment, men fordi dette bare er en test er 2.arealmoment satt like $100 \cdot 10^6$ for at det skal være lettere å regne. Dette er i samme størrelsesorden som de vi får når vi bruker tversnittene fra hovedoppgaven. Lengdene på elementene er henholdsvis $1 \cdot 10^3$ og $2 \cdot 10^3$ [mm]. Som tidligere beskrevet opererer programmet med millimeter og newton som standard inputenheter, men man kunne også ha brukt meter og kilonewton dersom man hadde vært konsekvent med enhetene i inputfilene. Oppgave c er levert i en egen mappe der de eneste forskjelle fra hovedoppgaven er at funksjonen for 2.arealmoment er fjernet og lesinput-funksjonen leser inn riktig inputfil H.2. I appendikset er kun koden for hovedprogrammet inkludert fordi forskjellene er så små.

3.3.2 Resultater

Våre resultater fra MATLAB-programmet finnes i appendiks G.1. Dessverre stemmer disse ikke overens med håndberegningene med enhetslastmetoden slik de er presentert i appendiks A. Selv om vi er trygge på at håndberegningene med enhetslastmetoden er riktige har vi likevel valgt å gå videre med programmet. Dette er gjort fordi resultatene logisk stemmer godt overens med det man skulle forvente, og vi derfor kan anta at ulikhettene i resultatene kommer av en mindre feil i koden.

Slik vi ser i appendiks G.1 er alle rotasjonene i samme størrelsesorden 10^{-6} , og de er naturlig nok symmetriske. Man kan også se at rotasjonene er meget små noe som stemmer godt overens med at vi har valgt sterke bjelker sammenliknet med lasten. Systemstivhetsmatrisen er også symmetrisk og midtmomentene på henholdsvis element 1 og 2 er like store, men med motsatt fortegn, slik det må være for en symmetrisk ramme. Midtmomentet i element 2 er også betraktlig større enn midtmomentene på de vertikale bjelkene, noe som må være rimelig da den er mer belastet.

Verdiene for endemomentene er også rimelige. De er ikke nøyaktig lik null slik man skulle ønske, men i en størrelsesorden 10^{-12} . Når vi nå har laster i [Nmm] er dette neglisjerbart lite, spesielt når vi sammenligner med endemomentene i rammehjørnene, som er i størrelsesorden 10^6 .



Figur 4: Diskretisering av skipsramme

Vi ser at

$$\frac{(EI)_1}{(EI)_2} \rightarrow 0 \quad (7)$$

medfører at

$$\begin{aligned}(EI)_1 &\rightarrow 0 \\ (EI)_2 &\rightarrow \infty\end{aligned}\tag{8}$$

og at

$$\frac{(EI)_1}{(EI)_2} \rightarrow \infty\tag{9}$$

medfører at

$$\begin{aligned}(EI)_1 &\rightarrow \infty \\ (EI)_2 &\rightarrow 0\end{aligned}\tag{10}$$

Det vil si at $\frac{EI_1}{EI_2} \rightarrow 0$ tilsvarer dette at skipssidene er uendelig mye sterkere enn bunnen. Med andre ord kan vi betrakte dette som at skipsbunnen er fast innspent og midtmomentet vil derfor gå mot $\frac{1}{24}q_0l^2$ hvor l er lengden av den horisontale bjelka og q_0 amplituden for den jevnt fordelt lasten.

Dersom $\frac{EI_1}{EI_2} \rightarrow \infty$ tilsvarer dette at skipsbunnen er uendelig mye sterkere enn sidene. Sidene vil derfor ikke ta opp moment og vi vil modellere bunnen som en fritt opplagt bjelke. Midmomentet vil derfor gå mot $\frac{1}{8}q_0l^2$ hvor l er lengden av den horisontale bjelka og q_0 amplituden for den jevnt fordelt lasten.

3.4 Oppgave d)

I oppgaven er det spesifisert at de vertikale søylene skal bestå av rør-profiler og de horisontale av I-profiler, men dimensjonene var hittil ukjente. I tillegg til dette er det oppgitt at flytespenningen i alle elementene er 320 [MPa] og at den største bøyespenningen på hvert horisontalplan skal være mellom 30% og 70% av flytespenningen. Vi sjekket spenningene ved hjelp av MATLAB-Programmet og Nauticus 3D-beam.

3.4.1 Nauticus 3D Beam

Nauticus 3D Beam er et modellerings- og analyseringsprogram gitt av DNV GL. Ved hjelp av denne programvaren kan man gjennomføre enkel modellering og analyser av konstruksjoner. I dette prosjektet er 3D Beam brukt for å dobbelsjekke MATLAB-programmet da tverrsnittsdimensjonene skulle bestemmes i oppgave d).

For å minimere avvik mellom MATLAB-programmet og Nauticus 3D Beam har vi satt at alle knutepunktene er i 3D Beam er fastholdt mot translasjon. Dette er fordi stivhetsmatrisene som benyttes i 3D beam er basert på at man analyserer en forsylig ramme noe som ikke er tilfelle i MATLAB-programmet. Bjelkeformuleringen som 3D Beam er basert på inkluderer bidrag fra skjærdeformasjoner, men i vårt prosjekt kan dette bidraget neglisjeres fordi bjelkene er slanke. For å fikse dette problemet settes skjærfaktorene lik 1000 når tverrsnittene defineres i Nauticus 3D Beam.

Vi tegnet opp rammen vår med tilhørende fordelt laster, punktlaster, momenter, samt valgte dimensjoner på rør- og I-profil. Da dette var gjort beregnet Nauticus 3D Beam ut momenter, skjærkrefter og aksialkrefter, og tegnet opp de tilhørende diagrammene.

3.4.2 Fremgangsmåte

Her oppsto det et dilemma for oss. Vi viste fra oppgave c at programmet vårt ikke stemte og at vi derfor ikke kunne forvente å få riktige verdier for spenningene. Vi hadde allerede modellert konstruksjonen i Nauticus 3D-beam og bestemte oss derfor for å dimensjonere tversnittene på bakgrunn av verdiene fra Nauticus 3D-beam i håp om at feilen i MATLAB-programmet ble rettet opp. Dimensjonene som ble brukt finnes i 2 og 3.

Å finne verdiene i Nauticus 3D Beam ble gjort på akkurat samme måte som vi ville gjort i MATLAB-programmet med å teste ulike tversnittsdimensjoner helt til vi fant noen som oppfylte de ønskede kraftene. MATLAB-programmet inneholder også en funksjon for å analysere spenningene ???. Når vi sammenlikner med Nauticus ser vi at spenningene er mye høyere når de blir regnet ut i MATLAB-programmet. Den maksimale spenningen som blir funnet i MATLAB-programmet er over dobbelt så høy som flytespenningen.

Vi ser derimot at de er logiske akkurat slik vi så i oppgave c. Det største endemomentet er i knutepunkt der den jevnt fordelt lasten virker over to elementer, midtmomentene er under de jevnt fordelt lastene er størst osv. Vi konkluderer derfor med at feilen i spenningene er en følgefeil av feilen i MATLAB som også førte til galt svar i oppgave c.

I MATLAB-programmet er det implementert en funksjon som sjekker om den maksimale spenningen er innenfor de gitte rammene. Denne er presentert i appendiks H.15. Egentlig skulle denne testet maks-spenningen på hvert plan, men fordi hele grunnen til at vi ønsker å teste det er for å kunne dimensjonere tversnittene regner programmet ut den maksimale spenningen totalt. Dette er tilstrekkelig i vår oppgave da profilene til alle de henholdsvis vertikale- og horisontale belkene er like. Med andre ord vil den maksimale spenningen på hvert plan uansett føre til at man måtte endre dimensjonene på de andre horisontale bjelkene også, dermed er dette overflødig å regne dem ut separat. Dette burde derimot vært implementert i et mer generelt program.

