2021•2022

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: bouwkunde

Masterthesis

Ontwikkeling van scripts voor de simulatie van kolomverlies in commerciële analysesoftware

PROMOTOR :

Prof. dr. ing. Bram VANDOREN

PROMOTOR :

dr. ir. Tom MOLKENS

Stijn Caerels, Niels Kusters

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: bouwkunde

Gezamenlijke opleiding UHasselt en KU Leuven

2021•2022

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: bouwkunde

Masterthesis

Ontwikkeling van scripts voor de simulatie van kolomverlies in commerciële analysesoftware

PROMOTOR :

Prof. dr. ing. Bram VANDOREN

PROMOTOR :

dr. ir. Tom MOLKENS

Stijn Caerels, Niels Kusters

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: bouwkunde

UHASSELT

KU LEUVEN

Woord vooraf

Deze masterproef brengt een eind aan de opleiding industrieel ingenieur bouwkunde. Het onderzoek liep niet altijd vlekkeloos en daarom willen we graag een dankwoord uitspreken aan de personen die ons zowel begeleidt als ondersteunt hebben gedurende het onderzoek.

Graag zouden we in het bijzonder onze promotoren prof dr. ing. Bram Vandoren en dr. Ir. Tom Molkens willen bedanken voor de feedback, begeleiding en opvolging om deze masterproef mee tot stand te brengen. Daarnaast bedanken we graag familie en vrienden voor hun steun en toeverlaat tijdens dit proces. Tenslotte willen we zowel het bedrijf SCIA als zijn werknemers bedanken voor het steeds spoedig beantwoorden van onze vragen in verband met de software en het voorzien van een proefversie van de software gedurende dit onderzoek.

Inhoud

Lijst van tabellen

Lijst van figuren

Abstract

Robuustheidcontroles omtrent kolomverlies zijn erg complex en tijdrovend. Daarom tracht dit onderzoek de berekening voor kolomverlies te automatiseren. Om dit mogelijk te maken wordt er nagegaan welke software de juiste mogelijkheden biedt. Dit wordt onderzocht door op een eenvoudige structuur manueel kolomverlies te simuleren en de resultaten te vergelijken. Zo is het zowel in de software Buildsoft Diamonds als in SCIA Engineer mogelijk kolomverlies te simuleren op eenvoudige structuren. Wanneer er evenwel gekeken wordt naar complexere structuren en de hiermee samenhangende noodzaak tot automatisatie van kolomverlies, is de software Buildsoft Diamonds niet geschikt. Hierdoor wordt er geopteerd om het onderzoek uit te voeren met de software SCIA Engineer.

Aan de hand van enkele scripts opgesteld in PyCharm worden structuren opgesteld en aangepast. Deze structuren kunnen vervolgens uitgerekend worden op de achtergrond aan de hand van de ESA_XML-tool die SCIA Engineer aanbiedt. Gedurende dit onderzoek zijn twee verschillende algoritmen opgesteld. Enerzijds een algoritme dat in staat is om zelf een symmetrische raamwerkstructuur te genereren en hierop kolomverlies toe te passen. Anderzijds een algoritme dat kolomverlies toepast op door de gebruiker aangeleverde structuren. Het onderzoek valideert het algoritme door de resultaten van de simulatie van kolomverlies van het algoritme te vergelijken met een theoretisch voorbeeld.

Abstract

Robustness checks for column losses are very complex and time-consuming. Therefore, this research tries to automate the calculation for column loss. In order to make this possible, the possibilities of multiple commercial software needed to be researched. This was investigated by manually simulating column loss on a simple structure and comparing the results. Both Buildsoft Diamonds and SCIA Engineer offer the possibility to simulate column loss on simple structures. However, when looking at more complex structures and the automation of column loss, the Buildsoft Diamonds software is not suitable. Therefore, it was decided to conduct the research with the software SCIA Engineer.

Through a few scripts written in PyCharm, structures are generated and customized. These structures can then be calculated in the background using the ESA_XML tool offered by SCIA Engineer. During this research, two different algorithms are created. On the one hand, an algorithm that is able to generate a symmetrical truss structure and apply column loss to it. On the other hand, an algorithm that applies column loss to structures provided by the user. The research validates the algorithm by comparing the results of the simulation of column loss of both the algorithm and a theoretical example.

1 Inleiding

Structuren zijn onderhevig aan allerhande belastingen. Deze belastingen kunnen onderverdeeld worden in permanente belastingen, variabele belastingen en accidentele belastingen. Permanente en variabele belastingen zijn belastingen die zich in de structuur begeven, zoals het eigengewicht van de structuur, de onderhoudslast en de belastingen door het interieur van de structuur. Accidentele belastingen zijn belastingen zoals natuurrampen, ontploffingen of terroristische aanslagen.

De doelstelling van deze masterproef is het opstellen van een algoritme dat kolomverlies simuleert voor structuren op basis van de huidige normen op vlak van robuustheid. Er dient dus een algoritme opgesteld te worden dat kolomverlies simuleert voor iedere kolom binnen een structuur. Alvorens dit algoritme opgesteld kan worden, dient er onderzoek gedaan te worden naar de mogelijkheden van de simulatie van kolomverlies in verschillende commerciële analysesoftware. Zo komen de software Buildsoft Diamonds en SCIA Engineer aan bod in het vooronderzoek. In het vooronderzoek wordt bepaald welke analysesoftware verder gebruikt zal worden voor dit onderzoek. Er dient niet alleen rekening gehouden te worden met de mogelijkheden omtrent kolomverlies, maar ook met de mogelijkheid tot automatisatie van kolomverlies. Het vooronderzoek is uitgewerkt in hoofdstuk 6.

Gedurende dit onderzoek worden twee algoritmen opgesteld aan de hand van het programma PyCharm. Het eerste algoritme genereert een raamwerkstructuur op basis van inputs die de gebruiker aanlevert. Het algoritme maakt gebruik van een XML-bestand waarin het alle elementen van de structuur wegschrijft. Dit bestand is de basis om kolomverlies voor iedere kolom te simuleren. Het tweede algoritme maakt gebruik van een model dat de gebruiker aanlevert. Beide algoritme voeren de simulatie van kolomverlies op de structuren uit en berekenen de resultaten van de verscheidene modellen. De resultaten van deze analyses zijn terug te vinden in een tekstbestand als engineering rapporten in PDF. Beide algoritmen worden verduidelijkt in respectievelijk paragraaf 7.2 en 7.3.

Tenslotte zullen de resultaten van de simulatie van kolomverlies door het opgestelde algoritme op een specifieke structuur vergeleken worden met de resultaten van een theoretisch voorbeeld [13]. Voor deze vergelijking te realiseren wordt gebruik gemaakt van het tweede algoritme dat opgesteld wordt in dit onderzoek. De vergelijking van de resultaten van het algoritme met een theoretisch voorbeeld is beschreven in hoofdstuk 8.

2 Progressieve instorting

In volgende paragrafen van deze masterproef wordt eerst het begrip progressieve instorting besproken. Zo komen ook recente en bekende voorbeelden van progressieve instorting aan bod. Nadien zal de kans op optreden van progressieve instorting toegelicht worden.

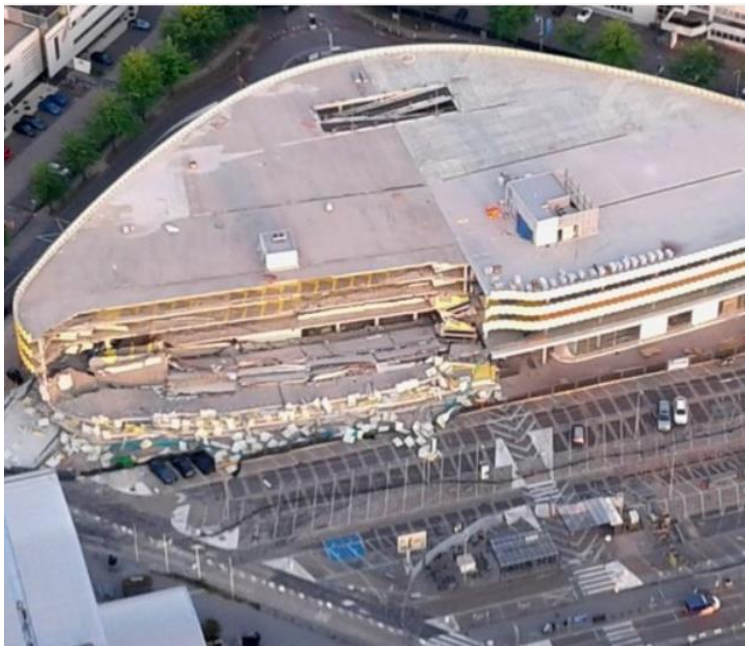
2.1 Het begrip progressieve instorting

Het begrip progressieve instorting heeft geen eenduidige definitie, aangezien het begrip geen duidelijke afbakening kent. Progressieve instorting kan omschreven worden als het lokaal bezwijken van een constructieonderdeel die een instorting veroorzaakt die niet evenredig is met de oorzaak ervan. Een voorbeeld van lokaal bezwijken is het falen van een kolom/constructieonderdeel omwille van een accidentele belasting. Een accidentele belasting is bijvoorbeeld een explosie of een stootbelasting omwille van een aanrijding van het constructieonderdeel. Het is echter zo dat de grootte van een accidentele belasting moeilijk in te schatten is. Daardoor is het dus zeer moeilijk om te voorspellen hoe de structuur gaat reageren op dergelijke belasting. [1]

Progressieve instorting is samenhangend met de robuustheid van een constructie. Zo is robuustheid het vermogen van een constructie om weerstand te bieden tegen onvoorziene gebeurtenissen. Voorbeelden van onvoorziene gebeurtenissen zijn brand, ontploffingen en accidentele stootbelastingen. Robuustheid is dus een essentiële eigenschap van constructies. Het gaat vaak gepaard met de ductiliteit en continuïteit van de constructieonderdelen alsook de aanwezigheid van alternatieve draagwegen in de constructie. Indien een constructie een voldoende robuustheid bevat, vermindert de kans op progressieve instorting. In dergelijke constructies komt er dus bijna nooit schade voor die disproportioneel is ten opzichte van de oorzaak. [1]

Zoals hierboven vermeld wordt, kan een onderscheid gemaakt worden in lokaal bezwijken van de structuur op basis van de gevolgen ervan. Zo is er enerzijds het lokaal bezwijken waarbij de schade beperkt blijft tot het constructieonderdeel zelf en zijn directe omgeving, anderzijds het lokaal bezwijken waar een belangrijk deel van de structuur schade lijdt. In het geval waarbij een belangrijk deel van de structuur schade lijdt, wordt er gesproken over progressieve instorting. Bekende voorbeelden van progressieve instorting zijn de instorting van het appartementsgebouw Ronan point en een parkeergarage in Eindhoven. Beide voorbeelden worden respectievelijk voorgesteld in Figuur 1 en Figuur 2.

Figuur 1: Ronan Point appartementsgebouw [2]



Figuur 2: Parkeergarage Eindhoven [3]

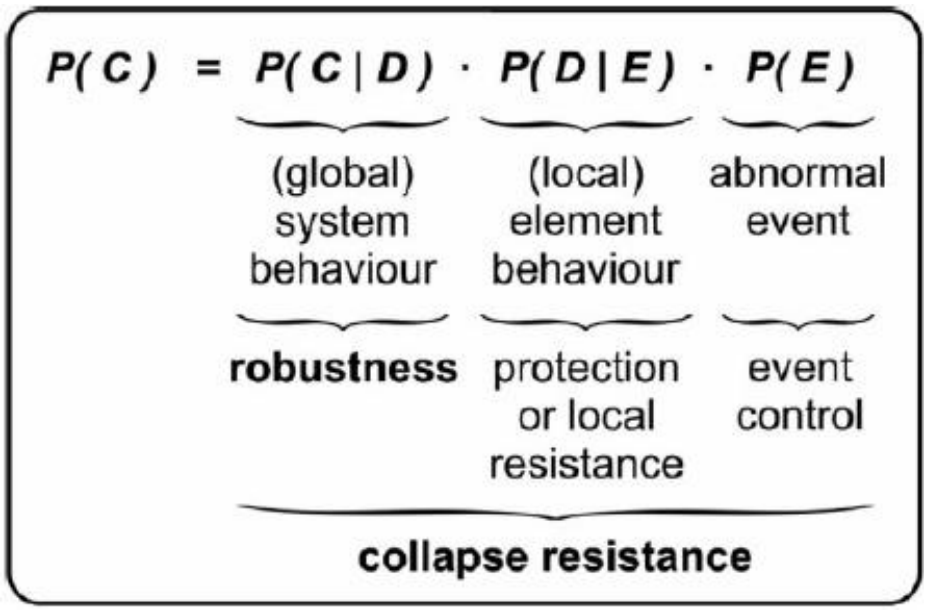
Een ander, meer recent voorbeeld van progressieve instorting is de instorting van een loods in de Antwerpse haven op 21 september 2021. De instorting van de loods is een gevolg van een aanrijding van een heftruck tegen een kolom. Het aanrijden van de kolom resulteerde in het totale verlies van deze kolom en uiteindelijk tot het instortten van 1600m² oppervlakte loods. De instorting van de loods in de Antwerpse haven is weergegeven in Figuur 3. [4]

Figuur 3: Ingestorte loods in Antwerpse haven [4]

2.2 Kans op Progressieve instorting

Zoals reeds vermeld werd, is het bijna onmogelijk om de grootte en de gevolgen van een accidentele belasting in te schatten. Dit is niet anders bij de kans dat dergelijke belasting optreedt. Het is echter wel mogelijk om een goede benadering te verkrijgen door gebruik te maken van kansberekeningen. Zo is bijvoorbeeld de kans dat een aanrijding van een kolom optreedt waarbij kolomverlies zich voordoet groter in industriehallen waar voertuigen rondrijden ten opzichte van in een woning. [5]

De kans op progressieve instorting, *P(C)*, kan worden onderverdeeld in verschillende deelkansen. Een overzicht van deze deelkansen is voorgesteld in Figuur 4. Een eerste deelkans, *P(E)*, is het optreden van een accidentele belasting. Het is niet mogelijk deze kans te bepalen. Daarnaast is er ook de kans dat er lokale schade optreedt als gevolg van een accidentele belasting *E*. Deze kans wordt voorgesteld door *P(D|E)*. De laatste deelkans, *P(C|D)*, bestaat uit de kans dat een constructie instort als er sprake is van lokale schade. Aan de hand van deze drie deelkansen kan de kans op progressieve instorting bepaald worden. [1]



Figuur 4: kansbepaling van progressieve instorting [1]

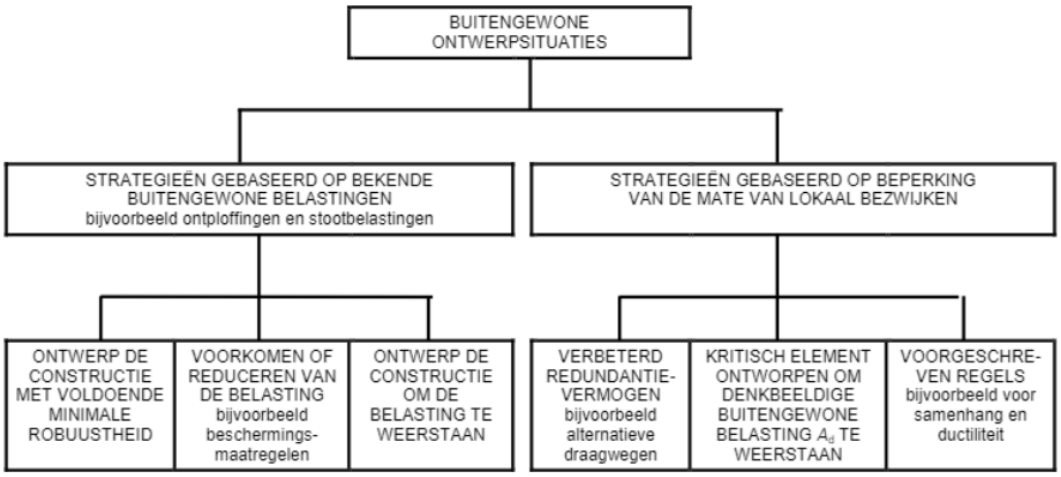
Op basis van de drie deelkansen uit Figuur 4 kunnen drie strategieën opgesteld worden die progressieve instorting trachten te vermijden. Deze drie strategieën worden zowel in de Europese norm NBN EN 1991-1-7 als in deze masterproef als volgend benoemd: voorkomen of reduceren van de belasting, kritisch element ontwerpen om denkbeeldige buitengewone belasting A_d te weerstaan en als laatste enkele voorgeschreven regels. De eerste strategie bestaat eruit om de nodige maatregelen te treffen om het optreden van een accidentele belasting te vermijden. Dit is echter geen gemakkelijke opgave aangezien de aard van de belasting onbekend is. Ten tweede kan een structuuronderdeel ontworpen worden zodat het onderdeel niet lokaal kan bezwijken. Ten laatste kan men proberen het gedrag van een constructie bij lokaal bezwijken te sturen. [1]

3 Studie van NBN EN 1991-1-7 - Buitengewone ontwerpsituaties

In dit hoofdstuk van deze masterproef wordt de Europese norm NBN EN 1991-1-7 onderzocht voor buitengewone ontwerpsituaties. Dit deel kan onderverdeeld worden in enerzijds de strategieën voor bekende buitengewone belasting en anderzijds strategieën voor de beperking van de mate van lokaal bezwijken.

3.1 Strategieën voor bekende buitengewone belastingen

Deze paragraaf bespreekt de strategieën, die beschreven zijn in NBN EN 1991-1-7, voor bekende buitengewone belastingen. Linksonder in Figuur 5 kunnen de drie strategieën, gebaseerd op bekende buitengewone belastingen, teruggevonden worden. Zowel ontploffingen als stootbelastingen maken deel uit van deze buitengewone belastingen. [6]



Figuur 5: Strategieën voor buitengewone ontwerpsituaties [6]

Naast de maatregelen en richtlijnen die toegepast zijn in de drie strategieën gebaseerd op bekende buitengewone belastingen, zijn er enkele richtlijnen die niet specifiek tot een strategie behoren. Desalniettemin moet er rekening worden gehouden met deze richtlijnen tijdens het ontwerp van een constructie. [6]

Eén van deze richtlijnen is dat de belasting afhankelijk moet zijn van enkele factoren. Volgens [6] is een eerste factor de genomen maatregel(en) om een buitengewone belasting te voorkomen of de ernst ervan te verminderen. Naast deze eerste factor zijn ook de kans dat een bekende buitengewone belasting optreedt, net zoals de gevolgen van bezwijken door dergelijke belasting, alsook de waarneming door het publiek en het aanvaardbaar risiconiveau factoren waarvan de belasting afhankelijk moet zijn. Het risiconiveau van een constructie kan worden geassocieerd met de mogelijke gevolgen die buitengewone belastingen met zich mee kunnen brengen. Dit niveau is afhankelijk van verschillende factoren. Het mogelijk aantal slachtoffers, de economische gevolgen of de kosten van veiligheidsmaatregelen zijn enkele voorbeelden van die factoren. Indien het risiconiveau niet kan worden aanvaard, zijn aanvullende maatregelen verreist. [6]

Een volgende richtlijn is dat een constructie aanvaard kan worden onder bepaalde voorwaarden, zelfs als lokaal bezwijken van een constructieonderdeel optreedt. Ten eerste moet er de mogelijkheid zijn om de noodzakelijke maatregelen te treffen. Daarnaast mag de stabiliteit van de constructie niet in het gedrang komen bij lokaal bezwijken. Tenslotte moet het draagvermogen van de constructie steeds gegarandeerd blijven. [6]

De overige twee richtlijnen houden in dat buitengewone belastingen in rekening moeten worden gebracht met permanente en andere veranderlijke belastingen. Dit gebeurt aan de hand van veiligheidsfactoren. De andere richtlijn stelt dat tijdens het ontwerpproces rekening moet worden gehouden met de veiligheid van de constructie na progressieve instorting. Hierbij behoort ook de beschouwing van de progressieve instorting. In de volgende paragrafen worden de verschillende ontwerpstrategieën besproken. [6]

3.1.1 Ontwerp de constructie met voldoende minimale robuustheid

De eerste strategie, gebaseerd op bekende buitengewone belastingen, bestaat eruit om een constructie zo te ontwerpen zodat deze een voldoende minimale robuustheid bevat. Deze strategie kan toegepast worden door gebruik te maken van één of meer van de volgende methoden. De eerste methode houdt in dat de onderdelen, waarvan de stabiliteit van de constructie afhangt, als kritische elementen ontworpen moeten worden. Op deze manier is de kans groter dat de constructie intact blijft na lokaal bezwijken van een constructieonderdeel. Een tweede methode bestaat eruit dat de constructie en de onderdelen ervan een voldoende ductiliteit moeten bevatten zodat vormveranderingsenergie geabsorbeerd kan worden zonder dat dit leidt tot breuk. Doormiddel van het ontwerp van de constructieonderdelen en de juiste materiaalkeuze wordt beoogd deze ductiliteit te behalen. De laatste methode is gericht om een voldoende redundantie in de constructie op te nemen tijdens het ontwerp. Dit betekent concreet dat er alternatieve draagwegen aanwezig moeten zijn binnen de structuur voor de belasting na lokaal bezwijken van een constructieonderdeel. Indien dus aan deze voorwaarde voldaan is, zal lokaal bezwijken niet leiden tot progressieve instorting van de structuur. [6]

3.1.2 Voorkomen of reduceren van de belasting

De volgende strategie is om de constructie zo te ontwerpen dat de buitengewone belasting kan voorkomen of gereduceerd worden. In de Europese norm worden specifiek twee maatregelen opgesomd. Ten eerste wordt er getracht het optreden van de belasting te voorkomen. Ten tweede dient de grootte van de belasting tot een aanvaardbaar niveau gereduceerd te worden. De tweede maatregel houdt in dat de gevolgen van een buitengewone belasting op de structuur beperkt worden. [6]

3.1.3 Ontwerp de constructie om de belastingen te weerstaan

De strategie om het ontwerp van de constructie te baseren op het weerstaan van belastingen is een zeer ruim en open concept zonder duidelijke afbakening. In de norm NBN EN 1991-1-7 wordt hier niet verder op ingegaan. Bijgevolg wordt dit onderdeel ook niet verder besproken in deze masterproef.

3.2 Strategieën voor de beperking van de mate van lokaal bezwijken

Waar de richtlijnen bij de strategieën gebaseerd op bekende buitengewone belasting al niet concreet en eenduidig waren, zijn deze voor de strategieën voor de beperking van de mate van lokaal bezwijken niet anders. Het is zelfs zo dat de informatie hieromtrent nog geringer is. Zoals eerder vermeld in paragraaf 3.1 kan een constructie, waarin lokaal bezwijken optreedt, toch aanvaard worden. Dit is wel enkel het geval als de constructie voldoet aan de opgestelde voorwaarden. Deze voorwaarden zijn terug te vinden in paragraaf 3.1.

Rechtsonder in Figuur 5 zijn de drie strategieën gebaseerd op de beperking van de mate van lokaal bezwijken weergegeven. De drie strategieën zullen hieronder besproken worden.

3.2.1 Verbeterd redundantievermogen

Tijdens het ontwerp van een constructie zullen structuuronderdelen, waarvan de stabiliteit van de constructie afhangt, als kritische elementen beschouwd worden. Zodoende dat de constructie een buitengewone belasting zou kunnen weerstaan. [6]

3.2.2 Kritisch element ontwerpen om denkbeeldige buitengewone belasting te weerstaan

De constructie moet zo ontworpen zijn dat de stabiliteit van de hele constructie of een deel ervan niet in gevaar komt indien er lokaal bezwijken van een constructieonderdeel optreedt. [6]

3.2.3 Voorgeschreven regels

De laatste mogelijkheid bestaat eruit om voorgeschreven ontwerp/detailleringsregels toe te passen. Deze regels zorgen voor een aanvaardbare robuustheid van de constructie. Enkele voorbeeld van deze regels zijn het gebruik van trekbanden of door materialen te kiezen die een voldoende ductiliteit bevatten. [6]

3.3 Gevolgklassen

De strategieën voor buitengewone ontwerpsituaties die eerder besproken zijn, zijn gebaseerd op gevolgklassen. Doorheen de verschillende normen zijn er verschillende benamingen voor iedere gevolgklasse. Zo geeft Figuur 6 een overzicht van deze verschillende benamingen van de gevolgklassen en de bijhorende gebouwtypes. [6]

Gebouwtypes	EN 1990 (tabel B.1)	EN 1991-1-7 (tabel A.1)	EN 1998-1 (tabel 4.3)
Agrarische gebouwen of gewoonlijk niet-bezette gebouwen	CC1	CCA1	I
Eengezinswoningen of gebouwen ≤ 2 niveaus en ≤ 100 m² in totaal	CC2	CCA1	II
Gebouwen waarbij de gevolgen van de instorting middelmatig zijn en welke niet behoren tot de andere categorieën, met een maximale gelijktijdige bezetting ≤ 500 personen	CC2	CCA2a	II
Gebouwen waarbij de gevolgen van de instorting belangrijk zijn (school, vergaderzaal, cultureel centrum, handelscentrum), met in totaal ≤ 15 niveaus en een maximale gelijktijdige bezetting ≤ 5000 personen	CC2	CCA2b	III
Gebouwen waarbij de gevolgen van de instorting belangrijk zijn (school, vergaderzaal, cultureel centrum, ...) > 15 niveaus	CC2	CCA3	III
Gebouwen waarbij de gevolgen van gebreken zeer belangrijk zouden zijn met een maximale gelijktijdige bezetting > 5000 personen (bijv. concertzalen, tribunes)	CC3	CCA3	III
Gebouwen waarin zich gevaarlijke stoffen of producten bevinden	CC3	CCA3	IV
Elektriciteitscentrales, ziekenhuizen, kazernes, ... en andere gebouwen van vitaal belang voor de burgerbescherming	CC3	CCA3	IV

Figuur 6: Definitie van de gevolgklassen van gebouwen [7]

Deze masterproef kadert voornamelijk de Europese norm uit 1991. Er zal dan ook gebruik gemaakt worden van de benamingen gebruikt in deze norm. Gevolgklasse CCA1 wordt toegepast voor gebouwen waarbij de gevolgen van progressieve instorting enkel van materiele aard en beperkte menselijke aard zijn. Bij deze gevolgklasse is het niet vereist om specifiek buitengewone belastingen te controleren aangezien er bij bezwijken weinig gevolgen zijn voor de structuur. Een extra controle op buitengewone belastingen kan nog uitgevoerd worden om te verzekeren dat voldaan is aan de regels opgesteld door NBN EN 1991 tot en met NBN EN 1999 omtrent robuustheid. Gevolgklasse CCA2 kan onderverdeeld worden in twee groepen, gebouwen met laag risico en gebouwen met hoog risico. Op basis van de situatie waarin de structuur zich bevindt, wordt aangeraden enerzijds vereenvoudigde berekeningen te gebruiken en anderzijds voorgeschreven ontwerp/detailleringsregels toe te passen. De laatste gevolgklasse is CCA3. Deze gevolgklasse bevat de gebouwen waarbij de gevolgen van bezwijken groot zijn. Om het vereiste betrouwbaarheidsniveau en de nodige berekeningen te bepalen is het verreist een onderzoek naar de specifieke structuur uit te voeren. Een risicoanalyse kan al dan niet uitgevoerd worden. Bovendien kan gebruik gemaakt worden van dynamische berekening en niet-lineaire modellen. [6]

De benamingen van de gevolgklassen van NBN EN 1990 en NBN EN 1991-1-7 komen in grote maten overeen. Zoals hierboven vermeldt is er een verschil in gevolgklasse 2. Deze gevolgklasse kan nog onderverdeeld worden in een lage risicogroep en een hoge risicogroep. Respectievelijk staat CCA2a voor de lage risicogroep en CCA2b voor de hoge risicogroep. [6]

