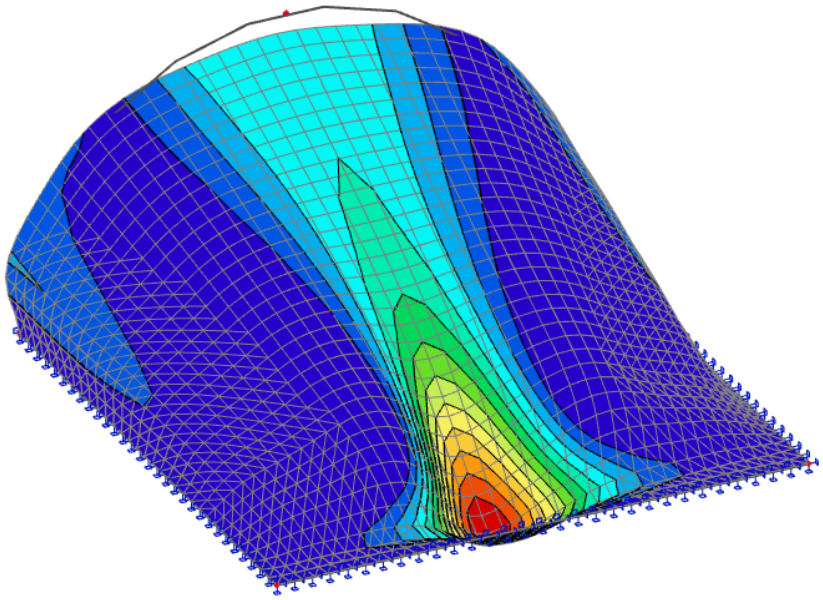
TU Delft – Bachelor Eindproject CTB3000

Invarianten van gecombineerde tensoren

Uitknikken van schaalconstructies



Jeffrey van Hulst | 4474392 | Juni 2018

1e begeleider dr.ir. P.C.J. Hoogenboom
2e begeleider dr.ir. F.P. van der Meer

TU Delft Delft University of Technology

Invarianten van gecombineerde tensoren

Uitknikken van schaalconstructies

Bachelor Eindproject

Jeffrey van Hulst

Juni 2018

Faculteit Civiele Techniek en Geowetenschappen – Technische Universiteit Delft

Voorwoord

Dit rapport is tot stand gekomen tijdens het Bachelor Eindproject in het derde jaar van de studie Civiele Techniek aan de Technische Universiteit Delft. Dit onderzoek is ter afronding van de Bachelor fase voor de titel *Bachelor of Science* in Civiele Techniek.

In dit rapport is onderzoek gedaan naar de algemene knikformule van schaalconstructies. Het rapport is geschreven in de periode van april 2018 tot en met juni 2018.

Dit rapport was niet mogelijk geweest zonder de hulp van mijn begeleiders, dr.ir. P.C.J. Hoogenboom en dr.ir. F.P. van der Meer. Ik wil ze dan ook hartelijk bedanken voor de tips en feedback tijdens het onderzoek. Ik heb tijdens dit onderzoek veel nieuwe dingen geleerd en ben ook blij dat ik dit onderwerp heb kunnen onderzoeken.

Ik wens u veel leesplezier toe,

Jeffrey van Hulst

Delft, juni 2018

Abstract

Shell structures are made up of elements that are relatively thin compared to the other dimensions. In addition, these elements are curved in one or two directions. These constructions are very economical in terms of the power transfer and the material used. Nevertheless, shell structures can collapse and buckle at a certain load. The behaviour of the buckling can currently be described using finite element methods. However, this method requires a lot of time and computing power, so it is preferable to have a general formula that can predict buckling. It is currently known that the curvatures and membrane forces play a role in this formula. The exact connection between them is not yet known. In addition, the starting point is that the formula which describes buckling is independent of the coordinate system and thus contains an invariant. A coordinate system is an arbitrary concept, how a system of axes is defined should not affect the physical laws that apply to the buckling of shell structures.

To find the formula which describes buckling, there is searched for invariants of combined tensors. Several programs have been written for this in the Python programming environment that systematically try all possible combinations. Then these combinations can be tested in two different ways whether they are invariant. Firstly, there is the analytical method that simplifies a certain combination as much as possible through the built-in simplification function in Python. When the angle α disappears after the simplification, the specific combination is an invariant. This method takes a relatively long time. Nevertheless, about 200,000 combinations have been investigated with this method. The second method to investigate invariants is the numerical method. In this method, a certain combination is plotted against the angle of the coordinate system α. If this yields a constant value, the combination is an invariant (after all, it is then independent of the angle α). This method is a lot faster than the analytical method. In total, approximately twelve million combinations have been tested with these two methods. However, this did not yield any new invariants, all invariants could be reduced to already known solutions. From this, the following conjecture is formulated:

Conjecture:

Of all combinations that can be made of the elements of two tensors only six combinations are independent invariants.

Invariants of single and combined tensors after investigation and described in the literature are shown below. Combinations of these invariants also yield invariants.

$$n_{xx} + n_{yy} \quad \& \quad k_{xx} + k_{yy}$$

$$n_{xx}n_{yy} - {n_{xy}}^2 \quad \& \quad k_{xx}k_{yy} - {k_{xy}}^2$$

$$n_{xy}(k_{xx} - k_{yy}) - k_{xy}(n_{xx} - n_{yy})$$

$$n_{xx}k_{yy} - 2n_{xy}k_{xy} + n_{yy}k_{xx}$$

Subsequently, buckling formulas have been formulated using these invariants, which can be further investigated. These buckling formulas also have to be accurate in terms of dimensions. These buckling

formulas were then programmed into SCIA Engineer using Design Forms Builder. This results in contour plots in SCIA Engineer of each buckling formula for different shell structures. These contour plots show the values of the load factor λ. From this it can be concluded that buckling formula 1 (λ_1) has the best prediction of the buckling behaviour for the time being. This conclusion might still change if the curvatures are also calculated in the local coordinate system. This has not yet been done in this study, so drawing real conclusions is therefore very difficult.

$$\lambda_1 = \mathrm{E} \cdot t^2 \cdot \frac{k_{xx} + k_{yy}}{n_{xx} + n_{yy}}$$

To get better results from the buckling formulas in SCIA Engineer, it is necessary to work with coordinates in the local coordinate system. Currently, the calculations are made in the global coordinate system, this causes deviations in the results of the curvatures. These curvatures are then used in the calculation of the load factor λ. Implementing the coordinates in the local coordinate system is not an easy task, extra research will be needed to get it working properly.

Inhoudsopgave

Lijst met figuren en tabellen

Symbolenlijst

$a = Straal\ [m]$

$a_i = Gradiënten\ (umbilic)$

$\alpha = Rotatie\ assenstelsel\ [rad]$

$\beta = Factor$

$E = Elasticiteitsmodulus\ \left[{}^{N}/_{mm^2}\right]$

$I = Eenheidsmatrix$

$I_i = Invarianten\ van\ gecombineerde\ tensoren$

$k_{ij} = Kromming\ \left[{}^{1}/_{mm}\right]$

$\lambda =$ Belastingfactor $[-]$

$\lambda =$ Schaalfactor (eigenwaarde probleem)

$\lambda_i = Knikformules$

$n_{ij} = Membraankracht\ \left[{}^{N}/_{mm}\right]$

$N_i = Interessante\ invarianten$

$R = Rotatiematrix$

$R^{-1} = Inverse\ rotatiematrix$

$R^{T} = Getransponeerde\ rotatiematrix$

$\sigma = Spanning\ \left[{}^{N}/_{mm^2}\right]$

$t = Schaaldikte\ [mm]$

$U_i = Invarianten\ van\ umbilics$

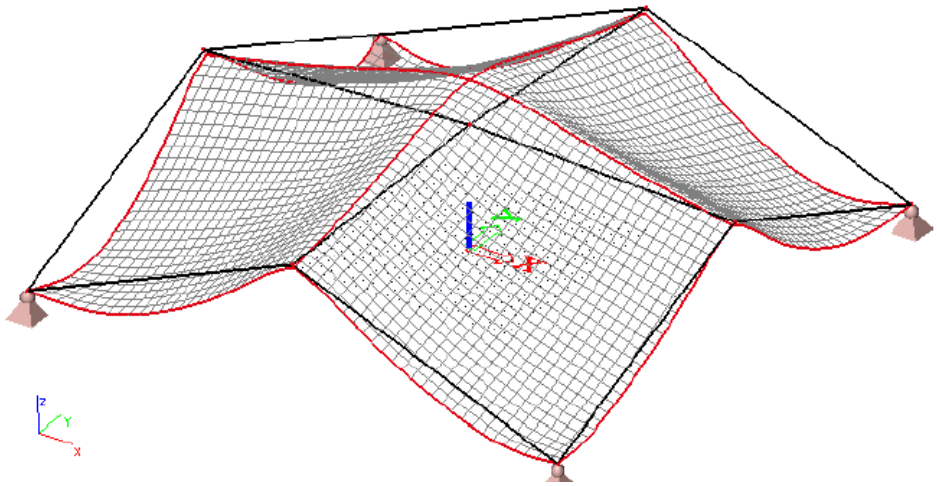
$x, y = Oorspronkelijke\ coördinaten$

$\bar{x}, \bar{y} = Nieuwe\ coördinaten$

$z(x, y) = Vergelijking\ van\ het\ vlak$

Inleiding

Met de huidige technologie is het mogelijk om te berekenen vanaf welke belasting een schaalconstructie uitknikt. Dit wordt momenteel gedaan met behulp van Eindige Elementenmethoden (E.E.M). Schaalconstructies worden daarbij in computersoftware opgedeeld in kleine elementen (*figuur I*). Door eisen te stellen aan de afzonderlijke elementen onderling (gemeenschappelijke randvoorwaarden), kan een oplossing voor het gehele systeem worden verkregen. Op deze manier ontstaan vele vergelijkingen – bestaande uit constitutieve en kinematische vergelijkingen – die in een matrix kunnen worden gezet. Oplossen van de matrix geeft de totale oplossing van de gehele constructie. Om tot een goede oplossing te komen is het noodzakelijk om de elementen zo klein mogelijk te maken. Dit vergt echter wel meer rekenkracht van de computer om het systeem op te lossen. Een compromis tussen precisie van de oplossing en benodigde rekentijd zal gemaakt moeten worden.



Figuur I Voorbeeld Eindige Elementenmethode (SCIA Engineer) (Hoogenboom, Design of a reinforced concrete 4-hypar shell with edge beams, 2016)

In dit Bachelor Eindproject zal de tegenhanger van de hierboven genoemde Eindige Elementenmethode centraal staan. Namelijk een algemene formule die geschikt is om knik van schaalconstructies te voorspellen. Momenteel is deze formule door wetenschappers nog niet gevonden. Mocht deze formule wel gevonden worden, dan zou het proces om de kniklast te bepalen een stuk makkelijker worden. Ook is er dan geen speciale software en computer rekenkracht meer nodig.

In de zoektocht naar deze algemene formule zijn twee belangrijke zaken ontdekt;

- Knik van schaalconstructies hangt af van twee tensoren, namelijk de membraankrachttensor en de krommingstensor
- Met bovendien als uitgangspunt dat knik niet afhangt van de oriëntatie van het assenstelsel

Uit deze zaken volgt een belangrijke constatering; aangezien knik onafhankelijk is van het assenstelsel, betekent dit dat de algemene formule van knik een *invariant* (zie ook *§1.3*) moet bevatten. Daarnaast is

knik niet afhankelijk van één specifieke tensor, maar van twee verschillende tensoren. De algemene formule moet dus ook een combinatie bevatten van deze tensoren, hoe deze combinatie gemaakt moet worden blijft de grote vraag.

Om de algemene knikformule te achterhalen, zal een stapje terug gedaan moeten worden, door het probleem op wiskundige wijze te benaderen. Dit probleem kan dan worden geformuleerd als; vind combinaties van elementen van tensoren die invariant zijn. Oplossingen van dit wiskundig probleem kunnen mogelijk de oplossing zijn van het knikprobleem.

Probleemstelling

Het uitknikken van schaalconstructies kan momenteel worden berekend met een Eindige Elementenmethode, echter vergt deze methode de nodige rekentijd. Een algemene formule die het uitknikken van schaalconstructies beschrijft is er daarentegen nog niet.

Om dit probleem op te lossen, moeten invarianten van gecombineerde tensoren worden gevonden. Er zijn al een aantal invarianten bekend. Zo heeft Marijn Drillenburg in mei 2017 de volgende invariant gevonden:

$$n_{xx}k_{yy} - 2n_{xy}k_{xy} + n_{yy}k_{xx}$$

Andere invarianten van twee tensoren die al bekend zijn;

$$n_{xy}(k_{xx} - k_{yy}) - k_{xy}(n_{xx} - n_{yy})$$

$$(n_{xx} + n_{yy})(k_{xx} + k_{yy})$$

Daarbij worden alleen invarianten meegerekend die niet uit te drukken zijn in bestaande invarianten. Als een invariant is uit te drukken in een bestaande invariant is de oplossing een triviale oplossing, omdat er in wezen geen echte nieuwe oplossing is gevonden.

Doelstelling

Vind invarianten van gecombineerde tensoren, die mogelijk gebruikt kunnen worden in de algemene formule voor knik bij schaalconstructies. Als deze formule bekend is, kan het uitknikken van een schaalconstructie zoals in *figuur II* relatief eenvoudig worden voorspeld.

Voor dit Bachelor Eindproject is het echter niet realistisch om te denken dat de algemene formule eenvoudig kan worden gevonden. Deze zoektocht is als het ware zoeken naar een speld in de hooiberg, het is niet onmogelijk, maar enige vorm van geluk is wel noodzakelijk.

De focus in dit rapport zal dan ook met name liggen op het vinden van een invariant, of deze dan ook gebruikt kan worden voor de algemene knikformule, zal in eerste instantie minder belangrijk zijn. Het zou natuurlijk kunnen dat een gevonden invariant een rol speelt in een ander natuurkundig proces, in plaats van het knikproces. Het is dus noodzakelijk om alle opties open te houden.

Figuur II Uitknikken schaalconstructie (Skogh, 1979)

Methode

Om mogelijke knikformules op te stellen, zal dit probleem eerst vereenvoudigd moeten worden tot een puur wiskundig probleem. Aangezien bekend is dat knik afhangt van twee tensoren en dat deze onafhankelijk zijn van het assenstelsel, zal er eerst gezocht moeten worden naar invarianten van gecombineerde tensoren. Voor het vinden van nieuwe invarianten zijn er verschillende methoden te gebruiken.

Ten eerste is het zoeken op een systematische wijze, door middel van programmeren, een erg efficiënte methode. Mocht dit nog niet het gewenste effect hebben, dan kan ook handmatig via een trial-and-error methode combinaties worden getoetst op invarianten. Daarnaast kunnen combinaties van tensoren op visuele wijze worden uitgezet tegen de hoek van het assenstelsel, als dit een constante waarde oplevert is de combinatie invariant.

Nadat verschillende invarianten zijn gevonden, kunnen deze worden gebruikt om mogelijke knikformules te formuleren. Deze knikformules kunnen met behulp van SCIA Engineer (een Eindige Elementenmethode) in combinatie met Design Forms Builder (programma waarmee zelf dingen geprogrammeerd kunnen worden in SCIA Engineer) aanschouwelijk worden gemaakt in de vorm van contourplots. Op deze wijze kunnen de knikformules worden gecontroleerd. Geven de knikformules een realistisch beeld? Is de initiatie van knik in de constructie hetzelfde zoals deze is berekend met de eindige elementenmethode in SCIA Engineer?

1 Theorie

In dit hoofdstuk staat de benodigde theorie centraal. Zo zal eerst het begrip schaalconstructies uiteengezet worden. Daarna volgt een korte introductie over tensoren. Tenslotte zullen invarianten besproken worden; zowel qua wiskunde alsook het verband met de tensoren.

1.1 Schaalconstructies

Schaalconstructies bestaan uit gekromde elementen, zowel in één richting als in twee richtingen, waarvan de dikte relatief klein is ten opzichte van de andere dimensies. Bij het classificeren van schaalconstructies kan onderscheid worden gemaakt naar het type materiaal waarvan het is gemaakt, denk hierbij aan (gewapend) beton, staal, hout. Elke materiaalsoort heeft andere karakteristieke eigenschappen. Ook kan een schaalconstructie geclassificeerd worden naar de schaaldikte. Er kan grofweg worden gezegd dat voor dunne schaalconstructies de verhouding tussen de straal (a) en de schaaldikte (t) minimaal 30 is (met een maximum van 4000, vanaf 4000 wordt het beschouwd als een membraan), voor dikke schaalconstructies is deze verhouding maximaal 30 (Hoogenboom, CIE4143 Shell Analysis, Theory and Application). In het algemeen is een schaalconstructie qua materiaalgebruik erg economisch.

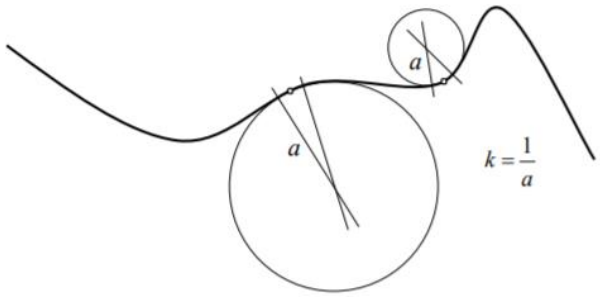
Schaalconstructies bevatten krommingen (voor een algemene schaalconstructie zijn er twee krommingen). Deze krommingen kunnen als volgt gedefinieerd worden. Waarbij z loodrecht op de schaal staat in een lokaal coördinaten systeem.

$$k_{xx} = \frac{\partial^2 z}{\partial x^2} = \frac{1}{a_{xx}}$$

$$k_{yy} = \frac{\partial^2 z}{\partial y^2} = \frac{1}{a_{yy}}$$

De krommingen kunnen ook gerelateerd worden aan de straal van de cirkel (a) in het bijbehorende punt van de schaalconstructie. Dit is in *figuur 1.1* weergegeven. Daarnaast wordt het torderen van de schaal beschreven met de volgende vergelijking.

