Norges miljø- og biovitenskapelige universitet

Masteroppgave 2018 30 stp
REALTEK

Multivariabel studie av et høyt trehus

Multivariate study of a high-rise timber building

Jonas Aasebø Stafsberg
Byggeteknikk og arkitektur

Forord

Denne masteroppgaven er skrevet ved fakultetet for realfag og teknologi ved Norges miljø- og biovitenskapelige Universitet (NMBU). Oppgaven markerer avslutningen på et toårig masterstudie innen Byggeteknikk og Arkitektur. Oppgaven er utarbeidet våren 2018 og utgjør 30 studiepoeng.

Takket være NMBU har jeg blitt tildelt mye god kunnskap innen trekonstruksjoner og programmering. Dette ga meg inspirasjon for tema og arbeidsmetode for denne oppgave.

Oppgaven har gitt meg ny kunnskap innen konstruksjonsteknikk og programmering. Jeg har satt stor pris på muligheten til å få kombinere min interesse for tre-konstruksjoner, konstruksjonsteknikk og programmering.

Takk til min veileder Anders Björnfot som har bistått med god veiledning og oppfølging av oppgaven. I tillegg vil jeg gi en takk til fremtidige kollegaer på Sweco Hamar som har gitt meg arbeidsplass for oppgaveskrivingen og vist interesse for oppgaven. I tillegg vil jeg gi en takk til Magne Bjertnæs på Sweco Lillehammer som har gitt meg tilgang til alle beregningene og informasjon om Mjøstårnet i Brumunddal

Til slutt vil jeg takke familie og venner som har støttet og hjulpet meg på alle mulige måter.

Jonas Aasebø Stafsberg
Hamar, Mai 2018

Sammendrag

Det er vekket stor interesse i høye trehus i nyere tid og vi er nå i en tid hvor høye trehus går fra teoretiske konsepter til virkelige bygg. Først kom Treet i Bergen som var rangert som verdens høyeste trehus helt til Mjøstårnet i Brumunddal ble reis med sine 84 meter over bakken. De teoretiske konseptene utledet i akademiske tekster er nå en realitet. Og lysten til å bygge enda høyere er fortsatt til stede. Treet i Bergen var et pilotprosjekt. Takket være Treet i Bergen har man fått dypere forståelse for de dynamiske parameterne som er så viktig for disse konstruksjonene.

Forskningen og de levende bevisene for hva som er mulig å bygge er utgangspunktet for problemstillingen i denne masteroppgaven. Målet med oppgaven var å se hvordan bæresystemet til Mjøstårnet presterer i en høyde på 100 meter med varierende dybde, bredde, massefordeling og forskjellige tverrsnitt med tanke på de dynamiske bruks-kravene.

For å se hvordan de forskjellige variablene ga innvirkning på de dynamiske bruks-kravene ble det konstruert et Python-program som itererte seg igjennom alle mulige sammensetninger av dybde, bredde, massefordeling og tverrsnitt. Begrensningen til fotavtrykket var med dybde mellom 14 og 25 meter, bredden mellom 14 og 35 meter og en massefordeling hvor det ble introdusert ett betongdekket i øverste etasje, de to øverste etasjene osv... Totalt ble det kalkulert 1 520 640 modeller, men det ble brukt 380 160 modeller til mer detaljert analyse. Datasettet med de 380 160 modellene ble analysert og diskutert for å finne underliggende sammenhenger mellom de gitte variablene.

Av de 380 160 modellen som ble analysert var det 373833 modeller som overholdte akselerasjonskravet for kontorbygg etter ISO 10137. I tillegg ble det registrert at byggets bredde hadde veldig liten innflytelse på utnyttelsen av akselerasjonskravet for kontorbygg. Det ble observert modeller som fikk en reduksjon i utnyttelse av akselerasjonskravet med økt bredde.

Abstract

There is a great interest in high-rise timber buildngs. We are now in a time when high-rise timber budilings has gone from theoretical concepts to real buildings. First we had Treet in Bergen which was ranked the world's tallest high-rise timber building until Mjøstårnet in Brumunddal was built with its 84 meters. Theoretical concepts published in academic texts are now a reality, and the desire to build even higher is still present. Treet in Bergen was a pilot project. Thanks to Treet in Bergen, we have gained a deeper understanding of the dynamic parameters that are so important for these constructions.

The research and living evidence of what is possible to build was the starting point for this thesis. The aim of this thesis was to see how the structural system of Mjøstårnet performed at a height of 100 meters with varying depth, width, mass distribution and different cross sections in regard to the wind-induced vibrations.

To see how the different variables affected the dynamic requirements described in ISO 10137, a Python program was constructed to calculate all possible combinations of depth, width, mass distribution and cross-section defined in this thesis. The building depth was set between 14 and 25 meters. The width was set between 14 and 35 meters. And a mass distribution where a concrete deck was installed on the roof, the two top floors and so on. In total, 1 520 640 buildings were calculated, but only 380 160 models were used for further analysis. The dataset with 380 160 models was analyzed and discussed to find underlying relationships between the given variables.

Of the 380 160 buildings that were analyzed, there were 373 833 buildings that met the acceleration requirement for office buildings according to ISO 10137. In addition, it was noted that the width of the building had very little impact on the utilization of the acceleration requirement for office buildings. There were even buildings that got a reduction of the utilization of acceleration requirements with increased width.

Innholdsfortegnelse

Figurliste

Tabeliste

1 Innledning

1.1 Bakgrunn

7 februar 2018 publiserte Aftenposten en artikkel med følgende overskrift «*Planlegger over 40 nye høyhus i Oslo – Her kan de komme*» i artikkelen er det oppsummert høyde, bruksområde og form på byggene (Sørgjerd, 2018). En gjenganger er konstruksjoner på rundt 100 meter med bruksområde som kontor. Nå som verdens høyeste trehus snart står oppreist med sine 84 meter over bakken kan det være interessant å se hvordan samme bæresystem hadde prestert med en høyde rundt 100 meter. Mjøstårnet i Brumunddal er presset til det ytterste når det kommer til de dynamiske brukskravene (Bjertnæs, 2017). Det er ikke unaturlig da kortsiden av Mjøstårnet er 14800 mm fra senterlinjen til senterlinjen kantbjelke. Og med en bredde på 37000 mm, hvor vinden får ett stort lastareal er det ikke rart Mjøstårnet er presset til det ytterste av hva ett tre-bygg kan prestere med de gitte parameterne. Hovedgrunnene til at Mjøstårnet er så smalt er på grunn av planløsningen som må tilfredsstille arkitektoniske krav.

Hadde man bygget et rent kontorbygg er det en rekke forutsetninger som kunne gjort dimensjoneringen av ett høy-trehus mer gunstig. Ett eksempel på dette er de dynamiske kravene. I ISO-10137 er det en veiledning for hvilken vindindusert akselerasjon man kan tillate for ett bygg. ISO-10137 har to anbefalinger for vindindusert akselerasjon, disse er brukskrav for boliger og kontorer. Anbefalingen for kontorbygg tillater en høyere akselerasjon. I tillegg kan man tillate en høyere akselerasjon jo lavere egen-frekvens det er i konstruksjonen. Jo høyere konstruksjon man har desto lavere egen-frekvens får man. Ser man på ett rent kontorbygg med en større høyde en Mjøstårnet vil man kunne observere å få litt «snillere» brukskrav.

1.2 Målsetting og problemstilling

En av de største utfordringene ved å dimensjonere høye trekonstruksjoner er å tilfredsstille de dynamiske brukskravene. Tidligere forskning har sett på hvilke bæresystem som kan være det mest gunstige for høye trehus med tanke på vindindusert akselerasjon (Juveli, 2016). Det er også bygget enn rekke høye trehus i nyere tid (Liven, 2017). Det har vært stort fokus på forskjellige bæresystemer og kanskje ikke så mye forskning på hvordan de forskjellige bæresystemene presterer under forskjellige forutsetninger som økt bredde, dybde og høyde. Det kan derfor være interessant å se hvordan de dynamiske egenskapene til et bæresystem endrer seg med forskjellige forutsetninger og dette er grunnlaget for problemstillingen:

Hvordan endres de dynamiske egenskapene for et høyt trehus ved å endre fotavtrykk, massefordeling og tverrsnitt?

For å komme frem til en konklusjon av gitt problemstilling er det introdusert noen delproblemstillinger.

- Hvilket avvik man får man ved å redusere en fullstendig 3D modell til en 2D modell.
- Hvordan endres Egenfrekvens, Akselerasjon og Akselerasjonsutnyttelsen ved å endre fotavtrykk, massefordeling og tverrsnitt på en gitt modell?
- Har gitte modeller realistiske forskyvninger som gjenspeiler noe som er gjennomførbart i ett ekte byggeprosjekt?

For å svare på spørsmålene over er det tenk å sette opp et Python program som kalkulerer ett gitt bæresystem med en gitt dybde mellom $a \rightarrow b$, et begrenset antall tverrsnitt, et sett med massefordelinger samt en gitt bredde på konstruksjonen mellom $c \rightarrow d$.

1.3 Avgrensninger

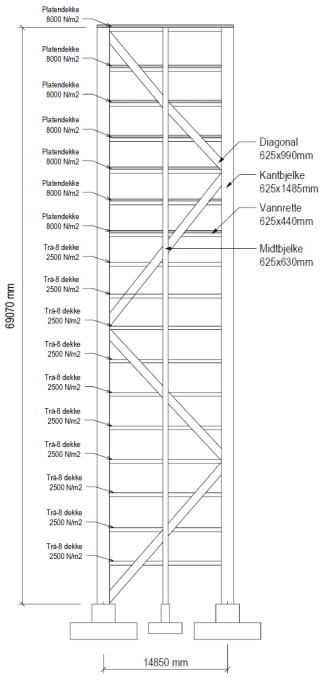
- Oppgavene er begrenset til de dynamiske egenskapene til kortsiden av modellene.
- Oppgaven er begrenset til første egensvingningsperioden og akselerasjonskrav for kontorbygg.
- Det er benyttet en 2D modell for finite-element modellering i Python.
- Det er antatt at vinden virker vinkelrett på byggets langside og at det ikke virke noe torsjon i konstruksjonen.
- For å begrense dataanalyseringen er det avgrenset til et bygg på 25 etasjer (100 meter).

2 Mjøstårnet i Brumunddal

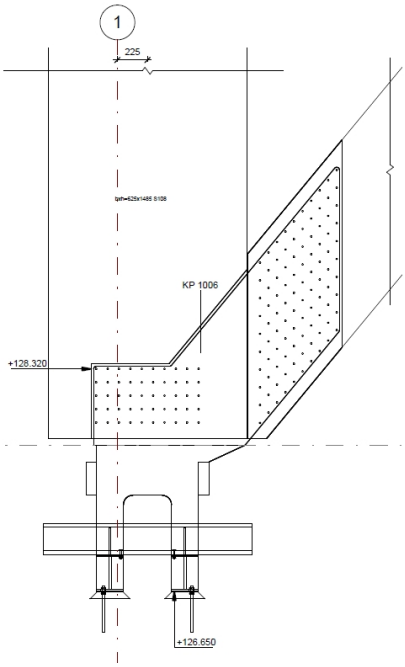
Bæresystemet til Mjøstårnet i Brumunddal er brukt som grunnlag i denne oppgaven. Ser man på en konstruksjon som en helhet er det mye mer enn bare statikk og dynamikk, en av de viktigste aspektene er gjennomførbarhet. Det ville vært likegyldig om en konstruksjon var «stabil» så lengde det ikke var «byggbart» eller «økonomisk gunstig». Siden Mjøstårnet i Brumunddal er et utprøvd konsept vil det være mye mindre usikkerhet rundt begrensningene i oppgaven.

Bygget er det som skal bli verdens høyeste trehus på 84 meter med 17 etasjer. Bygget skal bestå av hotell, kontor og boliger. Det er 5 etasjer med kontor, 4 etasjer med hotell og 6 etasjer med boliger i toppen. Netto areal er på 11300 m^2 med ett fotavtrykk på ca 629 m^2, hvor bredde med utv. fasade er 37 meter, dybde med utv. fasade er på 17 meter (Abrahamsen, 2017).

Bæresystemet består av blokklimte limtrebjelker og søyler. Horisontalavstivningen er ivaretatt med limtrediagonalene som vist i Figur 1. Midt i bygget er det en massivtre-sjakt for trapperom og heis, men denne brukes ikke som horisontalavstivning.



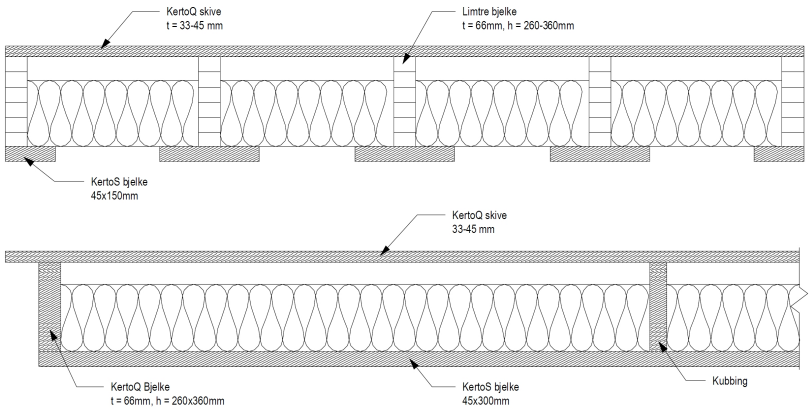
Figur 1 – Bæresystemet til Mjøstårnet



Figur 2 - Forankring diagonal/kantbjelke

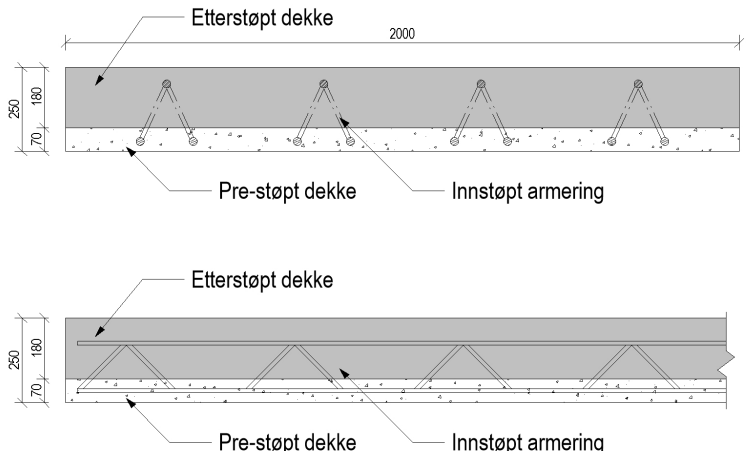
Etasjeskillene i Mjøstårnet består av Trä-8 bjelkelagselementer og plattendekker. Trä-8 bjelkelagselementene kommer fra bjelkelangselementene Ripa, forskjellen er at Ripa elementene har Kerto-S bjelker i steget. Trä-8 er bygget opp med Kerto-Q skive i overflensen, Limtrebjelke i steget og Kerto-S bjelker i underflensen (Ivarsson & Nellber, 2016) som vist i *Figur 3*. Tykkelsen på disse elementene er mellom 338 og 450 mm og maks spennvidde er rundt 7-8 meter.

Egenvekten av bjelkelangselementene uten gulvoppbygning eller himling er på 2000 N/m^2. Med ferdiggulv og himling er egenvekten 2500 N/m^2.



Figur 3 – *Trä-8 bjelkelagselement*

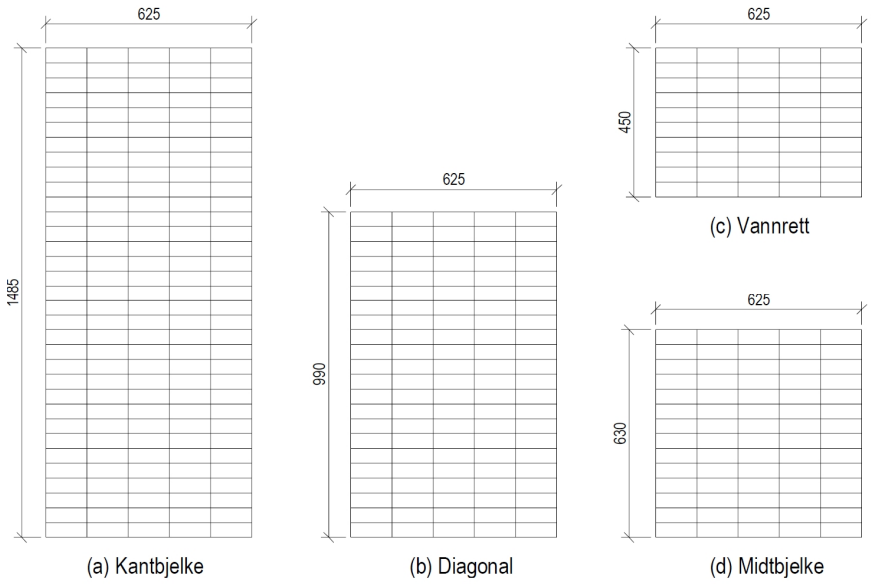
I de syv øverste etasjene er det etasjeskiller av 300mm betong. Det er benyttet Plattendekke-elementer i disse etasjeskillene. Disse elementene er konstruert som en type forskaling. Hovedargumentet for å benytte plattendekke er for å spare forskalings-arbeid. I Mjøstårnet er det helt nødvendig å benytte ett slik dekke da det ville vært svært kostbart om ikke umulig å bygge en forskaling til et plasstøpt dekke, noe annet pre-fabrikkert element som for eksempel hulldekker hadde blitt vanskelig på grunn av «lav» egenvekt.



Figur 4 – Plattendekke

Til vanlig trengs det stempling av plattendekker, men dekke i 10 etasje er spesialtilpasset med forhåndsstøpte bjelker for å slippe stempling av første etasjeskille i betong, resterende etasjeskiller stemples mot det ferdigstøpte dekke. For å tilfredsstille de dynamiske brukskravene var det nødvendig at etasjeskillen hadde en egenvekt på 8000 N/m^2. Forskning har vist at det er vanskelig å tilfredsstille lydkrav med lette tredekker (Sættem, 2016), dette var argumentet for å fordele den ekstra massen lengere ned i konstruksjonen, da det er lettere å tilfredsstille lydkrav med etasjeskiller av betong.

De blokklimte tverrsnittene i Mjøstårnet består av kvaliteten GL30c og GL30h. I bæresystemet for fagverket i kortsiden av bygget er det GL30c i alle elementer utenom kantbjelkene fra fundamentnivå og opp til fjerde etasje. Basert på kapasitets-beregningene i ULS er det å anta at disse elementene har klasse GL30h på grunn av høyere elastisitetsmodul og SLS krav. Oppbygningen av tverrsnittene er illustrert i Figur 5.



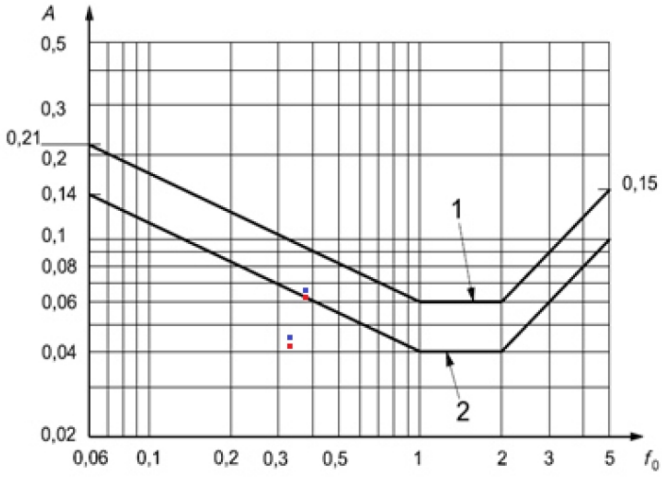
Figur 5 - Tverrsnitt i Mjøstårnet. Plassering er illustrert i Figur 1

Bredden på limtreblokkene er styrt av utformingen av slisseplatene. Som man kan se er bredden på alle elementene i fagverket mot kortsiden 625mm. Dette er for å få en enkel forbindelse mellom elementene. Figur 2 illustrerer forankringen mellom diagonalene og kantbjelkene til fundamentnivå. For å sette ting i perspektiv har hele detaljen en vekt på rundt 900 kg og er ca 3 meter høy. Alle forbindelsene i bæresystemet er dybel-forbindelser. De største trykk- og strekkreftene som virker i kantbjelkene er på 11500 kN og -5500 kN i strekk. De store strekkreftene kommer av det lave fotavtrykket samt den lette konstruksjonen. Senterlinje mellom kantbjelken er 14800 mm mot vindens angrepsflate på 37000 x 68000mm + pergola. Diagonalene får en kraft på ± 5000 kN. Beregninger i SLS tilstand viser en topp forskyvning på 140 mm, dette er akkurat innenfor anbefalingen på $H/500$.

Som nevnt tidligere har de dynamiske brukskravene i Mjøstårnet vært en utfordring. De mekaniske egenskapene til treverk gir en konstruksjon med høy stivhet i forhold til tyngde. Dette resulterer i konstruksjoner med lav egensvingningsperiode. Dette er som nevn tidligere grunnen til at de 7 øverste etasjene har etasjeskiller av betong. Egensvingningsperioden for kortsiden av bygget er på 2.70 sekunder (Abrahamsen, 2017). Den ekvivalente massen pr kvm er på 3730 kg/m^2.

Ved beregning av den vindinduserte akselerasjonen er det benyttet beregningsmetode C i NS-EN 1991-1-1-4 hvor den teoretiske egensvingningsformen fra *Robot structural analysis* er benyttet. Egenfrekvens, masse og egensvingeformer er beregnet i Robot Structural analysis heretter kalt RSA. Dempingsfaktoren som er benyttet i beregningen av grunnakselerasjonen er 1.90%, begrunnelsen her er basert på dempingsfaktoren angitt for limtrebruer og en masteroppgave fra NTNU som kom frem til at dempingen i Treet i Bergen var i nærheten av denne verdien (Hansen & Fjeld Olsen, 2016).

Akselerasjonen som virker i 17 etasje er 0.0628 m/s^2. kravet i ISO-10137 er på 0.062 m/s^2, dette gir en utnyttelse av brukskravet på $\approx$ 100%. I senere tid er det blitt bestemt å oppføre ekstra boenheter på taket til bygget. Akselerasjonen i denne etasjen vil være på 0.0669 m/s^2.



Figur 6 - Vindindusert akselerasjon i Mjøstårnet (Abrahamsen, 2017).

Røde punkter i Figur 6 illustrerer akselerasjonen i 17 etasje for første og andre egensvingeform. Blått punkt illustrerer akselerasjonen i 18 etasje.

Det statiske systemet er beregnet med en fullskala finite-element modell konstruert i RSA.

3 Teori

3.1 Finite element method

Finite element method heretter kalt FEM er en numerisk metode for å løse differensialligninger ved å dele opp en komplisert struktur til mindre avhengige elementer. FEM brukes i en rekke temaer som Statikk, Varmetransport og Fluiddynamikk. Selv om FEM i bunn og grunn er en metode for å løse differensial ligninger skal vi se nærmere på bruksområde for Statikk og dynamikk i konstruksjoner.

3.1.1 Statikk

Om man har studert Euler-Bernoulli bjelke teori vet man at forholdet mellom last og deformasjon er en differensialligning. Ligningen er vist under

$$\frac{d^2}{dx^2}\left(EI\frac{d^2\omega}{dx^2}\right) = q \qquad (1)$$

Hvor ω er deformasjon og q er lasten i kN/m. For at man skal kunne se på bjelken i sin helhet med en funksjon må q være kontinuerlig eller fult deriverbare over helle lengden x.

En viktig del av FEM innen konstruksjonsteknikk er den lokale stivhetsmatrisen $[k]$. Også kalt Elementmatrise. Denne elementmatrisen beskriver stivheten til alle frihetsgradene til ett element. Den baser seg på hook's lov som sier

$$F = kX \qquad (2)$$

Dette betyr i denne sammenheng at ett deformert element vil trenge en gitt mengde kraft for å deformere seg en gitt lengde basert på stivheten til elementet. Presentert i en modell med stort antall frihetsgrader får man hook's lov på matriseform som vist under

$$[k]\{d\}=\{F\} \qquad (3)$$

Hvor $\{d\}$ er en vektor som representerer deformasjon i en gitt frihet og retning. $\{F\}$ er en vektor som presenterer en gitt ekstern kraft i en gitt frihet og retning. Informasjon om hvordan man konstruerer en slik stivhetsmatrise er fint forklart i en rekke

faglitteratur som for eksempel «*matrisestatikk*» av Kolbein Bell (Bell, 2011) eller «*The Finit Element method in engineering*» av Singiresu S. Rao (Rao, 2017). Den mest kjente stivhetsmatrisen er den konstruert etter Euler Bernoullis bjelketeori med tre frihets-grader.

$$k_{elem} = \begin{bmatrix} \frac{AE}{L} & 0 & 0 & \frac{-AE}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & \frac{-12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & \frac{-6EI}{L^2} & \frac{2EI}{L} \\ \frac{-AE}{L} & 0 & 0 & \frac{AE}{L} & 0 & 0 \\ 0 & \frac{-12EI}{L^3} & \frac{-6EI}{L^2} & 0 & \frac{12EI}{L^3} & \frac{-6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & \frac{-6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \quad (4)$$

Begrensingene til denne stivhetsmatrisen er at man kun kan sette inn krefter i nodene på elementet, med noder mener man Start og slutt punkt på et element. Er man i ett tilfelle der man vil sette inn en kraft midt på en bjelke må man splitte elementet til to mindre elementer. Da er man innen på den globale delen av FEM for konstruksjoner. For å analysere en hel konstruksjon må man sette de lokale stivhetsmatrisene i en global stivhetsmatrise som beskriver en konstruksjon i sin helhet. Metoden for å konstruere en slik global stivhetsmatrise er på engels kalt «*Direct stiffness method*» og er godt forklart i boken «*The Finit Element method in engineering*» av Singiresu S. Rao. Prinsippet går ut på å bygge stivhetsmatrisen på en systematisk og generalisert metode som gjør det enkelt å automatisere mye av beregningene med hjelp av programmering.

På samme måte som for den lokale stivhetsmatrisen får man et likningssett som ser slik ut

$$[K]\{d\} = \{F\} \quad (5)$$

Hvor [K] er den globale stivhetsmatrisen, {d} er forskyvning av de globale frihetene og $\{d\}$ er kraften som virker i de globale frihetene. I Teorien kunne man løst dette likningssettet med kjent $[K]$ og $\{F\}$ for å finne den ukjente forskyvningen $\{d\}$, men for at likningsettet ikke skal resultere i å være singulært må man introdusere noen begrensninger. Og begrensninger er her opplagerene, for å tilføre opplagere til modellen er det vanlig å nulle ut raden og kolonnen til friheten som har en begrensning. Definerer

man nok opplagere vil man slippe unna problemet med en singulær løsning, altså problemet med at det er uendelig mange løsninger av {d}.

Vil man se på de lokale kreftene som oppstår i ett lokalt element $[k]$ kan man multiplisere elementmatrisen $[k]$ med de tilhørende frihetene fra den globale deformasjonsmatrisen {d}. Denne operasjonen er ofte benevnt med

$$\{f\} = [k][d] \tag{6}$$

Hvor $\{f\}$ er den lokale kraften som virker i ett gitt element. $[k]$ er den lokale stivhetsmatrisen og $[d]$ representerer de globale forskyvningene som tilhører frihetene til den lokale stivhetsmatrisen [k].

3.1.2 Dynamikk

Grunnprinsippet med dynamiske problemer innen konstruksjonsteknikk går ut på at forskyvninger, bevegelser og krefter er tidsavhengige. Ser man på ett system med 1 frihetsgrad kan man definere en bevegelseslikningen som følger

$$m\,\ddot{u}(t) + C\,\dot{u}(t) + k\,u(t) = p(t) \tag{7}$$

Hvor

- $m = masse$
- $\ddot{u} = m/s^2$
- $c = Dempningsfaktor$
- $\dot{u} = m/s$
- $k = Stivhet$
- $u = m$

I ett system med flere frihetsgrader kan likning (7) omformuleres til matriseform som vist under.

$$[\boldsymbol{m}]\,\{\ddot{\boldsymbol{u}}\} + [\boldsymbol{c}]\,\{\dot{\boldsymbol{u}}\} + [\boldsymbol{k}]\,\{\boldsymbol{u}\} = \{\boldsymbol{p}(t)\} \tag{8}$$

Hvor

- $[\boldsymbol{m}] = massematrise$
- $[\boldsymbol{c}] = dempningsmatrise$
- $[\boldsymbol{k}] = Stivhetsmatrisen\ som\ beskrevet\ i\ kapittelet\ over.$

Om man legger til en forskyvning ved tiden t = 0 og fjerner begrensningen for forskyvning vil man oppleve at systemet vil få en harmonisk svingning. Svingningen vil være avhengig av massen og stivheten til konstruksjonen. Om det er demping i systemet vil svingningenes amplitude avta med tiden t. Om man kan tenkte seg ett system hvor det ikke eksisterer noe demping vil svinginene aldri avta med tiden t. Svingningene til ett slik system vil opptre i frekvenser kalt naturlige egenfrekvenser, i tillegg vil det opptrer deformasjonsmønstre kalt egensvingningsformer. Den laveste egenfrekvensen i systemet blir ofte kalt den fundamentale egenfrekvensen.