I oppgaven skal konstruksjonen fra figur 1 analyseres med MATLAB-programmet. Bøyespenningen og tversnittsdimensjonene avhenger av hverandre, og er avgjørende for flytespenningen. Det mest utsatte elementet skulle ha en maksimal bøyespennning i området 30-70 % av flytespenningen, som er gitt til å være 320 MPa i alle elementene. Største bøyespennning skal dermed ligge mellom 96 [MPa] og 224 [MPa]. Oppgaven informerer om å bruke I-profil i de horisontale bjelkene og rørprofil i de vertikale sylinderne. Tverrsnittsdimensjonene ble valgt i en iteringsprosess er gitt i tabellene 2 og 3 for henholdsvis rør- og I-profil.

3.5 Oppgave e)

I denne oppgaven skal momentdiagrammet tegnes for rammen på grunnlag av valgte tverrsnittsdimensjoner og outputverdier fra MATLAB-programmet. Ut i fra dette skal også aksial- og skjærkraftdiagram tegnes. Det er akseptert å kun angi momentverdien på midten av bjelker med fordelt last selv om maksimalverdien ikke befinner seg der.

Ettersom det er en ukjent feil i MATLAB-programmet har momentene for hele rammen også blitt regnet ut for hånd, slik at kanskje feilen ville vise seg. Dette var en lang og krevende prosess som viste seg å lønne seg. Resultatene viste oss at formen på momentdiagrammene stemmer overens og verdiene på endemomentene er tilnærmet like. I håndberegningene er det brukt litt andre tverrsnitt, fordi det ble gjort ørsmå endringer i etterkant, og det er derfor momentene ikke er helt like som i MATLAB-programmet. Midtmomentene stemmer ikke overens, og man kan derfor gå ut i fra at det er i forbindelse med denne feilen i programmet ligger. Selv om vi finner feil, ser vi likevel igjen at resultatene er logiske. Vi får store momenter på de elementene der det er fysisk logisk, og null

moment i bjelkeendene som ikke tar opp moment. Vi har også likevekt i rammehjørnene. Logikken bak MATLAB-programmet er dermed riktig.

Selve MATLAB-koden ligger som vedlegg i appendiks H og er meget nøyne kommentert inkludert en rute øverst i hver funksjon som nøyne beskriver hva funksjonen gjør og hvordan. I tillegg er selve Matrisemetoden beskrevet i teoridelen. Vi går derfor ikke i detalj på selve implementasjonen, men leseren oppfordres til å lese rutene dersom det skulle være uklarheter i hva de ulike underfunksjonene gjør.

Selv formlene for utregningene som er implementert er i all hovesak funnet i

$$M_x = \frac{gL\frac{L}{2}}{6} \left(1 - \frac{\frac{L}{4}}{L^2}\right) = \frac{gL^2}{12} \left(\frac{3}{4}\right) = \frac{qL^2}{16} \quad (11)$$

Eksempel på fremgangsmåten er vist på bjelke 5 i appendiks C.

Moment-, skjærkraft- og normalkraftdiagram for konstruksjonen finnes i appendiks B, D, E, og F.

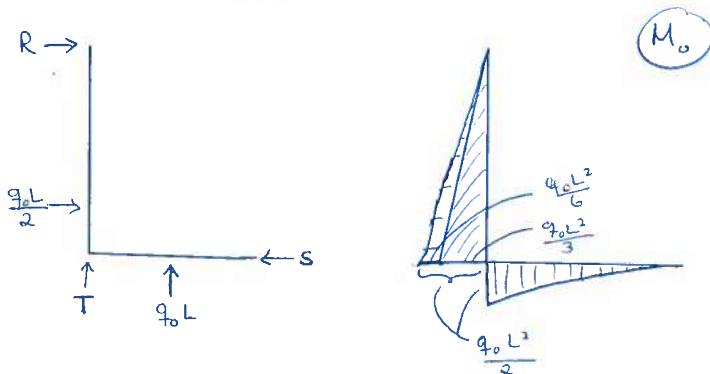
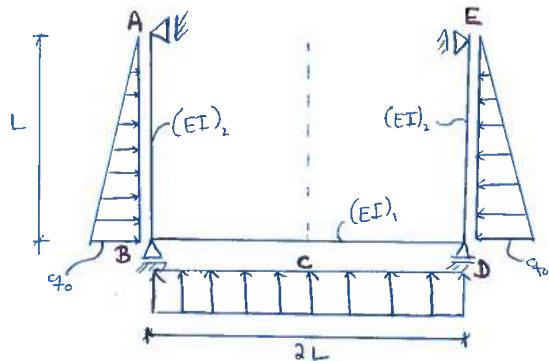
4 Oppsummering og videre arbeid

Som konstatert underveis i oppgavene er det en eller flere regnfeil i forbindelse med midtmomentene og rotasjonene i MATLAB-programmet. Resultatene er fysisk logiske da vi har store og små midtmomenter henholdsvis der det fysisk skal være det, og rotasjoner som går riktig vei og er symmetriske der de skal være det. Det eneste som er feil er verdien. Vi konkluderer derfor med at feilen i programmet ikke kan være kritisk stor da logikken på resultatene tydelig er riktig.

Videre arbeid ville vært å feilsøke MATLAB-programmet enda mer i håp om å finne feilen(e). Da vi laget funksjonen for 2.arealmoment for et I-profil og da vi laget inputfilen tok vi høyde for at tversnittene kunne være usymmetriske, men dette gikk vi senere bort fra. Dette ville ikke vært mye jobb å implementere i resten av programmet og ville vært ett naturlig neste steg. Ved en å legge til en ny variabel i inputfilen som også ga tversnittet ett nummer ville vi også kunne ha åpnet for at profiler av samme type kunne ha ulike dimensjoner, noe som kunne ha vært gunstig i de horisontale bjelkene. Da kunne vi ikke lenger ha brukt den nåværende funksjonen som sjekker om det maksimale spenningen. Den naturlige neste steget ville være å utvide denne til å beregnet flytespenningen til å vurdere for hvert høydenivå hver for seg.

5 Appendix

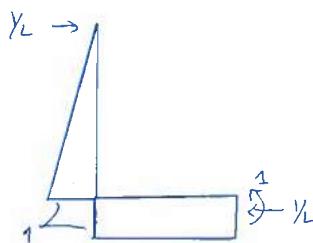
A Oppgave C: Håndberegninger



$$\uparrow \sum F_y = 0 \Rightarrow T = -q_0 L$$

$$\sum M_{0T} = 0 \Rightarrow R \cdot L + \frac{q_0 L^2}{6} - \frac{q_0 L^2}{2} = 0 \Rightarrow R = \frac{q_0 L}{3}$$