3.4 Trekbanden

Om zeker te zijn dat een constructie geen progressieve instorting ondergaat bij lokaal bezwijken, is er in bijlage A van NBN EN 1991-1-7 een methode ontwikkeld die gebruik maakt van trekbanden. Hierin kan een onderscheid gemaakt worden tussen verticale en horizontale trekbanden. In het kader van dit onderzoek zijn de verticale trekbanden minder relevant en zullen deze dan ook niet besproken worden. Binnen de horizontale trekbanden is er ook een onderscheid tussen constructies met dragende kolommen en constructies met dragende wanden. [6]

3.4.1 Horizontale trekbanden voor constructies met kolommen

Horizontale trekbanden dienen zowel toegepast te worden langs de omtrek van iedere vloer en van ieder dak, alsook binnenin de constructie. De trekbanden die binnenin de constructie zijn toegepast kunnen worden opgedeeld in twee onderling loodrechte richtingen. Deze trekbanden mogen vervaardigd zijn uit gewalste staalprofielen, wapeningstaven of wapeningsnetten voor betonnen vloerplaten. Voor staal-betonvloeren worden deze vervaardigd uit geprofileerde staalplaten. Dit is enkel toegelaten wanneer de vloer direct bevestigd wordt aan de stalen liggers door gebruik te maken van afschuifverbindingsmiddelen. Een combinatie van materialen is ook mogelijk. [6]

Voor de berekening van de trekbanden wordt er gebruik gemaakt van de uiterste grenstoestand. Er wordt een onderscheid gemaakt tussen interne en externe trekbanden. De interne trekbanden zijn gelegen binnen de constructie en de externe aan de rand van de constructie. De interne trekbanden dienen een trekkracht T_i te kunnen weerstaan en de externe trekbanden een trekkracht T_p. Formule 3.1 en formule 3.2 geven de formules weer van deze trekkrachten. [6]

$$T_i = 0{,}8(g_k + \Psi q_k)s \cdot L \quad (3.1)$$

$$T_p = 0{,}4(g_k + \Psi q_k)s \cdot L \quad (3.2)$$

Waarbij: s = de afstand tussen de verschillende trekbanden
L = de lengte van de trekband
Ψ = de combinatiefactor van toepassing voor de buitengewone ontwerpsituatie

Zowel de interne als de externe trekbanden moeten kunnen weerstaan aan een minimum trekkracht die 75 kN bedraagt. Deze waarden van de trekkracht waaraan de trekbanden aan moeten voldoen kan vergeleken worden met de waarden die zijn voorgeschreven in de norm NBN EN 1992-1-1. Deze norm maakt gebruik van een andere formule voor het berekenen van trekbanden. Formule 3.3 geeft de formule voor de externe trekbanden. [6], [8]

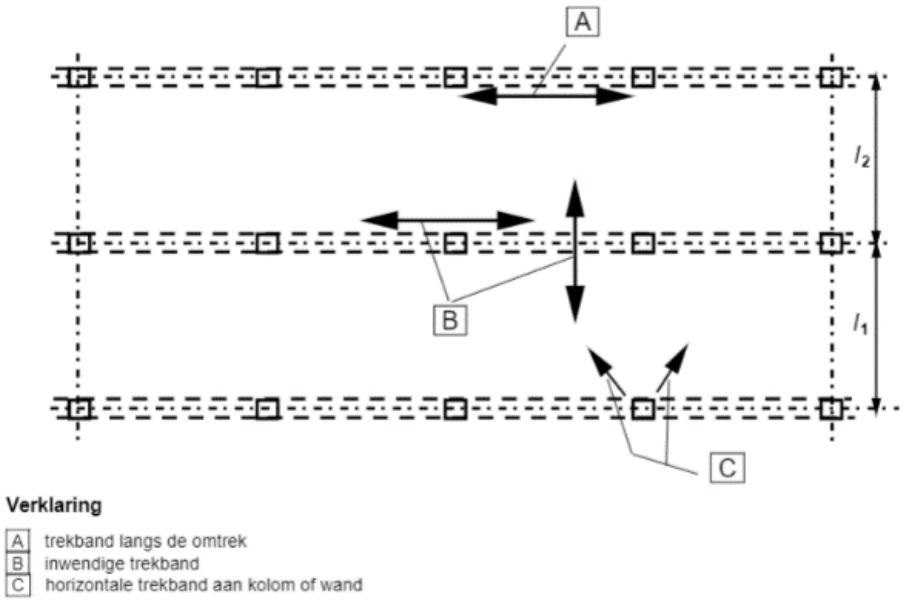
$$F_{tie} = l_i \cdot q_1 \leq Q_2 \quad (3.3)$$

Waarbij: F_{tie} = de trekbandkracht
l_i = de lengte van de eindoverspanning

De waarde van q_1 en Q_2 uit formule 3.3 worden bepaald aan de hand van de nationale bijlage in [8]. De aanbevolen waarden bedragen respectievelijk 10 kN/m voor q_1 en 70 kN voor Q_2. De maximale waarde die een trekband moet kunnen opnemen is hier dus lager dan in de norm NBN EN 1991-1-7. Formule 3.4 geeft weer hoe de interne trekbanden moeten worden berekend. [8]

$$F_{tie} = ((l_1 + l_2)/2) \cdot q_3 \leq Q_4 \qquad (3.4)$$

Waarbij: l_1, l_2 = de overspanningen van de vloerplaat weergegeven door Figuur 7



Figuur 7: Trekbanden volgens NBN EN 1992-1-1 [8]

De waarde van q_3 en Q_4 uit formule 3.4 zijn terug te vinden in de nationale bijlage in [8]. Ze bedragen 20 kN/m voor q_3 en 70 kN voor Q_4. [8]

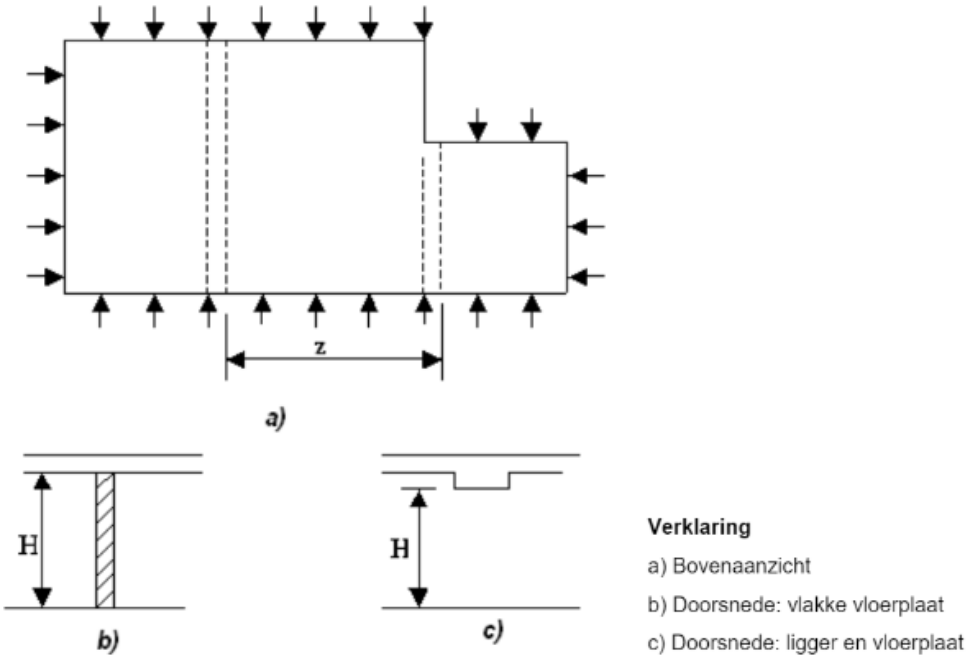
3.4.2 Horizontale trekbanden voor constructies met dragende wanden

In deze paragraaf wordt een onderscheid gemaakt op basis van de gevolgklasse die weergegeven zijn in Figuur 6. Voor constructies uit gevolgklasse 2 met een lage risicogroep dient er geen gebruik gemaakt te worden van trekbanden. Echter dient men er wel voor te zorgen dat er een voldoende robuustheid aanwezig is, alsook een geschikte verankering van de vloeren aan de wand. Voor constructies met een gevolgklasse 2 met een hoge risicogroep dienen er trekbanden voorzien te worden. Deze worden zowel inwendig orthogonaal verspreid over de vloer, als extern aan de omtrek van het vloer- of daksysteem. De externe trekbanden dienen geplaatst te worden maximaal 1,2 m van de rand van de constructie. Formule 3.5 geeft de trekkracht T_i die de interne trekbanden moet kunnen weerstaan. Figuur 8 geeft een verduidelijking van de factoren *H* en *z* weer. [6]

$$T_i = \frac{F_t \cdot (g_k + \Psi q_k)}{7.5} \frac{z}{5} \qquad (3.5)$$

Waarbij: F_t = de kleinste waarde van:
60 kN/m
$20 + 4 \cdot n_s$ (met n_s = aantal bouwlagen)

z = de kleinste waarde van
5 keer de vrije bouwlaaghoogte *H*
De grootste overspanning in de richting van de trekband



Figuur 8: Verduidelijking factoren van formule 3.3 [6]

4 Studie van UFC 4-023-03

Naast de Europese norm die eerder besproken werd, heeft ook de Amerikaanse norm UFC 4-023-03 informatie over de ontwerpvoorschriften voor structuren. Specifiek wordt in deze norm vermeld dat structuren die drie of meer verdiepingen bevatten, een groter risico hebben op progressieve instorting. Daardoor wordt voorgeschreven dat alle structuren moeten voldoen aan deze norm. [9]

4.1 Risicocategorieën

Zowel nieuwe als bestaande structuren worden ontworpen of gecontroleerd volgens ontwerpvoorwaarden. Elke constructie behoort tot één van vier risicocategorieën. De risicocategorieën maken deel uit van de ontwerpvoorwaarden. Specifiek worden zij gebruikt om het niveau te definiëren waarop een structuur ontworpen moet worden of ontworpen is geweest om progressieve instorting te weerstaan. Concreet betekent dit dat de risicocategorie bepaalt welke ontwerpstrategie moet toegepast worden om de structuur te ontwerpen. Welke risicocategorieën er bestaan en welke ontwerpstrategie bij welke risicocategorie behoort is samengevat in Tabel 1. [9], [10]

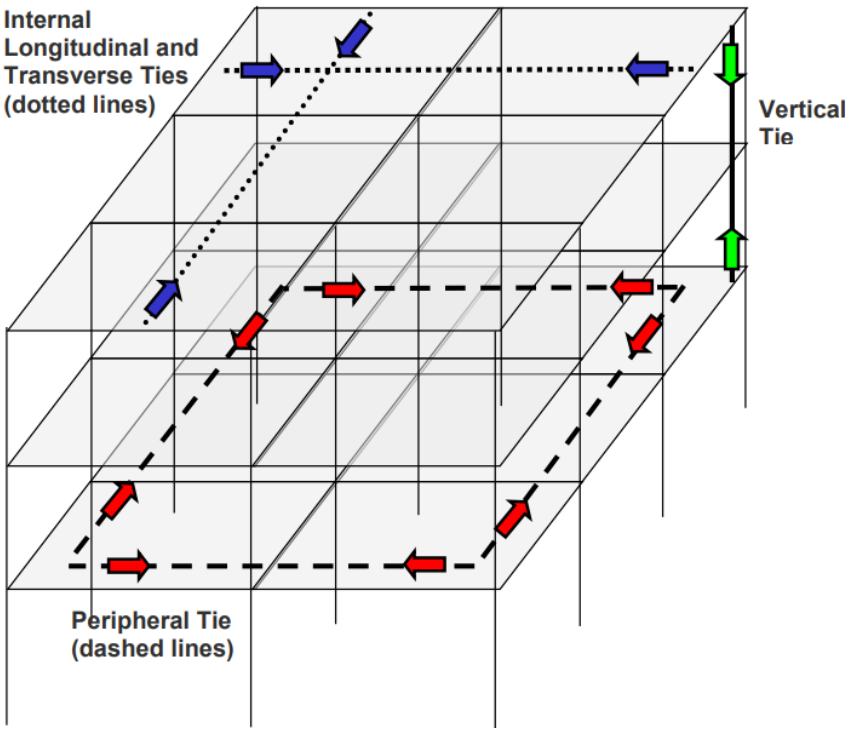
Tabel 1: Soorten risicocategorieën [9], [10]

Risico-categorie	Aard van de bezetting	Ontwerpvoorwaarden
I	Gebouwen en andere constructies die in geval van falen een gering gevaar voor mensenlevens vormen	Geen specifieke ontwerpvoorwaarden
II	Gebouwen en andere bouwwerken, met uitzondering van die worden genoemd in risicocategorie I, III en IV	Optie 1: kettingkrachten Optie 2: alternatieve pad methode
III	Gebouwen en andere constructies die een aanzienlijk gevaar voor mensenlevens of een aanzienlijk economisch verlies betekenen bij falen	Alternatieve pad methode en verbeterde lokale weerstand voor alle kolommen en wanden aanwezig op de eerste verdieping
IV	Gebouwen en andere constructies die ontworpen zijn als essentiële faciliteiten	Alle drie de bovenstaande ontwerpvoorwaarden

Over de verschillende risicocategorieën zijn er drie mogelijke ontwerpstrategieën. Ten eerste is er de methode van de kettingkrachten. Deze methode wordt verder in deze masterproef toegelicht. Ook zal deze methode gebruikt worden om een vergelijking te maken tussen de Amerikaanse en Europese normen. Vervolgens zijn er nog de alternatieve pad methode en de verbeterde lokale weerstand methode. Deze zullen niet verder toegelicht worden. [9]

4.2 Methode van de kettingkrachten

Bij de methode van de kettingkrachten wordt verondersteld dat de structuur mechanisch aan elkaar geknoopt is, waardoor de continuïteit en de ductiliteit en de ontwikkeling van alternatieve draagwegen voor de belasting worden verbeterd. Binnen constructies kunnen kettingkrachten geleverd worden door bestaande constructieonderdelen. Zo zijn er drie verschillende horizontale trekbanden die voorzien moeten worden in een constructie. Namelijk, Longitudinale (in de lengterichting), transversale (dwars) en perifere (langs de rand van de constructie). Verticale trekbanden zijn verreist in kolommen en dragende muren. De verticale trekbanden zullen niet verder besproken worden in deze masterproef. Een overzicht van de verschillende trekbanden in een structuur is weergegeven in Figuur 9. [9]



Figuur 9: Overzicht verschillende soorten trekbanden [9]

Indien de constructiedelen en de verbindingen niet in staat zijn om de vereiste longitudinale, transversale en perifere krachten op te vangen bij een hoekrotatie van 0,20 radialen oftewel 11,3 graden zullen deze krachten opgevangen moeten worden door het vloer- en daksysteem. Vloer- en daksystemen die beschikbaar zijn voor deze toepassing zijn ter plaatse gestort beton, composietvloeren en geprefabriceerde betonnen vloerplanken met betonnen deklaag. Ook andere vloer- en daksystemen zijn toegelaten indien het vermogen om de vereiste treksterkte te dragen bij een hoekrotatie van 0,20 radialen oftewel 11,3 graden kan aangetoond worden. Ook is het belangrijk dat een positieve mechanische verankering of verbinding tussen de constructiedelen die kettingkrachten bezitten en de andere constructiedelen voorzien wordt. Concreet wil dit zeggen dat het kettingkrachten model zo opgebouwd moet worden zodat er geen mogelijkheid bestaat dat delen van een constructie afbreken en op de onderliggende vloer vallen. [9]

4.2.1 Longitudinale en transversale trekbanden

Indien de structuur een geschikt vloer- of daksysteem bevat, zal een kracht met een longitudinale en transversale component steeds orthogonaal ontbonden kunnen worden. De uiteinden van longitudinale en transversale trekbanden moeten steeds verankerd zijn aan de perifere trekbanden. Aangezien in dit onderzoek gewerkt wordt met 2D-structuren zullen de trekbanden afzonderlijk bekeken worden. Voor overspanningen in één richting, de longitudinale richting, mag de afstand tussen twee trekbanden in dezelfde richting niet groter zijn dan 0,2 L_L. De afkortingen L_L en L_T staan respectievelijk voor de grootste afstand tussen de twee centra van de kolommen in de longitudinale en transversale richting. In de andere richting, de transversale richting is de voorwaarde dat de waarde L_T 5 keer h_w bedraagt, wat voor de hoogte van een verdieping staat. Met dergelijke voorwaarden wordt in de Amerikaanse norm volgende formule gesteld die de mogelijkheid geeft om de vereiste trekkracht F_T in zowel de longitudinale als in de transversale richting te berekenen. De eenheid van deze kracht bedraagt kN/m. De term L_1, die terug te vinden is in formule 4.1, is afhankelijk van de richting. Zo bedraagt deze in de longitudinale richting L_L en in de transversale richting L_T. In tegenstelling tot de Europese norm, geldt in de Amerikaanse norm eenzelfde formule voor zowel constructies met dragende kolommen als dragende wanden. [9]

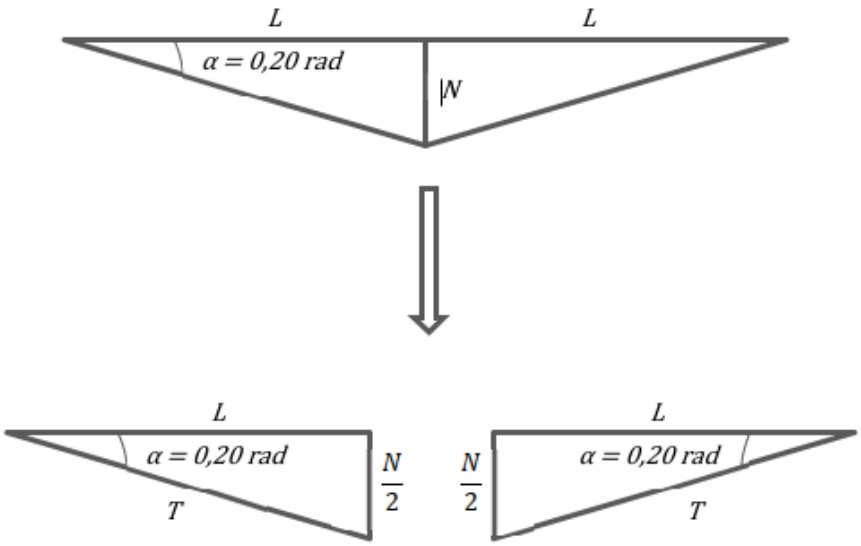
$$F_i = 3\, w_f\, L_1 \qquad (4.1)$$

Waarbij: w_f = de belasting op de vloer in kN/m²
L_1 = L_L → in longitudinale richting
= L_T → in transversale richting

Naast de belasting op de vloer en de afstand in beide richtingen is in de formule ook de factor drie terug te vinden. Deze factor is op volgende manier bepaald.

Een constructie bestaande uit drie kolommen en twee liggers is de basis voor het onderzoek van deze factor. Om vast te stellen hoeveel deze constante bedraagt, zal een accidentele belasting aangrijpen op de constructie die lokaal bezwijken veroorzaakt. Concreet betekent dit het verlies van de middelste kolom. De constructie voor en na het kolomverlies kan vereenvoudigd worden tot enerzijds een ligger op drie steunpunten en anderzijds een ligger op twee steunpunten. [5]

Omdat de middelste kolom instort en er dus kolomverlies optreedt op deze plaats is het belangrijk om de kracht te berekenen die deze kolom ondervond. Deze kracht is gelijk aan de reactiekracht van het middelste steunpunt in de vereenvoudigde voorstelling. De kracht is aangeduid met de letter *N* op Figuur 10. Door het kolomverlies zal de ligger een rotatiehoek ondergaan en zullen er trekkrachten optreden in de liggers. Deze trekkrachten zijn aangeduid met de letter *T* op Figuur 10. De optredende trekkrachten zullen de kracht die normaal opgevangen werd door de middelste kolom, de kracht *N*, afdragen naar de overige twee kolommen of steunpunten. [5]



Figuur 10: Deelsystemen van kettinglijnsysteem [5]

Zoals vermeld wordt in [5] kan het kettinglijnsysteem voorgesteld in Figuur 10 opgesplitst worden in twee deelsystemen. Indien de goniometrische meetkunde wordt toegepast op een van deze deelsystemen volgt formule 4.2. [5]

$$\sin(\alpha) = \frac{N/2}{T} \tag{4.2}$$

Aangezien de rotatiehoek α dermate klein is mag volgende vereenvoudiging toegepast worden, $\sin(\alpha) = \alpha$.

$$T = \frac{N}{2\alpha} \tag{4.3}$$

Eerder in deze masterproef werd aangehaald dat in de Amerikaanse norm 0,2 radialen aangenomen werd voor de rotatiehoek.

$$T = \frac{N}{2 \cdot 0{,}2} = 2{,}5\ N \tag{4.4}$$

De berekende factor bedraagt nu 2,5. Volgens [5] wordt deze factor afgerond om dynamische effecten in rekening te brengen. Dit verklaart de afwijking ten opzichte van de eerder vermelde factor drie.

4.2.2 Perifere trekbanden

Net zoals bij de longitudinale en transversale trekbanden zal een geschikt vloer- of daksysteem de krachten opnemen die normaal in de perifere trekbanden voorkomen. Ook kunnen deze krachten opgenomen worden door structuurelementen waarvan aangetoond kan worden dat deze de perifere kracht kunnen weerstaan als ze een hoekrotatie van 0,2 radialen oftewel 11,3 graden kunnen ondergaan. Dit soort trekbanden dient binnen één meter van de rand van de vloer of het dak geplaatst te worden. Om de kracht te berekenen die de perifere trekbanden moeten kunnen weerstaan wordt in de Amerikaanse norm volgende formule voorgesteld. [9]

$$F_p = 6\, w_f\, L_1\, L_p + 3\, W_c \qquad (4.5)$$

Waarbij:
- w_f = de belasting op de vloer in kN/m²
- W_c = 1,2 keer de belasting van de vloerafwerking over een lengte van L_1 in kN
- L_1 = trekbanden aan de rand van structuur → De grootste afstand tussen de centra van de kolommen
  = trekbanden aan openingen in de structuur → De lengte van de opening in die richting
- L_p = maximum afstand waar een trekband gelegen mag zijn ten opzichte van de rand van de structuur → 1 m

5 Vergelijking van NBN EN 1991-1-7 en UFC 4-023-03

Indien de Europese en Amerikaanse norm vergeleken worden, wordt duidelijk dat er enkele opmerkelijke verschillen zijn. In dit hoofdstuk worden deze verschillen verder toegelicht. Vooraleer de verschillen besproken worden, worden de belangrijkste formules nog kort herhaald. Enkel de constructies met dragende kolommen worden hier in acht genomen.

In de Europese norm, NBN EN 1991-1-7, zijn onderstaande formules voor de interne en externe trekbanden van kracht. [6]

Voor de interne trekbanden in een orthogonale richting geldt formule 3.1.

$$T_i = 0{,}8(g_k + \Psi q_k)s \cdot L \qquad (3.1)$$

Voor de externe trekbanden geldt formule 3.2.

$$T_p = 0{,}4(g_k + \Psi q_k)s \cdot L \qquad (3.2)$$

Voor zowel formule 3.1 als formule 3.2 geldt:

- s = de afstand tussen de verschillende trekbanden
- L = de lengte van de trekband
- Ψ = de combinatiefactor van toepassing voor de buitengewone ontwerpsituatie

In de Amerikaanse norm UFC 4-023-03 zijn onderstaande formules voor de interne en externe trekbanden van toepassing. [9]

Voor de interne trekbanden (Longitudinale en transversale trekbanden) geldt formule 4.1.

$$F_i = 3\, w_f\, L_1 \qquad (4.1)$$

Voor de externe trekbanden (perifere trekbanden) geldt formule 4.5.

$$F_p = 6\, w_f\, L_1\, L_p + 3\, W_c \qquad (4.5)$$

Waarbij:
- w_f = belasting op de vloer in kN/m²
- W_c = 1,2 keer de belasting van de vloerafwerking over een lengte van L_1 in kN
- L_1 = L_L → in longitudinale richting
  = L_T → in transversale richting

Zoals eerder vermeld werd, kan de toegelaten rotatie worden bepaald aan de hand van een kettinglijnsysteem. Door deze methode toe te passen op de formules van beide normen kunnen de verschillen worden aangetoond. Zo schrijft de Amerikaanse norm een toegelaten rotatie van 0,20 radialen of 11,3 graden voor. Op basis van deze toegelaten rotatie wordt een factor drie gevonden die geldt voor zowel de interne als externe trekbanden. [5], [9]

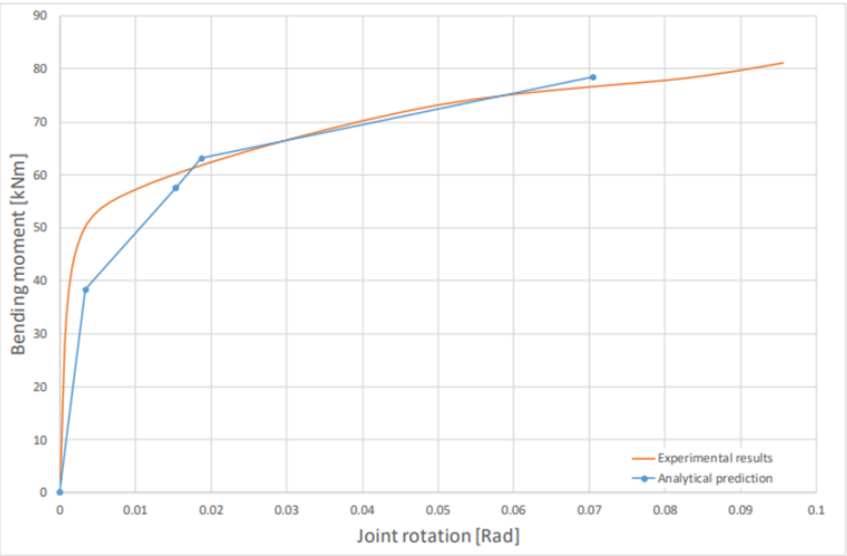
De Europese norm maakt wel een onderscheid tussen interne en externe trekbanden. Zo is in de formules voor de interne trekbanden een factor 0,8 terug te vinden. Met behulp van formule 5.1 kan vervolgens de toegelaten rotatie worden berekend. Hieruit volgt dat de toegelaten rotatie 0,31 radialen bedraagt, wat overeenkomt met een hoek van 17,8 graden. [5], [6]

$$\sin\alpha \approx \alpha = \frac{N}{4 \cdot 0{,}8 \cdot N} \tag{5.1}$$

Voor de externe trekbanden kan een factor 0,4 teruggevonden worden. Aan de hand van formule 5.2 wordt bepaald dat de factor 0,4 overeenkomt met een toegelaten rotatiehoek van 0,625 radialen of 35,8 graden. [5], [6]

$$\sin\alpha \approx \alpha = \frac{N}{4 * 0{,}4 * N} \tag{5.2}$$

Zowel voor de interne als externe trekbanden verschillen de waarden van factoren tussen de Europese en Amerikaanse norm beduidend. Zo kunnen de waarden van de toegelaten rotatiehoek van beide normen in perspectief geplaatst worden aan de hand van een onderzoek [11]. Dit onderzoek onderzoekt de ductiliteit van staal en composiet verbindingen in raamwerkstructuren aan de hand van experimenten. Figuur 11 toont de grafiek waarop het buigmoment in functie van de rotatiehoek, van het onderzoek weergegeven wordt. [11]



Figuur 11: Enkelzijdige verbinding met een verlengde eindplaatverbinding [11]