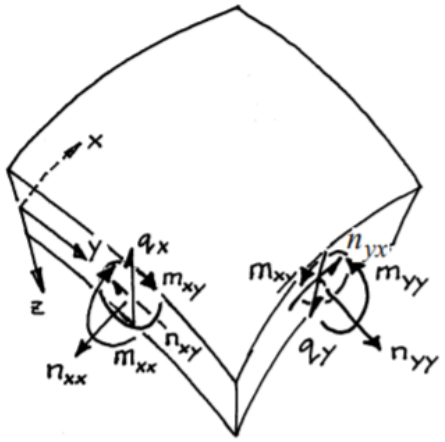
$$k_{xy} = \frac{\partial^2 z}{\partial x \partial y}$$



Figuur 1.1 Krommingen (Hoogenboom, CIE4143 Shell Analysis, Theory and Application)

Als we een deel van de schaalconstructie beschouwen (*figuur 1.2*), is te zien dat er verschillende krachten werken op dit element. Waaronder momenten, schuifkrachten en ook membraankrachten die werken in het vlak van het schaalelement. Met name deze membraankrachten zijn belangrijk voor de beschouwingen die volgen over het uitknikken van schaalconstructies. De membraankracht n_{xx} werkt in de x-richting en de normaal van het oppervlak is ook in de x-richting. Hetzelfde geldt voor de n_{yy}. Daarnaast zijn er de n_{xy} en de n_{yx}, waarbij 1e index staat voor de richting van het oppervlak en de 2e index voor de richting van de kracht. De membraankracht is een soort gelijkmatig verdeelde belasting. Het is te bepalen via de spanningen in het vlak (σ) en de bijbehorende dikte (t).

$$n = \sigma \cdot t$$



Figuur 1.2 Krachten op schaalelement (Hoogenboom, CIE4143 Shell Analysis, Theory and Application)

Het uitknikken van schaalconstructies (zie ook *figuur 1.3* voor knikvormen van een cilinder) hangt af de membraankrachten en krommingen in de constructie. In dit rapport wordt getracht (een deel van) deze formule te vinden. Het precieze verband tussen de membraankrachten en krommingen is nog niet bekend. Wel kan alvast de globale vorm van belastingfactor λ worden gemaakt. Deze belastingfactor geeft aan vanaf wanneer er knik optreedt. Als voorbeeld de formule hieronder. Waarbij E de elasticiteitsmodulus is van het materiaal en t de dikte van de schaal. k_{ij} is een element van de krommingstensor en n_{ij} is een element van de membraankrachttensor.

$$\lambda = 0.1 \cdot \mathrm{E} \cdot t^2 \cdot \frac{k_{xx} + k_{yy} + k_{xy}}{n_{xx} + n_{yy} + n_{xy}}$$

De dimensies van de afzonderlijke elementen:

$$\lambda = [-]$$

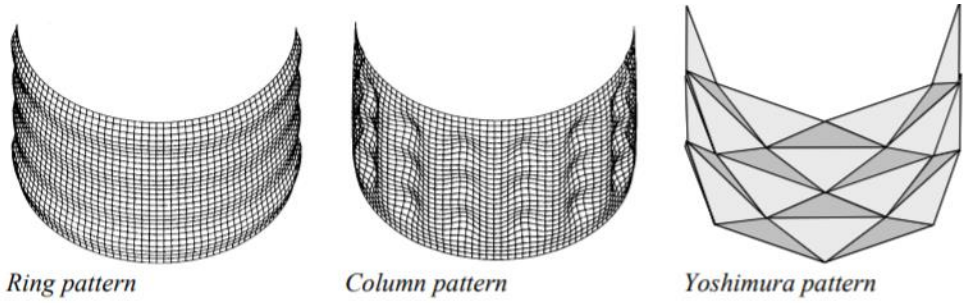
$$E = \left[\frac{N}{mm^2}\right]$$

$$t = [mm]$$

$$k_{ij} = \left[\frac{1}{mm}\right]$$

$$n_{ij} = \left[\frac{N}{mm}\right]$$

In dit rapport staat met name het laatste gedeelte uit de vergelijking van λ centraal. Welke combinatie van elementen van zowel de krommingstensor als de membraankrachttensor resulteren in de juiste dimensies voor λ [–] en zijn onafhankelijk van het assenstelsel (zie ook de *inleiding*).



Figuur 1.3 Knikvormen cilinder (Hoogenboom, CIE4143 Shell Analysis, Theory and Application)

Ten slotte moet worden opgemerkt dat schaalconstructies erg gevoelig zijn voor imperfecties. In de praktijk zal namelijk nooit een perfecte schaal gerealiseerd kunnen worden. Er zullen altijd kleine afwijkingen zijn, in de orde van millimeters, die ook het knikgedrag van de schaal zullen beïnvloeden. Deze imperfecties kunnen resulteren in een zes keer kleinere draagkracht dan tijdens perfecte omstandigheden.

1.2 Tensor

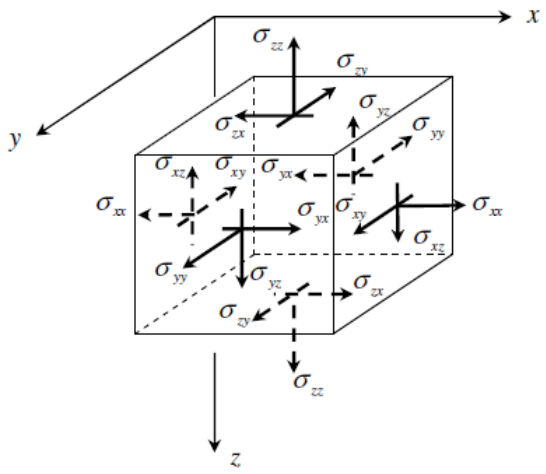
Tensoren zijn een reeks van getallen of functies, die transformeren volgens bepaalde regels bij verandering van het assenstelsel (Physlink, sd). Daarbij wordt onderscheid gemaakt naar de orde van de tensor. In theorie is elke orde mogelijk, maar in de praktijk zullen met name de 0^e, 1^e, 2^e en 3^e orde veel voorkomen.

Een nulde-orde tensor is een scalair. Zo'n scalair bevat alleen informatie over de grootte. Dit in tegenstelling tot de eerste-orde tensor, ook wel een vector genoemd. Een vector heeft naast de grootte, ook de richting als component in zich. Daarnaast kunnen op vectoren transformatie-regels worden toegepast die betrekking hebben op de rotatie van het assenstelsel. Een tweede-orde tensor is een matrix die een verband legt – lineaire relatie – tussen twee eerste-orde tensoren (vectoren). Ook voor tweede-orde tensoren zijn transformatieregels ten opzichte van het assenstelsel bekend, deze regels volgen uit de transformatieregels voor eerste-orde tensoren. Daarnaast zijn er derde-orde tensoren,

deze bestaan uit kubusmatrices opgebouwd in de driedimensionale ruimte. Hogere orde tensoren zijn in theorie ook mogelijk, in dit rapport ligt de focus echter op tweede-orde tensoren.

Een voorbeeld van een tweede-orde tensor is de spanningstensor σ_{ij} in 3D (*figuur 1.4*) met onderstaande spanningsmatrix.

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}$$



Figuur 1.4 Spanningen in drie dimensies

1.3 Invariant

Zoals hierboven is beschreven kunnen tensoren ten opzichte van het assenstelsel worden getransformeerd met transformatieformules. In het algemeen zullen de elementen van de tensor veranderen na de transformatie van het assenstelsel. Indien deze elementen van een tensor na een willekeurige transformatie niet veranderen is er sprake van een invariant. In dat geval is een combinatie van de elementen van de tensor onafhankelijk van de oriëntatie van het assenstelsel.

Een eerste-orde tensor (vector) heeft zowel een grootte als een richting. Deze richting hangt af van het assenstelsel en zal na rotatie van het assenstelsel veranderen, de richting is dus geen invariant. De grootte van de vector blijft echter gelijk na rotatie van het assenstelsel en is dus wel een invariant.

Daarnaast is er een verband te leggen tussen invarianten van een tensor en de bijbehorende eigenwaarden. Om het eigenwaarde probleem op te lossen moet een schaalfactor λ en eenheidsmatrix I worden ingevoerd.

$$(K - \lambda \mathrm{I}) = \begin{bmatrix} k_{xx} - \lambda & k_{xy} \\ k_{yx} & k_{yy} - \lambda \end{bmatrix}$$

Hierbij is er alleen sprake van een niet-triviale oplossing als de determinant van het stelsel nul is.

$$Det = (k_{xx} - \lambda)(k_{yy} - \lambda) - k_{xy}k_{yx} = 0$$

Hieruit volgt het karakteristiek polynoom. Deze bevat ook informatie over de invarianten van het stelsel. De niet-diagonaalelementen k_{xy} en k_{yx} zijn aan elkaar gelijk.

$$\lambda^2 - \left(k_{xx} + k_{yy}\right)\lambda + \left(k_{xx}k_{yy} - {k_{xy}}^2\right)$$

De constanten van het polynoom zijn de invarianten van het stelsel. In dit geval zijn $\left(k_{xx} + k_{yy}\right)$ en $\left(k_{xx}k_{yy} - {k_{xy}}^2\right)$ de invarianten. Een combinatie van elementen van de tweede orde tensor, zoals bijvoorbeeld $\left(k_{xx} + k_{yy}\right)$, zal onder rotatie van het assenstelsel niet veranderen.

In *Bijlage G* staat de oplossing van de vermenigvuldiging van twee karakteristieke polynomen van twee tensoren.

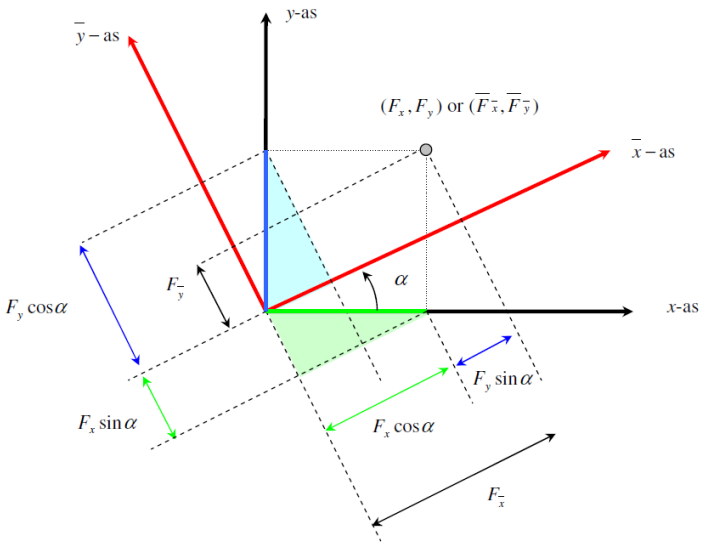
2

Inventarisatie invarianten

Dit hoofdstuk begint met een overzicht van alle transformatieformules die nodig zijn om een draaiing van het assenstelsel te beschrijven. Deze formules zullen ook in volgende hoofdstukken gebruikt worden als basis van het programmeren. Vervolgens zal een overzicht gegeven worden van alle al bekende invarianten, zoals deze in de literatuur zijn beschreven. Ook onderzoek naar umbilics en de bijbehorende invarianten zullen in beschouwing genomen worden.

2.1 Transformatie

Om invarianten te kunnen vinden, zal er eerst naar de onderliggende wiskunde gekeken moeten worden. Hoe veranderen alle elementen na rotatie van het assenstelsel. In *figuur 2.1* wordt het assenstelsel over een hoek α gedraaid. Voor de rotatie werd het punt beschreven door de x en y, na rotatie gaan deze over op $\bar{x}$ en $\bar{y}$.



Figuur 2.1 Rotatie assenstelsel (Hartsuijker & Welleman, 2018)

Vanuit de bovenstaande figuur kan de rotatiematrix R worden bepaald. Deze matrix beschrijft hoe alle elementen transformeren na de rotatie van het assenstelsel. De rotatiematrix staat hieronder genoteerd.

$$R = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

De nieuwe coördinaten – na rotatie van het assenstelsel – kunnen eenvoudig worden gevonden door de rotatiematrix R te vermenigvuldigen met de vector van de oorspronkelijke coördinaten.

$$\bar{F} = R \cdot F$$

Om een tweede orde tensor (zoals de membraankrachttensor N en de krommingstensor K) te transformeren, moet deze vermenigvuldigd worden met de rotatiematrix R en de inverse van de rotatiematrix R^{-1}. De rotatiematrix heeft als speciale eigenschap dat de inverse gelijk is aan de getransponeerde matrix R^{T}. De transformatie van de krommingstensor K gaat dan als volgt:

$$\bar{K} = R \cdot K \cdot R^T$$

$$\bar{K} = \begin{bmatrix} k_{\bar{x}\bar{x}} & k_{\bar{x}\bar{y}} \\ k_{\bar{y}\bar{x}} & k_{\bar{y}\bar{y}} \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} k_{xx} & k_{xy} \\ k_{yx} & k_{yy} \end{bmatrix} \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

Deze matrixvermenigvuldiging is hieronder uitgeschreven.

$$k_{\bar{x}\bar{x}} = k_{xx}cos^2(\alpha) + k_{xy}\sin(\alpha)\cos(\alpha) + k_{yx}\sin(\alpha) + k_{yy}sin^2(\alpha)$$

$$k_{\bar{x}\bar{y}} = -k_{xx}\sin(\alpha)\cos(\alpha) + k_{xy}cos^2(\alpha) - k_{yx}sin^2(\alpha) + k_{yy}\sin(\alpha)\cos(\alpha)$$

$$k_{\bar{y}\bar{x}} = -k_{xx}\sin(\alpha)\cos(\alpha) - k_{xy}sin^2(\alpha) + k_{yx}cos^2(\alpha) + k_{yy}\sin(\alpha)\cos(\alpha)$$

$$k_{\bar{y}\bar{y}} = k_{xx}sin^2(\alpha) - k_{xy}\sin(\alpha)\cos(\alpha) - k_{yx}\sin(\alpha) + k_{yy}cos^2(\alpha)$$

Met behulp van onderstaande dubbele-hoekformules kunnen de transformatieregels nog wat worden vereenvoudigd.

$$2cos^2(\alpha) = 1 + \cos(2\alpha)$$

$$2sin^2(\alpha) = 1 - \cos(2\alpha)$$

$$2\sin(\alpha)\cos(\alpha) = \sin(2\alpha)$$

De uiteindelijke transformatieregels voor de krommingstensor K staan hieronder. De niet-diagonaal elementen k_{xy} en k_{yx} zijn aan elkaar gelijk en daarom zal dus alleen k_{xy} worden bepaald.

$$k_{\bar{x}\bar{x}} = \frac{1}{2}\left(k_{xx} + k_{yy}\right) + \frac{1}{2}\left(k_{xx} - k_{yy}\right)\cos(2\alpha) + k_{xy}\sin(2\alpha)$$

$$k_{\bar{y}\bar{y}} = \frac{1}{2}\left(k_{xx} + k_{yy}\right) - \frac{1}{2}\left(k_{xx} - k_{yy}\right)\cos(2\alpha) - k_{xy}\sin(2\alpha)$$

$$k_{\bar{x}\bar{y}} = -\frac{1}{2}\left(k_{xx} - k_{yy}\right)\sin(2\alpha) + k_{xy}\cos(2\alpha)$$

Voor de transformaties van de membraankrachttensor N kunnen dezelfde regels worden toegepast. Met behulp van deze transformatieregels kan relatief eenvoudig worden gecontroleerd of een bepaalde combinatie een invariant is. Als na het toepassen van deze transformatieregels alles onafhankelijk is van de oriëntatie van het assenstelsel (hoek α), ofwel α valt uit alle vergelijkingen, dan is het een invariant.

2.2 Bekende invarianten

2.2.1 Gecombineerde tensoren

In *§1.3* zijn de invarianten van een enkele tensor omschreven, namelijk $(k_{xx} + k_{yy})$ en $(k_{xx}k_{yy} - {k_{xy}}^2)$. Met deze bekende invarianten kunnen nieuwe invarianten worden gemaakt, aangezien elke willekeurige combinatie van invarianten ook onafhankelijk is van het assenstelsel. Zo kan op eenvoudige wijze een invariant voor een combinatie van twee tensoren worden gevonden. Namelijk door de eerste invariant te gebruiken en van beide tensoren te vermenigvuldigen.

$$(n_{xx} + n_{yy})(k_{xx} + k_{yy})$$

De invarianten van gecombineerde tensoren zullen worden genummerd met I_i. Een al bekende invariant is I_1. Hierbij worden ook de niet-diagonaal elementen meegenomen. Dit in tegenstelling tot bovenstaande invariant waarbij de kruistermen niet terugkomen.

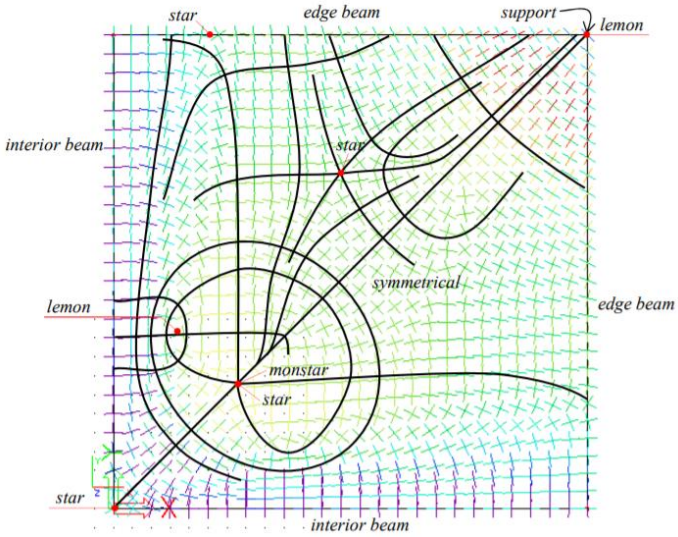
$$I_1 = n_{xy}(k_{xx} - k_{yy}) - k_{xy}(n_{xx} - n_{yy})$$

Daarnaast heeft Marijn Drillenburg in mei 2017 tijdens onderzoek naar dit onderwerp de volgende invariant I_2 gevonden.

$$I_2 = n_{xx}k_{yy} - 2n_{xy}k_{xy} + n_{yy}k_{xx}$$

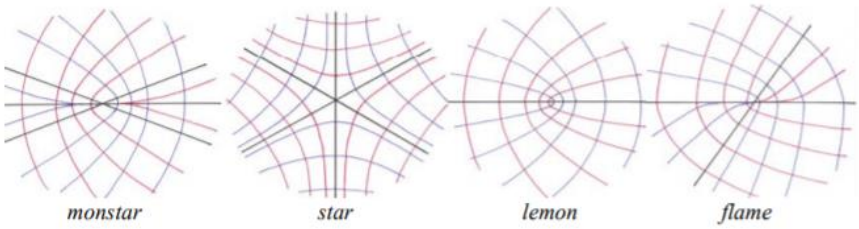
2.2.2 Umbilics

In de literatuur is ook veel onderzoek gedaan naar *umbilics*, dit zijn specifieke punten in een tensorveld waarbij de hoofdspanningen aan elkaar gelijk zijn. Deze punten worden ook wel isotropische punten genoemd, aangezien er geen schuifspanningen zijn. Een voorbeeld van zo'n tensorveld met umbilics is te zien in *figuur 2.2*. Ook bij umbilics is het zinvol om invarianten te bepalen. Zo zijn er een aantal invarianten bekend die in deze paragraaf worden besproken.



Figuur 2.2 Tensorveld met umbilics (Hoogenboom, CIE4143 Shell Analysis, Theory and Application)

Zoals in bovenstaande figuur is te zien, zijn er verschillende soorten umbilics (*figuur 2.3*). De naamgeving is gedaan aan de hand van de vorm van de umbilic. Zo is *monstar* een combinatie tussen *star* en *lemon*.



