Bachelor Eindwerk

Invloedslijnen voor platen

Lars Imholz
4090497
November 2016

Eerste begeleider: J.W. Welleman
Tweede begeleider: P.C.J. Hoogenboom

Inhoudsopgave

Samenvatting 1

Inleiding 2

Ontwikkeling van het model 3

De verplaatsingenmethode 3

De stijfheidsmatrices 4

Wringing 7

Randbalken 9

Benodigde invoer 9

Snedekrachten 10

Python 10

Validatie van het model 12

Model zonder wringing 12

Belastingsgeval 1 12

Belastingsgeval 2 15

Belastingsgeval 3 17

Conclusie 21

Model met wringing 22

Belastingsgeval 1 23

Belastingsgeval 2 24

Belastingsgeval 3 25

Belastingsgeval 4 26

Belastingsgeval 5 27

Belastingsgeval 6 28

Belastingsgeval 7 29

Conclusie 29

Invloedslijnen 30

Zakking 30

Overige invloedslijnen 35

Conclusie 37

Aanbevelingen 38

Referentielijst 39

Bijlage 40
Tabellen van Guyon-Massonnet 40
Python code 42
Systeemmatrix 42
Plaatmodel 49
BruteForceMethode 53

Samenvatting

In dit rapport wordt een model voor de plaatwerking ontwikkeld om zodoende de invloedslijnen voor verschillende doorsnedekrachten en verplaatsingen te kunnen vinden. Omdat de plaatwerking geen stof is uit de Bachelor van civiele techniek, wordt de plaat opgedeeld in balkjes in beide richtingen. Op de plaatsen waar deze balkjes elkaar kruisen moet gelden dat de zakking en rotatie gelijk zijn.

Door de stijfheidsmatrix van de gehele constructie te bepalen, kunnen de verplaatsingen door middel van de verplaatsingenmethode worden opgelost. Vervolgens kunnen met deze verplaatsingen de doorsnedekrachten worden bepaald. De verkregen resultaten zijn gecontroleerd met al bestaande modellen en met analytische oplossingen uit de literatuur.

De resultaten uit het model zonder wringing kwamen nagenoeg perfect overeen met de modellen DIMOD en Femdem, wat betekent dat het model werkt. In de vergelijking met de analytische oplossingen van Timoshenko kwamen grotere verschillen naar voren, afhankelijk van welke oplegging werd gebruikt. Bij een volledige inklemming bleef de fout beperkt tot ongeveer 2 procent, maar bij een scharnierend opgelegde plaat nam de fout toe tot ongeveer 12 procent.

De invloedslijnen voor de momenten, dwarskrachten en zakkingen zijn bepaald met het model. Echter is alleen de invloedslijn voor de zakking gecontroleerd, vanwege tijdgebrek. In deze controle zit waarschijnlijk ook een fout, wat resulteerde in erg grote verschillen tussen de invloedslijn bepaald met het model en die volgde uit de analytische methode van Guyon-Massonnet.

Inleiding

Het doel van dit project is het ontwikkelen van een model voor de werking van een plaat, waarmee de invloedslijnen voor verschillende krachtsgrootheden en verplaatsingen kunnen worden bepaald, waarna deze getoetst kunnen worden aan de analytische methode.

In voorgaande jaren is al eerder onderzoek gedaan naar de modellering van platen door studenten van de TU Delft als onderwerp van het Bachelor Eindwerk. Hierbij werd gebruik gemaakt van de krachtenmethode. Eén van de aanbevelingen die volgden uit deze rapporten was het gebruik van de verplaatsingenmethode voor verdere modellering ten behoeve van de flexibiliteit.
Eind vorig jaar is er een model voor de plaatwerking ontwikkeld in Maple gebaseerd op de verplaatsingenmethode (Aarts, 2016). Het uitgangspunt voor dit project is het Maple-model gemaakt door deze student.
Tijdens het gebruik van het model bleek echter al snel dat de capaciteit van Maple niet toereikend was als het aantal balkjes werd opgevoerd om een nauwkeuriger resultaat te verkrijgen, wat resulteerde in langdurige berekeningstijden. Als gevolg hiervan is besloten om voor dit project gebruik te maken van een andere programmeertaal, namelijk Python, met de verwachting dat er grotere modellen doorgerekend kunnen worden.

Een belangrijke vraag die beantwoord moet worden is hoe het model omgeschreven kan worden naar de Python programmeertaal. Vervolgens moet het model uitgebreid worden om ook de dwarskrachten en momenten in de plaat te bepalen, zodat ook van deze krachtsgrootheden invloedslijnen kunnen worden opgesteld. Om te controleren of de door het model berekende waardes voor de krachtsgrootheden en verplaatsingen nauwkeurig genoeg zijn, zal het nieuwe model gevalideerd moeten worden met al bestaande modellen en theorieën. Tot slot kunnen de verschillende invloedslijnen worden bepaald en vergeleken met de analytische methode.

Ontwikkeling van het model

Hieronder zal ingegaan worden op de theorie die is gebruikt om de werking van een plaat te modelleren. De gebruikte formules zullen worden toegelicht, evenals eventuele versimpelingen die zijn doorgevoerd om tot het model te komen.

De verplaatsingenmethode

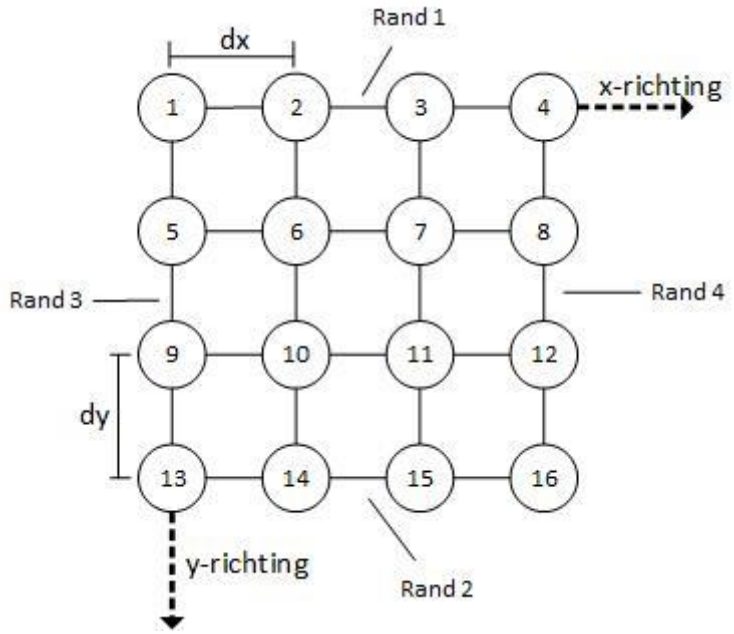
Zoals eerder vermeld wordt in dit model de verplaatsingenmethode gebruikt om de verplaatsingen en daaropvolgend de doorsnedekrachten te bepalen. Dit betekent dat de verplaatsingen als onbekend verondersteld worden. De formule die gebruikt wordt voor de verplaatsingenmethode is hieronder weergegeven.

$$[K] * [u] = [f]$$

Vergelijking 1: De verplaatsingenmethode in matrix notatie

Dit houdt in dat de stijfheid van een systeem (K) vermenigvuldigd met de verplaatsing (u) gelijk is aan de belasting op dat systeem (f). Deze begrippen worden meestal in matrixnotatie geschreven, zoals hierboven ook is gedaan. Aangezien de belasting en de stijfheid van het systeem normaliter bekend zijn, deze hangt namelijk af van de mechanische eigenschappen van het systeem, kunnen hieruit de verplaatsingen worden opgelost.

Om de werking van een plaat te modelleren, wordt de plaat in beide richtingen opgedeeld in balkjes. Op de punten waar deze balkjes elkaar kruisen, moet natuurlijk gelden dat de zakkingen van beide balkjes gelijk zijn, evenals de hoekverdraaiingen van twee opeenvolgende balkjes. Met andere woorden: er mogen geen sprongen (gelijke zakking) of knikken (gelijke hoekverdraaiing) aanwezig zijn in de plaat. Een voorbeeld van een model is hieronder weergegeven.



Figuur 1: Voorbeeld van een model met 3 balkjes in beide richtingen

De stijfheidsmatrices

Voordat de stijfheidsmatrix voor het gehele systeem kan worden opgesteld, wordt eerst gekeken naar de stijfheidsmatrix van één enkel balkje in de x-richting. Hiervoor zal gebruik gemaakt worden van de bekende differentiaalvergelijking voor een balk belast door een verdeelde belasting. Deze vergelijking is hieronder weergegeven.

$$EI_{zzx} * \frac{d^4}{dx^4} w(x) = \ q_x \ met \ I_{zzx} = \frac{1}{12} * dy * h^3$$

Vergelijking 2: De DV voor de zakking van een balk belast met een verdeelde belasting en de formule voor het traagheidsmoment

Als deze vergelijking viermaal wordt geïntegreerd, levert het de volgende uitdrukking op:

$$w(x) = \frac{1}{24}\frac{q_x x^4}{EI_{zzx}} + \frac{1}{6}C1x^3 + \frac{1}{2}C2x^2 + C3x + C4$$

Vergelijking 3: De oplossing voor de zakking met de bijbehorende integratieconstanten

Zoals te zien is zijn er 4 onbekende integratieconstanten aanwezig in de uitdrukking die moeten worden opgelost. Hiervoor zijn vier randvoorwaarden nodig:

$$\begin{aligned} op\ x = 0 \quad &\rightarrow \quad w = w_A \quad en \quad \varphi_y = \varphi_{yA} \\ op\ x = dx \quad &\rightarrow \quad w = w_B \quad en \quad \varphi_y = \varphi_{yB} \end{aligned}$$

Samen met de mechanica relatie voor de hoekverdraaiing $\varphi_y = -\frac{d}{dx} w(x)$ kunnen de integratieconstanten worden bepaald.



In de figuur hierboven is een balkje in de x-richting te zien met daarbij de verplaatsingen en doorsnedekrachten. De doorsnedekrachten worden bepaald met de volgende mechanica relaties:

$$\kappa_y = \frac{d}{dx}\varphi_y \qquad M_y = EI_{zzx} * \kappa_y \qquad V = \frac{d}{dx} M_y$$

Vergelijking 4: Bekende mechanica relaties voor de kromming, het moment en de dwarskracht

$$op\ x = 0 \quad \rightarrow \quad F_A = -V_z \quad en \quad M_{yA} = -M_y$$
$$op\ x = dx \quad \rightarrow \quad F_B = V_z \quad en \quad M_{yB} = M_y$$

Door bovenstaande relaties en randvoorwaarden in te vullen in de gevonden uitdrukking voor w(x) worden vergelijkingen gevonden voor de verticale krachten en de momenten. Deze vergelijkingen zijn hieronder weergegeven.

$$F_A = -\frac{1}{2}\frac{dx^4 q_x - 24EI_{zzx}w_A + 12EI_{zzx}dx\varphi_{yA} + 24EI_{zzx}w_B + 12EI_{zzx}dx\varphi_{yB}}{dx^3}$$

$$F_B = \frac{1}{2}\frac{-dx^4 q_x - 24EI_{zzx}w_A + 12EI_{zzx}dx\varphi_{yA} + 24EI_{zzx}w_B + 12EI_{zzx}dx\varphi_{yB}}{dx^3}$$

$$M_{yA} = \frac{1}{12}\frac{dx^4 q_x - 72EI_{zzx}w_A + 48EI_{zzx}dx\varphi_{yA} + 72EI_{zzx}wB + 24EI_{zzx}dx\varphi_{yB}}{dx^2}$$

$$M_{yB} = \frac{1}{12}\frac{-dx^4 q_x - 72EI_{zzx}w_A + 24EI_{zzx}dx\varphi_{yA} + 72EI_{zzx}wB + 48EI_{zzx}dx\varphi_{yB}}{dx^2}$$

Vergelijking 5: Uitdrukkingen voor de krachten op een element in x-richting

Deze vergelijkingen kunnen in matrix notatie geschreven worden, wat hieronder is gedaan.

$$\begin{bmatrix} \frac{12EI_{zzx}}{dx^3} & -\frac{6EI_{zzx}}{dx^2} & 0 & -\frac{12EI_{zzx}}{dx^3} & -\frac{6EI_{zzx}}{dx^2} & 0 \\ -\frac{6EI_{zzx}}{dx^2} & \frac{4EI_{zzx}}{dx} & 0 & \frac{6EI_{zzx}}{dx^2} & \frac{2EI_{zzx}}{dx} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{12EI_{zzx}}{dx^3} & \frac{6EI_{zzx}}{dx^2} & 0 & \frac{12EI_{zzx}}{dx^3} & \frac{6EI_{zzx}}{dx^2} & 0 \\ -\frac{6EI_{zzx}}{dx^2} & \frac{2EI_{zzx}}{dx} & 0 & \frac{6EI_{zzx}}{dx^2} & \frac{4EI_{zzx}}{dx} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} w_A \\ \varphi_{yA} \\ \varphi_{xA} \\ w_B \\ \varphi_{yB} \\ \varphi_{xB} \end{bmatrix} = \begin{bmatrix} F_A \\ M_{yA} \\ M_{xA} \\ F_B \\ M_{yB} \\ M_{xB} \end{bmatrix}$$

Hiermee is de stijfheidsmatrix van een enkel element in de x-richting dus bekend.

Voor een balkje in de y-richting gaat dit op dezelfde manier, wat resulteert in de volgende stijfheidsmatrix:

$$\begin{bmatrix} \frac{12EI_{zzy}}{dy^3} & 0 & -\frac{6EI_{zzy}}{dy^2} & -\frac{12EI_{zzy}}{dy^3} & 0 & -\frac{6EI_{zzy}}{dy^2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{6EI_{zzy}}{dy^2} & 0 & \frac{4EI_{zzy}}{dy} & \frac{6EI_{zzy}}{dy^2} & 0 & \frac{2EI_{zzy}}{dy} \\ -\frac{12EI_{zzy}}{dy^3} & 0 & \frac{6EI_{zzy}}{dy^2} & \frac{12EI_{zzy}}{dy^3} & 0 & \frac{6EI_{zzy}}{dy^2} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{6EI_{zzy}}{dy^2} & 0 & \frac{2EI_{zzy}}{dy} & \frac{6EI_{zzy}}{dy^2} & 0 & \frac{4EI_{zzy}}{dy} \end{bmatrix}$$

Nu de elementstijfheidsmatrices in beide richtingen bekend zijn, kunnen deze geassembleerd worden voor meerdere balkjes in x en y-richting (Blauwendraad, 2000). Hieronder is dit voor gedaan voor twee opeenvolgende balkjes in x-richting.

$$\begin{bmatrix} \frac{12EI_{zzx}}{dx^3} & -\frac{6EI_{zzx}}{dx^2} & 0 & -\frac{12EI_{zzx}}{dx^3} & -\frac{6EI_{zzx}}{dx^2} & 0 & 0 & 0 & 0 \\ -\frac{6EI_{zzx}}{dx^2} & \frac{4EI_{zzx}}{dx} & 0 & \frac{6EI_{zzx}}{dx^2} & \frac{2EI_{zzx}}{dx} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{12EI_{zzx}}{dx^3} & \frac{6EI_{zzx}}{dx^2} & 0 & \frac{24EI_{zzx}}{dx^3} & 0 & 0 & -\frac{12EI_{zzx}}{dx^3} & -\frac{6EI_{zzx}}{dx^2} & 0 \\ -\frac{6EI_{zzx}}{dx^2} & \frac{2EI_{zzx}}{dx} & 0 & 0 & \frac{8EI_{zzx}}{dx} & 0 & \frac{6EI_{zzx}}{dx^2} & \frac{2EI_{zzx}}{dx} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{12EI_{zzx}}{dx^3} & \frac{6EI_{zzx}}{dx^2} & 0 & \frac{12EI_{zzx}}{dx^3} & \frac{6EI_{zzx}}{dx^2} & 0 \\ 0 & 0 & 0 & -\frac{6EI_{zzx}}{dx^2} & \frac{2EI_{zzx}}{dx} & 0 & \frac{6EI_{zzx}}{dx^2} & \frac{4EI_{zzx}}{dx} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

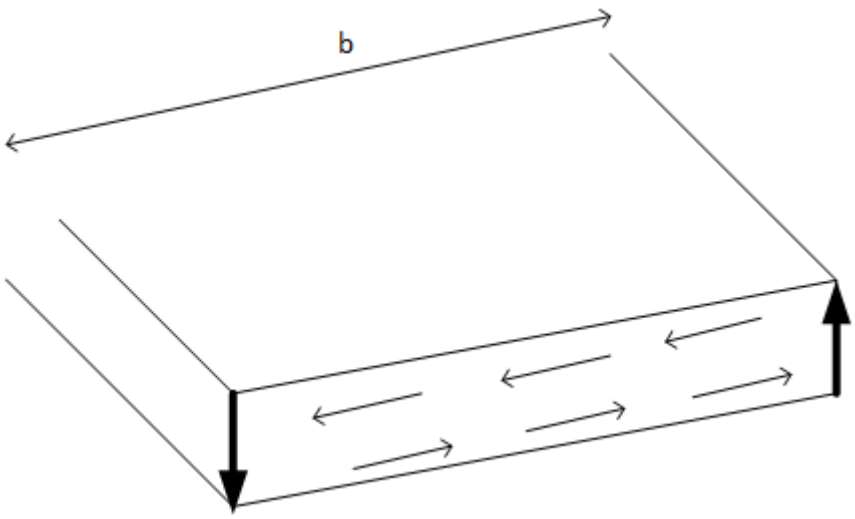
De twee omkaderde rechthoeken zijn allebei elemenstijfheidsmatrices. In het overlappende gedeelte worden de waardes van beide matrices opgeteld. Op de plekken van de rode nul-waardes heffen de factoren elkaar dus op: $\frac{-6EI_{zzx}}{dx^2}$ en $\frac{6EI_{zzx}}{dx^2}$

De systeemstijfheidsmatrix kan nu verkregen worden door per knooppunt de stijfheidsmatrices voor de x en y-richting op te tellen.

Wringing

Om een realistisch model van de plaatwerking te verkrijgen, is het noodzakelijk dat de wringing wordt meegenomen in het model. Daarom zal de torsiestijfheid aan de elementstijfheidsmatrix moeten worden toegevoegd. Omdat we het gedrag van een plaat zo goed mogelijk willen modelleren, kan niet zonder meer de torsiestijfheid van een balk gebruikt worden. Daarom wordt de benodigde uitdrukking voor de torsiestijfheid afgeleid.

Bij torsie in een balk wordt normaal gesproken gebruik gemaakt van de uitdrukking $M_w = GI_w\theta$. Bij platen spreekt men echter niet van de grote M_w, maar van de kleine m_w (Nm/m). Er moet dus een relatie gevonden worden tussen de M_w en de m_w in de plaat. In figuur 2 is de wringing in een plaatdeel geschematiseerd. De schuifspanningen lopen rond met een grootte m_w. Het moment dat wordt geleverd door deze schuifspanningen is dan $m_w b$ met b de breedte van het plaatdeeltje. Er werken echter geen verticale schuifspanningen in de doorsnede van een dunne plaat, zoals je gewend bent bij bijvoorbeeld een balk. In de hoeken van de plaat werkt echter wel een geconcentreerde dwarskracht (aangegeven met de dikke zwarte pijlen), ook met een grootte m_w. Het moment dat geleverd wordt door dit koppel is eveneens $m_w b$. Het totale moment dat dus geleverd wordt door de doorsnede $M_w = 2*m_w b$.