I boken «*The Finit Element method of structures*» (Rao, 2017) er det utledet hvordan man kan gå fra ligning (8) til ett egenproblem. Egenverdiproblemet er vist under

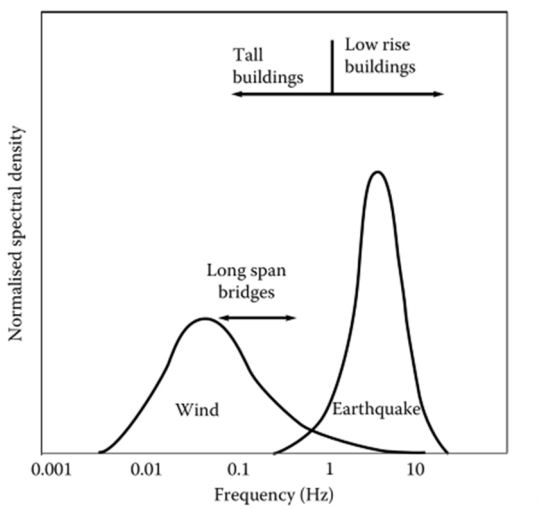
$$\left[[k] - \omega^2[m]\right] = 0 \qquad (9)$$

Det finnes flere måter å løse dette egenverdiproblemet på. Det finnes hovedsakelig to fremgangsmåter for å løse egenverdiproblemet. Det er transformasjonsmetoder som Jackobi, Givens og Householder schemes, eller så har man iterative metoder. Transformasjonsmetodene er som regel å foretrekke om man må ha alle egenverdiene, er man ute etter de første eller siste egenverdiene er det mer vanlig å bruke iterative metoder (Rao, 2017).

Det finnes en rekke metoder for å konstruere massematriser, men de to vanligste metodene er på engelsk kalt *'consistent'* og *'lumped mass'*. *'consistent mass'* er utledet på samme måte som den for stivhetsmatrisen nevnt i delkapittel «3.1.1 - Statikk». Denne metoden blir ofte brukt der man ser på de dynamiske egenskaper for lokale elementer. Noen eksempler hvor det kan være hensiktsmessig å bruke *'consistent mass'* matrise er med utkragede bjelker eller der hvor det vil være nødvendig å studere egensvingningsformen til lokale elementer. En av de negative effektene til en *'consistent mass'* matrise er kompleksiteten, det kan by på utfordringer ved beregningene av egenløsningene da man får en komplisert matrise. *'Lumped mass'* er en enklere, men nødvendigvis ikke en mer unøyaktig metode sammenlignet med *'cosistent mass'*. Metoden *'lumped mass'* egner seg mest i situasjoner hvor man setter inn store masser i lette systemer. *'lumped mass'* matrise er også fordelaktig med det faktumet at masse matrisen blir en diagonal n*n matrise, noe som resulterer i mye enklere beregninger for å løse egenproblemet nevnt i likning (9).

3.2 Vind

I Norge er vind stort sett den dominerende horisontallasten som virker på en konstruksjon. vindlast er stort sett er den dominerende lasten for høye konstruksjoner på grunn av frekvensspekteret til vind og egenfrekvensen til høye konstruksjoner. Figur 7 viser hvordan vinden opererer i en mye lavere frekvens enn det jordskjelv gjør.



Figur 7 – Spektraldensitet (Holmes, 2015)

Vindkraften F_w som virker på en konstruksjon eller en konstruksjonsdel kan bestemmes ved å summere følgende krefter etter NS-1991-1-4 (Standard, 2009)

Utvendig krefter:

$$F_{w,e} = C_s C_d \sum_{overflate} W_e * A_{ref} \quad (10)$$

Innvendige krefter:

$$F_{w,i} = \sum_{overflate} W_i * A_{ref} \quad (11)$$

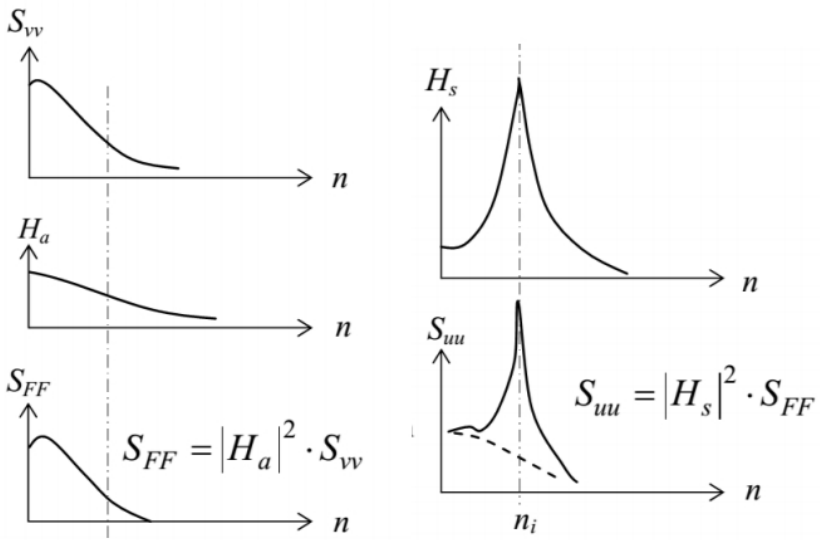
Friksjonskrefter:

$$F_{fr} = C_{fr} * q_p(z_e) * A_{fr} \quad (12)$$

Selv om det i NS-EN 1991-1-4 punkt 6.2 (Standard, 2009) fremmes en rekke punkter som gjør det mulig å neglisjere den dynamiske virkingen til vind er det i noen situasjoner

nødvendig å se på den dynamiske effekten til vinden. I høye konstruksjoner vil det være nødvendig å kontrollere konstruksjonen for forskyvninger og akselerasjon med hensyn til den dynamiske responsen til konstruksjonen.

Vind er et stokastisk fenomen, det betyr at det er svært vanskelig å beregne den eksakte responsen vinden vil ha på en gitt konstruksjon. Men med hjelp fra en spektralanalyse er det mulig å finne den verst mulige effekten som kan ramme en gitt konstruksjon. A.G Davenport utledet en slik spektralanalyse på 1960 tallet (Davenport, 1961).



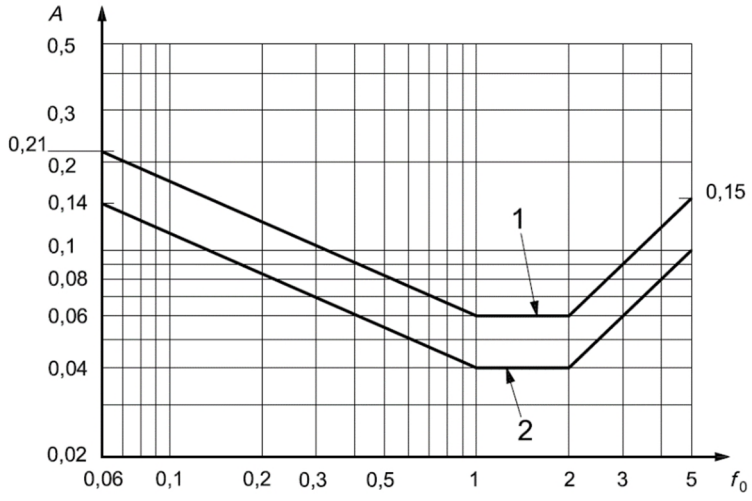
Figur 8- Spektralanalyse av vind (Steenbergen et al., 2009)

En generell fremgangsmetode for å finne den vindindusert responsen som er i en konstruksjon er illustrert i Figur 8. Første punkt er en spektralfremstilling av vindhastigheten S_{vv} som er avhengig av frekvensen (n). Lastspektret S_{FF} er kalkulert med influensfunksjonen H_a, som er avhengig av den eksponerte lastflaten samt type overflate. S_{FF} er en funksjon som beskriver vindlasten i kraft med hensyn til vindens frekvens. Når Vindkreftene er kjente vil neste mål være å estimere konstruksjonens respons med funksjonen S_{uu} ved å bruke influensfunksjonen H_s som reflekterer de mekaniske egenskapene til bygget. Responsen til bygget vil være avhengig av konstruksjonens frekvens. Så responsen til S_{uu} vil bli amplifisert i området nær egenfrekvensen til konstruksjonen (Talja & Fülöp, 2016).

NS-EN 1991-1-4 introduserer to forenklede metoder for å regne ut den vindinduserte-akselerasjonen i en gitt konstruksjon. Det er metode 1 og 2 (Steenbergen et al., 2009). Disse metodene ligger som tillegg B og C i NS-EN 1991-1-4. Det viser seg at de to

metodene kan gi svært forskjellige resultater. I en forskningsartikkel av Raphael Steenbergen med flere ble det observert at beregninger etter metode 1 kunne underestimere akselerasjonen ned mot 40% sammenlignet med numeriske beregninger. Metode 2 overestimerte akselerasjonen opp mot 5% sammenlignet med den teoretiske akselerasjonen. Konklusjonen var at metode 2 er å foretrekke da beregninger av den vindinduserte akselerasjonen med metode 1 gjorde forenklinger av egensvingeformen som skapte unøyaktige resultater (Steenbergen et al., 2009).

NS-EN 1991-1-4 forholder seg til vedlegg D i ISO 10137 for brukskrav med tanke på vindisert akselerasjon. I Figur 9 kan man se akselerasjonskravet for henholdsvis kontor og bolig. Hvordan menneske opplever bevegelser i høye konstruksjoner er subjektivt (Juveli, 2016). hver enkelt person vil oppleve bevegelser i høye konstruksjoner på sin egen måte, på grunn av dette er det gjort sensitivitets undersøkelser som har testet forskjellige akselerasjoner for å se hvor stor prosentandel testpersoner som opplev ubehag ved forskjellige frekvenser og akselerasjoner (Boggs, 1995), på grunn av dette er det ingen International standard for fremgangsmåte for å finne dette brukskravet (Juveli, 2016).



Figur 9 – Akselerasjonsgrense for 1-(kontor) og 2-(Bolig) fra ISO 10137.

3.3 Dataanalyse og forsøk

For å forstå deler av analysen i denne oppgaven er det nødvendig å gå gjennom noen konsepter innen dataanalyse.

I et vanlig 2-dimensjonalt plot med en x- og y-akse har man muligheten til å beskrive sammenhengen mellom to variabler på en visuell måte. Den mest fundamentale funksjonen man kan plote i en slik 2d plot er funksjonen $f(x_1) = y$. Med en slik funksjon plotet i ett 2d kartesisk koordinatsystem kan man intuitivt hente ut informasjon om sammenhengen mellom $f(x_1)$ og y. Man kan for eksempel se om den varierende variabelen x_1 gjør at den avhengige variabelen y går opp eller ned.

Introduserer man en ny variabel x_2 vil det bli verre å illustrere sammenhengene mellom $f(x_1, x_2) = y$ i ett 2d koordinatsystem. Med tre variabler vil man få ett 3 dimensjonalt datasett. Man kan introdusere en ny akse kalt z, da har man muligheten til å presentere variabel x_1 i akse-x, x_2 i akse-y og den siste avhengige variabelen i akse-z. da vil man gå fra en graf i et 2d plot til ett plan i ett 3d koordinatsystem.

Går man videre og introduserer enda en avhengig variabel får man i prinsippet to 3-dimensjonale datasett, i tilfeller der man sitter med en funksjon $f(x_1, x_2) = y_1, y_2$ kan man plote ett plan i ett 3d koordinatsystem hvor avhengig variabel y_1 er presentert i akse-z når avhengige variabel y_2 er presentert med en farge i punktet x_1, x_2, y_1. Dette kan man gjøre fordi de avhengige variablene y_1 og y_2 opptrer i samme punkt. Hadde man istedenfor hatt et datasett hvor man sitter med de varierende variablene $f(x_1, x_2, x_3) = y$ ville det blitt verre å presentere dataen visuelt i ett 3d koordinatsystem.

Informasjonen over er veldig elementært og vil virke som en selvfølge for de fleste, men poenget er å understreke problematikken med å visualisere og undersøke datasett med flere variabler en de tre dimensjonene x, y og z som man er så kjent med fra det kartesiske koordinatsystemet.

3.3.1 Datasett med to eller flere dimensjoner.

Det finnes flere metoder for å undersøke sammenhenger i et datasett. En metode blir kalt COST som står for *'One factor at a time'* den går ut på å forandre en variabel i forvente av å registrere endringen av den forventete variabelen. En slik metode vil i

mange tilfeller gi feiltolkninger. Grunne til dette er fordi det kan være underliggende variabler som er avhengig av den variabelen man ønsker å forandre (Dunn, 2018). Ser man for eksempel på et konstruksjonseksempel kan man vise hvordan en slik COST analyse vil kunne gi feiltolkning. Ett kjapt eksempel på dette kan være om man skulle ha foretatt ett eksperiment ved å teste hvordan en limtrebjelke bøyer seg uten å ta forbehold om de klimatiske forholdene under forsøket. Selvfølgelig vet man pr dags dato at fuktigheten til en limtrebjelke har stor innvirkning på nedbøyningen, men i tiden før forsøket vet man ikke alltid hvordan kjente og ukjente variabler henger sammen.

En vanlig metode for å kontrollere sammenhengen som nevn over er *full factorial design*. I en *full factorial design* finner man ekstremalpunkter av variablene man benytter i ett forsøk. Man kan teste korrelasjonen mellom så mange variabler man ønsker med det vanligste er å teste to og to variabler. Antall variabler man ønsker å teste opp mot hverandre kalles nivåer.

Se for seg ett forsøk med faktoren f_1, f_2 som vist i Tabell 1.

Tabell 1 - *Faktorer*

Faktor	Lavt nivå [mengde]	Høy nivå [mengde]
f_1	a_1	a_2
f_2	b_1	b_2

Prinsippet for en *full factorial design* er da å gjøre forsøk med alle mulige kombinasjoner av faktorene vist i Tabell 1.

Tabell 2 - *Antall mulige forsøk*

Forsøk	x_1	x_2	$Resultat$
1	a_1	b_1	y_1
2	a_2	b_1	y_2
3	a_1	b_2	y_3
4	a_2	b_2	y_4

Når man har gjennomført alle forsøkene kan man studere hovedeffekten mellom variablene. En tolkning av hovedeffekten kan være hvordan stigningstallet til f_1 med

hensyn til y endrer seg på grunn av f_2. Matematisk kan man se på dette som ett regresjonsproblem. Ser man for seg det vanlige tilfelle med en regresjon som vist under

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + e = y \qquad (13)$$

For det arbitrære forsøket vist i Tabell 2 - *Antall mulige forsøk* kan ligning (13) settes opp i matriseformen $\boldsymbol{y} = \boldsymbol{Xb} + \boldsymbol{e}$ som vist under

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & -x_1 & -x_2 & (-x_1 - x_2) \\ 1 & +x_1 & -x_2 & (+x_1 - x_2) \\ 1 & -x_1 & +x_2 & (-x_1 + x_2) \\ 1 & +x_1 & +x_2 & (+x_1 + x_2) \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \qquad (14)$$

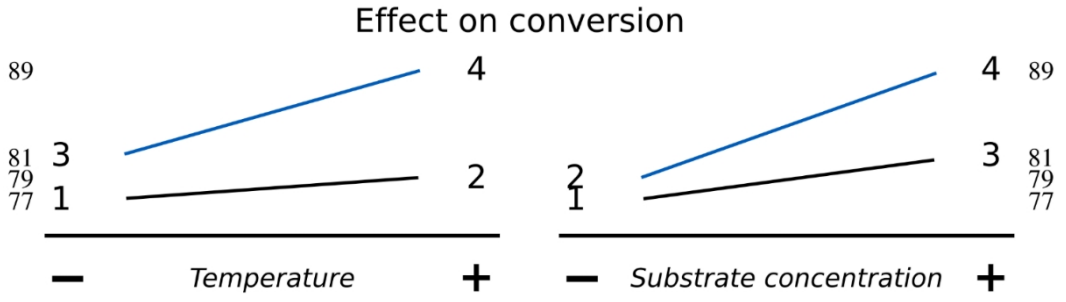
For å se på effekten som skjer inne i forsøksrommet $a_1 \rightarrow a_2 \; og \; b_1 \rightarrow b_2$ kan man normalisere variablene til $a_1 = 0, a_2 = 1, b_1 = 0, b_2 = 1$, man får da ett ligningssett som vist under.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} \qquad (15)$$

Da finne man regresjonskoeffisientene på følgende måte.

$$\boldsymbol{b} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \qquad (16)$$

Vektoren $\boldsymbol{b}$ er regresjonskoeffisienten som forteller oss hvordan de forskjellige variablene gjør utslag på den avhengige variabelen y. regresjonskoeffisienten β_3 forteller oss hvordan stigningstallet til x_1 endres på grunn av x_2 når x_2 er konstant og vice versa. En vanlig måte å visualisere en slik interaksjon er med et interaksjonsplot.



Figur 10 – Integrasjons plot (Dunn, 2018)

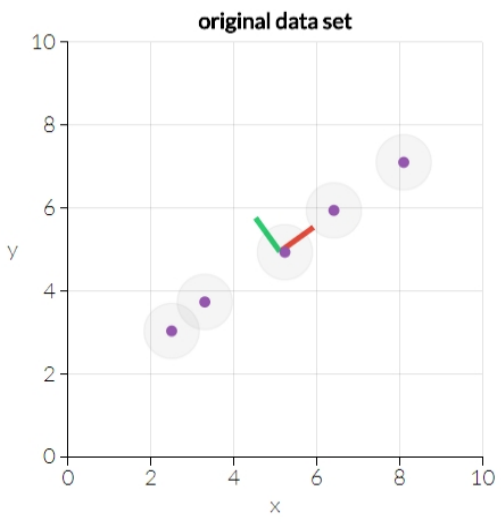
Figur 10 er tatt ut fra ett eksempel i boken «*Process imporvment using data*» *(Dunn,*

2018) hvor det er foretatt en full factorial design av to variabler med to nivåer. Tallene 1,2,3 og 4 i Figur 10 representerer forsøkene som er foretatt på samme måte som illustrert i Tabell 2. Tallene ved siden av forsøksnummereringen representerer den resulterende variabelen y_i. Ploten til venstre illustrerer hvordan temperaturen endres fra laveste nivå til høyeste nivå. Den svarte grafen viser hvordan temperatur endres når *'Substrate concetration'* er på sitt laveste nivå, blå graf viser hvordan temperatur endres fra laveste til høyeste nivå når *'Substrate concetration'* er på sitt høyete nivå. Ploten til venstre er satt sammen på samme måte. Der kan man visuelt se hvordan *'Substrate concetration'* endrer oppførselen til temperatur med hensyn til den resulterende variabelen y. om den svarte og blå grafen hadde ligget helt parallelt med hverandre ville det vært indikasjon på at *'Substrate concetration'* ikke hadde noe effekt på temperaturen.

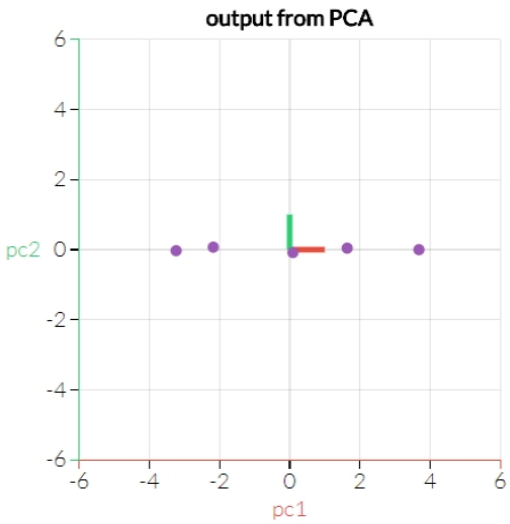
Det fullstendige eksemplet er i kapittel 5.8.1 i boken *Process imporvment usning data av Kevin Dunn*.

En annen måte å visualisere multivarierende data er ved hjelp av latente variabler. En latent variabel er en variabel som ikke kan måles direkte, latente variabler blir ofte brukt i statistikken der den predikerende variabelen ikke har en bestemt variabel som definerer den resulterende verdien. Eksempler på dette kan være alt fra følelser, smak og hvordan man opplever temperatur i ett rom. Det er kanskje vanskelig å se hvordan dette kan relateres til en multivariabel studie av et høyt trehus, men poenget er at man kan finne underliggende variabler som beskriver noe man ikke har full kontroll på. En type analyse som tar for seg disse latente variablene er '*Principal component analysis*' heretter kalt PCA. Grunnprinsippet i en PCA er å finne 'retning' i ett multivariabelt datasett som beskriver mest mulig av variansen til dataen.

Victor powel og Lewis Lehe har laget en interaktiv visualisering som viser hvordan en PCA fungerer og hva man kan bruke en slik PCA til (Powell & Lehe).

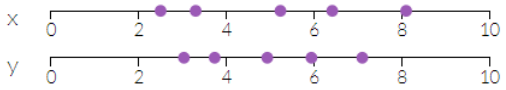


Figur 11 - 2D datasett (Powell & Lehe)

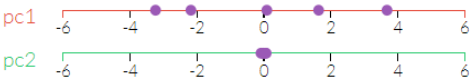


Figur 12 - 2D datasett gjenspeilet i Prinsipalkomponenter (Powell & Lehe)

På Figur 11 ser man en 2D plot, det er ikke så vanskelig i dette plotet å se sammenhengen mellom x og y, men ved hjelp av en PCA kan man gjøre en dimensjonsreduksjon ved å transformere x- og y-koordinatene slik at de målte verdiene i Figur 11 ligger vannrett i PCA plotet vist i Figur 12. I det originale 2D plotet må man ha både x og y koordinatene for å kunne beskrive variansen til datasettet, men i PCA plotet ser man hvordan det er mulig å forkaste koordinat pc2 fordi det ikke er noe variansen i pc2-koordinatet. Sammenhengen er videre illustrert i Figur 13 og Figur 14 hvor koordinatene x- og y samt koordinatene pc1- og pc2 er plotet i separate 1dimensjonale plot.



Figur 13 - 2 x 1D plot av 2D datasettet (Powell & Lehe)



Figur 14 - 2 x 1D plot av prinsipalkomponentene (Powell & Lehe)

Det er ikke alltid slik at man kan gjøre en PCA i forvente av å kunne redusere dimensjonene på datasettet. Grunnen til at det var mulig å redusere dataen i Figur 11 ned til 1 dimensjoner er fordi den underliggende funksjonen som definerte dataen hadde kun 1 dimensjon.

Prinsipalkomponentene $pc1, pc2. pc3\ osv..$ vil være rangert etter hvilke komponent som beskriver mest mulig variansen i datasettet. Pc1 vil være den prinsipalkomponenten som beskriver mest av variansen til det n-dimensjonale datasettet. Pc2 vil være den prinsipalkomponenten med nest mest varians osv…

En forgrening av PCA er *partial least squares* heretter kalt PLS. På samme måte som med PCA finner man retninger som forklarer mest mulig varians av ett datasett. Forskjellen mellom PCA og PLS er at man i PCA prøver å forklare hele datasettet, med PLS er målet å forklare mest mulig varians med hensyn til en resulterende variabel y (Dunn, 2018). Ved å foreta en slik PLS analyse kan man få ett innblikk i hvordan datasettet er satt sammen og hvor mange underliggende funksjoner som er med på å beskrive variansen til en gitt resulterende variabel. For å få en bedre forståelse på hvordan resultatene til en slik PLS analyse kan tolkes er det anbefalt å lese artikkelen «*Interpretation of partial least squares regression models by means of target projection and selectivity ratio plots*» (Kvalheim, 2010)

4 Metode

Metoden i denne oppgaven baserer seg på en fremgangsmåte veldig likt en '*full factorial design*' som vist i kapittel «3.3.1 - Datasett med to eller flere dimensjoner.». Ved å lage ett Python-program som regner ut de dynamiske egenskapene med de gitte variablene Bredde, Dybde, Massefordeling og Tverrsnitt kan man lage ett sett modeller som inneholder alle mulige sammensetninger av disse variablene. Videre kan man kartlegge hvilke variabler som gir varians til de dynamiske egenskapene.

Sammensetningen av variablene nevnt over er definert i kapittelet *«4.1*-Variabler». Med de gitte variablene vil man få ett 5-dimensjonalt datasett, det vil gi utfordringer med å visualisere sammenhengen til de varierende variablene. For å løse denne utfordringen er det gjort *Partial least square* analyser for de resulterende variablene for å få en overordnet oversikt over sammenhengen mellom variablene og de dynamiske egenskapene til modellene. Videre er det tenkt å se på sammenhengen mellom variablene og de dynamiske egenskapene ved å plote 2-dimensjonale figurer hvor et sett variabler blir satt til konstante verdier.

4.1 Variabler

Det var i utgangspunktet tenkt å se på varierende høyder mellom 60- og 100 meter men det har blitt begrenset til 100 meter. I Mjøstårnet varierer etasjehøyden mellom 3800 mm og 4400 mm. For å gjøre beregningene enklest mulig er etasjehøyden satt til 4000 mm. Dette betyr at man må ha 25 etasjer for å komme opp til 100 meter.

4.1.1 Geometri

Det ligger ikke noe spesielt til grunn for valg av grensene til geometrien annet enn erfaringstall fra typiske planløsninger (Svendsen, 2016). Gjennomgående planløsninger er breie med slanke bygg for å tilfredsstille kravene om dagslys. Det er ikke sikkert at en sammensetning av bredde og dybde vil tilfredsstille arkitektoniske krav som for eksempel dagslys og arealeffektivitet, men denne oppgaven er avgrenset til de dynamiske egenskapene så dette blir sett bort fra. Grensene for dybde variabelen i denne oppgaven er begrenset til minstemål på 14000 mm, største dybdemål er satt til 25000

mm. Bredde variabelen er begrenset til minstemål lik dybde, største mål er begrenset til 35000 mm.

Økningen på geometri variablene vil være 1000 mm. Dette resulterer i 253 forskjellige sammensetninger av geometri parameterne.

4.1.2 Masse

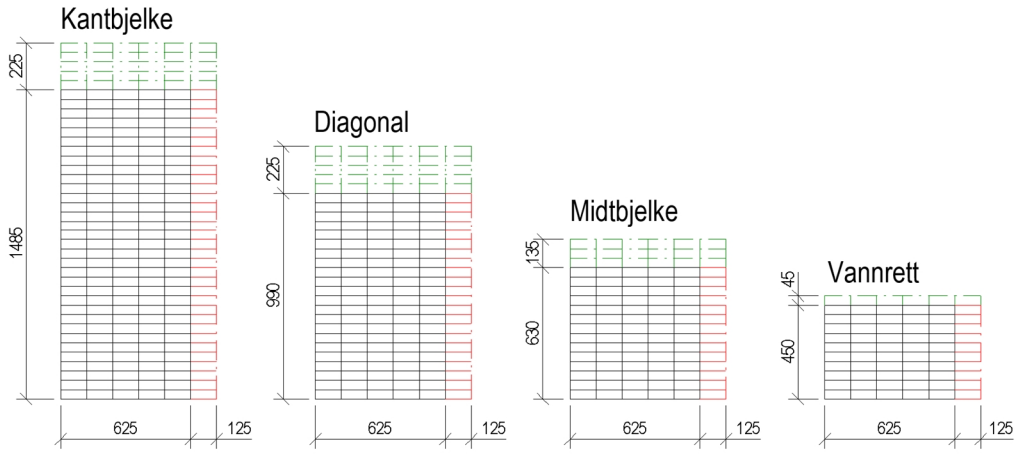
Massen til alle modellene vil variere med massefordelingen av dekkene. Det er benyttet to forskjellige dekker i denne oppgaven. Første dekke er Trä-8 dekke som har en relativ lav egenvekt da det er bygget opp med kerto-bjelker og finerplater som vist i kapittel «*2 - Mjøstårnet i Brumunddal*». Egenvekt av trä-8 dekke med ferdiggulv er antatt å være $2.5\ kN/m^2$. Det andre dekke som er benyttet er plattendekke. Egenvekt av disse er med ferdiggulv $8.0\ kN/m^2$.

Fordelingen av disse dekkene er tenkt med utgangspunkt i trä-8 dekke i alle etasjer utenom øverste etasje som har plattedekke. Neste beregning er tenkt med plattendekke i de to øverste etasjene osv.

Sidene det er 25 etasjer vil det bli 24 forskjellige massefordelinger.

4.1.3 Tverrsnitt

Det vil bli stor variasjon i kreftene på de forskjellige modellene. Det er derfor nødvendig å se på forskjellige sammensetninger av tverrsnitt. Minstemål på tverrsnittene vil være målene til Mjøstårnet med forskjellige økninger i høyde og bredde. I Figur 15 er økningen for de forskjellige tverrsnittene illustrert. Økningen av høyde vil være styrt av antall lameller hvor en lamell er 45mm. Økningen av bredde er styrt av bredden til limtreblokkene. En limtreblokk er 125mm.



Figur 15 – Tverrsnitt - modell

Bredden av alle tverrsnitt er like. Dette er på grunn av innfestningen mellom tverrsnittene som trenger like mål på grunn av slisseplatene til forbindelsene. Økningen på bredden er tenkt 625mm + 4 ganger 125mm. Tabell 3 viser utgangspunktet og antall økninger for de forskjellige tverrsnittene. Benevningen og plassering av tverrsnittene er de samme som beskrevet i kapittelet *«2 - Mjøstårnet i Brumunddal»*

Tabell 3 - Tverrsnitts sammensetning

Type	Minste høyde	1 økning	Antall økninger
Diagonal	990mm (22 lameller)	5 lameller	3 stk
Kantbjelke	1485mm (33 lameller)	5 lameller	3 stk
Midtbjelke	630mm (14 lameller)	Kvadratisk (styrt av bredde)	
vannrett	450mm (10 lameller)	1 lamell	3 stk

Basert på dette ender det opp med 324 forskjellige tverrsnittkombinasjoner.

