NTNU
Det skapende universitet

Videreføring av lineær statisk analyse i programmet fap2D

Frans Erstad

Master i ingeniørvitenskap og IKT
Innlevert: juni 2013
Hovedveileder: Kjell Magne Mathisen, KT
Medveileder: Kolbein Bell, KT

Norges teknisk-naturvitenskapelige universitet
Institutt for konstruksjonsteknikk

Institutt for konstruksjonsteknikk
Fakultet for ingeniørvitenskap og teknologi

NTNU- Norges teknisk- naturvitenskapelige universitet

TILGJENGELIGHET

Åpen

MASTEROPPGAVE 2013

FAGOMRÅDE:	DATO:	ANTALL SIDER:
Konstruksjonsteknikk/ programutvikling	10.06.2013	

TITTEL:

Videreføring av lineær statisk analyse i programmet fap2D

Further development of linear static analyses in program **fap2D**

UTFØRT AV:

Frans Erstad

SAMMENDRAG:

Denne rapporten beskriver utviklingen av analysen «sanntids» beregninger i **fap2D**, et program for statisk og dynamisk analyse av 2D rammekonstruksjoner. I tillegg omhandler den også arbeidet med forbedringer og stabilisering av programmet.

Rapporten er skrevet med bakgrunn i en tidligere prosjektoppgave fra forrige semester som betrakter muligheter for implementasjon av «sanntids» beregninger. Innholdsmessig tar rapporten for seg arbeidsprosess, utvikling, funksjonalitet og bruk av implementasjonene som har blitt laget i løpet av masteroppgaven.

Det har også blitt testet at implementasjonene fungerer som påtenkt, og forslag til videre arbeid er presentert for fremtidige utviklere på **fap2D**. I denne masteroppgaven er mesteparten av arbeidet gjort i kildekoden til **fap2D**, en logg er vedlagt som beskriver arbeidet med masteroppgaven i detalj.

FAGLÆRER: Kjell Magne Mathisen

VEILEDER(E): Kolbein Bell

UTFØRT VED: NTNU

Summary

This report describes the development of real time analysis in **fap2D**, a program for static and dynamic analysis of 2D frame type structures. The report also concerns the task of improving and stabilizing the program in addition.

The report is based on a previous project from last semester concerning the possibilities of an implementation of real time analysis. In content, the report covers the working process, development, functionality and user instructions for the implementations that have been made during this thesis.

The implementations have also been tested to confirm that they are working properly, and suggestions for future work are presented for future developers of **fap2D**. Most of the work in this thesis is done in the source code, a log is attached describing the work in detail.

Forord

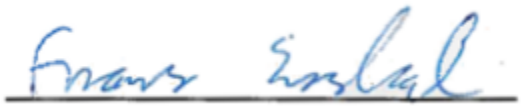
Programmet **fap2D** har vært under utvikling som masteroppgaver siden 2006 med forskjellige studenter ansvarlige for utvidelser av programmet. Utvidelsene er forbedringer og videreføringer av tidligere arbeid.

På samme måte er denne masteroppgaven en utvidelse av **fap2D** siden den bygger videre på allerede eksisterende struktur og metoder. I tillegg er masteroppgaven en videreføring av min egen prosjektoppgave fra høsten 2012, hvor gjennomførbarheten av denne oppgaven ble vurdert og gjennomførelsen planlagt.

Denne masteroppgaven fungerer som dokumentasjon for arbeidet som er gjort programmeringsmessig med **fap2D**, da mesteparten av tiden dette semesteret er tillagt programmeringen av mine oppgaver. Fremgangen fra dag til dag med hensyn på programmering er videre beskrevet i en logg som ble oppdatert kontinuerlig.

To andre studenter har jobbet med **fap2D** samtidig som meg i løpet av dette semesteret. Disse er Erik Aasmundrud og Kristian Pedersen som respektivt har jobbet med implementering av jordskjelvsberegninger og forbedring/kvalitetssikring av programmet. Sammen har vi delt kildekode på et fellesområde under utviklingen, samarbeidet og kommet med innspill på hverandres arbeid.

Jeg vil gjerne takke disse to medstudentene for samarbeidet og Professor emeritus Kolbein Bell ved NTNU for all hjelp og tilrettelegging med masteroppgaven og for oppfølging gjennom ukentlige møter.



Frans Erstad – Trondheim, vår 2013

Institutt for konstruksjonsteknikk
FAKULTET FOR INGENIØRVITENSKAP OG TEKNOLOGI
NTNU – Norges teknisk-naturvitenskapelige universitet

MASTEROPPGAVE 2013

for

Frans Erstad

Videreføring av lineære statiske beregninger i programmet **fap2D**

Further development of linear static analyses in program **fap2D**

Basert på utført prosjektoppgave går oppgaven ut på å videreføre utviklingen av **fap2D**, et program for statiske og dynamiske beregninger av plane rammekonstruksjoner som ble påbegynt allerede i 2006. Frem til 2012 har 8 studenter deltatt i utviklingen av programmet, som prosjekt- og masteroppgaver. I inneværende semester er dette en av tre oppgaver knyttet til utviklingen av **fap2D**.

Denne oppgaven skal primært implementere løsninger knyttet til lineære statiske beregninger av konstruksjoner som påkjennes av bevegelige laster på klart definerte «kjørebaner», som «sanntids» (*real time*) visning av responsdiagrammer; mulighet for stopp og start underveis vurderes. Gradvis fremføring av «dast-tog» bør også vurderes.

I den grad tiden tillater skal det også vurderes å implementere «omhyllingskurver» for snittkraftsdiagrammene når det er spesifisert to eller flere lastkombinasjoner.

Oppgaven forutsettes koordinert med oppgavene til Aasmundrud og Pedersen, og som for deres oppgaver har også denne oppgaven et ansvar for å implementere robuste løsninger, eventuelt modifisere allerede implementerte løsninger med tanke på å fremskaffe et effektivt, robust og stabilt program tuftet på enhetlige løsninger.

Besvarelsen er først og fremst selve programmet, men det skal også utarbeides en kortfattet rapport som beskriver problemstillinger, utfordringer og valgte løsninger. Det bør også samt sannsynliggjøres at implementeringene fungerer korrekt. For å kunne bedømme innsatsen er det viktig at rapporten omfatter en kronologisk "logg" som angir hva som er gjort og når; loggen plasseres i et vedlegg. Kandidaten bør også ha i mente at en av rapportens viktigste oppgaver er å sette andre i stand til å gå inn i koden og eventuelt modifisere/videreføre den.

Oppgaven lar seg vanskelig innordne den vanlige "malen", men så langt råd bør besvarelsen organiseres i henhold til gjeldende retningslinjer. Selv om rapporten blir relativt kortfattet stilles det de samme formelle krav til den som for masteroppgaver generelt.

Hovedveileder: Kjell Magne Mathisen
Medveileder: Kolbein Bell

Besvarelsen skal leveres til Institutt for konstruksjonsteknikk innen 10. juni 2013.

NTNU, 18. januar, 2013

Kjell Magne Mathisen

Innholdsfortegnelse

1. Innledning

fap2D er et Windows-basert program for statisk og dynamisk analyse av 2D konstruksjonsrammer som er utviklet ved institutt for konstruksjonsteknikk ved NTNU. Virkemåten og egenskapene til programmet er nærmere spesifisert i brukermanualen[2] og i prosjektoppgaven[1] som masteroppgaven bygger på.

I sammenheng med prosjektoppgaven ble det skrevet kode i programmet for å teste hvorvidt implementasjonen av «sanntids» beregninger er mulig å gjennomføre. Koden har senere blitt fjernet fra **fap2D**s kildekode da den ikke hadde noen praktisk verdi utover testingen. Det ble konkludert med at slike beregninger lot seg gjøre å implementere på grunn av rimelige resultater for beregningshastighet.

Motivasjonen for å lage en slik analyse er å kunne kjøre en serie med statiske beregninger for å se endringen i respons over hele modellen. I den sammenheng er lasttog en passende konnotasjon siden det er et av de mer iøynefallende bruksområdene for analysen.

Målet med denne masteroppgaven var å implementere «sanntids» beregninger basert på konklusjonene og planen som ble presentert i prosjektoppgaven. Mesteparten av tiden dette semesteret har gått med på å programmere selve implementasjonen av «sanntids» beregninger, mens selve rapporten inneholder den daglige loggen, begrunnelser for programmeringsvalgene som er gjort underveis og annen nyttig informasjon for fremtidige brukere av programmet.

I tillegg blir bruken av analysene som har blitt implementert forklart, sammen med en oversikt over funksjonaliteten til disse. Analysene har også blitt testet for å sikre at de fungerer som påtenkt ved slutten av denne masteroppgaven. Selve koden er inkludert i tillegg til rapporten.

2. Terminologi

Kapittelet gir en oversikt over valgene som er gjort i forhold til formatering og notasjon og andre standarder i masteroppgaven.

2.1. Notasjon

Tabell 1 viser hvilke konvensjoner som er brukt for ulike typer tekst gjennom masteroppgaven. Disse formateringskonvensjonene er de samme som i prosjektoppgaven[1] som denne masteroppgaven bygger på.

Tabell 1: Tekstkonvensjoner

Formatering	Beskrivelse
Normal tekst	Dette er standard format for tekst i prosjektoppgaven
Kursiv tekst	Brukes på uttrykk på engelsk som ikke har god norsk oversettelse
Fet tekst	Matrise- og vektorsymboler
[*Kursiv tekst mellom klammer*]	Metoder og objekter i **fap2D**

Foruten disse konvensjonene blir navnet på programmet, **fap2D**, og dets beregningskjerne, **Frame2D**, skrevet konsekvent med fet skrift gjennom hele masteroppgaven. Kapittel- og avsnittsoverskrifter, tabeller og formler samt bilde-, tabell- og figurtekst er skrevet i Microsoft Word 2010-stil. Referanser til referanselisten er gitt med opphøyd tekst ved siden av det som refereres. Referanser til tabeller og figurer skrives med stor forbokstav, da dette er standarden i Word 2010-stil.

Noen uttrykk på engelsk har også blitt oversatt til norsk i denne rapporten. En oversikt over alle uttrykkene med tilhørende norsk oversettelse finnes i Tabell 2.

Tabell 2: Oversikt over uttrykk på engelsk med norsk oversettelse

Uttrykk på engelsk	Uttrykk på norsk
Real time analysis	«Sanntids» beregninger
Xtrm analysis	Analyse for ekstreme responser
Modeling panel	Modellvinduet
Toolbox panel	Verktøylinjen
Meshing	Elementinndeling
Load path	Lastbane
Load case	Lasttilfelle

2.2. Kodekonvensjoner

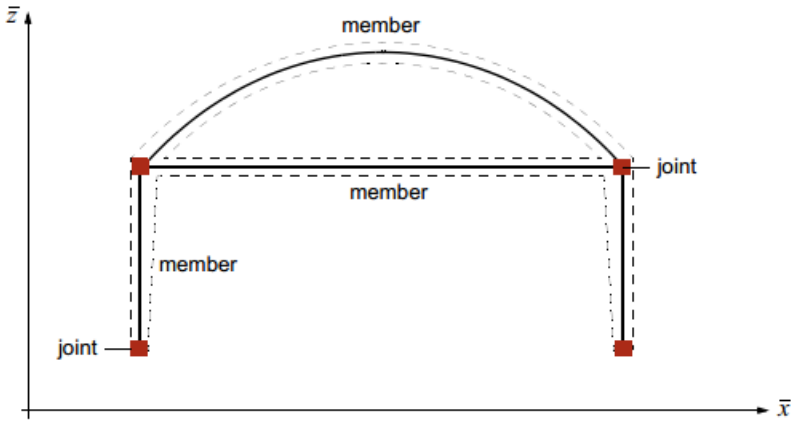
Alle som har vært medvirkende til å arbeide med **fap2D** dette semesteret har måttet forholde seg til samme sett med kodekonvensjoner, som er nærmere beskrevet i vedlegg D.

2.3. Definisjoner

2.3.1. Modellen

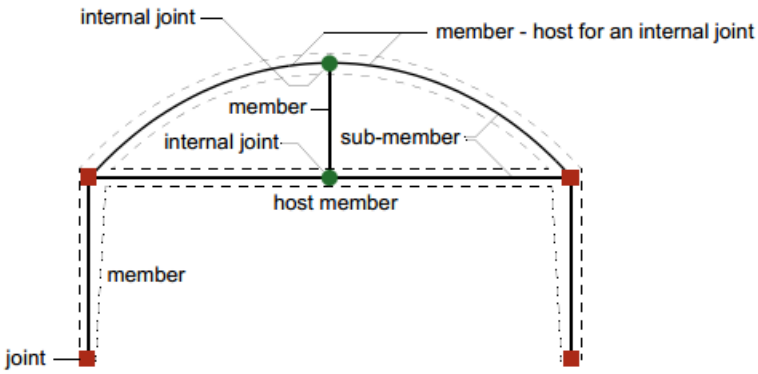
Definisjoner og forklaringer til uttrykk som eksisterer i sammenheng med en modell i **fap2D** forklares her.

I masteroppgaven refereres det til *members* og *joints*. I en konstruksjonsmodell som modelleres i **fap2D** er disse definert som konstruksjonselementer i konstruksjonsmodellen og kan illustreres med Figur 1.



Figur 1: Definisjon av members og joints

Figurene i dette delkapittelet er hentet fra brukermanualen til **fap2D**[2]. *Joints* kan også plasseres inne på *members*, disse er da definert som *internal joints*. *Members* som deles av *internal joints* blir definert som *host member* og nye *members* som oppstår på grunn av delingen er definert som *submembers*, jamfør Figur 2.



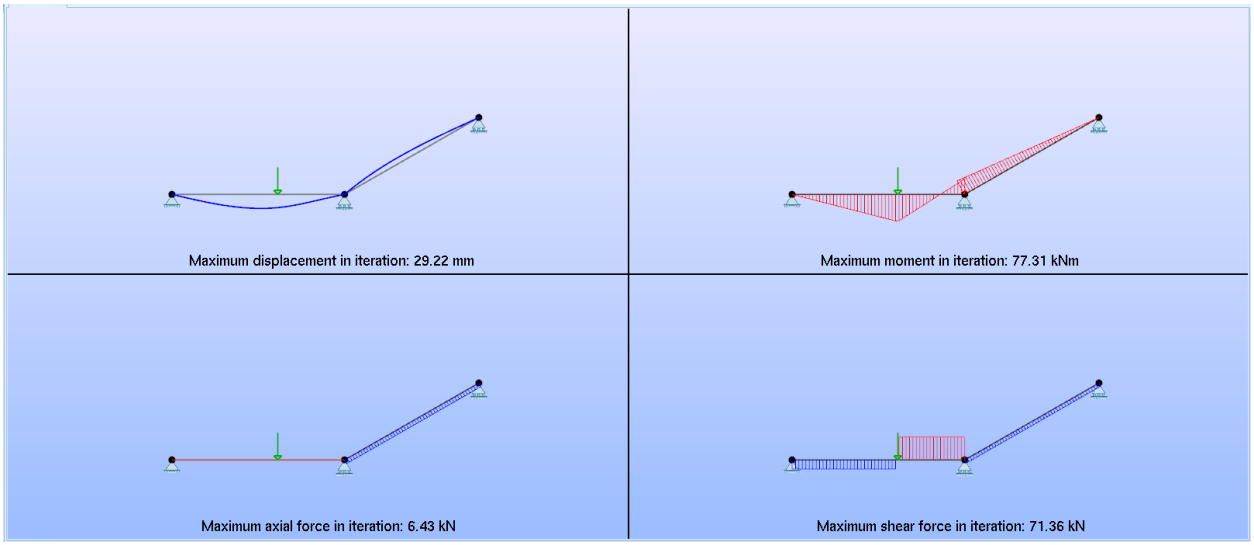
Figur 2: Internal joints, host members og submembers

Faguttrykk om modeller i **fap2D** som nevnes i rapporten er definert på samme måte som i brukermanualen[2]. Disse inkluderer:

- Konstruksjonsmodell: Blir tegnet av brukeren av **fap2D** og består av *members*, *joints*, opplagere og andre konstruksjonsdetaljer brukeren hvelger å modellere på konstruksjonen.
- Beregningsmodell: En konstruksjonsmodell blir omgjort til en beregningsmodell i **fap2D** bestående av et gitt antall elementer på hvert *member*. Dette gjøres i forkant av analyser for at det skal bli mulig å gjennomføre beregninger på konstruksjonsmodellen. Elementene får også tilegnet sine respektive seksjonskrefter og forskyvninger når en beregning er fullført. Standard elementinndeling er 50 elementer per *member*.
- Responsparameter: Forskyvninger, momenter, aksial- og skjærkrefter på en joint, i et gitt element eller i et opplager er eksempler på responsparametre.

Responstyper refererer til de ulike responser som kan betraktes på en fullstendig modell. Blant disse finnes forskyvninger, momenter, aksial-, skjær- og reaksjonskrefter.

Videre er et responsdiagram en visualisering av en gitt responstype eller responsparameter tegnet på konstruksjonsmodellen i resultatvisningen av en beregning i **fap2D**, jamfør Figur 3.



Figur 3: Responsdiagrammer for fire ulike responstyper i fap2D

2.3.2. «Sanntids» beregninger og influenslinjer

Definisjoner og forklaringer til uttrykk som eksisterer i sammenheng analysene «sanntids» beregninger og influenslinjer forklares her.

Lasttogene i begge analysene inneholder et lokomotiv. Dette er ikke et objekt som blir satt av koden eller av brukeren, og blir brukt som referanse til den aller fremste posisjonen i lasttoget. Lokomotivet blir ikke referert ved navn til på noe tidspunkt i **fap2D** og brukes i denne rapporten som en definisjon. Lastene som blir laget i lasttogene blir oppgitt med en gitt avstand til lokomotivet, slik kommer nytteverdien til et slikt konsept inn.

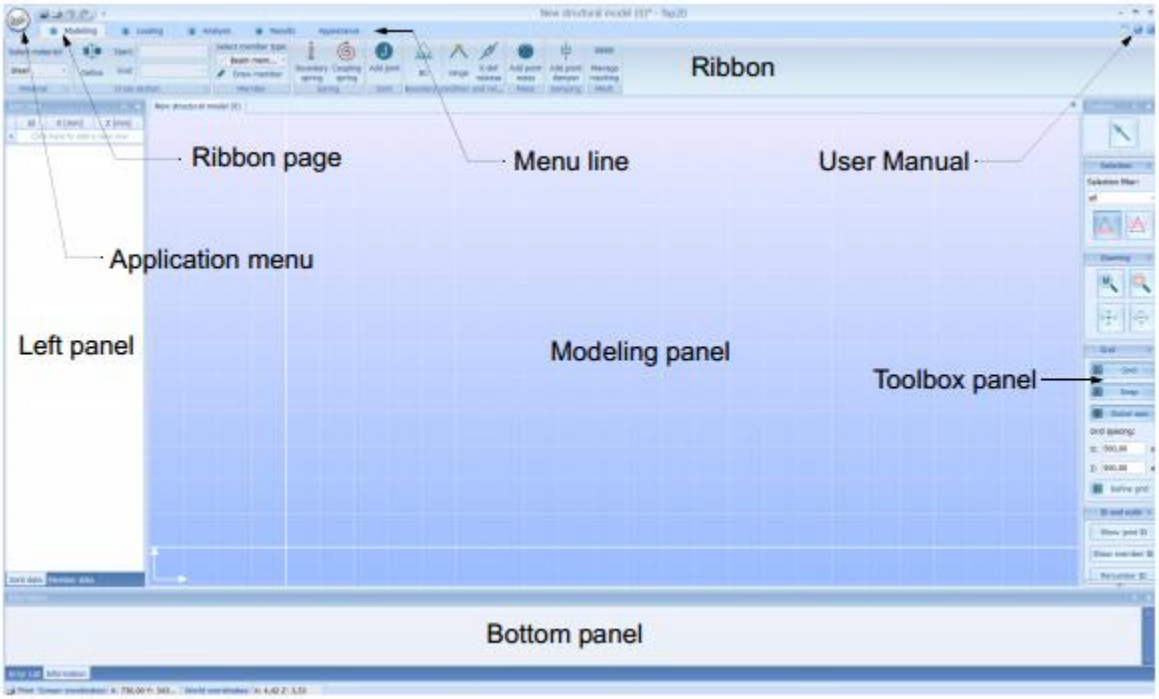
Lokomotivet er dermed en egenskap ved lasttoget og ikke en last, det er ens eget valg om man vil plassere en last i fremste posisjon på toget. Dette vil være nyttig i situasjoner hvor man vil neglisjere lokomotivets last i forhold til påfølgende laster og dermed ikke sette en last på lokomotivet.

Under influenslinjer blir de ekstreme responsene for hvert lasttog funnet ved hjelp av Xtrm-analysis. I denne rapporten har denne funksjonen blitt oversatt til analyse for ekstreme responser og heretter refereres denne funksjonen til dette navnet.

I oppgaveteksten er analysen som skal implementeres referert til som lineært statiske beregninger. For å unikt definere analysen er den heretter referert til som «sanntids» beregninger, altså samme navn som den ble referert til som i høstens prosjektoppgave[1]. I selve programmet er analysen implementert som «real time analysis». Lastbane er synonymt med kjørebane som nevnes i oppgaveteksten.

2.3.3. Grafisk grensesnitt

Modellvinduet og andre grafiske elementer refereres til gjennom hele rapporten og under er en oversikt over elementene i **fap2D** sitt grafiske grensesnitt. Figur 4 er hentet fra brukermanualen til **fap2D**[2].



Figur 4: Oversikt over de grafiske elementene i fap2D hentet fra[2]

I denne rapporten er «modeling panel» modellvinduet og «toolbox panel» er kjent som verktøylinjen.

3. Arbeidsprosessen

Planleggingen av implementasjonsarbeidet var allerede gjort i prosjektoppgaven og en grovskisse av fremdriften ble laget som et Ganttdiagram. Dette diagrammet er vedlagt i vedlegg 0. Det er planlagt at arbeidsprosessen skal følge den planen noenlunde. Den faktiske fremdriften som grafisk viser hvor mye tid som egentlig ble brukt på hver oppgave er lagt ved i vedlegg 0 til sammenligning.

Siden det ble programmert til i løpet av prosjektoppgaven har alt av nødvendig programvare blitt installert på forhånd før arbeidet med masteroppgaven startet. En SVN-server for deling av kode og versjonskontroll ble også satt opp i løpet av høsten 2012 slik at alle medvirkende var ferdig oppkoblet til denne innen arbeidets start.

Arbeidsprosessen med SVN-serveren har gått ut på å dele redigert kildekode med kommentarer for hver endring som er gjort, i tillegg til en oversikt over hva som er endret for hver deling. Programmeringen har fulgt kodekonvensjonene beskrevet i vedlegg D under hele arbeidet med masteroppgaven.

Selve rapporten har blitt skrevet på grunnlag av tilbakemeldingene fra veileder på prosjektoppgaven. Ellers har rapporten blitt skrevet på når det var behov for utfyllende forklaring om tankegangen bak valg som var gjort i programmeringssammenheng.

Den daglige loggen inneholder en oversikt over hva som er gjort programmeringsmessig. Den grafiske oversikten over hva som har blitt implementert og Ganttdiagrammet over den faktiske fremdriften har blitt oppdatert for hvert større tillegg som har blitt laget.

Hver enkelt av studentene som har jobbet med **fap2D** dette semesteret jobbet med hver sine oppgaver. Likevel hadde alle kryssende arbeidsprosesser blant annet når man skulle komme til enighet om kodekonvensjoner og andre fellesregler for programmeringen.

Videre har det vært ukentlige møter for å presentere egen fremgang og problemer som krever innspill og gjennomgang. Det har også vært individuelle møter med veileder, spesielt med tanke på å diskutere endringer som trengtes i beregningskjernen og veileders preferanser på grensesnittets utseende.

4. Utviklingen

Valgene som har blitt gjort med tanke på implementering under hele utviklingsprosessen er beskrevet i dette kapittelet. En grafisk oversikt over noe av koden som har blitt programmert i finnes i vedlegg C. Dette er for å gjøre det enklere for fremtidige utviklere å sette seg inn i virkemåten til den implementerte koden.

Koden er bygget opp rundt eksisterende strukturer i **fap2D**, sist beskrevet i forrige masteroppgave knyttet til programmet[16]. Arbeidet i denne masteroppgaven har ikke endret programmets overordnede struktur.

4.1. Brukergrensesnitt

Ved implementering av brukergrensesnitt for «sanntids» beregninger i form av knapper hvor brukeren kan velge blant annet analyse og innstillinger, har DevExpress sine automatiske funksjoner for å lage grensesnitt blitt brukt. På denne måten får hver knapp sine innstillinger som samsvarer med innstillingene som finnes på de eksisterende knappene i **fap2D**. I tillegg blir metoder for å håndtere hendelser relatert til knappene automatisk generert.

4.2. Samkjøring med metoder for influenslinjeberegning

I begynnelsen av 2013 var planen å samkjøre metodene for «sanntids» beregninger og influenslinjer. Dette var fordi influenslinjer inneholdt mange av de samme konseptene som «sanntids» beregninger, med tanke på lasttog, lastbaner, ekstreme responser og lasttogposisjoner, som nevnt i prosjektrapporten[1].

Det viste det seg at de samme konseptene innenfor begge analysene hadde så ulike egenskaper at de ikke burde slås sammen. Lasttog for «sanntids» beregninger hadde for eksempel individuelle retninger på sine laster, i tillegg til at de kunne håndtere momentlaster og generelt bevege seg friere enn togene for influenslinjer.

Det meste av logikken rundt plasseringer av lasttog og inndeling av lastbanen for «sanntids» beregninger eksisterte heller ikke i samme grad for influenslinjer. Dermed ble samkjøringen skrinlagt med lastbanene som unntak. Ved ferdigstillingen av implementasjonen kan lastbaner brukes i begge analyser, uavhengig av hvor de er laget.

4.3. Interpolering

Da beregningsdelen av «sanntids» beregninger skulle implementeres måtte det gjøres et valg om hvordan lastene i lasttoget skulle ha innvirkning på beregningsmodellen. Enten skulle alle lasttogposisjoner plasseres etter nodeinndelingen på modellen, eller så måtte lastene plasseres i egne posisjoner og lineærinterpoleres mellom nærmeste noder.

Etter konsultasjon med veileder ble det bestemt at interpolering skulle benyttes. Dette var fordi det var viktig at lasttogposisjonene hadde faste avstander langs lastbanen. En slik inndeling var verre å få til dersom lastene skulle stå i posisjonen til hver enkelt node, siden nodenes plassering varierte etter elementinndelingen på hvert *member*.

Selv om interpolering gir et mer tilnærmet resultat, ble det bestemt etter samråd med veileder at løsningen var tilstrekkelig. Dersom det skulle være behov for mer nøyaktige resultater, ordnes dette med en finere elementinndeling og dermed kortere avstand mellom nodene under interpoleringen.

Momentlaster kan ikke interpoleres og forflyttes dermed til nærmeste node som en forenkling, uavhengig av om noden er foran eller bak lasten i forhold til togets kjøreretning.

4.4. Influenslinjer

Det meste av funksjonaliteten rundt «sanntids» beregninger ble ferdig implementert før påske. I følge oppgaveteksten skulle da omhyllingskurver bli den neste oppgaven etter påskeferien var over. Dette ble derimot ikke tilfelle da influenslinjer fikk prioritet over omhyllingskurver etter samråd med veileder.

Dette var fordi influenslinjer hadde funksjonelle feil som trengte oppmerksomhet og det var fokus på å få et mest mulig stabilt program fremfor å implementere flere analyser. Spesielt feil i metodene knyttet til responsparametre var viktige å fjerne, siden problemer med disse hadde også innvirkning på andre analyser. Dette er fordi responsparametre går inn som variabler i andre analyser.

Omhyllingskurver ble derfor sett bort fra i denne masteroppgaven.

4.5. Vedlikeholdsoppgaver

I tillegg til selve analysene har det også blitt gjort en del arbeid med generelt vedlikehold av **fap2D**. I samspill med de andre medvirkende på programmet ble det fordelt oppgaver oss imellom for å forbedre koden og gjøre den stabil og enklere å videreføre for nye utviklere.

I den sammenheng ble kommentering av kode en av oppgavene i denne masteroppgaven. For å gjøre koden oversiktlig har rundt 4500 plasser i koden fått kommentarer angående funksjonalitet og egenskaper. En del tid i etterkant av påsken gikk med på denne jobben.

Det ble også lagt ned arbeid i å finne en passende lisenstjeneste som kunne brukes til å administrere lisenser for en lansert versjon av **fap2D**. Dette viste seg å bli vanskelig å utføre innenfor den tidsrammen som var til rådighet, og ble derfor et prosjekt for fremtidige utviklere.

5. Funksjonalitet

Kapittelet forklarer funksjonaliteten til de ulike implementasjonene som det er arbeidet med i løpet av denne masteroppgaven. Oversikten er delt inn etter analyse og videre etter hovedpunkter og funksjoner under hver analyse. Den er også ment å belyse alle sider ved analysene for å gjøre det enklere å forstå virkemåten for fremtidige utviklere. Selve koden er også kommentert med forklaringer til hver av metodene.

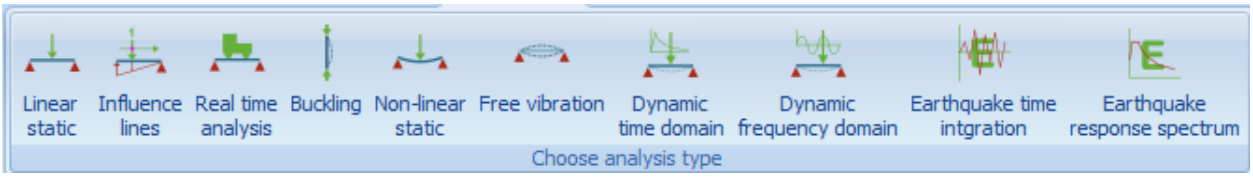
5.1. «Sanntids» beregninger

Funksjonaliteten til «sanntids» beregninger kan deles inn i to kategorier, analyse og resultater. Under analysedelen blir alle detaljer rundt lastbanen og lasttoget definert. Resultatdelen består av å plassere toget som ønsket og betrakte responsen i konstruksjonsmodellen avhengig av plasseringen.

5.1.1. Analyse

Oppbygning av lasttog og lastbane

«Sanntids» beregninger er implementert som en egen analyse med knapp under «Analysis»-fanen, på lik linje med de andre analysene i **fap2D**, jamfør Figur 5.



Figur 5: De ulike analyseknappene i fap2D

Ved å velge knappen «sanntids» beregninger, utvides antall valg under «Analysis»-fanen og innstillingene for denne fanen kommer til syne, jamfør Figur 6.

Figur 6: Analyseknappene med instillingene til «sanntids» beregninger

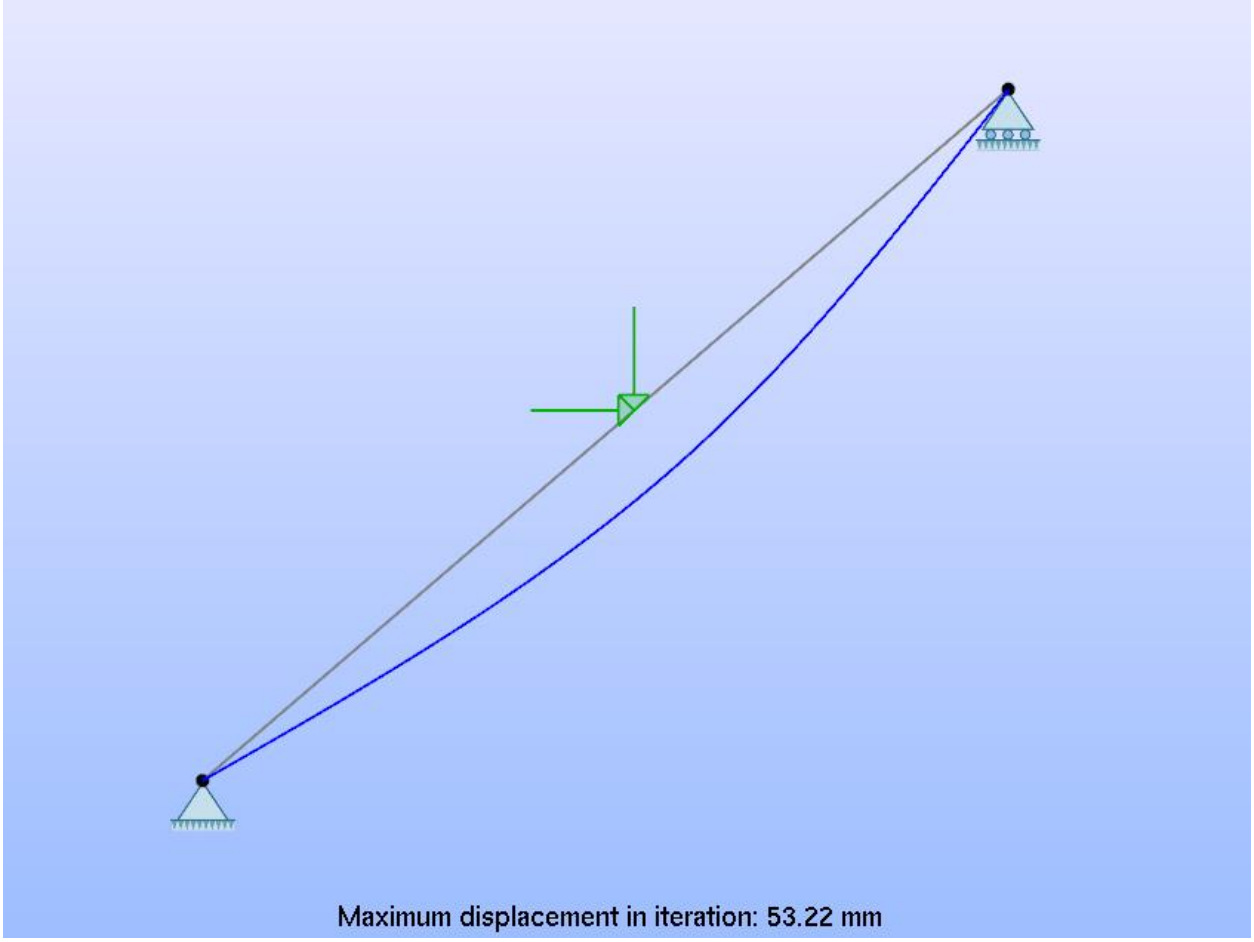
Man har blant annet mulighet til å ta hensyn til skjærdeformasjoner i beregningsmodellen. Slike deformasjoner blir ikke lagt til med mindre det velges eksplisitt.

Før en analyse kan starte må lasttog og lastbane defineres. Symbolene for disse er forskjellige fra de som benyttes for influenslinjer. Dette er fordi «sanntids» beregninger alltid kan benytte seg av lastbaner som er laget for influenslinjer, mens det motsatte ikke alltid er tilfelle. Dermed er det hensiktsmessig å skille mellom de.

Lastbanen defineres ved å velge knappen for lastbane, forlate dialogboksen som kommer opp og deretter merke de *members* på modellen man vil ha med i lastbanen. Eventuelt er det mulig å merke *members* på forhånd og velge å legge disse til i lastbanen via dialogboksen som presenteres.

Lasttog defineres ved sin egen knapp, som åpner en dialogboks. I dialogboksen kan man definere så mange lasttog som man ønsker. Når man har opprettet et tog, kan man legge til et vilkårlig antall enkeltlaster i toget, samt deres avstand til lokomotivet. Som nevnt under definisjoner trenger det ikke nødvendigvis å være en last på selve lokomotivet. Følgende regler gjelder for lastene i lasttoget:

- Man kan legge til tre typer laster: punktlast i global x- og z-retning og konsentrert moment.
- Lastene virker kun i globale retninger (ikke i forhold til lokale aksesystemer).
- Lastene kan ha både positiv og negativ retning, men ikke verdien 0. Negative punktlaster virker i negativ global retning og negative momenter virker mot urviseren. Dette er i samsvar med standarden i **fap2D**.
- Laster kan ikke ha negativ avstand til lokomotivet, men avstanden kan være lik 0. Det vil si at lasten er fremst i toget. Alle avstander er målt bakover langs toget i forhold til lokomotivet.
- Laster kan plasseres med samme avstand til lokomotivet som andre laster. Lastene er da plassert i samme punkt. Dette for eksempel nyttig for dekomponering av laster som skulle stått normalt på et skråplan, jamfør Figur 7.



Figur 7: Eksempel på bruk av «sanntids» beregninger med to laster i samme punkt

Lasttoget som er valgt i listen over opprettede tog er det aktive toget som vises når resultatene skal betraktes. Det er kun mulig å vise ett tog på lastbanen av gangen. Både lasttog og lastbanen lagres med modellen.

Det er også nødvendig å definere antall lasttogposisjoner langs lastbanen før en analyse kan starte. Lasttogposisjonene er punkter langs lastbanen, med innbyrdes lik avstand, som lasttoget skal innom. Posisjonene skiller seg fra nodeinndelingen, ved at det alltid er like langt mellom hver posisjon, uavhengig av hvordan man har delt inn modellen i elementer. Prinsippet om fast avstand mellom posisjonene blir også overholdt ved krumme *members*.