Figuur 2: Wringing in een plaatelement

De grootte van m_w wordt gegeven door (Blaauwendraad, 2010)

$$m_w = (1 - \nu)D(-\frac{d^2w}{dxdy})$$

Dit kan omgeschreven worden naar

$$\frac{M_w}{2b} = \frac{(1 - \nu)Eh^3}{12(1 - \nu^2)}\theta$$

In ons geval is de breedte van een element gelijk aan dy en $(1 - \nu)$ kan worden weggedeeld.

$$\frac{M_w}{2dy} = \frac{Eh^3}{12(1+\nu)}\theta$$

$$M_w = \frac{Eh^3 dy}{6(1+\nu)}\theta$$

$$M_w = G\frac{h^3 dy}{3}\theta \; met \; G = \frac{E}{2(1+\nu)} \; en \; \theta = \frac{\varphi_x}{dx}$$

$$M_w = \frac{GIwx}{dx} * \varphi_x$$

De uitdrukking voor de torsiestijfheid is nu dus bekend en kan worden toegevoegd aan de elementstijfheidsmatrices. Dit levert de volgende elementstijfheidsmatrices op voor de x-richting

$$\begin{bmatrix} \frac{12EI_{zzx}}{dx^3} & -\frac{6EI_{zzx}}{dx^2} & 0 & -\frac{12EI_{zzx}}{dx^3} & -\frac{6EI_{zzx}}{dx^2} & 0 \\ -\frac{6EI_{zzx}}{dx^2} & \frac{4EI_{zzx}}{dx} & 0 & \frac{6EI_{zzx}}{dx^2} & \frac{2EI_{zzx}}{dx} & 0 \\ 0 & 0 & \frac{GI_{wx}}{dx} & 0 & 0 & -\frac{GI_{wx}}{dx} \\ -\frac{12EI_{zzx}}{dx^3} & \frac{6EI_{zzx}}{dx^2} & 0 & \frac{12EI_{zzx}}{dx^3} & \frac{6EI_{zzx}}{dx^2} & 0 \\ -\frac{6EI_{zzx}}{dx^2} & \frac{2EI_{zzx}}{dx} & 0 & \frac{6EI_{zzx}}{dx^2} & \frac{4EI_{zzx}}{dx} & 0 \\ 0 & 0 & -\frac{GI_{wx}}{dx} & 0 & 0 & \frac{GI_{wx}}{dx} \end{bmatrix}$$

en voor de y-richting:

$$\begin{bmatrix} \frac{12EI_{zzy}}{dy^3} & 0 & -\frac{6EI_{zzy}}{dy^2} & -\frac{12EI_{zzy}}{dy^3} & 0 & -\frac{6EI_{zzy}}{dy^2} \\ 0 & \frac{GI_{wy}}{dy} & 0 & 0 & -\frac{GI_{wy}}{dy} & 0 \\ -\frac{6EI_{zzy}}{dy^2} & 0 & \frac{4EI_{zzy}}{dy} & \frac{6EI_{zzy}}{dy^2} & 0 & \frac{2EI_{zzy}}{dy} \\ -\frac{12EI_{zzy}}{dy^3} & 0 & \frac{6EI_{zzy}}{dy^2} & \frac{12EI_{zzy}}{dy^3} & 0 & \frac{6EI_{zzy}}{dy^2} \\ 0 & -\frac{GI_{wy}}{dy} & 0 & 0 & \frac{GI_{wy}}{dy} & 0 \\ -\frac{6EI_{zzy}}{dy^2} & 0 & \frac{2EI_{zzy}}{dy} & \frac{6EI_{zzy}}{dy^2} & 0 & \frac{4EI_{zzy}}{dy} \end{bmatrix}$$

Randbalken

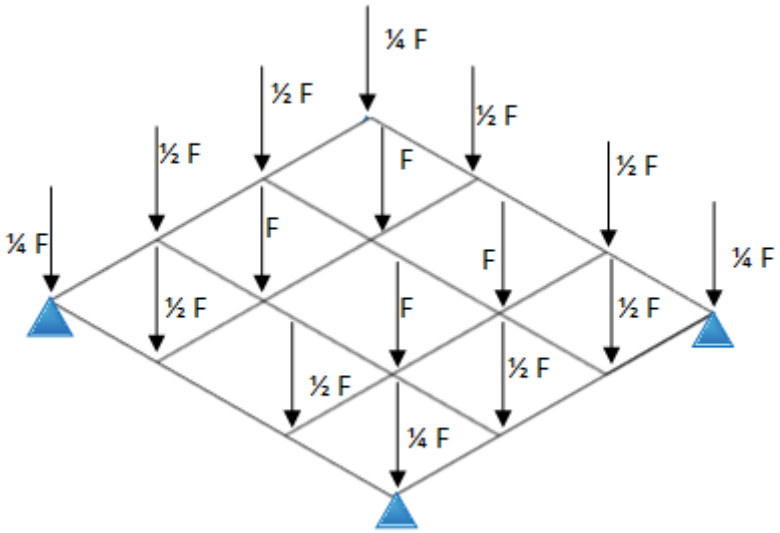
Voor de afleiding van de stijfheidsmatrices is gekeken naar een element in het midden van het veld. Deze balkjes zullen echter een ander gedrag vertonen dan de balkjes aan de rand van de plaat. Dit komt ten eerste omdat de randbalken minder belasting dragen dan balken in het midden van het veld. Ten tweede hebben de randbalken niet een breedte dx of dy, maar slechts de helft hiervan. Daarom is de elementstijfheidsmatrix voor de randbalken gehalveerd ten opzichte van de overige balkjes. Ten slotte vereist de torsie in de randbalken nog extra de aandacht, zoals ook al opgemerkt was door Steven Oomen (Oomen, 2012). Hij ondervond dat torsiestijfheid in de randbalken ervoor zou kunnen zorgen dat de constructie zich daar gaat gedragen als een schijnbare inklemming. Om dit probleem te voorkomen wordt de torsiestijfheid in de randbalken nul gesteld.

Benodigde invoer

Allereerst moet aangegeven worden met hoeveel balkjes er gerekend moet worden. Dit wordt gedaan voor beide richtingen: n staat voor het aantal balkjes in de x-richting, m voor het aantal balkjes in de y-richting. Vervolgens moeten de afmetingen van de plaat ingevoerd worden, oftewel de lengte in x-richting (Lx), de lengte in y-richting (Ly) en de dikte van de plaat (h).

De wijze waarop de plaat is opgelegd kan per rand worden ingevoerd. Er kan gekozen worden uit een scharnierende oplegging, een volledige inklemming, een inklemming waarbij de rotatie om de x-as vrij is en een inklemming waarbij de rotatie om de y-as vrij is. De scharnierende oplegging en de volledige inklemming worden in de literatuur het meest gebruikt, dus de nadruk in dit rapport zal liggen op deze twee opleggingen.

Ten slotte moet de buigstijfheid van de plaat worden ingevoerd en de belasting die op de plaat werkt. De buigstijfheid kan voor zowel de x-richting als de y-richting apart worden ingevoerd: Ex staat voor buigstijfheid in de x-richting, Ey voor de buigstijfheid in de y-richting. De belasting wordt ingevoerd als een gelijkmatig verdeelde belasting die werkt over de gehele plaat. Deze verdeelde belasting wordt vervolgens omgezet naar puntlasten die aangrijpen op de knooppunten. De grootte van de puntlasten is afhankelijk van de locatie van de knoop. Als de knoop in het midden van het veld ligt, dan is de grootte q*dx*dy. Ligt de knoop aan de rand, dan wordt de grootte q*1/2*dx*dy. Wordt er gekeken naar een knoop in de hoek van de plaat, dan wordt de grootte van de puntlast q*1/2*1/2*dx*dy. Dit is in de figuur hieronder weergegeven.



Deze versimpeling van de afdracht van de belasting brengt uiteraard een onnauwkeurigheid met zich mee. Echter, als de hoeveelheid balkjes in x en y-richting wordt opgevoerd, wordt de verdeelde belasting steeds beter benaderd omdat de puntlasten meer verdeeld worden over de plaat. Het verhogen van het aantal balkjes zorgt dus voor een betere benadering van de werkelijkheid. De q_x die te zien is in de uitdrukkingen voor de krachten op een enkel element (vergelijking 5), wordt in het model op nul gesteld. Deze q_x staat namelijk voor de verdeelde belasting in x-richting. In de praktijk komt deze manier van belasten eigenlijk nooit voor. De verdeelde belasting over de gehele plaat, zoals hierboven is beschreven, is een veel vaker gebruikte belastingswijze. Voor de q_y geldt hetzelfde verhaal.
Als alle gegevens zijn ingevoerd in het model, kunnen de verplaatsingen en rotaties worden berekend. Het resultaat is een matrix met daarin de zakking en de rotatie voor zowel de x en y-richting voor elke knoop.

Snedekrachten

Nu alle verplaatsingen en rotaties van de knooppunten bekend zijn, kunnen deze gebruikt worden om de snedekrachten te bepalen. Omdat de plaat enkel in de knooppunten van de balkjes wordt belast, verlopen de momenten lineair over de lengte van een balkje. Deze momenten kunnen gevonden worden door de verplaatsingen en rotaties in te vullen in de uitdrukkingen die eerder gevonden zijn voor de momenten bij het opstellen van de elementstijfheidsmatrices.
Omdat de belasting enkel op de knooppunten aangrijpt, is de dwarskracht constant over de balk. Daarom de dwarskrachten simpelweg verkregen worden door het verschil in moment links en rechts van een balkje te delen door de lengte van de balk.

Python

Zoals al eerder vermeld, wordt het model van dit bachelor eindwerk gemaakt in Python. Waar Maple uit zichzelf al een compleet rekenprogramma is waarin gerekend kan worden met matrices, stelsels van vergelijkingen kunnen worden opgelost en dataplots kunnen worden gemaakt, is dit bij Python niet het geval. Python biedt wel de mogelijkheid tot het

toevoegen van ‘externe bibliotheken’ waardoor deze functies aan Python kunnen worden toegevoegd. Voor dit model zijn een aantal externe bibliotheken gebruikt.

Numpy
Numpy is essentiële toevoeging aan Python voor het maken van en rekenen met matrices, hetgeen erg veel gebruikt is in dit model.

Sympy
Sympy bevat handige functies voor het rekenen met symbolen en het oplossen van stelsels vergelijkingen, evenals voor het vereenvoudigen van uitdrukkingen en het verzamelen van coëfficiënten.

Matplotlib
Matplotlib is onmisbaar voor het plotten van data in Python, waardoor het een noodzakelijke toevoeging was voor dit project.

Voor de volledige Python-code van het model wordt verwezen naar de bijlage.

Validatie van het model

Om er zeker van te zijn dat het model een realistische weergave van de werkelijke plaatwerking is, wordt het model gevalideerd met behulp van al bestaande modellen en analytische oplossingen uit de literatuur. Tijdens de validatie zal er gekeken worden naar twee verschillende situaties: het model zónder invloed van wringing en het model waarbij de wringing wel is meegenomen in de berekening. Voor beide situaties zullen verschillende belastingsgevallen bekeken worden, waarbij gevarieerd wordt met de opleggingen, de geometrie en de wijze van belasten.

Model zonder wringing

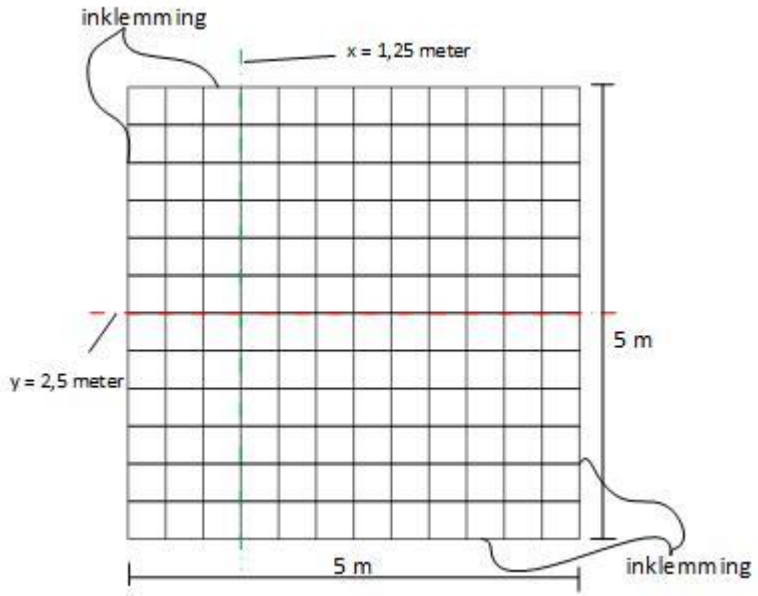
Het model zonder wringing zal vergeleken worden met DIMOD en Femdem. DIMOD, wat staat voor 'Discrete modellering van plaatwerking' is een model gebaseerd op de krachtenmethode, gemaakt door Evert van Vliet voor diens Bachelor Eindwerk (Vliet, 2009). Femdem is een modelleringsprogramma ontwikkeld door Dhr. J.W. Welleman, wat gebruik maakt van de eindige elementen methode.

Belastingsgeval 1

Voor belastingsgeval 1 wordt gekeken naar een vierkante plaat die aan alle randen volledig is ingeklemd. Dit houdt in dat de verplaatsingen en de hoekverdraaiingen aan alle randen nul zijn. Er is gekozen voor een uniform verdeelde belasting van 100 kN/m^2. Voor de overige gegevens wordt verwezen naar de onderstaande tabel.

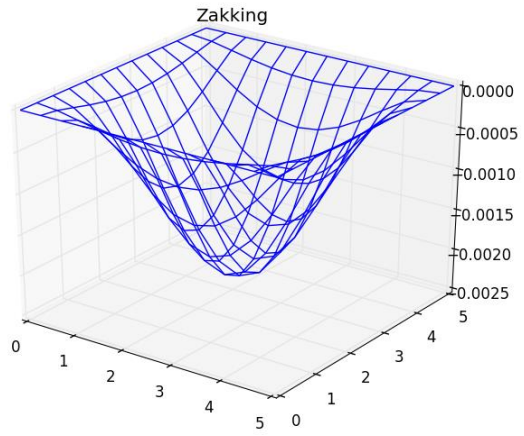
n	12
m	12
Lx	5 m
Ly	5 m
EIx	20000 kNm^2
EIy	20000 kNm^2
q	100 kN/m^2
h	0,25 m
GIw_x	0
GIw_y	0

Tabel 1: Invoer voor BG 1

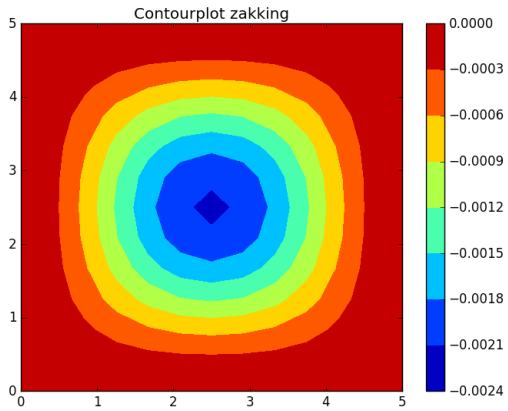


Figuur 3: Model voor BG1

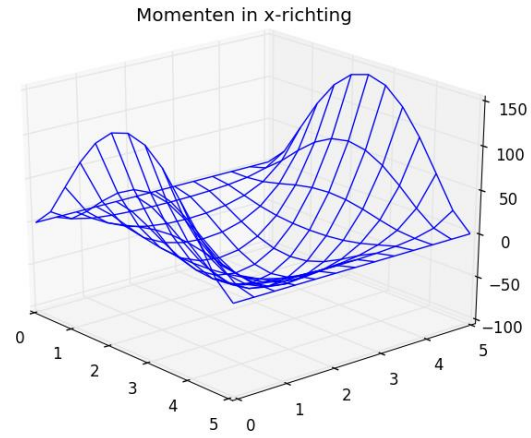
DIMOD geeft als uitkomsten de momenten in x en y-richting en de zakkingen. Deze waardes zullen dan ook met elkaar vergeleken worden. Als eerste wordt een snede gemaakt bij y = 2,5 meter om de verplaatsingen en momenten in x-richting te bekijken. Dit is halverwege de plaat. De resultaten zijn weergegeven in onderstaande tabel.



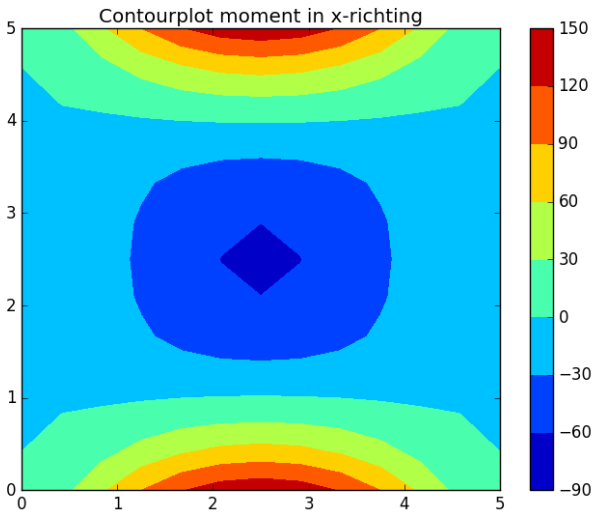
Figuur 4: Zakking BG1



Figuur 5: Contourplot zakking BG1



Figuur 6: Momenten in x-richting



Figuur 7: contourplot moment in x-richting

	Zakkingen			Momenten		
Punt	Python	DIMOD	Verschil (%)	Python	DIMOD	Verschil (%)
1	0	0	0,000	-141,7020	-141,7015	0,000
2	0,000214	0,000214	0,000	-71,0118	-71,0118	0,000
3	0,000694	0,000694	0,000	-17,1862	-17,1862	0,000
4	0,001246	0,001246	0,000	21,1582	21,1582	0,000
5	0,001730	0,001730	0,000	46,0586	46,0586	0,000
6	0,002054	0,002054	0,000	59,7745	59,7745	0,000
7	0,002167	0,002167	0,000	64,1148	64,1148	0,000
8	0,002054	0,002054	0,000	59,7745	59,7745	0,000
9	0,001730	0,001730	0,000	46,0586	46,0586	0,000
10	0,001246	0,001246	0,000	21,1582	21,1582	0,000
11	0,000694	0,000694	0,000	-17,1862	-17,1862	0,000
12	0,000214	0,000214	0,000	-71,0118	-71,0118	0,000
13	0	0	0,000	-141,7020	-141,7015	0,000

Tabel 2: BG1. Zakkingen en momenten in x-richting bij y = 2,5 meter

Zoals te zien is komen de resultaten nagenoeg exact met elkaar overeen.

Om nu de zakkingen en momenten in de y-richting te bekijken, wordt een snede gemaakt bij x = 1,25 meter. Dit is op een kwart van de breedte van de plaat. De resultaten zijn weergegeven in onderstaande tabel.

	Zakkingen			**Momenten**		
Punt	Python	DIMOD	Verschil (%)	Python	DIMOD	Verschil (%)
1	0	0	0,000	-89,6225	-89,6225	0,000
2	0,000132	0,000132	0,000	-39,9673	-39,9673	0,000
3	0,000419	0,000419	0,000	-6,4511	-6,4511	0,000
4	0,000736	0,000736	0,000	14,3907	14,3907	0,000
5	0,001007	0,001007	0,000	26,5519	26,5519	0,000
6	0,001185	0,001185	0,000	32,8719	32,8719	0,000
7	0,001246	0,001246	0,000	34,8304	34,8304	0,000
8	0,001185	0,001185	0,000	32,8719	32,8719	0,000
9	0,001007	0,001007	0,000	26,5519	26,5519	0,000
10	0,000736	0,000736	0,000	14,3907	14,3907	0,000
11	0,000419	0,000419	0,000	-6,4511	-6,4511	0,000
12	0,000132	0,000132	0,000	-39,9673	-39,9673	0,000
13	0	0	0,000	-89,6225	-89,6225	0,000

Tabel 3: BG1. Zakkingen en momenten in y-richting bij x = 1,25 meter

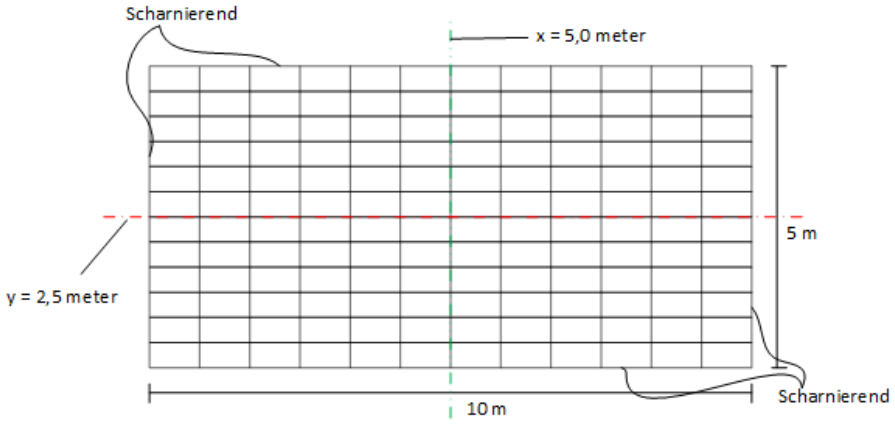
Ook hier komen de resultaten nagenoeg perfect met elkaar overeen.