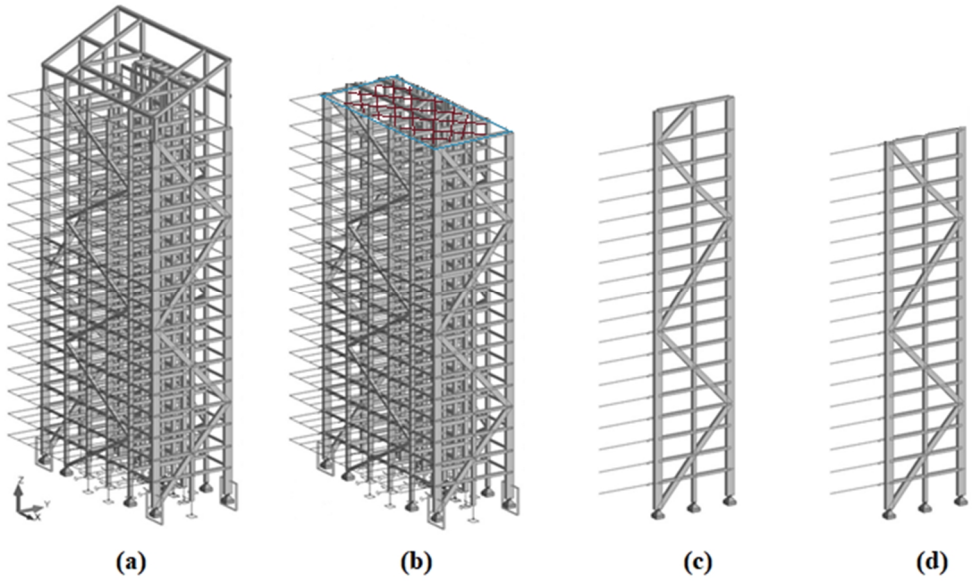
4.1.4 Oppsummering

Basert på variablene over vil det bli generert ca 1 500 000 modeller som skal beregnes I Python-programmet. Basert på erfaringer tar det ca $\approx 0.2\,Sekunder$ å kalkulere en modell. Dette resulterer i en antatt kalkuleringstid på 84 timer. Estimert filstørrelse på resultatene er 1 gigabyte.

4.2 Verifisering av arbeidsmetode

Det er gjort en rekke kontroller for å sjekke hvilken innvirkning forenklingene gjort i denne oppgaven har på resultatene. For å kunne sammenligne Python modellene med tanke på forenklingene har det vært nødvendig å modifisere den originale 3D modellen til Sweco. Modell (a) og (b) viser forenklingene som er gjort på 3D modellen til Sweco. I modell (a) er opplagerene med fjærstivhet mht. pelene til bygget byttet ut med vanlige leddlagere uten noe fjærstivhet. I modell (b) er pergolaen fjernet. Det var nødvendig å fjerne pergolaen for å kontrollere parametersettingen av Python modellene, da pergolaen bidrar til kompliserte beregninger av vind på lokalt nivå.

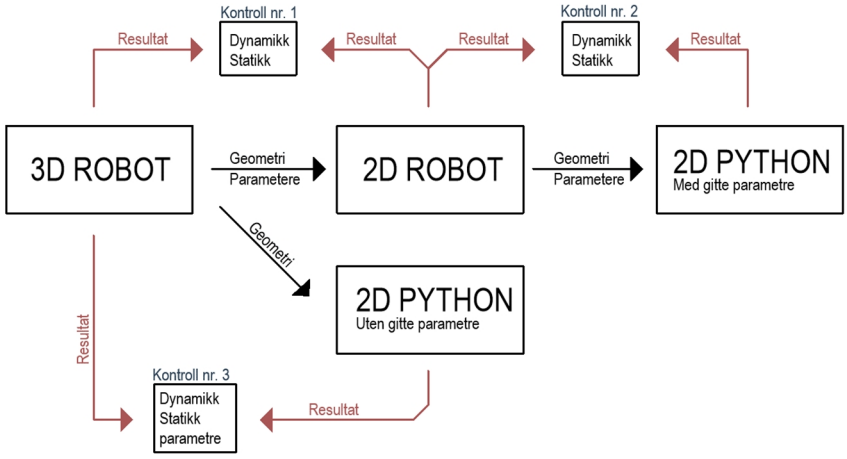
Modell (a) har en masse i 18 etasje (Pergola), for å kunne sammenligne modell (a) og (c) er det nødvendig å få lagt in masse i samme høyde. Det er derfor lagt inn en ekstra etasje på fagverket i modell (c) som skal simulere denne massen. I modell (d) er fagverket redusert ned til 17 etasjer. Grunnen til dette er fordi denne modellen brukes til å sammenligne Python-programmet med Robot strucutural analysis modell heretter kalt RSA.



Figur 16 – Beregningsmodeller

Som forklart under er det 3 kontroller som blir gjennomført for å verifisere og avklare avvik som kan komme fra forenklingene gjort i denne oppgaven. Figur 17 illustrerer fremgangsmåten for disse kontrollene. De svarte pilene representerer hvilken

informasjon som blir sendt over fra en modell til en annen. Røde pilene illustrerer hvilke modell-resultater som sammenlignes med hverandre.



Figur 17 – Visualisering av resultat verifisering.

4.2.1 Kontroll 1

Kontroll 1 består av en sammenligning av resultatene til modell (a) og (c). Formålet med kontroll 1 er å sjekke at oppførelsen til de to modellene er like. For å teste forskyvningene til de to modellene er det introdusert ett forenklet lasttilfelle, som er en linjelast 7.72 kN/m på dekkekant oppover i modell (a). modell (c) har en punktlast på 140 kN oppover i etasjene. Punktlasten på 140 kN er resultantkraften til linjelasten nevnt over. Massen fra modell (a) er tatt ut fra fanen *Values* i tabellen *Stories* i RSA modellen og omgjort til nodal-masser i modell (c). Massen fra RSA modellen og omgjøringen til nodal-masse er presentert i «Vedlegg A: Tabell 11 - Masse fra 3D robot modell.» fordelingen av masse pr etasje er gjort ved å tildele kant nodene 1/8 av etasjemassen, ¼ av etasjemassen er tildelt midtnoden. Resultatene fra disse kalkulasjonene er Frekvens, Periode, Egensvingeform, relativmasse, forskyvning og forskyvningsform. Formen kontrolleres fra grunnivå til dekke i 17 etasje.

4.2.2 Kontroll 2

Kontroll 2 har som formål å kontrollere at finite-element beregningen i Python programmet er riktig. Det er da modell (c) som sammenlignes med modell (d) beregnet i Python. Last- og masseforutsetningene er de samme som for modell 1 med ett unntatt.

Massen fra pergolaen er fjernet i denne kontrollen. Resultatene her er de samme som for kontroll 1 med unntak av den relative massen som ikke er kontrolert.

4.2.3 Kontroll 3

Kontroll 3 er en sammenligning av frekvensen og maks forskyvning av modell (b) og modell (d) beregnet i Python uten å overføre last- og massefordeling. I denne kontrollen er det tenkt å se hvordan parametriseringen av last og masse fra Python programmet fungerer. Resultatene som er tenkt sammenlignet er Frekvens, periode og maks forskyvning.

4.3 Python-program

Python-programmet har en rekke moduler med forskjellige funksjoner.

Det er to hovedmoduler i python-programmet. Den første modulen er lagd for å sette sammen alle mulige modeller og behandle resultatene til alle modellene. Dette er modulen som ligger i «*Vedlegg C – Python-modul modeller.py*». Andre modulene er Parametrisering. Denne modulen går gjennom alle beregningene for en spesifikk modell. Denne ligger i «Vedlegg D – Python-modul Parametrisering.py»

Parametrisering.py benytter i hovedsak tre moduler for å beregne en gitt modell. De er Statikk.py, Dynamikk.py og Elementsjekker.py. Statikk.py og Dynamikk.py er moduler som gjør beregninger etter finite element method. Elementsjekker.py består av funksjoner som gjør beregninger fra eurokodene, fordeler laster og masser osv.

Python-programmet gjør ingen forhåndsdimensjonering. Tanken er at programmet tester en rekke tverrsnitt for så å sortere ut irrelevante resultater i ettertid. For å kontrollere at man sitter igjen med fornuftige resultater blir de lokale elementene for hver modell kontrollert etter NS-EN 1995. Mer detaljert fremgang for dette er forklart i avsnitt «4.3.3».

4.3.1 Modellbygging (Stivhetsmatrise)

Programmet importerer en CSV (*Comma-separated values*) fil med en liste over elementer som illustrert i Tabell 4 – *Første 7 kolonene i Elementer.csv*. Programmet

kjører deretter en funksjon som bytter ut parameterene gitt i Tabell 4 og Tabell 6 med forhåndsvalg *dybde* og *tverrsnitt* generert i modeller.py med *E_i, A_i, I_i, b_i, h_i*.

Etter oppdateringen av geometri og stivhetstall vil elementlisten gå gjennom en funksjon som konstruerer lokale stivhetsmatriser for alle elementene. Det er de samme matrisene som er beskrevet i kapittelet «3.1.1 - Statikk». Programmet bestemmer om den skal bruke fagverksfunksjonen eller bjelkefunksjonen basert på *Type* parameteren i Tabell 6. Om *Type* er lik 'Vannrett' vil det bli konstruert en lokal fagverksmatrise, resterende er konstruert som bjelkeelementer. De vannrette bjelkene er konstruert som fagverks-elementer for å slippe å 'Løsne' bjelkene i koblingen mellom andre elementer.

Tabell 4 – *Første 7 kolonene i Elementer.csv*

Element	start node	slutt node	Start Koordinat x	Start Koordinat y	Slutt Koordinat x	Slutt Koordinat y
1	1	5	0	0	1/4*d	4900
2	5	9	1/4*d	4900	1/2*d	8900
3	9	13	1/2*d	8900	3/4*d	13100
…n	13	17	3/4*d	13100	d	17100

Tabell 5 - *Kolonne 8-13 i Elementer.csv*

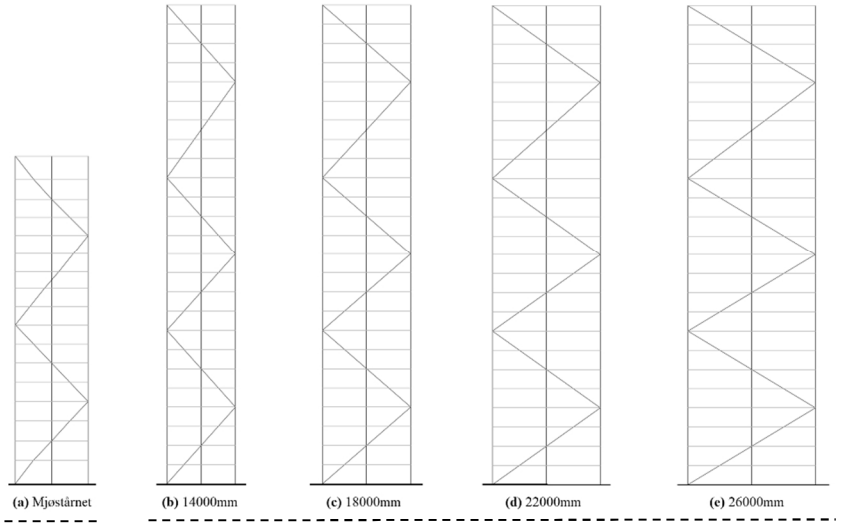
Element	…	Start frihet x	start frihet y	Start frihet rot	Slutt Frihet x	Slutt Frihet y	Slutt Frihet rot
1	…	0	1	2	12	13	14
2	…	12	13	14	24	25	26
3	…	24	25	26	36	37	38
…n	…	36	37	38	48	49	50

Tabell 6 – *Siste 6 kolonene i Elementer.csv*

Element	…	Type	E_i	A_i	I_i	b_i	h_i
1	…	Diagonal	E_1	A_1	I_1	b_1	h_1
2	…	Kant	E_2	A_2	I_2	b_2	h_2
3	…	midt	E_3	A_3	I_3	b_3	h_3
…n	…	Vannrett	E_4	A_4	I_4	b_4	h_4

Når de lokale stivhetsmatrisene er konstruert vil de bli satt inn i en global stivhetsmatrise basert på frihetsgradene gitt i Tabell 5.

Fordi elementene er representert i en liste som er relativt uoversiktlig er det lagd en funksjon som ploter alle elementene i en grafisk fremstilling. Figur *18* viser hvordan disse grafiske plotene ser ut for henholdsvis dybde lik 14000, 18000, 22000 og 26000 mm. Det er verdt å merke seg at diagonalene strekker seg over fem etasjer mellom 16 og 21 etasje. Det er tenkt at man strekker diagonalene over fem etasjer i toppen konstruksjonen da trykk og strekk-kreftene er vesentlig mindre mot toppen.



Figur 18 - *Geometri plotet med elementlisten som grunnlag.*

4.3.2 Massematrise og beregninger av dynamiske egenskaper

Massematrisen er konstruert med «*lumped mass*» som beskrevet i *«3.1.2*- Dynamikk» Det vil si at massen blir satt inn i nærmeste node. Massen som kommer fra hvert enkelt element blir fordelt med halve masse til henholdsvis start og slutt node for elementene. Dette resulterer i en massematrise med størrelse lik antall frihetsgrader, med massen liggende i diagonalen til massematrisen.

I avsnitt «4.3.3 - *Kraftmatrise og beregning av forskyvninger*» er egenvekt distribuert til kraftmatrisen i Newton, masse fra egenvekt og nyttelast er distribuert på samme måte bare i kg. Første funksjon er masse fra bæresystemet. Den er beregnet som vist i funksjonen under.

$$\frac{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} * Tv_{areal} * 390\ {}^{kg}/_{m^3}}{2} = masse\ til\ node$$

Masse fra dekke er beregnet og distribuert til noder ved å finne den totale massen per etasje og fordele en fjerdedel av den totale massen til kantnodene og halve masse til den midterste noden. Første funksjon under viser fordelingen til kant-nodene, andre funksjonen viser fordelingen til midt-noden.

$$\frac{b * d}{8} \times \begin{cases} 255\, kg/m^2\ \ for\ trä - 8\ dekke \\ 815\ kg/m^2\ \ for\ betong\ dekke \end{cases} = Masse\ til\ kantnode$$

$$\frac{b * d}{4} \times \begin{cases} 255\, kg/m^2\ \ for\ trä - 8\ dekke \\ 815\ kg/m^2\ \ for\ betong\ dekke \end{cases} = Masse\ til\ Midtnode$$

Grunne til at bredden ganger dybden blir delt på fire og åtte er på grunn av 2D gjenspeilingen som gjør at halve masse til bygget går til fagverket som beregnes i Python-programmet.

Siden de dynamiske egenskapene brukes til beregning av bruksgrensetilstand kan man konvertere noe av nyttelasten til masse. Her er det fulgt samme tankegang som beregningene til Mjøstårnet som resulterer i at 30% av nyttelasten blir omgjort til masse. Dette er på grunn av ligning 6.16b i NS-EN 1990 og Ψ_2 faktor for last i kategori B og C i tabell NA.A.1.1 i NS 1990. Fordelingen er gjort på samme måte som for masse fra dekke beregningen.

$$\frac{b * d}{8} \times\ 0{,}1\ \times \begin{cases} 305\, kg/m^2\ \ for\ Kategori\ B{:}\ kontor \\ 204\ kg/m^2\ \ for\ Kategori\ A{:}\ Bolig \end{cases} = Masse\ til\ kant\ node$$

$$\frac{b * d}{4} \times\ 0{,}1\ \times \begin{cases} 305\, kg/m^2\ \ for\ Kategori\ B{:}\ kontor \\ 204\ kg/m^2\ \ for\ Kategori\ A{:}\ Bolig \end{cases} = Masse\ til\ Midt\ node$$

Til slutt er det masse som kommer fra det innvendige bæresystemet. Det er gjort et anslag på masse per kvm ved å se på beregningene i Mjøstårnet. Det er sett på den totale massefordelingen per etasje og trukket fra all masse som kommer fra Dekke, Utvending bæresystem, fasade og nyttelast. Etter å ha gjort dette anslaget sitter man igjen med en

rest masse på 1200 N/m^2 eller 122 kg/m^2. Funksjonen for massefordelingen bli da som følger

$$\frac{b * d}{8} \times 122\ kg/m^2 = Masse\ til\ kantnode$$

$$\frac{b * d}{4} \times 122\ kg/m^2 = Masse\ til\ midtnode$$

Når alle massebidragene er satt inn i den diagonale massematrisen er de dynamiske egenskapene beregnet ved å løse egenproblemet vist under. dette egenproblemet er introdusert i kapittel «*3.1.2* - Dynamikk»

$$[k]\emptyset_n - \omega_n^2[m]\emptyset_n = 0$$

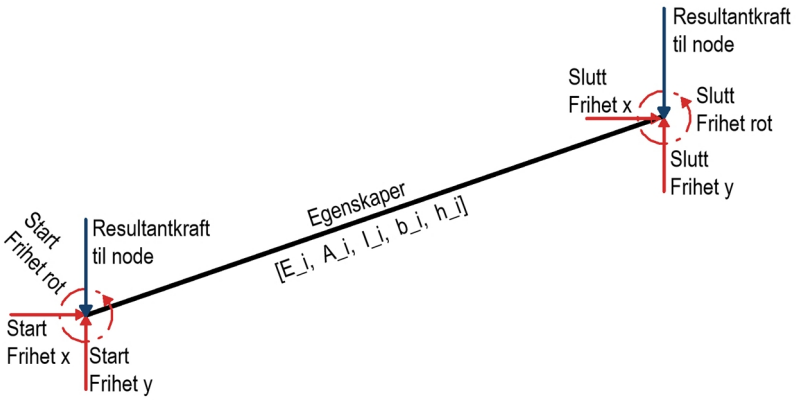
For å løse dette egenproblemet benyttes den innebygde Numpy funksjonen np.eig( ) (Community, 2018). Resultatet av denne funksjonen er $\Phi = [\emptyset_n] =$ er egensvingningsformene og $\Omega^2 = [\omega_n^2] =$ er de sirkulære frekvensene i andre. For å kunne beregne den vindindusert akselerasjon etter NS-EN 1991-1-4 tillegg C må man vite egenfrekvensen, egensvingeformen og modal massen. egensvingeformen og egenfrekvensen får man ut av np.eig() funksjonen. for å finne den ekvivalente massen per kvm er det benyttet fremgangsmåten definert i NS-EN-1991-1-4. Den er beregnet på følgende måte.

$$\frac{modal\ masse}{\sum b * \emptyset_n^2(etg) * \Delta etg} = ekv.masse\ i\ {}^{kg}/_{m^2}$$

Den vindinduserte akselerasjonen som virker på bygget er beregnet etter metode 2 i NS-EN 1991-1-4 tillegg C.

4.3.3 Kraftmatrise og beregning av forskyvninger

Etter at den globale stivhetsmatrisen er konstruert går programmet videre til å konstruere en kraft matrise. Der er det en rekke funksjoner som bruker variablene gitt i denne oppgaven som grunnlag og disse er presentert i teksten under. Figur 19 viser parameterne til de lokale elementene som brukes



Figur 19 - Grafisk fremstilling av de lokale elementene

Vindkraften blir beregnet etter NS-EN 1991-1-4:2005+NA:2009. For å begrense beregningsmengden er plasseringen av bygget begrenset til steder hvor referansevindhastighet $V_{b,0}$ = 22 m/s^2. I tillegg er det begrenset til Terrengkategori II (*Område med lav vegetasjon som gress og spredte hindringer(trær, bygninger) med avstand minst 20 ganger deres høyde*). For å beregne basisvindhastigheten er det antatt grunnverdier C_{dir} = 1, C_{seson} = 1 og C_{prob} = 1 noe som tilsier V_b = 22 m/s^2.

Vindkasthastighetstrykket blir beregnet etter punkt 4.5 (1) med forutsetningene nevn over. Referansehøydene er beregnet etter punkt 7.2.2 (1). bredde parameteren i Python-programmet er det utvendige målet som brukes til beregninger av vind parameterne. Dybde parameteren er Senter/Senter mål mellom kantbjelkene i bæresystemet. Det er derfor lagt på 2000mm på dette målet ved beregninger av vind påkjenninger.

Det utvendige vindtrykket er beregnet etter punkt 5.2 (1) med interpolering mellom C_{pe} faktorene hvor dette blir nødvendig. Vindkreftene på konstruksjonen er kalkulert etter 5.3 (3) som betyr at vindkreftene som angriper bygget blir beregnet fra formfaktorer. Det er hvert å nevne at den dynamiske analysen av bygget må være gjennomført før

vindberegningene kan utføres. Det er på grunn av C_sC_d faktoren som er avhengig av de dynamiske egenskapene til bygget. Forutsetningen for beregningen av C_sC_d faktorene etter tillegg B og C er vist i Tabell 7.

Tabell 7 - *Faktorer for beregning av C_sC_d*

Z_s	C_f	Z_0	L_t	Z_t	C_{prob}	demp	G_y	G_z	K_y	K_z
0.6*h	C_D+C_E	0.05	300	200	1	0.12	½	$^3/_8$	1	$^3/_2$

Fordelingen av krefter blir som følger.

$$A_{Ref}^{etg} = \left(\frac{høyde\ etasje\ under}{2} + \frac{høyde\ etasje\ over}{2}\right) \times \frac{bredde\ av\ bygget}{2}$$

$$W_e^{venstre} = q_p(z_e) * c_{pe,D}$$

$$W_e^{høyre} = q_p(z_e) * c_{pe,E}$$

$$Venstre\ node = F_w = C_sC_d * A_{Ref}^{etg} * W_e^{venstre}$$

$$Høyre\ node = F_w = C_sC_d * A_{Ref}^{etg} * W_e^{venstre}$$

Kraften fra egenvekt er delt inn i egenvekt fra bæresystem, egenvekt fra fasade og egenvekt fra dekke. Kraft fra bæresystemet er beregnet ved å finne lengden til elementene basert på x og y koordinatene i Elementer.csv samt arealet av tverrsnittet som ligger i samme liste. Beregningen er vist under

$$\frac{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} * Tv_{areal} * 390\ {}^{kg}/_{m^3} * 9{,}81\ {}^{m}/_{s^2}}{2} = Resultantkraft\ til\ node$$

Kraft fra egenvekt dekke er beregnet med hensyn til massefordelingen på etasjen samt lastarealet som er beregnet med hensyn til lengden av de vannrette bjelkene. Funksjonene for dette er som følger

$$\frac{(x_1 - x_0)\ \times 3{,}85\ m}{2} \times \begin{cases} 2500\ N/m^2\ for\ trä - 8\ dekke \\ 8000\ N/m^2\ for\ betong\ dekke \end{cases} = Resultantkraft\ til\ node$$

Kraft fra fasade er beregnet ved å ta lengden av de vannrette bjelkene og gange de opp med høyden av etasjen ganger kraften fra fasade som er gitt i N/m^2. fordelingen av

resultant kraften er gitt ved å ta halve resultantkraften til start og slutt node. Beregningen av denne funksjonen er gitt under

$$\frac{(x_1 - x_0) * etasjehøyde * fasadekraft\ i\ N/m^2}{2} = Resultanttkraft\ til\ node$$

Når alle kraftbidragene er lagt til kraftmatrisen blir deformasjonene beregnet etter samme fremgangsmåte som beskrevet i kapittel «*3.1.1 - Statikk*». Sitter da igjen med deformasjonene i alle nodene til konstruksjonen. Deformasjonen i den øverste noden i venstre side av konstruksjonen blir lagt til resultatliste.

4.3.4 Elementkontroller

Python programmet som er konstruert for denne oppgaven bygger modeller helt slavisk etter parameterene den bli tildelt. Det er derfor nødvendig å teste kapasiteten til de lokale elementene da det ikke er gitt at alle resultatene tilfredsstiller bruddgrensekravene etter NS-EN 1995.

Måten dette er gjort på er å ta ut de lokale kreftene kalkulert etter avsnitt «*4.3.3 Kraftmatrise og beregning av forskyvninger*» og sortert elementene etter «type» parameteren gitt i Elementer.csv. Basert på dette er elementene sortert etter følgende grupper. *Diagonal under trykk, Diagonaler under strekk, kantbjelker under trykk, Kantbjelker under strekk og vannrettebjelker under bøying.* Elementkontrollene for de forskjellige elementene er illustrert i Tabell 8.

Tabell 8 - *Elementsjekk etter NS-EN 1995*

Type	Kontroll etter NS-EN 1995				
	6.1.6	6.2.3	6.2.4	6.3.2	6.3.3
Diagonal under trykk			x	x	x
Diagonal under strekk		x			
Kantbjelke under trykk			x	x	x
Kantbjelke under strekk		x			
Midtbjelke under trykk			x	x	
Vannrette bjelker under bøying	x				

Etter at elementkontrollene er gjort etter NS-EN 1995 blir den største utnyttelsen for hver type sortert ut og lagret i Resultatlisten.

Maks tillat forskyvning er i denne teksten er satt til $h/400$. Den mest tradisjonelle verdien er $h/500$, men det ble bestemt i denne oppgaven at man skulle se på muligheten til å redusere forskyvningskravet. Maks tillat forskyvning er en veiledning og er på ingen måte ett bestemt krav.

4.3.5 Modeller

Resultatene for alle modellene blir lagret i en CSV fil hvor hver rad representerer en modell. De viktigste beregningene som brukes i analyse delen er Forskyvning, Forskyvningskrav, Utnyttelse av forskyvningene, Egenfrekvens, Vindindusert akselerasjon, Akselerasjonskrav for kontorbygg og Utnyttelse av akselerasjonen med hensyn til akselerasjonskravet for kontor bygg. Et utdrag av resultatene er vist i tabellene under.

Tabell 9 – *Kolonne 3-8 i Resultat_total_hash.csv*

Resultat	…	Høyde	Dybde	Bredde	Antall Trä-8	Antall Betong	Tverrsnitt bredde
1	…	100 000	14 000	14 000	24	1	625
2	…	100 000	14 000	15 000	24	1	625
3	…	100 000	14 000	16 000	24	1	625
…n	…	100 000	14 000	17 000	24	1	625

Tabell 10 - *Kolonne 29-32 og 34 i Resultat_total_hash.csv*

Resultat	…	Egenfrekvens	Periode	Akselerasjon	Akselerasjons krav kontor	Akselerasjons-Utnyttelse kontor
1	…	0.42	2.36	0.112	0.087	1.28
2	…	0.41	2.43	0.114	0.089	1.28
3	…	0.40	2.49	0.116	0.090	1.29
…n	…	0.39	2.56	0.118	0.091	1.29

5 Resultater og analyse

Arbeidsgangen i dette kapittelet går ut på å presentere resultatene som ligger i resultatlisten Resultat_total_hash.csv fra kapittel «4.3.5 - Modeller» samt presentere avvikene på grunn av forenklingene gjort i denne oppgave.

For å få ett innblikk i datasettet som inneholder alle modellene er det i kapittel «*5.2- Maks grunnakselerasjon og forskyvning.*» gjort en overordnet analyse slik at leser kan se noen overordnede sammenhenger i datasettet.

I den mer detaljerte analysen er det foretatt en PLS analyse for å finne korrelasjoner i datasettet, ved å foreta en slik PLS analyse kan man se nærmere på de korrelasjonene som kan være av interesse. Som nevn i kapittel *«3.3.1 - Datasett med to eller flere dimensjoner.»* Kan det være vanskelig å illustrere dataen i ett to-dimensjonalt plot. Variablene som er satt i denne oppgaven resulterer i flere dimensjoner enn det man kan vise på ett 2-dimensjonalt plot, men det kan fortsatt være mulig å plotte deler av resultatene i 2d om de underliggende sammenhengende ikke er avhengig av mer enn 2 variabler. Selv når dataen er avhengig av mer en 2 variabler kan en PLS analyse vise om den tredje variablene har stor eller liten innvirkning, om det viser seg at den tredje avhengige variabelen ikke har stor innvirkning kan det være mulig å forkaste den uten for stort tap av informasjon.

Den detaljerte delen av resultater og analyse er delt inn i tre deler som skal besvare del-problemstillingen *«Hvordan endres Egenfrekvens, Akselerasjon og Akselerasjons-utnyttelsen ved å endre fotavtrykk, massefordeling og tverrsnitt på en gitt modell?»*

5.1 Kontroll av forenklingene i oppgaven.

5.1.1 Kontroll 1

I kapittel «*4.2.1 - Kontroll 1*» ble det forklart hvordan man skulle teste forenklingene med å gjenspeile en fullstendig 3D-RSA modell til et 2D system hvor massen til konstruksjonen ble satt inn som «*Lumped mass*» i noder på fagverket som skulle gjenspeile avstivningen til den fullstendige 3D modellen.

I Tabell 12 - *Dynamiske resultater 3D / 2D robot* fra «*Vedlegg A – Verifisering av arbeidsgang*» ser man at det ikke er noe problem i å gjenspeile 3D modellen til ett 2 dimensjonalt fagverk. Egenfrekvensen i 3D modellen uten fjærstivhet i opplagerene var på 0.405 Hz, egenfrekvensen til 2D modellen er på 0.403 Hz. Den relative massen var 77.92% for 3D modellen, 2D modellen hadde en relativ masse på 77.81%.

i Tabell 14 fra *«Vedlegg A – Verifisering av arbeidsgang»* er egensvingningsformen til modellene presentert. Avviket som forekommer i Tabell 14 er for formålet godt nok. Fra kapittel «*3.2 -Vind*» er det allerede kjent hvordan den vindinduserte akselerasjonen kan ha et avvik opp mot 5% fra den teoretiske verdien beregnet etter numeriske metoder. Så lenge forenklingene ikke resulterer i noe drastisk endring i egensvingningsform er det vanskelig å se for seg noe store endringer av den vindinduserte akselerasjonen. Fra sammenligningen av forskyvningen for de to modellen viser det seg at modellene oppfører seg veldig likt. I Tabell 13 fra vedlegg A kan man se differansen mellom forskyvningen i toppetasjene. Differansen mellom modellene er på 2.43mm. Det er heller ingen indikasjon på at modellene har forskjellige bøyeformer.

5.1.2 Kontroll 2

kontroll 2 som tester 2D-RSA modell mot 2D-Python modell viser ikke noe større differanse en det kontroll 1 gjorde. Tabell 16 viser at frekvensen for 2D RSA modellen uten pergola var på 0.408 Hz mot 0.405 Hz for Python-programmet. I Tabell 17 ser man hvor liten differansen mellom forskyvningen til 2D-RSA modellen er mot Python-modellen. En differanse på 0.75mm i forskyvning i toppetasjen kan sees på som minimale med tanke på formålet til oppgaven. I Tabell 18 kan man se hvordan egen svingningsformen til de to modellene differensierer seg. På samme måte som for

sammenligningen av 3D-RSA/2D-RSA er det ingen tegn til forandring av oppførsel eller bøyeform.

