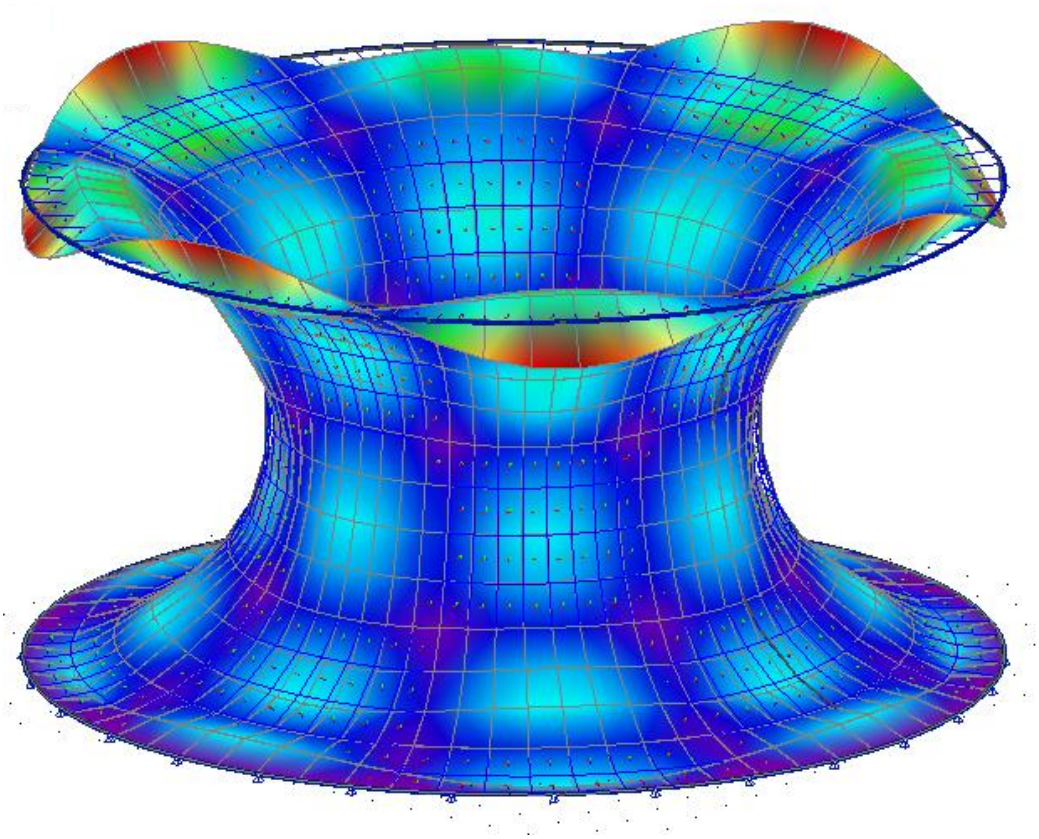
TU Delft



Ontwerptoetsen van kniklast op Schaalconstructies

Met SCIA Engineer

Bachelor Eindwerk
20-06-2020
Robin Streng

Bachelor Eindwerk

Auteur:	Robin Streng
Studienummer:	4696794
Opleiding:	Civiele Techniek
Track:	Structural Engineering
Product:	Eindrapport
Datum:	20-06-2020
Plaats:	Lopik
Begeleiders:	Pierre Hoogenboom
	Marco Schuurman



Voorwoord

Voor u ligt mijn rapport ‘Ontwerptoetsen van kniklast op schaalconstructies, met behulp van SCIA Engineer’. Dit rapport dient als bachelor eindwerk ter afronding van de bachelor civiele techniek aan de Technische Universiteit Delft. Deze studie bouwt voort op het onderzoek van Marieke Vergeer uit 2016 (Vergeer, 2016). Dit rapport is bestemd voor lezers die geïnteresseerd zijn in ontwerptoetsen voor knik in schaalconstructies. Ook kan hoofdstuk 3 interessant zijn voor lezers die informatie willen over verschillende functies van SCIA Engineer en Design Forms Builder. Graag wil ik Pierre Hoogenboom en Marco Schuurman bedanken voor de adviezen en inzichten die ze mij gegeven hebben.

Robin Streng

Lopik, 20 juni 2020

Samenvatting

Schaalconstructies zijn meestal slanke, gekromde en sterke constructies, die veel worden toegepast binnen de civiele techniek. Een belangrijk faalmechanisme is knik. Echter is het gedrag van schalen moeilijk te voorspellen en zijn er ingewikkelde niet-lineaire knikanalyses nodig om een ontwerp te evalueren. In dit rapport zijn verschillende methoden toegepast om krommingen in een constructie te benaderen en er zijn twee toetsen geprogrammeerd die snel inzicht geven in het gedrag van de schaalconstructies en welke factoren hier invloed op hebben. De codes zijn geschreven in Design Forms Builder en vervolgens toegepast op schaalconstructies in SCIA Engineer.

Voor beide toetsen is het noodzakelijk dat de krommingen bekend zijn. Deze zijn echter niet voorgeprogrammeerd in SCIA Engineer en moeten dus handmatig bepaald worden. Hier is vaak veel geheugen voor nodig, omdat er veel iteraties uitgevoerd moeten worden om alle knopen in een constructie te analyseren. De resultaten hiervan worden getoetst op een constructie waar de krommingen van bekend zijn.

De eerste toets bekijkt of de drukkracht kleiner is dan de trekkracht in de tegengestelde richting in schalen met een tegengestelde kromming. Als dit niet het geval is, knikt de schaalconstructie bij een lage drukkracht. Om de toets te verifiëren, worden de resultaten vergeleken met de knikvorm die volgt uit een lineaire stabiliteitsberekening. Bij de tweede toets wordt de lokale kniklast in positief en negatief gekromde schaalconstructies berekend. Hierbij is de lokale kniklast belangrijk om te weten, omdat een optimaal ontworpen schaal niet globaal, maar lokaal zal knikken. Als de schaalranden goed ondersteund zijn, zal er eerst lokale knik optreden. De toets berekent zelf eerst de kritische kniklastfactor, waarna deze wordt vergeleken met de lineaire stabiliteitsberekening. Als deze overeen komen, kan verondersteld worden dat de toets geldig is.

Er zijn verschillende resultaten vastgesteld. Uit de bepaling van krommingen volgt dat het benodigde geheugen een groot obstakel blijft en dat er geen goede benadering is gevonden. Dit heeft als gevolg dat de toetsen niet gebruiksvriendelijk zijn. De toetsen kunnen wel gebruikt worden, maar de krommingen moeten handmatig worden ingevoerd. Bij de resultaten van toets I bleek dat een op druk belaste constructie met een tegengestelde kromming altijd bij een lage belasting zal knikken. Het is echter niet geverifieerd of de toets geldig is, omdat de controleconstructie randknik vertoond, waar toets I uitgaat van een knikvorm met een ringpatroon of schaakbordpatroon. Toets II bleek een goed beeld te geven van de eigenschappen en dimensies die invloed hebben op de lokale knik. Hierbij is het wel noodzakelijk dat de lengte van de constructie voldoende lang is, omdat bij een kleine lengte de kritische kniklastfactoren van toets II en de lineaire knikberekening niet overeenkomen. Toets II is daarentegen geen vervanger van een niet-lineaire knikanalyse, die het knikgedrag gedetailleerd beschrijft.

Op basis van de resultaten wordt aanbevolen om gericht onderzoek te doen naar het benaderen van krommingen in schaalconstructies. Een gebruikte methode benadert krommingen in één richting door de kromtestraal van iedere knoop te bepalen met behulp van de twee dichtstbijzijnde knopen. De code werkt nog niet door een fout in de code, maar zou een goede benadering kunnen zijn als deze fout eruit is. Ook verdient het de aanbeveling om te onderzoeken of knopen logisch geordend kunnen worden in SCIA Engineer en deze vervolgens in te laden in Design Forms Builder. Dit zou een grote reductie van het benodigde geheugen opleveren.

Verder zou toets I geverifieerd kunnen worden door een randbalk aan de constructie toe te voegen en de toets opnieuw uit te voeren. Dit onderdrukt de randknik, waardoor er een ringpatroon of schaakbordpatroon als knikvorm zou moeten ontstaan. Ten slotte zou toets II anders geformuleerd kunnen worden, waardoor de focus op het verbeteren van ontwerpen ligt in plaats van een alternatief van de niet-lineaire knikanalyse.

Inhoudsopgave

Voorwoord ii
Samenvatting iii
Inhoudsopgave iv
Symbolenlijst vi
1. Inleiding 1
2. Theorie 2
2.1 Kromming 2
2.1.1 Gauss-kromming 3
2.2 Krachten in schaalconstructies 3
2.3 Knik in schaalconstructies 4
2.3.1 Knikvormen bij schaalconstructies 4
2.3.2 Kniklastfactoren 4
2.3.3 Kritische kniklast voor negatieve Gauss-kromming (TOETS I) 4
2.3.4 Invloed van imperfecties 5
2.3.5 Lineaire knikanalyse 5
2.3.6 Niet-lineaire eindige-elementenanalyse 5
2.3.7 Lokale kniklast (TOETS II) 5
3. Software 6
3.1 SCIA Engineer 6
3.1.1 Lineaire stabiliteitsberekening 6
3.2 Design Forms Builder 7
3.2.1 Hoe open je Design Forms Builder? 7
3.2.2 Calculation 7
3.2.3 Header 7
3.3 Koppeling tussen SCIA Engineer & Design Forms Builder 8
3.2.3 Stappenplan en aandachtspunten 8
4. Codes voor krommingen 9
4.1 Complexiteiten 9
4.1.1 Lokale assenstelsel 9
4.1.2 Beschikbare geheugen 9
4.2 Controleconstructie voor krommingen 9
4.3 Methode 1: Code van Marieke Vergeer 10
4.3.1 Werking van de code 10
4.3.2 Resultaat 11
4.4 Methode 2: Kromtestraal in x- en y-richting 12
4.4.1 Werking van de code 12
4.4.2 Resultaat 13
4.5 Methode 3: Parametrisatie 13

5. Toets I: Knik bij negatieve Gauss-kromming 14
5.1 Controleconstructie 14
5.2 Controle bij negatieve Gauss-kromming 14
5.3 Resultaten bij negatieve Gauss-kromming 15
5.3.1 Eerste controle 16
5.3.2 Tweede controle 17
6. Toets II: Lokale kniklast 19
6.1 Controleconstructies 19
6.2 Controle lokale kniklast 20
6.3 Resultaten 21
6.3.1Eerste controle 21
6.3.2 Tweede controle 22
7. Conclusie 25
8. Aanbevelingen 26
Verwijzingen 27
Bijlage A: Code voor kromming van methode 1 28
Bijlage B: Code voor kromming van methode 2 31
Bijlage C: Code toets I 33
Bijlage D: Overige resultaten van toets I 34
Bijlage E: Code van toets II 35
Bijlage F: Overige resultaten van toets II 36

Symbolenlijst

a_1 , a_2 , a_3 , a_4 , a_5 , a_6	= Constanten in vergelijking van kwadratisch oppervlak (-)
a , a_x , a_y	= Kromtestraal (m)
C	= Knock down factor (-)
d	= Imperfectie-amplitude (m)
E	= Elasticiteitsmodulus (N/m^2)
h	= Stapgrootte voor numerieke differentiatie (-)
k_G	= Gauss-kromming (m^{-2})
k , k_{xx} , k_{yy} , k_{xy}	= Kromming in lokale assenstelsel (m^{-1})
$k_{\overline{xx}}$, $k_{\overline{yy}}$	= Kromming in globale assenstelsel (m^{-1})
l_x , l_y	= Kniklengte (m)
m_{xx} , m_{yy} , m_{xy}	= Moment (Nm)
n_{xx} , n_{yy} , n_{xy}	= Membraankracht (N)
q_x , q_y	= Uitwendige schuifkracht (N)
t	= Schaaldikte (m)
η	= Variabele in lokale knock down factor (-)
λ_{cr}	= Kritische kniklastfactor (-)
λ_{cr1} , λ_{cr2} , λ_{cr3}	= Kleinste kritische kniklastfactor (-)
λ_{lok}	= Lokale kniklastfactor (-)
ν	= Poisson's ratio (-)

1. Inleiding

Schaalconstructies zijn bijna altijd slanke, gekromde constructies. Als dit constructietype goed ontworpen is, kan het gemakkelijk grote belastingen afdragen over grote overspanningen. Vanwege de mechanische voordelen, relatief weinig gebruik van materiaal en vaak elegante vorm worden schaalconstructies regelmatig gebruikt binnen de civiele techniek, bijvoorbeeld bij grote daken van grote gebouwen.

Voor veel schaalconstructies is de maatgevende bezwijkmodus knik. De knikbelasting kan berekend worden met de eindige-elementenmethode. Met deze methode wordt, met behulp van een computerprogramma (bijvoorbeeld SCIA Engineer), een constructie beschreven als heel veel kleine elementen, waarin elk element zijn eigen krachtenverdeling en vervorming heeft. Een groot nadeel van deze methode is de grote hoeveelheid informatie die verkregen wordt. Hier is niet direct uit af te leiden hoe een ontwerp verbeterd kan worden. Moet de schaal dikker gemaakt worden, zo ja hoeveel dikker? Of is het beter om een grotere kromming toe te passen? Of is het verstijven van de randbalken een betere oplossing? Het kost veel tijd om al deze testen uit te voeren, vandaar dat er twee toetsen zijn om snel antwoord te kunnen geven op deze vragen. Deze toetsen zijn alleen nog niet geprogrammeerd in SCIA Engineer voor constructieberekeningen.

De eerste toets bekijkt of de drukkracht kleiner is dan de trekkracht in de tegengestelde richting in schalen met een tegengestelde kromming. Als dit niet het geval is, knikt de schaalconstructie bij een lage drukkracht. De tweede toets berekent de lokale kniklast in positief en negatief gekromde schaalconstructies. De lokale kniklast is belangrijk om te weten, omdat een optimaal ontworpen schaal niet globaal, maar lokaal zal knikken. Als de schaalranden goed ondersteund zijn, zal er eerst lokale knik optreden.

Het doel van deze studie is het programmeren van krommingen en bovenstaande toetsen in SCIA Engineer. Dit wordt overzichtelijk weergegeven op het oppervlak van verschillende schaalconstructies. Deze studie bouwt voort op het onderzoek van Marieke Vergeer uit 2016 (Vergeer, 2016). Deze resultaten zullen worden geverifieerd met constructies waar de resultaten bekend van zijn.

De opbouw van het rapport is als volgt: In hoofdstuk 2 zal de theorie behandeld worden, onder andere krommingen, krachten in schaalconstructies en knik in schaalconstructies komen aan bod. Hoofdstuk 3 geeft een beknopte uitleg over SCIA Engineer, Design Form Builder en de koppeling tussen deze twee programma's. Hoofdstuk 4 beschrijft verschillende methoden en codes die gebruikt zijn om de krommingen te bepalen. In hoofdstuk 5 wordt een beschrijving gegeven van de eerste toets en in hoofdstuk 6 van de tweede toets. In hoofdstuk 7 worden de resultaten vergeleken en wordt er een conclusie getrokken. Ten slotte worden er in hoofdstuk 8 aanbevelingen gedaan.

2. Theorie

In dit hoofdstuk wordt de benodigde theorie voor het onderzoek beschreven. De theorie van schaalconstructies kan heel diepgaand zijn, maar hier wordt niet te diep op ingegaan. Het blijft bij oppervlakkige achtergrondinformatie en noodzakelijke formules en aannames die nodig zijn om de toetsen van dit onderzoek toe te lichten. Dit hoofdstuk is gebaseerd op hand-out 1, 2, 12 en 13 van CIE4143 Shell Analysis, Theory and Application (Hoogenboom, Notes on Shell structures, 2020). Hier is ook meer informatie te vinden over de bewijzen van theorieën en formules.

2.1 Kromming

De theorie over kromming is gebaseerd op pagina 20 t/m 25 van hand-out 2 (Hoogenboom, Handout 2, 2020).

Een kromme lijn kan beschreven worden als een cirkel met een bepaalde straal a dat de curve op één punt raakt. Deze cirkel kan geschat worden door de twee dichtstbijzijnde punten, naast het te bepalen punt, te bepalen en een lijn door deze punten haaks op de curve te tekenen. Het snijpunt van deze lijnen is het middelpunt van de cirkel (zie *figuur 2.1a*). De kromming k van de curve op punt X wordt gegeven door: $k = \frac{1}{a}$.

Schaalconstructies bestaan vaak uit gekromde elementen, zowel in één of in twee richtingen. Daarom zijn ook hier krommingen voor gedefinieerd. Deze zijn gedefinieerd in het lokale assenstelsel voor schaalconstructies. In het lokale assenstelsel staat de z-as altijd loodrecht op het oppervlak van de schaal (zie *figuur 2.1b*). De krommingen in xx-, yy en xy-richtingen zijn dan als volgt:

$$k_{xx} = \frac{\partial^2 z}{\partial x^2} = \frac{1}{a_x} \quad [1]$$

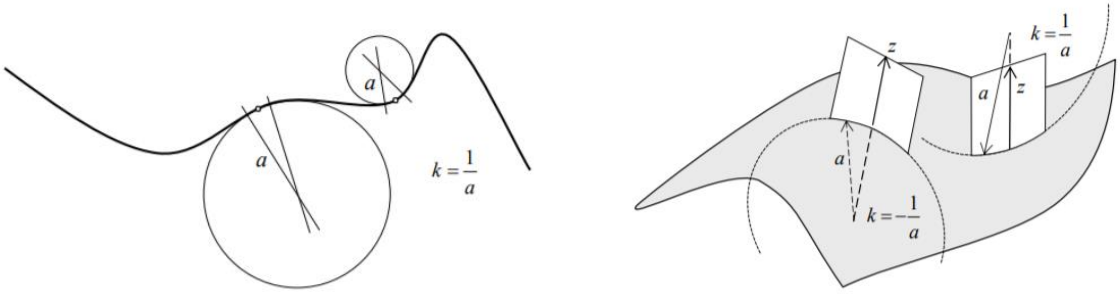
$$k_{yy} = \frac{\partial^2 z}{\partial y^2} = \frac{1}{a_y} \quad [2]$$

$$k_{xy} = \frac{\partial^2 z}{\partial x \partial y} \quad [3]$$

$\boldsymbol{k_{xy}}$ geeft de torsie van het schaaloppervlak aan. In het globale assenstelsel zijn de krommingen anders. De formules hiervoor zijn:

$$k_{\overline{xx}} = \frac{\frac{\partial^2 \bar{z}}{\partial \bar{x}^2}}{(1+\frac{\partial \bar{z}}{\partial \bar{x}}^2)^{\frac{3}{2}}} \quad [4]$$

$$k_{\overline{yy}} = \frac{\frac{\partial^2 \bar{z}}{\partial \bar{y}^2}}{(1+\frac{\partial \bar{z}}{\partial \bar{y}}^2)^{\frac{3}{2}}} \quad [5]$$

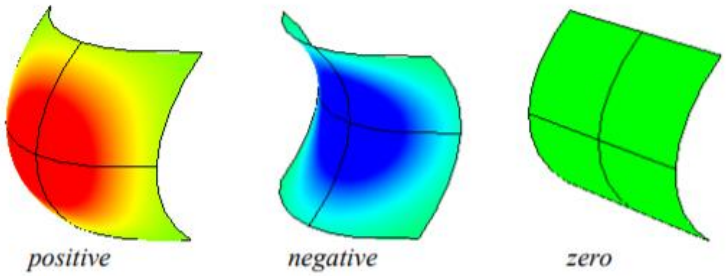


a. Kromming van een lijn *b. Krommingen van een oppervlak*

Figuur 2.1: Krommingen (Hoogenboom, Handout 2, 2020)

2.1.1 Gauss-kromming

De Gauss-kromming geeft aan wat voor type kromming een constructie heeft. De Gauss-kromming kan berekend worden met $\boldsymbol{k_G = k_{xx}k_{yy} - k_{xy}}$. De Gauss-kromming kan positief, negatief of nul zijn. Bij veel standaard constructies kan snel gezien worden wat voor type kromming het oppervlak is. Zo is de Gauss-kromming nul voor bijvoorbeeld platen en cilinders en positief voor een bol. In *figuur 2.2* zijn voorbeelden van verschillende typen oppervlakken weergegeven.

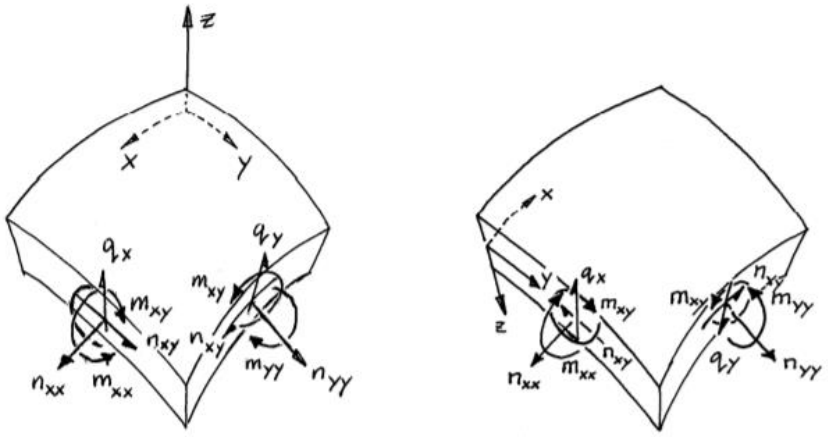


Figuur 2.2: Gauss-kromming (Hoogenboom, Handout 2, 2020)

2.2 Krachten in schaalconstructies

De theorie over krachten in schaalconstructies is gebaseerd op pagina 12 en 13 van hand-out 1 (Hoogenboom, Handout 1, 2020).

Normaal gesproken worden inwendige krachten uitgedrukt als spanningen [N/mm²]. Omdat de dikte t bij schaalconstructies (zo goed als) constant is, worden inwendige krachten in schalen uitgedrukt als verdeelde belastingen [N/mm]. De membraankrachten n_{xx} en n_{yy} zijn normaalkrachten en n_{xy} is de schuifkracht in het vlak. q_x en q_y zijn de schuifkrachten loodrecht op het vlak. De momenten m_{xx} en m_{yy} zijn buigmomenten en m_{xy} is het torsiemoment. Een doorsnede van een deel van een schaalconstructie met de inwendige krachten is weergegeven in *figuur 2.3*. Voor dit onderzoek zijn alleen de membraankrachten relevant.



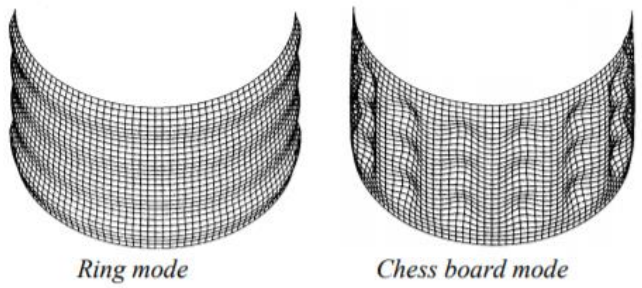
Figuur 2.3: Inwendige krachten in schaalconstructies (Hoogenboom, Handout 1, 2020)

2.3 Knik in schaalconstructies

Voor veel schaalconstructies is de maatgevende bezwijkmodus knik. Daarom zijn er veel studies naar verricht en zijn er vele theorieën beschikbaar. De theorie over knik in schaalconstructies is gebaseerd op hand-out 12 (Hoogenboom, Handout 12, 2020) en hand-out 13 (Hoogenboom, Handout 13, 2020).

2.3.1 Knikvormen bij schaalconstructies

Wanneer schaalconstructies axiaal op druk belast worden, kunnen deze twee knikvormen aannemen: ringpatroon of schaakbordpatroon (zie *figuur 2.4)*. Ook kunnen schalen bij de randen knikken, maar dit wordt in de praktijk voorkomen door de randen te verstevigen (bijvoorbeeld met randbalken).



Figuur 2.4: Knikvormen van schaalconstructies (Hoogenboom, Handout 12, 2020)

2.3.2 Kniklastfactoren

Om de knikbelasting voor een lineaire analyse te definiëren, is er een kniklastfactor λ_{cr} afgeleid. Bij de bepaling van deze kniklastfactor is er uitgegaan van een knikvorm met een ringpatroon of een schaakbordpatroon. Deze patronen geven beide dezelfde kniklastfactoren. De knikbelasting wordt dan bepaald door deze kniklastfactor te vermenigvuldigen met de belastingcombinatie. Als de knikbelasting lager is dan de belastingcombinatie, dan zal de constructie op knik bezwijken. De formule luidt:

$$\lambda_{cr} = \frac{-Et}{\frac{n_{xx}}{l_x^2}+\frac{n_{yy}}{l_y^2}}\left(\frac{\pi^2 t^2}{12(1-\nu^2)}\left(\frac{1}{l_x^2}+\frac{1}{l_y^2}\right)^2 + \frac{\left(\frac{k_{xx}}{l_y^2}+\frac{k_{yy}}{l_x^2}\right)^2}{\pi^2\left(\frac{1}{l_x^2}+\frac{1}{l_y^2}\right)^2}\right) \qquad [6]$$

Wanneer $0 < \lambda_{cr} < 1$ → Constructie bezwijkt

Als er vanuit wordt gegaan dat de randen van de constructie geen invloed hebben op de knikmodus, dan gedragen de kniklengtes l_x en l_y zich zo, dat de kniklastfactoren zo klein mogelijk zijn. Dit geeft de volgende drie kniklastfactoren:

$$\lambda_{cr1} = \frac{-Et^2}{\sqrt{3(1-\nu^2)}}\frac{|k_{yy}|}{n_{xx}} \qquad \lambda_{cr2} = \frac{-Et^2}{\sqrt{3(1-\nu^2)}}\frac{|k_{xx}|}{n_{yy}} \qquad \lambda_{cr3} \approx 0 \qquad [7]$$

λ_{cr1} en λ_{cr2} kunnen in alle schaalconstructies optreden. λ_{cr3} kan alleen optreden in schaalconstructies met een negatieve Gauss-kromming. Deze is dan klein, dus als deze positief is, zal de constructie bij een lage belasting bezwijken.

2.3.3 Kritische kniklast voor negatieve Gauss-kromming (TOETS I)

Voor schaalconstructies met een negatieve Gauss-kromming wordt gecontroleerd of de trekkracht overal in de constructie groter is dan de drukkracht in tegengestelde richting. Als dit niet het geval is, knikt de schaal al bij een lage belasting. Dit wordt gedaan met de volgende check:

$n_{xx}|k_{xx}| + n_{yy}|k_{yy}| > 0 \rightarrow \lambda_{cr3} < 0$ (trekkracht groter dan drukkracht = niet gevaarlijk) [8]

$n_{xx}|k_{xx}| + n_{yy}|k_{yy}| < 0 \rightarrow \lambda_{cr3} > 0$ (trekkracht kleiner dan drukkracht = gevaarlijk) [9]

2.3.4 Invloed van imperfecties

Verschillende experimenten hebben bewezen dat de maximale knikbelasting vaak veel lager ligt dan de theoretische kritische belasting, zelfs als de imperfectie heel klein is.

2.3.5 Lineaire knikanalyse

Bij een lineaire knikanalyse worden de kritische kniklastfactoren λ_{cr} berekend met behulp van eindige-elementenprogramma's (bv. SCIA Engineer). Als de op-een-na kleinste kniklastfactor heel dicht bij de kleinste kniklastfactor ligt, dan is de constructie waarschijnlijk erg gevoelig voor imperfecties. Het heeft geen zin om vooraf imperfecties aan te brengen, omdat deze in de realiteit langzaam groeien bij een oplopende belasting. Een lineaire knikanalyse houdt hier geen rekening mee, een niet-lineaire knikanalyse doet dat wel.

Om bij het ontwerp van schaalconstructies toch rekening te houden met imperfecties, wordt er een knock down factor C gebruikt. Deze knock down factor wordt vermenigvuldigd met de kniklastfactor. Als $0 < C\lambda_{cr} < 1$, wordt ervan uitgegaan dat de constructie bezwijkt. Als er weinig informatie over een constructie beschikbaar is, wordt bij het ontwerp als aanname vaak $C = \frac{1}{6}$ gebruikt.

2.3.6 Niet-lineaire eindige-elementenanalyse

Een niet lineaire eindige-elementenanalyse houdt rekening met vormimperfecties, imperfecties in de stijfheid van de opleggingen en plastische effecten. Deze imperfecties zijn niet van tevoren te bepalen, dus worden deze geschat. Vaak kiezen constructeurs ervoor om de eerste knikvorm aan te nemen als vormimperfectie. Vervolgens wordt de belasting langzaam hoger gemaakt, waarbij steeds de verplaatsingen worden bepaald. Deze analyses kosten veel tijd en het geeft weinig informatie over hoe een ontwerp verbeterd kan worden.

2.3.7 Lokale kniklast (TOETS II)

Een optimaal ontworpen schaal zal niet globaal, maar lokaal knikken. Dit komt door langzaam groeiende imperfecties van de schaal bij een toenemende belasting. Als de belasting langzaam toeneemt, zal de kromming op de ene plek toenemen en op een andere plek afnemen. Deze imperfecties zijn niet met het blote oog te zien, maar hebben wel een grote invloed. Ook veranderen hierdoor de membraankrachten in de schaal. Op de plekken waar een negatieve Gauss-kromming ontstaat, nemen de drukkrachten toe. Hier ontstaat een lokale knik.
Als de schaalranden goed ondersteund zijn, wordt de globale knik onderdrukt. Om de invloed van verschillende factoren beter in te kunnen schatten, is er een methode voorgesteld. Hierbij wordt wederom een knikvorm van een ringpatroon of schaakbordpatroon (zie *figuur 2.4*) aangenomen met een imperfectie amplitude d. Het resultaat van de lokale kniklastfactor λ_{lok} is als volgt:

$$\lambda_{lok1} = \frac{-Et^2}{\sqrt{3(1-\nu^2)}}\frac{|k_{yy}|}{n_{xx}}C \qquad C = \frac{\left(\frac{k_{xx}}{k_{yy}} - 1 - 2\eta\frac{d}{t}\right)^2}{4\left(\frac{k_{xx}}{k_{yy}} - \frac{n_{yy}}{n_{xx}} - 3\eta\frac{d}{t}\right)\left(\frac{k_{xx}}{k_{yy}} - 2\eta\frac{d}{t}\right)} \qquad \eta = \frac{\sqrt{3(1-\nu^2)}}{1-C} \qquad [10]$$

$$\lambda_{lok2} = \frac{-Et^2}{\sqrt{3(1-\nu^2)}}\frac{|k_{xx}|}{n_{yy}}C \qquad C = \frac{\left(\frac{k_{yy}}{k_{xx}} - 1 - 2\eta\frac{d}{t}\right)^2}{4\left(\frac{k_{yy}}{k_{xx}} - \frac{n_{xx}}{n_{yy}} - 3\eta\frac{d}{t}\right)\left(\frac{k_{yy}}{k_{xx}} - 2\eta\frac{d}{t}\right)} \qquad \eta = \frac{\sqrt{3(1-\nu^2)}}{1-C} \qquad [11]$$

Hierin is η een nieuwe variabele die geïntroduceerd wordt, die afhankelijk is van C. Zoals te zien in de vergelijkingen, is dit een iteratief proces. Om C te bepalen, moet een initiële waarde gekozen worden. Ook is de imperfectie amplitude d meestal niet bekend, dus zal deze ook geschat moeten worden.

3. Software

In dit hoofdstuk wordt beschreven hoe relevante opties in SCIA Engineer en Design Forms Builder zijn gebruikt. Dit geeft duidelijkheid in hoe de programma's werken en hoe hier informatie uit is verkregen.

3.1 SCIA Engineer

SCIA Engineer is een programma dat veel gebruikt wordt voor constructieberekeningen. Wanneer er een constructie gemodelleerd is, de opleggingen zijn gedefinieerd en een belasting is toegebracht, kan SCIA Engineer verschillende berekeningen uitvoeren. Na deze berekeningen kunnen bijvoorbeeld de verplaatsingen en inwendige spanningen weergegeven worden op het oppervlak van de constructie. Een handige berekening voor dit onderzoek is de 'lineaire stabiliteitsberekening'. Hoe dit gebruikt kan worden, is in paragraaf 3.1.1 kort toegelicht. Meer informatie kan gevonden worden op de sites van SCIA (zie (SCIA webhelp, 2019) en (SCIA, 2020)).

3.1.1 Lineaire stabiliteitsberekening

Met de functie lineaire stabiliteitsberekening kunnen de kritische kniklastfactoren worden bepaald. Ook kunnen de vervormingen en spanningen in de constructie worden weergegeven. Een kort stappenplan voor de lineaire stabiliteitsberekening wordt hieronder gegeven:

1. Voeg belastingen toe aan een "stabiliteitscombinatie".
2. Klik op "berekening" en zorg ervoor dat "lineaire stabiliteit" geselecteerd is.
3. Kies het aantal knikvormen in de "geavanceerde instellingen".
4. Voer de berekening uit. De kleinste kniklastfactor zal in het scherm te zien zijn.
5. Ga vervolgens naar "resultaten" en selecteer "kritische belastingcoëfficiënten".

De kritische belastingcoëfficiënten zijn de kritische kniklastfactoren. Verder kunnen de knikvormen met bijbehorende belastingcoëfficiënten weergegeven worden door bij "3D verplaatsing" het type last te veranderen naar "Stabiliteitscombinaties". Het is belangrijk om te beseffen dat SCIA Engineer de vervormingen in de knikvorm normaliseert. Dit betekent dat de knikvorm bekend is, maar de grootte van de vervormingen geschat zijn. In de praktijk kunnen de vervormingen dus groter of kleiner zijn en op een andere plek in de constructie optreden.

3.2 Design Forms Builder

Design Forms Builder is een programma waar gebruikers zelf een code kunnen schrijven. Deze code kan vervolgens ingeladen worden in SCIA Engineer. Design Forms Builder bestaat uit 5 tabs: Calculation, Header, Dialog, Images en Translations. In deze paragraaf wordt er beschreven hoe Design Forms Builder geopend kan worden en worden de eerste drie tabs toegelicht. Meer informatie kan gevonden worden op de sites van SCIA Design Forms (zie (SCIA Design Forms, sd)).

3.2.1 Hoe open je Design Forms Builder?

Design Forms Builder is niet zoals de meeste programma's vanaf het bureaublad te openen. Dit moet vanuit SCIA Engineer gebeuren. Een kort stappenplan is hieronder gegeven.

1. Open SCIA Engineer en open een bestaand project of maak een nieuwe aan.
2. Klik op "Geïntegreerde Design Forms".
3. Klik op "Controle manager". Hier zijn al een aantal voorbeeldchecks gegeven.
4. Om een nieuwe code te schrijven, klik op "Nieuw" en klik vervolgens op "Nieuw script".
   OF
   Om een bestaande code te openen, klik op de bestaande code en klik op "Open in de Script Editor".
5. Er verschijnt een melding in beeld. Klik op "OK".
6. Een scherm genaamd "Program settings" verschijnt nu. Hier kunnen verschillende instellingen aangepast worden. Eén van de instellingen is "Max. number of WHILE cycle iterations". De beginwaarde hiervan is 0. Dit is voor de meeste codes prima, maar hier kan een limiet aan gesteld worden. Verder hoeft er niets aangepast te worden. Druk op "OK".
7. SCIA Design Forms Builder wordt nu geopend.

3.2.2 Calculation

In de Calculation-tab wordt de code geschreven en vormgegeven. Het bestaat uit de volgende tools:
Script editor - Hierin wordt de code geschreven.
Table of variables – De variabelen die gebruikt worden in de code worden hier weergegeven. Hier kan de koppeling worden gemaakt met gegevens uit SCIA Engineer, zoals de krachten, constructie-eigenschappen en constructie-dimensies. Dit wordt gedaan met zogenaamde ESA_ID's.
Lay-out editor – Hier wordt weergegeven wat de output van de code is. Dit kan hier ook aangepast worden, maar is niet relevant voor de toetsen in dit rapport. Het is wel handig om te controleren of de code goed weergegeven wordt.
"Refresh" knop – Als er op deze knop geklikt wordt, dan wordt de code uitgevoerd en weergegeven in de Lay-out editor. Mocht er een fout in de code zitten, dan verschijnt een tool met "errors".

3.2.3 Header

In de Header-tab worden de uitvoergegevens bepaald, zoals de naam van de check, welke norm er gebruikt wordt, voor welke elementtypes de check gebruikt kan worden en voor welk materiaal.

3.2.4 Dialog

In de Dialog-tab worden de invoergegevens gedefinieerd. Een belangrijke tool hier is om variabelen aan "Member data" toe te voegen. Deze variabelen kunnen dan gemakkelijk handmatig in SCIA Engineer aangepast worden, zonder de code te moeten aanpassen in Design Forms Builder.

3.3 Koppeling tussen SCIA Engineer & Design Forms Builder

Als er een model gemaakt is in SCIA Engineer en er een code geschreven is in Design Forms Builder, is het tijd om de code te toetsen op het model. In deze paragraaf wordt een stappenplan beschreven hoe de koppeling tussen SCIA Engineer en Design Forms Builder tot stand gebracht kan worden en aandachtspunten om op te letten.

3.2.3 Stappenplan en aandachtspunten

In Design Forms Builder:

1. In de Calculation-tab, vul alle benodigde ESA_ID's in de Table of Variables in. Een lijst met ESA_ID's kan gevonden worden met de Help-functie van SCIA. Voor resultaten, gebruik Result.<nummer> (bijvoorbeeld Result.1).
2. In de Header-tab, voer de gegevens in. Zorg ervoor dat het juiste elementtype en materiaal geselecteerd zijn. Ook kunnen eenheden en limieten aan de resultaten toegevoegd worden.
3. Exporteer de code als CLC door op "Export CLC" te klikken. Zorg dat het bestand in de map /Documenten/ESA19.1/OpenChecks staat.

In SCIA Engineer:

4. Zorg dat er een berekening is uitgevoerd. Anders zijn de inwendige krachten, spanningen en vervormingen nog niet bepaald en kunnen de checks niet worden uitgevoerd.
5. Ga naar "Geïntegreerde Design Forms" en klik op "Controle manager". Klik op "Invoegen" en selecteer het CLC-bestand. Sluit vervolgens de "Controle Manager".
6. Onder "Andere controles" is de check nu weergegeven. Als er variabelen aan de Member data toegevoegd zijn, definieer die dan door middel van "2D Elementgegevens" onder de check aan te klikken en de variabelen aan te passen. Klik daarna op "Toets 1".
7. Rechts in beeld verschijnen eigenschappen van de toets. Onder "Waardes" kan bepaald worden welk resultaat weergegeven moet worden. Klik vervolgens op "Herlees". De resultaten worden weergegeven op de constructie.

4. Codes voor krommingen

De grootste uitdaging is het bepalen van de krommingen in de constructie. Om deze reden is er dan ook veel tijd gestoken in het vinden en bedenken van een goede methode om deze krommingen te benaderen. In dit hoofdstuk wordt uitgelegd wat de complexiteiten zijn en welke methoden er zijn toegepast voor een benadering van de krommingen in SCIA Engineer.

4.1 Complexiteiten

4.1.1 Lokale assenstelsel

Zoals in de theorie al is aangegeven, worden de krommingen gedefinieerd in het lokale assenstelsel. Echter is het lastig om in Design Forms Builder punten te definiëren in dit lokale assenstelsel. De krommingen moeten dus vanuit het globale assenstelsel omgerekend worden.

4.1.2 Beschikbare geheugen

Om een code van Design Forms Builder in SCIA Engineer te runnen, is er veel geheugen vereist. Als er in de code loops worden gebruikt, dat wil zeggen dat een berekening meerdere malen wordt doorgerekend, is er soms te veel geheugen nodig en kan de computer het niet aan. Dit is het geval bij veel methoden, omdat er meerdere malen de afstanden tussen knopen bepaald moet worden. Van tevoren bepaald SCIA Engineer of dit haalbaar is en geeft bij veel iteraties een foutmelding.

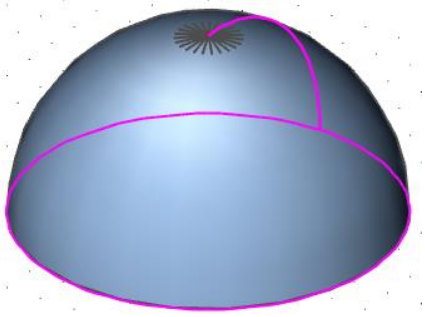
Het benodigde geheugen kan op drie manieren beperkt worden:

1. Het aantal knopen in de constructie wordt beperkt. Voor de code van Marieke Vergeer (Vergeer, 2016) konden maximaal 700 knopen gebruikt worden. Daarom zal daar ook ruim onder gebleven worden, om de kans op een tekort aan geheugen te minimaliseren.
2. Zo min mogelijk loops gebruiken in de code. Dit is echter lastig, omdat loops nodig zijn om een redelijke benadering van de krommingen te bepalen. Toch zal hier, waar mogelijk, rekening mee gehouden worden.
3. Als de buurknopen bepaalt kunnen worden met behulp van knoopnummering van de mesh-generatie en dit ingeladen kan worden in Design Forms Builder, dan zijn er nog maar heel weinig loops nodig. Het is niet bekend of de knopen logisch genummerd kunnen worden in SCIA Engineer en of dit gekoppeld kan worden aan Design Forms Builder.

4.2 Controleconstructie voor krommingen

Om de geprogrammeerde codes van de krommingen te kunnen testen, wordt een halve bol gebruikt. De gegevens van de constructie zijn weergegeven in *tabel 4.1* Deze constructie is om twee redenen gunstig voor de tests:

1. De kromming van een halve bol is bekend en bovendien overal hetzelfde, namelijk $k_{xx} = k_{yy} = \frac{1}{a}$.
2. In het rapport van Marieke Vergeer (Vergeer, 2016) wordt dezelfde halve bol gebruikt als controleconstructie, dus daarmee kunnen de verschillende resultaten vergeleken worden. Ook kan de code van Marieke Vergeer gebruikt worden (deels of volledig) om na te gaan of deze methode om krommingen te bepalen gebruikt kan worden.



Figuur 4.1: Controleconstructie halve bol

Tabel 4.1: Gegevens controleconstructie halve bol

Betonsoort	C45/55
Straal	5 m
Dikte	200 mm

4.3 Methode 1: Code van Marieke Vergeer (Vergeer, 2016)

Als eerste methode is de code van Marieke Vergeer gebruikt. Vergeer gebruikte de code om de Gauss-kromming en de verandering van de Gauss-kromming te bepalen. Dit is voor dit onderzoek niet nodig. Echter, voor de Gauss-kromming zijn eerst de krommingen bepaald en deze kunnen wel gebruikt worden. De code bepaalt eerst de vergelijking van het kwadratisch oppervlak en leidt daar de krommingen van af. De vergelijking heeft zes onbekenden, dus er zijn zes vergelijkingen nodig om deze op te lossen. De vergelijking voor een kwadratisch oppervlak is:

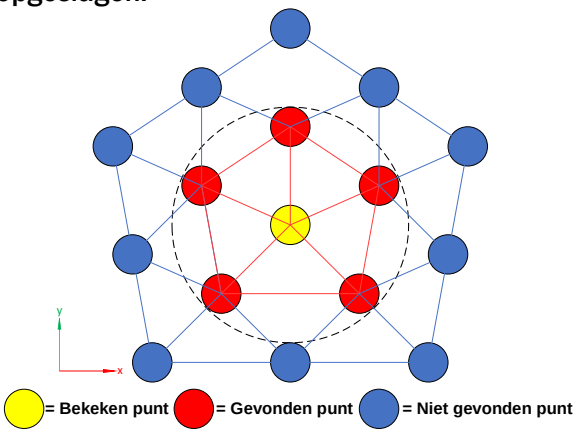
$$z(x,y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6 \quad [12]$$

4.3.1 Werking van de code

De code werkt als volgt:

1. Er wordt een bol gedefinieerd om het bekeken punt heen. Er worden zes punten gezocht binnen die bol. Als er minder dan zes punten gevonden zijn, wordt de cirkel vergroot en wordt er opnieuw naar zes punten gezocht.

2. Als er uiteindelijk zes punten zijn gevonden, worden de coördinaten van deze zes punten opgeslagen.



Figuur 4.2: Zoekmethode relevante punten van methode 1

3. Met deze zes punten kan de vergelijking van het kwadratisch oppervlak bepaald worden. Design Forms Builder is niet in staat om een matrixvergelijking op te lossen, dus dit moet 'handmatig' geprogrammeerd worden. Dit is in deze code gedaan met Gauss-eliminatie, ook wel rijreductie genoemd. Voor meer informatie over deze stap, zie (Vergeer, 2016)
4. Design Forms Builder bevat ook geen functie om afgeleiden te bepalen. Daarom worden de afgeleiden benaderd met behulp van numerieke differentiatie met een stapgrootte $h = 0.1$. De afgeleiden zijn dan als volgt (Vuik, 2018):

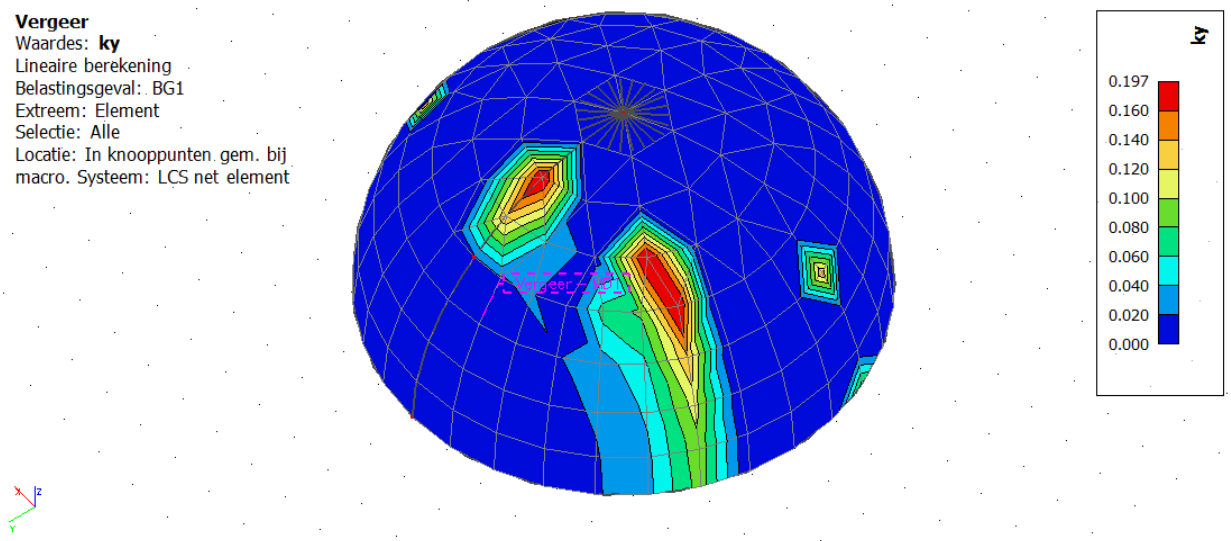
$$\frac{dz}{dx} = \frac{z(x+h,y)-z(x-h,y)}{2h} \qquad \frac{dz}{dy} = \frac{z(x,y+h)-z(x,y-h)}{2h} \quad [13]$$

$$\frac{d^2z}{dx^2} = \frac{z(x+h,y)-2z(x,y)+z(x-h,y)}{2h} \qquad \frac{d^2z}{dy^2} = \frac{z(x,y+h)-2z(x,y)+z(x,y-h)}{2h} \quad [14]$$

5. Met de afgeleiden kunnen de krommingen met formules [4] en [5] in het globale assenstelsel bepaald worden.

4.3.2 Resultaat

Bij de uitvoering van de code bleek dat er niet genoeg geheugen beschikbaar is om de code te kunnen runnen. In de code wordt het commando 'WHILE' gebruikt en dit zorgt ervoor dat Design Forms Builder en SCIA Engineer oneindig lang blijven itereren, totdat er wordt voldaan aan dit commando. Er is een maximale hoeveelheid iteraties 'WHILE" opgelegd. Dit kan gedaan worden in de "Program settings", die verschijnt voordat Design Forms Builder geopend wordt (zie paragraaf 3.2).
Dit geeft het resultaat dat te zien is in *figuur 4.3.*



Figuur 4.3: Resultaat kromming methode 1

Verwacht wordt een waarde van $k_y = \frac{1}{a} = \frac{1}{5} = 0.2\ m^{-1}$. Het overgrote deel van de constructie heeft als waarde 0. Dit heeft te maken met de maximaal toegestane iteraties van het 'WHILE' commando. SCIA Engineer kan niet genoeg iteraties uitvoeren, waardoor niet alle punten bekeken zijn en er dus geen krommingen van deze punten zijn bepaald. Daarom geeft SCIA Engineer hier een waarde van 0. Op een aantal punten wordt er wel een andere waarde gevonden en benaderen soms ook de verwachte kromming, zoals het rode gebied in *figuur 4.3*. Echter zijn veel benaderde krommingen erg onnauwkeurig en wordt er getwijfeld aan de juistheid van deze methode.

Voor deze methode is meer geheugen nodig dan beschikbaar is. Dit kan beperkt worden door een maximale hoeveelheid iteraties toe te passen. Echter, dit zorgt ervoor dat de krommingen niet op alle punten bepaald kunnen worden. In de punten waar de kromming wel bepaald is, is deze erg onnauwkeurig. Dit kan komen door de benadering van de krommingen in het globale assenstelsel of er zitten onjuistheden in de code.

Deze methode is dus niet voldoende. Ook zit er weinig potentie in om de code te verbeteren, aangezien de benodigde iteraties te hoog blijft en er dus veel geheugen nodig is.

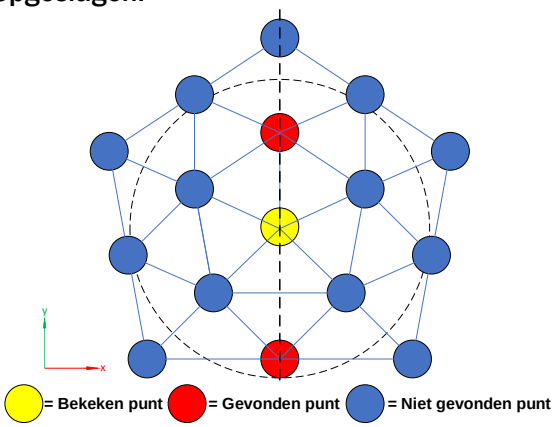
4.4 Methode 2: Kromtestraal in x- en y-richting

Deze methode benadert de kromtestraal in één specifieke richting (x of y) in iedere knoop in de constructie. Dat wordt gedaan door de twee dichtstbijzijnde punten, die ongeveer hetzelfde coördinaat (x of y) in één richting hebben. Op deze manier wordt de 3D-constructie als het ware 'platgeslagen', waardoor de krommingen in een 2D-vlak worden bepaald. Deze methode lijkt op methode 1, maar er zijn minder punten nodig om de krommingen te bepalen. Dit zou dus minder geheugen moeten kosten.

4.4.1 Werking van de code

De code voor de kromming k_{yy} werkt als volgt:

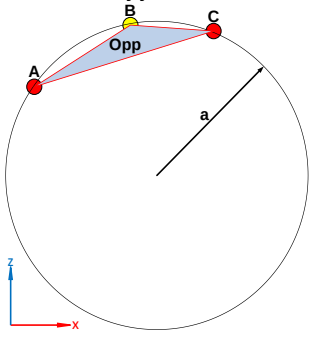
1. Er wordt een bol gedefinieerd om het bekeken punt heen. Er worden twee punten gezocht binnen die bol die ongeveer dezelfde y-coördinaat heeft. Als er minder dan twee punten gevonden zijn, wordt de cirkel vergroot en wordt er opnieuw naar twee punten gezocht.
2. Als er uiteindelijk twee punten zijn gevonden, worden de coördinaten van deze twee punten opgeslagen.



Figuur 4.4: Zoekmethode relevante punten van methode 2

3. Nu de coördinaten van drie punten bekend zijn (twee gevonden punten en bekeken punt), moeten de afstanden ten opzichte van elkaar bepaald worden. Dit is een eenvoudig geometrisch probleem, dat met de stelling van Pythagoras opgelost wordt.
4. De oppervlakte van de driehoek, die gevormd wordt door de drie coördinaten, wordt berekend.
5. De kromtestraal van de kromtecirkel wordt vervolgens berekend met de volgende formule ((curvature, sd):

$$a_y = \frac{|AB||AC||BC|}{4Opp} \qquad [15]$$



Figuur 4.5: Afstanden tussen punten en oppervlakte kromtedriehoek

6. De kromming wordt gegeven door $k_{yy} = \frac{1}{a_y}$.

Voor de x-richting werkt de code hetzelfde, waarbij er bij stap 1 gezocht wordt naar twee x-coördinaten die bijna hetzelfde zijn.

4.4.2 Resultaat

De code is alleen voor de y-richting uitgewerkt en getest. Bij het testen in SCIA Engineer werd er geconstateerd dat de resultaten van a_y en k_{yy} over de hele constructie 'NaN' (=Not a Number) is. Dit komt waarschijnlijk door een fout in de code. Dit is wegens tijdgebrek niet opgelost.

De code werkt nog niet door een fout in de code. Als deze fout opgelost wordt, zou de code kunnen werken. Het is niet bekend of de resultaten een goede benadering van de krommingen zou zijn.

Verwacht wordt dat de code relatief snel en eenvoudig verbeterd kan worden en om deze reden is het de moeite waard om hier verder onderzoek naar te doen. Mocht het dan blijken dat de resultaten niet overeenkomen of er blijkt toch nog te veel geheugen nodig zijn, dan is er niet al te veel tijd verloren gegaan. Echter, als de resultaten wel een goede benadering blijken te zijn, kunnen de krommingen voor willekeurige constructies bepaald worden en zal dit goed van pas komen voor verschillende toetsen en verder onderzoek. Kortom, verder onderzoek zou kunnen leiden tot een doorbraak.

4.5 Methode 3: Parametrisatie

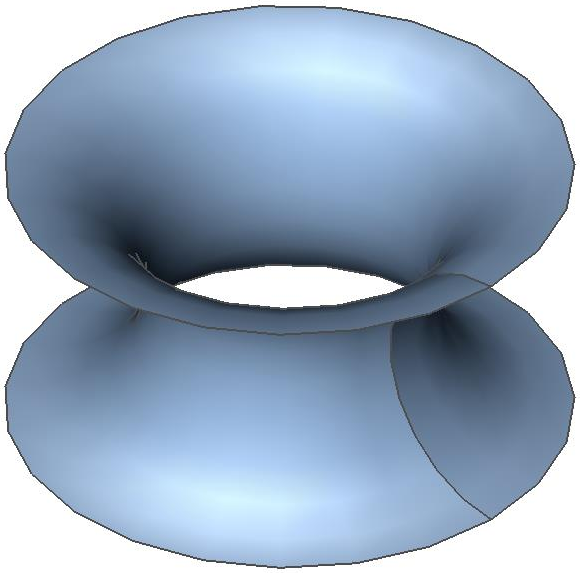
Een andere methode zou de krommingen kunnen bepalen met behulp van parametrisatie. Hiermee zouden de krommingen nauwkeurig bepaald kunnen worden. Het nadeel van deze methode is dat niet alle constructies geparametriseerd kunnen worden. Voor constructies waar parametrisatie wel mogelijk is, moeten de parameters voor elke constructie handmatig bepaald en ingevuld worden. Hiervoor moet de code dus voor iedere constructie aangepast worden. Dit zou niet erg gebruiksvriendelijk zijn en om deze reden is deze methode ook niet verder uitgewerkt.

5. Toets I: Knik bij negatieve Gauss-kromming

Schalen met een negatieve Gauss-kromming zijn erg gevoelig voor knik. Als ergens in de constructie de drukkracht groter is dan de trekkracht, dan kan een constructie al bij een lage belasting knikken. Dit wordt gecontroleerd met deze toets.

5.1 Controleconstructie

De toets wordt gecontroleerd met een hyperbool, waar de krommingen gemakkelijk bepaald kunnen worden. De ene kromming is in de vorm van een halve cirkel, waardoor die kromming constant is. De belasting op de constructie is een verticale drukkracht van 1 kN/m op de bovenrand.



Figuur 5.1: Hyperbool toets I

Tabel 5.1: Gegevens hyperbool toets I

Betonsoort	C45/55
Straal	10 m
Middenstraal	5 m
Dikte	20 mm
Hoogte	10 m

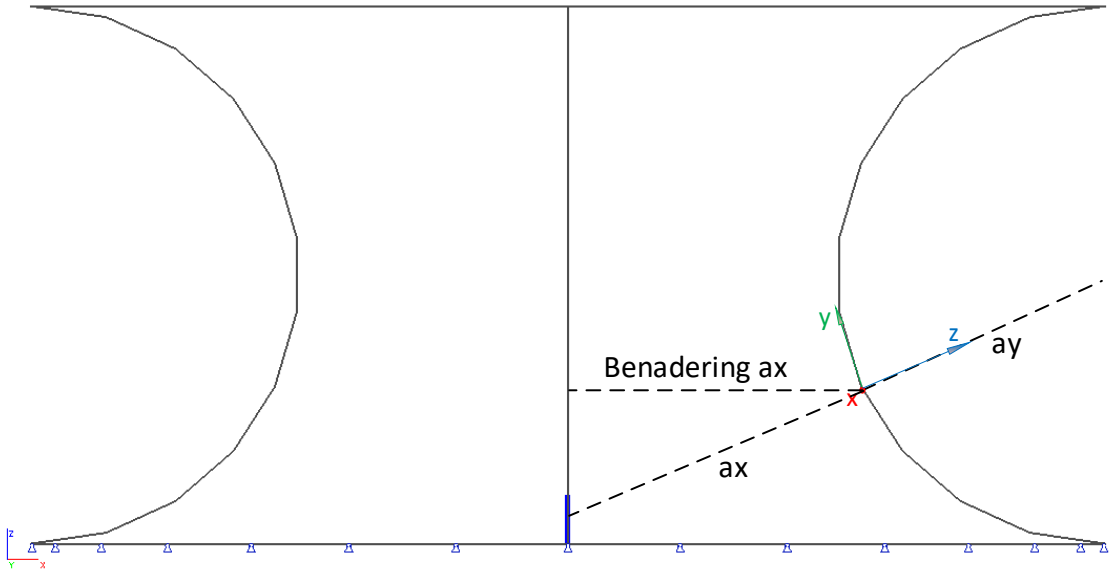
5.2 Controle bij negatieve Gauss-kromming

Zoals in de theorie is beschreven, wordt gecontroleerd of de trekkracht groter is dan de drukkracht in tegengestelde richting met de volgende check:

$n_{xx}|k_{xx}| + n_{yy}|k_{yy}| > 0 \rightarrow \lambda_{cr3} < 0$ (trekkracht groter dan drukkracht = niet gevaarlijk) [8]

$n_{xx}|k_{xx}| + n_{yy}|k_{yy}| < 0 \rightarrow \lambda_{cr3} > 0$ (trekkracht kleiner dan drukkracht = gevaarlijk) [9]

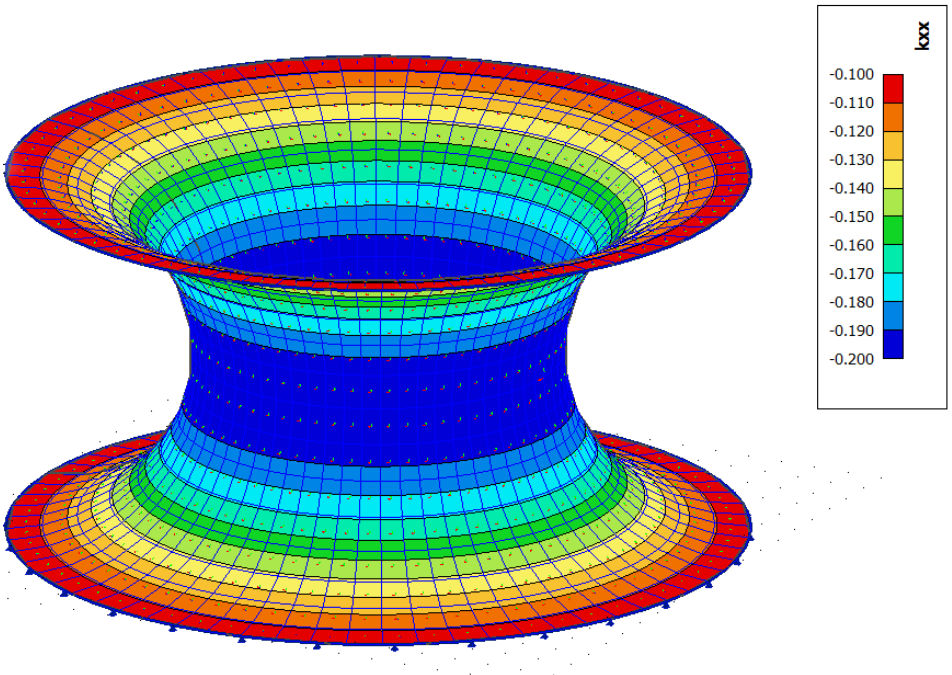
Hiervoor is het ook weer belangrijk dat de krommingen bekend zijn. In *figuur 5.2* is geschetst hoe de kromtestralen bepaald worden. De kromtestraal a_y is bekend, omdat dit een halve cirkel is. a_x is lastiger te bepalen. Dit is benaderd met $a_x = \sqrt{x^2 + y^2}$, met andere woorden de constructie wordt in het xy-vlak als cirkel gezien en de lokale z-as wijst naar het middelpunt van deze cirkel. Dit is niet zo, omdat de z-as altijd loodrecht op het oppervlak is. Dit is een redelijke benadering in het midden van de constructie, maar aan de randen van de constructie is de benadering van de kromming niet nauwkeurig. De membraankrachten kunnen worden ingeladen en de controle kan dus worden uitgerekend.



Figuur 5.2: Schematische weergave voor de bepaling van de kromtestralen

5.3 Resultaten bij negatieve Gauss-kromming

Zoals in *figuur 5.3* te zien is, varieert de kromming k_{xx} tussen -0.1 en -0.2 m^{-1}. Bij de randen is de waarde van de krommingen -0.1, waar deze eigenlijk 0 moeten zijn, omdat de lokale z-as loodrecht op het oppervlak staat. Dit zou een oneindige kromtestraal moeten geven en dus een kromming van 0 m^{-1}. In het midden van de constructie zijn de krommingen wel goed benaderd. Dit betekent dat de toets erg onnauwkeurig is op de randen en hier niet valide is. In het middendeel van de constructie is de benadering van de kromming wel voldoende en is de toets wel valide.

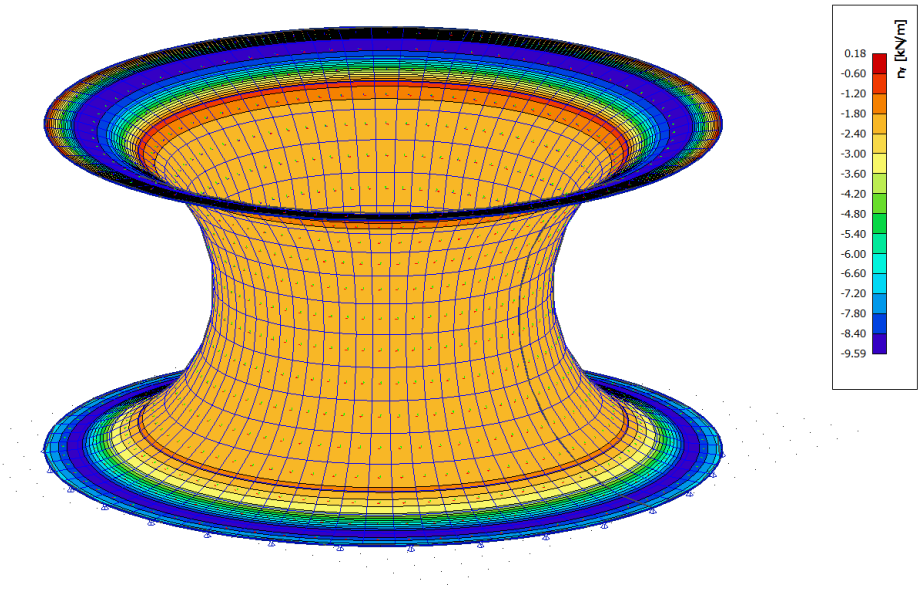


Figuur 5.3: Benadering van de kromming k_{xx} van hyperbool

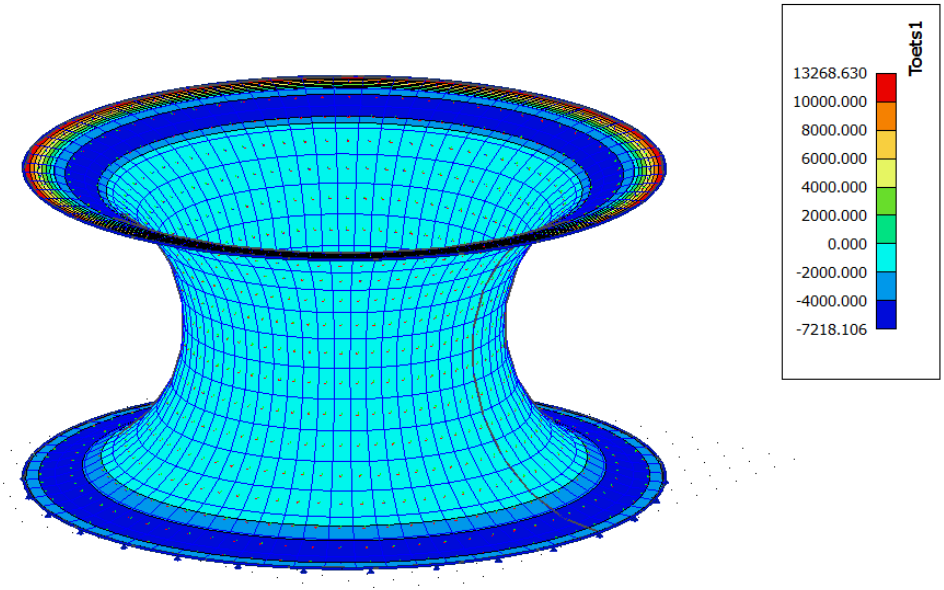
5.3.1 Eerste controle

Bij de eerste controle leek de toets realistisch, totdat deze vergeleken werd met de knikvorm die volgt uit de lineaire stabiliteitsberekening. Hieruit volgt namelijk dat er bij de bovenrand vervormingen plaatsvinden bij de kniklast (zie *figuur 5.6*). Deze vorm van knikken is randknik. De toets gaat uit van een knikvorm met een ringpatroon of schaakbordpatroon en is dus niet valide.

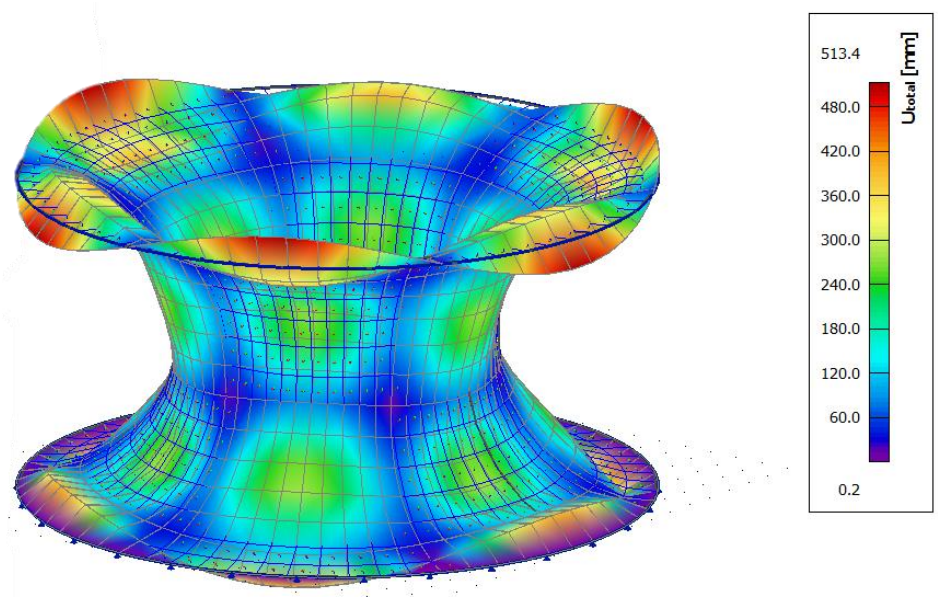
Het is bekend dat kromming $k_{xx} = 0$, dus membraankracht n_{xx} heeft geen invloed op de toets. Uit *figuur 5.4* blijkt dat n_{yy} op de bovenrand trekkracht is, wat impliceert dat er geen gevaar schuilt voor knikken. Dit komt niet overeen met de lineaire stabiliteitsberekening, dus moet het model aangepast worden. Om dit op te lossen, wordt de bovenrand opgelegd in x- en y-richting en in z-richting vrijgelaten. Dit beïnvloedt de mechanica van de schaal, maar er kunnen bij de bovenrand nog wel verticale verplaatsingen optreden.



Figuur 5.4: Membraankracht n_{yy} bij de eerste controle



Figuur 5.5: Resultaten toets I bij de eerste controle



Figuur 5.6: Knikvorm ($\lambda_{cr} = 12.22$) volgens lineaire stabiliteitsberekening bij eerste controle

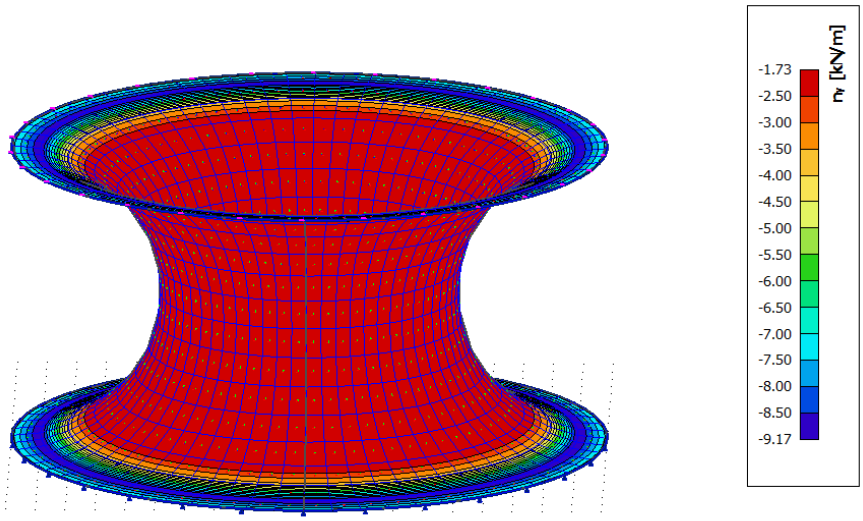
5.3.2 Tweede controle

Als de bovenrand in x- en y-richting opgelegd is, wordt er een realistischer resultaat weergegeven. n_{yy} is boven in de constructie een drukkracht (zie *figuur 5.7*). Dit komt overeen met de knikvorm die volgt uit de lineaire stabiliteitsberekening.

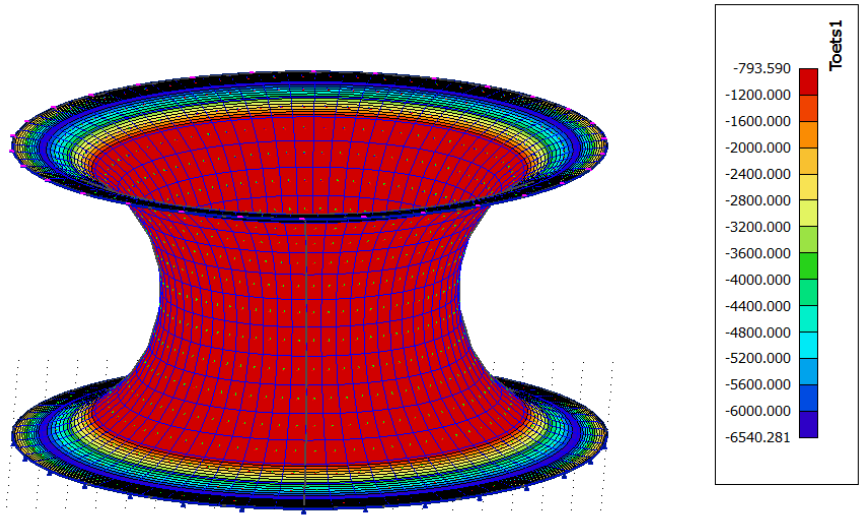
De toets is overal in de constructie negatief. Dit betekent dat de constructie bij een lage belasting zal knikken en dit op ieder punt in de constructie kan gebeuren. Dit blijkt ook uit de lineaire stabiliteitsberekening, waar een kleinste kniklastfactor van 6.95 uit volgt. De krommingen kunnen aangepast worden, maar zo lang de Gauss-kromming negatief is, zullen er altijd negatieve waarden uit de toets komen en zal de constructie bij een lage belasting al knikken.

De krachtsverdeling moet dus aangepast worden. Dit is in de praktijk niet altijd even makkelijk, maar theoretisch gezien zou de kniksterkte groter zijn als de belasting omdraait. In de praktijk betekent dit dat de constructie bijvoorbeeld opgehangen wordt en dat er aan de onderzijde getrokken wordt.

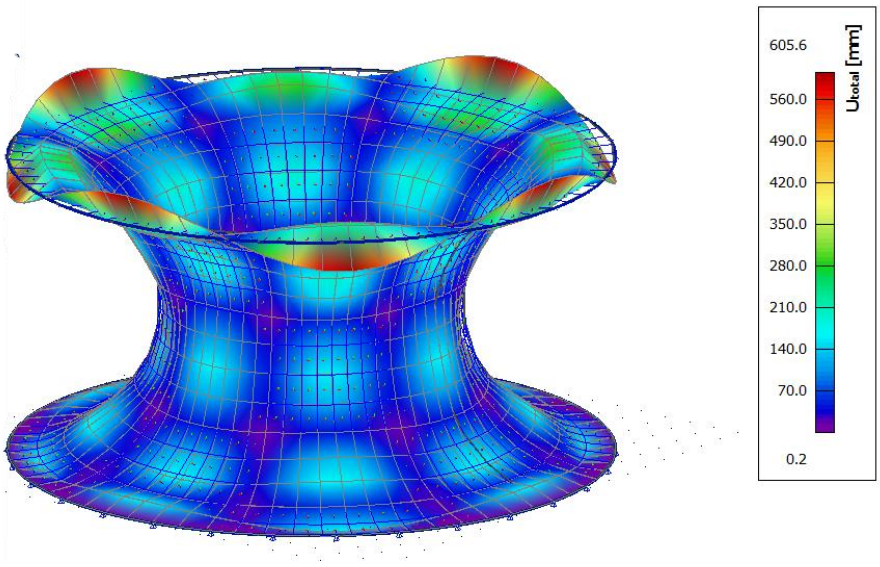
Ondanks het verwachte resultaat, blijft er randknik optreden en is de toets niet volledig betrouwbaar. Om er zeker van te zijn dat de toets betrouwbaar is, zal de randknik onderdrukt moeten worden met bijvoorbeeld een randbalk. Als de knikvorm vervolgens een ringpatroon of een schaakbordpatroon vertoont, dan kan er met zekerheid gezegd worden dat de toets betrouwbaar is.



Figuur 5.7: Membraankracht n_{yy} bij de tweede controle



Figuur 5.8: Resultaat toets I bij de tweede controle



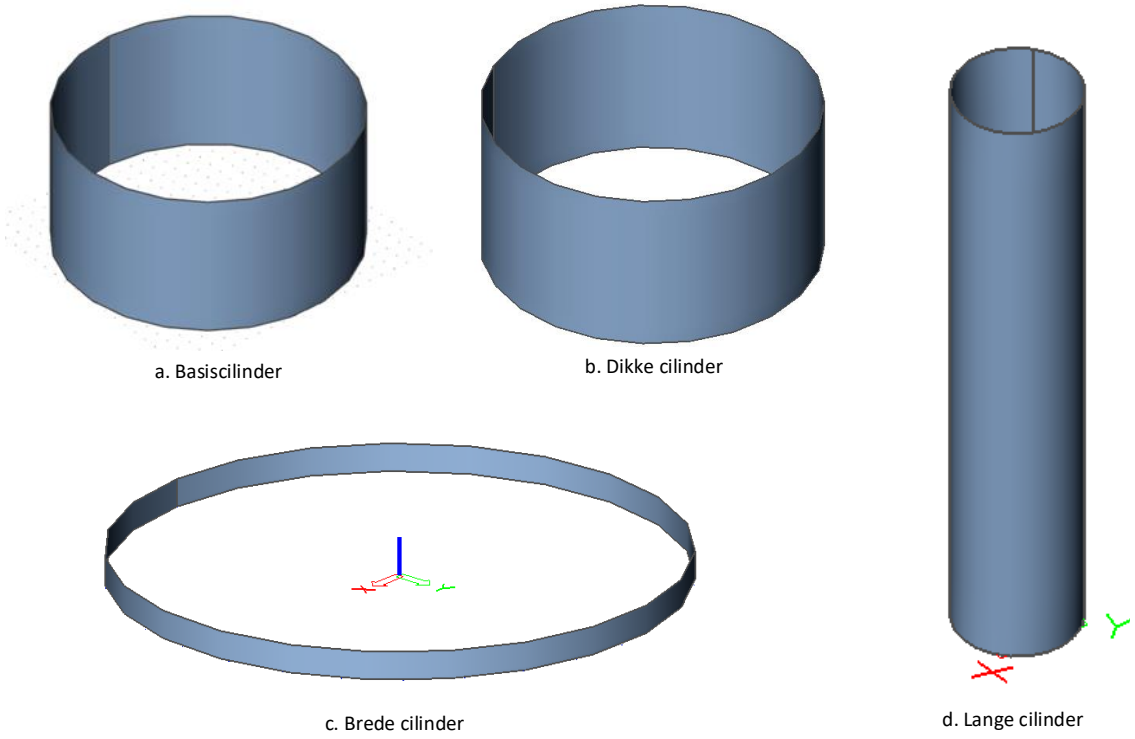
Figuur 5.9: Knikvorm ($\lambda_{cr} = 6.95$) volgens lineaire stabiliteitsberekening bij tweede controle

6. Toets II: Lokale kniklast

Als schalen bij de randen goed ondersteund zijn, zullen schalen niet globaal, maar lokaal knikken. De globale knikvorm wordt dan namelijk onderdrukt door de schaalranden. Bij deze toets worden de kritische en lokale kniklastfactor bepaald. Vervolgens wordt de kritische kniklastfactor vergeleken met de kleinste kritische kniklastfactor die bepaald is door de lineaire stabiliteitsberekening van SCIA Engineer. Als deze factoren overeenkomen, kan er geconcludeerd worden dat de toets betrouwbaar is.

6.1 Controleconstructies

Als controle zal er een cilinder gedefinieerd worden als basiscilinder. De check wordt dan op deze basiscilinder getest worden. Om te kijken welke factoren invloed hebben op de lokale kniklast, worden er meerdere cilinders getoetst, waarbij één van de dimensies 10x groter wordt gemaakt ten opzichte van de basiscilinder. De gegevens van de constructies zijn in de tabellen en figuren hieronder te zien. De verticale belasting op iedere constructie is drukkracht van 1 kN/m op de bovenrand.



Figuur 6.1: Controleconstructies Toets II

Tabel 6.1: Gegevens van de controleconstructies Toets II

	a. Basiscilinder	b. Dikke cilinder	c. Brede cilinder	d. Lange cilinder
Betonsoort	C45/55	C45/55	C45/55	C45/55
Straal	10 m	10 m	100 m	10 m
Dikte	20 mm	200 mm	20 mm	20 mm
Hoogte	10 m	10 m	10 m	100 m

6.2 Controle lokale kniklast

Zoals in de theorie is beschreven, wordt de lokale kniklastfactor bepaald met de volgende formules:

$$\lambda_{lok1} = \frac{-Et^2}{\sqrt{3(1-\nu^2)}} \frac{|k_{yy}|}{n_{xx}} C \qquad C = \frac{\left(\frac{k_{xx}}{k_{yy}} - 1 - 2\eta\frac{d}{t}\right)^2}{4\left(\frac{k_{xx}}{k_{yy}} - \frac{n_{yy}}{n_{xx}} - 3\eta\frac{d}{t}\right)\left(\frac{k_{xx}}{k_{yy}} - 2\eta\frac{d}{t}\right)} \qquad \eta = \frac{\sqrt{3(1-\nu^2)}}{1-C} \qquad [10]$$

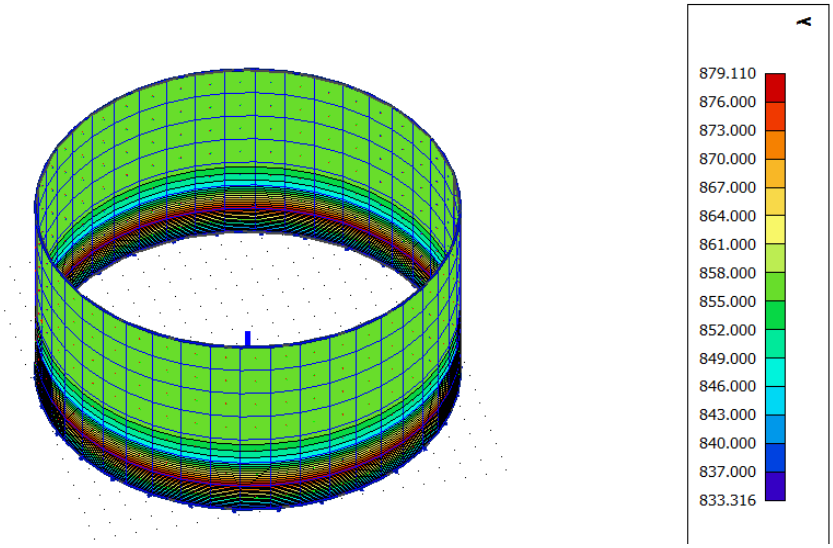
$$\lambda_{lok2} = \frac{-Et^2}{\sqrt{3(1-\nu^2)}} \frac{|k_{xx}|}{n_{yy}} C \qquad C = \frac{\left(\frac{k_{yy}}{k_{xx}} - 1 - 2\eta\frac{d}{t}\right)^2}{4\left(\frac{k_{yy}}{k_{xx}} - \frac{n_{xx}}{n_{yy}} - 3\eta\frac{d}{t}\right)\left(\frac{k_{yy}}{k_{xx}} - 2\eta\frac{d}{t}\right)} \qquad \eta = \frac{\sqrt{3(1-\nu^2)}}{1-C} \qquad [11]$$

Hierin dient de knock down factor iteratief bepaald te worden. Als initiële waarde wordt $C = \frac{1}{6}$ gebruikt. Vervolgens wordt deze waarde van C ingevuld in η en wordt de nieuwe C bepaald. Deze C wordt weer ingevuld in η enzovoorts. Na een aantal iteraties zal C bijna niet meer veranderen. Als het verschil tussen de een-na-laatste C en laatste C kleiner is dan 0.001, dan wordt het iteratieproces gestopt. De laatste waarde van C wordt gebruikt om de lokale kniklastfactor λ_{lok}te bepalen. De code van de controle is te vinden in ***bijlage E***. De krommingen konden niet nauwkeurig bepaald worden, dus deze moeten voor elke constructie handmatig ingevoerd worden. Voor imperfectie-amplitude d is aangenomen dat dit $d = \frac{1}{2}t$ is. Dit is aan de conservatieve kant, maar realistisch.

6.3 Resultaten

6.3.1Eerste controle

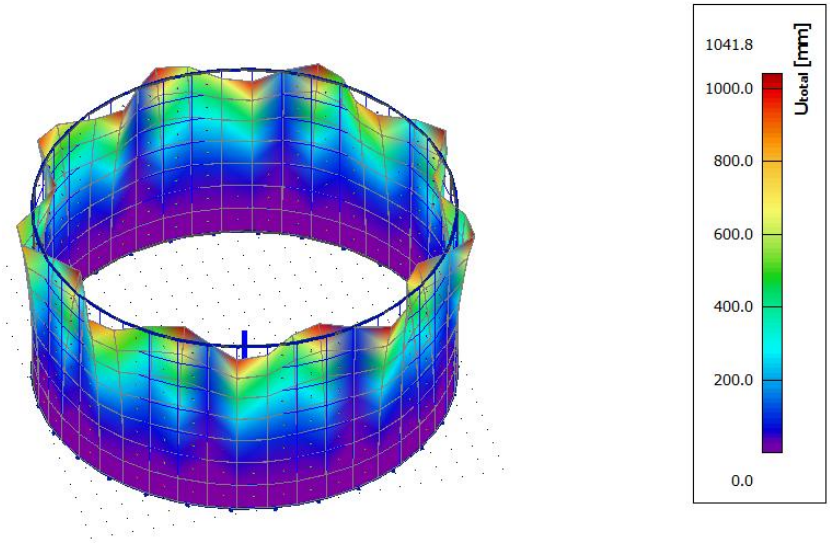
Bij de controle bleek dat de kritische kniklastfactor die berekend werd door de lineaire stabiliteitsberekening lager was dan de kritische kniklastfactor die met de check berekend was. Deze zouden ongeveer gelijk moeten zijn. De afwijking hierin komt doordat de lineaire stabiliteitsberekening ervan uitgaat dat de constructie aan de bovenrand (randknik) zal bezwijken (zie *figuur 6.3)*, terwijl de toets ervan uitgaat dat de constructie in een schaakbordpatroon bezwijkt. Om de resultaten van de toets en de lineaire stabiliteitsberekening met elkaar te kunnen vergelijken, worden de bovenranden van de constructies ook opgelegd. Deze opleggingen zijn vast in x- en y-richting, maar vrij in de z-richting.



Figuur 6.2: Kritische kniklastfactor basiscilinder volgens toets II

Tabel 6.2: Kritische kniklastfactor basiscilinder volgens lineaire stabiliteitsberekening

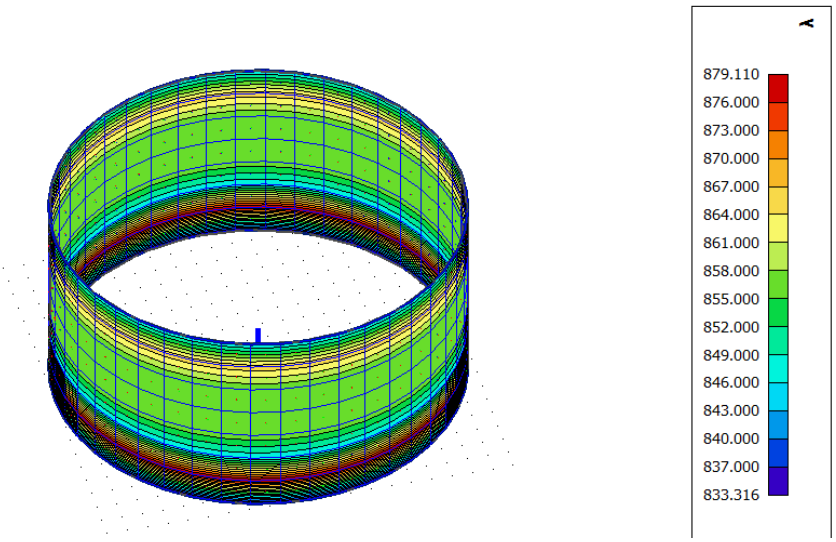
Kritische kniklastfactor
589.69
605.17
618.22
623.78
640.49



Figuur 6.3: Knikvorm ($\lambda_{cr} = 589.69$) basiscilinder volgens lineaire stabiliteitsberekening

6.3.2 Tweede controle

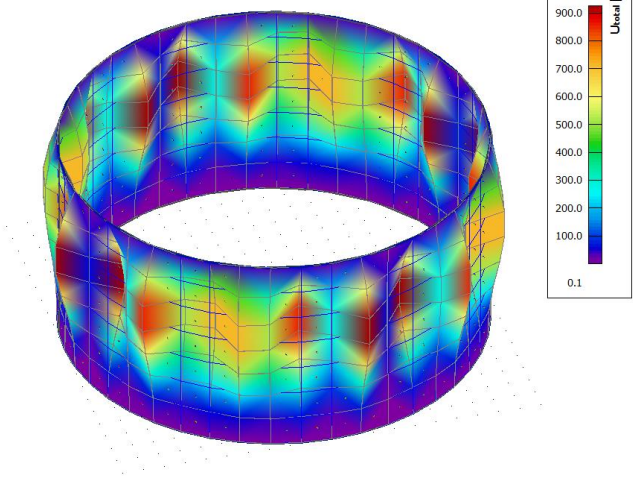
Bij de tweede controle zijn er dus opleggingen aan de bovenzijde van de constructie geplaatst. Bij de basiscilinder blijken de kritische belastingfactoren van de lineaire stabiliteitsberekening en de toets nog meer te verschillen (zie *figuur 6.4* en *tabel 6.3)*. Bij de toets is er qua input weinig veranderd, waardoor deze kritische kniklastfactoren niet veel veranderen. De knikvorm van de basiscilinder lijkt op een enkelvoudig schaakbordpatroon (zie *figuur 6.5*), dat wil zeggen dat er één laag variërend naar binnen en naar buiten knikt, in plaats van meerdere lagen onder elkaar. Dit komt doordat de constructie niet voldoende lang is gekozen, waardoor de kniklengtes zich niet zo gedragen dat de kniklastfactoren zo klein mogelijk zijn. In *figuur 6.6* is de knikvorm van de lange cilinder afgebeeld, waarbij het schaakbordpatroon wel duidelijk te zien is. Hierbij komen de kritische kniklastfactoren van de lineaire stabiliteitsberekening en toets II ook beter overeen (zie *tabel 6.4* en *figuur 6.8*).



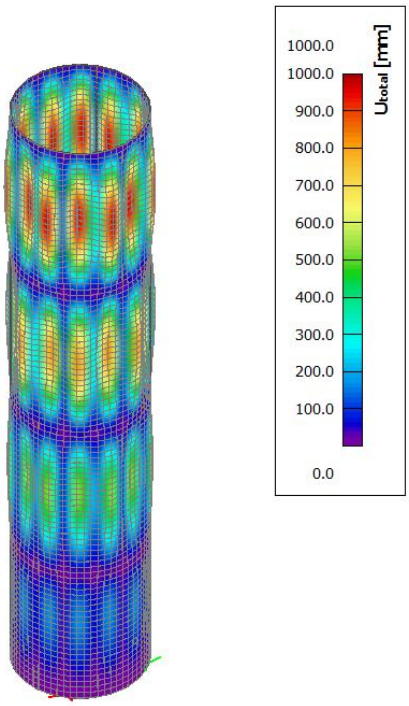
Figuur 6.4: Kritische kniklastfactor basiscilinder volgens toets II

Tabel 6.3: Kritische kniklastfactor basiscilinder volgens lineaire stabiliteitsberekening

Kritische kniklastfactor
1426.97
1461.94
1462.97
1514.38
1564.87



Figuur 6.5: Knikvorm ($\lambda_{cr} = 1426.97$) basiscilinder volgens lineaire stabiliteitsberekening



Figuur 6.6: Knikvorm ($\lambda_{cr} = 589.69$) lange cilinder volgens lineaire stabiliteitsberekening

In *figuur 6.7* is te zien hoe de knock down factor varieert over de constructie. Als de knock down factor klein is, betekent het dat de lokale kniklastfactor ook klein is. Dit is volgens de toets dus de zwakste locatie in de constructie en hier zal de constructie, volgens de theorie, dus ook lokale knik vertonen.

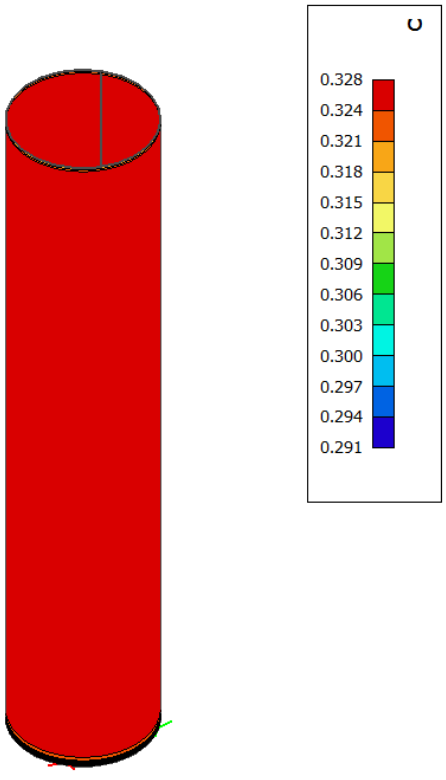
Als de resultaten uit *tabel 6.3* en *6.4* vergeleken worden, blijkt dat de lengte van de constructie invloed heeft op de globale kniklast. *Figuren 6.9* en *6.10* laten echter zien dat de lengte vrijwel geen invloed heeft op de lokale kniklast. Dit is logisch, omdat de cilinder zich bij een grote lengte meer gaat gedragen als een kolom, terwijl de lokale structuur en membraankrachten hetzelfde blijven als bij een kortere lengte. Zoals verwacht heeft de dikte van de constructie de meeste invloed op zowel de globale als de lokale kniklast. Dit blijkt uit *tabel 6.5* en *figuur 6.11*, waarbij de kniklastfactoren veel hoger zijn dan de kniklastfactoren van de basiscilinder, lange cilinder en brede cilinder.

Ook opvallend is dat de kniklastfactoren onder in de constructie erg veel variëren. Dit komt door de dwarscontractie. De constructie wil overal vervormen, maar onder in de constructie wordt dit belemmerd door de opleggingen. Dit veroorzaakt spanningen, die invloed hebben op de kniklastfactoren onder in de constructie.

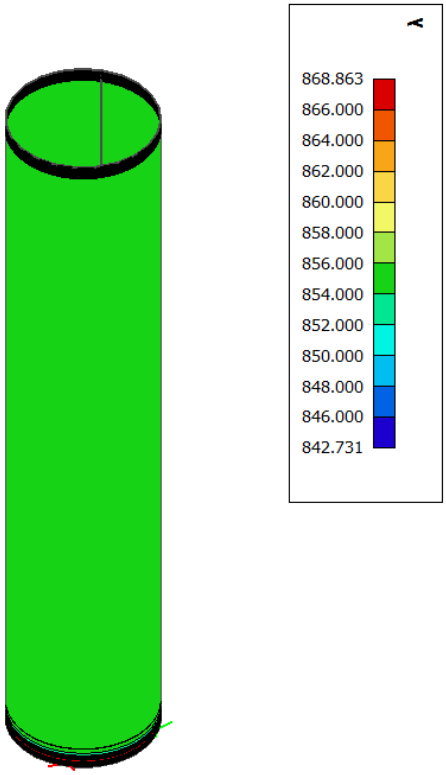
Overige figuren zijn te vinden in bijlage F.

Tabel 6.4: Kritische kniklastfactor lange cilinder volgens lineaire stabiliteitsberekening

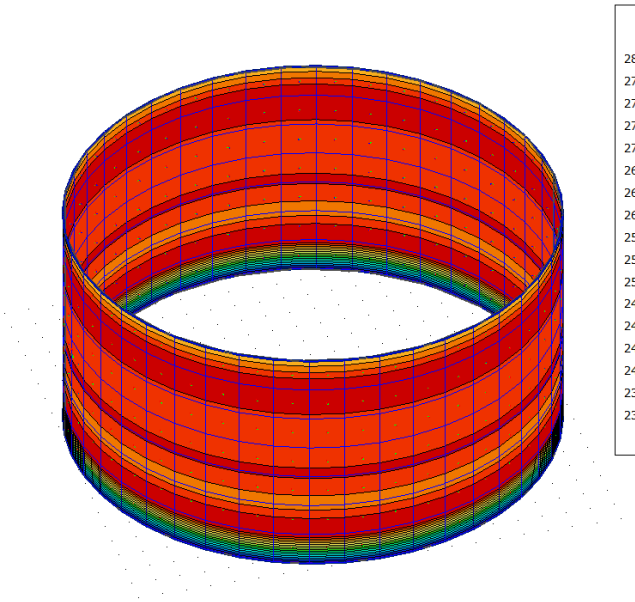
Kritische kniklastfactor
895.92
896.01
907.66
915.45
927.42



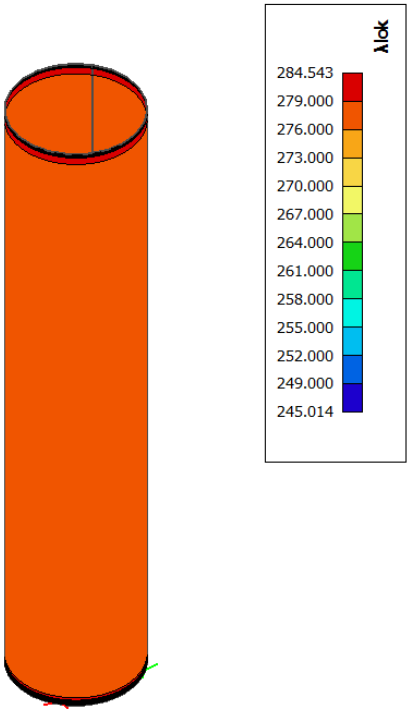
Figuur 6.7: Knock down factor lange cilinder



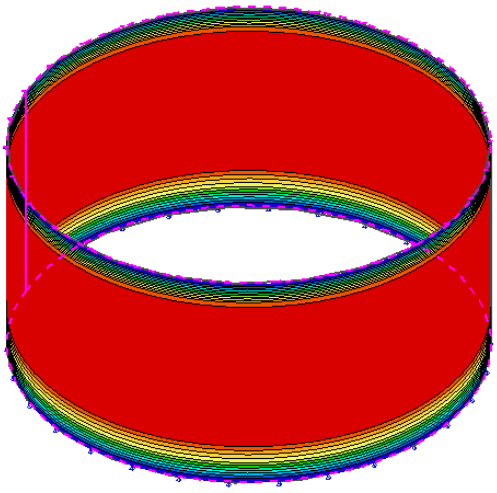
Figuur 6.8: Kritische kniklastfactor basiscilinder volgens toets II

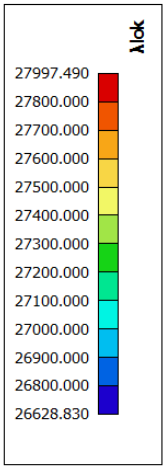


Figuur 6.9: Lokale kniklastfactor basiscilinder



Figuur 6.10: Lokale kniklastfactor lange cilinder





Figuur 6.11: Lokale kniklastfactor dikke cilinder

Tabel 6.5: Kritische kniklastfactor dikke cilinder volgens lineaire stabiliteitsberekening

Kritische kniklastfactor
90842.79
93192.33
94800.06
94804.48
95886.87

7. Conclusie

Het doel van deze studie was het programmeren van krommingen en twee toetsen, die het knikken van schaalconstructies evalueren, in SCIA Engineer. Dit heeft geleid tot de volgende conclusies:

Krommingen moeten handmatig bepaald worden, omdat in SCIA Engineer hier geen standaard functie voor gegeven is. Veel methodes om krommingen te bepalen hebben te veel geheugen nodig. Dit komt door de vele iteraties die uitgevoerd moeten worden om alle knopen in een model te analyseren. Hierdoor is het ingewikkeld om een nauwkeurige benadering te programmeren. Er is hier nog geen goede methode voor gevonden.

Een gebruikte methode benadert krommingen in één richting door een 3D-constructie in een 2D-vlak te plaatsen en de kromtestraal van iedere knoop te bepalen met behulp van de twee dichtstbijzijnde knopen. De code werkt nog niet door een fout in de code. Er zit potentie in om een goede benadering van krommingen te zijn als de fout eruit gehaald wordt.

Uit toets I is gebleken dat een op druk belaste constructie met een negatieve Gauss-kromming altijd zal knikken bij een lage belasting, ongeacht hoe groot de kromming is. Het is echter niet geverifieerd of de resultaten van deze toets betrouwbaar zijn, omdat de gekozen voorbeeldconstructie randknik vertoonde, waar de toets uitgaat van een knikvorm met een ringpatroon of schaakbordpatroon.

Ondanks dat toets II uitgaat van een versimpeling van de werkelijkheid, blijkt het wel een goed beeld te geven welke eigenschappen en dimensies invloed hebben op de lokale kniksterkte van een schaalconstructie. Een voorwaarde hiervoor is wel dat de lengte van de constructie voldoende lang is. De toets kan een handige tool zijn bij het ontwerpen van schaalconstructies, maar wegens versimpelingen en aannames is het wel beter een niet-lineaire analyse uit te voeren als de veiligheid van een definitief ontwerp bepaald moet worden.

Gebleken is dat de toetsen vooralsnog niet gebruiksvriendelijk zijn. Dit komt doordat de krommingen handmatig bepaald en ingevoerd moesten worden. Om beide toetsen goed te kunnen gebruiken, moeten de krommingen van een willekeurige constructie gemakkelijk bepaald kunnen worden.

8. Aanbevelingen

De volgende punten worden aanbevolen om verder uit te werken:

- Omdat er potentie voor een goede benadering van krommingen in de tweede methode zit, wordt het aanbevolen om de fout uit de code te halen en de toets te evalueren.
- Het benodigde geheugen blijft een groot obstakel bij de bepaling van krommingen. Dit geheugen wordt voornamelijk gebruikt om meerdere keren alle knopen in een model te bekijken. Verder onderzoek naar een methode om alle coördinaten in SCIA Engineer logisch te ordenen en in te laden in Design Forms Builder zou veel geheugen kunnen besparen. Het is mogelijk om coördinaten van de knopen in SCIA Engineer van klein naar groot te ordenen op bijvoorbeeld de x-coördinaten. Als de koppeling gemaakt kan worden met Design Forms Builder en deze coördinaten met deze ordening ingeladen kan worden, is het grootste probleem opgelost.
- Om te controleren of toets I betrouwbaar is, kan er een randbalk aan een constructie met een negatieve Gauss-kromming toegevoegd worden. Dit zou de randknik moeten onderdrukken en een knikvorm met een ringpatroon of schaakbordpatroon geven.
- De lineaire knikberekening is een belangrijke toets het constructiegedrag weergeeft bij knikken. Het geeft verder geen informatie over de oorzaak van het knikken. De geprogrammeerde toetsen in dit rapport kunnen de lineaire knikberekening niet vervangen. Als deze anders geformuleerd worden, kunnen ze wellicht inzicht geven in de oorzaak van het falen. Dit legt de focus niet op het vervangen en versimpelen van ingewikkelde knikberekeningen, maar op het verbeteren van ontwerpen van schaalconstructies.

Verwijzingen

curvature. (sd). Opgeroepen op 05 29, 2020, van web.ma.utexas: https://web.ma.utexas.edu/users/og/curvature.html?#globalCurvatureDefn

Hoogenboom. (2020). Opgeroepen op 06 17, 2020, van Handout 1: http://homepage.tudelft.nl/p3r3s/b17_handout_1_lockdown.pdf

Hoogenboom. (2020). Opgeroepen op 17 06, 2020, van Handout 12: http://homepage.tudelft.nl/p3r3s/b17_handout_12_lockdown.pdf

Hoogenboom. (2020). Opgeroepen op 06 17, 2020, van Handout 13: http://homepage.tudelft.nl/p3r3s/b17_handout_13_lockdown.pdf

Hoogenboom. (2020). *Handout 2*. Opgeroepen op 06 17, 2020, van http://homepage.tudelft.nl/p3r3s/b17_handout_2_lockdown.pdf

Hoogenboom. (2020). *Notes on Shell structures*. Opgeroepen op 05 14, 2020, van TU Delft: http://homepage.tudelft.nl/p3r3s/b17_schedule.html

SCIA. (2020). Opgeroepen op 06 15, 2020, van SCIA: https://www.scia.net/nl

SCIA Design Forms. (sd). Opgeroepen op 06 15, 2020, van design forms SCIA: https://design-forms.scia.net/help/50/en/Index.htm

SCIA webhelp. (2019). Opgeroepen op 06 15, 2019, van help.SCIA: https://help.scia.net/webhelplatest/en/

Vergeer, M. (2016, 06 13). *Checks voor schaalconstructies met SCIA Engineer.* Delft. Opgeroepen op 04 29, 2020, van http://homepage.tudelft.nl/p3r3s/BSc_projects/eindrapport_vergeer.pdf

Vuik, C. (2018). *Numerical Methods for Ordinary Differential Equations.* Delft: Delft Academic Press. Opgeroepen op 06 12, 2020

Bijlage A: Code voor kromming van methode 1

```
double[] X = new double[];
double[] Y = new double[];
double[] Z = new double[];
double[] a = new double[];
double[] x1 = new double[];
double[] y1 = new double[];
double[] z1 = new double[];
double[] p1 = new double[];
object[] α = new object[];
object M = new Matrix(6, 6);
object m = new Matrix(6, 6);

TEXT(" 5 dichtstbijzijnde punten");
FOR(i, 0, double punten) {
   X[i] = IO.Member2D.Point[i].X;
   Y[i] = IO.Member2D.Point[i].Y;
   Z[i] = IO.Member2D.Point[i].Z; }
s = 0.5;
WHILE(GetLength(a) < 6) {
   FOR(j, 0, GetLength(X)) {
      IF(SQRT((x-X[j])↑2←+(y-Y[j])↑2←+(z-Z[j])↑2←) <= s && j != p1[0] && j != p1[1] && j != p1[2]
&&
      j != p1[3] && j != p1[4] && j != p1[5]) {
         af = SQRT((x-X[j])↑2←+(y-Y[j])↑2←+(z-Z[j])↑2←);
         a.Add(af);
         p1.Add(j);
         x1.Add(X[j]);
         y1.Add(Y[j]);
         z1.Add(Z[j]); }
      IF(GetLength(a) == 6) { Break(); } }
   IF(GetLength(a) == 6) { Break(); }
   ELSE { s = s+0.1; } }

TEXT("matrix maken ");
FOR(i, 0, 5) {
   M[i][0] = (x1[i])↑2←;
   M[i][1] = x1[i]*y1[i];
   M[i][2] = (y1[i])↑2←;
   M[i][3] = x1[i];
   M[i][4] = y1[i];
   M[i][5] = 1; }

TEXT("swap rows ");
FOR(n, 0, 5) {
   double r = 0;
   IF(n < 5) {
      IF(ABS(M[n][0]) == ABS(M[n][1])) { double r = 1; } }
   IF(M[n][n] == 0) { IF(n < 1 && M[n+5][n] != 0) { r = 5; }
      IF(n < 2 && M[n+4][n] != 0) { r = 4; }
      IF(n < 3 && M[n+3][n] != 0) { r = 3; }
      IF(n < 4 && M[n+2][n] != 0) { r = 2; }
      IF(M[n+1][n] != 0) { r = 1; } }
   double swap_z = z1[n];
```

```
   z1[n] = z1[n+r];
   z1[n+r] = swap_z;
   FOR(p, 0, 5) {
      double swap = M[n][p];
      M[n][p] = M[n+r][p];
      M[n+r][p] = swap; } }

TEXT("matix oplossen ");
FOR(k, 0, 4) {
   FOR(i, k+1, 5) {
      m[i][k] = M[i][k]/M[k][k];
      IF(IsNaN(m[i][k])) { m[i][k] = 0; }
      IF(IsInfinity(ABS(m[i][k]))) { m[i][k] = 0; }
      M[i][k] = 0;
      FOR(j, k+1, 5) {
         M[i][j] = M[i][j]-m[i][k]*M[k][j]; }
      z1[i] = z1[i]-m[i][k]*z1[k]; } }

IF(IsNaN(z1[5]/M[5][5]) || IsInfinity(ABS(z1[5]/M[5][5]))) { α[5] = 0; } ELSE { α[5] = z1[5]/M[5][5]; }
IF(IsInfinity(1/M[4][4])) { α[4] = 0; } ELSE { α[4] = (1/M[4][4])*(z1[4]-(α[5]*M[4][5])); }
IF(IsInfinity(1/M[3][3])) { α[3] = 0; } ELSE { α[3] = (1/M[3][3])*(z1[3]-(α[5]*M[3][5]+α[4]*M[3][4])); }
IF(IsInfinity(1/M[2][2])) { α[2] = 0; } ELSE { α[2] = (1/M[2][2])*(z1[2]-
   (α[5]*M[2][5]+α[4]*M[2][4]+α[3]*M[2][3])); }
IF(IsInfinity(1/M[1][1])) { α[1] = 0; } ELSE { α[1] = (1/M[1][1])*(z1[1]-
   (α[5]*M[1][5]+α[4]*M[1][4]+α[3]*M[1][3]+α[2]*M[1][2])); }
IF(IsInfinity(1/M[0][0])) { α[0] = 0; } ELSE { α[0] = (1/M[0][0])*(z1[0]-
   (α[5]*M[0][5]+α[4]*M[0][4]+α[3]*M[0][3]+α[2]*M[0][2]+α[1]*M[0][1])); }

TEXT("vlak functie ");
double x2 = x;
double y2 = y;
double h = 0.1;
double f(double x2, double y2) {
   z = α[0]*x2↑2←+α[1]*x2*y2+α[2]*y2↑2←+α[3]*x2+α[4]*y2+α[5];
   return z; }
double z = f(x2, y2);

TEXT("afgeleiden ");
double fx(double x2, double y2, double h) {
   fx = (f(x2+h, y2)-f(x2-h, y2))/(2*h);
   return fx; }
double fy(double x2, double y2, double h) {
   fy = (f(x2, y2+h)-f(x2, y2-h))/(2*h);
   return fy; }
double fxx(double x2, double y2, double h) {
   fxx = (f(x2+h, y2)-2*f(x2, y2)+f(x2-h, y2))/h↑2←;
   return fxx; }
double fyy(double x2, double y2, double h) {
   fyy = (f(x2, y2+h)-2*f(x2, y2)+f(x2, y2-h))/h↑2←;
   return fyy; }
double fx = fx(x2, y2, h);
double fy = fy(x2, y2, h);
double fxx = fxx(x2, y2, h);
double fyy = fyy(x2, y2, h);
```

```
double fxy = α[1];
TEXT("kromming ");
kx = ABS(fxx)/((1+ABS(fx)↑2←)↑3/2←);
ky = ABS(fyy)/((1+ABS(fy)↑2←)↑3/2←);
```

ESA-ID's:

kx	Result.1
ky	Result.2
punten	MemberData.punten
x	Point.X
y	Point.Y
z	Point.Z

De variabele punten moet worden toegevoegd aan member data.

Bijlage B: Code voor kromming van methode 2

```
double[] X = new double[];
double[] Y = new double[];
double[] Z = new double[];
double[] a = new double[];
double[] a2 = new double[];
double[] x1 = new double[];
double[] y1 = new double[];
double[] z1 = new double[];
double[] p1 = new double[];
object[] α = new object[];
object M = new Matrix(6, 6);
object m = new Matrix(6, 6);

TEXT(" 2 dichtsbijzijnde punten");
FOR(i, 0, double punten) {
   X[i] = IO.Member2D.Point[i].X;
   Y[i] = IO.Member2D.Point[i].Y;
   Z[i] = IO.Member2D.Point[i].Z; }

s = 0.5;
WHILE(GetLength(a) < 2) {
   FOR(j, 0, GetLength(X)) {
      IF(SQRT((x-X[j])↑2←+(y-Y[j])↑2←+(z-Z[j])↑2←) <= s && ABS(y-Y[j]) <= 0.1
       && j != p1[0] && j != p1[1]) {

         af = SQRT((x-X[j])↑2←+(z-Z[j])↑2←);
         a.Add(af);
         p1.Add(j);
         x1.Add(X[j]);
         y1.Add(Y[j]);
         z1.Add(Z[j]); }
      IF(GetLength(a) == 2) { Break(); } }
   IF(GetLength(a) == 2) { Break(); }
   ELSE { s = s+0.1; } }

a[2] = SQRT((x1[1]-x1[0])↑2←+(z1[1]-z1[0])↑2←);

double fo(double x1, double z1) {
   Opp = 0.5*a[2]*((SQRT(-(a[1])↑2←+4*(a[2])↑2←)*a[1])/(2*a2));
   return Opp; }
double Opp = fo(a, a2);

ry = (ABS(a[0])*ABS(a[1])*ABS(a[2]))/(4*Opp);
kyy = 1/ry;
```

ESA-ID's:

ry	Result.1
kyy	Result.2
punten	MemberData.punten
x	Point.X
y	Point.Y
z	Point.Z

De variabele punten moet worden toegevoegd aan member data.