Belastingsgeval 2

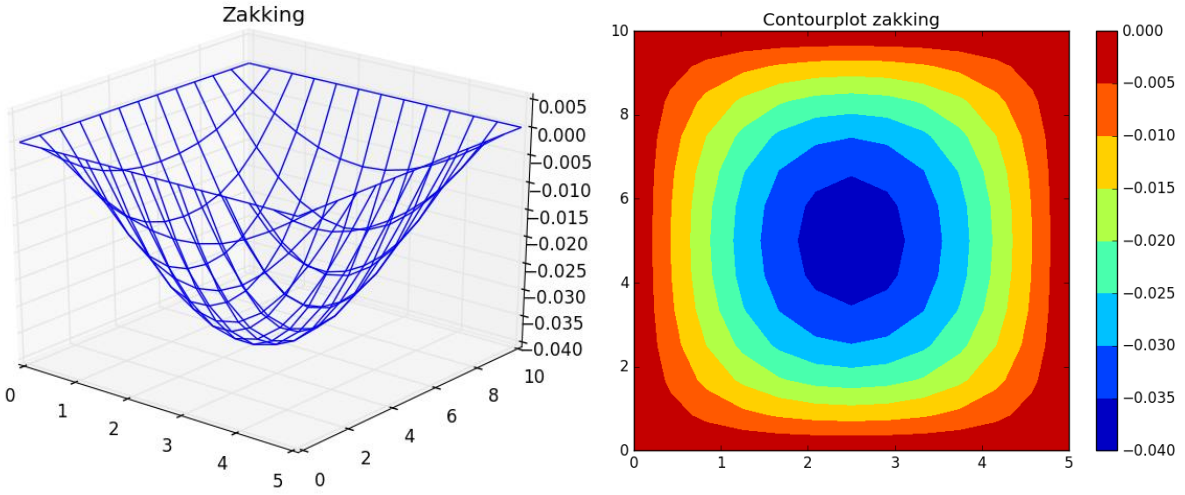
Voor belastingsgeval 2 wordt gekeken naar een rechthoekige plaat met een tweemaal zo grote lengte in de x-richting als in de y-richting. De plaat wordt weer belast met een verdeelde belasting van 100 kN/m^2. Verder is de plaat aan de randen scharnierend opgelegd, wat betekent dat de zakkingen op de randen verhinderd zijn, maar de rotaties niet. Tevens is de buigstijfheid in de x-richting gehalveerd. Voor de overige gegevens wordt verwezen naar onderstaande tabel.

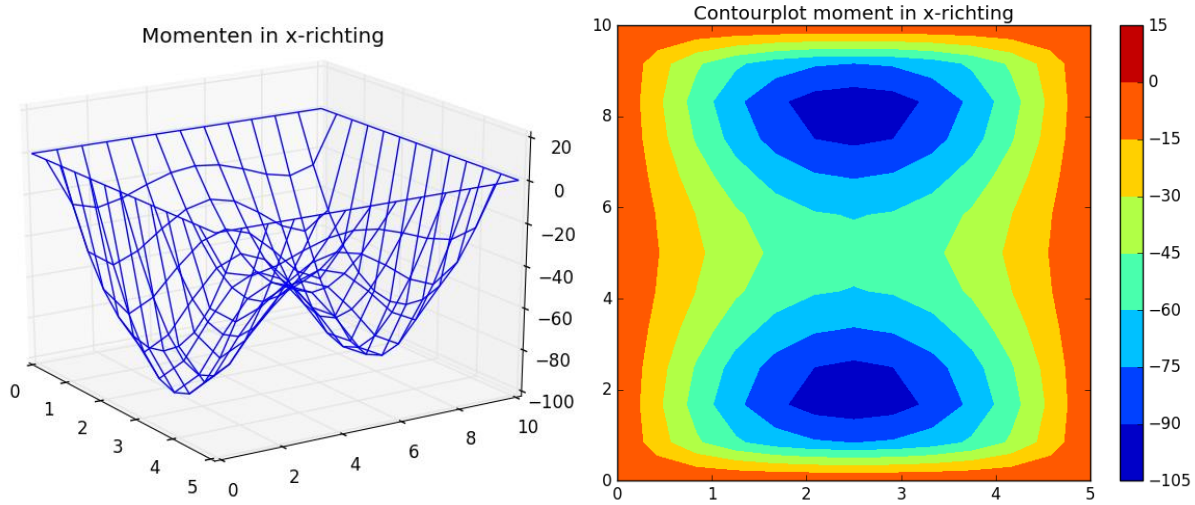
n	12
m	12
Lx	10 m
Ly	5 m
EIx	10000 kNm^2
EIy	20000 kNm^2
q	100 kN/m^2
h	0,25 m
GIw_x	0
GIw_y	0

Tabel 4: Invoer voor BG2



Figuur 8: Model voor BG2





Er wordt wederom een snede gemaakt bij y = 2,5 meter om de verplaatsingen en momenten in de x-richting te vergelijken. De resultaten zijn weergegeven in onderstaande tabel.

	Zakkingen			Momenten		
Punt	Python	DIMOD	Verschil (%)	Python	DIMOD	Verschil (%)
1	0	0	0,000	0	0	0,000
2	0,012141	0,012141	0,000	75,2451	75,2451	0,000
3	0,022350	0,022350	0,000	99,4617	99,4617	0,000
4	0,029828	0,029828	0,000	93,2351	93,2351	0,000
5	0,034662	0,034662	0,000	75,8715	75,8715	0,000
6	0,037290	0,037290	0,000	60,7836	60,7836	0,000
7	0,038114	0,038114	0,000	55,0403	55,0403	0,000
8	0,037290	0,037290	0,000	60,7836	60,7836	0,000
9	0,034662	0,034662	0,000	75,8715	75,8715	0,000
10	0,029828	0,029828	0,000	93,2351	93,2351	0,000
11	0,022350	0,022350	0,000	99,4617	99,4617	0,000
12	0,012141	0,012141	0,000	75,2451	75,2451	0,000
13	0	0	0,000	0	0	0,000

Tabel 5: BG2. Zakkingen en momenten in x-richting bij y = 2,5 meter

Wederom komen de resultaten nagenoeg exact overeen.

Vervolgens wordt er een snede gemaakt bij x = 5 meter om de verplaatsingen en momenten in de y-richting te bekijken. De resultaten zijn te vinden in onderstaande tabel.

	Zakkingen			Momenten		
Punt	Python	DIMOD	Verschil (%)	Python	DIMOD	Verschil (%)
1	0	0	0,000	0	0	0,000
2	0,010001	0,010001	0,000	106,3000	106,3004	0,000
3	0,019255	0,019255	0,000	194,5140	194,5140	0,000
4	0,027125	0,027125	0,000	263,9580	263,9583	0,000
5	0,033109	0,033109	0,000	314,0370	314,0369	0,000
6	0,036845	0,036845	0,000	344,2830	344,2834	0,000
7	0,038114	0,038114	0,000	354,4000	354,3998	0,000
8	0,036845	0,036845	0,000	344,2830	344,2834	0,000
9	0,033109	0,033109	0,000	314,0370	314,0369	0,000
10	0,027125	0,027125	0,000	263,9580	263,9583	0,000
11	0,019255	0,019255	0,000	194,5140	194,5140	0,000
12	0,010001	0,010001	0,000	106,3000	106,3004	0,000
13	0	0	0,000	0	0	0,000

Tabel 6: BG2. Zakkingen en momenten in x-richting bij y = 5 meter

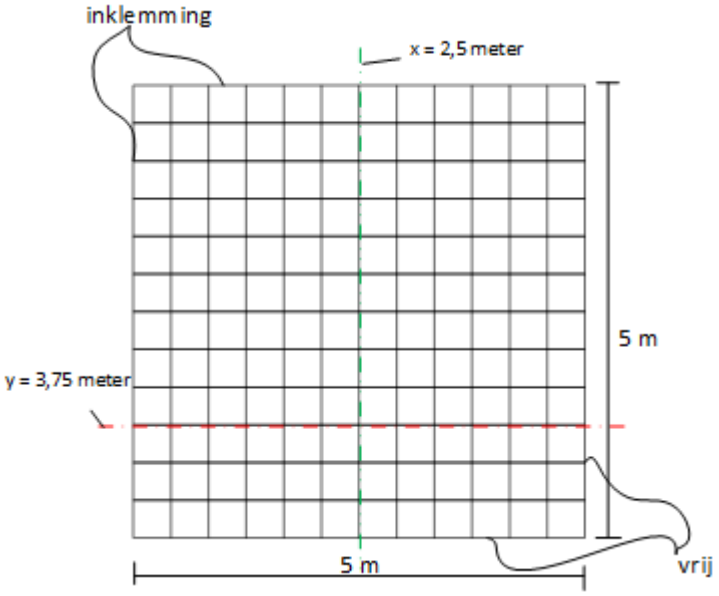
Ook hier komen de resultaten bijna perfect overeen.

Belastingsgeval 3

Voor belastingsgeval 3 wordt gekeken naar een vierkante plaat waar twee aangrenzende randen worden ingeklemd en de overige twee randen worden vrijgelaten. De plaat wordt dus in een hoek ingeklemd. De plaat wordt wederom belast door een verdeelde belasting van 100 kN/m^2. De buigstijfheden worden weer gelijkgesteld voor beide richtingen. Voor de overige gegevens wordt verwezen naar onderstaande tabel.

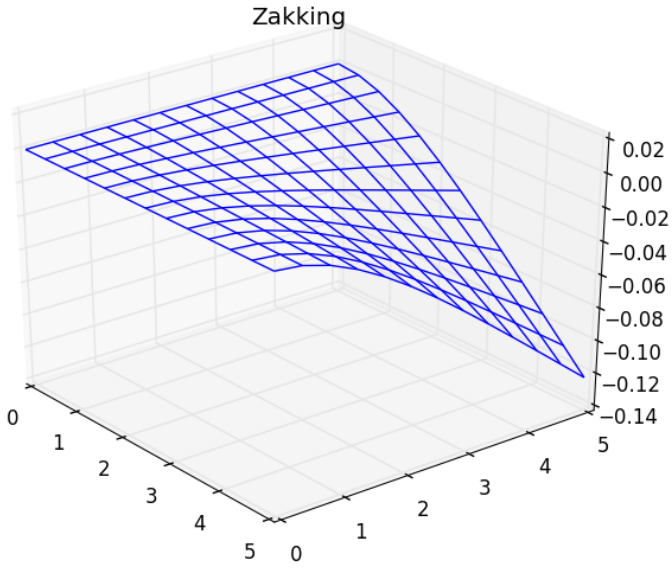
n	12
m	12
Lx	5 m
Ly	5 m
EIx	20000 kNm^2
EIy	20000 kNm^2
q	100 kN/m^2
h	0,25 m
GIw_x	0
GIw_y	0

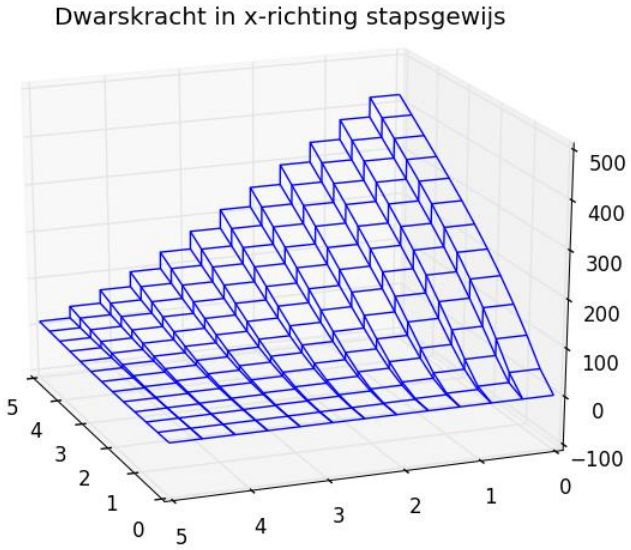
Tabel 7: Invoer voor BG3

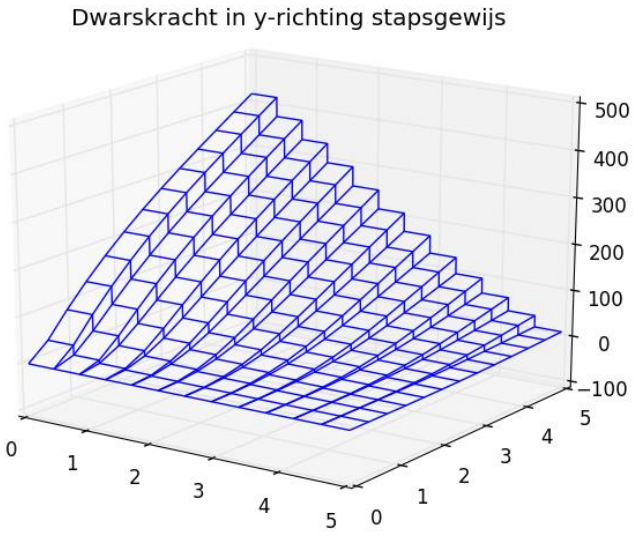


Figuur 9: Model voor BG3

Dit belastingsgeval zal gecontroleerd worden met Femdem. Anders dan bij DIMOD, bevat de uitdraai van Femdem naast de zakkingen en momenten ook de rotaties en dwarskrachten. Al deze gegevens zullen voor dit belastingsgeval dus vergeleken worden.


Zakking
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
−0.14
0
1
2
3
4
5


Dwarskracht in x-richting stapsgewijs
500
400
300
200
100
0
−100
5
4
3
2
1
0


Dwarskracht in y-richting stapsgewijs
500
400
300
200
100
0
−100
0
1
2
3
4
5

Er wordt een snede gemaakt op y = 3,75 meter om de verplaatsingen en snedekrachten in x-richting te bekijken. De resultaten zijn weergegeven in onderstaande tabellen.

	Zakkingen		
Punt	Python	Femdem	Verschil (%)
1	0	0	0,000
2	0,001207	0,001207	0,006
3	0,004500	0,004501	0,003
4	0,009439	0,009439	0,002
5	0,015641	0,015641	0,002
6	0,022785	0,022785	0,001
7	0,030606	0,030606	0,001
8	0,038890	0,038890	0,001
9	0,047472	0,047473	0,001
10	0,056232	0,056232	0,001
11	0,065083	0,065084	0,001
12	0,073973	0,073974	0,000
13	0,082873	0,082874	0,000

Tabel 8: BG3. Zakkingen in x-richting bij y = 3,75 meter

	Rotatie om x-as			Rotatie om y-as		
Punt	Python	Femdem	Verschil (%)	Python	Femdem	Verschil (%)
1	0	0	0,000	0	0	0,000
2	-0,000392	0,000392	0,004	-0,005590	-0,005590	0,000
3	-0,001501	0,001501	0,001	-0,010043	-0,010043	0,000
4	-0,003222	0,003222	0,001	-0,013509	-0,013509	0,000
5	-0,005456	0,005456	0,001	-0,016133	-0,016133	0,000
6	-0,008105	0,008105	0,000	-0,018053	-0,018053	0,000
7	-0,011078	0,011078	0,000	-0,019401	-0,019401	0,000
8	-0,014291	0,014291	0,000	-0,020297	-0,020297	0,000
9	-0,017672	0,017672	0,000	-0,020852	-0,020852	0,000
10	-0,021161	0,021160	0,000	-0,021161	-0,021160	0,000
11	-0,024710	0,024710	0,000	-0,021305	-0,021305	0,000
12	-0,028287	0,028287	0,000	-0,021355	-0,021355	0,000
13	-0,031872	0,031872	0,000	-0,021363	-0,021363	0,000

Tabel 9: Rotaties in x-richting bij y = 3,75 meter

	Momenten			Dwarskrachten		
Punt	Python	Femdem	Verschil (%)	Python	Femdem	Verschil (%)
1	-297,381	-297,381	0,000	139,5330	139,5334	0,000
2	-239,242	-239,242	0,000	122,2620	122,2622	0,000
3	-188,300	-188,300	0,000	105,2860	105,2863	0,000
4	-144,430	-144,430	0,000	88,8456	88,8455	0,000
5	-107,411	-107,411	0,000	73,2025	73,2024	0,000
6	-76,910	-76,910	0,000	58,6188	58,6187	0,000
7	-52,486	-52,486	0,000	45,3245	45,3245	0,000
8	-33,601	-33,601	0,000	33,4897	33,4897	0,000
9	-19,647	-19,646	0,000	23,2131	23,2131	0,000
10	-9,974	-9,974	0,000	14,5325	14,5325	0,000
11	-3,919	-3,919	0,000	7,4500	7,4500	0,000
12	-0,815	-0,815	0,001	1,9560	1,9560	-0,001
13	0	0	0,000			

Tabel 10: BG3. Momenten en dwarskrachten in x-richting bij y = 3,75 meter

De resultaten komen ook in dit geval nagenoeg perfect overeen, met uitzondering van het minteken bij de rotaties om de x-as. Dit komt door een ander gekozen assenstelsel.

Voor het vergelijken van de resultaten in y-richting wordt een snede gemaakt bij x = 2,5 meter. De resultaten zijn te vinden in onderstaande tabellen.

	Zakkingen		
Punt	Python	Femdem	Verschil (%)
1	0	0	0,000
2	0,000718	0,000718	0,008
3	0,002638	0,002638	0,004
4	0,005454	0,005454	0,003
5	0,008918	0,008919	0,002
6	0,012836	0,012836	0,002
7	0,017058	0,017058	0,001
8	0,021475	0,021476	0,001
9	0,026009	0,026009	0,001
10	0,030606	0,030606	0,001
11	0,035232	0,035232	0,001
12	0,039869	0,039869	0,001
13	0,044508	0,044509	0,001

Tabel 11: BG3. Zakkingen in y-richtingen bij x = 2,5 meter

	Rotatie om x-as			Rotatie om y-as		
Punt	Python	Femdem	Verschil (%)	Python	Femdem	Verschil (%)
1	0	0	0,000	0	0	0,000
2	-0,003300	0,003300	0,000	-0,000386	-0,000386	0,004
3	-0,005794	0,005794	0,000	-0,001463	-0,001463	0,002
4	-0,007624	0,007624	0,000	-0,003114	-0,003114	0,001
5	-0,008926	0,008926	0,000	-0,005223	-0,005223	0,001
6	-0,009819	0,009819	0,000	-0,007685	-0,007685	0,000
7	-0,010404	0,010404	0,000	-0,010404	-0,010404	0,000
8	-0,010767	0,010767	0,000	-0,013303	-0,013303	0,000
9	-0,010973	0,010973	0,000	-0,016317	-0,016317	0,000
10	-0,011078	0,011078	0,000	-0,019401	-0,019401	0,000
11	-0,011121	0,011121	0,000	-0,022520	-0,022520	0,000
12	-0,011134	0,011134	0,000	-0,025655	-0,025655	0,000
13	-0,011135	0,011135	0,000	-0,028794	-0,028794	0,000

Tabel 12: BG3. Rotaties in y-richting bij x = 2,5 meter

	Momenten			Dwarskrachten		
Punt	Python	Femdem	Verschil (%)	Python	Femdem	Verschil (%)
1	-179,563	-179,563	0,000	101,4986	101,4985	0,000
2	-137,272	-137,272	0,000	84,4340	84,4339	0,000
3	-102,091	-102,092	0,000	68,2623	68,2622	0,000
4	-73,649	-73,649	0,000	53,5519	53,5518	0,000
5	-51,335	-51,336	0,001	40,7463	40,7464	0,000
6	-34,358	-34,358	0,001	30,0414	30,0415	0,000
7	-21,841	-21,841	0,001	21,3608	21,3610	-0,001
8	-12,940	-12,940	0,001	14,4653	14,4654	-0,001
9	-6,913	-6,913	0,001	9,0895	9,0896	-0,001
10	-3,126	-3,126	0,001	5,0227	5,0227	-0,001
11	-1,033	-1,033	0,001	2,1310	2,1310	-0,001
12	-0,145	-0,145	0,004	0,3480	0,3480	-0,002
13	0	0	0,000			

Tabel 13: BG3. Momenten en dwarskrachten in y-richting bij x = 2,5 meter

Ook de resultaten in y-richting komen goed overeen. Vanzelfsprekend zijn ook hier de rotaties om de x-as tegengesteld van teken, wat eerder al verklaard is door een ander gekozen assenstelsel.

Conclusie

Zoals te zien is in de tabellen komen de resultaten van het model zonder wringing erg goed overeen met de al bestaande modellen DIMOD en Femdem. Hieruit kan geconcludeerd worden dat het model gebaseerd op de verplaatsingenmethode goed werkt en de verschillende opleggingen op de juiste wijze zijn geïmplementeerd.