5.1.3 Kontroll 3

Kontroll 3 var satt opp for å teste parametriseringen til Python-programmet. Ett premiss for at man i det heletatt kan sammenligne parametriseringen til python-programmet mot 3D-RSA modellen må det være gitt at de statiske og dynamiske egenskapene viser samme resultater, fra Kontroll 1 og 2 vet man at dette stemmer.

I Tabell 19 kan man se at frekvensen til 3D-RSA modellen uten pergola er på 0.411 Hz, egenfrekvensen for Python-programmet er på 0.414 Hz. avviket mellom disse modellene viser ikke noe større avvik en det de andre kontrollene har vist.

Forskyvningen i toppetasjen er 129.20 mm for 3D-RSA modellen uten pergola, Python-programmet kom frem til en forskyvning på 133.78 mm. Ser ett minimalt avvik mellom de to modellene. Det er selv opp til leser om avvikene presentert i dette kapittelet er for store, men forfatter konkluderer med at Python programmet klarer å fordele masse og legge til vindkrefter veldig presist. Samt at videre beregning med Python programmet representerer en god tilnærming til detaljerte beregninger.

5.2 Maks grunnakselerasjon og forskyvning.

Før man kan foreta en detaljert analyse av de dynamiske egenskapene er det nødvendig å se på de dynamiske brukskravene og utnyttelsen av disse for de forskjellige modellene. For ikke å dra irrelevante konklusjoner er det viktig at senarioene som blir analysert senere i dette kapittelet inneholder faktiske løsninger, med dette mener man konstruksjoner som har en utnyttelse under maks nivå for henholdsvis vindindusert akselerasjon og forskyvning. Basert på hvordan Python-programmet beregnet de forskjellige modellene kan det tenkes at det eksisterer en rekke løsninger hvor de dynamiske bruksravene med hensyn til vindindusert akselerasjon er godt innenfor kravene gitt i ISO-10137, men de samme modellene har en altfor stor utnyttelse av forskyvning. Dette gir mening da høy global forskyvning implisitt resulterer i en mye ‘mykere ’konstruksjon en det som er praktisk mulig å bygge. Som det kommer frem lenger ned i analysen finnes det modeller der den vindinduserte akselerasjonen er godt

nedenfor 50% utnyttet, der samme modeller har en utnyttelse opp mot 160% i topp forskyvning. Disse konstruksjonene er med på å forstyrre analysen for frekvens og vindindusert akselerasjon da man får urealistiske ekstremalpunkter i datasettet som ikke er av noe relevans.

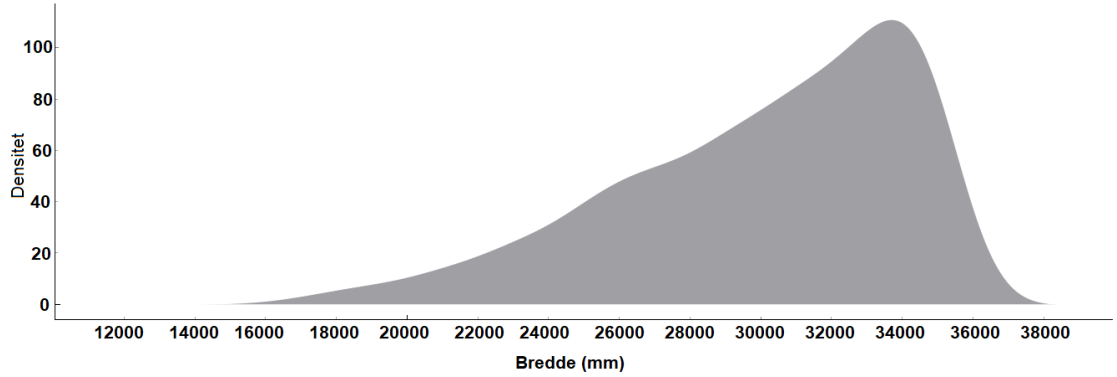
For å få en overordnet oversikt på hvilke brukskrav som er gjeldende er det gjort noen enkle spørringer i datasettet. Totalt er det 1 520 640 modeller i datasettet. Disse modellene representerer alle mulige sammensetninger av variablene nevnt på side 21. De vannrette tverrsnittene har minimal effekt på det globale systemet, så disse tverrsnittene er satt til en konstant høyde lik 450mm. Når tverrsnittet til de vannrette bjelkene er satt til en konstant verdi lik 450mm sitter man igjen med 380 160 modeller. dette er fordi det på side 22 ble bestemt at disse bjelkene skulle ha 4 forskjellige tverrsnitt. Fjerner man 3 stk tverrsnitt fra itereringen vil antall modeller reduseres ned til ¼

Sorterer man bort alle modeller som inneholder beregninger der utnyttelsen av akselerasjonskravet for kontorbygg overstiger 1.00 er det igjen 373833 modeller. Dette betyr at det finnes 380 160 – 373 833 = 6327 modeller der utnyttelsen av akselerasjons-kravet for kontorbygg overstiger 1.00.

Gjør man en spørring ved å filtrere bort alle modeller der utnyttelsen av forskyvningen ikke overstiger 1.00 sitter man igjen med 330 462 modeller. Dette betyr at det finnes flere modeller hvor forskyvningen ikke er innenfor kravet enn det det er modeller som ikke overholder akselerasjonskravet for kontorbygg.

Gjør man en spørring på alle modeller der utnyttelsen av forskyvningen og akselerasjonen for kontorbygg er under 1 sitter man igjen med 330462 modeller. Dette er samme antall modeller som man fikk ved å se på modeller der kun forskyvningskravet var under 1.00. Dette betyr at alle modeller som har en utnyttelse under 1 for akselerasjonskravet for kontorbygg også har en utnyttelse av forskyvningen under 1.

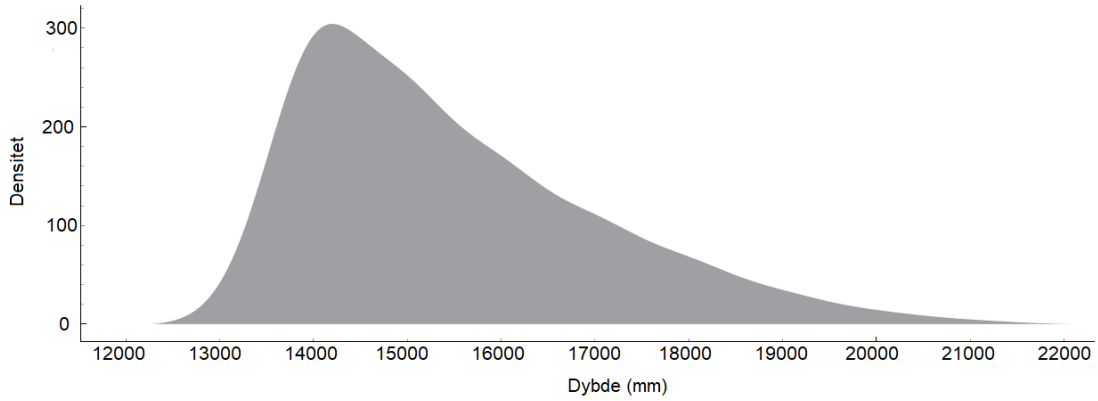
Figur 20 illustrerer bredden til de modellene som ikke overholder forskyvningskravet. Akse-x representerer bredden til disse modellene. Akse-y er densiteten til modellene, ett annet ord for dette kunne vært antall modeller i gitt x-akse.



Figur 20 – Antall modeller som ikke overholder forskyvningskravet, summert etter bredde

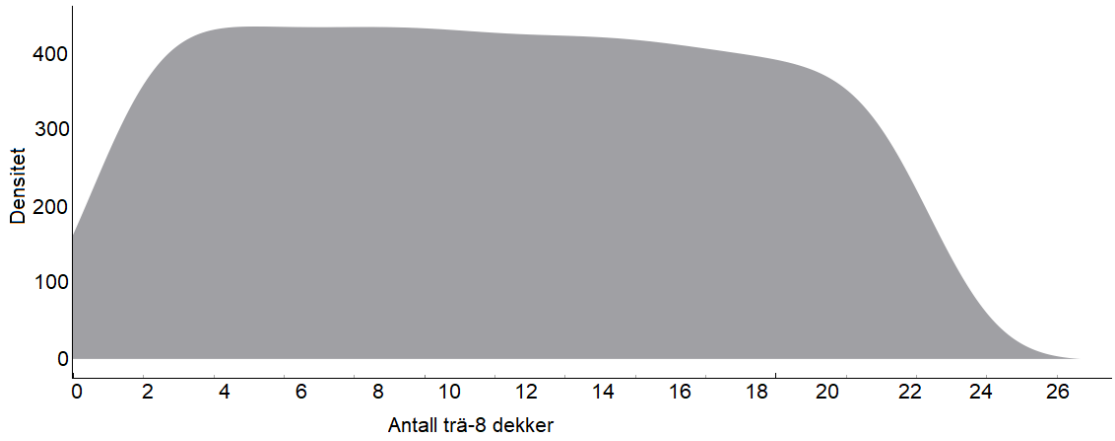
Ut fra Figur 20 kan man se at de fleste modellene som ikke overholder forskyvningskravet er i området rundt 25000-35000 mm fordi det er størst densitet i det området. Denne figuren er kun ment som en illustrasjon for å vise hvilke verdier av bredde som bidrar til ugunstige resultater.

Figur 21 er plotet på samme måte som Figur 20. Akse-x presenterer dybde, y-akse representerer antallet modeller som ligger i området med gitt dybde i akse x.



Figur 21 - Antall modeller som ikke overholder forskyvningskravet, summert etter dybde

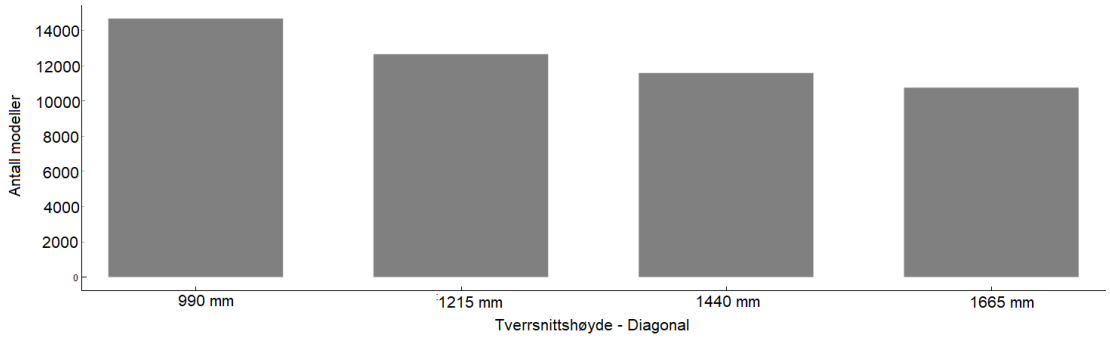
ut fra Figur 21 kan man se at de flere modellene som ikke overholder forskyvningskravet ligger i området med en dybde mellom 14000-18000mm.



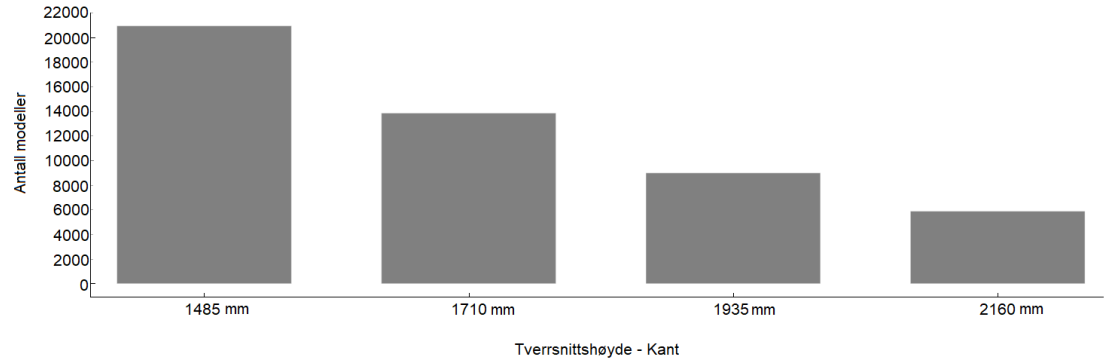
Figur 22 – Antall modeller som ikke overholder forskyvningskravet, summert etter antall trä-8

Figur 22 viser hvor mange modeller som ikke overholder forskyvningskravet med hensyn til antall trä-8 dekker i modellene. x-akse representerer antall trä-8 dekker, Y-akse representerer antall beregninger. Ser en liten tendens til at det er flere modeller med lavt antall trä-8 dekker enn der hvor modellene har et lavt antall trä-8 dekker med tanke på for høy utnyttelse av forskyvningene.

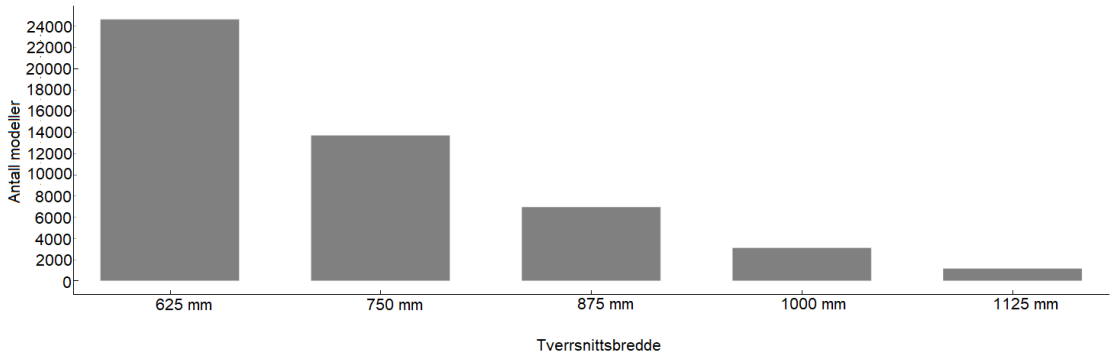
Figur 23, Figur 24 og Figur 25 er plotet på samme måte som figurene over, bare med en forandring. Istedenfor densitet er det målt frekvens (Antall modeller), dette kan være misvisende med tanke på temaet i denne oppgaven men *frekvensen* i dette tilfelle er antall beregninger. x-aksen representerer en tverrsnittshøyde, y-akse representerer antall modeller som ikke overholder forskyvningskravet. Søylene er derfor en representasjon av antall modeller med gitt tverrsnitt = x som ikke overholder forskyvningskravene. Ser at det er lite varians mellom søylene i Figur *23* sammenlignet med de to andre figurene. Ser man for eksempel på Figur *25* ser man at det inneholder veldig få modeller som overstiger forskyvningskravet. Det er faktisk bare rundt 1000 modeller i hele datasettet som ikke overholder forskyvningskravet når variabelen *tverrsnittsbredde* er 1125 mm. Ser man på samme figur og ser på søylen der variabelen *tverrsnittsbredde* er 725 mm ser man at det er 24 000 modeller med *tverrsnittsbredde* på 725 mm som ikke overholder forskyvningskravet. Fordi det er så stor variasjon mellom de to nevnte søylene er det en antydning til at variabelen *Tverrsnittsbredde* har en stor innflytelse på forskyvnings-kravet.



Figur 23 – Antall modeller som ikke overholder forskyvningskravet, summert etter diagonalbjelkene



Figur 24 - Antall modeller som ikke overholder forskyvningskravet, summert etter kantbjelkene



Figur 25 – Antall modeller som ikke overholder forskyvningskravet, summert etter tverrsnittsbredde

Selv om dette var en introduksjon av datasettet er det greit å ta med seg det faktumet at modeller med en dybde på 14000 mm med høy bredde har en veldig liten sannsynlighet for å tilfredsstille forskyvningskravene da det er mange modeller med lav dybde og høy bredde som ikke overholder forskyvningskravet. Kan også være greit å merke seg at det er mange modeller med tverrsnittsbredde under 725mm, som ikke overholder forskyvningskravet.

Basert på informasjonen over er det introdusert en ny «*gruppe*» variabel. Fordi det er så mange variabler i datasettet er det definert enn sammensetning av alle tverrsnittene som er *Diagonalbjelke* = 990mm, *Kantbjelke* = 1485 og *Tverrsnittsbredde* = 875. Denne sammensetningen av tverrsnitt vil bli kalt Standardtverrsnitt og tverrsnitts variablene kan være antatt å ha disse verdiene om ikke annet er definert.

5.3 Frekvens

Basert på teorikapittelet vet man at den generelle formelen for egenfrekvens er som følger

$$f_n = \sqrt{\frac{S}{m}} \tag{17}$$

Hvor

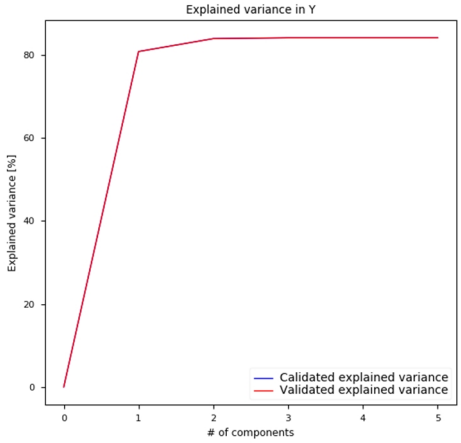
S = Stivhet
m = Masse

Ut fra formelen over ser forholdet til egenfrekvens ut til å være en enkel ligning. Men man vet fra teorikapittelet at egenfrekvensen er ett n-dimensjonalt egenproblem hvor det finnes en egenfrekvens for alle frihetsgradene i systemet. Det er verdt å merke seg at man kun er interessert i den fundamentale egenfrekvensen til bygget. Intuitivt kan man til en viss grad tolke hvordan oppgavens gitte variabler vil endre egenfrekvensen til bygget. Stivheten vil være styrt av tverrsnittene og dybde. Massen vil være styrt av bredde og antall trä-8 dekker samt dybde. Uten å se noe på resultatene fra Python beregningene klarer man å konkludere med at økning i tverrsnitt vil øke frekvensen, økning i bredde vil minske frekvensen samt at en økning i antall trâ-8 dekker vil minke frekvensen igjen. Dybde vil gi innvirkning på både Stivhet og masse.

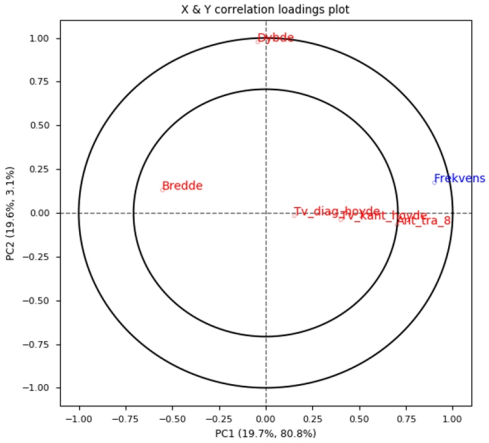
5.3.1 PLS analyse av frekvensen

For å se hvor stort bidrag de forskjellige variablene har på systemet er det gjort en PLS1 analyse. PLS analysene er gjennomført i Python med modulen «Hoggorm» (Tomic, 2017). Oppsettet av analysen finnes i vedlegg B, *«PLS1 analyse med Y = Frekvens»* og *«PLS1 analyse med Y = Frekvens og Xi = log(Xi)»*. PLS1 baserer seg på ett lineært system, for å minimere de ikke-lineære effekten ble det gjort ett forsøk med å normalisere X_i variablene logaritmisk. Den forklarte variansen av Y var henholdsvis $Log(X_i)$ = 68.21% og X_i = 83.9% om man ser på de to første prinsipalkomponentene. Ut fra dette ble forsøket med $log(X_i)$ forkastet da det forklarte mindre en forsøket med X_i.

Figur 26 Viser hvor mye av variansen til egenfrekvensen som er forklart med de forskjellige prinsipalkomponentene. Ut fra figuren ser man at prinsipalkomponent 1 forklarer 80.8% av variansen til frekvensen, prinsipalkomponent 2 forklarer 3.1% av variansen til frekvensen. Prinsipalkomponent 3 til 5 bidrar ikke til noe mer forklaring av variansen til Y. Siden det kun er to prinsipalkomponenter som er med på å definere frekvensen kan men se nærmere på korrelasjonen mellom variablene med ett korrelasjonslast plot. Figur 27 viser korrelasjonslast plotet til PLS1 analysen, x-aksen presenterer last til prinsipalkomponent 1, y-akse presenterer last til Prinsipalkomponent 2. Videre vil prinsipalkomponent bli benevnte med forkortelsen PC fra det engelske ordet *Principal Component*.



Figur 26 - Forklart varians av frekvens med X_i (PLS1)



Figur 27 - Korrelasjonslast plot for frekvens (PLS1)

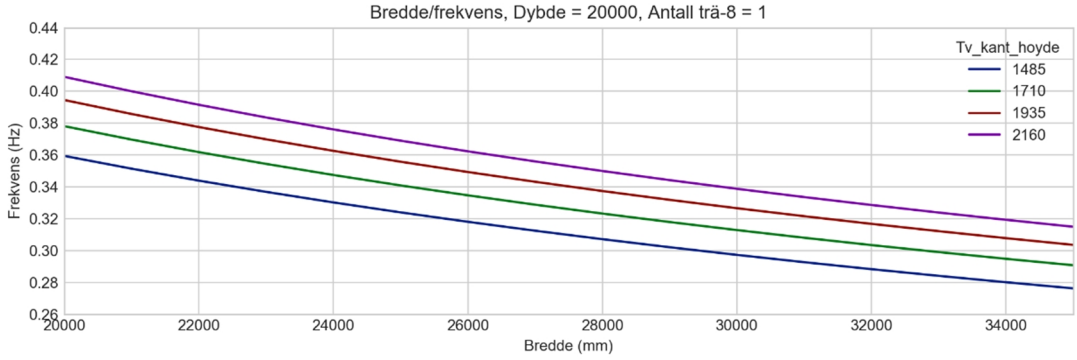
Ser man nærmere på PC1 kan man se at PC1 er høyt korrelert med frekvens, dette tyder på at PC1 beskriver det meste av frekvens. Videre i PC1 kan man se at Bredde er negativt korrelert, tverrsnittene *Diagonal* og *kant* samt *antall trä-8* er positivt korrelert i PC1. Dette betyr at frekvens og bredde er negativt korrelert, Frekvens og Tverrsnittene *Diagonal* og *Kant* samt *antall trä-8* er positivt korrelert. Korrelasjonen mellom bredde og de andre X-variablene kan tyde på at høyere tverrsnitt og økt antall trä-8 har en interaksjonseffekt på bredde og vice versa.

Det er ikke noe last på dybde i PC1. Siden PC1 presenterer det meste av varians mellom frekvens tyder det på at dybde har veldig lite å si for frekvensen. Går man videre til PC2 kan man se at PC2 har veldig høy last på Dybde, i tillegg er det en liten positiv

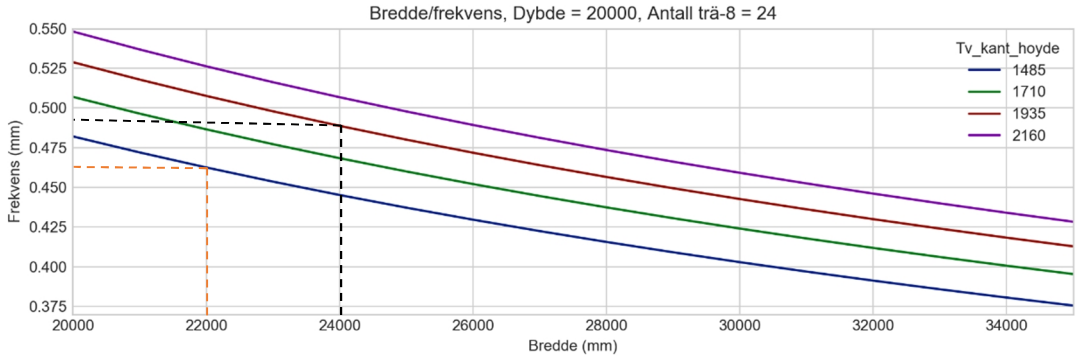
korrelasjon mellom PC2 og frekvens samt Bredde, men igjen så beskriver PC2 kun 3.1% av variansen til frekvens, noe som tyder på at det har liten innvirkning på den 'globale' variansen til frekvens.

5.3.2 Forholdet mellom Bredde, Tverrsnitt og Antall trä-8

I PLS analysen var det en sterk indikasjon på at det er mulig å gjenspeile dataen i 2-dimensjonale plot siden de underliggende sammenhengende er 2-dimensjonale. En gjenganger i kapittel «5.3 - Frekvens», «5.4 – Vindindusert akselerasjon» og «5.5 - Akselerasjonsutnyttelse» vil være at man ser to 2-dimensjonale plot der en variabel, for eksempel *Tverrsnittsbredde* er sortert som grupper og plotet som grafer i det 2-dimensjonale plotet.



Figur 28 - Bredde/frekvens, Dybde = 20000, Antall trä-8 = 1, Tverrsnitt = Kant



Figur 29 – Bredde/frekvens, Dybde = 20000, Antall trä-8 = 24, Tverrsnitt = Kant

Figur 28 og Figur 29 er to identiske plot med den forskjellen at Figur 28 representerer alle modeller hvor det er 1 trä-8 dekke, Figur 29 er da plotet med modeller hvor det er

24 trä-8 dekker i modellene. Det er satt en konstant verdi på *Dybde* variabelen lik = 20000 mm. Bredden til gitt modell er definert i x-aksen, frekvensen til en gitt modell er i y-akse. Orange stiplet linje i Figur 29 viser frekvensen til en spesifikk modell med Bredde = 22000 mm, Dybde = 20000 mm, Antall trä-8 dekker = 24, Tverrsnitt = Standard utenom høyden til kantbjelkene som er 1485mm. Den svarte stripete linjen i Figur 29 viser frekvensen til en modell som har Bredde = 24000mm, Dybde = 20000mm, Antall trä-8 dekker = 24, Tverrsnitt = Standard utenom høyden til kantbjelkene som er 1935mm. Den blå grafen i Figur 29 representerer alle modeller med Dybde = 20000 mm, Antall trä-8 dekker = 24, Kantbjelke = 1485 mm og varierende bredde fra 20000 mm til 35000 mm. Denne metoden for å plote sammenhengen mellom variablene er gjennomgående for den resterende teksten.

Informasjonen som kan hentes ut fra de to figurene over er hvordan bredden, Høyden på kantbjelkene og antall trä-8 dekker gir innvirkning på frekvensen når dybden er satt til 20000mm. I tillegg kan man sjekke interaksjonene mellom «*gruppene*» i dette tilfelle høyden til kantbjelkene. Husk tilbake til kapittel «3.3.1 - Datasett med to eller flere dimensjoner.» om *full factorial design* hvordan man kunne illustrere interaksjoner ved å se om to grafer ligger parallelt med hverandre eller ikke. Om to grafer ikke ligger parallelt er det det en indikasjon på interaksjon mellom variabelen i x-akse og variabelen som definerer «*gruppen*» i ett gitt plot.

Figur 28 og Figur 29 er plotet slik at differansen mellom topp og bunn frekvens på y-aksen er lik for begge grafene. Ved å studere figurene ser man at det er en stor endring av oppførsel av bredde/frekvens med forskjellige antall trä-8. man ser for eksempel at å endre bredden fra 20000 mm til 35000 mm med en høyde på kantbjelkene lik 1485mm vil gi en større endring om man har 24 trä-8 dekker i forhold til 1 trä-8 dekke.

$$f\left(bredde, dybde, ant\_tra\_8, tverrsnitt_{kantbjelke}\right) = frekvens$$

$$\Delta frekvens = f(20000, 20000, 24, 1485) - f(35000, 20000, 24, 1485)$$

$$\Delta frekvens = 0.423\ Hz - 0.0.320\ Hz = 0.103\ Hz$$

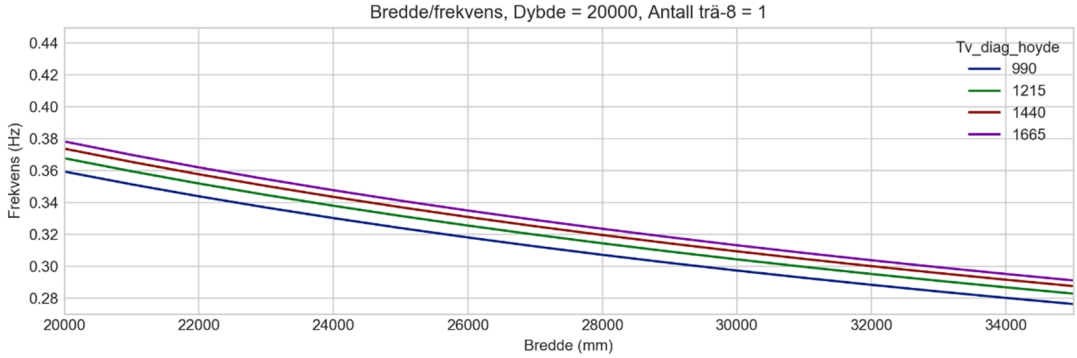
Ser man på endringen av frekvens med hensyn til bredde med 1 trä-8 dekke får man følgende endring.

$$f\left(bredde, dybde, ant\_tra\_8, tverrsnitt_{kantbjelke}\right) = frekvens$$

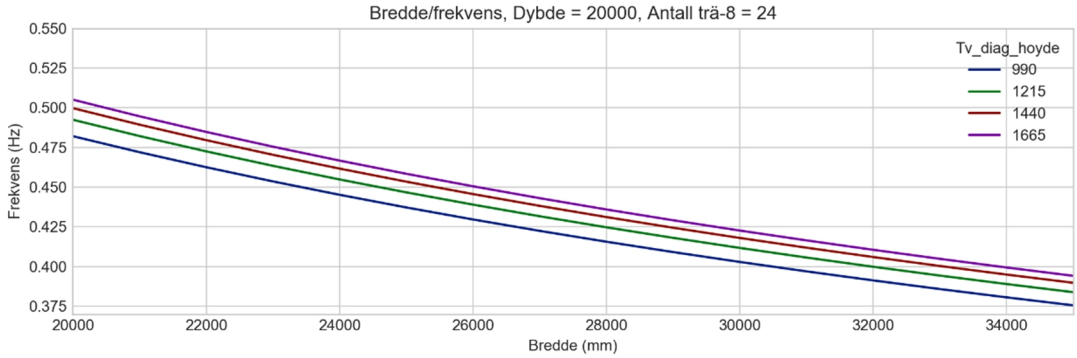
$$\Delta frekvens = f(20000, 20000, 1, 1485) - f(35000, 20000, 1, 1485)$$

$$\Delta frekvens = 0.306\ Hz - 0.234\ Hz = 0.072\ Hz$$

Her ser man at antall trä-8 har en innvirkning på forholdet mellom bredde og frekvens og tolkningen av PLS analysen ser ut til å stemme.