Figuur 2.3 Soorten umbilics (Hoogenboom, CIE4143 Shell Analysis, Theory and Application)

Het onderzoek naar umbilics en de verschillende soorten hiervan, maakt gebruik van de richtingen rondom de umbilic. Deze richtingen, ook wel gradiënten (a_i) genoemd, komen later terug in de invarianten van een umbilic.

$$a_1 = \frac{\partial n_{xx}}{\partial x} \qquad a_2 = \frac{\partial n_{xx}}{\partial y}$$

$$a_3 = \frac{\partial n_{yy}}{\partial x} \qquad a_4 = \frac{\partial n_{yy}}{\partial y}$$

$$a_5 = \frac{\partial n_{xy}}{\partial x} \qquad a_6 = \frac{\partial n_{xy}}{\partial y}$$

Specifieke combinaties van gradiënten kunnen onafhankelijk zijn van het assenstelsel. Deze bekende invarianten van umbilics staan hieronder genoemd (U_i). U_1, U_5 en U_6 zijn gevonden door A. Thorndike et al in 1978. U_2, U_3 en U_4 zijn gevonden door Wouter van Straalen in 2013.

$$U_1 = (a_1 - a_3)a_6 - (a_2 - a_4)a_5$$

$$U_2 = (a_1 + a_3)^2 + (a_2 + a_4)^2$$

$$U_3 = a_1 a_3 - {a_5}^2 + a_2 a_4 - {a_6}^2$$

$$U_4 = (a_3 - a_6)^2 + (a_2 - a_5)^2$$

$$U_5 = 4 \cdot (a_1 a_3 - {a_5}^2) \cdot (\, a_2 a_4 - {a_6}^2) - (a_1 a_4 + a_2 a_3 - 2 \cdot a_5 a_6)^2$$

$$U_6 = 4 \cdot \left({d_2}^2 - 3 \cdot a_6 d_1\right) \cdot \left({d_1}^2 + 3 \cdot a_5 d_2\right) - (d_1 d_2 + 9 \cdot a_5 a_6)^2$$

Met: $d_1 = a_1 - a_3 - a_6$ en $d_2 = a_2 - a_4 + a_5$

Invariant U_1 is ook te schrijven als de al bekende invariant $I_1 = n_{xy}\left(k_{xx} - k_{yy}\right) - k_{xy}(n_{xx} - n_{yy})$. Hiervoor moeten de gradiënten (a_i) omgeschreven worden:

$$a_1 = n_{xx} \qquad a_2 = k_{xx}$$

$$a_3 = n_{yy} \qquad a_4 = k_{yy}$$

$$a_5 = n_{xy} \qquad a_6 = k_{xy}$$

Ook invariant U_2 kan worden omgeschreven: $(n_{xx} + n_{yy})^2 + (k_{xx} + k_{yy})^2$. Deze invariant is opgebouwd uit meerdere invarianten, de bouwsteen is namelijk de invariant: $\left(k_{xx} + k_{yy}\right)$ en

$(n_{xx} + n_{yy})$ van een enkelvoudige tensor. In principe is U_2 dan niet te beschouwen als een nieuwe invariant.

Hetzelfde geldt voor U_3. Als deze wordt omgeschreven: $n_{xx}n_{yy} - {n_{xy}}^2 + k_{xx}k_{yy} - {k_{xy}}^2$, blijkt dat de basis van deze invariant is opgebouwd uit een invariant van een enkelvoudige tensor: $n_{xx}n_{yy} - {n_{xy}}^2$ en $k_{xx}k_{yy} - {k_{xy}}^2$. Het optellen van twee invarianten levert per definitie een nieuwe invariant op.

U_5 is ook opgebouwd uit bekende invarianten: $4 \cdot (n_{xx}n_{yy} - {n_{xy}}^2) \cdot (k_{xx}k_{yy} - {k_{xy}}^2) - (n_{xx}k_{yy} + k_{xx}n_{yy} - 2 \cdot n_{xy}k_{xy})^2$.

Een overzicht van de elementaire invariant bouwstenen staat hieronder in *tabel 2.1*. Met behulp van onderstaande invarianten kunnen combinaties worden gemaakt die ook invariant zijn.

Nummer	**Soort**	**Invariant**
I	Enkelvoudige tensor	$(k_{xx} + k_{yy})$ en $(n_{xx} + n_{yy})$
II	Enkelvoudige tensor	$k_{xx}k_{yy} - {k_{xy}}^2$ en $n_{xx}n_{yy} - {n_{xy}}^2$
III	Gecombineerde tensor	$n_{xy}(k_{xx} - k_{yy}) - k_{xy}(n_{xx} - n_{yy})$
IV	Gecombineerde tensor	$n_{xx}k_{yy} - 2n_{xy}k_{xy} + n_{yy}k_{xx}$

Tabel 2.1 Bouwstenen invarianten

3

Programmeren: Analytische methode

De eerste van de twee programmeermethodes zal in dit hoofdstuk uitgelegd worden. Om te beginnen volgt een uitleg over de methode zelf, daarna wordt deze methode verduidelijkt door middel van een voorbeeld. Tenslotte volgt een mogelijke oplossing die op deze manier is gevonden.

3.1 Methode

Om heel veel verschillende combinaties te toetsen, is het handig om gebruik te maken van de computer en deze op systematische wijze alle mogelijke combinaties te laten formuleren en naderhand te controleren of het een invariant is. De transformatieregels uit het vorige hoofdstuk dienen als basis voor het programmeren. Het gebruikte Python-script staat in *Bijlage A*.

In het Python-script wordt als input een mogelijke combinatie van elementen van tensoren aangegeven. Het script zal dan automatisch alle mogelijkheden uitproberen, als blijkt dat een bepaalde combinatie invariant is zal deze geprint worden op het scherm. Voor zeer lange combinaties kan de rekentijd erg groot worden. Alle mogelijkheden proberen is dus niet reëel, er zal gericht gezocht moeten worden. In totaal zijn er zo'n 200.000 combinaties getoetst, dit leverde met name al bekende oplossingen op (of combinaties van al bekende oplossingen). Alle onderzochte globale vormen staan opgesomd in *Bijlage D*.

Bij deze analytische methode wordt gebruik gemaakt van de *simplify* functie in Python. Deze functie probeert een bepaalde formule in de meest eenvoudige vorm terug te brengen. Als blijkt dat na deze vereenvoudiging de term *alpha* (rotatie van het assenstelsel α) wegvalt, dan is de specifieke combinatie een invariant. Echter is gebleken dat Python (en ook Maple) niet altijd de vereenvoudigingen goed doen. Af en toe blijft er ergens een *alpha* achter, die met verder vereenvoudigen wel weg zou moeten vallen.

Om de werkwijze te verduidelijken staat hieronder een simpel voorbeeld uitgewerkt.

Stap 1:

Om een groep combinaties te toetsen, moet een globale vorm als input worden gegeven.

```
A = ['+','-']
B = ['(',')']
C = ['*','+','-']
S = ['SXX', 'SXY', 'SYY']
K = ['KXX', 'KXY', 'KYY']
N = ['1*','2*','3*']
P = ['**2']
```

Voorbeeld:

```
combinations = [[S,'*',S,'-',S,P]]
```

Deze simpele globale vorm levert 27 combinaties op. Het aantal combinaties loopt snel op naarmate de globale vorm complexer is.

```
SXX*SXX-SXX**2
SXX*SXX-SXY**2
SXX*SXX-SYY**2
SXX*SXY-SXX**2
SXX*SXY-SXY**2
SXX*SXY-SYY**2
SXX*SYY-SXX**2
SXX*SYY-SXY**2
SXX*SYY-SYY**2
SXY*SXX-SXX**2
SXY*SXX-SXY**2
SXY*SXX-SYY**2
SXY*SXY-SXX**2
SXY*SXY-SXY**2
SXY*SXY-SYY**2
SXY*SYY-SXX**2
SXY*SYY-SXY**2
SXY*SYY-SYY**2
SYY*SXX-SXX**2
SYY*SXX-SXY**2
SYY*SXX-SYY**2
SYY*SXY-SXX**2
SYY*SXY-SXY**2
SYY*SXY-SYY**2
SYY*SYY-SXX**2
SYY*SYY-SXY**2
SYY*SYY-SYY**2
```

Stap 2:

Alle combinaties worden vervolgens getoetst of het een invariant is. Hiervoor wordt de functie *test_invariant* gebruikt. De combinatie wordt vereenvoudigd met behulp van de *simplify* functie. Als de hoek *alpha* wegvalt, is de combinatie een invariant.

Stap 3:

Nadat Python alle combinaties heeft doorgerekend, verschijnt de output (met mogelijke nieuwe invarianten) op het scherm.

Vervolg voorbeeld:

```
##### Total permutations: 27 ; in combination k: 1 / 1
INVARIANT: SXX*SYY-SXY**2
INVARIANT: SYY*SXX-SXY**2
Done! Amount of invariants found: 2
```

In dit simpele voorbeeld is de al bekende invariant $S_{xx}S_{yy} - {S_{xy}}^2$ gevonden. Zoals te zien is weet het Python script niet dat `SXX*SYY-SXY**2` en `SYY*SXX-SXY**2` eigenlijk hetzelfde is. De gegenereerde output moet dus alsnog goed bestudeerd worden.

3.2 Mogelijke oplossing

Met deze methode zijn honderden invarianten gevonden, deze waren echter allemaal eenvoudig te herleiden tot bestaande invarianten. Daarnaast is er een andere invariant gevonden (N_1), die aanvankelijk nieuw bleek te zijn. Echter is ook deze oplossing te herleiden naar al bekende invarianten.

$$N_1 = n_{xx}k_{xx} + 2n_{xy}k_{xy} + n_{yy}k_{yy}$$

Deze nieuwe invariant lijkt heel erg op de al gevonden invariant IV uit *hoofdstuk 2*.

$$n_{xx}k_{yy} - 2n_{xy}k_{xy} + n_{yy}k_{xx}$$

Door het optellen van deze twee invarianten valt de term $n_{xy}k_{xy}$ weg.

$$n_{xx}k_{xx} + n_{xx}k_{yy} + n_{yy}k_{xx} + n_{yy}k_{yy}$$

Dit is weer te herleiden tot de invariant:

$$(n_{xx} + n_{yy})(k_{xx} + k_{yy})$$

Hieruit kan worden opgemaakt dat N_1 niet een compleet nieuwe invariant is. Het is namelijk mogelijk om deze invariant te relateren aan al bekende invarianten. Omdat deze invariant samen met de al gevonden invariant IV een soort koppel vormen van twee simpele invarianten is deze nieuwe invariant het vermelden waard. Ook voor het formuleren van de knikformules kan deze invariant N_1 nuttig zijn.

De bovenstaande invariant N_1 kan ook worden geschreven als de dubbele contractie ($n{:}\,k$) van de twee tensoren n_{ij} en k_{ij}. Elke component van de ene tensor wordt vermenigvuldigd met de corresponderende component van de andere tensor.

$$n{:}\,k = n_{xx}k_{xx} + n_{xy}k_{xy} + n_{xy}k_{xy} + n_{yy}k_{yy} = n_{xx}k_{xx} + 2n_{xy}k_{xy} + n_{yy}k_{yy} = N_1$$

Deze dubbele contractie van twee tensoren vertoont veel gelijkenis met het dot product van twee vectoren ($a \cdot b$). Het dot product van twee vectoren is ook een invariant. Dit kan op een eenvoudige wijze worden aangetoond als $a = b$ en we weten dat de lengte van een vector invariant is:

$$a \cdot b = {a_x}^2 + {a_x}^2 + {a_x}^2 = lengte^2$$

4

Programmeren: Numerieke methode

In dit hoofdstuk staat de tweede programmeermethode centraal. Ook hier zal eerst de methode uitgelegd worden. Daarna volgen de oplossingen voor het algemene geval. Aan het einde van dit hoofdstuk zal de vraag gesteld worden of er überhaupt nog andere invarianten mogelijk zijn. Ook staan invarianten met quotiënten centraal in het laatste gedeelte van dit hoofdstuk.

4.1 Methode

Een andere methode om op een systematische wijze invarianten te vinden, is door gebruik te maken van de grafische numerieke methode. In plaats van de functie te vereenvoudigen in de simpelste vorm (zoals in *Hoofdstuk 3* is beschreven), kan ook gebruik worden gemaakt van het feit dat het plotten tegen de hoek van het assenstelsel, een constante waarde moet opleveren. Een invariant hangt immers niet af van de hoek, dit resulteert automatisch in een constante rechte lijn (zie ook *Bijlage C* voor een grafisch voorbeeld).

Om op een snelle wijze te toetsen of een functie constant is, wordt de waarde van de functie op diverse locaties uitgerekend. Als al deze waarden op de diverse locaties aan elkaar gelijk zijn (binnen een gestelde bandbreedte) is het een constante functie en dus een invariant.

Tijdens het programmeren valt meteen op dat deze methode veel sneller is. Het gebruikte Python script staat in *Bijlage B.* Om het berekenen sneller en makkelijker te maken, worden er in eerste instantie willekeurige waarden voor $n_{xx},\ n_{xy},\ n_{yy},\ k_{xx},\ k_{xy}\ en\ k_{yy}$ ingevoerd. Om de methode nog sneller te maken worden deze willekeurige waarden handmatig ingevoerd. Ze zouden ook via de random functie kunnen worden ingevoerd, maar dat vertraagd het proces (zeker als voor alle combinaties opnieuw zes waarden random gekozen moeten worden). Als het Python script een oplossing geeft, dient deze echter voor een tweede keer gecontroleerd te worden (om volledig zeker te zijn dat het een algemene oplossing is en in elke situatie geldig is). Een invariant is een oplossing die voor elke willekeurige waarden geldt. De tweede keer moeten andere waarden voor $n_{xx},\ n_{xy},\ n_{yy},\ k_{xx},\ k_{xy}\ en\ k_{yy}$ worden gekozen. Een andere optie is om de gevonden oplossingen voor de tweede keer te controleren via de analytische

methode (*Hoofdstuk 3*). Als blijkt dat ook via de *simplify* functie in python de term alpha er volledig uitvalt is de oplossing een echte invariant die altijd geldt.

4.2 Algemeen geval

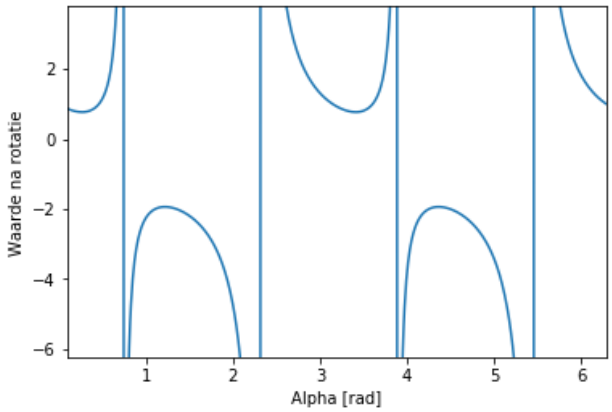
Voorlopig zijn er zo'n zes à zeven miljoen combinaties getoetst, tot op heden is er echter nog geen nieuwe invariant gevonden die in elke situatie geldt. Wel hebben de zes à zeven miljoen combinaties geleid tot ongeveer 2700 invarianten, na controle bleken deze allemaal opgebouwd te zijn uit al bekende invarianten. De onderzochte globale vormen staan in *Bijlage E*.

4.3 Quotiënten

Ook is er apart gezocht naar invariant met quotiënten. Dit is interessant omdat alle bekende invarianten zijn opgebouwd uit producten, dan rest natuurlijk de vraag of er ook invarianten mogelijk zijn met quotiënten. Om dit probleem op te lossen wordt gebruik gemaakt van de numerieke methode. Met name omdat deze methode vele malen sneller is dan de analytische methode.

Bij quotiënten moet wel worden opgepast met de input in het systeem, anders kunnen fouten ontstaan zoals delen door nul. Daarnaast is het gedrag van quotiënten veel grilliger dan bij de producten. Zo kunnen er best snel verticale asymptoten ontstaan (*figuur 4.1*). Om te controleren of een combinatie een invariant is, wordt gebruik gemaakt van de methode zoals deze in *§4.1* is uitgelegd.

De onderzochte vormen van quotiënten staan in *Bijlage F*. In totaal zijn er zo'n zes miljoen combinaties getoetst (specifiek gericht op quotiënten), deze zoekactie heeft uiteindelijk geen nieuwe invarianten opgeleverd.



Figuur 4.1 Asymptoten ontstaan makkelijk bij quotiënten

4.4 Bestaan er nog andere invarianten?

Tijdens dit onderzoek zijn er in totaal zo'n twaalf miljoen verschillende combinaties getoetst. Dit leverde veel invarianten op, echter bleek in de meeste gevallen al vrij snel dat deze waren opgebouwd uit bekende invarianten. Wel leverde het onderzoek aanvankelijk één nieuwe invariant op die voorheen nog niet bekend was. Later bleek dat ook deze is opgebouwd uit bekende invarianten. Al met al wordt het vermoeden groter dat er misschien helemaal geen andere invarianten meer zijn. Misschien zijn alle

bouwstenen voor de invarianten al ontdekt. Een wiskundig bewijs om dit aan te tonen ontbreekt momenteel.

Wat wel opvalt, is dat er voor enkelvoudige en gecombineerde tensoren in totaal zes invarianten gevonden zijn (zie ook *tabel 2.1*). Daarnaast zijn er ook zes elementen die de twee tensoren beschrijven; $n_{xx},\ n_{xy},\ n_{yy},\ k_{xx},\ k_{xy}\ en\ k_{yy}$. Is dit toeval, of is het daarom logisch dat er in totaal slechts zes invarianten gevonden zijn. Deze hypothese is geformuleerd in de onderstaande conjecture.

Conjecture:

Of all combinations that can be made of the elements of two tensors only six combinations are independent invariants.

Uit praktische overwegingen is besloten om na twaalf miljoen combinaties te stoppen met het zoeken naar andere invarianten. Puur vanwege het feit dat het zoeken relatief veel tijd in beslag neemt. Om echt te kunnen uitsluiten dat er niet nog meer invarianten zijn, is het aan te bevelen om verder te zoeken of een wiskundig bewijs te formuleren over het bestaan van invarianten.

Ook is het goed om op te merken dat er geen invarianten zijn gevonden opgebouwd uit quotiënten. Liggen hier wiskundige of misschien zelfs fysische principes ten grondslag? Ook op deze vraag is nog geen duidelijk antwoord te geven. Het onderzoek wat wereldwijd gedaan is naar invarianten beschrijft nergens de mogelijke oorzaak van dit fenomeen.

5

Knikformule opstellen

Verschillende knikformules zullen worden geformuleerd. Deze formules zijn opgebouwd uit invarianten. Voor het opstellen van de knikformules kunnen de volgende invarianten worden gebruikt:

$$n_{xx} + n_{yy}$$

$$k_{xx} + k_{yy}$$

$$n_{xx}n_{yy} - {n_{xy}}^2$$

$$k_{xx}k_{yy} - {k_{xy}}^2$$

$$n_{xy}(k_{xx} - k_{yy}) - k_{xy}(n_{xx} - n_{yy})$$

$$n_{xx}k_{yy} - 2n_{xy}k_{xy} + n_{yy}k_{xx}$$

Ook de invariant die is gevonden in *§3.2*, maar toch geen nieuwe invariant bleek te zijn, is meegenomen in het formuleren van mogelijke knikformules. Met deze invariant kunnen namelijk op eenvoudige wijze de dimensies van de knikformule kloppend gemaakt worden.

$$n_{xx}k_{xx} + 2n_{xy}k_{xy} + n_{yy}k_{yy}$$

Zoals ook is uitgelegd in *§1.1*, kan voor de knikformule een globale vorm worden gegeven. Deze belastingfactor λ geeft aan wanneer knik optreedt. De factor 0.1 kan worden aangepast om een correctie toe te passen.

$$\lambda = 0.1 \cdot \mathrm{E} \cdot t^2 \cdot (INVARIANT\ )$$

Nu volgen knikformules die mogelijk het verband beschrijven. Hierbij is veel variatie mogelijk, echter is getracht om de formules zo simpel mogelijk te houden. Ook is de factor 0.1 niet meegenomen in de knikformule, de formule kan namelijk altijd nog geschaald worden naargelang dit nodig is.