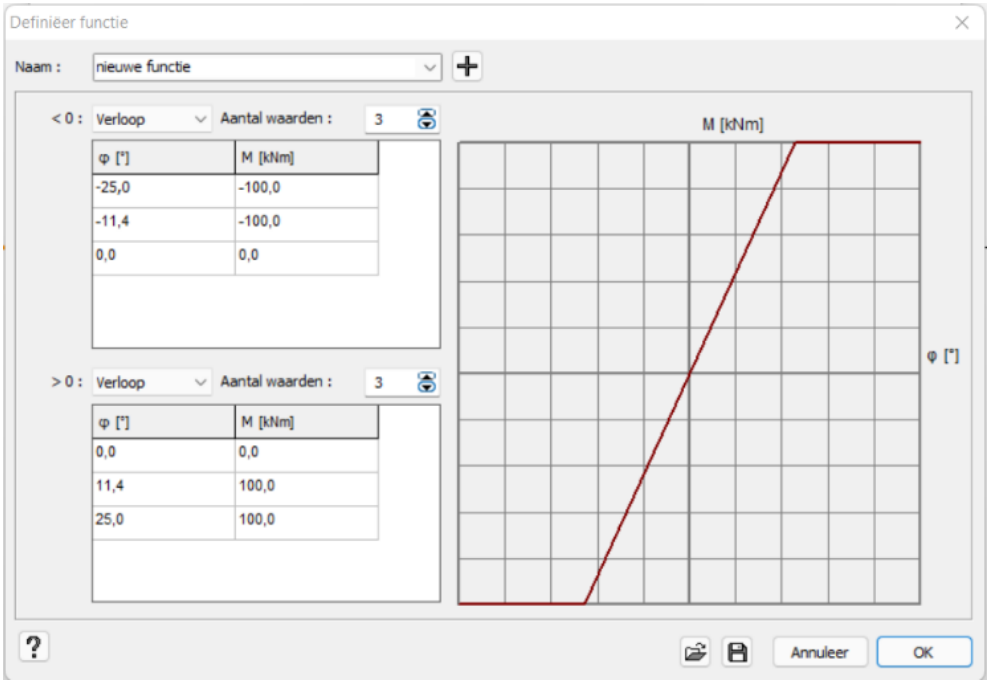
De maximale rotatiehoek is op Figuur 11 weergegeven en bedraagt 0,09 à 0,1 radialen. Indien deze waarde dan vergeleken wordt met de waarden van beide normen kan er geconcludeerd worden dat de waarde van beide normen groter zijn. Zo is de waarde van de Amerikaanse norm twee keer groter en die van de Europese norm maar liefst zes keer groter. Er kan dus geconcludeerd worden dat de Europese norm niet enkel minder streng maar ook minder veilig is. Omwille van deze vaststelling zal er deze masterproef gewerkt worden met een hoekrotatie van 0,2 radialen die voorgeschreven is door UFC 4-023-03. [11]

6 Mogelijkheden softwarepakketten

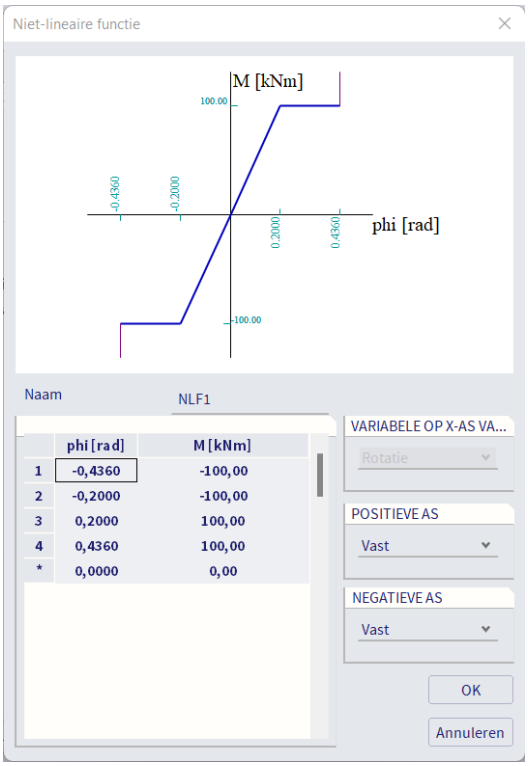
Het vooronderzoek van deze masterproef bestaat eruit om de mogelijkheden van de simulatie van kolomverlies te onderzoeken in verschillende commerciële analysesoftware. Zo worden eerst de mogelijkheden binnen Buildsoft Diamonds besproken en nadien die in SCIA Engineer.

In het onderzoek naar de mogelijkheden binnen de verschillende analysesoftware wordt er met een eenvoudige structuur gewerkt. Deze structuur wordt In Buildsoft Diamonds gemodelleerd als ligger op drie scharnierpunten. Om alle mogelijkheden In SCIA Engineer te onderzoeken wordt de structuur gemodelleerd als ligger op drie kolommen.

Het simuleren van kolomverlies komt overeen met het wegnemen van een steunpunt of kolom van de ligger. Naar aanleiding van de eerder uitgevoerde literatuurstudie zijn er een aantal zaken die in het model dienen voor te komen. Zo moeten de staven aan iedere kant van het knooppunt waarop het weg te nemen steunpunt aanwezig is, na een hoekrotatie van 0,2 radialen hun stijfheid verliezen. Het verliezen van de stijfheid kan worden gesimuleerd door het invoeren van een plastisch scharnier op de ligger. Dit is mogelijk in beide analysesoftware door een functie toe te kennen aan het plastisch scharnier. De functie die gemodelleerd werd voor Buildsoft Diamonds is weergegeven in Figuur 12. Voor de analysesoftware SCIA Engineer werd de functie opgesteld die weergegeven is in Figuur 13.



Figuur 12: Functie plastisch scharnier Buildsoft Diamonds [12]



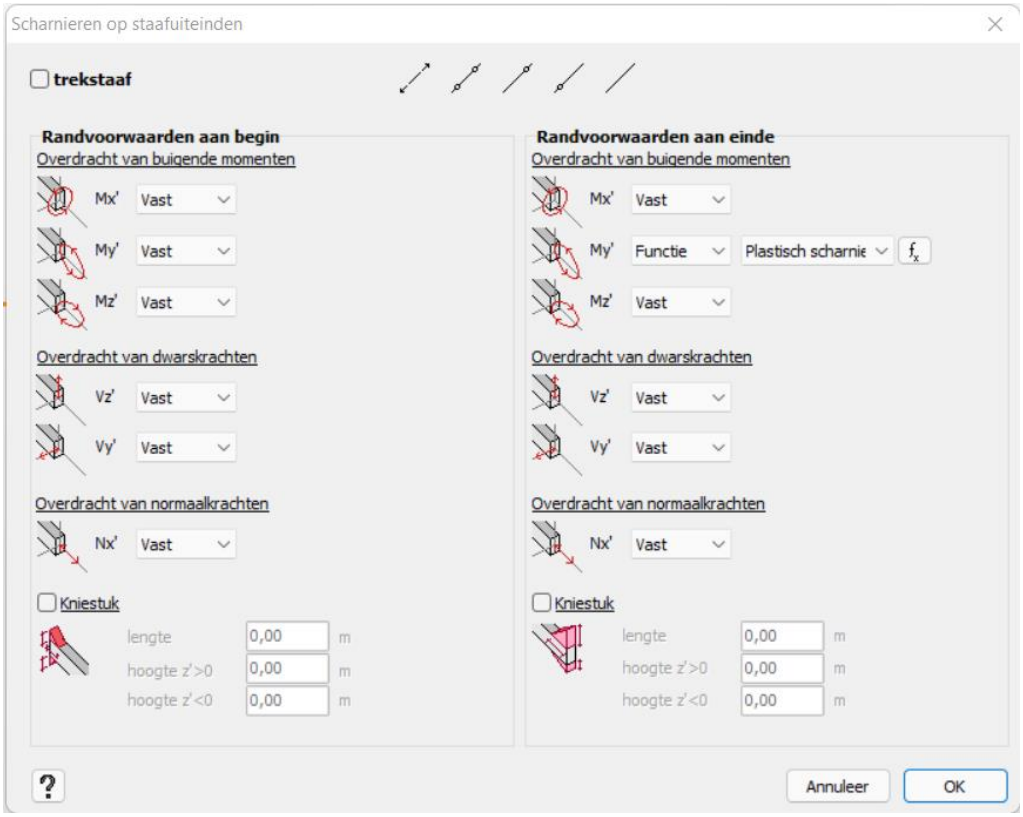
Figuur 13: Functie plastisch scharnier SCIA Engineer [13]

Het verloop van de functies weergegeven in bovenstaande figuren toont dat na het bereiken van een hoek van 0,2 radialen of 11,4° de richtingscoëfficiënt van de functie nul bedraagt. Dit betekent dat de staaf geen stijfheid meer bevat en dat het geen extra moment meer kan opnemen. Het moment gebruikt in beide functies bedraagt 100 kNm. Deze waarde is het plastisch moment van de ligger afgerond naar het dichtstbijzijnde honderdtal. Naast de waarde van 0,2 radialen of 11,4° en de oorsprong is er ook een de derde waarde van 0,436 radialen of 25° gebruikt. Deze is slechts toegevoegd ter verduidelijking van het verloop van de functie.

6.1 Mogelijkheden Buildsoft Diamonds 2021 R.03

In het volgende deel van dit onderzoek wordt er onderzocht welke mogelijkheden Buildsoft Diamonds biedt voor het doel van dit onderzoek. Met als hoofddoel op een haalbare manier kolomverlies te simuleren. Naast de simulatie dient het ook mogelijk te zijn een algoritme op te stellen dat voor iedere kolom in een structuur kolomverlies simuleert. Het programma Buildsoft Diamonds is een eindige-elementensoftware die kan gebruikt worden voor het modelleren van staal, hout en beton structuren. Daarnaast bezit het programma nog enkele mogelijkheden om speciale belastingen toe te voegen aan de structuur. Denk hierbij bijvoorbeeld aan brandbelasting, seismische belasting of gewoonweg een dynamische belasting.

Zoals eerder vermeld, kan het verlies van stijfheid van de ligger gesimuleerd worden door een plastisch scharnier toe te voegen op de ligger. Figuur 14 toont de randvoorwaarden van de ligger in de structuur.



Figuur 14: Randvoorwaarden ligger [12]

Dit plastisch scharnier dient aan beide zijden van het steunpunt te worden toegepast. Na het toekennen van het plastisch scharnier is de volgende stap het weghalen van het middelste steunpunt ter simulatie van kolomverlies. Het is mogelijk om de structuur voor iedere kolom te hertekenen maar dit geeft geen mogelijkheid tot automatisering van het proces. Dit maakt dat deze methode niet gepast is voor dit onderzoek. De simulatie van kolomverlies kan ook gebeuren op basis van dynamische lasten.

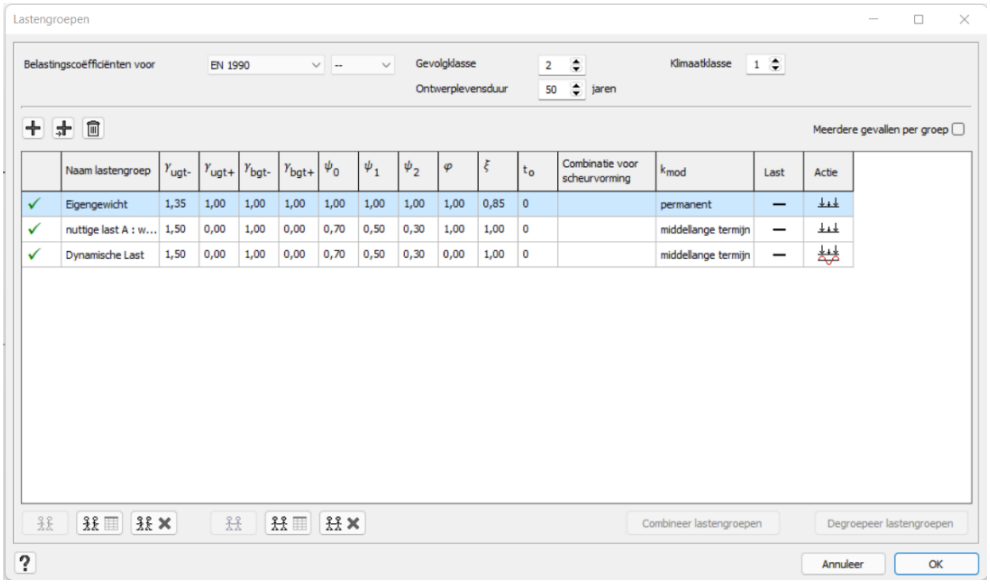
De andere parameters van dit model worden in deze paragraaf aangehaald. De ligger is een IPE 300 profiel van staalkwaliteit S235 en heeft een lengte van 10m. De drie steunpunten zijn inklemmingen en als permanente last wordt er enkel het eigengewicht in rekening gebracht. De variabele last is een lijnlast van 10 kN/m.

6.1.1 Gebruik dynamische belasting

Zoals eerder aangehaald is er de optie om gebruik te maken van een dynamische belasting om kolom verlies te simuleren. Dit kan op verschillende manieren gebeuren, de eerste manier maakt gebruik van een dynamische belasting. Het steunpunt dat verwijderd dient te worden, wordt vervangen door een dynamische belasting met dezelfde grootte als zijn reactiekracht. De aanwezigheid van deze dynamische belasting wordt beschreven door een blokgolf. Het probleem van deze methode bestaat eruit dat de structuur eerst moet worden uitgerekend op drie steunpunten om zo de steunpuntsreactie te kunnen bepalen. Daarna is het mogelijk om de structuur aan te passen met de steunpuntsreactie en tenslotte de analyse uit te voeren. Indien de eerder beschreven structuur op drie steunpunten uitgerekend wordt, wordt er een verticale reactiekracht in het middelste steunpunt van 59,1 kN verkregen. Deze reactiekracht is terug te vinden in Figuur 15.

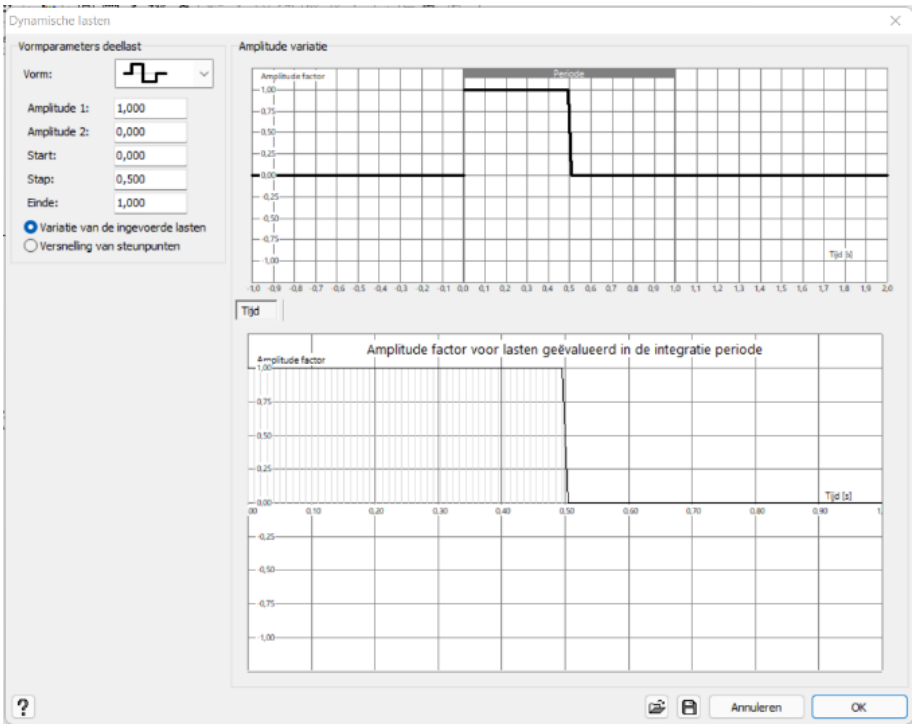
Figuur 15: Reactiekrachten steunpunten [12]

Eenmaal de grootte van de reactiekracht in het weg te nemen steunpunt gekend is, kan er een dynamische belasting worden gecreëerd. De eerste stap hiertoe is het aanmaken van een nieuwe lastengroep. Bij het aanmaken van deze lastengroep moet worden aangegeven dat er dynamische belastingen aanwezig zijn in deze lastengroep. De verschillende gebruikte lastengroepen in het model zijn opgesomd in Figuur 16.



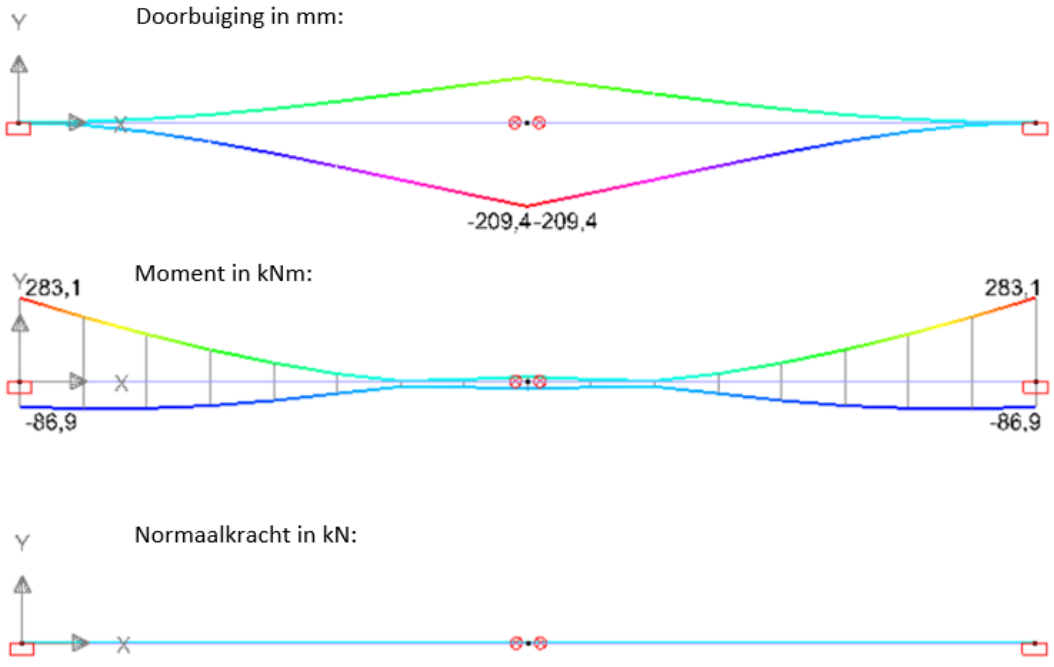
Figuur 16: Lastengroepen [12]

Eens de lastengroep voor de dynamische belasting is aangemaakt, kan het steunpunt vervangen worden door een dynamische belasting. De dynamische belasting wordt in de vorm van een puntlast met een grootte van 59,1 kN aangebracht op de plaats waar het weg te nemen steunpunt zich bevindt. Vervolgens dient de aard van de dynamisch belasting te worden gedefinieerd. Dit gebeurt aan de hand van de functie Dynamische belasting en is terug te vinden onder de sectie Dynamisch in Buildsoft Diamonds. De blokgolf die gebruikt wordt om de aanwezigheid van de dynamische belasting te beschrijven, is gedefinieerd in twee delen. Zo bevat het eerste deel van de blokgolf een amplitude van grootte 1 en het tweede deel een amplitude van grootte 0. Deze blokgolf is weergegeven in Figuur 17.



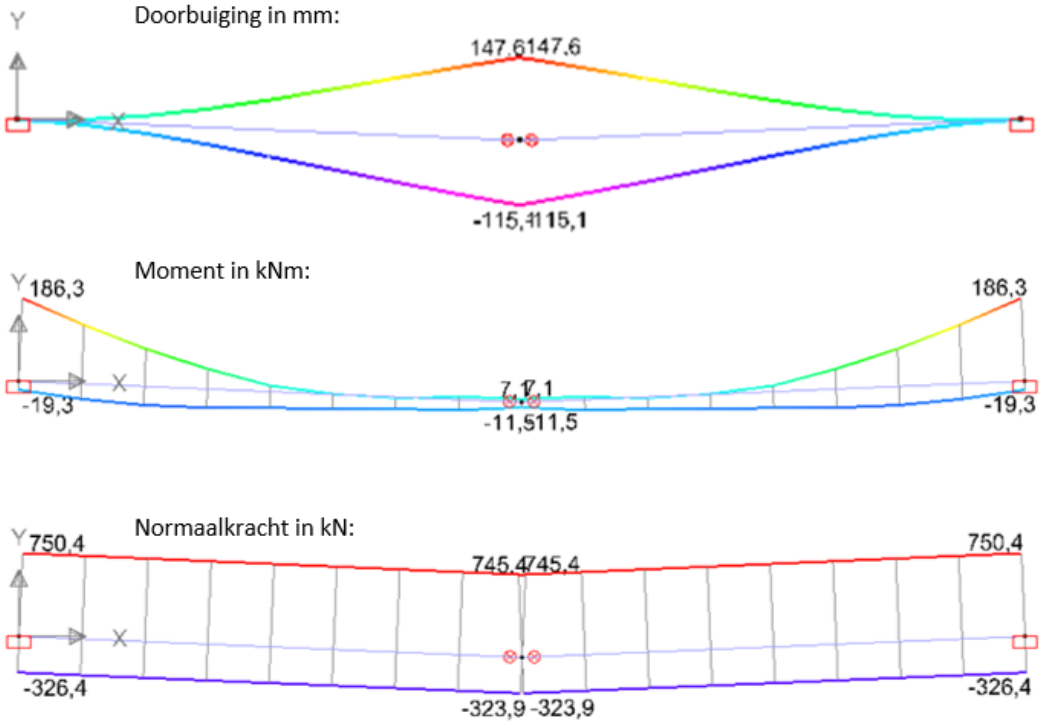
Figuur 17: Blokgolf dynamische belasting [12]

Na het toevoegen van de dynamische belasting aan zijn lastengroep, dienen nieuwe belastingcombinaties opgesteld te worden. Eenmaal dit gebeurd is, kan er opnieuw een elastische analyse uitgevoerd worden op de structuur. De belangrijkste resultaten van deze analyse zijn de doorbuiging, het moment en de normaalkrachten binnen de structuur. Deze resultaten zijn representatief voor de werking van de structuur. Figuur 18 toont de belangrijkste resultaten.



Figuur 18: Resultaten elastische analyse dynamische belasting [12]

Het gebrek aan normaalkrachten in de resultaten van de elastische analyse duidt dat het dynamische effect nog niet is gesimuleerd. Om de dynamische effecten in rekening te brengen dient er nog een verplaatsing van het knooppunt te gebeuren. Deze verplaatsing mag een maximale waarde hebben van 0,2 radialen keer de afstand tussen het middelste knooppunt en een knooppunt aan de rand van de balk. Indien de doorbuiging uit de resultaten van de elastische analyse kleiner is dan deze grenswaarde, dient de verplaatsing van het knooppunt te gebeuren met de doorbuiging van de resultaten van de elastische analyse. In dit geval bedraagt de doorbuiging 209,4 mm en is de grenswaarde 1000 mm. Het knooppunt wordt vervolgens 209,4 mm naar beneden verplaatst. Tenslotte wordt op deze aangepaste structuur opnieuw een elastische analyse uitgevoerd en zijn de resultaten weergegeven in Figuur 19.

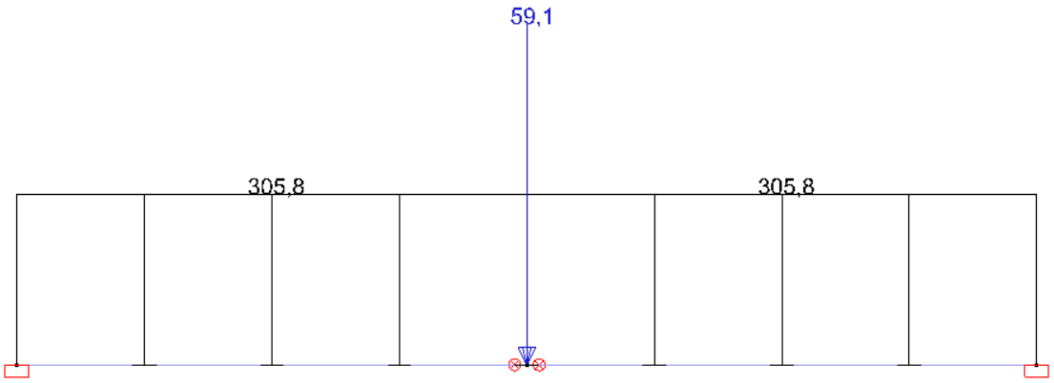


Figuur 19: Resultaten elastische analyse dynamische belasting met dynamisch effect [12]

De resultaten van de elastische analyse met dynamisch effect verschillen duidelijk van deze zonder dynamisch effect. Zo kan er een duidelijke daling van momenten en een stijging van normaalkrachten waargenomen worden.

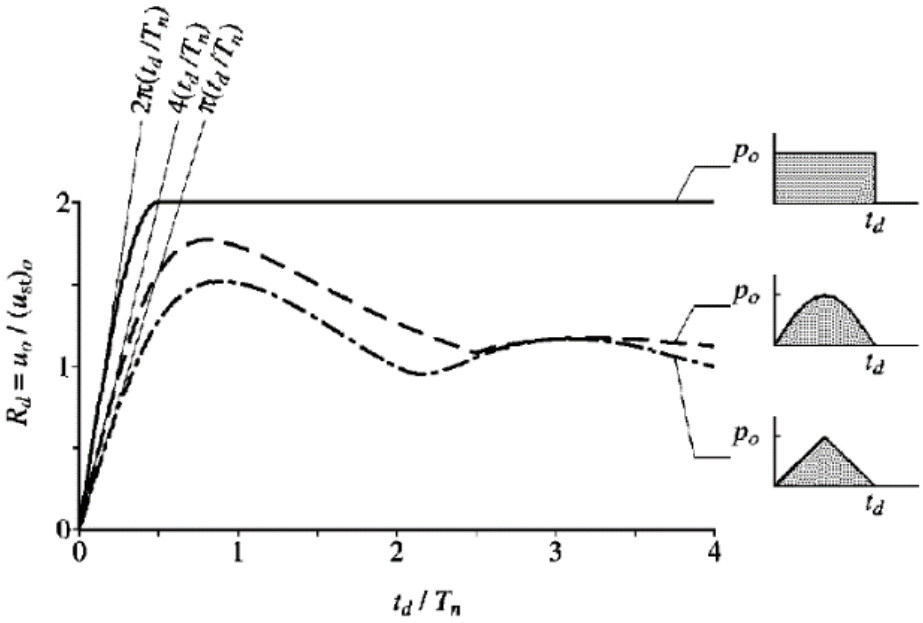
6.1.2 Gebruik impulsbelasting

Naast het gewoon hertekenen van de structuur en het gebruik van een dynamische belasting is het ook mogelijk om gebruik te maken van een impulsbelasting. Deze methode is in grote lijnen dezelfde als deze met een dynamische belasting, maar verschilt bij het aanbrengen van de puntlast. In tegenstelling tot bij de dynamische puntlast waar deze het knooppunt verving zal de impulsbelasting de reactiekracht van het knooppunt opheffen. Ook hier wordt een nieuwe lastengroep aangemaakt. In deze nieuwe aangemaakte lastengroep zal dan een impulsreactie geplaatst worden die tegengesteld is aan de steunpuntsreactie van het middelste steunpunt. De structuur met de aangebracht impulsbelasting wordt weergegeven in Figuur 20.



