Kaleidoscopisch Projectwerk:
Toegepaste Wiskunde

# *Vormoptimalisatie in soft robotics*

Auteur: *Vladomeare Obolonskyy*

Begeleider: *Wim Vanroose*

Academiejaar 2023-2024

# 1 Inleiding

Soft robotics, een relatief nieuwe stroming in robotica die steeds meer aandacht naar zich toe trekt. Gebruik van soft robots breidt de mogelijkheden van klassieke robots uit om bijvoorbeeld met breekbare voorwerpen te werken. Dit project is geboren uit een video van 10 seconden op twitter waar een aantal blaasjes, bestuurd met 1 pomp, een melodie spelen op elektronische piano. Met dit mooie beeld in het achterhoofd zijn we begonnen met een simulatie van een soft robot bestaande uit 1 blaasje.

Oorspronkelijk werken we in 2D-ruimte om de simulatie te vereenvoudigen. Nadat ons project met een ballon van willekeurige beginvorm in de 2D-ruimte een door ons verwacht gedrag vertoont, gaan we over naar dezelfde probleemstelling in 3D-ruimte.

# 2 Theorie

## 2.1 Druk

We starten met het bekijken van theorie voor een continue geval, waarna we ons probleem gaan discritiseren zodat we een numerieke model kunnen maken.We starten met definitie van druk. (Cursus KU Leuven) De druk $p$ op een oppervlak wordt gedefinieerd als de grootte van de normaalkracht (de kracht loodrecht op het oppervlak) per eenheid van oppervlakte $A$:

$$p = \frac{F_\perp}{A}$$

Uit deze formule krijgen we een uitdrukking voor de kracht $F_\perp$ op de hele oppevlakte $A$ van onze model.

$$F_\perp = p \cdot A$$

Deze formule geldt in een 3D-ruimte voor ons model. Als we de situatie willen reduceren tot een 2D-ruimte moeten we de situatie in doorsnede bekijken. De formule wordt er verkregen door vervanging van oppervlakte $A$ naar de omtrek $S$:

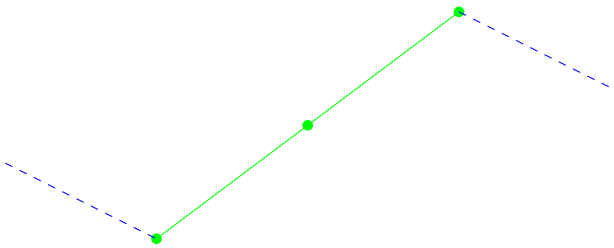
$$p = \frac{F_\perp}{S}$$

Net zoals hierboven krijgen we bij gevolg volgende formule voor de kracht $F_\perp$:

