GRUPPE NR.
23

TILGJENGELIGHET
Åpen

OSLO METROPOLITAN UNIVERSITY
STORBYUNIVERSITETET

Telefon: 67 23 50 00
www.oslomet.no

Institutt for Bygg- og energiteknikk - Bygg
Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Pilestredet 35, Oslo

BACHELOROPPGAVE

BACHELOROPPGAVENS TITTEL	DATO
Effektiv modellering ved bruk av parametrisk modellering	26.05.2021
	ANTALL SIDER / ANTALL VEDLEGG 57/10
FORFATTERE Jan Tidjani Aboubacar Jakob Henstein Jonas Kristiansen	VEILEDER Lasse Arnesen

UTFØRT I SAMARBEID MED	KONTAKTPERSON
Ingeniørene Seim & Hultgreen A/S	Gaute Øye-Tveit

SAMMENDRAG

Det har skjedd en digital utvikling i den norske byggenæringen de siste årene og interessen for å effektivisere og optimalisere arbeidsprosesser er stor. Formålet med bacheloroppgaven har vært å studere hvordan parametrisk modellering kan effektivisere 3D-BIM-modellering. Oppgaven tar for seg hovedproblemstillingen «hvordan kan parametrisk modellering ved bruk av Dynamo effektivisere 3D-BIM-modellering?» og delproblemstillingen «hvor mye kjennskap har byggebransjen til parametriskmodellering og hva brukes det til?». For å belyse hovedproblemstillingen ble det gjennomført to dybdeintervjuer av fagpersoner og en praktisk utførelse med parametrisk modellering av 3D-armering i heisgruber. Gjennom å gjøre dette har vi kunnet sammenligne våre egne observasjoner med erfaringer fra fagpersonene. For delproblemstillingen ble det gjennomført en spørreundersøkelse med respondenter fra norske aktører i byggebransjen.

Resultatene viser at bruken av parametrisk modellering vil være effektiviserende for 3D-BIM-modellering, dersom det brukes riktig. Oppgavens omfang bør være så begrenset som mulig for å gjøre den parametriske modellen enkel å utarbeide og behandle i ettertid. I tillegg bør prosessen som automatiseres være repetitiv, slik at skriptet kan gjenbrukes. Vi ser samtidig at begrepet er kjent i byggebransjen idag, men at det er få som praktiserer metoden.

3 STIKKORD
Parametrisk Modellering
3D-armering
Dynamo

Forord

Denne bacheloroppgaven er utarbeidet ved OsloMet, våren 2021. Oppgaven er utført som en avsluttende oppgave for studiet, byggingeniør. Oppgavens tema omhandler bruken av parametrisk modelleringer og bruken av dette til å effektivisere armeringsprosessen. Dette er et høyst aktuelt tema i bransjen i dag, da digitalisering og effektivisering står sentralt. Målgruppen for denne oppgaven er ingeniører og studenter innenfor ingeniørfag -bygg som ønsker å lære mer om noen av de digitale verktøyene som er tilgjengelige innenfor modellering i dag, samt få et innblikk i hvordan disse kan anvendes.

Utarbeidelsen av denne oppgaven har vært en spennende prosess med en bratt læringskurve. Det har til dels vært krevende å tilegne seg de ferdighetene som har vært nødvendig for å kunne løse problemstillingen. Etter mye jobbing og mange sene nattetimer har vi laget et produkt vi er meget fornøyd med. Alle eventuelle feil er våre egne.

Oppgaven er skrevet i samarbeid med Ingeniørene Seim & Hultgreen, et rådgivende ingeniørfirma innenfor byggeteknikk. Vi vil rette en stor takk til de, og spesielt Gaute Øye-Tvedt for støtte og oppfølging under arbeidet.

Tusen takk til alle som har tatt dere tid til å delta i vår spørreundersøkelse. Det setter vi stor pris på. Vi vil også takke Marcin Luczkowski og Fredrik Jacobsen som tok seg tid til å stille til intervju og dele sin kunnskap og sine erfaringer innenfor temaet.

Til slutt vil vi rette en stor takk til vår interne veileder ved OsloMet, Lasse Arnesen, for god veiledning gjennom hele oppgaveskrivingen. Samt takk til medstudentene Atle Berczelly Schwach, Ola Brende og Øystein Reitan for testing og tilbakemelding på skriptet.

Oslo, 26. mai 2021

Jan Tidjani Aboubacar

Jonas Kristiansen

Jakob Henstein

Sammendrag

Formålet med bacheloroppgaven har vært å studere hvordan parametrisk modellering kan effektivisere 3D-BIM-modellering. På bakgrunn av dette tar oppgaven for seg hovedproblemstillingen «hvordan kan parametrisk modellering ved bruk av Dynamo effektivisere 3D-BIM-modellering?» og delproblemstillingen «hvor mye kjennskap har byggebransjen til parametrisk modellering og hva brukes det til?». Oppgaven tar utgangspunkt i kriteriene vi har satt for effektivisering: struktur i skriptet, brukervennlighet, generaliserbarhet og tidsbesparelse. For å belyse hovedproblemstillingen ble det gjennomført to dybdeintervjuer av fagpersoner og en praktisk utførelse med parametrisk modellering av 3D-armering. Ved å gjøre dette kan vi sammenligne våre egne observasjoner med erfaringer fra fagpersonene. For delproblemstillingen ble det gjennomført en spørreundersøkelse med respondenter fra norske aktører i byggebransjen.

Resultatet av den praktiske utførelsen og dybdeintervjuene viser at bruk av parametrisk modellering kan være effektiviserende ved riktig bruk. Vi ser at generaliserbarheten til skriptet er en avgjørende faktor for om bruken av metoden vil være effektiviserende, altså i hvilken grad det kan gjenbrukes. Metoden vil ha størst verdi om oppgaven som skal løses er repetitiv, som vil være tidsbesparende for modelleringsprosessen over tid. Metoden baserer seg på at ved å endre parametere kan en enkelt tilpasse modellen etter behov. God struktur i skriptet vil påvirke om det vil la seg gjøres eller ikke, siden skript i Dynamo kan bli uoversiktlig om en har mange variabler. Dette kan gjøre det vanskelig for andre å bruke og gjøre endringer i skriptet, samtidig som det tar lengre tid å lage det. Til forskjell fra parametrisk modellering er tradisjonell modellering mest benyttet til modellering i dag. Dette innebærer manuell plassering av objekter i modellen.

Resultatene fra spørreundersøkelsen viser at parametrisk modellering har et bredt anvendelsesområde. Likevel er det få som besitter tilstrekkelig kunnskap og erfaring med metoden i bransjen i dag. Ut ifra dette bør det legges fokus på brukervennlighet. Vårt tiltak er å bruke Dynamo Player i Revit. Det muliggjør at brukeren av skriptet trenger minimalt med forkunnskaper for å anvende parametrisk modellering, og innad i en bedrift kan dette være med på å danne et felles grunnlag for modellering av prosjekter.

Abstract

The purpose of this bachelor thesis has been to study how parametric modeling can make 3D-BIM modeling more efficient. Based on this, the thesis addresses the research question "how can parametric modeling using Dynamo make 3D-BIM modeling more efficient?" and the sub-question "how much knowledge does the construction industry have of parametric modeling and what is it used for?". The thesis bases upon the criteria we have set for efficiency: structure in the script, user-friendliness, generalizability, and time savings. To shed light on the research question, two qualitative research interviews were conducted with professionals and a practical execution with parametric modeling of 3D reinforcement. By doing this, we can compare our observations with experiences from the professionals. For the sub-problem, we surveyed respondents from the Norwegian construction industry.

The results of the practical execution and qualitative research interviews show that the use of parametric modeling can be more efficient with the appropriate application. We recognize that the generalizability of the script is a decisive factor in whether the application of the practice will be more efficient, that is to what extent it is reusable. The practice will have the highest value if the task is repetitive, this will save time for the modeling process over time. The practice base on the fact that by changing parameters, one can easily adjust the model as needed. Good structure in the script will affect whether or not the practice is applicable, since scripts in Dynamo can become confusing with many variables. This can make it difficult for others to use and make changes to the script, while also taking longer to create. Unlike parametric modeling, traditional modeling is commonly used for modeling today. This involves manually placing objects in the model.

The results from the survey show that parametric modeling has a wide range of applications. Nevertheless, few possess sufficient knowledge and experience with the practice in the industry today. Based on this, the focus should be on user-friendliness. Our initiative is to use Dynamo Player in Revit. This enables the user to have minimal prior knowledge to apply parametric modeling, within a company this may help form a general basis for modeling projects.

Innholdsfortegnelse

Lister

Begrepsliste

Algorithms-Aided Design Design ved bruk av algoritmisk hjelp

Arbeidsflyt Oversikten over nodene i Dynamo

Boolean En datatype med to mulige verdier - sant eller usant. *True* eller *False*

CC Senteravstand

Cnom Nominell overdekning

Finite Element Method En metode for å analysere og beregne spenninger og deformasjoner

Industry Foundation Classes Et format som brukes for utveksling av BIM-filer.

Klasse (programmering) En mal for hvordan objekter lages og hvilke egenskaper de skal besitte

Kompleks geometri Geometri av prosjektdeler som modelleringsprogrammer har vanskelighet for å lage

Metode (programmering) En sett med kode utfører operasjoner på et objekt i en klasse for å endre dets egenskaper

OK Overkant

Parameter (kurver) Beskriver en posisjon langs en kurve. Parameter 0 er startpunktet og parameter 1 sluttpunktet

Skript En sett med instruksjoner som forteller et program hva det skal gjøre

UK Underkant

Variabel En lagringsplass for en verdi. Verdien kan endres hvis brukeren ønsker det

Vertsobjekt Revitelementet et armeringsjern er knyttet til. Eksempelvis et dekke.

Figurliste

Tabelliste

1 Innledning

1.1 Bakgrunn

Det har skjedd en digital utvikling i den norske byggenæringen de siste årene og interessen for å effektivisere og optimalisere arbeidsprosesser er stor (Bartolomei, 2019). Drivkreftene bak denne digitale utviklingen er blant annet ønsker om effektivisering, kostnads-besparelse og en bærekraftig bransje. I lys av dette mener BuildingSMART Norges armerings-gruppe at Bygge-, Anleggs- og Eiendomsnæringen (BAE) trenger en ny tilrettelagt arbeidsflyt ved å legge til rette for å arbeide mer med digitale armeringsprosesser (Offergaard, 2021).

Bygningsinformasjonsmodellering (BIM), blir ofte nevnt i forbindelse med digitalisering og teknologien har vært i bransjen i lang tid. BIM-teknologien har vært populær i lang tid. Begrepet omfatter mye, men kan deles inn til to hovedkategorier, prosess (bygningsinformasjonsmodellering) og produkt (bygningsinformasjonsmodell). Bruk av BIM fører til at bransjen beveger seg mer og mer bort fra 2D-tegninger over til 3D-BIM-modeller, og det finnes mange metoder for å bruke BIM.

En metode innenfor behandling av 3D modeller er parametrisk modellering. Parametrisk modellering innebærer bruk av algoritmer, som ut ifra gitte betingelser og parametere utfører oppgaver i modellen. Hensikten er å effektivisere og automatisere design- og modelleringsprosesser, samtidig til at det muliggjør å modellere mer komplekse geometrier til konstruksjoner. Parametrisk modellering gjør det mulig å forene flere disipliner i én og samme datamodell. Det gjør at fagpersoner kan enklere samarbeide og forstå hverandre. Ved å gjøre endringer i parameterne kan en se effekten av endringene på konstruksjonen automatisk. Noe som vil virke gunstig tidlig i prosjekteringsfasen (Fremtidens Byggenæring, 2017). Ut ifra dette stiller den krav til kompetanse. I en heldigital verden blir datakunnskaper en viktig egenskap, selv i hverdagen til en byggingeniør. I nyere tid har blant annet Revit lagt til et tilleggsprogram kalt Dynamo. Parametrisk modellering kan være med på å redusere tidsbruken og byggingeniører kan slippe mange dager med repetitivt arbeid. Det er derfor interessant å utforske parametrisk modeller-ing nærmere. Temaet vårt blir dermed å se på bruk av parametrisk modellering med hovedfokus nytteverdiene metoden har sammenlignet med tradisjonell modellering, og få innblikk i bransjens forhold til dette.

1.2 Formål

Formålet med oppgaven er å se nærmere på hvordan parametrisk modellering kan effektivisere 3D-BIM-modellering.

Samarbeidspartneren vår, Ingeniørene Seim & Hultgreen, er et rådgivende ingeniørfirma innen byggeteknikk. Bedriften arbeider med å betjene byggherrer, arkitekter og entreprenører hvor målsettingen er å oppnå den beste løsningen med hensyn til konstruksjon, estetikk og økonomi. Prosjektområder er typisk boliger, kontor, industribygg, broer, pleiehjem, skole og offentlige konstruksjoner (Seim & Hultgreen, u.å).

Seim & Hultgreen har et ønske om å effektivisere arbeidet med plassering av armeringsjern i 3D-modeller ved bruk av Revit og Dynamo. De ønsker at løsningen skal være tidsbesparende og kunne brukes som en standardisert metode for armeringprosessen. I oppgaven bruker vi et av deres tidligere prosjekter. Målet er å lage et skript for plassering av armeringsjern i en heisgrube. Grunnlaget for at vi ser på en heisgrube, er at det er en gjentagende bygningsdel som skal modelleres. De lager armeringstegninger for omlag tjue heisgruber i året. Tiden det tar å lage disse tegningene, varierer fra én til to arbeidsdager, avhengig av øvrige arbeidsoppgaver. Selve modelleringen ved tradisjonell modellering tar omlag femten minutter, men de har et ønske om en løsning som kan videreutvikles til å inkludere mer. Eksempelvis uthenting av snittegninger og bøyelister.

Modellering av heisgruber med parametrisk modellering har til hovedfokus å plassere armeringsjern korrekt ut ifra armeringstegninger. Kollisjoner mellom armeringsjern vil ikke være en prioritering. Løsningen vil fremvises i form av skript og overføring av kunnskap, som de kan benytte seg av til framtidige prosjekter.

Gruppens mål er å få en dyperer forståelse av parametrisk modellering gjennom bruk av Dynamo, og opparbeide kompetanse som kan brukes i videre arbeid.

1.3 Problemstilling

Delproblemstillingen

Vi ønsker å se nærmere på hvordan parametrisk modellering brukes i bransjen i dag, og stiller følgende spørsmål:

«Hvilken kjennskap har bransjen til parametrisk modellering og hva brukes det til?»

Hovedproblemstillingen

På bakgrunn av Seim & Hultgreen sitt ønske, og vår interesse for å utforske parametrisk modellering, har vi valgt følgende problemstilling:

«Hvordan kan parametrisk modellering ved bruk av Dynamo effektivisere 3D-BIM-modellering?»

For å svare på hovedproblemstillingen har vi valgt å gå frem på to måter:

1. Praktisk utførelse i form av modellering av 3D-armering i heisgruber ved bruk av Dynamo
2. Dybdeintervjuer

1.4 Hypotese

Vi tror parametrisk modellering kan være med på å effektivisere 3D-BIM-modelleringsprosessen. Tradisjonell modellering kan være tidkrevende og tungvint. Seim & Hultgreen presenterte modellering av 3D-armering i heisgruber som en tidkrevende prosess, grunnet store mengder armeringsjern i denne konstruksjonsdelen. Vår hypotese går ut på at parametrisk modellering generelt, vil være en mer effektiv måte å modellere på. For å teste hypotesen vil vi ta for oss plasseringen av armering i heisgruber. Vi tror at et generaliserbart skript som kan gjenbrukes til fremtidige prosjekter, vil være tidsbesparende og effektiviserende. For delproblemstillingen er hypotesen vår at parametrisk modellering er et kjent begrep i bransjen, men blir lite brukt til reelle prosjekter. Vi tror at bruksområdene er automatisering av repetitive oppgaver, og utforming av kompleks geometri.

1.5 Avgrensning

Hovedproblemstilling

Temaet *effektivitet av modellering* er nokså omfattende og påvirkes av flere faktorer som spiller inn prosjekteringsfasen. I denne oppgaven valgt å fokusere kun på modelleringsdelen av armeringsprosessen. Det innebærer at vi ikke gjør beregninger, kontroller eller annen dimensjonering av kontruksjonsdeler. All nødvendig informasjon for å lage modellene og plasseringen av armeringsjernene, er gitt på forhånd i form av 2D-armeringstegninger, bøyelister og prosjektfiler i Revit fra Seim & Hultgreen. Vi har valgt å begrense oppgaven til konstruksjonsdelen heisgrube. Vi sammenligner effektivitet ved bruk av parametrisk modellering med metoden vår samarbeidspartner bruker i dag. Løsninger som faller utenfor disse kategoriene er ikke relevant for oppgaven.

Delproblemstilling

Vi har valgt å rette spørreundersøkelsen mot den norske bransjen, og da spesielt store aktører i Osloregionen samt vår samarbeidspartner. Disse er valgt fordi vi antar at de største aktørene er ledende innenfor bruk av nye teknologiske løsninger. Samtidig er vår oppfatning at disse aktørene arbeider med de største og mest komplekse prosjektene og dermed har mest å vinne på å ta i bruk denne typen metoder. Alle respondentene som har deltatt i undersøkelsen er ansatt i bedrifter med hovedkontor i Oslo. Vi søkte personer med kompetanse innen BIM og modellering, fra entreprenører til rådgivere. Hensikten er å se flere disipliner opp mot bruk av parametrisk modellering.

1.6 Struktur

I teorikapittelet presenteres teorien som ligger til grunn for fremgangsmåten og resultatet. Her beskriver vi de ulike programvarene vi har brukt og hvordan de fungerer. I tillegg legger vi frem og beskriver fundamentale prinsippene for fremgangsmåten i kapittel 4. I metodedelen presenterer vi og forklarer metodene vi har valgt for å løse problemstillingene. Denne delen tar for seg hvordan vi har gått frem og kriteriene vi har lagt til grunn for å kunne analysere resultatene. I kapittel 4, viser vi gjennomføringen og fremgangsmåten for selve løsningen av oppgaven. Dette blir så fremlagt i resultatdelen og videre diskutert. Avslutningsvis konkluderer vi og ser på mulige veier videre for oppgaven.

2 Teori

I dette kapittelet skal vi presentere teorien som ligger til grunn for løsingen av problemstillingene, og funnene i denne oppgaven. For å løse den praktiske delen av oppgaven har vi brukt flere forskjellige verktøy i form av modelleringsprogram, programutvidelser og kodespråk. Teorikapittelet bygger på skriftlige kilder og muntlige uttalser fra Marcin Luczkowski og Fredrik Jacobsen.

Navn	**Versjon**
Revit	2021
Dynamo	2.6
Python	3

Tabell 1: Oversikt over programvare og versjon

2.1 Visuell programmering og algoritmer

Visuell Programmering

For å forstå hva visuell programmering er, må en først forstå hva som ligger i begrepene koding og programmering. Hensikten med programmering er å få en datamaskin til å utføre en oppgave. Dette kan være for å effektivisere eller automatisere en prosess. Beskrivelsen datamaskinen får må være på et språk den forstår, nemlig kode. Ved å kode kan vi fortelle datamaskinen hvordan den skal utføre oppgavene vi ønsker. All koden en setter sammen utgjør til slutt et program eller et *skript*. På grunnlag av denne oppgavens karakter tar vi ikke for oss forskjellen på disse. Vi bruker begrepet skript.

Visuell programmering er en form for programmering eller *skripting*. Visuell programmering innebærer at en beskriver måten programmet skal utføre oppgaver på gjennom et visuelt grensesnitt. Dette i motsetning til tradisjonell, tekstbasert koding. I praksis fungerer den visuelle programmeringen ved at en kobler sammen elementer som inneholder ulike *funksjoner*, altså kode. Disse elementene kalles noder. Nodene kobles sammen av tråder og danner til slutt et skript eller en *Work Flow*, arbeidsflyt. Data beveger seg så fra node til node gjennom trådene og blir underveis behandlet av funksjonene på ulike måter. Gjennom hele arbeidsflyten har en mulighet til å mate inn data som *input* eller å hente ut data som *output*.

Visuell programmering kan beskrives som mer intuitivt enn tekstbasert koding. Ved å gjøre programmeringen visuell «brytes skriptet ned» slik at det kan være enklere å se hvordan programmet løser hver deloppgave/delproblem. Skriptene en setter sammen er algoritmiske, og består av flere algoritmer som sammen kan løse komplekse problemer.

Algoritmer

Som grunnlag for å forstå hvordan Dynamo og skriptene vi lager fungerer, er det nyttig å ha en forståelse av hva en algoritme er. Algoritme er i matematikk og databehandling en fullstendig og nøyaktig beskrivelse av fremgangsmåten for løsning av en beregningsoppgave eller annen oppgave (Grønmo, 2021).

En kan sammenligne en algoritme med en kakeoppskrift. Oppskriften er en steg-for-steg veiledning som beskriver fremgangsmåten for å oppnå et resultat, nemlig en kake. Innenfor programmering kan et skript være en sammensetning av mange algoritmer, der hver enkelt løser et delproblem. Selv om parameterene som mates inn i programmet endres, vil en algoritme være en presis beskrivelse av serien med arbeidsoppgaver datamaskinen skal utføre.

2.2 Python

Python er et tolket, objektorientert, høynivå programmeringsspråk med dynamisk semantikk (Python.org, 2021). At språket er tolket gjør at det kan brukes fritt uten behov for kompilatorer. Python er brukervennlig og relativt enkelt å lære sammenlignet med mange andre programmeringsspråk. Disse faktorene, i kombinasjon med kraften og at det enkelt kan brukes på tvers av plattformer, gjør Python til et populært kodespråk. For denne oppgaven er det ikke relevant å dykke dypere inn i hvordan Python er bygd opp. En beskrivelse av Pythons standard bibliotek produsert av Python Software Foundation (u.å) finnes på deres egen hjemmeside.

Dynamo tillatter brukeren å plassere egen pythonkode direkte inn i skriptet ved bruk av egne *Python-noder*. Disse beskrives nærmere i kapittel 2.5.

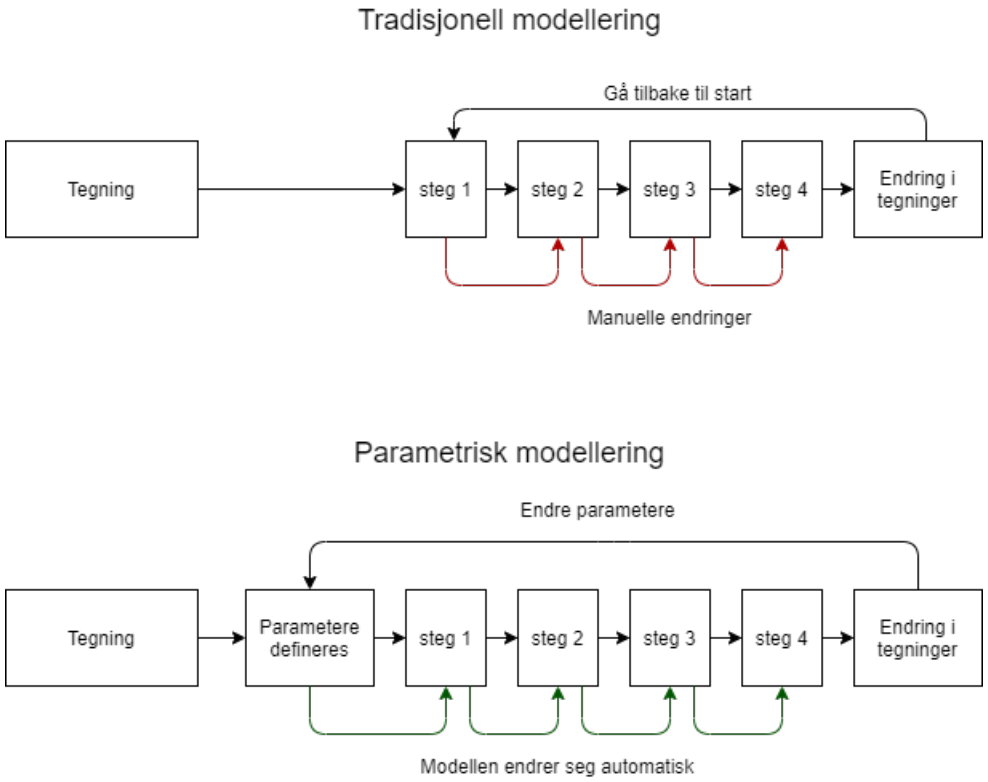
2.3 Parametrisk modellering

Parametrisk modellering er en måte å utarbeide/lage digitale modeller ved hjelp av forhåndde-finerte regler og algoritmer. Marcin Luczkowski beskriver det slik fra dybdeintervjuet:

> "*...Parametric modeling form is mostly about creating codes and algorithms which will help the design process, and if you as an user have the ability to change the parameters and automatically observe the change in your model, then you are doing parametric modeling*"

Kort fortalt handler det om å knytte elementer sammen slik at ved å endre parametere vil alle objekter som er knyttet til disse endre seg i takt (Fu, Feng, 2018). Ved å gjøre endringer på dimensjoner i en parametrisk modell vil alle objekter som er knyttet til disse verdiene også endre seg. Innenfor modellering av konstruksjoner kan det innebære at plasseringen til et armeringsjern automatisk endrer seg hvis tykkelsen til dekket det ligger i blir endret. Et viktig aspekt ved parametrisk modellering er at endringene kan visualiseres gjennom programvare. Hensikten med metoden er å redusere antall manuelle oppgaver og heller automatisere de.

Det er viktig å understreke at parametrisk modellering er en måte å arbeide på og ikke et program i seg selv, men for å bruke denne metoden trengs programvare som tillater det.



Figur 1: Tradisjonell modellering sammenlignet med parametrisk modellering

For å sette opp algoritmene og definere reglene til den parametriske modellen brukes *skript*. Designeren utarbeider/bruker et skript som inneholder funksjoner og velger selv hvilke parametere som skal legges inn. Typiske programmer som brukes til parametrisk modellering innefor bygg er Grasshopper og Dynamo. Bruk av parametrisk modellering tillater at en kan gjenbruke de samme skriptene flere ganger i løpet av samme prosjekt og mellom ulike prosjekter. Parametriske modeller kan skapes gjennom bruk av både visuell programmering og tekstbasert koding. I denne oppgaven bruker vi ren visuell programmering i kombinasjon med egne Pythonskript i noder i den visuelle programvaren.

Ut ifra resultatene i dybdeintervjuene (se kapittel 5.2), observerer vi at det er forskjeller på hvordan fagpersoner definerer begrepet. Uavhengig av dette, bygger metoden på prinsippene presensert over. Vi bruker i denne oppgaven begrepet parametrisk modellering for å beskrive metoden.

2.4 Revit

Revit® er et program fra Autodesk som lar brukeren designe og modellere i 3D. Programmet er et BIM-verktøy, som innebærer at en i tillegg til å lage visuelle modeller også har mulighet til å tilegne egenskaper til alle komponenter. Egenskapene som er knyttet opp mot modellen, kan være alt fra dimensjoner og kvaliteter til mengder og pris. Ingeniører og arkitekter, er

hovedsakelig programmets brukergruppe for design og modellering av bygg- og anleggsprosjekter. Videre kan modeller fra Revit eksporteres som *Industry Foundation Classes*-filer (IFC) og dermed brukes på tvers av ulike plattformer, både til prosjektering, bygging og forvaltning drift og vedlikehold (FDV). Komponentene, med deres egenskaper, kan eksporteres til andre programmer og utvidelser til Revit, som Dynamo (Autodesk Inc, u.å).

2.5 Dynamo

Dynamo generelt

"Dynamo is, quite literally, what you make it." (Modelab & Parallax Team, 2019)

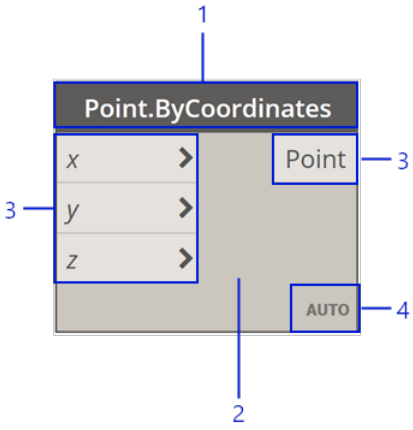
Dynamo er en visuell programmeringsplattform med åpen kildekode utviklet av Autodesk. Programvaren er en utvidelse til Revit, og kan gjennom de nyere utgavene åpnes direkte derfra. Dynamo kan også kjøres som en frittstående versjon, men da uten mulighet til å behandle data fra Revit direkte. Dynamo bygger på bruken av algoritmer og kan anvendes til en rekke operasjoner innefor alt fra databehandling til å generere geometrier (Modelab & Parallax Team, 2019). Programmet gir brukeren mulighet til å utføre operasjoner på data fra Revit, som normalt ville vært komplisert eller tilnærmet umulig å gjøre direkte i Revit. Uavhengig av dataene eller geometrien har opphav i Dynamo eller Revit, kan Dynamo behandle- og eksportere den til Revit.

Dynamo har en stor brukerskare og av den grunn finnes det mange forum der en kan finne informasjon for hjelp og veiledning til bruk av programvaren. Dette tar vi for oss nærmere i kapittel 6.3. I dette delkapitlet skal vi beskrive nærmere hvordan Dynamo fungerer.

Noder

Den visuelle programmeringen i Dynamo bygger på bruken av noder. For å forstå hvordan Dynamo fungerer i praksis, er det derfor grunnleggende å ha en forståelse av hvordan noder er satt sammen og hvordan de fungerer. I denne delen skal vi se nærmere på den generelle oppbyggingen av en node og de fem hoveddelene den består av. Oppbyggingen er uavhengig av hvilken oppgave noden har, altså hvilken type operasjon den skal utføre. Inndelingen av hva de fem delene nodene består av, er hentet fra Dynamo Primer (Autodesk & Modelab, u.å).

1.Navn Navnet gis ved "Kategori.navnet" som er beskrivende for arbeidsoppgaven. Alle noder som ligger i Dynamos bibliotek er kategorisert. Den første delen av navnet er kategorien, den sier ikke eksakt hva noden gjør, men forteller som regel hvilken type data som går inn eller ut. Kategoriseringen gjør det enklere å finne akkurat den noden en til enhver tid trenger for å bygge skriptet, samt at det gjør det enklere å navigere gjennom større skript. Den andre delen av navnet gir oss en beskrivelse av hva noden gjør eller hvordan den gjør det. Kombinasjonen av navn og kategori gir ofte god indikator på hvilken måte noden behandler data.

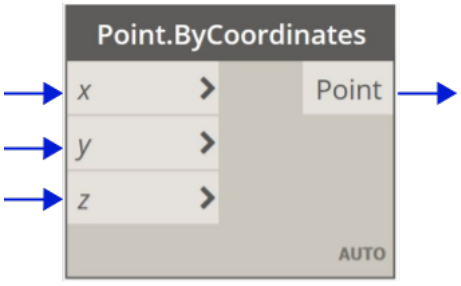


Figur 2: Oversikt over node

Etter hvert som en blir bedre kjent med dynamo er spesielt kategorien på nodene et nyttig verktøy. Hvis en vet hvilken type data som skal behandles i noden kan en ved å søke på denne, i søkefeltet, ofte finne en node som gjør operasjoner på denne type data.

2. Kropp Dette er selve kroppen til noden. Ved å klikke på denne får en tilgang til valgene som kan utføres i noden. Det omfatter alt fra forhåndsvisning, frysing, snøring og gruppering av noder.

3. Porter Elementene noden bruker til å ta imot og utleverer data kalles porter. Disse er mottakere for trådene som kobler nodene sammen. De fleste noder har en eller flere porter både for input og output, men det finnes også noder som kun har input- eller outputport(er). Måten noden arbeider på er at data kommer inn gjennom input porten(e) til venstre, noden utfører en operasjon på dataene, før den sendes ut gjennom output porten(e) til høyre. Vi sier derfor at dataflyten er mot høyre. For at noden skal klare å behandle data som vi ønsker, er den avhengig av å få inn riktig datatype. Det vil si at hvis en node for eksempel forventer å få input i form av heltall, men får en streng vil den ikke klarer å utføre operasjonen. Hvis en da forsøker å kjøre skriptet vil noden gi en feilmelding og ikke gi output, som fører til at dataflyten stanser.



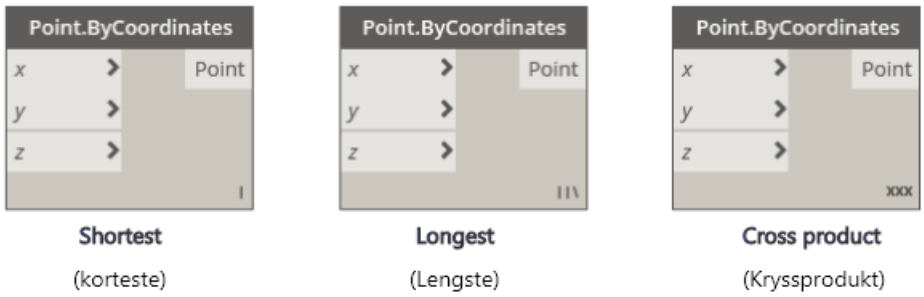
Figur 3: Porter node

4. Snøring Hvordan noden skal koble sammen dataene den får inn. Dersom noden mottar datasett med ulike lengder avgjør snøringen hvordan disse skal kobles sammen. Hvis en ek-

sempelvis har fire punkter, finnes det mer enn en måte å trekke linjer mellom disse. Snøringen forteller hvordan/i hvilken rekkefølge noden skal gjøre dette. Det finnes fire alternativer for snøring; Auto, Shortest, Longest og Cross-product. Auto innebærer at Dynamo selv velger hvilken snøring som brukes (se figur 3).

Korteste - Elementer med lik indeks kobles opp mot hverandre til slutten på den korteste listen.
Lengste - Elementer med lik indeks kobles opp mot hverandre til slutten på den korteste listen, deretter kobles det siste elementet i den korsteste listen opp mot de resterende i den lengste.
Kryssprodukt - Alle elementer kobles opp de alle elementer i den andre listen.

Dette visualiseres i vedlegg A.



Figur 4: Ulike typer snøring

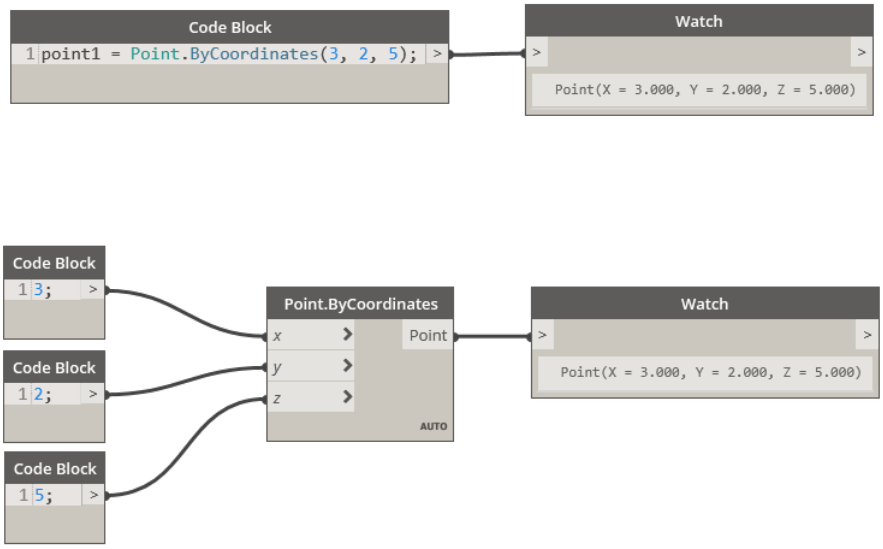
5. Standardverdi For mange noder ligger det allerede en standardverdi lagret. Det vil si at hvis noden ikke får en spesifikk input vil den benytte standardverdien for å utføre sin operasjon. Disse verdiene er som regel tilpasset slik at de påvirker skriptet i minst mulig grad. Eksempelvis så vil nøyaktighet settes som stor, vektorer og koordinater lik null og *booleans* (variabel som kan ha to verdier, sant/usant) lik sant (se figur 5).

Figur 5: Viser output med og uten definert input

Kodeblokker

Kodeblokker er en form for node som skiller seg fra gruppen beskrevet i avsnittet over. Kodeblokker tillater brukeren å skrive inn alt fra enkeltverdier og enkle utregninger til mer komplekse

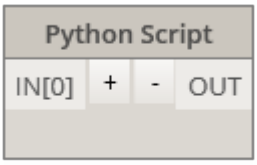
funksjoner. I kodeblokkene har en stor frihet til å selv utforme hvilke operasjoner den skal utføre, velge hvilke(n) input og output blokken skal gi. Bruken av disse kan være nyttig for å definere verdier som skal tas med videre inn i andre deler av skriptet eller for å regne ut formler. Kodeblokkene tillater også at en kaller direkte på *metoder* og funksjoner som er innebygd i Dynamo, i steden for å bruke egne noder. I figur 6 ser vi eksempelvis at metoden *ByCoordinate* kalles for å lage et punkt ut i fra gitte koordinater. Gjennom noden *Watch* ser vi hvilken output kodeblokkene gir. Ved å bruke kodeblokker kan en også unngå å bruke egne noder for å legge inn strenger, tall og formler i skriptet. Dette gjøres da heller rett i kodeblokken. I denne oppgaven bruker vi kodeblokker hovedsakelig til å definere tallverdier skal tas inn i andre noder eller kodeblokker og formler for utregning.



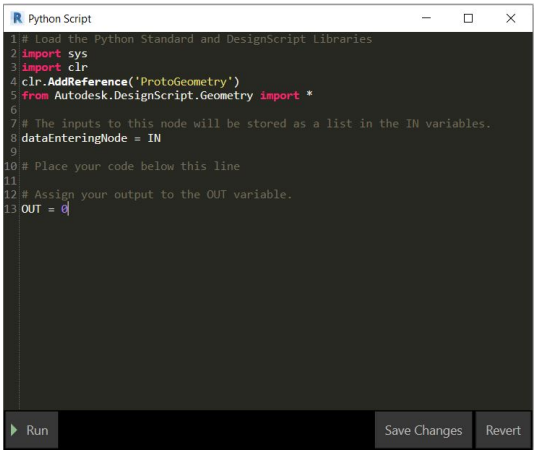
Figur 6: Kodeblokker

Python-noder

Python noder tillater brukeren å skripte Python direkte inn i Dynamo. Disse nodene gir muligheten til å utføre operasjoner som av ulike årsaker kan være komplisert eller tilnærmet umulig å utføre med "vanlige" noder. Antallet input i Python-noden kan endres, men det er kun mulig å ha en outputport. Som vi nevner i kapittel 2.5 kan dog store mengder data samles i én liste. Ved å gå inn i Pyton-noden (se figur 7b) har en mulighet til å skripte Python direkte. Skriptene her, kan en likhet med i kodeblokkene kalle på funksjoner og metoder som er innebygd i Dynamo. Pythonskriptene vi bruker i denne oppgaven er stort sett for å behandle lister. Enten for å sortere, flate ut eller sette de sammen på visse måter. Alle operasjonene kunne vært gjort ved bruk av Dynamos egne noder, men ved bruk av Python-noder har redusert antallet noder totalt i skriptene. For denne oppgaven er det ikke hensiktsmessig å gå dypere inn i Python-nodenes oppbygning.



(a) Python-node



(b) Innsiden av Python-noden

Figur 7: Pythonnode

Egendefinerte noder og pakker

I arbeidsflyten til Dynamo har en muligheten til å samle flere noder i en felles node, en egendefinert node. Når en lager egne skript kan dette være med på å gjøre de mer oversiktlige, samtidig som det gir muligheten til å enkelt kunne gjenbruke enkelte deler. En samling av egendefinerte noder kan settes sammen til en pakke. Pakkene en setter sammen kan lagres lokalt og/eller distribueres videre på nett. Det er mulig å laste ned en stor mengde pakker til Dynamo, enten direkte i Dynamo eller fra ulike nettsider. Pakkene kan være satt sammen og publisert av privatpersoner eller utviklere, og inneholder som regel noder innenfor visse kategorier, for eksempel armering eller rørgjennomføringer.

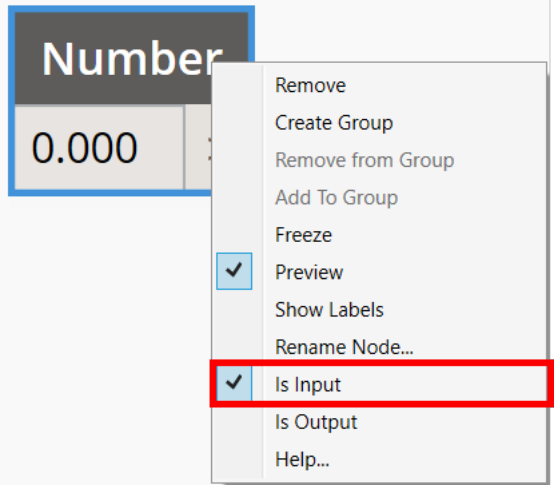
Datatyper

Data er et sett med verdier eller variabler. I programmering opererer en med mange forskjellige datatyper. Datatypen forteller programmet hvordan det skal behandle informasjonen. En kan se på datatyper som et lagringsformat som kan inneholde en type verdi. Dette fører til at dataene i Dynamo må behandles ulikt. Som tidligere nevnt, er noder avhengige av å få riktig datatype inn for å kunne behandle den på korrekt måte. Som et resultat av dette må en ofte konvertere mellom ulike datatyper for å kunne gjøre de operasjonene en ønsker på dataene. Fellesnevneren for alle datatypene og variablene de tilhører er at de kan lagres i lister. Lister er kort fortalt en samling av elementer. På denne måten kan en flytte og behandle store mengder data på en gang. En fordel med lister er også at en har mulighet til å hente ut elementer etter ønskede kriterier. Det kan for eksempel være datatype, navn, plassering.