$$\rightarrow \sum F_x = 0 \Rightarrow S = -\frac{5}{6} q_0 L$$

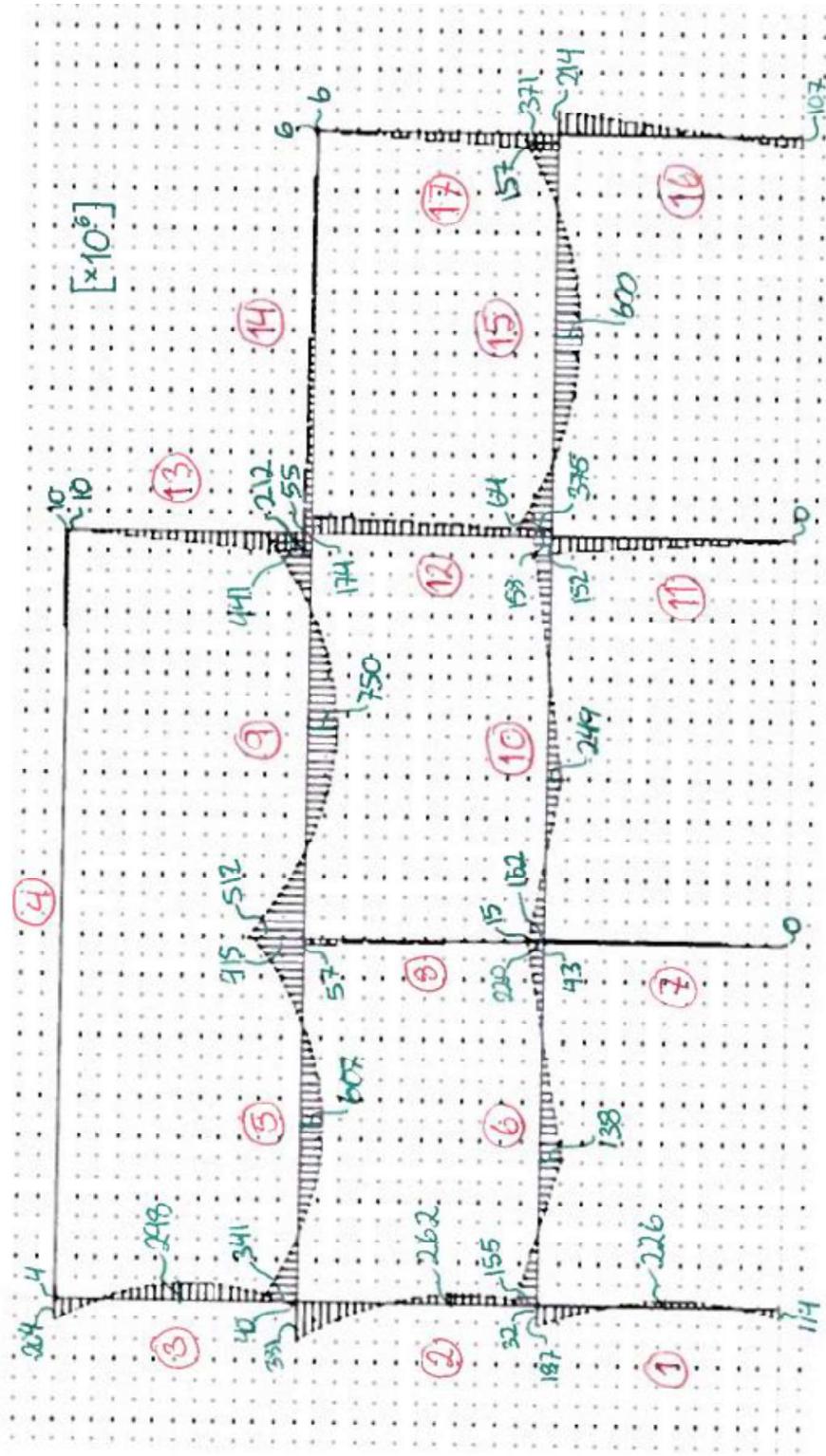


$$\Theta_{10} = \frac{13}{90} \frac{q_0 L^3}{(EI)_2} + \frac{1}{6} \frac{q_0 L^3}{(EI)_1}$$

$$\Theta_{11} = \frac{1}{3} \frac{L}{(EI)_2} + \frac{L}{(EI)_1}$$

$$\Rightarrow X = -\frac{\Theta_{10}}{\Theta_{11}} = -\frac{\frac{13}{90} \frac{q_0 L^2}{(EI)_2} + \frac{1}{6} \frac{q_0 L^2}{(EI)_1}}{\frac{1}{3} \frac{1}{(EI)_2} + \frac{1}{(EI)_1}}$$

B Oppgave E: MATLAB-resultat



C Oppgave E: Håndberegninger

| | | | |
|----------|--|--------------|-------|
| | | Oppdr.nr.: | Side: |
| Oppdrag: | | Utført: | Dato: |
| | | Kontrollert: | Dato: |

(5) :



$$h = 12 \quad L_1 = 1800 \text{ mm}$$

$$L_2 = 2000 \text{ mm}$$

1) L_1
3) L_2

2)
5)

$$\vec{f}_o = \begin{bmatrix} -\frac{15 \cdot 18^2}{12} \\ \frac{15 \cdot 18^2}{12} \end{bmatrix} = \begin{bmatrix} -405 \\ 405 \end{bmatrix} \text{ kNm}$$

$$K_N = 2,552 \cdot 10^9 \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 0,0057 \\ 0,0023 \end{bmatrix} = \begin{bmatrix} 69,9 \\ 52,5 \end{bmatrix} \text{ kNm}$$

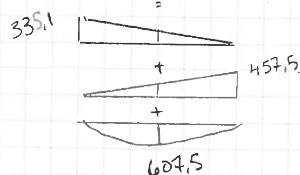
$$\vec{S} = \begin{bmatrix} -405 + 69,9 \\ 405 + 52,5 \end{bmatrix} = \begin{bmatrix} -335,1 \\ 457,5 \end{bmatrix} \text{ kNm}$$

M_o

$$335,1 \quad M_F = 546,3$$

$$M_f = 607,5 - \left(\frac{457,5 + 335,1}{2} \right)$$

$$= 546,3 \text{ kNm}$$



$$\vec{V} = \begin{bmatrix} (+) \\ (+) \end{bmatrix} \sim \frac{335,1}{18} = 18,6 \text{ kN}$$

$$\begin{bmatrix} (+) \\ (+) \end{bmatrix} \sim \frac{457,5}{18} = 25,42 \text{ kN}$$

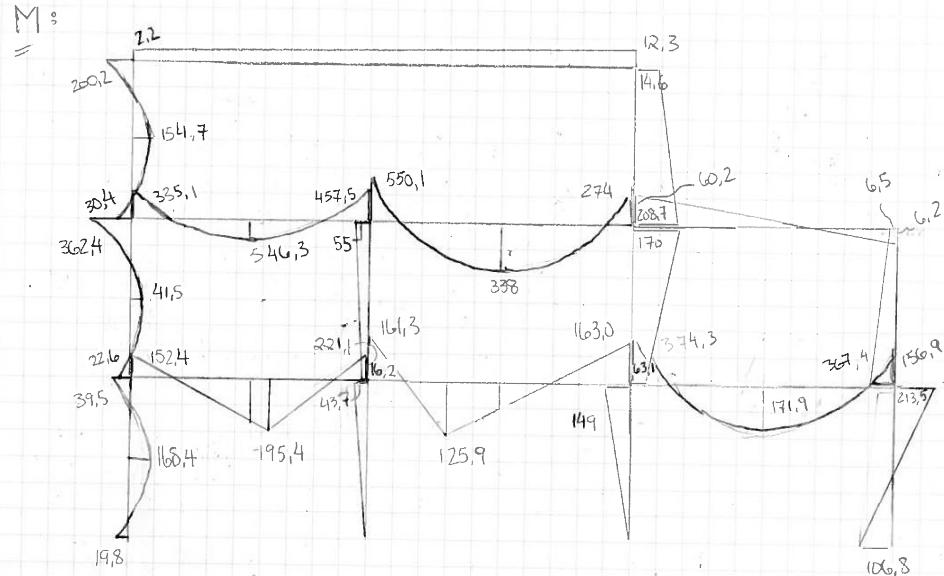
$$\begin{bmatrix} (+) \\ (-) \end{bmatrix} \sim \frac{15 \cdot 18}{2} = 135 \text{ kN}$$