Standard antall lasttogposisjoner er 100, man kan velge færre eller flere og antallet lagres med modellen. Maksimalt antall posisjoner er 2 000 000, og minimum er 2, siden lasttoget må minst innom begynnelsen og slutten av lastbanen.

Man kan også velge hvilken type respons som først skal betraktes når resultater skal vises. Dette kan gjøres før analysen starter. Man har følgende typer responser å velge mellom:

- Forskyvninger
- Moment
- Aksialkrefter
- Skjærkrefter
- Alle de fire ovennevnte samtidig
- Reaksjonskrefter

Dersom ingen type respons blir valgt starter analysen med moment som standard responstype.

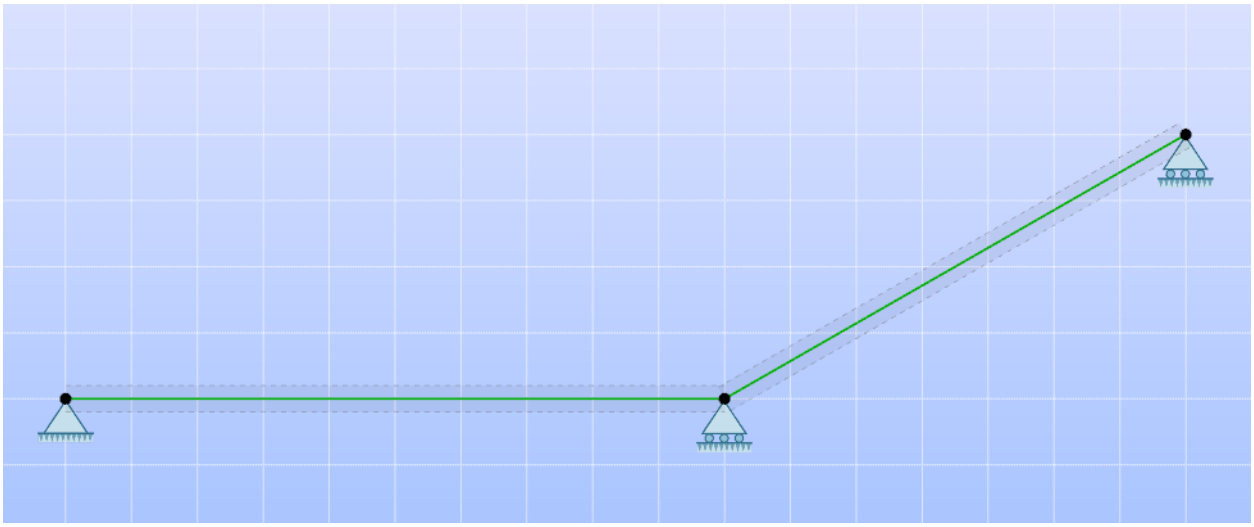
Oppstart av analysen

Når man velger å starte analysen fra «Analysis»-fanen, starter **fap2D** med å lage elementinndelingen som er standard i alle andre analyser. Deretter gjøres lasttogene om til objekter som kan brukes i beregningsmodellen. Grunnen til dette er at lasttogene må benytte seg av noder som er tilordnet elementene og dermed kun eksisterer i beregningsmodellen, som ikke blir laget før analysen startes.

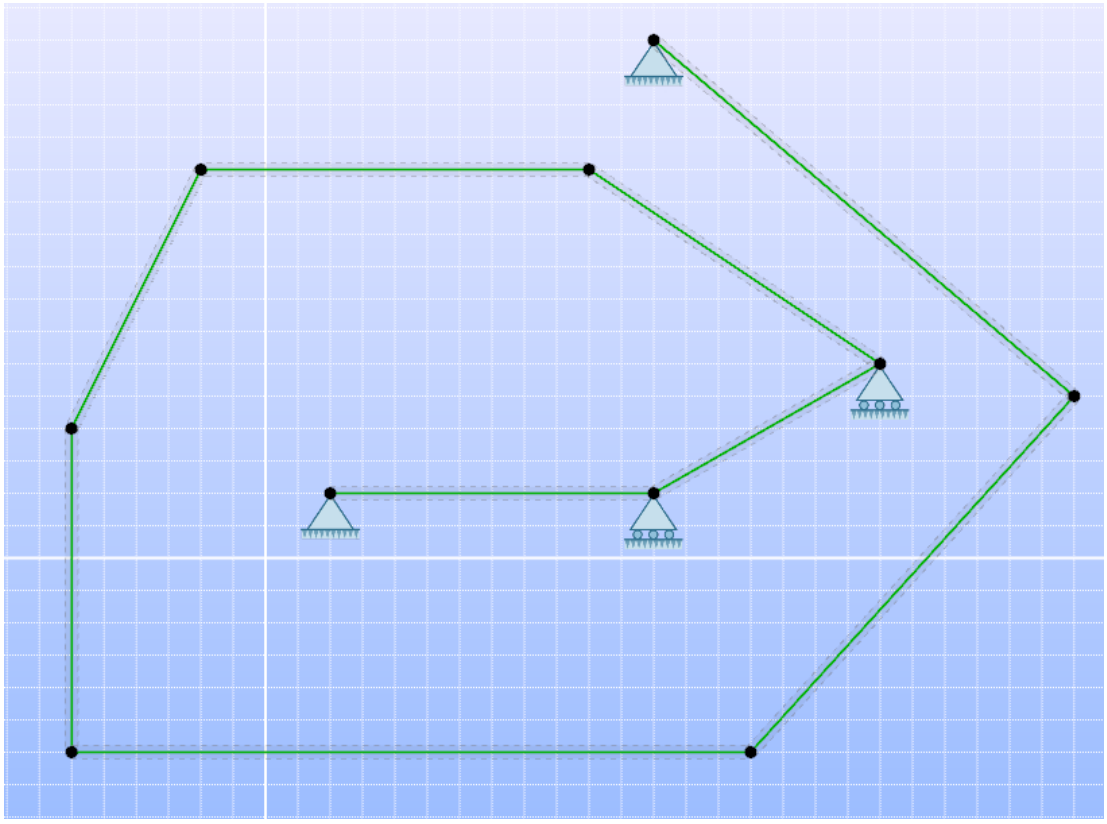
Etterpå må både lastbanen og lasttoget sjekkes for feil. Følgende krav må tilfredsstilles før lastbanen kan godkjennes:

- Lastbanen kan ikke inneholde forgreninger. Lasttoget kan ikke velge hvilken vei det skal gå, i tilfelle en forgrening oppstår.
- Lastbanen må inneholde minst en *member*.
- Lastbanen kan ikke starte og slutte i samme punkt.
- Lastbanen må være sammenhengende.
- Lastbanen kan kun bestå av rette og krumme *beam members*.

I motsetning til influenslinjer trenger ikke en lastbane for «sanntids» beregninger å bestå av *members* i påfølgende rekkefølge fra venstre mot høyre. En slik lastbane kan inneholde loddrette elementer og ha utstrekning i hvilken som helst retning så lenge den tilfredstiller kravene ovenfor. Selv om influenslinjer og «sanntids» beregninger kan benytte seg av de samme lastbanene, er det ikke sikkert at samme lastbane blir godkjent for bruk av begge analyser. Dette er eksemplifisert i Figur 8 og Figur 9.



Figur 8: Eksempel på lastbane som blir godkjent av begge analyser



Figur 9: Eksempel på lastbane som kun blir godkjent av «sanntids» beregninger

For at de definerte lasttogene skal bli godkjent må de tilfredsstille følgende krav:

- Det må eksistere minst ett lasttog.
- Alle lasttog må ha minst en last.

Etter godkjenningen blir lastbanen delt opp i lasttogposisjoner som lagres i en liste. Posisjonene i listen er sortert etter avstanden fra starten av lastbanen.

Starten av lastbanen er definert som den enden av lastbanen som er lengst til venstre og nederst. Dermed er det naturlig å tenke seg at lasttoget beveger seg fra venstre mot høyre langs lastbanen, selv om lastbanen har mulighet til å ende hvor som helst. Lasttoget er i stand til å følge lastbanen uansett, da den bare følger sammenhengenhet mellom de enkelte *members*.

Lastene i lasttoget blir sortert etter avstand fra togets lokomotiv. Dette er nødvendig for påfølgende metoder, som er avhengig av korrekt rekkefølge for å kunne plassere lastene riktig.

Med både lasttogposisjonene og lastene sortert, begynner søket etter ekstreme responser innenfor alle fem nevnte responstyper for det aktive lasttoget. En ekstrem respons er maksimalverdien langs lastbanen for en gitt type respons. Andre lasttog får ikke gjennomført dette søket før det blir valgt under resultatvisningen, for å spare tid under oppstarten av analysen.

Søket gjennomføres ved at lasttoget forflyttes en lasttogposisjon av gangen langs lastbanen i begge retninger uten at noe visualiseres. Etter hvert som ekstreme responser blir funnet, lagres togets plassering i en liste over plasseringer for hver type respons. Hver respons har to lister knyttet til seg, en for ekstreme responser som blir funnet i hver retning. Dersom flere plasseringer av lasttoget gir maksimalverdi i samme retning langs lastbanen, blir alle plasseringene lagret i den tilhørende listen. Verdiene for to plasseringer av lasttoget er definert som like dersom absoluttverdien av differansen mellom de er mindre enn 10^{-6}.

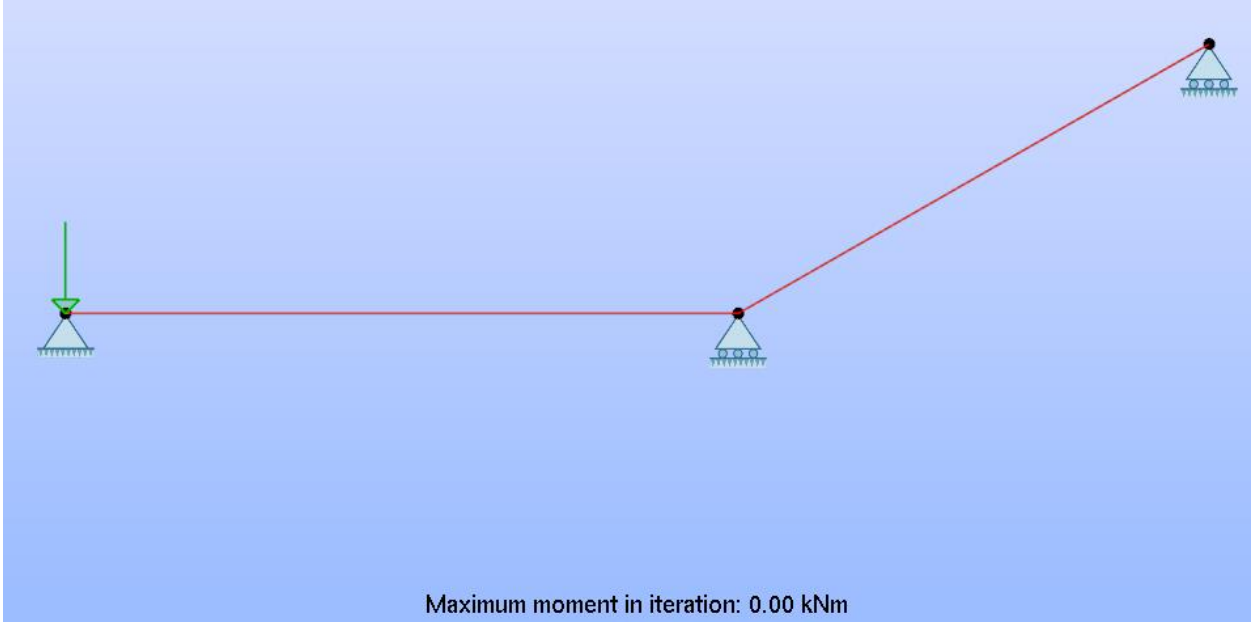
For å finne de ekstreme responsene må det gjennomføres en beregning i **Frame2D** for hver gang lasttoget forflyttes. Det er her lagringen av faktorisert stivhetsmatrise fra første kall til **Frame2D** kommer til nytte. Gjenbruk av den lagrede matrisen gjør at kjøretiden for søket etter ekstreme responser blir betydelig redusert, i tillegg til at resultatene ikke visualiseres. Dermed omgås denne flaskehalsen for kjøretiden. Dette samsvarer med resultatene av tidstestene fra høstens prosjektoppgave[1], som viste tidsgevinster ved gjenbruk av den faktoriserte stivhetsmatrisen og at visualisering var det mest tidkrevende aspektet ved en slik analyse i **fap2D**.

Når søket er ferdig settes lasttoget tilbake til startposisjonen til venstre på lastbanen, med kjøreretning fra venstre mot høyre. Modellens respons i togets første posisjon blir så beregnet og toget er klart til å betraktes under «Results»-fanen. Når man velger «Results»-fanen etter en gjennomført analyse, står det aktive toget klar i første posisjon.

5.1.2. Resultater

Resultatvisning og relativ posisjonering

Valg av resultatfanen gjør at innstillingene for kontroll av lasttoget vises og resultater presenteres i modellvinduet med lasttoget i første lasttogplassering på lastbanen, som i Figur 10.



Figur 10: Lasttogets initielle plassering

Ved hjelp av posisjoneringsknappene under «Results»-fanen har lasttoget mulighet til å bevege seg i begge retninger langs lastbanen. Det er alltid lokomotivet som beveger seg en lasttogposisjon fremover av gangen, så lenge lokomotivet er tilstede på lastbanen. Lokomotivet har mulighet til å bevege seg utenfor lastbanen så lenge det er andre laster bak den langs lastbanen.

Dersom lokomotivet er utenfor lastbanen er det lasten som er lengst fremme langs lastbanen som beveger seg en lastposisjon om gangen. Den fremste lasten flyttes da frem til nærmeste lasttogposisjon og følger deretter posisjonene i listen. Lokomotivet og laster som er utenfor lastbanen kommer tilbake ved å flytte toget tilbake til en tidligere lasttogposisjon. Det er også mulighet for å flytte toget mer enn en lasttogposisjon av gangen, dette er en innstilling man kan styre selv under «Results»-fanen.

Alle laster, inkludert den fremste, posisjoneres relativt til den fremste enheten på toget, altså enten lokomotivet eller den fremste lasten, avhengig av om lokomotivet er på lastbanen eller ikke. Posisjoneringen er uavhengig av hvor lasttogposisjonene er. Punktlastene plasseres alltid med nøyaktig avstand fra fremste enhet i forhold til hva man har definert når lasttoget ble opprettet. Unntaket er momentlaster, som plasseres i nærmeste node fra deres avstand til fremste enhet.

Når en last blir posisjonert i forhold til fremste enhet i lasttoget, blir det funnet hvilken *member* lasten er på og hvor langt inne på sin *member* lasten er. Dette gjennomføres uavhengig av om den gitte *member* er krummet eller rett. Denne fremgangsmåten sikrer at det alltid er fast avstand mellom lastene, noe som ikke hadde vært mulig dersom man for eksempel hadde plassert hver last i nærmeste node i forhold til avstanden til fremste enhet. Dette er fordi avstanden mellom hver node kan variere etter elementinndelingen.

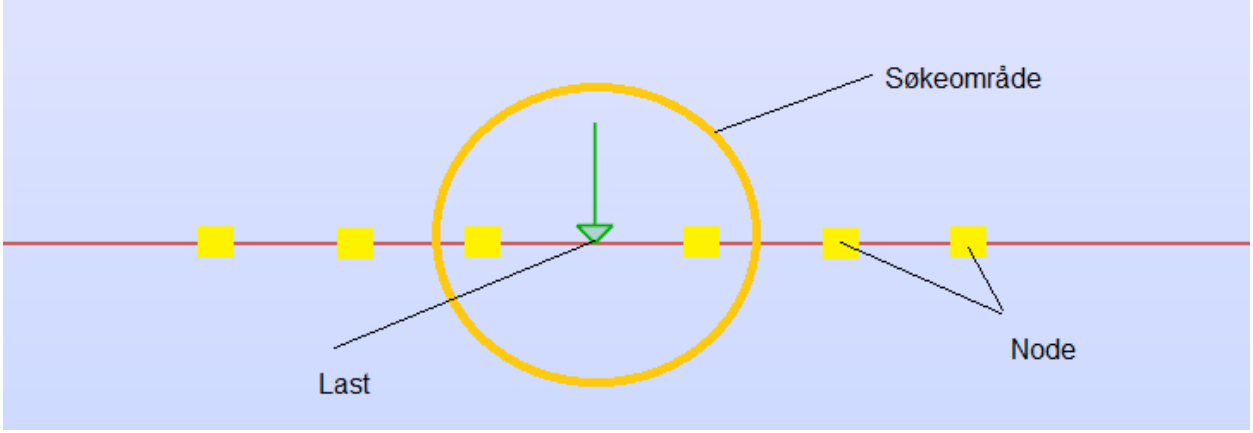
Hver gang toget endrer plassering langs lastbanen posisjoneres lastene relativt til fremste enhet. For hver last sjekkes det også om toget har beveget seg langt nok inn på lastbanen til at lasten kan bli aktiv på lastbanen, eller om den har forlatt lastbanen. Laster som ikke er på lastbanen blir ikke tatt med i beregninger og er ikke aktive.

Dersom avstanden mellom to laster er lengre enn lengden på lastbanen, og den fremste lasten forlater lastbanen, plasseres den påfølgende lasten i starten av lastbanen. På den måten vil det alltid vere en last på banen.

Nodesøk og interpolering

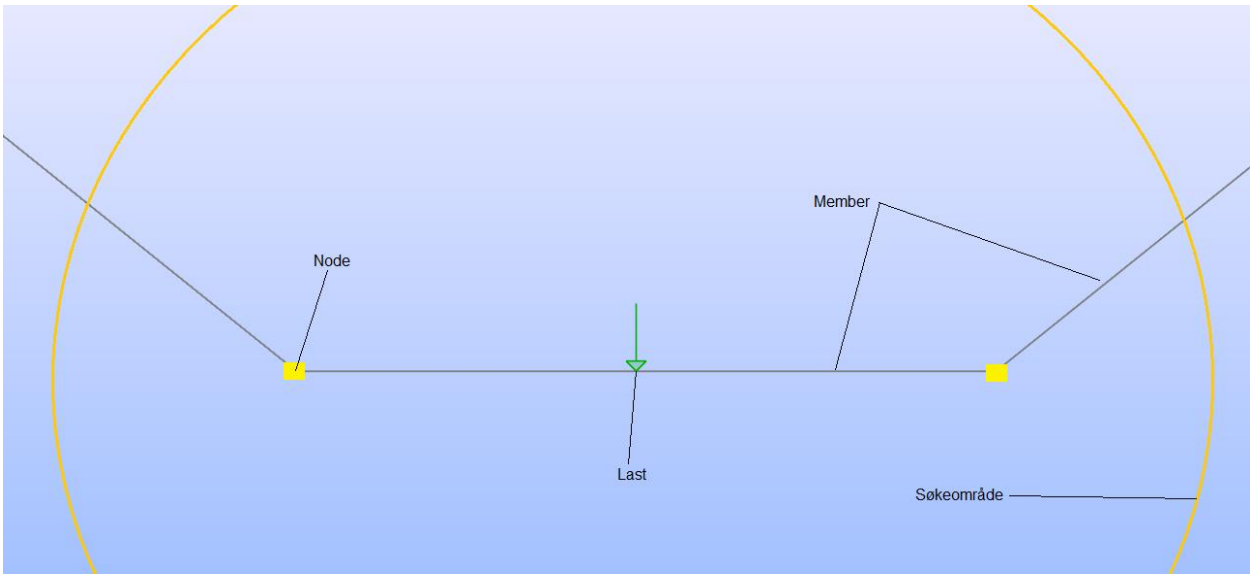
Når lastene er posisjonert på hvert sine respektive members, må de interpoleres til de nærmeste nodene i forhold til sin egen posisjon. Dette er fordi **fap2D** kun kan gjennomføre beregninger på laster som er plassert i noder. Dette fører til at resultatene fra en «sanntids» beregning blir tilnærmede verdier som blir mer nøyaktige desto flere elementer beregningsmodellen er delt inn i.

Prosessen starter med et søk rundt posisjonen til lasten etter de to nærmeste nodene innenfor en radius av lasten, jamfør Figur 11.



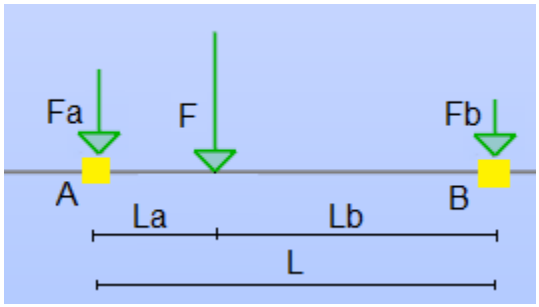
Figur 11: Illustrasjon av søk etter noder rundt lasten

Radiusen er lengden på lengste *member* i lastbanen, fordi det alltid vil være to noder innenfor radius av lengste *member* i lastbanen. Dette er fordi *joints* i hver ende av en *member* blir beregnet som noder i **fap2D** og i verste tilfelle vil en last vere posisjonert i enden av lengste *member*. Endenodene på lengste *member* vil da være innenfor et område med radius med lik lengde som nevnte *member*, jamfør Figur 12.



Figur 12: Søk etter noder i situasjonen med færrest mulig noder, søkeradius er lengden av lengste member

Søkemetoden fungerer både for rette og krumme *members*. Det er også et krav under søket at noden må vere på samme *member* som lasten er plassert på, hvis ikke kan det oppstå feil i interpoleringen.



Figur 13: Interpolering av lasten F

Interpoleringen illustreres i formel 2 og Figur 13. Metoden er generell og fungerer for alle tilfeller i begge retninger. Avstanden til nodene A og B fra posisjonen til lasten F er henholdsvis L_a og L_b. Avstanden mellom A og B er L. Dermed:

$$L = L_a + L_b \tag{1}$$

F blir lineært interpolert til lastene F_a og F_b, der størrelsene på F_a og F_b blir beregnet ved:

$$F_a = F\frac{L_b}{L}, \quad F_b = F\frac{L_a}{L}, \quad F_a + F_b = F \tag{2}$$

Momentlaster kan ikke interpoleres mellom to noder. Derfor blir slike laster unntatt fra interpoleringsmetoden og alle momenter blir automatisk forflyttet til nærmeste node fra sin posisjon.

Lagring til lister

Hvert lasttog inneholder en liste over absolutt alle laster som er inkludert i toget. Det er likevel kun lastene som er aktive på lastbanen som skal tegnes og kun lastene som er interpolerte i noder som beregninger skal utføres på.

For å holde orden på dette er det egne lister som tar seg av disse lastene. Etter at en last er ferdig interpolert, plasseres de interpolerte lastene i en egen liste for beregning og den originale lasten i en egen liste for visualisering. Når alle lastene i lasttoget har blitt gjennomgått brukes disse listene respektivt til beregning og visualisering. Listene tømmes for hver gang lasttoget forflyttes, det er kun den originale listen over laster i lasttoget som beholdes.

Visualisering og beregning

Lastene i listen for visualisering blir sendt til **fap2D** sine tegnemetoder og tegnes på modellen i sin nøyaktige posisjon i forhold til fremste enhet i lasttoget. Unntaket er momentlaster som tegnes i noden de er plassert i. Lastene blir også tegnet etter skala til største last i lasttoget.

Lastene i listen for beregning blir sendt til **fap2D** sin beregningsmodell og konvertert til et format som kan behandles av beregningskjernen **Frame2D**. Lastene blir omgjort til ytre krefter i modellen og legges i den vedrørende listen for ytre krefter, Trf[3].

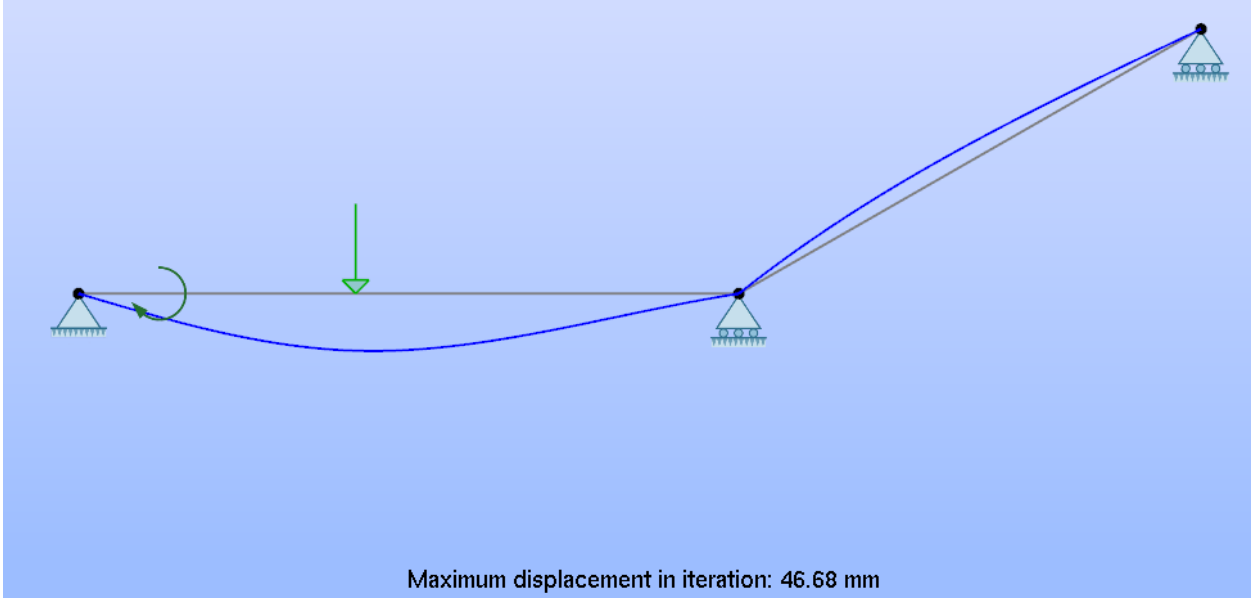
Deretter blir beregninger gjennomført og resultatet fra forflyttingen av lasttoget sammen med responsen som blir tegnet med tegnemetodene i **fap2D**. Alle beregninger som blir gjort under resultatvisningen av «sanntids» beregninger benytter seg av gjenbruk av lagret faktorisert stivhetsmatrise.

Vending av toget

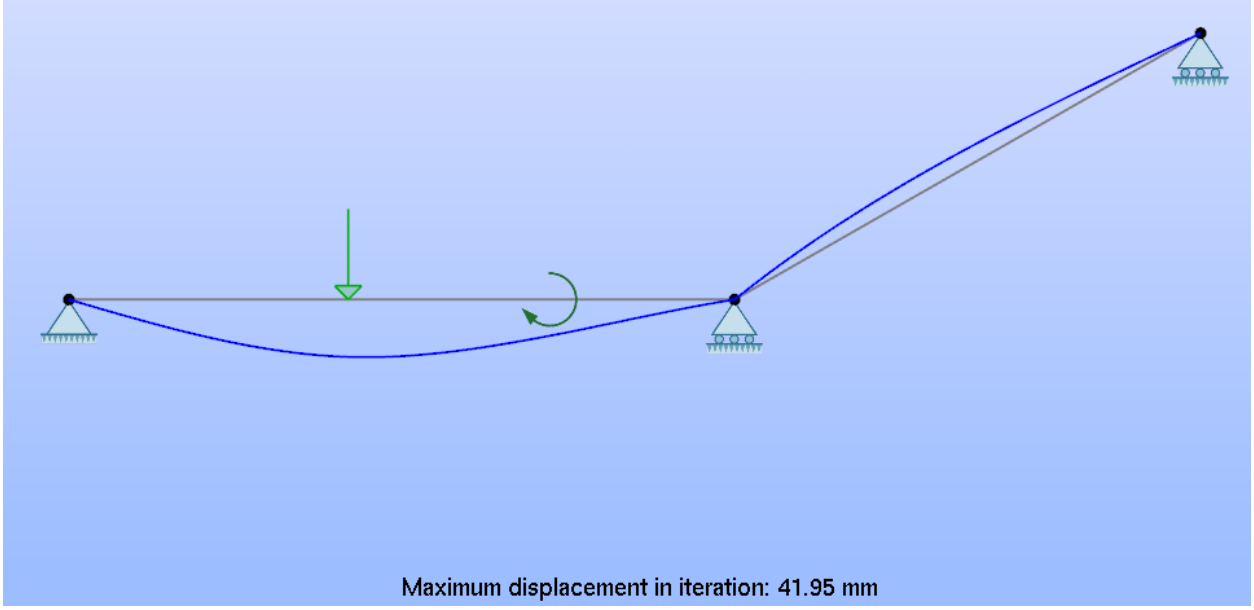
Lasttoget har en egenskap som gjør at den kan bevege seg i motsatt retning av retningen det allerede beveger seg i. De samme reglene for posisjonering gjelder fremdeles. Den funksjonelle endringen er at for hver gang lasttoget beveger seg i retning høyre mot venstre forflytter det seg en posisjon nedover i listen over lasttogposisjoner. Toget beveger seg oppover i listen dersom det forflytter seg fra venstre mot høyre.

Egenskapen er viktig fordi resultatene langs lastbanen kan bli forskjellige avhengig av hvilken retning toget beveger seg i.

Når toget blir snudd blir lastene i toget plassert speilvendt i forhold til lokomotivet. Laster som er 1500 mm til venstre for lokomitivet når det kjører fra venstre mot høyre vil nå vere 1500 mm til høyre for lokomotivet og toget vil nå kjøre fra høyre mot venstre, jamfør Figur 14 og Figur 15.



Figur 14: Lasttoget før vending



Figur 15: Lasttoget etter vending

Dersom toget snus når lokomotivet har passert høyre side av lastbanen, starter toget på nytt igjen med kjøreretning høyre mot venstre med lokomotivet plassert lengst til høyre på lastbanen. Det samme gjelder dersom lokomotivet er til venstre for lastbanen. I så tilfelle starter toget på nytt igjen med kjøreretning venstre mot høyre med lokomotivet plassert lengst til venstre på lastbanen.

Tilbakestilling i begge retninger

Tilbakestilling er en metode for å endre lasttogets plassering til enten begynnelsen eller enden av listen for lasttogposisjoner. Konsekvensene av denne metoden avhenger av hvilken retning lasttoget beveger seg i.

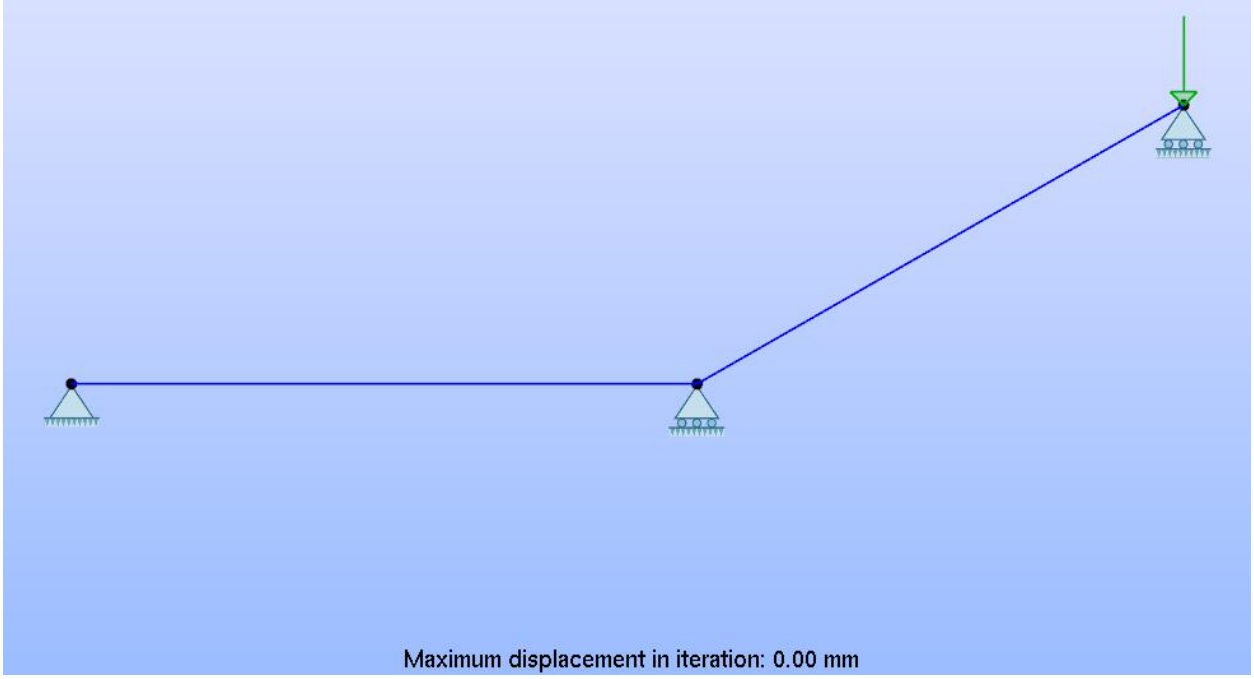
Dersom lasttoget beveger seg fra venstre mot høyre langs lastbanen, altså i stigende rekkefølge etter avstand fra start i listen over lasttogposisjoner, vil tilbakestilling til begynnelsen gjøre at lokomotivet plasseres lengst til venstre på lastbanen. Alle laster som har avstand 0 til lokomotivet vil være aktive på lastbanen, alle andre laster er da ikke aktive på lastbanen.

Når lasttoget tilbakestilles til enden plasseres den bakerste lasten lengst til høyre på lastbanen og alle andre laster gjøres inaktive på lastbanen, jamfør Figur 16.

Figur 16: Lasttoget med den bakerste lasten lengst til høyre på lastbanen med kjøreretning venstre mot høyre

Dersom toget beveger seg fra høyre mot venstre, vil tilbakestilling til begynnelsen gjøre at bakerste last plasseres lengst til venstre på lastbanen og alle andre laster gjøres inaktive på lastbanen.

En tilbakestilling til enden gjør at lokomotivet plasseres lengst til høyre på lastbanen og alle laster som har avstand 0 til lokomotivet vil også være aktive på lastbanen. Alle andre laster er da ikke aktive på lastbanen, jamfør Figur 17.



Figur 17: Lasttoget med den fremste lasten lengst til høyre på lastbanen med kjøreretning høyre mot venstre

Når bakerste last i toget er på slutten av lastbanen for sin kjøreretning har det ikke lenger mulighet til å bevege seg mer i samme retning. Dette gjelder også for vanlig forflytting av lasttoget. Bakerste last blir da stående i siste posisjon.

Valg av responser

Når som helst under resultatvisningen er det mulig å endre hvilke responser man vil betrakte. Man kan velge mellom de samme responsene som før gjennomføringen av analysen. Dette er mulig fordi alle responser som blir beregnet i **Frame2D** blir lagret for hver gang lasttoget forflyttes. Dermed er det komputasjonsmessig kostnadsfritt å skifte mellom hver av de.

I motsetning til andre analyser skaleres ikke responsene i forhold til resultatet som betraktes når toget er i en gitt posisjon. I «sanntids» beregninger skaleres alle responstyper i resultatvisningen etter den mest ekstreme responsen langs hele lastbanen. Den mest ekstreme responsen er allerede funnet i oppstarten av analysen. Det er også mulig å betrakte fire responser samtidig.

Informasjonen om resultantkreftene på modellen blir oppdatert for hver gang lasttoget forflytter seg. Denne informasjonen blir presentert i en egen dialogboks og er unik for hver lasttogets plassering.

Animasjon

Animasjonen kaller på posisjoneringsmetodene i løkke når den blir kjørt. Den er en organisert visning av plasseringen kjørt i løkke til siste last har nådd slutten av lastbanen. Animasjonen kan kjøre i begge retninger langs lastbanen.

Animasjonen blir kjørt som en bakgrunnsprosess, noe som gjør at brukere kan ha interaksjon med pauseknappen, og responsvalgene under «Results»-fanen mens den kjører. Forskjellige responser kan velges når som helst under kjøringen. Det er mulig å animere mens man betrakter alle fire responser samtidig. Når animasjonen blir satt i pause, avbrytes bakgrunnsprosessen og toget stanser i animasjonens siste posisjon.

Man kan kjøre animasjonen fra hvilken som helst plassering toget er i. Det er også mulig å overta navigeringen manuelt fra en hvilken som helst plassering og plassere toget selv ved bruk av posisjoneringsknappene.

Det er også mulig å spesifisere en forsinkelse mellom hver posisjonering i animasjonen. Når størrelsen på forsinkelsen er gitt i millisekunder, venter animasjonen med å gå til neste posisjon før den gitte tiden har passert.

Ekstreme responser

Dersom en bruker velger å betrakte et annet tog under resultatvisningen og ekstreme responser ikke har blitt funnet for dette toget, gjennomføres et søk etter ekstreme responser før det valgte toget kan vises. Dette fungerer på samme måte som for det aktive lasttoget under oppstarten av analysen.