Bijlage C: Code toets I

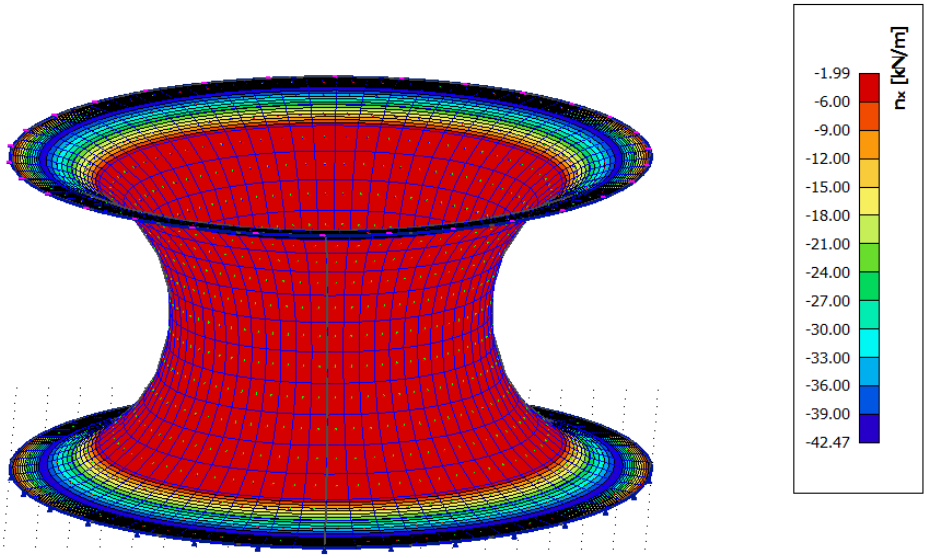
```
ax = SQRT(x↑2←+y↑2←);

kxx = 1/ax;
kyy = -1/ay;

Toets1 = nxx*ABS(kxx)+nyy*ABS(kyy);
```

ESA-ID's:

kxx	Result.1
kyy	Result.2
Toets1	Result.3
nxx	InternalForces.nx
nyy	InternalForces.ny
ay	MemberData.ay
x	Point.X
y	Point.Y

De variabele punten moet worden toegevoegd aan member data.

Bijlage D: Overige resultaten van toets I



Figuur D1: Membraankracht n_{xx} bij tweede controle

Bijlage E: Code van toets II

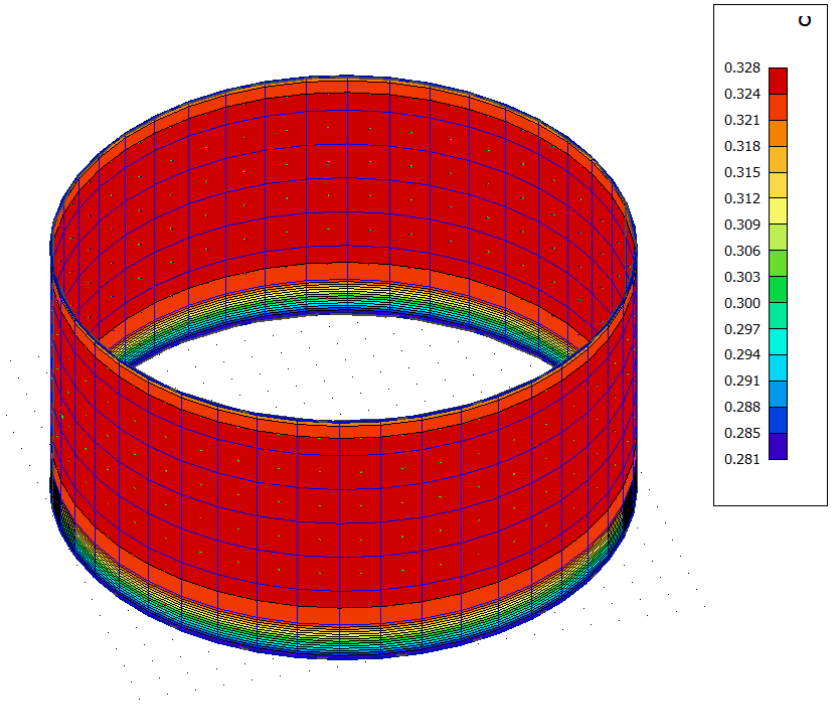
```
kxx = 0;
kyy = 1/a;

double d = (1/2)*t;
object η = new object[];
object C1 = new object[];

C1[0] = 1/6;
FOR(i, 1, 100) {
   η[i] = SQRT(3*(1-v↑2←))/(1-C1[i-1]);
   C1[i] = (kxx/kyy-1-2*η[i]*(d/t))↑2←/(4*(kxx/kyy-nyy/nxx-3*η[i]*(d/t))*(kxx/kyy-2*η[i]*(d/t)));
   IF(ABS(C1[i]-C1[i-1]) < 0.001) { Break(); } }

double C = C1[i];

λ = ((-E*t↑2←)/SQRT(3*(1-v↑2←)))*(ABS(kyy)/nxx);
λlok = λ*C;
```

ESA-ID's:

kxx	Result.1
kyy	Result.2
C	Result.3
λ	Result.4
λlok	Result.5
nxx	InternalForces.nx
nyy	InternalForces.ny
a	MemberData.a
t	Point.H
E	Point.Material.EC.Ecm
v	Point.Material.EC.Poisson

De variabele punten moet worden toegevoegd aan member data.

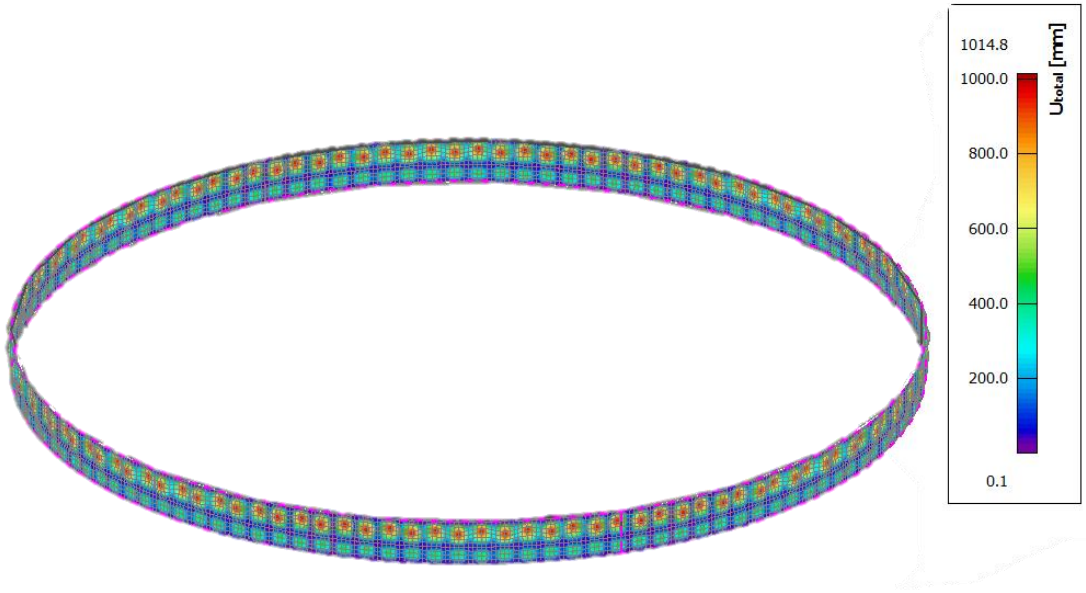
Bijlage F: Overige resultaten van toets II



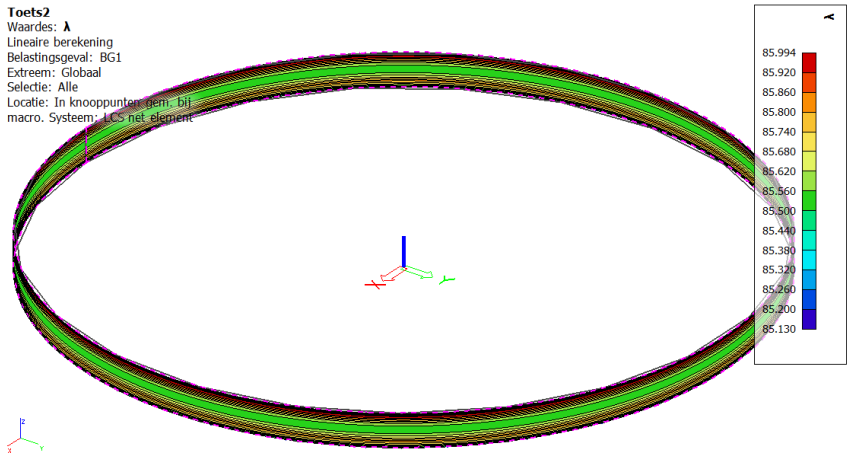
Figuur F1: Knock down factor basiscilinder

Tabel F1: Kritische kniklastfactor lange cilinder volgens lineaire stabiliteitsberekening

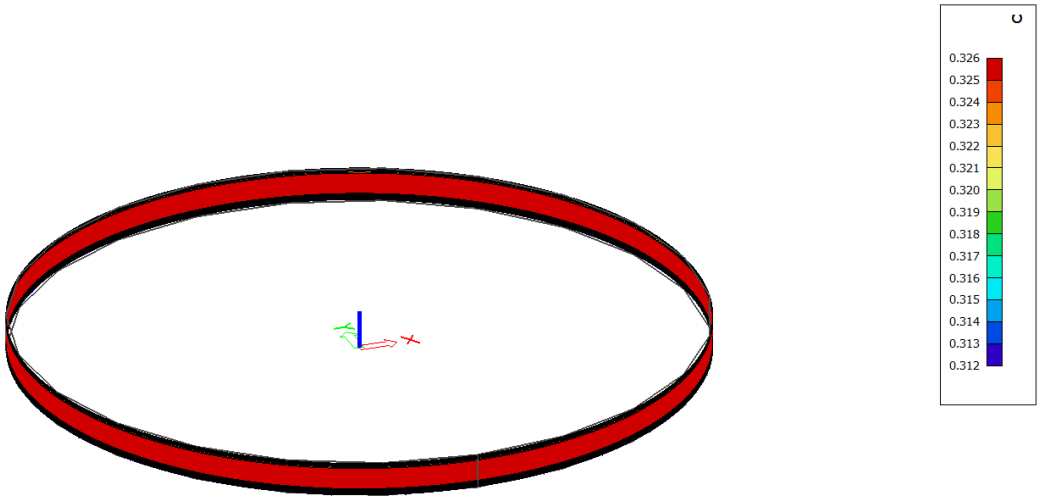
Kritische kniklastfactor
39.74
39.76
39.77
39.83
39.86



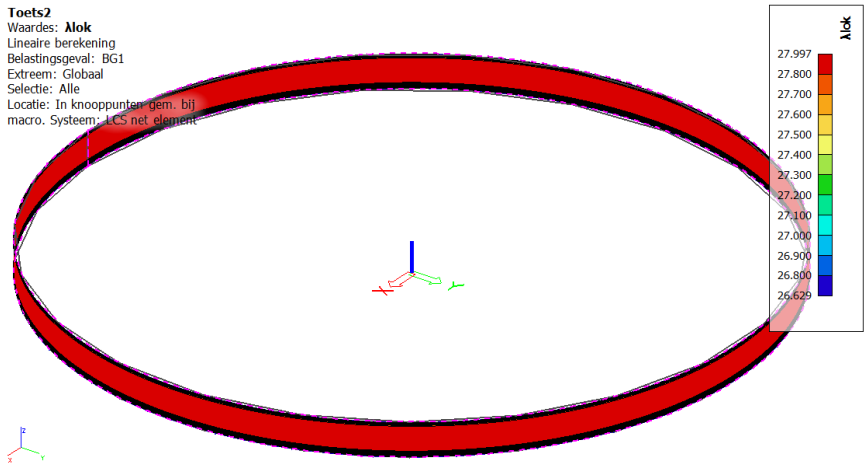
Figuur F2: Knikvorm ($\lambda_{cr} = 39.74$) brede cilinder volgens lineaire stabiliteitsberekening



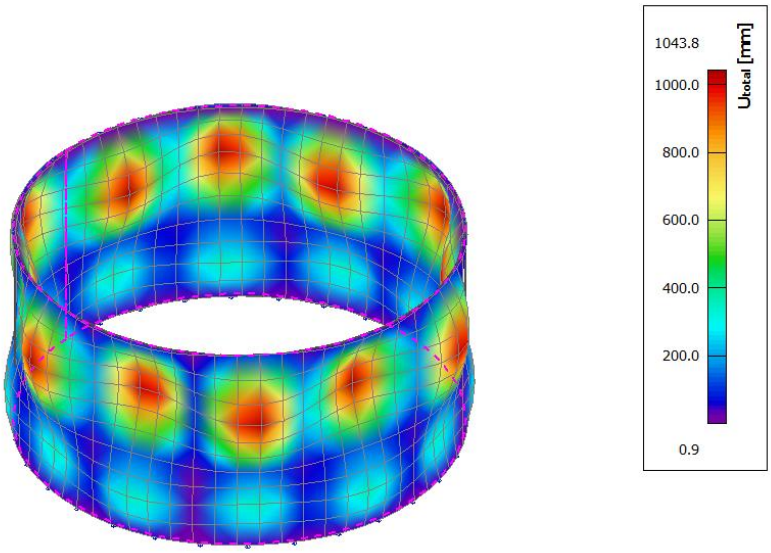
Figuur F3: Kritische kniklastfactor brede cilinder volgens toets II



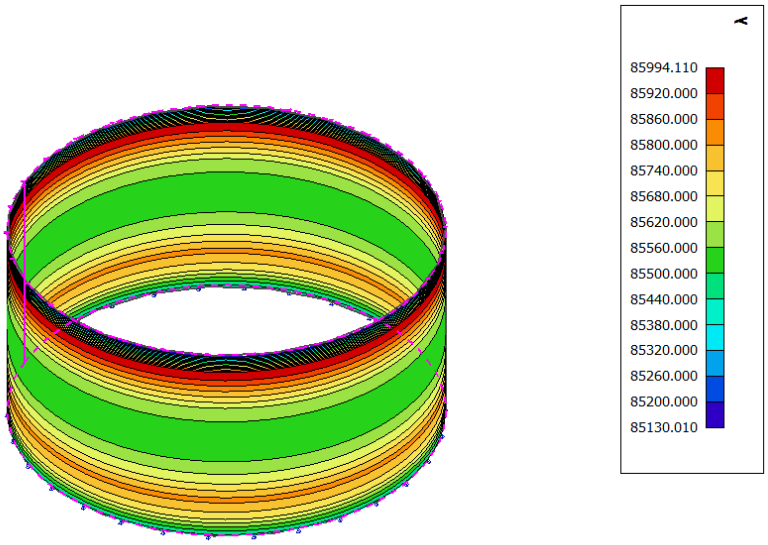
Figuur F4: Knock down factor brede cilinder



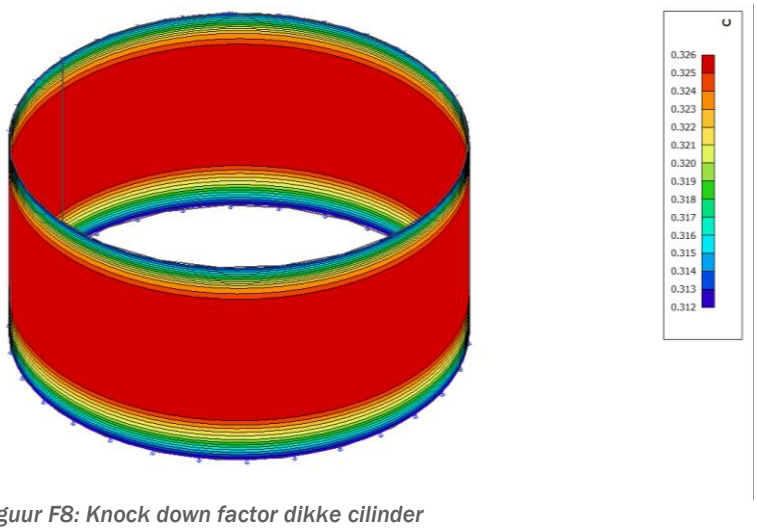
Figuur F5: Lokale kniklastfactor brede cilinder



Figuur F6: Knikvorm ($\lambda_{cr} = 90842.79$) dikke cilinder volgens lineaire stabiliteitsberekening



Figuur F7: Kritische kniklastfactor brede cilinder volgens toets II



Figuur F8: Knock down factor dikke cilinder