128,2

$$\begin{bmatrix} (+) \\ (-) \end{bmatrix} \sim 141,8$$

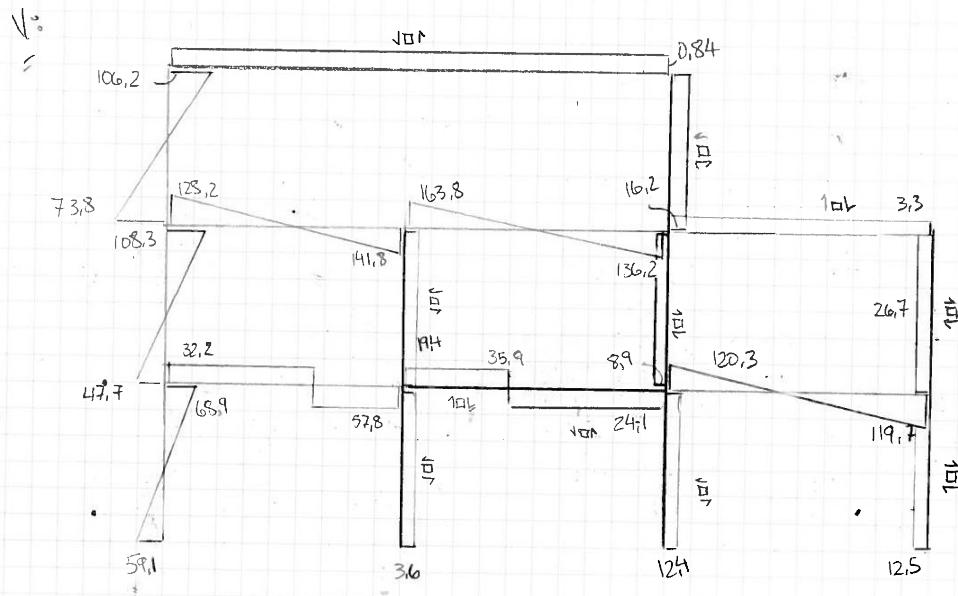
D Oppgave E: Håndberegninger

| | | |
|----------------------|----------------------|----------------------|
| <input type="text"/> | <input type="text"/> | <input type="text"/> |
| Oppdrag: | Uført: | Dato: |
| | Kontrollert: | Dato: |



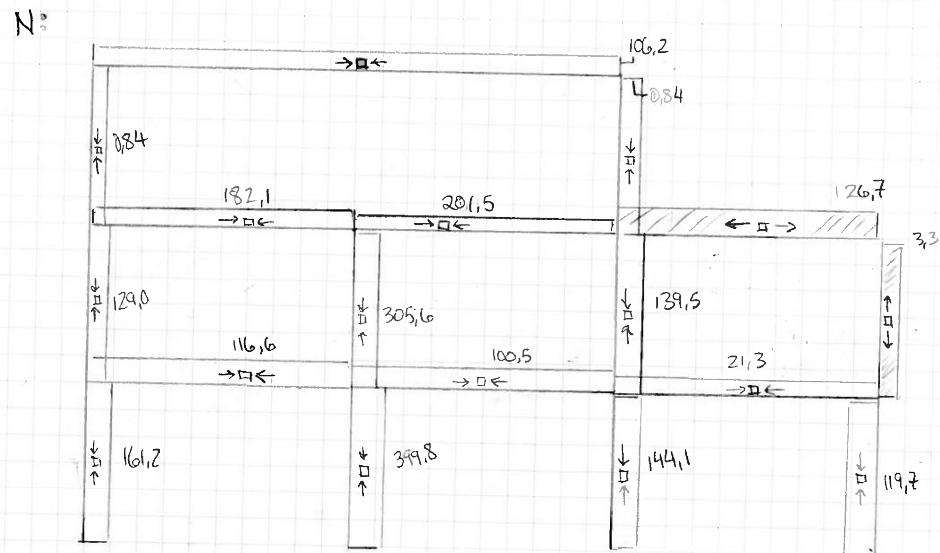
E Oppgave E: Håndberegninger

| | | |
|---|--------------|-------|
|  | Oppdr.nr.: | Side: |
| Oppdrag: | Utført: | Dato: |
| | Kontrollert: | Dato: |



F Oppgave E: Håndberegninger

| | | | | | | |
|----------|---|---|---|---|--------------|-------|
| | ↓ | ↑ | ↓ | ↓ | Oppdr.nr.: | Side: |
| Oppdrag: | | | | | Utført: | Dato: |
| | | | | | Kontrollert: | Dato: |



G Resultater

G.1 Resultater oppgave c

```
1 Restultater oppgave c:  
2  
3 rot =  
4  
5     8.7302e-006  
6     -25.3968e-006  
7     25.3968e-006  
8     -8.7302e-006  
9  
10 endeMom =  
11  
12     -2.2667e+006   -232.8306e-012  
13     2.2667e+006    -2.2667e+006  
14     2.2667e+006    -58.2077e-012  
15  
16 midtMom =  
17  
18     458.3333e+003  
19     -5.0000e+006  
20     -458.3333e+003  
21  
22 K =  
23  
24     84.0000e+009    42.0000e+009    0.0000e+000    0.0000e+000  
25     42.0000e+009    168.0000e+009   42.0000e+009    0.0000e+000  
26     0.0000e+000    42.0000e+009    168.0000e+009   42.0000e+009  
27     0.0000e+000    0.0000e+000    42.0000e+009    84.0000e+009
```

G.2 Resultater Hovedoppgaven

```
1 Restultater oppgave c:  
2  
3 rot =  
4  
5     8.7302e-006  
6     -25.3968e-006  
7     25.3968e-006  
8     -8.7302e-006  
9  
10 endeMom =  
11  
12     -2.2667e+006   -232.8306e-012  
13     2.2667e+006    -2.2667e+006  
14     2.2667e+006    -58.2077e-012  
15  
16 midtMom =  
17  
18     458.3333e+003  
19     -5.0000e+006  
20     -458.3333e+003  
21  
22 K =  
23  
24     84.0000e+009    42.0000e+009    0.0000e+000    0.0000e+000  
25     42.0000e+009    168.0000e+009   42.0000e+009    0.0000e+000  
26     0.0000e+000    42.0000e+009    168.0000e+009   42.0000e+009  
27     0.0000e+000    0.0000e+000    42.0000e+009    84.0000e+009
```

H Kode

H.1 Main

```
1 %%%%%%
2 % Titel: MATLAB-Prosjektet %
3 % Emne: TMR4167 – Marin Teknikk 2 %
4 % Semester: Høst 2017 %
5 % Forfattere: 10024, 10048, 10076, 10011 %
6 % Oppdatert: 2017–11–04 %
7 % Funksjon: Main analyserer en konstruksjon med gitte profiler og laster %
8 % ved hjelp av en rekke underfunksjoner. Den skriver ut %
9 % rotasjonene i knutepunktene og endemomentene i knutepunktene %
10 % til konstruksjonen. %
11 %
12 %%%%%%
13
14 clear all; format shorteng;
15
16 %% 1. Initialiserer
17 fprintf('
18 fprintf('\t\t\t\t MATLAB-Prosjektet\n');
19 fprintf('
20
21 [npunkt, punkt, nelem, elem, nTver, profil, nForL, ForL, nPktL, PktL, ...
22 nMom, Mom] = lesinput(); % Leser inputdata
23
24 %% 2. BEREGNINGER ELEMENTER OG PROFILER
25 I = I_areal(nTver, profil) % 2. Arealmoment for profiltypene
26 l = lengder(punkt, elem, nelem); % Elementlengder
27 EI_l = elemStivhet(nelem, elem, I, I); % Elementenes bøyestivhet
28
29
30 %% 3. BEREGNINGER LASTER
31 % Lastamplituder i hvert knutepunkt når en fordelt last går over mer enn
32 % et element
33 q0_KPkt = q0_KPktFunc(nelem, elem, I, nForL, ForL, npunkt);
34
35
36 %% 4. FASTINNSPENNINGSMOMENTER
37 fim = moment(npunkt, punkt, nelem, elem, I, nPktL, PktL, nForL, ...
38 ForL, nMom, Mom, q0_KPkt);
39 ytreMom = ytreMomFunc(npunkt, nMom, Mom);
40
41 %% 5. LASTVEKTOR OG SYSTEMSTIVHETSMATRISER
42 b = lastvektor(fim, ytreMom, npunkt, nelem, elem); % Lastvektoren
43 K = stivhet(nelem, elem, npunkt, EI_l); % Systemstivhetsmatrisen
44
45
46 %% 6. LØSER LIGNINGSSYSTEMET
47 [Kn, Bn] = bc(npunkt, punkt, K, b); % Innfører grensebetingelser
48 rot = Kn\Bn; % Beregner rotasjonene
49
50 %% 7. MOMENTER
51 endeMom = endeM(npunkt, punkt, nelem, elem, rot, fim, EI_l); % Endemomentene
52 BoyPktL = BoyPktL(nelem, elem, I, nPktL, PktL, fim); % Punktlast
53 BoyForL = BoyForL(nelem, I, q0_KPkt, fim); % Fordelte laster
54 midtMom = BoyPktL + BoyForL; % Totalt
55
56
57 %% 8. SPENNINGER
58 sigma = spenning(profil, nelem, elem, I, endeMom, midtMom);
59 testSigma = spenningstest(sigma);
```

```
60
61 %% ----- 9. RESULTATER -----
62 disp('Rotasjonane i de ulike punktene: ')
63 rot
64
65 disp('Elementvis endemoment: ')
66 endeMom
```