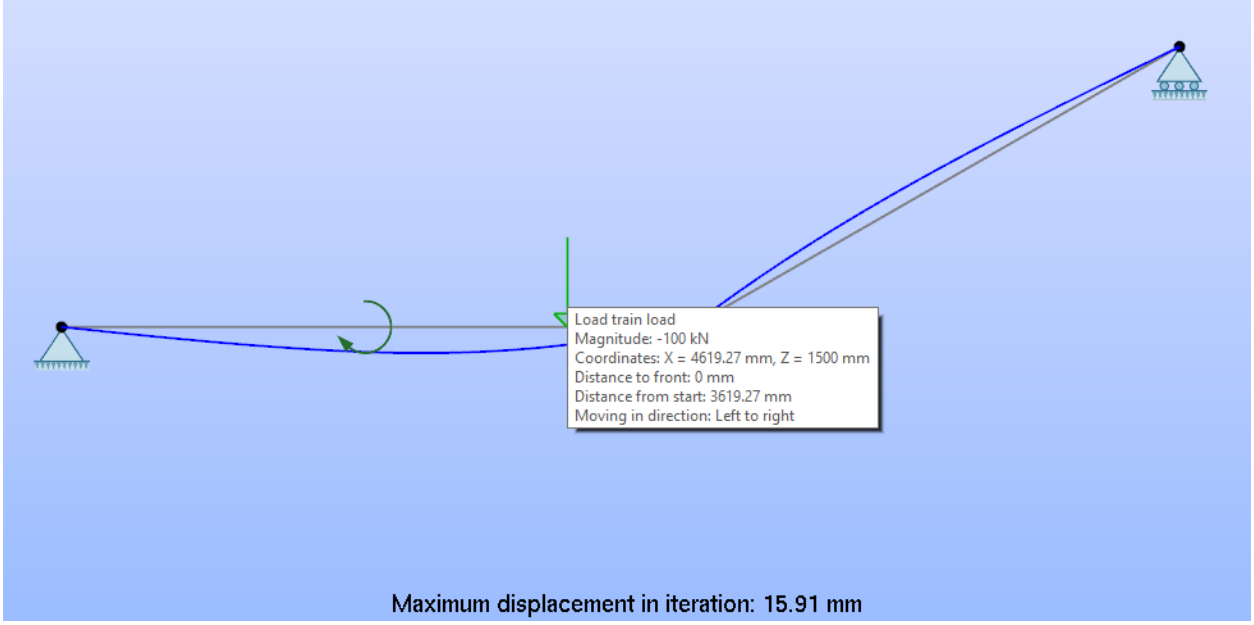
Det er også mulig å veksle mellom ulike ekstremresponser innenfor samme type respons, siden den mest ekstreme verdien kan oppstå mange plasser langs lastbanen. Når man veksler mellom responsene flytter toget seg til neste eller forrige posisjon med ekstrem respons, avhengig av hvilken man vil betrakte. Dersom lasttoget vendes, benytter metoden for ekstrem respons listen for ekstreme responser i den aktuelle kjøreretningen.

Tool tips

Lastene i lasttoget har en egenskap som gjør at man kan se informasjon om hver enkelt last ved å holde musepekeren stille over lasten. Informasjonen som da kommer opp er en *tool tip*, *som* forklarer følgende om lasten:

- Avstand fra lastbanens start.
- Posisjon.
- Avstand til lokomotivet.
- Kjøreretning.

Dette gjør det enkelt for brukere å vise informasjon når det er nødvendig. Kun laster som er aktive på lastbanen kan betraktes, jamfør Figur 18.



Figur 18: Eksempel på tool tip for last

Konvertering til lasttilfeller

I likhet med influenslinjer har også «sanntids» beregninger mulighet til å lagre en gitt lasttogposisjon som et lasttilfelle. Alle laster som til en hver tid er aktive på lastbanen blir lagret i et gitt lasttilfelle. Når lastene lagres blir det opprettet *joints* under lastene og lasttilfellet blir tilgjengelig for valg under «Loading»-fanen. De fullstendige lastene i hvert punkt blir lagret, og ikke de interpolerte. De blir lagret nøyaktig i sin posisjon med korrekt avstand til fremste enhet.

Dersom en last står på et element som grenser til en eksisterende *joint*, og er på den halvdelen av elementet som er nærmest den gitte *joint*, vil lasten bli lagret i denne *joint* i et lasttilfelle.

Betrakte elementkrefter

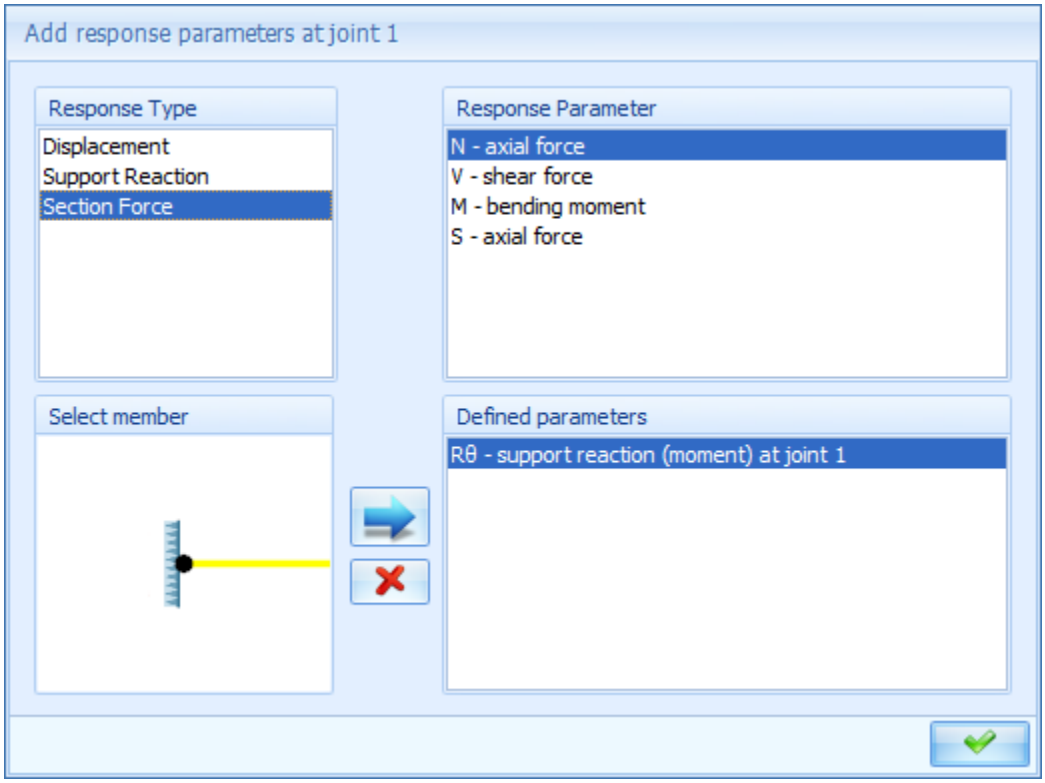
Ved å venstreklikke på et element under resultatvisningen for «sanntids» beregninger, er det mulig å betrakte elementkreftene i hver ende av elementet, i likhet med alle andre analyser. Det aktuelle elementet blir da gult og merket med en sirkel på modellen ssom indikerer dets plassering.

5.2.Influenslinjer

Funksjonaliteten til influenslinjeanalysen kan også deles inn etter analyse og resultater. Mye av analysen var allerede implementert i forkant av denne masteroppgaven, men trengte forbedringer for å gjøre den mer stabil. Endringene i funksjonalitet blir derfor oppsummert i dette delkapittelet.

5.2.1. Analyse

Responsparametre



Figur 19: Dialogboks for responsparametre i fap2D

I analysedelen av influenslinjeberegningen trengte metodene rundt responsparametre forbedringer. Aasmundrud oppdaget tidligere i dette semesteret at det var en del unøyaktigheter relatert til presentasjonen av resultater knyttet til responsparametre. Det ble funnet at metoden for responsparametre ikke klarte å tilegne rett beregningselement til rett responsparameter.

Hver responsparameter hører til i et enkelt opplager eller joint og er knyttet til en valgt *member*. Denne *member* velges i dialogboksen under vinduet «Select member», jamfør Figur 19. For beregningsformål må også et beregningselement på den valgte *member* som grenser til den responsparameterens joint finnes.

Selv om responsparametre knyttes til flere typer analyser ble oppgaven med å løse dette problemet knyttet til influenslinjer og dermed denne masteroppgaven. Problemet ble løst ved å søke gjennom elementene til den valgte *member* til man finner elementet som inneholder en node med samme nummer som jointet.

Dette er fordi noder får samme nummer som sin tilhørende joint under elementinndelingen. Siden en *joint* kun kan være på enden av en *member*, vil det kun være ett element som kan velges.

Videre var det også problemer med vinduet «Select member» i den samme dialogboksen for responsparametre. Feilen opptrådte kun på enkelte datamaskiner og førte til at *members* i vinduet ble forskjøvet når de ble merket. Det viste seg at problemet ikke var i selve koden, men at driverene til skjermkortet ikke var kompatible med OpenGL 2.1 som **fap2D** bruker. En oppgradering av driverene førte til at problemet ble løst.

5.2.2. Resultater

Tegning

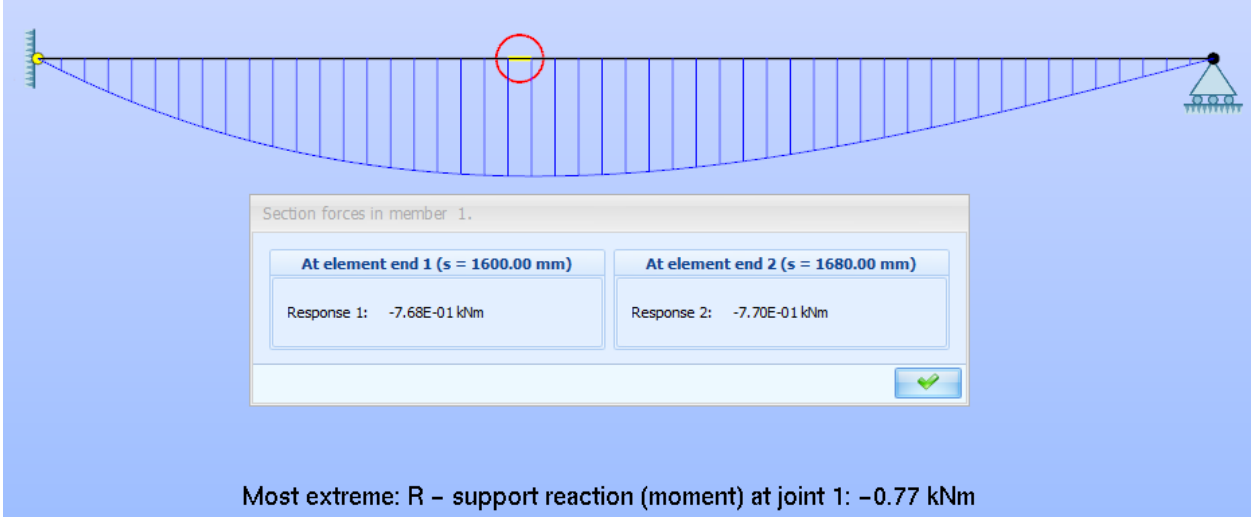
Innenfor tegningen av resultater for influenslinjer er det gjort endringer. Når laster i lasttogene til influenslinjer har posisjon utenfor lastbanen, blir disse ikke lenger tegnet på modellen. I tillegg blir laster i lasttoget nå skalert i størrelse i forhold til den største opptredende lasten i toget.

Det oppstod også et problem på roterte *members*. Dette oppsto dersom man lagde en *member* ved å tegne fra venstre mot høyre, for deretter å ta tak i venstre ende og rotere den om høyre ende. Det samme gjladt for rotering i motsatt retning. Da **fap2D** skulle tegne influenslinjer på roterte *members*, førte det til at influenslinjene for responsparametre som inkluderte momenter i *members* fikk motsatt fortegn i forhold til det som var påtenkt.

I tillegg ble influenslinjene for alle typer responsparametre tegnet slik at hvert element fikk tilordnet korrekte ordinater, men ordinatene ble plassert i feil ender når en *member* var rotert. (Dette problemet krevde tid og testing og hjelp fra veileder for å sikre at alle responsparametre ble tegnet rett og at problemet ikke var knyttet til beregningskjernen).

Problemet ble løst ved å gjøre tegnemetoden for influenslinjer sensitiv for om *members* er roterte. Det sjekkes også om responsparameteren er knyttet til momenter i *members* og i så fall snus fortegnet på influenslinjen. For andre analyser oppstod det ingen problemer for roterte *members*, noe som støttet konklusjonen om at det ikke var feil i beregningskjernen.

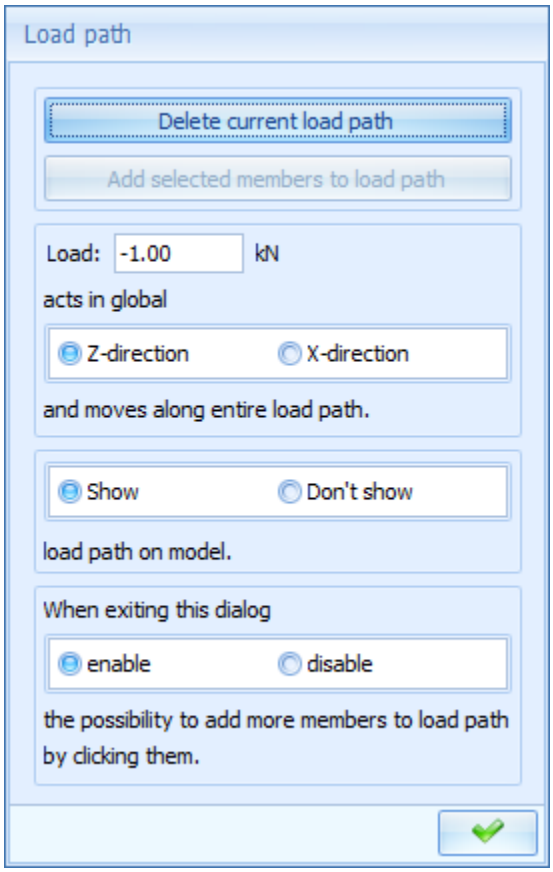
Betraktningen av krefter på elementer under influenslinjer er også forbedret. Når en bruker velger et element for å se på kreftene er elementet nå merket med en rød sirkel i modellvinduet, jamfør Figur 20.



Figur 20: Merking av elementer under influenslinjeberegningen

Analyse for ekstreme responser

Denne analysen tar inn ordinatene fra influenslinjeberegningen som en variabel. Ordinatene blir skalert i forhold til størrelsen på influenslasten som blir satt i dialogboksen for lastbane for influenslinjer, jamfør Figur 21.



Figur 21: Dialogboks for lastbane for influenslinjer

Siden man kan definere influenslasten til å være hvilken som helst verdi, med unntak av 0, må dette tas høyde for. Dette var ikke tilfelle tidligere. Tdiligere ble det antatt at influenslasten alltid var en standard enhetslast med negativt fortegn og dermed ble det ikke gjennomført noen skalering. Dersom ordinatene ikke er skalerte i forhold til influenslasten, blir verdien for den gitte ekstreme responsen også skalert feil.

Dette problemet ble løst ved å skalere ordinatene med å dele på verdien til influenslasten. Skaleringen ble først gjort i koden (.NET-delen) til **fap2D**, men ble senere forflyttet til beregningskjernen. (Dette problemet krevde en del samkjøring med veileder og noen møter for å få løst).

Det er også gjort endringer i beregningskjernen for å sikre at laster i lasttoget blir beregnet med korrekt størrelse og fortegn. Endringene er gjort i analysen for ekstreme responser og går ut på å ikke endre fortegnet som lasten ble definert med under initialiseringen av lasttoget. Samtidig blir lastene som er resultater av analysen skalert korrekt i forhold til opptredende influenslast. Tidligere ble resultatene for eksempel 1 000 ganger for store dersom influenslasten var på 1 000 kN.

Dette førte til en endring i beregningskjernen for analysen for ekstreme responser, den tar nå inn influenslasten som en variabel for å sikre at det ferdige resultatet blir korrekt. Tabellen «Tloads», som er knyttet til samme analyse og inneholder de ferdig beregnede lastene, er også forbedret slik at den skriver ut resultater i alle tilfeller og sikrer at lastene i lasttoget er korrekte.

Selv om et flertall av endringene endte opp med å bli implementert i beregningskjernen av veileder, krevde dette en god del testing og samkjøring for å få gjennomført.

Den siste endringen innenfor denne analysen er at brukere nå kan bestemme antallet lasttogposisjoner man vil bruke for beregningen. Tidligere var dette antallet 1000 posisjoner som standard, men antallet er nå valgbart for brukere via grensesnittet.

5.3.Andre oppgaver

I tillegg til analysene som har blitt fikset på har det også blitt lagt ned arbeid i andre mindre oppgaver.

5.3.1. Liste over rapporterte feil

Med hensyn på stabiliteten og videreførbarheten til programmet ble det opprettet en liste over feil som har blitt funnet og rapportert under utviklingen. Denne listen ble opprettet av undertegnede. Listen ble oppdatert og rettet på av alle og feilene som er nevnt i vedlegg H ble rapportert og løst i denne masteroppgaven. Feilene er ikke nevnt tidligere i rapporten. Flere feil ble funnet og rettet av de andre som arbeidet med **fap2D** dette semesteret.

5.3.2. Lagring av faktoriserte matriser

For å få raskere beregninger av lineære statiske problemer under «sanntids» beregninger ble faktoriserte stivhetsmatriser gjenbrukt mellom hver iterasjon, som foreslått i høstens prosjektoppgave[1]. Dermed ble modellens faktoriserte stivhetsmatrise liggende lagret i minnet til beregningskjernen selv etter at man var ferdig å bruke «sanntids» beregninger.

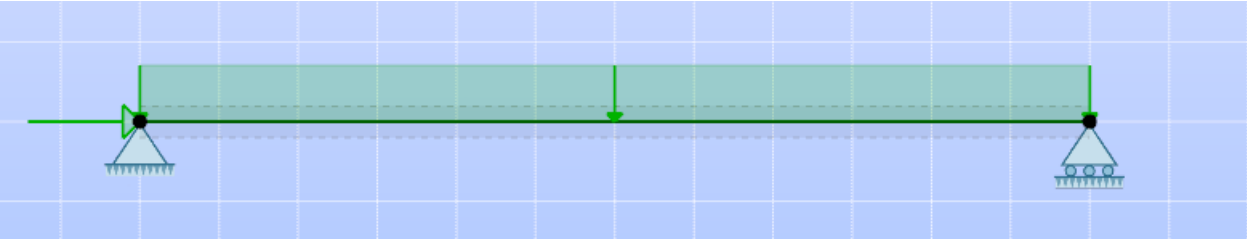
Dersom man deretter gikk videre til å bruke andre analyser oppstod det derfor problemer med minnet i beregningskjernen siden det ikke var tomt. Løsningen på dette problemet ble at absolutt alle analyser i **fap2D** nå tømmer minnet ved oppstart av analysen. For «sanntids» beregninger er det mulig å sette en variabel etter første iterasjon som gjør at beregningskjernen ikke tømmer minnet for neste iterasjon.

Alternativet til denne løsningen kunne vært at minnet tømmes når «sanntids» beregninger er ferdig. Dette ble derimot vanskelig å gjennomføre siden «sanntids» beregninger gjør en beregning for hver gang lasttoget forflyttes. Fordi man har mulighet til å gjøre så mange forflyttinger som man vil, er det umilig å vite når beregningene til «sanntids» beregninger er ferdig. Dermed vet man aldri når minnet skal tømmes.

5.3.3. Reaksjonskrefter

Reaksjonskrefter er endret til å tegnes i skala til største opptredende punktlast i alle analyser som inneholder punktlaster. Det innebærer lineær og ikkelineær analyse og «sanntids» beregninger.

Reaksjonskreftene tegnes aldri større enn punktlastene. Dette kan være tilfelle dersom både en jevnt fordelt last og en punktlast eksisterer på samme modell og den jevnt fordelte lasten fører til at reaksjonskreftene blir større enn største punktlast, jamfør Figur 22. Arbeidet med reaksjonskreftene er relatert til skaleringen av krefter under «sanntids» beregninger i forhold til største respons langs lastbanen og ble fikset i denne sammenhengen.



Figur 22: Eksempel på modell hvor reaksjonskrefter kan bli større enn største opptredende punktlast

5.3.4. Lisenstjeneste

Det ble forsøkt å finne en passende lisenstjener til **fap2D** blant ferdige kommersielle produkter. Da dette viste seg å bli for dyrt å kjøpe og for vanskelig å implementere selv på tiden som var til rådighet, ble det bestemt å implementere en sperre i beregningskjernen som sørger for at analysene ikke vil fungere etter en gitt tidsperiode. Funksjonaliteten ble implementert av veileder, men nevnes likevel siden noe tid har gått med på å finne alternativer for lisenstjener.

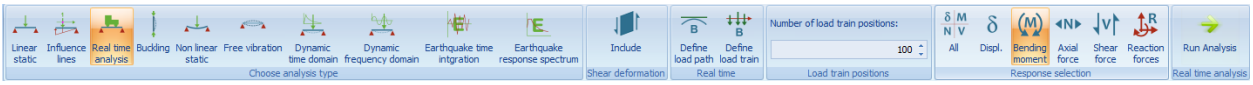
6. Bruk av nye analyser

Bruken av nye analyser som har blitt implementert eller endret på i løpet av masteroppgaven er forklart i dette kapittelet, dette innebærer knappene og dialogboksene som man skal forholde seg til og benytte seg av. Forklaringen har et brukerperspektiv i fokus siden funksjonalitet har blitt forklart i detalj tidligere. Felles for alle dialogbokser er at de kan lukkes med «Escape»-knappen, uansett situasjon.

6.1. «Sanntids» beregninger

Brukergrensesnittet for «sanntids» beregninger er delt opp i knapper for analyse og resultatvisning. Samtlige interaktive elementer under analysen har blitt implementert i løpet av denne masteroppgaven.

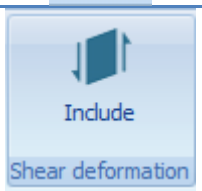
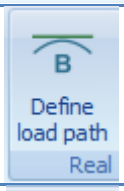
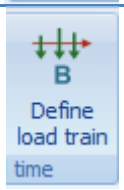
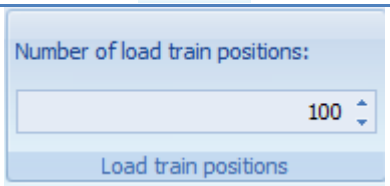
6.1.1. Analyse

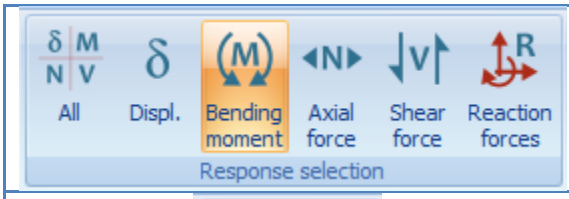
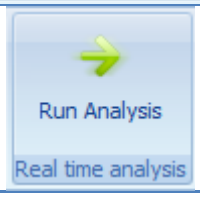


Figur 23: Oversikt over valgene under analyse for «sanntids» beregninger

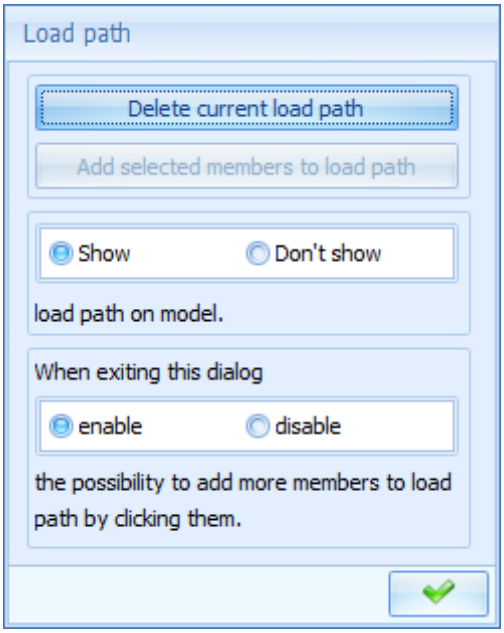
Knappene som finnes under analyse for «sanntids» beregninger er sortert som i Figur 23. En nærmere forklaring av hver enkelt knapp finnes i Tabell 3.

Tabell 3: Forklaring av knappene under analyse i «sanntids» beregninger

Knapp	Beskrivelse
	Gjør innstillingene for «sanntids» beregninger synlige. Valg av en annen analyse gjør at innstillingene forsvinner og blir byttet ut med den respektive analysens innstillinger.
	Inkluderer skjærdeformasjoner i beregningene. Standard innstilling er at skjærdeformasjoner ikke er inkludert.
	Åpner dialogboksen for lastbane for «sanntids» beregninger.
	Åpner dialogboksen for lasttog for «sanntids» beregninger.
	Setter antallet lasttogposisjoner for beregningsmodellen. Verdien lagres med modellen og blir husket selv om skifter mellom faner. Standard inndeling er 100 posisjoner.

	Velger hvilken type respons man vil se på under resultatvisningen. Man kan velge en av de fem responstypene som er vist. Dersom ingen respons er valgt blir momentbetraktninger valgt som standard.
	Starter analysen og gjennomfører et søk etter de mest ekstreme responsene for det aktive toget. Lasttoget kan vises under resultatfanen når beregningene er over.

Ved valg av lastbane for «sanntids» beregninger vises dialogboksen i Figur 24.



Figur 24: Dialogboks for lastbane for «sanntids» beregninger

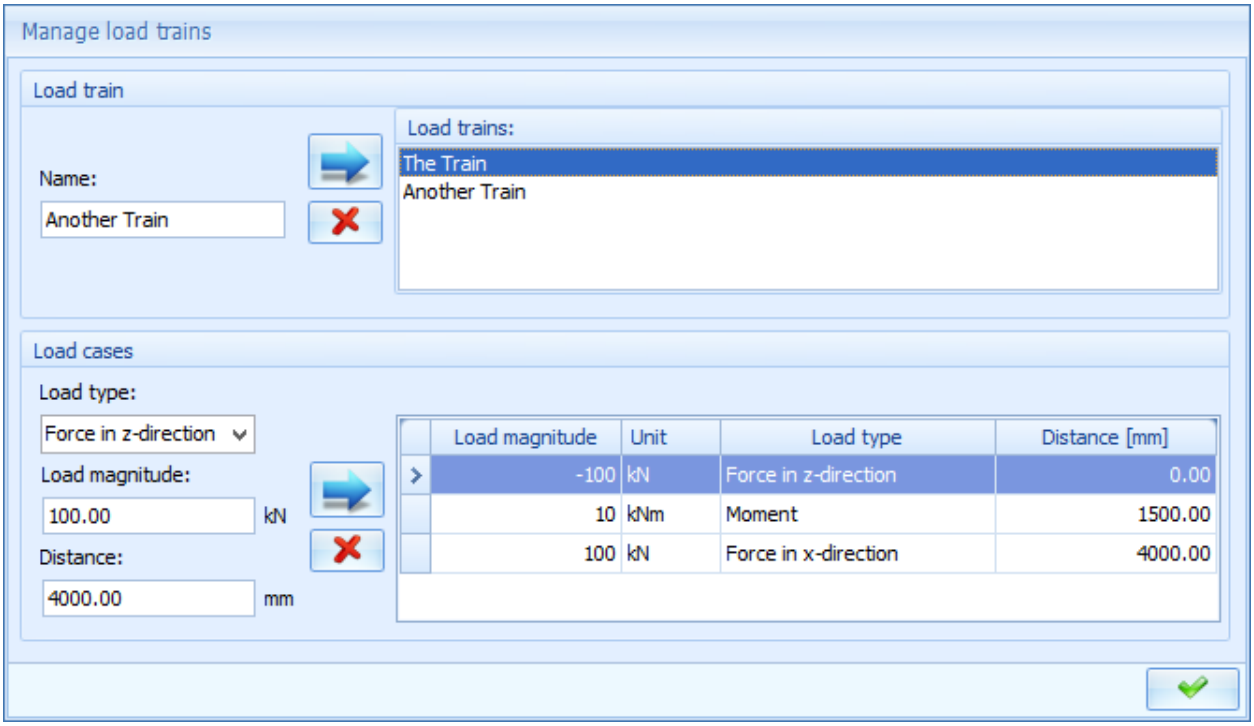
Dialogboksen er nokså lik den samme dialogboksen man kjenner fra lastbane for influenslinjer. Forskjellen er at man i dette tilfellet kun har fire valg. De samme fire valgene eksisterer også for influenslinjer og har samme effekt på modellen med hensyn på å lage lastbaner. På denne måten er det mulig å lage lastbaner som fungerer både for «sanntids» beregninger og influenslinjer.

Dersom man allerede har merket ut hvilke *members* man vil ha med i lastbanen før dialogboksen åpnes blir knappen «Add selected members to load path» aktiv. Trykker man på denne blir de valgte *members* lagt til og man forlater da dialogboksen.

Dersom det allerede eksisterer en lastbane på modellen blir knappen «Delete current load path» aktiv og ved å trykke på denne fjernes lastbanen fra modellen. Man forlater ikke dialogboksen ved å trykke på denne knappen.

På den øverste radioknappgruppen kan man velge hvorvidt man skal vise eller ikke vise lastbanen på modellen. En eventuell lastbane eksisterer uansett på modellen. På den nederste gruppen velger man om man skal få muligheten til å legge til nye *members* i lastbanen enkeltvis når man forlater dialogboksen. Ingen av endringer i gruppene tar effekt før man forlater dialogboksen via den grønne «OK»-knappen.

Har man ikke lenger behov for å legge til *members* i lastbanen, kan man tilbakestille funksjonen ved å velge «Neutral pointer» i verktøylinjen til høyre i **fap2D**.



Figur 25: Dialogboks for lasttog for «sanntids» beregninger

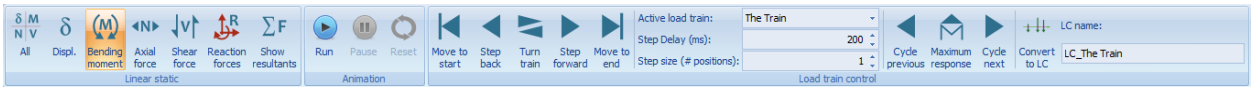
Dialogboksen for lasttog for «sanntids» beregninger i Figur 25 kan kun brukes i denne analysen, da den har flere funksjoner enn den samme for influenslinjer. Man kan legge til så mange lasttog man vil i isten over lasttog, så lenge de har et navn.

Når man velger et lasttog i listen får man muligheten til å legge til laster i toget. Laster blir lagt til så lenge de har størrelse ulik 0 og positiv avstand fra togets lokomotiv.

Man kan legge til og fjerne lasttog og laster via sine respektive tillegg- og fjerneknapper (blå pil og rødt kryss). Alternativt kan man legge til nye instanser av begge deler ved å trykke «enter» når man har skrevet inn verdiene. Dette gjelder også dersom listen for lasttyper har fokus i vinduet.

Lasttogene med laster blir lagret til modellen når man forlater dialogboksen via den grønne «OK»-knappen.

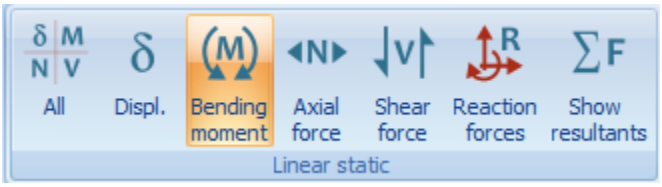
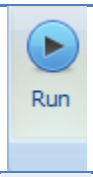
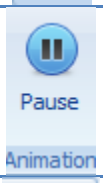
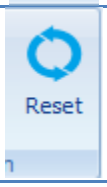
6.1.2. Resultater

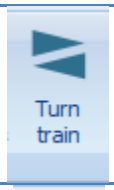
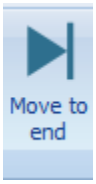
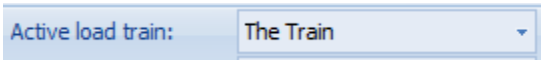
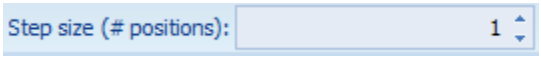


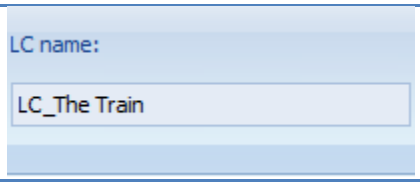
Figur 26: Oversikt over knappene under resultater for «sanntids» beregninger

Knappene som finnes under resultater for «sanntids» beregninger er sortert som i Figur 26. En nærmere forklaring av hver enkelt knapp finnes i Tabell 4.

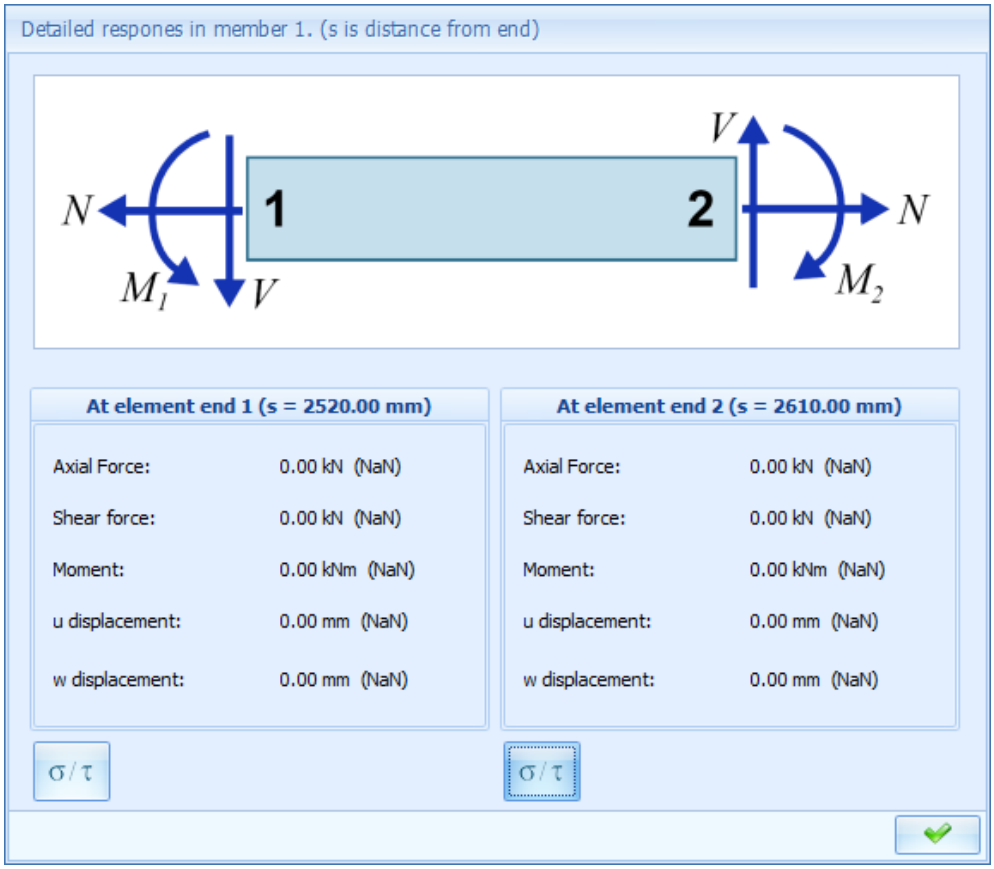
Tabell 4: Forklaring av knappene under resultater for «sanntids» beregninger

Knapp	Beskrivelse
	Velger hvilken type respons man vil se på. Man kan velge en av de fem responstypene som er vist. Velger man derimot «Show resultants», vises en dialogboks med resultantkreftene i modellen for den gitte plasseringen av lasttoget. Valgene i denne gruppen av knapper er de samme som under vanlig lineær og ikkelineær analyse. Alle knappene er aktive når en animasjon kjører.
	Starter animasjonen av lasttoget fra dets plassering. Knappen er deaktivert når lasttoget ikke lenger kan fortsette i den retningen den kjører i.
	Pauser animasjonen av lasttoget. Knappen er deaktivert når det ikke kjøres en animasjon.
	Flytter lasttoget tilbake til sin opprinnelige konfigurasjon ved starten av analysen, med kjøreretning venstre mot høyre.
	Flytter lasttoget til starten av lastbanen, avhengig av kjøreretning. Dersom kjøreretningen er venstre mot høyre plasseres lokomotivet lengst til venstre langs lastbanen. Dersom kjøreretningen er høyre mot venstre plasseres lokomotivet lengst til høyre langs lastbanen. Knappen er ikke aktiv når en animasjon kjører.
	Rygger fremste enhet i lasttoget et gitt antall steg bakover i forhold til togets kjøreretning. Knappen er ikke aktiv når en animasjon kjører.