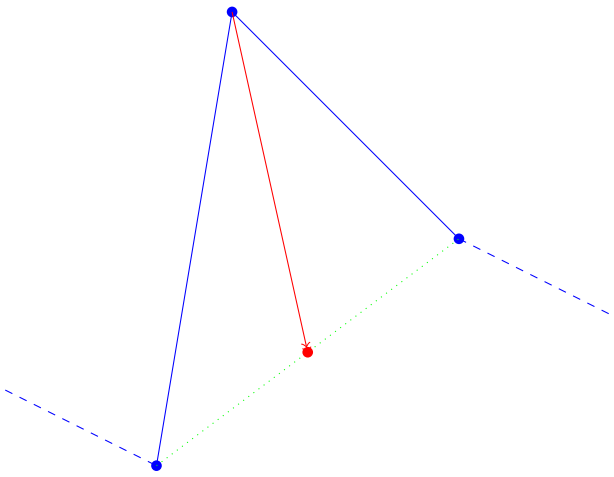
$$F_\perp = p \cdot S$$

## 2.2 Laplaciaan

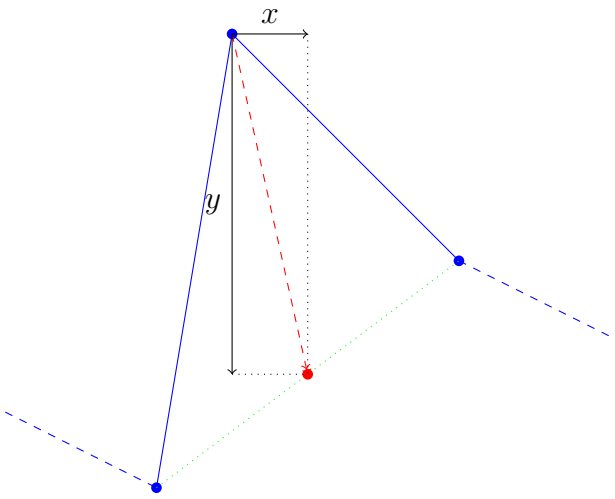
Wanneer we een oppervlak/omtrek discretiseren, willen we wel nog altijd een gladde oppervlak/omtrek krijgen. We kijken eerst naar fysisch voorstelling van dit probleem. Laat ons eerst eens kijken op een klein deel van omtrek die wordt voorgesteld door 3 punten. Hierbij kunnen we de verbinding tussen de punten benaderen als veertjes. Het is makkelijk in te zien dat dit 3-puntensysteem stabiel is wanneer dat deze 3 punten op een rechte liggen en wanneer de middelste punt zich op gelijke afstand bevindt van de 2 andere punten. [1]



Laat ons eens kijken naar het geval wanneer het 3-puntensysteem uit evenwicht wordt gebracht. Dan zal de kracht wijzen in de richting van evenwichtspunt.



Nu kunnen we deze kracht ontbinden in 2 componenten langs $x$ en $y$.




We kunnen dit probleem oplossen door een Laplaciaan te gebruiken. Een 1D-Laplaciaan neemt het verschil tussen de 2 punten en deelt die lengte door 2 waardoor we een punt ertussen vinden die juist in het midden ligt. Omdat we een systeem hebben die cyclisch gesloten is, moeten nog $\frac{1}{2}$ toevoegen in het rechter bovenhoek en linker onderhoek. Indien we een laplaciaan willen uitbreiden naar een hogere dimensie, moeten we een grotere matrix construeren die op meerdere 1D-Laplacianen op haar diagonaal bevat.

$$L_{1D} = \frac{1}{2}\begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{bmatrix}$$

Voor een 2D-geval krijgen we dan een Laplacian door op $x$- en $y$-component de $L_{1D}$ toe te passen. Als gevolg krijgen we volgende matrix.

$$L_{2D} = \begin{bmatrix} L_{1D} & 0 \\ 0 & L_{1D} \end{bmatrix}$$

Nu willen we de Laplaciaan uitbreiden naar een 3D-geval. Zoals we al hadden gezien stelt een Laplaciaan het gemiddelde voor tussen al de buren. Neem daarom $k_i$ voor het aantal buren voor een $i$-de punt. We krijgen dan volgende uitdrukking voor elke rij $i$.

$$L_{1D}(i,j) = \begin{cases} -1 & i = j \\ \frac{1}{k_i} & \text{als } j\text{-de punt een buur is} \\ 0 & \text{anders} \end{cases}$$

Omdat $x$, $y$ en $z$-componenten niet van elkaar afhangen, kunnen we alweer een diagonaalmatrix opstellen van volgende vorm.

$$L_{3D} = \begin{bmatrix} L_{1D} & 0 & 0 \\ 0 & L_{1D} & 0 \\ 0 & 0 & L_{1D} \end{bmatrix}$$

## 2.3 Krachtenanalyse

De oorzakken van verschillende krachten in een systeem kunnen verschillend zijn. Hierboven hadden we een kracht afgeleid veroorzaakt door druk en een kracht veroorzaakt door elasticiteit van het materiaal. We gaan bij onze model deze 2 krachten gebruiken en proberen hierbij om de toestand van deze systeem te balanceren. Elke kracht wordt voorgesteld door een vector. De vectoren worden opgeteld om een resulterende vector te vinden voor elk punt. Om het aantal vectoren te verminderen, kunnen we in plaats van 2 krachtvectoren die door druk worden veroorzaakt, $1^{\text{ste}}$ door binnendruk en $2^{\text{de}}$ door buitendruk, de relatieve druk gebruiken door het drukverschil te nemen.

# 3 2D-versie

## 3.1 Stappenplan

Hier worden de te ondernemen stappen beschreven die tot gewild resultaat zouden moeten leiden. Zo is er op basis van theorie een stappenplan uitgewerkt die als kern van ons programma wordt gebruikt. Hierbij gaan we volgende aannames maken: De punten die figuur vormen zijn geordend, voor waarden worden SI-eenheden gebruikt.

### 3.1.1 Smoothing

Zoals in theorie beschreven gebruiken we een Laplaciaan om onze figuur te vergladden. Zo zou elk punt tot haar stabiele toestand proberen te gaan. Hierdoor zal elk figuur zonder extra voorwaarden convergeren naar een cirkel. Indien er geen kracht van binnen naar buiten aanwezig gaat zijn, is het makkelijk in te zien dat alle punten zich naar een uniek punt gaan verplaasten.

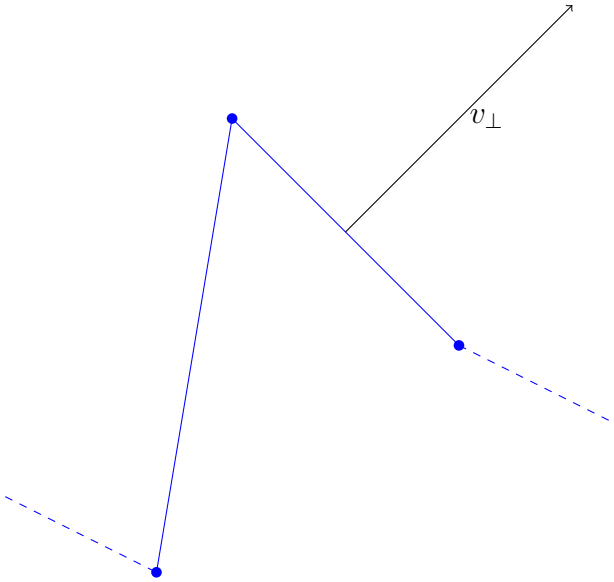
### 3.1.2 Krachten uitwerking

In ons model zijn er 2 krachten die op elk punt inwerken. Eerste kracht wordt veroorzaakt door de omhulsel van ons model die naar de stabiele toestand streeft. Deze kracht hebben we in voorgaande stuk besproken. Tweede kracht wordt veroorzaakt door de resulterende (relatieve) druk. De som van deze 2 vectoren geeft ons na berekening de resulterende krachtvector.

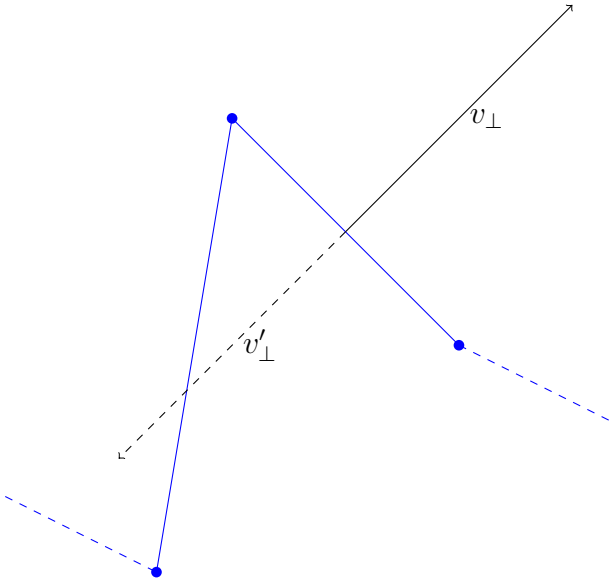
Laat ons eens kijken naar de vector veroorzaakt door de druk. Hiervoor berekenen we eerst de richting van de kracht. Daarna berekenen we de grootte en de zin ervan aan de hand van de druk. Voor elke berekening van kracht gebruiken we telkens 3 opeenvolgende punten.

Om te beginnen moeten we hier een orthogonale vector definieren op een verzameling punten. Hierbij merken we dat een verzameling punten geen raaklijn of raakvector kan hebben. Om dit probleem op te lossen, gaan we het begrip virtuele rand toevoegen en daarop een orthogonale vector definieren, die we erna gaan verdelen over onze randpunten. Deze benaderd de ontbrekende informatie over ons figuur door de kortste weg tussen de naburige punten.