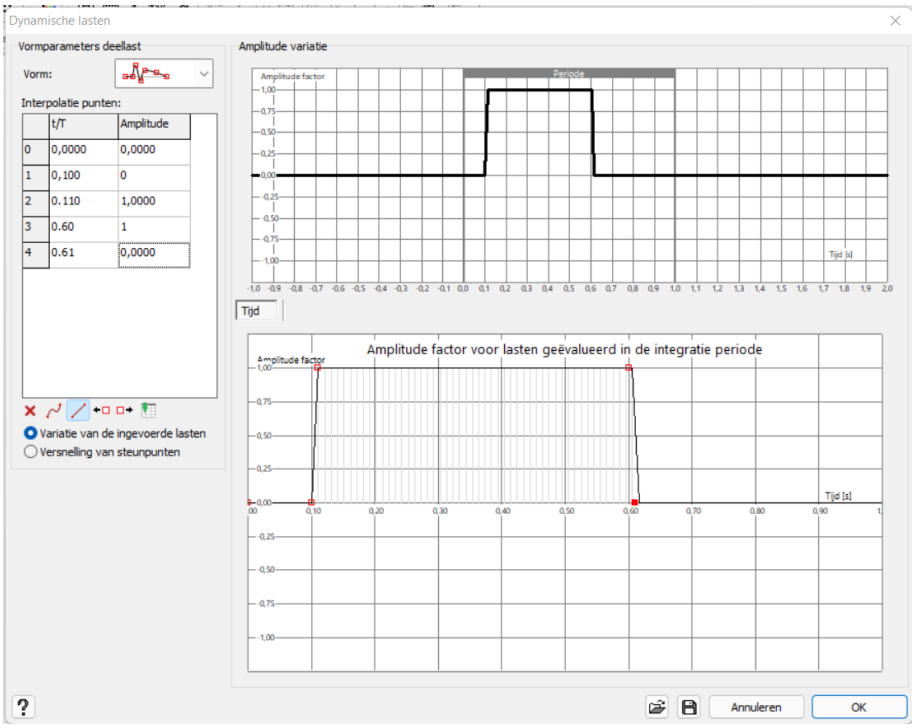
Figuur 20: Impuls belasting [12]

Net zoals bij het gebruik van de dynamische belasting is er ook in dit geval een functie vereist om de aanwezigheid van de impulsbelasting te beschrijven. De waarden waaruit de functie is opgebouwd zijn zorgvuldig gekozen. Volgens [14] zijn de effecten van een impulsbelasting het grootst wanneer de functie een rechthoekige golf is. Indien er gebruikt gemaakt zou worden van een driehoekige golf zal het effect van de impulsbelasting beduidend kleiner zijn. Figuur 21 beschrijft de effecten van de verschillende golfvormen.



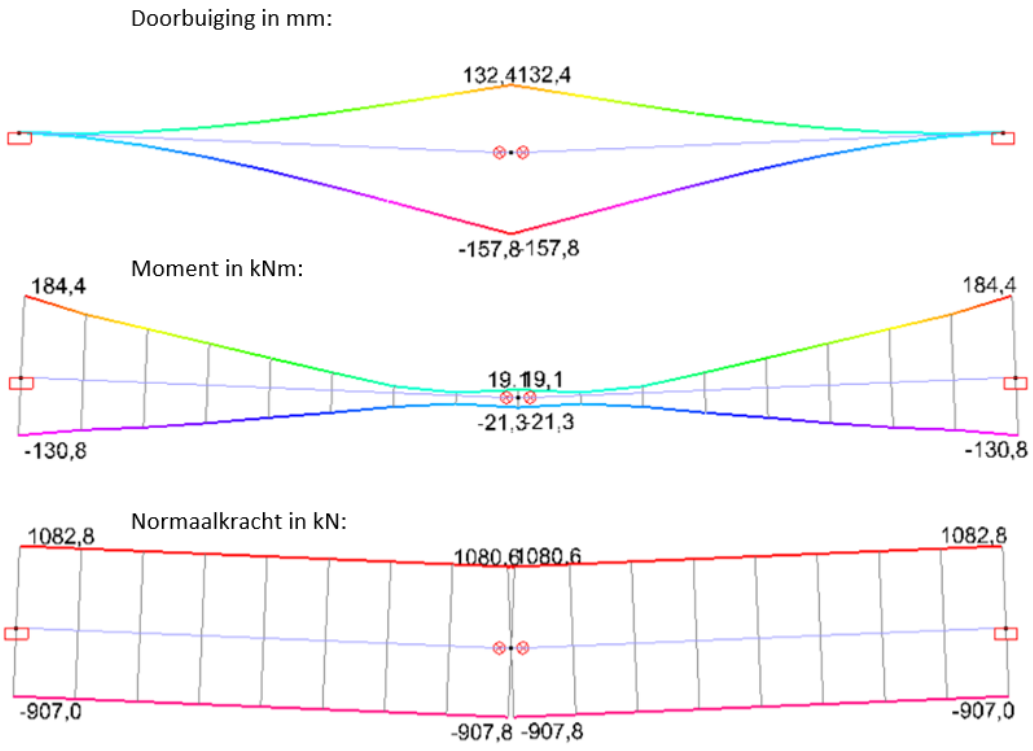
Figuur 21: Effecten van verschillende pulsvormen [14]

Aangezien in dit onderzoek de meest nadelige situatie aan de orde is, wordt gekozen om met een rechthoekige golf te werken. Ook kan uit Figuur 21 geconcludeerd worden dat een maximumreactie bekomen wordt rond een puls duur van 0,5. De gebruikte rechthoekige golf is terug te vinden in Figuur 22.



Figuur 22: Blokgolf impulsbelasting [12]

Eenmaal de impulsbelasting gemodelleerd is en een rotatie van de staaf wordt gerealiseerd met een grootte van de eerder besproken grenswaarde van 0,2 radialen, kan er ook op dit model een elastische analyse uitgevoerd worden. De resultaten van deze analyse zijn weergegeven in Figuur 23.



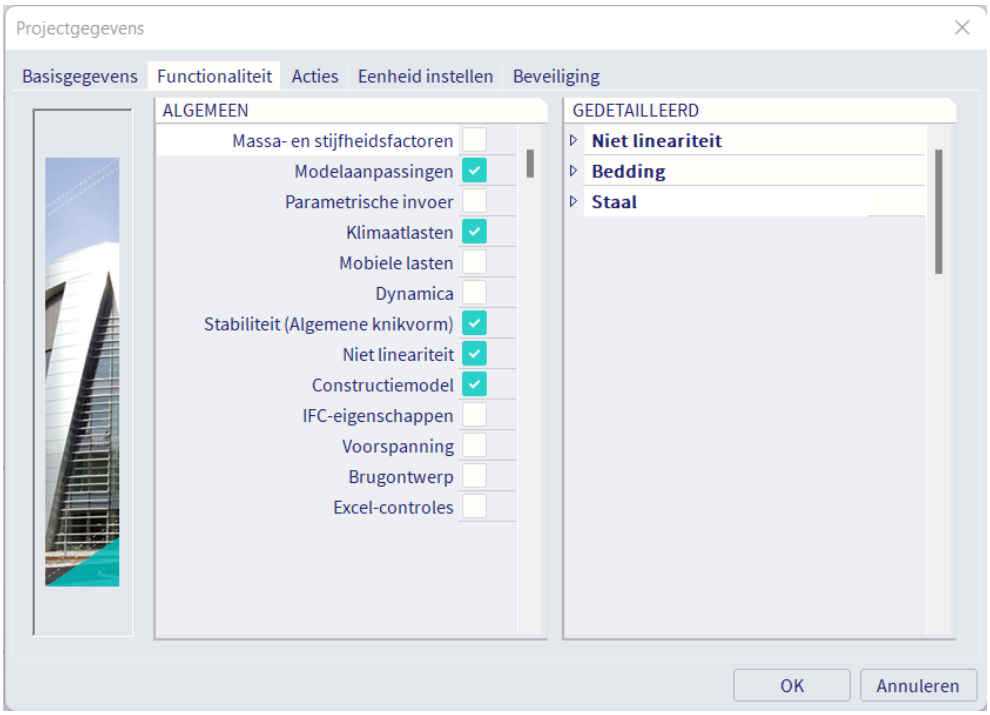
Figuur 23: Resultaten elastische analyse impulsbelasting [12]

Uit de resultaten bekomen door zowel de elastische analyse van de structuur met een dynamische belasting als die van de structuur met een impulsbelasting kan geconcludeerd worden dat de verlopen van de interne krachten gelijkaardig zijn. Echter verschillen de waarden van de interne krachten toch sterk. Een verklaring schuilt in de wijze waarop kolomverlies gesimuleerd wordt. In de eerder besproken methode in paragraaf 6.1.1 werd de steunpuntsreactie gedefinieerd als een dynamische last naast de reeds aanwezige nuttige last. Deze dynamische belasting wordt in de belastingscombinatie dan slechts voor 0,7 keer 1,5 toegepast. De factor 0,7 is een combinatiefactor. In de tweede methode, besproken in paragraaf 6.1.2, wordt het steunpunt ook vervangen door zijn reactiekracht, maar ook wordt een tegengestelde impulsbelasting geplaatst. De reactiekracht die het steunpunt vervangt wordt in de nuttige lastengroep gemodelleerd en wordt doorgerekend in de belastingscombinaties aan 1 keer 1,5. Dit is een mogelijke verklaring van het verschil in resultaten van beide methoden.

Niet alleen de resultaten komen in grote lijnen overeen voor beide methoden, ook delen ze dezelfde problemen. Een van deze problemen is het feit dat meerdere elastische analyses verreist zijn vooraleer kolomverlies gesimuleerd kan worden. Dit zorgt ervoor dat het automatiseren van dit proces niet haalbaar is. Op basis van deze redenen is er besloten om de methoden in Buildsoft Diamonds enkel handmatig toe te lichten en verder op zoek te gaan naar een andere analysesoftware die het wel mogelijk maakt om de simulatie van kolomverlies te automatiseren. Zo biedt de software SCIA Engineer deze mogelijkheden en zal deze software verder onderzocht worden.

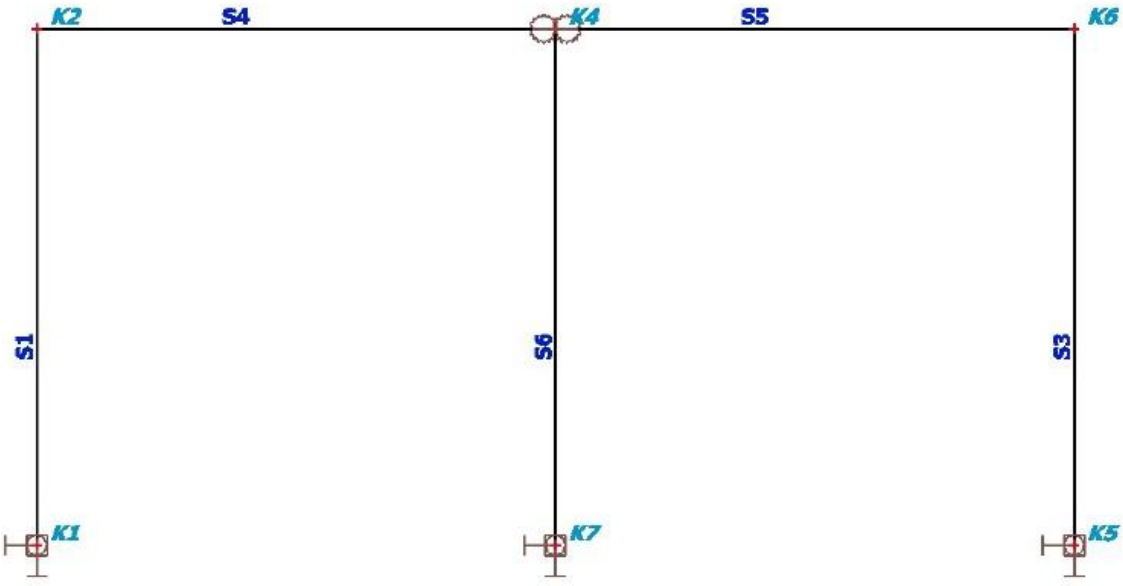
6.2 Mogelijkheden SCIA Engineer 21.1.1028

Om een zo goed mogelijke vergelijking te maken tussen de twee analysesoftware wordt er voor het onderzoek naar de mogelijkheden van SCIA Engineer een gelijkaardige structuur gebruikt als basis. Ook hier werden de verschillende mogelijkheden bekeken om het middelste steunpunt van de balk weg te nemen. In tegenstelling tot Buildsoft Diamonds is het in SCIA Engineer niet mogelijk om gebruikt te maken van een dynamische belasting. Wel kunnen seismische en harmonische belastingen worden gebruikt door middel van een speciale berekening. Deze soorten belastingen zijn echter niet bruikbaar in kader van dit onderzoek. Een andere manier is net zoals in Buildsoft Diamonds het steunpunt manueel te verwijderen. Alvorens het onderzoek naar SCIA Engineer gestart kon worden diende er rekening gehouden te worden met enkele zaken. Zo moet tijdens het creëren van het bestand waarin de structuur gemodelleerd wordt specifiek aangegeven worden dat er gebruik zal gemaakt worden van enkele functionaliteiten. Zowel de functionaliteit Modelaanpassingen als Niet lineariteit zijn nodig om het onderzoek naar SCIA Engineer uit te voeren. Zo is in Figuur 24 weergegeven hoe aangegeven moet worden dat er gebruik zal gemaakt worden van deze functionaliteiten.



Figuur 24: Projectgegevens SCIA Engineer [13]

De functionaliteit Modelaanpassingen maakt het mogelijk om een belastingscombinatie te creëren waarin bepaalde elementen van de structuur geen stijfheid bezitten gedurende de berekening voor deze belastingscombinatie. Om de methode aan de hand van de functionaliteit Modelaanpassingen verder te onderzoek wordt er gebruik gemaakt van een nieuwe structuur die bestaat uit een balk ondersteund door drie kolommen. De structuur is gelijkaardig met de eerder gebruikte structuur van de balk op drie steunpunten maar verschilt enkel dat de steunpunten hier vervangen zijn door kolommen. De nieuwe structuur is weergegeven in Figuur 25.

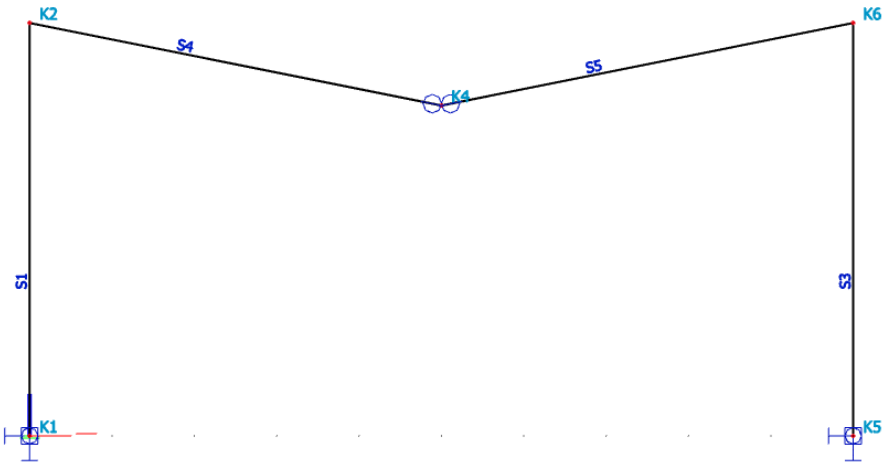


Figuur 25: Structuur Modelaanpassingen SCIA Engineer [13]

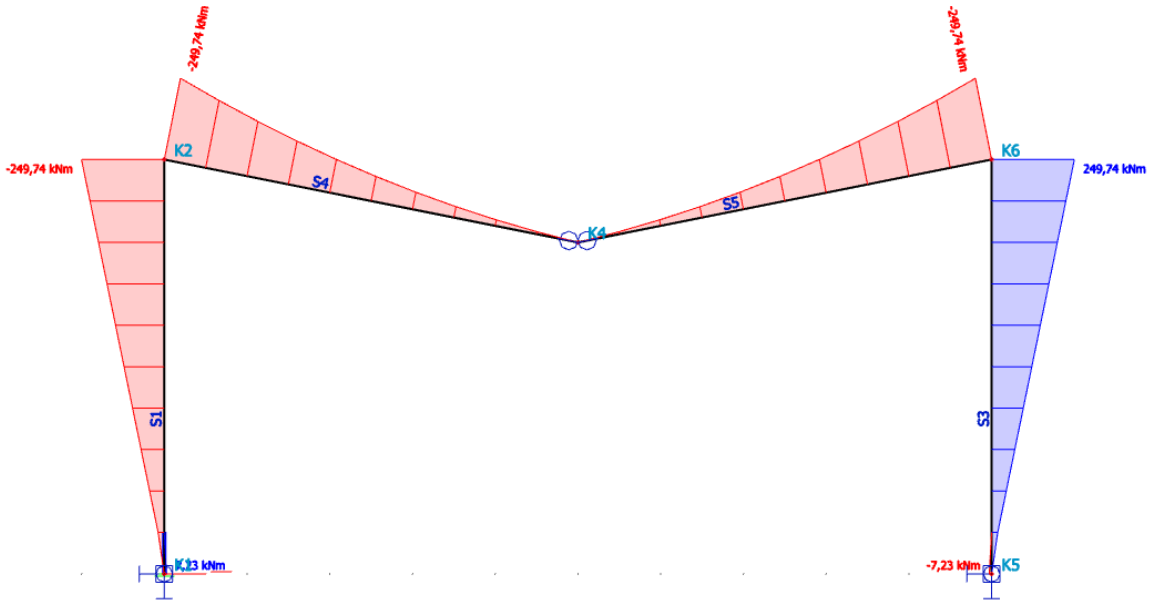
Zoals weergegeven is op Figuur 25 zijn de drie kolommen die de balk ondersteunen onderaan ingeklemd. Verder zijn de eigenschappen van de structuur dezelfde als deze van de balk op drie steunpunten. Ook hier zijn het IPE 300 profielen en bedraagt de afstand tussen de knooppunten 5 meter. En net zoals in de vorige structuur wordt zowel het eigengewicht als een nuttige lijnlast van 10 kN/m in rekening gebracht.

6.2.1 Manueel verwijderen kolom

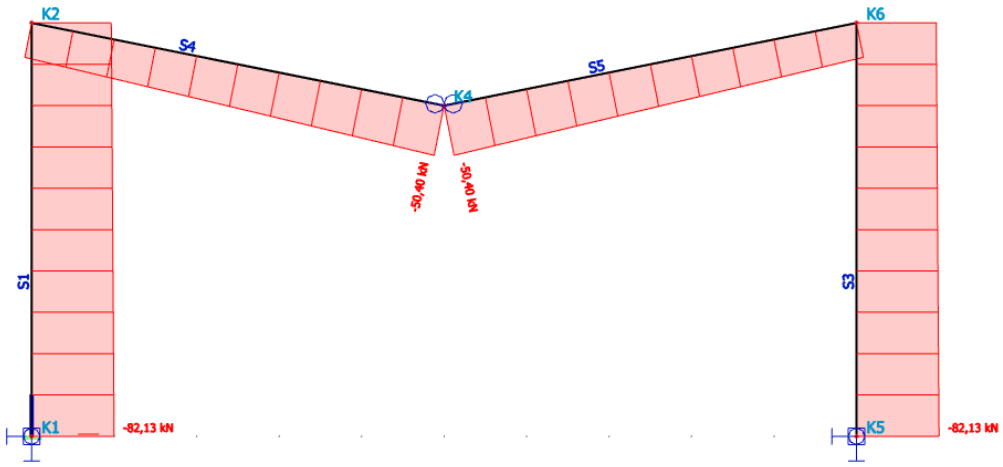
Om verder in het onderzoek de functionaliteit “modelaanpassingen” te valideren wordt eerst het kolomverlies gesimuleerd aan de hand van manueel de middelste kolom te verwijderen. Net zoals bij de simulatie in Buildsoft Diamonds is een knoopverplaatsing verreist om het dynamische effect in kaart te brengen. Deze verplaatsingen maakt ook gebruik van de maximale hoekverdraaiing van 0,2 radialen die voorgeschreven is in [9]. De structuur van een balk op drie kolommen waarop kolomverlies is toegepast door het manueel verwijderen van de kolom is weergegeven in Figuur 26. De resultaten van de analyse van deze structuur zijn weergegeven in Figuur 27 en Figuur 28.



Figuur 26: Structuur met kolomverlies door manueel verwijderen kolom [13]



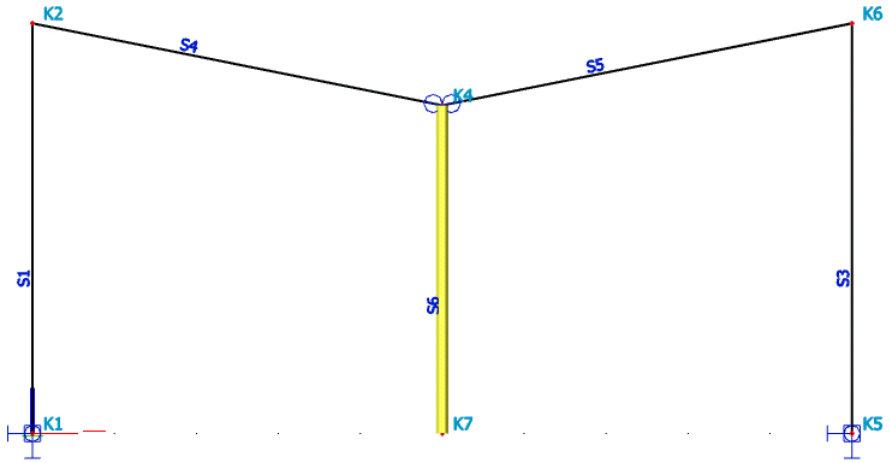
Figuur 27: Momenten structuur met kolomverlies door manueel verwijderen kolom [13]



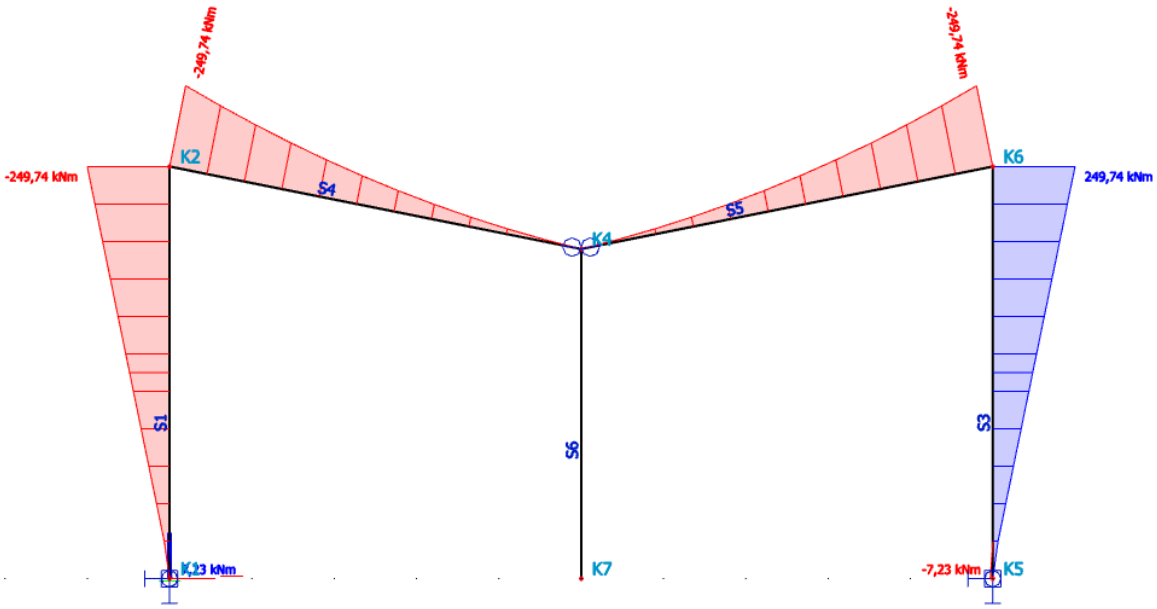
Figuur 28: Normaalkrachten structuur met kolomverlies door manueel verwijderen kolom [13]

6.2.2 Functionaliteit “modelaanpassingen”

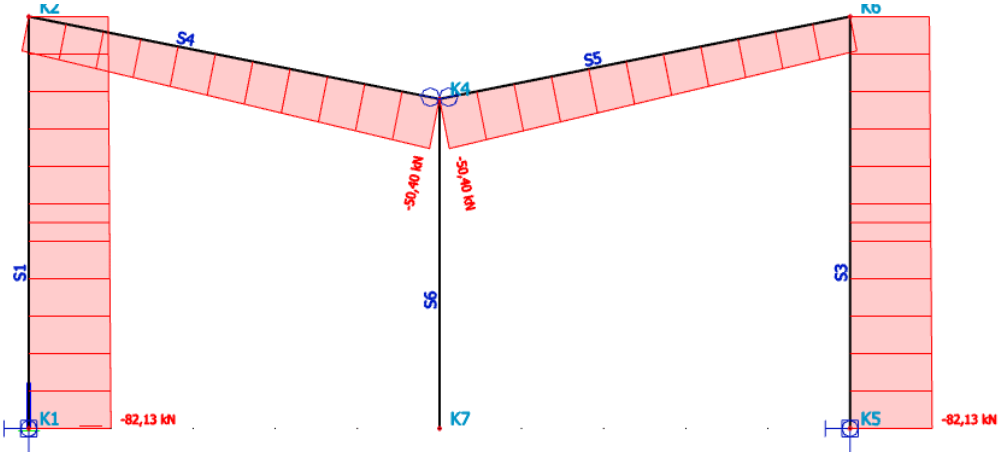
Nu dat het kolomverlies manueel gesimuleerd is kunnen de mogelijkheden aan de hand van de functionaliteit “modelaanpassingen” onderzocht worden. De eerste stap in dit proces is het aanmaken van een modificatiegroep. Vervolgens kan aan de hand van de functie 1D afwezigheden de kolom gemodelleerd worden zodat deze verloren zal gaan. Vooraleer een analyse kan worden uitgevoerd dient de eerder aangemaakte modificatiegroep toegekend te worden aan de verschillende belastingsgevallen waarop deze van toepassing is. Indien bovenstaande stappen voltooid zijn kan een analyse uitgevoerd worden op de structuur die weergegeven is in Figuur 29. De resultaten van de analyse van deze structuur zijn weergegeven in Figuur 30 en Figuur 31.



Figuur 29: Structuur met kolomverlies door Modelaanpassingen [13]



Figuur 30: Momenten structuur met kolomverlies door Modelaanpassingen [13]



Figuur 31: Normaalkrachten structuur met kolomverlies door Modelaanpassingen [13]

In tegenstelling tot de vaststelling tijdens het onderzoek naar de mogelijkheden van Buildsoft Diamonds zijn de resultaten voor de verschillende mogelijkheden binnen SCIA Engineer wel identiek aan elkaar. Wel is de voorstelling van de structuur verschillend. Zo is de middelste kolom bij de resultaten van de mogelijkheid Modelaanpassingen nog steeds zichtbaar terwijl dit niet is voor de andere mogelijkheid. Dit is te wijten aan het verschil in modeleren. Zo wordt bij de functionaliteit modelaanpassingen de kolom niet verwijderd, maar beschouwt als een kolom zonder stijfheid. De middelste kolom kan in beide gevallen geen krachten opnemen en dus ook geen invloed uitoefenen op de structuur.

6.3 Mogelijkheid tot automatisatie

SCIA Engineer biedt naast de *interface* zelf ook een tool aan genaamd ESA_XML. Deze tool maakt het mogelijk om analyses van structuren op de achtergrond uit te voeren. Ook kan de gebruiker zelf kiezen hoe hij de resultaten wil ontvangen, dit kan zowel in XML-formaat, waar dit onderzoek verder gebruik van zal maken, maar dit kan ook in TXT-formaat alsook in PDF-formaat.