H.2 lesinput

```

1 function [npunkt, punkt, nelem, elem, nTver, profil, nForL, ForL, nPktL, ...
2     PktL, nMom, Mom] = lesinput()
3 %%%%%%
4 % Titel:    lesinput
5 %
6 % Funksjon: Leser inn nødvendig informasjon om konstruksjonen og lastene %
7 %             fra en forhåndsdefinert inputfil med nødvendig data.
8 %             tilhørende lastene
9 %
10 % Oppdatert: 2017-11-01
11 %
12 % Formater:
13 %         punkt = [x-koordinat [mm], y-koordinat [mm], ...]
14 %                     grensebetingelse (1 => fast innspent, ...)
15 %                     0 => fri rotasjon]
16 %
17 %         elem = [Knutpunkt ende 1,
18 %                   knutepunkt ende 2, E-modul [MPa],
19 %                   tversnittstype (1 = l-profil, 2 = Rør-profil)]
20 %
21 %         profil(l) = [profilnummer, profilhøyde [mm], ...]
22 %                     tykkelse bunnflens [mm], tykkelse toppflens [mm], ...
23 %                     tykkelse stag [mm], bredde bunnflens [mm], ...
24 %                     bredde toppflens [mm]]
25 %
26 %         profil(rør) = [profilnummer, ytrediameter [mm], tykkelse [mm], ...]
27 %                         0, 0, 0,0]
28 %
29 %         ForL = [antall element lasten går over, amplitude første %
30 %                  element, amplitude siste element([kN/m] = [N/mm]), %
31 %                  elementene lasten går over(maks 4)]
32 %
33 %         PktL = [elementnummer, amplitude [N], avstand til ....]
34 %                 KPkt 1 [mm]]
35 %
36 %         Mom = [Knutepunkt, moment [Nm]]
37 %%%%%%
38
39 %% Åpner inputfila
40 hoved = true; % Avgjør om hoved- eller testfila
41
42 if (hoved)
43     filid = fopen('input.txt','r'); % Åpner hovedoppgaven
44     fprintf('Leser inn hovedoppgaven...\n');
45 else
46     filid = fopen('inputC.txt','r'); % Åpner Oppgave c
47     fprintf('Leser inn oppgave c...\n');
48 end
49
50 %% Knutepunkt og elementer
51 npunkt = fscanf(filid, '%i',[1 1]); % Antall knutepunkt
52 punkt = fscanf(filid, '%f %f %i',[3 npunkt])'; % Informasjon om punktene
53
54 nelem = fscanf(filid, '%i',[1 1]); % Antall elementer
55 elem = fscanf(filid, '%i %i %f %i',[4 nelem])'; % Info om elementene
56
57 %% Tversnitt
58 nTver = fscanf(filid, '%i',[1 1]); % Antall tversnitt
59 profil = fscanf(filid, '%i %f %f %f %f %f %f',[7 nTver])'; % Info om tversnittene
60
61

```

```

62 %% Laster
63 nForL = fscanf(filid , '%i',[1 1]); % Antall fordelte laster
64 ForL = fscanf(filid , '%i %f %f %i %i %i %i ',... % Info fordelte laster
65 [7 nForL]) ';
66
67 nPktL = fscanf(filid , '%i',[1 1]); % Antall punktlaster
68 PktL = fscanf(filid , '%i %f %f ', [3 nPktL])'; % Info om punktlastene
69
70 nMom = fscanf(filid , '%i',[1 1]); % Antall ytre momenter
71 Mom = fscanf(filid , '%i %f ', [3 nMom])'; % Informasjon om momentene
72
73 %% Avslutter
74 fclose(filid); % Lukker input-filen
75 end

```

H.3 2.Arealmoment

```

1 function I_areal = I_areal(nTver, profil)
2 % Titel: I_areal %
3 % Funksjon: Leser inn tversnittsdata og regner ut 2.arealmoment for de %
4 % ulike typene profiltypene, her I- og Rør-Profil. %
5 % Oppdatert: 2017-11-01 %
6 %%%%%%
7 %%%%%%
8
9 I_areal = zeros(2,1);
10
11 for i = 1: nTver
12     y_global = 0;
13 %----- I-PROFIL -----
14     if(profil(i,1) == 1)
15         % Henter informasjon
16         h = profil(i,2);           % Tversnittshøyde [mm]
17         t_bf = profil(i,3);        % Tykkelse brunnflens [mm]
18         t_tf = profil(i,4);        % Tykkelse toppflens [mm]
19         t_s = profil(i,5);         % Tykkelse stag [mm]
20         b_bf = profil(i,6);        % Bredde bunnflens [mm]
21         b_tf = profil(i,7);        % Bredde toppflens [mm]
22
23         % Beregner arealer (I-Profil)
24         a_bf = b_bf * t_bf;        % Areal bunnflens [mm^2]
25         a_tf = b_tf * t_tf;        % Areal toppflens [mm^2]
26         a_s = (h - b_bf - b_tf) * t_s; % Arealet av staget [mm^2]
27         a_tot = a_bf + a_tf + a_s; % Totalt tversnittsareal [mm^2]
28
29         % Beregner lokale arealsenter (I-Profil)
30         y_tf = h - (0.5*t_tf);    % [mm]
31         y_bf = 0.5*t_bf;          % [mm]
32         y_s = 0.5*h;              % [mm]
33
34         % Beregner globalt arealsenter (I-Profil)
35         if ((t_bf ~= t_tf) || (b_bf ~= b_tf))
36             y_c = (1/a_tot) * ((a_tf * y_tf) + (a_bf * y_bf)+ ...
37             (a_s * y_s)); % [mm]
38
39             if (y_c >= (h - y_c))
40                 y_global = y_c;      % [mm]
41             else
42                 y_global = h - y_c; % [mm]
43             end
44
45         elseif (((t_bf == t_tf) && (b_bf == b_tf)))
46             y_c = h/2;
47             y_global = y_c;        % [mm]
48
49         else
50             fprintf('Error');
51         end
52
53         % Beregner bidrag fra hver del (I-Profil)
54         A_bf = ((1/12) * (t_bf^3 * b_bf)) + (y_c - (t_bf/2))^2 * a_bf; % bunnflens [mm^4]
55         A_tf = ((1/12) * (t_tf^3 * b_tf)) + (h - y_c - (t_tf/2))^2 * a_tf; % Toppfleens [mm^4]
56         A_s = (1/12)*((h - t_tf - t_bf)^3 * t_s) + (y_s - y_c)^2* a_s; % Stag [mm^4]
57
58         % Totalt 2. Arealmoment (I-Profil)