Figur 30 - Bredde/frekvens, Dybde = 20000, Antall trä-8 = 1, Tverrsnitt = Diagonal

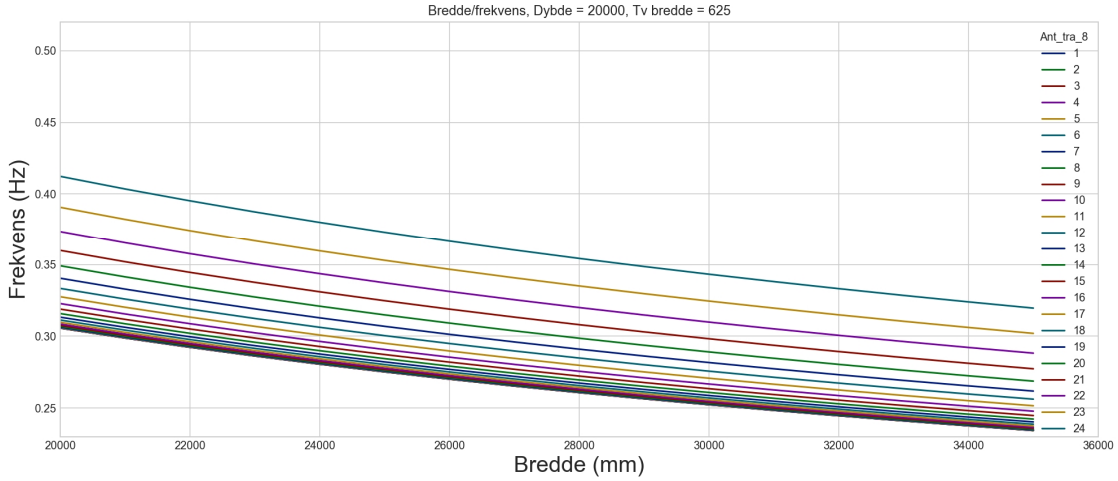


Figur 31 - Bredde/frekvens, Dybde = 20000, Antall trä-8 = 24, Tverrsnitt = Diagonal

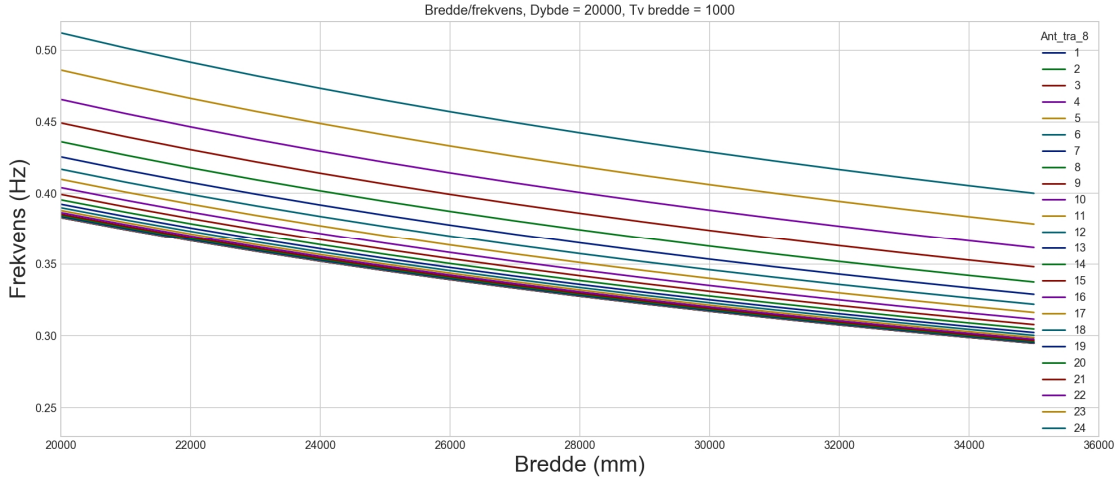
Figur 30 og Figur 31 er differansen mellom frekvensen i y-akse den samme. Ser man nærmere på variansen i disse grafene ser man den samme tendensen som er i Figur 28 og Figur 29. Ser også her at antall tra-8 er med på å endre breddens evne til å endre frekvens, mer spesifikt så øker breddens evne til å endre frekvens med økende antall trä-8 dekker. Den er faktisk tilnærmet lik da nederste graf i Figur 29 er den samme grafen som den nederste grafen i Figur 31. Det er en fin illustrasjon som viser hvordan en endring av Kantbjelkene har mye større evne til å endre på frekvensen en det en økning av høyden til diagonalbjelkene.

5.3.3 Forholdet mellom Bredde, Tverrsnitt og antall trä-8 dekker når dybde er konstant

I PLS analysen var det tegn til at bredden til modellene var negativt korrelert med både trä-8 og tverrsnittsøkning. I Figur 32 og Figur 33 er antall trä-8 dekker sortert inn i grupper og plotet som grafer i de to figurene. Øverste graf i begge figurene representerer modeller med 24 trä-8 dekker. Nærmeste graf for den med 24 trä-8 dekker er modellene med 23 trä-8 dekker osv.



Figur 32 - Bredde/frekvens, Dybde = 20000, Tverrsnittsbredde = 625



Figur 33 - Bredde/frekvens, Dybde = 20000, Tverrsnittsbredde = 1000

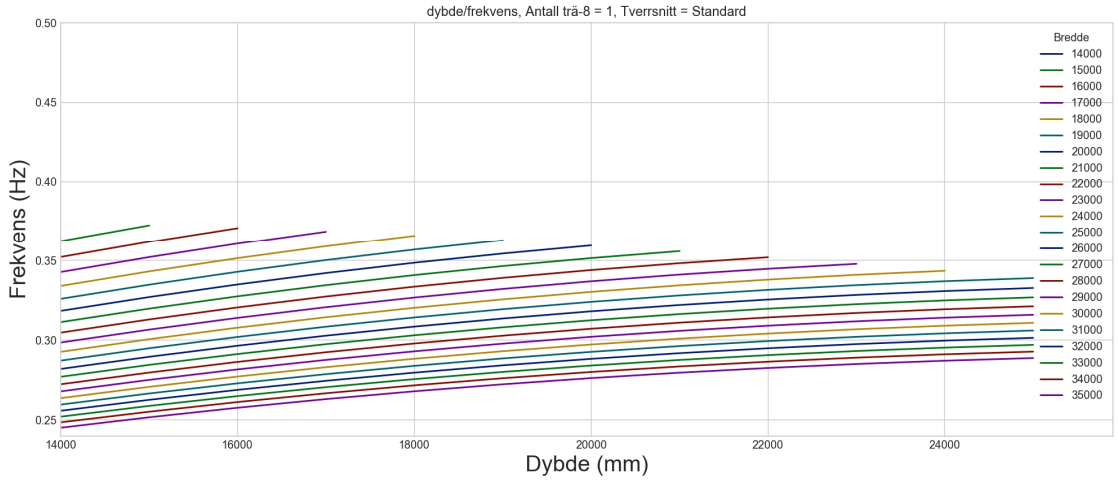
Man kan ved å se på avstanden mellom grafene se hvor stor innvirkning det å endre antall trä-8 dekker når det allerede er ett høyt antall trä-8 dekker i modellene. Ser man på de nederste grafene kan man se hvordan de er klynget sammen. Dette viser hvor liten

innvirkning det gir å endre antall trä-8 dekker når det allerede er ett lite antall trä-8 dekker i modellene.

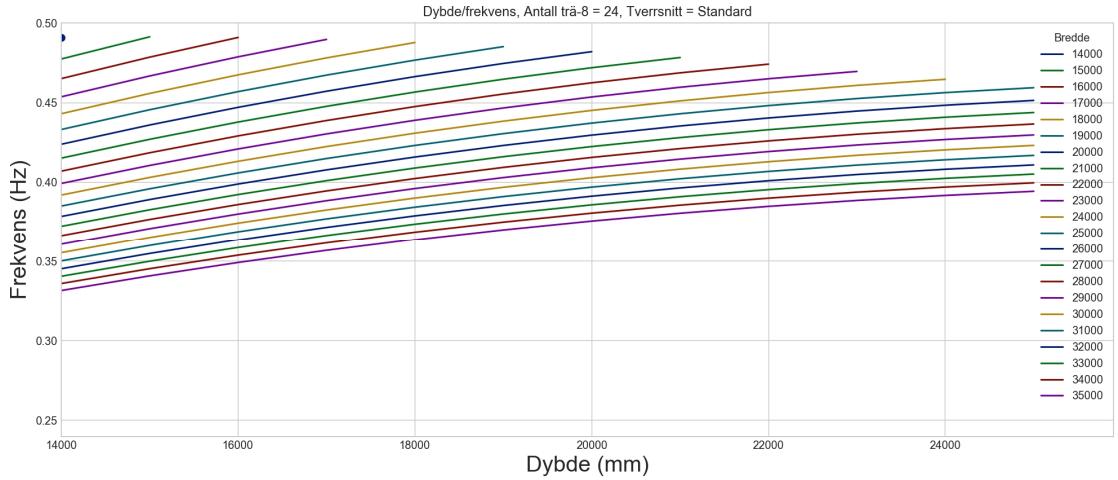
Ser også at endring av tverrsnittbredde endrer forholdet mellom bredde/frekvens. Det er ikke bare det at frekvensen øker med økt tverrsnittsbredde, men også det at en endring av bredden gir mer effekt når man har høy tverrsnittsbredde sammenlignet med modeller hvor det er en liten tverrsnittsbredde.

5.3.4 Forholdet mellom Dybde, Antall trä-8 og bredde når tverrsnitt er konstant

Videre til endring av dybde. Fra PLS analysen var det antydninger til at dybden forklarte veldig lite av variansen til frekvens, men igjen så var det 16,1% av variansen til frekvens som ikke var registrert i PLS analysen.



Figur 34 - Dybde/frekvens, Antall trä-8 = 1, Tverrsnitt = Standard



Figur 35 - Dybde/frekvens, Antall trä-8 = 24, Tverrsnitt = Standard

Basert på tolkningen av PLS analysen skulle egentlig grafene i Figur 34 og Figur 35 vært en tilnærmet flat linje siden det egentlig ikke skulle vært noe endring med forskjellige dybder, men det er ikke tilfelle. På Figur 34 ser man veldig klart at det ikke er tilfelle. Ved lav dybde (mellom 14000 mm og 20000 mm) vil en økning av dybde gi større utslag på frekvensen enn det en reduksjon av bredden vil gi. Med høyere dybde vil denne effekten avta, med ved dybde over 20000 mm kan man begynne å diskutere om en enhets-reduksjon av bedde gir høyere utslag enn enhetsendring av dybde. Plukker man ut modellene som inneholder følgende variabelverdier, Tverrsnitt = Standard og antall trä-8 = 24. kan man lage følgende spørring av bredde og dybde.

$$f(bredde, dybde) = frekvens$$

Videre kan man se på endringen av en enhets-endring dybde mot en enhetsendring av bredde i området med høy bredde.

$$f\left(Bredde_{høy}, Dybde_{konstant}\right) - f\left(Bredde_{Lav}, Dybde_{konstant}\right) = \Delta f_{bredde}$$

$$f(35000, 24000) - f(34000, 24000) = \Delta f_{bredde}$$

$$\Delta f_{bredde} = 0.392Hz - 0.397Hz = -0.005Hz$$

Mot

$$f\left(Bredde_{konstant}, Dybde_{lav}\right) - f\left(Bredde_{konstant}, Dybde_{høy}\right) = \Delta f_{bredde}$$

$$\Delta f_{dybde} = f(35000, 24000) - f(35000, 25000)$$

$$\Delta f_{dybde} = 0.392Hz - 0.394Hz = -0.002Hz$$

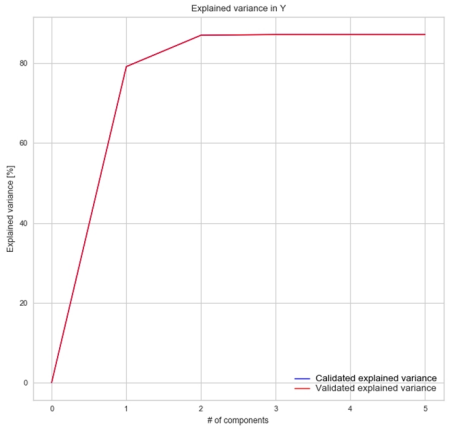
Dette er ganske interessant da man i området med høy dybde kan endre dybden uten å øke frekvensen noe spesielt. Det er en økning på 0.002Hz i eksemplet over, men det er utrolig lite i en praktisk sammenheng. Ser også ved å sammenligne grafene for antall trä-8 = 1 i Figur 34 mot grafen med antall trä-8 = 24 i Figur 35. Denne effekten økes om det er ett lite antall trä-8 dekker i modellene.

5.4 Vindindusert akselerasjon

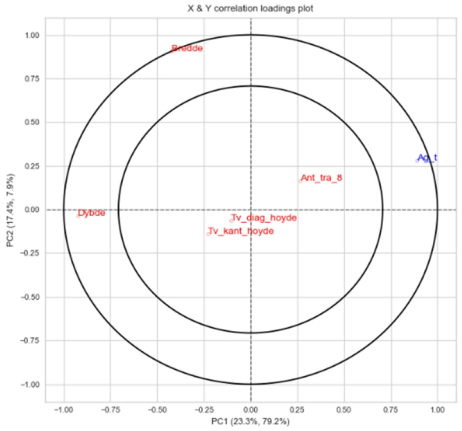
Fremstilling av sammenhengen mellom variablene med hensyn til akselerasjon er foretatt på samme måte som for frekvens.

5.4.1 PLS analyse for vind-indusert akselerasjon

På samme måte som for egenfrekvensen er det gjort en PLS1 analyse med Y = $A_g(t)$ og X_i = [Bredde, Dybde, Antall trä-8, Tv diagonal, Tv kant]. Oppsettet av analysen finnes i vedlegg B, «*PLS1 Analyse med Y = Agt.*»



Figur 36 - forklart varians av akselerasjon (PLS1)



Figur 37 - korrelasjonslast plot for akselerasjon (PLS1)

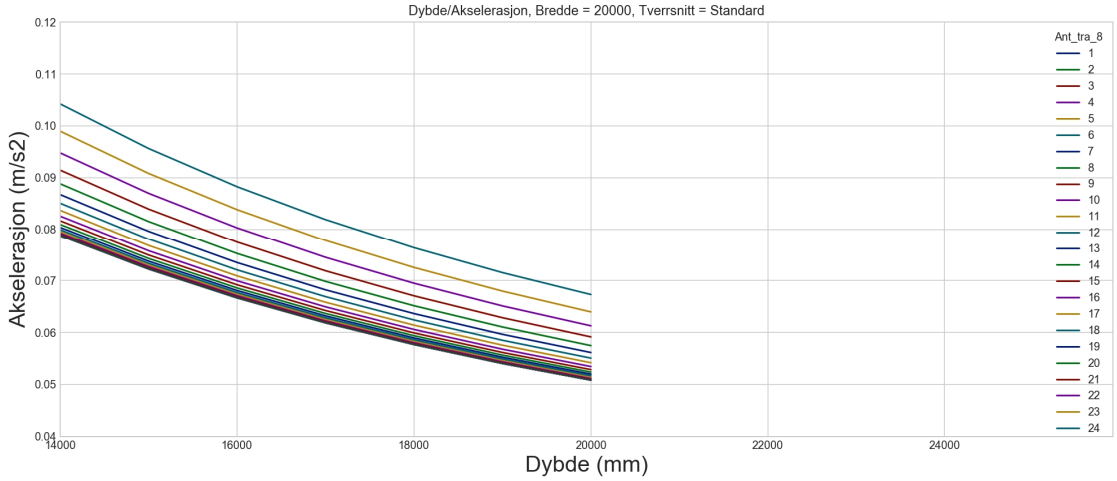
På samme måte som for PLS-analysen for egenfrekvens er det meste av variansen til Akselerasjonen forklart i de to første prinsipalkomponentene. PC1 representerer 79.2% av variansen til Akselerasjonene, 7.9% er representert i PC2. PC3 forklarer 0.16%. Dette er så lite at prinsipalkomponent 3 og de øvrige prinsipalkomponentene er neglisjert fra videre tolkning.

Ser i Figur 37 at dybde er negativt korrelert med akselerasjonen, dette betyr at akselerasjonen har en tendens til å gå ned med økt dybde. Ser man på de andre variablene så vil økt tverrsnitt bidra til lavere akselerasjon samt at økt antall trä-8 dekker vil bidra til høyere akselerasjon. Overaskende nok er det en liten positiv korrelasjon mellom bredde og dybde som igjen er negativt korrelert med akselerasjonen. Dette er til en viss grad motsigende fra teorien som tilsier at økt bredde på bygget vil bidra til økt angrepsflate for vinden som vil kunne bidra til høyere akselerasjon, men i tillegg til denne effekten har man PC2 som antyder at bredden er positivt korrelert med frekvensen, det er vanskelig å si noe konkret om korrelasjonen mellom frekvens og bredde i PC1 og PC2 uten noe videre undersøkelse. I PC1 er det lav last på bredde med

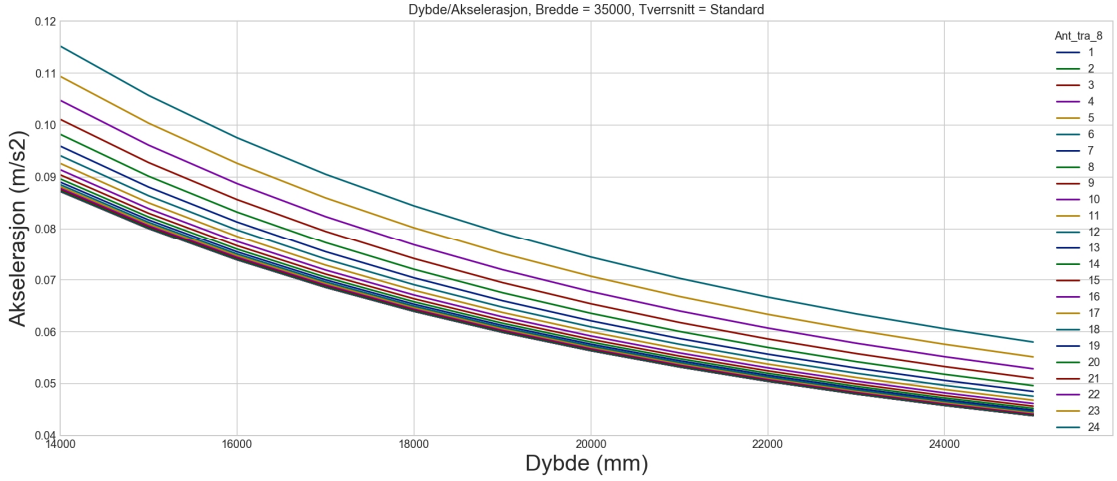
høy frekvens, i PC2 er det høy last på bredde med en mindre last på frekvens samt en mindre global forklaring av variansen til frekvens.

5.4.2 Endring av Dybde, Bredde og Antall trä-8 dekker med tverrsnitt konstant

Grunnen til at grafene i Figur 38 stopper ved dybde = 20000 mm er fordi bredden er satt til 20000 mm. Husker man tilbake til delkapittelet «*4.1.1 - Geometri*» vet man at modellens bredde ikke kunne være mindre enn modellens dybde. Grunnen til at x-aksen ikke er skalert til å være mellom 14000-20000 er for at man skal kunne sammenligne Figur 38 og Figur 39.



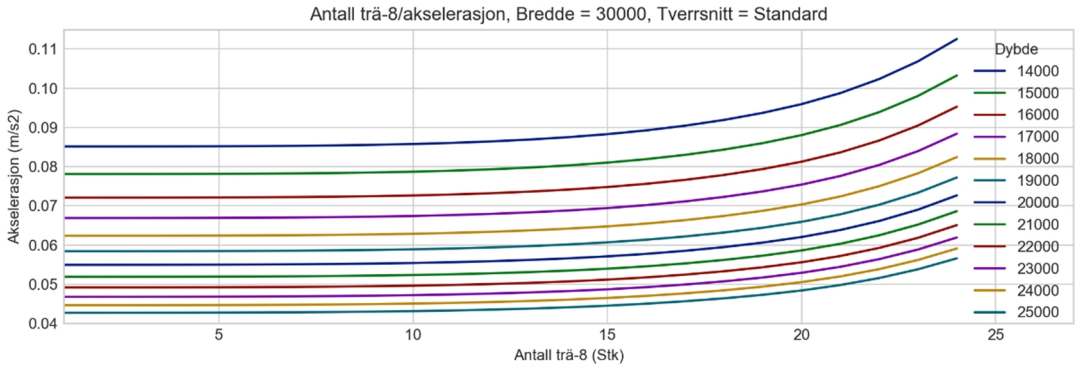
Figur 38 - Dybde/akselerasjon, Bredde = 20000, Tverrsnitt = Standard



Figur 39 - Dybde/akselerasjon, Bredde = 35000, Tverrsnitt = Standard

Figur 38 og Figur 39 viser hvordan akselerasjonen endrer seg med hensyn til dybde og antall trä-8 dekker med en bredde på 20000 mm og 35000 mm. Den øverste grafen representerer modeller med 24 trä-8 dekker, neste graf representerer 23 trä-8 dekker osv... Akselerasjonen går ned ved økt dybde, samt at en reduksjon i trä-8 dekker er med på å senke akselerasjonen. Ser også hvordan endringen av antall trä-8 dekker har veldig stor innvirkning ved de første reduksjonene av trä-8 dekker. I tillegg ser man en liten interaksjon mellom antall trä-8 dekker og dybden, man kan se at de to øverste grafene i Figur 39 ikke ligger parallelt med hverandre, og at denne interaksjonen er størst ved lav dybde.

man kan se på ett sett modeller med Tverrsnitt = Standard og Bredde = 30000mm som er plotet i Figur 40.



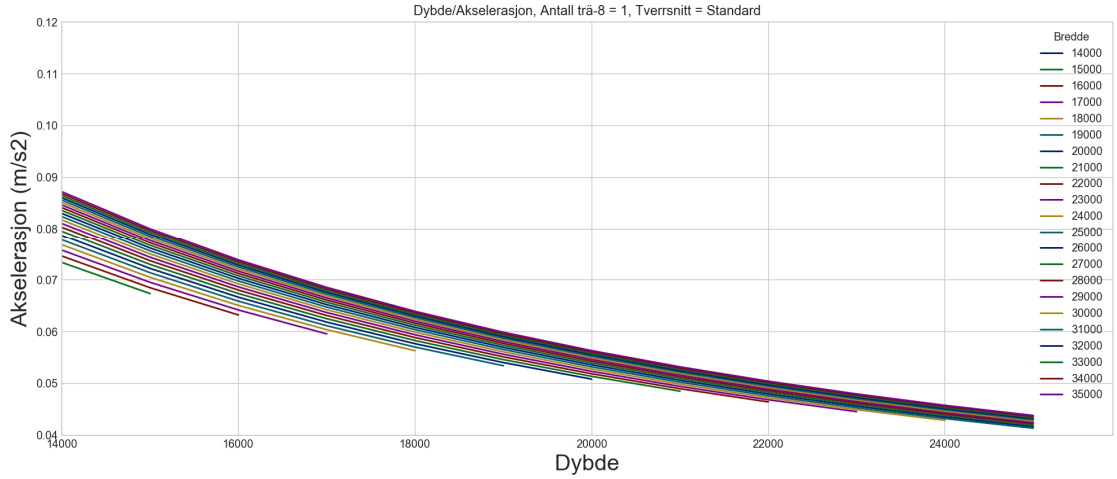
Figur 40 - Antall trä-8/Akselerasjon, Bredde =30000, Tverrsnitt = Standard

I Figur 40 ser man hvordan en økning av antall trä-8 har på modeller med forskjellige dybder. En reduksjon av antall trä-8 dekker har selvfølgelig enn sammenheng med massen til bygget, men reduksjonen av trä-8 dekker impliserer også at den ekstra massen fra betongdekkene som erstatter trä-8 dekkene legges lenger ned i bygget. Og det er grunne til at antall trä-8 har en eksponentiell effekt på akselerasjonen.

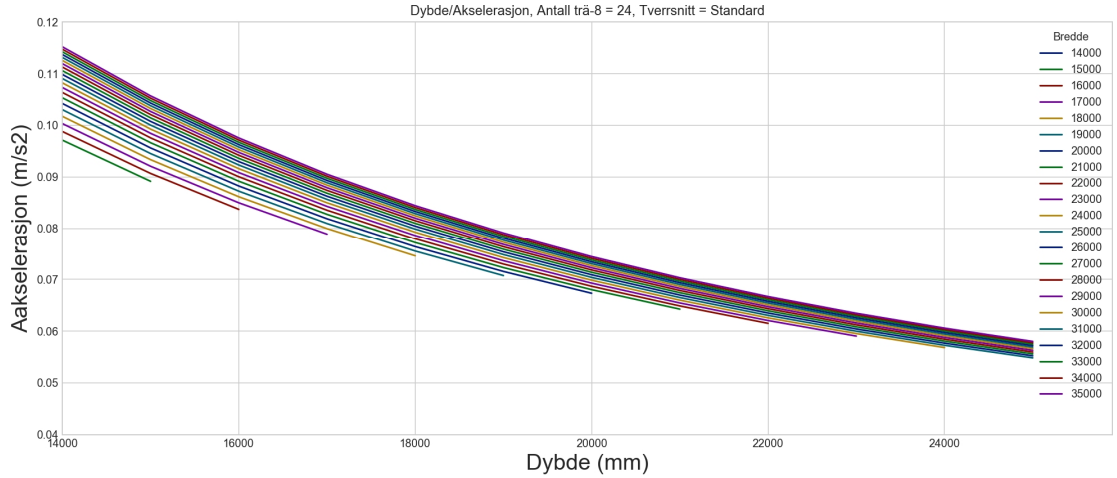
5.4.3 Endring av Dybde, Bredde og Antall trä-8 dekker med tverrsnitt konstant

Nederste graf i Figur 41 og Figur 42 representerer modeller med en bredde = 15000, nest nederste graf representerer modeller med bredde = 16000 osv… Grunne til at grafene som representerer modeller med lav bredde ikke eksisterer med modellens

dybde større en modellens bredde er igjen fordi det ble bestemt at modellenes bredde ikke kunne være mindre en modellens dybde.



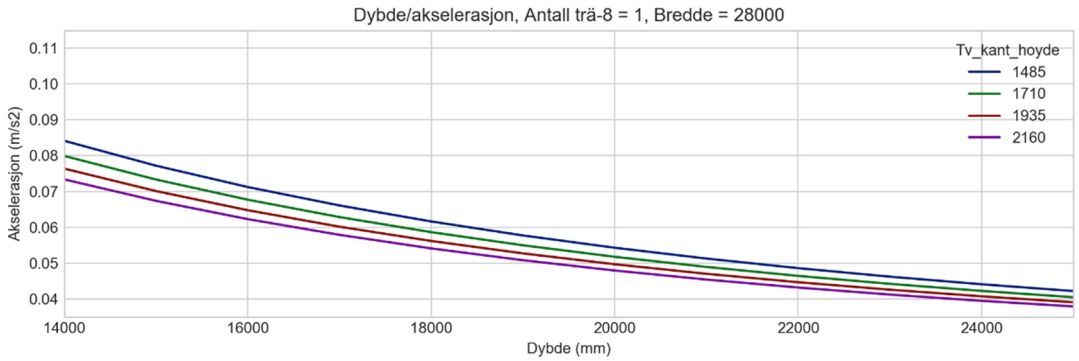
Figur 41 - Dybde/akselerasjon, Antall trä-8 = 1, Tverrsnitt = Standard



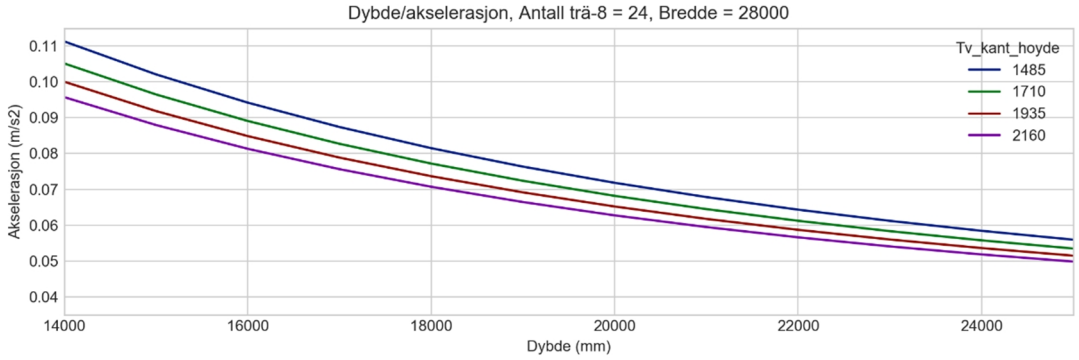
Figur 42 - Dybde/akselerasjon, Antall trä-8 = 24, Tverrsnitt = Standard

Ser på figuren over at bredden ikke har på langt nær samme effekten som dybde når det kommer til å endre akselerasjon. Ved å endre bredden fra 15000 mm til 35000 mm når dybden er 14000 kan man potensielt endre akselerasjonen med ca $\approx 0.02\, m/s^2$. Ser også at akselerasjonsdifferansen mellom bredde = 15000 og 35000mm er mye høyere med ett stort antall trä-8 dekker.