Knikformule 1

$$\lambda_1 = \mathrm{E} \cdot t^2 \cdot \frac{k_{xx} + k_{yy}}{n_{xx} + n_{yy}}$$

Knikformule 2

$$\lambda_2 = \mathrm{E} \cdot t^2 \cdot \frac{n_{xx}k_{xx} + 2n_{xy}k_{xy} + n_{yy}k_{yy}}{(n_{xx} + n_{yy})^2}$$

Knikformule 3

$$\lambda_3 = \mathrm{E} \cdot t^2 \cdot \frac{(k_{xx} + k_{yy})^2}{n_{xx}k_{xx} + 2n_{xy}k_{xy} + n_{yy}k_{yy}}$$

Knikformule 4

$$\lambda_4 = \mathrm{E} \cdot t^2 \cdot \frac{k_{xx} + k_{yy} + k_{xx}k_{yy} - {k_{xy}}^2}{n_{xx} + n_{yy} + n_{xx}n_{yy} - {n_{xy}}^2}$$

Van deze knikformules voldoet alleen de vierde formule niet volledig aan de eisen qua dimensies. Van deze formules kunnen diverse variaties worden gemaakt. Dit kan door het toevoegen van een invariant (bijvoorbeeld: $n_{xx}n_{yy} - {n_{xy}}^2$, $k_{xx}k_{yy} - {k_{xy}}^2$, $n_{xy}(k_{xx} - k_{yy}) - k_{xy}(n_{xx} - n_{yy})$ of $n_{xx}k_{yy} - 2n_{xy}k_{xy} + n_{yy}k_{xx}$), of het kwadrateren van een invariant. Om het aantal te onderzoeken knikformules niet te groot te laten worden, zullen alleen de vier bovenstaande knikformules onderzocht worden.

6

SCIA Engineer

Dit hoofdstuk begint met een korte introductie over de gebruikte software SCIA Engineer en Design Forms Builder. Vervolgens worden de berekeningen toegelicht. Tenslotte een overzicht van de gemodelleerde schaalconstructies in SCIA Engineer.

6.1 Software

Om de kwaliteit van de nieuwe knikformule te verifiëren zal gebruik worden gemaakt van SCIA Engineer. In deze software kunnen constructies in 3D worden gemodelleerd. Vervolgens kunnen diverse berekeningen (bijvoorbeeld sterkte, stabiliteit en vervorming) worden uitgevoerd met behulp van een Eindige Elementenmethode. Een voordeel van SCIA Engineer is dat schaalconstructies relatief eenvoudig kunnen worden ingevoerd.

Daarnaast wordt Design Forms Builder[1] gebruikt om eigen formules te kunnen programmeren in SCIA Engineer. Om deze twee programma's te kunnen combineren, moet Open Design Checks worden gebruikt. De schaalconstructies zullen eerst in SCIA Engineer worden gemodelleerd. Bepaalde eigenschappen kunnen door SCIA Engineer worden uitgerekend en vervolgens gebruikt worden als input voor het programmeren in Design Forms Builder.

6.2 Programmeren

6.2.1 Invoer in Design Forms Builder

Om de knikformules te kunnen programmeren in Design Forms Builder, zullen verschillende variabelen uit het SCIA Engineer model gehaald moeten worden. Al deze variabelen hangen af van de schaalconstructie. Deze variabelen kunnen daarna als input dienen in de berekeningen. Om de connectie te maken met het model en de variabelen, moeten deze gelinkt worden met behulp van ESA_ID's.

[1] Voorheen was Design Forms Builder een onderdeel van het SCIA pakket. Via SCIA konden ook eenvoudig studentenlicenties worden aangevraagd. Sinds 2018 is het onderdeel Design Forms Builder van SCIA echter verkocht aan een ander bedrijf. Om toch gebruik te kunnen maken van Design Forms Builder kan via https://designforms.net/LP/Default.aspx?LangCode=en-US een account worden aangemaakt. Na het inloggen kan in het scherm rechtsboven geklikt worden op uw account (emailadres met daarnaast een pijltje naar beneden). Vervolgens kan via *My Calculations* en dan *Design Forms Builder* het programma gedownload worden (30 dagen proefversie).

Om de knikformules te kunnen doorrekenen, zijn de volgende variabelen nodig:

- E elasticiteitsmodulus
- t dikte van de schaalconstructie
- k_{xx} kromming
- k_{yy} kromming
- k_{xy} kromming
- n_{xx} membraankracht
- n_{yy} membraankracht
- n_{xy} membraankracht

De elasticiteitsmodulus E is eenvoudig als input te krijgen door het invoeren van de volgende ESA_ID: CONCRETE.EC.Ecm / Point.Material.EC.Ecm (voor beton) of STEEL.EC.Es / Point.Material.EC.Es (voor staal).

De dikte van de schaalconstructie t kan worden ingeladen via: Point.H of CS.Geometry.H . Deze dikte heeft als dimensie $[m]$ in SCIA Engineer.

Om de andere zes variabelen te bepalen, moet wat meer werk worden gedaan. SCIA Engineer heeft geen ingebouwde functie om de krommingen van de schaalconstructie te bepalen. De krommingen zullen eerst zelf berekend moeten worden.

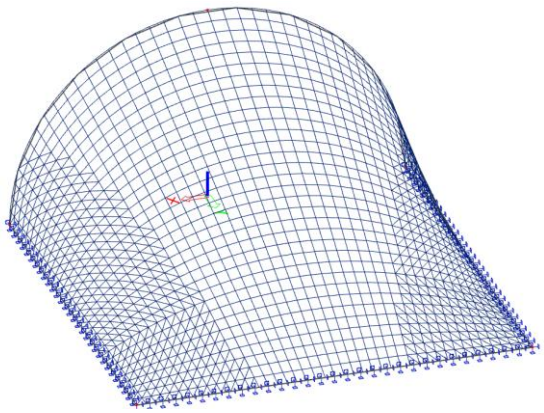
Om de membraankrachten te bepalen zullen eerst de normaalkrachten worden ingeladen uit SCIA Engineer met behulp van de volgende ESA_ID's:

N_x = InternalForces.nx

N_y = InternalForces.ny

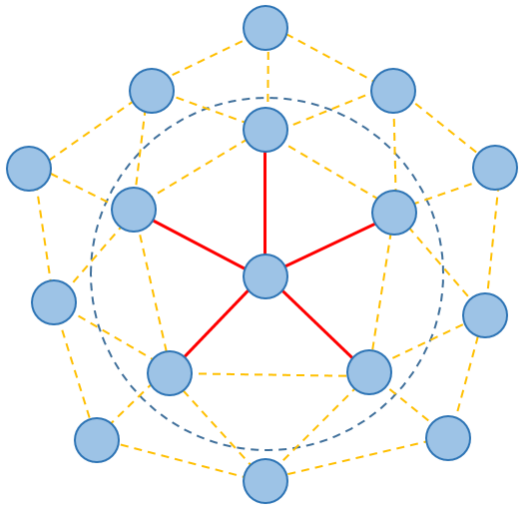
N_{xy} = InternalForces.nxy

Uit SCIA Engineer blijkt dat deze N_x, N_y en N_{xy} waarden al in de dimensies van de membraankrachten $\left[\frac{kN}{m}\right]$ staan. Het is dus niet nodig om de normaalkrachten nog eens om te rekenen naar de membraankrachten. Deze membraankrachten zijn van toepassing in de knopen van het *mesh*.



Figuur 6.1 Mesh van Stalen Conoid (SCIA Engineer)

Om de krommingen te bepalen zijn de coördinaten van het *mesh* (vereenvoudiging schaalconstructie door middel van een raster, *figuur 6.1*) nodig. Met behulp van deze coördinaten is het mogelijk om per punt in het *mesh* een benadering van het vlak te genereren. Deze vergelijking kan dan worden gebruikt om de krommingen te bepalen. Dit moet wel voor elk punt in het *mesh* gebeuren, wat niet optimaal is. Een benadering van het vlak rond een punt kan met de vergelijking: $z(x,y) = a \cdot x^2 + b \cdot x \cdot y + c \cdot y^2 + d \cdot x + e \cdot y + f$. Waarbij a t/m f onbekenden zijn die moeten worden opgelost om de algemene vergelijking te kunnen gebruiken. Door punten te gebruiken in de nabije omgeving van het punt waar het om gaat, kunnen deze zes onbekenden worden opgelost. Om alle zes onbekenden op te lossen, moeten vijf omringende punten worden beschouwd. In *figuur 6.2* is te zien dat binnen een bepaalde straal rondom het punt, naar vijf andere punten wordt gezocht. Het mesh bestaat natuurlijk uit heel veel punten, maar per punt zijn alleen de omringende vijf punten interessant. Nadat deze vijf punten zijn gevonden kunnen de x, y en z waarden worden gebruikt om de zes onbekenden op te lossen.



Figuur 6.2 Mesh met omringende punten

Nu zes punten zijn gevonden en de bijbehorende x, y en z waarden bekend zijn, kunnen deze gebruikt worden om de formule van $z(x,y) = a \cdot x^2 + b \cdot x \cdot y + c \cdot y^2 + d \cdot x + e \cdot y + f$ op te lossen. Als de zes onbekenden zijn opgelost kan de vergelijking van het vlak worden beschreven met $z(x,y)$.

De x en y waarden kunnen in een matrix worden gezet. De zes bijbehorende z waarden komen in een vector te staan.

$$M = \begin{bmatrix} {x_1}^2 & x_1 \cdot y_1 & {y_1}^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {x_6}^2 & x_6 \cdot y_6 & {y_6}^2 & x_6 & y_6 & 1 \end{bmatrix}$$

$$z = \begin{bmatrix} z_1 \\ \vdots \\ z_6 \end{bmatrix}$$

Om de onbekenden a t/m f op te lossen uit bovenstaande matrix en vector zal de matrix via gaussian eliminatie geveegd moeten worden naar echelon vorm. Dit is noodzakelijk omdat het in Design Forms Builder niet mogelijk is om matrices op te lossen. Het veegproces is door M.C. Vergeer in 2016

uitgewerkt in SCIA Design Forms. Deze methode kan ook gebruikt worden in Design Forms Builder. Deze methode maakt gebruik van een algoritme ontwikkeld door Vanden Berghe in 2007.

Na het toepassen van de hierboven genoemde methode zijn de onbekenden a t/m f opgelost. De vergelijking van het vlak in een bepaald punt $z(x, y)$ is hiermee bekend. Met behulp van deze vergelijking kunnen de krommingen k_{xx}, k_{yy} en k_{xy} worden uitgerekend. De krommingen zijn gelijk aan de tweede afgeleide van de vergelijking van het vlak, aangezien de tweede afgeleiden de buigpunten van het vlak zijn. Om de kromming k_{xx} te bepalen moet de vergelijking $z(x, y)$ twee keer gedifferentieerd worden naar x. Via hetzelfde principe: k_{yy} bepalen door twee keer te differentiëren naar y en k_{xy} bepalen door eerst te differentiëren naar x en daarna naar y. Dit geldt alleen indien het lokale assenstelsel is genomen wat overeenkomt met het assenstelsel van de membraankrachten. In dat geval zouden de termen d, e en f alle drie nul moeten zijn. Aangezien de krommingen in dit onderzoek via het globale assenstelsel zijn bepaald, kloppen de onderstaande formules niet helemaal.

$$k_{xx} = \frac{\partial z(x,y)}{\partial x^2} = 2a$$

$$k_{yy} = \frac{\partial z(x,y)}{\partial y^2} = 2c$$

$$k_{xy} = \frac{\partial z(x,y)}{\partial x \partial y} = b$$

Nu zijn alle onderdelen bekend om de vergelijking voor λ uit te rekenen. Vervolgens kunnen deze waarden voor elk punt in het *mesh* worden uitgerekend. Met behulp van de knikformules zouden in het vervolg relatief eenvoudig constateringen kunnen worden gedaan over het uitknikken van een constructie en de initiatie hiervan. Een betere knikformule is beter in staat om dit proces te beschrijven.

De check is nu afgerond en kan worden ingevoerd in SCIA Engineer. Hierbij moet gebruik worden gemaakt van de open check functie van SCIA. Ook moeten alle juiste ESA_ID's worden ingevoerd, dit zorgt namelijk voor de koppeling tussen het model in SCIA Engineer en de berekening in Design Forms Builder. Daarnaast moet de EDM-loader aan het systeem worden toegevoegd. Zorg dat de EDM-loader refereert naar de data uit het model, dit kan via: deze pc > Windows (C:) > gebruikers > *eigen naam* > ESA18.0 > Temp > Esa_Model_Data > datamap (bijvoorbeeld Member2D.1). Als er hierbij een foutmelding optreedt kan het mapje Esa_Model_Data bijvoorbeeld worden verplaatst naar het bureaublad en refereer hier vervolgens naar. Als er geen datamap (bijvoorbeeld Member2D.1) beschikbaar is, moet SCIA Engineer op een andere manier opgestart worden. Namelijk via de functie in Windows '*Uitvoeren'*. Typ hier: *"C:\Program Files (x86)\SCIA\Engineer18.0\Esa.exe" –OCFILES*. SCIA Engineer zal op deze manier alle data uit het model in de map Esa_Model_Data zetten. De Design Forms Builder code staat in *Bijlage I*.

Om de check te kunnen gebruiken in SCIA Engineer moet de check als .clc-bestand worden geëxporteerd. Vervolgens moet dit bestand in de open checks map komen van SCIA: deze pc > Windows (C:) > Gebruikers > *eigen naam* > Documenten > ESA18.0 > OpenChecks.

Nadat het bestand in de juiste map is geplaatst, kan de check in SCIA Engineer worden geladen. Open eerst het model wat onderzocht moet worden. Aan de linkerkant van het scherm staat *Integrated Design Forms.* Klik dit aan en kies vervolgens voor de *Check manager*. Via dit menu kun je de nieuwe check

inladen in het systeem. De nieuwe check staat nu onder het kopje *Other checks*. Klik op de check en pas eventueel nog wat dingen aan (bijv. welke waarden geplot moeten worden) in het eigenschappen paneel aan de rechterkant. Druk op de pijltjes bij *refresh* en de berekening wordt uitgevoerd.

6.2.2 Opmerkingen

Tijdens het gebruik van Design Forms Builder is gebleken dat het programma nog niet erg gebruiksvriendelijk is. Het invoeren van een check en daarna de juiste output krijgen in SCIA Engineer is niet erg eenvoudig. Daarnaast moet worden opgemerkt dat Design Forms Builder en de koppeling met SCIA Engineer nog niet zo lang bestaat en er dus ook nog veel bugs in zitten. Het proces van het opstellen van de berekening tot aan de gewenste output krijgen in SCIA Engineer kent als het ware een ontwerpcyclus, waarbij men stapje voor stapje naar het gewenste eindresultaat komt. Er kunnen zich diverse problemen voordoen, een overzicht van de mogelijke problemen staan hieronder.

- Zorg dat de licentie van SCIA Engineer op actief staat (dit is te zien in de *SCIA Activatie Manager*). Als SCIA Engineer niet geopend kan worden terwijl de licentie wel actief is, kunnen de instellingen worden bekeken via *FlexNet License Administrator*. Inloggen kan via *Administration* rechtsboven (eerste keer inloggen met User Name: admin, Password: admin). Controleer de status van SCIA bij *Vendor Daemon Configuration*, de status moet op *Up* staan. Controleer ook of de juiste server port is gekozen bij *Server Configuration*.
- In Design Forms Builder moeten ESA_ID's worden ingevoerd, die zorgen voor de koppeling tussen de berekening en het model in SCIA Engineer. Een mogelijke fout die kan optreden is dat het bestand niet opgeslagen kan worden. Als dat het geval is moeten alle ESA_ID's worden gecontroleerd. Er mogen namelijk geen ESA_ID's voorkomen in de *Table of variables* die niet in de berekening worden gebruikt.
- Om de check in SCIA Engineer te kunnen gebruiken moet het model zijn doorgerekend.
- Tijdens het uitvoeren van de check volgt een foutmelding dat niet alle elementen zijn gecheckt. Dit heeft te maken de hoeveelheid geheugen die noodzakelijk is en hoe het programma hiermee omgaat in het *memory management*. Het probleem geldt specifiek voor SCIA Engineer 17 en oudere versies. Het probleem is opgelost in SCIA Engineer 18 (vanaf juni 2018 verkrijgbaar).
- Indien het model erg veel knopen bevat en veel geheugen inneemt (aantal knopen is terug te vinden via: *Berekening, net* > *netgeneratie*), is het verstandig om het aantal knopen te verminderen (dit gaat wel ten koste van de nauwkeurigheid). Het aantal knopen verminderen kan via: *Instellingen net* > *Gemiddelde grootte van 2D element / gekromd element [m]*. Vul i.p.v. de gebruikelijke 1 m een hogere waarde in.

De berekening zoals deze hierboven is uitgelegd gaat uit van het globale assenstelsel van het model. Om de berekeningen nauwkeuriger te maken en ook beter te laten aansluiten op de werkelijkheid, zouden de krommingen bepaald moeten worden aan de hand van het lokale assenstelsel.

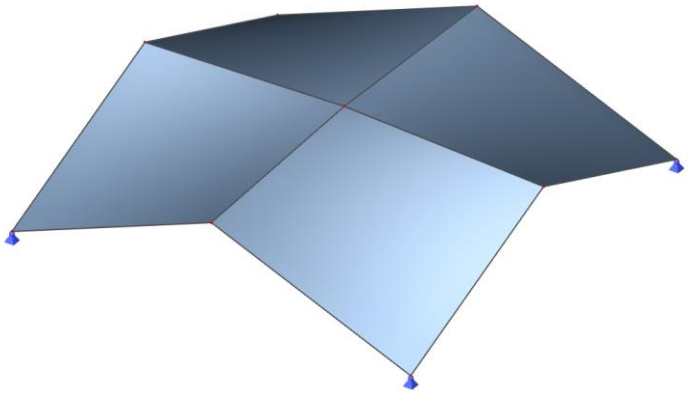
6.3 Gemodelleerde schaalconstructies

Om de knikformules te toetsen, is het noodzakelijk om schaalconstructies te modelleren waarop berekeningen kunnen worden gedaan. De vervormingen door de belasting via de eindige elementen methode kan dan worden vergeleken met de nieuwe knikformules.

In totaal zijn er drie schaalconstructies gemodelleerd. De eigenschappen van deze schaalconstructies staan hieronder genoemd. De constructies worden belast door het eigengewicht.