Model met wringing

Het model met wringing zal gevalideerd worden aan de hand van de analytische oplossing voor een homogeen isotrope plaat, zoals deze te vinden zijn in de plaattheorie van Timoshenko. (S. Timoshenko, 1989). Timoshenko heeft voor veel verschillende randvoorwaarden en opleggingen de zakkingen bepaald. Aan de hand van deze oplossingen zal het model gecontroleerd worden. Voorafgaand zal eerst een nieuwe grootheid geïntroduceerd worden, namelijk de buigstijfheid van een plaat D. Zoals al eerder vermeld beperken we ons tot de isotrope plaat, wat betekent dat de buigstijfheid in beide richtingen gelijk is. Deze stijfheid D wordt gevonden met de volgende formule

$$D = \frac{Eh^3}{12(1 - v^2)}$$

De invoer voor de belastingsgevallen is weergegeven in tabel 14. Deze waardes zijn voor alle belastingsgevallen gelijk. De verschillen in opleggingen zijn per belastingsgeval hieronder beschreven, samen met de resultaten.

a (=Lx=Ly)	5 m
E	$20*10^6$
h	0,25 m
q	100 kN/m^2
v	0,3

Tabel 14: Invoer voor de belastingsgevallen 1 t/m 7

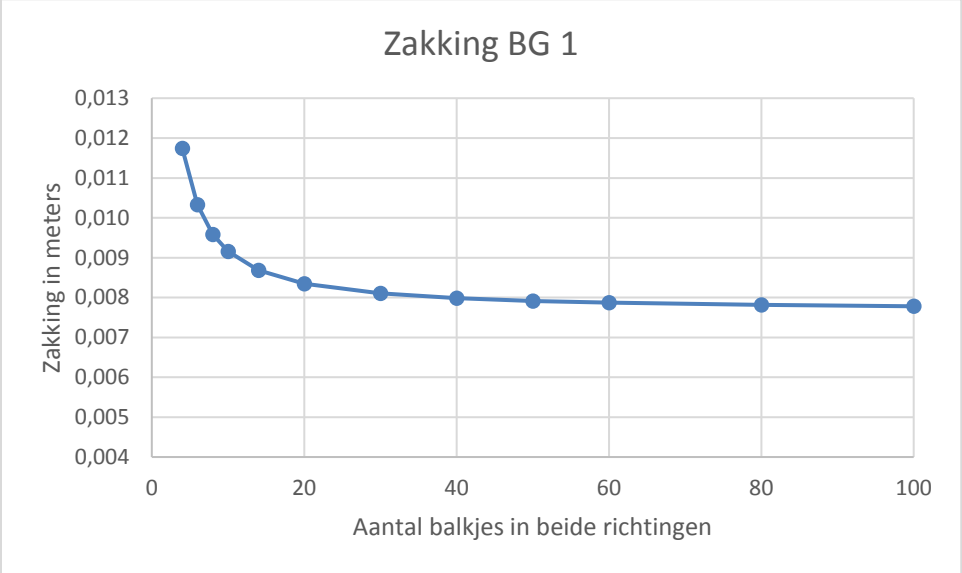
Belastingsgeval 1

Voor belastingsgeval 1 wordt gekeken naar een vierkante plaat die op alle randen scharnierend is opgelegd. De plaat is belast met een verdeelde belasting van 100 kN/m^2. De zakking in het midden van de plaat volgens Timoshenko wordt gegeven door

$$w_{midden} = 0{,}00406\frac{qa^4}{D}$$

Met de invoer uit bovenstaande tabel levert dit volgens Timoshenko een maximale doorbuiging van $w_{midden} = 0{,}00886704\ m$.

De resultaten uit het Python model zijn uiteraard afhankelijk van de hoeveelheid balkjes waarmee gerekend wordt. De verwachting is wel dat de oplossing convergeert als het aantal balkjes wordt opgevoerd. De resultaten zijn in onderstaande figuur weergegeven.



De verwachte convergentie van de zakking is duidelijk te zien in de figuur. De zakking bij 100 balkjes in beide richtingen bedraagt 0,00778086 m. Dit resulteert in een procentuele fout van

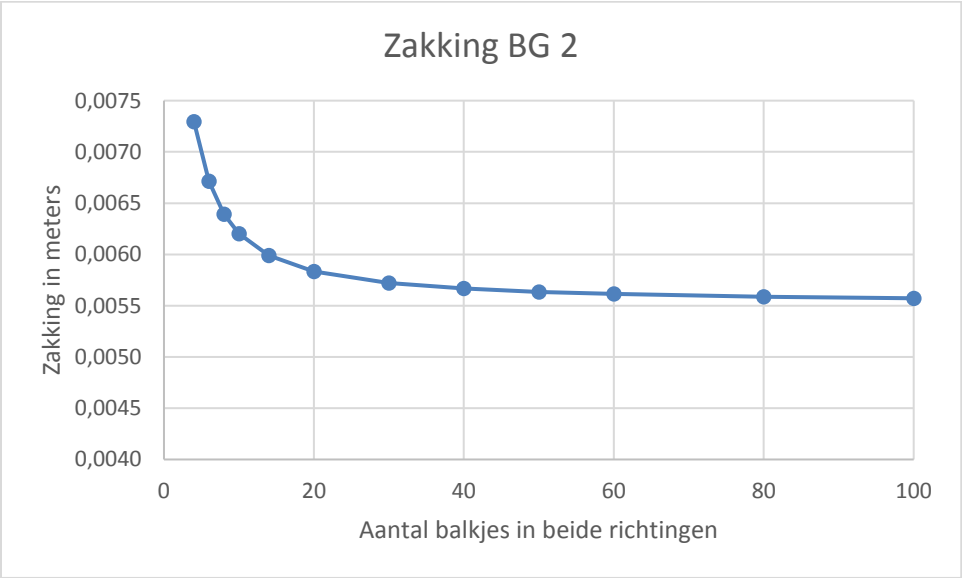
$$\frac{0{,}00886704 - 0{,}00778086}{0{,}00886704} * 100\% = 12{,}588\%$$

Belastingsgeval 2

Voor belastingsgeval 2 wordt gekeken naar een vierkante plaat met één rand ingeklemd en de resterende drie randen scharnierend opgelegd. De plaat is belast met een verdeelde belasting van 100 kN/m^2. De zakking in het midden van de plaat volgens Timoshenko wordt gegeven door

$$w_{midden} = 0{,}0028\frac{qa^4}{D}$$

Met de invoer uit tabel 14 geeft dit een zakking van $w_{midden} = 0{,}0061152\ m$. De resultaten uit Python zijn weergegeven in onderstaande figuur.



De zakking bij 100 balkjes in beide richtingen bedraagt 0,00557248 m. Dit resulteert in een procentuele fout van

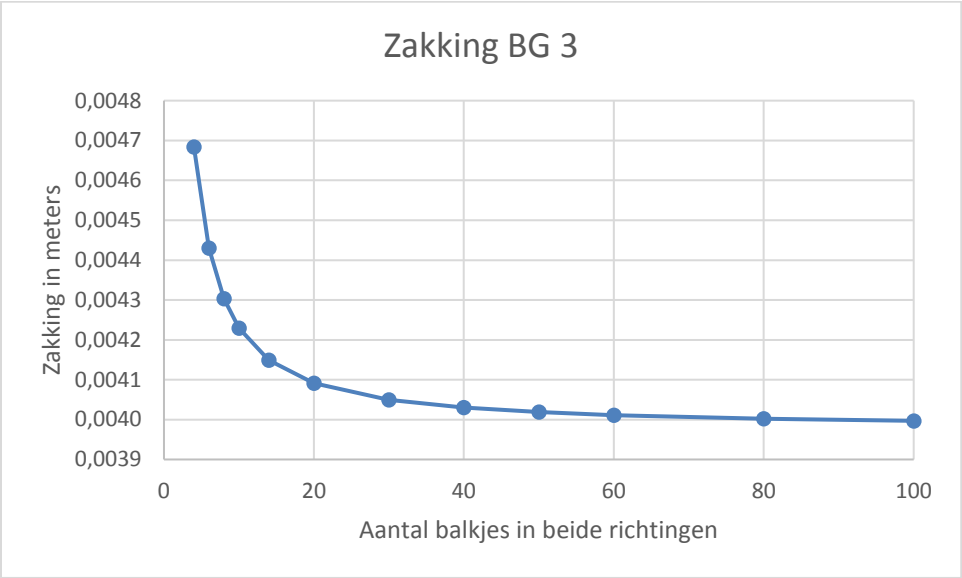
$$\frac{0{,}0061152 - 0{,}00557248}{0{,}0061152} * 100\% = 8{,}875\%$$

Belastingsgeval 3

Voor belastingsgeval 3 wordt gekeken naar een vierkante plaat met twee tegenover elkaar gelegen randen ingeklemd en de resterende twee randen scharnierend opgelegd. De plaat is belast met een verdeelde belasting van 100 kN/m^2. De zakking in het midden van de plaat volgens Timoshenko wordt gegeven door

$$w_{midden} = 0{,}00192\frac{qa^4}{D}$$

Met de invoer uit tabel 14 geeft dit een zakking van $w_{midden} = 0{,}00419328\ m$. De resultaten uit Python zijn weergegeven in onderstaande figuur.



De zakking bij 100 balkjes in beide richtingen bedraagt 0,00399694 m. Dit resulteert in een procentuele fout van

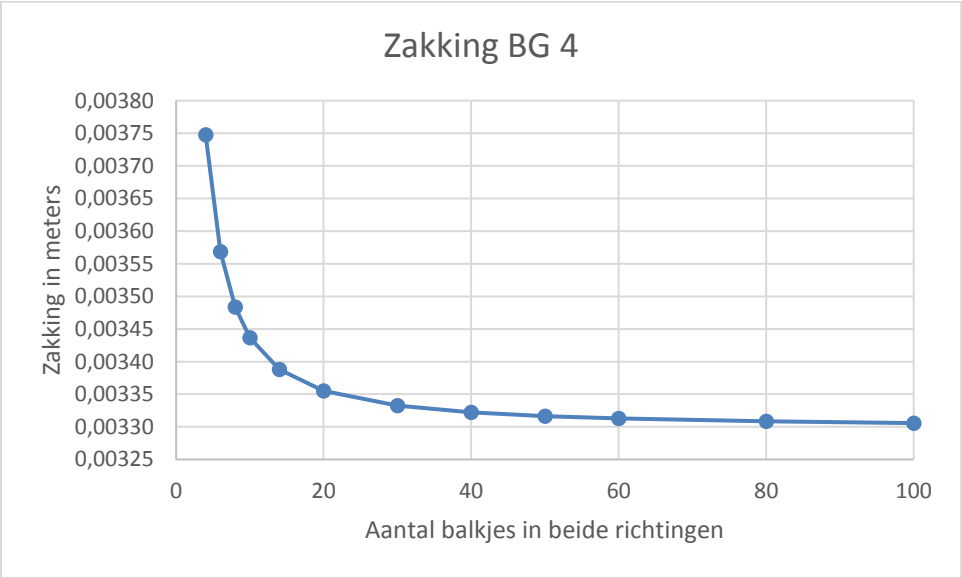
$$\frac{0{,}00419328\ - 0{,}00399694}{0{,}00419328} * 100\% = 4{,}682\%$$

Belastingsgeval 4

Voor belastingsgeval 4 wordt gekeken naar een vierkante plaat met drie randen ingeklemd en de resterende rand scharnierend opgelegd. De plaat is belast met een verdeelde belasting van 100 kN/m^2. De zakking in het midden van de plaat volgens Timoshenko wordt gegeven door

$$w_{midden} = 0{,}00157\frac{qa^4}{D}$$

Met de invoer uit tabel 14 geeft dit een zakking van $w_{midden} = 0{,}00342888\ m$. De resultaten uit Python zijn weergegeven in onderstaande figuur.



De zakking bij 100 balkjes in beide richtingen bedraagt 0,00330576 m. Dit resulteert in een procentuele fout van

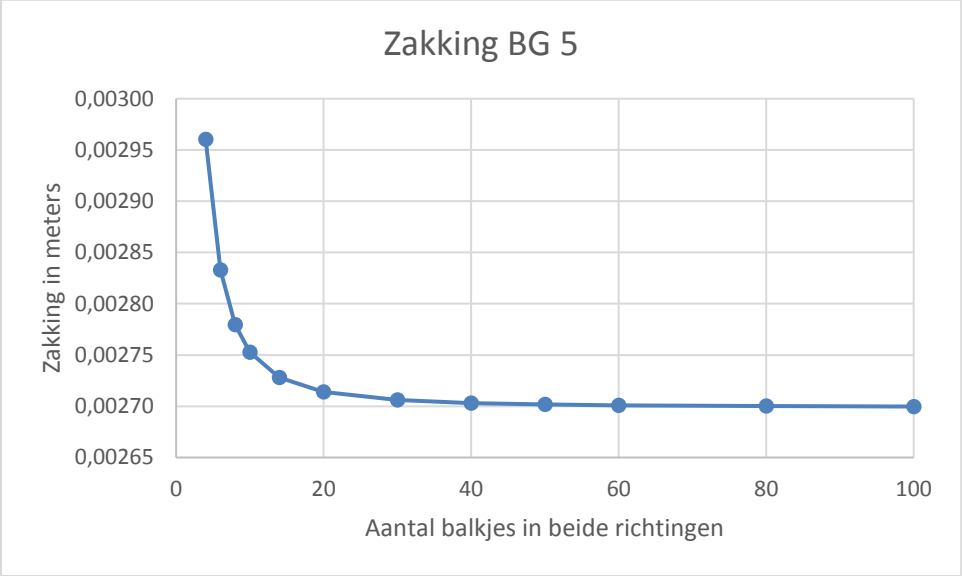
$$\frac{0{,}00342888\ - 0{,}00330576}{0{,}00342888} * 100\% = 3{,}591\%$$

Belastingsgeval 5

Voor belastingsgeval 5 wordt gekeken naar een vierkante plaat met alle randen ingeklemd. De plaat is belast met een verdeelde belasting van 100 kN/m^2. De zakking in het midden van de plaat volgens Timoshenko wordt gegeven door

$$w_{midden} = 0{,}00126\frac{qa^4}{D}$$

Met de invoer uit tabel 14 geeft dit een zakking van $w_{midden} = 0{,}00275184\ m$. De resultaten uit Python zijn weergegeven in onderstaande figuur.



De zakking bij 100 balkjes in beide richtingen bedraagt 0,00269973 m. Dit resulteert in een procentuele fout van

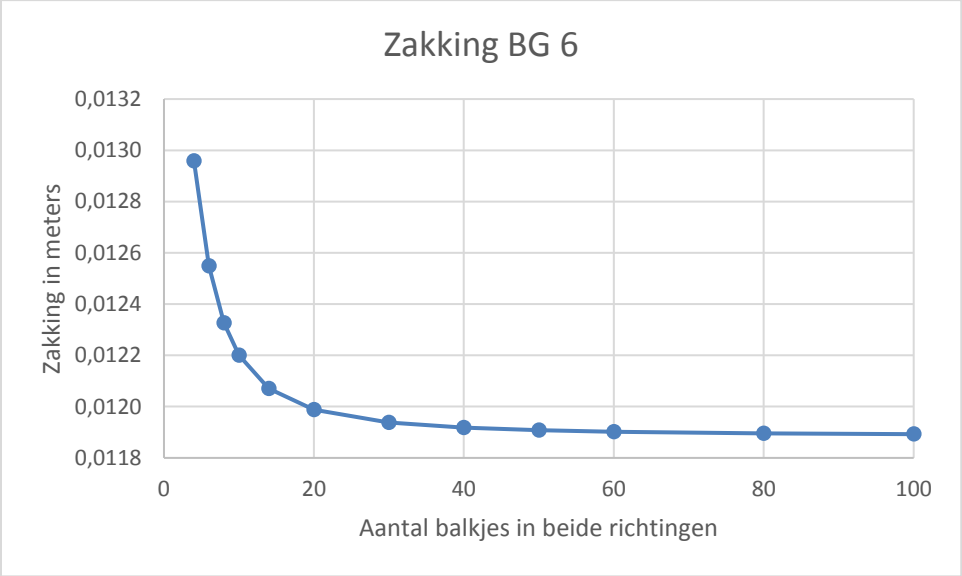
$$\frac{0{,}00275184\ -0{,}00269973}{0{,}00275184} * 100\% = 1{,}894\%$$

Belastingsgeval 6

Voor belastingsgeval 6 wordt gekeken naar een vierkante plaat met alle randen ingeklemd. De plaat wordt belast met een puntlast in het midden van de plaat. De puntlast heeft een grootte $P = qa^2$. De zakking in het midden van de plaat volgens Timoshenko wordt gegeven door

$$w_{midden} = 0{,}00560 * \frac{Pa^2}{D}$$

Met de invoer uit tabel 14 geeft dit een zakking van $w_{midden} = 0{,}0122304\ m$. De resultaten uit Python zijn weergegeven in onderstaande figuur.



De zakking bij 100 balkjes in beide richtingen bedraagt 0,01189262 m. Dit resulteert in een procentuele fout van

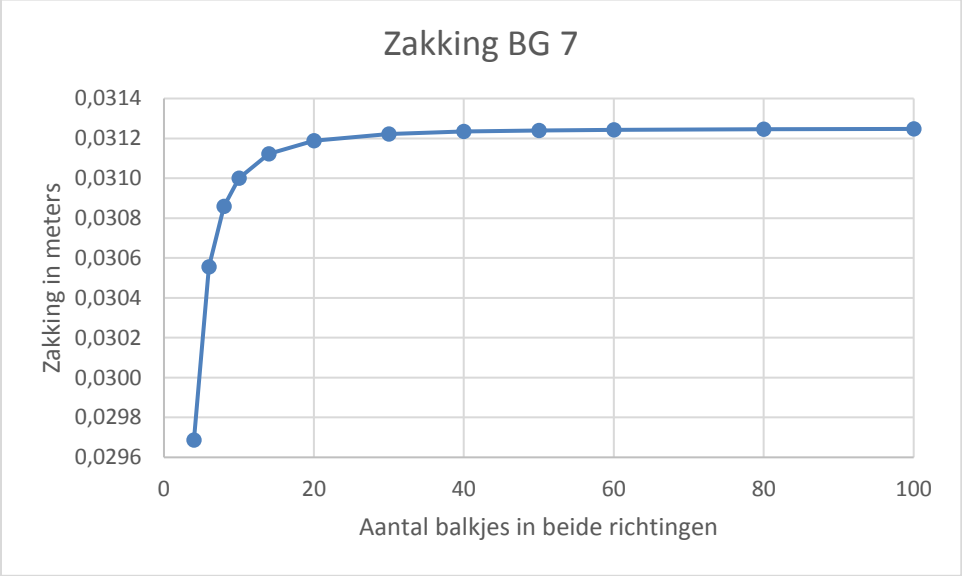
$$\frac{0{,}0122304\ - 0{,}01189262}{0{,}0122304} * 100\% = 2{,}762\%$$

Belastingsgeval 7

Bij dit belastingsgeval wordt er gekeken naar een vierkante plaat die op twee tegenover elkaar liggende randen scharnierend is op gelegd. De twee overige randen blijven vrij. Dit is een belangrijk geval om te controleren, omdat voor deze situatie later de invloedslijn voor de zakking zal worden bepaald en vergeleken. De plaat wordt wederom belast door een verdeelde belasting van 100 kN/m^2. De zakking in het midden van de plaat wordt volgens Timoshenko gegeven door

$$w_{midden} = 0{,}01309\frac{qa^4}{D}$$

Met de invoer uit tabel 14 geeft dit een zakking van $w_{midden} = 0{,}02858856\ m$. De resultaten uit Python zijn weergegeven in onderstaande figuur.



De zakking bij 100 balkjes in beide richtingen bedraagt 0,03124750 m. Dit resulteert in een procentuele fout van

$$\frac{0{,}02858856\ - 0{,}03124750}{0{,}02858856} * 100\% = -9{,}301\%$$

Conclusie

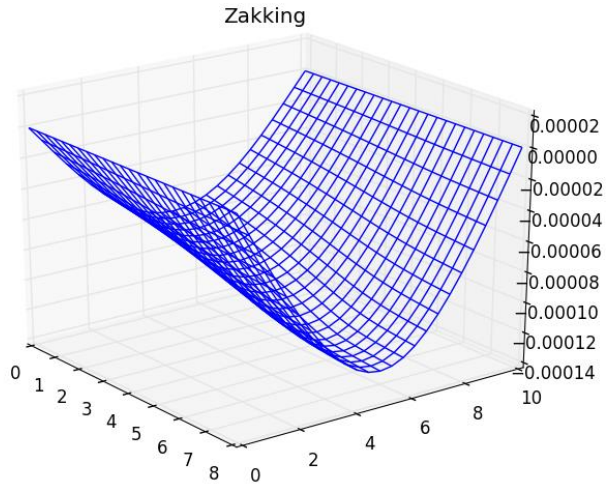
Het model werkt vrij aardig wat betreft de zakkingen. Zeker bij de gevallen waar de plaat is ingeklemd komen de resultaten goed overeen met maar een klein verschil van 2-3%. De afname van de fout naarmate de plaat meer wordt ingeklemd is ook heel sterk zichtbaar. Wat verder nog opvalt is het feit dat het model overal te stijf reageert, behalve bij belastingsgeval 7, waar de plaat aan twee zijden is opgelegd met een scharnierende oplegging. Hier is de zakking gevonden met het model namelijk 9,3% groter.

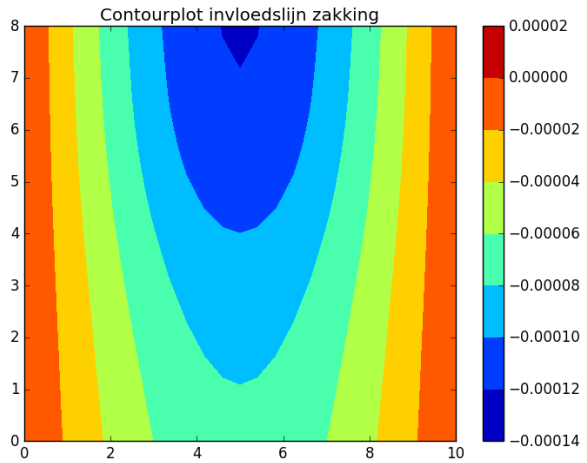
Invloedslijnen

De invloedslijnen voor de dwarskracht, het moment, de zakking en de oplegreacties kunnen worden bepaald met het model. Deze invloedslijnen worden bepaald door op een knooppunt een eenheidslast te plaatsen, de verplaatsingen en krachtsgrootheden voor deze situatie te bepalen, en vervolgens de eenheidslast te verplaatsen naar het volgende knooppunt. Wanneer de eenheidslast alle knooppunten langs is geweest, zijn de invloedslijnen bekend en kunnen ze geplot worden.

Zakking

De invloedslijn voor de zakking zal vergeleken worden met de analytische oplossing van Guyon-Massonnet. Het model bestaat uit een rechthoekige plaat van 8 bij 10 meter die op twee tegenoverliggende randen scharnierend is opgelegd. De overige twee randen zijn vrij. Er wordt gebruik gemaakt van 24 balkjes in beide richtingen. Het punt waar de invloedslijn voor bepaald is, ligt op (x, y) 6; 5. De invloedslijn uit het Python-model is hieronder weergegeven.



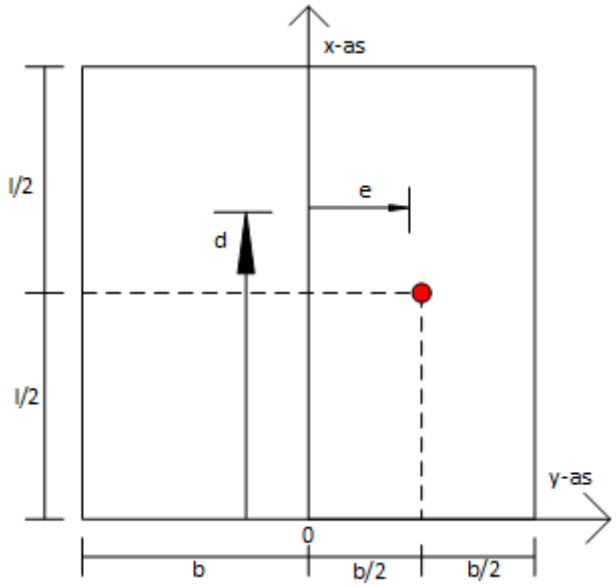


Om de invloedslijn met de theorie van Guyon-Massonnet te bepalen, moet wat meer werk worden verricht. De invloedslijn wordt gevonden met de volgende vergelijking (Bares, 1968):

$$w(x,y) = \frac{l^3}{b\varrho_T \pi^4} \sum_{m=1}^{\infty} K_m(y) \frac{1}{m^4} sin\frac{m\pi d}{l} sin\frac{m\pi x}{l}$$

$$waarbij\ \varrho_T\ staat\ voor\ de\ plaatstijfheid\ D = \frac{Eh^3}{12(1-\ v^2)}$$

In figuur 7 is een overzicht te zien van het model om de invloedslijn volgens de theorie van Guyon-Massonnet te bepalen. Het rode punt geeft vanzelfsprekend de plek aan waarvoor de invloedslijn moet worden bepaald.



Figuur 10: Model invloedslijn Guyon-Massonnet

Als eerst moeten de parameters voor de stijfheid, α en ϑ, worden gevonden voor deze situatie. Deze parameters zijn gegeven in de literatuur voor de hier gekozen situatie, namelijk α = 0,49 en ϑ = 0,7. Met deze waardes kunnen achter in het boek uit de tabellen de waardes voor K_0 en K_1 worden afgelezen bij y = b/2. Om de K-waardes voor ϑ = 2,1 te vinden moet er geïnterpoleerd worden met de waardes voor ϑ = 2,0 en ϑ = 2,2.
Vervolgens moeten de K_α waardes bepaald worden met behulp van de interpolatieformules

$$K_\alpha = K_0 + (K_1 - K_0)\alpha^{\left(1-e^{\frac{0{,}065-\vartheta}{0{,}663}}\right)}\ voor\ 0{,}1 <\ \vartheta\ \leq 1{,}0$$

$$K_\alpha = K_0 + 0{,}645(K_1 - K_0)\ \ voor\ 0{,}1 <\ \vartheta\ \leq 1{,}0$$

$$K_\alpha = K_0 + (K_1 - K_0)\sqrt{\alpha}\ \ voor\ \ \vartheta > 1{,}0$$

$$K_\alpha = K_0 + 0{,}7(K_1 - K_0)\ \ voor\ \ \vartheta > 1{,}0$$

Alle K-waardes zijn verzameld in onderstaande tabellen.

	K_0	K_1	$(K_1 - K_0)$	$0{,}645(K_1 - K_0)$	$K\alpha$
e			$\vartheta = 0{,}7$		
-b	-0,5114	0,3342	0,8456	0,5454	0,0340
-3/4b	-0,1756	0,4253	0,6009	0,3876	0,2120
-1/2b	0,1798	0,5535	0,3737	0,2410	0,4208
-1/4b	0,5862	0,7407	0,1545	0,0997	0,6859
0	1,0580	0,9923	-0,0657	-0,0424	1,0156
1/4b	1,5548	1,2824	-0,2724	-0,1757	1,3791
1/2b	1,9392	1,5134	-0,4258	-0,2746	1,6646
3/4b	2,0554	1,5539	-0,5015	-0,3235	1,7319
b	2,0618	1,5262	-0,5356	-0,3455	1,7163

Tabel 15: K-waardes bij $\vartheta = 0{,}7$

	K_0	K_1	$(K_1 - K_0)$	$0{,}7(K_1 - K_0)$	K_α
e			$\vartheta = 2{,}1$		
-b	0,0140	0,0029	-0,0111	-0,0078	0,0062
-3/4b	0,0062	0,0090	0,0029	0,0020	0,0081
-1/2b	-0,0483	0,0354	0,0837	0,0586	0,0103
-1/4b	-0,1828	0,0683	0,2511	0,1758	-0,0070
0	0,0147	0,5276	0,5129	0,3590	0,3737
1/4b	1,9332	1,6906	-0,2426	-0,1698	1,7633
1/2b	4,7938	3,3496	-1,4442	-1,0109	3,7828
3/4b	1,9308	1,8404	-0,0903	-0,0632	1,8675
b	-1,2341	0,8621	2,0961	1,4673	0,2332

Tabel 16: K-waardes bij $\vartheta = 2{,}1$

Nu kunnen de termen achter het som-teken berekend worden. Om een nauwkeurig resultaat te krijgen is het noodzakelijk minstens 3 m-termen te gebruiken. Deze worden hieronder uitgerekend.

$\frac{d}{l}$	$\frac{1}{m^4}\sin\frac{m\pi d}{l}$			$\sin\frac{m\pi x}{l}$			$\frac{1}{m^4}\sin\frac{m\pi d}{l}\sin\frac{m\pi x}{l}$		
	m = 1	m = 2	m = 3	m = 1	m = 2	m = 3	m = 1	m = 2	m = 3
1/8	0,3827	0,0442	0,0114	1	0	-1	0,3827	0	-0,0114
1/4	0,7071	0,0625	0,0087	1	0	-1	0,7071	0	-0,0087
3/8	0,9239	0,0442	-0,0047	1	0	-1	0,9239	0	0,0047
1/2	1	0	-0,0123	1	0	-1	1	0	0,0123
5/8	0,9239	-0,0442	-0,0047	1	0	-1	0,9239	0	0,0047
3/4	0,7071	-0,0625	0,0087	1	0	-1	0,7071	0	-0,0087
7/8	0,3827	-0,0442	0,0114	1	0	-1	0,3827	0	-0,0114

Tabel 17: Termen achter het som-teken voor m = 1 t/m 3

Zoals te zien is in tabel 17 vallen de termen met m = 2 weg. Daarom was het niet nodig om de K-waardes voor $\vartheta = 1{,}4$ op te zoeken in de tabellen.

Deze waardes uit tabel 17 kunnen nu vermenigvuldigd worden met de eerder gevonden K_α waardes, zoals in onderstaande tabel is te zien.

e	$a_m = K_m \frac{1}{m^4} sin\frac{m\pi d}{l} sin\frac{m\pi x}{l}$							
	d/l = 1/8; 7/8		d/l = 1/4; 3/4		d/l = 3/8; 5/8		d/l = 1/2	
	m = 1	m = 3	m = 1	m = 3	m = 1	m = 3	m = 1	m = 3
-b	0,01302	-0,00007	0,02405	-0,00005	0,03142	0,00003	0,03401	0,00008
-3/4b	0,08112	-0,00009	0,14989	-0,00007	0,19584	0,00004	0,21198	0,00010
-1/2b	0,16105	-0,00012	0,29758	-0,00009	0,38880	0,00005	0,42084	0,00013
-1/4b	0,26246	0,00008	0,48497	0,00006	0,63365	-0,00003	0,68585	-0,00009
0	0,38866	-0,00426	0,71815	-0,00326	0,93831	0,00177	1,01562	0,00461
1/4b	0,52776	-0,02011	0,97517	-0,01539	1,27412	0,00833	1,37910	0,02177
1/2b	0,63700	-0,04315	1,17702	-0,03302	1,53785	0,01787	1,66456	0,04670
3/4b	0,66278	-0,02130	1,22466	-0,01630	1,60010	0,00882	1,73193	0,02306
b	0,65681	-0,00266	1,21363	-0,00204	1,58569	0,00110	1,71634	0,00288

Tabel 18: Het uitrekenen van a_m

Nu zijn alle gegevens bekend om de factoren voor de zakking te bepalen. De waardes voor a_m voor m = 1 en m = 3 in bovenstaande tabel moeten bij elkaar opgeteld worden en vervolgens vermenigvuldigd worden met de uitdrukking die voor het sommeerteken staat. Dit is gedaan in onderstaande tabel.

e	$w(l/2, b/2) = \frac{l^3}{b\varrho_T\pi^4} \sum_{m=1}^{3} a_m$							
	d/l = 1/8; 7/8		d/l = 1/4; 3/4		d/l = 3/8; 5/8		d/l = 1/2	
	$\sum_{m=1}^{3} a_m$	zakking factoren	$\sum_{m=1}^{3} a_m$	zakking factoren	$\sum_{m=1}^{3} a_m$	zakking factoren	$\sum_{m=1}^{3} a_m$	zakking factoren
-b	0,01424	1,28E-06	0,02639	2,37E-06	0,03458	3,10E-06	0,03747	3,3605E-06
-3/4b	0,08195	7,35E-06	0,15152	1,36E-05	0,19810	1,78E-05	0,21448	1,9236E-05
-1/2b	0,16150	1,45E-05	0,29854	2,68E-05	0,39023	3,50E-05	0,42246	3,7888E-05
-1/4b	0,26278	2,36E-05	0,48547	4,35E-05	0,63418	5,69E-05	0,68638	6,1557E-05
0	0,38430	3,45E-05	0,71471	6,41E-05	0,93984	8,43E-05	1,01997	9,1475E-05
1/4b	0,50723	4,55E-05	0,95901	8,60E-05	1,28145	1,15E-04	1,39978	1,2554E-04
1/2b	0,59320	5,32E-05	1,14279	1,02E-04	1,55415	1,39E-04	1,70956	1,5332E-04
3/4b	0,64071	5,75E-05	1,20694	1,08E-04	1,60707	1,44E-04	1,75298	1,5721E-04
b	0,65333	5,86E-05	1,21008	1,09E-04	1,58481	1,42E-04	1,71707	1,5399E-04

Tabel 19: berekening factoren invloedslijn zakking

De invloedslijn volgens de analytische methode is nu bekend. In de tabel hieronder zijn de zakkingen weergegeven in een matrix.

0	0	0	0	0	0	0	0	0
1,28E-06	7,35E-06	1,45E-05	2,36E-05	3,45E-05	4,55E-05	5,32E-05	5,75E-05	5,86E-05
2,37E-06	1,36E-05	2,68E-05	4,35E-05	6,41E-05	8,60E-05	1,02E-04	1,08E-04	1,09E-04
3,10E-06	1,78E-05	3,50E-05	5,69E-05	8,43E-05	1,15E-04	1,39E-04	1,44E-04	1,42E-04
3,36E-06	1,92E-05	3,79E-05	6,16E-05	9,15E-05	1,26E-04	1,53E-04	1,57E-04	1,54E-04
3,10E-06	1,78E-05	3,50E-05	5,69E-05	8,43E-05	1,15E-04	1,39E-04	1,44E-04	1,42E-04
2,37E-06	1,36E-05	2,68E-05	4,35E-05	6,41E-05	8,60E-05	1,02E-04	1,08E-04	1,09E-04
1,28E-06	7,35E-06	1,45E-05	2,36E-05	3,45E-05	4,55E-05	5,32E-05	5,75E-05	5,86E-05
0	0	0	0	0	0	0	0	0

Tabel 20: Invloedslijn zakking in matrix

De waardes berekend met Python en met de analytische methode kunnen nu vergeleken worden.

0	0	0	0	0	0	0	0	0
2,81E-05	3,00E-05	3,20E-05	3,43E-05	3,69E-05	3,94E-05	4,16E-05	4,32E-05	4,45E-05
5,25E-05	5,58E-05	5,95E-05	6,40E-05	6,90E-05	7,41E-05	7,84E-05	8,13E-05	8,34E-05
6,90E-05	7,34E-05	7,84E-05	8,45E-05	9,16E-05	9,91E-05	0,0001053	0,0001089	0,0001111
7,49E-05	7,96E-05	8,51E-05	9,19E-05	1,00E-04	1,09E-04	0,0001166	0,0001197	0,0001216
6,90E-05	7,34E-05	7,84E-05	8,45E-05	9,16E-05	9,91E-05	0,0001053	0,0001089	0,0001111
5,25E-05	5,58E-05	5,95E-05	6,40E-05	6,90E-05	7,41E-05	7,84E-05	8,13E-05	8,34E-05
2,81E-05	3,00E-05	3,20E-05	3,43E-05	3,69E-05	3,94E-05	4,16E-05	4,32E-05	4,45E-05
0	0	0	0	0	0	0	0	0

Tabel 21: Invloedslijn van Python in matrix

De verschillen zijn in procenten uitgedrukt in onderstaande tabel.

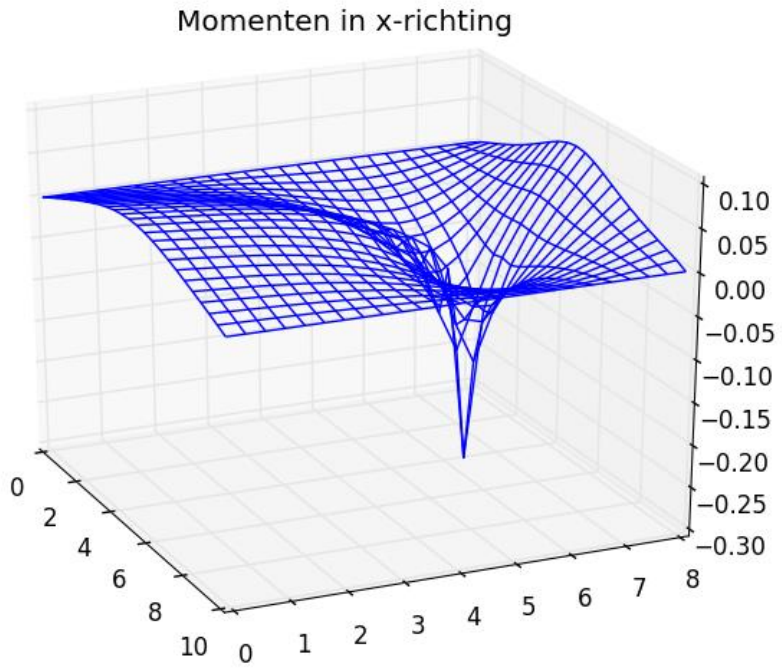
0	0	0	0	0	0	0	0	0
-2103,49	-307,58	-120,67	-45,49	-6,98	13,30	21,75	24,74	24,08
-2118,97	-310,49	-122,30	-46,91	-7,64	13,87	23,55	24,90	23,19
-2126,69	-312,86	-123,90	-48,51	-8,66	13,81	24,47	24,45	21,80
-2129,30	-313,81	-124,60	-49,26	-9,28	13,35	23,94	23,87	21,02
-2126,69	-312,86	-123,90	-48,51	-8,66	13,81	24,47	24,45	21,80
-2118,97	-310,49	-122,30	-46,91	-7,64	13,87	23,55	24,90	23,19
-2103,49	-307,58	-120,67	-45,49	-6,98	13,30	21,75	24,74	24,08
0	0	0	0	0	0	0	0	0

Tabel 22: Verschillen in procenten

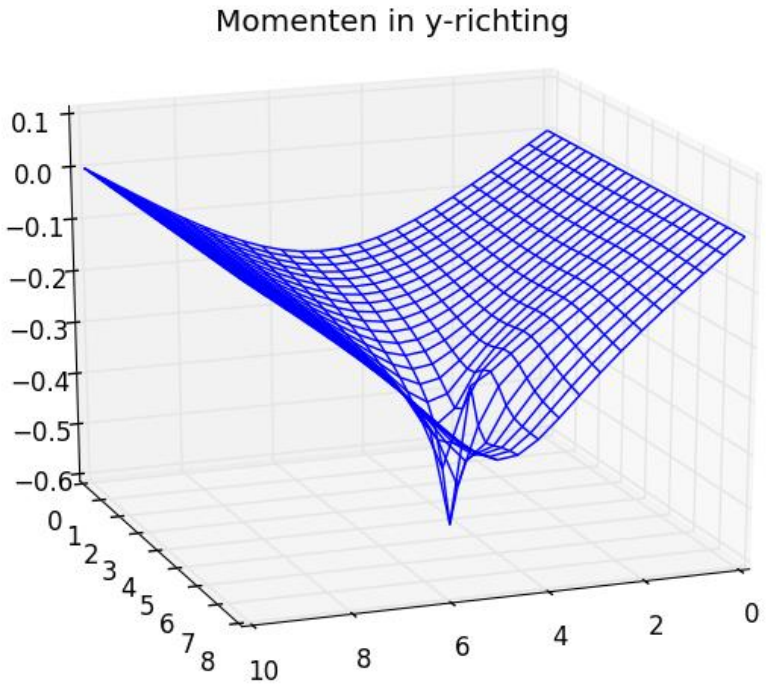
Het moge duidelijk zijn dat er ergens iets mis is gegaan in de berekening, waarschijnlijk in het bepalen van de invloedslijn met de analytische oplossing. Ik heb de fout zelf echter niet kunnen vinden. Verder bevatte dit belastingsgeval, een plaat aan twee zijden scharnierend opgelegd en de andere randen vrij, ook al een foutmarge van 9,3%, wat hoogstwaarschijnlijk door de torsiestijfheid komt.

Overige invloedslijnen

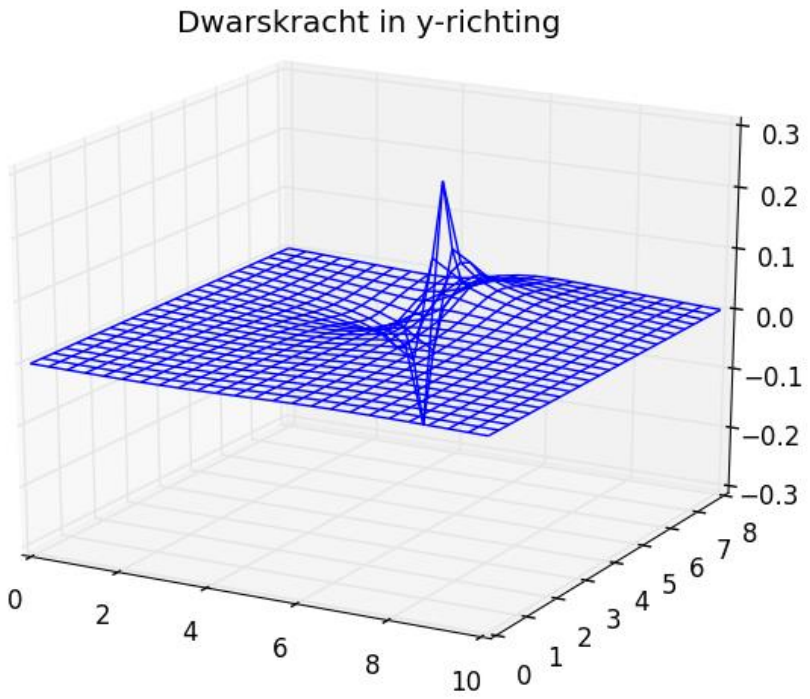
De invloedslijnen voor de momenten en dwarskrachten voor zowel de x als y-richting zijn ook bepaald met het model. Deze zijn echter niet gecontroleerd wegens een gebrek aan tijd. De plots van deze invloedslijnen worden hieronder weergegeven.



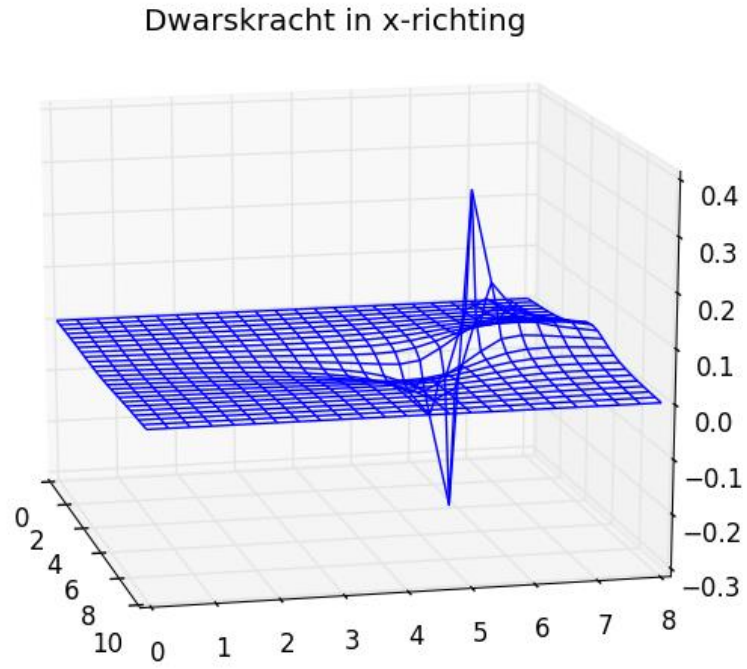
Figuur 11: Invloedslijn voor momenten in x-richting



Figuur 12: Invloedslijn voor momenten in y-richting



Figuur 13: Invloedslijn voor dwarskracht in y-richting



Figuur 14: Invloedslijn voor dwarskracht in x-richting

Conclusie

Ten eerste kan geconcludeerd worden dat, door het model om te schrijven naar Python, de capaciteiten drastisch zijn toegenomen. In dit rapport is meermaals een berekening gemaakt met 100 balkjes in beide richtingen, waar het model in Maple al problemen kreeg bij een hoeveelheid van 20 balkjes. Ook het berekenen van de invloedslijnen met de brute kracht methode verloopt vrij vlot voor modellen rond de 20-30 balkjes in beide richtingen.

De resultaten van het model zonder wringing komen nagenoeg perfect overeen met de al bestaande modellen zoals DIMOD en Femdem. Ook de doorsnedekrachten zijn met succes gecontroleerd in deze situatie.

Op het gebied van de wringing is het model flink verbeterd. Waar de resultaten eerst niet echt convergeerden naar een uiteindelijke oplossing, is deze convergentie nu heel sterk zichtbaar. Ook de resultaten voor de zakkingen kloppen vrij goed voor de volledig ingeklemde plaat. In andere situaties met bijvoorbeeld scharnierende opleggingen reageert het model nog wel te stijf, wat resulteert in lagere zakkingen dan die gevonden worden met de analytische methode.

De invloedslijnen voor de momenten, dwarskrachten en zakkingen zijn ook bepaald. De invloedslijnen voor de momenten en dwarskrachten zijn echter nog niet geverifieerd. Bij het doorlopen van de theorie van de analytische methode voor het bepalen van de invloedslijn voor de zakking is waarschijnlijk een fout gemaakt, wat resulteerde in erg grote verschillen tussen de numerieke en analytisch gevonden invloedslijn.

Aanbevelingen

De verificatie van de momenten en dwarskrachten is nog niet voltooid. Dit is een belangrijke stap alvorens er andere dingen met het model gedaan kunnen worden.

Verder kan er gekeken worden naar de orthotrope plaatwerking. Tot nu toe heeft de nadruk vooral gelegen op isotrope situaties, maar dit gaat in de praktijk lang niet altijd op. Ook kan er gekeken worden naar andere wijze van opleggingen. De plaat hoeft bijvoorbeeld niet langs de gehele rand opgelegd te worden, maar hij kan ook steunen op kolommen. Hiervoor zal het model aangepast moeten worden, zodat individuele knooppunten een oplegging kunnen krijgen.

Ten slotte blijft de torsie van het model een vervelend punt. Zoals bleek uit de verschillende belastingsgevallen neemt de foutmarge toe naarmate de plaat meer vrijheden krijgt (meer scharnierende opleggingen). Wellicht kan er gekeken worden naar een uitdrukking voor de torsiestijfheid die afhankelijk is van welke opleggingen gebruikt zijn.

Referentielijst

Bares, R. (1968). *Analysis of beam grids and orthotropic plates by the Guyon-Massonnet-Bares method.*

Aarts, A. (2016) *Modellering invloedslijnen platen.*

Oomen, S. (2012) *Vlakke plaatvloer modellering.*

Vliet, E. v. (2009). *Discrete modellering van plaatwerking*.

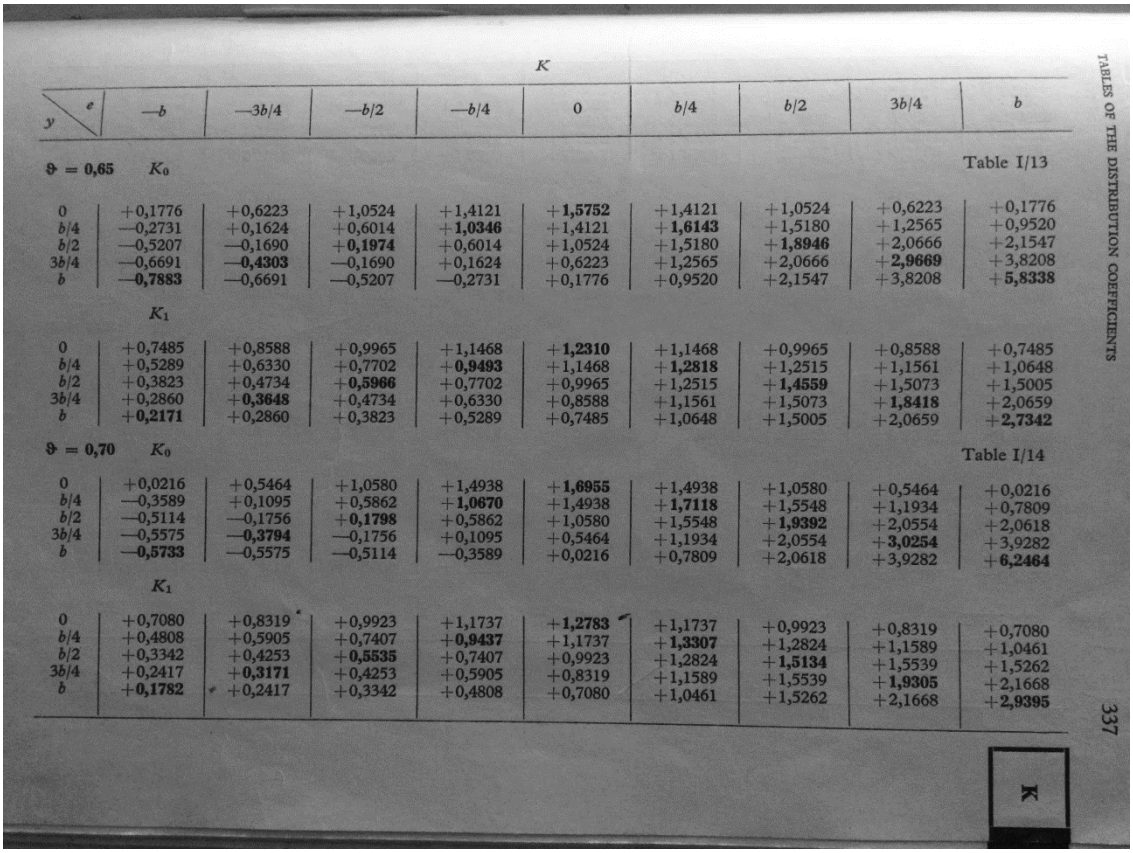
Blaauwendraad, J. (2000) *Eindige-elementenmethode voor staafconstructies.*

S. Timoshenko (1989) *Theory of plates and shells.*

Blaauwendraad, J. (2010) *Plates and FEM.*

Bijlage

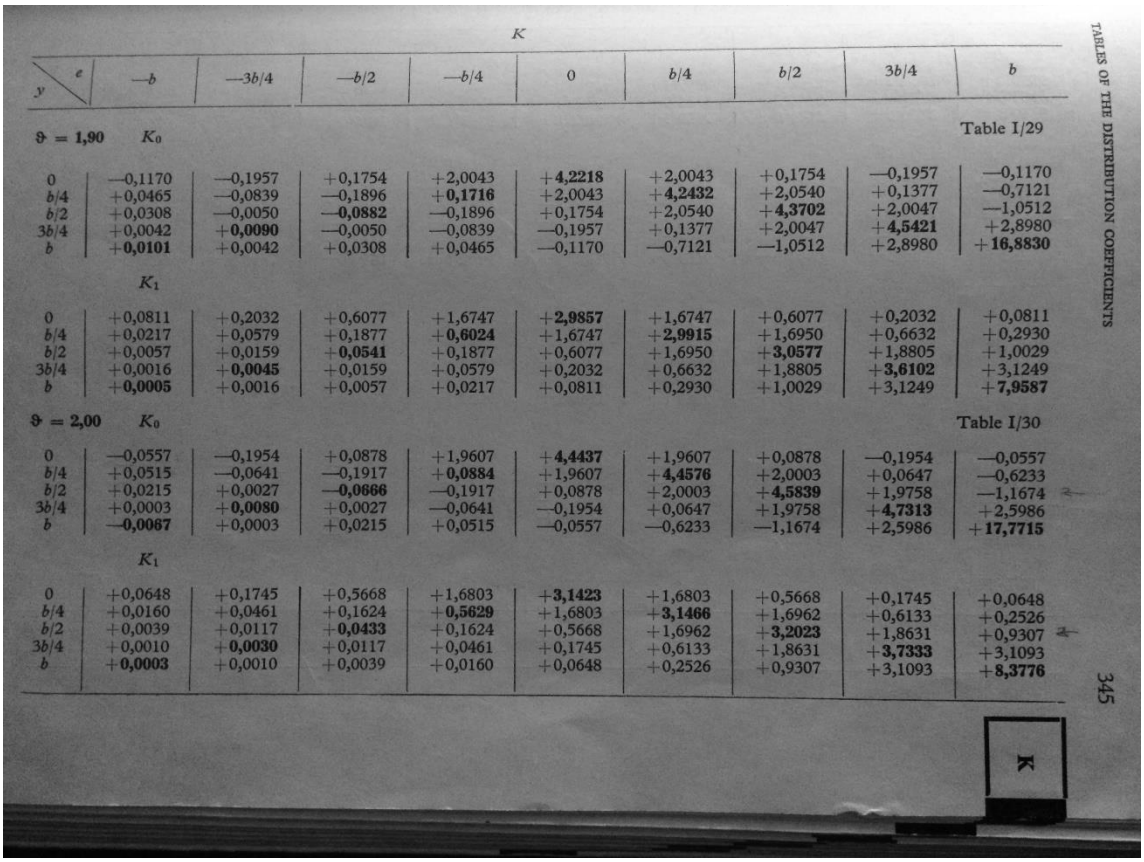
Tabellen van Guyon-Massonnet



TABLES OF THE DISTRIBUTION COEFFICIENTS 337

K

y \ e	−b	−3b/4	−b/2	−b/4	0	b/4	b/2	3b/4	b
ϑ = 0,65	K_0								Table I/13
0	+0,1776	+0,6223	+1,0524	+1,4121	**+1,5752**	+1,4121	+1,0524	+0,6223	+0,1776
b/4	−0,2731	+0,1624	+0,6014	**+1,0346**	+1,4121	**+1,6143**	+1,5180	+1,2565	+0,9520
b/2	−0,5207	−0,1690	**+0,1974**	+0,6014	+1,0524	+1,5180	**+1,8946**	+2,0666	+2,1547
3b/4	−0,6691	**−0,4303**	−0,1690	+0,1624	+0,6223	+1,2565	+2,0666	**+2,9669**	+3,8208
b	**−0,7883**	−0,6691	−0,5207	−0,2731	+0,1776	+0,9520	+2,1547	+3,8208	**+5,8338**
	K_1								
0	+0,7485	+0,8588	+0,9965	+1,1468	**+1,2310**	+1,1468	+0,9965	+0,8588	+0,7485
b/4	+0,5289	+0,6330	+0,7702	**+0,9493**	+1,1468	**+1,2818**	+1,2515	+1,1561	+1,0648
b/2	+0,3823	+0,4734	**+0,5966**	+0,7702	+0,9965	+1,2515	**+1,4559**	+1,5073	+1,5005
3b/4	+0,2860	**+0,3648**	+0,4734	+0,6330	+0,8588	+1,1561	+1,5073	**+1,8418**	+2,0659
b	**+0,2171**	+0,2860	+0,3823	+0,5289	+0,7485	+1,0648	+1,5005	+2,0659	**+2,7342**
ϑ = 0,70	K_0								Table I/14
0	+0,0216	+0,5464	+1,0580	+1,4938	**+1,6955**	+1,4938	+1,0580	+0,5464	+0,0216
b/4	−0,3589	+0,1095	+0,5862	**+1,0670**	+1,4938	**+1,7118**	+1,5548	+1,1934	+0,7809
b/2	−0,5114	−0,1756	**+0,1798**	+0,5862	+1,0580	+1,5548	**+1,9392**	+2,0554	+2,0618
3b/4	−0,5575	**−0,3794**	−0,1756	+0,1095	+0,5464	+1,1934	+2,0554	**+3,0254**	+3,9282
b	**−0,5733**	−0,5575	−0,5114	−0,3589	+0,0216	+0,7809	+2,0618	+3,9282	**+6,2464**
	K_1								
0	+0,7080	+0,8319	+0,9923	+1,1737	**+1,2783**	+1,1737	+0,9923	+0,8319	+0,7080
b/4	+0,4808	+0,5905	+0,7407	**+0,9437**	+1,1737	**+1,3307**	+1,2824	+1,1589	+1,0461
b/2	+0,3342	+0,4253	**+0,5535**	+0,7407	+0,9923	+1,2824	**+1,5134**	+1,5539	+1,5262
3b/4	+0,2417	**+0,3171**	+0,4253	+0,5905	+0,8319	+1,1589	+1,5539	**+1,9305**	+2,1668
b	**+0,1782**	+0,2417	+0,3342	+0,4808	+0,7080	+1,0461	+1,5262	+2,1668	**+2,9395**



TABLES OF THE DISTRIBUTION COEFFICIENTS 345

K

y \ e	−b	−3b/4	−b/2	−b/4	0	b/4	b/2	3b/4	b
ϑ = 1,90	K_0								Table I/29
0	−0,1170	−0,1957	+0,1754	+2,0043	**+4,2218**	+2,0043	+0,1754	−0,1957	−0,1170
b/4	+0,0465	−0,0839	−0,1896	**+0,1716**	+2,0043	**+4,2432**	+2,0540	+0,1377	−0,7121
b/2	+0,0308	−0,0050	**−0,0882**	−0,1896	+0,1754	+2,0540	**+4,3702**	+2,0047	−1,0512
3b/4	+0,0042	**+0,0090**	−0,0050	−0,0839	−0,1957	+0,1377	+2,0047	**+4,5421**	+2,8980
b	**+0,0101**	+0,0042	+0,0308	+0,0465	−0,1170	−0,7121	−1,0512	+2,8980	**+16,8830**
	K_1								
0	+0,0811	+0,2032	+0,6077	+1,6747	**+2,9857**	+1,6747	+0,6077	+0,2032	+0,0811
b/4	+0,0217	+0,0579	+0,1877	**+0,6024**	+1,6747	**+2,9915**	+1,6950	+0,6632	+0,2930
b/2	+0,0057	+0,0159	**+0,0541**	+0,1877	+0,6077	+1,6950	**+3,0577**	+1,8805	+1,0029
3b/4	+0,0016	**+0,0045**	+0,0159	+0,0579	+0,2032	+0,6632	+1,8805	**+3,6102**	+3,1249
b	**+0,0005**	+0,0016	+0,0057	+0,0217	+0,0811	+0,2930	+1,0029	+3,1249	**+7,9587**
ϑ = 2,00	K_0								Table I/30
0	−0,0557	−0,1954	+0,0878	+1,9607	**+4,4437**	+1,9607	+0,0878	−0,1954	−0,0557
b/4	+0,0515	−0,0641	−0,1917	**+0,0884**	+1,9607	**+4,4576**	+2,0003	+0,0647	−0,6233
b/2	+0,0215	+0,0027	**−0,0666**	−0,1917	+0,0878	+2,0003	**+4,5839**	+1,9758	−1,1674
3b/4	+0,0003	**+0,0080**	+0,0027	−0,0641	−0,1954	+0,0647	+1,9758	**+4,7313**	+2,5986
b	**−0,0067**	+0,0003	+0,0215	+0,0515	−0,0557	−0,6233	−1,1674	+2,5986	**+17,7715**
	K_1								
0	+0,0648	+0,1745	+0,5668	+1,6803	**+3,1423**	+1,6803	+0,5668	+0,1745	+0,0648
b/4	+0,0160	+0,0461	+0,1624	**+0,5629**	+1,6803	**+3,1466**	+1,6962	+0,6133	+0,2526
b/2	+0,0039	+0,0117	**+0,0433**	+0,1624	+0,5668	+1,6962	**+3,2023**	+1,8631	+0,9307
3b/4	+0,0010	**+0,0030**	+0,0117	+0,0461	+0,1745	+0,6133	+1,8631	**+3,7333**	+3,1093
b	**+0,0003**	+0,0010	+0,0039	+0,0160	+0,0648	+0,2526	+0,9307	+3,1093	**+8,3776**

K

y \ e	−b	−3b/4	−b/2	−b/4	0	b/4	b/2	3b/4	b
ϑ = 2,20	K_0								Table I/31
0	+0,0256	−0,1771	−0,0585	+1,8454	**+4,8880**	+1,8454	−0,0585	−0,1771	+0,0256
b/4	+0,0428	−0,0299	−0,1739	**−0,0536**	+1,8454	**+4,8925**	+1,8660	−0,0662	−0,4333
b/2	+0,0064	+0,0096	**−0,0300**	−0,1739	−0,0585	+1,8660	**+5,0036**	+1,8857	−1,3007
3b/4	−0,0026	**+0,0046**	+0,0096	−0,0299	−0,1771	−0,0662	+1,8857	**+5,1381**	+1,9700
b	**−0,0013**	−0,0026	+0,0064	+0,0428	+0,0256	−0,4333	−1,3007	+1,9700	**+19,5487**
	K_1								
0	+0,0409	+0,1275	+0,4883	+1,6754	**+3,4560**	+1,6754	+0,4883	+0,1275	+0,0409
b/4	+0,0087	+0,0289	+0,1204	**+0,4863**	+1,6754	**+3,4583**	+1,6849	+0,5202	+0,1856
b/2	+0,0018	+0,0063	**+0,0274**	+0,1204	+0,4883	+1,6849	**+3,4968**	+1,8177	+0,7934
3b/4	+0,0004	**+0,0014**	+0,0063	+0,0289	+0,1275	+0,5202	+1,8177	**+3,9803**	+3,0517
b	**+0,0001**	+0,0004	+0,0018	+0,0087	+0,0409	+0,1856	+0,7934	+3,0517	**+9,2153**
ϑ = 2,40	K_0								Table I/32
0	+0,0599	−0,1431	−0,1650	+1,6976	**+5,3321**	+1,6976	−0,1650	−0,1431	+0,0599
b/4	+0,0252	−0,0060	−0,1391	**−0,1600**	+1,6976	**+5,3330**	+1,7048	−0,1696	−0,2561
b/2	−0,0011	+0,0093	**−0,0056**	−0,1391	−0,1650	+1,7048	**+5,4190**	+1,7538	−1,3184
3b/4	−0,0020	**+0,0016**	+0,0093	−0,0060	−0,1431	−0,1696	+1,7538	**+5,5736**	+1,3256
b	**+0,0006**	−0,0020	−0,0011	+0,0252	+0,0599	−0,2561	−1,3184	+1,3256	**+21,3258**
	K_1								
0	+0,0255	+0,0922	+0,4157	+1,6516	**+3,7700**	+1,6516	+0,4157	+0,0922	+0,0255
b/4	+0,0046	+0,0179	+0,0880	**+0,4146**	+1,6516	**+3,7711**	+1,6572	+0,4372	+0,1347
b/2	+0,0008	+0,0033	**+0,0172**	+0,0880	+0,4157	+1,6572	**+3,7973**	+1,7608	+0,6686
3b/4	+0,0001	**+0,0006**	+0,0033	+0,0179	+0,0922	+0,4372	+1,7608	**+4,2298**	+2,9650
b	**0**	+0,0001	+0,0008	+0,0046	+0,0255	+0,1347	+0,6686	+2,9650	**+10,0531**