6.4 Conclusie vooronderzoek

Zowel in Buildsoft Diamonds als SCIA Engineer is het mogelijk om kolomverlies te simuleren. Na vergelijking van de resultaten kan vastgesteld worden dat er een lichte afwijking is tussen de verschillende methode binnen Buildsoft Diamonds. Deze afwijking is echter niet aanwezig bij de vergelijking van de resultaten van SCIA Engineer. Daarnaast is de mogelijkheid om analyses uit te voeren wel aanwezig in de analysesoftware SCIA Engineer maar niet in Buildsoft Diamonds.

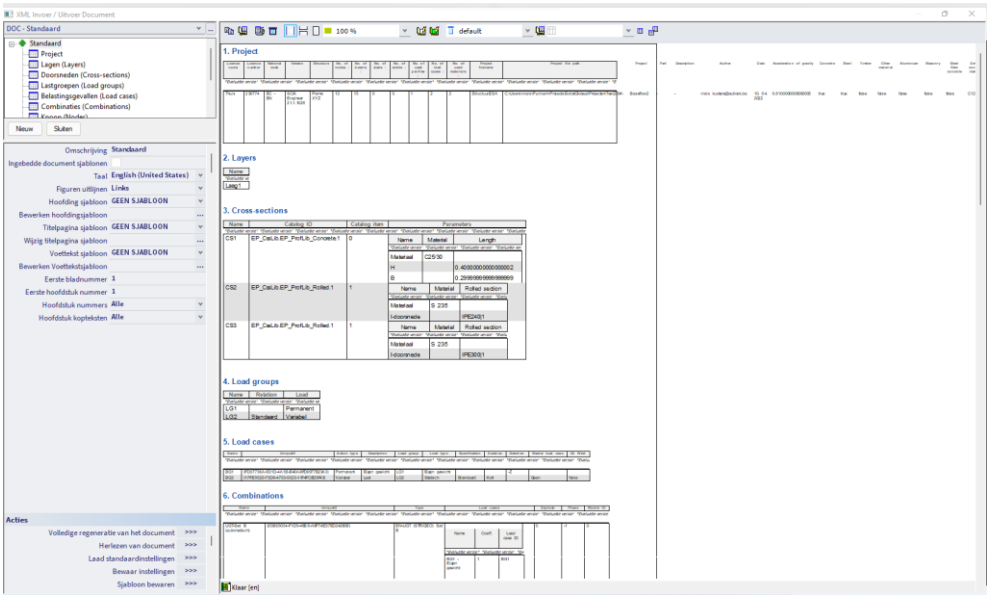
Na vergelijking van beide analysesoftware en na de vaststelling dat de software SCIA Engineer meer mogelijkheden biedt voor dit onderzoek, wordt besloten om het onderzoek verder te zetten in SCIA Engineer.

7 Scripts in SCIA Engineer

Na het bekijken van de verschillende mogelijkheden in zowel Buildsoft Diamonds als SCIA Engineer zullen in dit hoofdstuk de verschillende algoritmen die opgesteld zijn in dit onderzoek besproken worden.

7.1 Structuur XML-bestand

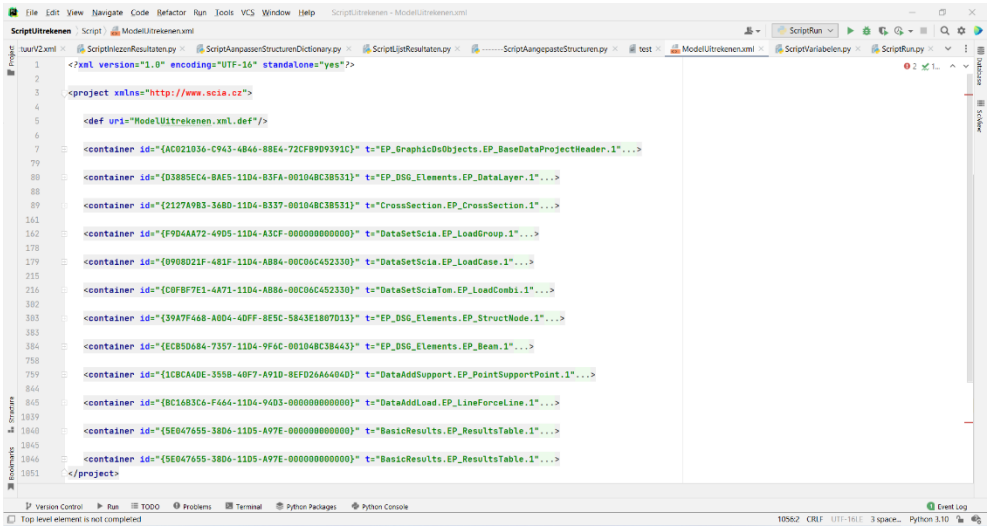
Gedurende dit onderzoek zullen er meerdere algoritmen aan bod komen. Echter hebben deze verschillende algoritmen één ding gemeenschappelijk, namelijk het gebruik van XML-bestanden. In SCIA Engineer dient het model dan ook geëxporteerd te worden naar een XML-bestand. De gebruiker kan zelf bepalen wat er in dit XML-bestand zal staan. Figuur 32 geeft het scherm weer waarin de gebruiker kan selecteren wat er al dan niet in het XML-bestand zal staan.



Figuur 32: Exporteren XML-bestand [13]

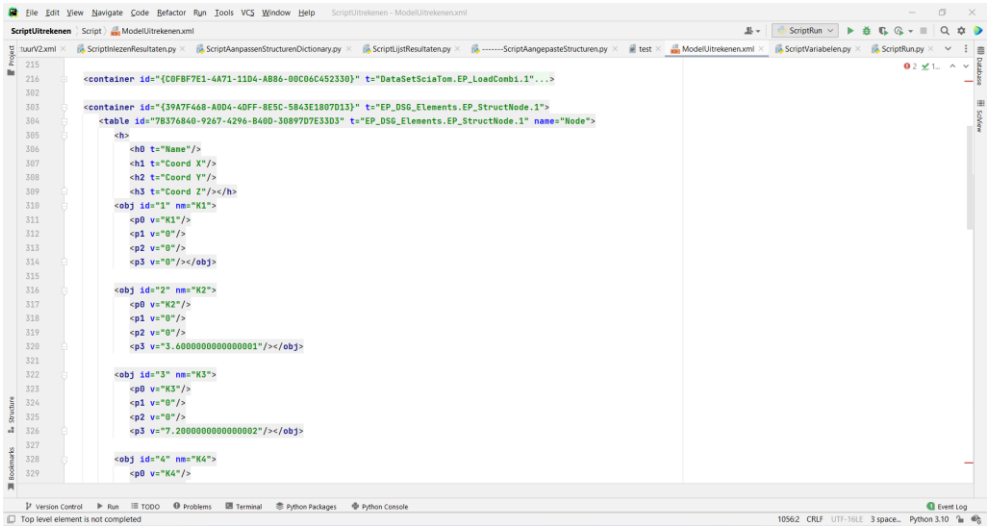
De opbouw van het XML-bestand kan afgeleid worden uit Figuur 32. Zo bestaat dit uit meerdere genummerde titels met daarbij de bijhorende tabel. In de tabellen wordt al de gevraagde informatie gebundeld over het model. Aan de linkerkant van het scherm kunnen er aanpassingen gebeuren aan het XML-bestand, gaande van het wijzigen van opmaak als het toevoegen of verwijderen van extra genummerde titels. Zo is het verreist voor dit onderzoek dat de resultaten *output* ook wordt toegevoegd aan het XML-bestand. Indien dit niet gebeurt zullen er geen resultaten verschijnen in het XML-bestand na uitvoeren van de analyse. Eenmaal dit document naar wens is, kan er overgegaan naar het effectief exporteren naar een XML-bestand.

Om een XML-bestand aan te passen dient er gebruik te worden gemaakt van tekst-editors, zoals Notepad++ of PyCharm. In dit onderzoek werd de tekst-editor PyCharm gebruikt aangezien de scripts ook in PyCharm geschreven zullen worden. Zo is er slechts één programma nodig om zowel het script te schrijven als de XML-bestanden aan te passen. De opbouw van een XML-bestand in de software PyCharm wordt weergegeven in Figuur 33.



Figuur 33: Opbouw XML-bestand [15]

De indeling die gebruikt werd tijdens het exporteren van het XML-bestand is ook in Figuur 33 zichtbaar. Zo maakt elk XML-bestand gebruik van verschillende *containers* met elk een unieke code. Echter zijn naast de unieke codes ook de genummerde titels onderdeel van deze verschillende containers. Hieruit kan afgeleid worden dat iedere container overeenkomt met een tabel uit het geëxporteerde XML-bestand. Iedere container bevat dus respectievelijke informatie die geraadpleegd kan worden door de container uit te klappen. Zo wordt als voorbeeld in Figuur 34 de opbouw van de container van de knooppunten, ook wel *nodes* weergegeven.



Figuur 34: Opbouw container knooppunten [15]

Uit de opbouw van de container van de knooppunten kan vastgesteld worden dat in iedere container een tabel zit die de gewenste informatie bevat. Zo staat er in de tabel als eerste een object “h” die de beschrijving weergeeft van wat de verdere objecten betekenen. Verder zijn er in deze opbouw meerdere “obj” oftewel objecten terug te vinden. Deze objecten stellen de verschillende knooppunten voor die gebruikt worden in dit model. Zo is “p0” de naam van het knooppunt en “p1”, “p2” en “p3” respectievelijk het x-, y- en z-coördinaat. Het zijn deze objecten die de structuur in SCIA Engineer vormen. Het aanpassen

of verwijderen van deze objecten lijdt dus tot het aanpassen of verdwijnen van het object uit de structuur. Door deze objecten aan te passen of te verwijderen, zal het simuleren van kolomverlies gebeuren.

Kort samengevat bestaat een XML-bestand dat gebruikt kan worden door SCIA Engineer uit een aantal containers met daarin de gegevens van de structuur. De gegevens die in dit XML-bestand terug te vinden zijn worden zelf gekozen door de gebruiker. Op basis van dit XML-bestand kan kolomverlies gesimuleerd worden op structuren.

7.2 Algoritme 1: genereren structuur en uitvoeren analyse

Om de doelstelling van dit onderzoek te behalen, dient één of meerdere algoritmen opgesteld te worden. Zo is het eerste opgestelde algoritme in staat om een structuur te genereren op basis van *inputs* van de gebruiker. Naast het genereren van een structuur, voert het algoritme de simulatie van kolomverlies uit en bundelt het uiteindelijk de resultaten in een tekstbestand. Voorlopig wordt er enkel gebruik gemaakt van staalstructuren met IPE 300 profielen met staalkwaliteit S235. In volgende paragrafen zullen de verschillende scripts beschreven worden die het algoritme vormen. Onder iedere paragraaf van een *script* is er een flowchart aanwezig waarop weergegeven is waar dit script zich bevindt in het algoritme.

7.2.1 Script 1: Script variabelen

Het eerste script, genaamd Script variabelen, zal aan de gebruiker een aantal inputs vragen, zoals de projectnaam, de verdiepingshoogte, aantal verdiepingen, de tussenafstand, aantal kolommen en de lijnlast op de structuur. Het script zal deze inputs wegschrijven naar een tekstbestand waaruit de andere scripts de nodige gegevens kunnen opvragen. Naast het wegschrijven van de inputs maakt het script ook een mappenstructuur aan op de computer van de gebruiker. Dit is voorzien in het script zodat het algoritme en de gebruiker zelf gebruik kunnen maken van een geordend project. Weliswaar is het ook belangrijk om te vermelden dat niet enkel de inputs worden weggeschreven naar het tekstbestand maar ook de bestandslocatie van de ESA_XML-tool die het mogelijk maakt om analyses op de achtergrond uit te voeren. Het script zoekt zelf op de computer van de gebruiker naar de bestandslocatie van deze tool en voegt deze toe aan het tekstbestand. Het script kan geraadpleegd worden in Bijlage A.

Figuur 35: Flowchart eerste algoritme – Script variabelen

7.2.2 Script 2: Script opstellen structuur

Nu de inputs van de gebruiker gekend zijn en weggeschreven zijn naar een tekstbestand kan er worden overgegaan naar het creëren van een basisstructuur waar nog geen sprake is van kolomverlies. Dit wordt in het algoritme verzorgd door het script opstellen structuur. Dit script maakt gebruik van een basefile. Deze basefile is een XML-bestand met de eerder besproken structuur uit paragraaf 7.1 waarbij de containers die de knooppunten, staven en lijnlasten beschrijven nog geen objecten bevatten. Wel bevat de basefile ook de belastingsgevallen en belastingscombinaties, de materialen en de doorsneden die gebruikt worden in deze structuur. Dit resulteert erin dat er enkel gebruik gemaakt kan worden van het IPE 300 profiel. Hieronder zijn de belangrijkste regels code uit dit script weergegeven met extra toelichting. Een deel van deze regels code wordt repetitief gebruikt om de structuur op te stellen. De regels code die hieronder geschreven staan dienen om de basefile in te lezen in PyCharm en deze nadien te kunnen bewerken.

```
Mytree = ET.parse(bestandslocatie+'\\'+'BasefileStructuurV2.xml')
myroot = mytree.getroot()
```

Eens de basefile is ingelezen, kan deze aangepast worden. Met onderstaande regels code kan een specifiek object in het XML-bestand aangepast of toegevoegd worden. Zo kunnen knooppunten, staven of lijnlasten aan de juiste containers toegevoegd worden.

```
root = ET.SubElement(myroot[7][0], 'obj')
root.set('id', X)
```

In de eerste regel code wordt in er in de tabel van de zevende container een nieuw object aangemaakt. Het enkel aanmaken van een object is niet voldoende. Verder dient er nog extra informatie aan dit object toegekend te worden, met name het nummer van bijvoorbeeld het knooppunt. Dit wordt verzorgd door de tweede regel code hierboven. Deze regel voegt het attribuut *id = X* toe aan het recent aangemaakte object. De variabele *X* staat hiervoor een getal startend van één die het nummer van het knooppunt voorstelt. De variabele wordt iedere keer doormiddel van een *while loop* vermeerderd met één zodat het script een nieuw object voor het volgend knooppunt kan aanmaken.

Indien het object correct werd aangemaakt en aangevuld werd met de nodige informatie kunnen enkele attributen toegewezen worden aan het object. Om deze attributen toe te wijzen wordt gelijkaardige code gebruikt met als enige verschil dat er nu een extra index verreist is in de eerste regel code. Deze extra index zorgt ervoor dat het object geselecteerd wordt door PyCharm en het kan worden aangepast. Welke attributen dienen toegevoegd te worden kan simpelweg uit het object *h* van iedere container van de basefile gehaald worden. Deze informatie is voor de zevende container oftewel deze van de knooppunten hieronder weergegeven.

```
<h>
  <h0 t="Name" />
  <h1 t="Coord X" />
  <h2 t="Coord Y" />
  <h3 t="Coord Z" /> </h>
```

Voor de zevende container, deze van de knooppunten, moeten er vier attributen toegevoegd worden aan ieder object van een knooppunt. Deze vier attributen zijn ten eerste de naam en vervolgens de x-, y- en z-coördinaat van het knooppunt. Voor de naam van het knooppunt hanteert SCIA Engineer de volgende wijze, de letter *K* met daaropvolgend het nummer van het knooppunt. Tijdens het automatisch aanpassen van de attributen van het object is het belangrijk dat er gekeken wordt naar de letter voor het gelijkteken, in dit geval *t*. Deze verandert immers nooit en mag niet aangepast worden. Indien dit wel gebeurt, zal SCIA Engineer niet in staat zijn om het model in te lezen.

Om de verschillende containers, objecten en attributen aan te passen worden bovenstaande regels code gebruikt die respectievelijk licht aangepast worden. Doormiddel van de combinatie van enkele simpele while loops die afhankelijk zijn van de input van de gebruiker en de bovenstaande regels code kan een volledige structuur verdiep per verdiep opgesteld worden. Indien alle aanpassingen en/of toevoegingen gebeurd zijn aan het basefile bestand wordt deze opgeslagen onder een andere naam. Het nieuw opgesteld XML-bestand kan nu gebruikt worden voor het verdere verloop van het algoritme. Dit script kan geraadpleegd worden in Bijlage B.

Figuur 36: Flowchart eerste algoritme – Script opstellen structuur

7.2.3 Script 3: Script aanpassen structuur

De volgende stap in het algoritme is om het nieuw aangemaakte XML-bestand waarin er nog geen sprake is van kolomverlies aan te passen. Het aanpassen van de structuur vindt plaats onder de vorm van het verwijderen van een kolom. Het verwijderen van een kolom is niets anders dan het verwijderen van een object uit de container met staven. Onderstaande code wordt gebruikt om een object te verwijderen uit het XML-bestand.

myroot[8][0].remove(myroot[8][0][X])

Net zoals in de regels code uit paragraaf 7.2.2 staat *X* hier voor het nummer van het object, in dit geval het nummer van de staaf die verwijderd dient te worden. Na het verwijderen van de staaf wordt het bestand opnieuw opgeslagen onder een andere naam om het eerder opgesteld XML-bestand waarin er nog geen sprake is van kolomverlies niet te wijzigen. Zoals besproken in paragraaf 6.2.1 volstaat het niet om enkel de kolom te verwijderen. Ook een verplaatsing van de knooppunten boven de te verwijderen kolom is verreist om de dynamische effecten in rekening te brengen. Deze verplaatsing is bepaald aan de hand van [9] en werd eerder ook al grenswaarde genoemd in dit onderzoek. Deze waarde bedraagt 0,2 radialen maal de afstand tussen twee knooppunten. Een knooppuntsverplaatsing kan gerealiseerd worden door het object van het knooppunt leeg te maken en de aangepaste gegevens als attributen toe te voegen. Het leeg maken van een object kan aan de hand van onderstaande regel code. Het toevoegen van attributen aan objecten werd eerder in paragraaf 7.2.2 reeds besproken.

myroot[7][0][X].clear()

Ook ditmaal wordt het nieuw opgestelde bestand opgeslagen onder een andere naam. Het is zo dat het bestand waarin enkel een kolom verwijderd wordt de basis is van het XML-bestand waarin de knooppunten verplaatst worden. Eens het XML-bestand van kolomverlies als dat voor kolomverlies en knooppuntsverplaatsing is aangemaakt voor één kolom, moet dit ook gebeuren voor de resterende kolommen in de structuur. Zodanig dat er voor elke kolom twee XML-bestanden beschikbaar zijn, één waarin er enkel kolomverlies plaatsvindt en een andere waarin kolomverlies en knooppuntsverplaatsing plaatsvindt. Het volledige script is toegevoegd in Bijlage C.

Naast het opstellen van de verschillende XML-bestanden genereert dit script ook een tekstbestand. Dit bestand bevat de bestandslocaties van de bestanden die opgesteld zijn om kolomverlies te simuleren.

Figuur 37: Flowchart eerste algoritme – Script aanpassen structuur

7.2.4 Script 4: Script lijst resultaten

Het volgend script is een hulpscript dat het gemakkelijker maakt om de resultaten na de analyse op te slaan. Zo stelt het script twee tekstbestanden op waarin de paden naar de bestanden van de resultaten terug te vinden zijn. Eén tekstbestand bevat de paden naar de resultaten in de vorm van de XML-bestanden en het ander de paden naar de engineering rapporten in de vorm van PDF-bestanden. Het script stelt de twee tekstbestanden op door het eerder aangemaakt bestand LijstBestanden uit 7.2.3 in te lezen en aan te passen doormiddel van respectievelijk Resultaten en Rapporten toe te voegen aan de naam van de bestanden. Dit script is terug te vinden in Bijlage D.

Figuur 38: Flowchart eerste algoritme – Script lijst resultaten

7.2.5 Script 5: Script SCIA berekening

Het opstellen van de XML-bestanden en de verschillende tekstbestanden is gebeurd en nu dienen deze XML-bestanden uitgerekend te worden. Voor het uitrekenen van de verschillende bestanden en structuren wordt er gebruik gemaakt van de ESA_XML-tool die bij de installatie van SCIA-engineer automatisch geïnstalleerd werd. Deze tool maakt het mogelijk bestanden en structuren uit te rekenen op de achtergrond. Het script dat hiervoor geschreven werd genereert in principe voor iedere berekening één regel code die dan verder in het *command promt* uitgevoerd wordt. Een voorbeeld van zo een regel code is hieronder terug te vinden. Voor de duidelijkheid werd in deze regel code gebruik gemaakt van de namen van de variabelen en niet van de gegevens zelf. Het script leest de verschillende tekstbestanden die in de vorige scripts zijn opgesteld in en wijst deze dan toe aan de respectievelijke variabele.

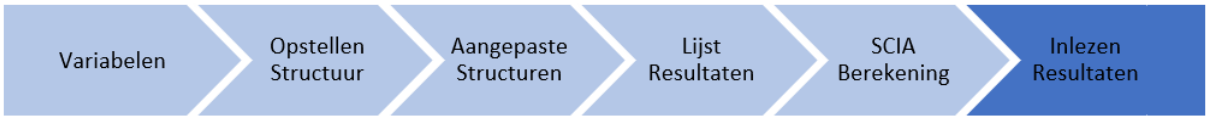
@"Bestandslocatie ESA_XML" LIN "%SCIABestand%" "!INPUTXML!" /sd /tPDF /x"!OUTPUTXML!" /o"!OUTPUTPDF!"

Zo staat de afkorting LIN in de code voor een lineaire analyse. De termen /sd en /tPDF duiden dat er een engineering rapport van de analyse zal worden opgesteld. De overige variabelen in de code spreken voor zich. Het is belangrijk te vermelden dat het op de achtergrond uitvoeren van analyses een aanzienlijke tijd in beslag neemt. De volledige code gebruikt in script 5 kan worden geraadpleegd in Bijlage E.

Figuur 39: Flowchart eerste algoritme – Script SCIA berekening

7.2.6 Script 6: Script inlezen resultaten

Na het uitvoeren van de analyses op de verschillende bestanden en structuren werd er voor ieder bestand of structuur een XML-bestand aangemaakt die de resultaten bevat. Om de resultaten gemakkelijk te raadplegen werd er besloten een script te schrijven die uit elk van deze resultaten bevattende XML-bestanden de gewenste resultaten opzocht en wegschreef naar een nieuw tekstbestand, genaamd LijstResultaten. De code gebruikt in dit script is toegevoegd in Bijlage F.



Figuur 40: Flowchart eerste algoritme – Script inlezen resultaten

Door de opgestelde scripts te combineren kan er kolomverlies gesimuleerd worden. De combinatie van de verschillende scripts gebeurt aan de hand van het script ScriptRun. De code van dit script is terug te vinden in Bijlage G.

Aan de hand van de resultaten van deze simulatie kan de robuustheid van de constructie worden gecontroleerd. Het eerste algoritme toont aan dat het mogelijk is om een algoritme op te stellen voor de simulatie van kolomverlies. Zoals eerder werd beschreven in 7.2.1 is de gegenereerde structuur afhankelijk van enkele inputs. Dit resulteert dat het algoritme enkel structuren genereert met eenzelfde verdiepingshoogte en eenzelfde tussenafstand tussen de kolommen. Dit betekent dat het eerste algoritme enkel in staat is om kolomverlies te simuleren op symmetrische raamwerkstructuren.

7.3 Algoritme 2: Uitvoeren analyse op door gebruiker aangeleverd model

Het eerste algoritme kan kolomverlies simuleren voor symmetrische raamwerkstructuren. Echter is het doel van dit onderzoek niet alleen het simuleren van kolomverlies voor gegenereerde symmetrische raamwerkstructuren, maar ook op door de gebruiker aangeleverde modellen moet kolomverlies kunnen gesimuleerd worden. Om dit mogelijk te maken werd een tweede algoritme opgesteld. De opbouw van dit algoritme diende dan ook te verschillen van dat van het eerste algoritme om zo de beperkingen van het eerste algoritme te overtreffen. Net zoals het eerste algoritme bestaat het tweede algoritme uit meerdere scripts. Onder iedere paragraaf van een script is er een flowchart aanwezig waarop weergegeven is waar dit script zich bevindt in het algoritme.

Alvorens het algoritme kan uitgevoerd worden, dient de gebruiker zelf nog enkele aanpassingen te maken aan zijn model. Zo is het verreist dat in het SCIA-model aan iedere ligger en kolom het juiste label is toegekend. Ook dient een tweede *layer* aangemaakt te worden in het bestand. Tijdens het aanmaken van deze layer dient de optie Enkel constructiemodel aangevinkt te worden en de optie Huidig gebruikte activiteit afgevinkt te worden. Daarnaast dient er in dit model ook het juiste engineering rapport geselecteerd te worden. Het sjabloon voor dit rapport wordt meegeleverd samen met het algoritme. Indien alle aanpassingen gebeurd zijn in het model en het kan uitgerekend worden, moet het geëxporteerd worden naar een XML-bestand. Net zoals bij het engineering rapport zal ook voor het exporteren een sjabloon meegeleverd worden. Het is belangrijk dat het sjabloon gebruikt wordt tijdens het exporteren van het XML-bestand aangezien het algoritme is opgesteld op basis van de structuur van het sjabloon. Indien alle aanpassingen gebeurd zijn en het model geëxporteerd is als XML-bestand kan het algoritme uitgevoerd worden.

7.3.1 Script 1: Script variabelen

Het eerste script in het tweede algoritme is een script genaamd variabelen. Deze zal net zoals bij het eerste algoritme enkele inputs vragen aan de gebruiker. In dit geval zijn dit niet de eigenschappen van de structuur, maar wel de namen van de verschillende bestanden. Zo zal het script zelf de locatie opzoeken van de bestanden. Dit is mogelijk aan de hand van de naam van het bestand en de harde schijf waarop deze is opgeslagen. Net zoals bij het eerste algoritme zal ook dit script de locatie van de tool ESA_XML opzoeken. Ook zal het script hier opnieuw de mappenstructuur aanmaken voor het project en de nodige bestanden verplaatsen naar de juiste mappen. Daarna worden de locaties van alle bestanden, aangemaakte mappen en de ESA_XML-tool weggeschreven naar een tekstbestand onder de naam LijstVariabelen. Dit script is terug te vinden in Bijlage H.

Figuur 41: Flowchart tweede algoritme – Script variabelen

7.3.2 Script 2: Script aanpassen structuren

Na het uitvoeren van het eerste script volgt het script aanpassen structuren. Dit script maakt de verschillende structuren aan waarin enerzijds kolomverlies gesimuleerd is en anderzijds kolomverlies en knooppuntsverplaatsing heeft plaatsgevonden. Dit doet het script door het, door de gebruiker aangeleverd, XML-bestand aan te passen. Door het verschil in opbouw van het eerste en tweede algoritme is er bij het tweede algoritme nog geen lijst beschikbaar met de eigenschappen van de structuur. Hiermee wordt het aantal verdiepingen, verdiepingshoogte, aantal kolommen en tussenafstand tussen de kolommen mee bedoeld. Hierdoor zal het script zelf op zoek moeten gaan naar deze eigenschappen door de coördinaten van de verschillende knopen te analyseren.