	Vender lasttoget om lokomotivet i dets plassering. Knappen er ikke aktiv når en animasjon kjører.
	Flytter fremste enhet lasttoget et gitt antall steg fremover i forhold til togets kjøreretning. Knappen er ikke aktiv når en animasjon kjører.
	Flytter lasttoget til enden avlastbanen, avhengig av kjøreretning. Dersom kjøreretningen er venstre mot høyre plasseres bakerste last lengst til høyre på lastbanen. Dersom kjøreretningen er høyre mot venstre plasseres bakerste last lengst til venstre på lastbanen. Knappen er ikke aktiv når en animasjon kjører.
	Veksler mellom hvilket lasttog som skal være aktivt og vises i resultatvinduet. Beregner også ekstreme responser for det valgte toget, dersom det ikke finnes fra før. Knappen er ikke aktiv når en animasjon kjører.
	Setter forsinkelsen mellom hver beregning under animasjonen. Knappen er ikke aktiv når en animasjon kjører.
	Setter antallet posisjoner som lasttoget skal forflytte seg over for hvert steg. Antallet tilsvarer inkrementet i antallet indekser i listen over lasttogposisjoner. Knappen er ikke aktiv når en animasjon kjører.
	Veksler til forrige ekstreme respons i innenfor den nåverende responstypen, gitt at det er en tidligere respons. Lasttoget forflyttes deretter til plasseringen som gir den ekstreme responsen. Denne knappen er ikke aktiv når alle responstyper vises samtidig. Knappen er ikke aktiv når en animasjon kjører.
	Forflytter lasttoget til plasseringen der den første ekstreme responsen langs lastbanen opptrer. Denne knappen er ikke aktiv når alle responstyper vises samtidig eller når en animasjon kjører.

	Veksler til neste ekstreme respons i innenfor den nåverende responstypen, gitt at det er en kommende respons. Lasttoget forflyttes deretter til plasseringen som gir den ekstreme responsen. Denne knappen er ikke aktiv når alle responstyper vises samtidig. Knappen er ikke aktiv når en animasjon kjører.
	Gjør om lasttogets plassering til et lasttilfelle som kan brukes i andre analyser. Kun laster som er til stede på lastbanen blir tatt med i lasttilfellet. Når lasttilfellet er lagret, får brukeren en bekreftelse i informasjonsvinduet og lasttilfellet blir valgbart under «Loading»-fanen. Lasten blir lagt til på konstruksjonsmodellen, med en node under lasten. Knappen er ikke aktiv når en animasjon kjører.
	Setter navnet på lasttilfellet. Lasttilfellet må ha et navn for å kunne lagres. Knappen er ikke aktiv når en animasjon kjører.

Dersom man høyreklikker et element i modellen under resultatvisningen for «sanntids» beregninger kommer dialogboksen i Figur 27 frem. Samtidig blir det valgte elementet merket i modellen med en rød sirkel. Elementkreftene i det valgte elementet vises i denne dialogboksen. Oppsettet av boksen er lik som i blant annet lineær og ikkelineær analyse i **fap2D**.

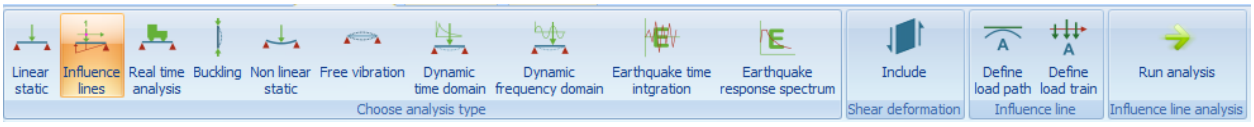


Figur 27: Dialogboks for elementkrefter for «sanntids» beregninger

6.2.Influenslinjer

Brukergrensesnittet for influenslinjer er også delt opp i knapper for analyse og resultatvisning. Mange av de interaktive elementene i analysen eksisterte i forkant av denne masteroppgaven. Det er hensiktsmessig å nevne elementene her siden funksjonaliteten er utvidet og flere av de tidligere har blitt forbedret. I tillegg er det nødvendig å forklare forskjellen i bruk mellom influenslinjer og «sanntids» beregninger og bruk av forbedrede funksjoner i samspill med de eksisterende.

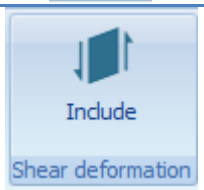
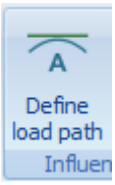
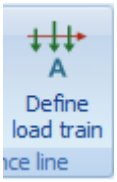
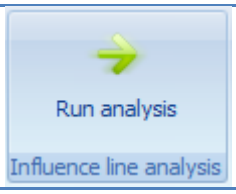
6.2.1. Analyse



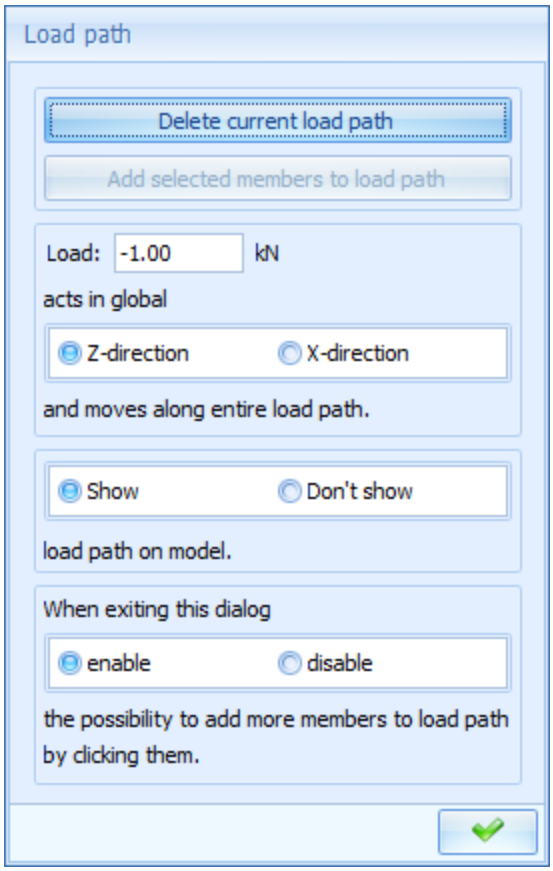
Figur 28: Oversikt over knappene under analyse for influenslinjer

Knappene som finnes under analyse for influenslinjer er sortert som i Figur 28. En nærmere forklaring av hver enkelt knapp finnes i Tabell 5.

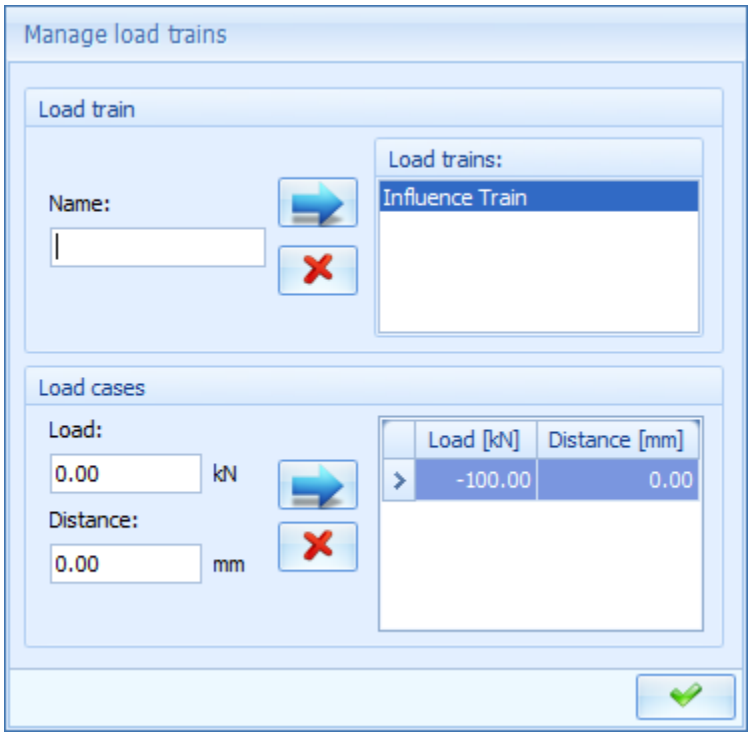
Tabell 5: Forklaring av knappene under analyse for influenslinjer

Knapp	Beskrivelse
	Gjør innstillingene for influenslinjer synlige. Valg av en annen analyse gjør at innstillingene forsvinner og blir byttet ut med den respektive analysens innstillinger.
	Inkluderer skjærdeformasjoner i beregningene. Standard innstilling er at skjærdeformasjoner ikke er inkludert.
	Knappen har blitt endret fra tidligere. Bokstaven A har blitt tilegnet influenslinjer og B har blitt tilegnet «sanntids» beregninger. Dette er for at brukere skal kunne se forskjell mellom de to funksjonene, selv om lastbaner kan gjenbrukes mellom analysene. Knappen åpner dialogboksen for lastbane for influenslinjer.
	Knappen har blitt endret fra tidligere. Bokstaven A har blitt tilegnet influenslinjer og B har blitt tilegnet «sanntids» beregninger. På denne knappen er det viktigere å markere at de to analysene er forskjellige, fordi lasttog for influenslinjer kan ikke brukes i «sanntids» beregninger og omvendt. Knappen åpner dialogboksen for lasttog for influenslinjer.
	Starter analysen. Resultatet kan vises under resultatfanen når beregningene er over. Det er kun nødvendig å ha definert en lastbane og en responsparameter på modellen for å kunne starte analysen.

Dialogboksen i Figur 29 har de samme funksjonene som den tilsvarende for «sanntids» beregninger. I tillegg til en funksjon for størrelsen på influenslasten og en funksjon for lastens retning. Virkemåten har også blitt endret slik at ferdige lastbaner som blir laget av denne dialogboksen også kan brukes i «sanntids» beregninger.



Figur 29: Dialogboks for lastbane for influenslinjer



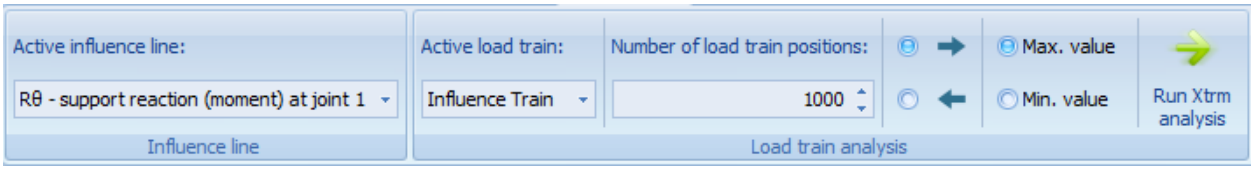
Figur 30: Dialogboks for lasttog for influenslinjer

Dialogboksen for lasttog for influenslinjer i Figur 30 har samme metode for å legge til lasttog og laster som «sanntids» beregninger. Den har også blitt endret slik at man kan bruke enter som hurtigtast for å legge til nye instanser og de samme metodene for validering av laster og avstander til lokomotivet.

Grunnen til at lasttogene under influenslinjer ikke kan brukes i begge analyser er at lastene i togene under influenslinjer ikke kan ha individuelle retninger. Lastene må alle virke i samme retning som influenslasten som ble definert i dialogboksen for lastbane for influenslinjer. Dette står i kontrast til lastene under «sanntids» beregninger, som kan virke i hver sin retning.

Lasttogene lagres til modellen når man forlater dialogboksen via den grønne «OK»-knappen.

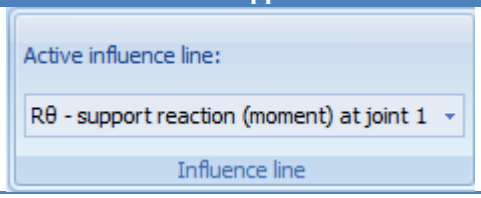
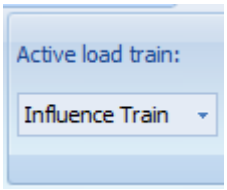
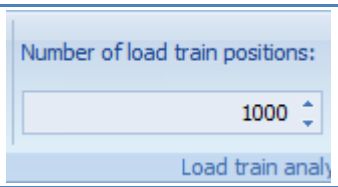
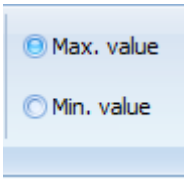
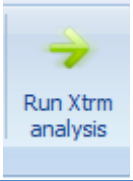
6.2.2. Resultater

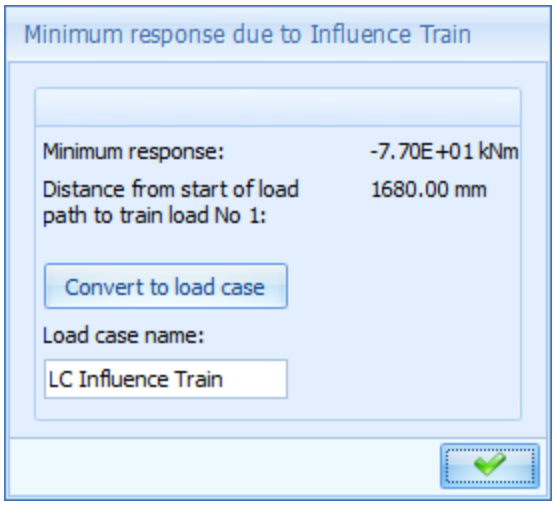


Figur 31: Oversikt over knappene under resultater for influenslinjer

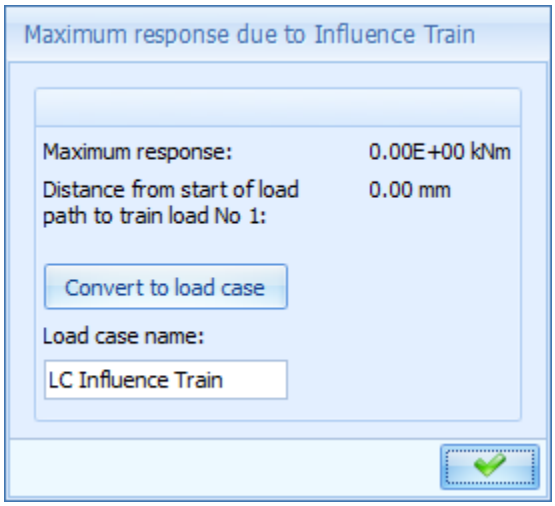
Knappene som finnes under resultater for influenslinjer er sortert som i Figur 31. En nærmere forklaring av hver enkelt knapp finnes i Tabell 6.

Tabell 6: Forklaring av knappene under resultater for influenslinjer

Knapp	Beskrivelse
	Veksler mellom hvilken responsparameter som skal betraktes.
	Veksler mellom hvilket lasttog som skal brukes for gjennomføring av analyse for ekstreme responser. Funksjonen har blitt endret til å ta høyde for hvilken ekstrem respons som skal betraktes når et nytt tog blir valgt. Når man velger et nytt tog skal alltid den ekstreme responsen som til en hver tid er merket av betraktes under analyse for ekstreme responser (enten maksimum eller mimum).
	Funksjonen har blitt lagt til i løpet av denne masteroppgaven. Tidligere var det ikke mulig å definere antallet lasttogposisjoner for influenslinjer. Minimum antall er 2 posisjoner og maksimalt er 2 000 000.
	Velger kjøreretning for lasttoget under analyse for ekstreme responser. Samme lasttog kan gi ulike resultater avhengig av kjøreretning.
	Velger hvilken ekstrem respons som skal betraktes under analyse for ekstrem respons. Enten den største (mest positive), eller den minste (mest negative) responsen innenfor responsparameteren på lastbanen. Fortegn er medregnet i denne funksjonen.
	Starter en analyse for ekstrem respons basert på hvilken responsparameter og hvilket lasttog som er valgt. Når analysen er over, åpnes dialogboksen for ekstrem respons. Knappen er ikke aktiv dersom ingen lasttog har blitt definert.



Figur 32: Dialogboks for minste respons for lasttog for influenslinjer



Figur 33: Dialogboks for største respons for lasttog under influenslinjer

Dialogboksen for minste og største respons i Figur 32 og Figur 33 er den samme boksen, dialogboks for ekstrem respons, men med ulik tekst avhengig av hvilken av responsene som betraktes. I denne boksen kan man se informasjon om den ekstreme responsen og avstanden fra fremste last til starten av lastbanen.

Samtidig som dialogboksen åpnes plasseres lasttoget i modellvinduet i posisjonen langs lastbanen som gir den ekstreme responsen. Dersom man velger å lagre den ekstreme responsen som et lasttilfelle, er det denne plasseringen som blir lagret på samme måte som under «sanntids» beregninger. Lasttilfellet må også ha et navn.

Dialogboksen og funksjonen har blitt endret i løpet av masteroppgaven. Teksten er blitt endret for å gjøre det enklere å skille mellom største og minste respons. I tillegg blir lasttoget tegnet riktig i modellvinduet med korrekt retning og størrelse på lastene.

«OK»-knappen i denne boksen har kun som funksjon å lukke den.

7. Testing

Virkemåten til analysene som har blitt implementert har blitt testet for å sannsynliggjøre at de fungerer korrekt.

7.1. «Sanntids» beregninger

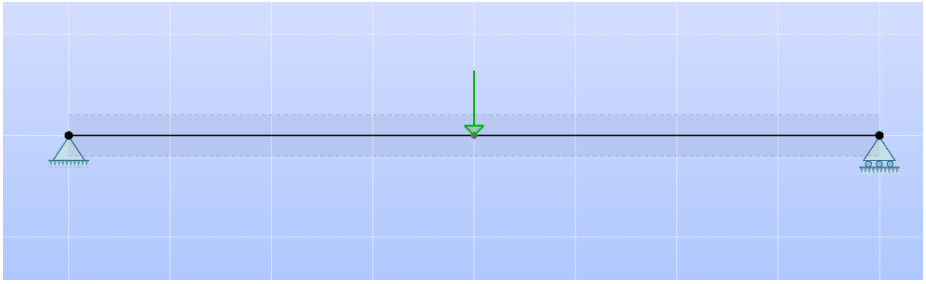
Analysen testes mot lineær statisk analyse siden denne benytter seg av samme rutine i **Frame2D** og gir dermed et godt sammenligningsgrunnlag i forhold til nøyaktighet og tidsbruk. I tillegg sammenlignes den med rammeprogrammet Focus Konstruksjon 2013[4] (heretter kalt Focus), siden det er et tilsvarende og sammenlignbart program som **fap2D**, som også inneholder en analyse for bevegelige laster.

7.1.1. Nøyaktighet

For å teste nøyaktigheten til analysen benyttes en fritt opplagt bjelke påkjent av en loddrett last i midtpunktet, jamfør Figur 34. Resultatene fra momentresponsen testes mot analytisk resultat som fasit. Analytisk resultat i midten av bjelken er beregnet til å være 10 kNm etter tabellen over enkle bjelker i formelsamlingen[5].

$$M = \frac{FL}{4} = \frac{10kN * 4m}{4} = 10kNm \quad \textbf{(3)}$$

Resultatene fra lineær statisk analyse i 2D i både **fap2D** og Focus tas med til sammenligning. Selv om modellen er triviell er den likevel egnet som testeksempel til å illustrere variasjonen i nøyaktighet for «sanntids» beregninger.



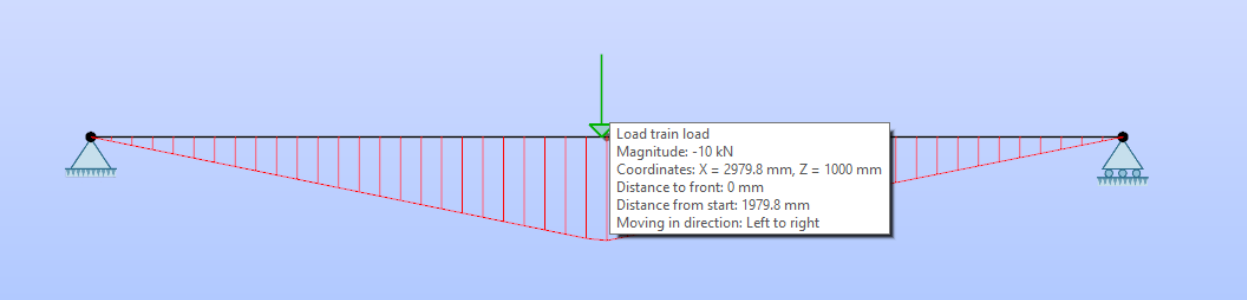
Figur 34: Fritt opplagt bjelke som brukes i nøyaktighetstesten

Følgende parametre gjelder for testen:

Tabell 7: Parametre i nøyaktighetstesten

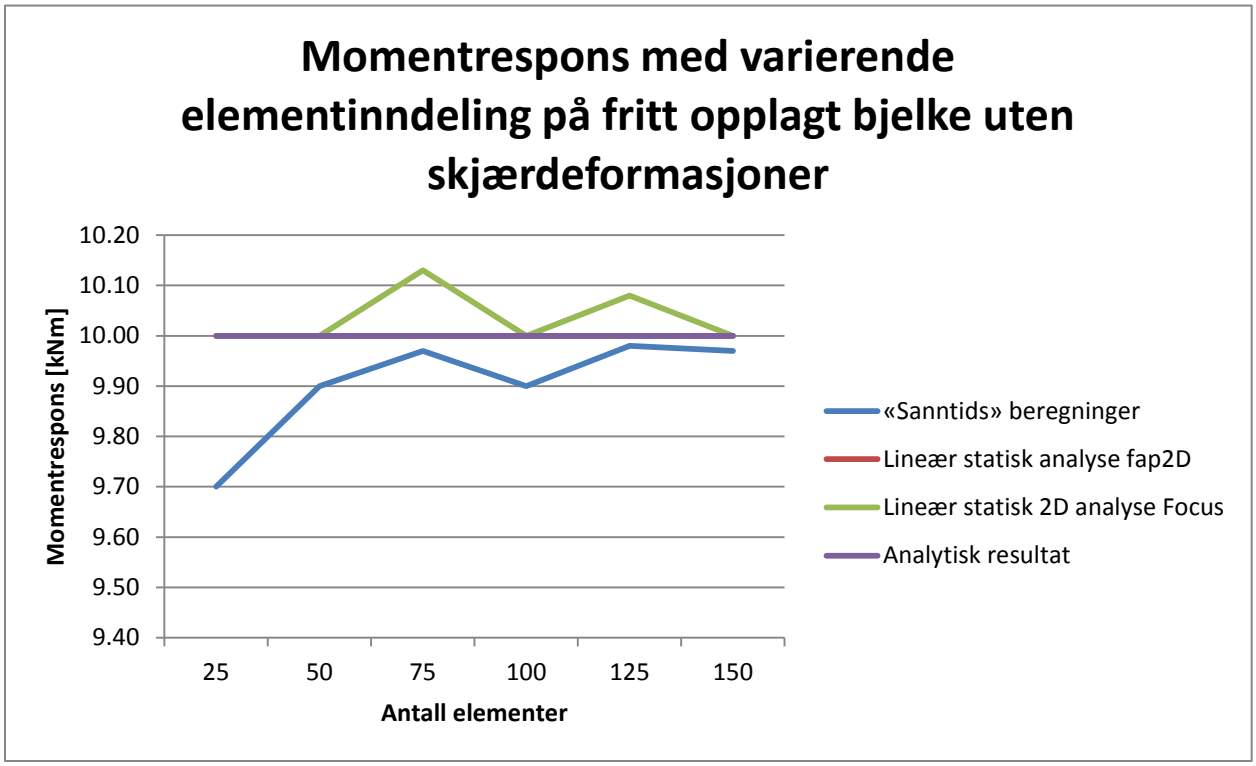
Parameter	Verdi
Bjelketverrsnitt	IPE200
Bjelkens lengde	4 000 mm
Lastens størrelse	10 kN
Materiale	Stål
Elastisitetsmodul	210 000 MPa
Skjærmodul	80 800 MPa
Antall lasttogposisjoner	100
Størrelse på lasten i lasttoget	10 kN

Testen gjennomføres med ulike elementinndelinger for å kunne betrakte resultatene i forhold til analytisk resultat etter som inndelingen endrer seg. Alle resultater er gitt i kNm. Resultatene fra «sanntids» beregninger er funnet i posisjonen hvor lasttoget er nærmest midtpunktet med det gitte antallet lasttogposisjoner. Denne posisjonen er 1979.8 mm fra venstre ende av bjelken, jamfør Figur 35. For de to andre analysene er lasten plassert nøyaktig i midtpunktet. Avviket mellom resultatene fra «sanntids» beregninger og lineær statisk analyse i **fap2D** beregnes på slutten. Skjærdeformasjoner er ikke inkludert i noen av analysene. Standard innstillinger er brukt i Focus utenom endringene i elementinndeling og ekskluderingen av skjærdeformasjoner.



Figur 35: Lasttogets nærmeste plassering i forhold til midtpunktet på bjelken

Resultatene kan sammenlignes med analytisk resultat på grafisk form i Figur 36. Verdiene som er bakgrunn for testen finnes i Tabell 18 i vedlegg J.



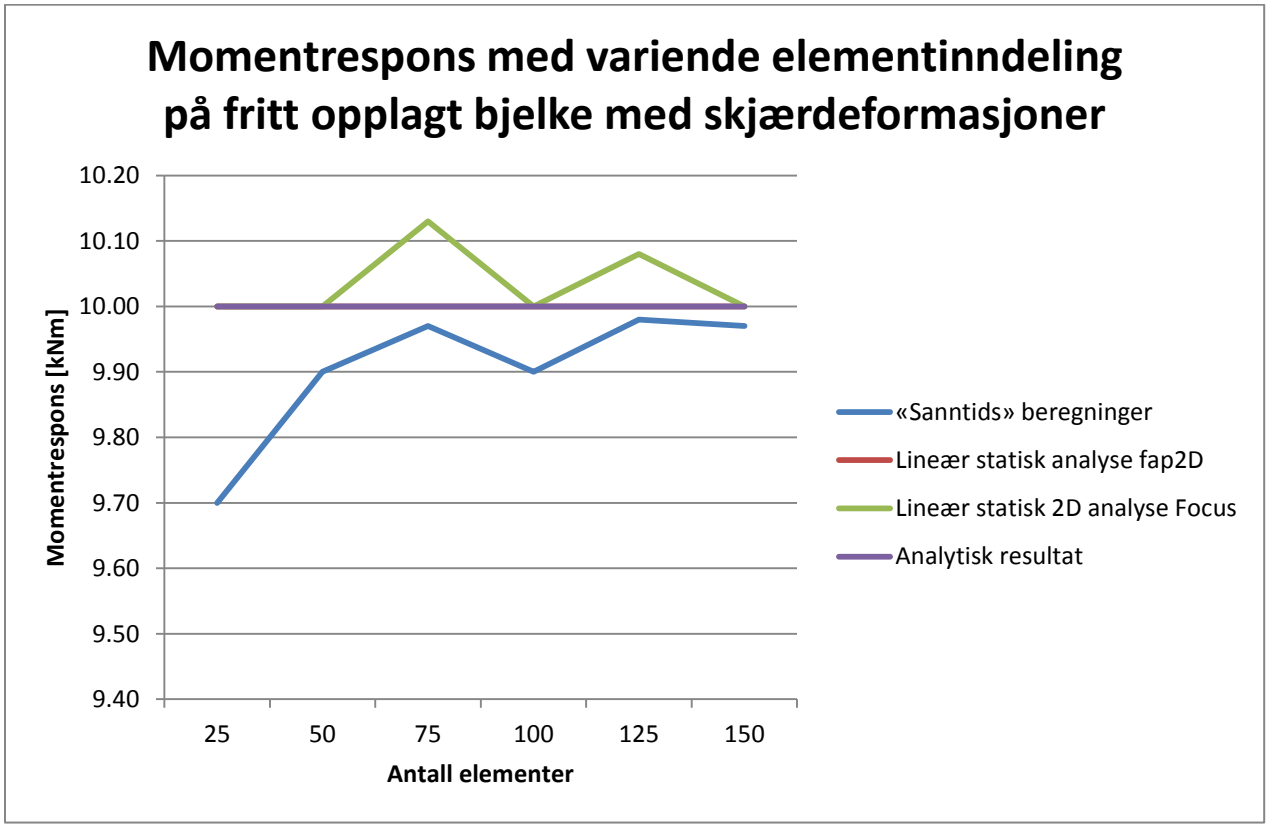
Figur 36: Variasjonen i momentrespons i analysene etter elementinndeling uten skjærdeformasjoner

Resultatene fra lineær statisk analyse i **fap2D** gir samme resultater som analytisk beregning for alle elementinndelinger. Linjene for disse resultatene overlapper dermed hverandre. Resultatene fra lineær statisk analyse i Focus varierer etter hvorvidt elementinndelingen fører til en jevn inndeling langs bjelken og overestimerer responsen i forhold til analytisk resultat i tilfellene hvor dette skjer.

«Sanntids» beregninger har mye av de samme utfordringene som Focus sin analyse, men «sanntids» beregninger får bedre resultater ved oddetallsinndelinger. Dette er fordi inndelingen fører til at nodene det interpoleres til havner nærmere midtpunktet av bjelken enn for partallsinndelinger av bjelken. Med oddetallsinndeling vil det plasseres en node i midtpunktet av bjelken og påfølgende noder vil da være nærmere midtpunktet enn for partallsinndelinger. Dermed interpoleres det til noder som er nærmere midtpunktet og man får et mer nøyaktig resultat.

Trenden i grafen er at resultatene fra «sanntids» beregninger konvergerer mot analytisk resultat og er best for oddetallsinndelinger. Desto finere oddetallsinndeling, desto nærmere analytisk resultat kommer man. Likevel er resultatene for «sanntids» beregninger alltid innenfor et 4 % avvik fra responsen i lineær statisk analyse for **fap2D**, selv om lasten i lasttoget ikke er plassert nøyaktig i bjelkens midtpunkt. Avviket beregnes i Tabell 18 i vedlegg J.

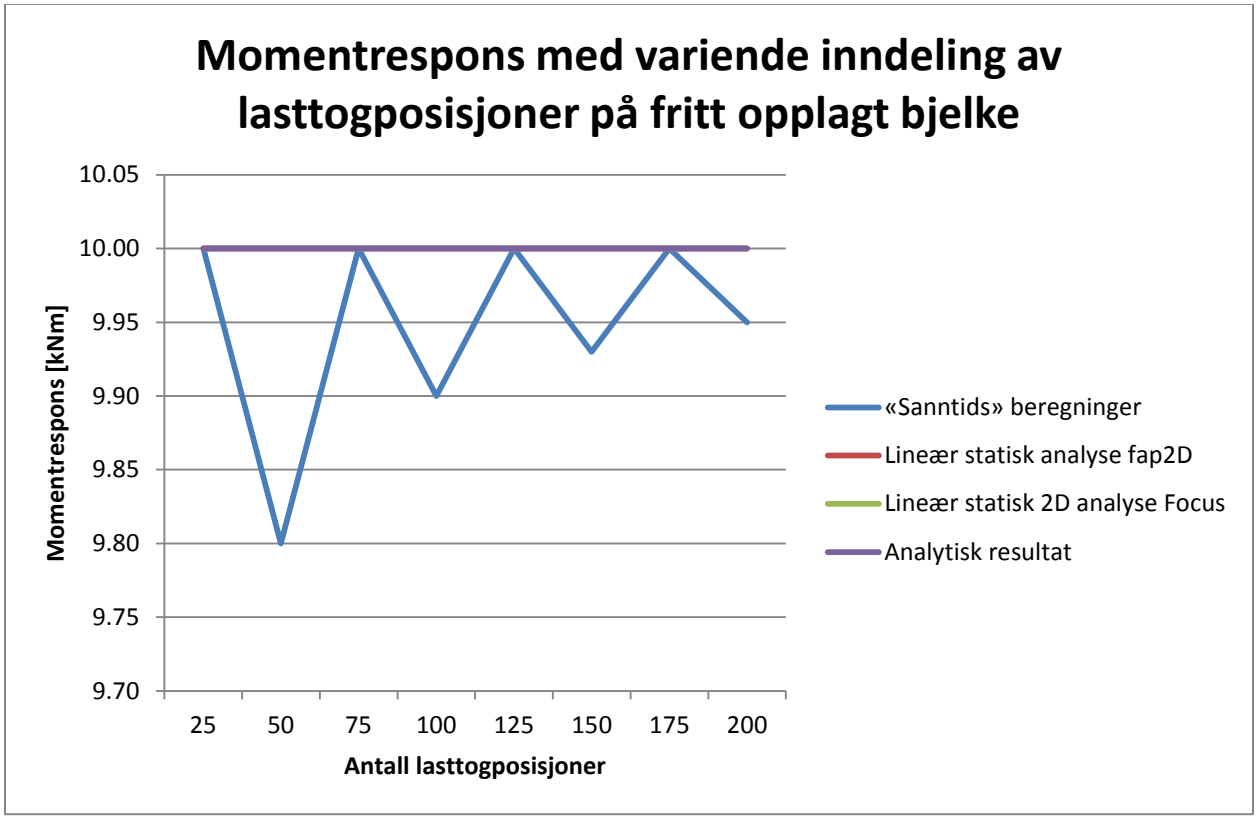
Inkludering av skjærdeformasjoner i analysene gir ingen forskjell i resultatene for noen av analysene i testen, som vist i Figur 37.



Figur 37: Variasjonen i momentrespons i analysene etter elementinndeling med skjærdeformasjoner

Antallet lasttogposisjoner som ble brukt i de foregående testene påvirket hvor nært midtpunktet lasten i lasttoget kunne være. Derfor er det interessant å betrakte effekten på resultatene for momentresponsen ettersom antallet lasttogposisjoner endres og elementinndelinger er konstant. I testen for lasttogposisjoner er antallet elementer konstant lik standarden i **fap2D** på 50 elementer og responsen i punktet nærmest midtpunktet av bjelken måles for «sanntids» beregninger.

Ellers gjelder de samme parameterne som i tidligere testene og samme modell blir brukt. «Sanntids» beregninger testes også mot de samme analysene. Skjærdeformasjoner er ikke inkludert. Avstanden fra lasten i lasttoget til venstre ende av bjelken når lasten er i punktet nærmest mulig midtpunktet registreres også for hver inndeling av lasttogposisjoner, jamfør Tabell 20 i vedlegg J.



Figur 38:Variasjonen i momentrespons etter inndeling av lasttogposisjoner

Resultatene fra alle analyser med unntak av «sanntids» beregninger er identisk lik det analytiske resultatet, jamfør Figur 38. Når antallet lasttogposisjoner er et oddetall får «sanntids» beregninger nøyaktig resultat for dette eksempelet. Grunnen til dette er at inndelingen gjør at en lasttogposisjon havner nøyaktig i midtpunktet av bjelken. I tillegg er antallet elementer et partall, som gjør at en node også blir plassert i midtpunktet. Dermed interpoleres alt av lasten til noden i midtpunktet når lasten står midt på bjelken. Nøyaktig resultat er derfor oppnåelig når antall lasttogposisjoner er oddetall og antall elementer er partall.

Selv om partallsinndelinger av lasttogposisjoner ikke gir nøyaktige resultater i denne testen konvergerer resultatene relatert til disse inndelingene mot et mer nøyaktig resultat ettersom antallet lasttogposisjoner økes. I tillegg konvergerer avstanden fra venstre ende til lastens posisjon mot avstanden fra venstre ende til midtpunktet for disse inndelingene. Avviket fra nøyaktig resultat er innenfor et 3 % avvik for alle resultater i denne testen. Oversikten over avvik og avstander fra venstre ende finnes i Tabell 20 i vedlegg J.

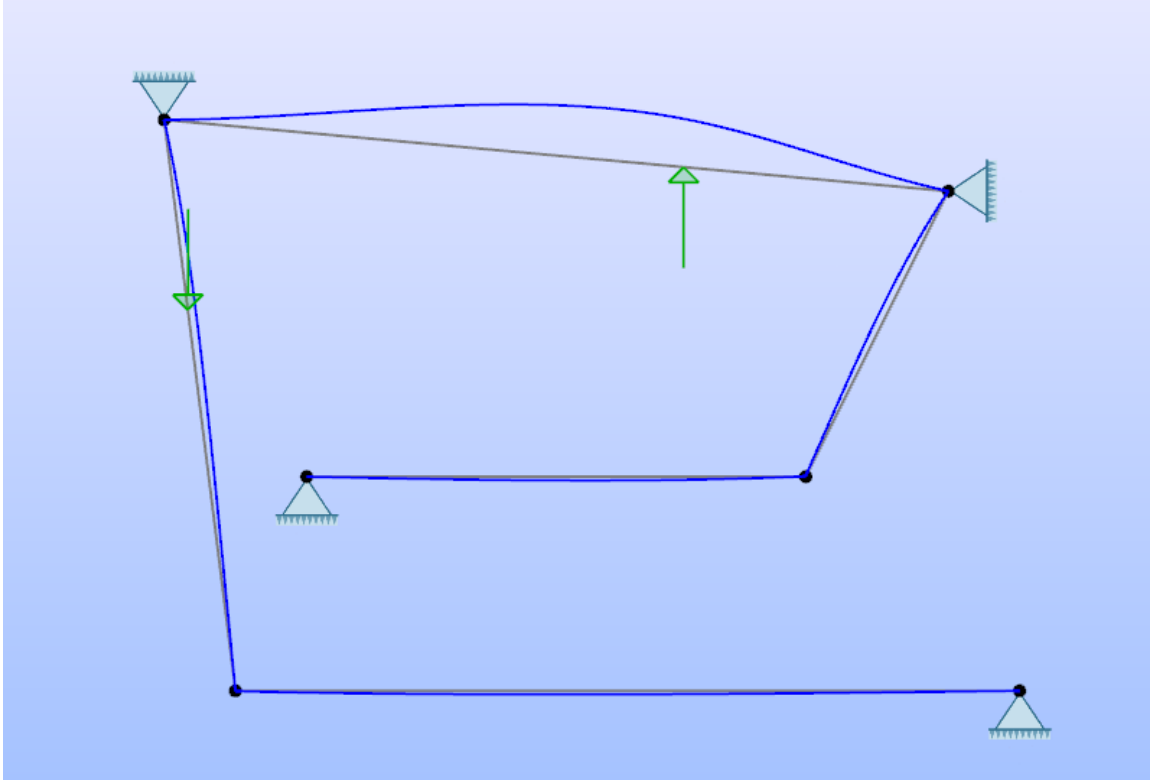
I stedet for å bruke den lineære statiske analysen i Focus for å sammenligne resultater med «sanntids» beregninger kan den tilsvarende 2D lasttoganalysen «Load train analysis» i Focus benyttes. Analysen benytter seg også av en serie lineært statiske analyser ettersom et lasttog beveger seg langs en lastbane, men skiller seg fra «sanntids» beregninger ved at den for eksempel ikke benytter seg av interpolering.