```

```

59     I_areal(i,1) = A_tf + A_bf + A_s;    % Total 2. Arealmoment I-Profil [mm^4]
60
61
62 % ----- RØRTVERSNIITT -----
63 elseif(profil(i,1) == 2)
64     r = 0.5*profil(i,2);                      % radius [mm]
65     t = profil(i,3);                         % Tykkelse [mm]
66     I_areal(i,1) = ((pi/4)*(r^4 - (r-t)^4)); % 2. Arealmoment rør.T [mm^4]
67     y_global = r;                            % [mm]
68
69
70 % ----- ANNEN -----
71 else
72     fprintf('Error: Ugyldig tversnittsnummer');
73     I_areal = 0;
74 end
75
76
77 fprintf('2. arealmoment beregnet for alle profiltyper\n');
78

```

H.4 lengder

```
1 function l = lengder(punkt, elem, nelem)
2 % Titel:    lengder
3 % Funksjon: Regner ut lengden på hver av elementene i konstruksjonen på %
4 %           bakgrunn av knutepunktskoordinatene. Finner endring i x- og z- %
5 %           retning separat og bestemmer den totale lengden med pytagoras %
6 %           Oppdatert: 2017-10-17
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8
9
10 l = zeros(nelem,1);          % Initialiserer vektor med lengder
11
12 for i=1:nelem
13     dx = punkt(elem(i,1),1) - punkt(elem(i,2),1); % Endring i x
14     dy = punkt(elem(i,1),2) - punkt(elem(i,2),2); % Endring i y
15     l(i) = sqrt(dx*dx + dy*dy);                   % Elementlengde
16 end
17
18 fprintf('Elementlengder regnet ut\n');
19
```

H.5 Elemtstivhet

```
1 function EI_L = elemStivhet(nelem, elem, I, l)
2 % Titel:    bjelkeStiv
3 % Funksjon: Beregner bjelkestivheten EI/l for alle elementene
4 %          %
5 % Oppdatert: 2017-10-18
6 %%%%%%
7
8 E = elem(:,3);           % E-modul
9 EI_L = zeros(nelem,1);   % Bjelkestivhetene
10
11 for i = 1: nelem
12 %----- I-Profil -----
13     if(elem(i,4) == 1)
14         EI_L(i) = E(i)*l(1)*(1/I(i));
15
16 %----- Rør-Profil -----
17     elseif (elem(i,4) == 2)
18         EI_L(i) = E(i)*l(2)*(1/I(i))';
19
20 %----- Ukjent profil -----
21     else
22         fprintf('Error\n')
23         EI_L(i) = 0;
24     end
25 end
26
27 fprintf('Elementstivheter beregnet\n');
28 end
```

H.6 Amplituder ved fordelt last

```
1 function q0_KPkt = q0_KPktFunc(nelem, elem, l, nForL, ForL, npunkt)
2 %%%%
3 % Titel: q_elem %
4 % Funksjon: Regner inn amplitudene i knutepunkt 1 og 2 for tilfellene der %
5 % vi har jevnt fordelt last over et eller flere elementer %
6 % Oppdatert: 2017-11-04 %
7 %%%%
8
9 q0_KPkt = zeros(nelem,2); % Amplitudene i hvert knutepunkt
10
11 %% Utregning
12 for i = 1:nForL
13     L_tot = 0; % totale lengden lasten går over
14     n_elem = ForL(i,1); % Antall elementer lasten går over
15
16     for j = 1:n_elem
17         element = ForL(i,j+3); % Elementnummeret
18         L = l(element); % Lengden av elementet
19         L_tot = L_tot + L; % Oppdaterer total lengde
20     end
21
22     q1 = ForL(i,2); % Amplitude hovedende 1
23     q2 = ForL(i,3); % Amplitude hovedende 2
24     s = (q2-q1)/L_tot; % Stigningstall
25     q_a = q1; % 1.hovedende = 1.knutepunkt
26
27     for h = 1:n_elem
28         element = ForL(i,h+3); % Elementnummeret
29         L = l(element); % Lengden av elementet
30         q_b = q_a + s*L; % Beregner amplituden i "motsatt" ende
31         q0_KPkt(element,1) = q_a; % Tilordner til hovedmatrisa
32         q0_KPkt(element,2) = q_b; % Tilordner til hovedmatrisa
33
34         q_a = q_b; % Oppdaterer
35     end
36 end
37
38 fprintf('Laster i knutepunktene beregnet\n')
39
40 end
```

H.7 moment

```

1 function fim = moment(npunkt,punkt,nelem,elem,l,nPktL,PktL,nFlast,...%
2 %last,nMom,Mom,q0_KPkt)%
3 % Titel: Moment %
4 % Funksjon: Beregner fastinnspenningsmomentene i enden av hvert element %
5 % pga. punktlast og fordelte laster. Dersom vi har en trapes- %
6 % last deles den opp i to bidrag. Et som en jevnt fordelte last %
7 % og en som en trekantlast som superposisjoneres. Jevnt fordelte %
8 % laster og trekantlaste regnes direkte. For trapes- og %
9 % trekantlaste ligger det inne en test om hvilken side som er %
10 % størst slik at riktig formel blir brukt. %
11 % Oppdatert: 2017-11-04 %
12 %%%%%%
13 %%%%%%
14
15 fim = zeros(nelem,2); % Fastinnspenningsmomentene
16 counter = 0;
17 %----- PUNKTLAST [N] -----
18
19 for counter = 1:nPktL
20     element = PktL(counter,1); % Elementnummeret
21     P = PktL(counter,2); % Lastamplituden
22     alpha = PktL(counter,3); % Avstand (0-1) på lasten fra ende a
23     L = l(element); % Elementlengden
24
25     a = l(element)*alpha; % Avstand fra last til ende a
26     b = l(element) - a; % Avstand fra last til ende b
27     m_ab = -P*(a*b^2/L^2); % Momentet i ende a pga. lasten
28     m_ba = P*(a^2*b/L^2); % Momentet i ende b pga. lasten
29
30     fim(element,1) = fim(element,1) + m_ab; % Fastinnspenningsmoment i a
31     fim(element,2) = fim(element,2) + m_ba; % Fastinnspenningsmoment i b
32 end
33
34 %----- FORTDELT LAST ([kN/m] = [N/mm]) -----
35 for j = 1:nelem
36     q_a = q0_KPkt(j,1); % Lastamplituden ende 1
37     q_b = q0_KPkt(j,2); % Lastamplituden ende 1
38     L = l(j); % Elementlengden
39
40 %----- Jevnt fordelt last -----
41     if q_a == q_b
42         m_ab = -(1/12)*q_a*L^2; % Bidrag i ende 1 og ende 2. Like store
43         m_ba = (1/12)*q_b*L^2;
44
45 %----- Trekantlast -----
46     elseif (q_a == 0) || (q_b == 0)
47         if abs(q_a) > abs(q_b) % Størst i ende a
48             m_ab = -(1/20)*q_a*L^2;
49             m_ba = (1/30)*q_a*L^2;
50
51         else % Størst i ende b
52             m_ab = -(1/30)*q_b*L^2;
53             m_ba = (1/20)*q_b*L^2;
54     end
55
56 %----- Trapeslast -----
57     else
58         m_ab_jf = -(1/12)*q_a*L^2; % Bidrag ende 1 jevnt fordelt last
59         m_ba_jf = (1/12)*q_b*L^2; % Bidrag ende 2 jevnt fordelt last
60
61

```

```

62      if (abs(q_b)) > (abs(q_a))           % Tester hvilken ende som er
63      størst
64          q_c = q_b - q_a;      % Amplitude trekantlast, ende b størst
65          m_ab_trapes = -(1/30)*q_c*L^2; % Bidrag ende 1 trekantlast
66          m_ba_trapes = (1/20)*q_c*L^2; % Bidrag ende 2 trekantlast
67      else
68          q_c = q_a - q_b;      % Amplitude trekantlast, ende a størst
69          m_ab_trapes = -(1/20)*q_c*L^2; % Bidrag ende 1 trekantlast
70          m_ba_trapes = (1/30)*q_c*L^2; % Bidrag ende 2 trekantlast
71      end
72
73      m_ab = m_ab_jf + m_ab_trapes;    % Superposisjonerer i ende 1
74      m_ba = m_ba_jf + m_ba_trapes;    % Superposisjonerer i ende 2
75  end
76
77  fim(j,1) = fim(j,1) + m_ab; % Totalt fastinnspenningsmoment i a
78  fim(j,2) = fim(j,2) + m_ba; % Totalt fastinnspenningsmoment i b
79 end
80
81 fprintf('Fastinnspenningsmomentene beregnet\n');
82 end