Het grootste deel van dit script bestaat er dan ook uit om deze eigenschappen te verkrijgen en verder de nodige gegevens te bepalen. Deze eigenschappen en gegevens worden steeds weggeschreven naar lijsten of *dictionaries*. Een *dictionary* is een datatype in PyCharm die *keys* verbindt met *values*. Een voorbeeld van een dictionary die gebruikt wordt in het tweede script is hieronder voorgesteld. Waarin respectievelijk Rij1, Rij2 en Rij3 de keys zijn en de staven, S1, S2, … de values zijn. Deze dictionary geeft weer welke kolommen zich in welke rij van de structuur bevinden en is gegenereerd door het script door de begin- en eindcoördinaten van de kolommen te analyseren. [16]

dict Rijen Kolommen = {'Rij1': ['S1', 'S2', 'S3'], 'Rij2': ['S10', 'S11', 'S14'], 'Rij3': ['S12', 'S13', 'S15']}

Het script maakt gebruik van meerdere dictionaries, de belangrijkste twee zullen verder besproken worden. De eerste belangrijke dictionary is degene die hierboven weergegeven is. Aan de hand van deze dictionary weet het script welke kolom in welke rij staat en kan het deze kolom verwijderen voor kolomverlies te simuleren. Zoals eerder besproken in paragraaf 6.1.1 dient er naast het verwijderen van de kolom ook een knooppuntsverplaatsing te gebeuren om de dynamische effecten in rekening te brengen. Het script moet dus in staat zijn om te weten welke knopen er boven de te verwijderen kolom staan. De dictionary die deze gegevens bevat voor een voorbeeld is hieronder voorgesteld.

dict Knopen boven kolom = {'S1': ['K2', 'K3', 'K4'], 'S2': ['K3', 'K4'], 'S3': ['K4'], 'S10': ['K7'], 'S11': ['K6', 'K7'], 'S12': ['K8'], 'S13': ['K9', 'K8'], 'S14': ['K5', 'K6', 'K7'], 'S15': ['K10', 'K9', 'K8']}

Nadat het script de nodige gegevens over de structuur verzameld heeft, kan het de structuren aanmaken waarin kolomverlies en knooppuntsverplaatsing plaatsvindt. Zo zal het script aan de hand van de dictionary “dict Rijen Kolommen” weten voor welke kolom kolomverlies gesimuleerd moet worden. Het kolomnummer wordt verkregen door de value op te vragen. Aangezien het tweede algoritme rekent met een door de gebruiker aangeleverd model is het SCIA-bestand dus niet leeg maar bevat het de structuur. In het eerste algoritme werd het verwijderen van een kolom in de structuur gedaan door het volledig object van deze kolom uit het XML-bestand te verwijderen. Dit is echter geen mogelijkheid voor het tweede script aangezien het XML-bestand enkel de structuur in het SCIA-bestand update. Het is wel mogelijk om coördinaten of gegevens van de kolom aan te passen maar niet om deze te verwijderen uit de structuur. De alternatieve methode die gebruikt werd in het tweede algoritme bestaat eruit om gebruik te maken van meerdere layers waarbij de berekening enkel rekent met de structuuronderdelen aanwezig in een bepaalde layer. Om het kolomverlies te simuleren dient de ligger dus enkel verplaatst te worden naar een andere layer. Het veranderen van de layer waarin de staaf zich bevindt wordt gedaan door het

attribuut van de layer in het XML-bestand van deze ligger te verwijderen en nadien opnieuw toe te voegen met een andere layer. Na het veranderen van de layer van de kolom wordt het bestand een eerste keer onder een andere naam opgeslagen. Ook wordt het volledige pad van dit bestand weggeschreven naar het tekstbestand “LijstBestanden”. Een voorbeeld van een naam waaronder het bestand wordt opgeslagen is “StructuurS1ZK”, waarbij ZK staat voor zonder knooppuntsverplaatsing. Doordat het script het kolomnummer van de te verwijderen kolom als value heeft verkregen kan het deze opzoeken in de dictionary “dict Knopen boven kolom”. Nu het script weet welke knopen een verplaatsing moeten ondergaan zal het de coördinaten aanpassen van deze knopen. Hierna zal het script dit bestand opnieuw opslaan onder een andere naam. Ook het pad van dit bestand wordt weggeschreven naar het tekstbestand LijstBestanden. Een voorbeeld van een naam waaronder het bestand wordt opgeslagen is “StructuurS1”. De nodige bestanden voor kolomverlies te simuleren voor één kolom zijn nu aangemaakt. Nu dient het script bovenstaande stappen te herhalen zodat voor iedere kolom in de structuur kolomverlies is gesimuleerd. De volledige code gebruikt in script 2 kan worden geraadpleegd in Bijlage I.

Figuur 42: Flowchart tweede algoritme – Script aanpassen structuren

7.3.3 Script 3: Script lijst resultaten

Het script wat de tekstbestanden aanmaakt met de paden van de verschillende resultaten voor het tweede algoritme is identiek aan Script 4: Script lijst resultaten van het eerste algoritme. De uitleg over dit script is terug te vinden in 7.2.4. Het script is toegevoegd in Bijlage J.

Figuur 43: Flowchart tweede algoritme – Script lijst resultaten

7.3.4 Script 4: SCIA berekening

Het script wat de analyses uitvoert en de resultaten opslaat voor het tweede algoritme is identiek aan Script 5: SCIA berekening van het eerste algoritme. De uitleg over dit script is terug te vinden in 7.2.5. De code va het script kan geraadpleegd worden in Bijlage K.

Figuur 44: Flowchart tweede algoritme – Script SCIA berekening

7.3.5 Script 5: Script Inlezen Resultaten

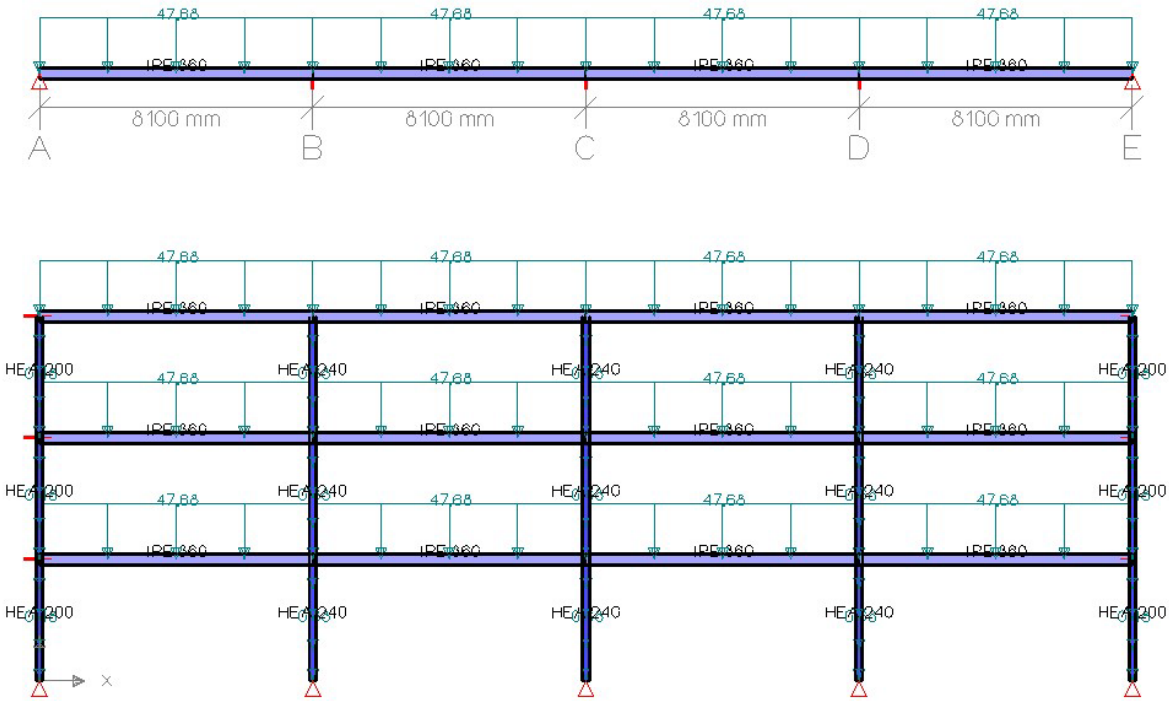
Het script wat de resultaten inleest en wegschrijft naar het tekstbestand LijstResultaten voor het tweede algoritme is identiek aan Script 6: Script inlezen resultaten van het eerste algoritme. De uitleg over dit script is terug te vinden in 7.2.6. Het script is toegevoegd in Bijlage L.

Figuur 45: Flowchart tweede algoritme – Script inlezen resultaten

Door bovenstaande scripts te combineren kan er kolomverlies gesimuleerd worden op door de gebruiker aangeleverde raamwerk structuren. De combinatie van de verschillende scripts gebeurt aan de hand van het script ScriptRun. De code van dit script is terug te vinden in Bijlage M. Met dit algoritme zijn de beperkingen van het eerste algoritme tenietgedaan en is het doel van dit onderzoek bereikt.

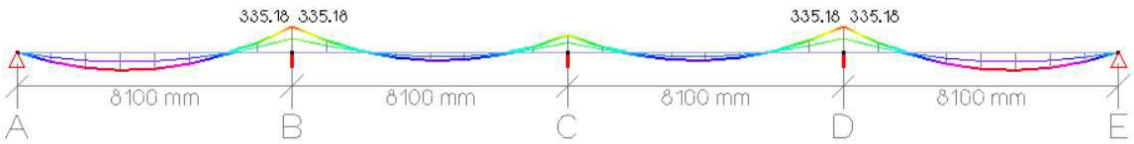
8 Verificatie algoritme

In voorgaande Hoofdstukken werd beschreven hoe de verschillende algoritmen werden opgesteld. Verder zal aan de hand van de resultaten verkregen door een simulatie van kolomverlies het algoritme geverifieerd worden. De verificatie van het algoritme gebeurt door een vergelijking te maken tussen de resultaten van het algoritme en de resultaten van een voorbeeldoefening [17]. De structuur die wordt gebruikt om de vergelijking te maken is weergegeven in Figuur 46. De liggers zijn IPE 360 profielen, De twee buitenste kolommen zijn HEA 200 profielen en de overige kolommen zijn HEA 240 profielen.

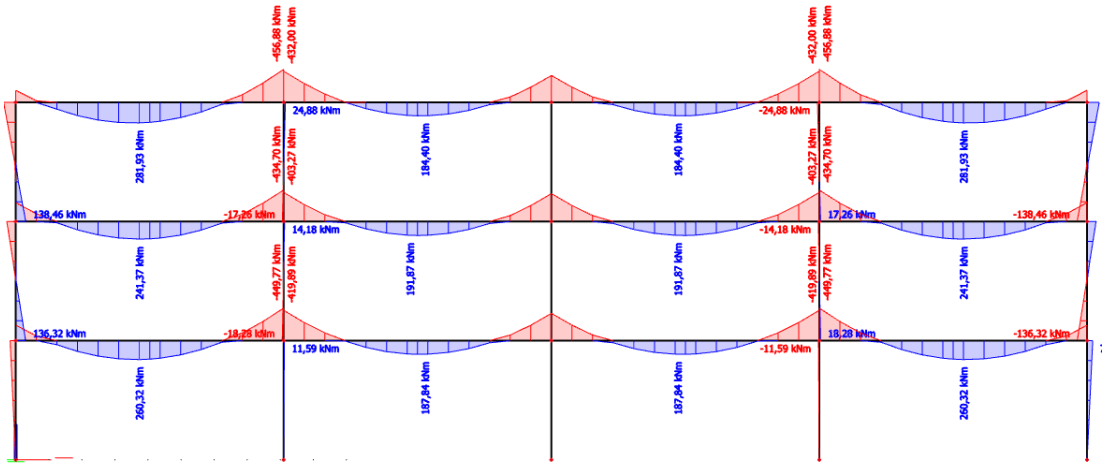


Figuur 46: Structuur Verificatie [17]

Om de resultaten van het algoritme te kunnen verifiëren, dient de structuur ingetekend te worden in SCIA Engineer. Het is niet voldoende om enkel de structuuronderdelen en lasten te modeleren, ook moet het engineering rapport aangepast worden. Een sjabloon van het engineering rapport wordt meegeleverd bij het algoritme. Eens het model klaar was, werd een eerste manuele analyse uitgevoerd. Deze analyse werd uitgevoerd om te controleren of het model correct was gemodelleerd tegenover de structuur uit [17]. De resultaten van de momenten van de structuur uit [17] zijn weergegeven in Figuur 47. Het maximaal moment bedraagt hier 335,18 kN/m. De resultaten van de manuele analyse binnen SCIA Engineer zijn gepresenteerd in Figuur 48. Het maximaal moment bedraagt hier 411,19 kN/m.



Figuur 47: Resultaten momenten vereenvoudigde berekening [17]



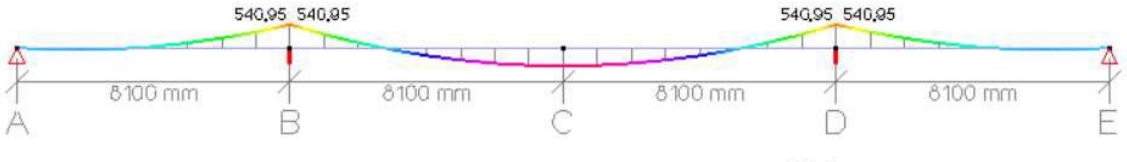
Figuur 48: Resultaten momenten SCIA [13]

Uit deze vergelijking kan geconcludeerd worden dat de resultaten gelijkaardig zijn maar niet exact overeenkomen. Het verschil tussen deze resultaten kan te wijten zijn aan de opbouw van de structuur. Zo wordt de balk op vijf steunpunten in [17] ondersteund door twee scharnieren aan de uiteinden en drie over het midden van de balk verspreide verticale ondersteuningen. Indien de raamwerkstructuur weergegeven in Figuur 48 vereenvoudigd wordt naar een balk op meerdere steunpunten dient er rekening gehouden te worden met de ingeklemde randvoorwaarden van de verbindingen tussen kolommen en liggers. Zo verschillen de randvoorwaarden voor beide modellen volgens de verschillende methoden. Aangezien de resultaten van beide methoden toch gelijkaardig waren werd dit model verder gebruikt voor het algoritme te verifiëren.

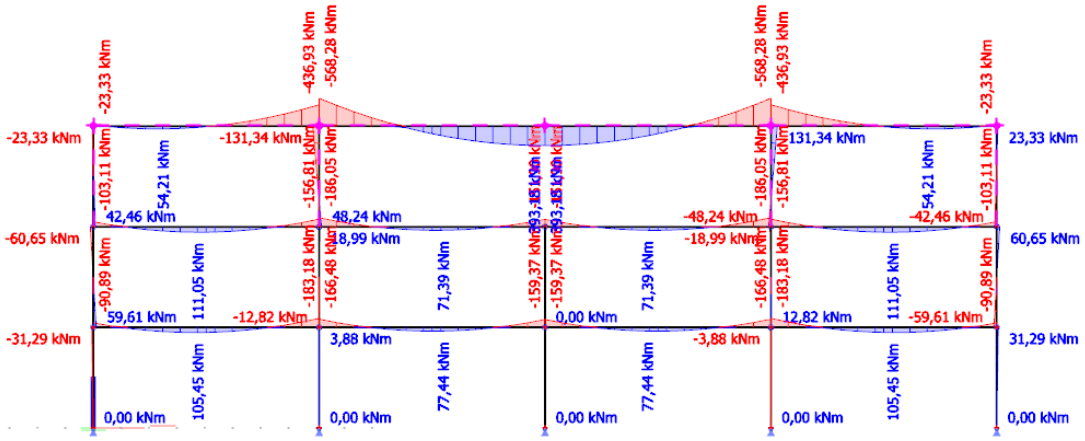
Kolomverlies wordt gesimuleerd in [17] door een accidentele belasting aan te brengen op de structuur. De accidentele belasting die aangebracht werd, bedraagt 29,31 kN/m. Deze accidentele belasting bestaat uit het eigengewicht, de permanente belasting en de variabele belasting. Het eigengewicht van de balk, een IPE 360 ligger, bedraagt 0,56 kN/m. De permanente belasting en variabele belastingen bedragen respectievelijk 25,11 kN/m en 12,15 kN/m. Aangezien het algoritme het nog niet mogelijk maakt om met dergelijke lastencombinaties te werken werd deze accidentele belasting omgerekend naar een permanente belasting. Door de accidentiele belasting te delen door de combinatiefactor van een permanente belasting, die 1,35 bedraagt, kan de accidentiele belasting omgerekend worden naar een permanente belasting. Echter moet van deze permanente belasting het eigengewicht nog afgetrokken worden aangezien SCIA Engineer dit zelf bepaalt. De lijnlast Q die dient aangebracht te worden op de structuur rekening houdend met het eigengewicht is berekend in formule 8.1.

$$Q = (29{,}31/1{,}35) - 0{,}56 = 21{,}15\ kN/m \qquad (8.1)$$

Aan de hand van de lijnlast bepaald in formule 8.1 kan het model in SCIA Engineer aangepast worden zodat kolomverlies op deze structuur kan gesimuleerd worden door het algoritme. Na de analyse van de simulatie van kolomverlies door het algoritme kunnen ook deze resultaten vergeleken worden. De resultaten van de momenten van de structuur waarin kolomverlies optreedt uit [17] zijn weergegeven in Figuur 49. Het maximaal moment bedraagt hier 540,95 kN/m. De resultaten van de simulatie van kolomverlies door het algoritme zijn weergegeven in Figuur 50. Het maximaal moment bedraagt hier 568,28 kN/m.

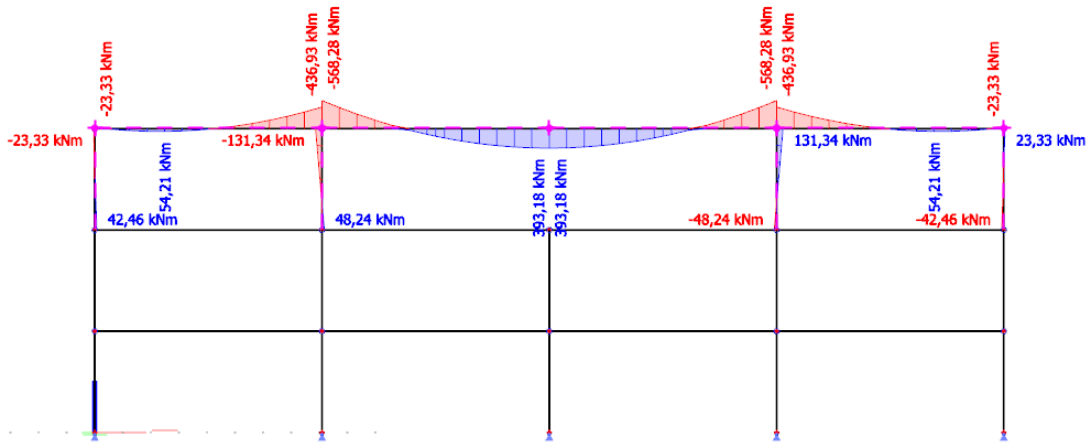


Figuur 49: Resultaten momenten met kolomverlies [13]



Figuur 50: Resultaten momenten simulatie kolomverlies algoritme [13]

Ter verduidelijking van de resultaten van het algoritme werd het uitgerekende model geopend in SCIA Engineer. Hier werd de visuele weergave van de resultaten beperkt tot enkel de bovenste verdieping. De resultaten van enkel de bovenste verdieping worden weergegeven in Figuur 51.



Figuur 51: Resultaten momenten simulatie kolomverlies algoritme bovenste verdieping [13]

De resultaten van de simulatie van kolomverlies verschillen zeer weinig ten opzichte van elkaar. Het verschil van deze afwijking kan te wijten zijn aan de al eerder besproken randvoorwaarden van de bovenste verdieping. Zo wordt de structuur in [17] zowel scharnierend als verticaal ondersteund terwijl het algoritme deze steunpunten als ingeklemd beschouwd. Daarnaast Is er een sprong in de momentenlijn in de resultaten uit het algoritme waar te nemen. De sprong in de momentenlijn Is het gevolg van een overdracht van moment naar de ondersteunende kolom. Deze overdracht vindt plaats omwille van de ingeklemde modellering van de kolommen ten opzichte van de liggers. Uit de vergelijking van de simulatie van kolomverlies kan geconcludeerd worden dat het algoritme in staat is om correct kolomverlies te simuleren voor raamwerkstructuren.

9 Conclusie

De doelstelling van deze masterproef is het opstellen van een algoritme dat kolomverlies simuleert voor volledige structuren op basis van de huidige normen op vlak van robuustheid. Dit onderzoek kan onderverdeeld worden in een vooronderzoek en een hoofdonderzoek.

Voorafgaand aan het vooronderzoek werd een studie van de literatuur uitgevoerd. Aan de hand van deze studie werden de randcondities voor de simulatie van kolomverlies bepaald. Uit de literatuurstudie kan geconcludeerd worden dat de huidige Europese norm als eerder onveilig beschouwt moet worden. Daarnaast wordt duidelijk dat de Amerikaanse norm een veiligere rotatiehoek voorstelt.

Het vooronderzoek bestond eruit om de mogelijkheden van de simulatie van kolomverlies in verschillende commerciële analysesoftware te onderzoeken. Uit dit vooronderzoek kon geconcludeerd worden dat beide commerciële analysesoftware de mogelijkheden bevatten om kolomverlies te simuleren. Echter biedt SCIA Engineer de mogelijkheid om gebruik te maken van de ESA_XML tool. Dit maakt het mogelijk om gebruik te maken van algoritmen en analyses in de achtergrond uit te voeren.

De doelstelling van het onderzoek is behaald aangezien er twee algoritmen opgesteld zijn die beide kolomverlies simuleren op structuren. Door de resultaten te vergelijken van de simulatie van kolomverlies door het algoritme in dit onderzoek met een voorbeeldoefening, kon het algoritme geverifieerd worden. Uit de verificatie van het algoritme kan geconcludeerd worden dat het opgestelde algoritme in staat is om correct kolomverlies te simuleren voor raamwerkstructuren.

Deze masterproef biedt de basis naar verder onderzoek omtrent simulatie van kolomverlies. Zo kan het algoritme zelf zowel geoptimaliseerd worden als verder uitgebreid worden. Daarnaast kan dit onderzoek de informatie bieden aan de verschillende commerciële software om hun software verder uit te breiden.

Referentielijst

[1] U. Starossek en M. Haberland, „Approaches to measures of structural robustness,” *Structure and Infrastructure Engineering,* pp. 625-631, 2011 vol. 7, 7-8.

[2] C. Pearson en N. Delatte, „Ronan Point Apartment Tower Collapse and its Effecton Building Codes,” *Journal of Performance of Constructed Facilities,* vol. 19, nr. 2, pp. 172-177, 2005.

[3] „Bouwsector faalt bij constructieve veiligheid,” *De ingenieur,* 18 oktober 2018.

[4] D. Van den Buijs, „Loods ingestort in de Antwerpse haven: "Heftruck reed tegen steunpilaar",” *VRT NWS,* 21 september 2021.

[5] L. Saelens, Simulatie van progressieve instroting van raamwerken in rekensoftwaremet behulp van plastische schranieren [Eindwerk], 2020.

[6] NBN EN 1991-1-7, „Eurocode 1 - Belastingen op constructies - Deel 1-7: Algemene belastingen - Buitengewone belastingen: stootbelastingen en ontploffingen,” CEN, 2006.

[7] NBN EN 1990 ANB, „Eurocode 0 - Grondslag voor het constructief ontwerp - Nationale bijlage,” CEN, 2021.

[8] CEN, „NBN EN 1992-1-1,” 2005.

[9] U.S. Army corps of engineers; Naval facilities engineering command; Airforce civil engineer support agency, „UFC 4-023-03: Design of buildings to resist progressive collapse,” 2016.

[10] U.S. Army corps of engineers; Naval facilities engineering command; Airforce civil engineer support agency, „UFC 3-301-01: Structural engineering,” 2022.

[11] J.-F. Demonceau, J.-P. Jaspart en A. Corman, „Ductility assessment of structural steel and composite joints,” 2019.

[12] Buldsoft, *Diamonds 2021 R.03,* 2021.

[13] SCIA nv, *SCIA Engineer 21.1.1028,* 2021.

[14] A. k. Chopra, Dynamic of structures: Theory and applications to earthquake engineering, New Jersey: Prentice-Hall, 1995.

[15] JetBrains, *PyCharm 2021.3.3,* 2021.

[16] N. Novak, *Python dictionary,* New York, 2019.

[17] Project team WG6.T2, „Robustness rules in material related eurocode parts,” 2020.