5.4.4 Endring av Dybde, Antall trä-8 og kant tverrsnitt med bredde konstant



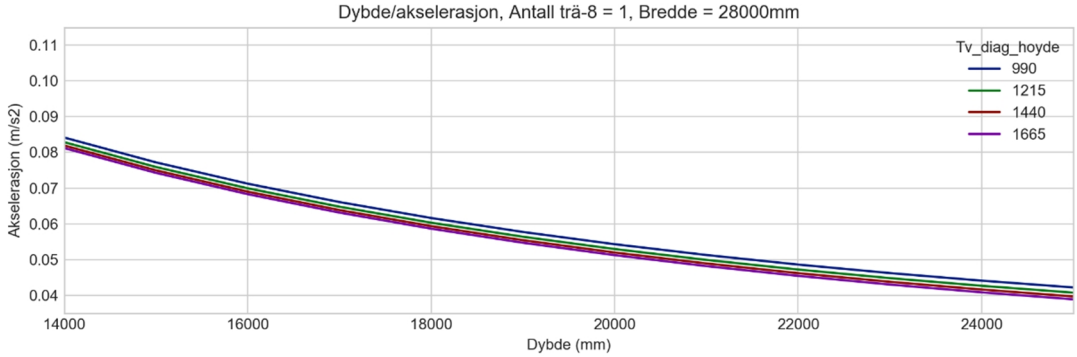
Figur 43 - Dybde/akselerasjon, Antall trä-8 = 1, Bredde = 28000 Tverrsnitt = Kant



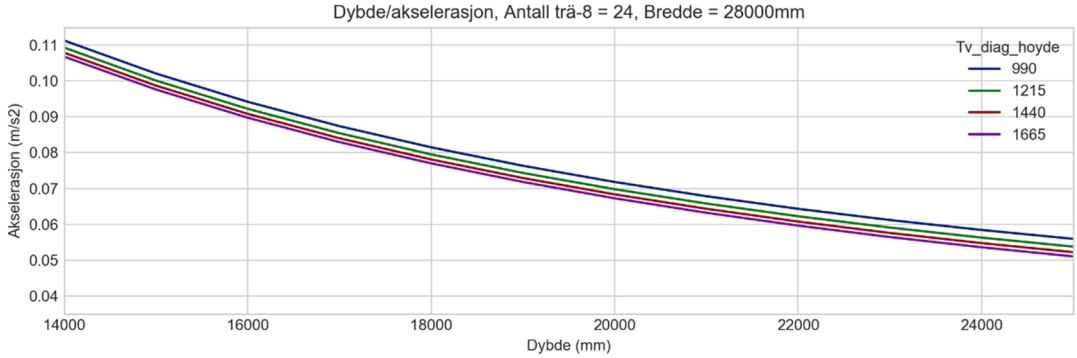
Figur 44 - Dybde/akselerasjon, Antall trä-8 = 24, Bredde = 28000 Tverrsnitt = Kant

I Figur 43 og Figur 44 er det avstanden mellom grafene som representerer endringen av akselerasjon med tanke på kantbjelkene. Ser her at endringen av kantbjelke har størst innvirkning på akselerasjonen med modeller som har liten dybde. Effekten av å endre høyden til kantbjelkene avtar med høyere dybde. Akselerasjonsdifferansen mellom kant tverrsnitt på 1485mm og 2160mm ved dybde = 14000mm er på rundt $0.017\ m/s^2$. Med en lik dybde = 25000 mm er den samme akselerasjonsdifferansen på rundt $0.007\ m/s^2$. Ser også ved å sammenligne grafene i Figur 43 og Figur 44 at et lavt antall trä-8 dekker bidrar til mindre effekt av å øke/minke høyden på kantbjelkene.

5.4.5 Endring av Dybde, Antall trä-8 dekker og diagonalbjelkene med bredde konstant



Figur 45 - Dybde/akselerasjon, Antall trä-8 = 1, Bredde = 28000 Tverrsnitt = Diagonal



Figur 46- Dybde/akselerasjon, Antall trä-8 = 24, Bredde = 28000 Tverrsnitt = Diagonal

Ser man på Figur 45 og Figur 46 kan man på samme måte som med Figur 43 og Figur 44 se endring av høyde på diagonalbjelkene med hensyn til akselerasjonen. Avstanden mellom grafene er mye mindre her enn den var for endring av kantbjelkene. I tillegg har man en motsatt effekt fra det man så for kantbjelkene, endringer av akselerasjon er høyere med modeller med høy dybde. Og på samme måte som for kantbjelken er det en høyere varians når det er mange trä-8 dekker.

5.5 Akselerasjonsutnyttelse

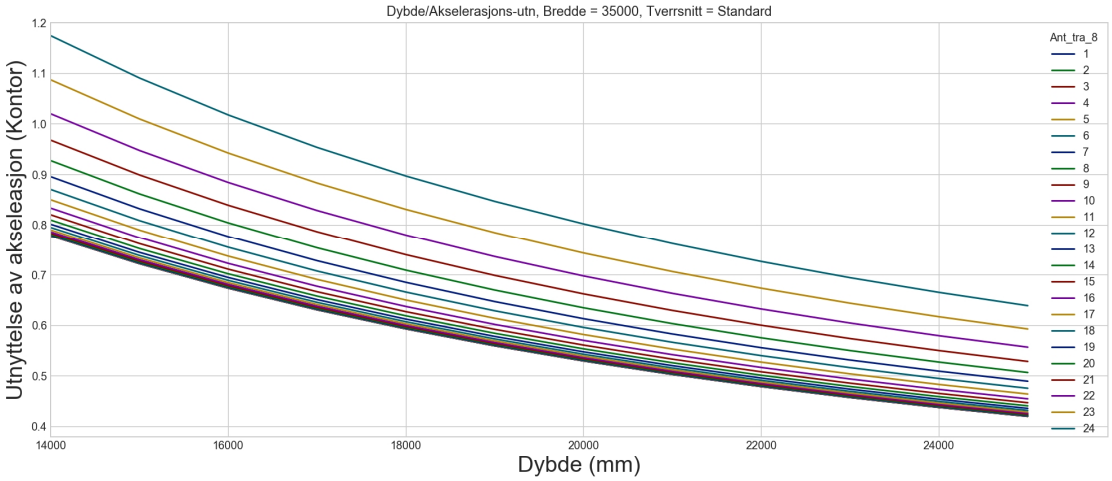
Med akselerasjonsutnyttelse mener man i hvor stor grad man utnytter konstruksjonens vindinduserte akselerasjon mot kravet i ISO 10137. Måten man regner dette på er

$$a_{g,opptredende} / a_{g,krav\ kontor}$$

Verdien vil variere fra null og oppover. Er akselerasjonsutnyttelsen over 1 betyr det at konstruksjonen ikke overholder akselerasjonskravet gitt i ISO 10137.

Man kunne tenkt seg at akselerasjonskravet ville fulgt samme trend som den opptredende akselerasjonen, men det er ikke helt tilfelle. Fra kapittelet «3.2 - Vind» vet man at kravet om maks akselerasjon varierer med egenfrekvensen til konstruksjonen. Dette åpner opp for mulige endringer fra det man tolket om den opptredende akselerasjonen da man får ett samspill mellom opptredende akselerasjon og egenfrekvens.

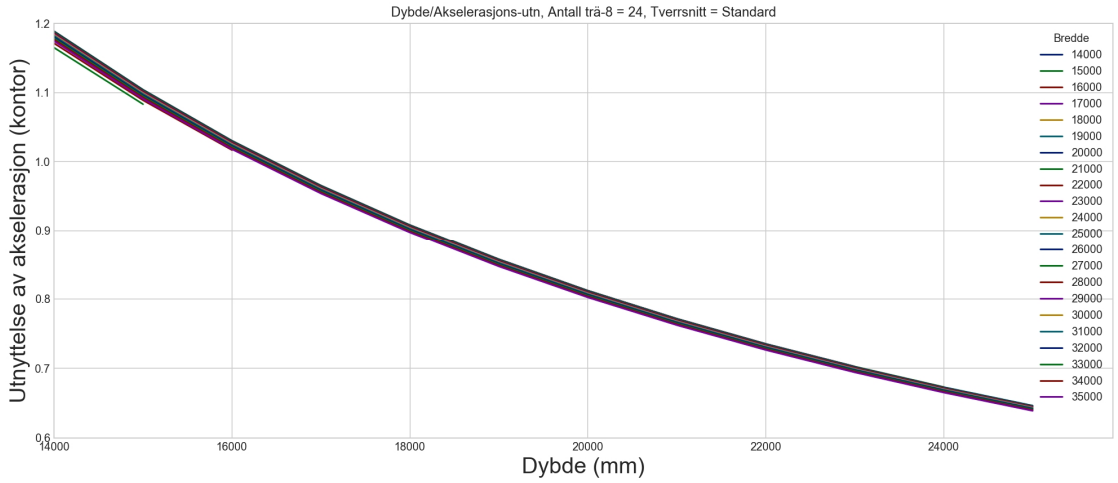
5.5.1 Endring av Dybde og Antall tra-8 dekker med konstant bredde og tverrsnitt



Figur 47 - Dybde/akselerasjons-utn, bredde = 35000, Tverrsnitt = Standard.

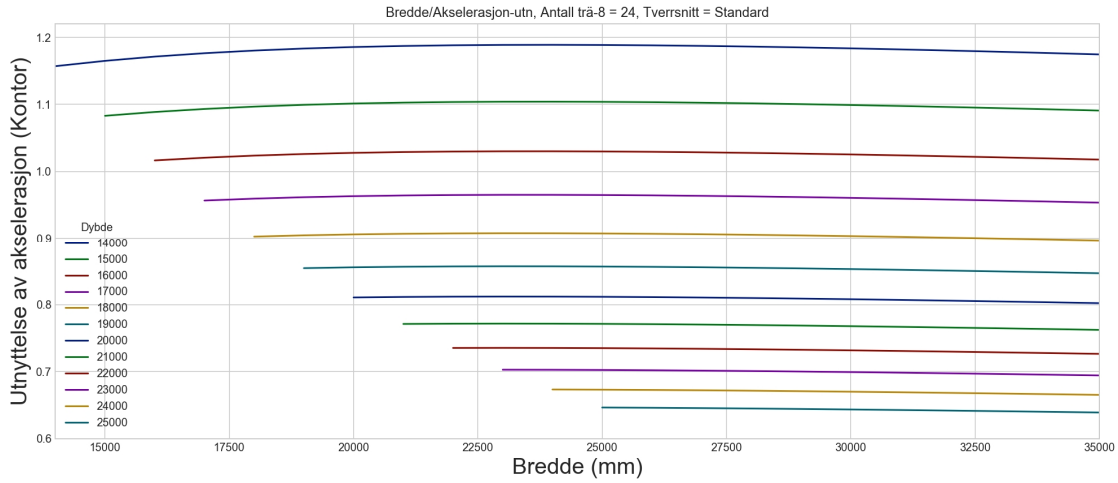
Ser man på figuren over ser man i stor grad den samme tendensen som beskrevet i kapittelet *«5.4 - Vindindusert* akselerasjon». Utnyttelsen synker med høyere dybde. Det er noe interessant at en modell med en bredde på 35000mm, lav dybde og høyt antall trä-8 dekker kan komme under akselerasjonskravet.

5.5.2 Endring av Dybde og Bredde med konstant antall trä-8 dekker og tverrsnitt



Figur 48 - Dybde/Akselerasjons-utn, Antall trä-8 = 24, Tverrsnitt = Standard.

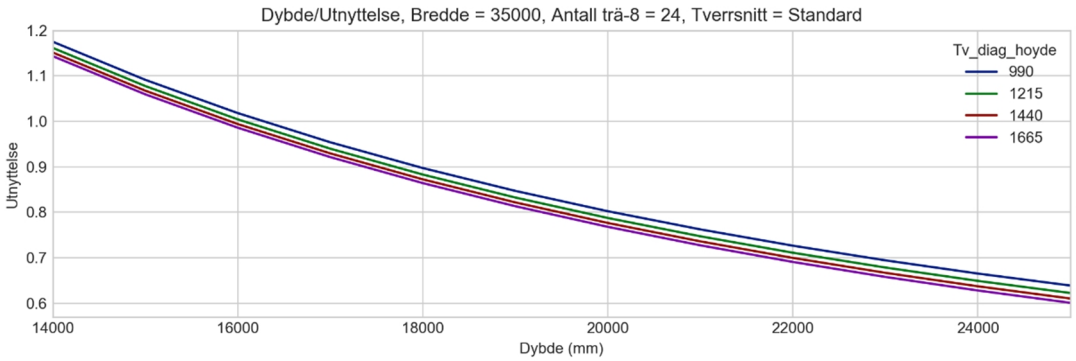
På Figur 48 ser man hvordan en endring av dybde har på konstruksjoner med en standard sammensetning av tverrsnitt samt 24 trä-8 dekker. Det er nesten umulig å se hvilen graf som representerer hvilken bredde i figuren, men at alle grafene er i en så tett klynge indikerer at bredden har veldig lite å si for akselerasjonskravet. Det kan altså tyde på at massebidraget fra økt bredde overstiger effekten av økt angrepsflate for vindende i disse situasjonene. Man kan plote grafene med bredde i x-aksen og dybden satt inn som gruppe. Da får man figuren under.



Figur 49 - Bredde/Akselerasjons-utn, Antall trä-8, Tverrsnitt = Standard

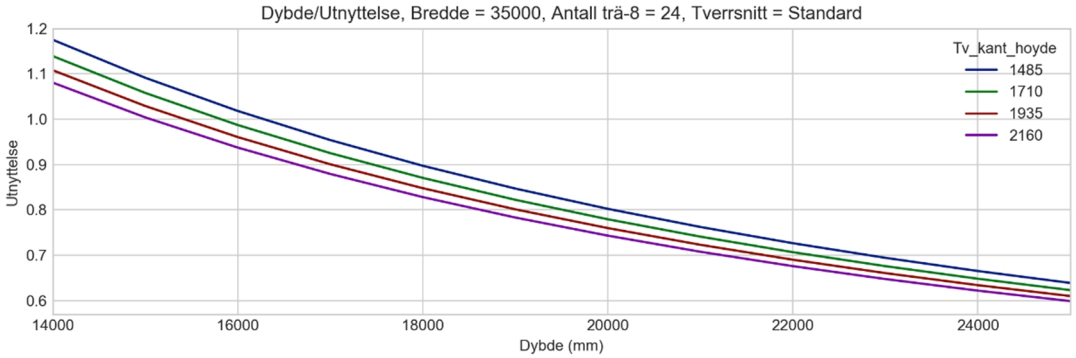
Ser på Figur 49 hvor liten effekten endring av bredde har på utnyttelsen. Dette er fordi stigningstallet til grafene i figuren over er tilnærmet lik null sammenlignet med dybden. Det er en lokal effekt som gjør at utnyttelsen får ett topp-punkt i figuren over. Gjør man en liten oppsummering av alle bidragene til bredde kan man si at bredden er med på å senke frekvensen og øke akselerasjonen, men det siste bidraget er mye mindre enn for dybde. I tillegg vet man at akselerasjonskravet blir strengere med økt frekvens. Det kan da tenkes at man får ett topp-punkt som definerer punktet hvor bidraget fra redusert frekvens med tanke på høyere bredde gir mer utslag enn det økt akselerasjon gjør.

5.5.3 Endring av Dybde og Tverrsnitt med konstant antall trä-8 og bredde



Figur 50 - Dybde/Akselerasjons-utn, Bredde = 35000, Antall trä-8 = 24, Tverrsnitt = Diagonal

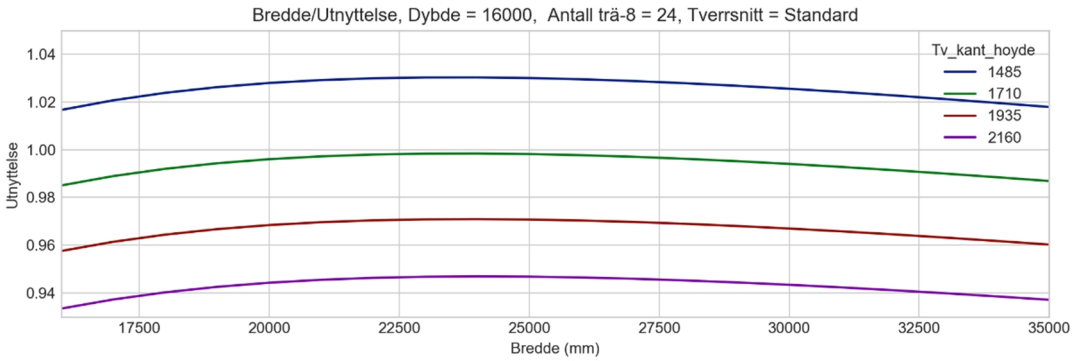
Utnyttelsen av akselerasjonskravet for endring av diagonal tverrsnittet er vist i Figur 50. Endring av høyden til diagonalbjelkene følger samme oppførsel som den for akselerasjon. Effekten av endring av tverrsnittene har mer å si ved høy dybde, men fortsatt en liten lokal effekt i forhold til å endre dybden på bygget.



Figur 51 - Dybde/Akselerasjons-utn, Bredde = 35000, Antall trä-8 = 24, Tverrsnitt = Kantbjelke

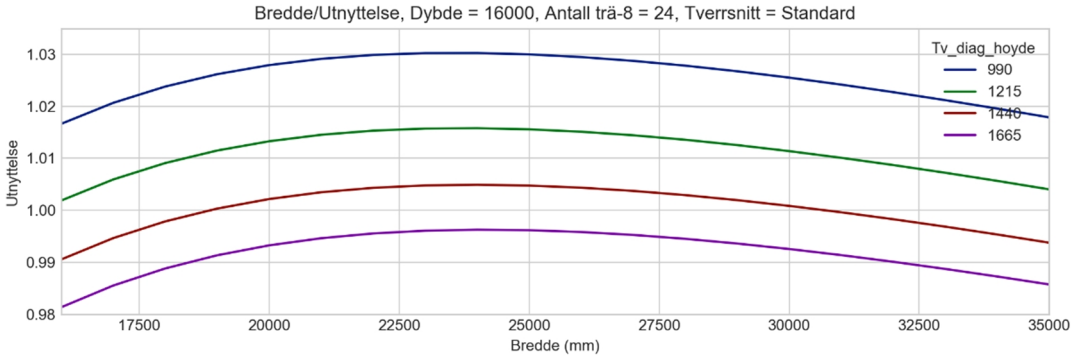
Endring av høyden til kantbjelkene oppfører seg likt som for akselerasjonsøkningen. og endringen er størst ved lav dybde. Sammenligner man Figur 50 og Figur 51 ser man hvordan endring av høyden til kantbjelkene gir større endring av utnyttelsen enn det endring av høyden til diagonalbjelkene. Det ser man fordi avstanden mellom grafene er større i Figur 51

5.5.4 Endring av Bredde og tverrsnitt med konstant antall trä-8 og Dybde.



Figur 52 - Bredde/Akselerasjons-utn, Dybde = 16000, Antall trä-8 = 24, Tverrsnitt = Kantbjelke

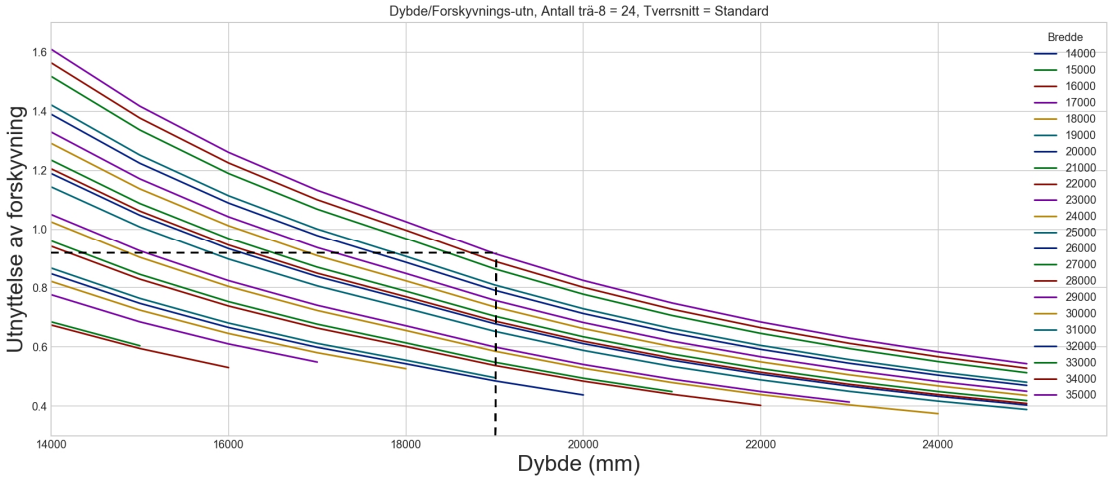
For å se om oppførselen til bredde/Akselerasjons-utn endrer seg ved å endre egenfrekvens eller akselerasjon utenom bredde er det plotet Figur 52 og Figur 53. Ser her at endring av høyden til kantbjelkene ikke gir noe endring av oppførselen til bredde/akselerasjons-utn.



Figur 53 - Bredde/Akselerasjons-utn, Dybde = 16000, Antall trä-8 = 24, Tverrsnitt = Diagonal

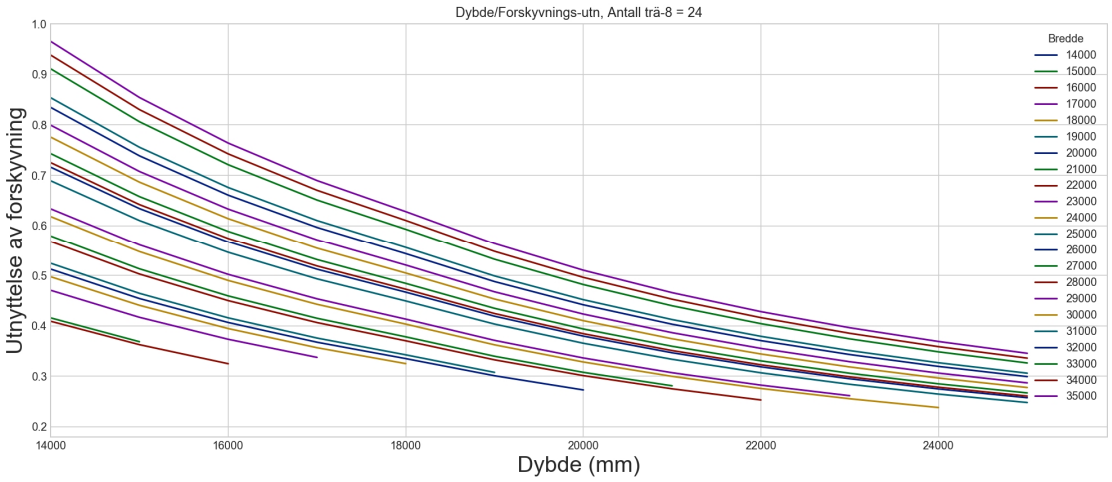
5.5.5 Problematikken med urealistisk myke konstruksjoner

Som nevnt tidligere er det en sannsynlighet for at man ser på modeller som viser tendenser til å komme innenfor de dynamiske brukskravene, men ikke gjenspeiler en reell løsning da modellene kan være urealistisk 'myke'. Figur 54 representerer de samme konstante variablene som plotet i Figur 48 bare med forskyvningsutnyttelsen istedenfor akselerasjonsutnyttelsen. Ser her at det er en rekke beregninger som er mye 'mykere' en det som er tillatt. Ser man for eksempel på beregningene med bredde på 35000 mm må man helt opp til en dybde på 19000 mm før man kommer under forskyvningskravet. Det er illustrert med en svart dottet linje i Figur 54.

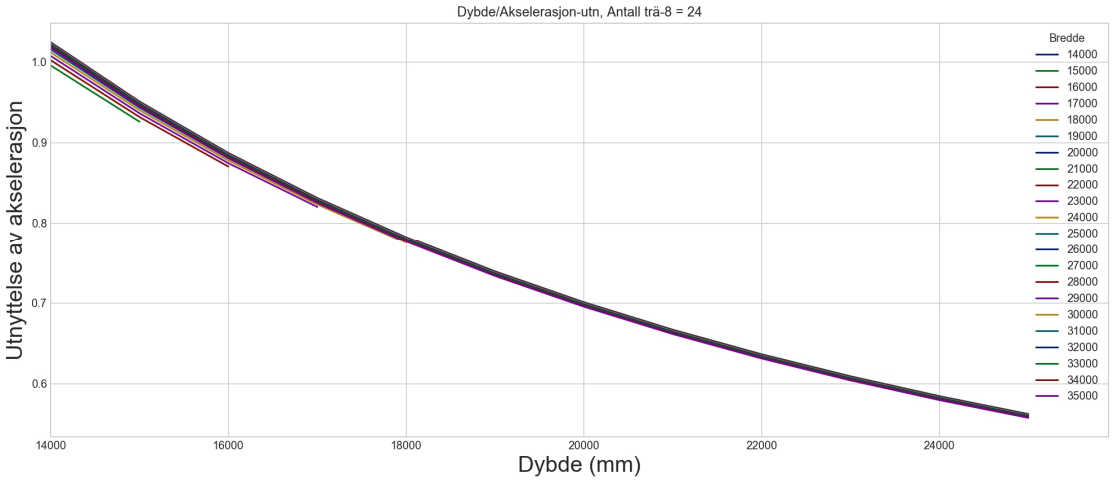


Figur 54 - Dybde/Forskyvnings-utn, Antall trä-8 = 24, Tverrsnitt = Standard

Basert på det over kan det være interessant å se hvilken effekt man får på akselerasjons-kravet om man øker stivheten til konstruksjonene med større tverrsnitt slik at man kommer under forskyvningskravet. Nå vet man jo hvordan økt tverrsnitt bidrar til lavere akselerasjon samt høyere frekvens. Igjen så er akselerasjonskravet avhengig av frekvensen som gjør akselerasjonskravet strengere ved økt frekvens. Det er derfor ikke sikkert at økt tverrsnitt gir en 'positiv' effekt. Figur 55 Figur 56 er plotet med tverrsnittsbredde = 1125 mm, kantbjelke høyde = 1935 mm og diagonalbjelke høyde = 990.



Figur 55 - Dybde/Forskyvnings-utn, Antall trä-8 = 24, Kant = 1935, Tv-bredde = 1125.



Figur 56 - Dybde/Akselerasjon-utn, Antall trä-8 = 24, Kant = 1935, Tv-bredde = 1125.

Ser nå at forskyvningskravet er under 1. Ser da på Figur 56 at utnyttelsen av akselerasjonen er lavere enn for den vist i Figur 48. Dette betyr at økning av stivhet fra tverrsnittene gir mer bidrag til redusert akselerasjon en det strengere akselerasjonskrav gir på grunn av økt frekvens. Det kan diskuteres om tverrsnittene brukt i Figur 55 og Figur 56 er realistiske med tanke på størrelse, men det er i hovedsak det faktumet at man er relativt frie til å øke stivheten uten for store konsekvenser for akselerasjonskravet som er interessante i denne situasjonen.

6 Konklusjon

Problemstillingen i denne oppgaven er «*Hvordan endres de dynamiske egenskapene for ett høyt trehus ved å endre fotavtrykk, massefordeling og tverrsnitt?*» for å finne svaret på dette problemet ble det bygget ett Python-program som itererte seg frem til alle mulige sammensetninger av variablene presentert i problemstillingen. Totalt ble det konstruert 1 520 640 modeller som ble beregnet for å finne forskyvningen i topp-etasjen, vindindusert akselerasjon, egenfrekvens og andre resultater. I ettertid av beregningene fant man ut at det i realiteten var kun nødvendig å jobbe videre med 380 160 modeller fordi variabelen *Tverrsnitt vannrett* ikke ga noe bidrag til de dynamiske egenskapene for modellene.

Forenklingene gjort i denne oppgaven viser seg å gi overaskende presise resultater, de dynamiske resultatene beregnet i Python-programmet hadde veldig små avvik fra den fullverdige Robot structural analysis modellen.

De mest interessante observasjonene som ble gjort i resultater og analyse kapittelet var i første omgang antall modeller som var innenfor brukskravene. Det var veldig få modeller som ikke var innenfor de dynamiske brukskravene, det var faktisk flere modeller som var innenfor de dynamiske brukskravene enn det var modeller som var innenfor forskyvningskravet. Ved analysering av utnyttelsen til akselerasjonskravet kom det frem at endring av bredde ga tilnærmet ingen effekt på utnyttelsen av akselerasjonskravet. Dette var tilfelle selv med modeller som hadde lav dybde og høyt antall trä-8 dekker. Det kom også frem i modellene at man kommer til ett punkt hvor økt bredde reduserte utnyttelsesgraden til akselerasjonskravet.

På ett litt mer lokalt nivå kom det frem at endring av høyden på kantbjelkene ga størst endring på frekvens, akselerasjon og utnyttelse av akselerasjonskravet når modellene var veldig smale. Endring av høyden til diagonalbjelkene ga størst innvirkning på frekvens, akselerasjon og utnyttelse av akselerasjonskravet ved stor dybde. og innvirkningen beskrevet i dette avsnittet ble forsterket i situasjoner hvor det var ett høyt antall trä-8 dekker. Om man skulle sammenlignet kantbjelkene og diagonalene ville det gitt størst endring på egenfrekvens og akselerasjon ved å endre høyden på kantbjelkene.