```

H.8 lastvektor

```
1 function b = lastvektor(fim ,ytreMom ,npunkt ,nelem ,elem )
2 %%%%%%
3 % Titel: Lastvektor %
4 % Funksjon: Setter sammen lastvektoren. Tar inn fastinnspenningsmomentene %
5 %           og de ytre momentene. Omgjør slik at vi har informasjon om %
6 %           hver node i stede for hver ende av et element. Sørger for %
7 %           riktig fortegn. %
8 % Oppdatert: 2017-11-03 %
9 %%%%%%
10
11 b = zeros(npunkt,1);    % Lastvektoren
12
13 %----- Fra fordelte laster og punktlaster -----
14 for i = 1:nelem
15     KPkt1 = elem(i,1);
16     KPkt2 = elem(i,2);
17
18     b(KPkt1,1) = b(KPkt1,1) - fim(i,1);      % Summerer i lokalt KPkt1
19     b(KPkt2,1) = b(KPkt2,1) - fim(i,2);      % Summerer i lokalt KPkt2
20 end
21
22 %----- Fra ytre momenter -----
23 b = b + ytreMom;          % Legger til ytre momenter
24
25 fprintf('Lastvektor beregnet.\n')
26 end
```

H.9 stivhet

```
1 function K = stivhet(nelem, elem, npunkt, EI_L)
2 % Titel: stivhet %
3 % Funksjon: Regner ut elementstivhetsmatrisa for hvert element og %
4 % plasserer verdiene riktig i den globale stivehetsmatrisa %
5 % Oppdatert: 2017-11-02 %
6 %%%%%%
7 %%%%%%
8
9 K = zeros(npunkt);
10 konst = [4 2; 2 4];
11
12 for i=1:nelem
13     KPkt1 = elem(i,1);      % Knutepunkt ende 1
14     KPkt2 = elem(i,2);      % Knutepunkt ende 2
15
16     k = konst*EI_L(i);    % Lokal elementstivhetsmatrise
17
18     % Tilordner i den globale stivhetsmatrisen
19     K(KPkt1,KPkt1) = K(KPkt1,KPkt1) + k(1,1);
20     K(KPkt2,KPkt1) = K(KPkt2,KPkt1) + k(2,1);
21     K(KPkt1,KPkt2) = K(KPkt1,KPkt2) + k(1,2);
22     K(KPkt2,KPkt2) = K(KPkt2,KPkt2) + k(2,2);
23 end
24
25 fprintf('Systemstivhetsmatrise definert\n')
26 end
```

H.10 bc

```
1 function [Kn, Bn] = bc(npunkt, punkt, K, b)
2 %%%%%%
3 % Titel: Bc %
4 % Funksjon: Oppdaterer systemstivhetsmatrisa og lastvektoren slik at de %
5 % tar hensyn til grensebetingelsene. I vårt tilfelle er noen av %
6 % knutepunktene fast innspent og om de er det må vi "nulle ut" %
7 % tilhørende rad og kolonne slik at vi ikke får rotasjoner i %
8 % de fast innspente punktene. Grensebetingelsene er lest inn %
9 % sammen med resten av informasjonen om knutepunktene og det er %
10 % fast innspent dersom verdien er 1. %
11 % Oppdatert: 2017-11-02 %
12 %%%%%%
13
14 Kn = K;      % Systemstivhetsmatrise inkludert grensebetingelser
15 Bn = b;      % Lastvektor inkludert grensebetingelser
16
17 for i = 1:npunkt
18     bc = punkt(i,3);    % Gensebetingelse
19
20     if bc == 1          % Fast innspent
21         Kn(:,i) = 0;    % Nuller ut rad
22         Kn(i,:) = 0;    % Nuller ut kolonne
23         Kn(i,i) = 1;
24
25         Bn(i) = 0; % Nuller ut i lastvektoren
26
27     elseif (bc == 1) && (bc == 0) % Test for feil
28         fprintf('Error! Verdien tilsvarer ikke en definert bc!\n')
29
30     end
31 end
32
33 fprintf(' Grensebetingelser tatt hensyn til.\n')
34 end
```

H.11 endeM

```
1 function S = endeM(npunkt,punkt,nelem,elem,rot,fim,El_L)
2 % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
3 % %Titel:      endeM                                %
4 % %Funksjon: Regner ut endemomentet på alle bjelkene ved å regne momentet %
5 % %           som følge av rotasjonen og legge til fastinnspenningsmomentet %
6 % %Oppdatert: 2017-11-05                                %
7 % % % % % % % % % % % % % % % % % % % % % % % % % % % %
8
9 S = zeros(nelem,2);
10 momentRotasjon1 = zeros(nelem,2);
11
12 for i = 1:nelem
13     KPkt1 = elem(i,1);
14     KPkt2 = elem(i,2);
15
16     rot1 = rot(KPkt1);
17     rot2 = rot(KPkt2);
18
19     momentRotasjon1(i,:) = El_L(i)*[4*rot1 + 2*rot2;
20                                         2*rot1 + 4*rot2];
21     S(i,:) = momentRotasjon1(i,:) + fim(i,:);
22
23 end
24
25 fprintf('Endemomenter regnet ut.\n'); % Melding til bruker
26
27 end
```

H.12 Bøyemoment pga fordelt last

```

1 function BoyForL = BoyForL(nelem,l,q0_KPkt,endeMom)
2 % Titel: BoyForL %
3 % Funksjon: Beregner bøyemomentet på midten av bjelken for fordelte %
4 % laster. Dersom lasten er jevnt fordelt brukes formelen direkte %
5 % for å finne momentet på midten. Dersom det er en trapeslast %
6 % deles den opp i en jevnt fordelt last og en trekantlast som %
7 % senere superposisjoneres. %
8 % Oppdatert: 2017-11-05 %
9 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 BoyForL = zeros(nelem,1); % Momentet på midten pga. jevnt fordelt last
13
14 for i = 1:nelem
15     L = l(i); % Lengden av elementet
16     q_a = q0_KPkt(i,1); % Amplitude ende a
17     q_b = q0_KPkt(i,2); % Amplitude ende b
18
19     endeMom_a = endeMom(i,1); % Endemoment ende a
20     endeMom_b = endeMom(i,2); % Endemoment ende b
21
22 %----- Jevnt fordelt last -----
23 if q_a == q_b
24     midtMom_jf = q_a*L^2*(1/8); % midtmoment fra jevnt fordelt last
25     BoyForL(i) = midtMom_jf + (endeMom_a + endeMom_b);
26
27 %----- Trekantlast -----
28 elseif (q_a == 0) || (q_b == 0)
29     if abs(q_a) > abs(q_b) % Størst i ende a
30         midtMom_trekant = q_a*L^2*(1/16);
31     else % Størst i ende b
32         midtMom_trekant = q_b*L^2*(1/16);
33     end
34     BoyForL(i) = midtMom_trekant + (endeMom_a + endeMom_b);
35
36 %----- Trapeslaster -----
37 else
38     % Jevnt fordelt bidrag
39     midtMom_jf = q_a*L^2*(1/8);
40
41     % Trekantbidrag
42     if abs(q_a) > abs(q_b)
43         midtMom_trapes = (q_a-q_b)*L^2*(1/16);
44     else
45         midtMom_trapes = (q_b-q_a)*L^2*(1/16);
46     end
47
48     % Superposisjonerer
49     BoyForL(i) = midtMom_jf + midtMom_trapes + (endeMom_a + endeMom_b);
50 end
51 end
52 fprintf('Bøyemoment midt på bjelkene med fordelt last regnet ut\n')
53 end
54