Python code

Systeemmatrix

```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# m = 40
n = m = 12
knpM = 2
h = 0.25
Ex = 20*10**6
Ey = 20*10**6
qx = 0
qy = 0
Lx = 5
Ly = 5
F = 0
q = 100

dx = Lx/n
dy = Ly/m
Izz_x = 1/12 * dy * h**3
Izz_y = 1/12 * dx * h**3
# EIzz_x = Ex * Izz_x
# EIzz_y = Ey * Izz_y
EIzz x = 20000
EIzz_y = 20000
Iwx = (dy * h ** 3) / 3
Iwy = (dx * h ** 3) / 3
# reductiefactortorsie = 0.0625/(0.1 * Lx)
# GIwy = Ey/(2 * (1 + 0.3)) * Iwy * reductiefactortorsie
GIwy = Ey/(2 * (1 + 0.3)) * Iwy * 1 * 0
# GIwx = Ex/(2 * (1 + 0.3)) * Iwx * reductiefactortorsie
GIwx = Ex/(2 * (1 + 0.3)) * Iwx * 1 * 0
"""
1 staat voor inklemming
2 staat voor inklemmingX
3 staat voor inklemmingY
4 staat voor scharnier
"""
rand1 = 1
rand2 = 0
rand3 = 1
rand4 = 0

support = np.zeros(((3 * (n + 1) * (m + 1)), 1))
knoop = np.zeros(((n + 1) * (m + 1), 1))
R = np.zeros(((n + 1) * (m + 1), 1))
My = np.zeros(((n + 1) * (m + 1), 1))
Mx = np.zeros(((n + 1) * (m + 1), 1))

for t in range(1, ((n + 1) * (m + 1) + 1)):
    if rand1 == 1:  # Volledige inklemming
        for r in range(1, ((n + 1) + 1)):
            knoop[r-1] = 1
            R[r-1] = 1
            My[r-1] = 1
            Mx[r-1] = 1
    elif rand1 == 2:  # InklemmingX
```

```
    for r in range(1, ((n + 1) + 1)):
        knoop[r-1] = 2
        R[r-1] = 1
        My[r-1] = 1
        Mx[r-1] = 0
elif rand1 == 3:  # InklemmingY
    for r in range(1, ((n + 1) + 1)):
        knoop[r-1] = 3
        R[r-1] = 1
        My[r-1] = 0
        Mx[r-1] = 1
elif rand1 == 4:  # Scharnier
    for r in range(1, ((n + 1) + 1)):
        knoop[r-1] = 4
        R[r-1] = 1
        My[r-1] = 0
        Mx[r-1] = 0
if rand2 == 1:  # Volledige inklemming
    for r in range(((n + 1)*m + 1), ((n + 1)*m + 1 + n + 1)):
        knoop[r-1] = 1
        R[r-1] = 1
        My[r-1] = 1
        Mx[r-1] = 1
elif rand2 == 2:  # InklemmingX
    for r in range(((n + 1)*m + 1), ((n + 1)*m + 1 + n + 1)):
        knoop[r-1] = 2
        R[r-1] = 1
        My[r-1] = 1
        Mx[r-1] = 0
elif rand2 == 3:  # InklemmingY
    for r in range(((n + 1)*m + 1), ((n + 1)*m + 1 + n + 1)):
        knoop[r-1] = 3
        R[r-1] = 1
        My[r-1] = 0
        Mx[r-1] = 1
elif rand2 == 4:  # Scharnier
    for r in range(((n + 1)*m + 1), ((n + 1)*m + 1 + n + 1)):
        knoop[r-1] = 4
        R[r-1] = 1
        My[r-1] = 0
        Mx[r-1] = 0
if rand3 == 1:  # Volledige inklemming
    for r in range(1, (m*(n + 1) + 1 + 1), (n + 1)):
        knoop[r - 1] = 1
        R[r - 1] = 1
        My[r - 1] = 1
        Mx[r - 1] = 1
elif rand3 == 2:  # InklemmingX
    for r in range(1, (m*(n + 1) + 1 + 1), (n + 1)):
        knoop[r - 1] = 2
        R[r - 1] = 1
        My[r - 1] = 1
        Mx[r - 1] = 0
elif rand3 == 3:  # InklemmingY
    for r in range(1, (m*(n + 1) + 1 + 1), (n + 1)):
        knoop[r - 1] = 3
        R[r - 1] = 1
        My[r - 1] = 0
        Mx[r - 1] = 1
elif rand3 == 4:  # Scharnier
    for r in range(1, (m*(n + 1) + 1 + 1), (n + 1)):
```

```
            knoop[r - 1] = 4
            R[r - 1] = 1
            My[r - 1] = 0
            Mx[r - 1] = 0
    if rand4 == 1:  # Volledige inklemming
        for r in range((n + 1), ((n + 1)*(m + 1) + 1), (n + 1)):
            knoop[r - 1] = 1
            R[r - 1] = 1
            My[r - 1] = 1
            Mx[r - 1] = 1
    elif rand4 == 2:  # InklemmingX
        for r in range((n + 1), ((n + 1)*(m + 1) + 1), (n + 1)):
            knoop[r - 1] = 2
            R[r - 1] = 1
            My[r - 1] = 1
            Mx[r - 1] = 0
    elif rand4 == 3:  # InklemmingY
        for r in range((n + 1), ((n + 1)*(m + 1) + 1), (n + 1)):
            knoop[r - 1] = 3
            R[r - 1] = 1
            My[r - 1] = 0
            Mx[r - 1] = 1
    elif rand4 == 4:  # Scharnier
        for r in range((n + 1), ((n + 1)*(m + 1) + 1), (n + 1)):
            knoop[r - 1] = 4
            R[r - 1] = 1
            My[r - 1] = 0
            Mx[r - 1] = 0
    support[(t - 1)*3] = R[t - 1]
    support[(t - 1)*3 + 1] = My[t - 1]
    support[(t - 1)*3 + 2] = Mx[t - 1]

C1, C2, C3, C4, x = sp.symbols('C1 C2 C3 C4 x')
w = qx*x**4/(24*EIzz_x)+(1/6*C1*x**3)+(1/2*C2*x**2)+(C3*x)+C4
D1, D2, D3, D4, y = sp.symbols('D1 D2 D3 D4 y')
u = qy*y**4/(24*EIzz_y)+(1/6*D1*y**3)+(1/2*D2*y**2)+(D3*y)+D4

phiy = -sp.diff(w, x)
phix = -sp.diff(u, y)
kappay = sp.diff(phiy, x)
kappaz = sp.diff(phix, y)

My1 = EIzz_x*kappay
Mx1 = EIzz_y*kappaz
Vy = sp.diff(My1, x)
Vz = sp.diff(Mx1, y)

wA, wB, uA, uB, phiyA, phiyB, phixA, phixB = sp.symbols('wA wB uA uB phiyA
phiyB phixA phixB')
eq1 = sp.Eq(w.subs(x, 0), wA)
eq2 = sp.Eq(phiy.subs(x, 0), phiyA)
eq3 = sp.Eq(w.subs(x, dx), wB)
eq4 = sp.Eq(phiy.subs(x, dx), phiyB)
sol1 = (sp.solve([eq1, eq2, eq3, eq4], [C1, C2, C3, C4]))
FxA = sp.simplify(-Vy.subs({x: 0, C1: sol1[C1]}))
TyA = sp.simplify(-My1.subs({x: 0, C1: sol1[C1], C2: sol1[C2]}))
FxB = sp.simplify(Vy.subs({x: dx, C1: sol1[C1]}))
TyB = sp.simplify(My1.subs({x: dx, C1: sol1[C1], C2: sol1[C2]}))
eq5 = sp.Eq(u.subs(y, 0), uA)
eq6 = sp.Eq(phix.subs(y, 0), phixA)
eq7 = sp.Eq(u.subs(y, dy), uB)
```

```
eq8 = sp.Eq(phix.subs(y, dy), phixB)
sol2 = (sp.solve([eq5, eq6, eq7, eq8], [D1, D2, D3, D4]))
FyA = sp.simplify(-Vz.subs({y: 0, D1: sol2[D1]}))
TxA = sp.simplify(-Mx1.subs({y: 0, D1: sol2[D1], D2: sol2[D2]}))
FyB = sp.simplify(Vz.subs({y: dy, D1: sol2[D1]}))
TxB = sp.simplify(Mx1.subs({y: dy, D1: sol2[D1], D2: sol2[D2]}))

k11 = sp.expand(FxA).coeff(wA, 1)
k12 = sp.expand(FxA).coeff(phiyA, 1)
k13 = 0
k14 = sp.expand(FxA).coeff(wB, 1)
k15 = sp.expand(FxA).coeff(phiyB, 1)
k16 = 0
k21 = sp.expand(TyA).coeff(wA, 1)
k22 = sp.expand(TyA).coeff(phiyA, 1)
k23 = 0
k24 = sp.expand(TyA).coeff(wB, 1)
k25 = sp.expand(TyA).coeff(phiyB, 1)
k26 = 0
k31 = 0
k32 = 0
k33 = GIwx/dx
k34 = 0
k35 = 0
k36 = -GIwx/dx
k41 = sp.expand(FxB).coeff(wA, 1)
k42 = sp.expand(FxB).coeff(phiyA, 1)
k43 = 0
k44 = sp.expand(FxB).coeff(wB, 1)
k45 = sp.expand(FxB).coeff(phiyB, 1)
k46 = 0
k51 = sp.expand(TyB).coeff(wA, 1)
k52 = sp.expand(TyB).coeff(phiyA, 1)
k53 = 0
k54 = sp.expand(TyB).coeff(wB, 1)
k55 = sp.expand(TyB).coeff(phiyB, 1)
k56 = 0
k61 = 0
k62 = 0
k63 = -GIwx/dx
k64 = 0
k65 = 0
k66 = GIwx/dx

Kelemx = np.array([[k11, k12, k13, k14, k15, k16], [k21, k22, k23, k24,
k25, k26],
                   [k31, k32, k33, k34, k35, k36], [k41, k42, k43, k44,
k45, k46],
                   [k51, k52, k53, k54, k55, k56], [k61, k62, k63, k64,
k65, k66]])

Kelemxrand = Kelemx*0.5
Kelemxrand[2, 2] = 0
Kelemxrand[2, 5] = 0
Kelemxrand[5, 2] = 0
Kelemxrand[5, 5] = 0

k11 = sp.expand(FyA).coeff(uA, 1)
k12 = 0
k13 = sp.expand(FyA).coeff(phixA, 1)
k14 = sp.expand(FyA).coeff(uB, 1)
```

```
k15 = 0
k16 = sp.expand(FyA).coeff(phixB, 1)
k21 = 0
k22 = GIwy/dy
k23 = 0
k24 = 0
k25 = -GIwy/dy
k26 = 0
k31 = sp.expand(TxA).coeff(uA, 1)
k32 = 0
k33 = sp.expand(TxA).coeff(phixA, 1)
k34 = sp.expand(TxA).coeff(uB, 1)
k35 = 0
k36 = sp.expand(TxA).coeff(phixB, 1)
k41 = sp.expand(FyB).coeff(uA, 1)
k42 = 0
k43 = sp.expand(FyB).coeff(phixA, 1)
k44 = sp.expand(FyB).coeff(uB, 1)
k45 = 0
k46 = sp.expand(FyB).coeff(phixB, 1)
k51 = 0
k52 = -GIwy/dy
k53 = 0
k54 = 0
k55 = GIwy/dy
k56 = 0
k61 = sp.expand(TxB).coeff(uA, 1)
k62 = 0
k63 = sp.expand(TxB).coeff(phixA, 1)
k64 = sp.expand(TxB).coeff(uB, 1)
k65 = 0
k66 = sp.expand(TxB).coeff(phixB, 1)
Kelemy = np.array([[k11, k12, k13, k14, k15, k16], [k21, k22, k23, k24,
k25, k26],
                   [k31, k32, k33, k34, k35, k36], [k41, k42, k43, k44,
k45, k46],
                   [k51, k52, k53, k54, k55, k56], [k61, k62, k63, k64,
k65, k66]])

Kelemyrand = Kelemy*0.5
Kelemyrand[1, 1] = 0
Kelemyrand[1, 4] = 0
Kelemyrand[4, 1] = 0
Kelemyrand[4, 4] = 0

if n > 0:
    fx1 = -(sp.expand(FxA).coeff(qx, 1)*qx) + 0.25*q*dx*dy
    fx2 = -sp.expand(TyA).coeff(qx, 1)*qx
    fx3 = 0
    fx4 = -(sp.expand(FxB).coeff(qx, 1)*qx) + 0.25*q*dx*dy
    fx5 = -sp.expand(TyB).coeff(qx, 1)*qx
    fx6 = 0
elif n == 0:
    fx1 = 0
    fx2 = 0
    fx3 = 0
    fx4 = 0
    fx5 = 0
    fx6 = 0
fsysx = np.array([[fx1], [fx2], [fx3], [fx4], [fx5], [fx6]])
fsysxrand = 0.5*fsysx
```

```
if m > 0:
    fy1 = -(sp.expand(FyA).coeff(qy, 1)*qy) + 0.25*q*dx*dy
    fy2 = 0
    fy3 = -sp.expand(TxA).coeff(qy, 1)*qy
    fy4 = -(sp.expand(FyB).coeff(qy, 1)*qy) + 0.25*q*dx*dy
    fy5 = 0
    fy6 = -sp.expand(TxB).coeff(qy, 1)*qy
elif m == 0:
    fy1 = 0
    fy2 = 0
    fy3 = 0
    fy4 = 0
    fy5 = 0
    fy6 = 0
fsysy = np.array([[fy1], [fy2], [fy3], [fy4], [fy5], [fy6]])
fsysyrand = 0.5*fsysy

belastingsvector = np.zeros(((3 * (n + 1) * (m + 1)), 1), dtype=float)
u = np.zeros(((3 * (n + 1) * (m + 1)), 1))
dof = np.zeros((6, 1), dtype=int)
Ksys = np.zeros((((n + 1)*(m + 1)*3), ((n + 1)*(m + 1)*3)), dtype=float)

for j in range(1, (m + 1 + 1)):
    for i in range(1, n + 1):
        knoop1 = (n + 1)*(j - 1) + i
        knoop2 = knoop1 + 1
        dof[0] = (knoop1 - 1)*3 + 1
        dof[1] = dof[0] + 1
        dof[2] = dof[1] + 1
        dof[3] = (knoop2 - 1)*3 + 1
        dof[4] = dof[3] + 1
        dof[5] = dof[4] + 1
        if j == 1:
            for k in range(1, 7):
                belastingsvector[(dof[k - 1]-1)] = belastingsvector[(dof[k
- 1]-1)] + fsysxrand[k - 1]
                for l in range(1, 7):
                    Ksys[(dof[k - 1]-1), (dof[l - 1]-1)] = Ksys[(dof[k -
1]-1), (dof[l - 1]-1)] + Kelemxrand[(k - 1),

(l - 1)]
        elif j == (m + 1):
            for k in range(1, 7):
                belastingsvector[(dof[k - 1]-1)] = belastingsvector[(dof[k
- 1]-1)] + fsysxrand[k - 1]
                for l in range(1, 7):
                    Ksys[(dof[k - 1]-1), (dof[l - 1]-1)] = Ksys[(dof[k -
1]-1), (dof[l - 1]-1)] + Kelemxrand[(k - 1),

(l - 1)]
        else:
            for k in range(1, 7):
                belastingsvector[(dof[k - 1]-1)] = belastingsvector[(dof[k
- 1]-1)] + fsysx[k - 1]
                for l in range(1, 7):
                    Ksys[(dof[k - 1]-1), (dof[l - 1]-1)] = Ksys[(dof[k -
1]-1), (dof[l - 1]-1)] + Kelemx[(k - 1),

(l - 1)]
```

```
for i in range(1, (n + 1 + 1)):
    for j in range(1, m + 1):
        knoop1 = (n + 1)*(j - 1) + i
        knoop2 = knoop1 + n + 1
        dof[0] = (knoop1 - 1)*3 + 1
        dof[1] = dof[0] + 1
        dof[2] = dof[1] + 1
        dof[3] = (knoop2 - 1)*3 + 1
        dof[4] = dof[3] + 1
        dof[5] = dof[4] + 1
        if i == 1:
            for k in range(1, 7):
                belastingsvector[(dof[k - 1]-1)] = belastingsvector[(dof[k
- 1]-1)] + fsysyrand[k - 1]
                for l in range(1, 7):
                    Ksys[(dof[k - 1]-1), (dof[l - 1]-1)] = Ksys[(dof[k -
1]-1), (dof[l - 1]-1)] + Kelemyrand[(k - 1),

(l - 1)]
        elif i == (n + 1):
            for k in range(1, 7):
                belastingsvector[(dof[k - 1]-1)] = belastingsvector[(dof[k
- 1]-1)] + fsysxrand[k - 1]
                for l in range(1, 7):
                    Ksys[(dof[k - 1]-1), (dof[l - 1]-1)] = Ksys[(dof[k -
1]-1), (dof[l - 1]-1)] + Kelemyrand[(k - 1),

(l - 1)]
        else:
            for k in range(1, 7):
                belastingsvector[(dof[k - 1]-1)] = belastingsvector[(dof[k
- 1]-1)] + fsysy[k - 1]
                for l in range(1, 7):
                    Ksys[(dof[k - 1]-1), (dof[l - 1]-1)] = Ksys[(dof[k -
1]-1), (dof[l - 1]-1)] + Kelemy[(k - 1),

(l - 1)]

check = np.zeros((((n + 1)*(m + 1)), 1), dtype=object)
for t in range(1, ((m + 1)*(n + 1) + 1)):
    check[t - 1] = belastingsvector[(t - 1)*3]
```

Plaatmodel

```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
from SysteemMatrix import Ksys, belastingsvector, m, n, knpM, support,
GIwx, GIwy, TyA, TxA, TyB, TxB, Lx, Ly, wA, wB, phiyA, phiyB, uA, uB,
phixA, phixB, dy, dx, R

Ksystijdelijk = np.copy(Ksys)
belastingsvectortijdelijk = np.copy(belastingsvector)
for i in range(1, ((n + 1)*(m + 1)*3 + 1)):
    if support[i - 1] == 1:
        Ksystijdelijk[(i - 1), :] = 0
        Ksystijdelijk[:, (i - 1)] = 0
        Ksystijdelijk[(i - 1), (i - 1)] = 1
        belastingsvectortijdelijk[i - 1] = 0
    elif Ksystijdelijk[(i - 1), (i - 1)] == 0:
        Ksystijdelijk[(i - 1), (i - 1)] = 1

u = np.linalg.solve(Ksystijdelijk, belastingsvectortijdelijk)

w_ = np.zeros((((n + 1)*(m + 1)), 1))
phiy = np.zeros((((n + 1)*(m + 1)), 1))
phix = np.zeros((((n + 1)*(m + 1)), 1))
for t in range(1, ((m + 1)*(n + 1) + 1)):
    w_[t - 1] = u[(t - 1)*3]
    phiy[t - 1] = u[(t - 1)*3 + 1]
    phix[t - 1] = u[(t - 1)*3 + 2]
results = np.zeros((((n + 1)*(m + 1)), 4))
for i in range(1, ((n + 1)*(m + 1) + 1)):
    results[(i - 1), 0] = i
    results[(i - 1), 1] = w_[i - 1]
    results[(i - 1), 2] = phiy[i - 1]
    results[(i - 1), 3] = phix[i - 1]
zakking = np.zeros(((m + 1), (n + 1)))
phiyMatrix = np.zeros(((m + 1), (n + 1)))
phixMatrix = np.zeros(((m + 1), (n + 1)))
for j in range(1, ((m + 1) + 1)):
    for i in range(1, ((n + 1) + 1)):
        zakking[(j - 1), (i - 1)] = (w_[(j - 1)*(n + 1) + i - 1])
        phiyMatrix[(j - 1), (i - 1)] = phiy[(j - 1) * (n + 1) + i - 1]
        phixMatrix[(j - 1), (i - 1)] = phix[(j - 1) * (n + 1) + i - 1]

MyLinksMatrix = np.zeros(((m + 1), (n + 1)))
MyRechtsMatrix = np.zeros(((m + 1), (n + 1)))
MxBovenMatrix = np.zeros(((m + 1), (n + 1)))
MxOnderMatrix = np.zeros(((m + 1), (n + 1)))
TyAW = -TyA
TxAW = -TxA
for j in range(1, ((m + 1) + 1)):
    for i in range(1, ((n + 1) + 1)):
        if i == (n + 1):
            MyRechtsMatrix[(j - 1), (i - 1)] = (m/Ly)*(TyB.subs({wA:
zakking[j - 1, i - 2], wB: zakking[j - 1, i - 1],
                                                phiyA: phiyMatrix[j - 1, i
- 2], phiyB: phiyMatrix[j - 1, i - 1]}))
            MyLinksMatrix[(j - 1), (i - 1)] = (m/Ly)*(TyB.subs({wA:
zakking[j - 1, i - 2], wB: zakking[j - 1, i - 1],
                                                phiyA: phiyMatrix[j - 1, i
```

```
- 2], phiyB: phiyMatrix[j - 1, i - 1]}))
        if i == 1:
            MyRechtsMatrix[(j - 1), (i - 1)] = (m/Ly)*(TyAW.subs({wA:
zakking[j - 1, i - 1], wB: zakking[j - 1, i],
                                                    phiyA: phiyMatrix[j - 1, i
- 1], phiyB: phiyMatrix[j - 1, i]}))
            MyLinksMatrix[(j - 1), (i - 1)] = (m/Ly)*(TyAW.subs({wA:
zakking[j - 1, i - 1], wB: zakking[j - 1, i],
                                                    phiyA: phiyMatrix[j - 1, i
- 1], phiyB: phiyMatrix[j - 1, i]}))
        if 1 < i < (n + 1):
            MyRechtsMatrix[(j - 1), (i - 1)] = (m/Ly)*(TyAW.subs({wA:
zakking[j - 1, i - 1], wB: zakking[j - 1, i],
                                                    phiyA: phiyMatrix[j - 1, i
- 1], phiyB: phiyMatrix[j - 1, i]}))
            MyLinksMatrix[(j - 1), (i - 1)] = (m/Ly)*(TyB.subs({wA:
zakking[j - 1, i - 2], wB: zakking[j - 1, i - 1],
                                                    phiyA: phiyMatrix[j - 1, i
- 2], phiyB: phiyMatrix[j - 1, i - 1]}))
for j in range(1, ((n + 1) + 1)):
    for i in range(1, ((m + 1) + 1)):
        if i == (m + 1):
            MxOnderMatrix[i - 1, j - 1] = (n/Lx)*(TxB.subs({uA: zakking[i -
2, j - 1], uB: zakking[i - 1, j - 1],
                                                    phixA: phixMatrix[i - 2, j
- 1], phixB: phixMatrix[i - 1, j - 1]}))
            MxBovenMatrix[i - 1, j - 1] = (n/Lx)*(TxB.subs({uA: zakking[i -
2, j - 1], uB: zakking[i - 1, j - 1],
                                                    phixA: phixMatrix[i - 2, j
- 1], phixB: phixMatrix[i - 1, j - 1]}))
        if i == 1:
            MxOnderMatrix[i - 1, j - 1] = (n/Lx)*(TxAW.subs({uA: zakking[i
- 1, j - 1], uB: zakking[i, j - 1],
                                                phixA: phixMatrix[i - 1, j -
1], phixB: phixMatrix[i, j - 1]}))
            MxBovenMatrix[i - 1, j - 1] = (n/Lx)*(TxAW.subs({uA: zakking[i
- 1, j - 1], uB: zakking[i, j - 1],
                                                phixA: phixMatrix[i - 1, j -
1], phixB: phixMatrix[i, j - 1]}))
        if 1 < i < (m + 1):
            MxOnderMatrix[i - 1, j - 1] = (n/Lx)*(TxAW.subs({uA: zakking[i
- 1, j - 1], uB: zakking[i, j - 1],
                                                phixA: phixMatrix[i - 1, j -
1], phixB: phixMatrix[i, j - 1]}))
            MxBovenMatrix[i - 1, j - 1] = (n/Lx)*(TxB.subs({uA: zakking[i -
2, j - 1], uB: zakking[i - 1, j - 1],
                                                    phixA: phixMatrix[i - 2, j
- 1], phixB: phixMatrix[i - 1, j - 1]}))

VxMatrix = np.zeros(((m + 1), n))
VyMatrix = np.zeros((m, (n + 1)))
for j in range(1, ((m + 1) + 1)):
    for i in range(1, (n + 1)):
        VxMatrix[(j - 1), (i - 1)] = (MyLinksMatrix[(j - 1), i] -
MyRechtsMatrix[(j - 1), (i - 1)])/dx
for j in range(1, ((n + 1) + 1)):
    for i in range(1, (m + 1)):
        VyMatrix[(i - 1), (j - 1)] = (MxBovenMatrix[i, (j - 1)] -
MxOnderMatrix[(i - 1), (j - 1)])/dy

MyStap = np.zeros(((m + 1), (n*2)))
```

```
MxStap = np.zeros(((m*2), (n + 1)))
VxStap = np.zeros(((m + 1), (n*2)))
VyStap = np.zeros(((m*2), (n + 1)))
for j in range(1, ((m + 1) + 1)):
    for i in range(1, (n + 1)):
        VxStap[(j - 1), (2*i - 1)] = VxMatrix[(j - 1), (i - 1)]
        MyStap[(j - 1), (2*i - 1)] = MyLinksMatrix[(j - 1), i]
for j in range(1, ((m + 1) + 1)):
    for i in range(1, (n + 1)):
        VxStap[(j - 1), (2*i - 2)] = VxMatrix[(j - 1), (i - 1)]
        MyStap[(j - 1), (2*i - 2)] = MyRechtsMatrix[(j - 1), (i - 1)]
for j in range(1, ((n + 1) + 1)):
    for i in range(1, (m + 1)):
        VyStap[(2*i - 1), (j - 1)] = VyMatrix[(i - 1), (j - 1)]
for j in range(1, ((n + 1) + 1)):
    for i in range(1, (m + 1)):
        VyStap[(2*i - 2), (j - 1)] = VyMatrix[(i - 1), (j - 1)]

np.set_printoptions(formatter={'float': '{:g}'.format})

x = np.linspace(0, Lx, (n + 1))
y = np.linspace(0, Ly, (m + 1))
X, Y = np.meshgrid(x, y)
f1 = plt.figure()
ha = f1.add_subplot(111, projection='3d')
ha.plot_wireframe(Y, X, -zakking)
plt.title('Zakking')
plt.figure()
CS = plt.contourf(Y, X, -zakking)
# plt.clabel(CS, inline=1, fontsize=10)
plt.colorbar(CS)
plt.title('Contourplot zakking')

f2 = plt.figure()
ha2 = f2.add_subplot(111, projection='3d')
ha2.plot_wireframe(Y, X, -MyLinksMatrix)
plt.title('Momenten in x-richting')
plt.figure()
CS2 = plt.contourf(Y, X, -MyLinksMatrix)
# plt.clabel(CS, inline=1, fontsize=10)
plt.colorbar(CS2)
plt.title('Contourplot moment in x-richting')
f3 = plt.figure()
ha3 = f3.add_subplot(111, projection='3d')
ha3.plot_wireframe(Y, X, -MxBovenMatrix)
plt.title('Momenten in y-richting')


xVx = np.linspace(0, Lx, n)
yVx = np.linspace(0, Ly, (m + 1))
xVy = np.linspace(0, Lx, (n + 1))
yVy = np.linspace(0, Ly, m)
XVx, YVx = np.meshgrid(xVx, yVx)
XVy, YVy = np.meshgrid(xVy, yVy)
f4 = plt.figure()
ha4 = f4.add_subplot(111, projection='3d')
ha4.plot_wireframe(YVx, XVx, VxMatrix)
plt.title('Dwarskracht in x-richting')
f5 = plt.figure()
ha5 = f5.add_subplot(111, projection='3d')
ha5.plot_wireframe(YVy, XVy, VyMatrix)
```

```
plt.title('Dwarskracht in y-richting')

delta = 0.001
xVxStap = np.zeros((1, (n*2)))
xVyStap = np.linspace(0, Lx, (n + 1))
for j in range(1, (n + 1)):
    if j == n:
        xVxStap[0, (2 * j - 1)] = j*(Lx/n)
    else:
        xVxStap[0, (2 * j - 1)] = (j*(Lx/n) - delta)
for j in range(1, (n + 1)):
    if j == 1:
        xVxStap[0, (2 * j - 2)] = 0
    else:
        xVxStap[0, (2 * j - 2)] = ((j - 1)*(Lx/n) + delta)
yVxStap = np.linspace(0, Ly, (m + 1))
yVyStap = np.zeros((1, (m*2)))
for j in range(1, (m + 1)):
    if j == m:
        yVyStap[0, (2 * j - 1)] = j*(Ly/m)
    else:
        yVyStap[0, (2 * j - 1)] = (j*(Ly/m) - delta)
for j in range(1, (m + 1)):
    if j == 1:
        yVyStap[0, (2 * j - 2)] = 0
    else:
        yVyStap[0, (2 * j - 2)] = ((j - 1)*(Ly/m) + delta)
XVxStap, YVxStap = np.meshgrid(xVxStap, yVxStap)
XVyStap, YVyStap = np.meshgrid(xVyStap, yVyStap)

f5 = plt.figure()
ha5 = f5.add_subplot(111, projection='3d')
ha5.plot_wireframe(YVxStap, XVxStap, VxStap)
plt.title('Dwarskracht in x-richting stapsgewijs')
f6 = plt.figure()
ha6 = f6.add_subplot(111, projection='3d')
ha6.plot_wireframe(YVyStap, XVyStap, VyStap)
plt.title('Dwarskracht in y-richting stapsgewijs')
f7 = plt.figure()
ha7 = f7.add_subplot(111, projection='3d')
ha7.plot_wireframe(YVxStap, XVxStap, -MyStap)
plt.title('Moment in x-richting stapsgewijs')
plt.figure()
CS3 = plt.contourf(YVxStap, XVxStap, VxStap)
# plt.clabel(CS, inline=1, fontsize=10)
plt.colorbar(CS2)
plt.title('Contourplot moment in x-richting')

supportres = Ksys.dot(u) - belastingsvector
Rres = np.zeros((((n + 1)*(m + 1)), 1))
Myres = np.zeros((((n + 1)*(m + 1)), 1))
Mxres = np.zeros((((n + 1)*(m + 1)), 1))

for t in range(1, ((n + 1)*(m + 1) + 1)):
    Rres[t - 1] = supportres[(t - 1) * 3]
    if R[t - 1] == 0:
        Rres[t - 1] = 0
RresMatrix = np.zeros(((m + 1), (n + 1)))

for j in range(1, ((m + 1) + 1)):
    for i in range(1, ((n + 1) + 1)):
```

```
        RresMatrix[(j - 1), (i - 1)] = Rres[(j - 1) * (n + 1) + i - 1]

plt.show()
```

BruteForceMethode

```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from SysteemMatrix import n, m, Ksys, support, Ly, Lx, dx, dy, TyA, TyB,
TxA, TxB, wA, wB, uA, uB, phiyA, phiyB, phixA, phixB

# Het punt waarvoor de invloedslijnen bepaald moeten worden
kpnr = 319
w_ = np.zeros((((n + 1)*(m + 1)), 1))
phiy = np.zeros((((n + 1)*(m + 1)), 1))
phix = np.zeros((((n + 1)*(m + 1)), 1))
Mylinks = np.zeros((((n + 1)*(m + 1)), 1))
Myrechts = np.zeros((((n + 1)*(m + 1)), 1))
Mxboven = np.zeros((((n + 1)*(m + 1)), 1))
Mxonder = np.zeros((((n + 1)*(m + 1)), 1))
Vlinks = np.zeros((((n + 1)*(m + 1)), 1))
Vrechts = np.zeros((((n + 1)*(m + 1)), 1))
Vboven = np.zeros((((n + 1)*(m + 1)), 1))
Vonder = np.zeros((((n + 1)*(m + 1)), 1))

TyAW = -TyA
TxAW = -TxA

for j in range(1, ((m + 1)*(n + 1) + 1)):
    belastingsvector = np.zeros(((3 * (n + 1) * (m + 1)), 1))
    belastingsvector[((j - 1)*3), 0] = 1
    belastingsvectortijdelijk = np.copy(belastingsvector)
    Ksystijdelijk = np.copy(Ksys)
    for i in range(1, ((n + 1)*(m + 1)*3 + 1)):
        if support[i - 1] == 1:
            Ksystijdelijk[(i - 1), :] = 0
            Ksystijdelijk[:, (i - 1)] = 0
            Ksystijdelijk[(i - 1), (i - 1)] = 1
            belastingsvectortijdelijk[i - 1] = 0
        elif Ksystijdelijk[(i - 1), (i - 1)] == 0:
            Ksystijdelijk[(i - 1), (i - 1)] = 1
    u = np.linalg.solve(Ksystijdelijk, belastingsvectortijdelijk)
    w_[j - 1] = u[(kpnr - 1)*3]
    phiy[j - 1] = u[(kpnr - 1)*3 + 1]
    phix[j - 1] = u[(kpnr - 1)*3 + 2]
    Mylinks[j - 1] = (m / Ly) * (TyB.subs({wA: u[((kpnr - 1) - 1) * 3], wB:
u[(kpnr - 1) * 3],
                                           phiyA: u[(((kpnr - 1) - 1) * 3) + 1],
phiyB: u[((kpnr - 1) * 3) + 1]}))
    Myrechts[j - 1] = (m / Ly) * (TyAW.subs({wA: u[(kpnr - 1) * 3], wB:
u[kpnr * 3],
                                             phiyA: u[((kpnr - 1) * 3) + 1],
phiyB: u[(kpnr * 3) + 1]}))
    Mxboven[j - 1] = (n / Lx) * (TxB.subs({uA: u[((kpnr - (n + 1)) - 1) *
3], uB: u[(kpnr - 1) * 3],
                                           phixA: u[(((kpnr - (n + 1)) - 1) * 3) +
2], phixB: u[((kpnr - 1) * 3) + 2]}))
    Mxonder[j - 1] = (n / Lx) * (TxAW.subs({uA: u[(kpnr - 1) * 3], uB:
u[((kpnr + (n + 1)) - 1) * 3],
```

```
                                        phixA: u[((kpnr - 1) * 3) + 2], phixB:
u[(((kpnr + (n + 1)) - 1) * 3) + 2]}))
    Vlinks[j - 1] = (TyB.subs({wA: u[((kpnr - 1) - 1) * 3], wB: u[(kpnr -
1) * 3],
                               phiyA: u[(((kpnr - 1) - 1) * 3) + 1], phiyB:
u[((kpnr - 1) * 3) + 1]}) -
                     TyAW.subs({wA: u[((kpnr - 1) - 1) * 3], wB: u[(kpnr -
1) * 3],
                                phiyA: u[(((kpnr - 1) - 1) * 3) + 1],
phiyB: u[((kpnr - 1) * 3) + 1]}))/dx
    Vrechts[j - 1] = (TyB.subs({wA: u[(kpnr - 1) * 3], wB: u[kpnr * 3],
                                phiyA: u[((kpnr - 1) * 3) + 1], phiyB:
u[(kpnr * 3) + 1]}) -
                      TyAW.subs({wA: u[(kpnr - 1) * 3], wB: u[kpnr * 3],
                                 phiyA: u[((kpnr - 1) * 3) + 1], phiyB:
u[(kpnr * 3) + 1]}))/dx
    Vboven[j - 1] = (TxB.subs({uA: u[((kpnr - (n + 1)) - 1) * 3], uB:
u[(kpnr - 1) * 3],
                               phixA: u[(((kpnr - (n + 1)) - 1) * 3) + 2],
phixB: u[((kpnr - 1) * 3) + 2]}) -
                     TxAW.subs({uA: u[((kpnr - (n + 1)) - 1) * 3], uB:
u[(kpnr - 1) * 3],
                                phixA: u[(((kpnr - (n + 1)) - 1) * 3) + 2],
phixB: u[((kpnr - 1) * 3) + 2]}))/dy
    Vonder[j - 1] = (TxB.subs({uA: u[(kpnr - 1) * 3], uB: u[((kpnr + (n +
1)) - 1) * 3],
                               phixA: u[((kpnr - 1) * 3) + 2], phixB:
u[(((kpnr + (n + 1)) - 1) * 3) + 2]}) -
                     TxAW.subs({uA: u[(kpnr - 1) * 3], uB: u[((kpnr + (n +
1)) - 1) * 3],
                                phixA: u[((kpnr - 1) * 3) + 2], phixB:
u[(((kpnr + (n + 1)) - 1) * 3) + 2]}))/dy

zakking = np.zeros(((m + 1), (n + 1)))
phiyMatrix = np.zeros(((m + 1), (n + 1)))
phixMatrix = np.zeros(((m + 1), (n + 1)))
MylinksMatrix = np.zeros(((m + 1), (n + 1)))
MyrechtsMatrix = np.zeros(((m + 1), (n + 1)))
MxbovenMatrix = np.zeros(((m + 1), (n + 1)))
MxonderMatrix = np.zeros(((m + 1), (n + 1)))
VlinksMatrix = np.zeros(((m + 1), (n + 1)))
VrechtsMatrix = np.zeros(((m + 1), (n + 1)))
VbovenMatrix = np.zeros(((m + 1), (n + 1)))
VonderMatrix = np.zeros(((m + 1), (n + 1)))
for j in range(1, ((m + 1) + 1)):
    for i in range(1, ((n + 1) + 1)):
        zakking[(j - 1), (i - 1)] = (w_[(j - 1)*(n + 1) + i - 1])
        phiyMatrix[(j - 1), (i - 1)] = phiy[(j - 1) * (n + 1) + i - 1]
        phixMatrix[(j - 1), (i - 1)] = phix[(j - 1) * (n + 1) + i - 1]
        MylinksMatrix[(j - 1), (i - 1)] = Mylinks[(j - 1) * (n + 1) + i -
1]
        MyrechtsMatrix[(j - 1), (i - 1)] = Myrechts[(j - 1) * (n + 1) + i -
1]
        MxbovenMatrix[(j - 1), (i - 1)] = Mxboven[(j - 1) * (n + 1) + i -
1]
        MxonderMatrix[(j - 1), (i - 1)] = Mxonder[(j - 1) * (n + 1) + i -
1]
        VlinksMatrix[(j - 1), (i - 1)] = Vlinks[(j - 1) * (n + 1) + i - 1]
        VrechtsMatrix[(j - 1), (i - 1)] = Vrechts[(j - 1) * (n + 1) + i -
1]
        VbovenMatrix[(j - 1), (i - 1)] = Vboven[(j - 1) * (n + 1) + i - 1]
```

```
        VonderMatrix[(j - 1), (i - 1)] = Vonder[(j - 1) * (n + 1) + i - 1]

x = np.linspace(0, Lx, (n + 1))
y = np.linspace(0, Ly, (m + 1))
X, Y = np.meshgrid(x, y)
f1 = plt.figure()
ha = f1.add_subplot(111, projection='3d')
ha.plot_wireframe(Y, X, -zakking)
plt.title('Zakking')

plt.figure()
CS = plt.contourf(Y, X, -zakking)
# plt.clabel(CS, inline=1, fontsize=10)
plt.colorbar(CS)
plt.title('Contourplot invloedslijn zakking')

f2 = plt.figure()
ha2 = f2.add_subplot(111, projection='3d')
ha2.plot_wireframe(Y, X, -MylinksMatrix)
plt.title('Momenten in x-richting')
f3 = plt.figure()
ha3 = f3.add_subplot(111, projection='3d')
ha3.plot_wireframe(Y, X, -MxbovenMatrix)
plt.title('Momenten in y-richting')
f4 = plt.figure()
ha4 = f4.add_subplot(111, projection='3d')
ha4.plot_wireframe(Y, X, VlinksMatrix)
plt.title('Dwarskracht in x-richting')
f5 = plt.figure()
ha5 = f5.add_subplot(111, projection='3d')
ha5.plot_wireframe(Y, X, VbovenMatrix)
plt.title('Dwarskracht in y-richting')

plt.show()
```