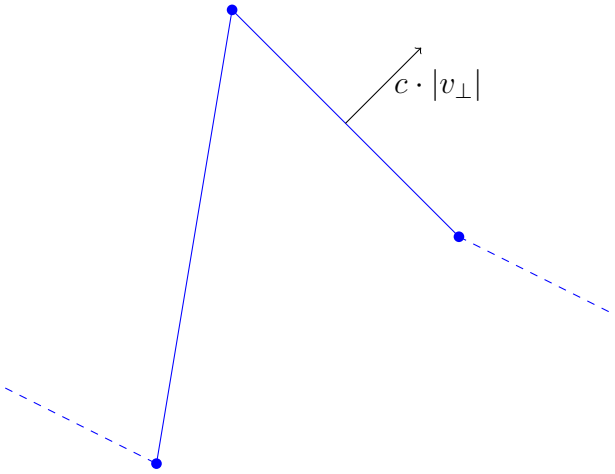
We beginnen met zoeken van de normaal. Hiervoor nemen we voor elke 2 opeenvolgende punten en gaan een orthogonale vector erop zoeken. Omdat we delingen zo veel mogelijk willen vermijden, gaan we de eerstkomende in gedachte manier met richtingscoeffiecienten negeren. Volgende manier die ons zeer snel een orthogonale vector is het volgende. Neem een vector tussen de naburige punten. Dit kan snel worden gedaan door verschil te nemen van de twee punten. Zo krijgen we een vector $\vec{v} = (x_{i-1} - x_{i+1}, y_{i-1} - y_{i+1})$ en door volgende bewerking krijgen we een orthogonale vector $\vec{v}_\perp = (y_{i-1} - y_{i+1}, -(x_{i-1} - x_{i+1}))$ of $\vec{v}_\perp = (-(y_{i-1} - y_{i+1}), x_{i-1} - x_{i+1})$.



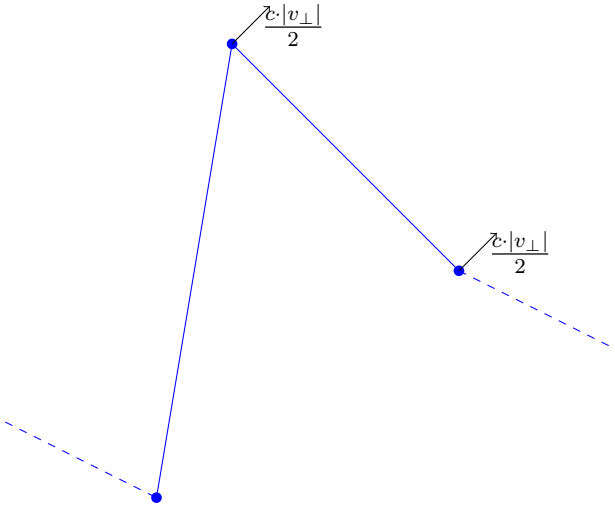

Zo zijn we nu aangekomen aan de zin van onze orthogonale vector. De zin van onze vector hangt enkel af van de relatieve druk. Hiervoor voeren we een coefficient in, die afhankelijk is van de orientatie van onze puntenverzameling en de keuze van formule voor normaalvector. Deze coefficient kan 1 of -1 zijn.



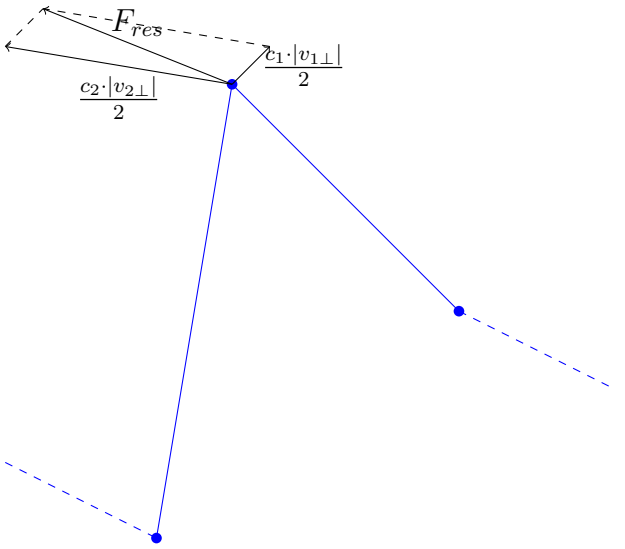

De moeilijkste om te bepalen is de grootte van onze normaalvector in elk punt. Hiervoor gaan we formule $F_\perp = p \cdot S = c \cdot |v_\perp|$ gebruiken om deze grootte te benaderen met $S$ de lengte van rechte tussen de 2 opeenvolgende punten. Deze geschaalde kracht wordt dan gegeven door een vector die loodrecht staat op ons benaderende vlak.




Nu verdelen we deze kracht over de 2 punten omdat we een discreet probleem hebben, want we gaan ervan uit dat de druk in alle richtingen gelijk is. We krijgen dan het volgende resultaat.