Lasttoganalysen i Focus har derimot en elementinndeling som sørger for at det alltid er en node under hver last i lasttoget ettersom det beveger seg langs lastbanen. Dermed er det aldri interpolerte resultater som blir presentert for brukeren for hver plassering toget er i.

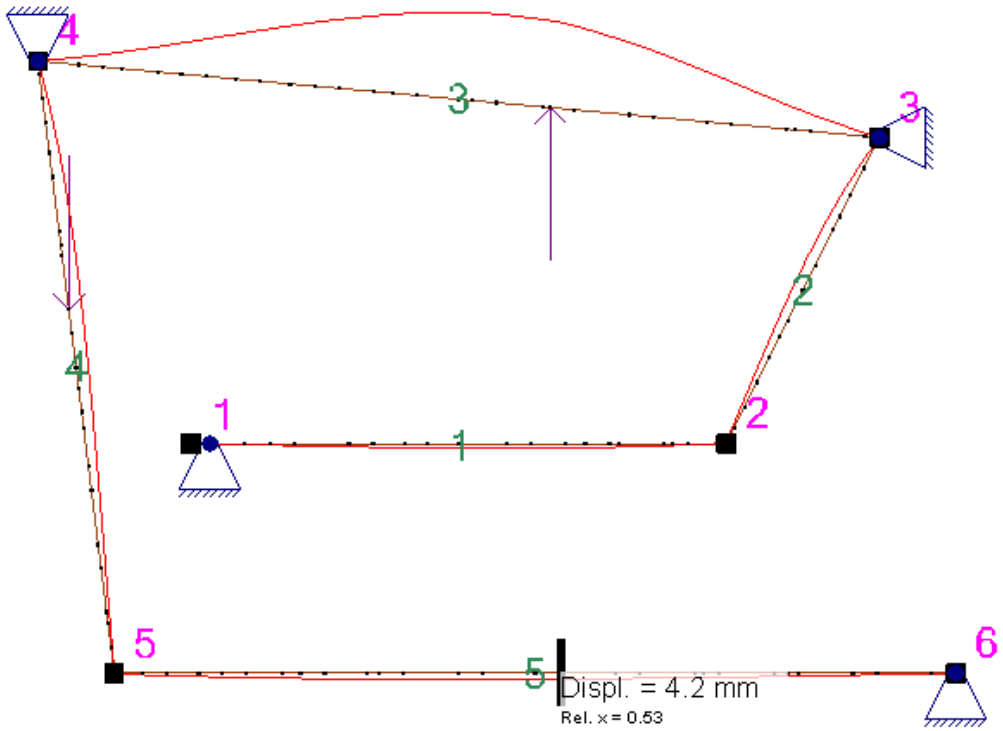
Siden formålet med nøyaktighetstesten er å sammenligne responsene i et enkelt punkt i midten av bjelken, er det ikke nødvendig å benytte seg av lasttoganalysen i stedet for den lineært statiske analysen i Focus. Dette er fordi lasten ikke beveger seg og uansett vil ha en node under lasten, selv om «sanntids» beregninger er funksjonelt mer lik lasttoganalysen.

7.1.2. Lastbaner

«Sanntids» beregninger har blitt implementert for å kunne håndtere lastbaner som ikke nødvendigvis representerer fysiske problemstillinger, som for eksempel baner som går loddrett eller går rundt startposisjonen. Slike egenskaper finnes også i kommersielle produkter som lasttoganalysen til Focus og motivasjonen er at «sanntids» beregninger skal klare de samme type lastbaner. Figur 39 og Figur 40 er stillbilder av tester på vanskelige lastbaner i både «sanntids» beregninger og lasttoganalysen til Focus.



Figur 39: Stillbilde fra testing av lastbane for «sanntids» beregninger



Figur 40: Stillbilde fra testing av 2D lasttoganalyse i Focus[4]

Modellene er fullstendig like i begge programmer, men det er unødvendig å referere til parameterne i testen siden kun lasttogets evne til å følge lastbanen testes. I begge tilfeller inkluderer lastbanen alle *members* i modellen og lasttoget starter i enden som er lengst til venstre på modellen. Modellene inneholder også de fleste orienteringer *members* på modellen kan ha og er dermed nokså dekkende for de fleste lastbanetilfeller. Lasttogene kjørte gjennom hele lastbanen og testingen viste at «sanntids» beregninger klarer avanserte lastbaner på lik linje med Focus.

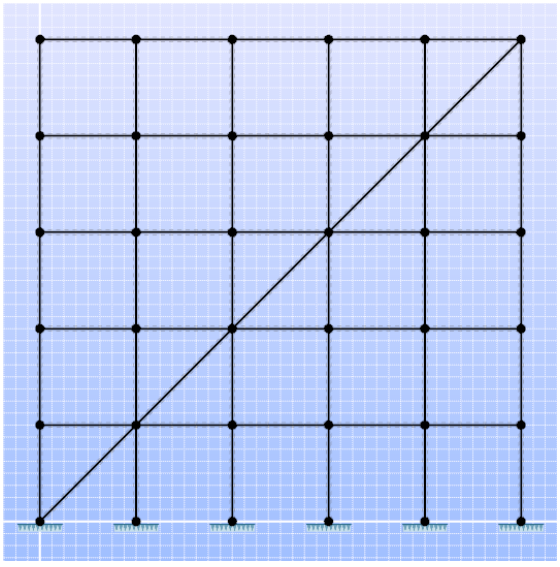
7.1.3. Tidsbruk

For hver beregning av en ny lasttogposisjon i «sanntids» beregninger blir den lagrede faktoriserte stivhetsmatrisen fra første beregning gjenbrukt. Det er derfor interessant å sjekke om tidsgevinsten som ble funnet i høstens prosjektoppgave[1] er tilstede i implementasjonen. Dette kan testes ved å betrakte oppstartstidene for henholdsvis lineær statisk analyse i **fap2D** og «sanntids» beregninger. Disse tidene blir skrevet ut til informasjonsvinduet ved starten av hver analyse.

Testen gjennomføres på en rammemodell med fem etasjer med mange koblinger for å sikre at ligningssystemet er tilstrekkelig stort til å gi utslag i tidsbruken, jamfør Figur 41. Lastbanen går langs alle de øverste bjelkene på modellen. Følgende parametre gjelder for testen:

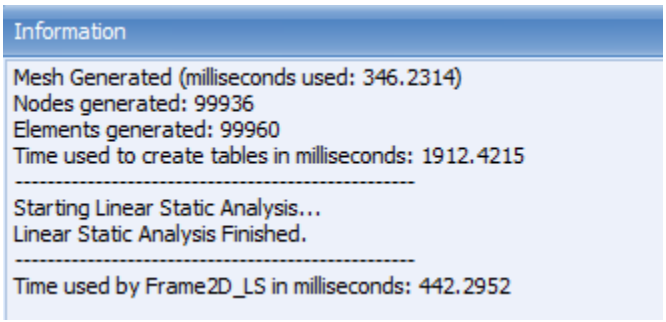
Tabell 8: Parametre i testen for tidsbruk

Parameter	Verdi
Tverrsnitt for alle bjelker	IPE200
Antall elementer per *member*	1 666
Antall *members*	60
Antall elementer totalt	99 960
Lengde for alle horisontale og vertikale *members*	4 000 mm
Materiale	Stål
Elastisistetsmodul	210 000 MPa
Antall lasttogposisjoner	100



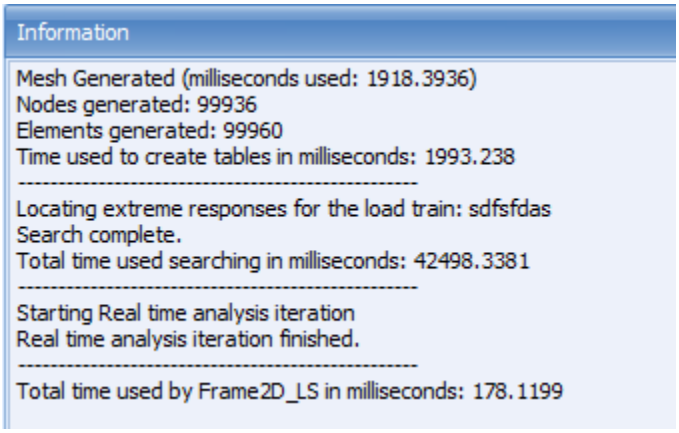
Figur 41: Rammemodell brukt i testen for tidsbruk

Lineær statisk analyse gjennomfører kun ett kall til metoden Frame2D_LS, som ikke benytter seg en tidligere lagret faktorisert stivhetsmatrise siden alle metoder tømmer minnet i beregningskjernen ved oppstart. Dermed har denne metoden ingen tidsgevinst ved gjenbruk.



Figur 42: Utskrift fra informasjonsvinduet etter en beregning av lineær statisk analyse i fap2D

Det ene kallet til Frame2D_LS fra lineær statisk analyse brukte 442 millisekunder på å beregne rammemodellen, jamfør Figur 42.



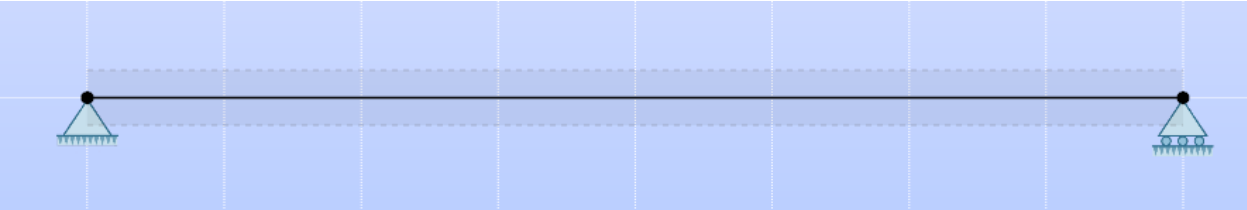
Figur 43: Utskrift fra informasjonsvinduet etter en beregning av «sanntids» beregninger

«Sanntids» beregninger gjennomfører 201 kall til Frame2D_LS uten å visualisere lasttoget på modellen i søket etter ekstreme responser og benytter seg av gjenbruk av den faktoriserte stivhetsmatrisen hver gang. Dette er med unntak av første iterasjon hvor den faktoriserte stivhetsmatrisen blir lagret i etterkant. Søket bruker 201 kall fordi man flytter lasttoget langs de 100 lasttogposisjonene i hver av kjøreretningene og til slutt plasserer lasttoget i startposisjonen med korrekt kjøreretning. «»

Tidbruken til det siste kallet til Frame2D_LS er 178 millisekunder, jamfør Figur 43. Dette er en betydelig forbedring i forhold til lineær statisk analyse og vitner om at gjenbruket av lagret faktorisert stivhetsmatrise fungerer korrekt.

7.1.4. Ekstreme responser

For å teste om metodene for å beregne ekstreme responser i «sanntids» beregninger er korrekte, sammenlignes moment- og forskyvningsresponsene i det mest utsatte punktet med analytisk verdi fra formelsamlingen[5]. Modellen som testes er den samme fritt opplagte bjelken som i nøyaktighetstesten med de samme parametre, jamfør Figur 44 og Tabell 7, siden det er enkelt å verifisere at den ekstreme responsen for bjelken er korrekt. For begge responstyper er det mest kritiske punktet i midten av bjelken.

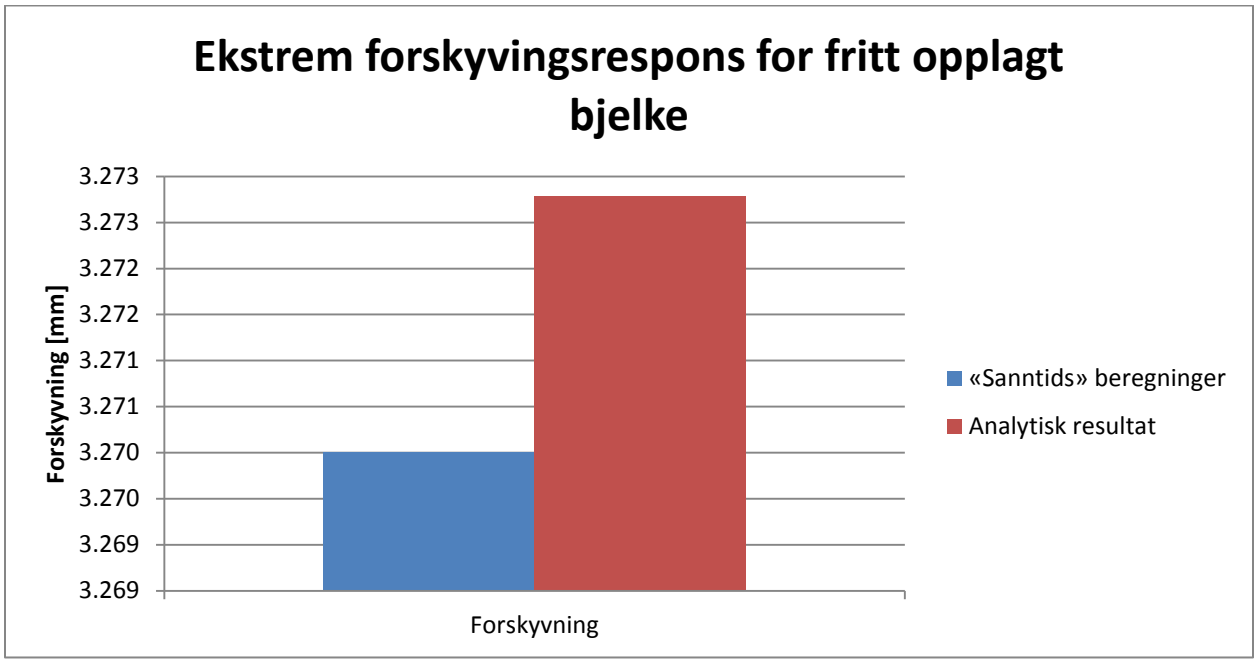


Figur 44: Modellen som brukes for å teste ekstreme responser

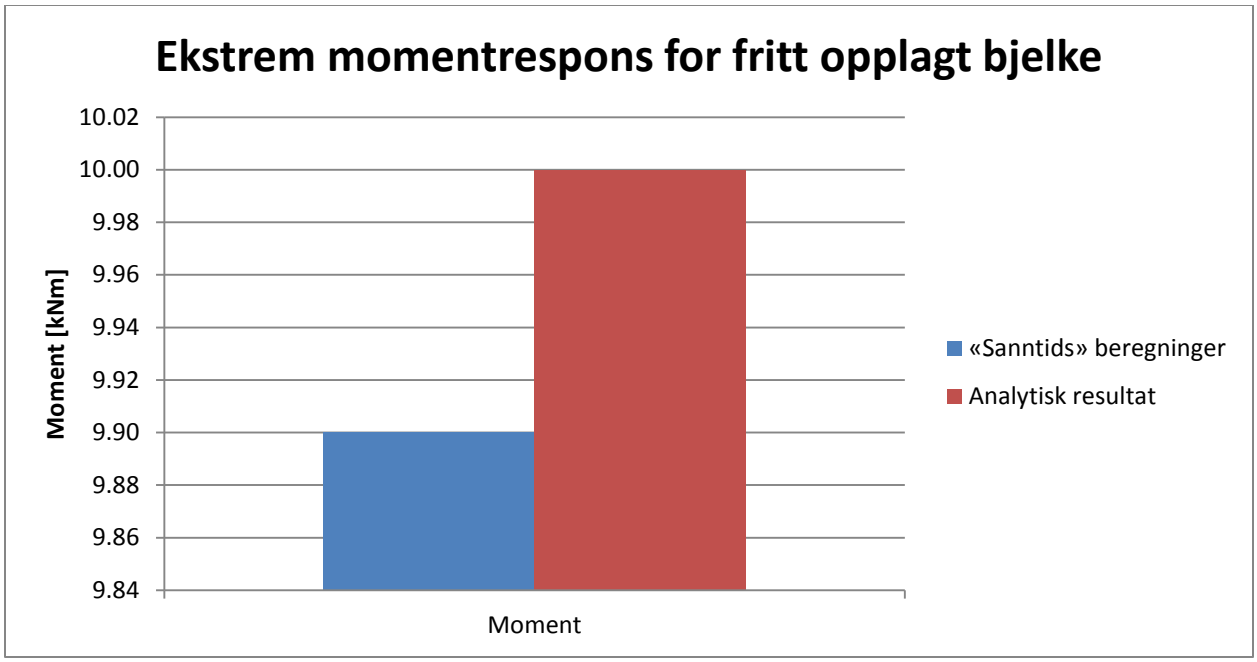
Analytisk verdi for forskyvning er gitt ved formel 4:

$$\delta = \frac{FL^3}{48EI} = \frac{10*10^3 N*(4m)^3}{48*2.1*10^{11}\frac{N}{m^2}*1.94*10^{-6}m^4} = 0.003723m = 3.273mm \quad \textbf{(4)}$$

I testen ble responsene funnet ved å trykke knappen for maksimalverdi under «sanntids» beregninger og lese av responsen og lasttogets avstand fra venstre ende av bjelken for hver av responsene. Avviket i forhold til analytisk verdi er vedlagt i Tabell 21 i vedlegg J, resultatet på grafisk form er gitt i Figur 45 og Figur 46.



Figur 45: Ekstrem forskyvningsrespons for fritt opplagt bjelke



Figur 46: Ekstrem momentrespons for fritt opplagt bjelke

Forskyvningsresponsen hadde 0.09 % avvik og momentresponsen hadde 1.01 % avvik i forhold til sine respektive analytiske verdier. Med disse resultatene er det rimelig å anta at metodene for ekstremrespons fungerer korrekt for «sanntids» beregninger.

Det skal også tas hensyn til at responsene for «sanntids» beregninger ble funnet mens lasttoget stod i en posisjon 1979.8 mm fra venstre ende av bjelken. Altså 20.2 mm fra midtpunktet. Dette kan føre til noen unøyaktigheter i testresultatene.

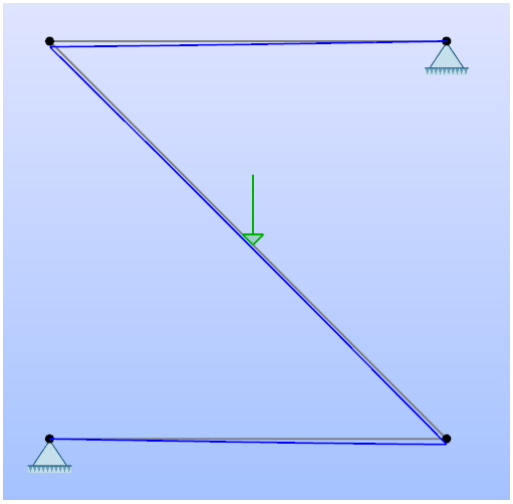
Interpoleringsmetoden har en annen effekt på maksimalresponsen for bjelken i testtilfellet. Begge lasttogposisjonene som er nærmest midtpunktet av bjelken ble funnet som plasseringer hvor ekstremrespons opptrer for begge responstypene som ble testet. Opptredenden var henholdsvis i punktene 1979.8 mm og 2020.2 mm fra venstre ende av bjelken. Dette er fordi like store andeler av lasten blir interpolert til noden i midtpunktet i begge tilfeller, fordi hver lasttogposisjon er like nært midtpunktet som er 2000 mm fra venstre ende.

Selv om en fritt opplagt bjelke egentlig kun skal ha en ekstremverdi for disse responstypene fører interpoleringen til at to blir funnet. Denne effekten er tilfelle når både antallet lasttogposisjoner og antallet elementer er partall, som beskrevet i drøftingen under nøyaktighetstesten.

7.1.5. Konvertering av lasttogposisjon til lasttilfelle

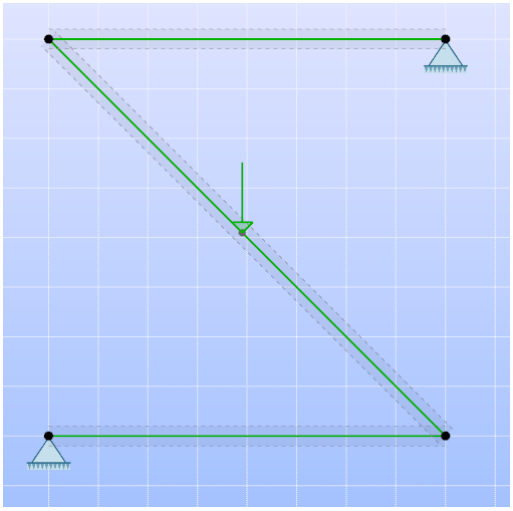
Det oppstod problemer med metoden for konvertering av lasttogposisjoner under implementeringen når lasten står på *members* som er tegnet i retning fra høyre mot venstre. I tillegg er metoden sensitiv for elementinndelingen i modellen. Det er da disse egenskapene som skal testes for denne metoden.

Modellen i Figur 47 består av tre *members* som former en 5 talls-form, hvor den midterste er tegnet fra høyre mot venstre. Formålet er å teste hvorvidt metoden klarer å konvertere en lasttogposisjon på den midterste *member* til et lasttilfelle. Alle *members* har en elementinndeling på 50 elementer per *member*.



Figur 47: 5-talls-formet modell for å teste konvertering til lasttilfelle

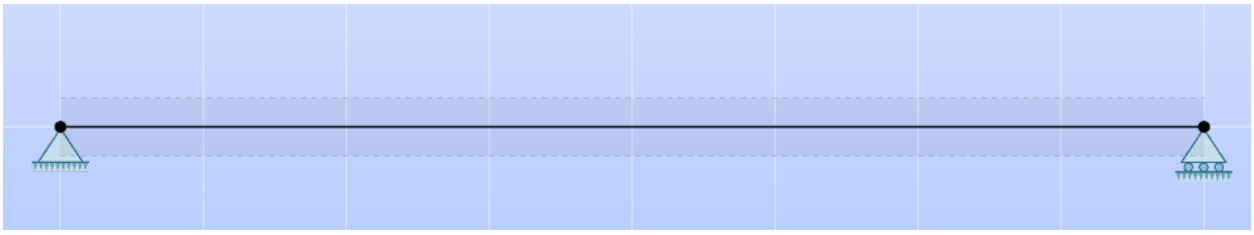
Da lasten var i samme posisjon som på Figur 47, ble det knappen for konvertering trykket. Konverteringen var vellykket og et lasttilfelle ble lagret med lasten i lasttoget i nøyaktig samme posisjon som i Figur 47. Dette tilsier at metoden for konvertering fungerer som tiltenkt. Resultatet er gitt i Figur 48.



Figur 48: 5-talls-formet modell med ferdig konvertert lasttilfelle

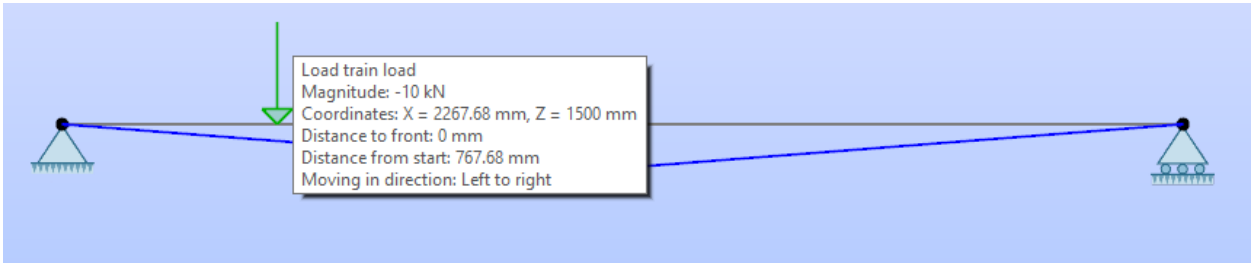
Siden konverteringsmetoden er sensitiv for elementinndelingen i modellen, testes også metoden for en veldig grov inndeling. Modellen det testes på er en fire meter lang fritt opplagt bjelke med to beregningselementer, jamfør Figur 49. Formålet med testen er å undersøke om lasten i lasttilfellet blir forflyttet til venstre ende av bjelken når lasttogposisjonen som blir konvertert er nærmere venstre ende enn midtpunktet av bjelken.

Dette er ønsket oppførsel for metoden siden metoden vil plassere lasten i nærmeste ende av elementet det er på og deretter plassere en node under lasten. Dette forhindrer at to noder havner oppå hverandre.



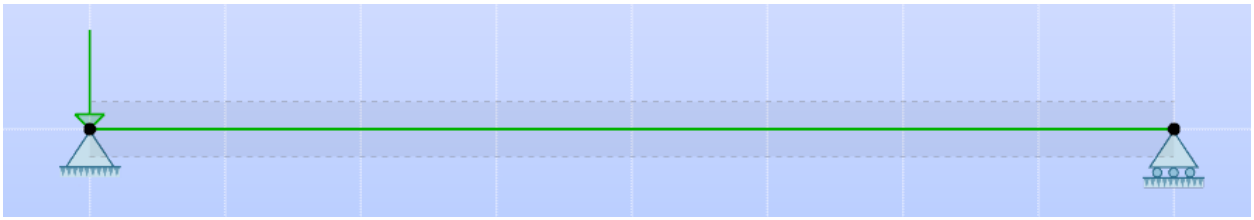
Figur 49: Fritt opplagt bjelke for konverteringstesten

Lasttogposisjonen blir konvertert til et lasttilfelle når lasten er 767,68 mm fra venstre ende av bjelken, jamfør Figur 50.



Figur 50: Lasttogets posisjon før konvertering

Etter konverteringen har lasten i lasttilfellet fått ønsket posisjon på venstre ende av bjelken, jamfør Figur 51.



Figur 51: Lastens posisjon i lasttilfellet etter konvertering

Testtilfellet ovenfor er vel å merke et ekstremtilfelle i forhold til elementinndelingen. Man vil ikke oppnå like store avrundinger ved bruk av standard elementinndeling i **fap2D**. Konverteringsmetoden vil derfor fungere tilstrekkelig for standard elementinndeling.

7.2.Influenslinjer

Analysen testes mot håndregnede eksempler for å sikre korrekthet i tillegg til tester for at de implementerte forbedringene fungerer som påtenkt.

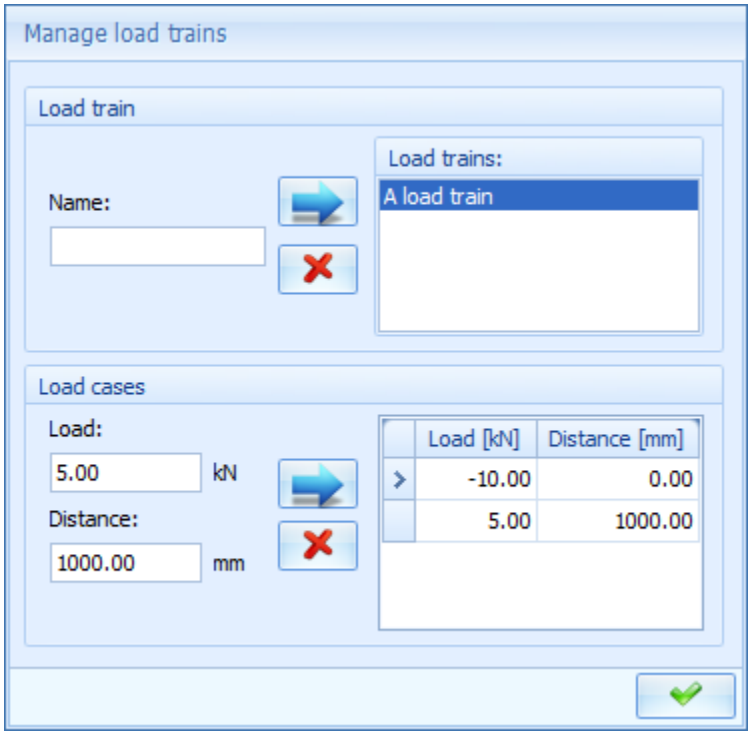
7.2.1. Korrekthet

Det er nødvendig å teste om forbedringene i influenslinjeberegningene regner korrekt. For å teste at analysen for ekstreme responser er korrekt benyttes en fritt opplagt bjelke med et lasttog med flere laster. Momentresponsen i midtpunktet av bjelken blir brukt som responsparameter i testen. Modellen har parameterene i Tabell 9.

Tabell 9: Parametre for modellen i korrekthetstesten

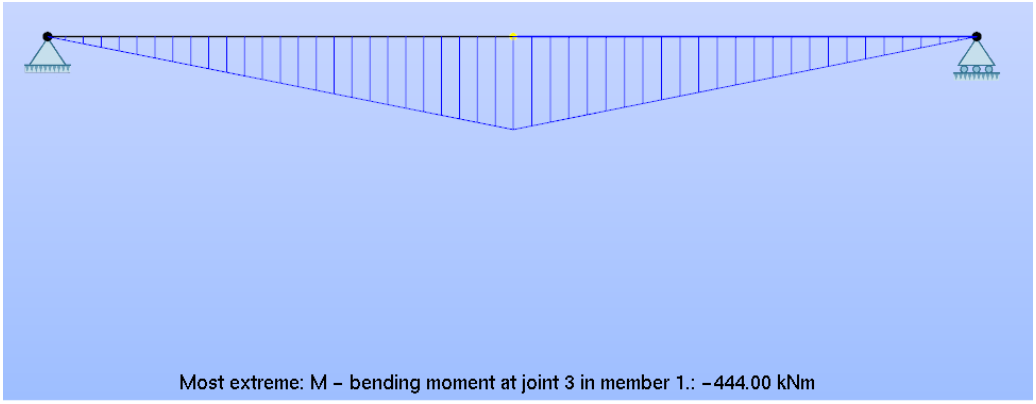
Parameter	Verdi
Bjelketverrsnitt	IPE200
Bjelkens lengde	4 000 mm
Materiale	Stål
Elastisitetsmodul	210 000 MPa
Skjærmodul	80 800 MPa
Antall lasttogposisjoner	100
Influenslast	-444 kN

Lasttoget har to laster, den fremste lasten er på 10 kN og er plassert på lokomotivet og den andre er på 5 kN og er plassert 1000 mm bak lokomotivet. Den fremste lasten virker i negativ retning, mens den bakerste virker i positiv retning, jamfør Figur 52.



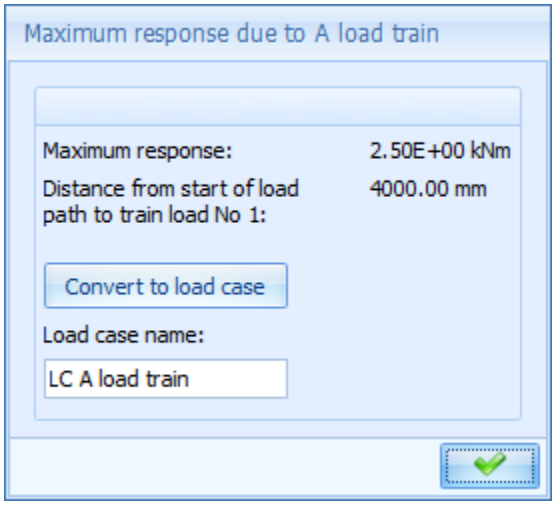
Figur 52: Lasttogets definisjon i korrekthetstesten

En kjøring av analysen gir forventet influenslinje, jamfør Figur 53. Influenslasten er endret til å ikke være en enhetslast for å kontrollere at skaleringen fungerer korrekt.



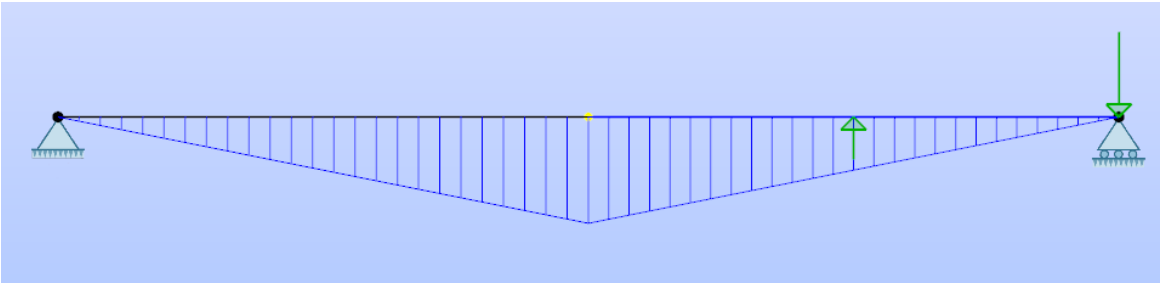
Figur 53: Influenslinje for korrekthetstesten

Deretter gjennomføres en analyse for ekstreme responser for å finne maksimalresponsen, jamfør Figur 54.



Figur 54: Resultat for maksimalresponsen

Analysen finner maksimal respons når den fremste lasten er i høyre ende av bjelken. Lastene i lasttoget blir også tegnet med korrekt fortegn og i skala i forhold til største last, jamfør Figur 55. Den bakerste lasten er halvparten så stor som den fremste.



Figur 55: Lastene i posisjonen langs influenslinjen som forårsaker maksimal respons

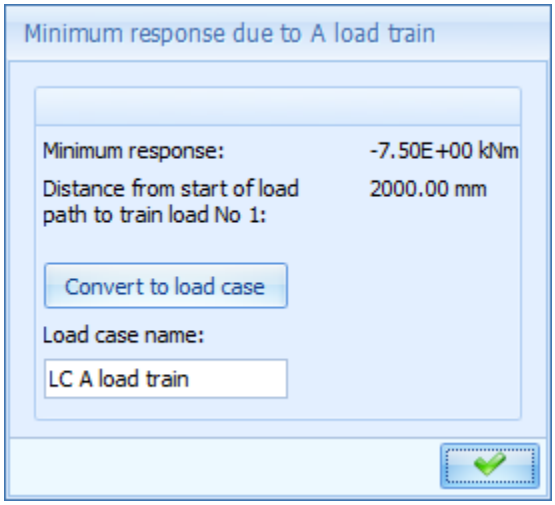
For å kontrollere at beregningen er korrekt må hver last ganges med absoluttverdien til ordinaten i punktet til lasten delt på influenslasten og legges sammen til den totale responsen. Den fremste lasten er i enden av lastbanen og har dermed 0 kNm som ordinat og bidrar ikke til den totale responsen. Den bakerste lasten er plassert midt mellom midten av bjelken og høyre ende. Siden influenslinjen varierer lineært fra midten til enden av bjelken, vil ordinaten i punktet til lasten være halvparten av ordinaten i midten. Ordinaten i midten er -444 kNm.

Ordinater deles på influenslasten før de ganges med sin respektive last. Formel 5 gir et uttrykk for maksimalresponsen når lastene er i posisjonen som i Figur 55.

$$Max = \sum_{i=1}^{2} \left| \frac{o_i}{L} \right| F_i = \left| \frac{0\,kNm}{-444\,kN} \right| * -10\,kN + \left| \frac{-222\,kNm}{-444\,kN} \right| * 5\,kN = 2.5\,kNm \qquad \textbf{(5)}$$

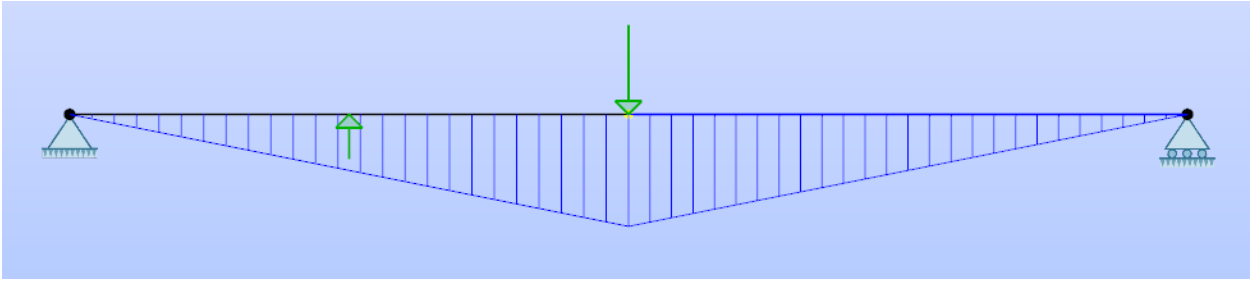
Der i er indeksen til hver last i lasttoget, målt fra fremst i toget, o_i er ordinaten i posisjonen til den gitte lasten, L er influenslasten og F_i er den gitte lasten. Håndberegningen viser at analysen for ekstreme responser regner korrekt i dette tilfellet.