Dynamo Player

Dynamo Player er i likhet med Dynamo en utvidelse til Revit som kan kjøres direkte derfra. Dynamo Player gjør at en kan kjøre Dynamoskript uten å måtte åpne fila i Dynamo og forlate Revits grensesnitt (se vedlegg B). Når Dynamo Player er åpnet kan brukeren selv velge hvilken

filplassering på datamaskinen programmet skal hente skript fra. Når ønsket skript er funnet kan brukeren fylle inn input og kjøre skriptet. Hvilken input som kan/må legges inn for å kjøre skriptet defineres i Dynamo. Dette gjøres ved å høyreklikke på noden og huke av på "Is input" (se figur 8). Dynamo Player kan forenkle bruken av skript.



Figur 8: Input i Dynamo player

3 Metode

3.1 Prosjektutforming og drøfting

Hovedproblemstillingen for oppgaven er «hvordan kan parametrisk modellering ved bruk av Dynamo effektivisere 3D-BIM-modellering?». For å svare på dette har vi to tilnærminger. En kvalitativ tilnærming i form av dybdeintervju og en praktisk utførelse. Dybdeintervjuene gir oss mulighet til å stille spørsmål direkte til kilden, og vi kan få et dypere innblikk i parametrisk modellering av fagpersoner med relevant kunnskap og erfaring. Den praktiske utførelsen gjøres gjennom bruk av parametrisk modellering til å modellere armering i heisgruber. Denne metoden vil være både kvalitativ og kvantitativ. Delproblemstillingen er «hvilken kjennskap har bransjen til parametrisk modellering og hva brukes det til?». Den har som formål å kartlegge bransjens forhold til parametrisk modellering opp mot kjennskap og bruksområder. Her vil vi benytte en kvantitativ metode gjennom spørreundersøkelse.

3.2 Valgt metode og utvalg

3.2.1 Delproblemstilling

For å belyse delproblemstillingen henter vi inn kvantitative data gjennom en spørreundersøkelse. Dette gjør at vi kan tallfeste dataene. Målgruppen for spørreundersøkelsen er norske aktører. Utvalget for undersøkelsen er respondenter fra aktørene; Sweco, Seim & Hultgreen, Multiconsult, Rambøll, Bane NOR, Skanska og AF-gruppen.

Datainnsamlingen er i form av et nettskjema. Nettskjema inneholder spørsmål om intervjuobjektenes kjennskap til- og bruk av parametrisk modellering. Forholdet vårt til datakilden blir dermed selektiv, som vil si at vi henter inn en bestemt type informasjon som er påvirket av oss (Halvorsen, Knut, 2008). Spørsmålene vil være konkrete, med svaralternativer slik at vi kan systematisere og tallfeste dataene. Analysen vil hovedsaklig legge vekt på sammenlikningsprosenter av antall respondenters svar på spørsmålene. I tillegg vil det være spørsmål der respondentene har mulighet til å legge inn tekstsvar for å tilføye eller utdype svarene.

3.2.2 Hovedproblemstilling

For å belyse hvordan parametrisk modellering ved bruk av Dynamo kan effektivisere 3D-BIM-modellering skal vi, som tidligere nevnt, gjennomføre en praktisk utførelse. Den praktiske utførelsen innebærer å modellere 3D-armering i heigruber. Referansen til modellen er armeringstegningene mottatt fra Seim & Hultgreen (se vedlegg C). Vi skal bruke parametrisk modellering gjennom Dynamo til å lage et fungerende skript. Tradisjonell metode for plassere 3D-armeringen, er direkte i Revit. Noe som kan være en repetitiv og tidkrevende prosess for heisgruber. Målet vårt er å lage en bærekraftig løsning i form av en parametrisk modell, som kan automatisere prosessen. Videre ønsker vi at skriptet vi lager skal være generelt, slik at det kan også kan brukes til fremtidige prosjekter. Parametrisk modellering er en modelleringsmetode

som ikke har blitt undervist i på universitetet. Derfor vil mye av arbeidet gå ut på prøve-feile metode, siden det ikke finnes en offentlig praktisk løsning på modellering av armeringsjern i heisgruber med parametrisk modellering.

For å sammenligne og utlede funnene til hovedproblemstillingen, vil vi i tillegg benytte oss av en kvalitativ tilnærming i form av dybdeintervju. Fagpersonene vi har intervjuet er valgt på grunnlag av deres erfaring innen modellering og skripting, og at de kjenner godt til parametrisk modellering. Målet for dybdeintervjuene er å gå mer inn på det faglige og bruken av parametrisk modellering. Tilnærmingen anser vi som et effektivt verktøy for å gi oss bedre en innsikt i metoden. Intervjuobjektene er Marcin Luczkowski og Fredrik Jacobsen. Luczkowski jobber som senior byggingeniør for Multiconsult og gjennomfører sin postdoktor på NTNU. Arbeidet hans omhandler ny teknologi og automatisering av bygningsdesign. Luczkowski har kompetanse innen FEM-modellering (Finite Element Method), BIM-modellering og parametrisk modellering. Modelleringsprogrammene han benytter hovedsakelig Rhino og Grasshopper, som likhet til Revit og Dynamo, kan benyttes til parametrisk modellering. Andre intervjuobjektet er Fredrik Jacobsen. Han jobber for Reope, et firma som består av ingeniører og arkitekter som arbeider med koding i Revit for å automatisere arbeidsprosesser. Utdannet som byggingeniør og master i anvendt matematikk og mekanikk, og har jobbet fem år i Rambøll og fem år i Johs Holt.

3.3 Reliabilitet, validitet og generaliserbarhet

Reliabilitet og validitet

Reliabilitet vil si hvor pålitelig kildene er. Målet med kvalitativ tilnærming ved dybdeintervju er å gå mer i dybden på temaet som en helhet. Intervjuobjektene er fagpersoner med erfaring og kunnskap innenfor området. Vi anser derfor dataene vi samler inn som pålitelige og valide. Reliabiliteten og validiteten for den praktiske utførelsen vil komme tydelig frem i ferdigstillingen. Vi vil få et konkret resultat som forteller om skiptet fungerer eller ikke. For å fremheve gyldigheten i resultatet skal vi teste skriptet på flere heisgruber. For å måle tidsbesparelse får vi medstudenter til å teste skriptet opp mot tidsbruk. Dataene vi samler inn her vil være kvantitative. Resultatene fra testene vil gi oss en god indikasjon på om det fungerer opp mot hensikt. Deretter skal vi vurdere om det kan brukes videre til andre prosjekter.

For å svare på delproblemstillingen vil de kvantitative dataene fra spørreundersøkelsens reliabilitet vises gjennom den faglige bakgrunnen til respondentene. Respondentene vi har oppsøkt er fagpersoner innen en rekke disipliner innen bransjen, men likevel fokusert på fagpersoner som har kjennskap til BIM og ulike modelleringsverktøy. Gyldigheten av resultatene avhenger av antallet respondenter som deltar i undersøkelsen. En utfordring kan være at feilmarginen blir for stor for til å kunne generalisere. Resultatene vil da heller brukes som en indikasjon ovenfor hypotesen vår.

Generaliserbarhet

Generalisering defineres som allmenngjøring utifra sammenlikning av enkelttilfeller (Olsvik, 2020). I denne oppgaven tar vi for oss generalisering innenfor to områder, generaliser-ing av resultatene fra spørreundersøkelsen og generaliserbarhet av skriptet. Knyttet opp mot spørreundersøkelsen omhandler dette hvorvidt svarene vi får kan ses på som representative for hele bransjen. Dersom utvalget ses på som tilstrekkelig, kan summen av alle enkelttilfellene (hver enkelt respondent) gi undersøkelsen høy nok validitet til at resultatet kan legges til grunn for å generalisere. Avgrensningene vi har valgt å forholde oss til må tas til betrakning når vi tar for oss i hvilken grad resultatene kan generaliseres.

Sett i sammenheng med skriptet knytter vi også generalisering opp mot summen av enkelttilfeller. Målet for hvor generelt skriptet vil være, er hvor mange enkelttilfeller det kan benyttes i. Vi anser det som *totalt* generelt hvis det kan brukes i alle tilfeller, altså alle oppgaver en står ovenfor der det er realistisk å bruke skriptet. Er skriptet *delvis* generelt kan hele eller deler av det benyttes i flere tilfeller, kun med små modifikasjoner. Som følge av dette vil et nøkkelord under generaliserbarheten til skriptet være *gjenbruk*. Hvorvidt resultatene kan generaliserer ut i fra disse kriteriene ser vi nærmere på i Resultat og Diskusjon

3.4 Betingelser for målinger av effektivitet

For å måle resultatet opp mot effektivitet, er det essensielt å definere hva vi legger i begrepet.

Overordnet ser vi på effektivitet som reduksjon av tid og energi. Altså jo mindre tid og energi en bruker på å utføre en oppgave, jo mer effektiv er prosessen. For å kunne måle dette har vi valgt å se på den totale effektiviteten som en sum av fire faktorer (se tabell 2). Denne inndelingen vil gjøre det enklere å analysere resultatene. Struktur og brukervennlighet måles kvalitativ gjennom tilbakemeldinger fra vår samarbeidspartner med utgangspunkt i det ferdige resultatet. Hvorvidt skriptet er generaliserbart kontrolleres gjennom testing i ulike modeller. Resultatet for tidsbesparelsen vil være en kombinasjon av to faktorer, testing gjennomført av medstudenter, Seim & Hultgreen og forfatterne, og tiden det har tatt å lage skriptet. I tillegg til dette vil resultatene fra dybdeintervjuene knyttes opp mot resultatene fra den praktiske utførelsen.

	Kriterie	**Beskrivelse**
1.	Struktur	Hvordan er skriptets oppsett
2.	Brukervennlighet	Hvor enkelt er det å bruke skriptet
3.	Generaliserbarhet	Hvor generelt er skriptet til gjenbruk
4.	Tidsbesparelse	Tidsbesparelse sammenlignet med tradisjonell modellering

Tabell 2: Kriterier for effektivitet

3.5 Metodekritikk og utfordringer

Delproblemstilling

Utfordringen til resultatet kan være størrelsen på utvalget. Hensikten med spørreundersøkelsen er å bruke de innhentede dataene til å danne et bilde av bransjens forhold til parametrisk modellering. Hvis utvalget er for lite vil det senke validiteten til undersøkelsen. Vi anser det derfor som nødvendig at spørreundersøkelsen ikke blir for omfattende, som kan resultere til at færre respondenter vil sette av nødvendig tid til å gjennomføre den. For å hindre dette må spørreundersøkelsen være så kortfattet som mulig, samtidig som det inneholder alle nødvendige spørsmål og beskrivelser.

For å sikre at reliabiliteten til undersøkelsen er så høy som mulig er det essensielt at spørsmålene i spørreundersøkelsen er entydige. I og med at undersøkelsen gjennomføres digitalt, har vi ingen måte å utdype eller beskrive nærmere hva som faktisk spørres etter, annet enn det står skrevet. En mulig feilkilde kan være at formuleringen i spørsmålene ikke er presist nok, slik at spørsmålene kan feiltolkes. Siden spørsmål med forhåndsgitte svaralternativer i liten grad tar hensyn til det, vil en løsning være å legge til en mulighet for tekstsvar. Tekstsvarene vil ikke kunne kvantifiseres på samme måte, men kan gi en indikator på reliabiliteten.

Hovedproblemstilling

Et viktig punkt ved gjennomføringen av den praktiske utførelsen er omfanget. Uten forkunnskap i bruk av Dynamo er det vanskelig å anslå nøyaktig hvor komplisert utarbeidelsen av skriptet vil være. Da vår samarbeidspartner heller ikke innehar et høyt kompetansenivå innenfor bruken av parametrisk programvare, stilles det krav til at vi må finne løsninger på problemene vi kan stå overfor. Som forberedelse til gjennomføringen har vi funnet noen utfordringer.

Ferdigstillingen av den praktiske utførelsen ved bruk av parametrisk modellering har vært utfordrende. Oppgaven har introdusert oss for bruk av Dynamo, som vi aldri har praktisert før, og ved inngangen til oppgaven hadde vi lite forkunnskaper om tema. Resultatet vi har oppnådd er ved prøving og feiling. Som tidligere nevnt finnes det ikke offentlige, praktiske eksempeler med parametrisk modellering av 3D-armering i heisgruber. Vi har derfor måttet ferdigstille skriptet ut ifra kunnskapen vi har opparbeidet. Likevel ble utførelsen gjennomført innen de gitte rammene. Vi vil diskutere nærmere gjennomføring og resultat av skriptet i delkapittel 6.2.

Vi ser fra start potensielle utfordringer knyttet opp mot resultatets reliabilitet fra løsningen av hovedproblemstillingen, som beskrevet i tabell 2. Dette handler først om fremst om de kvantitative dataene vi skal hente inn. For å kunne kvantifisere dataene er vi avhengige av å få presise tall som kan sammenlignes. I et tilfelle der de kvantitative dataene er for upresise, vil de kvalitative tilbakemeldingene ha en større påvirkning på resultatet. Vi må da ta høyde for at svarene en testperson gir i stor grad vil være subjektiv. For å få et resultat som i størst

mulig grad kan peke mot en entydig konklusjon, vil vi trenge et stort utvalg av testpersoner.

4 Gjennomføring

I dette kapitellet skal vi gjøre rede for hvordan vi kom frem til våre resultater. Vi viser til dybdeintervjuene, spørreundersøkelsen, og gjennomføringen av den praktiske utførelsen av parametrisk modellering for heisgrube.

4.1 Spørreundersøkelse

Spørsmålene vi skal ta for oss er bruk av parametrisk modellering opp mot erfaring og arbeidsoppgaver. Vi oppsøkte norske aktører og fagpersoner som vi mente har en relevans til spørreundersøkelsen. Ved å ta direkte kontakt over telefon, fremfor kun mail, viste seg å være mer effektivt, og vi fikk raskere svar og deltakerne som ble med svarte på undersøkelsen. Spørsmålene var standardiserte, og er lik for alle respondenter som deltok i undersøkelsen. Svarene de avga var lukkede, som vil si at spørsmålene hadde svaralternativer og ikke åpent for respondentenes mulighet til å skrive fritt, utenom svaralternativ «annet». Svar avgitt med «annet» måtte respondenten selv utlede svaret hengitt til spørsmålet. Resultatene fra undersøkelsen er kvantitative i form av prosentfordeling av antall respondenter og svaralternativene som ble gitt. Resultatet fra spørreundersøkelsen presenteres i delkapittel Spørreundersøkelsen, og deretter analyseres i Evaluering av skript

4.2 Dybdeintervju

Tilnærmingen vi valgte foregikk ved dybdeintervju av intervjuobjektene Fredrik Jacobsen og Marcin Luczkowski. Vi kontaktet intervjuobjektene over mail med henvisning til temaer vi var nysgjerrige på, og gjennomførte et digitalt møte med hvert intervjuobjekt separat. Datainnsamlingen foregikk i form av transkribering av dybdeintervjuene som ble gjennomført. Et dybdeintervju er en konversasjon hvor intervjueren oppmuntrer informanten (intervjuobjektet) til å bruke egne ord for å fortelle om erfaringer og kunnskaper relevant for problemstillingen (Halvorsen, Knut, 2008). For å kunne analysere intervjuobjektenes svar er det nødvendig å transkribere materialet, altså meninger og påstander rundt temaet. Vi bruker primærdata, som er data samlet inn av oss gjennom bruk av en eller flere datainnsamlingsmetoder (Halvorsen, Knut, 2008). Siden oppgaven vil belyse bruk av parametrisk modellerings, vil transkriberingen kun inneholde deres egne argumenter og påstander presentert i dybdeintervjuene. Transkriberingen setter søkelys på innholdet for å kunne analysere svarene som blir gitt, og vurdere deretter.

4.3 Grunnlag for skript

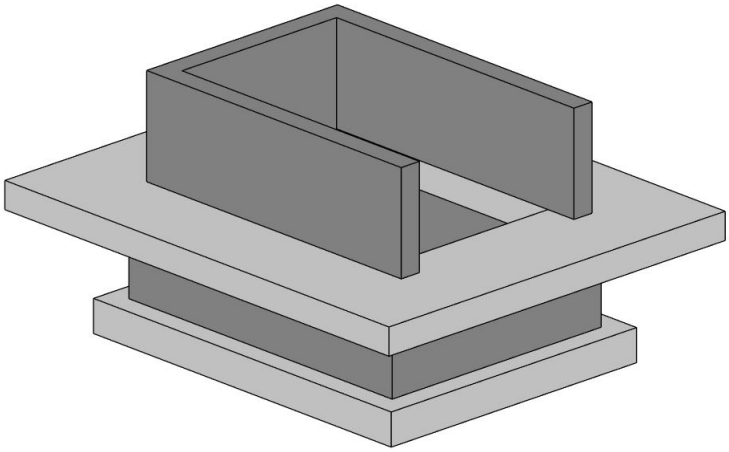
Oppdeling av skript

Vi har valgt å dele Dynamoskriptet opp i to deler, grubevegg og grubeplate. Årsaken til dette er todelt. For det første gjør oppdelingen skriptene mer oversiktlige. Dette gjør feilsøking og retting under utarbeidelsen av skriptet enklere. Et mer oversiktlig og forståelig skript gjør også

at det vil være enklere for personer som ikke har vært med å sette det sammen, å forstå hva de ulike delene gjør. Resultatet av et for stort skript kan fort være et "spaghettimonster". Med det mener vi en samling av noder og tråder som kan være svært utfordrende å navigere seg gjennom. For det andre gir en oppdeling av skriptet muligheten til at flere personer kan arbeide med hver sin separate del. Med hensyn til tidsrammen og omfanget av oppgaven har dette vært nødvendig.

Testing

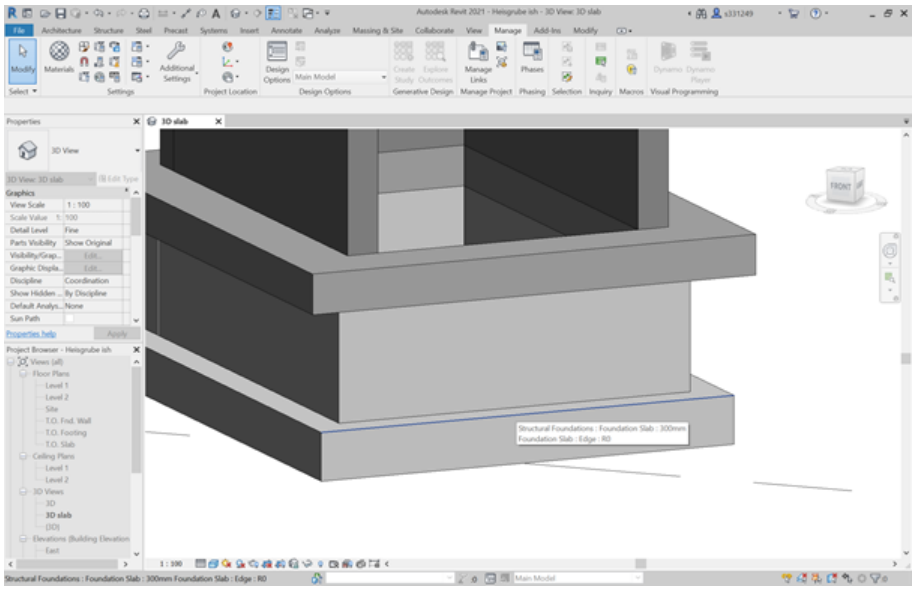
I arbeidet med oppgaven har vi valgt å lage en "testmodell" i Revit for å kjøre skriptene underveis. Modellen er laget med utgangspunkt i formtegningene. Dette gir oss muligheten til å kunne gjøre endringer i modellen underveis i arbeidet, slik at vi enkelt kan teste med ulike parametere. Alle figurer med visninger og snitt fra Revit i dette kapittelet er hentet fra testmodellen. I tillegg til i testmodellen har vi gjort regelmessige tester i prosjektfilene vi har mottatt fra Seim & Hultgreen.



Figur 9: Testmodell

Referanser fra Revit

I skriptene tar geometriene som lages i Dynamo utgangspunkt i en Revitmodell. Det betyr at referansene vi velger i Revit er kritiske for skriptet. Det finnes en rekke måter å hente ut referanser og verdier fra Revit til Dynamo på. Hvilken som er mest hensiktsmessig handler i stor grad om hvordan en ønsker at skriptet skal fungere. Referansen vi har valgt å bruke er linjen som utgjør den øverste kanten på kortsiden av grubeplaten, der utsparingen for heisdøren er (se figur 10). For å velge denne referanselinjen bruker vi noden *Select.Edge*. Ved å bruke denne noden selekteres referanselinjen manuelt i Revitmodellen. Som referanse kunne vi i teorien brukt hvilken som helst linje i Revitmodellen og heller bygd opp skriptet på en annen måte. Vi har valgt å bruke denne linja som referanse da den er enkelt gjenkjennelig i modellen.



Figur 10: Referanselinje for geometrien

Koordinatsystemer og vektorer

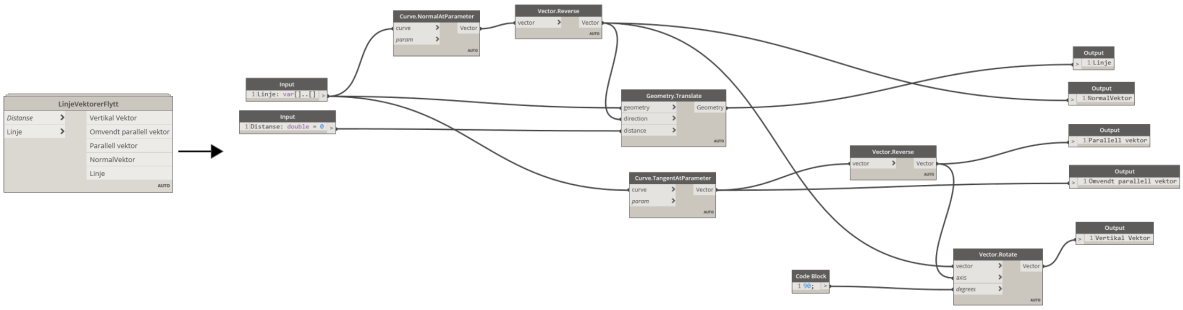
For at armeringsjernene vi lager i Dynamo skal ha riktig plassering når de importeres tilbake til Revit, er det essensielt at geometriene Dynamo behandler flyttes i riktige retninger, uansett orientering av *host element*, (vertsobjektet) altså Revitobjektet armeringsjernene knyttes til. I Revit ligger objektene knyttet til Revits koordinatsystem. Når en importerer objektene inn til Dynamo vil de orientere i samme retning i Dynamos koordinatsystem. Så lenge kurvene som definerer geometrien ligger langs x-, y- eller z-aksen, kan retningen på aksene brukes direkte for å forflytte geometri. Ligger kurvene derimot *ikke* langs aksene, må en bruke retningsvektorer for å til enhver tid gjøre forflytninger i riktig retning. I våre skript henter vi derfor ut og bruker retningsvektorer for alle forflytninger som skjer i xy-planet, siden denne orienteringen kan endre seg fra modell til modell. For forflytninger i z-retning bruker vi z-aksens retning som vektor, da vi antar at alle kurvene i geometrien til heisgruben ligger enten parallelt med- eller står vinkelrett på xy-planet.

Egendefinerte noder og pakker

Før vi ser på hvordan skriptene er satt sammen i sin helhet skal vi introdusere egendefinerte noder og pakker vi har bruker. De egendefinerte nodene er satt sammen av en eller to årsaker, enten at de gjenbrukes flere ganger eller for å gjøre skriptene mer oversiktlige. De egendefinerte nodene vi har laget behandler geometri, og brukes for å plassere linjer eller hente ut retningsvektorer. En nærmere beskrivelse av nodene vi bruker ligger i vedlegg D.

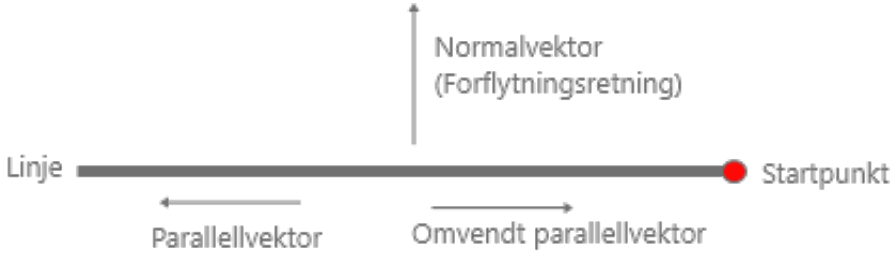
LinjeVektorerFlytt

Denne noden bruker vi for å hente ut vektorene til en kurve og flytte den i retningen av normalvektoren. Noden tar imot alle kurver for å hente ut vektorer og distanse for eventuell forflytning av curve.



Figur 11: Viser den egendefinerte noden LinjeVektorerFlytt

- Input
  - **Linje**. Linjen vi skal hente ut vektorene til og flytte
  - **Distanse**. Lengden linjen skal flyttes. Denne har standardverdi lik 0
- Output
  - **Linje**. Den nye linjen etter at den har blitt flyttet i retning av normalvektoren
  - **Normalvektor**. En vektor i xy-planet som peker vinkelrett ut av linjen som er input
  - **Parallellvektor**. En vektor i xy-planet som er en tangentvektor til linjen
  - **Omvendt parallell vektor**. En vektor i xy-planet som peker i motsatt retning av tangentvektoren
  - **Vertikalvektor**. En vektor i z-retning som peker vinkelrett ut av linjen



Figur 12: Visualisering av LinjerVektorerFlytt

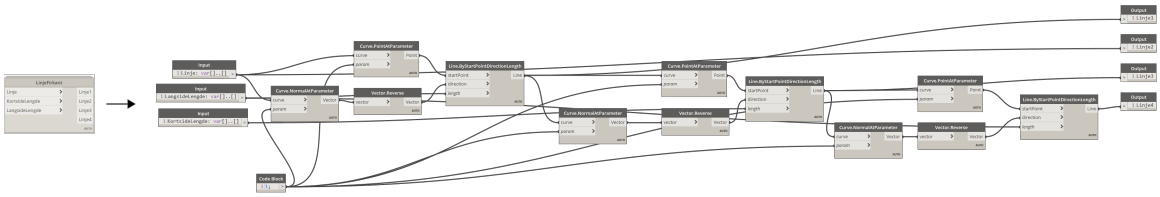
Curve.NormalAtParameter henter ut normalvektoren på parameter 0 til linjen som er input.

For å at vektoren skal peke i den retningen vi ønsker, snur vi denne med *Vector.Reverse* og får NormalVektor. Denne vektoren tar vi videre med inn i noden *Geometry.Translate* sammen med Linje og Distanse som er input. Linjen flyttes da distansen i retning av NormalVektor.

Curve.TangentAtParameter henter ut tangenten til Linje på parameter 0, Parallell vektor. Denne vektoren hentes deretter inn i node *Vector.Reverse* som gir Omvendt parallell vektor som output.

For å få den vertikale vektoren til linjen, bruker vi node *Vector.Rotate* og roterer NormalVektor 90 grader rundt Omvendt parallellvektor.

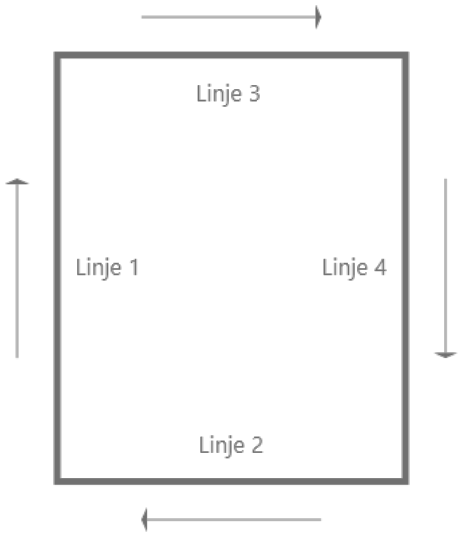
LinjeFirkant
Denne noden bruker vi for å trekke tre linjer, som sammen med referanselinjen utgjør et rektangel. Noden tar imot alle kurver og bruker endepunktet for å hente ut vektoren.



Figur 13: Vise den egendefinerte noden LinjeFirkant

- Input
  - **Linje**. Referanselinje. En av kortsidene i rektangelet
  - **KortsideLengde**. Lengden på de korteste sidene i rektangelet
  - **LangsideLengde**. Lengden på de lengste sidene i rektangelet
- Output
  - **Linje1**. Langside
  - **Linje2**. Kortside (samme linja som i input)
  - **Linje3**. Kortside
  - **Linje4**. Langside

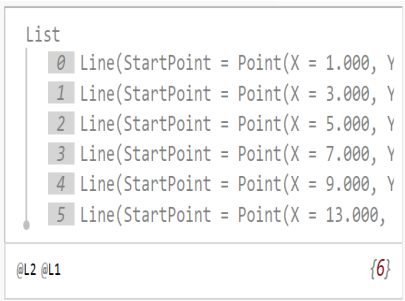
Referanselinjen fra input trekkes inn i noden Curve.NormalAtParameter. Siden vi ønsker å bruke endepunktet til linjen skriver vi inn tallet 1 i en kodeblokk og bruker den som input. Outputen fra noden er en normalvektor. For å få riktig retning på vektoren trekkes den videre inn i noden *Vector.Reverse*. I inn input til noden *Curve.PointAtParameter* trekker vi referanselinjen og verdien fra kodeblokken inn. Outputen er da endepunktet. Deretter trekkes endepunktet, normalvektoren og lengden på langsiden inn i noden *Line.ByStartPointDirectionLength*. Den trekker da en ny linje fra punktet, i retning vektoren med lengden LangsideLengde. Deretter gjennomføres det samme prosessen tre ganger slik at vi får fire linjer, inkludert startlinjen, som til sammen danner et lukket rektangel. Outputen fra den egendefinerte noden er alle linjene, inklusiv linjen fra input.



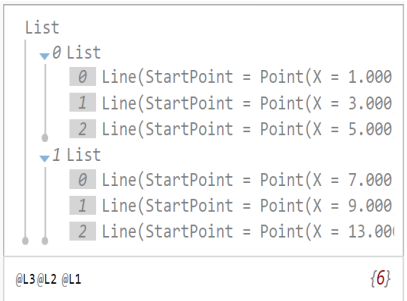
Figur 14: Visualisering av LinjeFirkant

Structural Design

For å sette samme skriptene importerer vi og bruker én pakke, *Structural Design* som er laget av Fudala (2018). Nodene i pakken brukes til å omforme geometriene vi lager i Dynamo til armeringsjern og fordele disse. Når armeringsjernene er skapt kan de eksporteres til Revit. En komplett oversikt over nodene vi har brukt fra pakken ligger i vedlegg E. Nodene i pakken som faktisk skaper armeringsjernelementer er *Create.FromCurve* og *Create.FromCurve*. Det eneste som skiller disse nodene er hvilken input de tar i mot i *Curve*-porten. Begge nodene tar imot lister med kurver i denne porten, men listen som mates inn *Create.FromCurve* kan kun inneholde en kurve per liste i listen, i motsetning til *Create.FromCurves* som kan inneholde flere kurver per liste. Det gjør at en kan bruke *Create.FromCurve* til rette armeringsjern (f.eks. lengdearmering), men for armeringsjern som kurver/har vinkler (eksempelvis bøyler) må *Create.FromCurves* brukes.



(a) flat liste med kurver



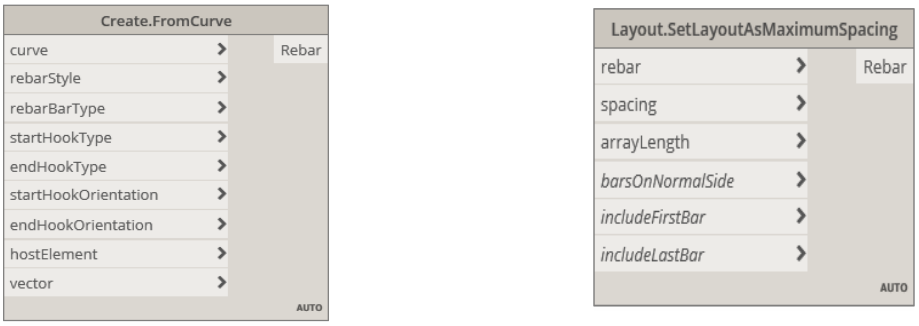
(b) tre kurver per liste i lista

Figur 15: Lister med kurver

Alle input portene som har navn som avslutter på "Style", "Type" eller "HookOrientation" tar imot input som definerer armeringsjernets egenskaper i Revit. Hvilke valgalternativer en har bestemmes ut ifra biblioteket i Revitmodellen som Dynamo er knyttet opp mot. Nodene som kobles til disse inputportene er også en del av pakken. Vertsobjektetet (*host.Element*) er Re-

vitelementet armeringsjernene skal knyttes til. Input til *vector*-porten er kurvens normalvektor.

Create.From.Curve(s) noden har ett element (armeringsjern) som output for hver liste med kurver den får som input. For å plassere disse over et gitt område bruker vi noden *Layout.SetLayoutAsMaximumSpacing*. Denne noden tar armeringselement, senteravstand og distansen armeringsjernene skal legges over som input. I tillegg kan en velge i hvilken retning armeringsjernene skal legges, definert av en vektor, samt om første og siste armeringsjern skal inkluderes.



(a) Opprettelse av armeringsjern (b) Fordeling av armeringsjern

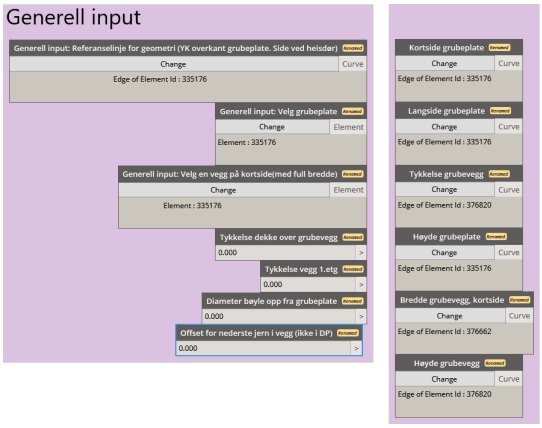
Figur 16: Noder fra Structural Design

Input

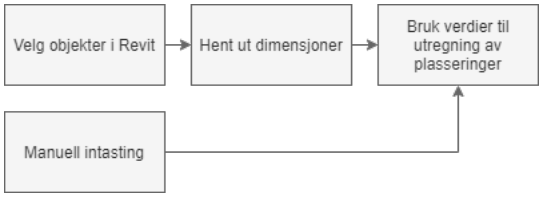
Generell input

For å systematisere skriptene mest mulig har vi valgt å sette opp en gruppe i starten av begge, der all den nødvendige generelle input mates inn. Som beskrevet i avsnitt "Referanser fra Revit" er referanselinjen vi henter ut fra Revit en essensiell del av inputen. Alle dimensjoner som trengs for å kjøre hver enkelt del av skriptene legges inn her. Dette gjøres på to måter, ved bruk av noder som krever at brukeren selekterer linjer eller elementer i Revit og gjennom manuell inntasting. Heisgrubens dimensjoner importeres gjennom *Select edge* noder (kolonnen til høyre i figur 17a). Disse importerer kantene på betongelementene i Revit til Dynamo som kurver. Lengden til kurvene hentes ved bruk av *Curve.Length*-noder. Vertsobjektet importeres gjennom noden *select model element* (den øverste noden i kolonnen til venstre). Den nominelle overdekningen hentes ut fra dette elementet ved bruk av noden *Properties.RebarHostCover*. De fire nederste nodene i kolonnen til venstre krever manuell inntasting av brukeren. Verdiene som hentes inn gjennom den generelle inputen videreføres deretter til hver enkelt det av skriptene. All inputen er definert som input i Dynamo Player som beskrevet i kapittel 2.5.

For å systematisere det hele og gjøre skriptet mer oversiktlig, plasseres alle verdiene vi skal ha med videre i separate kodeblokker. Alle videre deler av skriptene henter verdiene fra disse kodeblokkene.

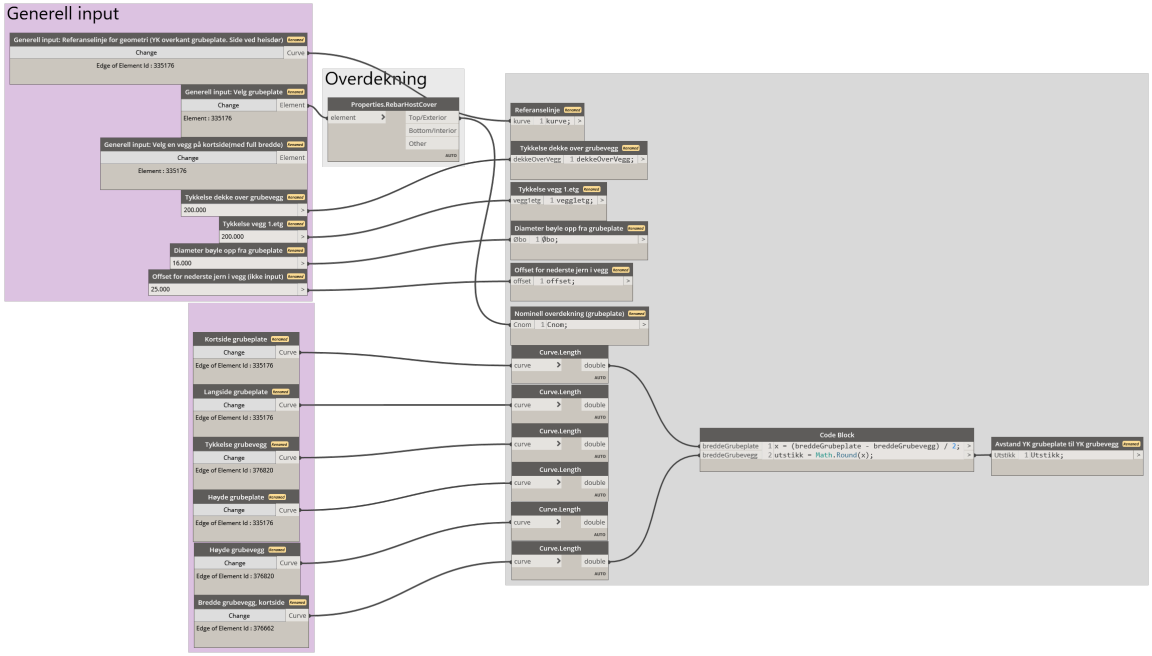


(a) Noder for input



(b) flyten til inputen

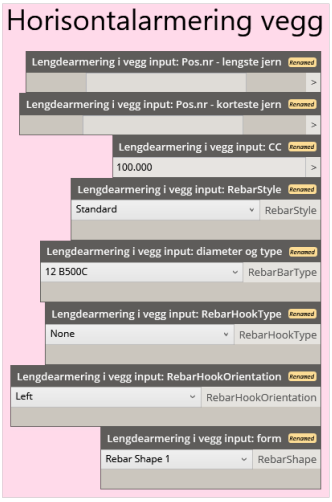
Figur 17: Generell input



Figur 18: Verdiene vi tar med videre

Spesiell input

I tillegg til den generelle inputen for hele skriptet er det en egen blokk med input for hver enkelt del av skriptet, altså for hver enkelt armeringsjernstype. Her må brukeren definere utformingen av armeringsjernet, posisjonsnummer og senteravstand. Armeringsjernets diameter brukes til mellomregninger, og den resterende verdien brukes i nodene som omformer geometri til Revitelementer og fordeler armeringsjernene utover. Alle verdier som definerer armeringjernenes utforming er nødvendig input i noden *Create.FromCurve(s)*, som vises i figur 16. Det er dog ikke alle som trenger å endres. Eksempelvis skal lengdearmeringen i veggen eller platen aldri ha krok. Derfor er det kun verdiene som kan variere de vi har definert som input i Dynamo Player.

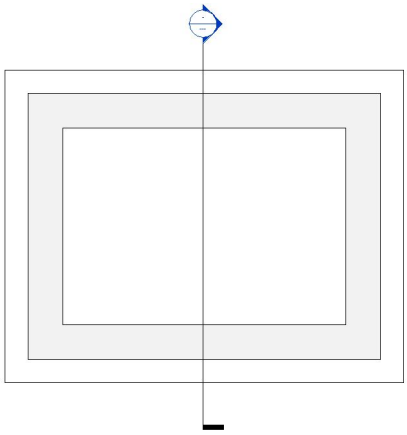


Figur 19: Eksempel på spesiell input

4.4 Introduksjon til gjennomgang

I kapittel 4.5 og 4.6 skal vi gå gjennom skriptene for all plasseringen av alle armeringsjernene. Fremgangsmåtene som beskrives er forenklet, og beskriver stegene som gjennomføres i Dynamo, men ikke hvilke noder som brukes i detalj. Skriptene i sin helhet finnes i vedlegg F og G. Innledningsvis i hvert delkapittel forklares hvilket armeringsjern som skal plasseres og dets posisjonsnummer (P.xx). Plasseringen av jernene gjøres med utgangspunkt i armeringstegningene mottatt fra Seim & Hultgreen. Fremgangsmåten og lengden armeringsjernene skal plasseres utover forklares gjennom tekst og visualisering i form av figurer. Avslutningsvis vises armeringsjernenes plasseringer i Revit i 3D-visning og snitt. Alle snitt er vertikalsnitt gjennom heisgrubens langsider (se figur 20).

Alle delene av skriptet bygger på samme prinsipp. Vi tegner først opp linjene som definerer overflaten til grubeplaten. Deretter brukes disse linjene for å finne punktene som definerer armeringsjernenes hjørner og/eller endepunkter.



Figur 20: Snittretning

4.5 Grubevegg

4.5.1 Vertikaler i vegghjørner

Formål

Dette skriptet skal plassere de vertikale armeringsjernene i alle vegghjørner (P.15). All armeringsjernene lages som separate elementer i Dynamo, og fordeles ikke utover en lengde. Stegene i fremgangsmåten gjøres for alle grubeplatens kantlinjer i OK.