6.3.1 4-Hypar

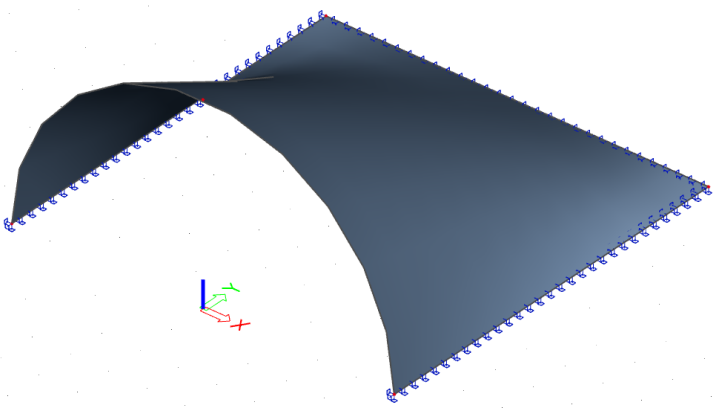
Overspanning:	50 m
Maximale hoogte:	11 m
Schaaldikte:	90 mm
Betonklasse:	C30/37
Wapening:	B400A
Betonnen staaf:	150 x 250 mm



Figuur 6.3 4-Hypar (SCIA Engineer)

6.3.2 Stalen conoid

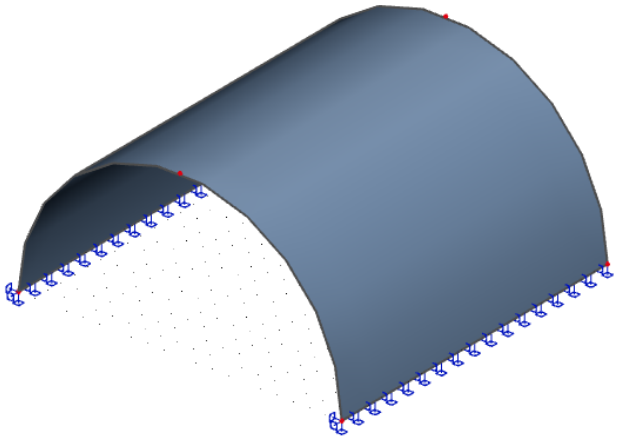
Overspanning:	10 m
Maximale hoogte:	5 m
Schaaldikte:	12 mm
Staalklasse:	S235



Figuur 6.4 Stalen conoid (SCIA Engineer)

6.3.3 Halve cilinder

Hoogte:	10 m
Breedte:	20 m
Lengte:	20 m
Schaaldikte:	100 mm
Betonklasse:	C30/37
Wapening:	B400A



Figuur 6.5 Halve cilinder (SCIA Engineer)

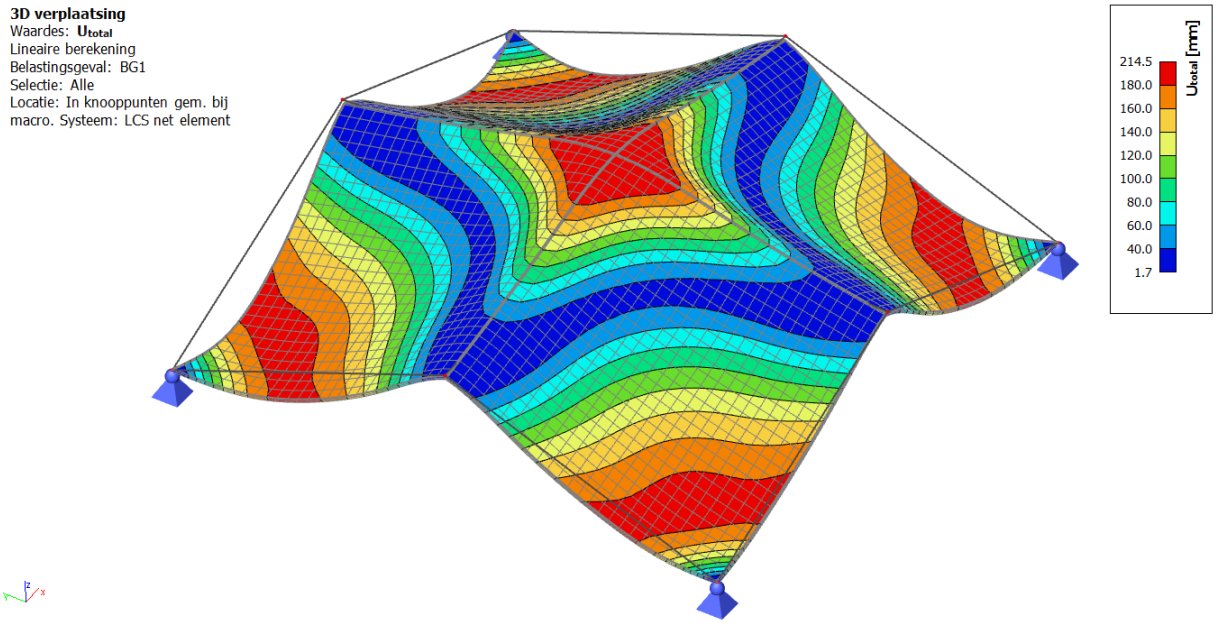
7

Beoordeling knikformules

Om de knikformules goed te kunnen beoordelen, zal het gedrag bij uitknikken worden vergeleken met de vervormingen van de constructie door de belasting volgens een eindige elementen berekening. Van de in *hoofdstuk 6* gemodelleerde schaalconstructies zien de vervormingen er als volgt uit. Tijdens de eindige elementen berekeningen is het *mesh* (netwerk van elementjes) automatisch gegenereerd. Bovendien is in de software aangegeven dat de buigtheorie van Kirchhoff gebruikt moet worden. De constructies ondervinden alleen de belasting van het eigengewicht. Overige gegevens per constructie staan in *Bijlage H*. Onder elke figuur is ook aangegeven wat de belastingfactor λ is van de eerste knikvorm.

7.1 Eindige elementenmethode

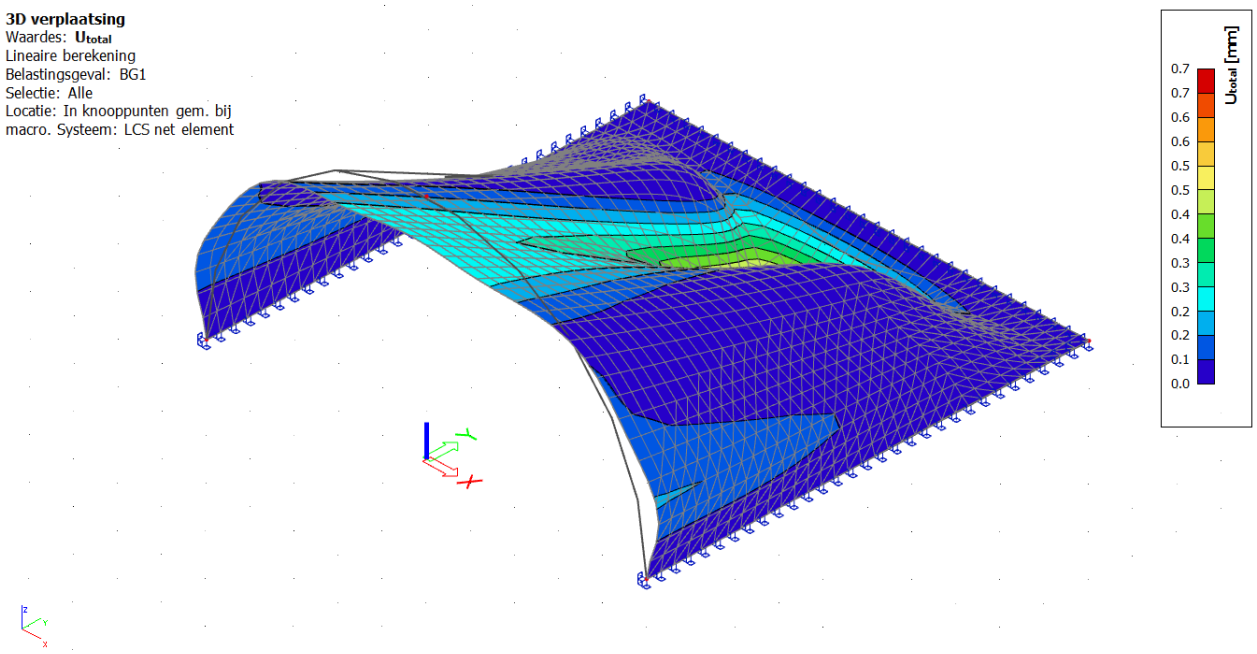
4-Hypar



Figuur 7.1 Vervormingen door de belasting (lineaire berekeningen): 4-Hypar (SCIA Engineer)

De eerste knikvorm heeft een belastingfactor λ van 0,90.

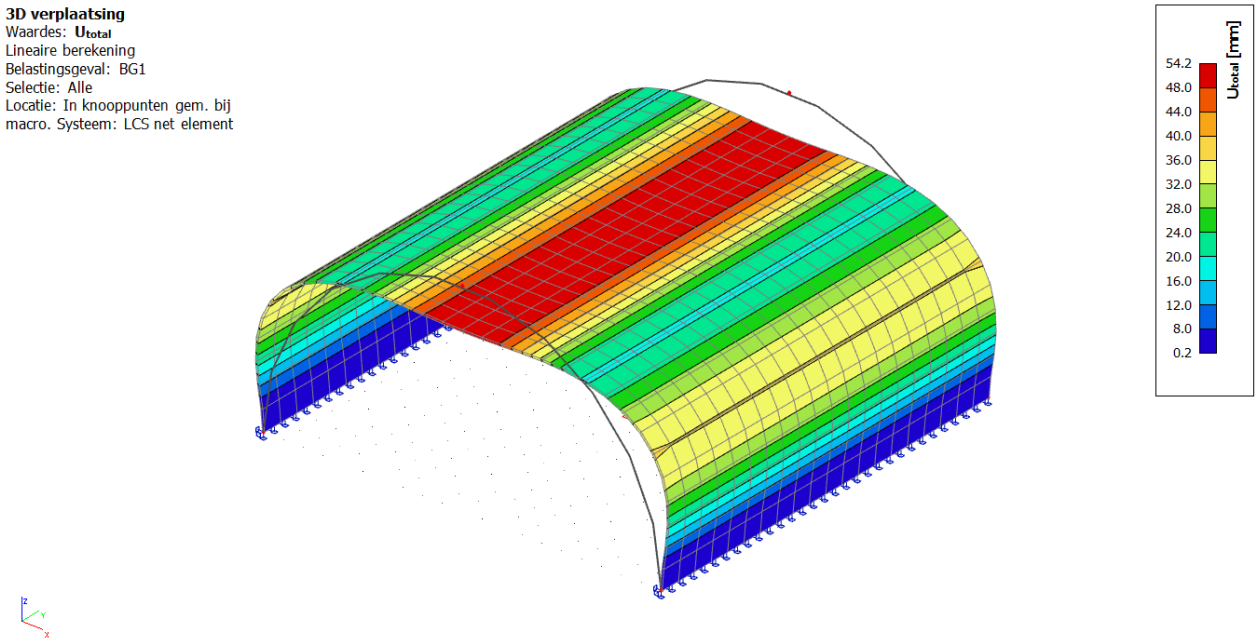
Stalen conoid



Figuur 7.2 Vervormingen door de belasting (lineaire berekeningen): stalen conoid (SCIA Engineer)

De eerste knikvorm heeft een belastingfactor λ van 26,19.

Halve cilinder



Figuur 7.3 Vervormingen door de belasting (lineaire berekeningen): halve cilinder (SCIA Engineer)

De eerste knikvorm heeft een belastingfactor λ van 9,90.

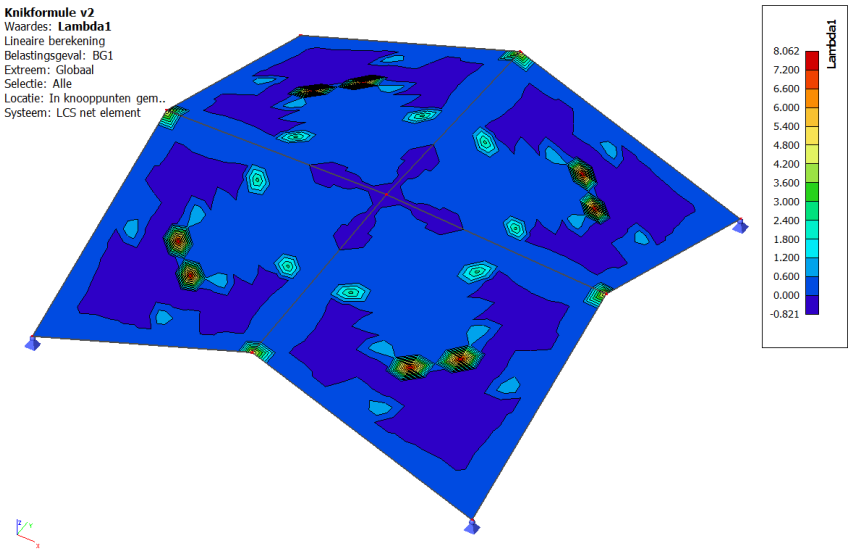
7.2 Knikformule

De vier verschillende knikformules λ_i uit *Hoofdstuk 5* worden voor de drie verschillende schaalconstructies onderzocht. Alle resultaten in de vorm van contourplots van de belastingfactor λ voor de drie verschillende schaalconstructies staan in *Bijlage J.* De grootte van de belastingfactoren in de contourplots zijn niet heel belangrijk, deze zouden namelijk toch altijd nog geschaald kunnen worden met een extra factor in de knikformule. Bij de beoordeling van de knikformules gaat het meer om de globale vorm en de locaties van de λ's in de buurt van nul. Juist op de locaties waar de belastingfactor nagenoeg nul is, zal de constructie makkelijk kunnen uitknikken. Het zijn dan ook deze plekken, welke vergeleken moeten worden met de vervormingen van de constructie gemaakt in SCIA Engineer. De contourplots kunnen in dat opzicht een vertekend beeld geven aangezien juist de locaties met een hoge belastingfactor (knikken minder snel uit), duidelijk aanwezig zijn met de kleur rood.

Bij het beoordelen van de knikformules zou ook gekeken kunnen worden of de symmetrie overeenkomt met de vervormingen op de contourplots. Echter is het verschil van symmetrie tussen de verschillende knikformules niet altijd toe te schrijven aan de knikformules zelf. De asymmetrie kan namelijk ontstaan doordat de berekeningen van de krommingen zijn gedaan in het globale assenstelsel en de berekeningen van de membraankrachten in het lokale assenstelsel. Het gebruiken van verschillende assenstelsels in een berekening is niet juist, echter is het in dit onderzoek, vanwege de beschikbare tijd, nog niet gelukt om de krommingen ook in het lokale assenstelsel uit te rekenen.

4-Hypar

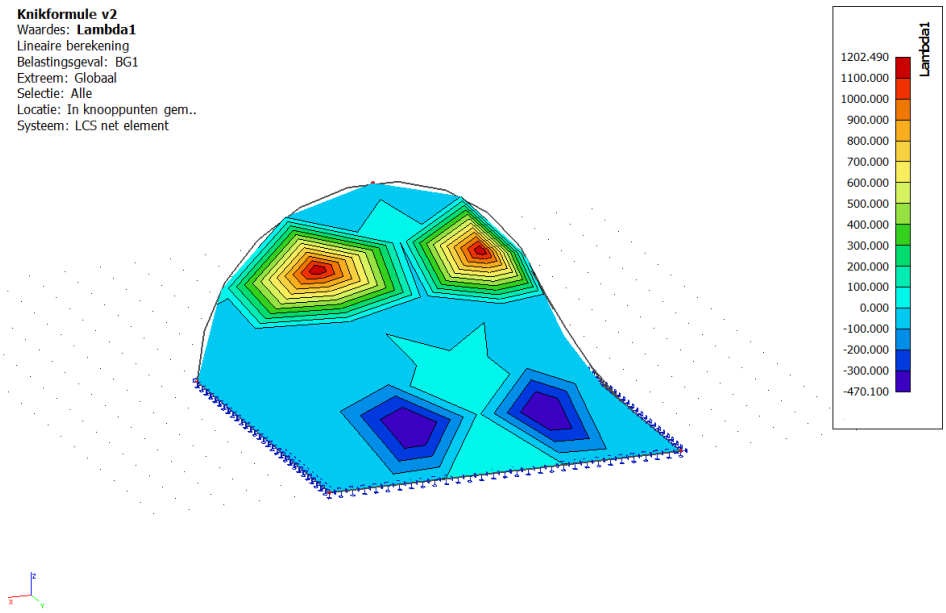
λ_2 en λ_3 geven qua totaalbeeld niet het gewenste resultaat, maar aangezien de krommingen en membraankrachten niet in hetzelfde assenstelsel zijn berekend, kunnen hieruit geen conclusies worden getrokken. Daarnaast laat λ_4 een apart resultaat zien, vrijwel de hele constructie zou eenvoudig kunnen uitknikken. Ook zijn er bij de randen interessante pieken te zien, wat zou kunnen duiden op een singulariteit. Al met al geeft λ_4 het minste resultaat. De knikvorm van de 4-hypar wordt echter wel redelijk goed beschreven door λ_1 (*figuur 7.4*).



Figuur 7.4 Knikformule 1 geeft beste resultaat bij 4-hypar (SCIA Engineer)

Stalen conoid

λ_2 en λ_3 vertonen heel onregelmatig gedrag. Net zoals bij de 4-Hypar zegt dit niet zo veel over de knikformule zelf, maar meer over de manier van het berekenen van de krommingen. De andere twee knikformules λ_1 en λ_4 vertonen een symmetrie. Van deze twee knikformules geeft λ_1 een beter resultaat. Dit gebied heeft een belastingfactor net iets groter dan nul wat ook overeenkomt met de vervormingen uit SCIA Engineer. Concluderend kan gesteld worden dat λ_1 het beste resultaat geeft (*figuur 7.5*).



Figuur 7.5 Knikformule 1 geeft beste resultaat bij stalen conoid (SCIA Engineer)

Halve cilinder

Bovenop de halve cilinder begint het uitknikken. De belastingfactor zou dan ook aan de bovenkant ongeveer nul moeten zijn. Voor de zijkanten van de constructie geldt een hogere belastingfactor. Dit onderscheidt is alleen terug te zien in λ_4. De andere knikformules hebben over het hele oppervlak van de constructie dezelfde belastingfactor. In dat opzicht zou λ_4 de knikvorm het beste beschrijven, echter heeft ook deze vierde knikformule ongewone eigenschappen; niet hetzelfde gedrag over de doorsnede en lokaal zijn er pieken te zien. Er kan dus niet duidelijk een beste knikformule worden aangewezen bij deze constructie. Daarbij moet worden opgemerkt dat de halve cilinder geen algemene schaalconstructie is aangezien er slechts in één richting sprake is van een kromming. De halve cilinder reageert in feite als een plaat.

Het moge dus duidelijk zijn dat conclusies trekken over de verschillende knikformules erg lastig is, aangezien de krommingen berekend in het globale assenstelsel een vertekend beeld kunnen geven. Pas nadat deze knikformules, in een mogelijk vervolgonderzoek, wel de juiste assenstelsels hanteren, kunnen duidelijke conclusies getrokken worden.