Den samme testen gjennomføres også for minimumsresponsen, jamfør Figur 56.



Figur 56: Resultat fra minimumsresponsen

Analysen finner minimumsresponsen når den fremste lasten er på midten av bjelken. Lastene er fremdeles tegnet i korrekt posisjon og skala, jamfør Figur 57.



Figur 57: Lastene i posisjonen langs lastbanen som forårsaker minimumsresponsen

Den bakerste lasten er midt mellom venstre ende og midtpunktet og dens ordinat er da halvparten av den i midtpunktet, på samme måte som for maksimalresponsen. Formel 6 gir et uttrykk for minimumsresponsen når lastene er i posisjonene som i Figur 57.

$$Min = \sum_{i=1}^{2} \left| \frac{o_i}{L} \right| F_i = \left| \frac{-444\,kNm}{-444\,kN} \right| * -10\,kN + \left| \frac{-222\,kNm}{-444\,kN} \right| * 5\,kN = -7.5\,kNm \quad \textbf{(6)}$$

Håndberegningen viser at analysen for ekstreme responser også regner korrekt i dette tilfellet.

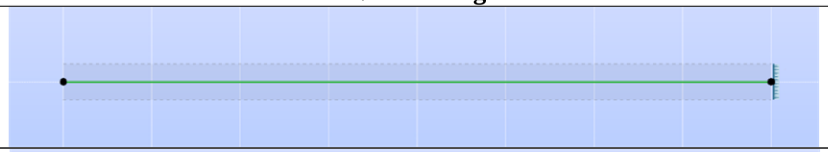
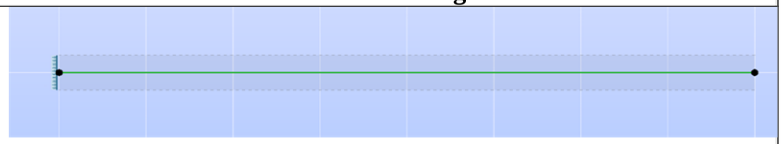
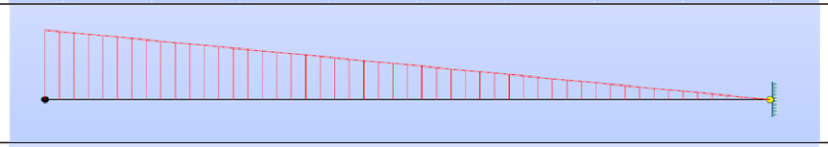
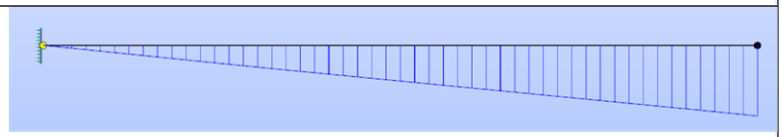
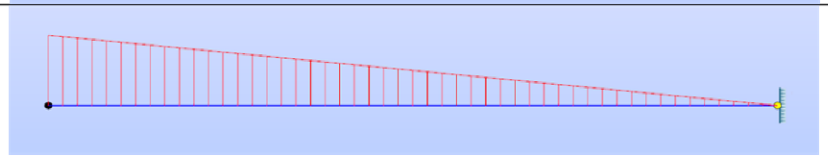
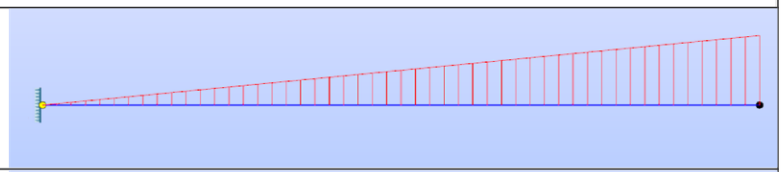
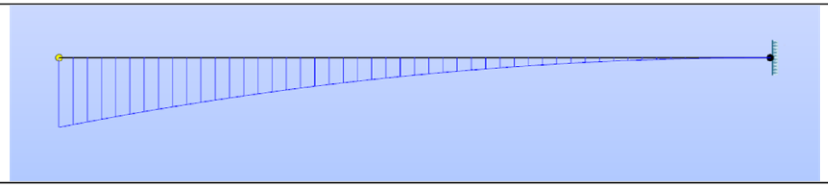
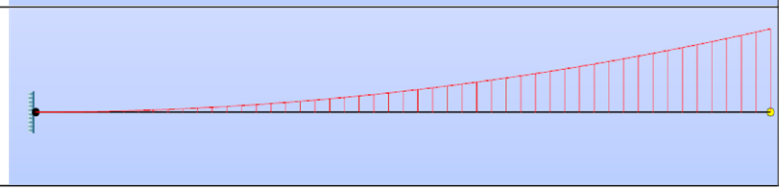
7.2.2. Visualisering

Som nevnt i kapittelet om funksjonalitet oppstod det problemer med å visualisere influenslinjer korrekt på *members* som har blitt rotert om startnoden, etter de først har blitt laget. Responsparameterne som er avhengig av orientering i forhold til lasten er sensitive for rotasjon og disse skal testes for å kontrollere om de visualiseres korrekt, også etter *member* har blitt rotert. Influenslasten er en enhetslast med negativ z-retning gjennom hele testen. Modellen er en utkragerbjelke og responsparameterne som skal testes er:

- Opplagermomentet
- Momentrespons på elementet nærmest opplageret
- Rotasjonsforskyvning i den frie enden av bjelken

Resultatene før og etter rotering er presentert i Tabell 10. Ut fra resultatene fra før og etter rotering av bjelken tegnes samtlige responsparametre korrekt.

Tabell 10: Influenslinjer for responsparametrene i visualiseringstesten før og etter rotering

	Før rotering	Etter rotering
Bjelkens orientering		
Opplagermomentet		
Momentrespons i elementet nærmest opplageret		
Rotasjonsforskyvning i enden av bjelken		

7.3.Konklusjon

Testene som har blitt gjennomført viste at det er grunn til å tro at programmet fungerer som påtenkt. For «sanntids» beregninger er beregningene nøyaktige med forbehold i lastens plassering i forhold til midtpunktet og eventuelle avrundingsproblemer knyttet til interpoleringen. Analysen er sensitiv til elementinndeling og en finere inndeling vil føre til mer nøyaktige resultater. Standard elementinndeling i **fap2D** gir brukbare resultater. Nøyaktigheten vil også forbedres desto flere lasttogposisjoner man bruker uten å endre elementinndelingen.

Tidsbruken til «sanntids» beregninger er som forventet ved gjenbruk av lagret faktorisert stivhetsmatrise. Metodene rundt ekstreme responser fungerer også med de samme bemerkningene til nøyaktighet som under nøyaktighetstesten. Standard elementinndeling i **fap2D** er også anbefalt for denne metoden. Konverteringen av lasttogposisjoner til lasttilfeller fungerer som påtenkt uansett hvilken orientering de ulike *members* har.

For influenslinjer sørger forbedringene som har blitt implementert i denne masteroppgaven for at beregninger gjennomføres korrekt. Dette gjelder med tanke på skalering av resultater mot influenslasten, plassering av laster og å finne orientering og ekstreme responser for lastene. Videre fungerer også visualisering av responsparametre som ønsket.

8. Videre arbeid

Kapittelet oppsummerer mangler og forbedringer ved masteroppgaven som kan videreføres i fremtidig arbeid med **fap2D**.

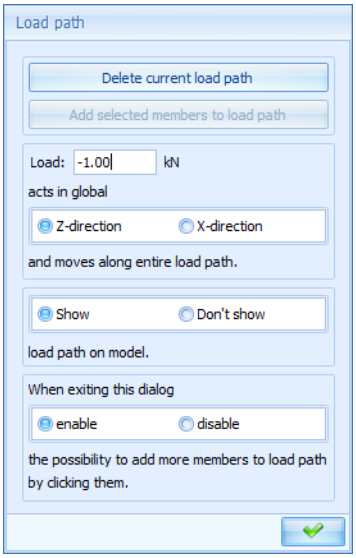
8.1. Omhyllingskurver

Omhyllingskurver er en analyse som ikke har blitt arbeidet med i denne masteroppgaven. Dette er fordi influenslinjer og programmets stabilitet heller fikk prioritet underveis.

8.2. Dialogboks for lastbane

Det kan også være nyttig å splitte dialogboksen for lastbane for influenslinjer i to slik at alt som omhandler lastbaner havner i en dialogboks, mens alt som omhandler influenslasten havner i en annen dialogboks. Slik dialogboksen er satt opp nå (vist i Figur 58) fungerer forsåvidt bra, men med en dialogboks for hver av konseptene er det mulig å gjenbruke den samme dialogboksen for lastbaner for både «sanntids» beregninger og influenslinjer. Dermed er det enklere for brukere av programmet å innse at lastbaner kan brukes på tvers av analysene.

Et eksempel på problemet er at en bruker lager en lastbane i dialogboksen for lastbane i «sanntids» beregninger, for deretter å starte en influenslinjeanalyse. Det vil da komme en feilmelding om at influenslasten ikke er definert . Dermed må brukeren inn i dialogboksen for lastbane for influenslinjer og definere influenslasten, selv om en lastbane allerede finnes på modellen . Dette er da ikke intuitivt og det vil derfor være hensiktsmessig å ha innstillingene knyttet til influenslast i en egen dialogboks.



Figur 58: Dialogboksen for lastbane for influenslinjer

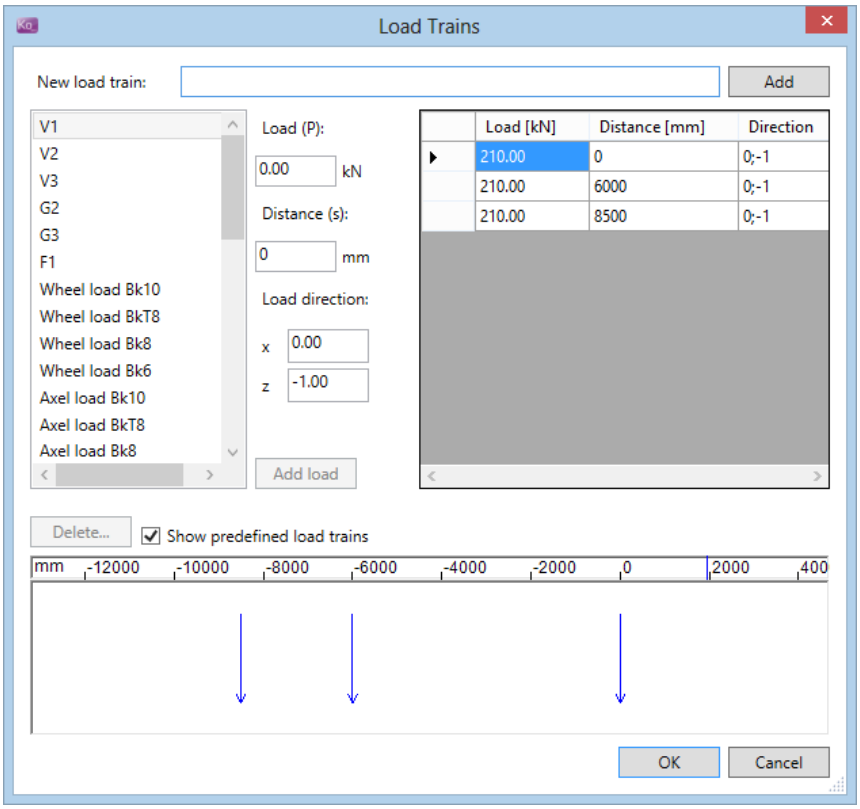
8.3. Kapasitetskontroll

Kapasitetskontroll innenfor stålkonstruksjoner kunne vært praktisk å fått implementert inn i «sanntids» beregninger som en respons som kan betraktes. En slik funksjonalitet finnes allerede under vanlig lineær statisk analyse i **fap2D** og vil også være nyttig i en fremtidig implementasjon av «sanntids» beregninger, jamfør Figur 59.

Figur 59: Valgene for kapasitetskontroll for stålkonstruksjoner under lineær statisk analyse

8.4. Forhåndsdefinerte lasttog

I likhet med lasttoganalysen i Focus[4], burde også «sanntids» beregninger inneholde en forhåndsdefinert samling med lasttog, jamfør Figur 60. Disse lasttogene representerer typiske lastkonfigurasjoner for ulike laster langs lastbanen, for eksempel bil-, hjul-, tog- og personlaster. Med en slik samling kan brukere av **fap2D** enkelt kunne velge et standard lasttog som passer for det gitte problemet man ønsker å betrakte.



Figur 60: Dialogboksen for lasttog i Focus[4] med forhåndsdefinerte lasttog i vinduet øverst til venstre

8.5. Behandling av lastbaner

En annen oppgave for videre arbeid vil være å gjøre lastbanene, som både er knyttet til «sanntids» beregninger og influenslinjer, om til objekter i beregningsmodellen når en av de nevnte analysene gjennomføres. I skrivende stund er lastbaner lagret som objekter under konstruksjonsmodellen, siden de blir definert av bruker mens modellen blir laget.

Dermed blir lastbanene fremdeles referert til som objekter under konstruksjonsmodellen når de brukes i «Results»-fanen til å plassere laster og andre operasjoner. Dette gjør de sårbare til endringer i konstruksjonsmodellen mens «Results»-fanen er åpen. Slike endringer vil da føre til feil under «Results»-fanen, siden beregningsmodellen ikke blir oppdatert med endringene i lastbanen på konstruksjonsmodellen.Dette problemet vil muligens ta en del tid å fikse og det ble ikke tid i løpet av masteroppgaven til å løse det, da det ble oppdaget sent.

8.6. Inndeling av lasttogposisjoner og noder

Et annet forbedringspotensiale til «sanntids» beregninger vil være innenfor inndelingen av lasttogposisjoner og noder. Målet er å ordne elementinndelingen slik at hver av lastene i et gitt lasttog alltid vil ha en node under sin posisjon. Dermed oppnås en mer nøyaktig løsning for responsene i plasseringen til lasttoget og interpolering blir unødvendig, siden laster plassert direkte i noder kan brukes direkte i beregningskjernen.

En slik forbedring kan gjennomføres på to måter. Den første går ut på å endre metodene for elementinndeling til å bruke en fin nok inndeling til å sikre at alle laster får en node under lasten. Inndelingen bestemmes ved å bruke avstanden til hver av lastene fra lokomotivet. Man deler lengden av lastbanen på en fellesnevner basert på avstandene som gjør at elementinndelingen blir fin nok til at alle laster i lasttoget får en node under lasten uansett hvor de er plassert langs lastbanen.

Selv om en slik metode vil føre til mer nøyaktige svar, vil dette kreve mye arbeid i omformulering av elementinndelingen i **fap2D**. I tillegg vil en såpass fin elementinndeling føre til et stort ligningssystem som det vil ta lang tid å løse.

Den andre metoden består av å endre elementinndelingen til å lage en ny inndeling for hver gang lasttoget plasseres i en lasttogposisjon. Denne inndelingen sørger da for at det alltid vil være en node under hver last som er tilstede på lastbanen og dermed nøyaktige svar.

Denne metoden vil også føre til mye arbeid i omformulering av den eksisterende elementinndelingen. I tillegg mister man fordelen med gjenbruk av faktorisert stivhetsmatrise, siden nye stivhetsmatriser må lages for hver iterasjon fordi elementinndelingen er endret. Likevel beregner **fap2D** ligningssystemer med akseptabel hastighet selv uten gjenbruk av faktorisert stivhetsmatrise, som vist i høstens prosjektoppgave[1].

Interpoleringsmetodene som allerede er implementert i «sanntids» beregninger har blitt testet og viser at de fungerer med tilstrekkelig nøyaktighet. Derfor er det nok ikke kritisk å få implementert en slik fullstendig omveltning av analysen og programmet for å oppnå fullstendig nøyaktighet.

9. Referanser

[1] Erstad, Frans: ««Sanntids» beregninger med **fap2D**», prosjektoppgave, NTNU, Trondheim 2012

[2] Bell, Kolbein: "**fap2D** – A WINDOWS-based program for static and dynamic analysis of 2D frame structures, User's manual, Preliminary version", May 2011.

[3] Bell, Kolbein: "**Frame2D** data structures", 2013.

[4] Focus Software AS. Focus Konstruksjon 13.3.5.0 Copyright © 2005 – 2013. http://www.focus.no/produkter/focus-konstruksjon.aspx, Mai 2013

[5] Irgens, Fridtjov: "Formelsamling mekanikk 3. utgave", Tapir Akademisk Forlag, Trondheim 1999

[6] Microsoft Corporation. Windows 8 64-bit. http://windows.microsoft.com/en-us/windows-8/meet, Mai 2013

[7] Microsoft Corporation. Visual Studio 2012. http://www.microsoft.com/visualstudio/eng/visual-studio-update, Mai 2013

[8] PuTTy Team. Putty. http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, Mai 2013

[9] DevExpress. Dxperience 13.1.4.0. http://www.devexpress.com/, Mai 2013

[10] The TortoiseSVN Team. TortoiseSVN 1.7.12 64-bit. http://tortoisesvn.net/downloads.html, Mai 2013

[11]Microsoft Corporation. Microsoft Word 2010. http://office.microsoft.com/en-us/word/, Mai 2013

[12] Microsoft Corporation. Microsoft Visio 2010. http://office.microsoft.com/en-us/visio, Mai 2013

[13] Microsoft Corporation. Microsoft Excel 2010. http://office.microsoft.com/en-us/excel/, Mai 2013

[14] Tao Framework project members. Tao Framework. http://sourceforge.net/projects/taoframework/, Mai 2013

[15] Microsoft Corporation. .NET Framework 4.5. http://msdn.microsoft.com/en-us/library/vstudio/w0x726c2.aspx, Mai 2013

[16] Aase, Daniel og Årvik, Brita: «**fap2D** – Forbedring og videreutvikling av GUI», masteroppgave, NTNU, Trondheim 2011.

[17] Flexera Software. Installshield 2012 http://www.installshield.com/, Mai 2013

Vedlegg

A. Logg

Logg over gjennomført arbeid fra dag til dag. Dette er ikke en akademisk skrevet tekst.

Tabell 11: Logg for utviklingen på fap2D

Dato	Tidspunkt	Filer som hovedsaklig ble endret	Beskrivelse
14.01	11-14	FrmMainGUI.cs	Satte meg inn i GUI for **fap2D** og hvordan man legger til knapper i ribbons
15.01	13-20	FrmMainGUI.cs FrmMainGUI.AnalysisPage.cs	Lagde ferdig GUI for analysis page for real time analysis Hadde også et møte med Kolbein, Erik og Kristian med gjennomgang av prosjektoppgaven og diskusjon rundt arbeidet med masteroppgaven
16.01	12-17	FrmMainGUI.AnalysisPage.cs ComputationalModel.cs	Tilknyttet funksjonalitet til knappene som bestemmer hvilke responser som skal betraktes under analysen. Lagde metode som håndterer kommandoen som starter analysen
26.01	18-23		Gikk gjennom kode for load path og load train for å planlegge en implementasjon av disse for real time analysis.
27.01	15-23		Skrev disposisjon, definisjoner og lagde grunnlaget for innholdet i rapporten
28.01	13-15 og 19-23		Skrev bruksanvisning for SVN server og skrev videre på rapporten Møte med Kolbein, Erik og Kristian
29.01	11-22	FrmInfluenceLineLoadPath.cs FrmRealTimeLoadPath.cs FrmInfluenceLineLoadTrain.cs FrmRealTimeLoadTrain.cs FrmMainGUI.cs FrmMainGUI.designer.cs FrmMainGUI.AnalysisPage.cs FrmMainGUI.LoadingPage.cs	Skrev kapittelet om innledning, forord, arbeidsprosess og utvikling i rapporten. Hjalp Kristian med å debugge et problem angående Joints som ikke kan plasseres på *members* som er oppdelt. Lagde dialogbokser og nye knapper for å skille mellom dialogboksene for influenslinjer og "sanntids» beregninger. De gikk ikke å forene disse funksjonalitetene.
31.01	16-18 og 21-00	FrmRealTimeLoadPath.cs FrmRealTimeLoadTrain.cs RealTime.cs ComputationalModel.cs	Gjorde ferdig funksjonalitetene I dialogboksene og startet å jobbe med å sortere og bruke noder for å bruke de i «sanntids» beregninger.

		FrmModel.cs	
01.02	12-21	FrmRealTimeLoadTrain.cs LoadTrainLoad.cs ComputationalModel.cs FrmModel.cs	Gjorde ferdig sortering av noder og bygget ut dialogboksen for lasttog for «sanntids» beregninger ved å utvide eksisterende metoder for punktlaster. Mange endringer i grensesnitt dialogboksen for lasttog
02.02	12-18	FrmRealTimeLoadTrain.cs LoadTrainLoad.cs ComputationalModel.cs	Arbeidde lenge med en feil i dialogboksen for lasttog, da den ikke ville vise aktuelle laster i tabellvinduet på skikkelig måte. Deretter ble et enkelt lasttog introdusert i utregningen av «sanntids» beregninger og første beregning gjennomført. Neste oppgave blir å forbedre resultatvisningen, samt gjøre det mulig å inkludere flere lasttog, med mange laster.
03.02	18-24	FrmMainGui.cs FrmMainGui.AnalysisPagecs FrmMainGui.Resultspage.cs FrmMainGui.Designer.cs ComputationalModel.cs	I dag brukte jeg lang tid på å designe knapper i grensesnittet for resultatvisning av «sanntids» beregninger. Underveis oppstod det forviklinger i forhold til annet arbeid som ble utført på FrmMainGui.cs av Kristian. Den enkleste løsningen ble at jeg måtte forkaste to timers arbeid og ta det på nytt. I tillegg knyttet jeg funksjonalitet til knappene i resultatvisningen som styrer hvilken type respons som vises.
04.02	15-23	ComputationalModel.cs ComputationalModel.Tables.cs FrmRealTimeLoadTrain.cs FrmMainGui.AnalysisPagecs FrmMainGui.Resultspage.cs	La til event handlers for keys i dialogboks for «sanntids» lasttog. Endret måten «sanntids» beregninger blir håndtert, analysen blir nå kjørt en iterasjon av gangen. For deretter å kalles av hendelser som er styrt av bruker. Nye klasser er blitt opprettet for å håndtere lasttoget under resultatvisningen.
05.02	13-15 og 18-23	ComputationalModel.cs ComputationalModel.Analyzes.cs ComputationalModel.Drawing.cs	Møte med alle i dag og diskusjon om gjennomført arbeid m.m. Lagde og fylte inn en mal for rapportering av feil i programmet som alle involverte nå deler på og

			oppdaterer jevnlig. Splittet klassen ComputationalModel.cs til to mindre klasser for å gjøre de mer oversiktlige
06.02	12-18	ComputationalModel.cs StructuralRealTimeLoadTrain.cs StructuralRealTimeLoadTrainLoad.cs ComputationallRealTimeLoadTrain.cs ComputationallRealTimeLoadTrainLoad.cs FrmInfluenceLineLoadTrain.cs FrmRealTimeLoadTrain.cs	Fullførte splitting mellom influenslinjer og «sanntids» beregninger, de har nå ingen felles funksjoner I koden utenom lastbane. «Sanntids» beregninger ble splittet i to typer klasser. En klasse for å håndtere lasttoget i modelleringsfasen og en for beregningsfasen. Dette var nødvendig fordi egenskapene i beregningsmodulen burde kun brukes av klasser i samme modul (det samme gjelder for analysemodulen) for å forhindre feil under kjøring av programmet.
07.02	15-23	ComputationalModel.cs ComputationalRealTimeLoadTrain.cs FrmMainGui.ResultsPage.cs FrmMainGui.cs	Lagde mye nytt grensesnitt for å kontrollere lasttoget og egen tegnemetode for å tegne lastene på lasttoget etter hvert som de er aktuelle for figuren. I tillegg ble logikken for å kontrollere lasttoget påbegynt og dagen ble avsluttet med at en enkel animasjon ved å flytte lasten fra høyre mot venstre ble gjennomført. Løsningen med at hver last «snapper» til nærmeste node fører til at bruker må velge en nokså fin inndeling for gode resultater.
08.02	12-20	ComputationalModel.Loadtrain.cs FrmMainGui.cs ComputationalRealTimeLoadTrain.cs FrmMainGui.ResultsPage.cs ComputationalModel.Tables.cs	Lagde nye knapper for å kontrollere lasttoget og logikk for å kontrollere lasttoget under kjøring av analysen. Logikken har tatt lengst tid å lage, da den må omfatte alle situasjoner som lasttoget kan havne i. Analysen beregner også tilsynelatene korrekte resultater nå.
09.02	12-17	ComputationalModel.Loadtrain.cs FrmMainGui.cs FrmMainGui.ResultsPage.cs	Lagde mer logikk for å kontrollere lasttoget. Det er nå mulig å velge mellom ulike lasttog under resultatfanen mens man betrakter resultatene.
10.02	12-20	ComputationalModel.Loadtrain.cs FrmMainGui.cs ModelSymbol.cs	Begynte å bygge om logikken til lasttoget for å håndtere varierende elementstørrelser. Dette var det ikke støtte for tidligere og trengte mye testing for å få til.

		ModelSymbolCollection.cs	Tegnemetoden for ulike punktlaster er også fikset og tegner laster i skala i forhold til største opptredende last og momenter tegnes med rett retning i forhold til fortegn.
11.02	11-18	ComputationalModel.Loadtrain.cs FrmMainGui.cs FrmMainGui.AnalysisPage.cs	I dag ble det gjennomført masse debugging for å sikre at lasttoget kunne håndtere varierende elementstørrelser, dette ble ferdigstilt i dag. I tillegg ble sorteringsmetoden for laster på lasttog funnet til å ikke fungere, så denne ble også fikset. Til slutt ble grensesnittet på resultatvisning gjort penere.
12.02	11-17	FrmMainGui.cs FrmMainGui.AnalysisPage.cs FrmMainGui.ResultsPage.cs ComputationalModel.Loadtrain.cs	Møte med alle i dag og diskusjon om hvordan lasttoget skal fungere. Spørsmålet er om lasttoget skal bevege seg med faste steglengder og heller fordele laster ut til noder hvis det havner inne på et element, dette er ikke avgjort ennå. Videre har det skjedd forbedringer i grensesnitt for resultatvisning for å gjøre det enklere å vedlikeholde og logikken for vending av lasttoget har blitt jobbet med.
14.02	11-18	RealTime.cs FrmMainGui.AnalysisPage.cs FrmMainGui.ResultsPage.cs StoringUtilities.cs ModelDataset.xsd	Møte med Kolbein i dag for å diskutere nøyaktig hvordan «sanntids» beregingerskal implementeres. Konklusjonen ble at man skal bruke faste avstander mellom lasttogposisjonene slik som i Xtrm-analysen og laster som havner mellom elementer skal lineærinterpoleres til nærmeste noder. I tillegg ble lagringen av lasttog for «sanntids» beregninger ferdigstilt og metoder for validering av lastbanen for å gjøre det enklere å unngå feil relatert til den.
17.02	16-24	FrmMainGui.cs LoadPath.cs LoadTrainPosition.cs FrmMainGui.AnalysisPage.cs ComputationalModel.cs ComputationalModel.LoadTrain.cs ComputationalModel.ModelTools.cs	Debugget feil i influenslinjer. Influenslinjer og «sanntids» beregninger bruker nå samme funksjoner for lastbane. Har også lagt til flere metoder for å dele inn lastbanen i lasttogposisjoner med innbyrdes jevnlange avstander til hverandre som lasttoget i «sanntids» beregninger kan bevege seg mellom.
18.02	12-18	ComputationalModel.cs ComputationalModel.LoadTrain.cs	Debugget en del feil I programmet som kom av endringer gjort i forhold til lastbane I går.

		ComputationalRealTimeLoadTrain.cs ComputationalRealTimeLoadTrainLoad.cs ComputationalModel.Analyses.cs StoringUtilities.cs	I dag la jeg til metoder for å lineærinterpolere en last til nærmeste noder, i tillegg til å lokalisere nærmeste noder og gjorde om en del på lasttoget for å håndtere posisjoner mellom noder. Jeg hadde også møte med Kolbein for å diskutere en utvidelse av rutinen Arc_to_X for å kunne bruke den i min analyse. Rutinen finner et punkt i lokale koordinater på en gitt avstand inne på en bue. Jeg vil gjerne ha dette punktet i globale koordinater. Vi ble enige om at dette burde kunne gjennomføres og jeg får snart en ny versjon av rutinen som gir globale koordinater.
19.02	17-00	ComputationalModel.LoadTrain.cs CurveUtils.cs	I dag brukte jeg mye tid på å debugge en feil i oppdelingen av lastposisjoner. Avrundingsfeil på grunn av dobbelpresisjonstall førte til at det siste punktet på lastbanen av og til ikke ble inkludert blant lastposisjoner. I tillegg ble også en ny metode fra Kolbein for å beregne posisjoner på kurver implementert i programmet. Debugging viste at den viser samme resultater for konvekse og konkave kurver og trenger derfor ombygging. I dag hadde vi også et møte med alle sammen hvor blant annet en del grafiske detaljer på programmet ble diskutert.
20.02	12-22	ComputationalModel.LoadTrain.cs LoadPath.cs ComputationalRealTimeLoadTrain.cs ComputationalRealTimeLoadTrainLoad.cs LoadTrainPosition.cs	Debugget posisjonering på kurver I samarbeid med Kolbein, posisjonering på kurver skal nå fungere uavhengig av hvordan kurven er rotert. Jeg brukte derfor en god del av dagen på møter med han for å finne løsningen på dette. Begynte også å endre metodene for å flytte lasttoget, siden posisjoneringen med faste avstander for lasttoget krever endringer i koden.
21.02	16-00	ComputationalModel.LoadTrain.cs LoadPath.cs ComputationalRealTimeLoadTrain.cs ComputationalRealTimeLoadTrainLoad.cs LoadTrainPosition.cs	Hele dagen gikk med på å ordne posisjonering av lastene til lasttoget. Løsningen ble at hver posisjon blir beregnet ut fra starten av lastbanen og plassert deretter, dette var også den enkleste løsningen sett med tanke på oversiktligheten i koden. Koden for posisjonering har også blitt delt opp i

			mindre, mer oversiktlige metoder.
24.02	16-23	ComputationalModel.LoadTrain.cs ComputationalRealTimeLoadTrain.cs	Også i dag ble det mye arbeid med posisjonering av lasttoget. Nesten hele bevegelsen av toget fra venstre mot høyre er nå tilpasset en fast lengde på mellom hver posisjon for lasttoget.
25.02	14-20	ComputationalModel.LoadTrain.cs FrmModel.InfoText.cs FrmModel.cs	Gjorde ferdig posisjonering for alle bevegelser når lasttoget beveger seg fra venstre mot høyre. Det eneste som gjenstår med posisjonering er nå bevegelser fra høyre mot venstre. Lagde også «Tool tips» for lastene i lasttoget, slik at brukere kan holde musepeker over laster for å få informasjon om disse. Innføringen av «Tool tips» førte til at bakgrunnen for modellen ble hvit, dette må fikses senere.
26.02	12-17	ComputationalModel.LoadTrain.cs FrmModel.InfoText.cs	Fjernet en feil i interpolasjonsmetoden og «Tool tips» etter noe testing og la til at man kan betrakte ekte forskyvninger i tillegg til normaliserte. Excel-arket med rapporterte feil har også blitt endret til å bli mer oversiktlig ved utskrift. Hadde et to timers møte med Kolbein og Erik for å demonstrere hva som har blitt gjennomført siden sist.
03.03	17-23	ComputationalModel.LoadTrain.cs	Fjernet feil i posisjoneringsmetoden som gjorde at momenter alltid ble feilplassert. I tillegg jobbet jeg en del med å få snudd toget slik at det kan bevege seg i motsatt retning på en konstruksjon. Til nå kan toget gjøre enkle bevegelser etter at det er snudd, det må mer testing og implementasjon til for at det skal fungere skikkelig.
04.03	19-23	ComputationalModel.LoadTrain.cs ComputationalRealTimeLoadTrain.cs	Fjernet flere feil i posisjoneringsmetoden både når lasttoget går fra venstre mot høyre og omvendt, etter mye testing. Posisjoneringsmetoden er nesten ferdig for bevegelser i begge retninger.
05.03	11-21	ComputationalModel.LoadTrain.cs	Forhåpentligvis er alt av posisjonering av lasttoget i begge retninger ferdig testet og implementert, etter en grundig gjennomgang i dag Koden knyttet til posisjonering ble også ryddet opp i og er nå mer oversiktlig. I tillegg var det møte med alle i dag hvor ulike bugs ble diskutert og posisjonering ved et snudd lasttog ble demonstrert.

07.03	12-20	FrmMainGui.cs FrmMainGui.AnalysisPage.cs FrmMainGui.ResultsPage.cs FrmModel.InfoText.cs	Begynte å lage animasjonen for lasttoget ved å kombinere de ulike posisjoneringsmetodene fra tidligere. Tooltips for lastene på lasttoget har også blitt forbedret.
08.03	16-22	FrmMainGui.cs FrmMainGui.AnalysisPage.cs FrmMainGui.ResultsPage.cs ComputationalModel.LoadTrain.cs RealTimeIteration.cs ComputationalModel.Drawing.cs	Første implementasjon av skalering av responser på modellen etter største opptredende respons på lastbanen. Tidligere har responser blitt skalert etter største opptredende respons per forflytning av toget, som virker misvisende for brukere.
09.03	11-19	FrmMainGui.cs FrmMainGui.AnalysisPage.cs FrmMainGui.ResultsPage.cs ComputationalModel.cs ComputationalModel.LoadTrain.cs ComputationalModel.ModelTools.cs ComputationalModel.Drawing.cs ComputationalModel.Tables.cs RealTimeIteration.cs	I dag ble det gjennomført masse testing og implementasjon av nye tegnemetoder som gjør at laster i «sanntids» beregninger tegnes I forhold til den største lasten på hele lastbanen og ikke i forhold til største last I hver iterasjon. Dette førte også til mye opprydning i eksisterende kode. Etter dette ble arbeidet med å lokalisere og plassere toget i de mest utsatte posisjonene på lastbanen påbegynt.
11.03	17-23	FrmMainGui.AnalysisPage.cs FrmMainGui.ResultsPage.cs ComputationalModel.cs ComputationalModel.LoadTrain.cs ComputationalModel.ModelTools.cs ComputationalModel.Drawing.cs RealTimeIteration.cs	Mye arbeid med å lokalisere og posisjonere lasttoget på de mest utsatte posisjonene på lastbanen. Løsningen ble å lagre de enkelte egenskapene for hvert tog i hver posisjon for seg selv for å hindre at noen endrer seg og posisjonene dermed ikke blir funnet.
12.03	13-20	FrmMainGui.AnalysisPage.cs FrmMainGui.ResultsPage.cs ComputationalModel.LoadTrain.cs	Møte med alle i dag hvor funksjonaliteten til «sanntids» beregninger ble presentert og diskutert. Konklusjonen ble at man er fornøyd med implementasjonen slik den er nå. I tillegg ble det gjennomført en god del feilsøking for å fjerne feil knyttet til ekstrem respons.
13.03	13-18	FrmMainGui.ResultsPage.cs	La til funksjonalitet som gjør det mulig for brukere

		ComputationalModel.LoadTrain.cs ComputationalModel.ModelTools.cs	å velge mellom ulike ekstremresponser innenfor en type respons. Ulike ekstremresponser er definert ved mange posisjoner langs lastbanen som gir samme ekstreme respons.
14.03	17-22	ComputationalModel.LoadTrain.cs ComputationalModel.cs FrmMainGui.cs FrmMainGui.AnalysisPage.cs FrmMainGui.ResultsPage.cs FrmModel.cs ComputationalRealTimeLoadTrain.cs ComputationalModel.Analyzes.cs	Fikset søket etter ekstremresponser slik at når en ekstrem respons finnes, leter programmet også etter verdier some er nesten like store som den som allerede er funnet. Forskjellen må ikke vere større enn 10^{-6}. Dermed finner man flere ekstremresponser. «Tool tips» for laster er også fikset, nå blinker ikke vinduet hvitt lenger når det «tool tip» blir vist. Implementering av lagring av faktoriserte matriser er også påbegynt. I tillegg er det lagt til flere metoder for at brukere kan velge responsen man vil se på i forkant av en analyse.
18.03	20-23	ComputationalModel.LoadTrain.cs ComputationalModel.Analyzes.cs	Fant en feil i posisjoneringen av lasttoget og brukte en del tid på å finne og rette denne. I tillegg ble nye metoder for benyttelse av faktoriserte matriser implementert, analysen gjennomføres mye raskere nå.
19.03	11-22	Flere klasser har blitt kommentert	Møte med alle i dag om blant annet hva oppgavene etter påske vil innebære og diskusjon om virkemåten til programmet. Det ble fastsatt at omhyllingskurver fremdeles er min oppgave fremover. For min del ble også de siste funksjonalitetene til «sanntids» beregninger presentert. For å sikre at koden er utviklervennlig og videreførbar må metoder og egenskaper kommenteres på ulike plasser i koden. Det er per i dag over 4000 plasser i koden som mangler kommentering. Jobben med å kommentere og forbedre koden har gått til meg, derfor har mye av dagen gått med på å kommentere, ved slutten av dagen var det bare 2750 plasser som var uten kommentarer.
08.04	10-15	Flere klasser har blitt kommentert	Antallet plasser som mangler kommentering er nå redusert til rundt 1600.