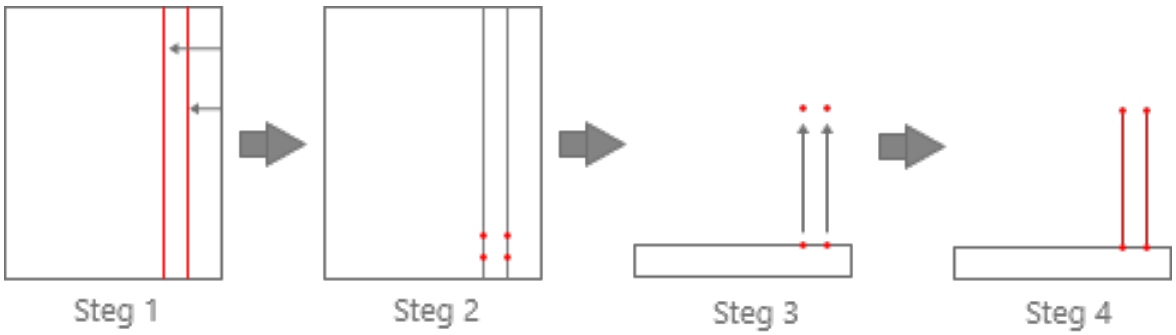
Fremgangsmåte

Steg 1: Bruker linjen som ligger langs kanten på grubeplaten. Kopierer og flytter denne slik at vi får to nye linjer. Disse ligger med riktige overdekninger i veggen og avstanden mellom dem er senteravstanden mellom de ferdige armeringsjernene.

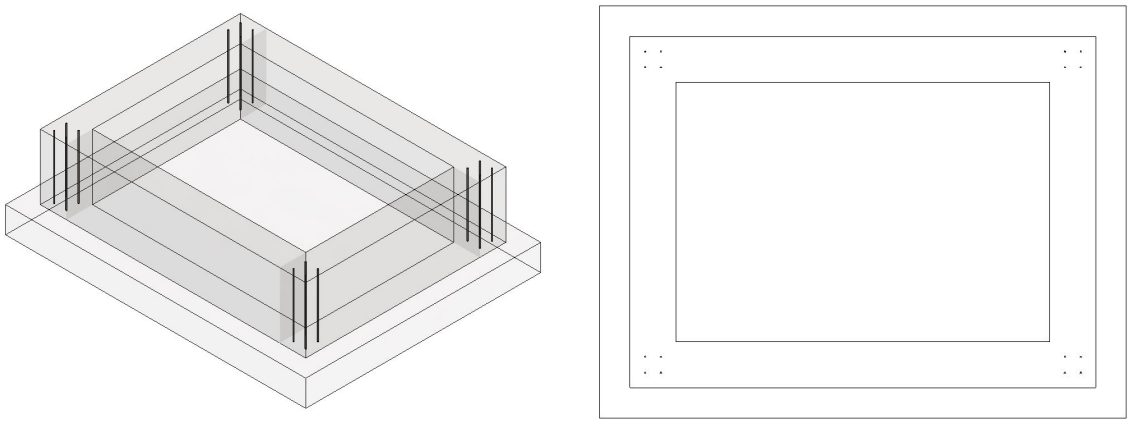
Steg 2: Vi beveger oss bortover linjene og plasserer to par med punkter. Hvert punkt representerer bunnpunktet til ett armeringsjern.

Steg 3: Alle punktene flyttes i positiv z-retning. De nye punktene representerer toppunktet til armeringsjernene.

Steg 4: Til slutt trekkes det linjer mellom bunn- og topppunktene. Hver linje er geometrien til ett armeringsjern. Deretter omformes geometrien til Revitelementer.



Figur 21: Stegene for de vertikale armeringsjernene



(a) 3D-visning

(b) Armering vist i plan

Figur 22: Vertikaler i vegghjørner i Revit

4.5.2 Horisontalarmering i vegger

Dette skriptet skal plassere de horisontale armeringsjernene i grubeveggen (P.12 og P.13). Armeringsjernene skal ligge symmetrisk i begge sider av veggen, plassert på innsiden av bøylene som kommer opp fra grubeplaten. Stegene i fremgangsmåten gjøres for alle grubeplatens kantlinjer i OK.

Fremgangsmåte

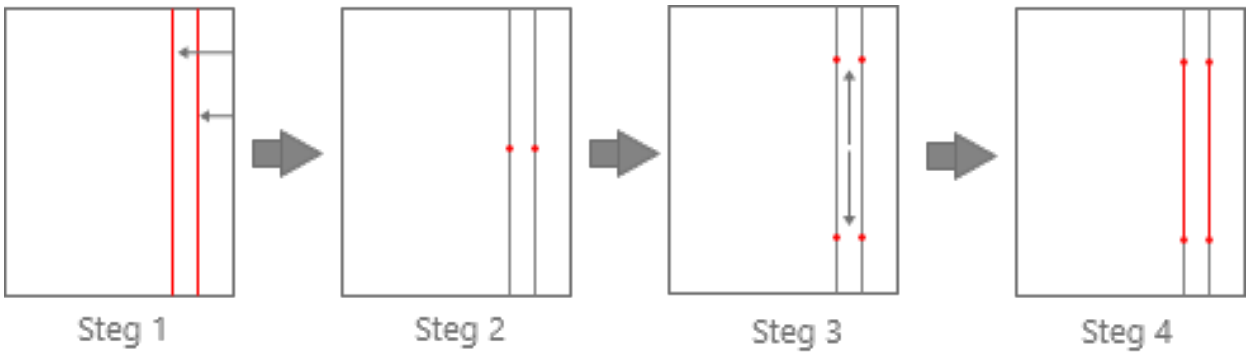
Steg 1: Bruker ut linjen som ligger langs kanten på grubeplaten. Kopierer og flytter denne slik at vi får to nye linjer. Disse ligger med riktige overdekninger i vegger og avstanden mellom dem er lik avstanden mellom armeringsjernene i hver sin ytterkant av veggen.

Steg 2: Finner midtpunktet til linjene.

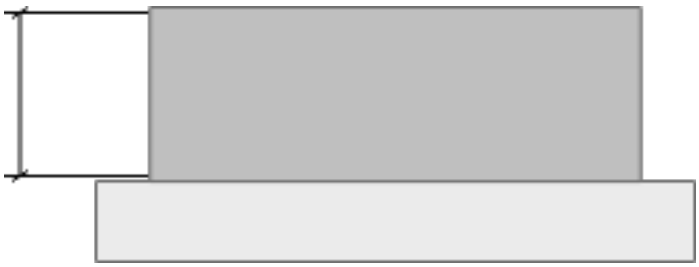
Steg 3: Fra midtpunkter følger vi linjen samme avstand i begge retninger og plasserer et punkt. Avstanden er lik halve lengde til armeringsjernene. De to punktene vi har plassert på hver linje, representerer start- og sluttpunktet til armeringsjernene.

Steg 4: Til slutt trekkes det en linje mellom start- og sluttpunktet. De to linjene er geometrien til armeringsjernene. Deretter omformes disse til Revitelementer og fordeles utover.

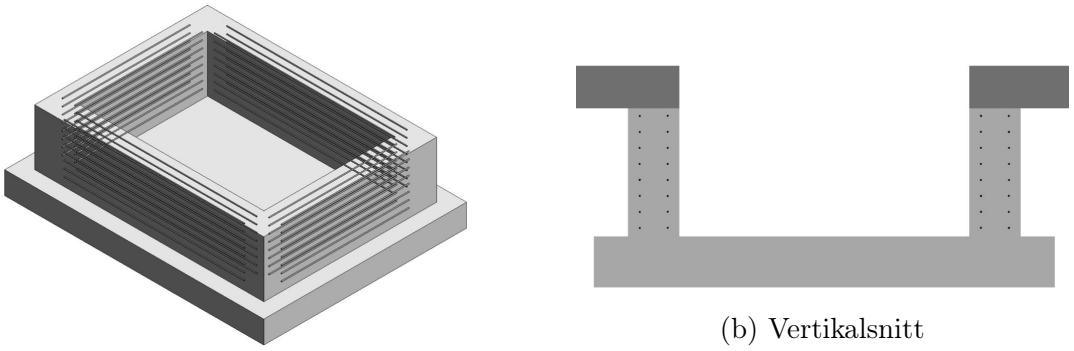
Fordelingslengde: For å hindre at armeringsjernene kommer i konflikt med grubeplaten og dekket i overkant av veggen legger vi inn en liten forskyvning i UK og OK. Fordelingslengden blir derfor lik vegghøyden fratrukket forskyvningen på begge sider. For å unngå kollisjon med bøylene i vegghjørnene forskyves bøylene.



Figur 23: Stegene for den horisontale lengdearmeringen i vegg



Figur 24: Fordelingslengden til de horisontale armeringsjernene i vegg



(a) 3D-visning

(b) Vertikalsnitt

Figur 25: Horisontalarmeringer i vegg i Revit

4.5.3 Horisontale bøyler i vegghjørner

Formål

Denne delen av skriptet skal plassere ut de horisontale bøylene i vegghjørnene (P.16). Stegene i fremgangsmåten gjøres for alle grubeplatens kantlinjer i OK.

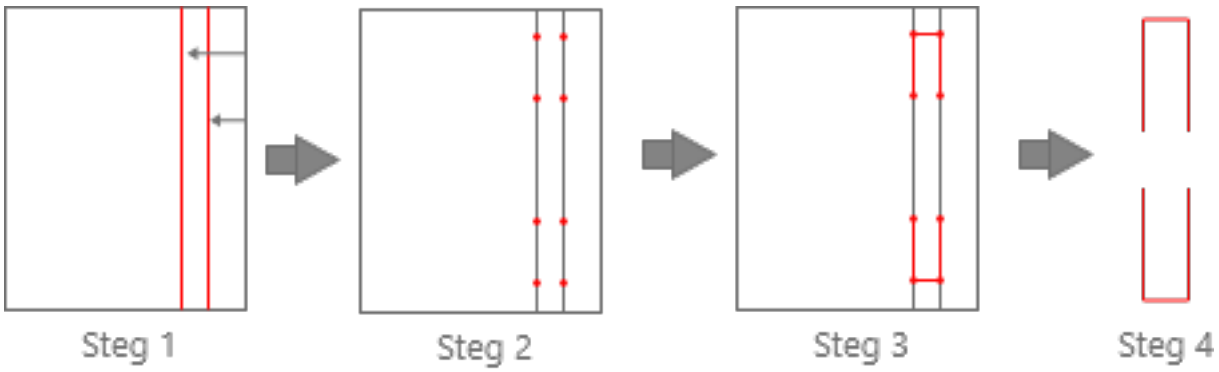
Fremgangsmåte

Steg 1: Bruker linjen som ligger langs kanten på grubeplaten. Kopierer og flytter denne slik at vi får to nye linjer. Disse ligger med riktige overdekninger og avstanden mellom dem er lik bøylens bredde.

Steg 2: Vi beveger oss bortover linjene og plasserer to par med punkter. Det ene paret representerer bøylens hjørner og det andre bøylens endepunkter.

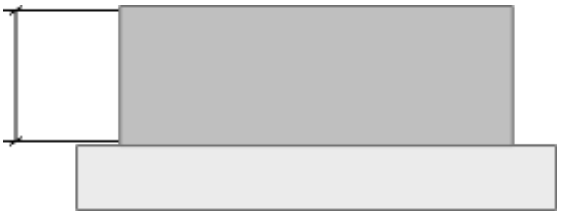
Steg 3: Trekker så en linje mellom punktene som ligger i det som skal være bøylens hjørner. Denne representerer den korteste linjen i bøylen. Vi trekker også en linje mellom startpunktet og endepunktet på begge linjene. Disse to linjene representerer bøylearmene.

Steg 4: Til slutt samles alle tre linjene i en liste. Listen sorteres slik at linjene kommer i riktig rekkefølge for at de kan omformes til Revitelementer. Bøylene i det ene hjørnet forskyves litt i positiv z-retning og bøylene i det andre i negativ z-retning. Dette er slik at bøylene ikke kolliderer, men ligger inntil den horisontale lengdearmeringen i veggen (P.12 og P.13).

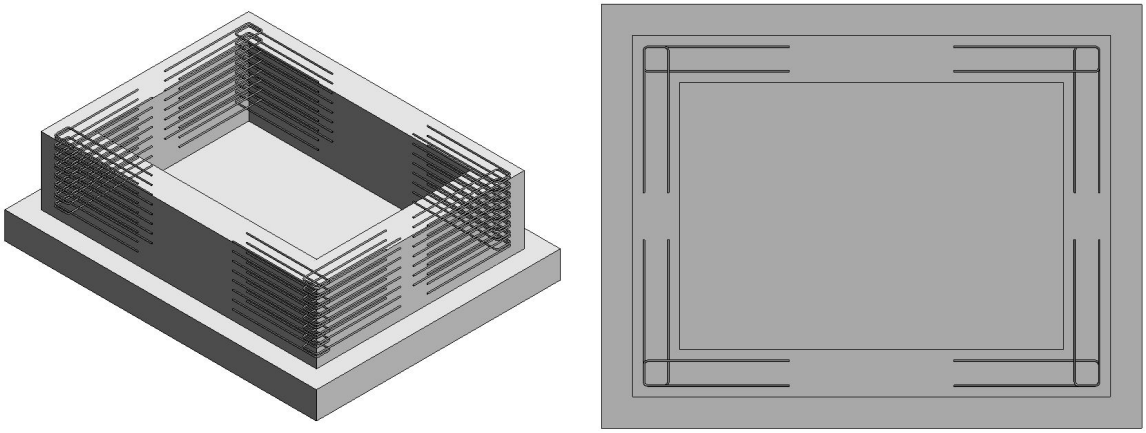


Figur 26: Stegene for de horisontale bøylene i vegghjørner

Fordelingslengde: For å hindre at bøylene kommer i konflikt med grubeplaten og dekket i overkant av veggen legger vi inn en liten forskyvning. Fordelingslengden blir derfor lik veghøyden fratrukket forskyvningen på begge sider. I tillegg tar vi hensyn armeringsjernets egen diameter.



Figur 27: Fordelingslengden til de horisontale bøylene i vegghjørner



(a) 3D-visning

(b) Plan

Figur 28: Horisontale bøyler i vegghjørner i Revit

4.5.4 Vertikale bøyler i vegg

Formål

Hensikten med denne delen av skriptet er å plassere ut bøylene i grubeveggen (P.18). Bøylene har startpunkt i grubeveggen og strekker seg opp i veggen i første etasje.

Fremgangsmåte

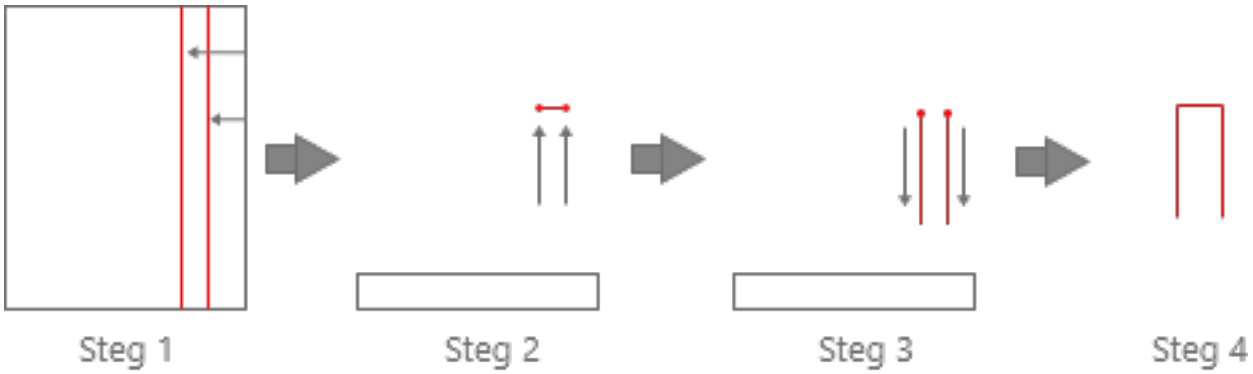
Stegene under viser fremgangsmåten for å finne geometrien til ett armeringsjern på en side. Alle operasjonene utføres for begge langsider og den ene kortsiden. Siden det skal være en utsparring for heisdør på den andre kortsiden, legges ikke armeringsjernene langs denne siden.

Steg 1: Henter ut linjen som ligger langs kanten på grubeplaten. Kopierer og flytter denne slik at vi får to nye linjer. Disse ligger med riktige overdekninger og avstanden mellom dem er lik bøylens bredde.

Steg 2: Flytter linjene i positiv z-retning slik at de skjærer gjennom punktene som skal være bøylens hjørner. Trekker så en linje mellom disse punktene. Denne representerer den korteste linjen i bøylen.

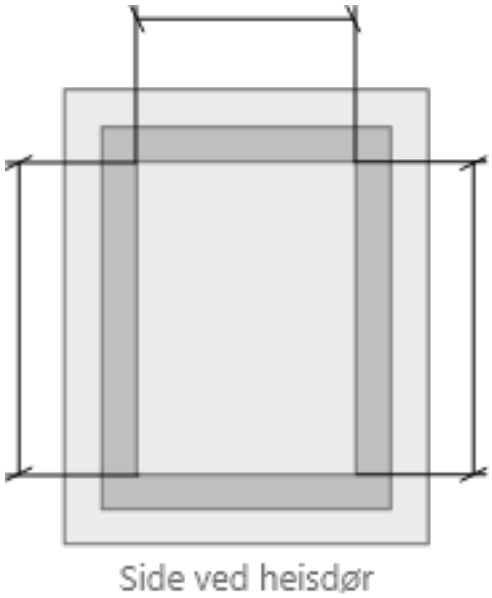
Steg 3: Med utgangspunkt i hjørnepunktene trekkes det en linje i negativ z-retning fra hvert av punktene. Disse representerer bøylearmene.

Steg 4: Linjene som utgjør bøylens geometri samles og sorteres, før de omformes til Revitelementer.



Figur 29: Stegene for de vertikale bøylene i vegg

Fordelingslengde: Siden det ligger vertikale armeringsjern i vegghjørnene avsluttes bøylene før hjørnene. Lengden blir derfor lik, lengden til hele siden fratrukket utstikket på begge sider og veggtykkelsen på begge sider. I tillegg må de forskyves slik at de ikke kolliderer med bøylene som kommer opp fra grubeplaten (P.14).



Figur 30: Fordelingslengden til de vertikale bøylene i vegg

4.6 Grubeplate

Vi har valgt å ikke legge inn V-bøylene i hjørnene av grubeplata (P.19). Etter å ha tatt en vurdering, ser vi at tiden det ville tatt sammenlignet med tiden vi har hatt til rådighet, har vi ikke sett på det som hensiktsmessig. Hadde tidsrammen til oppgaven vært større ville disse også blitt inkludert.

4.6.1 Vertikale Bøyler

Denne delen av skriptet plasserer de vertikale bøylene (P.14). Disse har bunnpunkt i grubeplaten og strekker seg opp i grubeveggen. De plasseres slik at de ligger på utsiden av alle horisontale armeringsjern i grubeveggen.

Fremgangsmåte

Steg 1: Henter ut linjen som ligger langs kanten på grubeplaten. Kopierer og flytter denne slik at vi får to nye linjer. Disse ligger med riktige overdekninger i veggen og avstanden mellom dem er lik bøylens bredde.

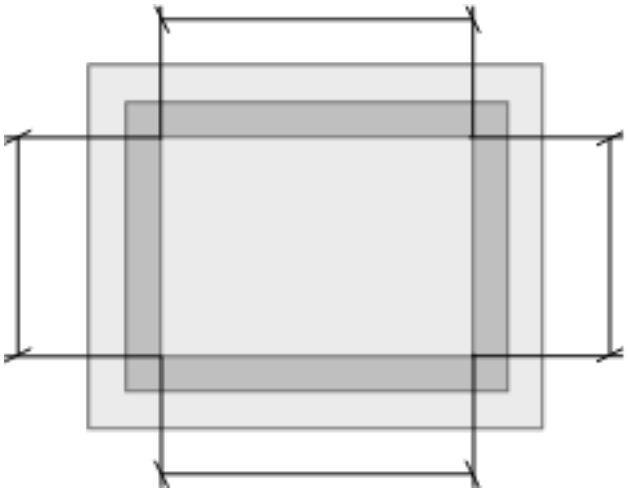
Steg 2: Flytter linjene i negativ z-retning slik at de skjærer gjennom punktene som skal være bøylens hjørner. Trekker så en linje mellom disse punktene. Denne representerer den korteste linjen i bøylen.

Steg 3: Med utgangspunkt i hjørnepunktene trekkes det en linje i positiv z-retning fra hvert av punktene. Disse representerer bøylearmene.

Steg 4: Deretter samles alle tre linjene og sorteres. Til slutt omformes geometrien til Revitelementer og fordeles utover.

Figur 31: Stegene for de vertikale bøylene i grubeplate

Fordelingslengde: Siden det ligger vertikale armeringsjern i vegghjørnene, avsluttes bøylene før hjørnene. Lengden blir derfor lik, lengden til hele siden fratrukket utstikket og veggtykkelsen på begge sider. I tillegg må de forskyves slik at de ikke kolliderer med de vertikale bøylene i grubeveggen (P.18).



Figur 32: Fordelingslengden til de vertikale bøylene i vegg

4.6.2 Kantbøyler

Formål

Denne delen av skriptet plasseres kantbøylene i grubeplaten (P.17). Bøylene skal ligge langs alle kanter i grubeplaten og ha omfar med lengdearmeringen i grubeplaten.

Fremgangsmåte

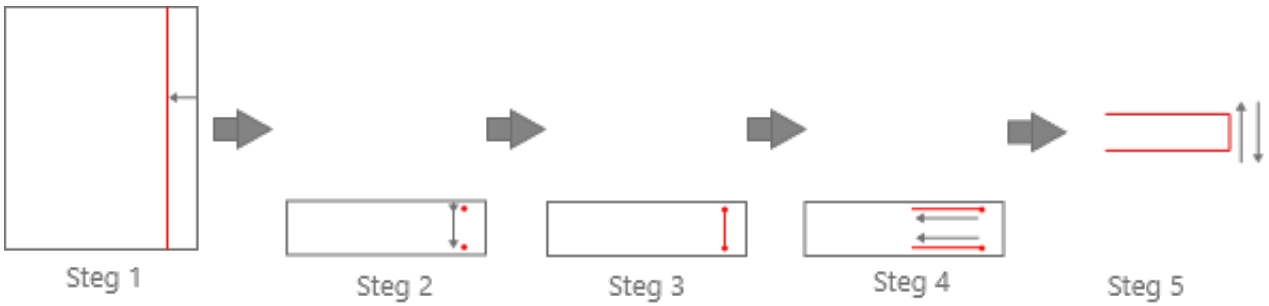
Steg 1: Henter ut linjen som ligger langs kanten på grubeplaten. Kopierer og flytter denne slik at vi får en ny linje som ligger med en avstand fra kanten som er lik overdekningen til bøylen.

Steg 2: Kopierer og flytter linjen i negativ z-retningen slik at vi får to nye linjer. Disser ligger med riktig overdekning til OK og UK grubeplate og avstanden mellom de er lik bøylebredden.

Steg 3: Vi beveger oss bortover linjene og plasserer ett punkt på hver linje. Disse punktene representerer bøylens hjørner. Vi trekker så en linje mellompunktene. Linjen representerer bøylens korteste side.

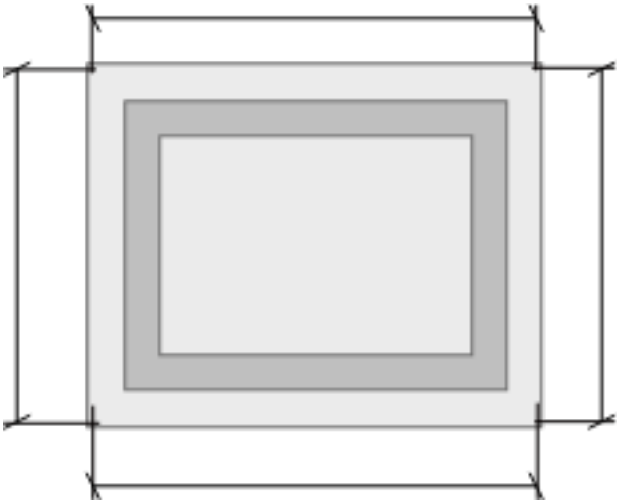
Steg 4: Med utgangspunkt i hjørnepunktene trekker vi to nye linjer, en fra hvert punkt. Disse representerer bøylearmene.

Steg 5: Til slutt samles alle tre linjene i en liste. Listen sorteres slik at linjene kommer i riktig rekkefølge for at de kan omformes til Revitelementer. For å unngå kollisjon mellom kantbøylene i hjørnene av platen forskyves bøylene som ligger langs langsidene en avstand lik diameteren til armeringsjernene i z-retning.



Figur 33: Stegene for kantbøylene i grubeplata

Fordelingslengde: Kantbøylene legges langs alle kanter i grubeplaten og ligger som ytterste armeringsjern. Fordelingslengden for hver side er derfor lengden til siden fratrukket Cnom på begge sider.

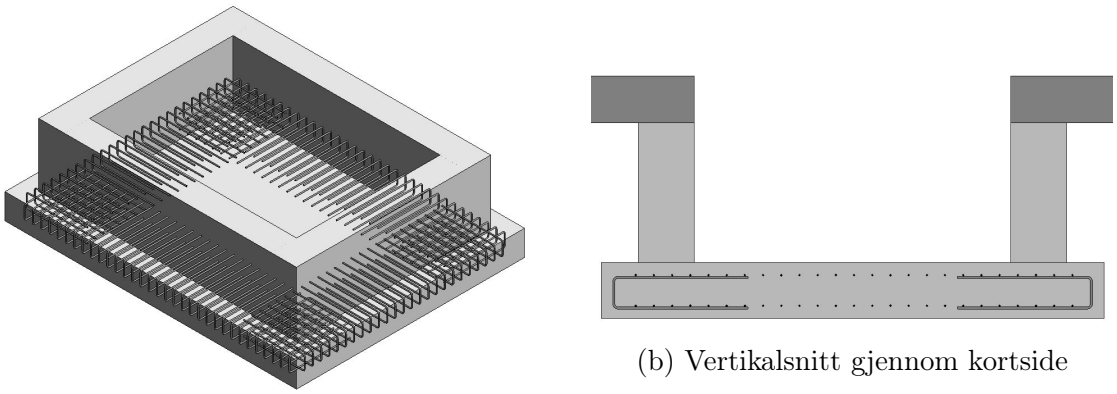


Figur 34: Fordelingslengden til kantbøylene i grubeplata

4.6.3 Lengdearmering i grubeplate

Formål

Denne delen skal plassere lengdearmeringen i grubeplata (P.10 og P.11)



(a) 3D-visning

(b) Vertikalsnitt gjennom kortside

Figur 35: Kantbøyler i grubeplate i Revit

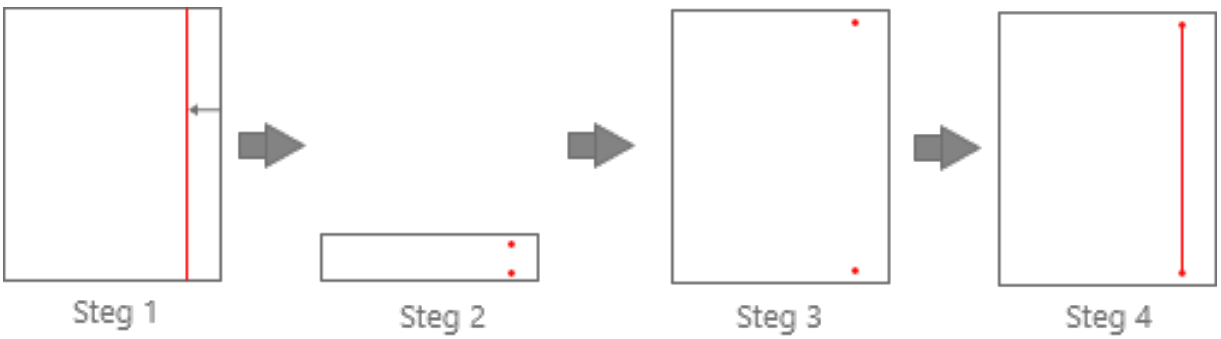
Fremgangsmåte

Steg 1: Henter ut linjen som ligger langs kanten på grubeplaten. Kopierer og flytter denne slik at vi får en ny linje som ligger med en avstand fra kanten som er lik overdekningen til bøylen. Avstanden er identisk til kantbøylenes forflytning.

Steg 2: Kopierer og flytter linjen i negativ z-retningen slik at vi får to nye linjer. Avstanden er identisk til bøylenes forflytning.

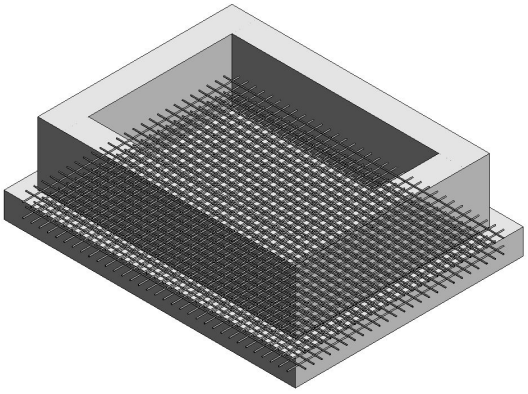
Steg 3: Vi finner så start- og endepunktet til begge linjene. Vi beveger oss langs linjen med samme distanse fra alle punktene og plasserer ut nye punkter. De nye punktene er start- og endepunktene for armeringsjernene.

Steg 4: Til slutt trekkes det linjer mellom start- og sluttpunktene. Linjene er geometrien til armeringsjernene. Deretter omformes disse til Revitelementer og fordeles utover.

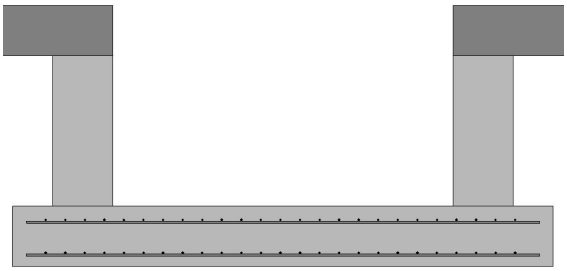


Figur 36: Stegene for lengdearmeringen i grubeplata

Fordelingslengde: Lengden lengdearmeringen skal fordeles over er den samme for kantbøylene (se figur 34). For å unngå kollisjon med bøylene forskyves armeringsjernene horisontalt (sett i vertikalsnitt) i en avstand lik halve av sin egen- og bøylens diameter.



(a) 3D-visning



(b) Vertikalsnitt gjennom kortside

Figur 37: Lengdearmering i grubeplate i Revit

5 Resultat

5.1 Spørreundersøkelsen

I dette delkapitlet skal vi legge frem resultatet av spørreundersøkelsen vi foretok. Undersøkelsen er anonymisert, dermed er dataene vi mottar presentert kun i prosentandel, og refererer ikke til hvem respondentene er. For spørsmålet som har «annet» som svaralternativ viser vi tekstsvarene. I denne delen av oppgaven skal vi kun fremvise dataene til spørsmålene vi anser som mest aktuelle og relevante til delproblemstillingen. Deretter analyserer vi resultatene i diskusjonskapittelet.

Undersøkelsen ble besvart av seksten fagpersoner fra de respektive aktørene tidligere nevnt i delkapittel 3.2.1. Respondentene har et faglig bredt spekter og har erfaring innen BIM og modellering. Vi anser dem derfor som pålitelige, og at dataene vi får er valide. De selekterte fagpersonene har stillinger som; fagspesialist, sivilingeniør, leder for digitale tjenester og samlinger (CDO), BIM-leder, BIM-koordinator, avdelingssjef og prosjektleder i digitalisering. Likevel vil vi påpeke at de empiriske dataene ikke er tilstrekkelige til å generalisere resultatene, men gir heller en indikasjon av bransjen. Spørreundersøkelsen henvises i vedlegg H.

Hva er din erfaring med parametrisk modellering?

Svar	**Antall**	**Prosent**
Har aldri hørt om	2	12,5
Har hørt om det, men har aldri brukt det	5	31,2
Kjenner til det og har jobbet litt med det utenom prosjekter	5	31,2
Kjenner veldig godt til det og bruker det ofte til modellering av prosjekter	4	25

Tabell 3: Spørsmål 1

Hvilke arbeidsoppgaver løser du ved hjelp av parametrisk modellering?

Svar	**Antall**	**Prosent**
Utforming av kompleks geometri	5	31,2
Armering	5	31,2
Bærekonstruksjoner (søyler, dekker, bjelker)	2	12,5
Annet	11	68,8

Tabell 4: Spørsmål 2

Annet:

- Automatisering av diverse
- Lage parametere, og flytte parametere for IFC-eksport
- Automatisering av informasjon i Revit
- Modellering av infrastruktur i Infraworks
- Bruker det ikke
- Kjenner til prinsippene, men bruker ikke programvaren selv
- Ikke brukt
- Kjenner til prinsippene, men bruker ikke programvaren selv
- Vet ikke hva som menes med parametrisk modellering, så svarene under blir nøytrale
- Bruker det ikke
- Bruker det ikke

5.2 Dybdeintervjuene

I denne delen har vi valgt å ikke bruke direktesitat fra transkriberingen av dybdeintervjuene med Marcin Luczkowski og Fredrik Jacobsen, men heller få frem essensen av svarene. Vi mener dette er en mer hensiktsmessig måte for gi en mer sammenhengende og relevant fremstilling av svarene. Vi henviser til transkriberingen av dybdeintervjuene i vedleggene I og J. For at vi skal svare på problemstillingen velger vi å ikke vise til spørsmålene, men heller tematikken i det som blir sagt. Vi har kontaktet intervjuobjektene og de har godkjent fremstillingen av dybdeintervjuene.

Definere parametrisk modellering

Her spør vi intervjuobjektene om deres tolkning av parametrisk modellering. Fra intervjuene merker vi at fagpersonene har forskjellige meninger rundt temaet. Marcin Luczkowski mener begrepet kan omfatte mye og mange misforstår hva det faktisk innebærer. Fredrik Jacobsen har også en oppfatning av at mange bruker begrepet uten å egentlig forstå hva det innebærer. Jacobsen mener bransjens oppfattelse av parametrisk modellering er synonymt med visuell programmering. Jacobsen utdyper videre at visuell programmering er et verktøy, mens parametrisk modellering bare er en generell beskrivelse av det som blir gjort. Luczkowski forteller at mange vektlegger programvaren for mye og ikke prosessene som faktisk blir gjort for å kalle det parametrisk modellering. Luczkowski mener parametrisk modellering ikke handler om hvilken programvare en bruker i seg selv, men hvordan en faktisk bruker den, ved å lage koder og algoritmer til å hjelpe design-/modelleringsprosessen. Det at brukeren av modellen kan endre parametere og observere automatisk endring. Luczkowski forteller videre at parametrisk modellering for ham i stor grad handler om å automatisere, optimalisere og overvinne begrensningene i modelleringsprogrammene. Luczkowski forklarer at et annet ord for denne metoden innenfor akademia er *Algorithms-Aided Design*, AAD.

Ulemper og fordeler med parametrisk modellering

Luczkowski ser mange fordeler. Gjennom arbeidet sitt i Multiconsult og som postdoktor på NTNU har han arbeidet mye med automatisering og optimalisering. Luczkowski mener begrensningen hovedsaklig ligger hos designeren, mer enn programvaren. Han mener også at omfanget på prosjektet i stor grad er med på å bestemme om bruken av parametrisk modellering vil skape økt effektivitet. Dersom det ikke er nødvendig kan parametrisk modellering ta opp mer tid og energi enn ved tradisjonell modellering. Jacobsen mener det samme, og forteller at hvis omfanget er for stort kan tradisjonell modellering være enklere og mer effektivt. Jacobsen påpeker også ulempen ved at det ikke finnes et standardisert oppsett for gjennomføring, og for utenforstående kan det være vanskelig å forstå skriptet. Det kan igjen føre til at færre kommer med innspill og tilbakemeldinger. Luczkowski mener også at programvaren i seg selv kan være en begrensning for bruken av parametrisk modellering. Han refererer eksempelvis til Dynamo, som han mener er tregt og har dårlig grafisk motor. Dette gjør det tidkrevende å modellere

eksempelvis armering, ettersom det er mange uavhengige variabler som skal inn i skriptet.

Effektivt arbeid med parametrisk modellering

Begge intervjuobjektene setter fokus på omfanget. Størrelsen av prosjektet og oppgavene en skal løse er avgjørende for om en bør bruke parametrisk modellering. Skriptet bør kunne brukes til fremtidige prosjekter og repetitive oppgaver. Luczkowski sier at mye av hans arbeid går ut på å lage koder for å automatisere designprosessen. Tiden som blir brukt til å lage en parametrisk modell kan være lenger enn ved å modellere tradisjonelt, men ved neste prosjekt kan den parametriske modellen bli brukt igjen. Hensikten er da at en kun trenger å endre på parametere eller mindre deler av skriptet for å oppnå ønsket resultat. Jacobsen forteller at parametrisk modellering oppleves effektivt der det ikke finnes god arbeidsflyt, om arbeidsoppgaver er vanskelige eller ikke direkte støttes av programvaren til 3D-modellering. Han fortsetter med at dersom en prøver å gjøre for mye, kan skriptet bli for stort og ugunstig.

Utviklingen av parametrisk modellering

Luczkowski sier at akkurat nå er ikke alle prosesser fullt automatisert, men i løpet av 50 år vil det skje en stor utvikling. Da vil kunstig intelligens ha en større rolle, og parametrisk modellering er et steg mot dette. I løpet av et tidsperspektiv på fem år vil parametrisk modellering fortsatt være på vei opp og flere bedrifter vil rette seg mot det. Jacobsen ser at flere nyutdannede retter seg mot metoden, og at bruken av programmering vil utvikle seg over tid. Avslutningsvis mener begge intervjuobjektene at ikke alt bør eller må gjennomføres ved parametrisk modellering.

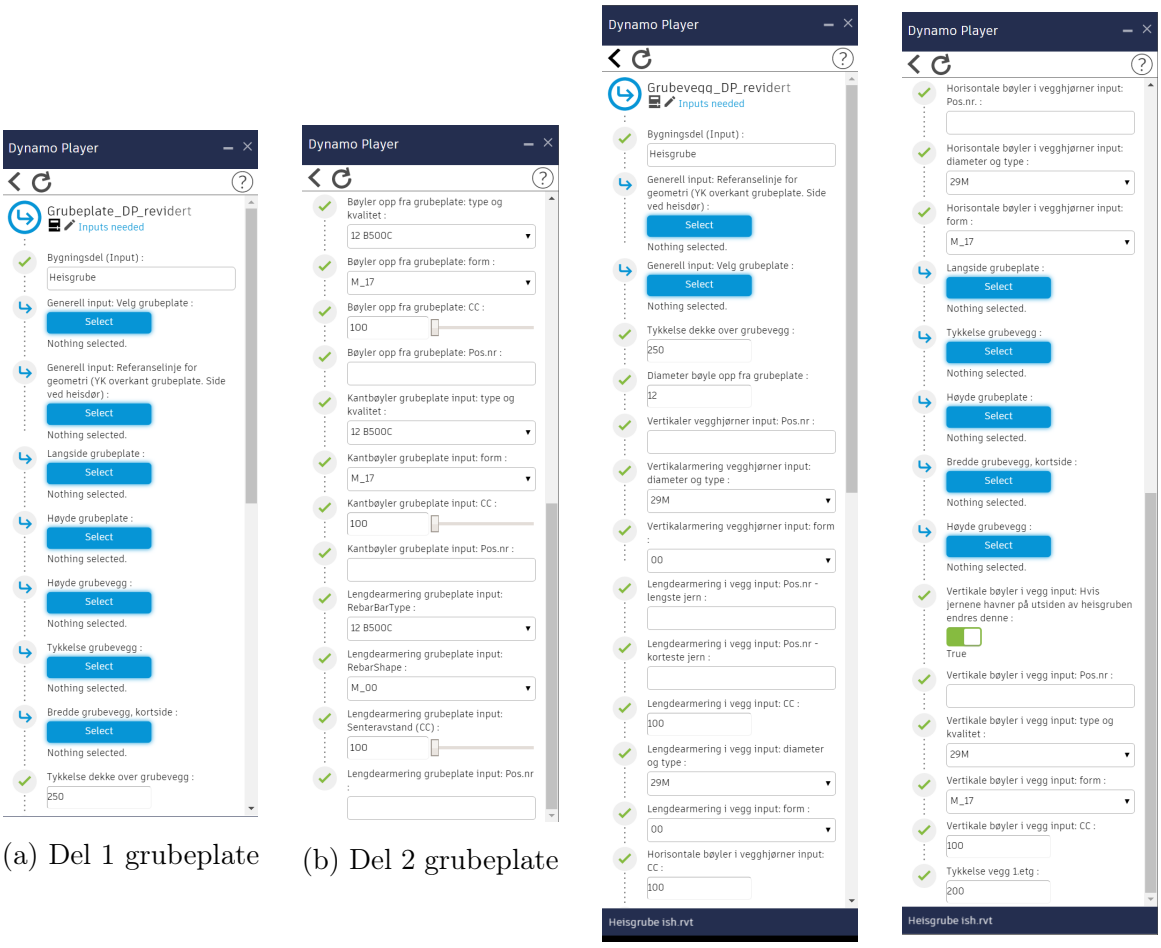
5.3 Dynamoskript

Den praktiske utførelsen ga oss en del resultater som vi skal fremlegge i dette delkapittelet. Resultatene deles inn etter kriteriene vi har satt for effektivisering (struktur, brukervennlighet, generaliserbarhet og tidsbesparelse). For å få bedre innsikt i brukervennligheten og tidsbesparelsen har vi fått hjelp av tre medstudenter som har testet skriptet. Medstudentene testet skriptet i Dynamo Player, der vi tok tiden på dem.

Struktur: Skriptene i Dynamo har en struktur som tydelig skiller de ulike delene fra hverandre. Inndelingen er gjennomfør i form av gruppering av noder, posisjonering av grupper og tekstbeskrivelser. Grupperingen er gjort med hensyn til hvilke oppgaver de ulike nodene har (input, mellomregninger, lage geometri, omdanne geometri til revitelementer). Fremgangsmåten for de ulike delene av skriptene er relativt lik.

Brukervennlighet: Resultatene for brukervennlighet er basert på hva som kreves av brukeren for å anvende skriptene. Begge skriptene vi har laget er tilgjengeliggjort i Dynamo Player. Dette tillater brukeren å kjøre skriptene uten å forlate Revits grensesnitt. Vi har også lagt inn tekstbeskrivelse i alle inputfeltene som beskriver hvilken informasjon/ hvilke verdier som må

fylles inn. De fleste dimensjoner som skriptet trenger fra Revit for å kjøre, hentes ut gjennom å klikke på elementer i modellen ved bruk av musepekeren.



(a) Del 1 grubeplate (b) Del 2 grubeplate (c) Del 1 grubevegg (d) Del 2 grubevegg

Figur 38: Dynamo Player input for grubeplate og grubevegg

Generaliserbarhet: Skriptene har blitt testet av oss i flere modeller med forskjellige senteravstander og dimensjoner på armeringsjernene. Disse testene har vist at armeringsjernene får riktig plassering, selv med ulike parametere som input (se figur 39 til 42). Skiptene er ikke prøvd til andre oppgaver enn heisgrube.

Tidsbesparelse: Tidsbesparelsen er delt i to; hvor lang tid det tar å bruke de ferdige skriptene og tiden det har tatt å lage de.

Testing av skript er gjennomført av fem testobjekter; tre medstudenter, en ansatt hos Seim & Hultgreen og en av forfatterne. Resultatene fra testene vises i tabell 5. For forfatteren og medstudentene startet tidtagningen når Dynamo Player ble åpnet, og stoppet når heisgruben var fullstendig 3D-armert. Medstudentene fikk tildelt testfilen og armeringstegninger, samt en rask veiledning i hvordan de skulle bruke Dynamo Player. Grunnet Covid-19-pandemien ble testingen til Seim & Hultgreen ikke utført på samme måte. Skriptene og testfilen ble oversendt per mail, og resultatet sent tilbake til oss etter gjennomført test.

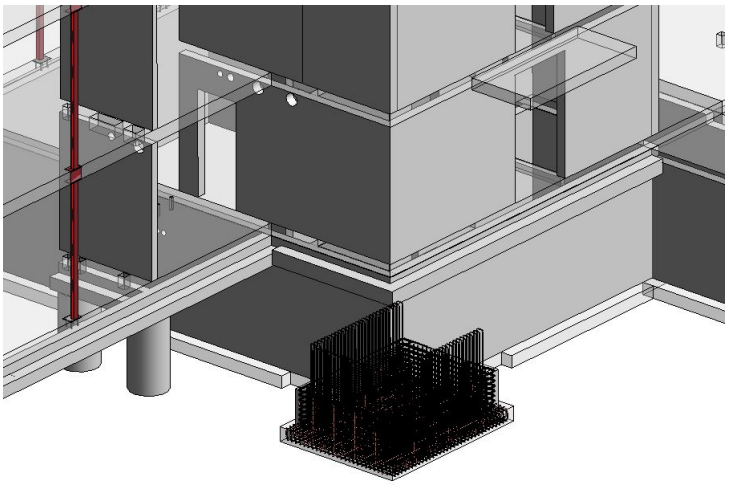
Vi har brukt ett semester på å lage skriptene fra bunnen av. Her inngår både tiden for å lære seg programvaren og for selve skriptingen.

Testpersoner	**Tid**
Forfatter	6 minutter
Medstudent 1	17 minutter
Medstudent 2	21 minutter
Medstudent 3	18 minutter
Seim & Hultgreen	10 minutter

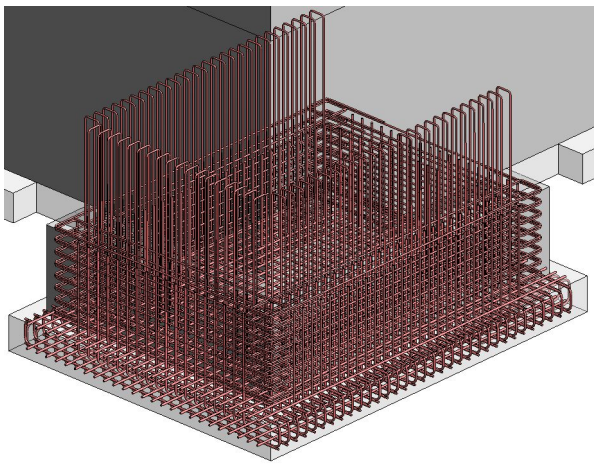
Tabell 5: Tidtagning av skript

Modeller i Revit

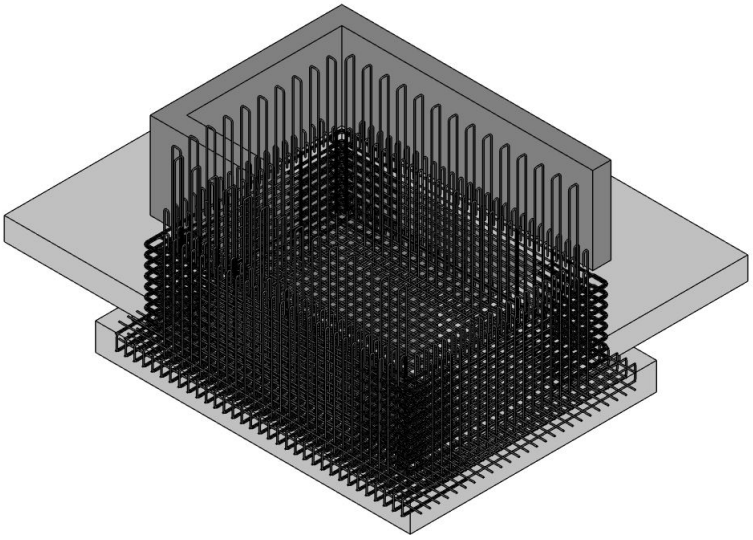
Figurene under viser ferdig armerte heisgruber i Revit. Alle modellene er armert ved bruk av Dynamo.



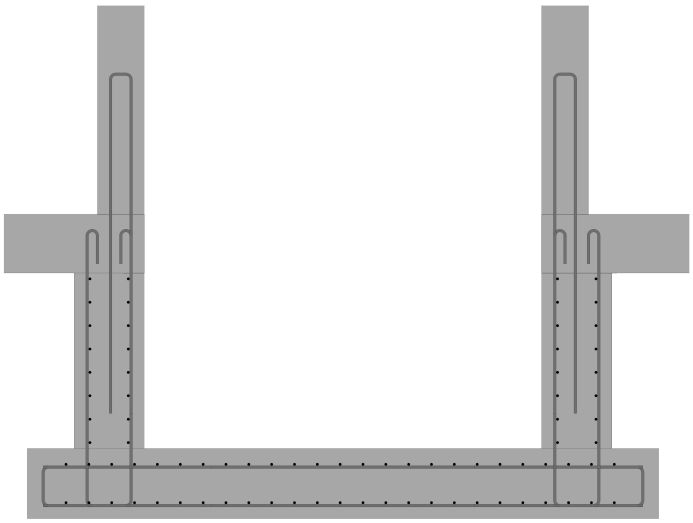
Figur 39: Heigrube i prosjektfil - figur 1



Figur 40: Heigrube i prosjektfil - figur 2



Figur 41: Heigrube i testmodell - 3D



Figur 42: Heigrube i testmodell - snitt gjennom langsider

6 Diskusjon

6.1 Analyse av spørreundersøkelsen

I denne delen skal vi analysere spørreundesøkelsen. Vi valgte å korte ned på antall spørsmål fordi det ga mer relevante svar til delproblemstillingen.

Hva er din erfaring med parametrisk modellering?

Spørsmålet i tabell 3 er ment for å kartlegge intervjuobjektenes kjennskap til begrepet parametrisk modellering. Dette hjelper oss med å konstatere validiteten hos respondentene for videre spørsmål. Vi ønsker å finne ut både om de har hørt om begrepet og i hvilken grad de bruker denne metoden til sitt daglige arbeid. Det er her viktig å påpeke at spørsmålet ikke tar hensyn til om respondentene forstår hva begrepet innebærer. Vi ser av resultatet at 88,5% har hørt om

det og 25% har benyttet det til prosjekter. Dette samsvarer med vår hypotese om at de fleste er kjent med begrepet, men få praktiserer metoden. En grunn til dette kan være at metoden ikke er godt nok implementert i bransjen enda. Samtidig må vi ta høyde for at stillingene til flere av respondentene ikke legger opp til bruk av metoden.

Hvilke arbeidsoppgaver løser du ved hjelp av parametrisk modellering?

Spørsmålet i tabell 4 er ment for å få en dypere innsikt i hvilke typer oppgaver som løses ved bruk av parametrisk modellering. Spørsmålet er flervalg, dermed kan respondentene huke av flere alternativer og de har mulighet til å utdype eller tilføye egne tekstsvar. Alternativene vi har lagt inn er valgt med tanke på hvilke oppgaver vi ser som mest interessante og relevante knyttet opp mot vår oppgave. Komplekse geometrier og armering er de vanligste bruksområdene med 31,2%. Videre ser vi at metoden brukes til et vidt spekter av oppgaver som innebærer alt fra bærekonstruksjoner til infrastruktur. Sett i sammenheng med vår hypotese er resultatet som forventet. Vi ser at det brukes til flere områder, men fritekstsvarene er upresise og dataene kan ikke kvantifiseres.

Refleksjon

Resultatene fra spørreundersøkelsen ville hatt større reliabilitet med flere respondenter. Vi ser på dataene som utilstrekkelige for å kunne generalisere. Grunnet tidsrammen ble ikke omfanget større. Dette begrunnes med at kun 56,25 % av respondenter har brukt parametrisk modellering til reell prosjektering. Ved å selektere respondenter etter relevant erfaring ville resultatet hatt en større validitet. På grunn av dette anser spørreundersøkelsen å ha lav statistisk signifikans. Likevel ser vi at de som har svart på spørsmål 2, har erfaring med bruk av metoden. Vi anser derfor svarene som valide.

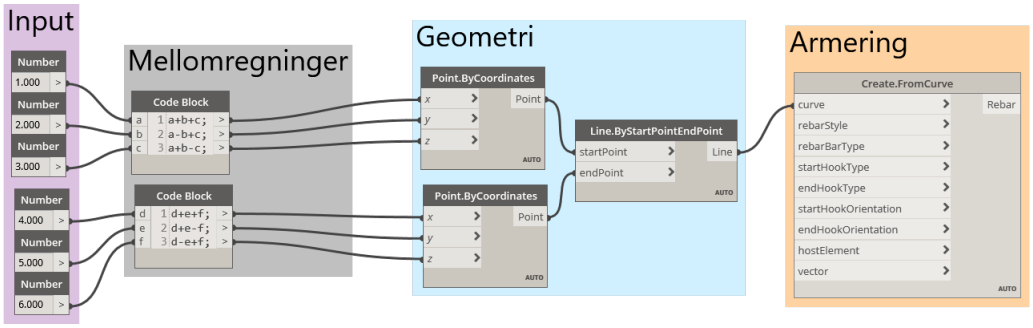
6.2 Evaluering av skript

I denne delen skal vi analysere skriptet. Analysen blir gjort med utgangspunkt i kriteriene vi har definert for effektivitet (se kapittel 3.4). Her legger vi også ekspertuttalelsene fra dybdeintervjuene til grunn for å drøftingen.

6.2.1 Struktur

Det har vært viktig for oss gjennom arbeidet med skriptene å holde en mest mulig ryddig struktur i arbeidsflyten. Struktureringen har foregått gjennom å samle noder inn i grupper, flytte all input til starten av arbeidsflyten og beskrive hva de ulike delene av skriptet gjør med tekst. Dette er gjort for å gjøre det mer oversiktlig for vår del under arbeidet, men samtidig med fokus på at andre brukere skal kunne ta det i bruk og gjøre endringer i ettertid. Gruppering og navnsetting vil ideelt sett gjøre det enklere for utenforstående å se hvilke oppgaver de ulike delene av skriptet utfører. Valgene vi har gjort har ført til at skriptene er "relativt" oversiktlige.

Dette kan vi si med utgangspunkt i våre egne erfaringer og tilbakemeldingene vi har fått fra Seim & Hultgreen underveis.



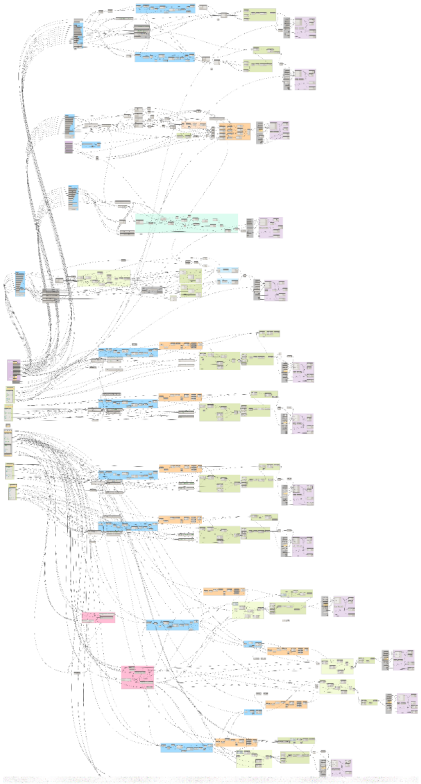
Figur 43: Illustrasjon av grupperingen av noder

Selv om vi har valgt en metode for å strukturere arbeidsflyten, har vi merket oss at det kan være utfordringer knyttet til det å definere hva som er en god struktur. Som Jacobsen fortalte i sitt dybdeintervju, finnes det ikke et fast oppsett for hvordan en arbeidsflyt skal se ut. I fraværet av et standardisert oppsett åpnes det for store variasjoner i hvordan arbeidsflyter kan settes sammen. Dette innebærer at hva som definerer en god struktur vil være subjektivt for den som lager skriptet. Det viktigste punktet når det gjelder struktur, mener vi, vil være at det er mest mulig intuitiv. For vår del har nøkkelen vært regelmessig kommunikasjon med personene som skal forvalte og bruke skriptet i sitt daglige virke.

På tross av at gruppering og gjennomtenkt plassering av gruppene kan gi en bedre oversikt over arbeidsflyten i sin helhet, ser vi også at det kan være med på å hindre en viss flyt når en skal analysere den stegvis i Dynamo. Valgene vi har tatt for gruppering og posisjonering av noder avhenger ikke nødvendigvis av hvor dataen de behandler skal videreføres i arbeidsflyten, men heller av typen oppgaver de utfører. Det innebærer at selv om verdien fra en mellomregning kanskje ikke skal brukes før i slutten av skriptet, er kodeblokken plassert allerede i starten av arbeidsflyten. Resultatet av dette kan være utfordringer knyttet til å følge arbeidsflyten i Dynamo, da inputen i en node kan være output fra en annen node som ikke har en nærliggende plassering. Hvorvidt dette er en god eller dårlig struktur er det ikke faglig enighet om. Det viktigste vil være å velge en metode, og deretter forholde seg til denne. En fast, dårlig struktur vil uansett være bedre enn ingen struktur.

En annen faktor, kanskje den viktigste, som påvirker strukturen i skriptet, er størrelsen. Helt fra starten av vårt arbeid i Dynamo har vi observert hvordan størrelsen, altså antallet noder og tråder, påvirker *opplevelsen* av strukturen. Vi tok tidlig et aktivt valg om å dele Dynamoskriptet i to deler, en for grubeveggen og en for grubeplaten. Bakgrunnen for vårt valg var hovedsakelig å gi en bedre oversikt over arbeidsflyten og mulighet for at flere brukere kan arbeide med ulike deler samtidig. I retrospekt ser vi på dette som et fornuftig valg. Allerede i startfasen av arbeidet i Dynamo opplevde vi at arbeidsflyten kunne bli uoversiktlig. Grup-

pering og tekstbeskrivelser hjalp i stor grad, men kun inntil et visst punkt. Når størrelsen på skriptet ble for stor var skapelsen av et "spaghettimonster" uunngåelig (se figur 44). Dette kan knyttes opp mot omfanget av oppgaven skriptet skal løse i sin helhet. Både Luczkowski og Jacobsen understreker at omfanget av prosjektet, altså oppgaven, i stor grad vil være avgjørende for om bruken av parametrisk modellering vil være hensiktsmessig. I arbeidet med å armere en heisgrube er det mange parametere som spiller inn i plasseringen og geometrien til armeringsjernene. Et resultat av dette er at det *vil* bli mange noder og tråder, og disse *må* sorteres og kobles riktig. Selv med en oppdeling mener vi å ha nærmet oss grensen for hva som faktisk er mulig å holde oversiktlig. Vi har selv erfart at størrelsen på, og antallet parametere i, våre skript er svært omfattende. Strukturen tatt i betraktning, vurderer vi den tradisjonelle metoden som mer hensiktsmessig for å armere heisgruber.



Figur 44: Spaghettimonster

6.2.2 Generaliserbarhet

Kriterier vi har satt for generaliserbarhet, er i hvilken grad skriptet kan benyttes på flere områder og for andre bygningsdeler enn heisgruber. Slik vi ser det, avhenger generaliserbarheten i stor grad av strukturen til skriptet. Som et resultat av at vi anser skriptet som godt strukturert, mener vi at det har grunnlag for å være generaliserbart.

Fra resultatene (se figurene 39 til 42), ser vi at skriptene fungerer i alle modellene vi har testet det i. Forskjellene mellom modellene er hovedsakelig dimensjoner og orienteringen av betongelementene. I tillegg har vi gjennomført tester med ulike diametere og senteravstander på armeringsjernene. Resultatene vi har fått viser seg å være positive. Ved å ta høyde for ulike

orienteringer, observerer vi at armeringsjernene plasseres riktig i forhold til vertsobjektene. Dette vil si at uansett hvordan vi har rotert Revitelementene i xy-planet, plasseres armeringsjernene riktig opp mot den valgte referanselinjen og Revitelementene. Ut ifra dette kan vi si at skriptene er et valid alternativ til modellering av armering for heisgruber i disse modellene. Vi kan dermed si at skriptet er delvis generelt. På tross av dette må vi ta høyde for at det kan være avvik som ikke har blitt fanget opp under våre tester. For å redusere sannsynligheten for eventuelle feil ville det vært hensiktsmessig å teste skriptene mer. Ved å bruke flere ulike testfiler og legge inn "alle" mulige kombinasjoner av input, ville resultatene fått større reliabilitet. Ideelt sett hadde skriptet blitt testet av Seim & Hultgreen over en lengre tidsperiode på ulike prosjekter. Grunnet oppgavens tidsramme legges resultatene vi har per nå til grunn for konklusjonen.

Tidsrammen er også en begrensende faktor når vi analyserer skriptets *totale* generaliserbarhet. Verken vi eller vår samarbeidspartner har hatt mulighet til å hente ut deler av skriptet og bruke det til andre konstruksjonsdeler enn heisgruber. På tross av dette, er vi av den oppfatning at dette kan være mulig. Gjennom vårt fokus på god struktur i arbeidsflyten, samt resultater som viser at armeringsplasseringer blir korrekt, mener vi at det er deler av skriptet som kan brukes til andre oppgaver. Som nevnt tidligere, er det mange faktorer og variabler vi ikke har fått kontrollert. Dette medfører at vi ikke ser på skriptet som totalt generalisert, men har potensiale til å brukes til flere oppgaver.

Som beskrevet i kapittel 4, har vi valgt å benytte de samme prinsippene for å lage geometri i alle delene av skriptene. Dette innebærer at all forflytning av geometri i Dynamo gjøres med utgangspunkt i samme referanselinje fra Revitmodellen. For vår del har dette vært en god løsning, ettersom vi har kunnet benytte relativt like mellomregninger til alle armeringsjernene. Forskjeller i mellomregningene er stort sett å legge til eller trekke fra diametere etter plassering opp mot nærliggende armeringsjern. Metoden har også gjort at nodene vi bruker i begge skriptene er tilnærmet like. Et færre antall ulike noder, og mest mulig identiske skript, er i våre øyne med på å gjøre det enklere å forstå hva de ulike delene av skriptet gjør. Dette, i kombinasjon med strukturen, vil gjøre det være enklere å hente ut deler eller gjøre endringer i skriptet i ettertid. Dette styrker generaliserbarheten. Samtidig ser vi at en tilnærming der vi ønsker å gjøre mest mulig likt kan være med på å komplisere noen operasjoner. Sannsynligvis kunne skriptet hatt færre mellomregninger ved bruk av ulike fremgangsmåter. I stedet for å konsekvent bruke én referanselinje for all geometri, kunne geometrien til hvert armeringsjern hatt utgangspunkt i en referanse i Revitmodellen med en nærmere plassering. Dette hadde sannsynligvis gitt færre mellomregninger, men samtidig skapt et behov for flere input og en større variasjon av fremgangsmåter i skriptet.

6.2.3 Brukervennlighet

Et av målene for skriptene, har vært at de skal være brukervennlige. Hensikten med dette, er å senke kompetansenivået som trengs for å bruke de. Når vi ser på brukervennligheten, fokuserer

vi på hvordan skriptene oppleves når de skal anvendes til å modellere. Fordi vi har valgt å tilgjengeliggjøre begge skriptene i Dynamo Player, trenger ikke brukeren å åpne Dynamo for å kjøre de. Dette innebærer at skriptenes struktur ikke påvirker brukervennligheten.

Hovedtiltaket vi har gjort for å oppnå en høyest mulig brukervennlighet er som tidligere nevnt, å tilgjengeliggjøre skriptene i Dynamo Player (se figur 38 i resultat). Dette gir brukeren muligheten til å fylle inn all input uten å forlate Revit sitt grensesnitt. Når skriptet kjøres gjennom Dynamo Player vil brukeren ikke ha noen annen informasjon å forholde seg til enn teksten som står ved inputfeltene. Vi har derfor fokusert på å gjøre denne så enkel og presis som mulig. For å oppnå dette har vi forsøkt å gjøre all tekst entydig, men samtidig kortfattet. Videre har vi lagt opp rekkefølgen på inputen på det vi mener er en intuitiv måte. Dette i form av at all input for hvert armeringsjern kommer gruppevis. Vi har redusert antallet tallverdier som må tastes inn manuelt ved å bruke noder som krever at brukeren klikker direkte i Revitmodellen med musepekeren. På denne måten kan vi hente ut de nødvendige dimensjonene til heisgruben, og samtidig redusere sannsynlighet for inntastingsfeil.

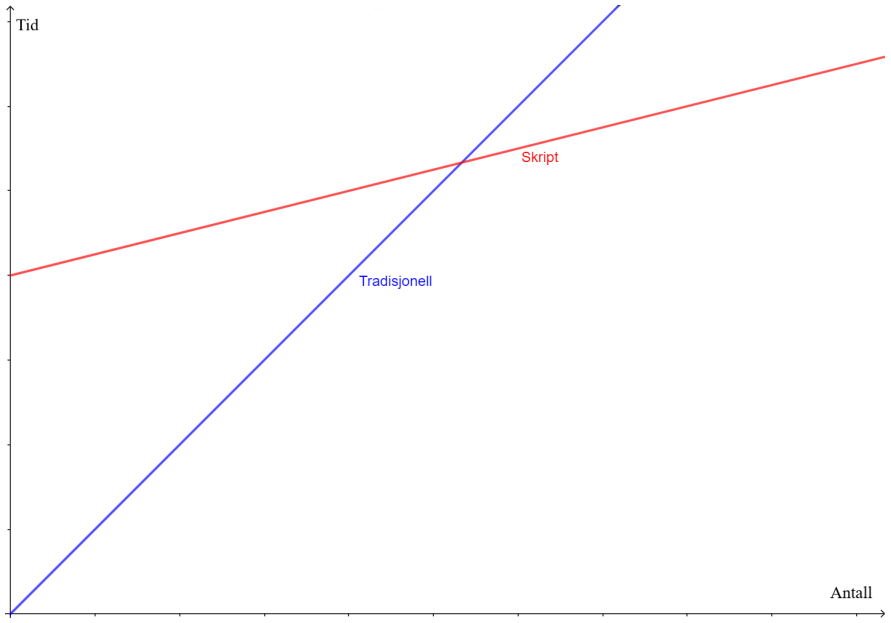
Selv om tiltakene vi har gjort er målrettet for å øke brukervennligheten, ser vi potensielle utfordringer knyttet til metoden. Spesielt med tanke på hvordan beskrivelser skal gjøres entydige. I likhet med strukturen i arbeidsflyten, vil oppfatningen av hva som er enkelt forståelig være subjektivt. Hvis skriptet skal benyttes av andre personer enn den/de som har laget det, vil det være ideelt med god kommunikasjon, slik at en at sikrer beskrivelsene er forståelige.

6.2.4 Tidsbesparelse

Det siste kriteriet vi legger til grunn for effektivisering, er tidsbesparelse. Det begrunnes med i hvilken grad tidsbruken reduseres ved bruk av Dynamo til forskjell fra tradisjonell modellering, og tidsbesparelsen over tid gitt at skriptet kan gjenbrukes.

Resultatene fra tidtakingen (se tabell 5) viser at det er relativt stor variasjon i tiden de ulike testobjektene har brukt. De med minst erfaring i modellering, medstudentene, brukte rundt tjue minutter, sammelignet med Seim & Hultgreen som brukte ti. Testobjektet blant forfatterne hadde den korteste tiden med seks minutter. Vi har valgt å fokusere på tidene med- og uten bruk av skript hos Seim & Hultgreens ansatte.

Figur 45 er ment som en illustrasjon på hvordan parametrisk modellering kan virke effektiviserende over tid (fremstillingen tar ikke utgangspunkt i reelle data). Her representerer y-aksen den totale tiden en har brukt på en prosess, og x-aksen antall ganger prosessen har blitt utført. Fra start ser vi at tidsbruken ved den tradisjonelle metoden vil være kortere enn ved parametrisk modellering (skript). Årsaken til dette er at utarbeidelsen av et skript sannsynligvis vil ta tid. I det samme tidsrommet vil en ved bruk av tradisjonell modelleringsmetode kunne gjennomføre prosessen flere ganger før skriptet er ferdig. Hensikten med Dynamo er å kunne effektivere selve prosessen. Selv om et skript på kort sikt sannsynligvis ikke vil være



Figur 45: Tidsbesparelse over tid

tidsbesparende, vil det kunne hente seg inn over en tidsperiode. En kan se på det som en investering som vil gi avkastning over tid. Hvor lang tid det vil ta før en faktisk kan se verdien rent tidsmessig, avhenger av to faktorer; hvor lang tid det tar å utarbeide skriptet og hvor mye raskere prosessen kan utføres ved bruk av skriptet.

Denne modellen kan vi også knytte opp mot våre resultater. I våre innledende samtaler med Seim & Hultgreen anslo de at de bruker omtrent femten minutter på selve modelleringen av en heisgrube direkte i Revit (se kapittel 1.2). Ved bruk av Dynamo ser vi at tidsbruken er ti minutter. Selv om vi tar høyde for at tiden kan reduseres etterhvert som brukeren blir bedre kjent med skriptet, vil differansen sannsynligvis være relativt liten. I tillegg må tiden vi har brukt på å lage skriptet, tas med i regnestykket. Vi har brukt omtrent ett semester på å ferdigstille produktet. Dette inkluderer tiden vi har brukt på å lære oss Dynamo, sette sammen skriptet og gjøre det mest mulig brukervennlig. Selv om mye av tiden for vår del har gått med til å lære Dynamo, anser vi skriptene vi har laget som relativt omfattende og krevende. Med den kunnskapen vi besitter i dag, kunne tiden vært betydelig redusert. Uansett ville det tatt flere dager å lage skriptene. Tidsbesparelsen per heisgrube vil derfor være liten sammenlignet med tiden som har gått med på å lage skriptene, noe som gjør at de må brukes over en lengre tidsperiode for å være tidsbesparende.

Det vi ser fra resultatene kan tyde på at skiptene vi har laget *ikke* er tidsbesparende. Sett i sammenheng med uttalelser fra Jacobsen og Luczkowski, kan dette knyttes opp mot omfanget av den parametriske modellen. Begge påpeker at hvis omfanget av oppgaven vi forsøker å automatisere er for stort, kan parametrisk modelling ta opp mer tid og energi enn tradisjonell modellering. I tilfellet med heisgruber ser vi på antallet variabler som en mulig årsak til at den parametriske modellen blir for komplisert.

Selv om skriptene ikke er tidsbesparende slik de står i dag, kan det gi positive effekter som vi ikke har fått målt. Bruken av en parametrisk modell kan standardisere prosesser innad i en bedrift, ettersom modellen alltid vil gi det samme resultatet.

6.3 Kilder til Dynamo

Et av nettstedene vi har benyttet oss mest av er *dynamobim.org*, som er utarbeidet av Autodesk&Modelab (u.å). Dette inneholder en rekke ressurser, blant annet *Dynamo Primer* og *dictionary*. Dynamo Primer er en beskrivelse av hvordan Dynamo fungerer, og har vært en nyttig introduksjon til programvare og bruken hvordan den brukes. Dictionary er et oppslagsverk med beskrivelser av alle nodene som finnes i Dynamos standardbibliotek. Primeren og oppslagsverket, i kombinasjon med opplæringsvideoene som også finnes på nettsiden, har dannet et godt grunnlag for å få en grunnleggende forståelse av hvordan Dynamo fungerer.

Selv om disse tre ressursene har gitt et godt grunnlag for bruk av Dynamo, beskriver de ikke alle utfordringer vi har møtt på undervies. For å finne løsninger på konkrete problemer har det åpne forumet på nettsiden til Dynamo (Forum, 2021) vært en god kilde til kunnskap. Forumet tillater alle med en brukerkonto å dele sine løsninger, men setter et merke ved forfatterens navn ut ifra hvilken kompetanse/erfaring personen har i Dynamo. Dette har gitt oss en større frihet til å velge de kildene vi har sett på som mest pålitelige.

Ved siden av Dynmos eget nettsted er *YouTube* og forumsiden *Stackoverflow* noen av de største gratissidene som tilbyr veiledning og løsninger til programmet. Til felles med Dynamos forum tillater disse sidene også at hvem som helst kan publisere. På YouTube har vi benyttet oss av et begynnerkurs i Dynamo laget av kanalen DiRoots. DiRoots er et firma som lager programvare og egendefinerte skript for sine kunder (DiRoots, u.å). I tillegg til gratissidene finnes det også kurs o.l. en kan betale for, men dette har vi ikke benyttet oss av.

Nødvendig kompetanse for Python-skriptingen i denne oppgaven er anskaffet gjennom selvlære før arbeidet med denne oppgaven startet, og et ekstrakurs på OsloMet.

Selv om det er mange gode kilder til Dynamo, har vi ikke funnet gode kilder som er direkte knyttet opp mot vår problemstilling. Dette har ført til mye prøv-og-feil underveis, spesielt for oppsett av struktur og arbeidsflyt i skriptene. Tilgang til slike kilder kunne potensielt ha kuttet ned tiden brukt på skripting.

7 Konklusjon

Gjennom denne oppgaven har vi utforsket mulighetene for å effektivisere 3D-BIM-modellering ved bruk av parametrisk modellering. Gjennom en praktisk utførelse har vi brukt Dynamo til å lage et skript for å armere en heisgrube i Revit. I tillegg har vi gjennomført to dybdeintervjuer og en spørreundersøkelse. Dybdeintervjuene har til hensikt å gi bedre forståelse av hvilket potensial og begrensninger som ligger i bruken av parametrisk modellering. Spørreundersøkelsens formål har vært å kartlegge bransjens kjennskap til begrepet, og få innblikk i hvordan metoden brukes i dag.

Grunnlaget for den praktiske utførelsen er et ønske fra Seim & Hultgreen om å finne en metode for å effektivisere armeringen av en heisgrube. Vi har vist at dette lar seg gjøre ved bruk av parametrisk modellering gjennom Dynamo. Ved bruk av Dynamo har tiden det tar å modellere armeringen i 3D-BIM-modellen blitt redusert med omtrent fem minutter. Sammenligner vi tiden det har tatt å utarbeide skriptene med tradisjonell metode, vil det ikke være tidsbesparende for oppgaven vi har løst. Samtidig ser vi muligheten for at deler av skriptet kan hentes ut, og brukes til å effektivisere andre oppgaver.

Resultatene viser også at omfanget av problemet Dynamo skal løse, er avgjørende for kompleksiteten til skriptet. Kompliserte skript kan bli store og uoversiktlige, og dermed utfordrende å behandle. Dette påvirker i stor grad hvor mye tid som vil gå med til å utarbeide den parametriske modellen. Dette underbygges av uttalelser fra Jacobsen og Luczkowski. Begrensning av omfanget til skriptet er derfor en essensiell del for effektiv bruk av parametrisk modellering. Det vi samtidig ser, er at kompleksiteten til skriptet ikke nødvendigvis påvirker brukervennligheten. Ved bruk av riktige tiltak kan kompetansenivået som kreves for å kjøre skriptene reduseres betraktelig. I Dynamo kan dette gjøres ved å tilgjengeliggjøre skriptet i Dynamo Player og legge inn entydige beskrivelser.

Av spørreundersøkelsen ser vi at selv om det kun er et fåtall som bruker parametrisk modellering i sitt daglige virke, er begrepet relativt kjent blant respondentene. Det er også en stor variasjon innenfor hvilke oppgaver det brukes til, som kan indikere at det har et bredt anvendelsesområde. På grunnlag av antallet respondenter har vi valgt å ikke legge for stor vekt på disse resultatene, men heller bruke de som en indikasjon.

For å konkludere, anser vi bruken av parametrisk modellering som effektiviserende for 3D-BIM-modellering, dersom det brukes riktig. Oppgavens omfang bør være så begrenset som mulig for å gjøre den parametriske modellen enkel å utarbeide og behandle i ettertid. I tillegg bør prosessen som automatiseres være repetitiv, slik at skriptet kan gjenbrukes. Vi ser samtidig at begrepet er kjent i byggebransjen i dag, men at det er få som praktiserer metoden.

8 Veien videre

Ettersom vi har opparbeidet oss erfaring og kompetanse innenfor bruken av Dynamo og parametrisk modellering, observerer vi flere punkter rundt utførelsen og strukturen av metoden som bør videreutvikles. Dette er punkter som kan effektivisere skriptene ytterlige:

- Utvikle skript til å generere merkelapper til armeringsjernene. Disse merkelappene skal inkludere senteravstand og diameter på armeringsjernene, samt posisjonsnummer.
- Utvikle skript til å generere armeringstegninger som inkluderer alle nødvendige snitt.
- Utvikle skript til å kjøre egne kollisjonstester etter skript kjøres.
- Utbedre skript til å legge armeringsjern ved siden av hverandre, korrigere automatisk om kollisjon skjer.
- Samle skriptene til ett skript, og fortsatt beholde strukturen og god brukervennlighet.
- Omgjøre skript til ren kode gjennom programmer som C Sharp (C#), og videre gjøre om til *plugin*. Utfordringer kan dog være at det blir vanskeligere å gjøre endringer uten kompetanse.

Kildeliste

Autodesk & Modelab. (u.å). *Dynamo primer.* https://primer.dynamobim.org. (Hentet 13. april 2021)

Autodesk Inc. (u.å). *Autodesk revit forside.* https://www.autodesk.no/products/revit/. (Hentet 25. april 2021)

Autodesk&Modelab. (u.å). *Dynamobim.* https://dynamobim.org/. (Hentet 5. mai 2021)

Bartolomei, R. T. (2019, 19. November). *Digitale verktøy gjør byggebransjen mer bærekraftig: Gir også færre feil.* https://www.fremtidensbygg.no/digitale-verktoy-gjor-byggebransjen-mer-baerekraftig-gir-ogsa-faerre-feil/.

DiRoots. (u.å). *Our main services.* https://www.diroots.com. (Hentet 12. mai 2021)

Forum. (2021). *Forum dynamobim.* https://forum.dynamobim.com/. (Hentet 25. april 2021)

Fremtidens Byggenæring. (2017, 09. august). *Parametrisk modellering - bedre enn standard bim.* https://www.fremtidensbygg.no/parametrisk-modellering-bedre-enn-standard-bim/.

Fu, Feng. (2018, februar). Design and analysis of complex structures. https://www.sciencedirect.com/topics/engineering/parametric-modeling. (Hentet 14. mai 2021)

Fudala, T. (2018). *Structural design.* https://dynamopackages.com/. (Versjon 2021.1.22)

Grønmo, S. (2021). *Algoritme. i store norske leksikon.* https://snl.no/algoritme. (Hentet 13. april 2021)

Halvorsen, Knut. (2008). *Å forske på samfunnet (5.utg.).* Oslo: J.W. Cappelens Forlag as.

Modelab & Parallax Team. (2019). *What is dynamo?* https://primer.dynamobim.org/$01_Introduction/1-2_what_is_dynamo.html$. (Hentet 04. mai 2021)

Offergaard, S. (2021, 05. mai). *Buildingsmart norge lanserer rapport om digital armering.* https://www.bygg.no/article/1466324.

Olsvik, E. H. (2020, 17. juli). *Generalisering. i store norske leksikon.* https://snl.no/generalisering.