```

H.13 Bøyemoment pga punktlast

```
1 function BoyPktL = BoyPktL(nelem, elem, l, nPktL, PktL, fim)
2 % Titel:      BoyPktL
3 % Funksjon:   Regner bøyemomentet under punktlaster. Tester først om lasten
4 %               er på riktig element.
5 % Oppdatert:  2017-11-05
6 %%%%%%
7
8
9 BoyPktL = zeros(nelem,1);
10
11 for i = 1:nPktL
12     element = PktL(i,1);      % Elementet lasten virker på
13     P = PktL(i,2);          % Lastamplituden
14     alpha = PktL(i,3);      % Avstandenskoeffisienten
15     L = l(i);              % Lengden av elementet lasten virker på
16     a = L*alpha;            % Avstand fra knutepunkt 1 til lasten
17     b = L - a;              % Avstand fra knutepunkt 2 til lasten
18
19     if (alpha > 0) && (alpha < 1)
20         m = -P*(a*b/L);
21
22     elseif (alpha == 0) || (alpha == 1)
23         fprintf('Lasten virker rett i et knutepunkt')
24         m = 0;
25
26     else
27         fprintf('Error: Lasten er "utenfor" elementet');
28         m = 0;
29     end
30
31 % Momentet på midten pga. fastinnspenningsmomentene.
32 midtMomEnde = (1/L)*((fim(i,1)*b) + fim(i,2)*a);
33
34 % Det totale momentet på midten n
35 BoyPktL(element,1) = BoyPktL(element,1) + m + midtMomEnde;
36 end
37
38 fprintf('Bøyemoment under punktlaster regnet ut\n')
39 end
```

H.14 Spenning

```
1 function sigma = spenning( profil ,nelem ,elem ,I ,endeMom ,m_middt )
2 %%%%%%
3 % Titel:    spenning                                %
4 % Funksjon: Lager en matrise med plass til momentene som virker i hvert %
5 %           knutepunkt, tar inn de påsatte momentene og organiserer dem %
6 %           til riktig knutepunkt                      %
7 % Oppdatert: 2017-11-04                                %
8 %%%%%%
9 sigma = zeros(nelem ,3);      % Spenningene
10
11 for i = 1:nelem
12     profilType = elem(i ,4); % Avgjør profiltypen
13     M1 = endeMom(i ,1);      % Endemoment ende 1
14     M2 = endeMom(i ,2);      % Endemoment ende 2
15     M3 = m_middt(i ,1);     % Midtmoment
16
17 %----- I-PROFIL -----
18 if profilType == 1
19     y_I = 0.5*profil(1,2);      % Profillets arealsenter
20     sigma(i ,1) = (M1*y_I)/I(1); % Spenning ende 1
21     sigma(i ,2) = (M2*y_I)/I(1); % Spenning ende 1
22     sigma(i ,3) = (M3*y_I)/I(1); % Spenning på midten av bjelken
23
24 %----- RØRTVERSNIKT -----
25 elseif profilType == 2
26     y_ror = 0.5*profil(2,2);    % Profillets arealsenter
27     sigma(i ,1) = (M1*y_ror)/I(2); % Spenning ende 1
28     sigma(i ,2) = (M2*y_ror)/I(2); % Spenning ende 1
29     sigma(i ,3) = (M3*y_ror)/I(2); % Spenning på midten av bjelken
30
31 %----- ANNEN -----
32 else
33     fprintf('Error: Ugyldig profil! Kun definert I- og rør-profil')
34 end
35
36 end
37
38 end
```

H.15 Spenningsstest

```
1 function testSigma = spenningsstest(sigma)
2 % Titel: spenningsstest %
3 % Funksjon: Finner den makstatile spenningen (i absoluttverdi) og tester %
4 % om denne er i område 30%–70% av flytespenningen. Gir beskjed %
5 % til brukeren om tversnittet er sterkt nok og eventuelt hva %
6 % som er galt. %
7 % til riktig knutepunkt %
8 % Oppdatert: 2017-11-04 %
9 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10 testSigma = false; % Antar at tversnittet ikke er stekt nok
11
12 sigma_flyt = 320; % Flytespenningen [MPa]
13 sigma_max = max(max(abs(sigma))); % Den største absolutte bøyespenninga
14
15 fprintf('\nMaksimal skjærspenning:\t %f\n', sigma_max);
16
17 if (sigma_max < 0.7*sigma_flyt) && (sigma_max > 0.3*sigma_flyt)
18     testSigma = true;
19     fprintf('Maksimal skjærspenning er innenfor grensene\n');
20
21
22 elseif (sigma_max >= 0.7*sigma_flyt)
23     fprintf('Maksimal skjærspenning er over flytspenningen\n');
24     testSigma = false;
25
26 elseif (sigma_max > 0.7*sigma_flyt)
27     fprintf('Maksimal skjærspenning er på over maksgrensen på 70');
28     fprintf(' prosent av flytespenningen\n');
29     testSigma = false;
30
31 elseif (sigma_max < 0.3*sigma_flyt)
32     fprintf('Maksimal skjærspenning er under 30 prosent av flytespenningen.\n');
33
34     testSigma = false;
35 end
36
37 end
```

H.16 Ytre moment

```
1 function ytreMom = ytreMomFunc(npunkt,nMom, Mom)
2 %%%%%%
3 % Titel:      ytreMomFunc                                %
4 % Funksjon: Lager en matrise med plass til momentene som virker i hvert %
5 %            knutepunkt, tar inn de påsatte momentene og organiserer dem %
6 %            til riktig knutepunkt                            %
7 % Oppdatert: 2017-11-04                                    %
8 %%%%%%
9
10 ytreMom = zeros(npunkt,1);
11
12 for i = 1:nMom
13     KPkt = Mom(i,1);        % Knutepunktet momentet virker på
14     M = Mom(i,2);          % Størrelsen på momentet
15
16     ytreMom(KPkt) = M;    % Tilordner til tilhørende knutepunkt
17 end
18
19 end
```

H.17 input

```
1 14
2 0 0 1
3 0 12e3 0
4 0 24e3 0
5 0 36e3 0
6 18e3 24e3 0
7 18e3 12e3 0
8 18e3 0 0
9 38e3 0 0
10 38e3 12e3 0
11 38e3 24e3 0
12 38e3 36e3 0
13 58e3 24e3 0
14 58e3 12e3 0
15 58e3 0 1
16 17
17 1 2 210e3 2
18 2 3 210e3 2
19 3 4 210e3 2
20 4 11 210e3 1
21 3 5 210e3 1
22 2 6 210e3 1
23 7 6 210e3 2
24 6 5 210e3 2
25 5 10 210e3 1
26 6 9 210e3 1
27 8 9 210e3 2
28 9 10 210e3 2
29 10 11 210e3 2
30 10 12 210e3 1
31 9 13 210e3 1
32 14 13 210e3 2
33 13 12 210e3 2
34 2
35 1 300 26 26 14 180 180
36 2 500 10 0 0 0 0
37 3
38 3 10 16 1 2 3 0
39 2 15 15 5 9 0 0
40 1 12 12 15 0 0 0
41 2
42 6 90e3 0.6
43 10 60e3 0.4
44 1
45 4 200e6
```

H.18 input oppgave c

```
1 4
2 0 1e3 0
3 0 0 0
4 2e3 0 0
5 2e3 1e3 0
6 3
7 2 1 210e3 1
8 2 3 420e3 1
9 3 4 210e3 1
10 2
11 1 300 13.5 13.5 8.6 180 180
12 2 500 10 0 0 0 0
13 3
14 1 10 0 1 0 0 0
15 1 -10 -10 2 0 0 0
16 1 -10 0 3 0 0 0
```