Bijlagen

Bijlage A: Algoritme 1 – Script variabelen

```
import os
import pathlib
import shutil


bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
LocatiemapProject = bestandslocatie[:-6]+"Projecten"

Projectnaam = '\\'+str(input('Projectnaam: '));
HardeSchijfSCIA = str(input('Harde schijf waarop SCIA is
geinstalleerd (bv C of D): '));
Verdiepingshoogte = str(input('Verdiepingshoogte: '));
AantalVerdiepingen = str(input('Aantal verdiepingen: '))
Tussenafstand = str(input('Tussenafstand: '))
AantalKolommen = str(input('Aantal kolommen: '))
Lijnlast = str(input('Lijnlast (N): '))

NaamSCIA = "ESA_XML.exe"
path = HardeSchijfSCIA.upper() + ":\\"
for root, dir, files in os.walk(path):
    if NaamSCIA in files:
        Pad_ESA_XML_esa=(os.path.join(root, NaamSCIA))


Pad_Project_Map = str(LocatiemapProject+Projectnaam)
Pad_Structuren_Kolomverlies = str(Pad_Project_Map + '\\' + '
Structuren' + '\\' + 'Kolomverlies')
Pad_Structuren_Knooppuntsverplaatsing = str(Pad_Project_Map + '
\\' + 'Structuren' + '\\' + 'Knooppuntsverplaatsing')

os.makedirs(Pad_Project_Map)
os.makedirs(Pad_Structuren_Kolomverlies)
os.makedirs(Pad_Structuren_Knooppuntsverplaatsing)

LijstVariabelen = bestandslocatie+ '\\' + "LijstVariabelen.txt"

bestand = open(LijstVariabelen,"w")
bestand.write("Verdiepingshoogte:")
bestand.write('\n')
bestand.write(Verdiepingshoogte)
bestand.write('\n')
bestand.write("Aantal Verdiepingen:")
bestand.write('\n')
bestand.write(AantalVerdiepingen)
bestand.write('\n')
bestand.write("Tussenafstand:")
bestand.write("\n")
bestand.write(Tussenafstand)
bestand.write("\n")
```

```
47 bestand.write("Aantal Kolommen:")
48 bestand.write("\n")
49 bestand.write(AantalKolommen)
50 bestand.write("\n")
51 bestand.write("Lijnlast:")
52 bestand.write("\n")
53 bestand.write(Lijnlast)
54 bestand.write("\n")
55 bestand.write("Pad Project Map:")
56 bestand.write("\n")
57 bestand.write(Pad_Project_Map)
58 bestand.write("\n")
59
60 bestand.write("Pad Structuren Kolomverlies:")
61 bestand.write("\n")
62 bestand.write(Pad_Structuren_Kolomverlies)
63 bestand.write("\n")
64
65 bestand.write("Pad Structuren Knooppuntsverplaatsing:")
66 bestand.write("\n")
67 bestand.write(Pad_Structuren_Knooppuntsverplaatsing)
68 bestand.write("\n")
69
70 bestand.write("Pad ESA_XML.esa:")
71 bestand.write("\n")
72 bestand.write(Pad_ESA_XML_esa)
73 bestand.write("\n")
74
75 bestand.close
76
77
78 bestand = open(LijstVariabelen,"r")
79
80 shutil.copyfile(LijstVariabelen, Pad_Project_Map+'\\'+"
   LijstVariabelen.txt")
81
82 shutil.copyfile(bestandslocatie+'\\'+"BasefileStructuurV2.esa"
   , Pad_Project_Map+'\\'+"BasefileStructuurV2.esa")
83 shutil.copyfile(bestandslocatie+'\\'+"BaseFileStructuurV2.xml.
   def", Pad_Project_Map+'\\'+"BaseFileStructuurV2.xml.def")
84
85 shutil.copyfile(bestandslocatie+'\\'+"BaseFileStructuurV2.xml"
   , Pad_Structuren_Kolomverlies+'\\'+"BaseFileStructuurV2.xml")
86 shutil.copyfile(bestandslocatie+'\\'+"BaseFileStructuurV2.xml.
   def", Pad_Structuren_Kolomverlies+'\\'+"BaseFileStructuurV2.xml
   .def")
87
88 shutil.copyfile(bestandslocatie+'\\'+"BaseFileStructuurV2.xml"
   , Pad_Structuren_Knooppuntsverplaatsing+'\\'+"
```

```
BaseFileStructuurV2.xml")
shutil.copyfile(bestandslocatie+'\\'+"BaseFileStructuurV2.xml.def", Pad_Structuren_Knooppuntsverplaatsing+'\\'+"BaseFileStructuurV2.xml.def")


print('ScriptVariabelen klaar')
```

Bijlage B: Algoritme 1 – Script opstellen structuur

```
import xml.etree.ElementTree as ET;
import pathlib

bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
bestand = open(bestandslocatie+'\\'+"LijstVariabelen.txt", "r")
b = bestand.readlines()
lijst = []
for line in b:
    if line[-1] == '\n':
        lijst.append(line[:-1])
    else:
        lijst.append(line)
Verdiepingshoogte = int(lijst[1])
AantalVerdiepingen = int(lijst[3])
Tussenafstand = int(lijst[5])
AantalKolommen = int(lijst[7])
Lijnlast = int(lijst[9])
Pad_Project_Map = lijst[11]
Pad_Structuren_Kolomverlies = lijst[13]
Pad_Pad_Structuren_Knooppuntsverplaatsing = lijst[15]
print(Pad_Project_Map)

mytree =ET.parse(bestandslocatie+'\\'+'BasefileStructuurV2.xml'
)
myroot= mytree.getroot()
X=int(1)

print('Knopen:')
i = 0
j = 0
H = 0

while i < AantalVerdiepingen+1:
    G = 0

    while j < AantalKolommen:
        K = str(G)
        L = str(H)
        B = str(X)
        A = str('K' + B)
        print("     "+A)
        root = ET.SubElement(myroot[7][0], 'obj')
        root.set('id', B)
        root.set('nm', A)

        root = ET.SubElement(myroot[7][0][X], 'p0')
        root.set('v', A)

        root = ET.SubElement(myroot[7][0][X], 'p1')
```

```
49          root.set('v', K)
50
51          root = ET.SubElement(myroot[7][0][X], 'p2')
52          root.set('v', '0')
53
54          root = ET.SubElement(myroot[7][0][X], 'p3')
55          root.set('v', L)
56
57          G = G + Tussenafstand
58          X = X + 1
59          j = j + 1
60
61      H = H + Verdiepingshoogte
62      j = 0
63      i = i + 1
64
65
66  print('kolommen:')
67  XX = int(1)
68  II = int(1)
69  JJ = int(0)
70  X = int(1)
71  while II < AantalVerdiepingen+1:
72      while JJ < AantalKolommen:
73          BB = str(XX)
74          AA = str('S' + BB)
75          CC = str('K' + BB)
76          DD = XX + AantalKolommen
77          EE = str(DD)
78          FF = str('K' + EE)
79          print("   "+AA)
80          print('      Begin ' + CC)
81          print('      Einde ' + FF)
82          root = ET.SubElement(myroot[8][0], 'obj')
83          root.set('id', BB)
84          root.set('nm', AA)
85
86          root = ET.SubElement(myroot[8][0][XX], 'p0')
87          root.set('v', AA)
88
89          root = ET.SubElement(myroot[8][0][XX], 'p1')
90          root.set('i', BB)
91          root.set('n', CC)
92
93          root = ET.SubElement(myroot[8][0][XX], 'p2')
94          root.set('i', EE)
95          root.set('n', FF)
96
97          root = ET.SubElement(myroot[8][0][XX], 'p3')
```

```
 98          root.set('i', '3')
 99          root.set('n', 'CS3 - IPE300')
100
101          root = ET.SubElement(myroot[8][0][XX], 'p4')
102          root.set('v', '0')
103          root.set('t', 'standaard')
104
105          root = ET.SubElement(myroot[8][0][XX], 'p5')
106          root.set('v', '1')
107          root.set('t', 'Midden')
108
109          root = ET.SubElement(myroot[8][0][XX], 'p6')
110          root.set('v', '0')
111
112          root = ET.SubElement(myroot[8][0][XX], 'p7')
113          root.set('v', '0')
114
115          root = ET.SubElement(myroot[8][0][XX], 'p8')
116          root.set('t', '')
117
118          root = ET.SubElement(myroot[8][0][XX][8], 'h')
119
120          root = ET.SubElement(myroot[8][0][XX][8][0], 'h1')
121          root.set('t', 'Node')
122
123          root = ET.SubElement(myroot[8][0][XX][8][0], 'h2')
124          root.set('t', 'Edge')
125
126          root = ET.SubElement(myroot[8][0][XX][8], 'row')
127          root.set('id', '0')
128
129          root = ET.SubElement(myroot[8][0][XX][8][1], 'p1')
130          root.set('i', BB)
131          root.set('n', CC)
132
133          root = ET.SubElement(myroot[8][0][XX][8][1], 'p2')
134          root.set('v', '0')
135          root.set('t', 'Lijn')
136
137          root = ET.SubElement(myroot[8][0][XX][8], 'row')
138          root.set('id', '1')
139
140          root = ET.SubElement(myroot[8][0][XX][8][2], 'p1')
141          root.set('i', EE)
142          root.set('n', FF)
143
144          JJ = JJ + 1
145
146          XX = XX+1
```

```
147     II = II+1
148     JJ = 0
149 
150 #liggers
151 print('liggers:')
152 III = int(1)
153 JJJ =int(0)
154 XX = (AantalKolommen*(AantalVerdiepingen))+1
155 XXX = AantalKolommen+1
156 while III != AantalVerdiepingen+1:
157     while JJJ != AantalKolommen-1:
158         BBB = str(XXX)
159         BB = str(XX)
160         AA = str('S' + BB)
161 
162         CC = str('K' + BBB)
163         DD = XXX+1
164         EE = str(DD)
165         FF = str('K' + EE)
166         print("   "+AA)
167 
168         print('      Begin ' + CC)
169         print('      Einde ' + FF)
170 
171         root = ET.SubElement(myroot[8][0], 'obj')
172         root.set('id', BB)
173         root.set('nm', AA)
174 
175         root = ET.SubElement(myroot[8][0][XX], 'p0')
176         root.set('v', AA)
177 
178         root = ET.SubElement(myroot[8][0][XX], 'p1')
179         root.set('i', BB)
180         root.set('n', CC)
181 
182         root = ET.SubElement(myroot[8][0][XX], 'p2')
183         root.set('i', EE)
184         root.set('n', FF)
185 
186         root = ET.SubElement(myroot[8][0][XX], 'p3')
187         root.set('i', '3')
188         root.set('n', 'CS3 - IPE300')
189 
190         root = ET.SubElement(myroot[8][0][XX], 'p4')
191         root.set('v', '0')
192         root.set('t', 'standaard')
193 
194         root = ET.SubElement(myroot[8][0][XX], 'p5')
195         root.set('v', '1')
```

```
196         root.set('t', 'Midden')
197 
198         root = ET.SubElement(myroot[8][0][XX], 'p6')
199         root.set('v', '0')
200 
201         root = ET.SubElement(myroot[8][0][XX], 'p7')
202         root.set('v', '0')
203 
204         root = ET.SubElement(myroot[8][0][XX], 'p8')
205         root.set('t', '')
206 
207         root = ET.SubElement(myroot[8][0][XX][8], 'h')
208 
209         root = ET.SubElement(myroot[8][0][XX][8][0], 'h1')
210         root.set('t', 'Node')
211 
212         root = ET.SubElement(myroot[8][0][XX][8][0], 'h2')
213         root.set('t', 'Edge')
214 
215         root = ET.SubElement(myroot[8][0][XX][8], 'row')
216         root.set('id', '0')
217 
218         root = ET.SubElement(myroot[8][0][XX][8][1], 'p1')
219         root.set('i', BB)
220         root.set('n', CC)
221 
222         root = ET.SubElement(myroot[8][0][XX][8][1], 'p2')
223         root.set('v', '0')
224         root.set('t', 'Lijn')
225 
226         root = ET.SubElement(myroot[8][0][XX][8], 'row')
227         root.set('id', '1')
228 
229         root = ET.SubElement(myroot[8][0][XX][8][2], 'p1')
230         root.set('i', EE)
231         root.set('n', FF)
232 
233         JJJ = JJJ + 1
234         XX = XX + 1
235         XXX = XXX + 1
236     III = III + 1
237     XXX = XXX + 1
238     JJJ = 0
239 X1 = int(1)
240 #supports
241 print('supports:')
242 while X1 != AantalKolommen+1:
243     A1 = str(X1)
244     B1 = str('Sn'+A1)
```

```
245     C1 = str('K' + A1)
246     print("   "+C1)
247
248     root = ET.SubElement(myroot[9][0], 'obj')
249     root.set('id', A1)
250     root.set('nm', B1)
251
252     root = ET.SubElement(myroot[9][0][X1], 'p0')
253     root.set('v', B1)
254
255     root = ET.SubElement(myroot[9][0][X1], 'p1')
256     root.set('t', '')
257
258     root = ET.SubElement(myroot[9][0][X1][1], 'h')
259
260     root = ET.SubElement(myroot[9][0][X1][1][0], 'h0')
261     root.set('t','Member Type')
262
263     root = ET.SubElement(myroot[9][0][X1][1][0], 'h1')
264     root.set('t', 'Member Type Name')
265
266     root = ET.SubElement(myroot[9][0][X1][1][0], 'h2')
267     root.set('t', 'Member Id')
268
269     root = ET.SubElement(myroot[9][0][X1][1][0], 'h3')
270     root.set('t', 'Member Name')
271
272     root = ET.SubElement(myroot[9][0][X1][1], 'row')
273     root.set('id','0')
274
275     root = ET.SubElement(myroot[9][0][X1][1][1], 'p0')
276     root.set('v', '{39A7F468-A0D4-4DFF-8E5C-5843E1807D13}')
277
278     root = ET.SubElement(myroot[9][0][X1][1][1], 'p1')
279     root.set('v', 'EP_DSG_Elements.EP_StructNode.1')
280
281     root = ET.SubElement(myroot[9][0][X1][1][1], 'p2')
282     root.set('v', A1)
283
284     root = ET.SubElement(myroot[9][0][X1][1][1], 'p3')
285     root.set('v', C1)
286
287     root = ET.SubElement(myroot[9][0][X1], 'p2')
288     root.set('v', '0')
289     root.set('t', 'Standaard')
290
291     root = ET.SubElement(myroot[9][0][X1], 'p3')
292     root.set('v', '1')
293     root.set('t', 'Vast')
```

```
294
295     root = ET.SubElement(myroot[9][0][X1], 'p4')
296     root.set('v', '1')
297     root.set('t', 'Vast')
298
299     root = ET.SubElement(myroot[9][0][X1], 'p5')
300     root.set('v', '1')
301     root.set('t', 'Vast')
302
303     root = ET.SubElement(myroot[9][0][X1], 'p6')
304     root.set('v', '1')
305     root.set('t', 'Vast')
306
307     root = ET.SubElement(myroot[9][0][X1], 'p7')
308     root.set('v', '1')
309     root.set('t', 'Vast')
310
311     root = ET.SubElement(myroot[9][0][X1], 'p8')
312     root.set('v', '1')
313     root.set('t', 'Vast')
314
315     X1 = X1 + 1
316 #Lasten Op Staaf
317 print('lasten:')
318 X2 = 1
319 C2 = 1
320 while C2 != (AantalKolommen-1)*AantalVerdiepingen +
    AantalKolommen*AantalVerdiepingen :
321     Lijnlaststring = str(Lijnlast)
322     LijnlastString = str('-'+ Lijnlaststring)
323     A2 = str(X2)
324     B2 = str('Lijnlast' + A2)
325
326     print("   "+B2)
327     C2 = AantalKolommen*(AantalVerdiepingen)+X2
328
329     StringC2 = str(C2)
330     D2 = str('S'+StringC2)
331     print("      "+D2)
332     root = ET.SubElement(myroot[10][0], 'obj')
333     root.set('id', A2)
334     root.set('nm', B2)
335
336     root = ET.SubElement(myroot[10][0][X2], 'p0')
337     root.set('v', B2)
338
339     root = ET.SubElement(myroot[10][0][X2], 'p1')
340     root.set('i', '2')
341     root.set('n', 'BG2')
```

```
342
343     root = ET.SubElement(myroot[10][0][X2], 'p2')
344     root.set('t', '')
345
346     root = ET.SubElement(myroot[10][0][X2][2], 'h')
347
348     root = ET.SubElement(myroot[10][0][X2][2][0], 'h0')
349     root.set('t', 'Member Type')
350
351     root = ET.SubElement(myroot[10][0][X2][2][0], 'h1')
352     root.set('t', 'Member Type Name')
353
354     root = ET.SubElement(myroot[10][0][X2][2][0], 'h2')
355     root.set('t', 'Member Name')
356
357     root = ET.SubElement(myroot[10][0][X2][2], 'row')
358     root.set('id', '0')
359
360     root = ET.SubElement(myroot[10][0][X2][2][1], 'p0')
361     root.set('v', '{ECB5D684-7357-11D4-9F6C-00104BC3B443}')
362
363     root = ET.SubElement(myroot[10][0][X2][2][1], 'p1')
364     root.set('v', 'EP_DSG_Elements.EP_Beam.1')
365
366     root = ET.SubElement(myroot[10][0][X2][2][1], 'p2')
367     root.set('v', D2)
368
369     root = ET.SubElement(myroot[10][0][X2], 'p3')
370     root.set('v', '2')
371     root.set('t', 'Z')
372
373     root = ET.SubElement(myroot[10][0][X2], 'p4')
374     root.set('v', '0')
375     root.set('t', 'Kracht')
376
377     root = ET.SubElement(myroot[10][0][X2], 'p5')
378     root.set('v', '0')
379     root.set('t', 'Uniform')
380
381     root = ET.SubElement(myroot[10][0][X2], 'p6')
382     root.set('v', LijnlastString)
383
384     root = ET.SubElement(myroot[10][0][X2], 'p7')
385     root.set('v', '0')
386
387     root = ET.SubElement(myroot[10][0][X2], 'p8')
388     root.set('v', '0')
389     root.set('t', 'GCS')
390
```

```
391     root = ET.SubElement(myroot[10][0][X2], 'p9')
392     root.set('v', '0')
393     root.set('t', 'Length')
394 
395     root = ET.SubElement(myroot[10][0][X2], 'p10')
396     root.set('v', '0')
397 
398     root = ET.SubElement(myroot[10][0][X2], 'p11')
399     root.set('v', '1')
400 
401     root = ET.SubElement(myroot[10][0][X2], 'p12')
402     root.set('v', '1')
403     root.set('t', 'Rela')
404 
405     root = ET.SubElement(myroot[10][0][X2], 'p13')
406     root.set('v', '0')
407     root.set('t', 'From start')
408 
409     root = ET.SubElement(myroot[10][0][X2], 'p14')
410     root.set('v', '0')
411 
412     root = ET.SubElement(myroot[10][0][X2], 'p15')
413     root.set('v', '0')
414 
415     X2 = X2 + 1
416 
417 
418 ET.indent(mytree, space="\t", level=0)
419 mytree.write(Pad_Project_Map+'\\'+'Structuur.xml', encoding="
    utf-8")
420 
421 print('ScriptOpstellenStructuur klaar')
422 
423 
```

Bijlage C: Algoritme 1 – Script aanpassen structuur

```
 1 import xml.etree.ElementTree as ET;
 2 import pathlib
 3
 4
 5 bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
 6 bestand = open(bestandslocatie+'\\'+"LijstVariabelen.txt","r")
 7 b = bestand.readlines()
 8 lijst = []
 9 for line in b:
10     if line[-1] == '\n':
11         lijst.append(line[:-1])
12     else:
13         lijst.append(line)
14 Verdiepingshoogte = int(lijst[1])
15 AantalVerdiepingen = int(lijst[3])
16 Tussenafstand = int(lijst[5])
17 AantalKolommen = int(lijst[7])
18 Lijnlast = int(lijst[9])
19 Pad_Project_Map = lijst[11]
20 Pad_Structuren_Kolomverlies = lijst[13]
21 Pad_Pad_Structuren_Knooppuntsverplaatsing = lijst[15]
22
23 i = 1
24 j = 1
25 k = 1
26 a = 0
27 b = 1
28 d = 1
29 rad = 0.2
30
31 bestand = open(Pad_Project_Map+'\\' + "LijstBestanden.txt", "w"
   )
32
33 while i != AantalVerdiepingen + 1:
34     Aantalknopenaantepassen =AantalVerdiepingen + 1 - i
35     while j != AantalKolommen + 1:
36
37
38         e = str(d)
39         mytree = ET.parse(Pad_Project_Map + '\\' + 'Structuur.
   xml')
40         myroot = mytree.getroot()
41         myroot[8][0].remove(myroot[8][0][d])
42
43         l = i
44         k=1
45
46         m = str(Pad_Structuren_Kolomverlies + '\\' + 'Structuur
   ' + e + 'ZK' + '.xml')
```

```
47          ET.indent(mytree, space="\t", level=0)
48          mytree.write(m, encoding="utf-8")
49
50
51          bestand.write(Pad_Structuren_Kolomverlies + '\\' + '
   Structuur' + e + 'ZK' + '.xml')
52          bestand.write('\n')
53
54          while k != Aantalknopenaantepassen + 1:
55              Kn = l * AantalKolommen + j
56              n = str(Kn)
57              s = str('K' + n)
58
59              xCo = a * Tussenafstand
60              xco = str(xCo)
61
62              zCo = (l * Verdiepingshoogte) - (rad *
   Tussenafstand)
63              zco = str(zCo)
64
65              myroot[7][0][Kn].clear()
66              myroot[7][0][Kn].set('id', n)
67              myroot[7][0][Kn].set('nm', s)
68
69              root = ET.SubElement(myroot[7][0][Kn], 'p0')
70              root.set('v', s)
71
72              root = ET.SubElement(myroot[7][0][Kn], 'p1')
73              root.set('v', xco)
74
75              root = ET.SubElement(myroot[7][0][Kn], 'p2')
76              root.set('v', '0')
77
78              root = ET.SubElement(myroot[7][0][Kn], 'p3')
79              root.set('v', zco)
80              l = l + 1
81              k = k+1
82
83          m = str(Pad_Pad_Structuren_Knooppuntsverplaatsing + '\\
   ' + 'Structuur' + e + '.xml')
84          ET.indent(mytree, space="\t", level=0)
85          mytree.write(m, encoding="utf-8")
86          j = j+1
87          a = a+1
88          d = d + 1
89
90          bestand.write(Pad_Pad_Structuren_Knooppuntsverplaatsing
    + '\\' + 'Structuur' + e + '.xml')
91          bestand.write('\n')
```

```
 92
 93
 94      i = i+1
 95      j = 1
 96      a = 0
 97
 98 bestand = open(Pad_Project_Map+'\\' + "LijstBestanden.txt","r"
    )
 99
100
101 print('ScriptAangepasteStructuren klaar')
102
103
```

Bijlage D: Algoritme 1 – Script lijst resultaten

```
1
2  import pathlib
3
4  bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
5  bestand1 = open(bestandslocatie+'\\'+"LijstVariabelen.txt","r")
6
7  b1 = bestand1.readlines()
8  lijst1 = []
9  for line in b1:
10     if line[-1] == '\n':
11         lijst1.append(line[:-1])
12     else:
13         lijst1.append(line)
14 Verdiepingshoogte = int(lijst1[1])
15 AantalVerdiepingen = int(lijst1[3])
16 Tussenafstand = int(lijst1[5])
17 AantalKolommen = int(lijst1[7])
18 Lijnlast = int(lijst1[9])
19 Pad_Project_Map = lijst1[11]
20 Pad_Structuren_Kolomverlies = lijst1[13]
21 Pad_Pad_Structuren_Knooppuntsverplaatsing = lijst1[15]
22
23 bestand2 = open(Pad_Project_Map+'\\' + "LijstBestanden.txt","r"
   )
24 b2 = bestand2.readlines()
25 lijst = []
26 i=0
27 for line in b2:
28     if line[-1] == '\n':
29         lijst.append(line[:-1])
30         i = i + 1
31     else:
32         lijst.append(line)
33
34 bestand3 = open(Pad_Project_Map+'\\' + "
   LijstBestandenResultaten.txt","w")
35 j = 0
36 while j < i:
37     var1 = str(lijst[j])
38     var2 = var1[:-4]
39     var3 = var2 + 'Resultaten.xml'
40
41     bestand3.write(var3)
42     bestand3.write('\n')
43
44     j = j + 1
45
46 bestand3.close
47
```

```
48 bestand4 = open(Pad_Project_Map+'\\' + "LijstBestandenRapporten
   .txt","w")
49 j = 0
50 while j < i:
51     var1 = str(lijst[j])
52     var2 = var1[:-4]
53     var3 = var2 + 'Rapport.pdf'
54
55     bestand4.write(var3)
56     bestand4.write('\n')
57
58     j = j + 1
59
60 bestand4.close
61
62
63 print('ScriptLijstResultaten klaar')
```

Bijlage E: Algoritme 1 – SCIA berekening

```
@echo off
@SETLOCAL enabledelayedexpansion

SET bestandslocatie1=%~dp0
set bestandslocatie= %bestandslocatie1:~0,-1%

set LijstVariabelen= %bestandslocatie%\LijstVariabelen.txt
set count1=0
for /f "tokens=*" %%x in (!bestandslocatie!\LijstVariabelen.txt
) do (
    set /a count1+=1
    set var1[!count1!]=%%x
)

set count2=0
for /f "tokens=*" %%x in (!var1[12]!\LijstBestanden.txt) do (
    set /a count2+=1
    set var2[!count2!]=%%x
)

set count3=0
for /f "tokens=*" %%x in (!var1[12]!\LijstBestandenResultaten.
txt) do (
    set /a count3+=1
    set var3[!count3!]=%%x
)

set count4=0
for /f "tokens=*" %%x in (!var1[12]!\LijstBestandenRapporten.
txt) do (
    set /a count4+=1
    set var4[!count4!]=%%x
)

@Call :Initialize

@SET SCIABestand=!var1[12]!\BaseFileStructuurV2.esa
for /L %%i in (1,1,%count2%) do (
    @SET INPUTXML=!var2[%%i]!
    @SET OUTPUTXML=!var3[%%i]!
    @SET OUTPUTPDF=!var4[%%i]!

    @Call :Calculate LIN "%SCIABestand%" "!INPUTXML!" "!
OUTPUTXML!" "!OUTPUTPDF!"
)

@ENDLOCAL

::@TIMEOUT /T 5
```

```
46 ping -n 5 127.0.0.1
47
48 Exit
49
50
51
52 :Calculate
53
54 @ECHO.--START--
55
56 @ECHO.Berekenen...
57
58 @ECHO.arg1--%1--
59 @ECHO.arg2--%2--
60 @ECHO.arg3--%3--
61 @ECHO.arg4--%4--
62 @ECHO.cmd--@%ProgExe% %1 %2 %3 /sd /tPDF /x%4 /o%5-
63
64 @%ProgExe% %1 %2 %3 /sd /tPDF /x%4 /o%5
65
66 @ECHO.--EINDE--
67
68 @ECHO.Error Level = %ERRORLEVEL%
69
70 @Exit /B
71
72
73
74 :Initialize
75
76 @SET Pad= !var1[18]!
77 @SET ProgExe="!Pad:~1!"
78
79
80 @Exit /B
```

Bijlage F: Algoritme 1 – Script inlezen resultaten

```
import xml.etree.ElementTree as ET;
import shutil
import pathlib

bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
bestand1 = open(bestandslocatie+'\\'+"LijstVariabelen.txt","r")
b1 = bestand1.readlines()

lijst1 = []
for line in b1:
    if line[-1] == '\n':
        lijst1.append(line[:-1])
    else:
        lijst1.append(line)
Verdiepingshoogte = int(lijst1[1])
AantalVerdiepingen = int(lijst1[3])
Tussenafstand = int(lijst1[5])
AantalKolommen = int(lijst1[7])
Lijnlast = int(lijst1[9])
Pad_Project_Map = lijst1[11]
Pad_Structuren_Kolomverlies = lijst1[13]
Pad_Pad_Structuren_Knooppuntsverplaatsing = lijst1[15]

bestand2 = open(Pad_Project_Map+'\\' + "
LijstBestandenResultaten.txt","r")
b2 = bestand2.readlines()
lijst2 = []
for line in b2:
    if line[-1] == '\n':
        lijst2.append(line[:-1])
    else:
        lijst2.append(line)

i = 0


bestand3 = open(Pad_Project_Map+'\\' + "LijstResultaten.txt","w
")

while i < len(lijst2):
    mytree = ET.parse(lijst2[i]);
    myroot = mytree.getroot();
    X = 1

    bestand3.write('Bestand: ')
    bestand3.write('\n')
    bestand3.write(str(lijst2[i]))
    bestand3.write('\n')

```

```
48      while X < len(list(myroot[12][0][0][0])):
49          varp0 = myroot[12][0][0][0][X][0].attrib
50          bestand3.write('Staafnummer: ')
51          bestand3.write('\n')
52          bestand3.write(str(varp0.get('v')))
53          bestand3.write('\n')
54          bestand3.write('Belastingscombinatie: ')
55          bestand3.write('\n')
56          varp2 = myroot[12][0][0][0][X][2].attrib
57          bestand3.write(str(varp2.get('v')))
58          bestand3.write('\n')
59          bestand3.write('Doorbuiging (m): ')
60          bestand3.write('\n')
61          varp3 = myroot[12][0][0][0][X][5].attrib
62          bestand3.write(str(varp3.get('v')))
63          bestand3.write('\n')
64          bestand3.write('Volgende staaf ')
65          bestand3.write('\n')
66          X = X + 1
67
68      i = i + 1
69      X = 1
70      mytree = ET.parse(lijst2[i]);
71      myroot = mytree.getroot();
72
73      bestand3.write('Bestand: ')
74      bestand3.write('\n')
75      bestand3.write(str(lijst2[i]))
76      bestand3.write('\n')
77
78      while X < len(list(myroot[11][0][0][0])):
79          varp0 = myroot[11][0][0][0][X][0].attrib
80          bestand3.write('Staafnummer: ')
81          bestand3.write('\n')
82          bestand3.write(str(varp0.get('v')))
83          bestand3.write('\n')
84          bestand3.write('Belastingscombinatie: ')
85          bestand3.write('\n')
86          varp2 = myroot[11][0][0][0][X][2].attrib
87          bestand3.write(str(varp2.get('v')))
88          bestand3.write('\n')
89          bestand3.write('Normaalkracht (N): ')
90          bestand3.write('\n')
91          varp3 = myroot[11][0][0][0][X][3].attrib
92          bestand3.write(str(varp3.get('v')))
93          bestand3.write('\n')
94          bestand3.write('Volgende staaf')
95          bestand3.write('\n')
96          X = X + 1
```

```
 97
 98     i = i + 1
 99
100 bestand3.close
101
102 print('ScriptInlezenResultaten klaar')
```

Bijlage G: Algoritme 1 – Script run

```python
import re
import subprocess
import pathlib



bestandslocatie = re.escape(str(pathlib.Path(__file__).parent.resolve())+'\\Script\\')
ScriptVariabelen = (bestandslocatie+"ScriptVariabelen.py")
ScriptOpstellenStructuur = bestandslocatie+"ScriptOpstellenStructuur.py"
ScriptAangepasteStructuren = bestandslocatie+"ScriptAangepasteStructuren.py"
ScriptLijstResultaten = bestandslocatie+"ScriptLijstResultaten.py"
SCIABerekening = bestandslocatie+"SCIABerekening.bat"
ScriptInlezenResultaten = bestandslocatie+"ScriptInlezenResultaten.py"



subprocess.call(ScriptVariabelen, shell=True)
subprocess.call(ScriptOpstellenStructuur, shell=True)
subprocess.call(ScriptAangepasteStructuren, shell=True)
subprocess.call(ScriptLijstResultaten, shell=True)
subprocess.call(SCIABerekening, shell=True)
subprocess.call(ScriptInlezenResultaten, shell=True)
```

Bijlage H: Algoritme 2 – Script variabelen

```
import os
import pathlib
import shutil

bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
print(bestandslocatie)
Locatie_map_Project = bestandslocatie[:-6]+"Projecten"


Naam_Model = str(input('Exacte naam model (Let op dat er geen spaties staan in de naam van het model): '));
Naam_XML_Model = str(input('Exacte naam XML model (Let op dat er geen spaties staan in de naam van het model): '));
HardeSchijf_Model = str(input('Harde schijf waarop model is opgeslagen (bv C of D): '));
HardeSchijf_SCIA = str(input('Harde schijf waarop SCIA is geinstalleerd (bv C of D): '));

bestandslocatie1 = bestandslocatie.replace('\\','')
Locatie_map_Project1 = Locatie_map_Project.replace('\\','')

LenNaamMap = len(Locatie_map_Project1)

LijstProjecten = []
rootdir = Locatie_map_Project
for it in os.scandir(rootdir):
    if it.is_dir():
        B = it.path.replace('\\', '')
        LijstProjecten.append(B[LenNaamMap:])
print(LijstProjecten)

Naam_Model2 = Naam_Model + str(".esa")
Naam_XML_Model2 = Naam_Model + str(".xml")

A = 0
i = 1
while A == 0:
    Naam_Project = Naam_Model + str(i)
    if Naam_Project not in LijstProjecten:
        A = 1
    i = i+1


path = HardeSchijf_Model.upper() + ":\\"
for root, dir, files in os.walk(path):
    if Naam_Model2 in files:
        Pad_Model=(os.path.join(root, Naam_Model2))


```

```
path = HardeSchijf_Model.upper() + ":\\"
for root, dir, files in os.walk(path):
    if Naam_XML_Model2 in files:
        Pad_XML_Model=(os.path.join(root, Naam_XML_Model2))



NaamSCIA = "ESA_XML.exe"
path = HardeSchijf_SCIA.upper() + ":\\"
for root, dir, files in os.walk(path):
    if NaamSCIA in files:
        Pad_ESA_XML_esa=(os.path.join(root, NaamSCIA))


Pad_Project_Map = str(Locatie_map_Project+ '\\' +Naam_Project)
Pad_Structuren_Kolomverlies = str(Pad_Project_Map + '\\' + '
Structuren' + '\\' + 'Kolomverlies')
Pad_Structuren_Knooppuntsverplaatsing = str(Pad_Project_Map + '
\\' + 'Structuren' + '\\' + 'Knooppuntsverplaatsing')

print(Pad_Project_Map)


os.makedirs(Pad_Structuren_Kolomverlies)
os.makedirs(Pad_Structuren_Knooppuntsverplaatsing)

LijstVariabelen = bestandslocatie+ '\\' + "LijstVariabelen.txt"

bestand = open(LijstVariabelen,"w")
bestand.write("Naam Project:")
bestand.write("\n")
bestand.write(Naam_Model)
bestand.write("\n")

bestand.write("Pad Project Map:")
bestand.write("\n")
bestand.write(Pad_Project_Map)
bestand.write("\n")

bestand.write("Pad Structuren Kolomverlies:")
bestand.write("\n")
bestand.write(Pad_Structuren_Kolomverlies)
bestand.write("\n")

bestand.write("Pad Structuren Knooppuntsverplaatsing:")
bestand.write("\n")
bestand.write(Pad_Structuren_Knooppuntsverplaatsing)
bestand.write("\n")

```

```
 93 bestand.write("Pad SCIA bestand: ")
 94 bestand.write("\n")
 95 bestand.write(Pad_Model)
 96 bestand.write("\n")
 97
 98 bestand.write("Pad XML bestand: ")
 99 bestand.write("\n")
100 bestand.write(Pad_XML_Model)
101 bestand.write("\n")
102
103 bestand.write("Pad ESA_XML.esa:")
104 bestand.write("\n")
105 bestand.write(Pad_ESA_XML_esa)
106 bestand.write("\n")
107 bestand.close()
108 bestand = open(LijstVariabelen,"r")
109
110 shutil.copyfile(bestandslocatie+'\\'+"LijstVariabelen.txt",
    Pad_Project_Map+'\\'+"LijstVariabelen.txt")
111
112 shutil.copyfile(Pad_Model, Pad_Project_Map+'\\'+Naam_Model+'.
    esa')
113 shutil.copyfile(Pad_XML_Model, Pad_Project_Map+'\\'+Naam_Model
    +'.xml')
114 shutil.copyfile(Pad_XML_Model+'.def', Pad_Project_Map+'\\'+
    Naam_Model+'.xml.def')
115
116 shutil.copyfile(Pad_XML_Model, Pad_Structuren_Kolomverlies+'\\
    '+'BasefileStructuurV2.xml')
117 shutil.copyfile(Pad_XML_Model+'.def',
    Pad_Structuren_Kolomverlies+'\\'+'BasefileStructuurV2.xml.def'
    )
118 shutil.copyfile(Pad_XML_Model, Pad_Structuren_Kolomverlies+'\\
    '+Naam_Model+'.xml')
119 shutil.copyfile(Pad_XML_Model+'.def',
    Pad_Structuren_Kolomverlies+'\\'+Naam_Model+'.xml.def')
120
121 shutil.copyfile(Pad_XML_Model,
    Pad_Structuren_Knooppuntsverplaatsing+'\\'+'
    BasefileStructuurV2.xml')
122 shutil.copyfile(Pad_XML_Model+'.def',
    Pad_Structuren_Knooppuntsverplaatsing+'\\'+'
    BasefileStructuurV2.xml.def')
123 shutil.copyfile(Pad_XML_Model,
    Pad_Structuren_Knooppuntsverplaatsing+'\\'+Naam_Model+'.xml')
124 shutil.copyfile(Pad_XML_Model+'.def',
    Pad_Structuren_Knooppuntsverplaatsing+'\\'+Naam_Model+'.xml.
    def')
125
```

```
126 print("scriptvariabelen klaar")
```

Bijlage I: Algoritme 2 – Script aanpassen structuren

```python
import xml.etree.ElementTree as ET
import pathlib

bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
bestand = open(bestandslocatie+'\\'+"LijstVariabelen.txt","r")
b = bestand.readlines()
lijst = []
for line in b:
    if line[-1] == '\n':
        lijst.append(line[:-1])
    else:
        lijst.append(line)
Naam_Model = lijst[1]
Pad_Project = lijst[3]
Pad_Structuren_Kolomverlies = lijst[5]
Pad_Pad_Structuren_Knooppuntsverplaatsing = lijst[7]
Pad_XML_Model = lijst[11]

mytree = ET.parse(Pad_XML_Model)
myroot = mytree.getroot()

myroot[11].remove(myroot[11][0])
myroot[12].remove(myroot[12][0])

Aantal_Knopen = len(myroot[7][0]) -1
Aantal_Staven = len(myroot[8][0]) -1

print ("Aantal knopen: " + str(Aantal_Knopen))
print ("Aantal staven: " + str(Aantal_Staven))

bestand = open(Pad_Project+'\\' + "LijstBestanden.txt", "w")

dict_Staven = {}
dict_Staven.update({"Staven":[],"Liggers":[],"Kolommen":[]})
dict_Knopen = {}
dict_Knopen.update({"Knopen":[]})
dict_Rijen_Knopen = {}
dict_Rijen_Knopen_Geordend = {}
dict_Tussenafstand = {}
dict_Kolommen = {}
dict_Rijen_Kolommen = {}
dict_Knopen_Boven_Kolom = {}

Lijst_Alle_Knopen_X = []
Lijst_Knopen_X = []
Lijst_Alle_Knopen_Z = []
Lijst_Knopen_Z = []

i = 1
```

```
50 while i < Aantal_Knopen+1:
51     VarKnoop = myroot[7][0][i][0].attrib
52     Knoop = str(VarKnoop.get('v'))
53 
54     dict_Knopen["Knopen"].append(Knoop)
55 
56     Naam_Knoop = str(Knoop)
57 
58     VarXCoord = myroot[7][0][i][1].attrib
59     XCoord = str(VarXCoord.get('v'))
60     Lijst_Alle_Knopen_X.append(float(XCoord))
61 
62     VarYCoord = myroot[7][0][i][2].attrib
63     YCoord = str(VarYCoord.get('v'))
64 
65     VarZCoord = myroot[7][0][i][3].attrib
66     ZCoord = str(VarZCoord.get('v'))
67     Lijst_Alle_Knopen_Z.append(float(ZCoord))
68 
69     dict_Knopen.update({Naam_Knoop: [float(XCoord),float(YCoord
   ),float(ZCoord)],})
70 
71     for XCoord in Lijst_Alle_Knopen_X:
72         if XCoord not in Lijst_Knopen_X:
73             Lijst_Knopen_X.append(XCoord)
74 
75     for ZCoord in Lijst_Alle_Knopen_Z:
76         if ZCoord not in Lijst_Knopen_Z:
77             Lijst_Knopen_Z.append(ZCoord)
78 
79     i = i+1
80 
81 Lijst_Knopen_X.sort(reverse=False)
82 Lijst_Knopen_Z.sort(reverse=False)
83 
84 i = 1
85 while i < len(Lijst_Knopen_X)+1:
86     Naam = "Rij" + str(i)
87     IF = 0
88     if i == 1:
89         Tussenafstand = Lijst_Knopen_X[i] - Lijst_Knopen_X[i -
   1]
90         IF = 1
91     if i == len(Lijst_Knopen_X):
92         Tussenafstand = Lijst_Knopen_X[i-1] - Lijst_Knopen_X[i
    - 2]
93         IF = 1
94     if IF == 0:
95         Tussenafstand1 =  Lijst_Knopen_X[i] - Lijst_Knopen_X[i
```

```
 95  - 1]
 96          Tussenafstand2 = Lijst_Knopen_X[i-1] - Lijst_Knopen_X[
    i - 2]
 97          Tussenafstand = min(Tussenafstand1,Tussenafstand2)
 98      dict_Tussenafstand[Naam] = Tussenafstand
 99      i = i +1
100
101 i = 1
102 while i < len(Lijst_Knopen_X)+1:
103      Naam = "Rij" + str(i)
104      dict_Rijen_Knopen[Naam]=[]
105      dict_Rijen_Knopen_Geordend[Naam] = []
106      dict_Rijen_Kolommen[Naam]=[]
107      i = i+1
108
109 i = 1
110 while i < Aantal_Knopen+1:
111      VarKnoop = myroot[7][0][i][0].attrib
112      Knoop = str(VarKnoop.get('v'))
113      Naam_Knoop = str(Knoop)
114
115      VarXCoord = myroot[7][0][i][1].attrib
116      XCoord = str(VarXCoord.get('v'))
117
118      VarYCoord = myroot[7][0][i][2].attrib
119      YCoord = str(VarYCoord.get('v'))
120
121      VarZCoord = myroot[7][0][i][3].attrib
122      ZCoord = str(VarZCoord.get('v'))
123
124      j = 0
125      k = 1
126      while j < len(Lijst_Knopen_X):
127          if float(XCoord) == Lijst_Knopen_X[j]:
128              Naam = "Rij" + str(k)
129              dict_Rijen_Knopen[Naam].append(Naam_Knoop)
130          j = j + 1
131          k = k + 1
132      i = i+1
133
134 i = 1
135 while i < len(dict_Rijen_Knopen)+1:
136      Naam = "Rij" + str(i)
137
138      j = 1
139      while j < len(dict_Rijen_Knopen[Naam])+1:
140          dict_Rijen_Knopen_Geordend[Naam].append(0)
141
142          j = j+1
```

```
143
144     i = i + 1
145
146 i = 1
147 while i < Aantal_Staven+1:
148     varType = myroot[8][0][i][1].attrib
149     Type = str(varType.get('t'))
150     varStaafNummer = myroot[8][0][i][0].attrib
151     StaafNummer = str(varStaafNummer.get('v'))
152
153     dict_Staven["Staven"].append(StaafNummer)
154
155     if Type == "Balk (80)":
156         dict_Staven["Liggers"].append(StaafNummer)
157
158     if Type == "column (100)" or Type == "Kolom (100)":
159         dict_Kolommen[StaafNummer] = []
160         dict_Staven["Kolommen"].append(StaafNummer)
161
162         VarBeginKnoop = myroot[8][0][i][2].attrib
163         BeginKnoop = str(VarBeginKnoop.get('n'))
164         dict_Kolommen[StaafNummer].append(BeginKnoop)
165
166         VarEindKnoop = myroot[8][0][i][3].attrib
167         EindKnoop = str(VarEindKnoop.get('n'))
168         dict_Kolommen[StaafNummer].append(EindKnoop)
169
170         if dict_Knopen[BeginKnoop][2] < dict_Knopen[EindKnoop
    ][2]:
171             BovensteKnoop = EindKnoop
172
173         if dict_Knopen[BeginKnoop][2] > dict_Knopen[EindKnoop
    ][2]:
174             BovensteKnoop = BeginKnoop
175
176         dict_Kolommen[StaafNummer].append(BovensteKnoop)
177
178         dict_Knopen_Boven_Kolom[StaafNummer]=[]
179
180     i = i + 1
181
182 i = 1
183 while i < len(dict_Rijen_Knopen)+1:
184     Naam = "Rij" + str(i)
185     j = 0
186
187     while j < len(dict_Rijen_Knopen[Naam]):
188
189         Geselecteerde_Knoop = dict_Rijen_Knopen[Naam][j]
```

```
190         ZCoord_Geselecteerde_Knoop = dict_Knopen[
    Geselecteerde_Knoop][2]
191 
192         k = 0
193         while k < len(Lijst_Knopen_Z):
194             if k < len(dict_Rijen_Knopen_Geordend[Naam]):
195                 if ZCoord_Geselecteerde_Knoop ==
    Lijst_Knopen_Z[k]:
196                     dict_Rijen_Knopen_Geordend[Naam][k]=
    Geselecteerde_Knoop
197 
198             k = k + 1
199 
200         j = j + 1
201 
202     i = i + 1
203 
204 i = 0
205 while i < len(dict_Staven["Kolommen"]):
206     Geselecteerde_Kolom = dict_Staven["Kolommen"][i]
207 
208     BovensteKnoop = dict_Kolommen[Geselecteerde_Kolom][2]
209 
210     j = 0
211     k = 1
212     while j < len(Lijst_Knopen_X):
213         if dict_Knopen[BovensteKnoop][0] == Lijst_Knopen_X[j]:
214             Naam = "Rij" + str(k)
215             dict_Rijen_Kolommen[Naam].append(
    Geselecteerde_Kolom)
216         j = j + 1
217         k = k + 1
218     i = i+1
219 
220 i = 1
221 while i < len(dict_Rijen_Kolommen)+1:
222     Rij = "Rij" + str(i)
223 
224     j = 0
225     while j < len(dict_Rijen_Kolommen[Rij]):
226         Geselecteerde_Kolom = dict_Rijen_Kolommen[Rij][j]
227 
228         BovensteKnoop = dict_Kolommen[Geselecteerde_Kolom][2]
229 
230         IndexStartKnoop = dict_Rijen_Knopen_Geordend[Rij].
    index(BovensteKnoop)
231 
232         while IndexStartKnoop < len(dict_Rijen_Knopen_Geordend
    [Rij]):
```

```
233             dict_Knopen_Boven_Kolom[Geselecteerde_Kolom].
    append(dict_Rijen_Knopen_Geordend[Rij][IndexStartKnoop])
234             IndexStartKnoop = IndexStartKnoop + 1
235 
236         j = j + 1
237     i = i + 1
238 
239 print("rtest")
240 bestand = open(Pad_Project+'\\' + "LijstBestanden.txt", "w")
241 i=0
242 while i < len(dict_Staven["Kolommen"]):
243     Staaf = str(dict_Staven["Kolommen"][i])
244     StaafID = int(Staaf[1:])
245     print("test")
246     j = 0
247     while j < len(Lijst_Knopen_X):
248         Naam = 'Rij' + str(j + 1)
249         if Staaf in dict_Rijen_Kolommen[Naam]:
250             Tussenafstand = dict_Tussenafstand[Naam]
251 
252         j = j + 1
253 
254     mytree = ET.parse(Pad_XML_Model)
255     myroot = mytree.getroot()
256     print(str(myroot[8][0][StaafID][10]))
257     myroot[8][0][StaafID].remove(myroot[8][0][StaafID][10])
258     ET.SubElement(myroot[8][0][StaafID], 'p10').set('i', '2')
259     ET.SubElement(myroot[8][0][StaafID], 'p10').set('n', '
    laag2')
260 
261     m = str(Pad_Structuren_Kolomverlies + '\\' + 'Structuur'
     + Staaf + 'ZK' + '.xml')
262     print(m)
263     ET.indent(mytree, space="\t", level=0)
264     mytree.write(m, encoding="utf-8")
265 
266     bestand.write(Pad_Structuren_Kolomverlies + '\\' + '
    Structuur' + Staaf + 'ZK' + '.xml')
267     bestand.write('\n')
268 
269     k = 0
270     while k < len(dict_Knopen_Boven_Kolom[Staaf]):
271         Knoop = dict_Knopen_Boven_Kolom[Staaf][k]
272         KnoopID = int(Knoop[1:])
273 
274         XCoord = int(dict_Knopen[Knoop][0])
275 
276         ZCoord1 = dict_Knopen[Knoop][2]
277         Verplaatsing = float(0.2 * Tussenafstand)
```

```
278         ZCoord = float(ZCoord1 - Verplaatsing)
279 
280         myroot[7][0][KnoopID].clear()
281         myroot[7][0][KnoopID].set('id', str(KnoopID))
282         myroot[7][0][KnoopID].set('nm', Knoop)
283 
284         ET.SubElement(myroot[7][0][KnoopID], 'p0').set('v',
    str(Knoop))
285 
286         ET.SubElement(myroot[7][0][KnoopID], 'p1').set('v',
    str(XCoord))
287 
288         ET.SubElement(myroot[7][0][KnoopID], 'p2').set('v', '0
    ')
289 
290         ET.SubElement(myroot[7][0][KnoopID], 'p3').set('v',
    str(ZCoord))
291 
292         k = k + 1
293 
294     m = str(Pad_Pad_Structuren_Knooppuntsverplaatsing + '\\'
     + 'Structuur' + Staaf + '.xml')
295     ET.indent(mytree, space="\t", level=0)
296     mytree.write(m, encoding="utf-8")
297 
298     bestand.write(Pad_Pad_Structuren_Knooppuntsverplaatsing +
    '\\' + 'Structuur' + Staaf + '.xml')
299     bestand.write('\n')
300 
301     i = i+1
302 
303 
304 print("ScriptAanpassenStructuren klaar")
305 
```

Bijlage J: Algoritme 2 – Script lijst resultaten

```
import shutil
import pathlib

bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
bestand1 = open(bestandslocatie+'\\'+"LijstVariabelen.txt","r")

b1 = bestand1.readlines()
lijst1 = []
for line in b1:
    if line[-1] == '\n':
        lijst1.append(line[:-1])
    else:
        lijst1.append(line)
Naam_Model = lijst1[1]
Pad_Project_Map = lijst1[3]
Pad_Structuren_Kolomverlies = lijst1[5]
Pad_Pad_Structuren_Knooppuntsverplaatsing = lijst1[7]
Pad_XML_Model = lijst1[11]

bestand2 = open(Pad_Project_Map+'\\' + "LijstBestanden.txt","r"
)
b2 = bestand2.readlines()
lijst = []
i=0
for line in b2:
    if line[-1] == '\n':
        lijst.append(line[:-1])
        i = i + 1
    else:
        lijst.append(line)

bestand3 = open(Pad_Project_Map+'\\' + "
LijstBestandenResultaten.txt","w")
j = 0
while j < i:
    var1 = str(lijst[j])
    var2 = var1[:-4]
    var3 = var2 + 'Resultaten.xml'

    bestand3.write(var3)
    bestand3.write('\n')

    j = j + 1

bestand3.close

bestand4 = open(Pad_Project_Map+'\\' + "LijstBestandenRapporten
.txt","w")
j = 0
```

```
47 while j < i:
48     var1 = str(lijst[j])
49     var2 = var1[:-4]
50     var3 = var2 + 'Rapport.pdf'
51 
52     bestand4.write(var3)
53     bestand4.write('\n')
54 
55     j = j + 1
56 
57 bestand4.close
58 
59 print('ScriptLijstResultaten klaar')
```

Bijlage K: Algoritme 2 – Script SCIA berekening

```
@echo off
@SETLOCAL enabledelayedexpansion

SET bestandslocatie1=%~dp0
set bestandslocatie= %bestandslocatie1:~0,-1%

set LijstVariabelen= %bestandslocatie%\LijstVariabelen.txt
set count1=0
for /f "tokens=*" %%x in (!bestandslocatie!\LijstVariabelen.txt
) do (
    set /a count1+=1
    set var1[!count1!]=%%x
)

set count2=0
for /f "tokens=*" %%x in (!var1[4]!\LijstBestanden.txt) do (
    set /a count2+=1
    set var2[!count2!]=%%x

)

set count3=0
for /f "tokens=*" %%x in (!var1[4]!\LijstBestandenResultaten.
txt) do (
    set /a count3+=1
    set var3[!count3!]=%%x
)

set count4=0
for /f "tokens=*" %%x in (!var1[4]!\LijstBestandenRapporten.txt
) do (
    set /a count4+=1
    set var4[!count4!]=%%x
)
@Call :Initialize

@SET SCIABestand=!var1[10]!
for /L %%i in (1,1,%count2%) do (
    @SET INPUTXML=!var2[%%i]!
    @SET OUTPUTXML=!var3[%%i]!
    @SET OUTPUTPDF=!var4[%%i]!

    @Call :Calculate LIN "%SCIABestand%" "!INPUTXML!" "!
OUTPUTXML!" "!OUTPUTPDF!"
)

@ENDLOCAL

::@TIMEOUT /T 5
```

```
46 ping -n 5 127.0.0.1
47
48 Exit
49
50
51
52 :Calculate
53
54 @ECHO.--START--
55
56 @ECHO.Berekenen...
57
58 @ECHO.arg1--%1--
59 @ECHO.arg2--%2--
60 @ECHO.arg3--%3--
61 @ECHO.arg4--%4--
62 @ECHO.cmd--@%ProgExe% %1 %2 %3 /sd /tPDF /x%4 /o%5-
63
64 @%ProgExe% %1 %2 %3 /sd /tPDF /x%4 /o%5
65
66 @ECHO.--EINDE--
67
68 @ECHO.Error Level = %ERRORLEVEL%
69
70 @Exit /B
71
72
73
74 :Initialize
75
76 @SET Pad= !var1[14]!
77 @SET ProgExe="!Pad:~1!"
78
79
80 @Exit /B
```

Bijlage L: Algoritme 2 – Script inlezen resultaten

```
import xml.etree.ElementTree as ET;
import shutil
import pathlib

bestandslocatie = str(pathlib.Path(__file__).parent.resolve())
bestand1 = open(bestandslocatie+'\\'+"LijstVariabelen.txt","r")
b1 = bestand1.readlines()

lijst1 = []
for line in b1:
    if line[-1] == '\n':
        lijst1.append(line[:-1])
    else:
        lijst1.append(line)
Verdiepingshoogte = int(lijst1[1])
AantalVerdiepingen = int(lijst1[3])
Tussenafstand = int(lijst1[5])
AantalKolommen = int(lijst1[7])
Lijnlast = int(lijst1[9])
Pad_Project_Map = lijst1[11]
Pad_Structuren_Kolomverlies = lijst1[13]
Pad_Pad_Structuren_Knooppuntsverplaatsing = lijst1[15]

bestand2 = open(Pad_Project_Map+'\\' + "
LijstBestandenResultaten.txt","r")
b2 = bestand2.readlines()
lijst2 = []
for line in b2:
    if line[-1] == '\n':
        lijst2.append(line[:-1])
    else:
        lijst2.append(line)

i = 0


bestand3 = open(Pad_Project_Map+'\\' + "LijstResultaten.txt","w
")

while i < len(lijst2):
    mytree = ET.parse(lijst2[i]);
    myroot = mytree.getroot();
    X = 1

    bestand3.write('Bestand: ')
    bestand3.write('\n')
    bestand3.write(str(lijst2[i]))
    bestand3.write('\n')

```

```
48      while X < len(list(myroot[12][0][0][0])):
49          varp0 = myroot[12][0][0][0][X][0].attrib
50          bestand3.write('Staafnummer: ')
51          bestand3.write('\n')
52          bestand3.write(str(varp0.get('v')))
53          bestand3.write('\n')
54          bestand3.write('Belastingscombinatie: ')
55          bestand3.write('\n')
56          varp2 = myroot[12][0][0][0][X][2].attrib
57          bestand3.write(str(varp2.get('v')))
58          bestand3.write('\n')
59          bestand3.write('Doorbuiging (m): ')
60          bestand3.write('\n')
61          varp3 = myroot[12][0][0][0][X][5].attrib
62          bestand3.write(str(varp3.get('v')))
63          bestand3.write('\n')
64          bestand3.write('Volgende staaf ')
65          bestand3.write('\n')
66          X = X + 1
67
68      i = i + 1
69      X = 1
70      mytree = ET.parse(lijst2[i]);
71      myroot = mytree.getroot();
72
73      bestand3.write('Bestand: ')
74      bestand3.write('\n')
75      bestand3.write(str(lijst2[i]))
76      bestand3.write('\n')
77
78      while X < len(list(myroot[11][0][0][0])):
79          varp0 = myroot[11][0][0][0][X][0].attrib
80          bestand3.write('Staafnummer: ')
81          bestand3.write('\n')
82          bestand3.write(str(varp0.get('v')))
83          bestand3.write('\n')
84          bestand3.write('Belastingscombinatie: ')
85          bestand3.write('\n')
86          varp2 = myroot[11][0][0][0][X][2].attrib
87          bestand3.write(str(varp2.get('v')))
88          bestand3.write('\n')
89          bestand3.write('Normaalkracht (N): ')
90          bestand3.write('\n')
91          varp3 = myroot[11][0][0][0][X][3].attrib
92          bestand3.write(str(varp3.get('v')))
93          bestand3.write('\n')
94          bestand3.write('Volgende staaf ')
95          bestand3.write('\n')
96          X = X + 1
```

```
 97
 98     i = i + 1
 99
100 bestand3.close
101
102 print('ScriptInlezenResultaten klaar')
```

Bijlage M: Algoritme 2 – Script run

```
 3 import pathlib
 4
 5
 6
 7 bestandslocatie = re.escape(str(pathlib.Path(__file__).parent.
   resolve())+'\\Script\\')
 8 ScriptVariabelen = (bestandslocatie+"ScriptVariabelen.py")
 9 ScriptAanpassenStructuren = (bestandslocatie+"
   ScriptAanpassenStructuren.py")
10 SCIABerekening = bestandslocatie+"SCIABerekening.bat"
11 ScriptLijstResultaten = bestandslocatie+"ScriptLijstResultaten.
   py"
12 ScriptInlezenResultaten = bestandslocatie+"
   ScriptinlezenResultaten.py"
13
14
15 subprocess.call(ScriptVariabelen, shell=True)
16 subprocess.call(ScriptAanpassenStructuren, shell=True)
17 subprocess.call(ScriptLijstResultaten, shell=True)
18 subprocess.call(SCIABerekening, shell=True)
19 subprocess.call(ScriptInlezenResultaten, shell=True)
20
21
22
23
24
```

Page 1 of 1