09.04	10-17	Flere klasser har blitt kommentert	Møte i dag med alle unntatt Erik. Gjennomgang av diverse problemer som er oppdaget i forhold til blant annet elementinndelingen i **fap2D**. Nye oppgaver har blitt tildelt hver person avhengig av hvilke problemer som er funnet. Jeg har fått i oppgave å undersøke mulighetene for å lage en lisensserver. Resten av dagen gikk med på å skrive kommentarer til klasser i **fap2D**.
10.04	11-19	Flere klasser har blitt kommentert	Mer kommentering. I tillegg brukte jeg en god del tid på å undersøke mulighetene for å lage lisensserver for **fap2D** ved å spørre blant de på NTNU som har fungerende systemer for lisenskontroll.
11.04	11-20	Flere klasser har blitt kommentert	Alle elementer i koden er nå ferdig kommentert. Det var over 4000 til sammen.
12.04	12-16	ComputationalModel.cs ComputationalModel.Analyzes.cs ReactionForce.cs FrmModel.Infotext.cs	Møte med Kolbein I dag for å diskutere ulike feil I programmet og hvordan de skal løses og hva som skal være fremtidige prioriteringer. Omhyllingskurver er ikke lenger en prioritet, det skal heller satses på å ordne «sanntids» beregninger og influenslinjer skikkelig. I tillegg har reaksjonskrefter blitt endret i programmet slik at de tegnes i skala til største opptredende last.
14.04	20-00	ComputationalRealTimeLoadTrain.cs ComputationalModel.LoadTrain.cs ComputationalModel.cs FrmMainGui.cs FrmMainGui.ResultsPage.cs StructuralModel.cs	Arbeidet med å finne ekstreme responser for reaksjonskrefter har blitt påbegynt. Systemet inneholder fremdeles noen feil.
15.04	10-20	ComputationalModel.cs ComputationalModel.Analyzes.cs ComputationalModel.Drawing.cs ComputationalModel.LoadTrain.cs ComputationalRealTimeLoadTrain.cs FrmMainGui.ResultsPage.cs FrmMainGui.AnalysisPage.cs	Reaksjonskrefter for alle analyser blir nå tegnet I korrekt retning og I skala for alle analyser. De har også fått sine egne tool tips for å informere brukere om reaksjonskreftene. Ekstremresponser blir nå funnet når brukere velger hvilket tog som skal betraktes, hvis de ikke allerede er funnet fra før. Ekstremresponser for reaksjonskrefter fungerer også nå. Animasjoner kan ikke lenger kjøres hvis toget er på enden av kjørebanen. Arbeidet med å gjøre

		FrmMainGui.cs FrmModel.cs FrmModel.InfoText.cs ReactionForce.cs	lasttogposisjoner om til lasttilfeller er også påbegynt.
16.04	11-19	ComputationalModel.LoadTrain.cs FrmMainGui.ResultsPage.cs FrmMainGui.AnalysisPage.cs FrmMainGui.cs StoringUtilities.cs ModelDataSet.Designer.cs LocalStructuralSettings.cs FrmModel.cs FrmRealTimeLoadTrainDialog.cs FrmInfluenceLineLoadTrainDialog.cs FrmInfluenceLineLoadPathDialog.cs FrmRealTimeLoadPathDialog.cs	Mange mindre endringer I dag. Metoden for å gjøre lasttogets posisjoner om til lasttilfeller er ferdig. I tillegg til at antall lasttogposisjoner lagres sammen med hver modell. Det har også blitt fjernet mange feil ved oppretting av lasttog og lastbaner for både «sanntids» beregninger og influenslinjer. Når brukere velger å se på detaljer ved elementer for influenslinjer er det aktive elementet merket på modellen.
17.04	10-17	FrmInfluenceLineLoadPathDialog.cs LocalStructuralSettings.cs FrmXtrmDialog.cs PointLoad.cs StructuralModel.cs ComputationalModel.Tables.cs FrmMainGui.cs	Oppretting av feil knyttet til influenslinjer i ble gjennomført i dag. Siden influenslinjene er beregnet på bakgrunn av en enhetslast på -1 kN, må ekstrem analysen for lasttogene for influenslinjer beregnes relativ til denne, dette ble ordnet. Laster i lasttog for influenslinjer blir nå tegnet i skala i forhold til hverandre. Dialogboksen for ekstreme responser for influenslinjer er blitt rettet på. Sendte også mail til Kolbein for å forhøre meg om hvordan andre feil ved influenslinjer skal håndteres.
18.04	11-19	FrmInfluenceLineResultsDialog.cs ResponseParameter.cs FrmMainGui.cs FrmInfluenceLineLoadTrainDialog.cs	Brukte masse tid I dag på å fjerne en feil knyttet til opprettelsen av en tabell for responsparametre som brukes I mange analyser. Problemet var å knytte rett responsparameter til rett element. Løsningen var at **fap2D** ikke hadde kode forå håndtere grensetilfellet hvor et element står loddrett og dermed ble elementene valgt tilfeldig. Dette er nå fikset.

			Videre er det fjernet en del feil knyttet til tegningen av «sanntids» beregninger og opprettelse av nye tog knyttet til influenslinjer. Arbeidet med å finne en passende programvare for en lisensserver er også påbegynt.
19.04	11-15		Dagens arbeid gikk med på å finne kandidater for en passende lisenstjeneste som kan brukes i samarbeid med **fap2D**. De sterkeste kandidatene er en tjeneste fra samme leverandør som leverer installeringstjenesten vi tenker å kjøpe. Den andre er en relativt billig og innbydende tjeneste som vi ikke har et forhold til fra før. Den første tjenesten er også den samme som blir brukt på institutt for produktutvikling og materialer.
20.04	20-22		Begynte å skrive en del om funksjonaliteten til implementasjonen i masterrapporten.
22.04	10-16		Møte med Kolbein i dag for diskutere endringene som har blitt gjort på influenslinjer og «sanntids» beregninger i forrige uke. Alle saker som kun angår mitt arbeid ble tatt opp i dag, slik at man kan fokusere på fellessaker på tirsdagens møte. Utover dette har det blitt skrevet mer om implementasjonens funksjonalitet i rapporten.
23.04	10-15		Møte med alle i dag hvor feilene rundt valg av elementer for responsparametre ble tatt opp, dette problemet trenger fremdeles en del oppmerksomhet også etter møtet. Videre ble det også diskutert en del om valg av lisenstjeneste. Resten av dagen gikk med på å skrive om funksjonalitet i rapporten.
24.04	10-18	ComputationalModel.LoadTrain.cs	Mye av dagen ble brukt til å feilsøke influenslinjeberegninger for å finne feil i Mpath. Arbeidet endte opp med å ordne sorteringsmetoden for noder i lastbanen for influenslinjer. Den skal fungere som påtenkt nå.

			Det er fremdeles problemer med responsparametre og tegning av modellen for influenslinjer. Mye tid ble også brukt på søket etter passende lisenstjener i samarbeid med Kristian.
25.04	10-19	ResponseParameter.cs HelpNode.cs	Var i dialog med Flexera om vi skulle kjøpe lisenstjeneren de selger, men de ville ha 20 000 NOK for den. Derfor bestemte eg og Kristian oss, i med samtykke fra Kolbein, for at vi begynner å jobbe med et billigere program. Programmet blir uansett ikke kjøpt før vi vet at vi kan få det til å fungere for **fap2D**. Problemet med å tilegne korrekt responsparameter til korrekt element ble løst i dag ved at hver responsparameter får tilknyttet elementet som grenser til knutepunktet sitt. Dermed har alle responsparametre fått entydig definert sitt element. Dette ble bestemt gjennom møter med Kolbein i dag. Resten av dagen gikk med på å prøve å finne en løsning som gjør at influenslinjer kan tegnes korrekt selv om en *member* blir rotert om sitt endepunkt. Dette er fremdeles ikke løst.
26.04	12-18	ComputationalModel.Drawing.cs ComputationalModel.Analyzes.cs FrmMainGui.ResultsPage.cs FrmMainGui.cs	Hadde et lengre møte med Kolbein i dag om hvordan resultatvisningen for influenslinjer skal være på elementer som blir satt ut ettersom de blir rotert. Konklusjonen ble at de fleste responsparametre blir tegnet rett, unntatt moment som trenger å skrifte fortegn fordi lokalt aksesystem også roteres når en *member* roteres. Siden influenslinjer blir beregnet etter lokalt aksesystem, trengte de derfor å endre fortegn når *member* roteres. Det skal også testes videre om fortegnet også trenger å skiftes for flere typer responsparametre. Videre har tegnemetoden blitt endret for å ta hånd om roterte *members* for influenslinjer. Det er nå også mulig å se fortegnet på den mest ekstreme responser i influenslinjer og å velge antall lasttogposisjoner for ekstremanalyse for lasttog på influenslinjer.
29.04	10-18	ComputationalModel.LoadTrain.cs	Fjernet «Tool tips» for reaksjonskrefter, dette var overflødig siden samme informasjon var

		ComputationalModel.cs ComputationalRealTimeLoadTrain.cs ComputationalRealTimeLoadTrainLoad.cs FrmModel.InfoText.cs FrmModel.cs	tilgjengelig ved å høyreklikke det tilhørende opplageret. Interpolasjonsmetoden har blitt forbedret, slik at kun noder som er på samme *member* som den aktuelle lasten blir valgt som interpolasjonspunkter. Det var tidligere problemer med at metoden valgte feil noder i overgangen til nye *members*. En feil som hindrer at alle noder i lastbanen ble inkludert er også blitt fjernet. Resten av dagen gikk med til å skrive om funksjonalitet i rapporten.
30.04	12-19		Møte med alle i dag hvor ukens arbeid ble diskutert og problemer med siste versjon av **fap2D** ble diskutert. Det ble også oppdaget noen feil knyttet til ekstremresponsen for influenslinjer som trenger å fikses. Dette ble min jobb. Resten av dagen gikk med på å skrive i masterrapporten.
06.05	12-18	ComputationalModel.Analyzes.cs FrmMainGui.ResultsPage.cs FrmXtrmDialog.cs ComputationalModel.Tables.cs Frame2D.cs	Møte med Kolbein i dag om endringer i Xtrm-analysen for å sikre at alle resultater har korrekt fortegn og størrelse. Løsningen ble å lansere en endring i **Frame2D** som tar hånd om dette. I tillegg gikk en del tid med på å fikse mindre funksjonelle feil og skrivefeil i influenslinjeanalysen. Resten av dagen gikk med på å ordne en feil knyttet til valg av *members* for responsparametre. Denne feilen ble rettet av å oppgradere skjermkortdriverene på det innebygde kortet på min maskin. Feilen lå med andre ord i et skjermkort som ikke var kompatibelt med siste versjon av OpenGL.
07.05	14-21		Møte med alle i dag hvor ulike feil i programmet ble diskutert. For min del gjaldt dette oppgradering av driverene skjermkortet for å forhindre tegnefeil i **fap2D**. Brukere av programmet skal nå anbefales å oppgradere sine skjermkortdrivere for å være kompatible med OpenGL 2.1. Videre ble problemet med korrekt fortegn for influenslinjene fikset ved å lansere en ny versjon av **Frame2D** som håndterer

			dette. Lisenstjenere ble også diskutert. Det ble bestemt at det ikke er en oppgave for årets utviklere og legges derfor på is. Til slutt ble det diskutert metoder for å finne ut om et punkt er på en gitt submember. Dette kom vi ikke frem til noe løsning på i dag og resten av dagen min gikk med til å finne løsninger for dette.
08.05	11-16	ComputationalModel.LoadTrain.cs	Møte med Kolbein for å finne løsningen på problemet med å finne korrekt submember til korrekt punkt. Løsningen ble å måle avstanden til punktet inne på en *member* og deretter sammenligne avstanden med lengdene på de ulike submembers fra starten av *member* for å finne ut hvilken som inneholder punktet. Dette ble implementert i dag. I tillegg var det en feil i posisjoneringsmetoden som gjorde at laster på slutten av lastbanen ikke fikk tilegnet korrekt *member* ved tilbakestilling. Denne feilen ble også fikset.
09.05	14-19		Skrev mer på rapporten om funksjonaliten til «sanntids» beregninger.
10.05	14-16		Skrev mer på rapporten om bruk av analysene som ble implementert i denne masteroppgaven.
11.05	12-19		Skrev om bruk av nye analyser i rapporten.
12.05	12-19		Skrev om definisjoner, innledning, arbeidsprosessen og utviklingen i masterrapporten.
14.05	13-16	FrmMainGui.cs FrmMainGui.ResultsPage.cs	Møte med alle I dag og mindre endringer I grensesnittet ble diskutert. I tillegg til å avtale en dato for demonstrering av programmet. Resten av dagen gikk med på å implementere det endringene i grensesnittet.
15.05	12-19		Skrev om utviklingen og funksjonaliteten til influenslinjer i masterrapporten. Møte med Kolbein for å diskutere innholdet i brukermanualen i forhold til det som allerede er skrevet om funksjonaliteten til «sanntids» beregninger. Møtet var først og fremst for å unngå misforståelser i teksten.
16.05	12-18		Skrev om utviklingen og funksjonalitet i masterrapporten.
18.05	19-24	ComputationalModel.Draw	Skrev om testing i rapporten og fikset mindre feil

		ing.cs FrmMainGui.AnalysisPage.cs	relatert til tegning og feilmeldinger i programmet.
19.05	12-20		Skrev om testing i rapporten
20.05	12-19		Skrev om testing i rapporten
21.05	12-19		Skrev om testing i rapporten og formaterte rapporten.
22.05	12-18		Skrev om testing i rapporten
23.05	12-18		Skrev om videre arbeid og vedlegg i rapporten
24.05	10-17		Møte med alle i dag hvor vi diskuterte mindre feil som har blitt funnet på programmet, i tillegg til hva som skal være innhold i rapporten og rutiner for innlevering. Det ble også bestemt at årets implementasjoner i **fap2D** skal presenteres i neste uke. Resten av dagen gikk med på å lage diagrammer for å beskrive dataflyten i implementasjonene og å skrive sammendrag i rapporten.
26.05	11-19		Arbeid med SVN-instruksjonene, tips og triks for fremtidige utviklere, kodekonvensjoner i rapporten. I tillegg til en del formatering av teksten.
27.05	19-23		Formatering og retting i rapporten.
28.05	11-19		Formatering og korrektur i rapporten.
29.05	13-20		Møte med alle i dag hvor mindre feil på programmet og planen for fredagens presentasjon ble diskutert. Resten av dagen gikk med på korrekturlesning og formatering av rapporten.
30.05	12-20		Korrekturlesing og formatering i rapporten. Arbeidet også med å tilrettelegge morgendagens presentasjon.
31.05	12-17		Presentasjon av fap2D til instituttet. Resten av dagen gikk med til forbedringer og korrekturlesing i rapporten.
01.06	13-19		Korrekturlesing og forberedelse til innlevering.
02.06	14-19		Korrekturlesing i rapporten.
03.06	12-16		Fiksing i rapporten.
06.06	12-17		Retting i rapporten.
07.06	15-20		Retting i rapporten.

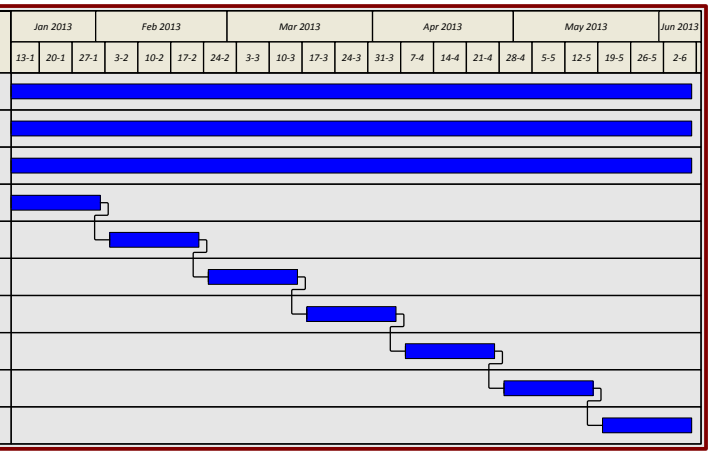
B. Ganttdiagram

Ganttdiagrammet viser hvor mye tid som er lagt ned i hver aktivitet i løpet av masteroppgaven.

B.1. Planlagt fremdriftsplan

Den planlagte fremdriftsplanen er en grovskisse for hvordan arbeidet skal gjennomføres.

ID	Task Name	Start	Finish	Duration
1	Skrive rapport	14-Jan-13	07-Jun-13	105d
2	Loggføre implementasjon	14-Jan-13	07-Jun-13	105d
3	Kode "sanntids" beregninger	14-Jan-13	07-Jun-13	105d
4	Lage klasse for "sanntids" beregninger	14-Jan-13	01-Feb-13	15d
5	Håndtere analysen	04-Feb-13	22-Feb-13	15d
6	Legge til rette for load path/load train	25-Feb-13	15-Mar-13	15d
7	Legge til rette for/forbedre visualiseringen	18-Mar-13	05-Apr-13	15d
8	Legge til rette for valg av responsparameter	08-Apr-13	26-Apr-13	15d
9	Implementere mulighet til å spole til tidligere beregninger	29-Apr-13	17-May-13	15d
10	Implementere mulighet til å finne ekstrem respons	20-May-13	07-Jun-13	15d

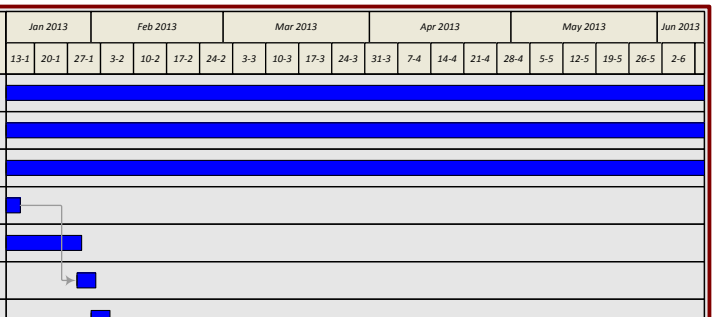


Figur 61: Ganttdiagram over fremdriftsplanen for arbeidet med masteroppgaven

B.2. Faktisk utført arbeid

Ganttdiagrammet viser omtrentlig utført arbeid og tar ikke hensyn til helger og ferier. Aktivitetene representerer de mest fremtredende trendene i løpet av arbeidet og er laget med bakgrunn i hendelsene i loggen.

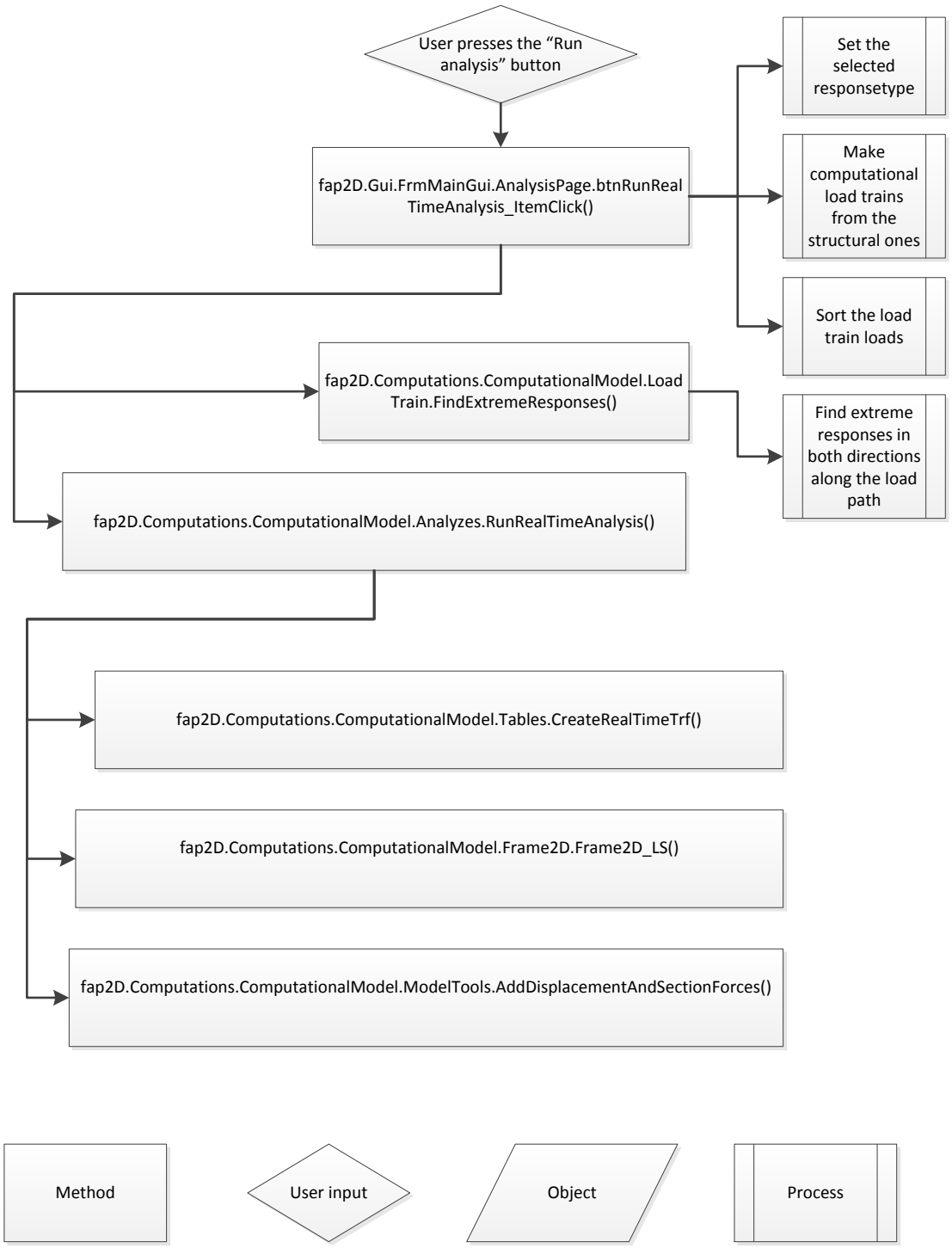
ID	Task Name	Start	Finish	Duration
1	Skrive rapport	14-Jan-13	10-Jun-13	21w 1d
2	Loggføre implementasjon	14-Jan-13	10-Jun-13	21w 1d
3	Kode implementasjonen	14-Jan-13	10-Jun-13	21w 1d
4	GUI-knapper for "sanntids" beregninger	14-Jan-13	16-Jan-13	3d
5	Gjennomgåelse av kode	14-Jan-13	29-Jan-13	2w 2d
6	Arbeid med dialogbokser	29-Jan-13	01-Feb-13	4d
7	Arbeid med resultatvisningen	01-Feb-13	04-Feb-13	2d
8	Arbeid med å lage togkontroller	07-Feb-13	12-Feb-13	4d
9	Arbeid med lasttogposisjoner, posisjonering av toget og interpolasjon	13-Feb-13	07-Mar-13	3w 2d
10	Arbeid med å finne ekstreme responser og maksimalverdier langs lastbanen	08-Mar-13	19-Mar-13	1w 3d
11	Kommentering av klasser	08-Apr-13	11-Apr-13	4d
12	Arbeid med reaksjonskrefter og konvertering til lasttilfeller	11-Apr-13	16-Apr-13	4d
13	Arbeid med influenslinjer og problemer med responsparametre	17-Apr-13	07-May-13	3w
14	Arbeid med å finne en passende lisenstjener for fap2D	19-Apr-13	07-May-13	2w 3d
15	Ferdigstilling av rapporten	22-Apr-13	10-Jun-13	7w 1d
16	Arbeid med rettelser I programmet på bakgrunn av tilbakemeldinger og testing	22-Apr-13	10-Jun-13	7w 1d



Figur 62: Ganttdiagram over faktisk utført arbeid

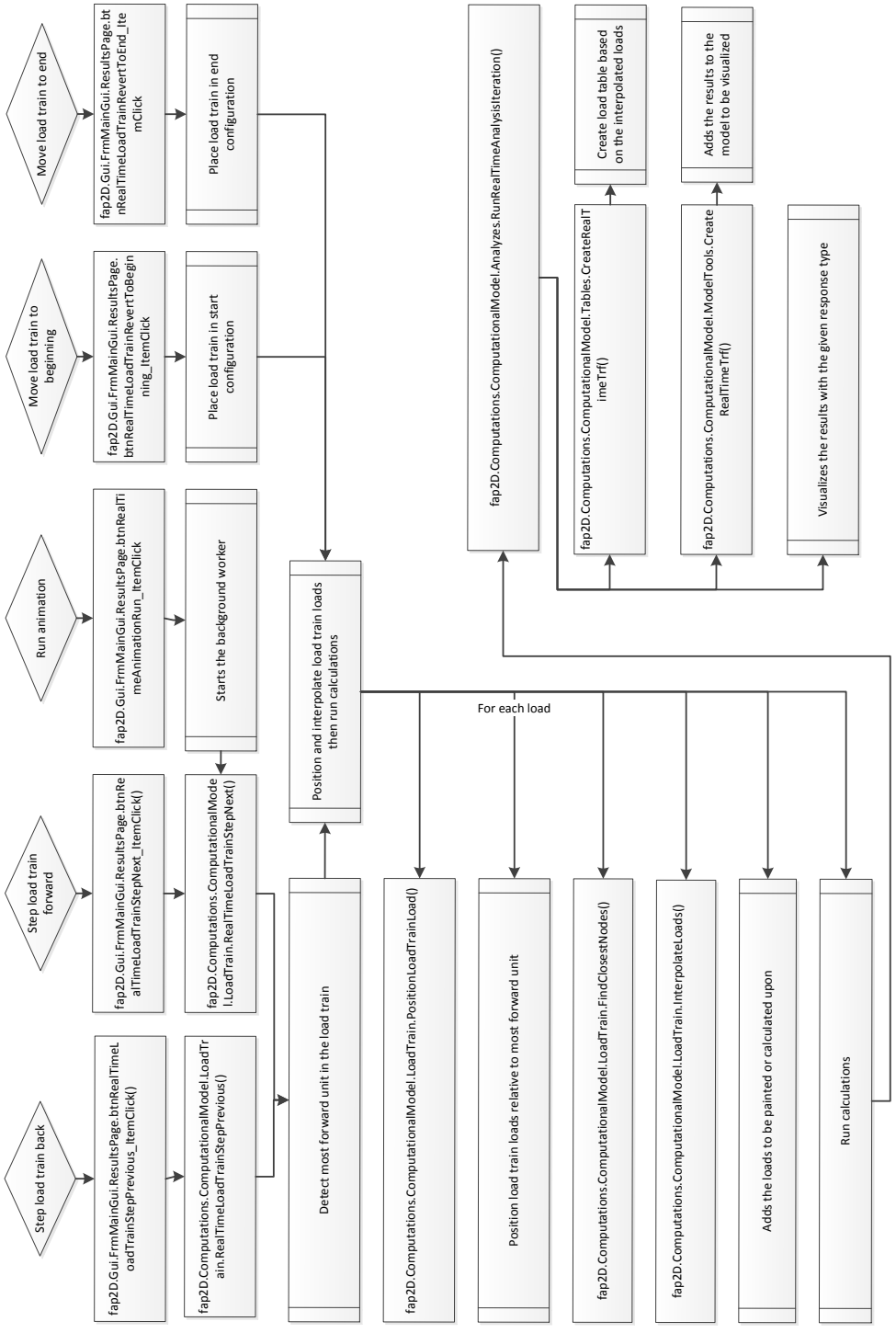
C. Kodens struktur

Diagrammene viser grafisk hvordan koden som har blitt implementert er satt sammen. Oversikten under viser metodekallene som gjennomføres når man starter «sanntids» beregninger. Tegnforklaringen i dette diagrammet gjelder for alle diagrammer i dette vedlegget.



Figur 63: Oversikt over metodekall i «Analysis»-fanen for «sanntids» beregninger

Figur 64 viser en overordnet oversikt over prosessene og metodekallene som gjennomføres hver gang en bruker velger en operasjon som forflytter lasttoget. Oversikten illustrerer at operasjonene har nokså lik virkemåte.



Figur 64: Oversikt over metodekallene ved forflytting av toget under «Results»-fanen for «sanntids» beregninger

D. Kodekonvensjoner

Vedlegget inneholder en oversikt over kodekonvensjonene som har blitt benyttet av alle utviklere.

D.1. Navnekonvensjoner:

Tabell 12: Navnekonvensjoner

Navnekonvensjon	Bruksområde
camelCasing	Brukes for å navngi variabler som ikke er private.
_camelCasing	Understrek brukes til å navngi lokale og private variabler.
KONSTANT	Konstante verdier navngis med kun store bokstaver.
PascalCasing	Pascal casing brukes for alle klasser, typer, metoder og navnerom, samt for navn som består av flere ord.
PrePascalCasing	Valgfri prefiks brukes for å indikere at klasser arver fra en klasse i standard biblioteket. Prefikset angir hvilken klasse de arver fra.

D.2. Kommentering

Alle metoder og events skal kommenteres. Eksempel:

```
/// <summary>
/// Implements the operator +.
/// </summary>
```

Det skal deretter komme en liste over eventuelle parameternavnene som benyttes i metoden / eventet og hva som returneres. Eksempel:`/// <param name="lv1">The first operand.</param>`

```
/// <param name="lv2">The second operand.</param>
/// <returns>The result of the operator.</returns>
```

Alle klasser skal kommenteres. Dette gjøres på lik måte som kommenteringen av et event eller metode. Eksempel:

```
/// <summary>
/// The settings form class.
/// Inherits from XtraForm.
///
/// Note:
/// Text
/// </summary>
```

Enums og Structs skal kommenteres, inkludert alle alternativene. Eksempel:

```
/// <summary>
/// The type of cursor
/// </summary>
public enum CursorType
{
    /// <summary>
    /// A beam member cursor.
    /// </summary>
    BeamMember,
    /// <summary>
    /// A bar member cursor.
    /// </summary>
    BarMember,
}
```

Det skal etterstrebes å kommentere koden i den grad det er nødvendig. Kommentering av selve koden skal foregå over koden, uten linjeskift mellom. Eksempel:

```
// Add mode shape displacement to nodes
AddModeShapeDisplacement();
```

Dersom det er nødvendig å kommentere variable gjøres dette etter variabelnavnet, for eksempel:

```
private ToolTip _toolTip; // The tooltip
```

Getter og settere skal kommenteres på følgende måte:

```
/// <summary>
/// Gets or sets the structural model.
/// </summary>
/// <value>The structural model.</value>
```

Dersom getter og setteren har parametre skal parameteren kommenteres på lik måte som i en metode / event. Eksempel:

```
/// <summary>
/// XXX
/// </summary>
/// <param name="i">XXX</param>
/// <value>XXX</value>
public double this[int i]
{
    get { return _matrix[i]; }
    set { _matrix[i] = value; }
}
```

Annet:

- Punktum etter param beskrivelsen, på slutt av summary, på slutt av return og på slutt av value.
- Det skal forekomme et mellomrom mellom /// og beskrivelsen

D.3. Regioner

Koden skal deles opp i regioner. Dette er for å få en mer oversiktlig kode. Regionene som koden skal deles opp i, og rekkefølgen, er:

- README
- Enums
- Structs
- Fortran import
- .dll import
- Names
- Variables
- Constructors
- Methods
    - Private methods (kun hvis andre eksisterer)
    - Internal methods (kun hvis andre eksisterer)
    - Protected methods (kun hvis andre eksisterer)
    - Virtual methods (kun hvis andre eksisterer)
    - Public methods (kun hvis andre eksisterer)
        - Analysis
            - En region per analyse
        - Other (kun hvis Analysis eksisterer)
- Events
    - Analysis
        - En region per analyse
    - Other (kun hvis Analysis eksisterer)
- Properties
- Interfaces
- Unsupported Properties/Methods etc.
- Abstract Methods

Det skal ikke forekomme tomme regioner. Regioner skal ikke deles inn i subregioner med mindre dette er hensiktsmessig, eller hovedregionen overgår 200 linjer.

Annet:

- Det skal forekomme et mellomrom mellom # og region / endregion

D.4. Linjeskift

Det skal etterstrebes å ikke ha for mange linjeskift i koden. Dette er for å få koden til å bli så kompakt som mulig, uten at koden blir uoversiktlig.

Det skal forekomme et linjeskift helt på slutten av en klasse. Eksempel:

```
        }
        #endregion
    }
}
```

Hver if-setning skal ha et linjeskift over og under seg. Eksempel:

```
    int numFrames = (int)e.Argument / 2;

    if (counter == 0)
        maxScaleFactor = Math.Abs(ComputationalModel.ResultsScaleFactor);

    double scaleFactor = ComputationalModel.ResultsScaleFactor;
```

Det skal ikke forekomme linjeskift mellom if og else if / else setninger. Eksempel:

```
    if (distance1 <= distance2)
    {
        snapX = Grid.GridLineDistanceX * atXGridLineNo;
        xFound = true;
    }
    else
    {
        snapX = Grid.GridLineDistanceX * (atXGridLineNo + 1);
        xFound = true;
    }
```

Det skal ikke forekomme et linjeskift over en if-setning hvis den er på innrykk. Eksempel:

```
    {
    if (scaleFactor > maxScaleFactor)
        {
            scaleFactor = maxScaleFactor;
            increment *= -1;
        }
    }
```

Dersom if-setningen skrives på 2 linjer skal det forekomme et linjeskift mellom if og else setningen. Eksempel:

```
    if (_drawableObjectClicked is Element)
        _drawableObjectClicked.IsSelected = false;

    else
        _drawableObjectClicked.IsSelected = true;
```

Hver foreach og for-løkke skal ha linjeskift over og under seg. Eksempel:

```
    bool toBeDeleted = false;

    foreach (SubMember sm in m.SubMembers)
        if (sm.IsSelected)
            toBeDeleted = true;

    if (toBeDeleted && !membersToDelete.Contains(m))
        membersToDelete.Add(m);
```

Return skal ha linjeskift over seg. Eksempel:

```
        private bool OpenConnection()
        {
            _connection = new OleDbConnection(_connectionString);
            _connection.Open();

            return true;
        }
```

Lukking av dialogbokser, altså this.close();, skal ha et linjeskift over seg. Eksempel:

```
        _frmMainGui.CurrentFrmModel.RePaintGLForm();

        this.Close();
    }
```

Det skal ikke forekomme variable som rett under et innrykk. Da skal det være et linjeskift over.
Eksempel:

```
        scaleFactor += increment;
    }

    ComputationalModel.ResultsScaleFactor = scaleFactor;
```

Det skal ikke forekomme noen linjeskift nedenfor #region og metoden / eventet etc. som kommer etter.
Eksempel:

```
        #region Events
        /// <summary>
        /// Handles the Click event of the btnOk control.
        /// </summary>
```

Dette gjelder også linjeskift over #endregion og metoden / eventet som kommer før. Eksempel:

```
            btnOk.Focus();
            base.OnActivated(e);
        }
        #endregion
```

Dersom det er en region som starter rett inni regionen skal det forekomme et linjeskift mellom disse. Eksempel:

```
#region Methods

#region Private methods
```

Det samme gjelder på slutten av en region. Eksempel:

```
}
#endregion

#endregion
```

D.5. Properties

Properties skal skrives på følgende måte:

```
public CursorType CursorType
{
    get { return  cursorType; }
    set {  cursor = value; }
}
```

Tomme properties skrives på følgende måte:'

```
public CursorType CursorType { get; set; }
```

D.6. Interface

Interface implementerers ved klasseavn : interface. Eksempel:

```
public class JointList : CollectionBase, IBindingList
```

D.7. Initialisering av dialogbokser

Ved initialisering skal hoveddialogboksen få fokus, ikke en controller.

D.8. Usings

Det skal etterstrebes å kutte ned på ubrukte usings. Dette gjøres lettest i Visual Studio ved å høyreklikke i koden → Organise Usings → Remove and Sort. Alle usings blir dermed tatt bort og alle usingsene blir sortert.

D.9. Generell opprydning av kode

For å rydde opp generelt i koden kan man trykke på Edit → Advanced → Format document i VS

D.10. Innkapsling

Alle variable bør innkapsles via gettere og settere.