Conclusie

De focus in dit onderzoek ligt op het vinden van mogelijke nieuwe invarianten van twee tensoren en hoe de algemene knikformules, opgebouwd uit deze invarianten, gebruikt zouden kunnen worden om het knikgedrag te voorspellen. Dit onderzoek heeft tot de volgende conclusies geleid.

Er zijn twee onafhankelijke invarianten gevonden van de elementen van twee tensoren. Samen met de invarianten voor een enkelvoudige tensor kan het onderstaande overzicht worden gemaakt. Elke willekeurige combinatie van deze invarianten levert per definitie een nieuwe invariant op.

Invarianten van gecombineerde tensoren:

$$n_{xy}(k_{xx} - k_{yy}) - k_{xy}(n_{xx} - n_{yy})$$

$$n_{xx}k_{yy} - 2n_{xy}k_{xy} + n_{yy}k_{xx}$$

Invarianten van een enkelvoudige tensor:

$$n_{xx} + n_{yy} \quad \& \quad k_{xx} + k_{yy}$$

$$n_{xx}n_{yy} - {n_{xy}}^2 \quad \& \quad k_{xx}k_{yy} - {k_{xy}}^2$$

Er zijn ongeveer twaalf miljoen mogelijke invarianten onderzocht waarvan ongeveer 3000 inderdaad invariant zijn. Echter zijn deze 3000 invarianten allemaal te herleiden naar de twee bekende invarianten van gecombineerde tensoren.

Er zijn bovendien geen invarianten met quotiënten gevonden. Wat zou kunnen duiden op het feit dat quotiënten niet de juiste eigenschappen bezitten om in invarianten voor te komen.

Als de aanname klopt dat er geen andere invarianten meer zijn, is het zoeken naar de algemene knikformule een stuk eenvoudiger geworden. Het aantal mogelijkheden zou dan drastisch kleiner zijn, wat voor het onderzoek naar deze knikformules erg goed uitkomt.

De gevonden invarianten zijn geïmplementeerd in het programma SCIA Engineer, met behulp van de functionaliteit Design Forms Builder. Hieruit is gebleken dat het programmeren van formules in SCIA Engineer wel mogelijk is, maar nog niet heel gebruiksvriendelijk. De mogelijke problemen die kunnen optreden en de bijbehorende oplossingen zijn te vinden in *§6.2.2*.

Er zijn vier knikformules (λ_i) onderzocht waarvan λ_1 vooralsnog het beste resultaat geeft. Echter zijn de resultaten niet heel betrouwbaar omdat voor de berekening van de krommingen niet het lokale assenstelsel is gebruikt.

$$\lambda_1 = \mathrm{E} \cdot t^2 \cdot \frac{k_{xx} + k_{yy}}{n_{xx} + n_{yy}}$$

Aanbevelingen

Naar aanleiding van het onderzoek dat is gedaan en de conclusies die daaruit volgen is het aan te bevelen om vervolgonderzoek te doen. Zoals uit dit onderzoek blijkt zijn invarianten erg zeldzaam, om aan te tonen dat er überhaupt nog invarianten bestaan verdient het de voorkeur om verder te zoeken naar deze invarianten. Ook kan er gezocht worden naar een wiskundige verklaring voor het feit dat er mogelijke geen andere invarianten meer zijn.

In dit onderzoek is het globale assenstelsel gebruikt voor de berekeningen, dit resulteert in afwijkingen ten opzichte van het echte resultaat. Om de kwaliteit van de berekeningen te verbeteren in Design Forms Builder zal uitvoerig gekeken moeten worden naar het implementeren van het lokale assenstelsel.

De knikformules uit dit rapport zouden dan opnieuw onderzocht kunnen worden volgens het lokale assenstelsel. Ook is het aan te bevelen om andere knikformules te onderzoeken die het gedrag van uitknikken nog beter beschrijven.

In een mogelijk vervolgonderzoek is het aan te raden om de pieken in de contourplots van de knikformules te verwijderen. Deze pieken geven een foutief beeld, want juist de waarden net iets groter dan nul zijn interessant. Om misverstanden te voorkomen en de resultaten makkelijker te bespreken zal dit aangepast moeten worden.

Bijlage A

Python script: Analytische methode

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
from scipy import stats
from sympy import *
import itertools
from pprint import pprint
```

```
Sxx = Symbol('Sxx')
Syy = Symbol('Syy')
Sxy = Symbol('Syy')
Kxx = Symbol('Kxx')
Kyy = Symbol('Kyy')
Kxy = Symbol('Kxy')
alpha = Symbol('alpha')
SXX = (0.5*(Sxx+Syy) + 0.5*(Sxx-Syy)*cos(2*alpha) + Sxy*sin(2*alpha))
SYY = (0.5*(Sxx+Syy) - 0.5*(Sxx-Syy)*cos(2*alpha) - Sxy*sin(2*alpha))
SXY = (-0.5*(Sxx-Syy)*sin(2*alpha) + Sxy*cos(2*alpha))
KXX = (0.5*(Kxx+Kyy) + 0.5*(Kxx-Kyy)*cos(2*alpha) + Kxy*sin(2*alpha))
KYY = (0.5*(Kxx+Kyy) - 0.5*(Kxx-Kyy)*cos(2*alpha) - Kxy*sin(2*alpha))
KXY = (-0.5*(Kxx-Kyy)*sin(2*alpha) + Kxy*cos(2*alpha))
```

```
def test_invariant(combination):
    test = str(simplify(combination))
    if 'alpha' in test:
        return False
    if test == '0':
        return False
    else:
        return True
```

```
A = ['+','-']
B = ['(',')']
C = ['*','+','-']
S = ['SXX', 'SXY', 'SYY']
K = ['KXX', 'KXY', 'KYY']
N = ['1*','2*','3*']
P = ['**2']
combinations = [### Combination input here###]
```

```
Invariants = []
for k in range(len(combinations)):
    inputdata = combinations[k]
    result = list(itertools.product(*inputdata))
    print('##### Total permutations:', len(result), '; in combination k:', k+1, '/', len(combinations))
    for i in range(len(result)):
        print('iteration:',i+1,'/', len(result), end='\r', flush=True)
        TEST = ''.join(map(str,result[i]))
        if test_invariant(eval(TEST)) == True:
            print('INVARIANT:',TEST)
            Invariants.append(TEST)
print('Done! Amount of invariants found:', len(Invariants))
```

Bijlage B

Python script: Numerieke methode

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
from scipy import stats
from sympy import *
import itertools
from pprint import pprint
```

```
Sxx = 4  #Random waarden voor Sxx tm Kxy
Syy = 1
Sxy = 2
Kxx = 1
Kyy = 5
Kxy = 3
alpha = np.arange(0, 2*np.pi, 0.01)
SXX = 0.5*(Sxx+Syy) + 0.5*(Sxx-Syy)*np.cos(2*alpha) + Sxy*np.sin(2*alpha)
SYY = 0.5*(Sxx+Syy) - 0.5*(Sxx-Syy)*np.cos(2*alpha) - Sxy*np.sin(2*alpha)
SXY = -0.5*(Sxx-Syy)*np.sin(2*alpha) + Sxy*np.cos(2*alpha)
KXX = 0.5*(Kxx+Kyy) + 0.5*(Kxx-Kyy)*np.cos(2*alpha) + Kxy*np.sin(2*alpha)
KYY = 0.5*(Kxx+Kyy) - 0.5*(Kxx-Kyy)*np.cos(2*alpha) - Kxy*np.sin(2*alpha)
KXY = -0.5*(Kxx-Kyy)*np.sin(2*alpha) + Kxy*np.cos(2*alpha)
def test_invariant(comb):
    slope = stats.linregress(alpha, comb)[0]
    if -0.1 <= slope <= 0.1:
        if (comb[0] - 0.1) <= comb[200] <= (comb[0] + 0.1):
            if (comb[50] - 0.1) <= comb[350] <= (comb[50] + 0.1):
                if (comb[123] - 0.1) <= comb[544] <= (comb[123] + 0.1):
                    if (comb[461] - 0.1) <= comb[611] <= (comb[461] + 0.1):
                        return True
```

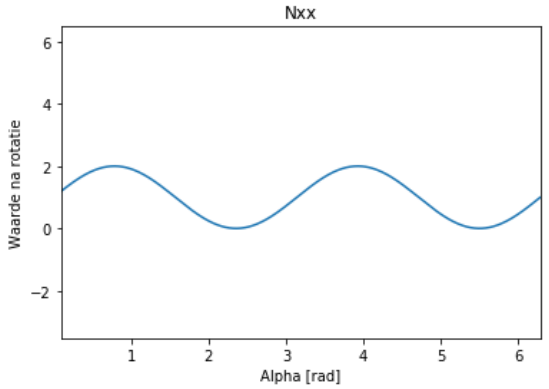
```
A = ['+','-']
B = ['(',')']
C = ['*','+','-']
S = ['SXX', 'SXY', 'SYY']
K = ['KXX', 'KXY', 'KYY']
N = ['1*','2*','3*']
P2 = ['**2']
P3 = ['**3']
D = ['1/']
combinations = [[### Combination input here ###]]
```

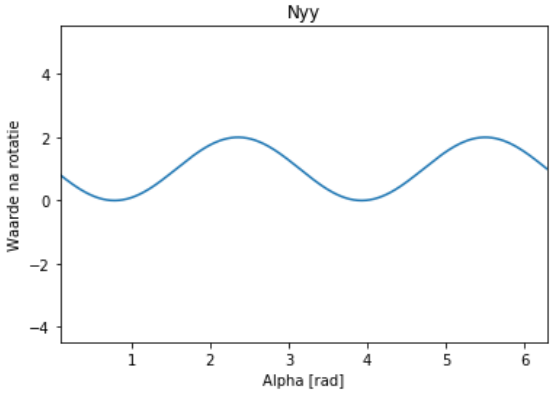
```
Invariants = []
for k in range(len(combinations)):
    inputdata = combinations[k]
    result = list(itertools.product(*inputdata))
    print('##### Total permutations:', len(result), '; in combination k:', k+1, '/', len(combinations))
    for i in range(len(result)):
        if (i+1) % 500 == 0:
            print('iteration:',i+1,'/', len(result), end='\r', flush=True)
        TEST = ''.join(map(str,result[i]))
        if test_invariant(eval(TEST)) == True:
            print('INVARIANT:',TEST)
            Invariants.append(TEST)
print('Done! Amount of invariants found:', len(Invariants))
```

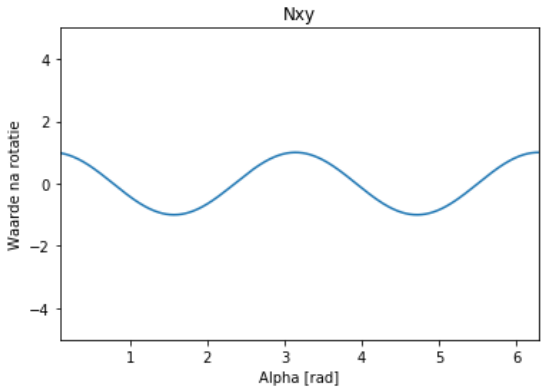
Bijlage C

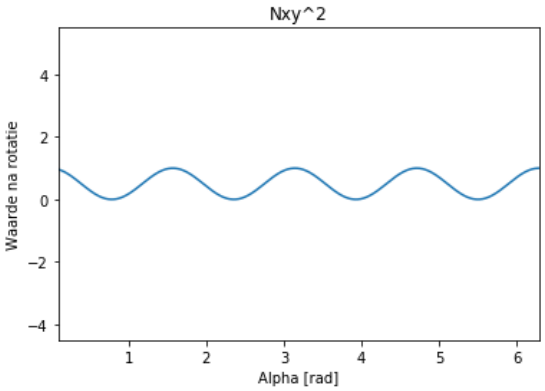
Grafisch voorbeeld

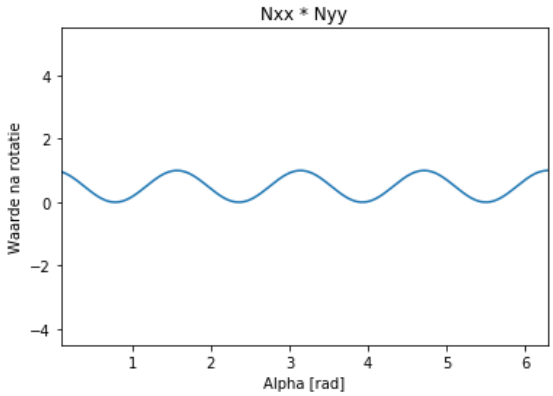
Hieronder is het verband te zien tussen de originele waarde en de waarde na rotatie (tussen 0 en 2π) van $n_{xx},\ n_{xy},\ n_{yy}$. Als deze op een juiste manier worden gecombineerd levert dit inderdaad een constante waarde op. Dit is het geval voor de bekende invariant $n_{xx}n_{yy} - {n_{xy}}^2$.

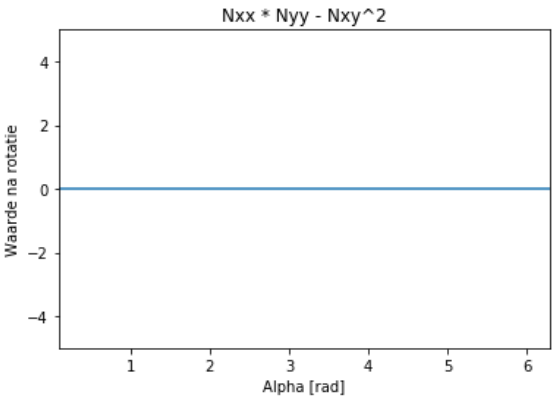





Nxy
Waarde na rotatie
Alpha [rad]


Nxy^2
Waarde na rotatie
Alpha [rad]


Nxx * Nyy
Waarde na rotatie
Alpha [rad]


Nxx * Nyy - Nxy^2
Waarde na rotatie
Alpha [rad]

Bijlage D

Analytische methode: onderzochte globale vormen

Globale vorm	Aantal combinaties
N,S,*,S,*,K,A,N,K,*,K,*,S	13.122
N,S,*,K,A,N,S,*,K	1.458
N,S,A,N,S,A,N,K,A,N,K	52.488
(,S,*,K,A,S,*,K,),C,(,N,S,*,K,)	13.122
S,*,K,A,S,P,A,S,*,K,A,K,P	5.832
S,*,S,A,S,P,A,K,*,K,A,K,P	5.832
(,S,*,K,),P,A,K,*,K,A,S,*,S	2.916
N,S,P,A,N,S,*,K,A,N,K,P	8.748
(,S,P,C,K,P,),C,(,K,P,C,S,P,)	13.122
(,S,*,K,*,S,A,K,*,S,*,K,),*,S,*,K	2.187
N,S,+,S,A,K,+,K,A,N,S,*,K	26.244
(,S,+,S,A,S,),C,(,K,A,K,+,K,)	8.748
(,S,A,K,+,S,),C,(,K,+,S,A,K,)	8.748
(,S,+,K,+,N,S,),C,(K,+,S,+,N,K,)	19.683
N,S,*,K,A,N,S,P,A,N,K,P	8.748
(,N,S,-,K,),P,+,(,N,K,-,S,),P	729
totaal aantal combinaties	191.727

Bijlage E

Numerieke methode: onderzochte globale vormen

Globale vorm	Aantal combinaties
N,S,*,S,*,K,A,N,K,*,K,*,S	13.122
N,S,*,K,A,N,S,*,K	1.458
N,S,A,N,S,A,N,K,A,N,K	52.488
(,S,*,K,A,S,*,K,),C,(,N,S,*,K,)	13.122
S,*,K,A,S,P,A,S,*,K,A,K,P	5.832
S,*,S,A,S,P,A,K,*,K,A,K,P	5.832
(,S,*,K,),P,A,K,*,K,A,S,*,S	2.916
N,S,P,A,N,S,*,K,A,N,K,P	8.748
(,S,P,C,K,P,),C,(,K,P,C,S,P,)	13.122
(,S,*,K,*,S,A,K,*,S,*,K,),*,S,*,K	2.187
N,S,+,S,A,K,+,K,A,N,S,*,K	26.244
(,S,+,S,A,S,),C,(,K,A,K,+,K,)	8.748
(,S,A,K,+,S,),C,(,K,+,S,A,K,)	8.748
(,S,+,K,+,N,S,),C,(K,+,S,+,N,K,)	19.683
N,S,*,K,A,N,S,P,A,N,K,P	8.748
(,N,S,-,K,),P,+,(,N,K,-,S,),P	729
N,S,C,S,C,S,A,K,C,K,P	39.366

N,S,*,S,C,N,S,*,K,C,N,K,*,K,P	177.147
N,S,P3,C,N,S,*,K,P3,C,N,K,P3	19.683
N,S,P3,C,N,S,P,*,K,P,C,N,K,P3	19.683
D,N,S,C,S,A,K,C,S	4.374
(,D,N,S,C,K,P3,),C,S,C,S,C,S	59.049
N,C,S,C,S,C,S,A,N,K,C,K,C,K	3.188.646
N,(,S,C,K,),P,C,N,(,K,C,S,),P	19.683
N,S,*,S,*,S,A,N,K,*,K,*,K,A,N,S,*,K,P	708.588
N,S,/,K,C,N,S,/,K,C,N,S,/,K	177.147
N,S,/,K,C,N,K,/,S,C,N,S,/,K,P	177.147
N,S,C,S,C,S,C,K,C,S	59.049
N,S,P3,C,K,C,S,C,N,K,P3,C,S	177.147
S,C,(,K,C,K,),C,(,S,C,S,)	19.683
N,K,C,S,P3,A,S,C,(,K,P,C,N,K,)	118.098
N,S,*,S,*,K,A,N,S,*,K,*,K,A,N,(,S,*,K,),P	708.588
N,S,/,K,C,N,S,/S,P3,C,N,S,/,K	177.147
N,S,*,K,C,N,S,/,S,P3,C,N,S,/,K	177.147
(,N,S,*,S,C,N,K,*,K,),/,(,S,*,N,K,P3,)	59.049
(,N,S,P,*,S,C,N,K,P,*,K,),/,(,S,*,N,K,P3,)	59.049
totaal aantal combinaties	6.337.197

Bijlage F

Numerieke methode: onderzochte globale vormen: quotiënten

Globale vorm	Aantal combinaties
N,S,D,K,A,N,S,D,K	19.683
N,S,P2,D,K,A,N,S,D,K	19.683
N,S,P3,D,K,A,N,S,D,K,P2	19.683
N,S,P2,D,K,A,N,S,P3,D,K	19.683
N,S,P3,D,K,P2,A,N,S,P2,D,K,P3	19.683
(,N,K,A,N,S,),D,N,S,P2	59.049
(,N,K,A,N,S,),D,N,S,P3	59.049
(,K,A,S,),P2,D,S	81
N,S,*,K,D,N,K,*,K,A,K,*,S	177.147
N,S,P2,*,K,D,N,K,P2,*,K,A,K,P2,*,S	177.147
S,*,K,P3,D,N,K,P3,*,K,A,N,K,P3,*,S	177.147
(,K,A,S,A,K,),D,(,S,A,K,A,S,)	59.049
(,N,K,P2,A,S,A,K,),D,(,S,A,N,K,)	531.441
K,D,S,A,S,D,K,A,S,D,S,A,K,D,K	52.488
K,D,S,A,N,S,D,K,A,N,S,D,S,A,K,D,K	4.251.528
totaal aantal combinaties	5.642.541

Bijlage G

Vermenigvuldiging twee karakteristieke polynomen en bijbehorende invarianten

Het karakteristieke polynoom van één tensor ziet er als volgt uit:

$$\lambda^2 - (k_{xx} + k_{yy})\lambda + (k_{xx}k_{yy} - {k_{xy}}^2)$$

De constanten zijn de invarianten van de tensor. Een eerste beschouwing van de mogelijke invarianten van twee gecombineerde tensoren kan gedaan worden door twee karakteristieke polynomen te vermenigvuldigen:

$$(\lambda^2 - (k_{xx} + k_{yy})\lambda + k_{xx}k_{yy} - {k_{xy}}^2) \cdot (\lambda^2 - (n_{xx} + n_{yy})\lambda + n_{xx}n_{yy} - {n_{xy}}^2)$$

Uitwerken levert de volgende vergelijking op:

$$\begin{aligned}\lambda^4 - (k_{xx} + k_{yy})\lambda^3 + (k_{xx}k_{yy} - {k_{xy}}^2)\lambda^2 - (n_{xx} + n_{yy})\lambda^3 + (n_{xx} + n_{yy})(k_{xx} + k_{yy})\lambda^2 \\ - (n_{xx} + n_{yy})(k_{xx}k_{yy} - {k_{xy}}^2)\lambda + (n_{xx}n_{yy} - {n_{xy}}^2)(k_{xx}k_{yy} - {k_{xy}}^2) \\ + (n_{xx}n_{yy} - {n_{xy}}^2)\lambda^2 - (n_{xx}n_{yy} - {n_{xy}}^2)(k_{xx} + k_{yy})\lambda \\ + (n_{xx}n_{yy} - {n_{xy}}^2)(k_{xx}k_{yy} - {k_{xy}}^2)\end{aligned}$$

Alle constanten in deze vergelijking zijn invarianten, namelijk:

$$I = (k_{xx} + k_{yy})$$

$$II = (k_{xx}k_{yy} - {k_{xy}}^2)$$

$$III = (n_{xx} + n_{yy})$$

$$IV = (n_{xx} + n_{yy})(k_{xx} + k_{yy}) = n_{xx}k_{xx} + n_{xx}k_{yy} + n_{yy}k_{xx} + n_{yy}k_{yy}$$

$$V = (n_{xx} + n_{yy})(k_{xx}k_{yy} - {k_{xy}}^2) = n_{xx}k_{xx}k_{yy} - n_{xx}{k_{xy}}^2 + n_{yy}k_{xx}k_{yy} - n_{yy}{k_{xy}}^2$$

$$VI = (n_{xx}n_{yy} - {n_{xy}}^2)(k_{xx}k_{yy} - {k_{xy}}^2) = n_{xx}n_{yy}k_{xx}k_{yy} - n_{xx}n_{yy}{k_{xy}}^2 - k_{xx}k_{yy}{n_{xy}}^2 + {n_{xy}}^2{k_{xy}}^2$$

$$VII = (n_{xx}n_{yy} - {n_{xy}}^2)$$

$$VIII = (n_{xx}n_{yy} - {n_{xy}}^2)(k_{xx} + k_{yy}) = n_{xx}n_{yy}k_{xx} + n_{xx}n_{yy}k_{yy} - k_{xx}{n_{xy}}^2 - k_{yy}{n_{xy}}^2$$

Bijlage H

Overige gegevens schaalconstructies

4-Hypar

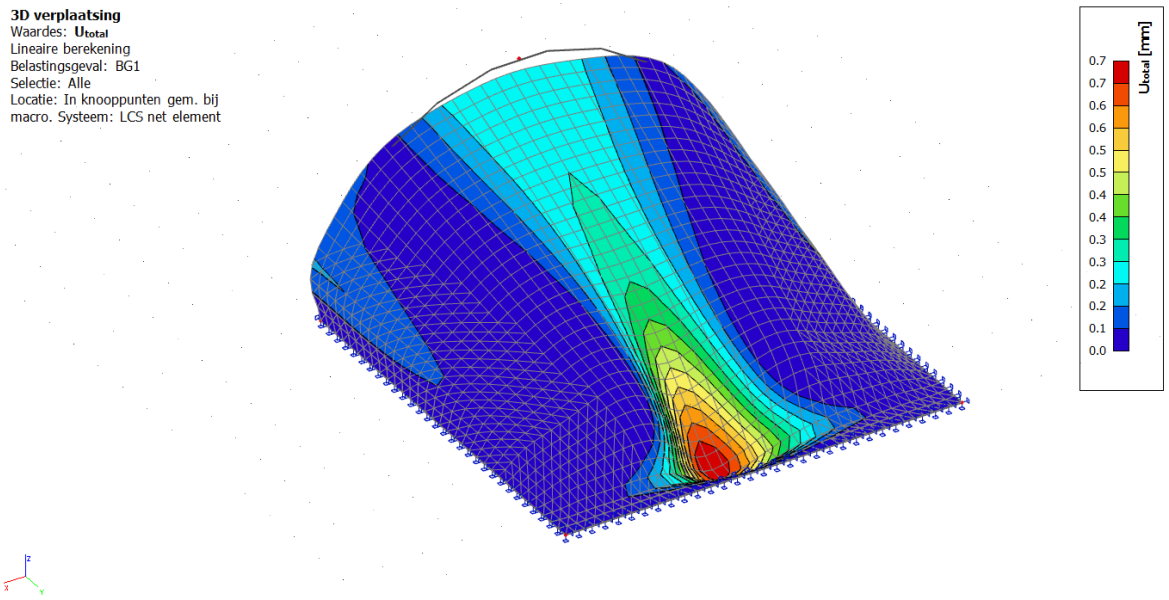
2809 knopen in mesh

Belastingfactoren	
1	0,90
2	1,24

Stalen conoid

1462 knopen in mesh

Belastingfactoren	
1	26,19
2	26,35



Halve cilinder

693 knopen in mesh

Belastingfactoren	
1	9,90
2	18,04

Bijlage

Design Forms Builder code

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


TEXT("5 Closest points");
double Lambda1 = 0.0;
double Lambda2 = 0.0;
double Lambda3 = 0.0;
double Lambda4 = 0.0;


double PointsCounter = IO.Member2D.Point.Count-1;
FOR(i, 0, PointsCounter) {
    X[i] = IO.Member2D.Point[i].X;
    Y[i] = IO.Member2D.Point[i].Y;
    Z[i] = IO.Member2D.Point[i].Z; }

double s;
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");
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; }
```

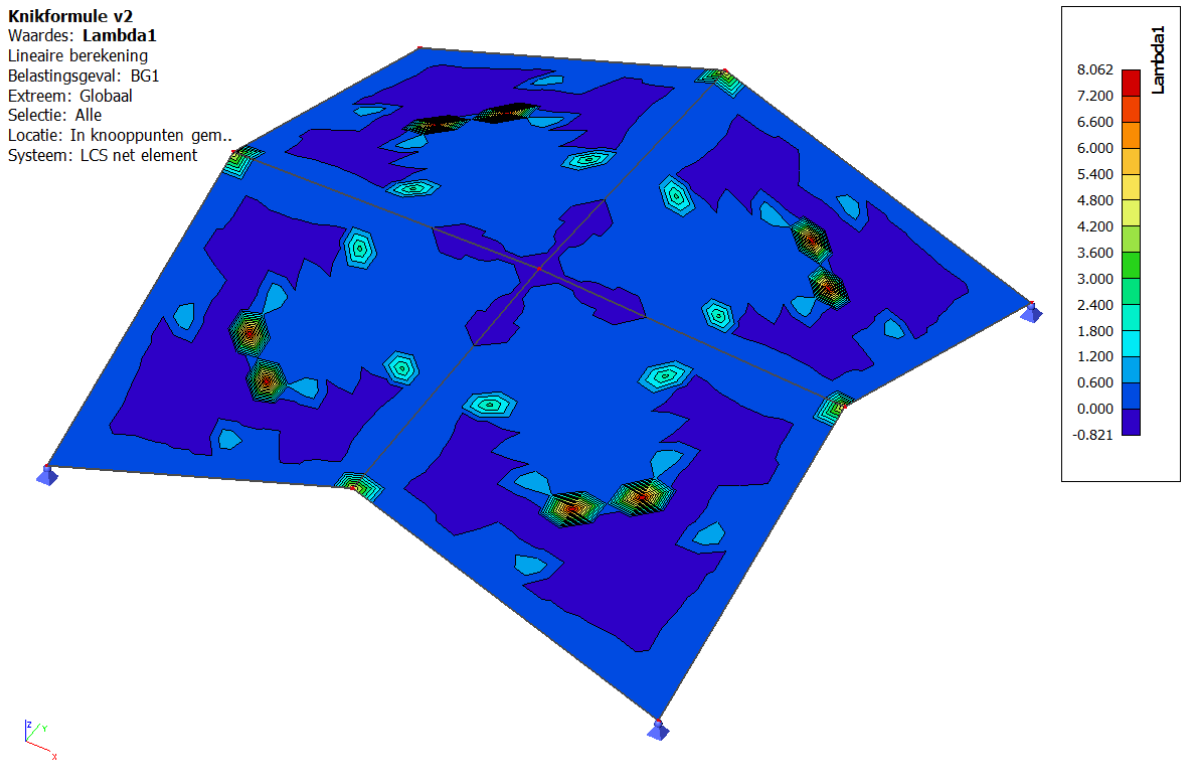
```
TEXT("Swap rows");
FOR(n, 0, 5) {
    double r = 0;
    IF(n < 5) {
        IF(ABS(M[n][0]) == ABS(M[n][1])) { double r = 1; } }
    IF(M[n][n] == 0) {
        IF(n < 1 && M[n+5][n] != 0) { r = 5; }
        IF(n < 2 && M[n+4][n] != 0) { r = 4; }
        IF(n < 3 && M[n+3][n] != 0) { r = 3; }
        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("Solve");
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("Curvatures");
kxx = 2*α[0];
kxy = α[1];
kyy = 2*α[2];