Litteraturliste

Abrahamsen, R. (2017). *Mjøstårnet-Construction of an 81 m tall timber building*. Internationales Holzbau-Forum IHF 2017 Garmisch-Partenkirchen, Congress Centrum.

Bell, K. (2011). *Matrisestatikk, Statiske beregninger av rammekonstruksjoner*: Tapir akademisk forl., Trondheim.

Bjertnæs, M. (2017). *Byggesystemer i høyden*. TREdagen, Thon Hotel Arena, Nesgata 1, Lillestrøm.

Boggs, D. (1995). Acceleration indexes for human comfort in tall buildings–Peak or RMS. *CTBUH monograph*.

Community, N. (2018). *NumPy Reference* (Versjon 1.14.2). Python modul.

Davenport, A. G. (1961). The spectrum of horizontal gustiness near the ground in high winds. *Quarterly Journal of the Royal Meteorological Society*, 87 (372): 194-211.

Dunn, K. (2018). Process Improvement Using Data.

Hansen, O. & Fjeld Olsen, M. (2016). *Measuring Vibrations and assessing Dynamic Properties of tall Timber Buildings-Måling av vibrasjoner og kartlegging av dynamiske egenskaper i høye trehus*: NTNU.

Holmes, J. D. (2015). *Wind loading of structures*: CRC press.

Ivarsson, N. & Nellber, A. (2016). *Trä8, ett pelar-balksystem i limträ: Tekniskt val av träbaserat stomsystem vid uppförande av ett flervåningshus i Sverige*.

Juveli, A. (2016). *Ulike løsninger for horisontal avstivnings effekt på den dynamiske responsen i høye hus*: Norwegian University of Life Sciences, Ås.

Kvalheim, O. M. (2010). Interpretation of partial least squares regression models by means of target projection and selectivity ratio plots. *Journal of Chemometrics*, 24 (7-8): 496-504.

Liven, H. (2017). *Mjøstårnet og andre høye bygg*. TREdagen, Thon Hotel Arena, Nesgata 1, Lillestrøm.

Powell, V. & Lehe, L. *Principal Component Analysis Explained Visually*. Tilgjengelig fra: http://setosa.io/ev/principal-component-analysis/.

Rao, S. S. (2017). *The finite element method in engineering*: Butterworth-heinemann.

Standard, N. (2009). *Eurokode 1: Laster på konstruksjoner: Del 1-4 : Allmenne laster. Vindlaster*. Eurocode 1: Actions on structures. Part 1-4: General actions. Wind actions, b. NS-EN 1991-1-4:2005+NA:2009. Lysaker: Standard Norge.

Steenbergen, R., Geurts, C. & Vrouwenvelder, A. (2009). *Dynamics of Tall Buildings Under Stochastic Wind Load, Applicability of Eurocode EN 1991-1-4 Procedures 1 and 2*. 5th European & African conference on wind engineering: Florence Italy, July 19th-23rd 2009: conference proceedings.: Firenze University Press.

Svendsen, L. (2016). *Tilpasningsdyktige kontorbygg-Erfaring fra 4 caser*: NTNU.

Sættem, J. L. (2016). *Lydkomfort i fleretasjes studentboliger i massivtre*: Norwegian University of Life Scineces, Ås.

Sørgjerd, C. (2018). Planlegger over 40 nye høyhus i Oslo – her kan de komme. *Aftenposten*. Tilgjengelig fra: https://www.aftenposten.no/osloby/i/KvpzG7/Planlegger-over-40-nye-hoyhus-i-Oslo--her-kan-de-komme.

Talja, A. & Fülöp, L. (2016). *Evaluation of wind-induced vibrations of modular buildings*.

Tomic, O. (2017). *Hoggorm documentation* (Versjon 0.11.3). Python modul.

Vedlegg A – Verifisering av arbeidsgang

Tabell 11 - Masse fra 3D robot modell.

Etasje	Høyde [mm]	Masse [kg]	1/2 masse [kg]	1/4 masse [kg]	1/8 masse [kg]	
1	4900	312030	156015	78008	39004	Trä-8 dekke
2	8900	292083	146041	73021	36510	
3	13100	289971	144986	72493	36246	
4	17100	287962	143981	71991	35995	
5	21100	286769	143385	71692	35846	
6	25100	288222	144111	72056	36028	
7	29100	295058	147529	73765	36882	
8	33100	273540	136770	68385	34193	
9	36900	269851	134926	67463	33731	
10	40700	269086	134543	67271	33636	
11	44250	601201	300600	150300	75150	Betong dekke
12	48050	606644	303322	151661	75830	
13	51850	610343	305172	152586	76293	
14	55650	608789	304395	152197	76099	
15	59450	611085	305542	152771	76386	
16	63650	607912	303956	151978	75989	
17	68450	597144	298572	149286	74643	
18	74450	97250	48625	24312	12156	
	Summert	**7204941 kg**	**3602471 kg**	-	-	

Tabell 12 - *Dynamiske resultater 3D / 2D robot*

Beskrivelse	Frekvens [Hz]	Periode [Sec]	masse [kg]	½ masse [kg]	Rel.mass [%]	Rel.mass [kg]	½ Rel.masse [kg]
Forenklet Sweco Robot 3D-modell	0.405	2.4692	7288979	3644489	77.92 %	5679806	2839902.77
Forenklet Robot 2D-modell	0.403	2.3756	3602470	-/-	77.81 %	2803082	-/-

Tabell 13 - *Forskyvninger i 3D / 2D modell*

		Robot 3D				Robot 2D				Differanse	
Etasje	**Høyde** [m]	**Venstre** [mm]	**Midt** [mm]	**Høyre** [mm]	**G.snitt** [mm]	**Venstre** [mm]	**Midt** [mm]	**Høyre** [mm]	**G.snitt** [mm]	**Løpende** [mm]	**ΔDiff** [mm]
1	4,90	5.22	5.22	5.18	5.21	5.33	5.22	5.10	5.22	0.12	0.12
2	8,90	9.95	9.95	9.94	9.94	10.11	9.52	9.81	9.81	0.16	0.04
3	13,10	16.30	16.30	16.25	16.28	16.61	16.43	16.28	16.44	0.32	0.16
4	17,10	22.76	22.76	22.51	22.68	23.36	23.07	22.77	23.07	0.60	0.28
5	21,10	29.04	29.04	28.97	29.02	29.73	29.54	29.38	29.55	0.69	0.09
6	25,10	35.39	35.39	35.36	35.38	36.11	35.63	35.81	35.85	0.72	0.03
7	29,10	42.25	42.25	42.29	42.27	42.97	42.84	42.76	42.86	0.72	0.00
8	33,10	48.93	48.93	49.33	49.06	49.61	49.69	49.77	49.69	0.67	-0.04
9	36,90	56.48	56.48	56.51	56.49	57.48	57.39	57.25	57.37	1.00	0.32
10	40,70	63.26	63.26	63.22	63.25	64.50	64.14	64.21	64.28	1.24	0.24
11	44,25	69.38	69.38	69.35	69.37	70.83	70.48	70.55	70.62	1.45	0.21
12	48,05	75.73	75.73	75.68	75.72	77.45	77.29	77.08	77.27	1.72	0.27
13	51,85	81.63	81.63	81.31	81.52	83.56	83.16	82.75	83.16	1.93	0.22
14	55,65	87.88	87.88	87.83	87.87	89.92	89.70	89.53	89.72	2.03	0.10
15	59,45	93.77	93.77	93.75	93.76	95.91	95.52	95.63	95.69	2.14	0.11
16	63,65	99.85	99.85	99.84	99.84	102.11	101.97	101.88	101.99	2.26	0.12
17	68,45	105.71	105.71	105.89	105.77	108.14	108.19	108.24	108.19	2.43	0.17

Tabell 14 – *Normalisert modshape i 3D / 2D modell i mode 1 (0.405Hz / 0.403 Hz)*

	Robot 3D				Robot 2D				Differanse	
Etasje	**Venstre**	**Midt**	**Høyre**	**G. snitt**	**Venstre**	**Midt**	**Høyre**	**G. snitt**	**Løpende**	**ΔDiff**
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0278	0.0278	0.0274	0.0277	0.0254	0.0256	0.0254	0.0255	0.0022	0.0022
2	0.0590	0.0574	0.0588	0.0584	0.0563	0.0544	0.0558	0.0555	0.0029	0.0007
3	0.1070	0.1070	0.1069	0.1070	0.1046	0.1044	0.1042	0.1044	0.0025	-0.0003
4	0.1600	0.1603	0.1610	0.1604	0.1587	0.1587	0.1591	0.1588	0.0016	-0.0009
5	0.2126	0.2123	0.2123	0.2124	0.2113	0.2107	0.2104	0.2108	0.0016	0.0000
6	0.2690	0.2673	0.2682	0.2682	0.2673	0.2651	0.2658	0.2661	0.0021	0.0005
7	0.3329	0.3323	0.3319	0.3324	0.3306	0.3295	0.3288	0.3297	0.0027	0.0006
8	0.4001	0.3995	0.3992	0.3996	0.3971	0.3960	0.3953	0.3962	0.0034	0.0007
9	0.4685	0.4678	0.4671	0.4678	0.4645	0.4633	0.4620	0.4633	0.0045	0.0011
10	0.5364	0.5342	0.5347	0.5351	0.5319	0.5288	0.5291	0.5299	0.0052	0.0007
11	0.6025	0.6005	0.6006	0.6012	0.5986	0.5954	0.5954	0.5964	0.0048	-0.0004
12	0.6735	0.6721	0.6710	0.6722	0.6710	0.6688	0.6664	0.6688	0.0034	-0.0013
13	0.7410	0.7383	0.7356	0.7383	0.7402	0.7355	0.7308	0.7355	0.0028	-0.0006
14	0.8109	0.8090	0.8078	0.8092	0.8105	0.8074	0.8050	0.8076	0.0016	-0.0012
15	0.8755	0.8723	0.8726	0.8735	0.8757	0.8704	0.8709	0.8723	0.0012	-0.0004
16	0.9403	0.9386	0.9376	0.9388	0.9407	0.9383	0.9364	0.9385	0.0004	-0.0008
17	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000

Tabell 15 - *Masse fra 2D robot modell*

Etg.	**Høyde** [mm]	**masse** [kg]	**Halve masse** [kg]	**Kvart masse** [Kg]	
1	4900	156016	78008	39004	**Trä-8 dekke**
2	8900	146041	73021	36510	
3	13100	144985	72493	36246	
4	17100	143981	71991	35995	
5	21100	143384	71692	35846	
6	25100	144112	72056	36028	
7	29100	147529	73765	36882	
8	33100	136771	68386	34193	
9	36900	134925	67463	33731	
10	40700	134543	67272	33636	
11	44250	300600	150300	75150	**Betong dekke**
12	48050	303321	151661	75830	
13	51850	305172	152586	76293	
14	55650	304395	152198	76099	
15	59450	305543	152772	76386	
16	63650	303956	151978	75989	
17	68450	298572	149286	74643	
	Summert	**3553846 kg**	-	-	

Tabell 16 - Dynamiske resultater 2D Robot / 2D Python

Beskrivelse	**Frekvens** (Hz)	**Periode** (Sec)
Robot 2D-modell	0.408	2.4510
Python 2D-modell	0.405	2.4691

Tabell 17 - *Forskyvninger i 2D Robot / 2D Python modell*

		Robot 2D				Python 2D				Differanse	
Etasje	**Høyde** [m]	**Venstre** [mm]	**Midt** [mm]	**Høyre** [mm]	**G.snitt** [mm]	**Venstre** [mm]	**Midt** [mm]	**Høyre** [mm]	**G.snitt** [mm]	**Løpende** [mm]	**ΔDiff** [mm]
1	4,90	6.77	6.70	6.27	6.58	7.03	6.93	6.46	6.81	0.27	0.27
2	8,90	10.85	10.01	10.49	10.45	11.15	10.23	10.79	10.72	0.29	0.03
3	13,10	16.84	16.63	16.52	16.66	17.25	17.14	16.99	17.13	0.41	0.12
4	17,10	23.54	23.29	23.02	23.28	23.82	23.54	23.26	23.54	0.28	-0.14
5	21,10	29.89	29.71	29.55	29.72	30.14	29.96	29.81	29.97	0.25	-0.02
6	25,10	36.28	35.78	35.97	36.01	36.54	36.06	36.25	36.28	0.26	0.01
7	29,10	43.23	43.10	43.01	43.11	43.54	43.42	43.33	43.43	0.31	0.05
8	33,10	50.00	50.08	50.16	50.08	50.37	50.46	50.55	50.46	0.37	0.05
9	36,90	58.01	57.92	57.78	57.90	58.61	58.52	58.38	58.50	0.60	0.23
10	40,70	65.14	64.77	64.84	64.92	65.78	65.46	65.48	65.57	0.65	0.05
11	44,25	71.55	71.19	71.26	71.33	72.07	71.68	71.78	71.84	0.53	-0.12
12	48,05	78.23	78.07	77.86	78.05	78.71	78.55	78.35	78.54	0.48	-0.04
13	51,85	84.37	83.97	83.56	83.97	84.92	84.53	84.14	84.53	0.54	0.06
14	55,65	90.78	90.56	90.39	90.58	91.38	91.17	91.00	91.18	0.60	0.06
15	59,45	96.80	96.43	96.52	96.58	97.45	97.09	97.18	97.24	0.65	0.05
16	63,65	103.02	102.88	102.82	102.91	103.74	103.61	103.54	103.63	0.72	0.07
17	68,45	109.29	109.33	109.37	109.33	110.04	110.08	110.13	110.08	0.75	0.03

Tabell 18 – Normalisert modshape i 2D Robot / 2D Python modell i mode 1

	Robot 2D (0.408 Hz)				Python 2D (0.405 Hz)				Differanse	
Etasje	**Venstre**	**Midt**	**Høyre**	**G. snitt**	**Venstre**	**Midt**	**Høyre**	**G. snitt**	**Løpende**	**ΔDiff**
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0278	0.0278	0.0274	0.0277	0.0254	0.0256	0.0254	0.0255	0.0022	0.0022
2	0.0590	0.0574	0.0588	0.0584	0.0563	0.0544	0.0558	0.0555	0.0029	0.0007
3	0.1070	0.1070	0.1069	0.1070	0.1046	0.1044	0.1042	0.1044	0.0025	-0.0003
4	0.1600	0.1603	0.1610	0.1604	0.1587	0.1587	0.1591	0.1588	0.0016	-0.0009
5	0.2126	0.2123	0.2123	0.2124	0.2113	0.2107	0.2104	0.2108	0.0016	0.0000
6	0.2690	0.2673	0.2682	0.2682	0.2673	0.2651	0.2658	0.2661	0.0021	0.0005
7	0.3329	0.3323	0.3319	0.3324	0.3306	0.3295	0.3288	0.3297	0.0027	0.0006
8	0.4001	0.3995	0.3992	0.3996	0.3971	0.3960	0.3953	0.3962	0.0034	0.0007
9	0.4685	0.4678	0.4671	0.4678	0.4645	0.4633	0.4620	0.4633	0.0045	0.0011
10	0.5364	0.5342	0.5347	0.5351	0.5319	0.5288	0.5291	0.5299	0.0052	0.0007
11	0.6025	0.6005	0.6006	0.6012	0.5986	0.5954	0.5954	0.5964	0.0048	-0.0004
12	0.6735	0.6721	0.6710	0.6722	0.6710	0.6688	0.6664	0.6688	0.0034	-0.0013
13	0.7410	0.7383	0.7356	0.7383	0.7402	0.7355	0.7308	0.7355	0.0028	-0.0006
14	0.8109	0.8090	0.8078	0.8092	0.8105	0.8074	0.8050	0.8076	0.0016	-0.0012
15	0.8755	0.8723	0.8726	0.8735	0.8757	0.8704	0.8709	0.8723	0.0012	-0.0004
16	0.9403	0.9386	0.9376	0.9388	0.9407	0.9383	0.9364	0.9385	0.0004	-0.0008
17	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000

Tabell 19 - Resultater 3D Robot / 2D Python

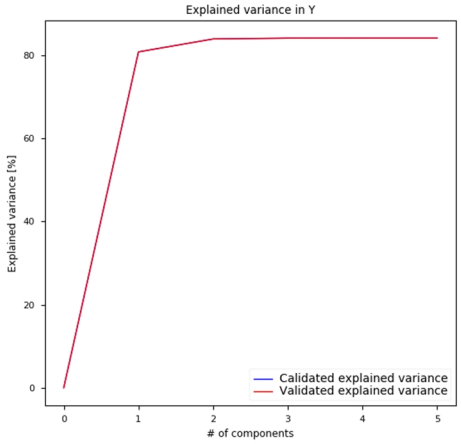
Beskrivelse	**Frekvens** [Hz]	**Periode** [Sec]	**Forskyvning** [kg]
Robot 3D-modell	0.4112	2.4319	129.20
Python 2D-modell	0.4143	2.4137	133.78

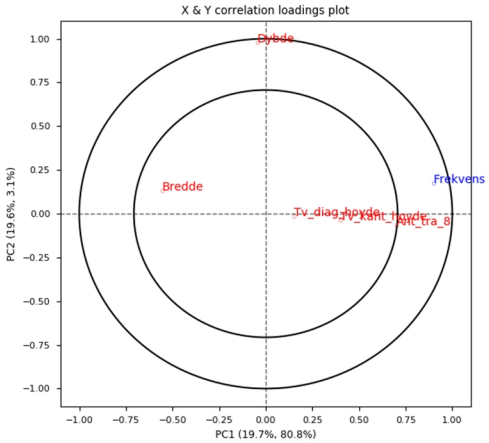
Vedlegg B – Dataanalyse

1. PLS1 analyse med Y = Frekvens

```
import hoggorm as ho
import hoggormplot as hopl
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('seaborn-paper')

#Plassering på resultatberegningene
hoit_trehus = 'C:/Users/nostaf/OneDrive - Sweco AB/Masteroppgave/03 -
Beregninger/Resultat_total_70874296e70983c81d34f6c4121b6bbaca757a3e.csv'



#Importerer resultatene i en pandas dataframe
beregning = pd.read_csv(hoit_trehus)
beregning = beregning[beregning.Tv_vannrett_hoyde == 450]
beregning = beregning[beregning.Tv_bredde == 625]
beregning = beregning[beregning.Ant_tra_8 != 2]
beregning = beregning[beregning.Ant_tra_8 != 3]
beregning = beregning[beregning.Ant_tra_8 != 4]
beregning = beregning[beregning.Ant_tra_8 != 6]
beregning = beregning[beregning.Ant_tra_8 != 7]
beregning = beregning[beregning.Ant_tra_8 != 8]
beregning = beregning[beregning.Ant_tra_8 != 10]
beregning = beregning[beregning.Ant_tra_8 != 11]
beregning = beregning[beregning.Ant_tra_8 != 12]
beregning = beregning[beregning.Ant_tra_8 != 14]
beregning = beregning[beregning.Ant_tra_8 != 15]
beregning = beregning[beregning.Ant_tra_8 != 16]
beregning = beregning[beregning.Ant_tra_8 != 18]
beregning = beregning[beregning.Ant_tra_8 != 19]
beregning = beregning[beregning.Ant_tra_8 != 20]
beregning = beregning[beregning.Ant_tra_8 != 22]
beregning = beregning[beregning.Ant_tra_8 != 23]

#splitter dataframen inn to lister som inneholder X og Y variabler
Liste_x = beregning.loc[:,['Dybde','Bredde','Ant_tra_8','Tv_diag_hoyde','Tv_kant_hoyde']]
Liste_y = beregning.loc[:,['Frekvens']]

#Gjor om til numpy liste
Liste_x_np = Liste_x.values
Liste_y_np = Liste_y.values

#Plotter størrelsene på X og Y
print ('Dimensjon på X og Y matrisene\n',np.shape(Liste_x_np))
print (np.shape(Liste_y_np),'\n')

#lager en nipalsPLS1 modell
model = ho.nipalsPLS1(arrX=Liste_x_np, Xstand = True,Ystand =
True,vecy=Liste_y_np,numComp=5)

#Henter ut loadingene
X_loadings = model.X_loadings()
Y_loadings = model.Y_loadings()

#Printer X og Y loads
print ('X ladings')
print (X_loadings)
```

```
print ('\nY loadings')
print (Y_loadings)

#Div plot
hopl.explVar(model)
hopl.correlationLoadings(model,comp=[1,2],XvarNames=['Dybde','Bredde','Ant_tra_8','Tv_dia
g_hoyde','Tv_kant_hoyde'],YvarNames=['Frekvens'])

#vektetloadingplot
hopl.loadingWeights(model,XvarNames=['Dybde','Bredde','Ant_tra_8','Tv_diag_hoyde','Tv_kan
t_hoyde'],YvarNames=['Frekvens'])

#Resultater ------------------------------------------------------------------------------

X ladings =
[[-4.63173675e-02  1.02239438e+00 -1.95027875e-01  6.36596567e-15  2.94447077e-15]
 [-5.69038758e-01  1.36975399e-01 -8.08039532e-01  1.96578536e-15  1.28398482e-15]
 [ 7.20723358e-01 -6.85404484e-02 -4.75066374e-01 -4.55883239e-01 -2.49453759e-01]
 [ 1.55915870e-01 -1.48275250e-02 -1.02772286e-01 -1.80593582e-01  9.68707287e-01]
 [ 4.09296804e-01 -3.89239313e-02 -2.69788882e-01  8.71551159e-01  7.02431874e-02]]

Y loadings =
[[ 9.23496749e-01  1.84169720e-01  4.33183385e-02  6.15243939e-17  7.08851275e-18]]

#PLOT
```

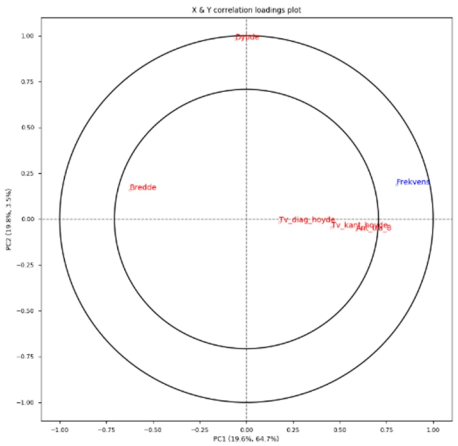


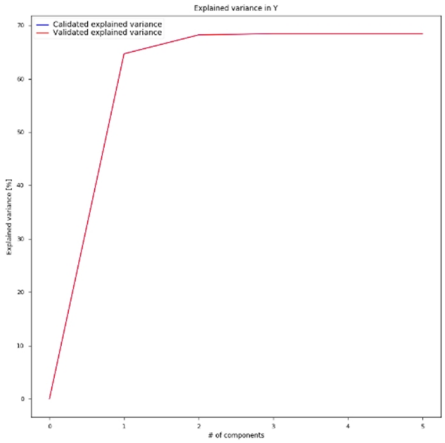


2. PLS1 analyse med Y = Frekvens og $X_i = \log(X_i)$

```
import hoggorm as ho
import hoggormplot as hopl
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-paper')

#Plassering på resultatberegningene
hoit_trehus = 'C:/Users/nostaf/OneDrive - Sweco AB/Masteroppgave/03 -
Beregninger/Resultat_total_70874296e70983c81d34f6c4121b6bbaca757a3e.csv'

#Importerer resultatene i en pandas dataframe
beregning = pd.read_csv(hoit_trehus)
beregning = beregning[beregning.Tv_vannrett_hoyde == 450]
beregning = beregning[beregning.Tv_bredde == 625]
beregning = beregning[beregning.Ant_tra_8 != 2]
beregning = beregning[beregning.Ant_tra_8 != 3]
beregning = beregning[beregning.Ant_tra_8 != 4]
beregning = beregning[beregning.Ant_tra_8 != 6]
beregning = beregning[beregning.Ant_tra_8 != 7]
beregning = beregning[beregning.Ant_tra_8 != 8]
beregning = beregning[beregning.Ant_tra_8 != 10]
beregning = beregning[beregning.Ant_tra_8 != 11]
beregning = beregning[beregning.Ant_tra_8 != 12]
beregning = beregning[beregning.Ant_tra_8 != 14]
beregning = beregning[beregning.Ant_tra_8 != 15]
beregning = beregning[beregning.Ant_tra_8 != 16]
beregning = beregning[beregning.Ant_tra_8 != 18]
beregning = beregning[beregning.Ant_tra_8 != 19]
beregning = beregning[beregning.Ant_tra_8 != 20]
beregning = beregning[beregning.Ant_tra_8 != 22]
beregning = beregning[beregning.Ant_tra_8 != 23]

#splitter dataframen inn to lister som inneholder X og Y variabler
Liste_x = beregning.loc[:,['Dybde','Bredde','Ant_tra_8','Tv_diag_hoyde','Tv_kant_hoyde']]
Liste_y = beregning.loc[:,['Frekvens']]

#Gjor om til numpy liste
Liste_x_np = Liste_x.values

#Transformerer X verdiene med log()
Liste_x_log = np.zeros((len(Liste_x), 5))

#
for j in range(len(Liste_x_np)):
    row = np.array([ np.log(Liste_x_np[j][0]),np.log(Liste_x_np[j][1]),np.log
(Liste_x_np[j][2]),np.log(Liste_x_np[j][3]),np.log(Liste_x_np[j][4])])
    Liste_x_log[j] = row
Liste_y_np = Liste_y.values

#Plotter størrelsene på X og Y
print ('Dimensjon på X og Y matrisene\n',np.shape(Liste_x_np))
print (np.shape(Liste_y_np),'\n')

#lager en nipalsPLS1 modell
model = ho.nipalsPLS1(arrX=Liste_x_log, Xstand = True,Ystand =
True,vecy=Liste_y_np,numComp=5)

#Henter ut loadingene
X_loadings = model.X_loadings()
```

```
Y_loadings = model.Y_loadings()

#Printer X og Y loads
print ('X ladings')
print (X_loadings)
print ('\nY loadings')
print (Y_loadings)

#printer forklart varians
print ('calibrated explained variance i Y')
print (model.Y_calExplVar())

#Div plot
hopl.explVar(model)
hopl.correlationLoadings(model,comp=[1,2],XvarNames=['Dybde','Bredde','Ant_tra_8','Tv_dia
g_hoyde','Tv_kant_hoyde'],YvarNames=['Frekvens'])

#vektetloadingplot
hopl.loadingWeights(model,XvarNames=['Dybde','Bredde','Ant_tra_8','Tv_diag_hoyde','Tv_kan
t_hoyde'],YvarNames=['Frekvens'])

#RESULTATER ------------------------------------------------------------------------------

X loadings =
[[-6.19478390e-02  1.01791998e+00 -1.88652549e-01  1.78910478e-13   2.67297479e-15]
 [-6.48361787e-01  1.63335823e-01 -7.53400494e-01  4.40869844e-14  -6.53671201e-16]
 [ 6.10877548e-01 -6.28318210e-02 -4.84632163e-01 -2.95134225e-01  -1.34101579e+00]
 [ 1.81535014e-01 -1.86717870e-02 -1.44018562e-01  9.55570293e-01  -4.42808204e-01]
 [ 4.73632157e-01 -4.87154438e-02 -3.75750225e-01  1.44023276e-02   1.89932550e+00]]

Y loadings =
[[ 8.35249133e-01  1.95456876e-01  4.62653587e-02  3.74159624e-15  -2.87195245e-17]]

calibrated explained variance i Y =
[64.67478676939152, 3.5385214157297895, 0.22082508064173112, 0.0, 0.0]
```