Python Software Foundation. (u.å). *The python standard library.* https://docs.python.org/3/library/. (Hentet 20. mai 2021)

Python.org. (2021). *What is python?* https://www.python.org/doc/essays/blurb/. (Hentet 13. april 2021)

Seim Hultgreen, S&H. (u.å). *Om oss.* http://www.s-h.no/om-oss/. (Hentet 05. april 2021)

10 Vedlegg

Vedlegg A - Illustrasjon av snøring

Vedlegg B - Dynamo Player fra Revit sitt brukergrensesnitt

Vedlegg C - Armering- og formtegninger

Vedlegg D - Nodebeskrivelse - Dynamo dictionary

Vedlegg E - Nodebeskrivelse - Structural Design

Vedlegg F - Gjennomgang av skript for grubevegg

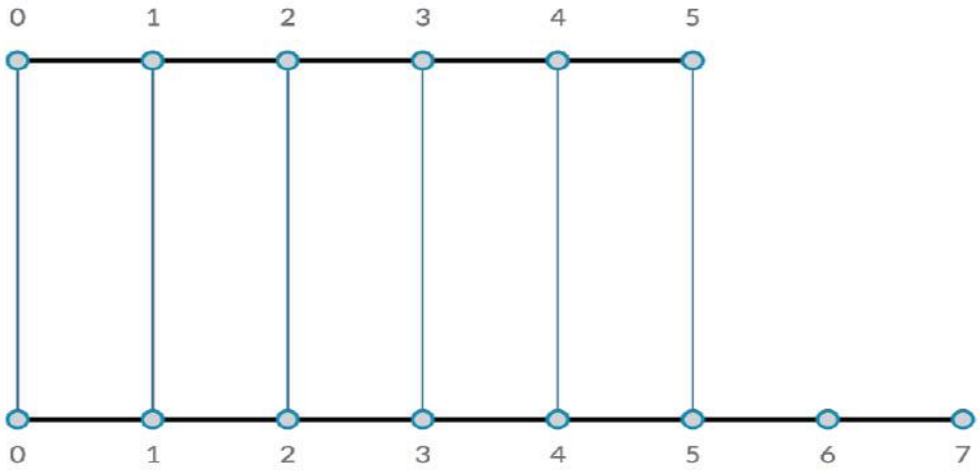
Vedlegg G - Gjennomgang av skript for grubeplate

Vedlegg H - Spørreundersøkelse

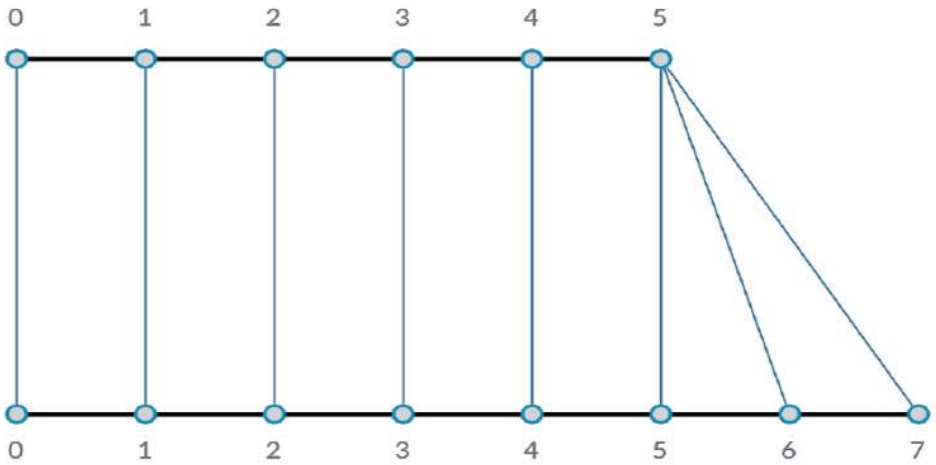
Vedlegg I - Transkribering av dybdeintervju med Marcin Luczkowski

Vedlegg J - Transkribering av dybdeintervju med Fredrik Jacobsen

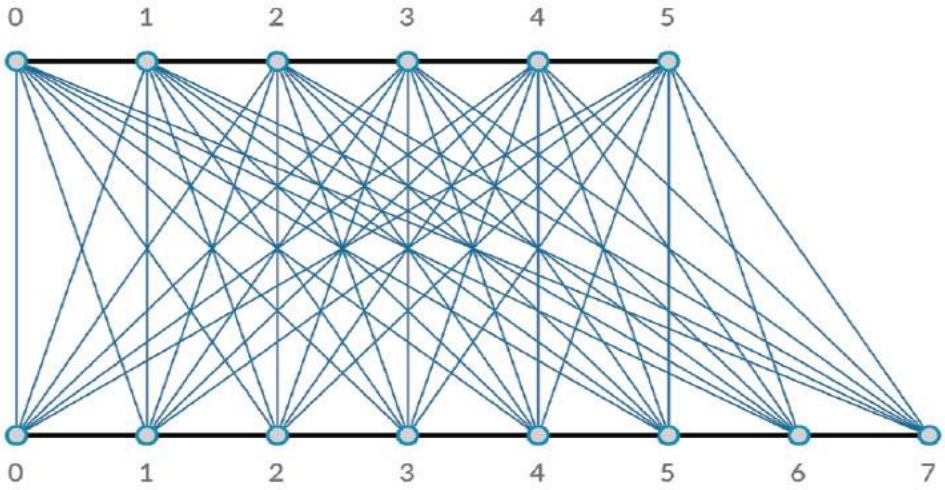
Vedlegg A - Snøring



Korteste

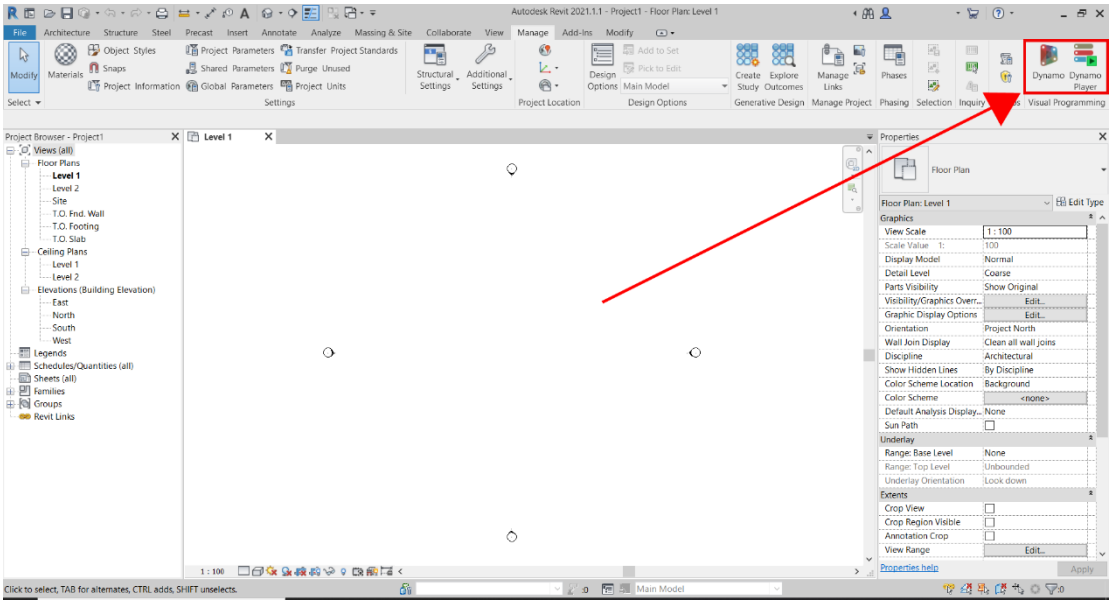


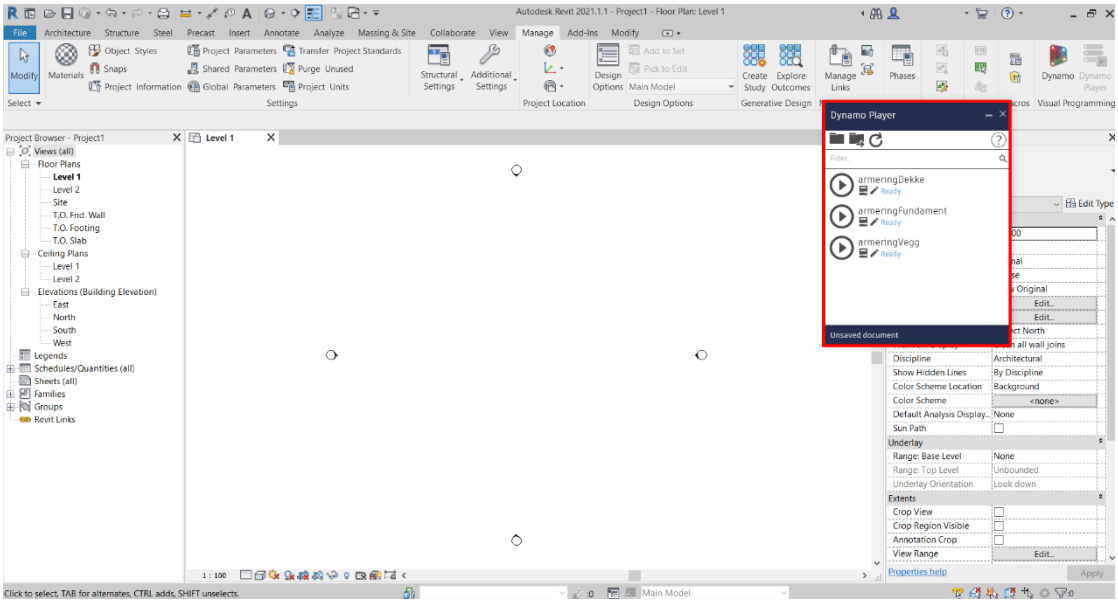
Lengste

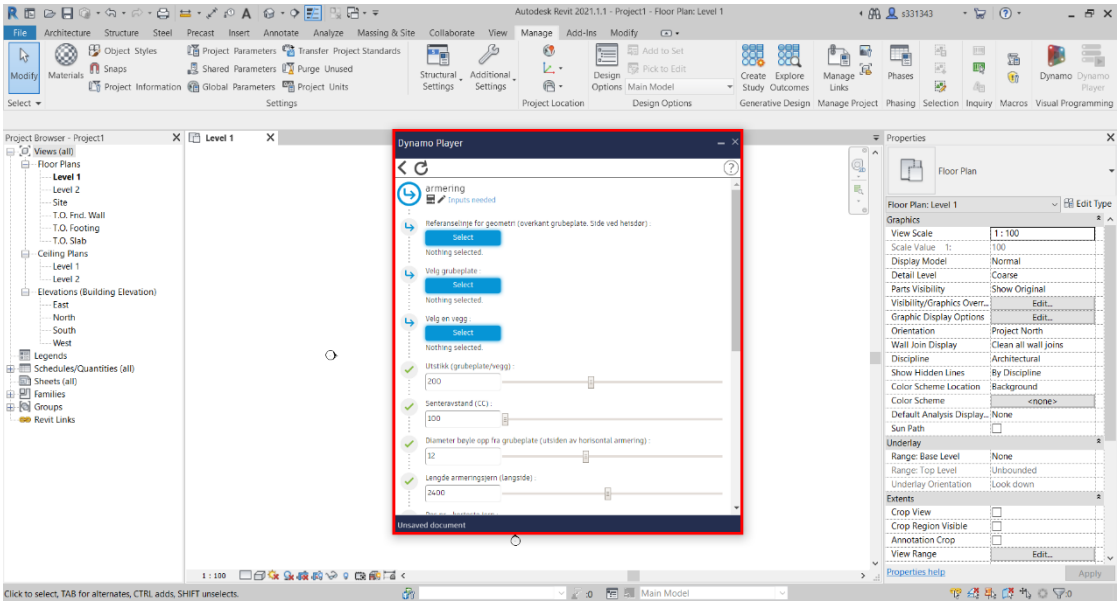


Kryssprodukt

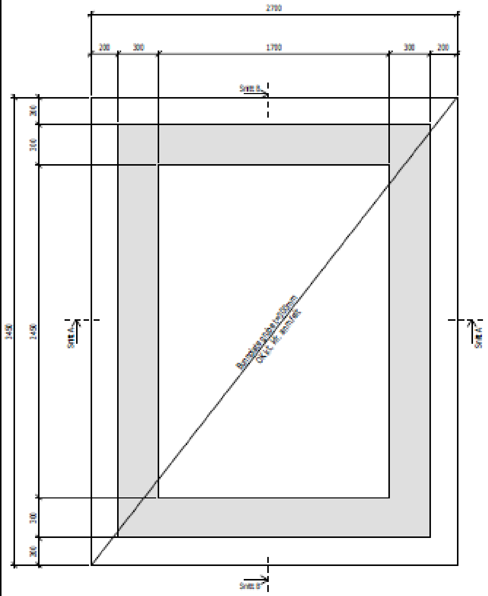
Vedlegg B – Dynamo Player fra Revit's brukergrensesnitt



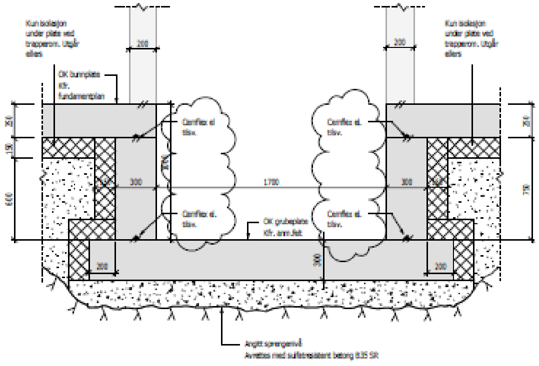




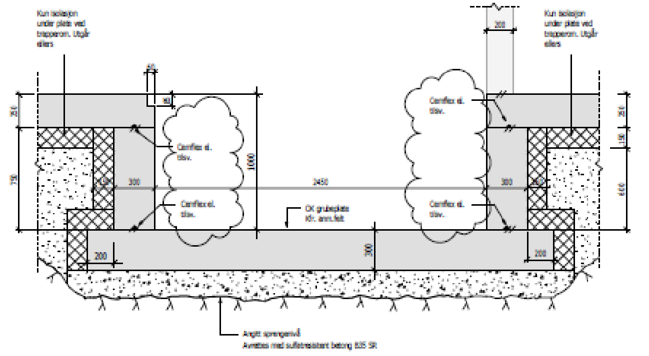
Vedlegg C – Armering- og formtegninger



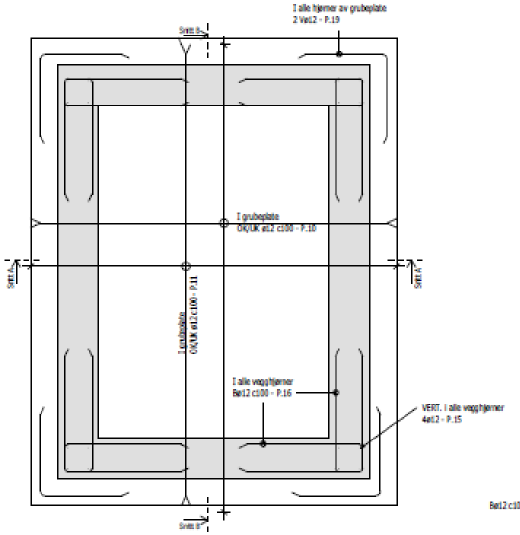
PLAN - FORM
1:20



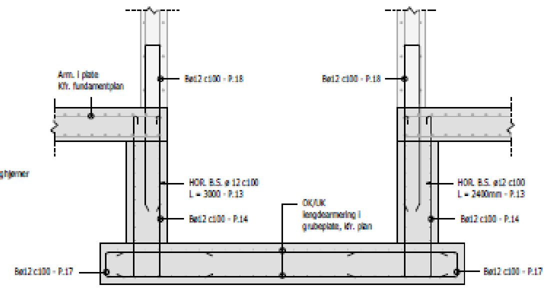
SNITT A - FORM
1:20



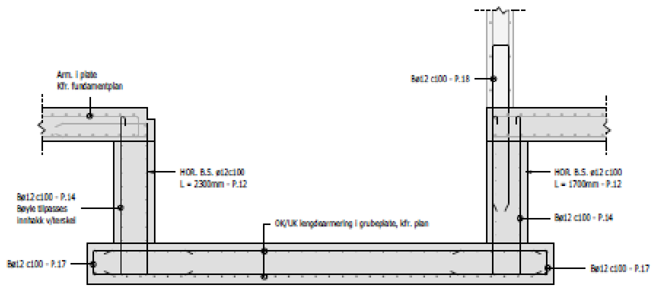
SNITT B - FORM
1:20



PLAN - ARMERING
1:20



SNITT A - ARMERING
1:20



SNITT B - ARMERING
1:20

ANMERKNINGER

Kontrollform: Normal
Utførelseskontrollklasse: UKK2
Utførelsesklasse betong: 2
Toleranseklasse betong: 1 (Normal)

PLASSTØPT BETONG
Betongkvalitet: B35
Betongkvalitet avretting: B35 SR
Armeringskvalitet: B500NC
Bestandighetsklasse: M40
Kloridklasse: cl 0,4
Herdeklasse: 3

ARMERING
Armeringskvalitet: B500NC
Nominell overdekning: 50mm ±10mm

Armering skjøtes med 50ø omfar
Omfaringsskjøt i ... - 100ø

ISOLASJON
Isolasjon rundt grubevegger: XPS-200 eller bedre
Isolasjon under trapperom: XPS-400 eller bedre

HENVISNINGER
Graveplan - X-E-PI-XX-00
Pilarplan, hus n - n-B-PI-U1-01
Fundamentplan, hus n - n-B-PI-U1-02/n-E-AR-U1-02 (form/armering)
Detaljsnitt fundament - X-E-D1-XX-05/X-E-AR-XX-05 (form/armering)

Bøyeliste heisgruber - BI-n-10

(n = A, B, C1, C2)

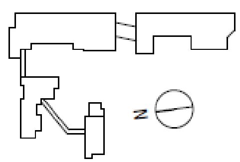
Bygg	OK kt grubeplate
A	+24,30
B	+23,03
C1	+25,93
C2	+23,94

PROSJEKTFASE: Arbeidstegning | FAG: RIB | TEGN.NR.: X-B-DT-XX-02 | REV.: E

TEGNINGSNAVN/TITTEL: Heisgruber

E	Spesifisert isolasjon. Endret vanntetting	19.06.20	DS	-
D	Korrigert isolasjon	20.05.20	DS	-
C	Korrigert heisgrubeplate	13.05.20	DS	-
B	Sulfatresistent betong.	04.05.20	DS	-
A	Arbeidstegning	24.04.20	DS	TGT
Rev	Beskr.	Dato	Tegn	Kontr

LOKALISERINGSSKISSE:



PROSJEKTNAVN:

OPPDRAGSGIVER:

PROSJEKTERENDE:

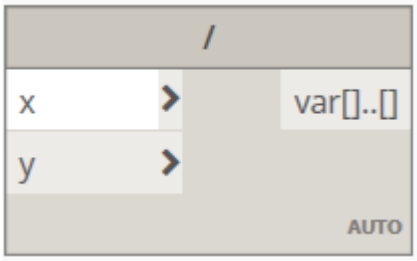
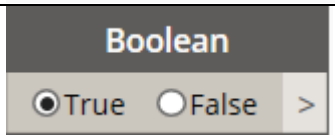
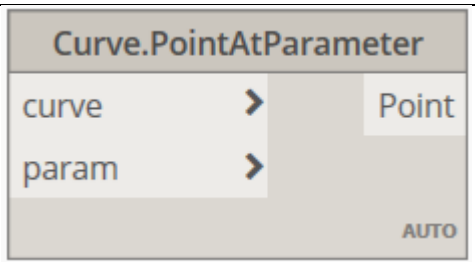
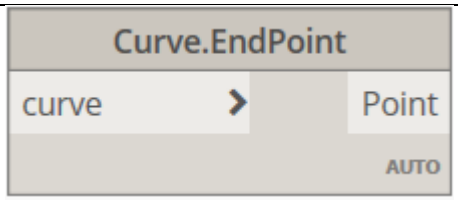
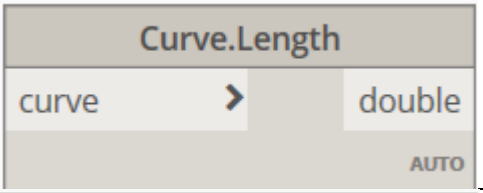
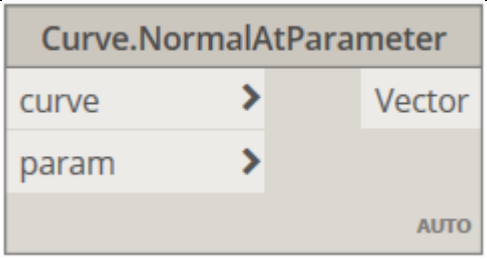
INGENIØRENE SEIM & HULTGREEN
RÅDGIVENDE INGENIØRER I BYGGETEKNIKK
PILESTREDET 75 C, N-0354 OSLO TELEFON 22 06 85 00 TELEFAX 22 06 85 01

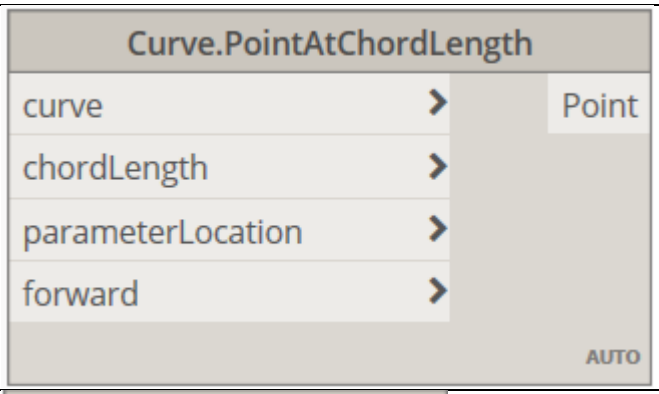
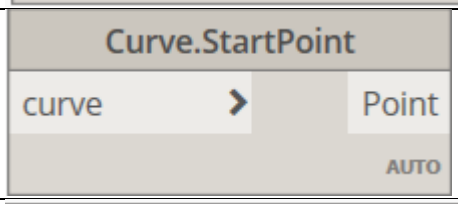
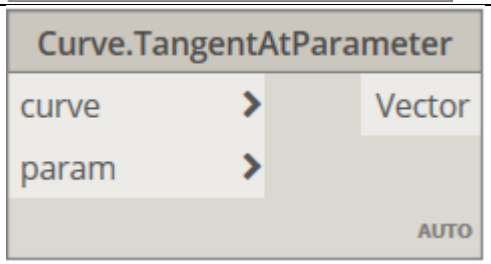
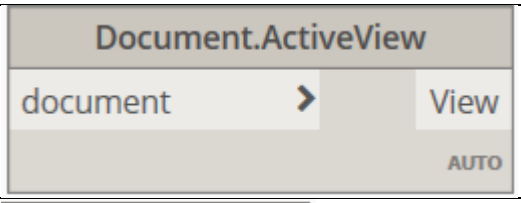
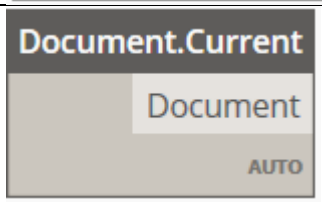
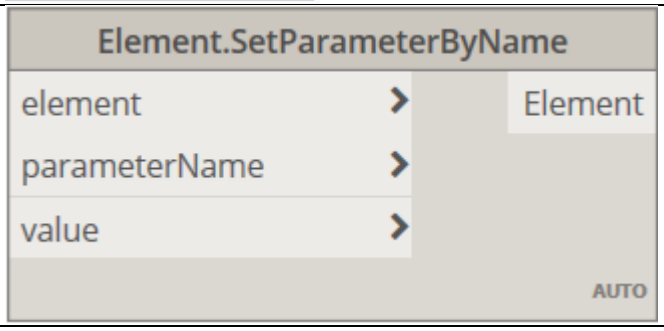
TEGNINGSNAVN / TITTEL:
Heisgruber
Form og armering
C

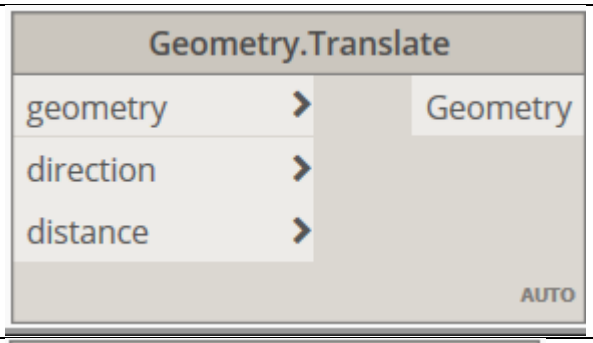
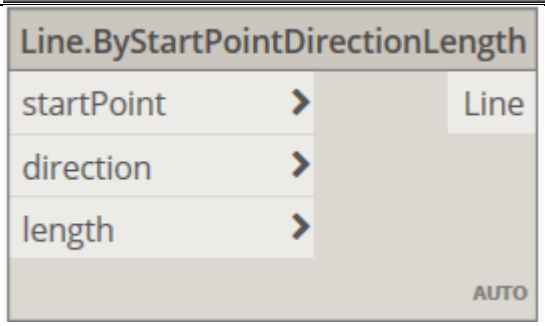
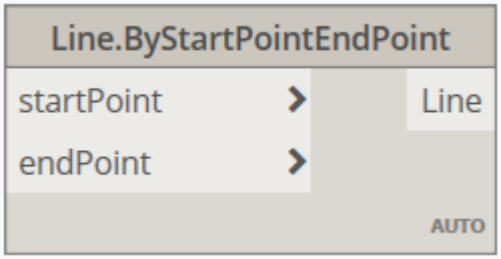
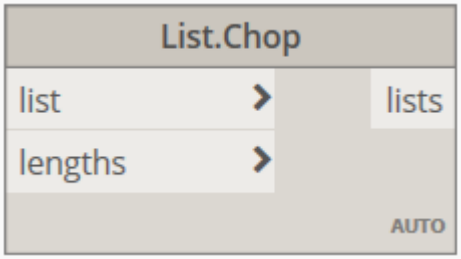
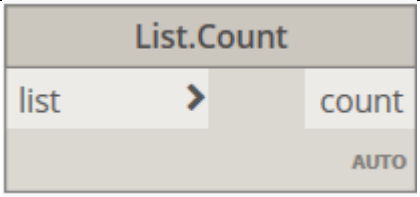
TEGNET AV: DS | KONTROLLERT AV: TGT | MÅLESTOKK: 1:20 | DATO: 24.04.20
PROSJEKTNR.: 18067 | TEGN.NR.: X-B-DT-XX-02 | REV.: E
PROSJEKTFASE: Arbeidstegning | FAG: RIB

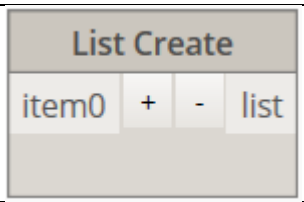
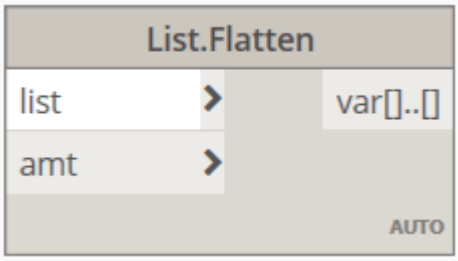
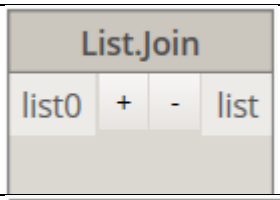
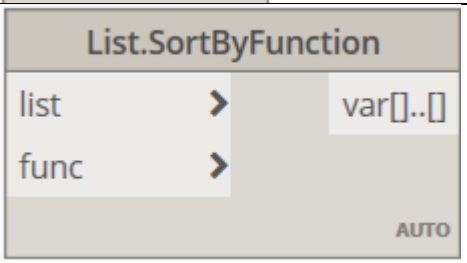
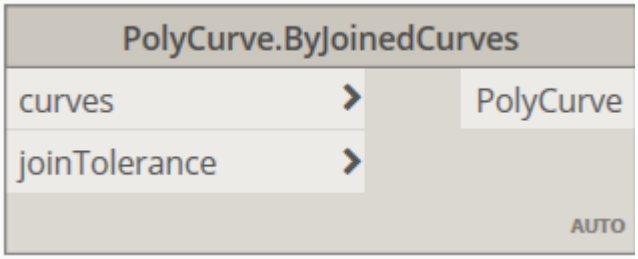
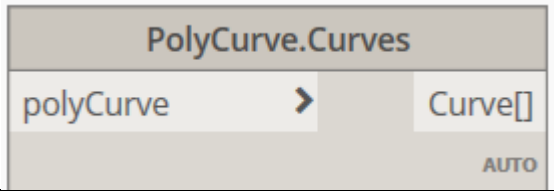
Vedlegg D – Nodeoversikt

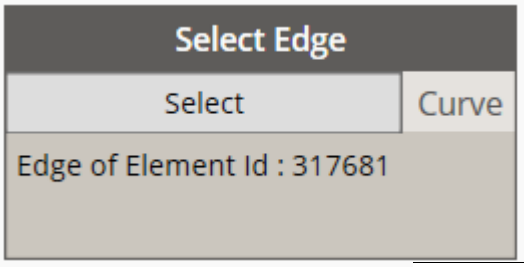
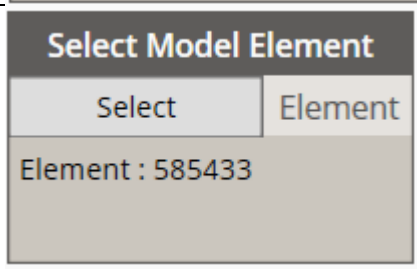
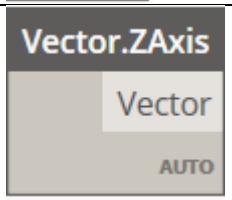
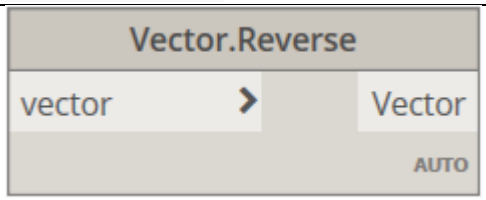
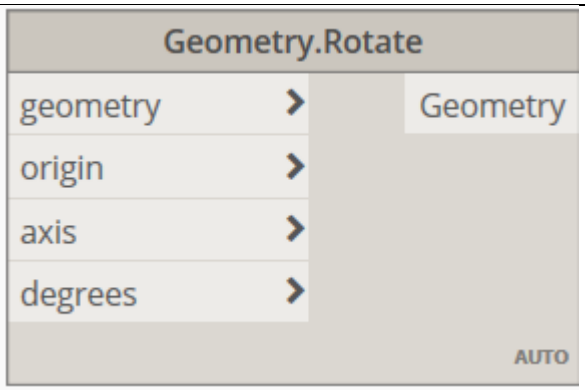
Oversikt over noder som brukes i skript (eks. Structural Design). Beksrivelser er gjengitt som de står i Dynamo og oversatt til norsk av oss.

	Node	Beskrivelse
1		*Divides x by y* Deler x på y
2		*Selection between a true and false* Velger mellom sann eller usann
3		*Get a Point on the Curve at a specified parameter between StartParameter() and EndParameter()* Gir et punkt gitt fra et parameter på en kurve, mellom start- og sluttparameteret
4		*Get the end Point along the Curve* Sluttpunktet fra en kurve
5		*The total arc length of the curve* Den totale lengden på en kurve.
6		*Get a Vector perpendicular to the curve at a specified parameter between StartParameter() and EndParameter()* Normalvektor mellom start- og sluttparameter

7		*Returns points spaced on the curve at given chord length starting from the given point* Returnerer punkter fordelt på kurven ved gitt akkordlengde fra det gitte punktet
8		*Get the start Point along the Curve* Få startpunktet langs kurven
9		*Get a Vector tangent to the curve at a specified parameter between StartParameter() and EndParameter()* Få en vektortangens til kurven ved en spesifisert parameter mellom StartParameter () og EndParameter ()
10		*Get the active view for the document* Få den aktive visningen for dokumentet
11		*Get the current document* Få det gjeldende dokumentet
12		*Set one of the element's parameters.* Sett en av elementets parametere.

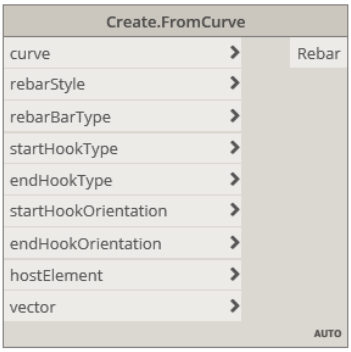
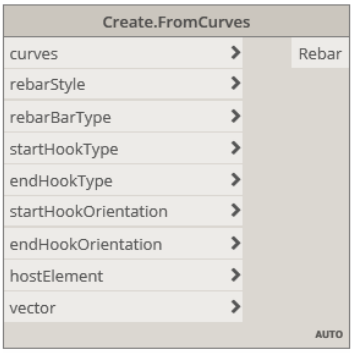
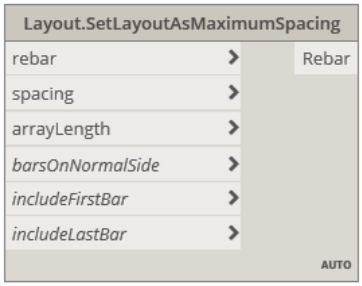
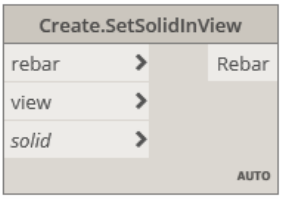
13		*Translates any geometry type by the given distance in the given direction.* Oversetter hvilken som helst geometritype med den angitte avstanden i den gitte retningen.
14		*Create a straight Line starting at start Point, extending in Vector direction by specified length.* Lag en rett linje som starter ved startpunktet, og strekker seg i vektorretning etter spesifisert lengde.
15		*Creates a straight Line between two input Points.* Oppretter en rett linje mellom to inngangspunkter.
16		*Chop a list into a set of consecutive sublists with the specified lengths. List division begins at the top of the list.* Hakk en liste i et sett med påfølgende underlister med de angitte lengdene. Listedeling begynner øverst på listen.
17		*Returns number of items in the specified list* Returnerer antall elementer i den angitte listen

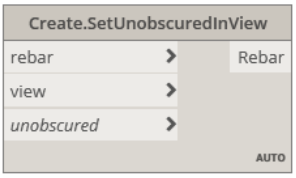
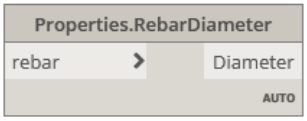
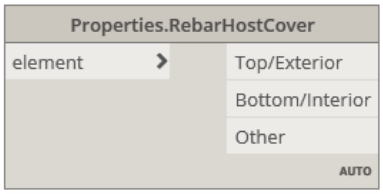
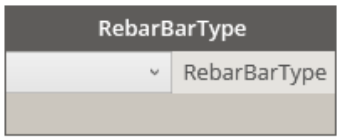
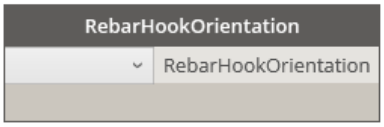
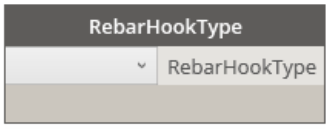
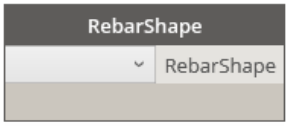
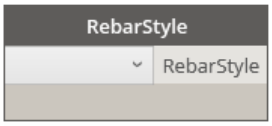
18		*Makes a new list out of the given inputs* Lager en ny liste ut av de gitte inngangene
19		*Returns the flattened 1D list of the multi-dimensional input list. If the input is a single value, returns that value.* Returnerer den flate 1D-listen til den flerdimensjonale inngangslisten. Hvis inngangen er en enkelt verdi, returnerer den verdien.
20		*Concatenates all given lists into a single list.* Sammenkobler alle gitte lister til en enkelt liste.
21		*Use a function to determine how list items should be sorted* Bruk en funksjon til å bestemme hvordan listeelementer skal sorteres
22		*Make PolyCurve by joining curves. Flips curve as needed for connectivity* Lag PolyCurve ved å samle kurver. Vender kurven etter behov for tilkobling
23		*Returns curves of the polycurve* Returnerer kurver for polykurven

24		*Select an edge.* Velg en kant.
25		*Select a model element from the document.* Velg et modellelement fra dokumentet.
26		*Creates a string.* Oppretter en streng.
27		*Get the Z component of a Vector* Få Z-komponenten i en vektor
28		*Get the reverse of the vector. Essentially this negates the X, Y, and Z components of the Vector.* Få baksiden av vektoren. I hovedsak negerer dette X-, Y- og Z-komponentene i Vector.
29		*Rotates an object around an origin and an axis by a specified degree* Roterer et objekt rundt en opprinnelse og en akse med en spesifisert grad

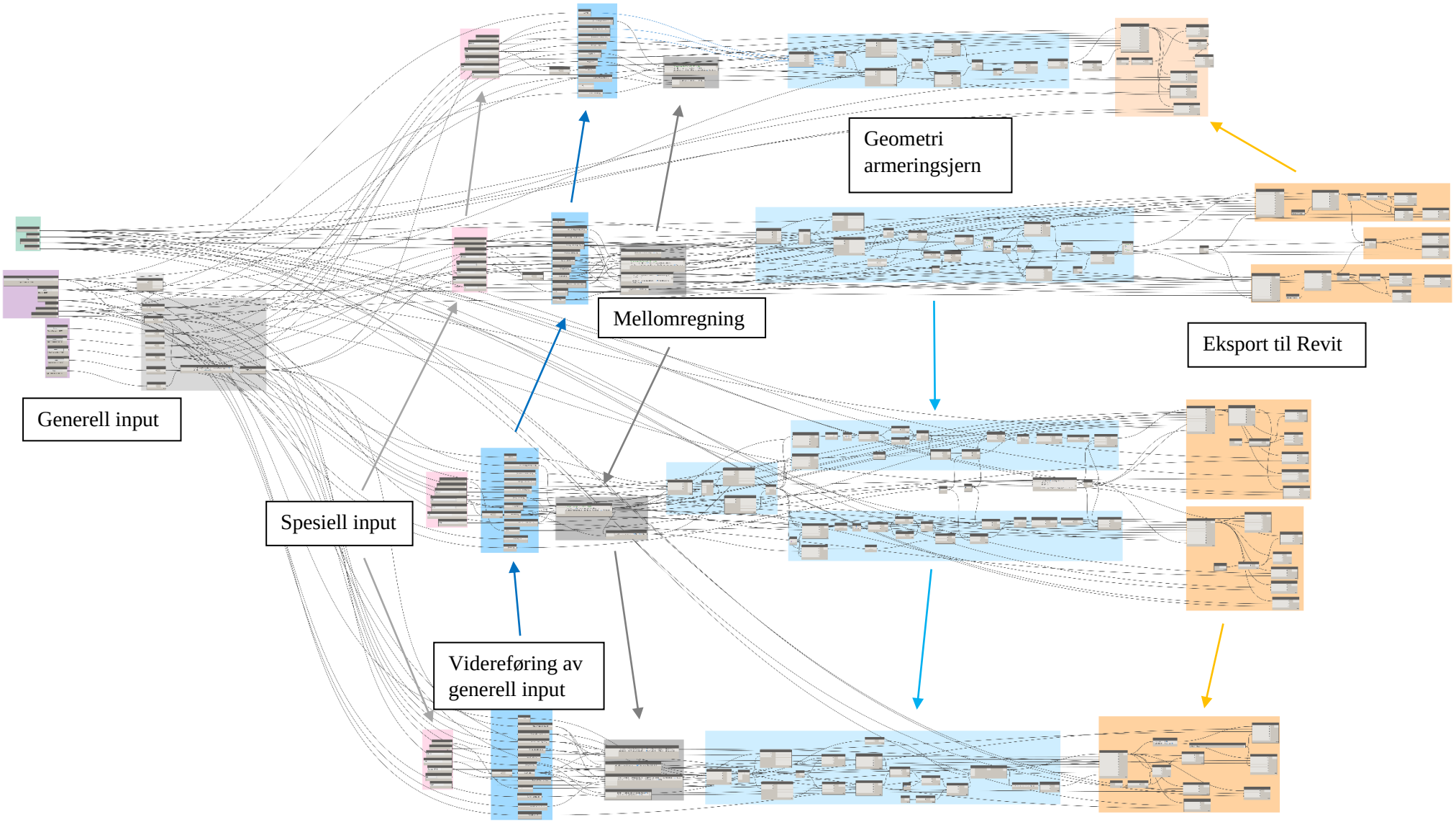
Vedlegg E – Nodeoversikt Structural Design

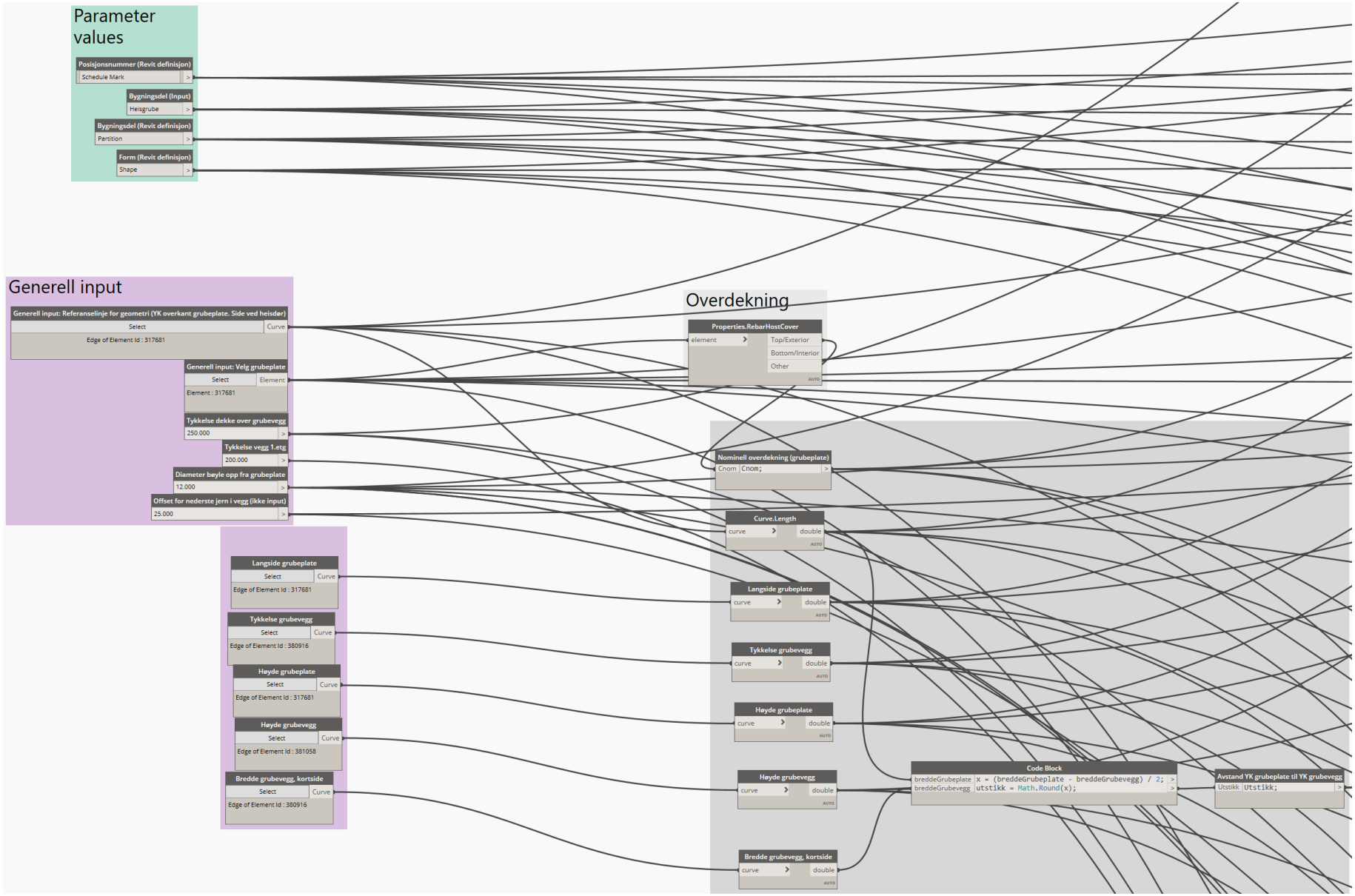
Oversikt over noder fra pakken Structural Desgin. Beksrivelser er gjengitt som de står i Dynamo og oversatt til norsk av oss.

	Node	Beskrivelse
1		*Creates a new instance of a shape driven rebar within the project.* Oppretter en ny forekomst av et formdrevet armeringsjern i prosjektet.
2		*Creates a new instance of a shape driven rebar within the project.* Oppretter en ny forekomst av et formdrevet armeringsjern i prosjektet.
3		*Sets the layout rule property of the rebar set to maximum spacing.* Setter armeringsjernets plasseringsregel til maks avstand
4		*Sets this rebar element to be shown solidly in a 3D view.* Angir at dette armeringsjernet skal vises solid i en 3D-visning.

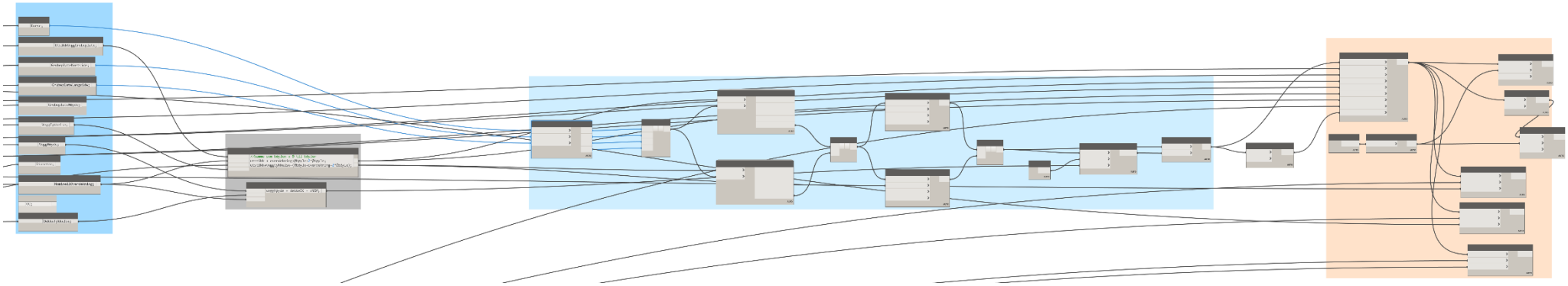
5		*Sets this rebar element to be shown unobscured in view.* Angir at dette armeringsjernet som skal vises uten synlighet.
6		*Gets rebar diameter* Henter armeringsjernets diameter
7		*Get the rebar cover dimensions for a selected element.* Henter overdekninger fra det valgte elementet
8		*Select rebar type* Velg armeringsjernstype
9		*Select rebar hook orientation* Velg armeringsjerns krokretning
10		*Select hook type* Velg armeringsjernets kroktype
11		*Select rebar shape* Velg armeringsjernets form
12		*Select rebar style* Velg armeringsjernet stil

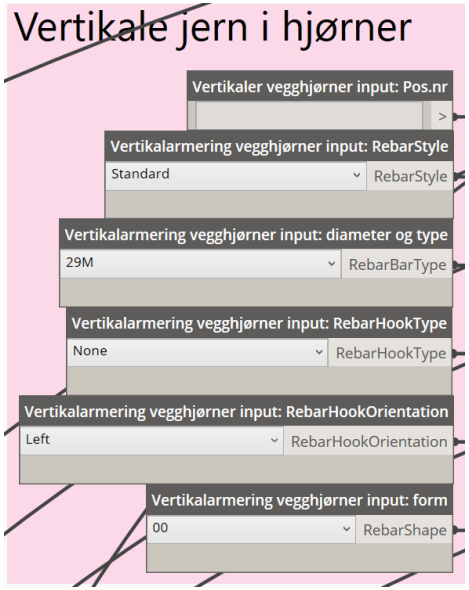
Vedlegg F – Dynamoskript for armering av grubevegg

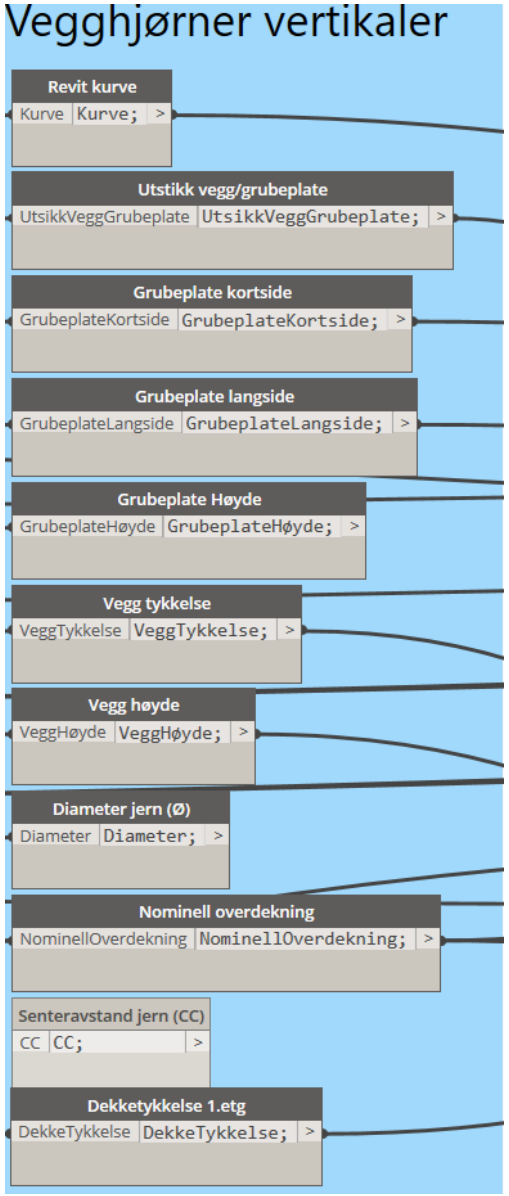


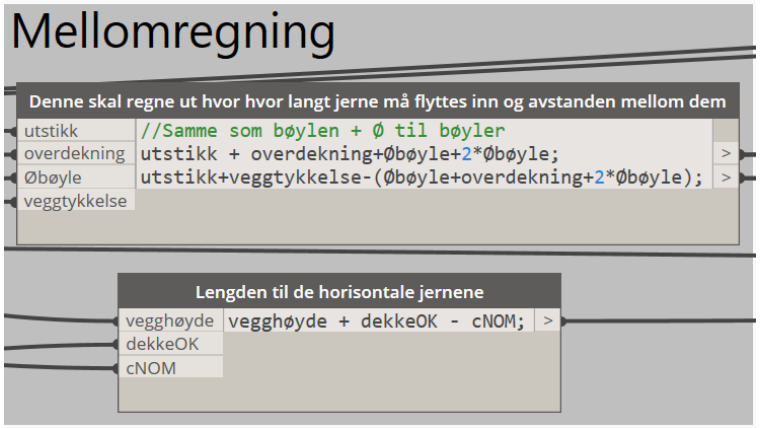

Parameter values
Posisjonsnummer (Revit definisjon)
Schedule Mark
Bygningsdel (Input)
Heisgrube
Bygningsdel (Revit definisjon)
Partition
Form (Revit definisjon)
Shape
Generell input
Generell input: Referanselinje for geometri (YK overkant grubeplate. Side ved heisdør)
Select
Curve
Edge of Element Id : 317681
Generell input: Velg grubeplate
Select
Element
Element : 317681
Tykkelse dekke over grubevegg
250.000
Tykkelse vegg 1.etg
200.000
Diameter bøyle opp fra grubeplate
12.000
Offset for nederste jern i vegg (ikke input)
25.000
Overdekning
Properties.RebarHostCover
element
Top/Exterior
Bottom/Interior
Other
AUTO
Nominell overdekning (grubeplate)
Cnom
Cnom;
Curve.Length
curve
double
AUTO
Langside grubeplate
Select
Curve
Edge of Element Id : 317681
Tykkelse grubevegg
Select
Curve
Edge of Element Id : 380916
Høyde grubeplate
Select
Curve
Edge of Element Id : 317681
Høyde grubevegg
Select
Curve
Edge of Element Id : 381058
Bredde grubevegg, kortside
Select
Curve
Edge of Element Id : 380916
Langside grubeplate
curve
double
AUTO
Tykkelse grubevegg
curve
double
AUTO
Høyde grubeplate
curve
double
AUTO
Høyde grubevegg
curve
double
AUTO
Bredde grubevegg, kortside
curve
double
AUTO
Code Block
breddeGrubeplate
breddeGrubevegg
x = (breddeGrubeplate - breddeGrubevegg) / 2;
utstikk = Math.Round(x);
Avstand YK grubeplate til YK grubevegg
Utstikk
Utstikk;

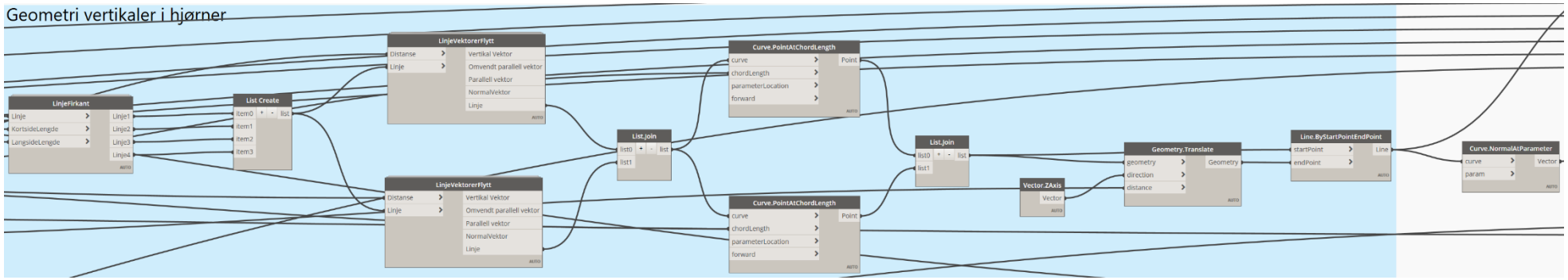
Vertikaler i vegghjørner

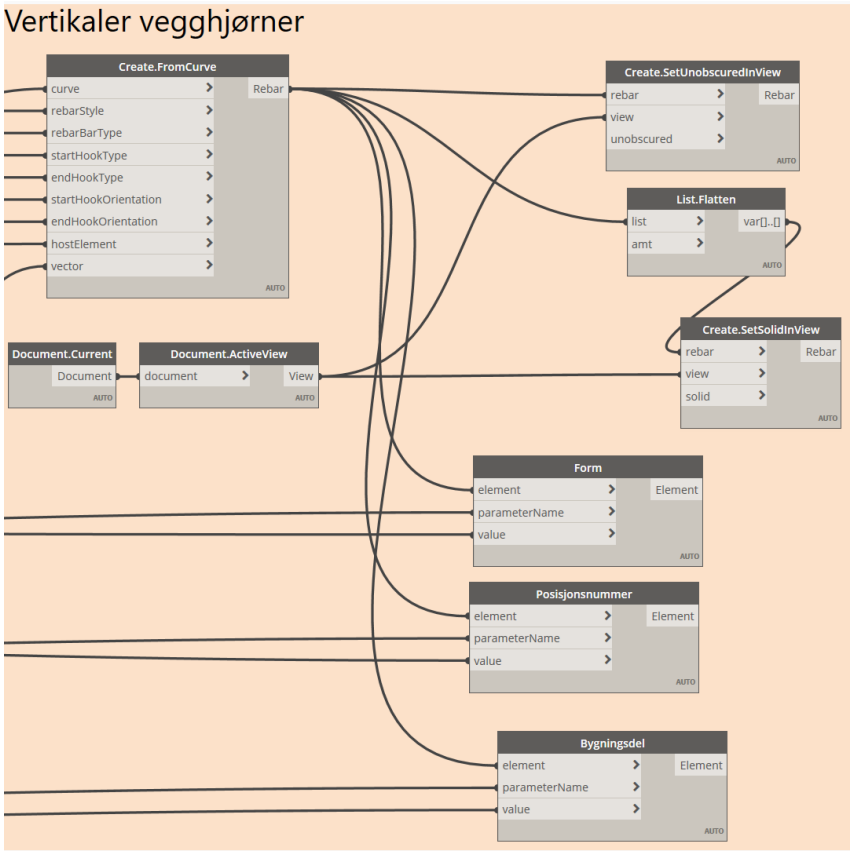



Vertikale jern i hjørner
Vertikaler vegghjørner input: Pos.nr
Vertikalarmering vegghjørner input: RebarStyle
Standard
RebarStyle
Vertikalarmering vegghjørner input: diameter og type
29M
RebarBarType
Vertikalarmering vegghjørner input: RebarHookType
None
RebarHookType
Vertikalarmering vegghjørner input: RebarHookOrientation
Left
RebarHookOrientation
Vertikalarmering vegghjørner input: form
00
RebarShape

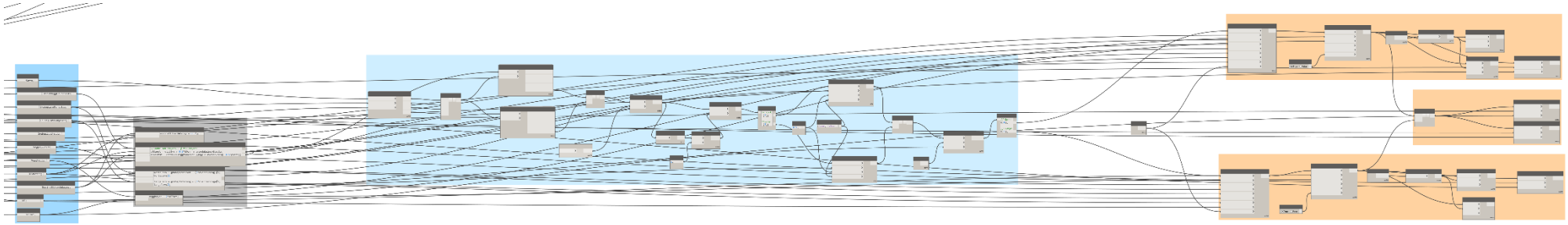

Vegghjørner vertikaler
Revit kurve
Kurve
Kurve;
Utstikk vegg/grubeplate
UtsikkVeggGrubeplate
UtsikkVeggGrubeplate;
Grubeplate kortside
GrubeplateKortside
GrubeplateKortside;
Grubeplate langside
GrubeplateLangside
GrubeplateLangside;
Grubeplate Høyde
GrubeplateHøyde
GrubeplateHøyde;
Vegg tykkelse
VeggTykkelse
VeggTykkelse;
Vegg høyde
VeggHøyde
VeggHøyde;
Diameter jern (Ø)
Diameter
Diameter;
Nominell overdekning
NominellOverdekning
NominellOverdekning;
Senteravstand jern (CC)
CC
CC;
Dekketykkelse 1.etg
DekkeTykkelse
DekkeTykkelse;

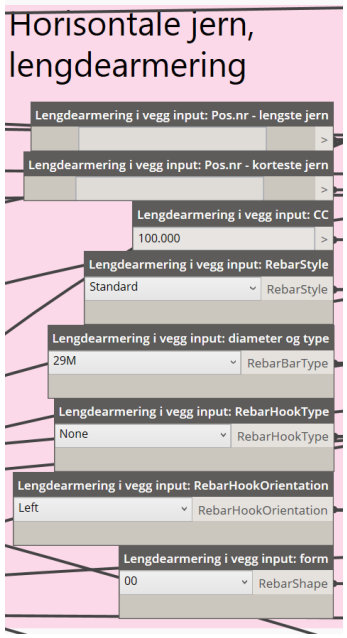

Mellomregning
Denne skal regne ut hvor hvor langt jerne må flyttes inn og avstanden mellom dem
utstikk
overdekning
Øbøyle
veggtykkelse
//Samme som bøylen + Ø til bøyler
utstikk + overdekning+Øbøyle+2*Øbøyle;
utstikk+veggtykkelse-(Øbøyle+overdekning+2*Øbøyle);
Lengden til de horisontale jernene
vegghøyde
dekkeOK
cNOM
vegghøyde + dekkeOK - cNOM;

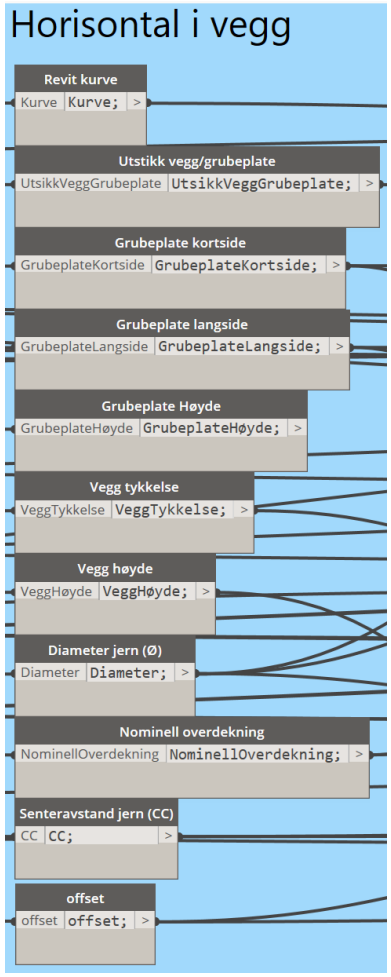

Geometri vertikaler i hjørner
LinjeFirkant
Linje
KortsideLengde
LangsideLengde
Linje1
Linje2
Linje3
Linje4
List Create
item0
item1
item2
item3
list
LinjeVektorerFlytt
Distanse
Linje
Vertikal Vektor
Omvendt parallell vektor
Parallell vektor
NormalVektor
Linje
List.Join
list0
list1
list
Curve.PointAtChordLength
curve
chordLength
parameterLocation
forward
Point
Vector.ZAxis
Vector
Geometry.Translate
geometry
direction
distance
Geometry
Line.ByStartPointEndPoint
startPoint
endPoint
Line
Curve.NormalAtParameter
curve
param
Vector

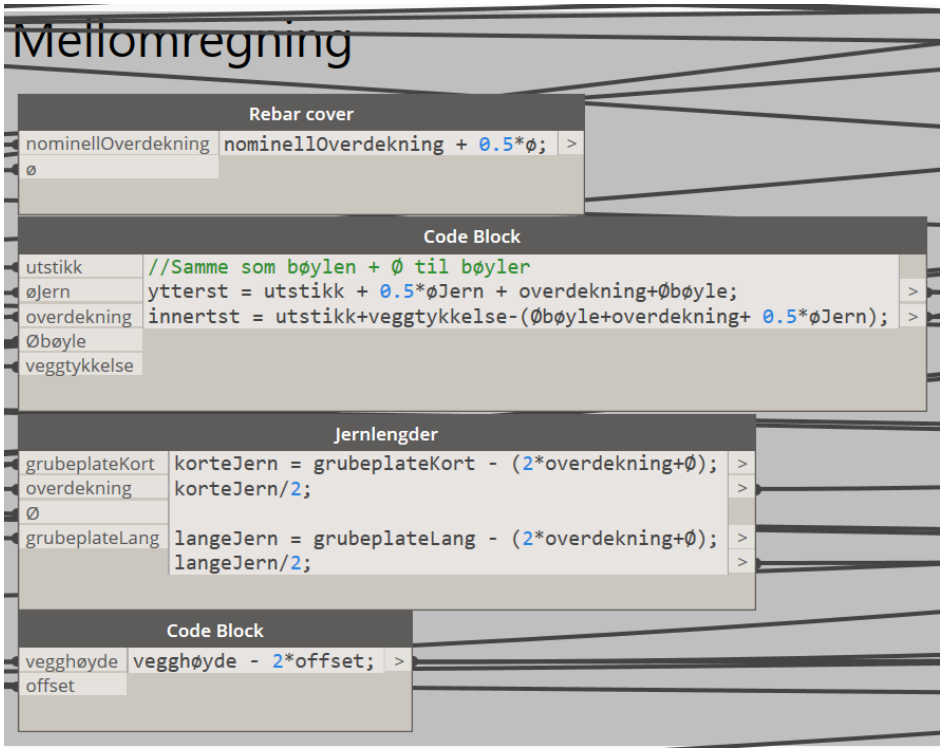

Vertikaler vegghjørner
Create.FromCurve
curve
rebarStyle
rebarBarType
startHookType
endHookType
startHookOrientation
endHookOrientation
hostElement
vector
Rebar
Create.SetUnobscuredInView
rebar
view
unobscured
Rebar
List.Flatten
list
amt
var[]..[]
Document.Current
Document
Document.ActiveView
document
View
Create.SetSolidInView
rebar
view
solid
Rebar
Form
element
parameterName
value
Element
Posisjonsnummer
element
parameterName
value
Element
Bygningsdel
element
parameterName
value
Element

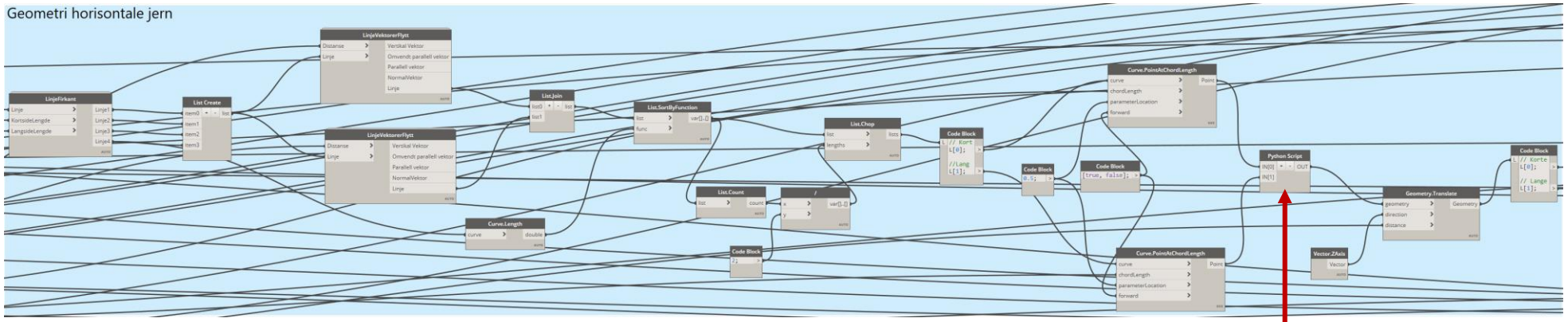
Horisontale armeringsjern




Horisontale jern, lengdearmering
Lengdearmering i vegg input: Pos.nr - lengste jern
Lengdearmering i vegg input: Pos.nr - korteste jern
Lengdearmering i vegg input: CC
100.000
Lengdearmering i vegg input: RebarStyle
Standard
RebarStyle
Lengdearmering i vegg input: diameter og type
29M
RebarBarType
Lengdearmering i vegg input: RebarHookType
None
RebarHookType
Lengdearmering i vegg input: RebarHookOrientation
Left
RebarHookOrientation
Lengdearmering i vegg input: form
00
RebarShape


Horisontal i vegg
Revit kurve
Kurve
Kurve;
Utstikk vegg/grubeplate
UtsikkVeggGrubeplate
UtsikkVeggGrubeplate;
Grubeplate kortside
GrubeplateKortside
GrubeplateKortside;
Grubeplate langside
GrubeplateLangside
GrubeplateLangside;
Grubeplate Høyde
GrubeplateHøyde
GrubeplateHøyde;
Vegg tykkelse
VeggTykkelse
VeggTykkelse;
Vegg høyde
VeggHøyde
VeggHøyde;
Diameter jern (Ø)
Diameter
Diameter;
Nominell overdekning
NominellOverdekning
NominellOverdekning;
Senteravstand jern (CC)
CC
CC;
offset
offset
offset;

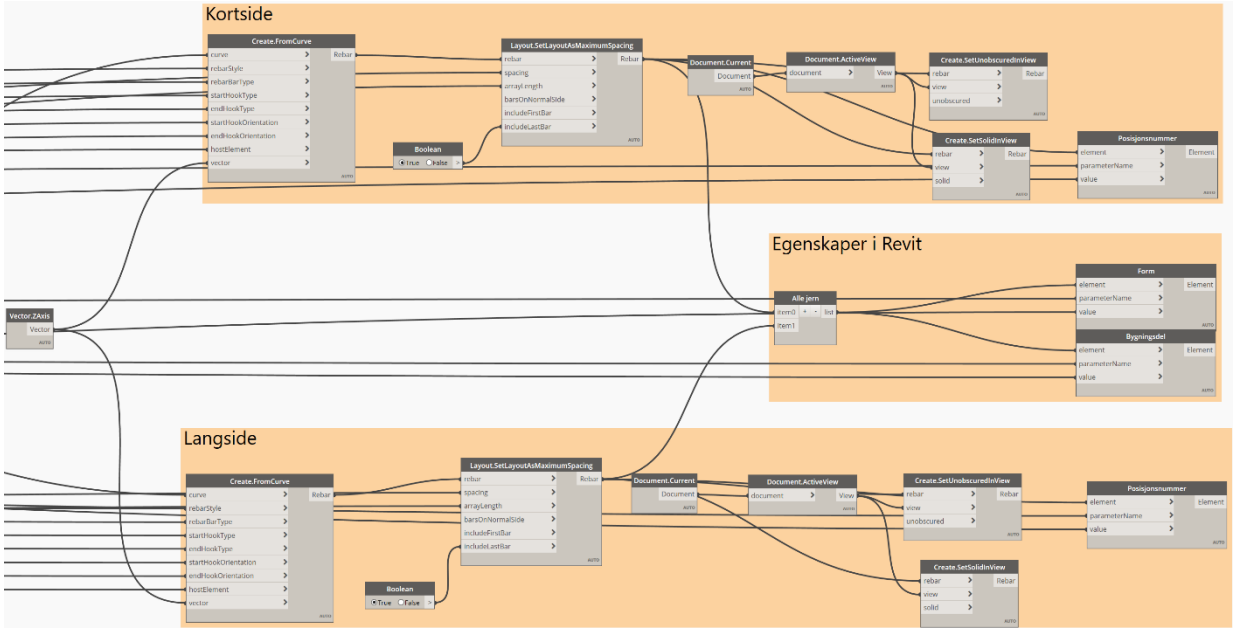

Mellomregning
Rebar cover
nominellOverdekning
nominellOverdekning + 0.5*ø;
ø
Code Block
utstikk
øJern
overdekning
Øbøyle
veggtykkelse
//Samme som bøylen + Ø til bøyler
ytterst = utstikk + 0.5*øJern + overdekning+Øbøyle;
innertst = utstikk+veggtykkelse-(Øbøyle+overdekning+ 0.5*øJern);
Jernlengder
grubeplateKort
overdekning
Ø
grubeplateLang
korteJern = grubeplateKort - (2*overdekning+Ø);
korteJern/2;
langeJern = grubeplateLang - (2*overdekning+Ø);
langeJern/2;
Code Block
vegghøyde
offset
vegghøyde - 2*offset;



Python Script

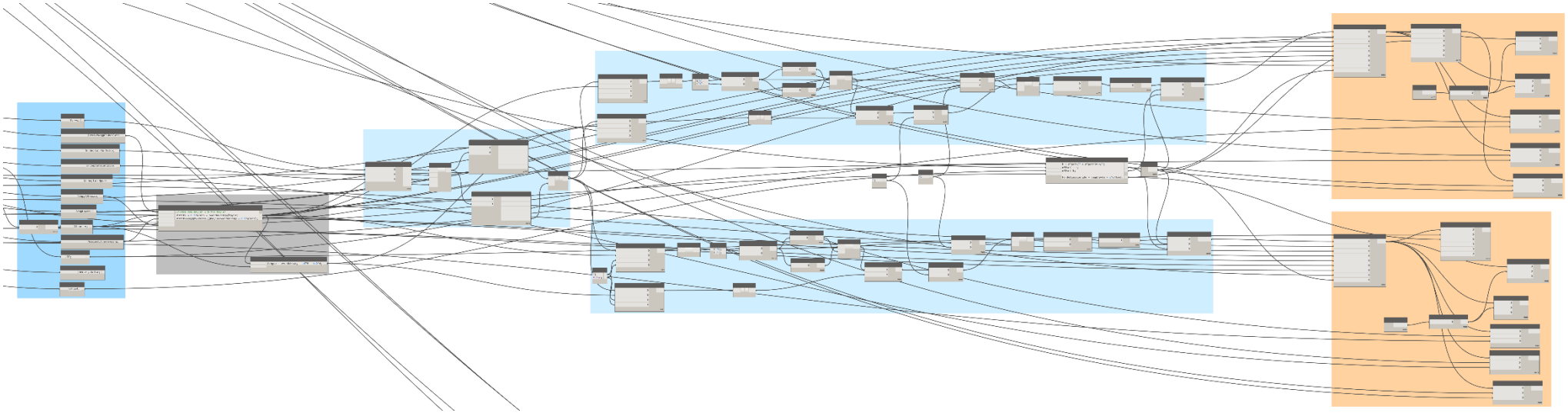
```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
kort = IN[0]
lang = IN[1]

flat_kort = []
flat_lang = []

for sublist in kort:
    for item in sublist:
        flat_kort.append(item)

for sublist in lang:
    for item in sublist:
        flat_lang.append(item)


flat_kort.sort()
flat_lang.sort()

start_kort = []
slutt_kort = []

start_lang = []
slutt_lang = []

# Place your code below this line
start_kort = flat_kort[0::2]
slutt_kort = flat_kort[1::2]

kortejern = []
for k,x in zip(start_kort, slutt_kort):
    line = Line.ByStartPointEndPoint(k, x)
    kortejern.append(line)

start_lang = flat_lang[0::2]
slutt_lang = flat_lang[1::2]

langejern = []
for a,b in zip(start_lang, slutt_lang):
    line = Line.ByStartPointEndPoint(a, b)
    langejern.append(line)

# Assign your output to the OUT variable.
OUT = [kortejern, langejern]
```

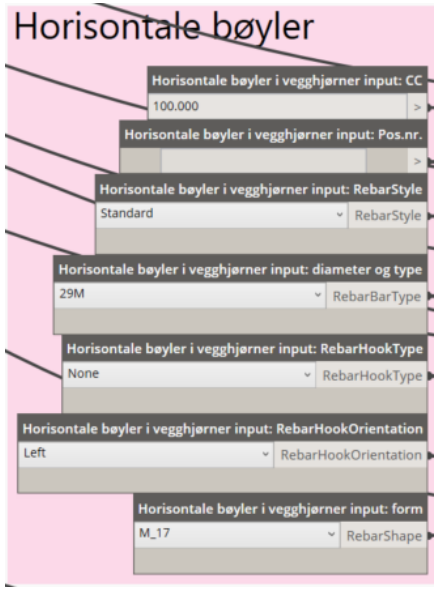
Run | Save Changes | Revert

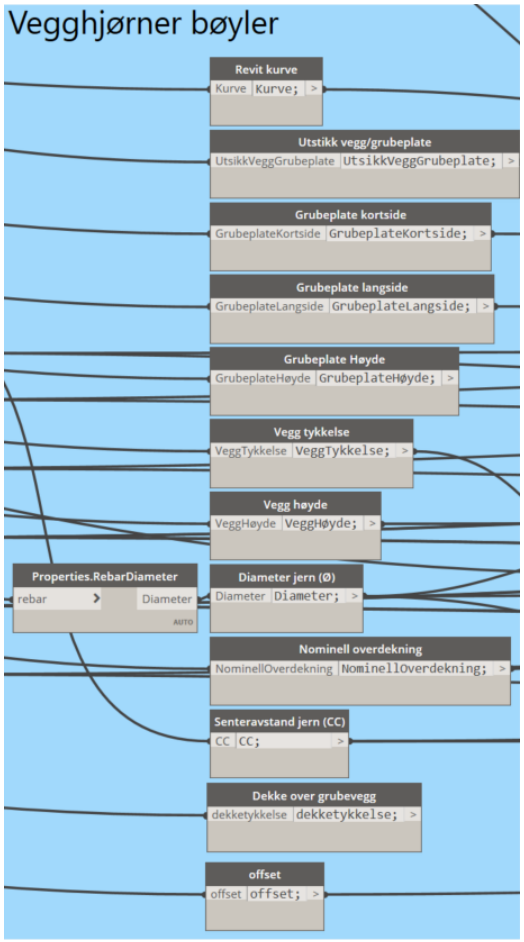


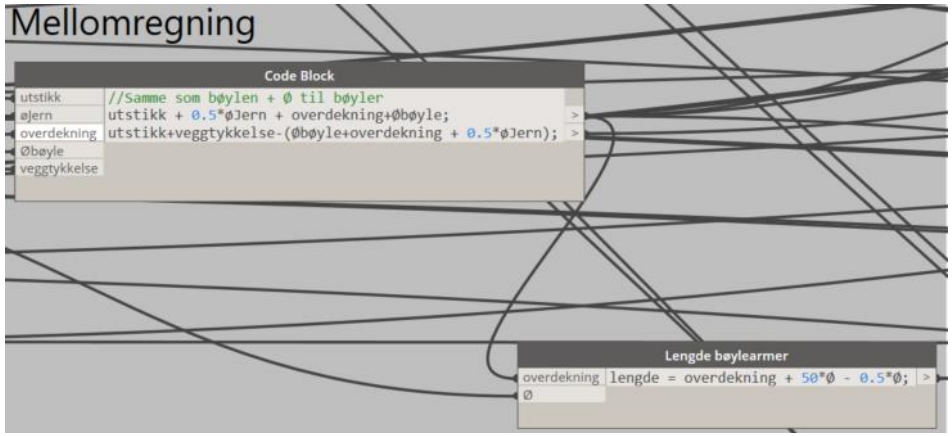
F

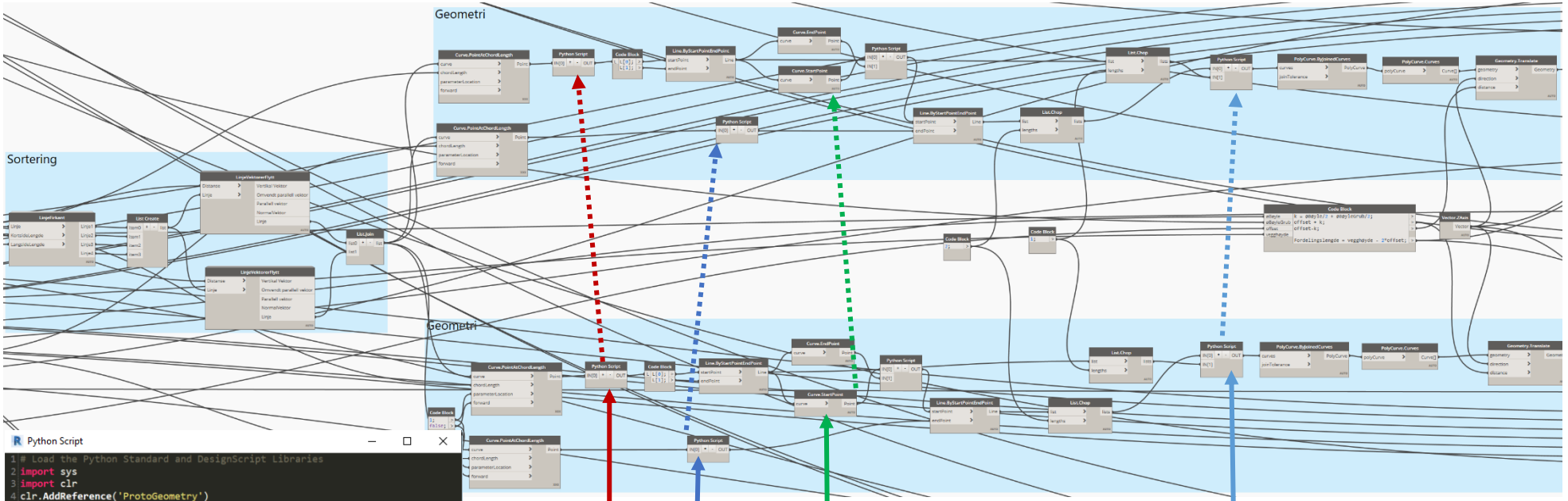
Bøyler i vegghjørner




Horisontale bøyler
Horisontale bøyler i vegghjørner input: CC
100.000
Horisontale bøyler i vegghjørner input: Pos.nr.
Horisontale bøyler i vegghjørner input: RebarStyle
Standard
RebarStyle
Horisontale bøyler i vegghjørner input: diameter og type
29M
RebarBarType
Horisontale bøyler i vegghjørner input: RebarHookType
None
RebarHookType
Horisontale bøyler i vegghjørner input: RebarHookOrientation
Left
RebarHookOrientation
Horisontale bøyler i vegghjørner input: form
M_17
RebarShape


Vegghjørner bøyler
Revit kurve
Kurve
Kurve;
Utstikk vegg/grubeplate
UtsikkVeggGrubeplate
UtsikkVeggGrubeplate;
Grubeplate kortside
GrubeplateKortside
GrubeplateKortside;
Grubeplate langside
GrubeplateLangside
GrubeplateLangside;
Grubeplate Høyde
GrubeplateHøyde
GrubeplateHøyde;
Vegg tykkelse
VeggTykkelse
VeggTykkelse;
Vegg høyde
VeggHøyde
VeggHøyde;
Properties.RebarDiameter
rebar
Diameter
AUTO
Diameter jern (Ø)
Diameter
Diameter;
Nominell overdekning
NominellOverdekning
NominellOverdekning;
Senteravstand jern (CC)
CC
CC;
Dekke over grubevegg
dekketykkelse
dekketykkelse;
offset
offset
offset;


Mellomregning
Code Block
utstikk
øjern
overdekning
Øbøyle
veggtykkelse
//Samme som bøylen + Ø til bøyler
utstikk + 0.5*øJern + overdekning+Øbøyle;
utstikk+veggtykkelse-(Øbøyle+overdekning + 0.5*øJern);
Lengde bøylearmer
overdekning
Ø
lengde = overdekning + 50*Ø - 0.5*Ø;



Python Script

```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
lines = IN[0]

# Place your code below this line


length = len(lines)
middle_index = length//2

start = lines[:middle_index]
slutt = lines[middle_index:]

# Assign your output to the OUT variable.

OUT = [start, slutt]
```

Run | Save Changes | Revert

Python Script

```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
punkter = IN[0]

# Place your code below this line
h1 = punkter[0::4]
h2 = punkter[1::4]
h3 = punkter[2::4]
h4 = punkter[3::4]

punkter = [h1,h2,h3,h4]
flat_list = []

for sublist in punkter:
    for item in sublist:
        flat_list.append(item)
# Assign your output to the OUT variable.
OUT = flat_list
```

Run | Save Changes | Revert

Python Script

```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
in1 = IN[0]
in2 = IN[1]

in1_flat = []
in2_flat = []
# Place your code below this line
l1 = [in1[0],in2[0]]
l2 = [in1[1],in2[1]]
l3 = [in1[2],in2[2]]
l4 = [in1[3],in2[3]]

lines = [l1,l2,l3,l4]
flat_list = []

for sublist in lines:
    for item in sublist:
        flat_list.append(item)

# Assign your output to the OUT variable.
OUT = flat_list
```

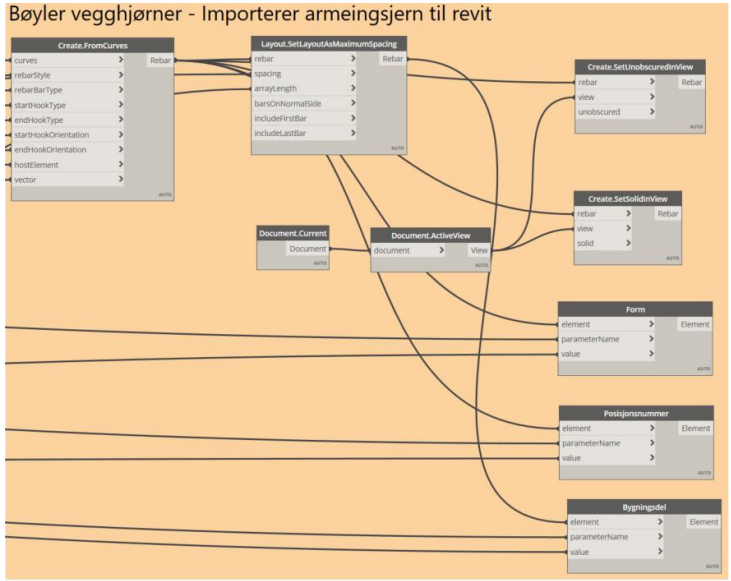
Run | Save Changes | Revert

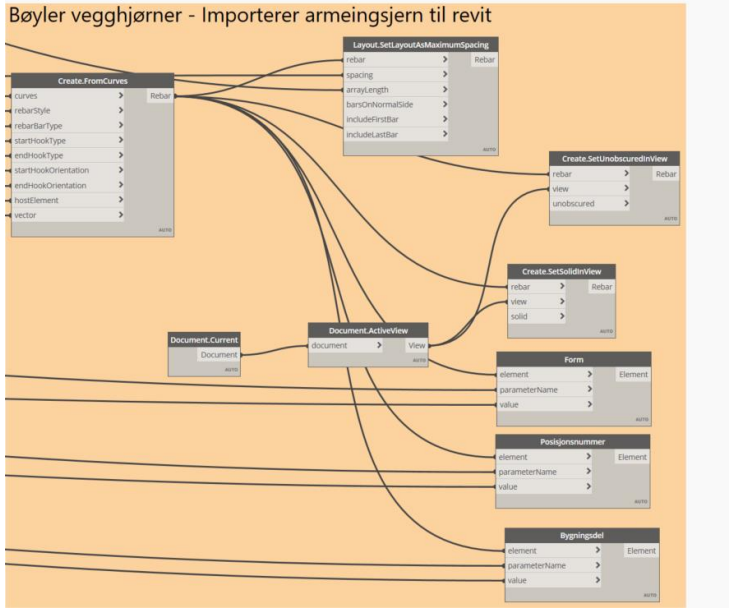
Python Script

```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
in1 = IN[0]
in2 = IN[1]

flat2 = []

for sublist in in2:
    for item in sublist:
        flat2.append(item)
# Place your code below this line
in1[0].append(flat2[0])
in1[1].append(flat2[1])
in1[2].append(flat2[2])
in1[3].append(flat2[3])

# Assign your output to the OUT variable.
OUT = in1
```

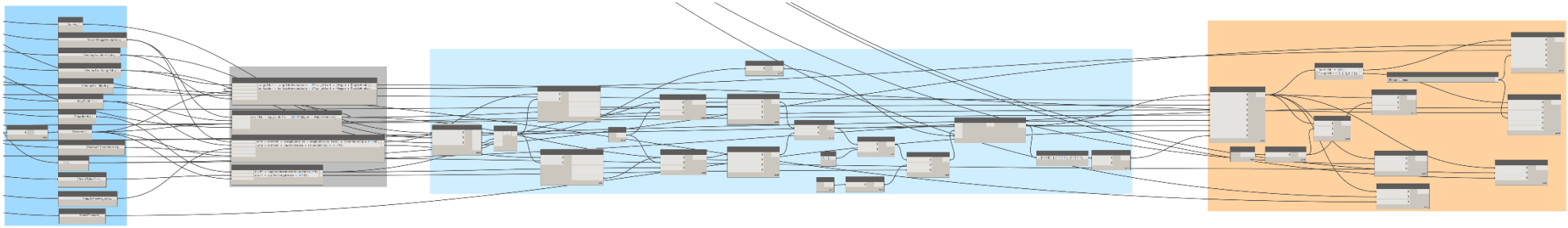
Run | Save Changes | Revert

F

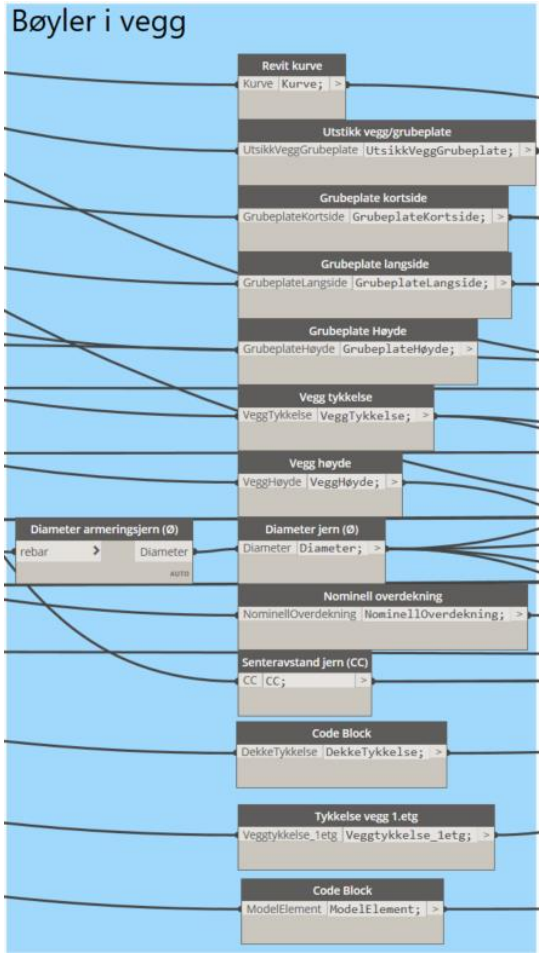

Bøyler vegghjørner - Importerer armeingsjern til revit
Create.FromCurves
curves
rebarStyle
rebarBarType
startHookType
endHookType
startHookOrientation
endHookOrientation
hostElement
vector
Rebar
Layout.SetLayoutAsMaximumSpacing
rebar
spacing
arrayLength
barsOnNormalSide
includeFirstBar
includeLastBar
Rebar
Create.SetUnobscuredInView
rebar
view
unobscured
Rebar
Create.SetSolidInView
rebar
view
solid
Rebar
Document.Current
Document
Document.ActiveView
document
View
Form
element
parameterName
value
Element
Posisjonsnummer
element
parameterName
value
Element
Bygningsdel
element
parameterName
value
Element

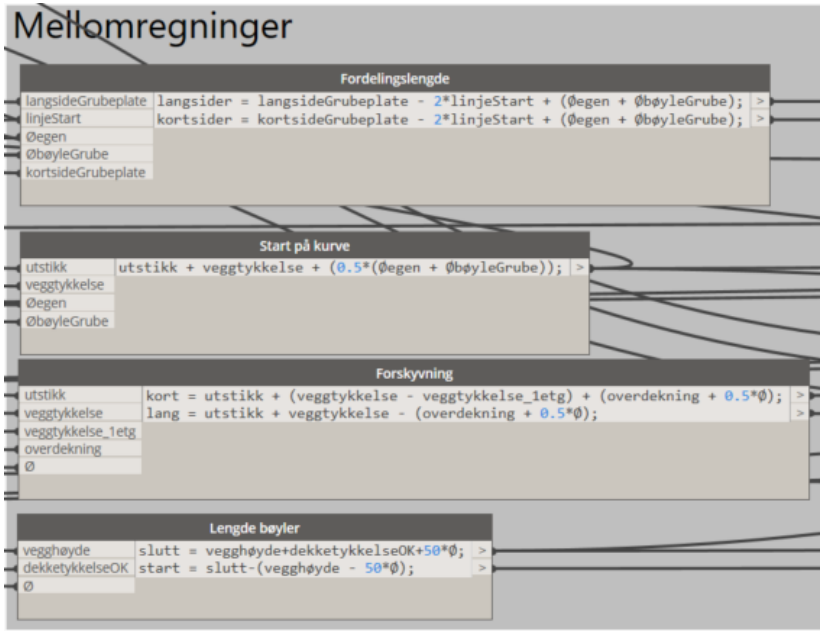

Bøyler vegghjørner - Importerer armeingsjern til revit
Layout.SetLayoutAsMaximumSpacing
rebar
spacing
arrayLength
barsOnNormalSide
includeFirstBar
includeLastBar
Rebar
Create.FromCurves
curves
rebarStyle
rebarBarType
startHookType
endHookType
startHookOrientation
endHookOrientation
hostElement
vector
Rebar
Create.SetUnobscuredInView
rebar
view
unobscured
Rebar
Create.SetSolidInView
rebar
view
solid
Rebar
Document.Current
Document
Document.ActiveView
document
View
Form
element
parameterName
value
Element
Posisjonsnummer
element
parameterName
value
Element
Bygningsdel
element
parameterName
value
Element

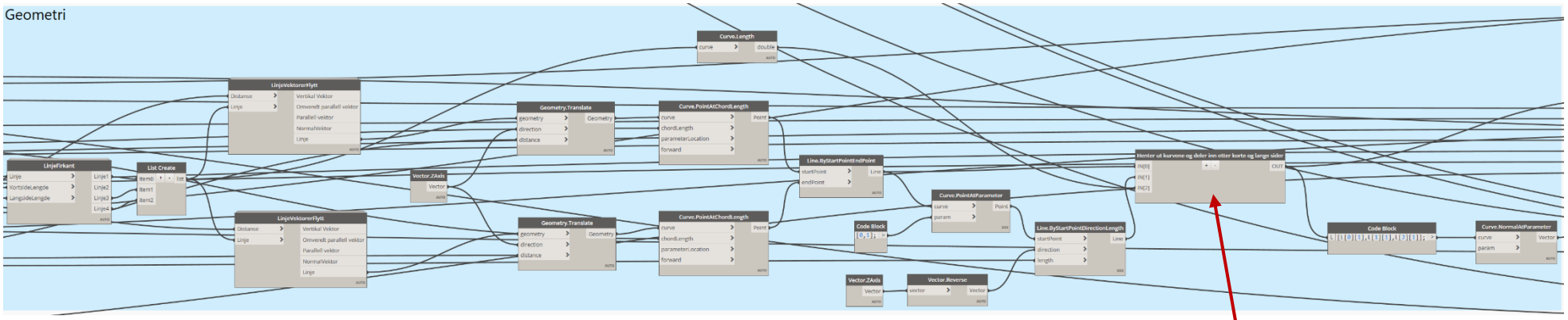
U-bøyler




Vertikale bøyler i vegg
Vertikale bøyler i vegg input: CC
100.000
Vertikale bøyler i vegg input: Pos.nr
Vertikale bøyler i vegg input: RebarStyle
Standard
RebarStyle
Vertikale bøyler i vegg input: type og kvalitet
29M
RebarBarType
Vertikale bøyler i vegg input: Kroktype
Standard - 180 deg.
RebarHookType
Vertikale bøyler i vegg input: Krokretning
Left
RebarHookOrientation
Vertikale bøyler i vegg input: form
M_17
RebarShape


Bøyler i vegg
Revit kurve
Kurve Kurve;
Utstikk vegg/grubeplate
UtsikkVeggGrubeplate UtsikkVeggGrubeplate;
Grubeplate kortside
GrubeplateKortside GrubeplateKortside;
Grubeplate langside
GrubeplateLangside GrubeplateLangside;
Grubeplate Høyde
GrubeplateHøyde GrubeplateHøyde;
Vegg tykkelse
VeggTykkelse VeggTykkelse;
Vegg høyde
VeggHøyde VeggHøyde;
Diameter armeringsjern (Ø)
rebar
Diameter
Diameter jern (Ø)
Diameter Diameter;
Nominell overdekning
NominellOverdekning NominellOverdekning;
Senteravstand jern (CC)
CC CC;
Code Block
DekkeTykkelse DekkeTykkelse;
Tykkelse vegg 1.etg
Veggtykkelse_1etg Veggtykkelse_1etg;
Code Block
ModelElement ModelElement;


Mellomregninger
Fordelingslengde
langsideGrubeplate
linjeStart
Øegen
ØbøyleGrube
kortsideGrubeplate
langsider = langsideGrubeplate - 2*linjeStart + (Øegen + ØbøyleGrube);
kortsider = kortsideGrubeplate - 2*linjeStart + (Øegen + ØbøyleGrube);
Start på kurve
utstikk
veggtykkelse
Øegen
ØbøyleGrube
utstikk + veggtykkelse + (0.5*(Øegen + ØbøyleGrube));
Forskyvning
utstikk
veggtykkelse
veggtykkelse_1etg
overdekning
Ø
kort = utstikk + (veggtykkelse - veggtykkelse_1etg) + (overdekning + 0.5*Ø);
lang = utstikk + veggtykkelse - (overdekning + 0.5*Ø);
Lengde bøyler
vegghøyde
dekketykkelseOK
Ø
slutt = vegghøyde+dekketykkelseOK+50*Ø;
start = slutt-(vegghøyde - 50*Ø);

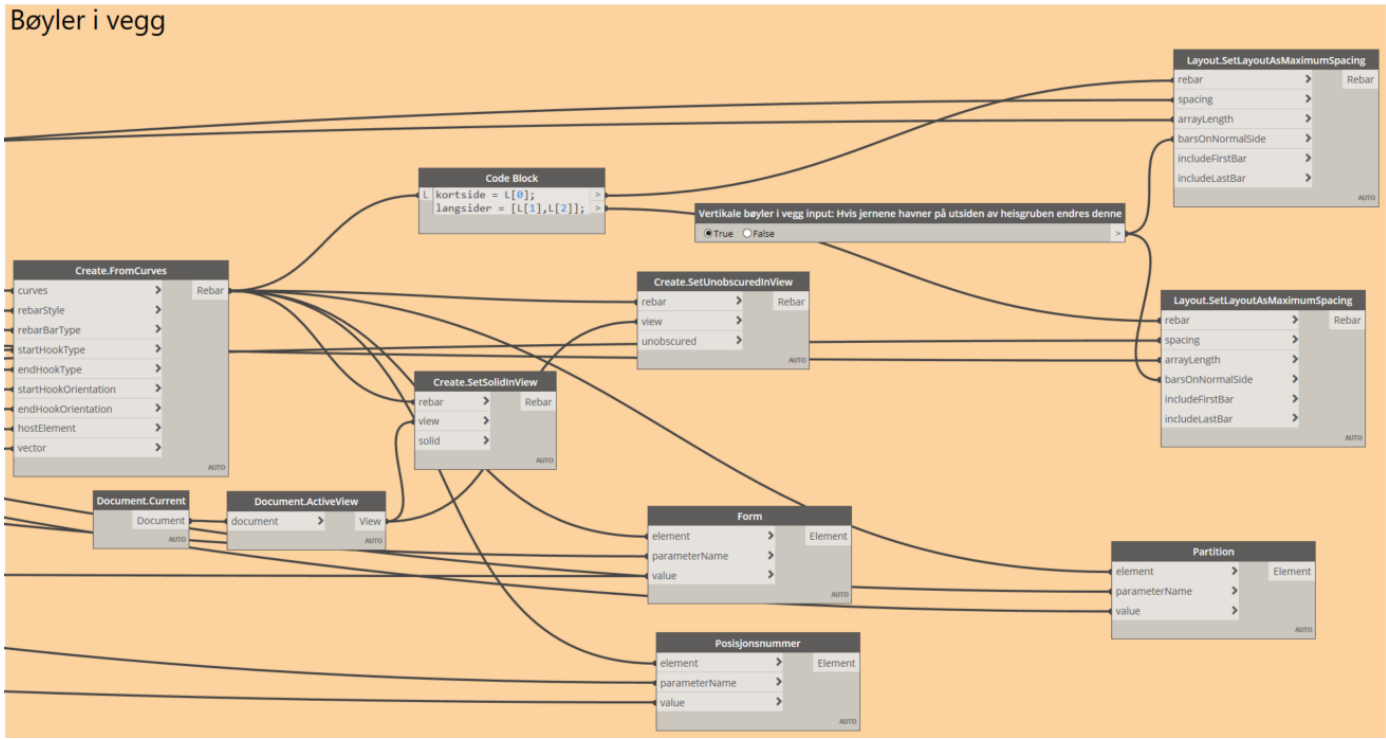


Henter ut kurvene og deler inn etter korte og lange sider

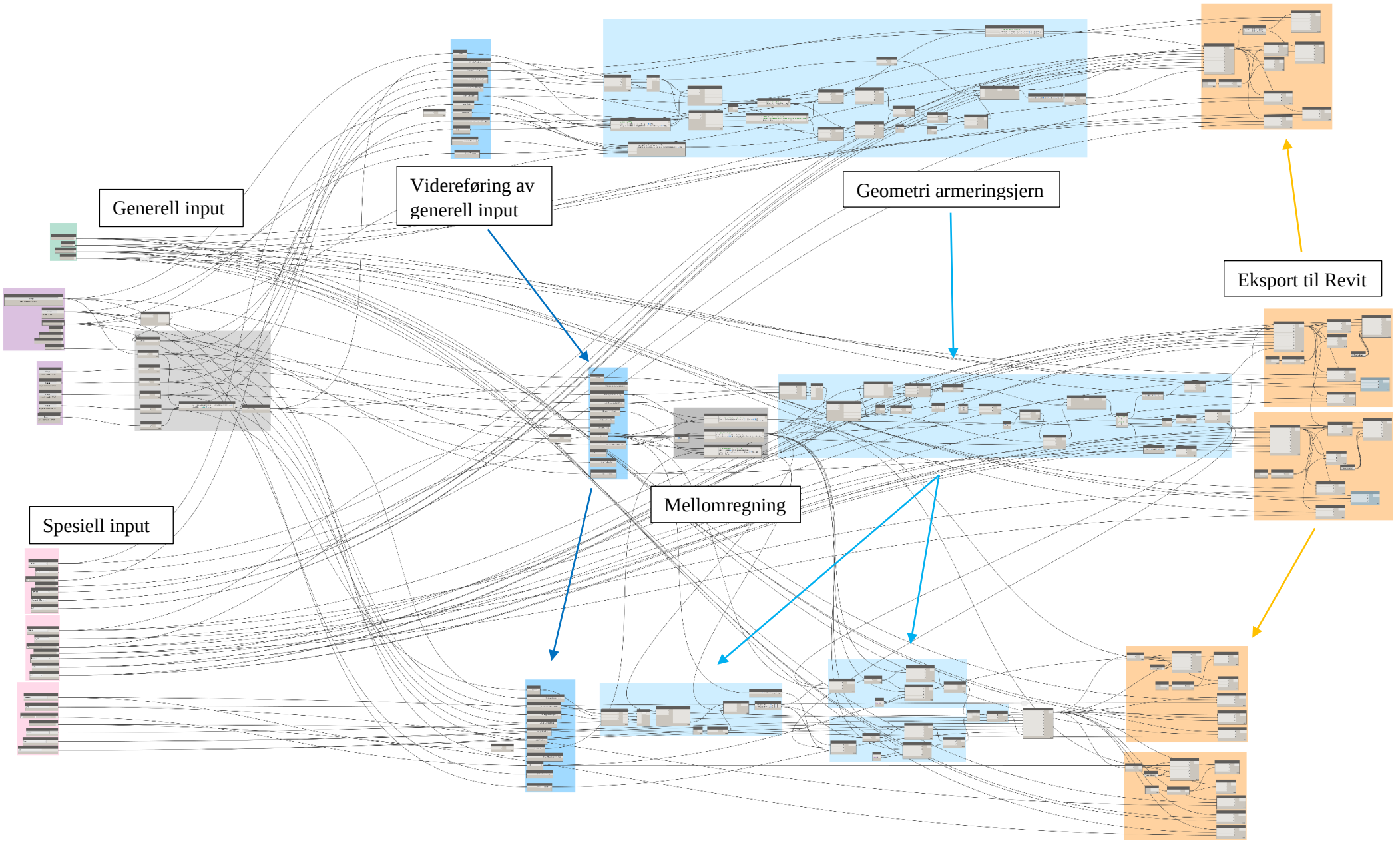
```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
korte = IN[0]
lange = IN[1]
lengder = IN[2]
# Place your code below this line
lange[0].append(korte[0])
lange[1].append(korte[1])
lange[2].append(korte[2])
#lange[3].append(korte[3])

polykurver = []

for i in lange:
    kurve = PolyCurve.ByJoinedCurves(i, 0.001)
    polykurver.append(kurve)

kurver = []
for x in polykurver:
    kurve = PolyCurve.Curves(x)
    kurver.append(kurve)

# sorterer etter kort og lange linjer på grubeplata
maximum = max(lengder)
minimum = min(lengder)

outKurver = []
korte = []
lange = []

for i in range(len(kurver)):
    if lengder[i] == maximum:
        outKurver.insert(-1, kurver[i])
    else:
        outKurver.insert(0,kurver[i])

# Assign your output to the OUT variable.
OUT = outKurver
```

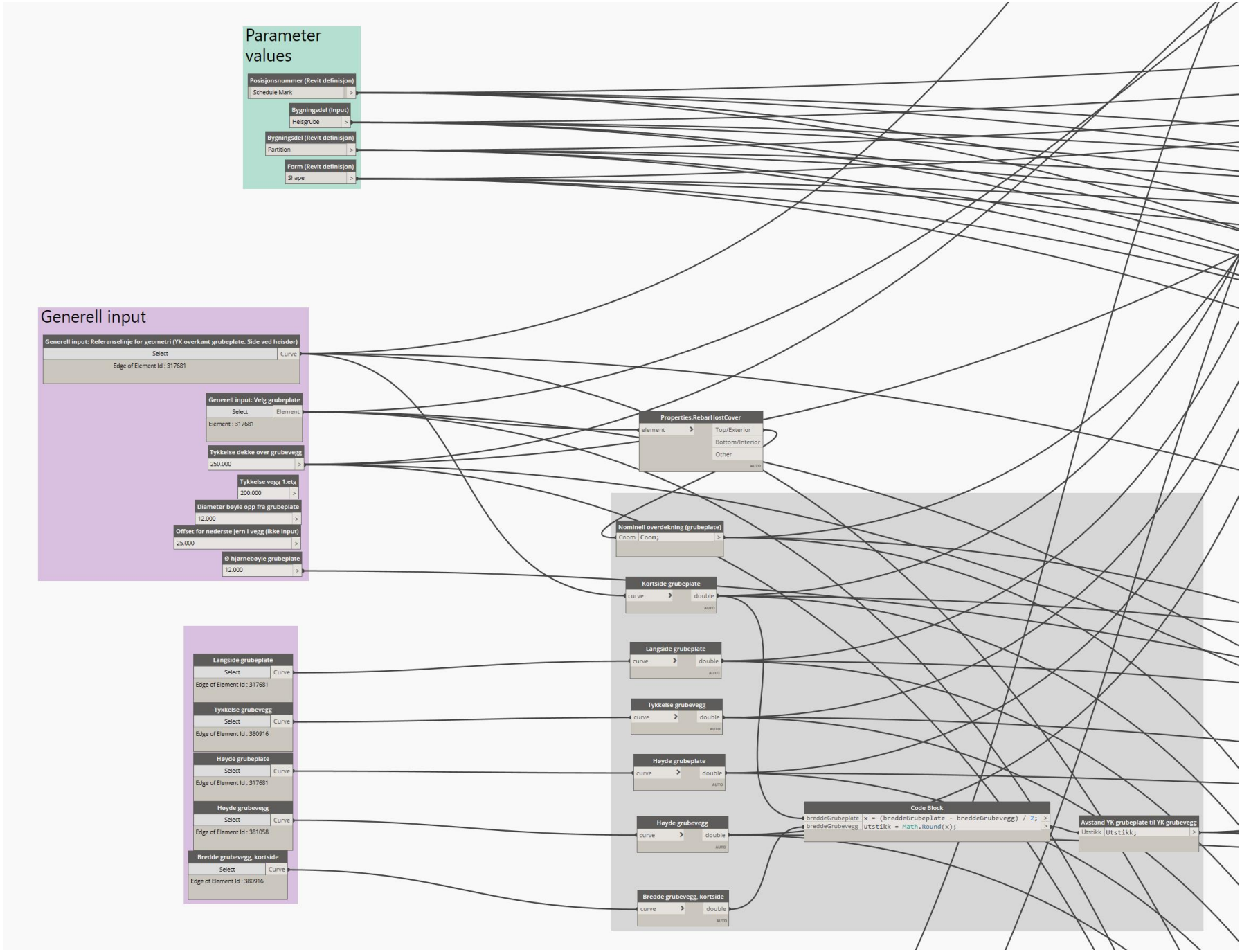
Run | Save Changes | Revert

F


Bøyler i vegg
Layout.SetLayoutAsMaximumSpacing
rebar
spacing
arrayLength
barsOnNormalSide
includeFirstBar
includeLastBar
Rebar
AUTO
Code Block
L kortside = L[0];
langsider = [L[1],L[2]];
Vertikale bøyler i vegg input: Hvis jernene havner på utsiden av heisgruben endres denne
True
False
Create.FromCurves
curves
rebarStyle
rebarBarType
startHookType
endHookType
startHookOrientation
endHookOrientation
hostElement
vector
Create.SetUnobscuredInView
view
unobscured
Create.SetSolidInView
solid
Document.Current
Document
Document.ActiveView
document
View
Form
element
parameterName
value
Element
Partition
Posisjonsnummer

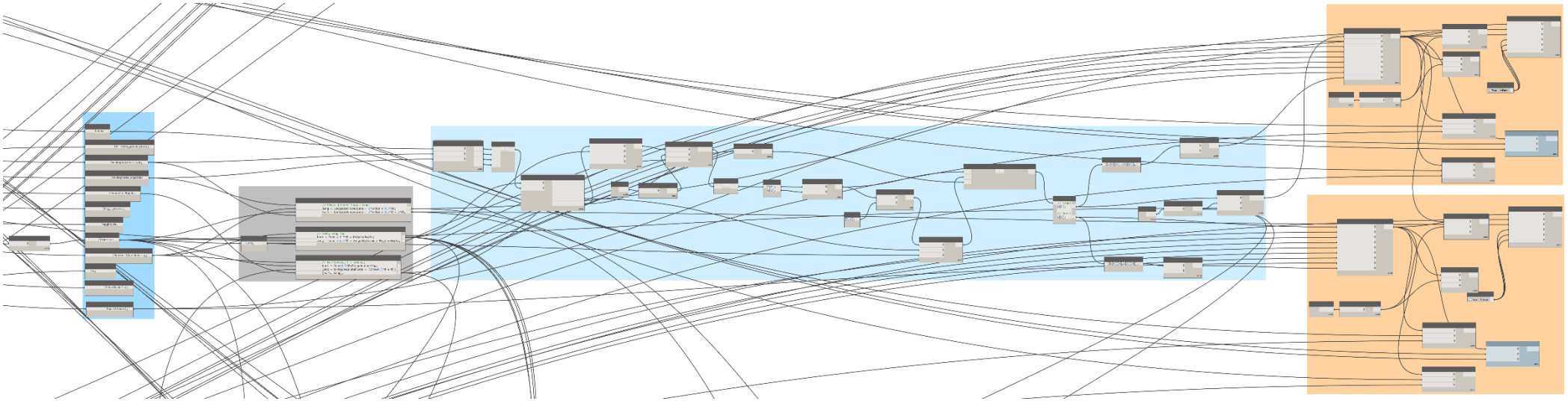
Vedlegg G – Dynamoskript for armering av grubeplate



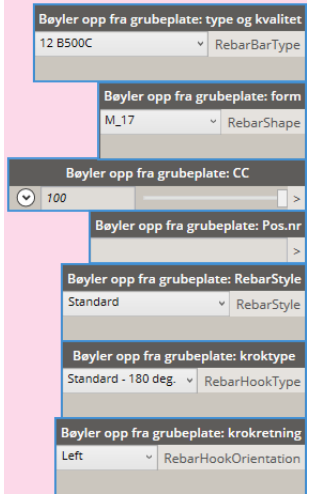

Parameter values
Posisjonsnummer (Revit definisjon)
Schedule Mark
Bygningsdel (Input)
Heisgrube
Bygningsdel (Revit definisjon)
Partition
Form (Revit definisjon)
Shape
Generell input
Generell input: Referanselinje for geometri (YK overkant grubeplate. Side ved heisdør)
Select
Curve
Edge of Element Id : 317681
Generell input: Velg grubeplate
Select
Element
Element : 317681
Tykkelse dekke over grubevegg
250.000
Tykkelse vegg 1.etg
200.000
Diameter bøyle opp fra grubeplate
12.000
Offset for nederste jern i vegg (ikke input)
25.000
Ø hjørnebøyle grubeplate
12.000
Properties.RebarHostCover
element
Top/Exterior
Bottom/Interior
Other
AUTO
Nominell overdekning (grubeplate)
Cnom
Cnom;
Kortside grubeplate
curve
double
AUTO
Langside grubeplate
Select
Curve
Edge of Element Id : 317681
Tykkelse grubevegg
Select
Curve
Edge of Element Id : 380916
Høyde grubeplate
Select
Curve
Edge of Element Id : 317681
Høyde grubevegg
Select
Curve
Edge of Element Id : 381058
Bredde grubevegg, kortside
Select
Curve
Edge of Element Id : 380916
Langside grubeplate
curve
double
AUTO
Tykkelse grubevegg
curve
double
AUTO
Høyde grubeplate
curve
double
AUTO
Høyde grubevegg
curve
double
AUTO
Bredde grubevegg, kortside
curve
double
AUTO
Code Block
breddeGrubeplate
breddeGrubevegg
x = (breddeGrubeplate - breddeGrubevegg) / 2;
utstikk = Math.Round(x);
Avstand YK grubeplate til YK grubevegg
Utstikk
Utstikk;

G

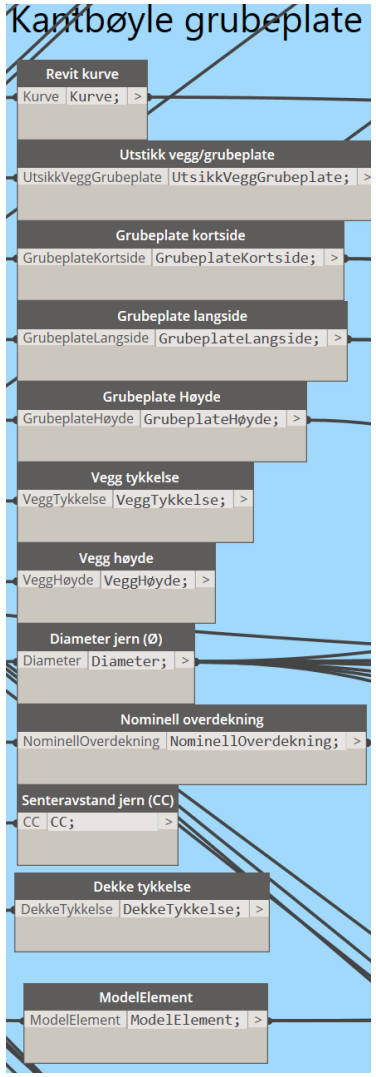
Kantbøyler



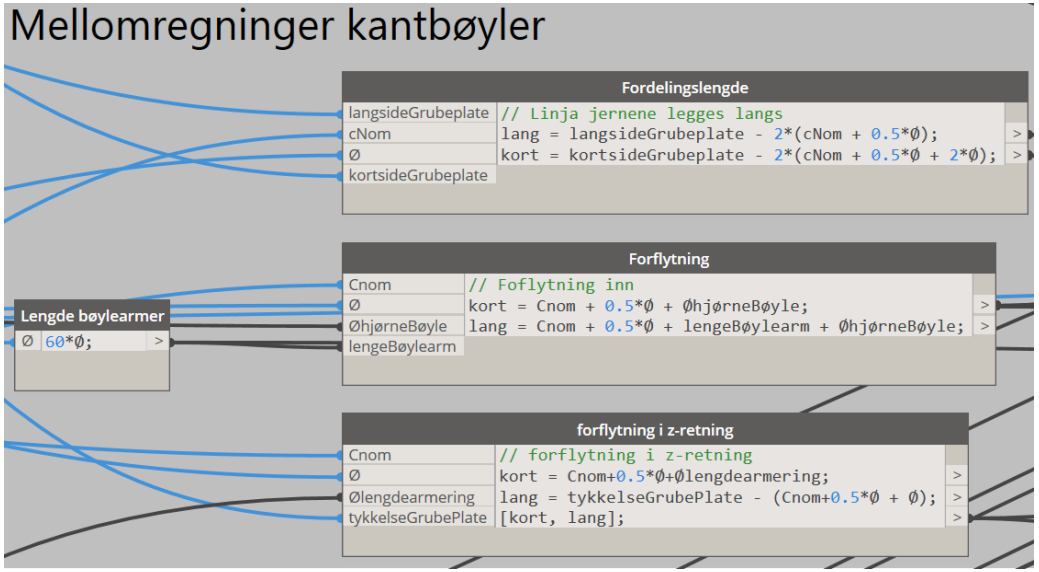
Bøyler opp fra grubeplate

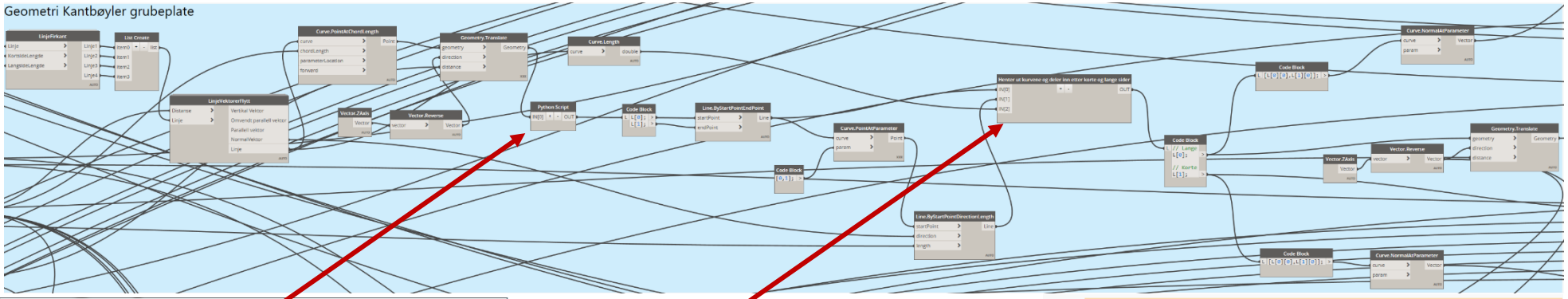


Kantbøyle grubeplate



Mellomregninger kantbøyler





Python Script

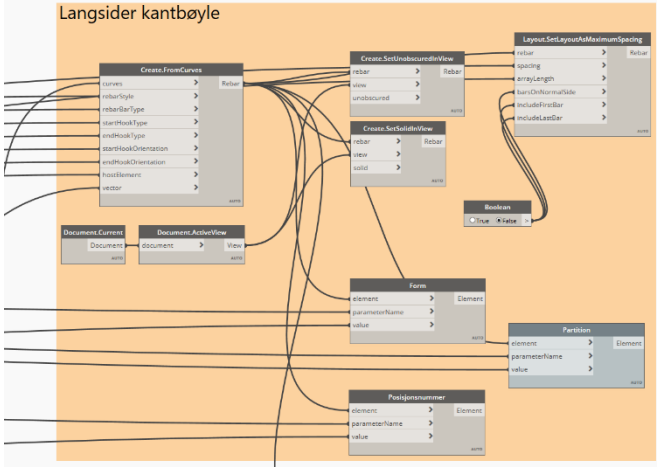
```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
lines = IN[0]

# Place your code below this line

flat_list = []

for sublist in lines:
    for item in sublist:
        flat_list.append(item)

start = flat_list[0::2]
slutt = flat_list[1::2]


# Assign your output to the OUT variable.

OUT = [start, slutt]
# Assign your output to the OUT variable.

```

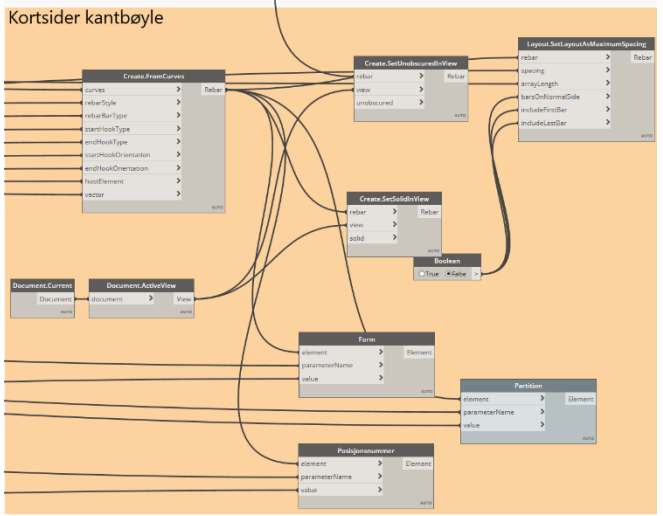
Run | Save Changes | Revert

Henter ut kurvene og deler inn etter korte og lange sider

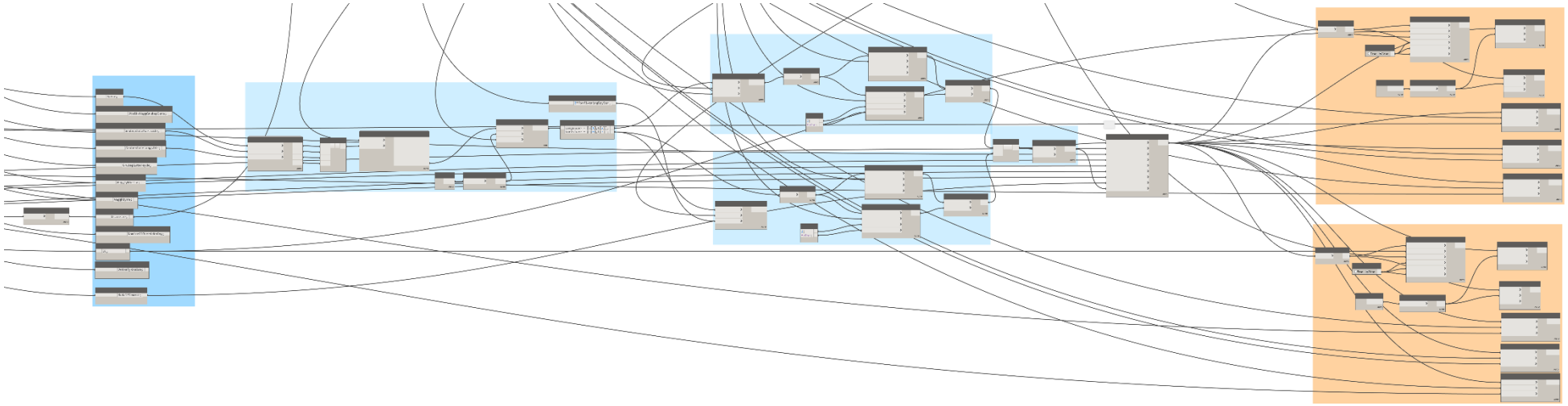
```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
korte = IN[0]
lange = IN[1]
lengder = IN[2]
# Place your code below this line
lange[0].append(korte[0])
lange[1].append(korte[1])
lange[2].append(korte[2])
lange[3].append(korte[3])

polykurver = []

for i in lange:
    kurve = PolyCurve.ByJoinedCurves(i, 0.001)
    polykurver.append(kurve)

kurver = []
for x in polykurver:
    kurve = PolyCurve.Curves(x)
    kurver.append(kurve)

# sorterer etter kort og lange linjer på grubeplata
maximum = max(lengder)
minimum = min(lengder)

outKurver = []
korte = []
lange = []

for i in range(len(kurver)):
    if lengder[i] == maximum:
        outKurver.insert(-1, kurver[i])
    else:
        outKurver.insert(0,kurver[i])

for i in range(len(kurver)):
    if lengder[i] == maximum:
        lange.append(kurver[i])
    else:
        korte.append(kurver[i])


# Assign your output to the OUT variable.
OUT = [lange, korte]