D.11. Switch

Switch statements skrives på følgende måte:

```
switch (_parent.CurrentFrmModel.StructuralModel.LocalStructuralSettings.A)
{
    case EigenValueAlgorithm.subspace:
        radioGroupEigenValueAlgorithm.EditValue = 0;
        break;
    case EigenValueAlgorithm.lanczos:
        radioGroupEigenValueAlgorithm.EditValue = 1;
        break;
    default:
        break;
}
```

E. Software og hardware

E.1. Utvikling

Utviklingen av **fap2D** og rapportskrivningen har blitt gjennomført med følgende programvare:

- Visual Studio 2012 Ultimate[7] (32-bit) som utviklingsområde
- Windows 8[6] (64-bit) som plattform.
- DevExpress[9] ble brukt i sammenheng med Visual Studio[7] for utvikling av brukergrensesnitt.
- PuTTy[8] ble brukt for å administrere tilgang til serveren hvor koden ble delt.
- Focus Konstruksjon[4] ble brukt som sammenligningsgrunnlag for **fap2D** i tester.
- TortoiseSVN[10] ble brukt for å dele kode mellom utviklerens datamaskiner og server.
- Word[11] ble brukt som tekstbehandler for å redigere rapporten
- Visio[12] ble brukt til å lage utviklings- og kodediagrammer for rapporten.
- Excel[13] ble brukt for å behandle verdiene fra testingen og lage grafer for rapporten.
- Tao Framework[14] ble brukt som en utvidelse til .NET-pakken under utviklingen
- .NET 4.5[15] er programmeringsspråket som ble brukt under utviklingen.
- Installshield 2012[17] er programvaren som har blitt brukt for å lage installasjonen til **fap2D**.

Datamaskinen som ble brukt har de følgende hardware-spesifikasjonene:

Tabell 13: Hardware-spesifikasjoner for datamaskinen brukt i denne masteroppgaven

Enhet	Type
Prosessor	Intel Core i5-2540M CPU 2.60 GHz
RAM	8 GB
Grafikkort	Nvidia GeForce GT 555M
Harddisk	Intel SSDSA2CW160G3

fap2D har de følgende kravende til hardware og software ved masteroppgavens slutt.

E.2. Krav til hardware

Tabell 14: Krav til hardware for fap2D

Enhet	Krav
Prosessor	1.5 GHz eller raskere
Minne	2 GB eller mer
Harddisk	Minimum 80 MB ledig diskplass
Skjerm	Minimum oppløsning på 1366*768
Grafikkort	Skjermkort med støtte for OpenGL 2.1

Det anbefales å ha nyeste driver til skjermkort installert. De nyeste driverne kan installeres fra nettsiden til skjermkortleverandøren. For å finne ut hva slags skjermkort pcen har, høyreklikk på Min Datamaskin og gå inn på Egenskaper. Klikk videre inn på Enhetsbehandlig som ligger i en liste til venstre under egenskapene. Under Enhetsbehandling finner man ut hvilken leverandør skjermkortet har.

Videre er det å gå inn på nettsiden til den leverandøren skjermkortet har og følge instruksjoner der for å laste ned nyeste driver.

Intel: http://www.intel.com/p/en_US/support/detect/graphics

Nvidia: http://www.nvidia.com/Download/index.aspx?lang=en-us

AMD: http://support.amd.com/us/gpudownload/Pages/index.aspx

E.3. Krav til software

fap2D er utviklet til bruk i Windows Vista, Windows 7 og Windows 8. Det er ikke støtte for Windows XP eller eldre utgaver.

Det kreves at .NET 4.5 er installert. Er det ikke allerede installert kan det enkelt installeres da dette følger med installasjonen til **fap2D** i en egen mappe.

Microsoft Office Access Runtime Engine kreves installert I tillegg. Denne er inkludert I Microsoft Office. Er ikke Microsoft Office allerede installert kan Microsoft Office Access Runtime Engine installeres på egen hånd, og denne installasjonen er inkludert i installasjonen til **fap2D** i en egen mappe.

F. Forklaring av koden

F.1. Prosjekter

Fap2D er bygd opp av 9 prosjekter, delt inn i hva slags formål de har. Det er `fap2D`, fap2D.Computations, `fap2D.Design`, `fap2D.Gui`, `fap2D.ModelData`, `fap2D.Storing`, `fap2D.StructuralData` og `fap2D.Utilities` og `Installer`. `fap2D`-prosjektet har kun en klasse, den består av en `main`-metode som kjører en ny instans av klassen `FrmMainGui` som ligger i prosjektet `fap2D.Gui.`

Fap2D.Gui

Dette er det klart største prosjektet og her ligger all koden til brukergrensesnittet. Når programmet starter er det `FrmMainGui` som starter opp. Det er en meget stor klasse som er delt inn i 14 delklasser. Her er all koden til hovedvinduet og boksene under de forskjellige fanene.

Alle dialogbokser som åpnes opp ligger i mappa `FrmDialogs`, de er delt inn etter hvilken fane de hører til.

Modellvinduene som man kan ha flere av åpne i FrmMainGui er instanser av klassen FrmModel. `FrmModel` er et lerret hvor alt av tegningen foregår og inneholder *events* for interaksjonen til muspekeren. `FrmModel` inneholder en instans av klassen `StructuralModel`, dvs hele konstruksjonsmodellen, og `ComputationalModel` som er resultater basert på konstruksjonsmodellen.

Fap2D.StructuralData

Her ligger all koden til konstruksjonsmodellen. Laster, material, tverrsnitt, *members*, *joints*, punktmasse, fjær osv. Alt er delt inn I mapper etter hvor de hører til. Klassen `StructuralModel` inneholder hele konstruksjonsmodellen som blir brukt i en modell åpnet i hovedvinduet.

Fap2D.Computations

Når man kjører en analyse så lages det en `ComputationalModel` som omgjør en `StructuralModel`, som den tar inn i konstruktøren som eneste parameter. `ComputationalModel` er delt inn i 9 delklasser og er en av de større klassene i prosjektet. Etter man har en `ComputationalModel` er det mulig å kjøre analyse på den som ligger under `ComputationalModel.Analyzes.`

Fap2D.Storing

Dette prosjektet inneholder alt av lagring. Klassen `ModelDataSet` er den klassen som inneholder alle felt som skal lagres med modellen, samtidig klassen GlobalSettingsDataSet som inneholder feltene som blir lagret av globale innstillinger. Klassen `StoringUtilities` inneholder metodene for lagring og åpning av en konstruksjonsmodell, og den tar nytte av `ModelDataSet` som har innebygde lagre og åpne metoder. `StoringUtilities` inneholder og metoder for å lagre og åpne de globale innstillingene. I dette prosjektet ligger og koden for *undo/redo* som det blir laget en instans av per `FrmModel`.

Fap2D.Utilities

Dette prosjektet inneholder mye hjelpeklasser og statiske klasser. Det er laget egne klasser for farge, matrise og punkter. De statiske klassene inneholder metoder som kan brukes overalt i prosjektet, f.eks `CurveUtils` inneholder metoder for å regne på buer. Her ligger og klassen som heter `MultiSampleGlControl`, som er hva som brukes i `FrmModel` til å tegne på. Det er en av de viktigste klassene i prosjektet og er en egenlagd *control* som openGL-metoder kan tegne på. Det er i tillegg klasse for kameraet og en statisk klasse med openGL-metoder.

Fap2D.ModelData

Dette prosjektet inneholder noen abstrakte klasser som igjen arves av flere andre klasser i andre prosjekter. Det inneholder og klassen som er ansvarlig for å tegne opp koordinatsystemet. Klassene i dette prosjektet brukes av alle andre prosjekter utenom `fap2D.Utilities`.

Fap2D.Design

Dette prosjektet inneholder alle klasser som er knyttet til kapasitetskontroll, som man kan utføre på en modell etter en analyse.

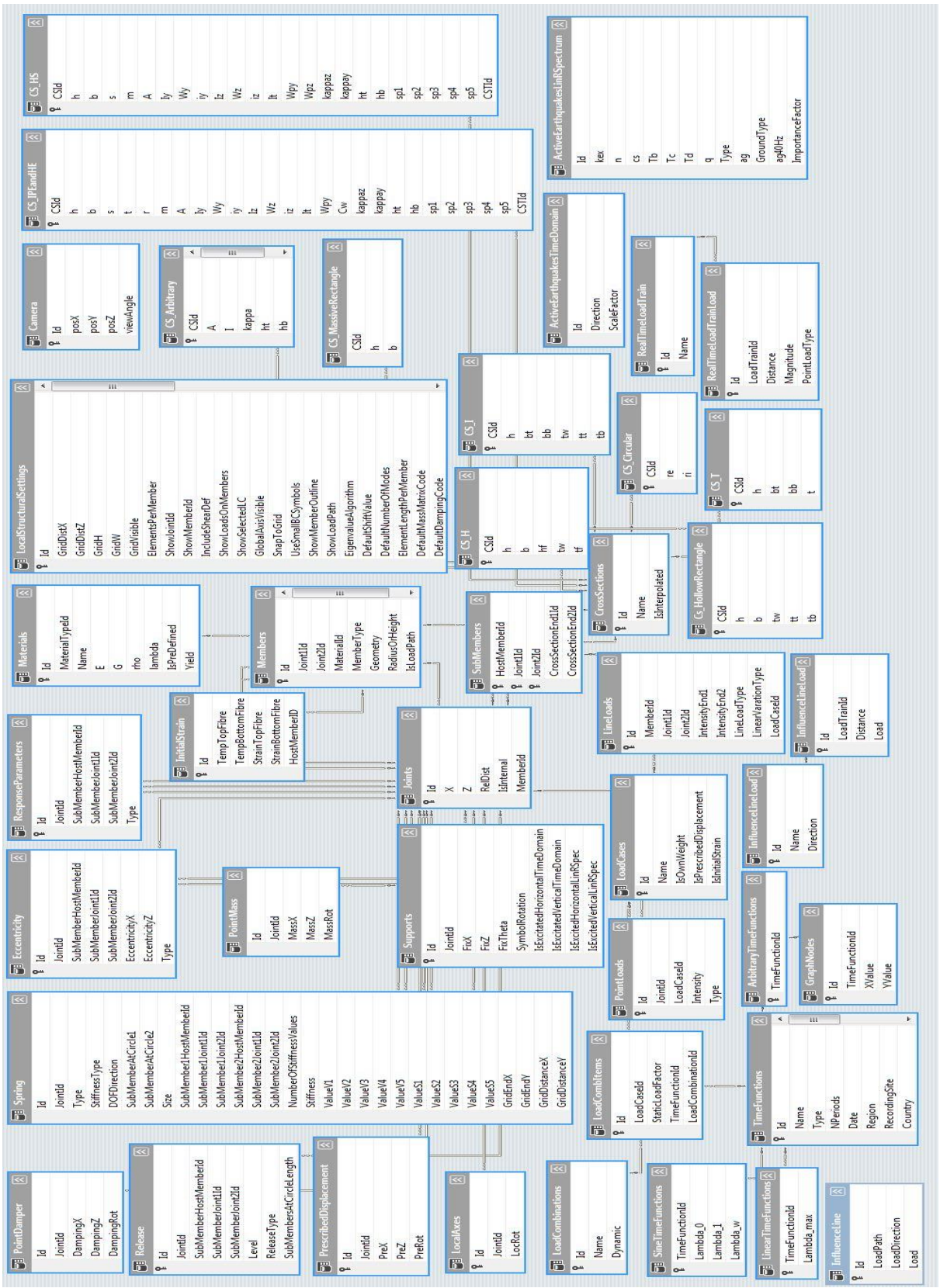
Installer

Dette prosjektet inneholder det som trengs for å lage en installasjon av programmet. Installshield 2012[17] må være installert for at denne skal fungere.

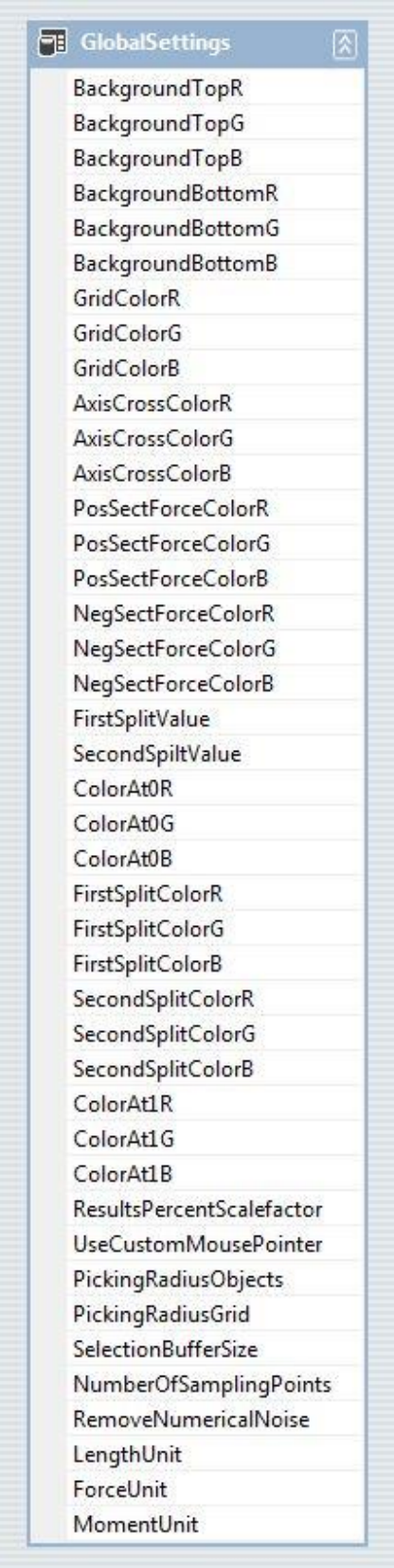
Alle nye og eksisterende metoder og *events* er blitt kommentert så det skal være enklere å forstå hva de gjør. Det er noen variabler som ikke er blitt kommentert, det er variabler som er blitt satt til `public` og som ikke blir aksessert gjennom `properties`.

F.2. Datasett

Figur 65 viser datasettet som brukes til lagring av modellen. Denne er blitt en del utvidet siden versjon 2.0, spesielt kolonnen `LocalStructuralSettings` er blitt en del større. Figur 66 viser datasettet som brukes til å lagre globale innstillinger som brukeren endrer i **fap2D Settings** under applikasjonsmenyen. Disse datasettene skriver og leser enkelt fra filer ved hjelp av kommandoene `WriteXml()` og `ReadXml()`.



Figur 65: Datasett for modeller



Figur 66: Datasett for globale innstillinger

F.3. Code metrics

Code metrics er en funksjon i Visual Studio som regner ut en del forskjellig om koden. Tabell 15 og Tabell 16 viser code metrics for respektivt versjon 2.0 og for versjon 3.0.

Maintainability index(MI)

Kalkulerer en indeks mellom 0 og 100 som representerer hvor enkelt det er å vedlikeholde koden. Jo høyere indeks, desto bedre. Denne bør ikke falle under 20, da blir det en vanskelig sak å vedlikeholde koden.

Cyclomatic complexity(CC)

Måler den strukturelle kompleksiteten til koden. Den blir skapt ved å beregne antall forskjellige kodebaner i flyten av programmet. Et program med høy kompleksitet vil kreve flere tester for å oppnå god kodedekning og vil være mindre vennlig mot vedlikehold.

Depth of inheritance(DOI)

Indikerer antall klassedefinisjoner som går til roten av klassens hierarki. Desto dypere hierarki, desto verre kan det være å forstå hvor koder og variabler er definert eller redefinert.

Class coupling(CCO)

Måler koplingen til unike klasser. Høy kopling kan indikere at designet er vanskelig å bruke om igjen på grunn av sin høye avhengihet av andre typer.

Lines of code(LOC)

Indikerer antall koder med linje i koden. Det er kun linjer med kode, så det er ikke nøyaktig antall linjer totalt i koden. Et veldig høyt tall indikerer at en metode kanskje prøver å gjøre litt for mye, og burde deles opp.

Tabell 15: Code metrics for versjon 2.0

Prosjekt	MI	CC	DOI	CCO	LOC
Fap2D	63	2	1	9	11
Fap2D.Gui	55	4455	9	563	31468
Fap2D.StructuralData	86	2287	3	125	4561
Fap2D.Computations	75	1736	2	107	4995
Fap2D.Storing	96	459	4	235	1888
Fap2D.Utilities	83	762	7	109	1985
Fap2D.ModelData	84	683	8	88	1534
Fap2D.Design	82	34	1	1	59

Totalt antall kodelinjer er da 46501. Totalt antall linjer inkludert all kommentering og annet er 157194.

Tabell 16: Code metrics for versjon 3.0

Prosjekt	MI	CC	DOI	CCO	LOC
Fap2D	61	2	1	8	12
Fap2D.Gui	57	5631	10	599	38922
Fap2D.StructuralData	85	4117	4	128	5926
Fap2D.Computations	76	2539	5	114	6958
Fap2D.Storing	97	477	4	249	2210
Fap2D.Utilities	83	681	7	98	1757
Fap2D.ModelData	83	252	9	94	779
Fap2D.Design	82	34	1	1	61

Totalt antall kodelinjer er da 56625. Totalt antall linjer inkludert all kommentering og annet er 187896.

Det klart største prosjektet er fap2D.Gui. Her er den største klassen FrmMainGui delt inn i 15 delklasser for at det ikke skal bli for mye kode i en klasse. FrmModel er og en stor klasse, og denne er delt inn i 5 delklasser. Og FrmSettings er delt inn i 3 delklasser.

I klassen fap2D.StructuralData er klassen StructuralSettings delt inn i 2 delklasser, og klassen CrossSection er delt inn i 2 delklasser.

I klassen fap2D.Computations er klassen ComputationalModel delt inn i 3 delklasser, og klassen Frame2D er delt inn i 2 delklasser.

I klassen fap2D.Utilities er klassen GlobalSettings delt inn i 9 delklasser, og klassen Units er delt inn i 2 delklasser.

Ved å dele opp store klasser minkes kompleksiteten. Av tabellene er det enkelt å se at fap2D.Gui har fått hele 7454 kodelinjer lagt til våren 2013, mens MI har gått opp to hakk. Dette er på grunn av god oppbygging av koden, og at koden er delt inn i delklasser.

G. SVN

For å dele filer og holde versjonskontroll mellom vi som programmerte i **fap2D** dette semesteret brukte vi en SVN-server på NTNUs gruppeområde. Hver deling som ble gjennomført med serveren hadde en kommentar som beskrev arbeidet som var gjort, i tillegg til en oversikt over filene som hadde blitt endret. For å koble opp mot serveren logget vi inn via Putty og brukte TortoiseSVN for selve opp- og nedlasting av endringer.

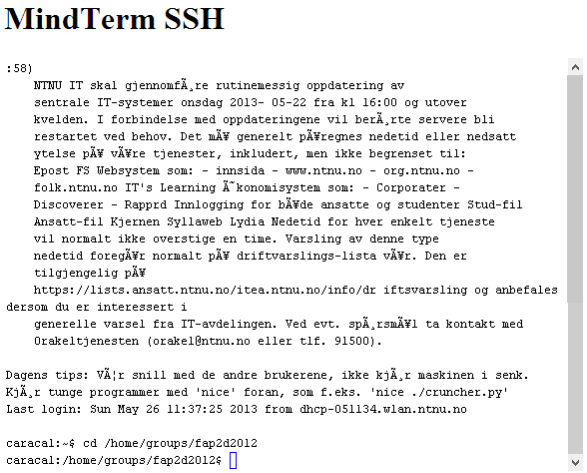
For fremtidige utviklere på **fap2D** som vil arbeide samtidig har vi laget en stegvis fremgangsmåte for å lage og administrere en slik server. Orakeltjenesten ved NTNU har vært svært hjelpsomme for vår del når vi skulle gjøre det samme, så hvis man står fast i instruksjonene anbefales det å spørre de.

Fremgangsmåte for oppsett og bruk av SVN-server på NTNUs gruppeområde:

1. Lag en gruppe på NTNUs gruppeområde http://www.stud.ntnu.no/kundesenter/groups/groups.php. Vi kalte vår gruppe fap2d2012.

Figur 67: Gruppeområdet etter at gruppen har blitt opprettet

2. Bruk kommandovinduet på http://login.stud.ntnu.no/, logg inn med vanlig studentpassord og bruk kommandoen «cd /home/groups/fap2d2012» til å navigere frem til gruppeområdet



Figur 68: Navigering til gruppeområdet på login.stud.ntnu.no

3. Bruk kommandoen «svnadmin create /rep» til å lage et *repository* på gruppeområdet kalt rep
4. Det vil også være nødvendig å åpne gruppeområdet for alle tilgangsrettigheter med kommandoen «chmod 777»
5. Serveren er nå opprettet. Likevel vil serveren kreve passordgodkjennelse for hver eneste fil som lastes opp, dette er tidkrevende og upraktisk. Derfor er det nødvendig å bruke Putty for å automatisk identifisere deg for hver forespørsel fra serveren.
6. Last ned og installer Putty-installasjonspakken fra http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. Bruk Windows installer-versjonen for å sikre at alle nødvendige komponenter i Putty-pakken kommuniserer skikkelig med hverandre.

Binaries

The latest release version (beta 0.62). This will generally be a version I think is reasonably likely to work well.

For Windows on Intel x86

PuTTY:	putty.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYtel:	puttytel.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSCP:	pscp.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSFTP:	psftp.exe	(or by FTP)	(RSA sig)	(DSA sig)
Plink:	plink.exe	(or by FTP)	(RSA sig)	(DSA sig)
Pageant:	pageant.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYgen:	puttygen.exe	(or by FTP)	(RSA sig)	(DSA sig)

A .ZIP file containing all the binaries (except PuTTYtel), and also the help files

Zip file:	putty.zip	(or by FTP)	(RSA sig)	(DSA sig)

A Windows installer for everything except PuTTYtel

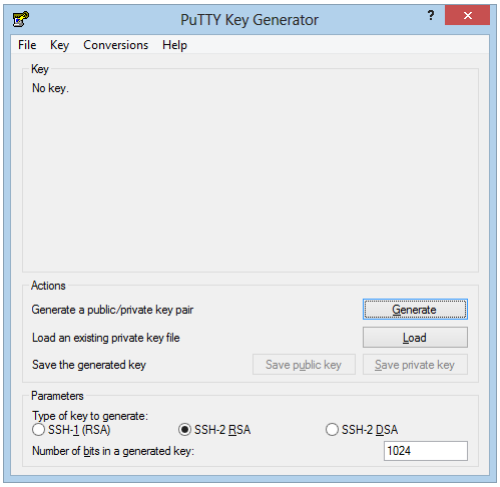
Installer:	putty-0.62-installer.exe	(or by FTP)	(RSA sig)	(DSA sig)

Checksums for all the above files

MD5:	md5sums	(or by FTP)	(RSA sig)	(DSA sig)
SHA-1:	sha1sums	(or by FTP)	(RSA sig)	(DSA sig)
SHA-256:	sha256sums	(or by FTP)	(RSA sig)	(DSA sig)
SHA-512:	sha512sums	(or by FTP)	(RSA sig)	(DSA sig)

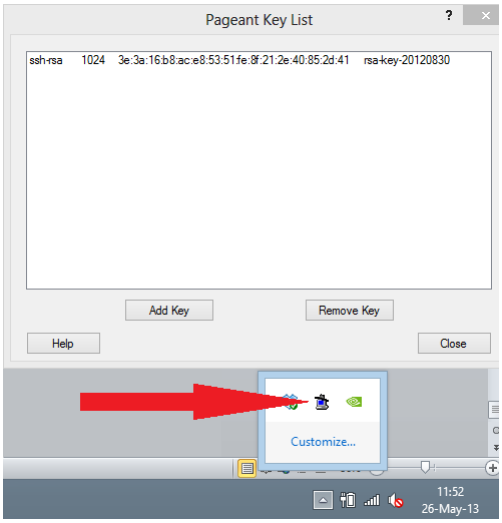
Figur 69: Hvilken Putty-pakke som skal velges på nettsiden for nedlasting

7. Bruk Puttygen fra Putty-pakken til å lage et sett med Private- og Public key. Det er ikke nødvendig å lage passord for nøklene.



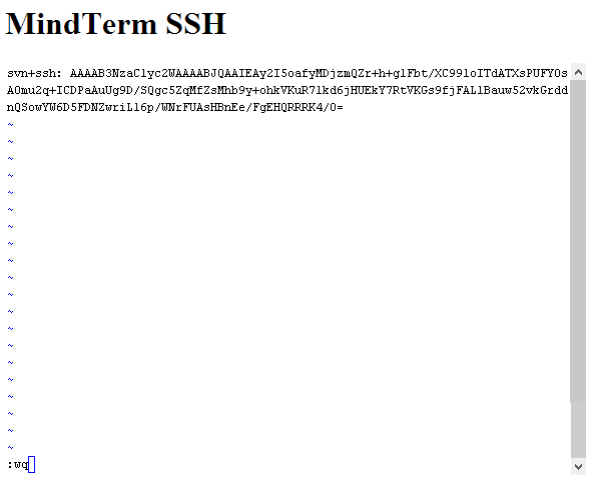
Figur 70: Puttygen som man bruker for å lage nøkler

8. Assosier filen Private-key med programmet Pageant fra Putty-pakken, ved å velge Pageant som standardprogram for Private-key-filen. Sett din datamaskin til å åpne Private-key ved oppstart slik at den alltid vil være aktiv i Pageant i bakgrunnen mens datamaskinen er på.
9. Pageant burde alltid være aktiv i bakgrunnen med Private key aktivert.



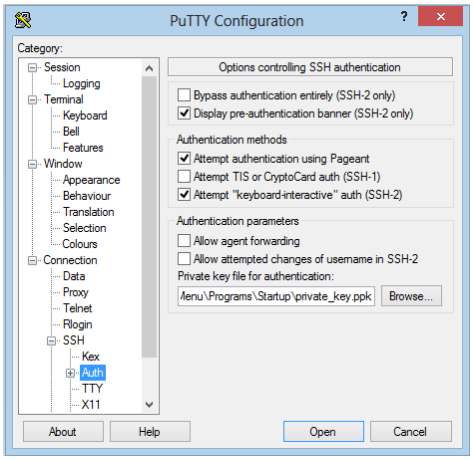
Figur 71: Pageant vinduet åpent med private key aktiv og pageant aktiv i bakgrunnen

10. Bruk kommandovinduet på http://login.stud.ntnu.no/ til å navigere til gruppeområdet med kommandoen «cd /home/groups/fap2d2012»
11. Bruk kommandoen «vim .ssh/authorized_keys» for å åpne tekstbehandleren vim og legge til innholdet Public key i filen .ssh/authorized_keys. I .ssh/authorized_keys skal det stå: «svn+ssh: (hele innholdet i Public key uten linjeskift eller mellomrom)».
12. Lagre og lukk .ssh/authorized_keys med vim-kommandoen «:wq».



Figur 72: Eksempel på public key og kommandoer i tekstbehandleren vim

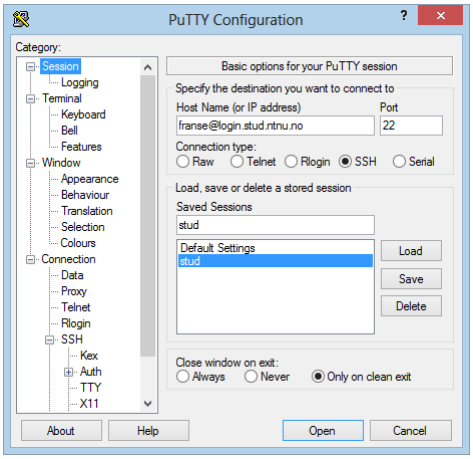
13. Åpne programmet Putty som ble installert sammen med Putty-pakken. Naviger deg frem til «Connection → SSH → Auth → Private key file for authentification» og kopier inn stien til Private key på din datamaskin.



Figur 73: Valg av Private Key i Putty

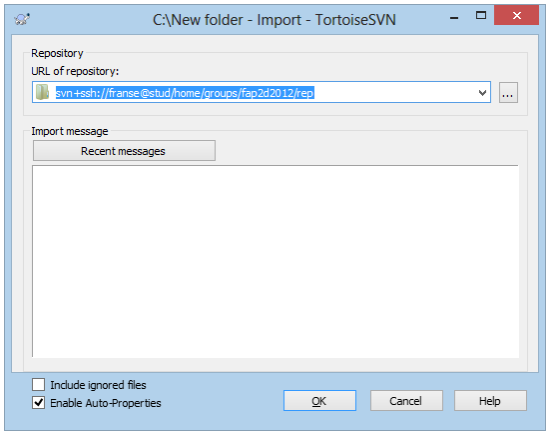
14. Deretter navigerer du deg frem til «Session» i Putty og setter følgende verdier:
    a. Host name = franse@login.stud.ntnu.no
    b. Port = 22
    c. Connection type = SSH

    Til slutt lagrer man som *session* med navnet «stud». Verdiene man nå har satt i Putty vil være inkludert i denne *session*.



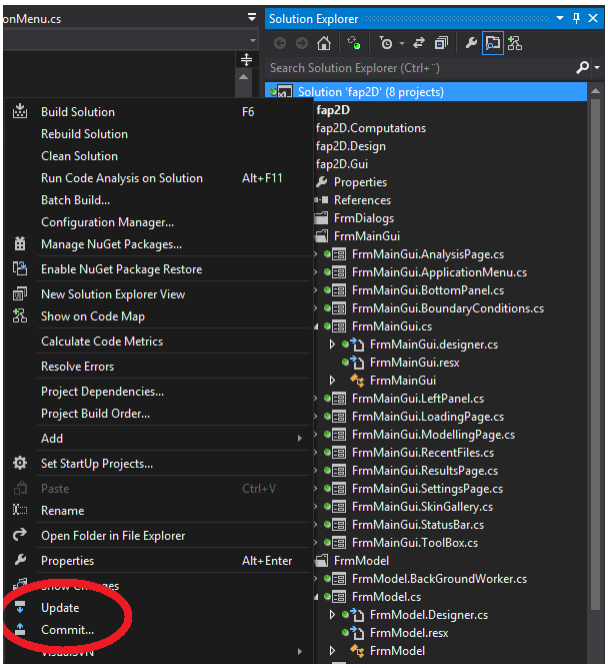
Figur 74: Valg av «session» innstillinger i Putty

15. Putty er da ferdig satt opp og serveren på grupperområdet vil da godkjenne datamaskinen du har privatnøkkelen på som din ID. Man slipper å gjenta passord for pålogging og for å laste opp og ned filer til gruppeområdet. Opplegget fungerer hvis man åpner *session* «stud» og får logget på uten å måtte skrive inn passord.
16. Deretter er det nødvendig å lage et område på din datamaskin hvor man kan lagre filer som deles med serveren og som kan være utviklingsområde for **fap2D**.
17. Last ned og installer TortoiseSVN http://tortoisesvn.net/downloads.html
18. Lag en tom mappe på datamaskinen der du vil ha utviklingsområdet. Etter installasjonen er Tortoise integrert i operativsystemet og man finner dermed Tortoise ved å høyreklikke på mappen, velge Tortoise og «import». Linken til repository er «svn+ssh://franse@stud/home/groups/fap2d2012/rep», der «stud» er navnet på den *session* som man lagret i Putty i steg 14.



Figur 75: Dialogboks for importering av repository

19. Innholdet på serveren kan da hentes ned i mappen med høyreklikk (Update) eller (Commit) for å laste opp. Disse kommandoene er også tilgjengelig fra å høyreklikke **fap2D**-prosjektet som er lastet inn i Visual Studio 2012. Visual Studio kan man få gratis hos Gurutjenesten ved NTNU.



Figur 76: Update og commit knappene i Visual Studio 2012

20. Ved å høyreklikke mappen for utviklingsområde på datamaskinen kan man velge hvilke filer som skal deles eller ikke med «TortiseSVN → add to ignore list». Man burde ikke dele filer som blir autogenerert av Visual Studio, da disse lager kluss med andre brukeres versjoner på serveren. Eksempler på slike er debug og release versjoner av programmet. Med dette er SVN-serveren og oppkoblingen ferdig satt opp.

H. Rapporterte feil i programmet

Liste over alle feil som er funnet av meg, hentet fra den en felles liste over rapporterte feil for alle utviklerne.

Tabell 17: Rapporterte og løste feil som ikke ellers er nevnt i denne masteroppgaven

Status	Dato funnet	Funnet av	Dato fikset	Fikset av	Beskrivelse
Løst	14.02.2013	Frans	15.03.2013	Kristian	Alle endrede egenskaper til en Structural Model lagres burde gi bruker en melding om lagring og et stjernemerke I tittelen for å gjøre det inuitivt for bruker at modellen burde lagres for at endringene skal ivaretas
Løst	02.02.2013	Frans	09.04.2013	Frans	Trd satt til å være lik Trf i ComputationalModel.Tables.cs linje 72, er dette rett? Det er misvisende med tanke på at de to tabellene har ulike funksjoner i følge dokumentasjonen. Dette er rett, det er bare forvirrende navnsetting.
Løst	14.02.2013	Frans	04.04.2013	Kristian	Getter/Setter IsModified I FrmModel.cs fungerer kun til å aksessere en Getter/Setter I StructuralModel.cs, dette er overflødig.
Løst	19.02.2013	Frans	15.05.2013	Kristian	Zoom to model mens eneste *member* er en krum *member* fører til at modellen ikke blir zoomet til. Modellen zoomer da heller kun til opplagerene.
Løst	11.03.2013	Frans	15.03.2013	Kristian	Det går an å bevege opplagere under «Results page». Det skal ikke være mulig.
Løst	18.04.2013	Frans	15.04.2013	Kristian. Den nye kan heller ikke skrive gresk	For å lage undertekst under resultatvisning OpenGL vinduet brukes glutStrokeString for å tegne teksten. Denne metoden klarer ikke å tegne greske bokstaver og burde byttes ut.

I. Tips og triks

Vi har hatt noen problemer i løpet av prosjekt- og masteroppgaven som vi gjerne vil at fremtidige utviklere skal unngå. Derfor har vi laget denne listen over tips og triks:

1. Bli enige om et forum hvor man sier ifra når man tar en commit til server. Vi har brukt facebook. Hvis man tar en commit skriver man «c» til de andre i gruppen. Dette har ført til at vi har unngått problemer når flere har jobbet med samme filer. En annen god regel er å ta en update fra server og teste om programmet fungerer som påtenkt før man tar en commit, dermed slipper man at alle andre får problemer. Forumet er også nyttig verktøy for å diskutere arbeidet som blir lastet opp.
2. Legg til databasen og obj- og bin mappene i *ignorelist*. Databasen er en oversikt over materialene og tverrsnittene som blir benyttet, og vi har flere ganger opplevd at modeller har sluttet å fungere når materialene eller tverrsnittene blir slettet grunnet at en ny database foreligger. Obj- og bin mappene består av kompilerte filer som er unike for hver datamaskin og er ikke nødvendig å laste opp. Disse er finnes i mappen til startprosjektet kalt **fap2D** og blir kompilert dit.
3. Når man commiter skriver man navnet sitt /implementasjonsnavnet og deretter hva commiten gjør i kommentarfeltet. Eksempel: Earthquake – Response spectrum analysis finished.

J. Testverdier

Tabellverdiene fra alle tester er samlet i dette vedlegget.

Tabell 18: Resultater fra nøyaktighetstesten for momentrespons med varierende elementinndeling uten skjærdeformasjoner

Antall elementer	Resultater [kNm]				Absolutt avvik
	«Sanntids» beregninger	Lineær statisk analyse fap2D	Lineær statisk analyse Focus	Analytisk resultat	
25	9.70	10.00	10.00	10.00	3.09%
50	9.90	10.00	10.00	10.00	1.01%
75	9.97	10.00	10.13	10.00	0.30%
100	9.90	10.00	10.00	10.00	1.01%
125	9.98	10.00	10.08	10.00	0.20%
150	9.97	10.00	10.00	10.00	0.30%

Tabell 19: Resultater fra nøyaktighetstesten for momentrespons med varierende elementinndeling og skjærdeformasjoner inkludert

Antall elementer	Resultater [kNm]				Absolutt avvik
	«Sanntids» beregninger	Lineær statisk analyse fap2D	Lineær statisk analyse Focus	Analytisk resultat	
25	9.70	10.00	10.00	10.00	3.09%
50	9.90	10.00	10.00	10.00	1.01%
75	9.97	10.00	10.13	10.00	0.30%
100	9.90	10.00	10.00	10.00	1.01%
125	9.98	10.00	10.08	10.00	0.20%
150	9.97	10.00	10.00	10.00	0.30%

Tabell 20: Resultater fra nøyaktighetstesten for momentrespons med varierende inndeling av lasttogposisjoner

Antall lasttogposisjoner	Lastens avstand fra venstre ende [mm]	Resultater [kNm]				Absolutt avvik
		«Sanntids» beregninger	Lineær statisk analyse fap2D	Lineær statisk analyse Focus	Analytisk resultat	
25	2,000.00	10.00	10.00	10.00	10.00	0.00%
50	1,959.18	9.80	10.00	10.00	10.00	2.04%
75	2,000.00	10.00	10.00	10.00	10.00	0.00%
100	1,979.80	9.90	10.00	10.00	10.00	1.01%
125	2,000.00	10.00	10.00	10.00	10.00	0.00%
150	1,986.58	9.93	10.00	10.00	10.00	0.70%
175	2,000.00	10.00	10.00	10.00	10.00	0.00%
200	1,989.95	9.95	10.00	10.00	10.00	0.50%

Tabell 21: Resultater fra testen for ekstreme responser

Responstype	Lastens avstand fra venstre ende [mm]	Resultater		Absolutt avvik
		«Sanntids» beregninger	Analytisk resultat	
Forskyvning	1,979.80	3.270 mm	3.273 mm	0.09%
Moment	1,979.80	9.90 kNm	10.00 kNm	1.01%