TEXT("Results");
Lambda1 = factor*E*t^2*(kxx+kyy)/(nxx+nyy);
Lambda2 = factor*E*t^2*(nxx*kxx+2*nxy*kxy+nyy*kyy)/(nxx+nyy)^2;
Lambda3 = factor*E*t^2*(kxx+kyy)^2/(nxx*kxx+2*nxy*kxy+nyy*kyy);
Lambda4 = factor*E*t^2*(kxx+kyy+kxx*kyy-kxy^2)/(nxx+nyy+nxx*nyy-nxy^2);
```

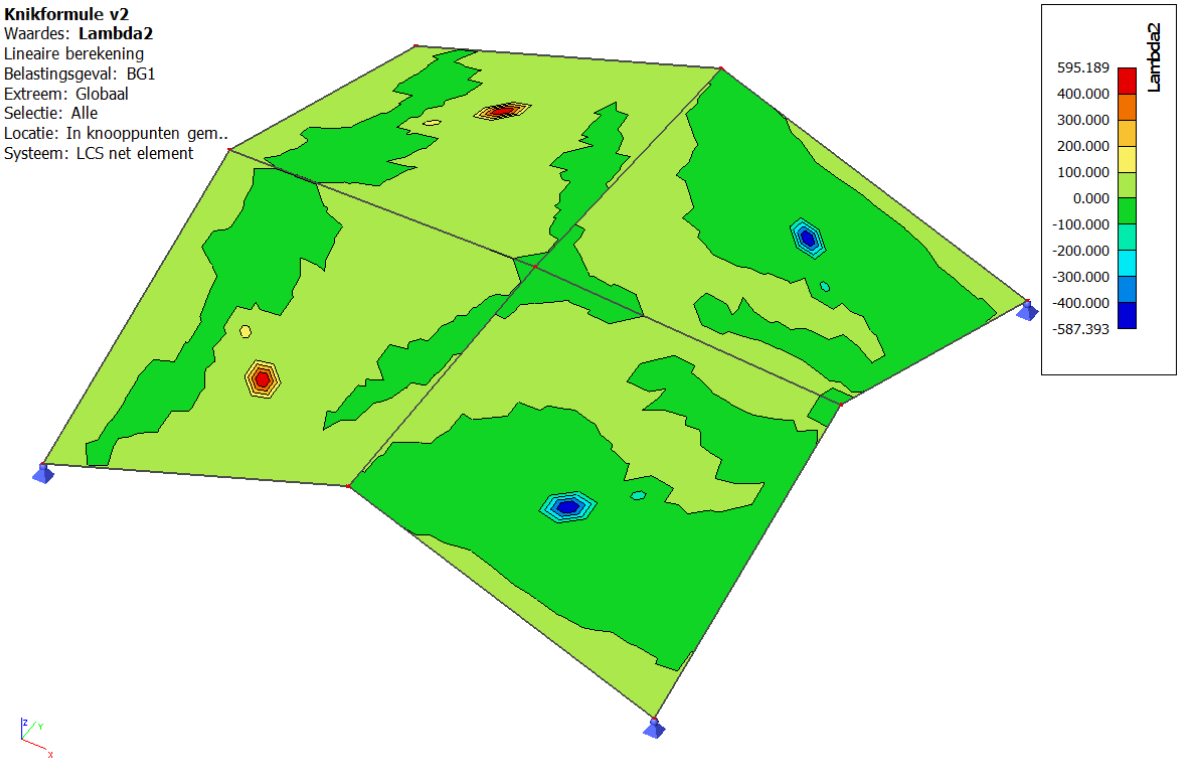
Bijlage J

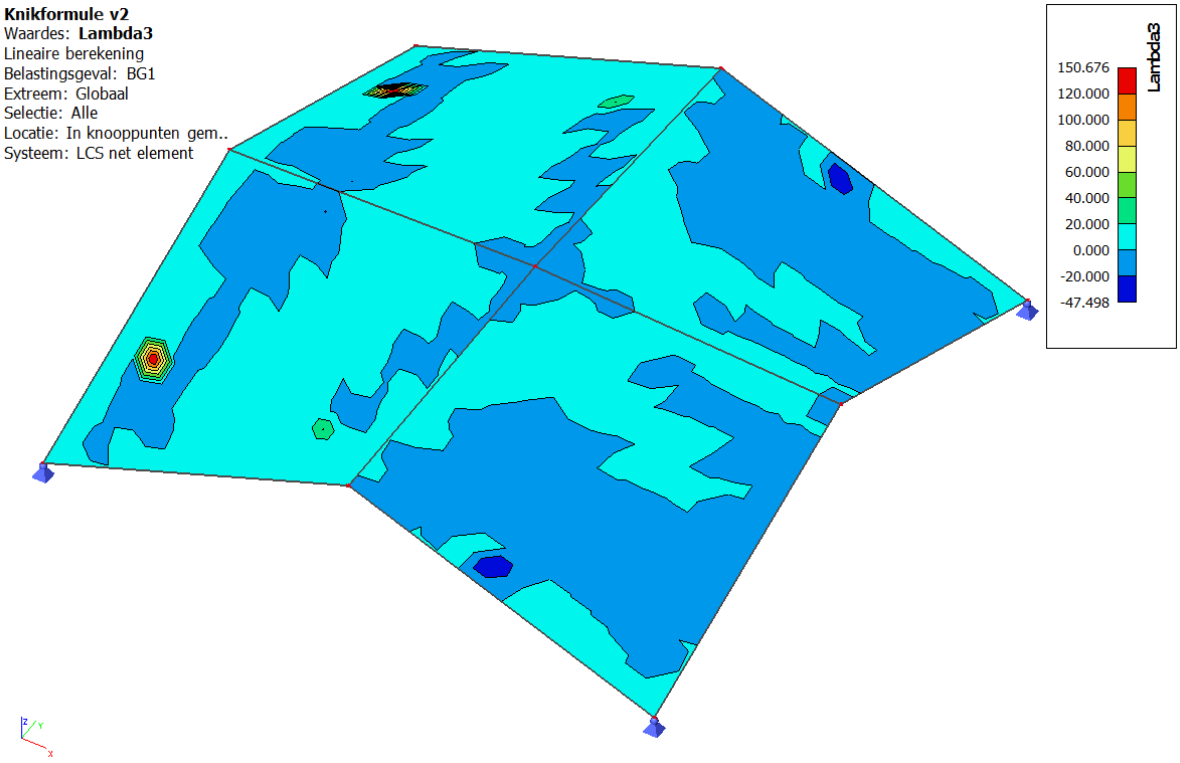
Contourplots knikformules in SCIA Engineer m.b.v. Design Forms Builder

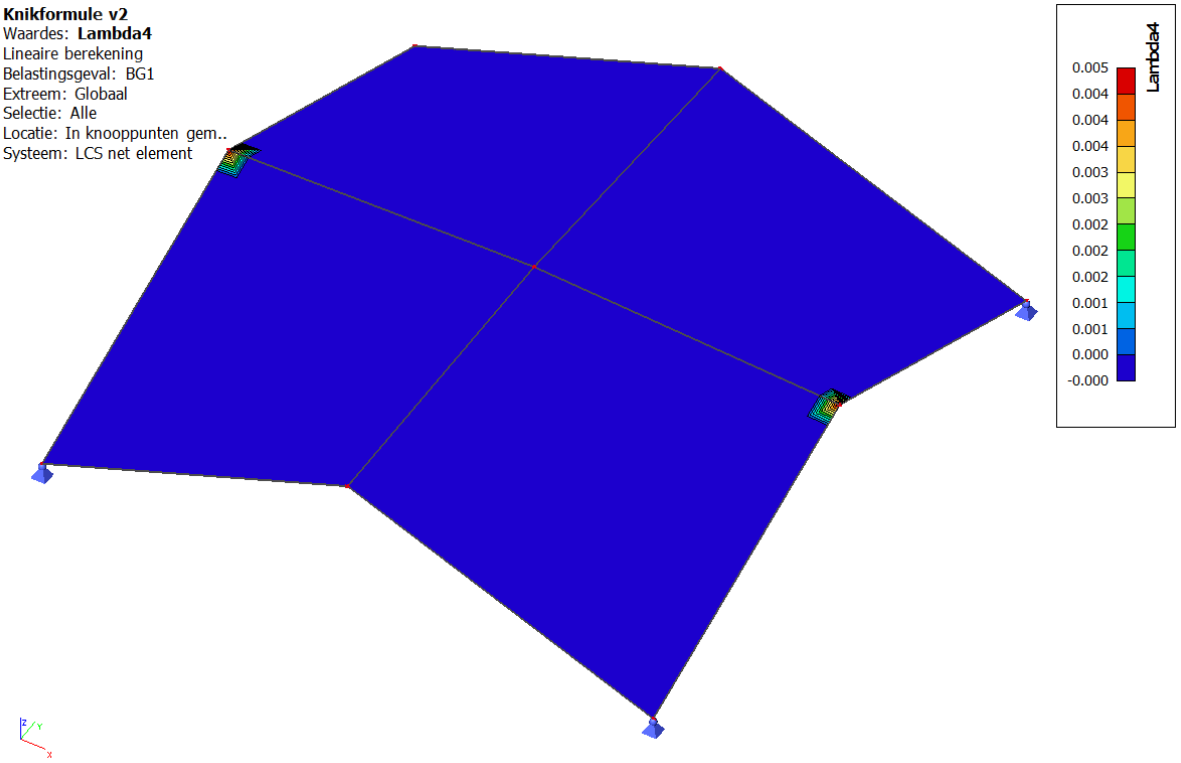
Linksboven in elke figuur staat om welke knikformule λ (Lambda) het gaat.

4-Hypar

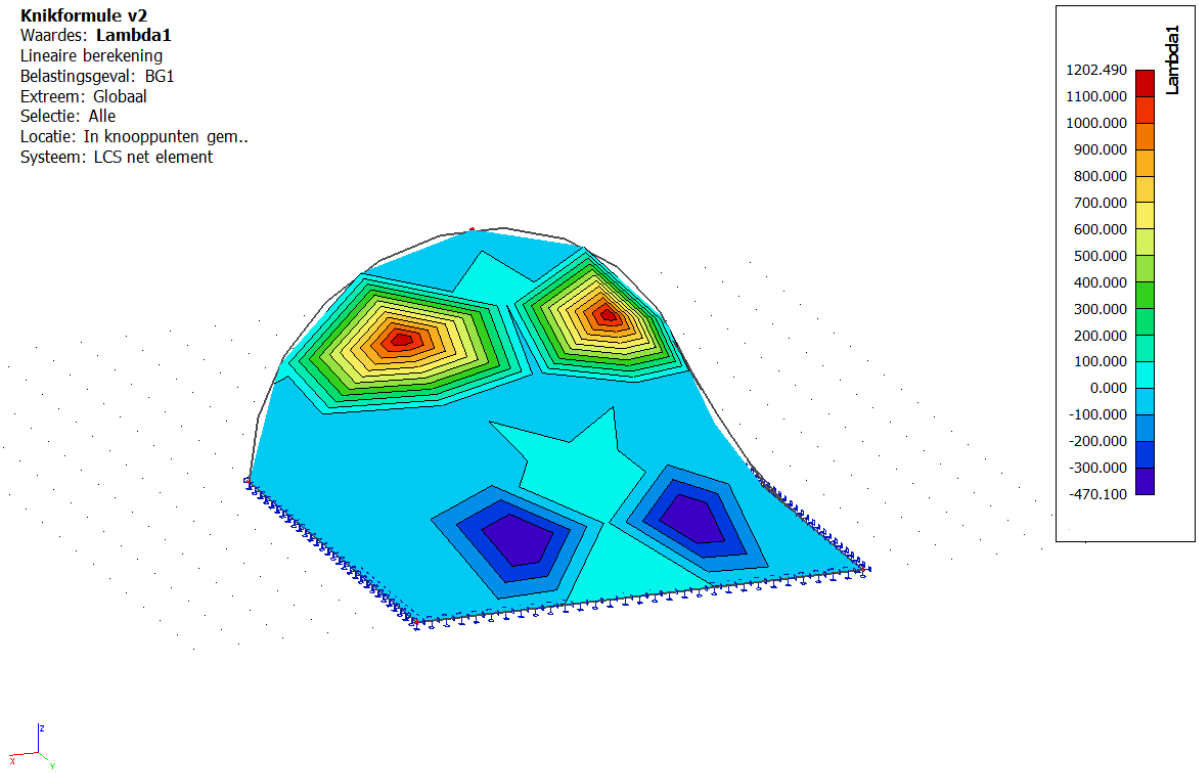


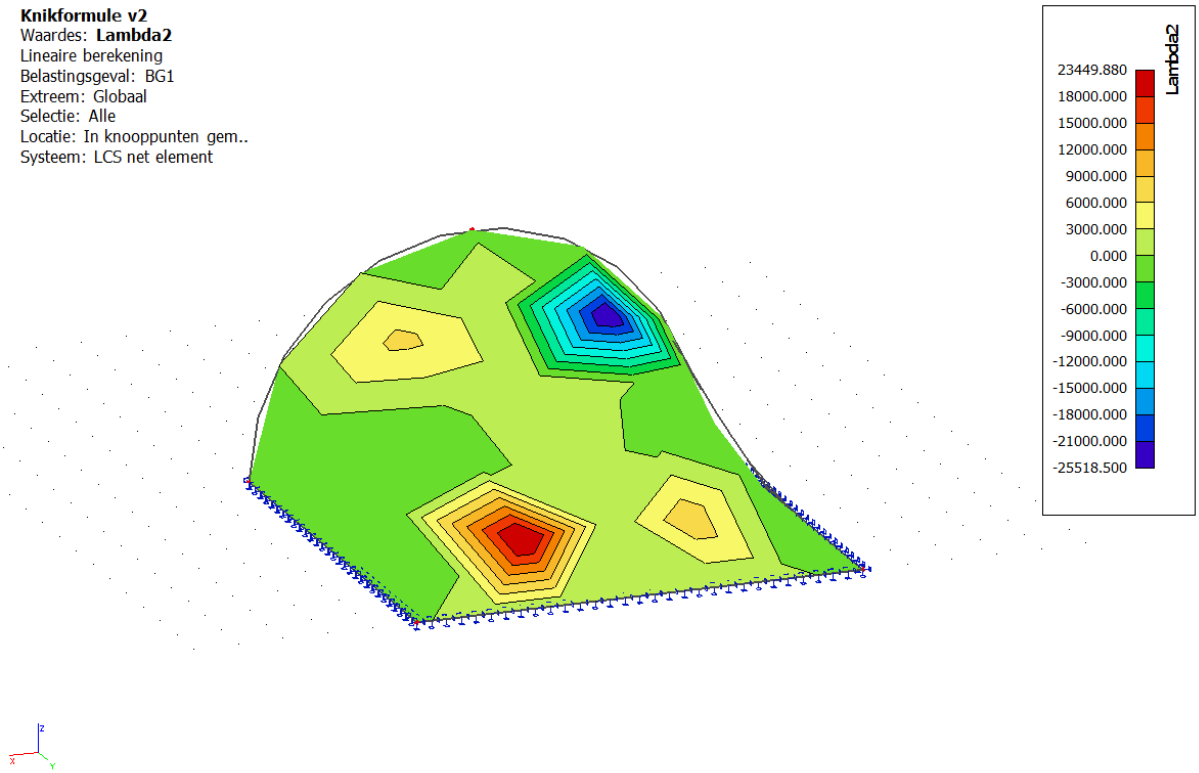

Knikformule v2
Waardes: Lambda2
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element
Lambda2
595.189
400.000
300.000
200.000
100.000
0.000
-100.000
-200.000
-300.000
-400.000
-587.393


Knikformule v2
Waardes: Lambda3
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element
Lambda3
150.676
120.000
100.000
80.000
60.000
40.000
20.000
0.000
-20.000
-47.498

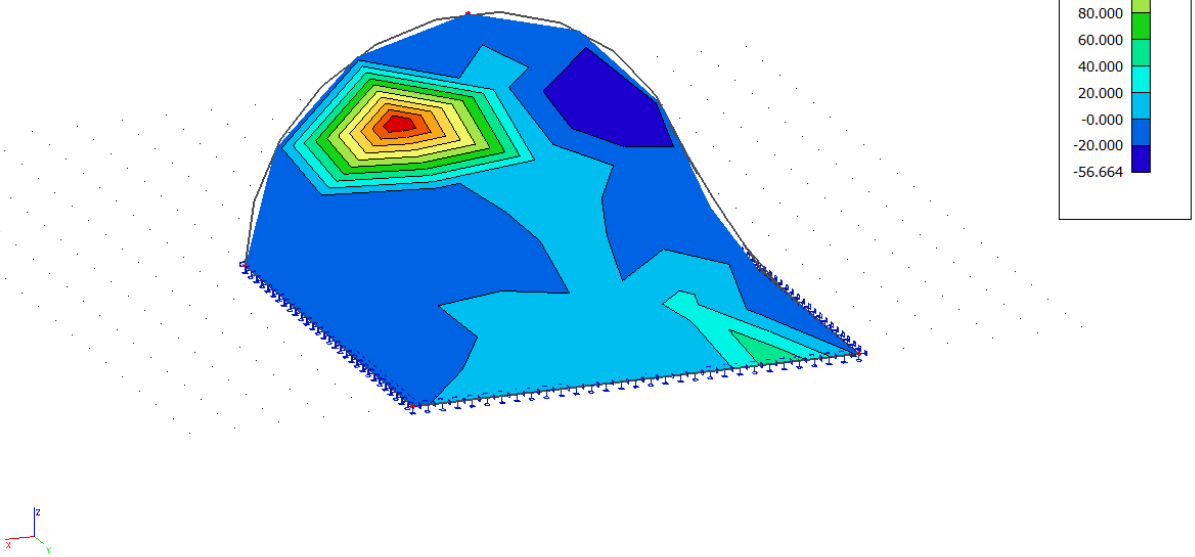

Knikformule v2
Waardes: Lambda4
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element
Lambda4
0.005
0.004
0.004
0.004
0.003
0.003
0.002
0.002
0.002
0.001
0.001
0.000
-0.000

Stalen conoid

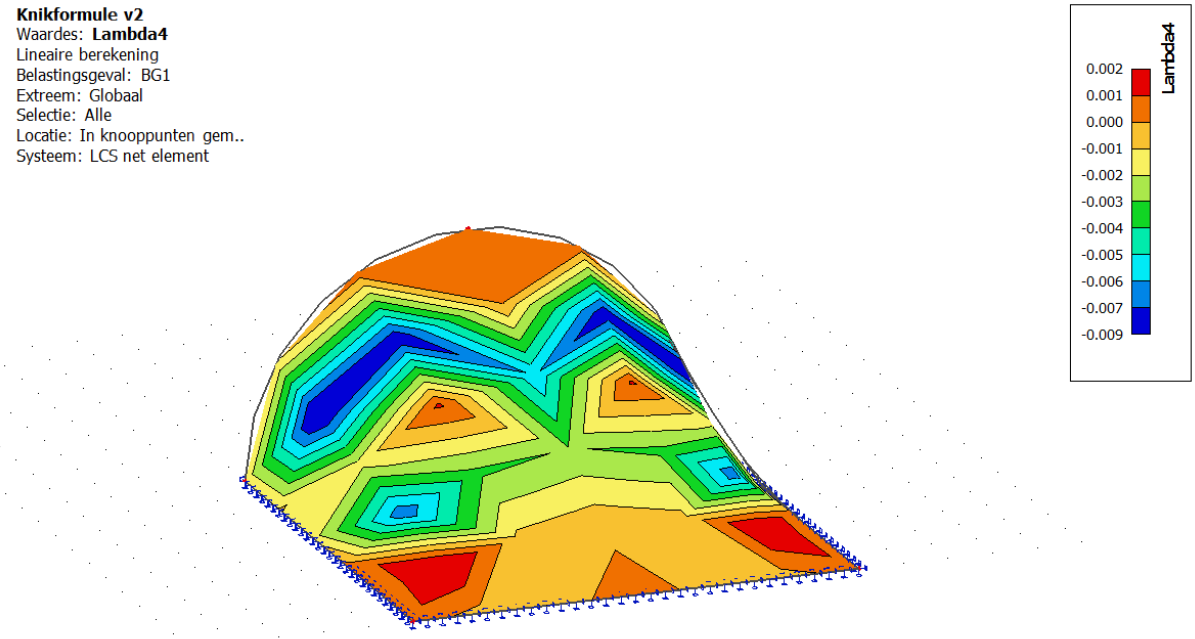




Knikformule v2
Waardes: **Lambda3**
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element

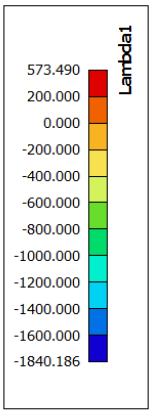


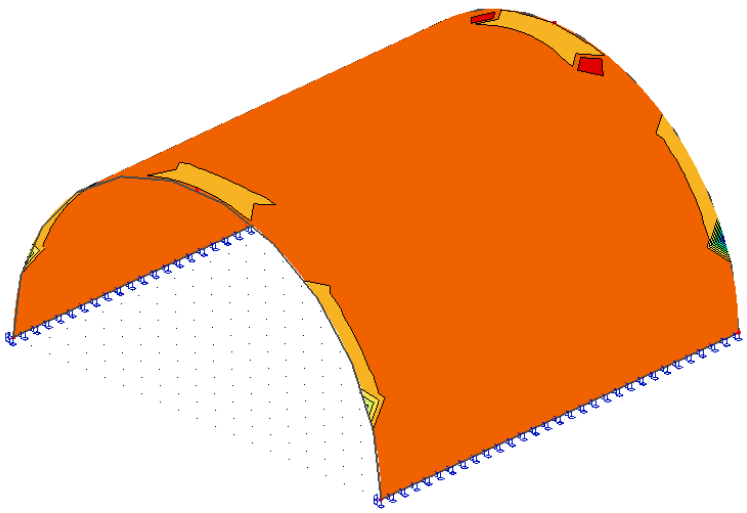
Knikformule v2
Waardes: **Lambda4**
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element



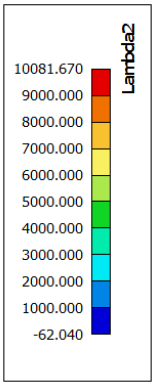
Halve cilinder

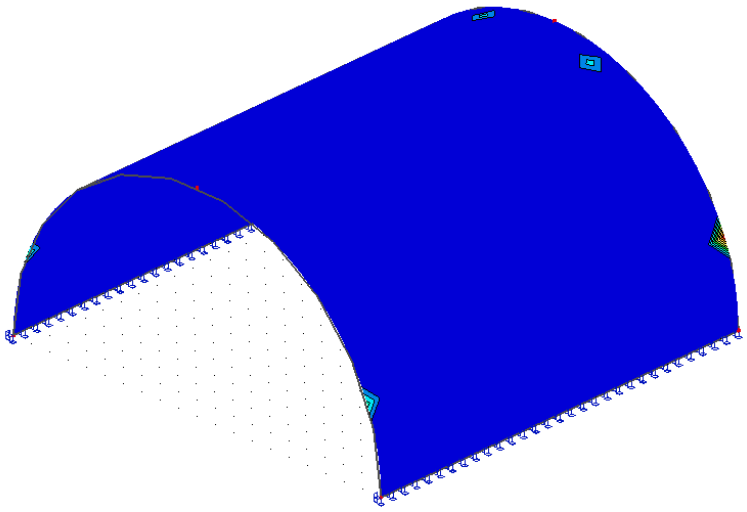
Knikformule v2
Waardes: **Lambda1**
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element



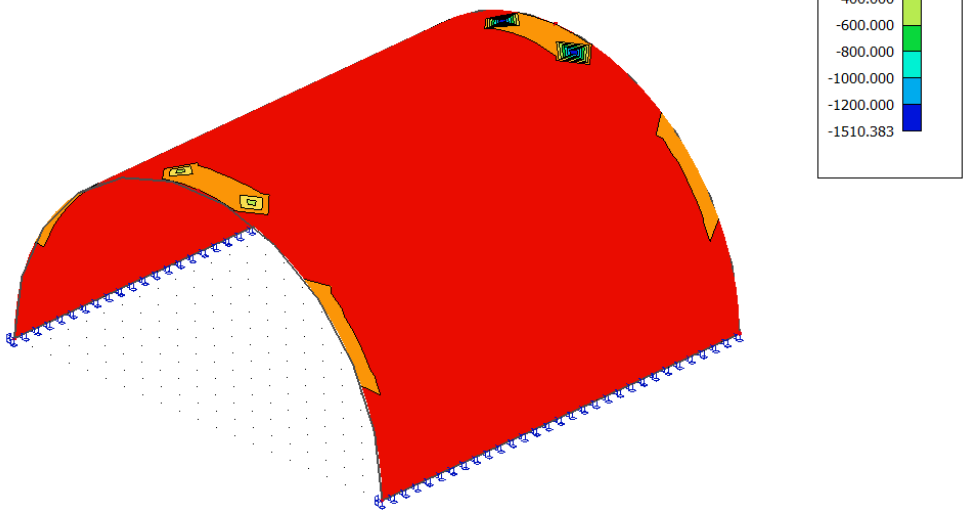


Knikformule v2
Waardes: **Lambda2**
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element

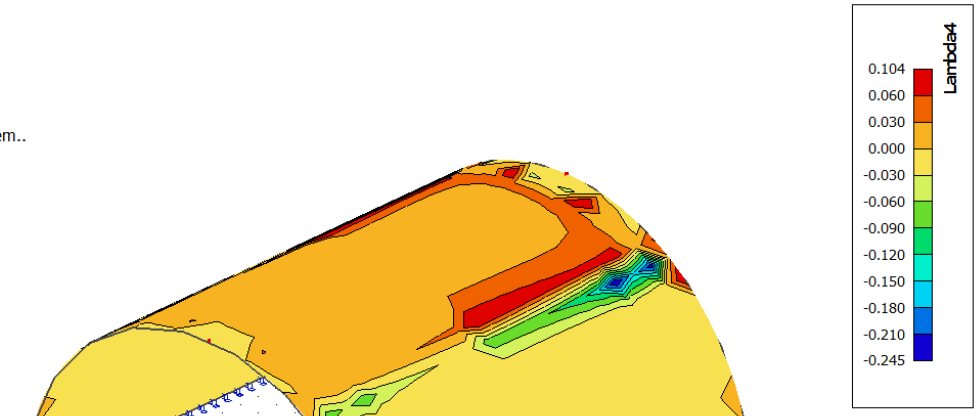




Knikformule v2
Waardes: **Lambda3**
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element



Knikformule v2
Waardes: **Lambda4**
Lineaire berekening
Belastingsgeval: BG1
Extreem: Globaal
Selectie: Alle
Locatie: In knooppunten gem..
Systeem: LCS net element



Literatuurlijst

Hartsuijker, C., & Welleman, H. (2018, januari). ConstructieMechanica 3. *Spanningsleer en bezwijkmodellen*.

Hoogenboom, P. (2016, mei 22). Design of a reinforced concrete 4-hypar shell with edge beams.

Hoogenboom, P. (sd). *CIE4143 Shell Analysis, Theory and Application.* Opgehaald van Dr P.C.J. Hoogenboom: http://homepage.tudelft.nl/p3r3s/b17_handout_1.pdf

Physlink. (sd). *What is a tensor?* Opgehaald van Physlink: http://www.physlink.com/education/askexperts/ae168.cfm

Skogh, J. (1979). *Buckled Shells.* Opgehaald van Shell Buckling: https://shellbuckling.com/buckledShells.php

Straalen, W. v. (2013). *Invarianten van tensoren - De onafhankelijkheid van assenstelsels rondom umbilics.* Delft.

Thorndike, A., Cooley, C., & Hye, J. (1978). *The structure and evolution of flow fields and other vector fields.* Journal of Physics A: Mathematical and General.

Vanden Berghe, G. (2007, december 12). *Numerieke Methodes in de Algebra.* Opgehaald van http://www.ilona.ugent.be/theorie/guido/H2.pdf

Vergeer, M. (2016). *Checks voor schaalconstructies met SCIA Engineer.* Dordrecht.