```

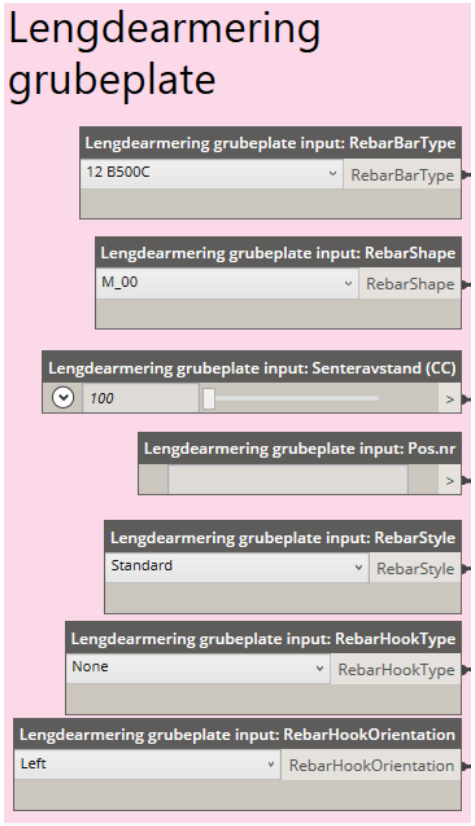
Run | Save Changes | Revert

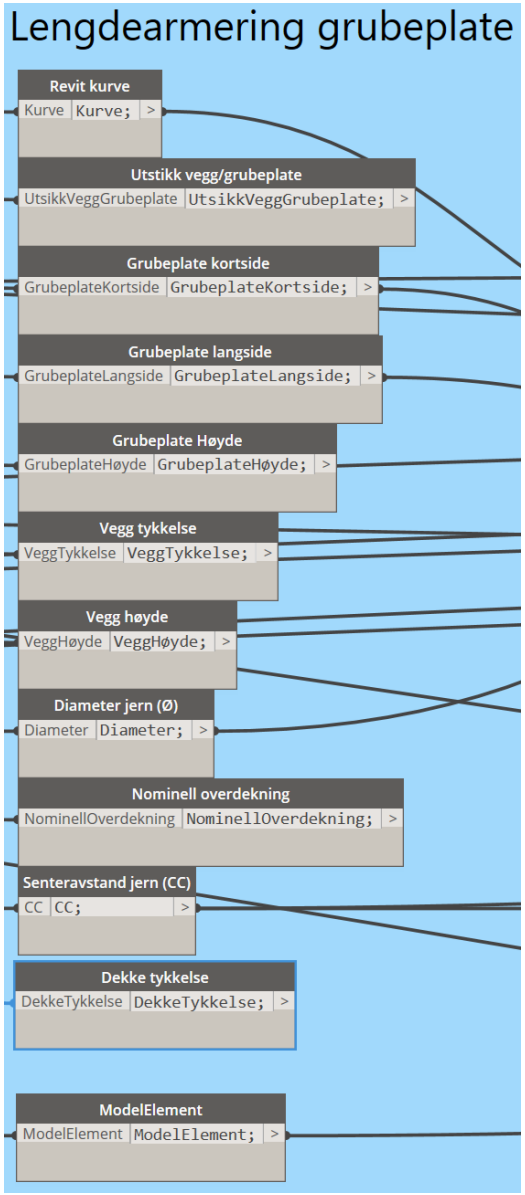


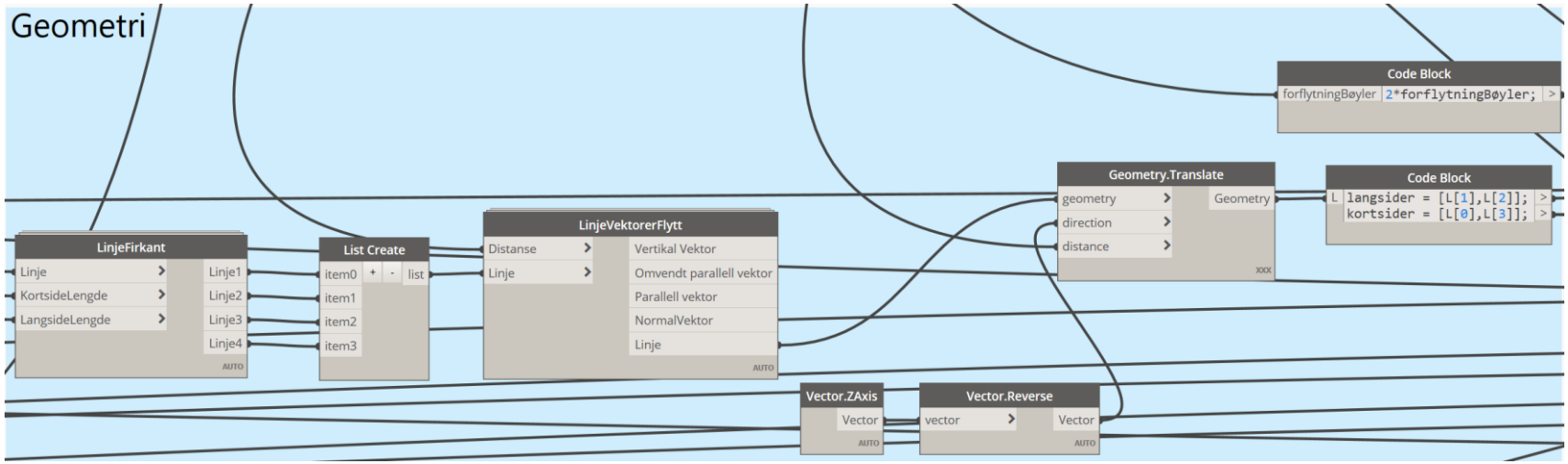


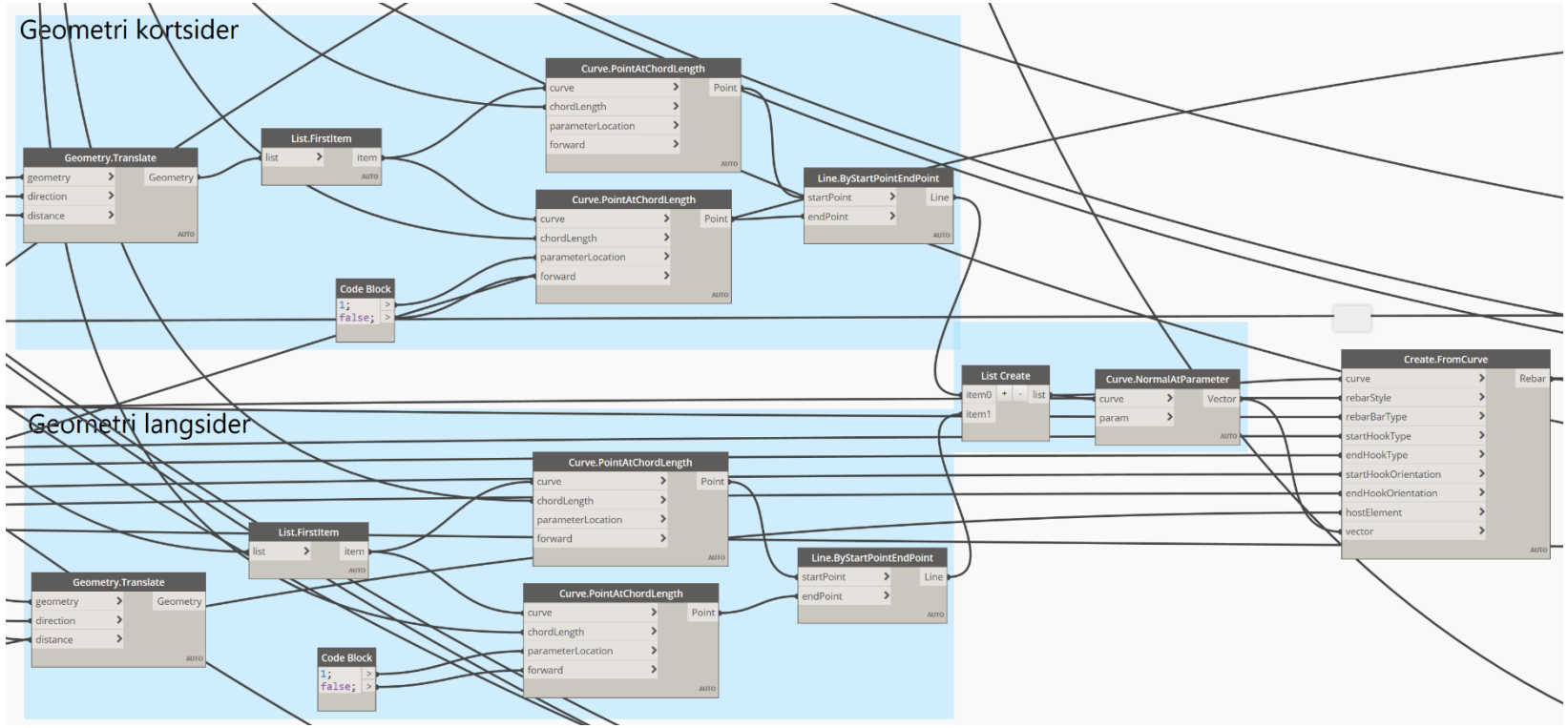
Lengdearmering

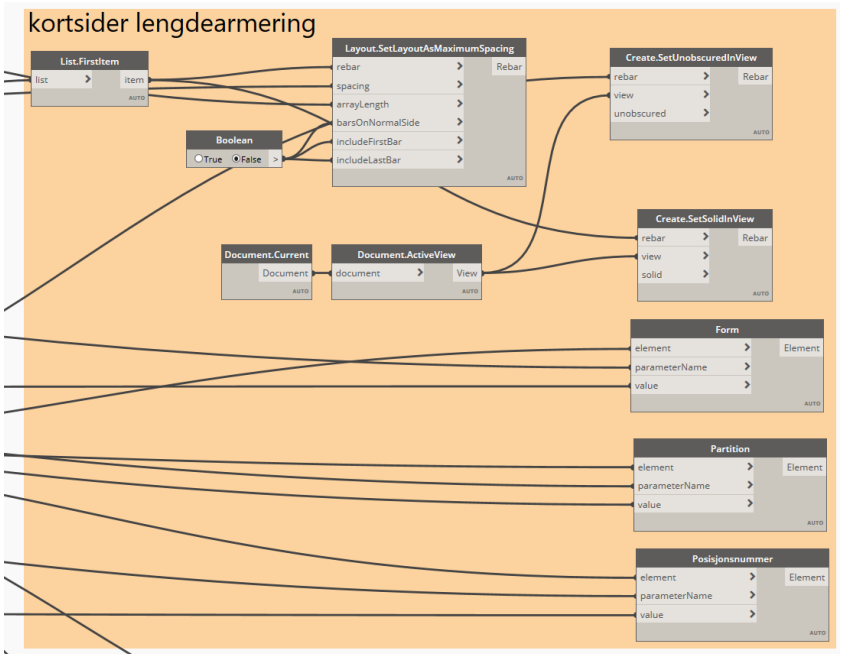


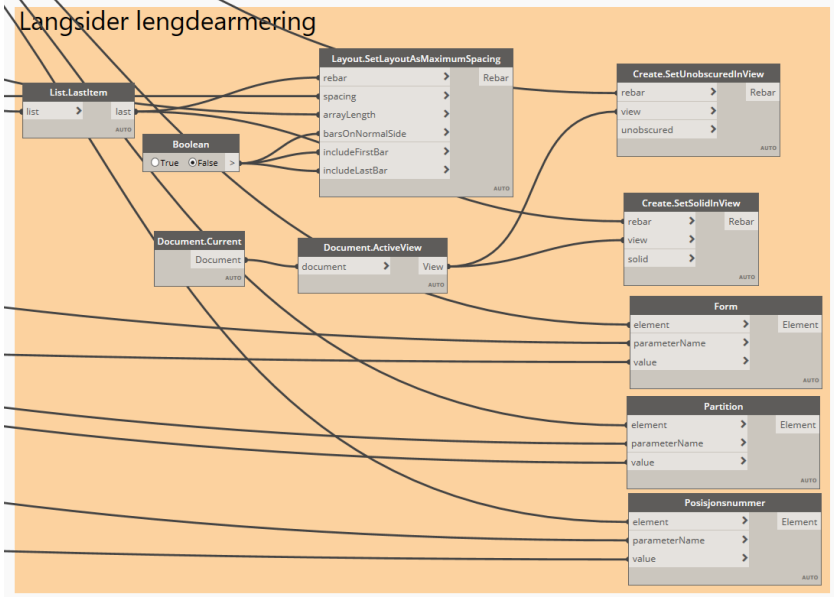

Lengdearmering grubeplate
Lengdearmering grubeplate input: RebarBarType
12 B500C
RebarBarType
Lengdearmering grubeplate input: RebarShape
M_00
RebarShape
Lengdearmering grubeplate input: Senteravstand (CC)
100
Lengdearmering grubeplate input: Pos.nr
Lengdearmering grubeplate input: RebarStyle
Standard
RebarStyle
Lengdearmering grubeplate input: RebarHookType
None
RebarHookType
Lengdearmering grubeplate input: RebarHookOrientation
Left
RebarHookOrientation


Lengdearmering grubeplate
Revit kurve
Kurve
Kurve;
Utstikk vegg/grubeplate
UtsikkVeggGrubeplate
UtsikkVeggGrubeplate;
Grubeplate kortside
GrubeplateKortside
GrubeplateKortside;
Grubeplate langside
GrubeplateLangside
GrubeplateLangside;
Grubeplate Høyde
GrubeplateHøyde
GrubeplateHøyde;
Vegg tykkelse
VeggTykkelse
VeggTykkelse;
Vegg høyde
VeggHøyde
VeggHøyde;
Diameter jern (Ø)
Diameter
Diameter;
Nominell overdekning
NominellOverdekning
NominellOverdekning;
Senteravstand jern (CC)
CC
CC;
Dekke tykkelse
DekkeTykkelse
DekkeTykkelse;
ModelElement
ModelElement
ModelElement;

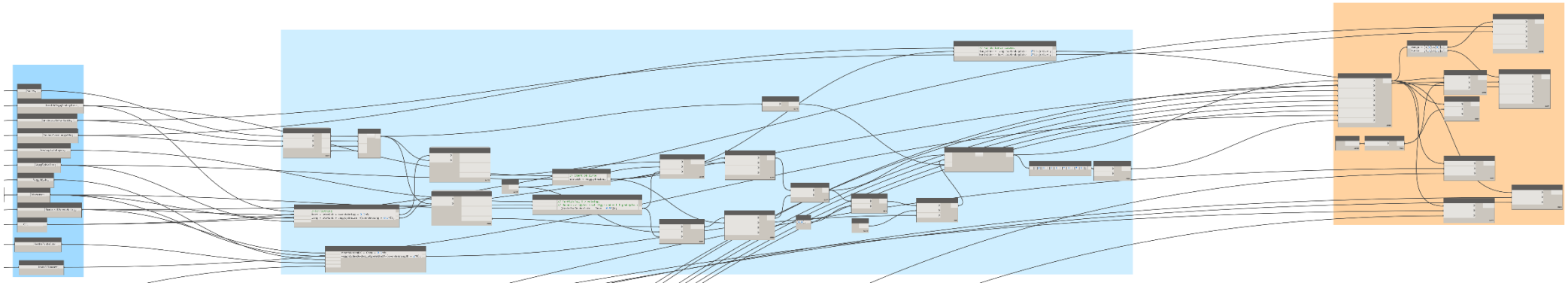

Geometri
Code Block
forflytningBøyler
2*forflytningBøyler;
Geometry.Translate
geometry
direction
distance
Geometry
Code Block
L
langsider = [L[1],L[2]];
kortsider = [L[0],L[3]];
LinjeFirkant
Linje
KortsideLengde
LangsideLengde
Linje1
Linje2
Linje3
Linje4
AUTO
List Create
item0
item1
item2
item3
list
LinjeVektorerFlytt
Distanse
Linje
Vertikal Vektor
Omvendt parallell vektor
Parallell vektor
NormalVektor
Linje
Vector.ZAxis
Vector
Vector.Reverse
vector

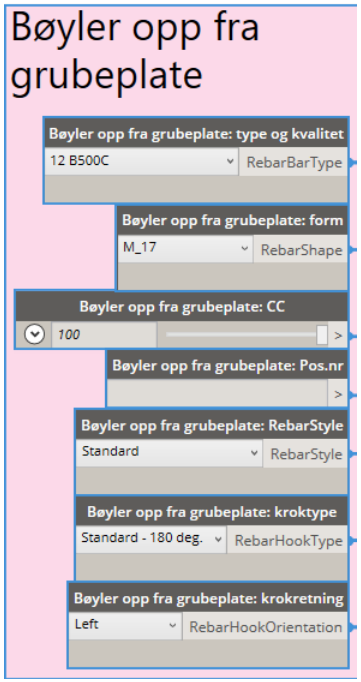

Geometri kortsider
Curve.PointAtChordLength
curve
chordLength
parameterLocation
forward
Point
List.FirstItem
list
item
Geometry.Translate
geometry
direction
distance
Geometry
Line.ByStartPointEndPoint
startPoint
endPoint
Line
Code Block
1;
false;
Geometri langsider
List Create
item0
item1
Curve.NormalAtParameter
curve
param
Vector
Create.FromCurve
rebarStyle
rebarBarType
startHookType
endHookType
startHookOrientation
endHookOrientation
hostElement
vector
Rebar

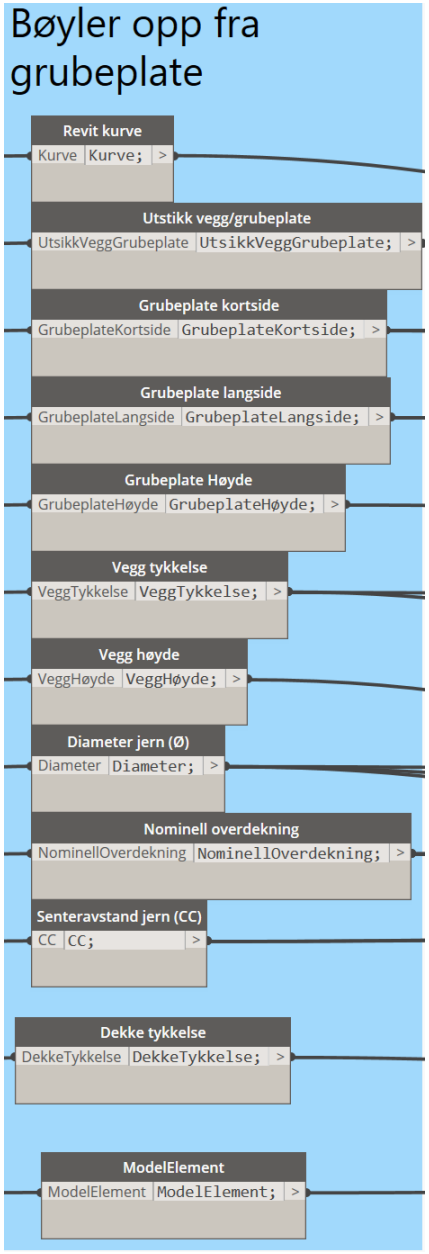

kortsider lengdearmering
List.FirstItem
list
item
AUTO
Boolean
True
False
Layout.SetLayoutAsMaximumSpacing
rebar
spacing
arrayLength
barsOnNormalSide
includeFirstBar
includeLastBar
Rebar
AUTO
Create.SetUnobscuredInView
rebar
view
unobscured
Rebar
AUTO
Create.SetSolidInView
rebar
view
solid
Rebar
AUTO
Document.Current
Document
AUTO
Document.ActiveView
document
View
AUTO
Form
element
parameterName
value
Element
AUTO
Partition
element
parameterName
value
Element
AUTO
Posisjonsnummer
element
parameterName
value
Element
AUTO

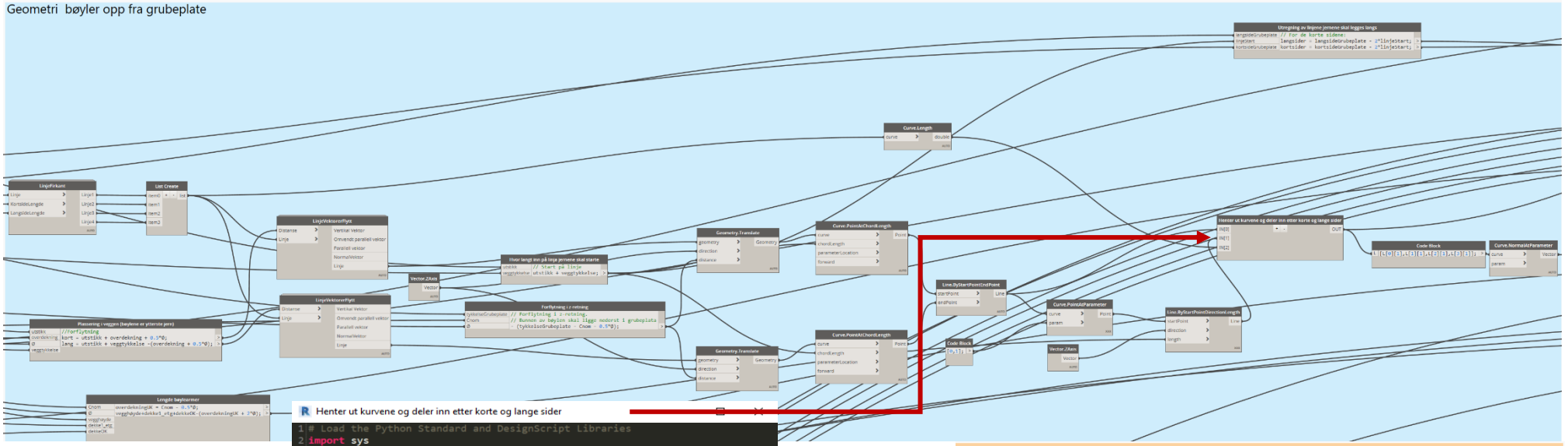

Langsider lengdearmering
List.LastItem
list
last
AUTO
Boolean
True
False
Layout.SetLayoutAsMaximumSpacing
rebar
spacing
arrayLength
barsOnNormalSide
includeFirstBar
includeLastBar
Rebar
AUTO
Create.SetUnobscuredInView
rebar
view
unobscured
Rebar
AUTO
Create.SetSolidInView
rebar
view
solid
Rebar
AUTO
Document.Current
Document
AUTO
Document.ActiveView
document
View
AUTO
Form
element
parameterName
value
Element
AUTO
Partition
element
parameterName
value
Element
AUTO
Posisjonsnummer
element
parameterName
value
Element
AUTO

Bøyler opp fra Grubeplate




Bøyler opp fra grubeplate
Bøyler opp fra grubeplate: type og kvalitet
12 B500C
RebarBarType
Bøyler opp fra grubeplate: form
M_17
RebarShape
Bøyler opp fra grubeplate: CC
100
Bøyler opp fra grubeplate: Pos.nr
Bøyler opp fra grubeplate: RebarStyle
Standard
RebarStyle
Bøyler opp fra grubeplate: kroktype
Standard - 180 deg.
RebarHookType
Bøyler opp fra grubeplate: krokretning
Left
RebarHookOrientation

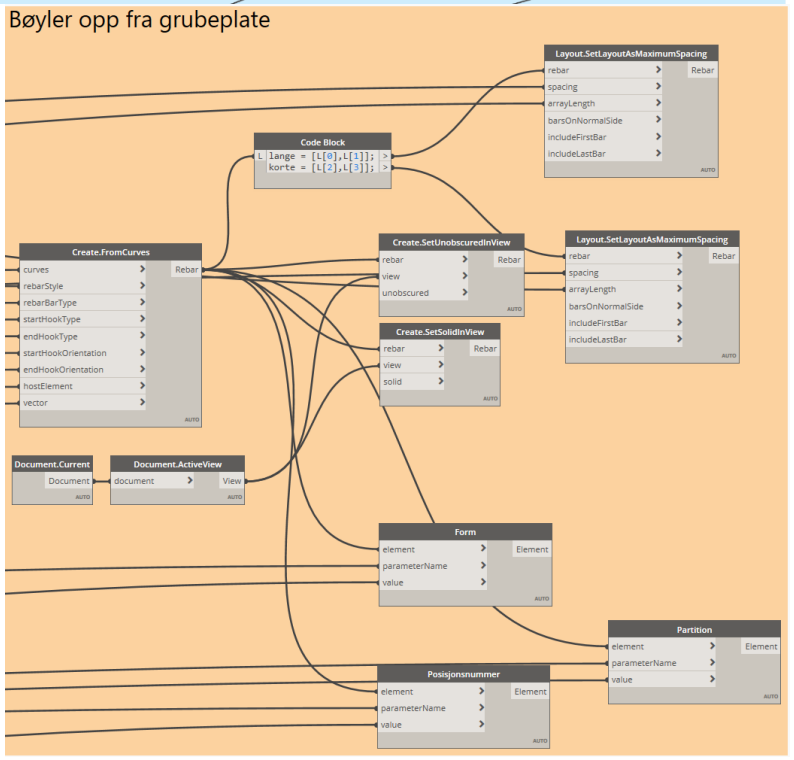

Bøyler opp fra grubeplate
Revit kurve
Kurve
Kurve;
Utstikk vegg/grubeplate
UtsikkVeggGrubeplate
UtsikkVeggGrubeplate;
Grubeplate kortside
GrubeplateKortside
GrubeplateKortside;
Grubeplate langside
GrubeplateLangside
GrubeplateLangside;
Grubeplate Høyde
GrubeplateHøyde
GrubeplateHøyde;
Vegg tykkelse
VeggTykkelse
VeggTykkelse;
Vegg høyde
VeggHøyde
VeggHøyde;
Diameter jern (Ø)
Diameter
Diameter;
Nominell overdekning
NominellOverdekning
NominellOverdekning;
Senteravstand jern (CC)
CC
CC;
Dekke tykkelse
DekkeTykkelse
DekkeTykkelse;
ModelElement
ModelElement
ModelElement;



Henter ut kurvene og deler inn etter korte og lange sider

```python
# Load the Python Standard and DesignScript Libraries
import sys
import clr
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

# The inputs to this node will be stored as a list in the IN variables.
korte = IN[0]
lange = IN[1]
lengder = IN[2]
# Place your code below this line
lange[0].append(korte[0])
lange[1].append(korte[1])
lange[2].append(korte[2])
lange[3].append(korte[3])

polykurver = []

for i in lange:
    kurve = PolyCurve.ByJoinedCurves(i, 0.001)
    polykurver.append(kurve)

kurver = []
for x in polykurver:
    kurve = PolyCurve.Curves(x)
    kurver.append(kurve)

# sorterer etter kort og lange linjer på grubeplata
maximum = max(lengder)
minimum = min(lengder)

outKurver = []
korte = []
lange = []

for i in range(len(kurver)):
    if lengder[i] == maximum:
        outKurver.insert(-1, kurver[i])
    else:
        outKurver.insert(0,kurver[i])

#for i in range(len(kurver)):
#   if lengder[i] == maximum:
#       lange.append(kurver[i])
#   else:
#       korte.append(kurver[i])


# Assign your output to the OUT variable.
OUT = outKurver
```

Run | Save Changes | Revert



Vedlegg H - Spørreundersøkelse

Rapport fra «Parametrisk modellering idag - Bacheloroppgave 2021»

Innhentede svar pr. 15. april 2021 16:10

- Leverte svar: **16**
- Påbegynte svar: **0**
- Antall invitasjoner sendt: **21**

Hvilken stilling har du? / What position do you have? *

- Fagspesialist
- Sivilingeniør
- CDO
- Seksjonsleder
- Avdelingssjef
- Sivilingeniør
- Seksjonsleder BIM
- BIM leder oppdrag
- BIM-koordinator
- Annlegsleder/ BIM koordinator
- Avdelingssjef konstruksjonsteknikk
- Ass. PL
- BIM-koordinator
- BIM Spesialist
- Prosjekteringsleder
- Prosjektleder digitalisering

BRUK AV PARAMETRISK MODELLERING / USE OF PARAMETRIC MODELING

Norsk:

I andre del ønsker vi å lære mer om din kjennskap til, og hvordan du bruker parametrisk modellering, hva du bruker det til, og dine tanker rundt bruken av dette.

Med parametrisk modellering mener vi: at det er en modelleringsprosess med mulighet til å endre geometrien i en modell gitt fra parametere.

English:

In the the second part, we want to learn more about your knowlage, and how you use parametric modeling, what you use it for, and your thoughts on its use.

By parametric modeling we refer to a modeling process with the ability to change the shape of the model geometry as soon as the dimension velue is modified.

6. Hva er din erfaring med parametrisk modellering? / What is your experience with parametric modeling? *

Norsk:

Dette skal hjelpe oss med å kartlegge forholdet du har til parametrisk modellering.

English:

This will help us map your relationship with parametric modelling.

Svar	Antall	Prosent	
Har aldri hørt om / Never heard of	2	**12,5 %**	
Har hørt om det, men har aldri brukt det / Have heard about it, but have never used it	5	**31,2 %**	
Kjenner til det og har jobbet litt med det utenom prosjekter / Know about it and have been working on it outside of work-related projects	5	**31,2 %**	
Kjenner veldig godt til det og bruker det ofte til modellering av prosjekter / Know it very well and often use it for modeling projects	4	**25 %**	

9. Hvilke arbeidsoppgaver løser du ved hjelp av parametrisk modellering? / What tasks do you solve using parametric modeling? *

Norsk:

Spørsmålet skal hjelpe oss å kartlegge i hvilke områder blir parametrsk modellering brukt mest.

English:

The question should help us map in which areas parametric modeling is used the most.

Svar	Antall	Prosent	
Utforming av kompleks geometri / Design of complex geometry	5	**31,2 %**	
Armering / Rebar	5	**31,2 %**	
Bærekonstruksjoner (søyler, dekker, bjelker) / Supporting structures (columns, slabs, beams)	2	**12,5 %**	
Annet / Other	11	**68,8 %**	

Hvilke andre felt jobber du med parametrisk modellering? / What other field do you work with parametric modeling?

- Automatisering av diverse
- Lage parametre, flytte parametre for IFC-eksport etc.
- Jeg kjenner til prinsippene, men bruker ikke programvare selv.
- Informasjon mm
- Aner ikke hva som menes med paremetriske modellering, så svarene under blir bare midt på treet/ nøytrale
- Ikke brukt
- -
- F.eks. modellering av infrastruktur i InfraWorks.
- Bruker Dynamo mye til Automatisering av informasjon i Revit.
- Bruker det ikke
- Bruker det ikke

Vedlegg I - Dybdeintervju Marcin Luczkowski

Dybdeintervju Marcin Luczkowski

So now you came up with a lot of pros for parametric modeling, but do you have any cons?

Yeah of course there is a lot of disadvantages and a lot of issues that is coming with it. First of all, this is coming, from a very general perspective, students and people have to start coding a bit more. Nothing is for free, so you have to spend time on it, so you have to take this time. So, the study will have to be rearranged with it, maybe the specialization has to come sooner in the study. If you choose this direction you should be program a bit more. The same is when you come to the company and you want to deal with those topics, fortunately or unfortunately you have to think about this programing a little bit more, and I assume it is happening, also with myself, that every company needs to have a specialist in this field. It´s hard to predict that there will only be one person that knows everything about everything. So, we will have to develop master's degree students with the knowledge about programming, FEM, building physics and all those other things. I think it is possible, I think it is more about the specialization, so we have to specialize people in this direction and this. The disadvantage is that you have to learn a little bit more about this topic, you will lose some knowledge about the building industry or design. Also, the disadvantage is that we are coming into the point where the automation is bigger and bigger, what I am doing in my job is to automate processes. When we get drawings from architect, I am trying to write a code that automatically transferrers this architect model to structural model. Previously this was done in 20-50 hours by some structural engineer that was redrawing this model from scratch. Right now, we automated, or we are trying to automate, in some instances we only need two minutes to do it, in some ways we need a couple of hours to do it. Anyways we will have a different job. There is a huge potential in also implementing machine learning into this, because everything in Grasshopper or Dynamo it's just about the numerical data, so if you also have like neural network that you teach it can be done more efficient. There is of course a possibility in the future that a lot of those jobs will just not exist anymore, I can give you also some examples about some companies which tried to involve already automatic joint calculation in the structure in the neural network, and it did this with 99% success. So, developing automating in the building design, of course it´s in the best scenario the structural engineer will just be a person that will have to decide what computer to calculate, this will shift the work from being a creator to be a supervisor. This is of course a long-standing process, for me it's at least 20-50 years of further exploration of this process. In today's model I think that some of the people that are using parametric modeling, to be honest, in these smaller projects, they are wasting their time because it could have been done without parametrization. If you don't need optimization, the structure is not repetitive, and it will not be built any structure like this, then a parametric model is not really needed. We should decide for making a parametric model only if we want to implement optimization, either we think this model can be used for later projects. I don't think we should make a parametric model, especially through Dynamo or Grasshopper, if there is no need for it. It is a bad thing that everybody wants to make these parametric models because in many cases it´s not really needed, and in some cases its really slowing down the processes. So, you could say this is a hype in some cases, or like a fairytale that a parametric model will be the answer for everything. I think it will be if you connect it to the neural network, but since we don't have this kind of AI at the moment, I don't think a parametric model is needed in many cases. There is a good chance that using a parametric model is a waste of time in many cases. Sorry, maybe I am elaborating to much?

And then we have the last question of this part, since we're most familiar with Dynamo and Grasshopper, we were wondering what the key points that set those two apart is?

So, you know the story, Grasshopper was created and then Autodesk say "Wow this is some cool stuff! Let's buy it!", but they couldn't buy it because Rhino wouldn't sell it. So instead they buy the guy that created Grasshopper and ask him if he could create Dynamo. So, then we have Dynamo. If you ask about the differences, in general they are almost the same. The problem is that Dynamo is much slower and does not have the right graphic engine, so you probably already see it's pretty slow in creating reinforcements and stuff like that, when Grasshopper can still be quite fast in doing this. It's all about the approach that should be the Dynamo subprogram of Revit, and Revit itself is quite slow because of this BIM concept in having these heavy objects with many properties, instead of just adding them as an attribute and focus on previewing just the geometry. Anyways Rhino Grasshopper is a quite fast and efficient software when it comes to all this geometry previewing. So, per definition it's very quick and fast engine to preview cut geometry. Dynamo, I like the sandbox Dynamo because you can as a standalone program, before it was Dynamo Studio, you could make quite efficient codes. When it comes to work with Revit I think it's the issue with the speed more than Revit with Dynamo itself. Per definition they are the same. The hierarchy of the classes of the objects most of them are really the same. Grasshopper has a longer history and a bigger community if people doing plug-ins, and it's better known in the market. First of all, the difference is the speed, which is why I generally always prefer to work in Grasshopper.

What kind of tasks do you solve using parametric modeling?

We can spend another 20 minutes on that, because to be honest I try to... but this is very specific to me, but I try to push everything to be done in automated way, it's hard to say, and here is also the problem of definition of parametric modeling. If I am doing my codes in C Sharp, in Visual Studio, and creating plug ins for the Grasshopper, am I still doing parametric modeling? Or am I software developer? I am just delivering the tool, I think I am doing parametric modeling, because I am influencing how the parameters are set to end at the connection between the input and output. So, in my case I'm doing finite element models out of the architectural models. I'm doing also... helping with achieving building physics models for the people who are responsible for it... Now I will talk about the work in Multiconsult, so mostly I'm doing Structural Engineering work, so mostly I'm trying to get from IFC-model, which is coming from Architects, model automatically which that can be calculated with all the loads and supports directly to the, for example FEM-design or Robot type software to calculate them, and then get back with those information to the Grasshopper to say it should be a change in this cross-section. But also, I was doing one project the last year when we wanted to combine all so the technological calculation and then I had to make a parametric model of the whole building and also, I have to create the geotechnical soil model, so all the layers of the soil with different properties and send it to the geotechnical software. So, I could say that everything that I got to be done I am trying to push through parametric modeling, through Grasshopper, through Rhino, so I try to automate everything what I got. When it comes to my work at the NTNU, it's completely like super strange projects, virtual reality and making a plug-in view in Grasshopper to enable the people to design directly FEM-design and FEM-model to be done in the virtual reality. So, I could say I am doing everything. Right now, there is nothing... I haven't had a project when I didn't use coding either visual either, just scripting.

That's interesting that you can basically use it for everything then?

Yeah, that's exactly interesting that already we have... Also the FEM is worth mentioning here, that also the software developers from the side of the FEM, Finite Element Method, like Autodesk Robot or FEM- design, they are all open for the visual programming and algorithm aided design and parametric modeling, so most of this companies all ready make plug ins for the Grasshopper, which and are enabling you to combine your design with their software, so it's not that I'm inventing the whole flow, there is a huge in for the marked, software developers, from many... or from the biggest companies, most of them have started to produce the plug ins for either Dynamo or Grasshopper. It's super interesting that we have the option to do it, you just have to know how to do it.

So, you could say that, if you see it long term, it may be more efficient? But project by project maybe not?

Maybe not, it's all with the parametrization, if the problem that... to build the parametric model, you need more time then to build the normal project, when everything is stiff. But with every change in the model that you are doing suddenly you don't have to delete everything, you just have to change the parameter. But to create a model which is really parametric and it's flexible to the change, and you also have to sit down first and think a lot how really build it to be able to create such a universal parametric model. It's really kind of the problem, so effectiveness, it's not truly... it's really hard to measure. I think that most of my work it's about automation, so I'm trying to create codes which will automate the design process. So probably on some of the projects I spend much more hours than it should be, but if I once do the code for creating for example the reinforcement, it will be used in many, many projects. So, it's hard to calculate it what was the effectiveness in the projects.

What criteria would you set for using parametric modeling in a project?

First of all, the criteria it's definitely, if the project will be repeated in such way. If not, what parts of the projects will be repeated in other projects. So sometimes the project itself, like a structure, it's very individual and you will see that it will not be used in another project, but then you came to for example end joint solutions, so the connector in between the steel and the beam, which was in the specific project and new stuff, but you know that you will use it in the next projects. Sometimes we have a project which... for example building physics, we are using the zones to the script where the light... So maybe this project it could be nice to create a flexible method to off populating your structure with the light and then you need a code which is doing it. So, criteria is very wide, again and you have to think what really can be used in the later... So first of all, the criteria should be that you look on the project and you try to think what can be used from this project in your later projects. So, it's hard for me to say what should be the criteria to use parametric modeling... Definitely if it's optimization, then you can be completely sure that the parametric modeling is the best way to do it. Now this is one strong criterion which is saying or giving you huge depth that this time you should use parametric modeling. But then the repetitiveness of the model or a part of the model should also be a big thing to say that you should use parametric modeling.

You talk about reusing parts of scripts from one project to another. Do you usually do this? You can reuse an entire script or just small parts from the scripts from project to project?

- When I was starting, I was more like... I didn't care about that so much, so in every project I was starting almost ten new scripts. It was very hard also to open old scripts sometimes

because of the software issue, because there was ten new version of Dynamo or of the Grasshopper, and suddenly if you scripted in one tool work and it was also one of the triggers to way to push me to build mostly with python and C sharp, because it's much less sensitive on the software change and development. So if you are doing it in the pure C sharp or Python, you don't care about the new version of the software. But right now most of my cases and projects that I have, I am trying to close the scripts in the form of the plug in. So in Multiconsult I'm developing for example Multiconsult plug-in, with every project I try to add extra node in Dynamo or extra... or the same component in the Grasshopper to this plug-in. So of course, sometimes when I say that this algorithm is very case-specific, I try to just do it in form of the code which I am doing just once and then probably, nobody else will use it. But most of the codes I always try to close the plug-in part. So yeah... Then with in every project I am just open my rebar with the plug-in and then I am dragging and developing the tools which I was doing before... So this is my way of just not reinventing the codes from project to project, and just stay with the codes which was created by me in the previous project. And in most of the cases it works, so I'm don't have to create the models from scratch, I can just re-code from scratch and I can just use once that I was creating previously.

In the last section we want to talk about the future of parametric modeling, and how will you see the developing of parametric modeling will go through time? And what will it take?

Like I said previously, I believe that there will be development of machine learning and artificial intelligence and that we are just right now have a step to pull out full automatization, right now we have like semi automation of many processes, but if you ask me what will happen in 50 years I would say, yes, the AI will take a bigger role and parametric modeling was just a step towards this, but if you ask me about five years in the future, I think parametric modeling and everything which is connected to it like Grasshopper, Dynamo, visual programming it will still be on the rise, so I think most of the companies will shift into this. And they will see... I think the marked of the people also with does skills will be just developing the next couple 10-15 years, so does are my predictions.

Vedlegg J - Dybdeintervju Fredrik Jacobsen

Transkribering av dybdeintervju med Fredrik Jacobsen

Hvordan du vil forklare parametrisk modellering? Litt med egne ord og erfaringer.

Jeg mener jo, fra eposter med Jonas, at det har blitt inflasjon i bruken av parametrisk modellering-begrepet. All modellering er parametrisk på en eller annen måte. Det blir nesten en fanesak for meg å prøve å være med spesifikk når jeg snakker om det, men det har jo blitt sånn i bransjen at parametrisk modellering er synonymt med visuell programmering. Forhåpentligvis blir man mer presise etter hvert.

Hva mener du er forskjellen på visuell programmering og parametrisk modellering?

Parametrisk modellering er jo egentlig bare å sette på parameter for.. eksempelvis sette ut et objekt da. Og hvis du har satt ut en vegg, med en høyde, så har man en høydeparameter, og ved å endre denne parameteren i Revit vi jo være parametrisk modellering per def. Mens i visuell programmering så bruker du jo disse boksene for å gjøre endringer i programmet. Visuell programmering er et verktøy, mens parametrisk modellering bare er en generell beskrivelse av noe man gjør på en måte. Jeg vet ikke om det gir mening, men man kan også bruke programmeringsspråk direkte for å parametrisere ting. Man må jo ikke bruke visuell programmering, det er jo det som er poenget, at det finnes andre veier til målet, Excel for eksempel.

I hvilke sammenhenger ser du at det er mest nyttig å bruket det?

Godt spørsmål, jeg tror at så lenge man klarer å holde omfanget av bruken til noe spesifikt. Slik som at dere skal modellere armeringen for heisgrube da, er det hele sjakta eller bunnen bare?

Det er fra grubeplata, og opp til første plan.

Ja, så hvis man vil klare å holde omfanget til ikke så mye mer enn det, for det blir ganske så mye armering som skal med som man skal kontrollere og dra i, så tror jeg at det er effektivt å bruke det og kanskje gjenbruke det. Jeg tror a det er veldig effektivt i forprosjekt ting og sånn er man bare vil vise frem veldig mye fort. Det jeg tror at det som er vanskelig med visuell programmering er at det er vanskelig med gjenbruk siden man tenker ulikt, og man har ikke det man kaller design patterns. I vanlig programmering som vil si at man har struktur og orden på disse boksene som gjør det gjenbrukbart da. Og det ser man også i mange prosjekter at når man kommer langt uti der da så er det veldig vanskelig for noen som, selv om man kan liksom Grasshopper så er det vanskelig å komme inn i andres kode da.

I hvilket omfang føler du at visuell programmering blir brukt i bransjen i dag?

Jeg tror at det har vært en sånn eksplosiv utvikling på det området. Det er veldig mange, spesielt nyutdanna som kommer inn og har lyst til å bruke dette her. Sånn alle som har starte de siste fem årene har kunnet noe Grasshopper eller en del Dynamo, eller starter å lære seg det veldig fort. Av unge er det sikkert 70% som kan noe om det i hvert fall, men jeg har ikke noe tall på det. Men det er i hvert fall ganske mange som begynner å bruke det da.

Ser du noen fordeler og ulemper med visuell programmering. Du har vært innpå noen fordeler, men er det noen ulemper?

Fordel er at man får til veldig mye, veldig fort. Uten å kunne programmering, som sådan. Ulempen er at eskaler barheten er veldig... skal ikke si at jeg kan det godt nok til at jeg kan si at det er veldig dårlig, men det blir veldig mye cliché – mange kokker mye søl. Og alle gjør ting forskjellig, alle tenker litt forskjellig. Så det er vanskelig å strømlinje formen for hvordan man skal gjøre ting. Og jeg tror

også disse scriptene, de kan bli veldig store til å gjenbrukes senere. Så lenge man klarer å begrense omfanget til mindre oppgaver som er litt kompliserte, der man virkelig trenger det, vil det være veldig bra.

Hvilke typer oppgaver ser du som hensiktsmessig å løse med visuell programmering?

Vi brukte Grasshopper til å modellere en del sånn rekkverk for eksempel, og i Tekla til et prosjekt som jeg jobbet med i hele fjor, så til på en måte har vært gjentakende og som ikke direkte støttes av Tekla er det vi har brukt det til foreløpig. Da jeg jobbet i Johs Holt så lenge ikke ting støttes enkelt internt i programmet så har jeg brukt eller fått noen til å hjelpe meg med å lage ting som er tydelige på en annen måte på scripting i Grasshopper. Planbruk også vært sånn at vi har satt ut telegrupper i et fundament for det, siden vi kan... konkretiserte systemet i veldig mange retninger og sånn, så sånne type oppgaver er liksom det jeg har vært med på.

Du sier du har brukt visuell programmering litt, opplever du at det effektiviserer jobben du skal gjøre i stor grad? I hvilken grad opplever du det?

Det er hvis man bruker det til de tingene som ikke støttes direkte av programvaren selv, eller finnes gode workflows og få til, så effektiviserer det helt klart jobben, men hvis man prøver å gjøre for mye så ender det opp med et spagetti-monster du ikke klare å temme til slutt, og da vil mye av tiden gå til å feil søke på hva som går feil her.

Du vil kanskje si det er tidsbesparende på mindre prosjekter?

Ikke mindre prosjekt, men mindre oppgaver i prosjektene.

Vi har vært innpå det med generaliserte script. Hva er dine tanker om generaliserende script? Er det mulig å få til eller er det for store forskjeller fra prosjekt til prosjekt eller fra oppgave til oppgave?

Godt spørsmål, jeg tror at man kan få til generelle script på spesifikke problemer, for eksempel modellering av en heisgrube. Betongen det kan man sikkert ganske greit få til i et script og ha parametere som man kan endre på de tingene som trenger å endres på, tykkelser, lengder, høyder, utsparinger osv. men jeg tror ikke man kan ta ut et helt bygg også generalisere det til å gjøre alt, da blir det for store forskjeller. Men ting som er gjentakende er jo selvfølgelig effektivt å bruke sånt på. En annen ting er landkar på bruer som finnes noen ganger i forskjellige løsninger, men det er også noe som stort sett har samme utforming på et vis, men på forskjellige høyder, lengder, bredder osv.

Hva føler du skal til for å kunne bruke det til alt? Hvis du tenker litt utenfor boksen. Du har mye begrensinger som du har listet opp, men hva må til for at visuell programmering skal ta over?

Jeg vet ikke. Jeg tror ikke det bør være noe mål om at det skal ta over heller kanskje. Hvis det skulle ta helt over, så vil jo prinsipielt i Rhino og Grasshopper, vil man ikke ha behov for de andre programmene hvis man har klart å ta over alt med visuell programmering. Da kan man pushe alt i Rhino egentlig. Men jeg tror ikke det er nyttig å gjøre alt i visuell programmering, uansett hvor bra det blir. Det å lage en utsparing i en vegg og sende til revisjon og da er det kanskje litt hardt å gå inn i scriptet og lage ny utsparing. Så får man feil også må man forfølge den feilen og lage den utsparingen direkte i Revit. Jeg tror man kommer aldri uten om å jobbe direkte i programmet.