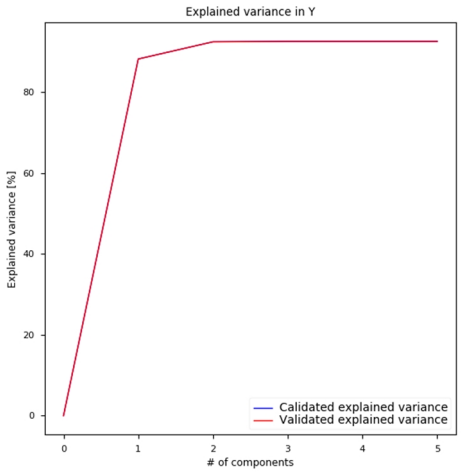


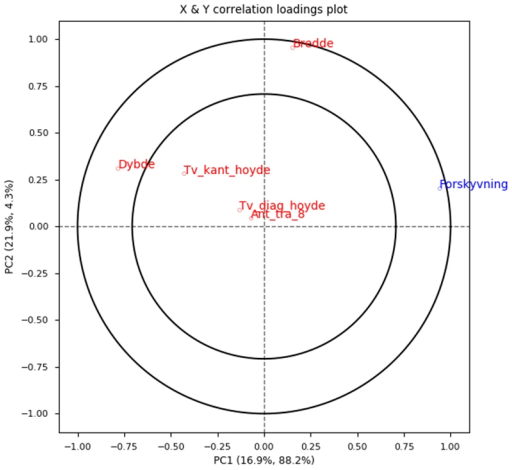


3. PLS1 analyse med Y = Forskyvning

```
import hoggorm as ho
import hoggormplot as hopl
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('seaborn-paper')

#Plassering på resultatberegningene
hoit_trehus = 'C:/Users/nostaf/OneDrive - Sweco AB/Masteroppgave/03 -
Beregninger/Resultat_total_70874296e70983c81d34f6c4121b6bbaca757a3e.csv'

#Importerer resultatene i en pandas dataframe
beregning = pd.read_csv(hoit_trehus)
beregning = beregning[beregning.Tv_vannrett_hoyde == 450]
beregning = beregning[beregning.Tv_bredde == 625]
beregning = beregning[beregning.Ant_tra_8 != 2]
beregning = beregning[beregning.Ant_tra_8 != 3]
beregning = beregning[beregning.Ant_tra_8 != 4]
beregning = beregning[beregning.Ant_tra_8 != 6]
beregning = beregning[beregning.Ant_tra_8 != 7]
beregning = beregning[beregning.Ant_tra_8 != 8]
beregning = beregning[beregning.Ant_tra_8 != 10]
beregning = beregning[beregning.Ant_tra_8 != 11]
beregning = beregning[beregning.Ant_tra_8 != 12]
beregning = beregning[beregning.Ant_tra_8 != 14]
beregning = beregning[beregning.Ant_tra_8 != 15]
beregning = beregning[beregning.Ant_tra_8 != 16]
beregning = beregning[beregning.Ant_tra_8 != 18]
beregning = beregning[beregning.Ant_tra_8 != 19]
beregning = beregning[beregning.Ant_tra_8 != 20]
beregning = beregning[beregning.Ant_tra_8 != 22]
beregning = beregning[beregning.Ant_tra_8 != 23]

#splitter dataframen inn to lister som inneholder X og Y variabler
Liste_x = beregning.loc[:,['Dybde','Bredde','Ant_tra_8','Tv_diag_hoyde','Tv_kant_hoyde']]
Liste_y = beregning.loc[:,['Forskyvning']]

#Gjor om til numpy liste
Liste_x_np = Liste_x.values
Liste_y_np = Liste_y.values

#Plotter størrelsene på X og Y
print ('Dimensjon på X og Y matrisene\n',np.shape(Liste_x_np))
print (np.shape(Liste_y_np),'\n')

#lager en nipalsPLS1 modell
model = ho.nipalsPLS1(arrX=Liste_x_np, Xstand = True,Ystand =
True,vecy=Liste_y_np,numComp=5)

#Henter ut loadingene
X_loadings = model.X_loadings()
Y_loadings = model.Y_loadings()

#Printer X og Y loads
print ('X ladings')
print (X_loadings)
print ('\nY loadings')
print (Y_loadings)
```

```
#printer forklart varians
print ('calibrated explained variance i Y')
print (model.Y_calExplVar())

#Div plot
hopl.explVar(model)
hopl.correlationLoadings(model,comp=[1,2],XvarNames=['Dybde','Bredde','Ant_tra_8','Tv_dia
g_hoyde','Tv_kant_hoyde'],YvarNames=['Forskyvning'])

#vektetloadingplot
hopl.loadingWeights(model,XvarNames=['Dybde','Bredde','Ant_tra_8','Tv_diag_hoyde','Tv_kan
t_hoyde'],YvarNames=['Forskyvning'])

#Resultater ------------------------------------------------------------------------------

X loadings =
[[-8.80764636e-01  3.01236518e-01 -5.21177631e-01  2.11952356e-15  -5.93458487e-16]
 [ 1.70715009e-01  9.21431932e-01 -2.50613613e-01  2.95765468e-15   6.39001729e-17]
 [-7.86085936e-02  4.44970533e-02  1.25691521e-01  2.52709320e-01   9.69443519e-01]
 [-1.51383835e-01  8.56920885e-02  2.42055780e-01  9.14655708e-01  -2.91824038e-01]
 [-4.80864739e-01  2.72197516e-01  7.68880572e-01 -3.29259353e-01  -6.66073994e-02]]


Y loadings =
[[1.05427653e+00 1.99168683e-01 3.07491537e-02 3.49128270e-17  1.41836178e-17]]

calibrated explained variance i Y = [88.16999158898152, 4.2537160046309594,
0.1000176017173402, 0.0, 0.0]
```

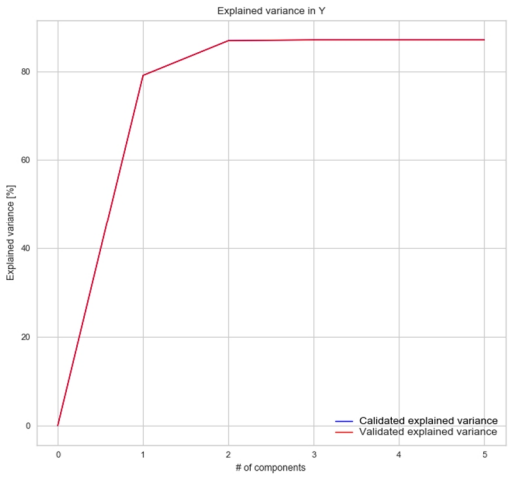


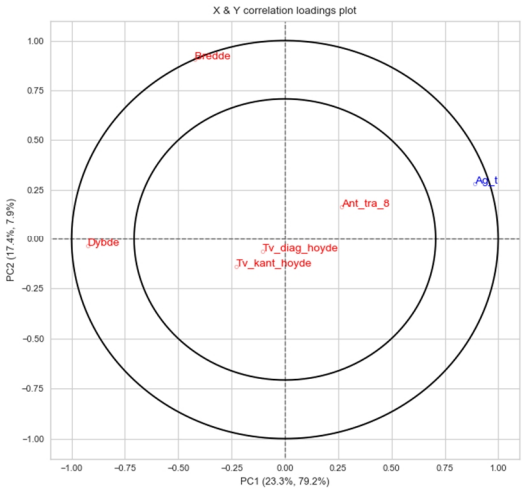


4. PLS1 Analyse med Y = Agt

```
import hoggorm as ho
import hoggormplot as hopl
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('seaborn-paper')

#Plassering på resultatberegningene
hoit_trehus = 'C:/Users/nostaf/OneDrive - Sweco AB/Masteroppgave/03
Beregninger/Resultat_total_70874296e70983c81d34f6c4121b6bbaca757a3e.csv'

#Importerer resultatene i en pandas dataframe
beregning = pd.read_csv(hoit_trehus)
beregning = beregning[beregning.Tv_vannrett_hoyde == 450]
beregning = beregning[beregning.Tv_bredde == 625]
beregning = beregning[beregning.Ant_tra_8 != 2]
beregning = beregning[beregning.Ant_tra_8 != 3]
beregning = beregning[beregning.Ant_tra_8 != 4]
beregning = beregning[beregning.Ant_tra_8 != 6]
beregning = beregning[beregning.Ant_tra_8 != 7]
beregning = beregning[beregning.Ant_tra_8 != 8]
beregning = beregning[beregning.Ant_tra_8 != 10]
beregning = beregning[beregning.Ant_tra_8 != 11]
beregning = beregning[beregning.Ant_tra_8 != 12]
beregning = beregning[beregning.Ant_tra_8 != 14]
beregning = beregning[beregning.Ant_tra_8 != 15]
beregning = beregning[beregning.Ant_tra_8 != 16]
beregning = beregning[beregning.Ant_tra_8 != 18]
beregning = beregning[beregning.Ant_tra_8 != 19]
beregning = beregning[beregning.Ant_tra_8 != 20]
beregning = beregning[beregning.Ant_tra_8 != 22]
beregning = beregning[beregning.Ant_tra_8 != 23]

#splitter dataframen inn to lister som inneholder X og Y variabler
Liste_x = beregning.loc[:,['Dybde','Bredde','Ant_tra_8','Tv_diag_hoyde','Tv_kant_hoyde']]
Liste_y = beregning.loc[:,[' Ag_t']]

#Gjor om til numpy liste
Liste_x_np = Liste_x.values
Liste_y_np = Liste_y.values

#Plotter størrelsene på X og Y
print ('Dimensjon på X og Y matrisene\n',np.shape(Liste_x_np))
print (np.shape(Liste_y_np),'\n')

#lager en nipalsPLS1 modell
model = ho.nipalsPLS1(arrX=Liste_x_np, Xstand = True,Ystand =
True,vecy=Liste_y_np,numComp=5)

#Henter ut loadingene
X_loadings = model.X_loadings()
Y_loadings = model.Y_loadings()

#Printer X og Y loads
print ('X ladings')
print (X_loadings)
print ('\nY loadings')
print (Y_loadings)

#Div plot
hopl.explVar(model)
hopl.correlationLoadings(model,comp=[1,2],XvarNames=['Dybde','Bredde','Ant_tra_8','Tv_dia
g_hoyde','Tv_kant_hoyde'],YvarNames=[' Ag_t'])
```

```
#vektetloadingplot
hopl.loadingWeights(model,XvarNames=['Dybde','Bredde','Ant_tra_8','Tv_diag_hoyde','Tv_kan
t_hoyde'],YvarNames=[' Ag_t'])

#Resultater ------------------------------------------------------------------------------

X loadings
[[-8.87905278e-01 -3.67748519e-02 -3.90912466e-01 -1.08357428e-14  -4.45781258e-15]
 [-4.06063626e-01  9.81450244e-01  5.52346258e-02  7.61949319e-14  -3.29396605e-14]
 [ 2.57170822e-01  1.77850864e-01 -6.68651234e-01 -2.98276914e-01   7.84615575e+00]
 [-1.00324237e-01 -6.93809355e-02  2.60845784e-01 -9.50793038e-01  -1.58280103e+00]
 [-2.20607888e-01 -1.52565144e-01  5.73586596e-01  8.46727051e-02   9.86635451e+00]]

Y loadings =
[[ 9.23496749e-01  1.84169720e-01  4.33183385e-02  6.15243939e-17  7.08851275e-18]]

#PLOT
```





Vedlegg C – Python-modul modeller.py

```
# ============================================================================
# ----------- IMPORT -----------------------------------------------------------
# ============================================================================

import os
path = ‘subfolder'
os.chdir(path)

#EGNE
import Parametrisering as para
import Elementsjekker as elem
import Statikk as st
import plot as plot
import importering as imp
import Dynamikk as dyn
import Parametrisering as para

#EKSTERNE
import os
from scipy import linalg
from pylab import array, shape, transpose, dot, copy, zeros
import numpy as np
import csv
from time import time
from copy import deepcopy
import sys
import multiprocessing
import hashlib
import base64

#=============================================================================
# BEREGNINGER ------------------------------------------------------------------
#=============================================================================

min_beregning = 1
max_beregning = 1520640

alle_resultater = []
alle_resultater_header =
['Beregningsnummer','beregnings_hash','høyde','Dybde','Bredde','Ant_trä-8',          \
       'Ant_betong' ,'Tv_bredde', 'Tv_diag_høyde','Tv_kant_høyde', 'Tv_midt_høyde',  \
       'Tv_vannrett_høyde','Forskyvning','Forskyvning_krav','Forskyvning_utn',       \
       'Reaksjon_venstre','Reaksjon_høyre','Kraft_diag_max_trykk',                   \
       'Kraft_diag_max_skjær', 'Kraft_diag_max_strekk','Kraft_diag_min_skjær',       \
       'kraft_diag_moment','Kraft_knat_max_trykk','Kraft_kant_max_skjær',            \
       'Kraft_kant_max_strekk','Kraft_kant_min_skjær','kraft_kant_moment',           \
       'Frekvens','Periode','Ag_t','Ag_krav_kontor','Ag_krav_bolig',                 \
       'Ag_utn_kontor','Ag_utn_bolig','ekv_masse','Cs','Cd','CsCd','Diag_T',         \
       'Diag_S','Midt_T','Kant_T','Kant_S','Van_Tre','Van_Bet','Diag_T_Brann',       \
       'Diag_S_Brann','Midt_T_Brann','Kant_T_Brann','Kant_S_Brann',                  \
       'Van_Tre_brann','Van_Bet_Brann']

alle_resultater.append(alle_resultater_header)

#=============================================================================
# Variabler ........-----------------------------------------------------------
#=============================================================================

#Tversnittsparametere

bredde = [625,750,875,1000,1125]
hoide_kant = [1485,1710,1935,2160]
hoide_diagonal = [990,1215,1440,1665]
hoide_vannrett = [450,495,540,585]
```

```
#Generere tversnitt i en global tverrsnitsmatrise med dict inne i matrise

tverrsnitt = elem.tverrsnitt_generering(bredde,hoide_kant,hoide_diagonal,            \
       hoide_vannrett)

#material = [f_m_g_k, f_t_0,f_c_0,f_v]

Material_k = [30, 19.5, 24.5,3.5]
Material_d = [28.69, 18.65, 23.43,3.35]

#Input for beresystem

hoid = ["Elementer-25.csv"]

#fotavtrykk parametere

dybde =  [14000,15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000]#mm

bredde = [14000,15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,   \
             26000,27000,28000,29000,30000,31000,32000,33000,34000,35000]#mm

#densiteten til tra-8 og betongdekkene.
masse_i_kraft = [2500,8000]

#=============================================================================
# INPUTGENERATOR -----------------------------------------------------------
#=============================================================================

innput = []
start_tid = time()
teller_1 = 0
for i in dybde:
    for j in bredde:
        if j < i:
            ost = 1
        else:
            teller_1 = teller_1 + 1
teller = 1
prosent = 0

#Antall beregninger er kalkulert her
ant_beregninger = 1
for i in hoid:
    etg = int(i[-6:-4])
    ant_beregninger = ant_beregninger + teller_1*(etg-1)*len(tverrsnitt)
gj_steg = deepcopy(ant_beregninger)

print ('')
print ('# ================================================================ ')
print ('# ===== Totalt antall beregninger:{} Stk ========'.format(ant_beregninger))
print ('# ================================================================\n')

print ('# Antall beregninger i denne økten er: {}'.format(max_beregning-min_beregning+1))
print ('# Utvalg er mellom {} og {}\n'.format(min_beregning,max_beregning))

#Itereringsprosessen for inndata
for tv in tverrsnitt:
    for y in hoid:
        etasjer = int(y[-6:-4])
        masse_fordeling = []

        for i in range (etasjer+1):
            type_masse = [0]

            for j in range(etasjer-i):
                type_masse.extend([masse_i_kraft[0]])

            for z in range(i):
```

```
                type_masse.extend([masse_i_kraft[1]])

            masse_fordeling.append(type_masse)

        for o in masse_fordeling[1:-1]:
            for i in dybde:
                for x in bredde:
                    if x < i:
                        ost = 1
                    else:
                        Id_string = str([y,tv,i,x,masse_i_kraft,o,teller])
                        Id = hashlib.sha1(Id_string.encode('utf-8')).hexdigest()
                        innput.append([y,tv,i,x,masse_i_kraft,o,teller,Id])
                        teller = teller +1

#krypterer sha1 krypteringen til inndata pga inputkontroll.
string_adder = ''

for i in range(int(len(innput)/1000)):
    string_adder = string_adder + innput[i*1000][7] has_kont = hashlib.sha1(         \
             string_adder.encode('utf-8')).hexdigest()

inndata_sti_navn = '{}/inndata_{}_{}_{}.csv'.format( path,min_beregning,             \
       max_beregning, has_kont)

print ('Inndata brukt i denne økten er lagret på stien {}\n'.format                  \
       (inndata_sti_navn))

del_innput = innput[min_beregning-1:max_beregning]

#Genererer en csv fil av alle inputverdier i denne okten

imp.writeCsvFile(r'{}'.format(inndata_sti_navn), del_innput)

#============================================================================
# OUTPUT -----------------------------------------------------------------
#============================================================================

start_time = time()
output_stream = sys.stdout
teller_2 = 0
ant_bereg = max_beregning-min_beregning+1
for steggg in del_innput:
    resu = para.parametrisk(steggg)
    alle_resultater.append(resu)

    teller_2 = teller_2 + 1
    step_time = time() - start_time
    gjen_tid = step_time/teller_2
    gj_tid = gjen_tid*ant_bereg - step_time
    prosent = teller_2/ant_bereg*100

    print_pros = 'Antall prosent:{:.2f}% | Tid gått:{:.2f} Sek | gjennstående tid:{:.2f}
             Min, {:.2f} Sek | G.snitt tid:{:.4f} | Beregninger:{}'.format(          \
                   prosent,step_time,gj_tid/60, gj_tid, gjen_tid,teller_2)          \
                         sys.stdout.write('\r'+print_pros)

utdata_sti_navn = '{}/utdata_{}_{}_{}.csv'.format(path,min_beregning,                \
       max_beregning,has_kont)

#Genererer en CSV fil av alle resultater i denne okten

imp.writeCsvFile(r'{}'.format(utdata_sti_navn), alle_resultater)

print ('\n\nUtdata generert i denne økten er lagret på stien {}\n'.                  \
       format(utdata_sti_navn))
```

```
#=============================================================================
# TIDSBRUK -----------------------------------------------------------------
#=============================================================================

end_time = time()
time_taken = end_time - start_time

print ('\n')
print ('# ========================================================================')
print ('# ===== Kalkulasjonstid:: {:.2f} Min, {:.2f} sek ====='.format(              \
       time_taken/60, time_taken))

print ('# ========================================================================\n')
```

Vedlegg D – Python-modul Parametrisering.py

```
#=============================================================================
# ----------- IMPORT ------------------------------------------------------
#=============================================================================

#EGNE
import Parametrisering as para
import Statikk as st
import plot as plot
import importering as imp
import Elementsjekker as elem
import Dynamikk as dyn

#EKSTERNE
from scipy import linalg
from pylab import array, shape, transpose, dot, copy, zeros
import numpy as np
import csv
from time import time
from copy import deepcopy

def parametrisk( plassering_elementer, tv, d, b, masse_i_kraft,    masse_fordeling ):

	tverrsnitt = tv
	start_time = time()

	#=============================================================================
	#====== VIND ================================================================
	#=============================================================================

	etasjer = int(plassering_elementer[-6:-4])
	#parametere fra EUROKODENE
	C_pe_D = [0.8]
	C_pe_E = [-0.7,-0.5]
	v_b0 = 22

	#terrengkategori
	# Type, z_0, z_min, z_max
	terreng = ['II',0.05,2,200]

	#C_parametere
	# C_para = [C_dir, C_season]
	C_para = [1,1]
	v_b = v_b0*C_para[0]*C_para[1]
	#Terrengformfaktor og turbulensfaktor
	TERRENGFAKTOR = [1,1]

	#MASSE-FORDELING ============================================================
	#masse representert i m*g i N/m2

	masse_etasje = zeros(etasjer+1)
	masse_innvendig_i_kraft = 1200     #N/m2
	masse_etg_treverk = 0
	masse_etg_betong = 0


	for i in masse_fordeling:
		if i == masse_i_kraft[0]:
			masse_etg_treverk = masse_etg_treverk + 1
		elif i == masse_i_kraft[1]:
			masse_etg_betong = masse_etg_betong + 1

	# NYTTELASTFORDELING =========================================================
```

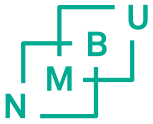
```
nytte_fordeling = [ 0, 3000 , 3000, 3000, 3000, 3000, 3000, 3000,    3000, 3000,
3000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 3000, 3000, 3000,
3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000, 3000,
3000, 3000, 3000, 3000 ]

#=============================================================================
# TOMME PARAMETERE ---------------------------------------------------------
#=============================================================================

kant_node_venstre = []
kant_node_hoire = []
midt_node = []
hoide = [0]
etasje_node = []

#=============================================================================
# RESULTERENDE PARAMETERE --------------------------------------------------
#=============================================================================

slutt_resultat_para = []
slutt_resultat_verdier = []

Elementer = imp.import_elementliste( plassering_elementer, d,
tverrsnitt)


friheter = []

for rad in Elementer:
       friheter.extend(rad[7:13])

frihetsgrader = int(friheter[-1]+1)
F = zeros(frihetsgrader)
F_brann = zeros(frihetsgrader)
F_test = zeros(frihetsgrader)
G = zeros((frihetsgrader,frihetsgrader))
G_brann = zeros((frihetsgrader,frihetsgrader))
G_newton_meter = zeros((frihetsgrader,frihetsgrader))
master_mass = zeros((frihetsgrader,frihetsgrader))

for i in range(len(master_mass)):
       master_mass[i][i] = 1




#=============================================================================
# GEOMETRI -----------------------------------------------------------------
#=============================================================================

# Etasje-noder i venstrkant av bygget
for i in Elementer:
       if i[3] == int(0):
              kant_node_venstre.extend([i[1]])
       elif i[5] == int(0):
              kant_node_venstre.extend([i[2]])

kant_node_venstre = list(set(kant_node_venstre))
kant_node_venstre.sort()


# Etasje-noder i høyrekant av bygget samt etasjehøyder
for i in Elementer:
       if i[3] == d:
              kant_node_hoire.extend([i[1]])
              hoide.extend([i[6]])
       elif i[5] == d:
              kant_node_hoire.extend([i[2]])
```

```
kant_node_hoire = list(set(kant_node_hoire))
kant_node_hoire.sort()

hoide = list(set(hoide))
hoide.sort()

# Etasje-noder i midten av bygget

for i in Elementer:
       if i[3] == (d/2) and i[4] in hoide:
              midt_node.extend([i[1]])
       elif i[5] == (d/2) and i[6] in hoide:
              midt_node.extend([i[2]])

midt_node = list(set(midt_node))
midt_node.sort()

#=============================================================================
# ELEMENTMODIFIKSJON ---------------------------------------------------------
#=============================================================================

for i in range(len(Elementer)):
       for j in range(len(hoide)):
              if Elementer[i][13] == 'vannrett' and Elementer[i][4] ==         \
                  hoide[j] and masse_fordeling[j] == masse_i_kraft[1]:
                     Elementer[i][18] = float(tverrsnitt['h_5'])
                     Elementer[i][17] = float(tverrsnitt['b_5'])
                     #print ('Element',i+1,'Høyde =',tverrsnitt['h_5'])

Elementer_newton_meter = imp.import_elementliste_newton_meter(Elementer)
Elementer_brann = deepcopy(Elementer)

for i in range(len(Elementer_brann)):
    if Elementer_brann[i][13] == 'vannrett':
        Elementer_brann[i][17] = Elementer_brann[i][17] - 206
        Elementer_brann[i][18] = Elementer_brann[i][18] - 103
    else:
        Elementer_brann[i][17] = Elementer_brann[i][17] - 206
        Elementer_brann[i][18] = Elementer_brann[i][18] - 206


# Finner noder som ligger i etasjene.
for j in range(etasjer+1):
    etasje_nr = []
    for i in Elementer:
        if i[4] in hoide and i[4] == hoide[j]:
            etasje_nr.extend([i[1]])
        elif i[6] in hoide and i[6] == hoide[j]:
            etasje_nr.extend([i[2]])
        etasje_nr = list(set(etasje_nr))
        etasje_nr.sort()
    etasje_node.append(etasje_nr)

#Massefordelinger ============================================================

#print (kant_node_venstre)
idof = array ([0,1,3,4,6,7])   #posisjon på kjent forflyttning
didof = array([0,0,0,0,0,0])   #forskyvning på kjent posisjon

#=============================================================================
# PARAMETERE -----------------------------------------------------------------
#=============================================================================

#print (f_il[80])
#print (f_ig[80])
#plot.print_node_f(D_node)
#plot.geoemtri_print(Elementer,d)
```

```
#plot.print_element_kraft(f_il)

#=============================================================================
# DYNAMISK-BEREGNING ---------------------------------------------------------
#=============================================================================

#STIVHETSMATRISE FOR DYNAMSKE BEREGNINGER MED BENEVNINGEN NEWTON OG METER ======

G_newton_meter,K_i_newton_meter = st.Elementer_parametrisk_stivhetsmatrise(    \
       Elementer_newton_meter ,idof, didof, F, G_newton_meter, d)

#KONSTRUERER EN MASSE MATRISE ================================================

master_mass, masse_etasje =   dyn.add_mass_fra_dekke( master_mass,hoide,       \
       kant_node_venstre,kant_node_hoire,midt_node ,masse_fordeling, d,        \
              b, Elementer, masse_etasje)

master_mass, masse_etasje = dyn.add_mass_fra_innvendig( master_mass,           \
       hoide,  kant_node_venstre, kant_node_hoire, midt_node,                  \
              masse_innvendig_i_kraft, d, b, Elementer, masse_etasje)

master_mass, masse_etasje = dyn.add_mass_fra_fasade(master_mass,hoide,         \
       kant_node_venstre,kant_node_hoire,midt_node,masse_fordeling,d,          \
              b,Elementer,masse_etasje)

master_mass, masse_etasje = dyn.add_mass_fra_elementer(Elementer,              \
       master_mass,masse_etasje,hoide )

master_mass, masse_etasje = dyn.add_mass_fra_nyttelast(master_mass,            \
       hoide, kant_node_venstre,kant_node_hoire,midt_node,                     \
              nytte_fordeling,d,b, Elementer, masse_etasje,0.3)


# EGENFREKVENS OG PERIODE =====================================================

f_1,p_1,mode_shape_etg_venstre,mode_shape_etg_midt, mode_shape_etg_hoire,      \
       mode_shape_etg_normal_venstre, mode_shape_etg_normal_midt,              \
                     mode_shape_etg_normal_hoire,modal_mass =                  \
                            dyn.egenfrekvens(master_mass, G_newton_meter,      \
                                   kant_node_venstre, kant_node_hoire, midt_node)

ekv_masse = dyn.ekvivalent_masse(mode_shape_etg_venstre, hoide,b,modal_mass)
resultat_dyn = [f_1,p_1,ekv_masse]
Ag_t = elem.vindinduserte_akselerasjoner(hoide,b,d,f_1,ekv_masse)
Ag_krav_kontor = bruks_krav_akselerasjon_kontor(f_1)
Ag_krav_bolig = bruks_krav_akselerasjon_bolig(f_1)

#=============================================================================
# STATISK BEREGNING ----------------------------------------------------------
#=============================================================================

# vind_kraft kalkulerer vindkraften i N/m2

z = elem.vind_hoide(hoide,b)

Vind_d,Vind_e = Vind_kraft(v_b0,terreng,TERRENGFAKTOR,z,C_pe_D,C_pe_E,         \
       hoide,d,C_para)

Cs,Cd,CsCd = Vindkraftfaktor(hoide[-1],b,d,f_1,C_pe_D,C_pe_E,v_b0)

#Legger til vindkraft og egenvekt fra dekke i F matrisen.

F = elem.add_vind_kraft(kant_node_venstre,kant_node_hoire, hoide, z, F,        \
       1.50,Vind_d,Vind_e,b,CsCd)

F = st.add_egenvekt_dekke_til_kraft(Elementer,hoide,F,masse_fordeling,1.0)
F = st.add_egenvekt_fasade_til_kraft(Elementer,F,1.0,hoide)
F = st.add_egenvekt_elementer_til_kraft(Elementer,F,1.0)
```

```
F = st.add_nyttelast_dekke_til_kraft(Elementer,hoide,nytte_fordeling,F,1.05)

#Legger til vindkraft og egenvekt fra dekke i F matrisen i brann situasjon.

F_brann = add_vind_kraft(kant_node_venstre,kant_node_hoire,hoide,z,           \
      F_brann,0.2,Vind_d,Vind_e,b,CsCd)

F_brann = st.add_egenvekt_dekke_til_kraft(Elementer,hoide,F_brann,            \
      masse_fordeling, 1.00)

F_brann = st.add_egenvekt_fasade_til_kraft(Elementer,F_brann,1.00,hoide)
F_brann = st.add_egenvekt_elementer_til_kraft(Elementer,F_brann,1.00)

F_brann = st.add_nyttelast_dekke_til_kraft(Elementer,hoide,                   \
      nytte_fordeling,F_brann,0.3)

#Løser de ukjente forskyvningene med hensyn fra F matrisen.

D_node,K_i,G = st.Elementer_parametrisk(Elementer,idof,didof,F,G,d)

D_node_brann,K_i_brann,G_brann = st.Elementer_parametrisk(                    \
      Elementer_brann, idof,didof,F_brann,G_brann,d)

#print (G_newtong_meter)


#Løser de interne kraftene i elementene, i lokale og globale kordinater.

f_il,f_ig = st.Interne_krefter(K_i,Elementer,D_node)

f_il_brann,f_ig_brann = st.Interne_krefter(K_i_brann,Elementer_brann,         \
      D_node_brann)

#=============================================================================
# ELEMENTKONTROLL ----------------------------------------------------------
#=============================================================================

uls_element_sjekk = element_sjekk (Elementer,f_il, masse_fordeling,           \
      nytte_fordeling,   hoide)

bra_element_sjekk = element_sjekk_brann(Elementer_brann, f_il_brann,          \
      masse_fordeling, nytte_fordeling,hoide)

#=============================================================================
# TESTER -------------------------------------------------------------------
#=============================================================================

Element_max,Element_max_label = max_element_sjekk(f_il,uls_element_sjekk,     \
      bra_element_sjekk, Elementer, hoide, masse_fordeling, masse_i_kraft)

Element_brann_max,Element_brann_max_label = max_element_sjekk(f_il_brann,     \
      bra_element_sjekk, bra_element_sjekk, Elementer, hoide,masse_fordeling  \
            masse_i_kraft)

#=============================================================================
# RESULTATSORTERING --------------------------------------------------------
#=============================================================================

Element_max_float = []
Element_brann_max_float = []


for i in range(len(Element_max)):
      Element_max_float.append(float(Element_max[i]))
      Element_brann_max_float.append(float(Element_brann_max[i]))

slutt_resultat_verdier.extend([hoide[-1],d,b])
slutt_resultat_verdier.extend([masse_etg_treverk,masse_etg_betong])
```

```
slutt_resultat_verdier.extend([D_node[-3][1],hoide[-1]/400,                    \
      D_node[-3][1]/(hoide[-1]/400)])
slutt_resultat_verdier.extend([f_1,p_1,Ag_t,Ag_krav_kontor,Ag_krav_bolig,      \
      Ag_t/Ag_krav_kontor,Ag_t/Ag_krav_bolig,ekv_masse,Cs,Cd,CsCd])

slutt_resultat_verdier.extend([tverrsnitt['b_1'],tverrsnitt['h_1'],           \
      tverrsnitt['h_2'],tverrsnitt['h_3'],tverrsnitt['h_4']])

slutt_resultat_verdier.extend(Element_max_float)
slutt_resultat_verdier.extend(Element_brann_max_float)

#Tid
end_time = time()
time_taken = end_time - start_time
#print ("Kalkulasjonstid:","%.4f"%time_taken, 'Sec')

#output
return slutt_resultat_verdier, Elementer
```



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway