Woord vooraf

Deze masterproef werd uitgevoerd in kader van het masterjaar industriële ingenieurswetenschappen van de gezamenlijke opleiding IIW UHasselt en KU Leuven. Dit was echter niet mogelijk zonder de hulp van bepaalde personen. Daarom willen wij ook alle personen bedanken die gezorgd hebben voor een goed verloop van deze masterproef.

Eerst willen we ing. Phil Melard bedanken voor alle tijd die hij heeft vrijgemaakt om ons bij te staan. De hulp bij het onderzoeken van de Eurocodes heeft ons goed vooruit geholpen. Ook zijn positieve ingesteldheid had een gunstige invloed op de voortgang van de masterproef. Verder willen we prof. dr. ing. Bram Vandoren bedanken voor alle hulp bij deze masterproef, niet alleen voor de uitleg bij moeilijke onderwerpen, maar ook voor de opvolging gedurende het jaar. Tot slot bedanken we ook ing. Rik Steensels. Zonder zijn uitleg over de eindige-elementenmethode was het niet mogelijk om deze masterproef tot een goed eind te brengen.

Inhoudsopgave

WOORD VOORAF ..... **I**

TABELLENLIJST ..... **V**

FIGURENLIJST ..... **VII**

ABSTRACT ..... **IX**

ABSTRACT (ENGLISH) ..... **XI**

1 INLEIDING **1**

1.1 ACHTERGROND 1

1.2 DOELSTELLINGEN 1

1.3 REKENMETHODE 2

1.4 TOEGEPASTE SOFTWARE 2

1.5 MANIER VAN WERKEN 2

2 ACHTERGRONDINFORMATIE **5**

2.1 EINDIGE-ELEMENTENMETHODE 5

2.1.1 Algemene werkwijze *5*

2.1.2 Randvoorwaarden *9*

2.1.3 Verplaatsingen berekenen *10*

2.1.4 Krachten berekenen *10*

2.2 STALEN LIGGER 10

2.2.1 Type ligger *10*

2.2.2 Eurocode 3 ..... *11*

2.3 BETONNEN LIGGER 14

2.3.1 Algemeen *14*

2.3.2 Eurocode 2 ..... *14*

3 ONTWIKKELING EN IMPLEMENTATIE APP **21**

3.1 ALGEMEEN 21

3.1.1 Werking app *21*

3.1.2 Xcode 8.3 ..... *22*

3.2 IMPLEMENTATIE EINDIGE-ELEMENTENMETHODE 25

3.2.1 Opstellen matrices *25*

3.2.2 Berekenen interne krachten *27*

3.2.3 Momenten- en dwarskrachtendiagram *29*

3.3 STALEN PROFIEL 29

3.3.1 Controles Eurocode 3 ..... *29*

3.3.1 Doorbuiging *31*

3.3.2 Optimale balk *31*

3.4 BETONNEN BALK 31

3.4.1 Controles Eurocode 2 ..... *32*

3.4.2 Optimale hoogte *33*

3.4.3 Wapening *34*

3.4.4 Doorbuiging *34*

4 TOEPASSING EN VALIDATIE APP **37**

4.1 TOEGEPAST VOORBEELD 37
4.1.1 Velden 37
4.1.2 Belastingen 37
4.1.3 Snedekrachten 38
4.1.4 Stalen profiel 39
4.1.5 Betonnen balk 39
4.2 RESULTATEN 40
4.2.1 Opstelling 1 40
4.2.2 Opstelling 2 42
4.2.3 Opstelling 3 44
4.2.4 Opstelling 4 45
5 BESLUIT 49
LITERATUURLIJST 51
BIJLAGEN 53
BIJLAGE A: BELANGRIJKE CODE 55
Bijlage A.1: Belastingen toevoegen 55
Bijlage A.2: Randvoorwaarden 56
Bijlage A.3: Verplaatsingen berekenen 57
Bijlage A.4: Berekenen momenten en dwarskrachten 58
Bijlage A.5: Kipweerstand 59
Bijlage A.6: Berekenen dwarskrachtwapening 60
Bijlage A.7: Scheurwijdtecontrole 61

Tabellenlijst

Tabel 4.1: snedekrachten opstelling 1 40

Tabel 4.2: profielen opstelling 1 41

Tabel 4.3: betonbalk opstelling 1 41

Tabel 4.4: snedekrachten opstelling 2 43

Tabel 4.5: profielen opstelling 2 43

Tabel 4.6: betonbalk opstelling 2 44

Tabel 4.7: snedekrachten opstelling 3 44

Tabel 4.8: profielen opstelling 3 45

Tabel 4.9: betonbalk opstelling 3 45

Tabel 4.10: snedekrachten opstelling 4 46

Tabel 4.11: profielen opstelling 4 46

Tabel 4.12: betonbalk opstelling 4 46

Figurenlijst

Figuur 1.1: manier van werken 2
Figuur 2.1: opdeling in n elementen en n+1 knooppunten 5
Figuur 2.2: puntlast op een element 6
Figuur 2.3: lijnlast op een element en equivalente situatie 7
Figuur 2.4: mogelijke verplaatsingen van een element 8
Figuur 2.5: globale stijfheidsmatrix 9
Figuur 3.1: flowchart werking app 21
Figuur 3.2: templates Xcode project [14] 23
Figuur 3.3: main.storyboard [14] 23
Figuur 3.4: voorbeeld in- en uitvoer 24
Figuur 3.5: configuratie voorbeeld 24
Code 3.1: structuur matrix 25
Code 3.2: opstellen matrices 26
Code 3.3: krachten toevoegen 26
Code 3.4: stijfheidsmatrix van één element 27
Code 3.5: globale stijfheidsmatrix 27
Code 3.6: inverse matrix 28
Code 3.7: vermenigvuldiging matrix 28
Code 3.8: momenten uitrekenen 28
Figuur 3.6: proces berekening M,V 29
Code 3.9: dwarskrachtcontrole 30
Code 3.10: controle buigend moment 30
Code 3.11: controle combinatie dwarskracht en buiging 31
Code 3.12: ontwerp betonbalk 32
Code 3.13: minimumwapening 32
Code 3.14: spaningscontrole 33
Code 3.15: optimale betonbalk 34
Code 3.16: wapening omzetten in staven 34

Code 3.17: doorbuiging betonbalk 35
Figuur 4.1: ingevoerde opstelling 37
Figuur 4.2: overspanningen ingeven 37
Figuur 4.3: belastingen ingeven 38
Figuur 4.4: maximale en minimale snedekrachten 38
Figuur 4.5: momenten- en dwarskrachtendiagram 39
Figuur 4.6: optimale stalen profielen 39
Figuur 4.7: optimale betonnen balk 40
Figuur 4.8: opstelling 1 40
Figuur 4.9: opstelling 2 43
Figuur 4.10: opstelling 3 44
Figuur 4.11: opstelling 4 45

Abstract

Studiebureau AB Associates krijgt op werven vaak vragen over aanpassingen van een structuur (plaatsing van kolommen e.d.). Daarom is er nood aan een handige tool om balken op de werf te controleren. Dit zou een grote tijdsbesparing voor het bedrijf opleveren. Het bedrijf kan zo een flexibelere houding aannemen tegenover de architect. De tijdswinst zorgt uiteindelijk ook voor een kostenbesparing.

De masterproef creëert deze tool in de vorm van een app om hyperstatische balken te berekenen. De app bepaalt het optimale stalen profiel en bij betonbalken optimaliseert de app de hoofdwapening, de dwarskrachtwapening en de hoogte. Uiteindelijk controleert de app ook de doorbuiging van de liggers.

Het ontwikkelen van de app gebeurt in Xcode, een ontwerpplatform voor Apple. Eerst wordt de eindige-elementenmethode geïmplementeerd in de app om de snedekrachten in de liggers te bepalen. Vervolgens hanteert de app de Eurocodes om de ligger te dimensioneren en de doorbuiging te bepalen. Uiteindelijk worden de resultaten uit de app vergeleken met resultaten uit bestaande software.

De snedekrachten en de dimensionering van de balken uit de app komt goed overeen met de resultaten uit bestaande software. Enkele controles in de app werden veilig benaderd. Dit leidt in sommige gevallen tot conservatieve berekeningen. De app kan dus verbeterd worden door deze controles verder uit te werken.

Abstract (English)

Engineers from the company AB Associates often receive questions about adjustments of the structure (positioning of columns etc.) when they are at the construction site. Therefore they need a tool to calculate beams at the construction site. The tool leads to big time savings for the company. In this way the company can have a flexible attitude against architects. Finally the time savings result in a reduction of the costs.

The thesis creates this tool as an app which calculates hyperstatic beams. The application determines for each situation the optimal steel profile. In concrete beams the reinforcement and the height are optimized. Finally the app checks if the deflection of the beam does not exceed the limit value.

The app is developed in Xcode, a design platform from Apple. First, the finite element method is implemented in the app to determine the internal forces in the beams. Then the Eurocodes are used to dimension the beams and to determine the deflection. Finally the results are verified with the help of existing software.

The internal forces, calculated by the app, correspond to the forces calculated by the software. Certain formulas in the app are conservative, which results in safe calculations. The app can be improved by more accurate estimation of these formulas.

1 Inleiding

1.1 Achtergrond

Stabiliteitsingenieurs komen vaak in aanraking met de dimensionering en de controle van hyperstatische balken. Vandaag de dag doet men dit door middel van lange tijdrovende berekeningen of door middel van software. Deze software is enkel beschikbaar voor computers en dus niet voor smartphones. Momenteel bestaat er dus geen snelle en eenvoudige methode om hyperstatische balken buiten op de werf te berekenen.

Wanneer stabiliteitsingenieurs op werven komen krijgen ze vaak vragen van aannemers over deze balken. Om op deze vraag een concreet en betrouwbaar antwoord te kunnen geven, is het aangewezen een tool te ontwikkelen die hyperstatische balken betrouwbaar berekent. Een bijkomend voordeel zou zijn dat de tool een optimalisatie uitvoert, waarmee aangegeven kan worden welk type profiel het best voldoet aan de eisen van de klant. Aan de hand van deze tool kan de ingenieur aantonen welk type balk voldoet, zodat verdere discussies op de werf vermeden kunnen worden. De tijdswinst kan ervoor zorgen dat er kosten worden bespaard binnen het project.

Hyperstatische balken zijn balken die statisch onbepaald zijn. Dit betekent dat men aan de hand van evenwichtsvergelijkingen de reactiekrachten niet kan bepalen. Om deze reacties te kunnen bepalen zijn er extra vergelijkingen nodig. Deze worden verkregen door middel van compatibiliteitsvoorwaarden. Dit zijn voorwaarden voor bijvoorbeeld verplaatsingen of rekken in de structuur [1]. Er zijn geen éénduidige formules beschikbaar voor het berekenen van hyperstatische balken. Met de hulp van een alternatieve rekenmethode zal de app op een snelle en efficiënte manier deze balken berekenen.

1.2 Doelstellingen

De doelstelling van deze masterproef is het ontwikkelen van een mobiele applicatie om hyperstatische balken uit te rekenen. De app moet de exacte snedekrachten kunnen berekenen en deze vervolgens visualiseren aan de hand van snedekrachtendiagrammen. In overleg met de promotoren is beslist om de reikwijdte van de app te beperken tot vier overspanningen van maximaal vijftien meter. Vervolgens moeten alle steunpunten scharnierend uitgevoerd worden, aangezien dit de meest voorkomende situatie is bij hyperstatische balken.

Wanneer de opstelling van de balk is ingesteld, moeten de belastingen worden kunnen toegevoegd. Er zullen twee types van belastingen mogelijk zijn, namelijk punt- en lijnlasten. Van elk type kunnen vier afzonderlijke belastingen worden ingevoerd met een exacte positie op de balk. Volgens de Eurocode moet er een onderscheid worden gemaakt tussen permanente en variabele lasten. Dit onderscheid wordt ook gehanteerd binnen de app. De gebruiker kan aangeven of er een permanente of variabele last ingevoerd moet worden.

Met deze gegevens is het mogelijk de snedekrachten te bepalen. Aan de hand van de berekende waarde voor de maximale dwarskracht en het maximaal buigmoment is het mogelijk om via Eurocode 2 (EC2) de optimale betonnen ligger te berekenen. De gebruiker van de app heeft de mogelijkheid om verschillende karakteristieken van de betonnen balk vast te leggen, namelijk de breedte van de balk, de betondekking en de betonkwaliteit. De app bepaalt dan de optimale hoogte van de betonnen balk. De nodige wapening wordt vervolgens bepaald via Eurocode 2.

Uiteindelijk zal ook de wapening concreet worden omgezet in een aantal staven met een bepaalde diameter. Zo heeft de gebruiker snel een idee van de benodigde veld- en steunpuntswapening.

Op dezelfde wijze wordt voor een stalen profiel via Eurocode 3 de optimale balk berekend. Voor de stalen profielen heeft de gebruiker niet de mogelijkheid om de karakteristieken vast te leggen, aangezien deze bepaald worden door de fabrikant. De app beperkt zich tot de vier meest courante types van profielen, namelijk IPE, HEA, HEB en HEM. Bij elke opstelling bepaalt de app voor elk profieltype het meest optimale profiel.

Tot slot zal voor elke berekende balk de maximale doorbuiging worden gecontroleerd. De uiteindelijke output van de app is voor staal het type profiel met zijn maximale doorbuiging. Voor de betonnen balk wordt deze output aangevuld met de minimale hoogte en de noodzakelijke wapening, omgezet in aantal en diameter.

1.3 Rekenmethode

De app moet de snedekrachten en doorbuigingen kunnen bepalen. Deze berekeningen zullen worden uitgevoerd met de eindige-elementenmethode (EEM). Er is gekozen voor de EEM omdat deze werkt met matrixvergelijkingen [2]. De matrixvergelijkingen kunnen altijd geïmplementeerd worden in de software voor de ontwikkeling van de app. Bijkomend is EEM ideaal voor het visualiseren van de snedekrachtendiagrammen.

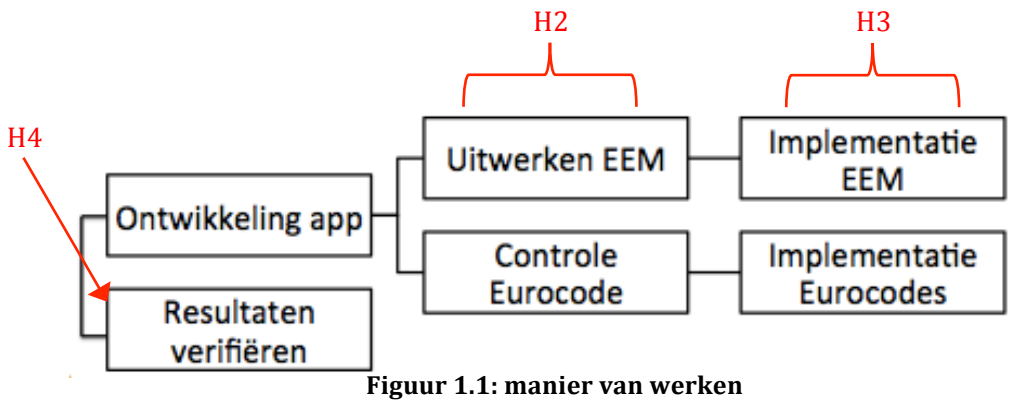
1.4 Toegepaste software

De app dient te worden ontwikkeld voor IPhone en eventueel IPad. Deze twee producten zijn allebei producten van fabrikant Apple. Om applicaties voor Apple-producten te kunnen ontwikkelen heeft de fabrikant een programma ontworpen, genaamd Xcode. De meest recente versie van dit programma is Xcode 8.3. Dit programma werkt met Swift 3.0 als programmeertaal [3].

De resultaten uit de ontwikkelde app zullen gecontroleerd worden met bestaande software. Er is geopteerd om de resultaten te vergelijken met het programma Diamonds van Buildsoft [4]. Deze keuze is te wijten aan het feit dat Diamonds dezelfde controles uit Eurocode 2 en Eurocode 3 toepast als de app.

1.5 Manier van werken

Het ontwerp van de app is gerealiseerd door het volgen van een vooropgesteld schema. De manier van werken is schematisch voorgesteld in onderstaande figuur.



Figuur 1.1: manier van werken

Eerst werd de eindige-elementenmethode uitgewerkt voor doorlopende liggers. Vervolgens werden deze matrices in het ontwerpplatform geïmplementeerd. Daarna werd de balk gedimensioneerd volgens Eurocode 2 of Eurocode 3. De uitwerking van de eindige-elementenmethode en de dimensionering van de balk zijn uitgelegd in hoofdstuk 2 van de masterproef. Vervolgens zijn voorgaande aspecten omgezet in codetaal en geïmplementeerd in de app. De ontwikkeling en implementatie van de app zijn weergegeven in hoofdstuk 3. Uiteindelijk werden de resultaten uit de app vergeleken met resultaten van bestaande software. Samen met een concreet voorbeeld vormt de validatie van de resultaten het vierde hoofdstuk van de masterthesis.

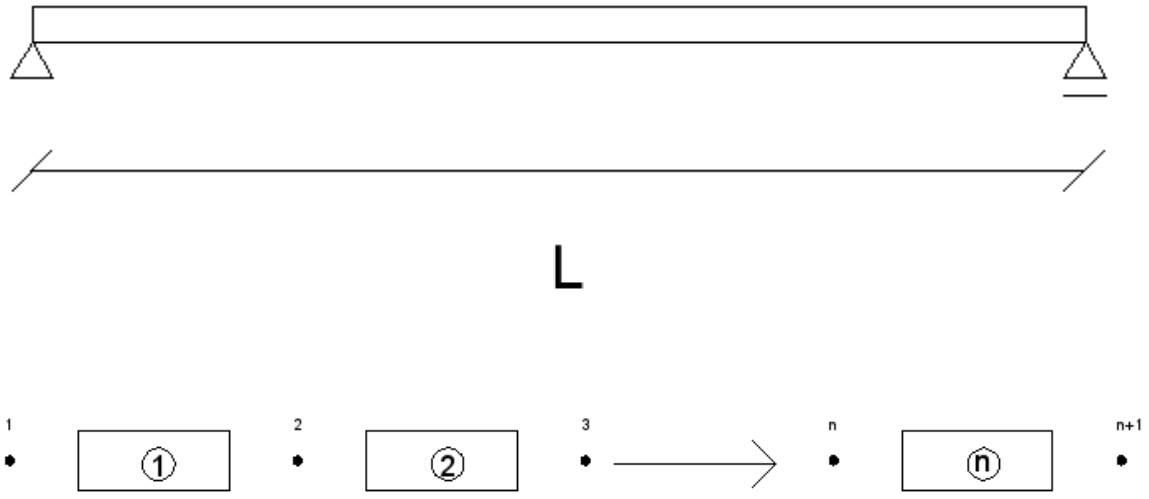
2 Achtergrondinformatie

2.1 Eindige-elementenmethode

Voor de berekening van de hyperstatische balken is er geopteerd om te werken met de eindige-elementenmethode. De algemene werking van deze methode is in volgende paragraaf beknopt beschreven. Verder zijn de verschillende aspecten met betrekking tot het correct berekenen van de snedekrachten op een horizontale ligger beschreven. Voor de implementatie van deze methode wordt verwezen naar het derde hoofdstuk.

2.1.1 Algemene werkwijze

Zoals de naam van de methode doet vermoeden, wordt een constructie opgedeeld in een eindig aantal elementen. De elementen komen samen in een knooppunt. Onderstaande figuur toont een mogelijkheid voor het opdelen en benoemen van een eenvoudige constructie, namelijk een opgelegde balk met lengte L. De balk kan opgedeeld worden in n elementen en n+1 knooppunten. De lengte van elk element is dan gelijk aan de totale lengte gedeeld door het aantal elementen. Als het gaat over een homogene balk, kan de balk worden opgedeeld in n identieke elementen.



Figuur 2.1: opdeling in n elementen en n+1 knooppunten

Elk knooppunt heeft randvoorwaarden afhankelijk van de eigenschappen van een constructie(deel). Het gedrag van de constructie kan worden benaderd door middel van een matrixvergelijking. Voor een lineair elastische berekening geldt volgende matrixvergelijking [2]:

$$[F] = [K] * [u] \tag{2.1}$$

met:

[F] = Belastingen op de knooppunten in alle richtingen (Krachtenmatrix)

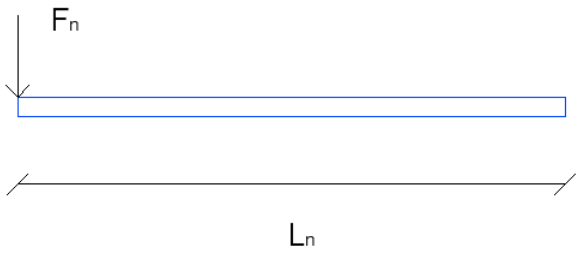
[K] = Stijfheidsmatrix van de constructie

[u] = verplaatsingen van de knooppunten in alle richtingen (Verplaatsingsmatrix)

Krachtenmatrix

De krachtenmatrix is een kolommatrix waarin alle lasten op elk knooppunt worden opgeslagen. Om de methode toe te kunnen passen op de app moeten op ieder knooppunt drie krachten kunnen aangrijpen: F_x, F_y en M. Aangezien er geen horizontale belastingen kunnen worden toegevoegd, geldt voor alle knooppunten $F_x = 0$.

De puntlasten kunnen eenvoudig worden toegevoegd aan de krachtenmatrix. Onderstaande figuur toont een element met lengte L_n, waar een puntlast is toegevoegd op knooppunt n.

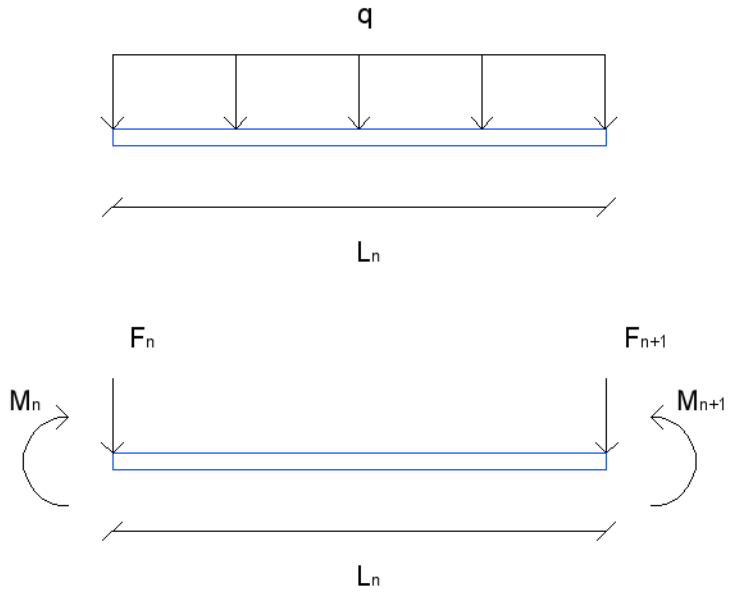


Figuur 2.2: puntlast op een element

De krachtenmatrix voor een element belast met een puntlast wordt dan als volgt:

$$[F_n] = \begin{bmatrix} F_{x,n} \\ F_{y,n} \\ M_n \\ F_{x,n+1} \\ F_{y,n+1} \\ M_{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ -F_n \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.2}$$

EEM heeft als nadeel dat enkel op de knooppunten krachten en verplaatsingen aangebracht kunnen worden. Om een lijnlast toe te kunnen voegen, moet deze last worden onderverdeeld in een equivalente situatie van momenten en puntlasten op de knooppunten. Voor alle elementen binnen het bereik van de lijnlast moet er een omzetting gebeuren zoals aangetoond op onderstaande figuur.



Figuur 2.3: lijnlast op een element en equivalente situatie

De waarde van de equivalente puntlasten en momenten zijn $F = q * \frac{L_n}{2}$ en $M = q * \frac{L_n^2}{8}$. De krachtenmatrix voor een element belast met een lijnlast wordt dan als volgt:

$$[F_n] = \begin{bmatrix} F_{x,n} \\ F_{y,n} \\ M_n \\ F_{x,n+1} \\ F_{y,n+1} \\ M_{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ -q * \frac{L_n}{2} \\ -q * \frac{L_n^2}{8} \\ 0 \\ -q * \frac{L_n}{2} \\ q * \frac{L_n^2}{8} \end{bmatrix} \tag{2.3}$$

Tot slot worden de krachten in elk knooppunt samengesteld tot één kolommatrix voor alle knooppunten. Deze matrix bevat $3 * (n + 1)$ elementen:

$$[F] = \begin{bmatrix} F_{x,1} \\ F_{y,1} \\ M_1 \\ \cdots \\ \cdots \\ \cdots \\ F_{x,n+1} \\ F_{y,n+1} \\ M_{n+1} \end{bmatrix} \tag{2.4}$$

Verplaatsingsmatrix

Aangezien de balk wordt opgelost als een 2D-balkelement, heeft de balk drie vrijheidsgraden [2]. De balk kan een horizontale verplaatsing, een verticale verplaatsing en een hoekverdraaiing ($u_x, u_y \; en \; \varphi_y$) ondergaan in elk knooppunt. De steunpunten worden scharnierend uitgevoerd, waardoor de horizontale verplaatsing overal gelijk wordt gesteld aan nul. Voor het opstellen van de matrixvergelijkingen wordt hier nog geen rekening mee gehouden. De verplaatsingen worden

achteraf aangepast met bijhorende randvoorwaarden. Onderstaande figuur is een illustratie van de zes mogelijke verplaatsingen die elk element kan ondergaan.

Figuur 2.4: mogelijke verplaatsingen van een element

De dimensie van de verplaatsingsmatrix is analoog aan de krachtenmatrix: een kolommatrix met $3 * (n + 1)$ elementen:

$$[u] = \begin{bmatrix} u_{x,1} \\ u_{y,1} \\ \varphi_{y,1} \\ \cdots \\ \cdots \\ \cdots \\ u_{x,n+1} \\ u_{y,n+1} \\ \varphi_{y,n+1} \end{bmatrix} \tag{2.5}$$

Stijfheidsmatrix

Alvorens de globale stijfheidsmatrix kan worden opgesteld, is het noodzakelijk om voor elk element afzonderlijk een stijfheidsmatrix op te stellen. In het geval van een homogene balk, verdeeld in elementen met constante breedte, zal voor ieder element de stijfheidsmatrix analoog zijn. Figuur 2.4 toont aan dat een element bestaat uit twee knooppunten met drie vrijheidsgraden.

In het kader van de masterproef is het niet relevant de uitgebreide afleiding van de stijfheidsmatrix te vermelden. Hoe de matrix tot stand is gekomen, wordt besproken aan de hand van de algemene theorie achter de afleiding. Om de afleiding mogelijk te maken zijn er twee belangrijke aannames gemaakt [5].

1. Een vlakke doorsnede voor vervorming blijft vlak na het optreden van de vervorming;
2. Een vlakke doorsnede loodrecht op de as van het element voor vervorming blijft loodrecht op de as in de vervormde toestand (dwarskrachtvervorming wordt verwaarloosd).

In het geval van de horizontale verplaatsing wordt een staafelement beschouwd waarbij een axiale verplaatsing mogelijk is. Voor het vinden van een juiste uitdrukking van de verplaatsing in functie van de normaalkracht is de differentiaalvergelijking voor de axiale verplaatsing van een staaf gehanteerd [6]. Het verband knoopverplaatsing-eindstaafkrachten kan vervolgens als volgt uitgedrukt worden:

$$F_{x,n} = EA\frac{du_x}{dx} = EA * \frac{u_{x,n} - u_{x,n+1}}{L_{element}} \tag{2.6}$$

De verticale verplaatsing en de hoekverdraaiing worden bepaald door de differentiaalvergelijking voor de doorbuiging van een 2D balkelement af te leiden [5]. het verband tussen het moment of de dwarskracht en de verticale verplaatsing wordt vervolgens weergegeven door onderstaande uitdrukkingen.

$$T_{y1} = -T_y(0) = \left(\frac{12EI_y}{L^3}\right) * u_{y,1} \tag{2.7}$$

$$T_{y2} = -T_y(L) = \left(-\frac{12EI_y}{L^3}\right) * u_{y,1} \tag{2.8}$$

$$M_{y,1} = M_y(0) = \left(\frac{6EI_y}{L^2}\right) * u_{y,1} \tag{2.9}$$

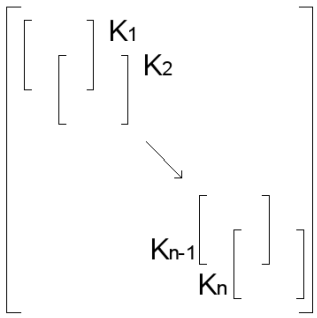
$$M_{y,1} = M_y(L) = \left(\frac{6EI_y}{L^2}\right) * u_{y,1} \tag{2.10}$$

Bovenstaande formules kunnen als matrixvergelijking worden genoteerd. De stijfheidsmatrix van één element ziet er dan uit als volgt:

$$[K_{element}] = \begin{bmatrix} k_1 & 0 & 0 & -k_1 & 0 & 0 \\ 0 & 12k_2 & 6k_3 & 0 & -12k_2 & 6k_3 \\ 0 & 6k_3 & 4k_4 & 0 & -6k_3 & 2k_4 \\ -k_1 & 0 & 0 & k_1 & 0 & 0 \\ 0 & -12k_2 & -6k_3 & 0 & 12k_2 & -6k_3 \\ 0 & 6k_3 & 2k_4 & 0 & -6k_3 & 4k_4 \end{bmatrix} \tag{2.11}$$

met: $k_1 = \frac{EA}{L}$, $k_2 = \frac{EI}{L^3}, k_3 = \frac{EI}{L^2}$ *en* $k_4 = \frac{EI}{L}$

In de app is de homogene balk onderverdeeld in elementen met constante lengte. Hierdoor is de stijfheidsmatrix van elk afzonderlijk element identiek. Alvorens de EEM kan worden toegepast, moeten de afzonderlijke stijfheidsmatrices van alle elementen worden geassembleerd tot een globale stijfheidsmatrix [7]. Onderstaande figuur toont schematisch aan hoe de globale stijfheidsmatrix wordt opgesteld.



Figuur 2.5: globale stijfheidsmatrix

Zoals figuur 2.5 aangeeft, worden op de diagonaal van de globale stijfheidsmatrix de elementen toegevoegd. De overige waarden binnen de matrix zijn gelijk aan nul. De stijfheidsmatrix is een vierkante matrix met $(n * 3 + 3)$ rijen en kolommen.

2.1.2 Randvoorwaarden

Wanneer de globale matrixvergelijking is opgesteld, is het noodzakelijk om de randvoorwaarden te verwerken binnen deze vergelijking. Het verhinderen van horizontale en verticale verplaatsing moet worden verwerkt binnen de methode. Ter hoogte van de scharnierende steunpunten worden de verticale en horizontale verplaatsing verhinderd.

Ter hoogte van een knooppunt kan een verplaatsing worden verhinderd door in de verplaatsingsmatrix de verplaatsing op de desbetreffende plaats gelijk te stellen aan nul. Verder

moet ook de globale stijfheidsmatrix worden aangepast. Stel dat een verplaatsing op knooppunt x binnen de verplaatsingsmatrix moet worden vermeden, dan zal op positie (x,x) binnen de globale stijfheidsmatrix de waarde op één worden gezet. De volledige rij x en kolom x krijgen de waarde nul, met uitzondering van deze plaats (x,x) [8]. Ter verduidelijking is onderstaand vereenvoudigd voorbeeld gegeven.

$$[K] * [u] = \begin{bmatrix} & & 0 & & & \\ & & 0 & & & \\ 0 & 0 & 1 & 0 & 0 & 0 \\ & & 0 & & & \\ & & 0 & & & \\ & & 0 & & & \end{bmatrix} \begin{bmatrix} \\ \\ 0 \\ \\ \\ \\ \end{bmatrix} \tag{2.12}$$

2.1.3 Verplaatsingen berekenen

Om de verplaatsingen van de balk in de knooppunten te berekenen volstaat het om de globale matrixvergelijking op te lossen in de vorm: $[u] = [K]^{-1} * [F]$. De horizontale en verticale verplaatsingen worden ter hoogte van de steunpunten op nul gehouden. De verplaatsingen $u_{x,n}, u_{y,n}$ *en* $\varphi_{y,n}$ voor knooppunt n zullen zich dan als volgt bevinden in de verplaatsingsmatrix:

$$[u] = \begin{bmatrix} u_{x,n} \\ u_{y,n} \\ \varphi_{y,n} \\ \\ \end{bmatrix} \tag{2.13}$$

2.1.4 Krachten berekenen

De krachten worden afzonderlijk berekend voor elk element. De verplaatsingen die zich voordoen voor een specifiek element zijn gekend. Door deze verplaatsingen te vermenigvuldigen met de stijfheidsmatrix van dat element, kunnen de krachten op de bijhorende knooppunten bepaald worden [8]. Onderstaand voorbeeld geeft aan hoe elke kracht afzonderlijk wordt berekend. De rood gekleurde waarden geven aan hoe kracht $F_{y,n}$ wordt bepaald.

$$\begin{bmatrix} F_{x,n} \\ F_{y,n} \\ M_n \\ F_{x,n+1} \\ F_{y,n+1} \\ M_{n+1} \end{bmatrix} = \begin{bmatrix} k_1 & 0 & 0 & -k_1 & 0 & 0 \\ 0 & 12k_2 & 6k_3 & 0 & -12k_2 & 6k_3 \\ 0 & 6k_3 & 4k_4 & 0 & -6k_3 & 2k_4 \\ -k_1 & 0 & 0 & k_1 & 0 & 0 \\ 0 & -12k_2 & -6k_3 & 0 & 12k_2 & -6k_3 \\ 0 & 6k_3 & 2k_4 & 0 & -6k_3 & 4k_4 \end{bmatrix} * \begin{bmatrix} u_{x,n} \\ u_{y,n} \\ \varphi_{y,n} \\ u_{x,n+1} \\ u_{y,n+1} \\ \varphi_{y,n+1} \end{bmatrix} \tag{2.14}$$

2.2 Stalen ligger

2.2.1 Type ligger

De app bepaalt voor vier types van liggers het optimale profiel. Deze vier types zijn: HEA-, HEB-, HEM- en IPE-profielen. De HE-profielen hebben bredere flenzen dan de IPE-profielen. Bij de HE-profielen hebben HEA-profielen de dunste flens en het dunste lijf, terwijl HEM-profielen de dikste flens en het dikste lijf hebben. Hierdoor hebben HEM-profielen een hoger draagvermogen, maar zijn ze ook zwaarder en dus duurder dan gelijkwaardige HE-profielen. De vier types profielen zijn allemaal te verkrijgen in meerdere staalkwaliteiten. De app bepaalt het optimale

profiel enkel voor een staalkwaliteit S235, dit zijn profielen met een vloeigrens $f_y = 235\ N/mm^2$.

2.2.2 Eurocode 3

De bepaling van het optimale profiel gebeurt aan de hand van verschillende controles uit Eurocode 3. Eerst bepaalt de app de maximale optredende dwarskracht en het maximaal optredend moment. Vervolgens wordt voor het kleinste profiel van elk type de dwarskrachtweerstand, de weerstand tegen buiging, de kipweerstand en de doorbuiging bepaald. Indien de optredende krachten kleiner zijn dan de weerstand van het profiel en de doorbuiging kleiner is dan de maximaal toelaatbare doorbuiging, dan heeft de app het optimale profiel gevonden. Indien een of meerdere optredende krachten groter zijn dan de weerstand van het profiel of indien de doorbuiging te groot is, dan volstaat dit profiel niet. Dan controleert de app een groter profiel. Deze handeling wordt herhaald tot er een profiel gevonden is dat voldoet aan alle controles.

De formules die gehanteerd worden om de sterkte en de stabiliteit van de profielen te controleren staan vermeld in Eurocode 3. De gegevens die nodig zijn om de controles uit te voeren, zijn terug te vinden in de staalcatalogus of worden berekend door de app. Bij al deze berekeningen is de elasticiteitsmodulus van de profielen gelijk aan 210 GPa en de Poisson-factor gelijk aan 0,3 [9].

Klasse van de doorsnede

De klasse van de doorsnede moet gekend zijn vooraleer de weerstand tegen buiging en de kipweerstand bepaald kunnen worden. De klasse is afhankelijk van de afmetingen van het profiel en de spanningsverdeling in het profiel. De doorsnede van het profiel kan geclassificeerd worden in vier klassen.

Als het profiel beschikt over een klasse één of een klasse twee doorsnede, dan kan het plastisch moment bereikt worden in deze doorsnede. Profielen met een doorsnede van klasse drie kunnen het plastisch moment niet bereiken door lokaal plooien. Bij deze profielen kan echter wel het elastisch moment in de doorsnede bereikt worden. Bij profielen van klasse vier daarentegen kan de doorsnede de vloeigrens niet bereiken omdat er dan lokaal plooien zal optreden.

Dwarskrachtweerstand

Eerst bepaalt de app de dwarskrachtweerstand van een profiel. De dwarskrachtweerstand wordt berekend aan de hand van de plastische berekeningsmethode. Dit wil zeggen dat de weerstand bij vloeien van het profiel wordt uitgerekend met volgende formule:

$$V_{pl,Rd} = \frac{A_v * (f_y/\sqrt{3})}{\gamma_{M0}} \tag{2.15}$$

In deze formule is A_v de oppervlakte van het werkzame afschuifoppervlak. Het afschuifoppervlak voor gewalste I- en H-profielen waarbij de belasting optreedt evenwijdig aan het lijf van het profiel kan op volgende manier bepaald worden:

$$A_v = A - 2 * b * t_f + (t_w + 2 * r) * t_f > h_w * t_w \tag{2.16}$$

De veiligheidsfactor γ_{M0} is gelijk aan 1,0. De dwarskrachtweerstand moet in elke doorsnede groter zijn dan de optredende dwarskracht in de corresponderende doorsnede.

Weerstand tegen buiging

Vervolgens controleert de app het profiel op buiging. De weerstand tegen buiging om de sterke as is afhankelijk van de klasse van de doorsnede. Als deze gekend is, wordt de weerstand tegen buiging met onderstaande formule bepaald:

$$M_{c,Rd} = \frac{W_y * f_y}{\gamma_{M0}} \tag{2.17}$$

De term W_y is het weerstandbiedend moment van het profiel. Bij doorsnedes van klasse één of klasse twee is deze gelijk aan het plastische weerstandbiedend moment. Dit in tegenstelling tot klasse drie doorsnedes waar deze gelijk is aan het elastisch weerstandbiedend moment. Bij doorsnedes van klasse vier wordt eerst het effectief weerstandbiedend moment van de doorsnede berekend vooraleer de app de weerstand tegen buiging kan bepalen. Deze weerstand tegen buiging moet in elke doorsnede van het profiel groter zijn dan het optredend buigmoment in deze doorsnede.

Samengestelde dwarskracht en buiging

De mogelijke belastinggevallen die in de app ingevoerd kunnen worden zorgen allemaal voor een combinatie van dwarskracht en buiging. Daarom wordt de invloed van dwarskracht op de weerstand tegen buiging in rekening gebracht. Indien de optredende dwarskracht kleiner is dan de helft van de dwarskrachtweerstand, bepaald in formule 3.1, dan heeft deze geen invloed op de weerstand tegen buiging. Indien dit niet het geval is wordt de weerstand tegen buiging gereduceerd. Dit wordt gedaan aan de hand van onderstaande formule. Deze formule is enkel geldig voor I-doorsneden met gelijke flenzen en buiging om de sterke as. Aangezien dit altijd het geval is in de app, is deze formule altijd van toepassing.

$$M_{v,y,Rd} = \frac{(W_{pl,y} - \frac{\rho * A_w^2}{4 * t_w}) * f_y}{\gamma_{M0}} \tag{2.18}$$

Hierbij is A_w de oppervlakte van het lijf. Deze wordt bepaald door de hoogte van het lijf h_w te vermenigvuldigen met de breedte van het lijf t_w. Het gereduceerd moment mag niet groter zijn dan de weerstand tegen buiging, bepaald met formule 3.3. Indien dit wel het geval is wordt er gewerkt met de weerstand tegen buiging uit deze formule. In elke doorsnede moet het optredend buigmoment kleiner zijn dan het gereduceerd moment.

Kipweerstand

De app heeft het profiel reeds gecontroleerd op sterkte. Nu volgt er nog een controle op de stabiliteit van het profiel. Aangezien er geen normaalkracht optreedt, is het profiel niet gevoelig voor knik. Daarom zal de app het profiel enkel controleren op kip.

Ter hoogte van de steunpunten is de balk zijdelings gesteund, er kan hier dus geen zijdelingse verplaatsing optreden. Daarom controleert de app elke overspanning van de ligger afzonderlijk op kip. De kiplengte is bijgevolg de lengte van de overspanning. Eerst berekent de app het elastisch kritisch moment voor een standaardsituatie. Dit is een dubbel symmetrische doorsnede met een constante momentverdeling. Vervolgens wordt er een correctie op dit elastisch kritisch moment toegepast die rekening houdt met het aangrijpingspunt van de belasting op het profiel en met de momentverdeling. Het elastisch kritisch moment voor de standaardsituatie wordt berekend met volgende formule [10]:

$$M_{cr}^{E} = \frac{\pi}{L} * \sqrt{G * I_T * E * I_z * \left(1 + \frac{\pi^2 * E * I_W}{L^2 * G * I_T}\right)} \quad (2.19)$$

Dit elastisch kritisch moment moet nog gecorrigeerd worden om rekening te houden de momentverdeling en met het aangrijpingspunt van de belasting:

$$M_{cr} = \alpha_m * M_{cr}^{E} * \left[\sqrt{1 + \left(\frac{0{,}4 * \alpha_m * y_Q}{M_{cr}^{E}/N_{cr,z}}\right)^2} + \frac{0{,}4 * \alpha_m * y_Q}{M_{cr}^{E}/N_{cr,z}}\right] \quad (2.20)$$

Waarbij:

$$N_{cr,z} = \frac{\pi^2 * E * I_z}{L^2} \quad (2.21)$$

De lengte in deze formules is de lengte tussen zijdelings gesteunde punten, dus tussen de steunpunten van de ligger. De term y_Q houdt rekening met het aangrijpingspunt van de belasting. De aangrijpingspunten van de belastingen die ingevoerd worden in de app zijn niet gekend. Dit kunnen welfsels zijn die tussen de flenzen van het profiel worden geplaatst, maar dit kan ook een muur zijn die bovenop het profiel wordt gebouwd. Daarom wordt de meest negatieve waarde voor deze term aangenomen. Dit is: $y_Q = -h_{profiel}/2$.

De factor α_m houdt rekening met de verschillende mogelijke momentverdelingen over het profiel. De momentverdeling is afhankelijk van het aantal lijn- en puntlasten, hun positie en hun grootte. Aangezien er vier lijn- en puntlasten ingevoerd kunnen worden, zijn er veel verschillende momentverdelingen mogelijk. Het is echter niet mogelijk om voor elke momentverdeling een exacte waarde voor α_m te bepalen. Daarom gaat de app uit van de slechts mogelijke situatie. Als er enkel puntlasten aangrijpen op het profiel, dan is α_m gelijk aan 1,35. In alle andere gevallen is α_m gelijk aan 1,13 [10].

Nu het kritisch moment gekend is, wordt er een kipcontrole uitgevoerd aan de hand van Eurocode 3. De kipweerstand van een ligger die belast wordt op buiging om de sterke as kan op volgende manier bepaald worden:

$$M_{b,Rd} = \chi_{LT} * \frac{W_y * f_y}{\gamma_{M1}} \quad (2.22)$$

Waarin:

$$\chi_{LT} = \frac{1}{\phi_{LT} + \sqrt{\phi_{LT}^2 - \bar{\lambda}_{LT}^2}} \leq 1{,}0 \quad (2.23)$$

$$\phi_{LT} = 0{,}5 * \left[1 + \alpha_{LT} * \left(\bar{\lambda}_{LT} - 0{,}2\right) + \bar{\lambda}_{LT}^2\right] \quad (2.24)$$

$$\bar{\lambda}_{LT} = \sqrt{\frac{W_y * f_y}{M_{cr}}} \quad (2.25)$$

De app maakt gebruik van gewalste I-profielen. Indien de verhouding van de hoogte op de breedte van het profiel groter is dan twee, dan is α_{LT} gelijk aan 0,34. Anders is α_{LT} gelijk aan 0,21.

In bovenstaande formules is W_y afhankelijk van de klasse van de doorsnede, zoals reeds vermeld bij het bepalen van de weerstand tegen buiging. Het profiel biedt voldoende weerstand tegen kip indien in elke doorsnede het optredend moment kleiner is dan de kipweerstand van het profiel.

Doorbuiging

De laatste controle die de app uitvoert is een controle op doorbuiging. Op basis van bovenstaande controles heeft de app reeds een optimaal profiel bepaald. Het traagheidsmoment van dit profiel is weergegeven in de staalcatalogus. Nu de elasticiteitsmodulus en het traagheidsmoment gekend zijn, bepaalt de app aan de hand van de EEM de optredende doorbuiging. Deze moet in elke doorsnede kleiner zijn dan de maximaal toelaatbare doorbuiging. De maximaal toelaatbare doorbuiging is de lengte van een overspanning gedeeld door 500. Indien de optredende doorbuiging groter is dan deze maximale waarde voldoet het profiel niet. Dan controleert de app een sterker profiel. Deze stap wordt herhaald tot er een profiel gevonden is dat aan elke controle voldoet.

2.3 Betonnen ligger

2.3.1 Algemeen

De app ontwerpt voor de gewenste situatie een gewapende betonbalk. Er wordt niet met voorspanning gewerkt. De gebruiker moet eerst een breedte ingeven. Er is geopteerd om de gebruiker de breedte te laten invoeren omdat er vaak maar een bepaalde breedte van de ligger mogelijk is, bijvoorbeeld de breedte van de kolom of de muur waar de balk wordt opgelegd. De betondekking staat standaard ingesteld op 30mm, maar de gebruiker kan deze waarde aanpassen. De balk wordt ontworpen uitgaande van betonstaal van kwaliteit BE500 en een betonkwaliteit C25/30.

2.3.2 Eurocode 2

Wanneer de breedte door de gebruiker is ingevoerd, wordt de optimale hoogte van de balk bepaald met volgende formule:

$$d_{opt} = 2{,}31 * \sqrt{\frac{M_d}{b * f_{cd}}} \tag{2.26}$$

In deze formule is M_d het maximaal optredend moment dat wordt berekend door de app. Dit kan zowel het veldmoment als het moment ter hoogte van het steunpunt zijn. De hoogte van balk wordt bekomen door de betondekking bij deze optimale hoogte op te tellen. De benodigde hoeveelheid wapening wordt met onderstaande formule bepaald:

$$A_s = \frac{M_d}{0{,}892 * d * f_{yd}} \tag{2.27}$$

De hoeveelheid wapening wordt zowel bepaald voor het gedeelte boven de steunpunten als voor de overspanning tussen de steunpunten. Boven de steunpunten is er een negatief buigmoment, dus de wapening moet bovenaan de balk geplaatst worden. In het veld tussen de steunpunten is er een positief buigmoment. Hier bevindt de wapening zich dus onderaan de balk. Er is nergens een dubbel gewapende doorsnede.

De app zet berekende hoeveelheid wapening om in een aantal staven met een bepaalde staafdiameter. Vervolgens controleert de app of er voldoende tussenafstand is tussen deze staven. De tussenafstand moet minimaal zo groot zijn als het minimum van de staafdiameter,

20mm of de grootste korreldiameter plus vijf. De app veronderstelt een korreldiameter van 32mm. Indien er niet voldoende tussenafstand is, kiest de app een grotere diameter die nog voldoet aan de wapeningseis. De app voert echter geen uitgebreide wapeningsdetaillering uit.

Nu de afmetingen van de balk en de hoeveelheid wapening gekend zijn, kan de dwarskrachtwapening bepaald worden. De controles uit de gebruiksgrenstoestand kunnen ook uitgevoerd worden. Deze controles zijn een spanningscontrole, een controle op minimumwapening, een scheurwijdtecontrole en een doorbuigingscontrole. Indien de balk niet voldoet aan deze controles wordt de balk verhoogd met vijf centimeter totdat deze wel voldoet aan alle controles.

De berekeningen in de volgende paragrafen zijn allemaal uitgevoerd uitgaande van een enkel gewapende doorsnede. Ter hoogte van de steunpunten is er enkel bovenwapening aanwezig, tussen de steunpunten is er enkel onderwapening aanwezig. De elasticiteitsmodulus van het betonstaal is in al deze berekeningen gelijk aan 200 GPa [11].

Dwarskrachtwapening

De app berekent de optredende dwarskracht in de uiterste grenstoestand. Aan de hand van deze dwarskracht kan de benodigde hoeveelheid dwarskrachtwapening worden bepaald. De dwarskrachtwapening wordt berekend voor beugels, dus deze wapening maakt een hoek van 90° met de as van de ligger. Hier moet rekening mee worden gehouden bij het plaatsen van de dwarskrachtwapening.

Eerst controleert de app of er dwarskrachtwapening nodig is. Vervolgens wordt er berekend of de drukdiagonaal in de balk bezwijkt. Als dit het geval is, is het niet mogelijk om te wapenen tegen dwarskracht. De gebruiker moet dan zelf een nieuwe situatie ingeven, bijvoorbeeld met andere afmetingen. Uiteindelijk berekent de app de benodigde dwarskrachtwapening.

De dwarskrachtweerstand van een balk zonder dwarskrachtwapening kan berekend worden met volgende formule:

$$V_{Rd,c} = \left[C_{Rd,c} * k * (100 * \rho * f_{ck})^{\frac{1}{3}} + k_1 * \sigma_{cp}\right] * b * d \tag{2.28}$$

Hiervoor geldt:

$$C_{Rd,c} = \frac{0{,}18}{\gamma_c} \tag{2.29}$$

$$k = 1 + \sqrt{\frac{200}{d}} \leq 2{,}0 \tag{2.30}$$

$$\rho = \frac{A_s}{b*d} \leq 0{,}02 \tag{2.31}$$

De term $k_1 * \sigma_{cp}$ valt weg omdat er geen normaalkracht aanwezig is. De minimumwaarde van de dwarskrachtweerstand van een balk zonder dwarskrachtwapening wordt gegeven door volgende formule:

$$V_{Rd,c,min} = v_{min} * b * d \tag{2.32}$$

Waarin:

$$v_{min} = 0{,}035 * k^{\frac{3}{2}} * f_{ck}^{\frac{1}{2}} \quad (2.33)$$

Indien de optredende dwarskracht in elke doorsnede van de balk kleiner is dan de dwarskrachtweerstand van een balk zonder dwarskrachtwapening, dan is er geen dwarskrachtwapening vereist. De balk wordt echter altijd voorzien van een minimale hoeveelheid dwarskrachtwapening. Deze wordt als volgt berekend:

$$A_{sw,min} = \frac{0{,}08*\sqrt{f_{ck}}*s*b}{f_{yk}} \quad (2.34)$$

Indien er wel dwarskrachtwapening vereist is, berekent de app eerst de maximale dwarskrachtweerstand. Als deze overschreden wordt, is het niet mogelijk om te wapenen tegen dwarskracht. De maximale dwarskrachtweerstand wordt als volgt bepaald:

$$V_{Rd,max} = \frac{\alpha_{cw}*b*z*\nu_1*f_{cd}}{\cot\theta + \tan\theta} \quad (2.35)$$

Hierbij is z de inwendige hefboomsarm. Deze mag gelijk genomen worden aan 0,9*d. ν_1 is een sterktereductiefactor voor beton dat gescheurd is door dwarskracht. Deze is gelijk aan 0,6. α_{cw} is een factor die rekening houdt met de spanning in de op druk belaste rand. Deze is gelijk aan 1,0. De hoek θ is de hoek die de drukdiagonaal van het beton maakt met de as van de ligger. Hiervoor wordt een gemiddelde waarde gehanteerd, namelijk $\theta = 26{,}57°$ ($\cot\theta = 2$). Nu alle termen gekend zijn kan de maximale dwarskrachtweerstand van de balk bepaald worden. De balk moet vervolgens ook voldoen aan volgende voorwaarde:

$$V_{Ed} \leq 0{,}5 * b * d * \nu * f_{cd} \quad (2.36)$$

Waarin:

$$\nu = 0{,}6 * \left(1 - \frac{f_{ck}}{250}\right) \quad (2.37)$$

Uiteindelijk kan de benodigde hoeveelheid dwarskrachtwapening bepaald worden met volgende formule:

$$A_{sw} = \frac{V_{Ed}*s}{f_{ywk}/1{,}15*z*\cot\theta} \geq A_{sw,min} \quad (2.38)$$

Hierbij is s de tussenafstand tussen de beugels. De app gaat uit van een tussenafstand van 25cm tussen de beugels. Bij de beugels wordt ook uitgegaan van staalsoort BE500. Bijgevolg is f_{ywk} gelijk aan 500 N/mm^2. De maximale oppervlakte dwarskrachtwapening wordt gegeven door volgende formule:

$$A_{sw,max} = \frac{\alpha_{cw}*\nu_1*f_{cd}*b*s}{2*f_{ywk}/1{,}15} \quad (2.39)$$

Indien de berekende dwarskrachtwapening deze waarde overschrijdt past de app de tussenafstand van de beugels aan met vijf centimeter. Vervolgens controleert de app opnieuw de dwarskrachtwapening totdat de balk voldoet.

Spanningscontrole

De spanningscontrole wordt uitgevoerd in de gebruiksgrenstoestand. De app stelt dus een nieuwe matrix op om het buigmoment in de gebruiksgrenstoestand te berekenen. Om deze

matrix op te kunnen stellen moeten er eerst belastingen ingevoerd worden. Dan wordt er een belastingscombinatie gemaakt. Deze belastingen worden in de matrix ingevoerd. De spanningscontrole wordt zowel voor de steunpunten als voor het veld uitgevoerd.

De belastingscombinatie in de app is niet conform met de Eurocode. Het merendeel van de belastingen zijn permanente belastingen, er zullen slechts sporadisch variabele belastingen voorkomen. Daarom wordt er bij de belastingscombinatie geen rekening gehouden met het gelijktijdig optreden van meerdere variabele belastingen (dus met de ψ-factoren). De belastingcombinatie wordt voor een algemene GGT-situatie uitgevoerd:

$$GGT\colon \sum G + \sum Q \tag{2.40}$$

Omdat er geen rekening gehouden wordt met de gelijktijdigheid van belastingen, is de belasting die in de matrix wordt ingevoerd een kleine overschatting van de werkelijke belastingscombinatie. Dit is dus een veilige benadering van de controles in de gebruiksgrenstoestand.

Bij deze spanningscontrole wordt er gecontroleerd of de staalspanning en de betonspanning de toelaatbare spanningen in het materiaal niet overschrijden. Aangezien er voor GGT een algemene belasting is bepaald, wordt de spanningscontrole enkel voor de quasi-blijvende gebruiksgrenstoestand uitgevoerd. In GGT-Q zijn de spanningseisen namelijk het strengst. De toelaatbare betonspanning is gelijk aan $0{,}45 * f_{ck}$ en de toelaatbare staalspanning is gelijk aan $0{,}8 * f_{yk}$.

Om de staalspanning en de betonspanning te berekenen, moet eerst de verhouding van de elasticiteitsmodulus van staal tot de elasticiteitsmodulus van beton gekend zijn. Deze wordt bepaald op volgende manier:

$$E_{cm} = 22\,000 * \left(\frac{f_{ck}+8}{10}\right)^{0,3} \tag{2.41}$$

$$\alpha = \frac{E_S}{E_{cm}} \tag{2.42}$$

Vervolgens berekent de app de betondrukhoogte van een gescheurde betondoorsnede:

$$x = \frac{-\alpha * A_S}{b} + + \sqrt{\left(\frac{\alpha * A_S}{b}\right)^2 + \frac{2 * \alpha * A_S * d}{b}} \tag{2.43}$$

Aan de hand van deze betondrukhoogte en het maximaal optredend buigmoment kan de betonspanning en daarna de staalspanning bepaald worden:

$$\sigma_c = \frac{M_{Ed,max} * x}{\frac{b * x^3}{3} + \alpha * A_S * (d-x)^2} \tag{2.44}$$

$$\sigma_s = \alpha * \sigma_c * \frac{(d-x)}{x} \tag{2.45}$$

Dit zijn de optredende spanningen in de materialen. Deze moeten voldoen aan de spanningseisen in GGT-Q:

$$\sigma_c \leq 0{,}45 * f_{ck} \tag{2.46}$$

$$\sigma_s \leq 0{,}8 * f_{yk} \tag{2.47}$$

Indien de balk aan deze voorwaarden voldoet, dan kan de volgende controle uitgevoerd worden. Als de balk niet voldoet wordt de balk vijf centimeter verhoogd en vervolgens opnieuw gecontroleerd. Deze stap wordt herhaald tot de balk voldoet aan deze controle.

Minimumwapening

Een minimum hoeveelheid wapening dient altijd aanwezig te zijn in een balk om de scheurvorming te beheersen en om een brosse breuk te vermijden. De minimale hoeveelheid wapening die nodig is om scheurbeheersing in de balk te garanderen wordt bepaald met volgende formule:

$$A_{s,min} = \frac{0{,}4 * k * 0{,}3 * f_{ck}^{\frac{2}{3}} * \frac{b*h}{2}}{500} \tag{2.48}$$

Hierbij wordt de term k op volgende manier bepaald:

$$h \leq 300 \quad \rightarrow \quad k = 1 \tag{2.49}$$

$$h \geq 800 \quad \rightarrow \quad k = 0{,}65 \tag{2.50}$$

$$300 \leq k \leq 800 \quad \rightarrow \quad k = 1{,}21 - 0{,}7 * \frac{h}{1000} \tag{2.51}$$

Vervolgens berekent de app ook de minimale hoeveelheid wapening die nodig is om brosse breuk te vermijden:

$$A_{s,min,BB} = 0{,}26 * \frac{0{,}3 * f_{ck}^{\frac{2}{3}}}{f_{yk}} * b * d \geq 0{,}0013 * b * d \tag{2.52}$$

Tenslotte controleert de app of de voorgestelde hoeveelheid wapening A_s groter is dan deze minimumwapening. Indien dit niet het geval is, wordt de voorgestelde wapeningshoeveelheid verhoogd naar deze minimumwaarde.

Scheurwijdte

De scheurwijdtecontrole wordt uitgevoerd in GGT-Q. De app voert deze controle uit op basis van de algemene belasting in GGT die bepaald is in de vorige paragraaf. Er wordt zowel voor de steunpunten als voor het veld een scheurwijdtecontrole uitgevoerd. Eerst berekent de app het scheurmoment van de ligger. De scheurwijdteberekening dient enkel uitgevoerd te worden indien het maximaal optredend moment groter is dan het scheurmoment. De scheurwijdte in de balk mag maximaal 0,3mm zijn.

De app berekent dus eerst het scheurmoment. Om dit te kunnen berekenen moet de betondrukhoogte en het traagheidsmoment van de ongescheurde betondoorsnede gekend zijn:

$$x = \frac{\frac{b*h^2}{2} + \alpha * A_s * d}{\alpha * A_s + b * h} \tag{2.53}$$

$$I = \frac{b*x^3}{3} + \frac{b}{3} * (h - x)^3 + \alpha * A_s * (d - x)^2 \tag{2.54}$$

De term α in deze formule is reeds bepaald in de vorige paragraaf. Met deze gegevens berekent de app het scheurmoment op de volgende manier:

$$M_{cr} = 0{,}3 * f_{ck}^{\frac{2}{3}} * \frac{I}{h-x} \tag{2.55}$$

Indien het optredend moment groter is dan dit scheurmoment wordt er een scheurwijdteberekening uitgevoerd voor een langdurige belasting. De app bepaalt enkel voor de langdurige belasting de scheurwijdte omdat het merendeel van de ingevoerde belastingen permanente belastingen zijn. De scheurwijdte in de balk wordt bepaald na enkele tussenberekeningen:

$$h_c = \min\left[\frac{h-x}{3}; 2{,}5(h-d); \frac{h}{2}\right] \tag{2.56}$$

$$\rho_{p,eff} = \frac{A_s}{h_c * b} \tag{2.57}$$

$$\epsilon_{sm} - \epsilon_{cm} = \frac{\sigma_s - 0{,}4 * \frac{0{,}3 * f_{ck}^{\frac{2}{3}}}{\rho_{p,eff}} * (1 + \alpha * \rho_{p,eff})}{E_s} \geq 0{,}6 * \frac{\sigma_s}{E_s} \tag{2.58}$$

In bovenstaande formule is σ_s de staalspanning. Deze wordt bepekt volgens Eurocode 2. Volgende formule is de laatste tussenberekening die nodig is vooraleer de scheurwijdte bepaald kan worden.

$$S_{r,max} = 3{,}4 * c + 0{,}8 * 0{,}5 * 0{,}425 * \frac{\phi}{\rho_{p,eff}} \tag{2.59}$$

Hierin is c de betondekking (30mm, tenzij anders ingevoerd) en ϕ de wapeningsdiameter. Met deze gegevens berekent de app de uiteindelijk scheurwijdte van de ligger:

$$w_k = S_{r,max} * (\epsilon_{sm} - \epsilon_{cm}) \tag{2.60}$$

Deze scheurwijdte moet kleiner zijn dan 0,3mm. Indien dit niet zo is wordt de hoogte van de balk aangepast tot de balk wel voldoet. Dan gaat de app verder met de volgende controle.

Doorbuiging

De belasting die aangrijpt op de betonbalk is de som van de ingevoerde belastingen en het eigengewicht van de betonbalk. Op basis van deze belasting en de afmetingen van de balk wordt de doorbuiging bepaald.

Eerst bepaalt de app de effectieve elasticiteitsmodulus van het beton. Omdat de fictieve dikte, de relatieve vochtigheid en de ouderdom van het beton op het moment van belasten bij elke situatie variëren, wordt er een vaste kruipfactor gehanteerd. Hierbij is de kruipfactor $\phi(\infty, t_0)$ gelijk aan 1,7.

$$E_{c,eff} = \frac{22\,000 * \left(\frac{f_{ck}+8}{10}\right)^{0{,}3}}{1 + \phi(\infty, t_0)} \tag{2.61}$$

Met deze effectieve elasticiteitsmodulus van beton kan α bepaald worden:

$$\alpha = \frac{E_s}{E_{c,eff}} \tag{2.62}$$

Vervolgens bepaalt de app de betondrukhoogte x en het traagheidsmoment I voor de ongescheurde en de gescheurde situatie.

$$x_{ongescheurd} = \frac{\alpha * A_s * d + b * h^2 * 0{,}5}{\alpha * A_s + b * h} \quad (2.63)$$

$$x_{gescheurd} = \frac{-\alpha}{b} * A_s + \left[\left(\frac{\alpha}{b} * A_s\right)^2 + 2 * \frac{\alpha}{b} * A_s * d\right]^{0,5} \quad (2.64)$$

$$I_{ongescheurd} = \frac{b * x_{ongescheurd}^3}{3} + \frac{b}{3} * \left(h - x_{ongescheurd}\right)^3 + \alpha * A_s * \left(d - x_{ongescheurd}\right)^2 \quad (2.65)$$

$$I_{gescheurd} = \frac{b * x_{gescheurd}^3}{3} + \alpha * A_s * \left(d - x_{gescheurd}\right)^2 \quad (2.66)$$

Aan de hand van deze berekende traagheidsmomenten en de effectieve elasticiteitsmodulus van beton berekent de app met behulp van de EEM de doorbuiging van de balk in de gescheurde en ongescheurde toestand. Deze worden gecombineerd tot de uiteindelijke doorbuiging van de ligger.

Vooraleer de doorbuigingen gecombineerd worden, berekent de app het scheurmoment van de balk met volgende formule:

$$M_r = 0{,}3 * f_{ck}^{\frac{2}{3}} * \frac{I_{ongescheurd}}{h - x_{ongescheurd}} \quad (2.67)$$

Aan de hand van dit scheurmoment kan de verdelingsfactor bepaald worden:

$$\zeta = 1 - \beta * \left(\frac{M_r}{M_{Ed}}\right)^2 \quad (2.68)$$

De app bepaalt de doorbuiging w op lange duur. Daarom is β gelijk aan 0,5. In bovenstaande formule is M_{Ed} het maximaal veldmoment. De uiteindelijke doorbuiging van de ligger wordt verkregen door de doorbuiging in gescheurde en in ongescheurde toestand te combineren door middel van de verdelingsfactor:

$$w = \zeta * w_{gescheurd} + (1 - \zeta) * w_{ongescheurd} \quad (2.69)$$

Deze doorbuiging moet in elke doorsnede kleiner zijn dan de maximaal toelaatbare doorbuiging.

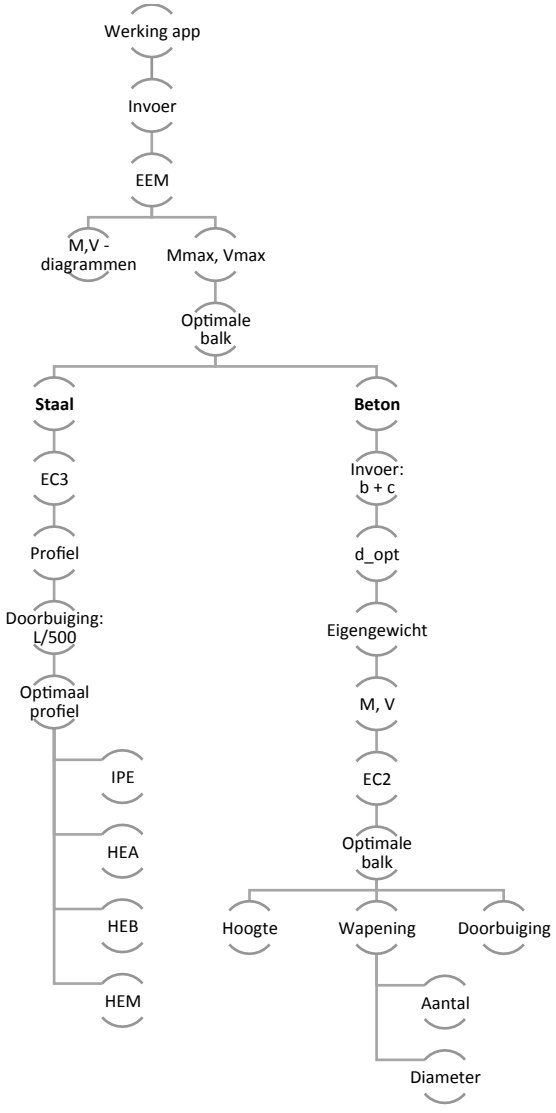
3 Ontwikkeling en implementatie app

3.1 Algemeen

Voordat er concreet wordt ingegaan op de code voor de ontwikkeling van de app, wordt er eerst een algemene toelichting gegeven. Volgende paragrafen bevatten de manier van werken binnen de app en een toelichting over Xcode. Deze twee punten zorgen ervoor dat de lezer al een beeld heeft over de app vooraleer de concrete ontwikkeling wordt beschreven. De elementaire delen van de code staan gegeven in de vorm van schermafbeeldingen. De overige belangrijke delen, daar deze meestal te uitgebreid zijn om te vermelden binnen een doorlopende tekst, staan vermeld in bijlage A. De volledige code en de documentatie over het ontwerp en implementatie van de app worden voorzien in de vorm van een extern opslagmedium.

3.1.1 Werking app

Onderstaande figuur toont de flowchart die schematisch de filosofie achter de werking van de app weergeeft.



Figuur 3.1: flowchart werking app

De eerste mogelijkheid van de app is dat de gebruiker de overspanningen en belastingen kan invoeren. De app is beperkt tot maximale overspanningen van vijftien meter. In totaal kunnen er vier overspanningen worden ingevoerd. De belastingen op de balk zijn punt- en/of lijnlasten. De gebruiker kan ook aanduiden of het een variabele of permanente last is. Wanneer de gegevens worden ingegeven, stelt de app de noodzakelijke matrices ([F], [K] en [u]) op. Aan de hand van de overspanningen berekent de app de dimensie van de drie matrices. Vervolgens worden de krachten toegevoegd aan de krachtenmatrix.

Wanneer de invoer door de gebruiker is afgerond gaat de app de EEM toepassen om de M, V–diagrammen te bepalen. De stijfheidsmatrix wordt opgesteld met gegevens van een stalen profiel IPE 100. Voor het opstellen van deze matrix zijn de exacte waarden van E, I en A nog niet van belang. Voor alle soorten balken moet het verloop van de M, V-diagrammen identiek zijn. Vervolgens stelt de app de M, V – diagrammen visueel voor. Ondertussen is het ook noodzakelijk dat de maximale dwarskracht (V_{max}) en het maximaal moment (M_{max}) op de balk worden bepaald. Onder een maximale waarde wordt de grootste absolute waarde verstaan.

Aan de hand van V_{max} en M_{max} kan de app voor elk soort balk een optimale balk berekenen. De optimalisatie van balken gebeurt voor een stalen en betonnen balk respectievelijk met behulp van Eurocode 3 en Eurocode 2. De werkwijze achter de dimensionering van stalen en betonnen balken zijn afzonderlijk opgedeeld, zoals getoond in figuur 3.1.

De keuze van het optimale stalen profiel gebeurt aan de hand van profielen uit de catalogi van ArcelorMittal [12]. De controles uit Eurocode 3 bepalen voor elk type profiel het kleinste profiel dat voldoet aan alle eisen. Vervolgens wordt met de gegevens van die profielen de doorbuiging bepaald. Wanneer de doorbuiging de waarde van $L/500$ overschrijdt, wordt met de gegevens van het volgende profiel uit de catalogus de doorbuiging berekend. Dit proces herhaalt zich totdat voor alle types (IPE, HEA, HEB en HEM) een optimaal profiel is geselecteerd.

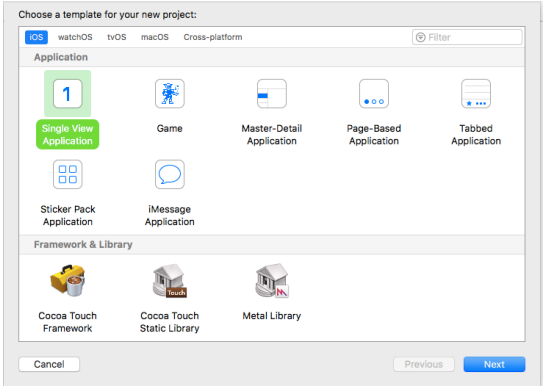
De optimalisatie van de betonnen balk is uitgevoerd met een ingevoerde breedte en betondekking. De gebruiker moet deze twee waarden ingeven. Hiermee zal de app vervolgens de optimale balk berekenen. Aan de hand van de breedte en M_{max} is het mogelijk een eerste schatting van de hoogte van de balk te maken: $h = d_{opt} + c$. Met deze schatting wordt vervolgens het eigengewicht van de betonnen balk toegevoegd aan de krachtenmatrix. Het gewicht van gewapend beton bedraagt afgerond 25 kN/m^3 [13]. Doordat er een extra kracht aangrijpt op de balk, rekent de app opnieuw V_{max} en M_{max} uit. Met deze waarden worden dan de bijhorende controles uit Eurocode 2 uitgevoerd. Indien de balk met de geschatte hoogte niet voldoet aan de controles, wordt de hoogte verhoogd met vijf centimeter en vervolgens worden voorgaande stappen herhaald totdat de balk wel voldoet. Op basis van de controles wordt ook de nodige hoeveelheid wapening bepaald. Deze wapening wordt dan concreet omgezet in een aantal staven met bijhorende diameter. Tot slot berekent de app via de EEM de doorbuiging van de betonnen balk.

3.1.2 Xcode 8.3

Het doel van de app is om hyperstatische balken uit te rekenen met behulp van een IPhone en eventueel een IPad. Na onderzoek blijkt dat Xcode 8.3 het best geschikt is voor de ontwikkeling van de app [3]. Hier volgt een beknopte uitleg over de werking van Xcode 8.3. De uitleg is enkel toegepast op de mogelijkheden van het programma die van nut zijn voor de app.

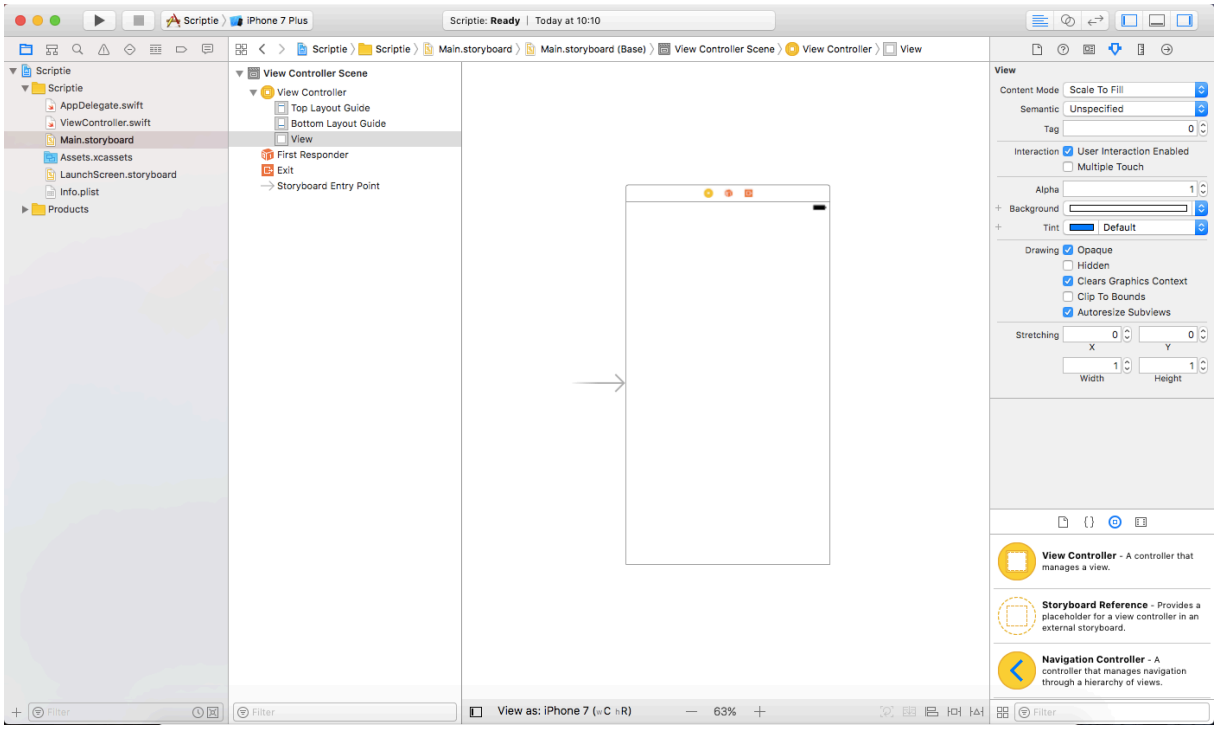
Als eerste stap moet er een nieuw project worden aangemaakt. Xcode geeft de mogelijkheid om aan de slag te gaan met verschillende templates. Onderstaande figuur geeft de templates weer

waaruit de gebruiker kan kiezen. De meest voor de hand liggende keuze is de single view application. De keuze voor deze template biedt de programmeur de meeste keuzevrijheid om de app te ontwerpen. Met deze template kan de gebruiker meerdere schermen creëren voor de app.



Figuur 3.2: templates Xcode project [14]

Nadat een project is aangemaakt kan de gebruiker aan de slag met de verschillende mogelijkheden die Xcode biedt. Toegepast op deze masterproef is er geopteerd om te beginnen met het opstellen van een voorlopige lay-out. De opstelling van deze lay-out is mogelijk met de toepassing main.storyboard. Dit is een functie binnen de software waarmee de gebruiker een configuratie kan maken aan de hand van voorgeprogrammeerde pictogrammen, labels, schermen… Onderstaande schermafdruk geeft weer hoe de main.storyboard is opgebouwd.

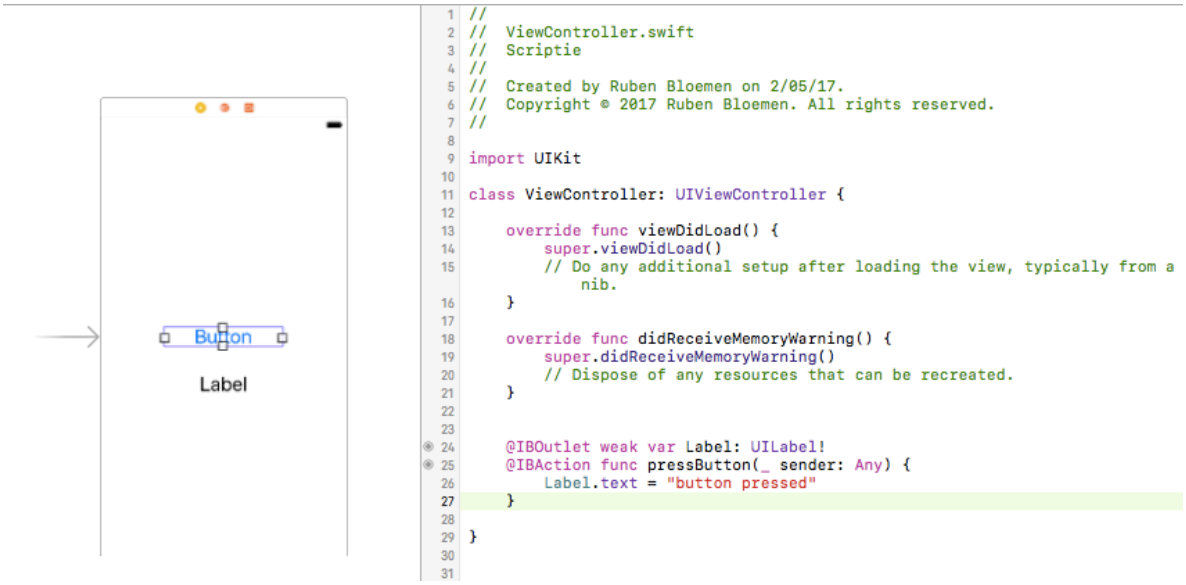


Figuur 3.3: main.storyboard [14]

Figuur 3.3 geeft de situatie weer bij de aanvang van het project. Xcode geeft een scherm weer in de vorm van een View Controller. De ontwerper kan een onbeperkt aantal View Controllers toevoegen. Deze schermen kunnen achteraf met elkaar worden verbonden. Binnen de main.storyboard is het mogelijk om verschillende pictogrammen toe te voegen. Deze pictogrammen hebben specifieke kenmerken met betrekking tot het doel waarvoor ze bestemd zijn. De pictogrammen behoren tot de UIKit [15]. Dit is een bestaande tool binnen Xcode om de gebruikersomgeving (userinterface) op te bouwen en te beheren.

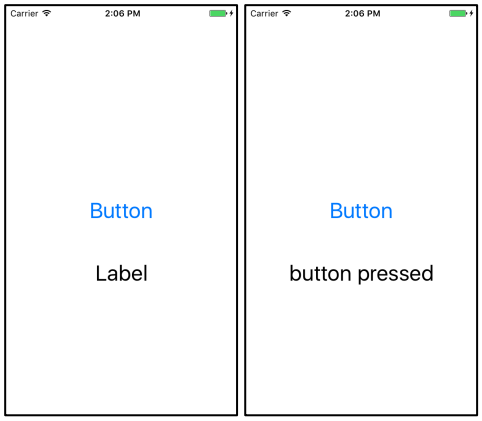
Op het moment dat de gebruikersomgeving is aangemaakt, kan de gebruiker van start gaan met het programmeren van de toepassingen. Voor een individueel scherm moet een aparte klasse worden aangemaakt. Voor de eenvoud is er geopteerd om deze klasses te benoemen in volgorde in de app. Zo zal de klasse van het derde scherm de naam viewController3.swift krijgen. Naargelang het doel van de klasse moeten er andere tools worden geïmporteerd. In het totaal zijn er drie tools van Xcode en een externe tool geïmporteerd. Voor de code binnen de verschillende klassen wordt er verwezen naar de volgende paragrafen: 3.2, 3.3 en 3.4.

Tenslotte is het van belang om te weten hoe men de invoer van de gebruiker kan weergeven in de vorm van een variabele of actie binnen de desbetreffende klasse [16]. De wijze waarop een actie omgezet wordt in een output is aangetoond aan de hand van een voorbeeld. Het voorbeeld bestaat uit de mogelijk om op een knop te duwen. Wanneer op deze knop wordt geduwd, zal er op het scherm verschijnen dat deze ingedrukt is geweest. De werking binnen Xcode is getoond in onderstaande code.



Figuur 3.4: voorbeeld in- en uitvoer

Xcode geeft de mogelijkheid om de opstelling van de app te tonen aan de hand van een configuratie. Wanneer de code, weergegeven in Figuur 3.4, wordt geconfigureerd zal het woord 'Label' vervangen worden door 'button pressed'. De configuratie is weergegeven in onderstaande afbeeldingen.



Figuur 3.5: configuratie voorbeeld

Voorgaande toelichting over de werking van Xcode 8.3 heeft als doel om de volgende informatie over de ontwikkeling van de app beter te kunnen begrijpen. Deze uitleg is slechts een beknopte versie om niet te uitgebreid stil te moeten staan bij het programma. De nadruk van de masterproef ligt op de berekeningen die de app moet uitvoeren en vooral de optimalisatie van de balken.

3.2 Implementatie eindige-elementenmethode

Zoals reeds is aangegeven in de uitleg over de werking, past de app meermaals de EEM toe. Xcode is echter geen programma om complexe wiskundige problemen op te lossen. Het opstellen van een matrix en het berekenen van M,V – diagrammen is niet zo vanzelfsprekend. Voor het oplossen van een matrixvergelijkingen zijn alternatieve oplossingen gevonden. Dit kan niet door slechts de bewerking in te moeten geven. Xcode beperkt zich tot het uitvoeren van de standaardbewerkingen (vermenigvuldiging, optelling, etc.).

3.2.1 Opstellen matrices

De eerste vereiste om de implementatie van de methode mogelijk te maken was het opstellen van matrices. De software zelf heeft geen functie die een matrix kan opstellen. Na verder onderzoek blijkt dat het mogelijk is om zelf een structuur op te stellen die het werken met matrices mogelijk maakt. Onderstaande code illustreert hoe een dergelijke structuur kan worden aangemaakt [17]. De software zet deze matrix om in één array. Deze functie maakt het echter wel mogelijk om waarden uit de matrix op te vragen aan de hand van de positie binnen deze matrix.

```
// het opstellen van een matrixfunctie
struct Matrix {
    let rows: Int, columns: Int
    var grid: [Double]
    init(rows: Int, columns: Int) {
        self.rows = rows
        self.columns = columns
        grid = Array(repeating: 0.0, count: rows * columns)
    }
    func indexIsValid(row: Int, column: Int) -> Bool {
        return row >= 0 && row < rows && column >= 0 && column < columns
    }
    subscript(row: Int, column: Int) -> Double {
        get {
            assert(indexIsValid(row: row, column: column), "Index out of range")
            return grid[(row * columns) + column]
        }
        set {
            assert(indexIsValid(row: row, column: column), "Index out of range")
            grid[(row * columns) + column] = newValue
        }
    }
}
```

Code 3.1: structuur matrix

Op het moment dat de gebruiker de overspanningen heeft ingevuld, gaat de app al de matrices opstellen. Alle waarden binnen de matrix staan op dat ogenblik nog op nul. De dimensie van een matrix kan de app berekenen aan de hand van de totale lengte van de overspanningen. De lengte van één element is vast ingesteld op vijf centimeter. Voor ieder knooppunt zijn er drie vrijheidsgraden. Met deze gegevens stelt de app de matrices op aan de hand van de volgende code.

```
//berekenen van het aantal elementen -> dimensie matrices
var elemPerMeter = 20.0
var aantal_elementen = Int((afstand_veld_1+afstand_veld_2+afstand_veld_3+afstand_veld_4)*elemPerMeter)
var dimensie_matrices: Int = aantal_elementen*3+2
var lengteElement = 1.0/elemPerMeter

//opstellen van matrices
var K_globaal_UGT = Matrix(rows:dimensie_matrices+1, columns:dimensie_matrices+1)
var K_globaal_GGT = Matrix(rows:dimensie_matrices+1, columns:dimensie_matrices+1)
var krachten_UGT = Matrix(rows: dimensie_matrices+1, columns:1)
var krachten_GGT = Matrix(rows: dimensie_matrices+1, columns:1)
var verplaatsingen_UGT = Matrix(rows:dimensie_matrices+1, columns:1)
var verplaatsingen_GGT = Matrix(rows:dimensie_matrices+1, columns:1)
```

Code 3.2: opstellen matrices

Belangrijk om te weten is dat binnen deze structuur het eerste element zich niet op positie (1,1) maar op positie (0,0) bevindt. Deze kanttekening moet duidelijk maken waarom de variabele dimensie_matrices één waarde lager ligt dan het aantal elementen rijen en/of kolommen. De variabele dimensie_matrices wordt gebruikt om het laatste element van een matrix op te vragen.

Het toevoegen van een punt– of lijnlast is de volgende stap van de app. Deze krachten worden toegevoegd aan de krachtenmatrix. Voor het opstellen van de krachtenmatrix is het voldoende om te weten dat deze worden toegevoegd op de juiste positie, bepaald door de gebruiker. De volledige code achter het ingeven en verwerken van de belastingen in de krachtenmatrix is gegeven in bijlage A.1. De krachten toevoegen gebeurt met de volgende twee functies:

```
    //een functie om een puntlast toe te voegen aan de globale krachtenmatrix
    func voegPuntlastToe_UGT (Grootte: Double, Positie: Double) {
        krachten_UGT[Int(Positie*elemPerMeter)*3+1,0] = krachten_UGT[Int(Positie*elemPerMeter)*3+1,0] - Grootte
    }
```

```
    //een functie om een lijnlast toe te voegen aan de krachtenmatrix
    //een lijnlast wordt gekenmerkt door 2 equivalente puntkrachten en momenten
    func VoegLijnlastToe_UGT (Grootte: Double, Beginpositie: Double, Eindpositie: Double) {
        if Eindpositie > Beginpositie{
        for i in Int(Beginpositie*elemPerMeter)...Int(Eindpositie*elemPerMeter)-2{
                krachten_UGT[i*3+1, 0] = krachten_UGT[i*3+1, 0] - Grootte*lengteElement/2           //equivalente puntlast links: F= -L/2
                krachten_UGT[i*3+4, 0] = krachten_UGT[i*3+4, 0] - Grootte*lengteElement/2           //equivalente puntlast rechts: F= -qL/2
                krachten_UGT[i*3+2, 0] = krachten_UGT[i*3+2, 0] - Grootte*pow(lengteElement,2)/8    //equivalent Moment links: F= -qL^2/8
                krachten_UGT[i*3+5, 0] = krachten_UGT[i*3+5, 0] + Grootte*pow(lengteElement,2)/8
            }
        }
        else if Beginpositie > Eindpositie{
            print("Lijnlast foutief ingegeven")
        }
    }
```

Code 3.3: krachten toevoegen

Deze functies worden gebruikt voor het invoeren van krachten in UGT. In het geval dat de krachten in GGT ingevoerd moeten worden, zijn de functies analoog. Het verschil zit in het feit dat er een andere veiligheid wordt toegepast in GGT. De invoer van de grootte wordt hierbij aangepast met de bijhorende veiligheidsfactor.

Tot slot moet ook de globale stijfheidsmatrix nog worden aangepast. De meest efficiënte manier is door eerst de stijfheidsmatrix voor een afzonderlijk element op te stellen en vervolgens te assembleren tot de globale matrix. De elementenmatrix is voor elk element identiek. Voor het berekenen van de dwarskrachten en momenten worden E, I en A van een IPE 100 profiel gebruikt. De stijfheidsmatrix van een element wordt opgesteld volgens onderstaande code.

```
//opstellen van een stijfheidsmatrix voor een afzonderlijk element. Deze is dezelfde voor elk element.
var K_element = Matrix(rows: 6, columns: 6)

//De stijfheidsmatrix van een element verandert bij een verschillend profiel door zijn traagheidsmoment I en oppervlakte A
func StijfheidsmatrixElement(I: Double, A: Double) {
    let term1 = A*E_staal/L                         // term1 = A*E/L
    let term2 = E_staal*I/pow(L,3)                  // term2 = EI/L^3
    let term3 = E_staal*I/pow(L,2)                  // term3 = EI/L^2
    let term4 = E_staal*I/L                         // term4 = EI/L

    for i in 0...5 {
        if i == 0{
            K_element[i,0] = term1
            K_element[i,3] = -term1
        }
        else if i == 1 {
            K_element[i,1] = 12*term2
            K_element[i,2] = 6*term3
            K_element[i,4] = -12*term2
            K_element[i,5] = 6*term3
        }
        else if i == 2 {
            K_element[i,1] = 6*term3
            K_element[i,2] = 4*term4
            K_element[i,4] = -6*term3
            K_element[i,5] = 2*term4
        }
        else if i == 3 {
            K_element[i,0] = -term1
            K_element[i,3] = term1
        }
        else if i == 4 {
            K_element[i,1] = -12*term2
            K_element[i,2] = -6*term3
            K_element[i,4] = 12*term2
            K_element[i,5] = -6*term3
        }
        else if i == 5 {
            K_element[i,1] = 6*term3
            K_element[i,2] = 2*term4
            K_element[i,4] = -6*term3
            K_element[i,5] = 4*term4
        }
    }
}
```

Code 3.4: stijfheidsmatrix van één element

Met deze elementenmatrix wordt vervolgens de globale stijfheidsmatrix geassembleerd. De waarden van twee elementenmatrix worden opgeteld. Dit houdt in dat er om de drie rijen en kolommen een nieuwe stijfheidsmatrix van een element wordt toegevoegd. Onderstaande code maakt de geassembleerde matrix aan zoals geïllustreerd in Figuur 2.5.

```
//globale stijfheidsmatrix [K] opstellen
func StijfheidsmatrixGlobaal() {
    for i in 0...Int(aantal_elementen)-1 {
        for j in 0...5 {
            for k in 0...5{
                K_globaal_UGT[i*3+j,i*3+k] = K_globaal_UGT[i*3+j,i*3+k] + K_element[j,k]
            }
        }
    }
}
```

Code 3.5: globale stijfheidsmatrix

3.2.2 Berekenen interne krachten

Nadat de matrices uit de matrixvergelijking zijn opgesteld, is het mogelijk om de verplaatsingen te berekenen. De krachtenmatrix en globale stijfheidsmatrix zijn ondertussen aangemaakt. Met de vergelijking $[u] = [K]^{-1} * [F]$ kan voor elk knooppunt de drie verplaatsingen worden bepaald. Zoals de formule aangeeft is er een functie nodig die de stijfheidsmatrix kan inverteren. Door de tool Accelerate van Xcode te importeren, is het mogelijk om een matrix te inverteren. [18] De geïnverteerde matrix zal dan wel een eendimensionale array worden met achtereenvolgens de verschillende rijen van de geïnverteerde matrix. Volgende code maakt het mogelijk een matrix om te zetten in zijn inverse vorm.

```
//functie om een NxN matrix te inverteren
func invert(matrix : [Double]) -> [Double] {
    var inMatrix:[Double]       = matrix
    // Get the dimensions of the matrix. An NxN matrix has N^2
    // elements, so sqrt( N^2 ) will return N, the dimension
    var N:__CLPK_integer        = __CLPK_integer( sqrt( Double( matrix.count ) ) )
    // Initialize some arrays for the dgetrf_(), and dgetri_() functions
    var pivots:[__CLPK_integer] = [__CLPK_integer](repeating: 0, count: Int(N))
    var workspace:[Double]      = [Double](repeating: 0.0, count: Int(N))
    var error: __CLPK_integer   = 0
    // Perform LU factorization
    dgetrf_(&N, &N, &inMatrix, &N, &pivots, &error)
    // Calculate inverse from LU factorization
    dgetri_(&N, &inMatrix, &N, &pivots, &workspace, &N, &error)
    return inMatrix
}
```

Code 3.6: inverse matrix

Vooraleer de verplaatsingen effectief berekend kunnen worden, moeten de randvoorwaarden worden toegevoegd. In bijlage A.2 bevindt zich de code voor het toevoegen van de randvoorwaarden. Als deze randvoorwaarden zijn toegevoegd, kunnen de verplaatsingen worden berekend. De volledige functie voor het berekenen van de verplaatsingen is weergegeven in bijlage A.3. De regels code die nodig zijn om de matrixbewerking uit te voeren, zijn nog eens herhaald in deze bijlage. Deze code is belangrijk om in te zien dat Xcode zelf geen matrixvergelijking kan oplossen. Onderstaande voorbeeld toont aan hoe een matrixvermenigvuldiging in zijn werk gaat. De code gaat deze bewerking handmatig uitvoeren en opslaan in de verplaatsingsmatrix.

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (1x + 2y) \\ (3x + 4y) \end{bmatrix} \tag{3.1}$$

```
 for j in 0...dimensie_matrices{
        verplaatsingen_UGT[i,0] = verplaatsingen_UGT[i,0] + krachten_UGT[j,0]*K_globaal_RVW_inv_UGT[dimensie_matrices*i+i+j]
}
```

Code 3.7: vermenigvuldiging matrix

Tot slot moeten de interne krachten worden berekend. Hiervoor zijn twee bijkomende kolommatrices opgesteld per grenstoestand. In deze twee matrices moeten respectievelijk de momenten en dwarskrachten op een knooppunt worden aangemaakt. De matrix bestaat logischerwijs uit evenveel waarden als het aantal knooppunten. Het verschil met het berekenen van de verplaatsingen is het feit dat momenten en dwarskrachten worden berekend met behulp van de stijfheidsmatrix van één element. De uiteindelijke momenten en dwarskrachten worden bepaald met onderstaande functies.

```
    func BepaalMomenten(){
        for i in 0...aantal_elementen-1{
            for j in 0...5{
                momenten_UGT[i,0] = momenten_UGT[i,0] + K_element[2,j]*verplaatsingen_UGT[i*3+j,0]
            }
        }
    }

    func BepaalDwarskrachten(){
        for i in 0...aantal_elementen-1{
            for j in 0...5{
                dwarskrachten_UGT[i,0] = dwarskrachten_UGT[i,0] + K_element[1,j]*verplaatsingen_UGT[i*3+j,0]
            }
            //verplaatsingen in steunpunt mee in rekening brengen
            if i == Int(scharnier_2*elemPerMeter), afstand_veld_1 > 0.0{
                dwarskrachten_UGT[i,0] = dwarskrachten_UGT[i,0] + K_element[1,1]*u_scharnier2
            }
            else if i == Int(scharnier_3*elemPerMeter), afstand_veld_2 > 0.0{
                dwarskrachten_UGT[i,0] = dwarskrachten_UGT[i,0] + K_element[1,1]*u_scharnier3
            }
            else if i == Int(scharnier_4*elemPerMeter), afstand_veld_3 > 0.0{
                dwarskrachten_UGT[i,0] = dwarskrachten_UGT[i,0] + K_element[1,1]*u_scharnier4
            }
            else if i == Int(scharnier_5*elemPerMeter), afstand_veld_4 > 0.0{
                dwarskrachten_UGT[i,0] = dwarskrachten_UGT[i,0] + K_element[1,1]*u_scharnier5
            }
        }
        //laatste element moet ook nog worden toegevoegd
        for j in 0...5{
            dwarskrachten_UGT[aantal_elementen,0] = dwarskrachten_UGT[aantal_elementen,0] - K_element[4,j]*verplaatsingen_UGT[aantal_elementen*3-3+j,0]
        }
    }
```

Code 3.8: momenten uitrekenen

Om alle waarden van deze vergelijkingen te bepalen zijn verschillende handelingen in een specifieke volgorde noodzakelijk. Wanneer de gebruiker op een knop duwt zal er een functie

worden aangesproken die deze handelingen in chronologische volgorde gaat uitvoeren. Deze functie is de kern voor het berekenen van dwarskrachten en momenten met de EEM. De functie die deze stappen achtereenvolgens uitvoert voor de app is weergegeven in bijlage A.4. Het proces voor het berekenen van de interne krachten wordt weergegeven in onderstaande figuur.

Figuur 3.6: proces berekening M,V

3.2.3 Momenten- en dwarskrachtendiagram

Één van de doelstellingen van de app is om de momenten- en dwarskrachtdiagrammen visueel voor te stellen. Aangezien er binnen Xcode zelf geen tool bestaat voor het tekenen van diagrammen is er een externe tool geïmporteerd binnen het project. Deze externe tool, genaamd Charts, biedt de mogelijkheid om lijngrafieken op te stellen. De tool is geïmporteerd en geïmplementeerd aan de hand van een online tutorial [19].

Aangezien de krachten worden opgeslagen in een kolommatrix, is het eenvoudig om de grafiek te plotten. De x-waarde is steeds de positie van het knooppunt op de balk. Als y-waarde geeft de app de bijhorende waarde van het moment en de dwarskracht. Deze waarden zijn in chronologische volgorde opgeslagen in de bijhorende matrix. Het volstaat om de momenten en dwarskrachten chronologisch te plotten met een lijngrafiek.

3.3 Stalen profiel

Tot slot bepaalt de app aan de hand van de berekende momenten en dwarskrachten een optimale balk. De app biedt vier mogelijke stalen profielen en een betonnen balk met opgelegde breedte en betondekking. Paragrafen 3.3 en 3.4 bevatten de code achter de berekeningen van respectievelijk de stalen profielen en de balk uit beton.

3.3.1 Controles Eurocode 3

De app controleert de sterkte van de balk aan de hand van vier controles uit eurocode 3: de dwarskrachtweerstand, de weerstand tegen buiging, de kipweerstand en de combinatie van dwarskracht en buiging. In hoofdstuk 2 zijn de controles al uitvoerig besproken. In dit hoofdstuk wordt er dieper ingegaan op de implementatie van deze controles.

Het lijf van een stalen profiel biedt de weerstand tegen de optredende dwarskracht. De oppervlakte van het lijf is minimaal de dikte van het lijf vermenigvuldigd met de hoogte. Vervolgens berekent de app de dwarskrachtweerstand. Deze waarde moet groter zijn dan de optredende dwarskracht op de balk. De berekening is omgezet in een controle met onderstaande code.

```
153     //functie om de dwarskrachte te controleren
154     var Av = 0.0
155     var VplRd = 0.0
156     func controleDwarskracht(profiel: String, A: Double, b: Double, tf: Double, tw: Double, hw: Double, r: Double, fy: Double){
157         Av = A - 2.0*b*tf + (tw + 2.0*r)*tf
158         if Av < hw*tw{
159             Av = hw * tw
160         }
161         VplRd = Av*fy/(pow(3,0.5))
162         if profiel == "IPE"{
163             if VplRd >= V_controle{
164                 dwarskracht_controle_IPE = true
165             }
166         }
167         else if profiel == "HEA"{
168             if VplRd >= V_controle{
169                 dwarskracht_controle_HEA = true
170             }
171         }
172         else if profiel == "HEB"{
173             if VplRd >= V_controle{
174                 dwarskracht_controle_HEB = true
175             }
176         }
177         else if profiel == "HEM"{
178             if VplRd >= V_controle{
179                 dwarskracht_controle_HEM = true
180             }
181         }
182 
183     }
```

Code 3.9: dwarskrachtcontrole

Vervolgens controleert de software de weerstand tegen buiging. Voor deze berekening moet een onderscheid gemaakt worden tussen de profielen met een doorsnede van klasse één, twee en drie. Doorsnedes van klasse vier zijn niet van toepassing op de profielen geïntegreerd in de app. De controle is gekenmerkt door het weerstandbiedend moment van de doorsnede. De app gaat na of de weerstand tegen buiging groter is dan het maximaal moment op de balk. Voor elk profiel geldt volgende code.

```
118     //functie om het buigend moment te controleren
119     func controleBuigendMoment(profiel: String, Wel: Double, Wpl: Double, fy: Double, klasse: Int) {
120         if klasse <= 2 {
121             let Mcrd = Wpl*fy
122             if Mcrd >= M_controle, profiel == "IPE"{
123                 buigendMoment_controle_IPE = true
124             }
125             else if Mcrd >= M_controle, profiel == "HEA"{
126                 buigendMoment_controle_HEA = true
127             }
128             else if Mcrd >= M_controle, profiel == "HEB"{
129                 buigendMoment_controle_HEB = true
130             }
131             else if Mcrd >= M_controle, profiel == "HEM"{
132                 buigendMoment_controle_HEM = true
133             }
134         }
135         else if klasse == 3{
136             let Mcrd = Wel*fy
137             if Mcrd >= M_controle, profiel == "IPE"{
138                 buigendMoment_controle_IPE = true
139             }
140             else if Mcrd >= M_controle, profiel == "HEA"{
141                 buigendMoment_controle_HEA = true
142             }
143             else if Mcrd >= M_controle, profiel == "HEB"{
144                 buigendMoment_controle_HEB = true
145             }
146             else if Mcrd >= M_controle, profiel == "HEM"{
147                 buigendMoment_controle_HEM = true
148             }
149         }
150 
151     }
```

Code 3.10: controle buigend moment

Er wordt één stabiliteitscontrole uitgevoerd op de profielen. Deze controle gaat na of de kans op kip niet overschreden wordt. De kipweerstand is verschillend voor elke overspanning van de ingevoerde situatie. Hierdoor is het een relatief uitgebreide functie binnen de app. De code die de kipweerstand controleert is weergegeven in bijlage A.5.

Tot slot gaat de app na of er een controle moet worden uitgevoerd op de combinatie van dwarskracht en buiging. Indien blijkt dat het nodig is om deze combinatie te testen, is er een onderscheid gemaakt tussen profielen met een doorsnede van klasse één, twee en drie. Ter verduidelijking is de code voor een IPE-profiel weergegeven.

```
195             Av = A - 2.0*b*tf + (tw + 2.0*r)*tf
196             if Av < hw*tw{
197                 Av = hw * tw
198             }
199             VplRd = Av*fy/pow(3,0.5)
200         if profiel == "IPE"{
201             if VplRd*0.5 > V_controle{
202                 voorwaardeControleDwarskrachtEnBuiging_IPE = false
203             }
204             else{
205                 rho = pow(2*V_controle/VplRd-1.0,2)
206                 MvyRD = (Wply - rho*pow(hw*tw,2)/(4*tw))*fy
207                 if klasse <= 2 {
208                     let Mcrd = Wply*fy
209                     if Mcrd >= M_controle{
210                         dwarskrachtEnBuiging_controle_IPE = true
211                     }
212                 }
213                 else if klasse == 3{
214                     let Mcrd = Wely*fy
215                     if Mcrd >= M_controle{
216                     dwarskrachtEnBuiging_controle_IPE = true
217                     }
218                 }
219             }
220         }
```

Code 3.11: controle combinatie dwarskracht en buiging

3.3.1 Doorbuiging

Indien de profielen voldoen aan alle voorgaande controles, dan wordt de doorbuigingseis van L/500 nog geverifieerd. Bij het berekenen van de maximale snedekrachten gaat de app tevens na in welk veld de grootste doorbuiging optreedt in verhouding met de lengte van de overspanning. Het programma neemt steeds een zwaarder profiel uit de catalogus totdat de doorbuiging voldoet. Met behulp van de EEM bepaalt de app voor iedere balk de doorbuiging. Deze functie is in voorgaande paragrafen geïllustreerd. Het enige bijkomende deel van de code bestaat erin om een volgend profiel te kiezen indien de toegelaten verticale verplaatsing overschreden wordt.

3.3.2 Optimale balk

De noodzakelijke waarden uit de catalogi van de stalen profielen zijn geïntegreerd in de vorm van een chronologische lijst. De app kan met behulp van een teller per type profiel al de nodige gegevens opvragen. Indien een profiel voldoet geeft de teller aan over welk profiel het gaat. De waarden uit de catalogi zijn omgezet naar de juiste eenheden van de app (kN en m).

3.4 Betonnen balk

In tegenstelling tot de stalen profielen heeft de gebruiker bij het berekenen van een balk uit beton wel de mogelijkheid om zelf twee gegevens vast te leggen. De app maakt het mogelijk om de breedte van de balk en de betondekking in te geven. Aangezien het gewicht van een betonnen balk 25 kN/m^3 bedraagt, kan dit een significant effect hebben op de berekening van de balk. Daarom wordt dit gewicht ook meegenomen in de berekeningen.

3.4.1 Controles Eurocode 2

De controles uit Eurocode 2 worden uitgevoerd in GGT. De app berekent eerst het maximaal moment en dwarskracht met de veiligheidsfactoren voor GGT. Aangezien het eigengewicht een belangrijk aandeel heeft in de berekeningen, wordt er eerst een balk ontworpen aan de hand van de optimale hoogte. Vervolgens bepaalt de software de wapening op basis van deze optimale hoogte. Op deze wijze heeft de app een beginwaarde voor de variabele hoogte en kan het eigengewicht worden geïntegreerd. Het ontwerpen van een initiële balk gebeurt met de gegeven code.

```
116     func OntwerpBalk() {
117         let f_cd = f_ck*0.85/1.5
118         let f_yd = f_yk/1.15
119 
120         d_opt = 2.31*pow(M_controle*1.35/(b*f_cd),0.5)
121         A_s_onder = M_onder*1.35/(0.892*d_opt*f_yd)
122         A_s_plot_onder = A_s_onder
123         A_s_boven = M_boven*1.35/(0.892*d_opt*f_yd)
124         A_s_plot_boven = A_s_boven
125         h = d_opt + dekking
126     }
```

Code 3.12: ontwerp betonbalk

Met de berekende hoogte bepaalt de app de vereiste wapening, inclusief het eigengewicht van de balk. De volgende stap is de verificatie of de wapening ligt tussen de berekende minimum- en maximumwapening. De maximale wapening bedraagt vier procent van de doorsnede van de balk. Het bepalen van de minimale wapening is complexer. Hiervoor is onderstaande code geschreven.

```
426     func minimumWapening(){
427         f_ctm = 0.3*pow(f_ck,2/3)
428 
429         if h <= 300{
430             k = 1.0
431         }
432         else if h >= 800{
433             k = 0.65
434         }
435         else{
436             k = 1.21 - 0.7*h/1000
437         }
438 
439         //scheurbeheersing
440         A_smin_1 = 0.4*k*f_ctm*b*h*0.5/500
441 
442         //brosse breuk
443         A_smin_2 = 0.26*f_ctm*b*d_opt/f_yk
444         if A_smin_2 < 0.0013*b*d_opt{
445             A_smin_2 = 0.0013*b*d_opt
446         }
447 
448         if A_smin_1 >= A_smin_2{
449             A_smin = A_smin_1
450         }
451         else{
452             A_smin = A_smin_2
453         }
454     }
```

Code 3.13: minimumwapening

De volgende controle uit de Eurocode is de spanningscontrole. De app gaat na of de drukspanning in het beton en de trekspanning in het wapeningsstaal worden overschreden. Deze controle voert de app uit bij het maximale en minimale moment. Voor beton is gebruik gemaakt van een sterkteklasse C25/30 en voor het wapeningsstaal BE500. Onderstaande functie controleert of de toegelaten spanningen niet overschreden worden.

```
487     func SpanningControle(M: Double, As: Double){
488         let f_cm = f_ck + 8
489         E_cm = 22000*pow(f_cm/10,0.3)
490         alfa = E_wapening/E_cm
491         let x = -alfa*As/b + pow(pow(alfa*As/b,2)+2*alfa*As*d_opt/b,1/2)
492         let sigma_c = M*x/(b*pow(x,3)/3+alfa*As*pow(d_opt-x,2))
493         let sigma_s = alfa*sigma_c*(d_opt-x)/x
494 
495         print("sigma_c: \(sigma_c)")
496         print("sigma_s: \(sigma_s)")
497         //voorlopig alle controles in UGT-B
498         if sigma_c <= 0.85*f_ck/1.5, sigma_s <= f_yk*0.8{
499             spanningControle = true
500         }
501         else{
502             spanningControle = false
503         }
504         print("Controle spanning: \(spanningControle)")
505     }
```

Code 3.14: spaningscontrole

Naast de boven- en onderwapening in de balk moet er ook nog dwarskrachtwapening aanwezig zijn. Er moet altijd een minimum dwarskrachtwapening aanwezig zijn. Vervolgens wordt gecontroleerd of er nog bijkomende wapening vereist is. Tot slot kan ook de afstand tussen de beugels worden aangepast, zodat het mogelijk is om voldoende wapening tegen dwarskracht te steken.

De laatste controle uit de eurocode gaat na of de maximale scheurwijdte overschreden wordt. Vooraleer deze controle kan worden toegepast, is het noodzakelijk om de diameter van de wapeningsstaven te bepalen. In paragraaf 3.4.3 staat uitgelegd met welke functies de app de diameter bepaalt. Vanwege de omvang van de code voor de berekeningen van de dwarskrachtwapening en de controle van de scheurwijdte zijn deze te vinden in bijlage A.6 en A.7.

3.4.2 Optimale hoogte

De hoogte van de balk is de variabele in de app die ervoor zorgt dat er een optimale balk wordt bepaald. Wanneer een balk niet voldoet aan de eisen van de app, wordt de optimale hoogte d_{opt} aangepast met een waarde van vijf centimeter. De controles worden vervolgens opnieuw toegepast met de aangepaste hoogte. Op deze wijze zoekt de software een hoogte die voldoet aan alle voorwaarden. Wanneer minstens één van de controles niet voldoet of wanneer er geen geschikte wapening is gevonden met de gegevens, past de app de hoogte aan. Volgende functie zal controleren of de voorwaarden voldoen en vervolgens de optimale hoogte en wapening weergeven op het scherm van de iPhone.

```
    @IBAction func controleBeton(_ sender: Any) {
        //terug beginwaarde  instellen (vermijden dat hij een andere hoogte pakt als er 2de keer op knop wordt geduwd
        beginwaarden_controles()
        bepaalKrachtenGGT()
        bepaalMoment()
        OntwerpBalk()
        controleerBalk()
        while geschikte_wapening ==  false{
            if controleWapening == true, spanningControle == true, dwarskrachtControle == true{
                //optimale wapening -> diameter staven -> scheurwijdte nog controleren
                optimaleWapening()
                print("hoogte \(d_opt+dekking)")
                ScheurwijdteControle(As: A_s_onder, M: M_onder, diam: diam_onder)
                ScheurwijdteControle(As: A_s_boven, M: M_boven, diam: diam_boven)
                if scheurwijdteControle == true{
                    HoogteBalk.text = "\((h/10.0).roundTo(places: 1))"
                    wapening.text = "\(A_s_plot_onder.roundTo(places: 1))"
                    bovenwapening.text = "\(A_s_plot_boven.roundTo(places: 1))"
                    wapeningStaven.text = "\(aantalStaven_onder) diam \(diam_onder)"
                    wapeningStaven_boven.text = "\(aantalStaven_boven) diam \(diam_boven)"

                    maximaleDoorbuigingBeton()
                    doorbuiging.text = "\((umax_beton*pow(10,3)).roundTo(places: 1))"
                    if dwarskrachtWapeningNodig == true, dwarskrachtwapeningMogelijk == true{
                        dwarskrachtwapening.text = "\(A_sw.roundTo(places: 1))"
                        tussenafstandBeugels.text = "\((s/10).roundTo(places: 0))"
                        geschikte_wapening = true
                    }
                    else if dwarskrachtWapeningNodig == false{
                        dwarskrachtwapening.text = "\(A_sw_min.roundTo(places: 1))"
                        tussenafstandBeugels.text = "\((s/10).roundTo(places: 0))"
                        geschikte_wapening = true
                    }
                }
                else{
                    d_opt = d_opt +  50
                    h = d_opt + dekking
                    controleerBalk()
                }
            }
            else{
                //hoogte met 5cm verhogen tot juiste is gevonden
                d_opt = d_opt +  50
                h = d_opt + dekking
                controleerBalk()
            }
        }
    }
}
```

Code 3.15: optimale betonbalk

3.4.3 Wapening

Indien al de controles, uitgezonderd van de scheurwijdtecontrole, voldoen bepaalt de app de wapening die het minste staal vereist voor zowel de onder- als bovenwapening. Er is geopteerd om te werken met veel gebruikte diameters van wapeningsstaven: 8, 10, 12, 16, 20, 25, 32 en 40 millimeter. Voor alle diameters berekent de app het aantal staven dat er nodig is om de nodige wapeningsoppervlakte te bereiken. Vervolgens is er een functie ontwikkeld die de minimale oppervlakte aan wapening bepaalt, groter dan de berekende wapening. Voor de verduidelijking is de functie voor het omzetten van de wapening in een concreet aantal voor een diameter 40 millimeter weergegeven.

```
    func bepaalWapeningStaaf40(){
        while geschikteWapening40_onder == false{
            if aantal_40_onder*pow(40.0,2)*3.14/4 < A_s_onder{
                aantal_40_onder = aantal_40_onder + 1.0
            }
            else{
                A_s_40_onder = aantal_40_onder*pow(40.0,2)*3.14/4
                geschikteWapening40_onder = true
            }
        }
        while geschikteWapening40_boven == false{
            if aantal_40_boven*pow(40.0,2)*3.14/4 < A_s_boven{
                aantal_40_boven = aantal_40_boven + 1.0
            }
            else{
                A_s_40_boven = aantal_40_boven*pow(40.0,2)*3.14/4
                geschikteWapening40_boven = true
            }
        }
    }
```

Code 3.16: wapening omzetten in staven

3.4.4 Doorbuiging

Tot slot berekent de app de doorbuiging van de balk met de berekende gegevens. De EEM berekent de verplaatsingen aan de hand van de aangrijpende krachten in GGT, inclusief het eigengewicht van de balk. Het is echter noodzakelijk voor de methode om een waarde te kennen voor E, I en A. De waarde van oppervlakte A is eenvoudigweg de breedte vermenigvuldigd met

de hoogte van balk. Voor E en I zijn er bijkomende berekeningen vereist. De doorbuiging zal in gescheurde en ongescheurde toestand worden berekend. De uiteindelijke doorbuiging van de balk is een combinatie van deze twee toestanden. Paragraaf 2.3.2 verwijst naar de theorie achter de doorbuiging. Voor de implementatie van deze theorie in de app is onderstaande functie gehanteerd.

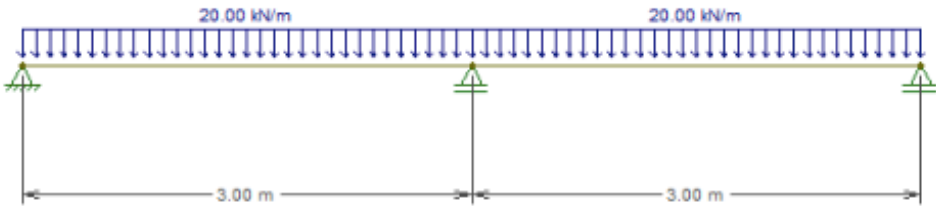
```
679     func maximaleDoorbuigingBeton(){
680         //E en I bepalen van ongescheurd beton
681         let Ecm = 22000*pow((f_ck+8)/10,0.3)
682         let Eceff = Ecm/(1+1.7)
683         let alfa = E_wapening/Eceff
684         let x_1 = (alfa*(A_s_onder*d_opt)+b*h*h*0.5)/(alfa*A_s_onder+b*h)
685         let I_1 = b*pow(x_1,3)/3+b/3*pow(h-x_1,3)+alfa*A_s_onder*pow(d_opt-x_1,2)
686 
687         berekenDoorbuiging(E: Eceff*pow(10,3), I: I_1*pow(10,-12))
688         //maximale doorbuiging bij ongescheurd beton
689         for i in 0...aantal_elementen{
690             if -verplaatsingen_GGT[i*3+1,0] >= umax_beton_ongescheurd{
691                 umax_beton_ongescheurd = -verplaatsingen_GGT[i*3+1,0]
692             }
693         }
694 
695         //E en I bepalen van gescheurd beton
696         let x_2 = -alfa/b*A_s_onder + pow(pow(alfa/b*A_s_onder,2)+2*alfa/b*A_s_onder*d_opt,0.5)
697         let I_2 = b*pow(x_2,3)/3+alfa*A_s_onder*pow(d_opt-x_2,2)
698         berekenDoorbuiging(E: Eceff*pow(10,3), I: I_2*pow(10,-12))
699         //maximale doorbuiging bij gescheurd beton
700         for i in 0...aantal_elementen{
701             if -verplaatsingen_GGT[i*3+1,0] >= umax_beton_gescheurd{
702                 umax_beton_gescheurd = -verplaatsingen_GGT[i*3+1,0]
703             }
704         }
705         //scheurmoment berekenen
706         let fctm = 0.3*pow(f_ck,2/3)
707         let Mr = fctm*pow(10,-6)*I_1/(h-x_1)
708         let M = M_onder*pow(10,-6)
709         let zeta = 1 - 0.5*pow(Mr/M,2)
710         //de uiteindelijke doorbuiging van de balk wordt samengesteld
711         umax_beton = zeta*umax_beton_gescheurd + (1-zeta)*umax_beton_ongescheurd
712     }
```

Code 3.17: doorbuiging betonbalk

4 Toepassing en validatie app

4.1 Toegepast voorbeeld

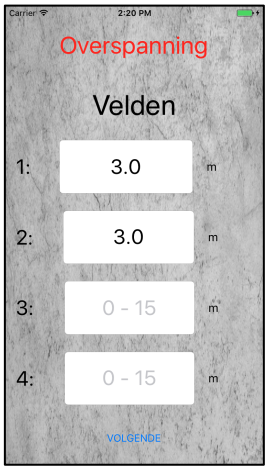
Deze paragraaf toont aan hoe de app wordt toegepast door een gebruiker. De toepassing van de app wordt aan de hand van een realistisch voorbeeld toegelicht. Het voorbeeld is ingegeven volgens onderstaande afbeelding.



Figuur 4.1: ingevoerde opstelling

4.1.1 Velden

De eerste handeling van de gebruiker is het ingeven van de overspanning. Zoals aangetoond in figuur 4.1 zijn er twee velden met een overspanning van drie meter. De gebruiker geeft enkel een lengte in bij twee velden, de andere velden blijven leeg. Wanneer een persoon een overspanning groter dan vijftien meter ingeeft, past de app deze waarde automatisch aan naar de maximale waarde van vijftien meter. In het andere uiterste geval, een waarde kleiner dan nul, gaat de overspanning als nul meter worden aangenomen. Onderstaand scherm geeft aan hoe de velden worden ingegeven.



Figuur 4.2: overspanningen ingeven

4.1.2 Belastingen

Op het moment dat de gebruiker verder gaat naar het volgende scherm, wordt de lengte van de lijnlast onmiddellijk vastgelegd op de lengte van balk. Deze instelling van de app is gekozen omdat dit het meest voorkomende geval is in hedendaagse toepassingen. De overspanningen worden getoond met behulp van een eenvoudige schets met bijhorende afstanden. De app maakt het mogelijk om de grootte, de veiligheidsfactor en de positie van vier punt- en lijnlasten in te geven. De belasting op de balk in het voorbeeld is een lijnlast van 20,0 kN/m. De veiligheid in UGT kan aangepast worden voor een permanente of mobiele last. Volgende figuur illustreert de

werking van het tweede scherm van de app. Het toetsenbord is mee afgebeeld om aan te tonen dat er enkel getallen kunnen worden ingevoerd. Deze instelling zorgt ervoor dat de app geen foutmelding geeft wanneer de gebruiker letters of leestekens invoert.

Figuur 4.3: belastingen ingeven

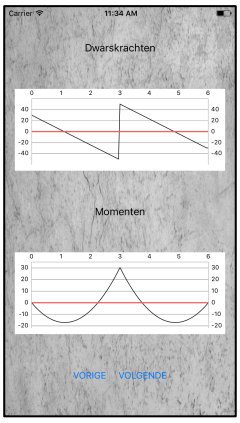
4.1.3 Snedekrachten

Eerst berekent de app de momenten en dwarskrachten die optreden op de balk. Deze resultaten worden weergegeven met behulp van twee schermen. Het eerste scherm berekent de maximale en minimale waarden van deze interne krachten. Het tweede scherm toont de diagrammen met behulp van twee lijngrafieken.

Om de minimale en maximale waarden te berekenen drukt de gebruiker op de drukknop 'bereken M,V'. Initieel was er geopteerd om de berekeningen te laten uitvoeren wanneer er bij het tweede scherm op de drukknop 'Volgende' wordt geduwd. Dit is echter niet praktisch. Wanneer de resultaten dan al berekend zijn en de gebruiker de verschillende schermen achteraf nog wil herbekijken, dan zou de app een te grote vertraging hebben. Onderstaand scherm toont aan hoe de app de berekende waarde weergeeft wanneer er op de knop is geduwd.

Figuur 4.4: maximale en minimale snedekrachten

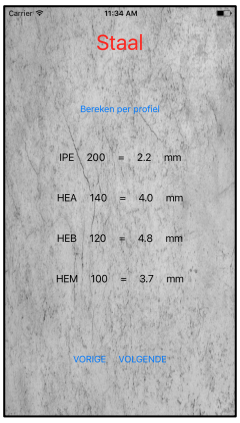
De snelheid waarmee de app de diagrammen grafisch voorstelt is beduidend sneller dan de snelheid waarmee de snedekrachten worden berekend. De functie om de grafieken op te stellen zit in dit geval wel achter de drukknop 'Volgende' van het vorige scherm. Indien de gebruiker op deze knop van het scherm uit figuur 4.4 duwt, toont de app onderstaand scherm. Dit is uiteraard ook het geval wanneer de gebruiker van het vijfde terugkeert naar het vierde scherm.



Figuur 4.5: momenten- en dwarskrachtendiagram

4.1.4 Stalen profiel

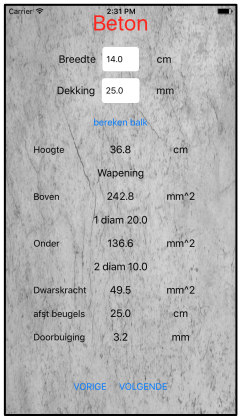
Om dezelfde reden als bij de berekening van de snedekrachten, is er voor de berekening van de optimale stalen profielen een drukknop 'bereken per profiel'. Op het ogenblik dat de gebruiker op de knop duwt gaat de app na welke profielen voldoen aan de vooropgestelde eisen. Voor elk type profiel geeft het scherm aan welk profiel optimaal is, aangevuld met de bijhorende doorbuiging. In het geval van het voorbeeld staan onderstaande resultaten weergegeven op het scherm van de app.



Figuur 4.6: optimale stalen profielen

4.1.5 Betonnen balk

De laatste mogelijkheid die de app biedt is het berekenen van een betonnen balk met optimale hoogte. Een realistische situatie is een balk met een breedte van 14,0 centimeter en betondekking van 25,0 millimeter. Met deze gegevens berekent de software de optimale balk. De belangrijkste uitkomsten met betrekking tot het toepassingsgebied van de app zijn weergegeven aan de gebruiker zoals getoond op volgende figuur.



Figuur 4.7: optimale betonnen balk

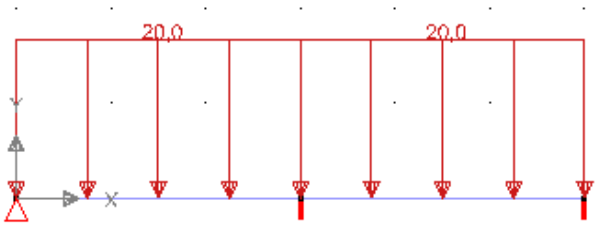
Tenslotte is er nog één laatste scherm gecreëerd voor de app. Dit scherm heeft maar één doel: de gegevens resetten voor een nieuwe berekening en een verbinding maken naar het startscherm van de app.

4.2 Resultaten

Om de betrouwbaarheid van de app te testen zijn de resultaten van vier verschillende opstellingen gecontroleerd met bestaande software. Deze controle werd uitgevoerd met Diamonds, een programma van Buildsoft. Diamonds maakt gebruik van dezelfde controles uit de Eurocodes als de app. De resultaten van beide berekeningen zijn weergegeven in tabellen. Verder is voor elke berekende waarde de procentuele afwijking tussen beide programma's bijgevoegd.

4.2.1 Opstelling 1

Het eerste voorbeeld is de situatie zoals besproken in het begin van dit hoofdstuk. Twee overspanningen van drie meter worden belast met een lijnlast van 20,0 kN/m. De situatie is visueel voorgesteld in onderstaande figuur.



Figuur 4.8: opstelling 1

De waarden van de snedekrachten uit de berekeningen met beide programma's zijn vergeleken met behulp van onderstaande tabel. In de rechterkolom is de procentuele afwijking tussen beide programma's berekend.

Tabel 4.1: snedekrachten opstelling 1

Snedekrachten				
	Diamonds	App		% afwijking
Mmax	17,1	17,1	kNm	0
Mmin	-30,4	-30,4	kNm	0
Vmax	50,6	50,6	kN	0
Vmin	-50,6	-50,6	kN	0

De resultaten uit tabel 4.1 tonen aan dat de snedekrachten exact overeenkomen bij beide programma's. De minimale en maximale dwarskracht treden op ter hoogte van het tweede steunpunt.

Zowel de app als Diamonds voert een optimalisatie van het stalen profiel uit die voldoet aan de eisen uit de Eurocode. Het verschil tussen de twee zit in het feit dat de app automatisch een profiel geeft waarvoor de doorbuiging voldoet aan de eis L/500. Diamonds bepaalt een optimaal profiel aan de hand van de snedekrachten. Vervolgens moet de gebruiker zelf nog controleren of de doorbuiging van een gekozen profiel voldoet. Volgende tabel toont aan welke profielen berekend zijn, elk met bijhorende doorbuiging.

Tabel 4.2: profielen opstelling 1

Staal						
	Diamonds	App	Diamonds	App		% afwijking
IPE	180	200	3,1	2,2	mm	/
HEA	140	140	4,0	4,0	mm	0
HEB	120	120	4,8	4,8	mm	0
HEM	100	100	3,6	3,7	mm	2,78

Uitgezonderd van de keuze van het optimaal IPE-profiel komen de berekeningen met beide programma's goed overeen. Het verschil in de keuze voor een IPE-profiel zit in de controle van de kipweerstand. Voor deze controle maakt de app gebruik van een veilige benadering van het kritisch kipmoment, zoals uitgelegd in de paragraaf over kipweerstand. Diamonds berekent dat het kritisch kipmoment bij een IPE 180 92% bedraagt van de maximaal toegelaten waarde. De app heeft aangegeven dat de kipweerstand niet voldoet en opteert vervolgens voor een optimaal profiel IPE 200. Deze afwijking is dus te wijten aan de veilige benadering in de app. De kleine afwijking van de doorbuiging bij de HEM 100 is te wijten aan een verschillende afronding van de berekende waarden. De doorbuiging van de IPE-profielen kan niet vergeleken worden omdat de doorbuiging berekend wordt voor een verschillend profiel.

Vervolgens werden de resultaten van de betonbalk gevalideerd. In Diamonds dient er eerst een betonbalk met een bepaald breedte, hoogte en betondekking ingegeven te worden vooraleer de benodigde wapening bepaald kan worden. Deze hoogte en breedte worden eerst bepaald met behulp van de app. Er zal dus bij geen enkele opstelling een afwijking zijn bij de breedte, de hoogte of de dekking. De resultaten van de betonbalk voor bovenstaande opstelling zijn te zien in onderstaande tabel.

Tabel 4.3: betonbalk opstelling 1

Beton				
	Diamonds	App		% afwijking
Veldwapening	128	136,6 2ϕ10	mm²	6,72
Steunpuntswapening	233	242,8 1ϕ20	mm^2	4,21
dwarskrachtwapening	241	247,5	mm²/m	2,70
doorbuiging	2,2	3,2	mm	45,45

Uit tabel 4.3 blijkt dat er relatief kleine afwijkingen zijn tussen de resultaten verkregen met Diamonds en deze van de app. Voor al deze kleine afwijkingen zijn er echter logische verklaringen. De app berekent de veldwapening met volgende formule:

$$A_s = \frac{M_d}{0{,}892 * d * f_{yd}} \tag{4.1}$$

In deze formule wordt er een vaste waarde voor ζ gebruikt, namelijk 0,892. Deze waarde is echter enkel geldig wanneer er met een optimale nuttige hoogte gewerkt wordt. In de app wordt deze hoogte echter aangepast totdat er een geschikte balk gevonden is. In de werkelijkheid zal de nuttige hoogte groter zijn dan de optimale nuttige hoogte. Als gevolg zal de waarde voor ζ ook groter zijn. Bijgevolg zal de wapeningshoeveelheid, verkregen met de app, groter zijn dan deze verkregen met Diamonds. De app berekent de wapening dus op een licht conservatieve manier. Deze kleine overschatting is dus een veilige benadering van de benodigde wapeningshoeveelheid.

De afwijking bij de dwarskrachtwapening is op een gelijkaardige manier te verklaren. De app bepaalt de benodigde hoeveelheid dwarskrachtwapening met onderstaande formule:

$$A_{sw} = \frac{V_{Ed} * s}{f_{ywk}/1{,}15 * z * \cot\theta} \tag{4.2}$$

In deze formule is θ de hoek die de drukdiagonaal maakt met de as van de ligger. De app hanteert een gemiddelde waarde voor $\cot\theta$ (namelijk $\cot\theta = 2$), terwijl Diamonds deze hoek exact zal bepalen. Bijgevolg is er dus een lichte afwijking tussen de waarden uit de app en deze verkregen met Diamonds.

De doorbuiging, verkregen met de app, heeft een procentuele afwijking van 45,45% met het resultaat verkregen met de software. Deze afwijking is te wijten aan de manier waarop de app de doorbuiging zal uitrekenen. Vooraleer de app de doorbuiging kan bepalen, moet de effectieve elasticiteitsmodulus van het beton gekend zijn. Deze wordt bepaald met behulp van een kruipfactor. Zoals reeds uitgelegd in de paragraaf over de doorbuiging van de betonbalk, hanteert de app een vaste waarde van 1,7 voor de kruipfactor. Diamonds daarentegen bepaalt voor elke situatie afzonderlijk de kruipfactor. Dit zorgt voor een verschil in effectieve elasticiteitmodus tussen de app en de software. Dit leidt uiteindelijk tot een kleine afwijking in de berekende doorbuiging. De zeer grote afwijking bij deze opstelling is te wijten aan het feit dat er hier slechts een zeer kleine doorbuiging optreedt. Het verschil bedraagt slecht één millimeter. Door het feit dat er slechts een doorbuiging van 2,2 millimeter is, is de procentuele afwijking relatief groot. In de volgende voorbeelden wordt aangetoond dat bij grotere doorbuigingen de procentuele afwijking sterk afneemt.

4.2.2 Opstelling 2

Het vorige voorbeeld is een situatie met beperkte overspanningen en een gelijkmatig verdeelde last. De volgende situatie die gecontroleerd wordt is een situatie met twee verschillende overspanningen, namelijk 3,5 en 6,0 meter. Verder grijpt er een verschillende belasting aan op het linker- en rechterveld. Het linkerveld wordt belast met een lijnlast van 15,0 kN/m over de volledige overspanning, terwijl op het rechterveld een puntlast van 20,0 kN aangrijpt in het midden van het veld. Onderstaande figuur schetst de situatie van opstelling twee.

Figuur 4.9: opstelling 2

Zoals bij elke opstelling is de eerste stap het controleren van de berekende snedekrachten. Volgende tabel geeft de vergelijking van de uiterste waarden van de momenten en dwarskrachten.

Tabel 4.4: snedekrachten opstelling 2

Snedekrachten				
	Diamonds	App		% afwijking
Mmax	25,2	25,2	kNm	0
Mmin	-30,6	-30,6	kNm	0
Vmax	26,7	26,7	kN	0
Vmin	-44,2	-44,2	kN	0

De resultaten uit tabel 4.4 bevestigen dat ook voor asymmetrische situaties de snedekrachten nauwkeurig worden berekend. Zowel de momenten als dwarskrachten komen exact overeen bij beide berekeningen.

De resultaten van de optimale stalen profielen zijn vergeleken aan de hand van onderstaande tabel.

Tabel 4.5: profielen opstelling 2

Staal						
	Diamonds	App	Diamonds	App		% afwijking
IPE	220	240	6,7	4,9	mm	/
HEA	160	160	11,1	11,3	mm	1,80
HEB	160	160	7,4	7,6	mm	2,70
HEM	120	120	9,2	9,4	mm	2,17

De keuze voor een zwaarder IPE-profiel door de app is opnieuw te wijten aan de berekening van de kipweerstand. Zoals tabel 4.5 aantoont komen de andere HE-profielen overeen. Er is echter wel een kleine afwijking op de doorbuiging. Deze afwijking is te verklaren door het feit dat Diamonds een geïsoleerde balk standaard opdeelt in acht elementen. In dit voorbeeld komt dat neer op een breedte van $6{,}0m/8 = 0{,}75m$. De nauwkeurigheid van de app is standaard ingesteld op vijf centimeter. De app geeft aan dat de maximale doorbuiging zich bevindt op 3,25 meter van het middelste steunpunt. De berekende doorbuiging door de app is dus nauwkeuriger dan de berekening uit Diamonds. Als de instellingen in Diamonds aangepast worden zodat de balk opgedeeld wordt in 100 elementen, dan geeft Diamonds dezelfde doorbuiging als de app.

In onderstaande tabel worden de specificaties van de betonbalk weergegeven. De app bepaalt voor de veld- en steunpuntswapening ook de benodigde staafdiameters en hun aantal.

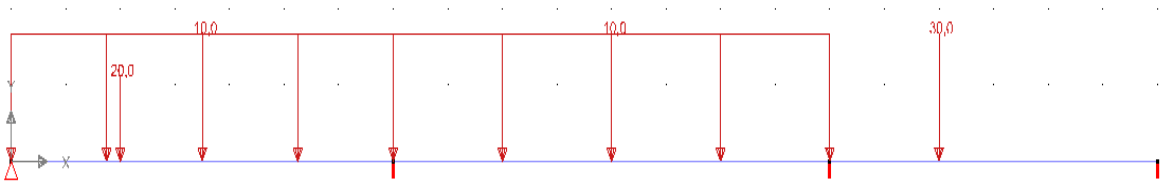
Tabel 4.6: betonbalk opstelling 2

Beton				
	Diamonds	App		% afwijking
Veldwapening	247	259 1ϕ20	mm²	4,86
Steunpuntswapening	304	314,9 3ϕ12	mm^2	3,59
dwarskrachtwapening	220	253	mm²/m	15,0
doorbuiging	13,4	13,9	mm	3,73

Bij deze opstelling valt opnieuw op dat er kleine afwijkingen zijn bij de berekende wapeningshoeveelheden en bij de doorbuiging. De dwarskrachtwapening vertoont een grote procentuele afwijking, namelijk 15%. Er is dus een grotere afwijking tussen de hoek van drukdiagonaal θ in de app en in Diamonds. Bij deze opstelling is er kleinere procentuele afwijking op de doorbuiging dan bij de vorige opstelling. Bij deze opstelling treedt er namelijk een veel grotere doorbuiging op. De kleine afwijking in de doorbuiging zorgt in dit geval dus voor een veel kleinere procentuele afwijking.

4.2.3 Opstelling 3

Aangezien er reeds is aangetoond dat de resultaten voor kleinere overspanningen identiek overeenkomen met beide programma's, is er in het derde voorbeeld geopteerd om een situatie na te bootsen met relatief grote overspanningen. De opstelling bestaat uit drie velden van respectievelijk zeven, acht en zes meter. Er wordt een lijnlast van 10,0 kN/m aangebracht op de twee linkse velden. Verder grijpt er een puntlast van 20,0 kN aan op twee meter van het uiterst linkse steunpunt en een puntlast van 30,0 kN op vier meter van het rechtersteunpunt. Onderstaande figuur geeft deze situatie weer.



Figuur 4.10: opstelling 3

Volgende tabel is een vergelijking van de snedekrachten berekend door Diamonds en de app.

Tabel 4.7: snedekrachten opstelling 3

Snedekrachten				
	Diamonds	App		% afwijking
Mmax	79,4	79,4	kNm	0
Mmin	-93,3	-93,3	kNm	0
Vmax	58,9	58,9	kN	0
Vmin	-68,3	-68,3	kN	0

Tabel 4.7 geeft aan dat bij grotere overspanningen de snedekrachten ook tot op één decimaal nauwkeurig worden berekend door de app.

Het bepalen van een optimaal profiel voor elk type is vergeleken met beide programma's. De resultaten worden gecontroleerd aan de hand van onderstaande tabel.

Tabel 4.8: profielen opstelling 3

Staal						
	Diamonds	App	Diamonds	App		% afwijking
IPE	330	400	8,8	4,5	mm	/
HEA	240	240	13,4	13,4	mm	0
HEB	220	220	12,8	12,8	mm	0
HEM	180	180	13,9	13,9	mm	0

De resultaten bevestigen het feit dat bij een IPE-profiel de kipweerstand de doorslaggevende factor is. Verder berekenen zowel de app als Diamonds dezelfde optimale HE-profielen . In tegenstelling tot het derde voorbeeld berekent de app de doorbuiging van de stalen HE-profielen tot op 0,1 millimeter nauwkeurig.

Vervolgens bepaalt de app de optimale betonbalk. De wapeningshoeveelheden en doorbuiging van deze betonbalk zijn uiteindelijk met behulp van Diamonds geverifieerd. De resultaten worden vergeleken in volgende tabel.

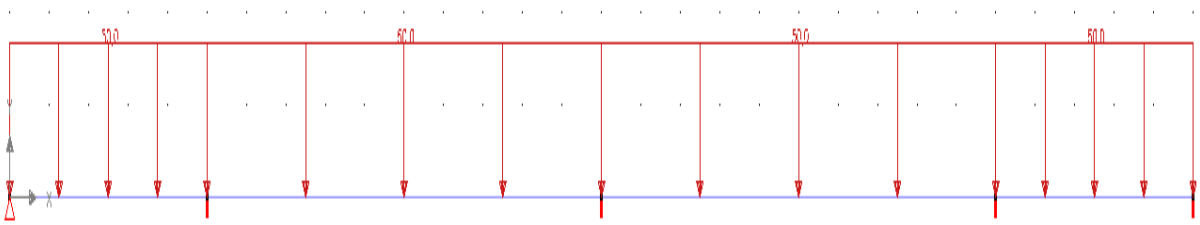
Tabel 4.9: betonbalk opstelling 3

Beton				
	Diamonds	App		% afwijking
Veldwapening	458	484,3 1ϕ25	mm²	5,74
Steunpuntswapening	586	612,2 2ϕ20	mm^2	4,47
dwarskrachtwapening	268	277	mm²/m	3,36
doorbuiging	14,5	15,5	mm	6,70

Er valt opnieuw op dat er kleine afwijkingen zijn bij de wapeningshoeveelheden en de doorbuiging. Deze verschillen zijn reeds verklaard bij de voorgaande opstellingen.

4.2.4 Opstelling 4

Als laatste voorbeeld is er geopteerd om te werken met het maximaal aantal overspanningen, namelijk vier. De buitenste velden hebben een lengte van vijf meter, terwijl de binnenste velden tien meter lang zijn. In deze laatste opstelling is een permanente last van 50,0 kN/m over de volledige lengte van de balk aangebracht. Onderstaande figuur is een grafische voorstelling van de situatie.



Figuur 4.11: opstelling 4

Volgende tabel toont de snedekrachten uit de app en uit Diamonds.

Tabel 4.10: snedekrachten opstelling 4

Snedekrachten				
	Diamonds	App		% afwijking
Mmax	316,4	319,7	kNm	1,04
Mmin	-632,8	-632,8	kNm	0
Vmax	358,6	358,6	kN	0
Vmin	-358,6	-358,6	kN	0

Uit de resultaten van tabel 4.10 blijkt dat er voor de berekening van het maximaal moment op de balk een afwijking van ongeveer één procent optreedt. De overige uiterste snedekrachten zijn wel nauwkeurig berekend door de app. Deze afwijking is te wijten aan het feit dat het maximaal moment niet exact in het midden van de overspanning ligt, maar op een afstand van 4,7 meter tot het middelste steunpunt. De app heeft een nauwkeurigheid van 0,05m, waardoor deze waarde nauwkeuriger wordt berekend dan in Diamonds. Zoals eerder vermeld deelt Diamonds een geïsoleerde balk namelijk op in slechts 8 elementen. Daarom is de app nauwkeuriger.

De resultaten van de optimale stalen profielen zijn vergeleken in onderstaande tabel.

Tabel 4.11: profielen opstelling 4

Staal						
	Diamonds	App	Diamonds	App		% afwijking
IPE	600	/	8,4	/	mm	/
HEA	450	550	12,2	7,0	mm	/
HEB	400	450	13,4	9,8	mm	/
HEM	280	300	19,6	13,2	mm	/

De resultaten van het IPE-profiel zijn in het rood gemarkeerd bij de berekening in Diamonds. Dit wil zeggen dat dit het grootste profiel is, maar dat deze nog steeds niet voldoet aan alle eisen. De app geeft in dat geval aan dat er geen IPE-profiel geschikt is door middel van twee schuine strepen. Verder blijkt dat bij grotere belastingen en grotere overspanningen de kipweerstand ook een invloed op de keuze van een HE-profiel. Voor elk HE-profiel geeft de app een groter profiel aan dan Diamonds. De keuze van de app voor een groter HE-profiel is bij deze opstelling dus te wijten aan de conservatieve benadering van de kipweerstand in de app. De vergelijking tussen de vier verschillende opstellingen stelt dat de invloed van de kipweerstand groter wordt naarmate de overspanningen groter worden.

Uiteindelijk werd ook deze betonbalk uit de app vergeleken met Diamonds. De resultaten zijn weergegeven in volgende tabel.

Tabel 4.12: betonbalk opstelling 4

Beton				
	Diamonds	App		% afwijking
Veldwapening	769	837,8 3ϕ20	mm²	8,95
Steunpuntswapening	1590	1658,3 4ϕ25	mm^2	4,30
dwarskrachtwapening	635	579,5	mm²/m	8,74
doorbuiging	7,5	9,0	mm	20,0

Ook bij deze opstelling zijn er kleine afwijking en vast te stellen. In tegenstelling tot alle andere opstellingen, berekent de app bij deze opstelling een kleinere hoeveelheid dwarskrachtwapening. De afwijking op de doorbuiging is in deze opstelling ook significanter dan bij de andere opstellingen, omdat er een kleinere totale doorbuiging is. Bij grote overspanningen en grote belastingen vertonen de resultaten uit de app grotere afwijkingen dan bij de andere opstellingen. De afwijking is echter nog aanvaardbaar omdat deze app slechts een tool is voor op de werf, het is geen gedetailleerde berekening van een balk.

5 Besluit

Het hoofddoel van de masterproef is een app ontwikkelen die hyperstatische balken uitrekent en vervolgens een optimale balk berekent. Een optimale balk kan bestaan uit een stalen profiel van het type IPE, HEA, HEB en HEM of een betonnen balk met ingevoerde breedte en betondekking.

De app rekent met behulp van de eindige-elementenmethode de snedekrachten in de balk uit. Deze methode maakt gebruik van een globale matrixvergelijking. De app rekent een homogene balk uit, waardoor het opstellen van deze vergelijking relatief eenvoudig is voor de software. Een grote beperking van de software is dat de breedte van één element beperkt blijft tot vijf centimeter. Indien een kleinere breedte wordt gebruikt van bijvoorbeeld één centimeter, heeft de EEM vijf keer zoveel elementen nodig. Dit heeft als gevolg dat de app matrices moet opstellen met vijftien keer de dimensie van de huidige matrices. De software is niet optimaal geschikt voor het berekenen van dit soort zware wiskundige berekeningen, waardoor dergelijke vergroting leidt tot een significante vertraging van de app. Daarom hanteert de app een elementbreedte en dus een nauwkeurigheid van vijf centimeter.

Vervolgens berekent de app een geschikt stalen profiel aan de hand van controles uit Eurocode 3. Dit profiel moet nog voldoen aan de doorbuigingseis: L/500. Indien er een profiel is gevonden die voldoet aan alle eisen, dan wordt dit profiel weergegeven als optimaal profiel. De app heeft een veilige benadering gehanteerd omtrent de kipweerstand van het profiel. Hierdoor bepaalt de app vaak een zwaarder IPE-profiel dan Diamonds. Wanneer kip een dreiging vormt, wordt er slechts zelden een IPE-profiel gebruikt in de praktijk, waardoor dit verschil niet relevant is voor de keuze van een optimaal profiel. Bij grotere overspanningen heeft de kipweerstand een grotere invloed dan bij kleinere overspanningen.

Uiteindelijk bepaalt de app ook de optimale betonbalk. De benodigde hoeveelheid langswapening en dwarskrachtwapening wordt bepaald. Vervolgens controleert de app de eisen uit de gebruiksgrenstoestand. Indien de balk niet voldoet aan deze eisen, wordt de hoogte van de balk gecorrigeerd. De resultaten uit de app vertonen een kleine afwijking met de resultaten uit bestaande software. Deze afwijkingen zijn allemaal te wijten aan het feit dat de app een vaste waarde toekent aan enkele variabelen. De afwijkingen blijven echter beperkt, waardoor de resultaten uit de app aanvaardbaar zijn. De app is namelijk slechts een tool, het is geen gedetailleerde berekening van een balk.

De mogelijkheid van uitbreiding van deze masterproef ligt vooral bij de software. Indien er een manier bestaat om de app de berekeningen sneller te laten uitvoeren, kan de nauwkeurigheid ook verhoogd worden met het gebruik van een kleinere breedte van een element. Vervolgens is het mogelijk om dieper in te gaan op de kipweerstand van een stalen profiel. Binnen deze masterproef zijn er verschillende veilige aannames gemaakt met betrekking tot het kipgevaar. Ook bij de dimensionering van de betonbalk zijn er enkele aannames gemaakt. Deze waardes kunnen nog verbeterd worden. Een andere mogelijke uitbreiding van de app is het detailleren van de wapening. Momenteel controleert de app enkel de staafafstanden, maar dit kan uitgebreid worden naar overlappinglassen en dergelijke. Op deze manier kan de app een grondige wapeningsdetaillering uitvoeren. Vervolgens kan de app dan een scherm tonen met de doorsnede van de betonbalk met daarop de diameters, de tussenafstand, de dekking, de breedte en de hoogte van balk. Dit zou een meerwaarde zijn voor de app. Tot slot kan de app uitgebreid worden door de implementatie van voorspanning in de betonbalk. De app kan dan het traject

van de voorspanning in de balk bepalen. Verder kan de app dan ook het aantal voorspanstrengen, de diameter van deze strengen en de voorspankracht per streng bepalen.

Literatuurlijst

[1] W. Van Paepegem, *Mechanica van materialen*, 1ste ed. Gent: Universiteit Gent, 2013.

[2] J. H. P. De vree, *Eindige Elementen Methode*. TU Eindhoven, 2001.

[3] Apple, "Xcode 8," 2017. [Online]. Beschikbaar: https://developer.apple.com/xcode/. [Geraadpleegd: 28-apr-2017].

[4] Buildsoft, "Diamonds." [Online]. Beschikbaar: http://www.buildsoft.eu/nl/product/diamonds. [Geraadpleegd: 28-apr-2017].

[5] G. Lombaert and G. De Roeck, "Hoofdstuk 3: 2D raamwerken," in *Eindige elementen, deel 1: staafconstructis*, KU Leuven, 2001.

[6] G. Lombaert en G. De Roeck, "Hoofdstuk 2: 2D vakwerken," in *Eindige elementen, deel 1: staafconstructis*, KU Leuven, 2001.

[7] D. J. Dean, "The Direct Stiffness method and the Global StiffnessMatrix," in *Introduction to the Finite Element Method (FEM)*, Cambridge: University of Cambridge, p. 4,5.

[8] Ing. R. Steensels, *persoonlijke communicatie*. maart 2017.

[9] *Eurocode 3: Ontwerp en berekening van staalconstructies - deel 1.1: Algemene regels en regels voor gebouwen (Geconsolideerde versie, inclusief AC: 2009)*, 1st ed. Brussel: CEN, 2015.

[10] L. Simoes da Silva, R. Simoes, and H. Gervasio, *Design of Steel structures*, 1st ed. Berlin: Ernst and Sohn, 2010.

[11] *Eurocode 2: Ontwerp en berekening van betonconstructies - Deel 1.1: algemene regels en regels voor gebouwen (+ AC: 2010)*, 1st ed. Brussel: CEN, 2005.

[12] ArcelorMittal, "products and services." [Online]. Beschikbaar: http://sections.arcelormittal.com/products-services/products-ranges.html. [Geraadpleegd: 10-apr-2017].

[13] L. Vanhooymissen, M. Spegelaere, A. Van Gysel, and W. De Vylder, *Gewapend beton*. Gent: Academia press, 2002.

[14] Apple, "Xcode 8." Apple, 2017.

[15] Apple, "UIKit," 2017. [Online]. Beschikbaar: https://developer.apple.com/reference/uikit#overview. [Geraadpleegd: 02-mei-2017].

[16] R. Shankar, "Different ways to connect IBAction to UIButton," 2012. [Online]. Beschibaar: http://rshankar.com/different-ways-to-connect-ibaction-to-uibutton/. [Geraadpleegd: 10-maa-2017].

[17] Apple, "Subscript," 2017. [Online]. Beschikbaar: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/Subscripts.html. [Geraadpleegd: 29-maa-2017].

[18] S. Canon, "Matrix inversion in Swift using Accelerate Framework," 2014. [Online]. Beschikbaar: http://stackoverflow.com/questions/26811843/matrix-inversion-in-swift-using-accelerate-framework. [Geraadpleegd: 01-apr-2017].

[19] J. Echessa, “How to Use iOS Charts API to Create Beautiful Charts in Swift,” 2015. [Online]. Beschikbaar: http://www.appcoda.com/ios-charts-api-tutorial/. [Geraadpleegd: 08-apr-2017].

Bijlagen

BIJLAGEN **53**
BIJLAGE A: BELANGRIJKE CODE 55
Bijlage A.1: Belastingen toevoegen *55*
Bijlage A.2: Randvoorwaarden *56*
Bijlage A.3: Verplaatsingen berekenen *57*
Bijlage A.4: Berekenen momenten en dwarskrachten *58*
Bijlage A.5: Kipweerstand *59*
Bijlage A.6: Berekenen dwarskrachtwapening *60*
Bijlage A.7: Scheurwijdtecontrole *61*

Bijlage A: Belangrijke code

Bijlage A.1: Belastingen toevoegen

De toegevoegde code is voor de functie die krachten uiteindelijk met de juiste veiligheid, grootte en afstand zal toevoegen aan de bijhorende krachtenmatrix.

```
391     func setKrachten(){
392         //puntlasten toevoegen
393         if positie_puntlast_1 > scharnier_5 {
394             print("puntlast1 buiten bereik")
395         }
396         else if positie_puntlast_2 > scharnier_5 {
397             print("puntlast2 buiten bereik")
398         }
399         else if positie_puntlast_3 > scharnier_5 {
400             print("puntlast3 buiten bereik")
401         }
402         else if positie_puntlast_4 > scharnier_5 {
403             print("puntlast4 buiten bereik")
404         }
405         else {
406             voegPuntlastToe_GGT(Grootte: grootte_puntlast_1, Positie: positie_puntlast_1)
407             voegPuntlastToe_GGT(Grootte: grootte_puntlast_2, Positie: positie_puntlast_2)
408             voegPuntlastToe_GGT(Grootte: grootte_puntlast_3, Positie: positie_puntlast_3)
409             voegPuntlastToe_GGT(Grootte: grootte_puntlast_4, Positie: positie_puntlast_4)
410             if VeiligheidPuntlast1.isOn {
411                 voegPuntlastToe_UGT(Grootte: grootte_puntlast_1*1.35, Positie: positie_puntlast_1)
412             }
413             else{
414                 voegPuntlastToe_UGT(Grootte: grootte_puntlast_1*1.50, Positie: positie_puntlast_1)
415             }
416             if VeiligheidPuntlast2.isOn {
417                 voegPuntlastToe_UGT(Grootte: grootte_puntlast_2*1.35, Positie: positie_puntlast_2)
418             }
419             else{
420                 voegPuntlastToe_UGT(Grootte: grootte_puntlast_2*1.50, Positie: positie_puntlast_2)
421             }
422             if VeiligheidPuntlast3.isOn {
423                 voegPuntlastToe_UGT(Grootte: grootte_puntlast_3*1.35, Positie: positie_puntlast_3)
424             }
425             else{
426                 voegPuntlastToe_UGT(Grootte: grootte_puntlast_3*1.50, Positie: positie_puntlast_3)
427             }
428             if VeiligheidPuntlast4.isOn {
429                 voegPuntlastToe_UGT(Grootte: grootte_puntlast_4*1.35, Positie: positie_puntlast_4)
430             }
431             else{
432                 voegPuntlastToe_UGT(Grootte: grootte_puntlast_4*1.50, Positie: positie_puntlast_4)
433             }
434         }
```

```
435         //lijnlasten toevoegen
436         if eindpositie_lijnlast_1 > scharnier_5 {
437             print("lijnlast1 buiten bereik")
438         }
439         else if eindpositie_lijnlast_2 > scharnier_5 {
440             print("lijnlast2 buiten bereik")
441         }
442         else if eindpositie_lijnlast_3 > scharnier_5 {
443             print("lijnlast3 buiten bereik")
444         }
445         else if eindpositie_lijnlast_4 > scharnier_5 {
446             print("lijnlast4 buiten bereik")
447         }
448         else{
449             //in GGT zijn de veiligheidsfactoren 1.0, 1 variabele last
450             VoegLijnlastToe_GGT(Grootte: grootte_lijnlast_1, Beginpositie: beginpositie_lijnlast_1, Eindpositie: eindpositie_lijnlast_1)
451             VoegLijnlastToe_GGT(Grootte: grootte_lijnlast_2, Beginpositie: beginpositie_lijnlast_2, Eindpositie: eindpositie_lijnlast_2)
452             VoegLijnlastToe_GGT(Grootte: grootte_lijnlast_3, Beginpositie: beginpositie_lijnlast_3, Eindpositie: eindpositie_lijnlast_3)
453             VoegLijnlastToe_GGT(Grootte: grootte_lijnlast_4, Beginpositie: beginpositie_lijnlast_4, Eindpositie: eindpositie_lijnlast_4)
454             if VeiligheidLijnlast1.isOn{
455                 VoegLijnlastToe_UGT(Grootte: grootte_lijnlast_1*1.35, Beginpositie: beginpositie_lijnlast_1, Eindpositie: eindpositie_lijnlast_1)
456             }
457             else{
458                 VoegLijnlastToe_UGT(Grootte: grootte_lijnlast_1*1.50, Beginpositie: beginpositie_lijnlast_1, Eindpositie: eindpositie_lijnlast_1)
459             }
460             if VeiligheidLijnlast2.isOn{
461                 VoegLijnlastToe_UGT(Grootte: grootte_lijnlast_2*1.35, Beginpositie: beginpositie_lijnlast_2, Eindpositie: eindpositie_lijnlast_2)
462             }
463             else{
464                 VoegLijnlastToe_UGT(Grootte: grootte_lijnlast_2*1.50, Beginpositie: beginpositie_lijnlast_2, Eindpositie: eindpositie_lijnlast_2)
465             }
466             if VeiligheidLijnlast3.isOn{
467                 VoegLijnlastToe_UGT(Grootte: grootte_lijnlast_3*1.35, Beginpositie: beginpositie_lijnlast_3, Eindpositie: eindpositie_lijnlast_3)
468             }
469             else{
470                 VoegLijnlastToe_UGT(Grootte: grootte_lijnlast_3*1.50, Beginpositie: beginpositie_lijnlast_3, Eindpositie: eindpositie_lijnlast_3)
471             }
472             if VeiligheidLijnlast4.isOn{
473                 VoegLijnlastToe_UGT(Grootte: grootte_lijnlast_4*1.35, Beginpositie: beginpositie_lijnlast_4, Eindpositie: eindpositie_lijnlast_4)
474             }
475             else{
476                 VoegLijnlastToe_UGT(Grootte: grootte_lijnlast_4*1.50, Beginpositie: beginpositie_lijnlast_4, Eindpositie: eindpositie_lijnlast_4)
477             }
478         }
479         for i in 0...dimensie_matrices-1{
480             print("\(i): \(krachten_UGT[i,0])")
481         }
482 
483     }
```

Bijlage A.2: Randvoorwaarden

Onderstaande code zijn de twee functies die de horizontale en verticale doorbuiging verhinderen waar nodig.

```
120     func randvoorwaardenHorizontaleVerplaatsing(scharnier: Double){
121         for i in 0...8{
122             if scharnier == 0{
123                 K_globaal_RVW_UGT[0,i] = 0.0
124                 K_globaal_RVW_UGT[i,0] = 0.0
125             }
126             else if Int(scharnier*elemPerMeter) == Int(aantal_elementen){
127                 K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3, Int(scharnier*elemPerMeter)*3-6+i]=0.0
128                 K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3-6+i, Int(scharnier*elemPerMeter)*3]=0.0
129 
130             }
131             else{
132                 K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3, Int(scharnier*elemPerMeter)*3-3+i]=0.0
133                 K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3-3+i, Int(scharnier*elemPerMeter)*3]=0.0
134             }
135         }
136         K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3, Int(scharnier*elemPerMeter)*3]=1.0
137     }
138     //Uy = 0 ter hoogte van een scharnier
139     func randvoorwaardenVerticaleVerplaatsing(scharnier: Double) {
140         if scharnier == 0{
141             for k in 0...5 {
142                 K_globaal_RVW_UGT[1, k] = 0.0
143                 K_globaal_RVW_UGT[k, 1] = 0.0
144             }
145         }
146         else if Int(scharnier*elemPerMeter) == Int(aantal_elementen){
147             for i in 0...8{
148                 //verticale verplaatsing = 0
149                 K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3-6+i, Int(scharnier*elemPerMeter)*3+1] = 0.0
150                 K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3+1, Int(scharnier*elemPerMeter)*3-6+i] = 0.0
151             }
152         }
153         else {
154             for i in 0...9{
155                 //verticale verplaatsing = 0
156                 K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3-4+i, Int(scharnier*elemPerMeter)*3+1] = 0.0
157                 K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3+1, Int(scharnier*elemPerMeter)*3-4+i] = 0.0
158             }
159         }
160         K_globaal_RVW_UGT[Int(scharnier*elemPerMeter)*3+1, Int(scharnier*elemPerMeter)*3+1] = 1.0
161 
162     }
```

Bijlage A.3: Verplaatsingen berekenen

De volledige functie voor het berekenen van de verplaatsingen met EEM. Ook de verplaatsingen die worden verhinderd ter hoogte van het steunpunt worden opgeslagen en vervolgens op nul gezet binnen de verplaatsingsmatrix.

```
183     func BerekenVerplaatsingen() {
184         for i in 0...dimensie_matrices{
185             if i == 1{
186                 //horizontale en verticale verplaatsing in scharnier = 0.0
187                 verplaatsingen_UGT[i-1,0] = 0.0
188                 verplaatsingen_UGT[i,0] = 0.0
189             }
190             else if i == Int(scharnier_2*elemPerMeter)*3+1{
191                 verplaatsingen_UGT[i-1,0] = 0.0
192                 verplaatsingen_UGT[i,0] = 0.0
193             }
194             else if i == Int(scharnier_3*elemPerMeter)*3+1{
195                 verplaatsingen_UGT[i-1,0] = 0.0
196                 verplaatsingen_UGT[i,0] = 0.0
197             }
198             else if i == Int(scharnier_4*elemPerMeter)*3+1{
199                 verplaatsingen_UGT[i-1,0] = 0.0
200                 verplaatsingen_UGT[i,0] = 0.0
201             }
202             else if i == Int(scharnier_5*elemPerMeter)*3+1{
203                 verplaatsingen_UGT[i-1,0] = 0.0
204                 verplaatsingen_UGT[i,0] = 0.0
205             }
206             else{
207                 for j in 0...dimensie_matrices{
208                     verplaatsingen_UGT[i,0] = verplaatsingen_UGT[i,0] + krachten_UGT[j,0]*K_globaal_RVW_inv_UGT[dimensie_matrices*i+i+j]
209                 }
210             }
211         }
212 
213     }
```

Bijlage A.4: Berekenen momenten en dwarskrachten

De uiteindelijke functie die alle handelingen in de juiste volgorde zal uitvoeren voor het bepalen van de momenten en dwarskrachten per knooppunt.

```
288     @IBAction func BerekenMVN(_ sender: Any) {
289         //eerst alle matrices op hun beginwaarde zetten, anders telt de methode de krachten elke keer op.
290         beginwaarden()
291 
292         //aanmaken van stijfheidsmatrix van een element -> globale stijfheidsmatrix
293         StijfheidsmatrixElement(I: Iy_IPE[1], A: A_IPE[1])
294         StijfheidsmatrixGlobaal()
295 
296         //nieuwe matrix aanmaken voor randvoorwaarden toe te voegen.
297         K_globaal_RVW_UGT = K_globaal_UGT
298 
299         randvoorwaardenHorizontaleVerplaatsing(scharnier: scharnier_1)
300         randvoorwaardenHorizontaleVerplaatsing(scharnier: scharnier_2)
301         randvoorwaardenHorizontaleVerplaatsing(scharnier: scharnier_3)
302         randvoorwaardenHorizontaleVerplaatsing(scharnier: scharnier_4)
303         randvoorwaardenHorizontaleVerplaatsing(scharnier: scharnier_5)
304         randvoorwaardenVerticaleVerplaatsing(scharnier: scharnier_1)
305         randvoorwaardenVerticaleVerplaatsing(scharnier: scharnier_2)
306         randvoorwaardenVerticaleVerplaatsing(scharnier: scharnier_3)
307         randvoorwaardenVerticaleVerplaatsing(scharnier: scharnier_4)
308         randvoorwaardenVerticaleVerplaatsing(scharnier: scharnier_5)
309 
310         //[K]^-1
311         K_globaal_RVW_inv_UGT = invert(matrix: K_globaal_RVW_UGT.grid)
312 
313         //uiteindelijk de matrixvergelijkingen oplossen
314         BerekenVerplaatsingen()
315         BepaalDwarskrachten()
316         BepaalMomenten()
```

Bijlage A.5: Kipweerstand

Onderstaande code geeft weer hoe voor elk veld de kipweerstand wordt gecontroleerd.

```
//functie om kip te controleren
func kipweerstandPerVeld(veld: Double, M: Double, profiel: String, It: Double, Iz: Double, Iw: Double, hoogte: Double, breedte: Double, Wply: Double, Wely: Double, klasse: Int){

    let G: Double = 81.0*pow(10.0,6)

    let Ncr = pow(3.14,2)*E_staal*Iz/pow(veld,2)
    //hulpfactor ingeroepen, omdat het anders te zwaar is voor xCode
    let Mcre_hulp_1 = G*It*E_staal*Iz
    let Mcre_hulp_2 = (3.14*3.14*E_staal*Iw/(veld*veld*G*It))
    let Mcre = 3.14/veld*pow(Mcre_hulp_1 + Mcre_hulp_1*Mcre_hulp_2,0.5)
    let yq = -hoogte/2
    var alfa = 0.0
    //alfa is altijd 1.13, behalve als er enkel puntlasten zijn toegevoegd
    if grootte_lijnlast_1 + grootte_lijnlast_2 + grootte_lijnlast_3 + grootte_lijnlast_4 != 0.0{
        alfa = 1.13
    }
    else{
        alfa = 1.35
    }
    let Mcr = alfa*Mcre*(pow(1+pow(0.4*alfa*yq/(Mcre/Ncr),2),0.5)+0.4*alfa*yq/(Mcre/Ncr))
    var lambda = 0.0
    if klasse <= 2{
        lambda = pow(Wply*fy/Mcr,0.5)
    }
    else if klasse == 3{
        lambda = pow(Wely*fy/Mcr,0.5)
    }
    var alfa_lt = 0.0
    if hoogte/breedte <= 2.0{
        alfa_lt = 0.21
    }
    else{
        alfa_lt = 0.34
    }
    let phi = 0.5*(1+alfa_lt*(lambda - 0.2)+pow(lambda,2))
    let chi = 1/(phi+pow(pow(phi,2)-pow(lambda,2),0.5))
    var Mbrd = 0.0
    if klasse <= 2{
        Mbrd = chi*Wply*fy
    }
    else{
        Mbrd = chi*Wely*fy
    }
    //controle per profiel
    if profiel == "IPE"{
        if Mbrd > M, kipweerstand_controle_IPE == true{
            kipweerstand_controle_IPE = true
        }
        else{
            kipweerstand_controle_IPE = false
        }
    }
```

Bijlage A.6: Berekenen dwarskrachtwapening

Volgende code stelt de functie voor die de dwarskrachtwapening controleert en eventuele bijkomende wapening berekent.

```
    func DwarskrachtControle(){
        let C_Rdc = 0.18/1.5
        var k = 1 + pow(200/d_opt,0.5)
        if k >= 2{
            k = 2.0
        }
        let rho = A_s_boven/(b*d_opt)
        let V_Rdc_1 = (C_Rdc*k*pow(100*rho*f_ck,1/3))*b*d_opt

        let v_min = 0.035*pow(k,1.5)*pow(f_ck,0.5)
        let V_Rdc_2 = v_min*b*d_opt

        var V_Rdc: Double = 0.0
        if V_Rdc_1 >= V_Rdc_2{
            V_Rdc = V_Rdc_1
        }
        else{
            V_Rdc = V_Rdc_2
        }

        //controleren of doorsnede van de balk voldoet aan dwarskracht
        let v = 0.6*(1-f_ck/250)
        if V_controle <= 0.5*b*d_opt*v*(f_ck*0.85/1.5){
            dwarskrachtControle = true
        }
        else{
            dwarskrachtControle = false
        }

        //controleren of er bijkomende dwarskracht moet gestoken worden
        if V_Rdc >= V_controle{
            dwarskrachtWapeningNodig = false
            A_sw_min = 0.08*pow(f_ck,0.5)*s*b/f_yk
        }
        else{
            //veronderstellen een cot(theta) van 2.0 (moet tussen 1 en 2.5 liggen)
            let cot = 2.0
            let z = 0.9*d_opt
            A_sw = s*V_controle*1.35/(z*f_yk/1.15*cot)

            let V_Rdmax =  1*b*z*v*(f_ck*0.85/1.5)/(cot+1/cot)
            if V_controle > V_Rdmax{
                dwarskrachtwapeningMogelijk = false
                print("niet mogelijk om voldoende dwarskrachtwapening te steken")
            }

            let A_sw_max = 1*v*f_ck*0.85/1.5*b*s/(2*f_yk/1.15)

            if A_sw <= A_sw_max{
                dwarskrachtwapeningMogelijk = true
            }
            else{
                tussenafstandDwarskrachtWapening = false
                dwarskrachtwapeningMogelijk =  false
                s = s + 50
            }
        }
        print("controle dwarskracht: \(dwarskrachtControle)")
    }
```

Bijlage A.7: Scheurwijdtecontrole

Deze functie controleert of het noodzakelijk is of de scheurwijdte moet worden gecontroleerd. Indien dit het geval is zal de maximale scheurwijdte 0,3 mm bedragen.

```
func ScheurwijdteControle(As: Double, M: Double, diam: Double){
    let x = (b*h*h*0.5+alfa*As*d_opt)/(As*alfa+b*h)
    let I = b*pow(x,3)/3+b*pow((h-x),3)/3+alfa*As*pow((d_opt-x),2)
    let v = h-x

    let W = I/v
    let M_cr = f_ctm*W

    if M > M_cr{
        scheurwijdteControleNodig = true

        //de scheurwijdtecontrole
        let h_c1 = (h-x)/3
        let h_c2 = 2.5*(h-d_opt)
        let h_c3 = h/2
        var h_c: Double
        if h_c1 <= h_c2, h_c1 <= h_c3{
            h_c = h_c1
        }
        else if h_c2 <= h_c1, h_c2 <= h_c3{
            h_c = h_c2
        }
        else{
            h_c = h_c3
        }
        let A_ceff = h_c*b
        let rho_p = As/A_ceff
        let Srmax = 3.4*dekking+0.8*0.5*0.425*diam/rho_p
        var sigma_s: Double = 0.0
        if diam == 8{
            sigma_s = 360
        }
        else if diam == 10{
            sigma_s = 320
        }
        else if diam == 12{
            sigma_s = 280
        }
        else if diam == 16{
            sigma_s = 340
        }
        else if diam == 20{
            sigma_s = 220
        }
        else if diam == 25{
            sigma_s = 200
        }
        else if diam == 32{
            sigma_s = 160
        }
        else if diam == 40{
            sigma_s = 160
        }

         var Eps = (sigma_s - 0.4*f_ctm/rho_p*(1+alfa*rho_p))/E_wapening

         if Eps < 0.6*sigma_s/E_wapening{
             Eps = 0.6*sigma_s/E_wapening
         }

         let Wk =  Srmax*Eps
         print("scheurwijdte: \(Wk) mm")
         if Wk <= 0.3{
             scheurwijdteControle = true
         }
         else{
             scheurwijdteControle = false
         }
    }
    else{
        scheurwijdteControleNodig = false
        scheurwijdteControle = true
    }
    print("controle scheurwijdte: \(scheurwijdteControle)")

}
```

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Ontwikkeling en implementatie van een ios-app voor de berekening en optimalisatie van hyperstatische balken

Richting: **master in de industriële wetenschappen: bouwkunde**
Jaar: **2017**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Bloemen, Ruben **Jehaes, Jelle**

Datum: **6/06/2017**