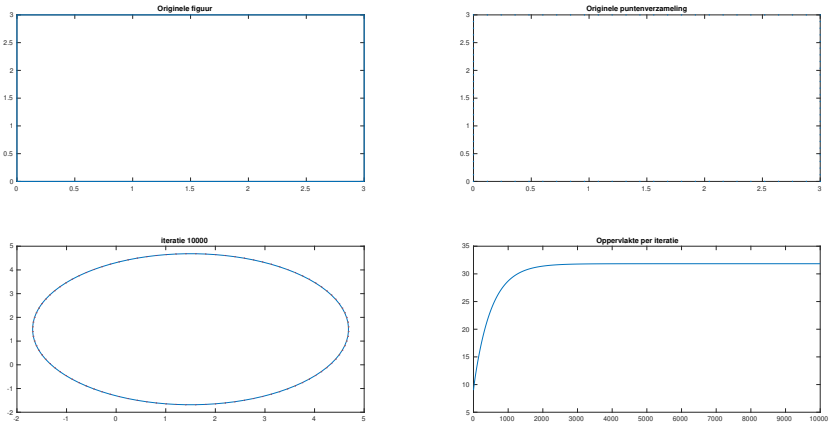

Als we zelfde berekening doorvoeren voor het volgende lijnstuk, krijgen we een $2^{de}$ krachtvector die op ons punt inwerkt. Daarna kunnen we deze 2 vectoren sommeren om resulterende krachtvector te krijgen.




Deze berekeningen voeren we voor elk punt uit. Hierna kunnen al deze vectoren worden gesommeerd met hun overeenkomstige laplaciaan om de iteratie af te sluiten.

## 3.2 Verloop

We beschouwen een uitgewerkt voorbeeld met onze code waar we beginen met een vierkant.




We kunnen als eerste opmerken dat ons vierkant omgevormd is tot een cirkel, zoals we het in theorie hadden voorspeld. Het tweede wat we kunnen opmerken is dat de oppervlate op een gegeven moment zich stabiliseert en niet meer veranderd. Dit betekent dat we een stabiele oplossing hebben

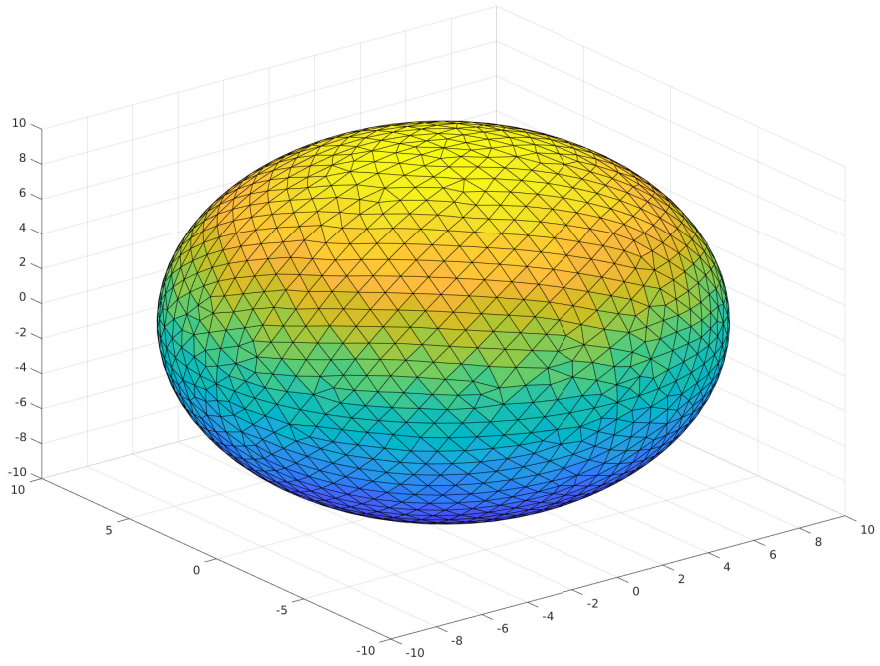
verkregen. Hieruit volgt dat we een stopcriterium kunnen toevoegen om niet meer berekeningen te doen dan nodig.

# 4 3D-versie

Nu kunnen we een stap dichter zetten bij onze poging om een zeer simpele soft robot te simuleren. In 3D-geval moeten we een aantal zaken veralgemenen. Zo hadden we al in theoretische gedeelte onze Laplaciaan veralgemeend naar een 3D-geval.

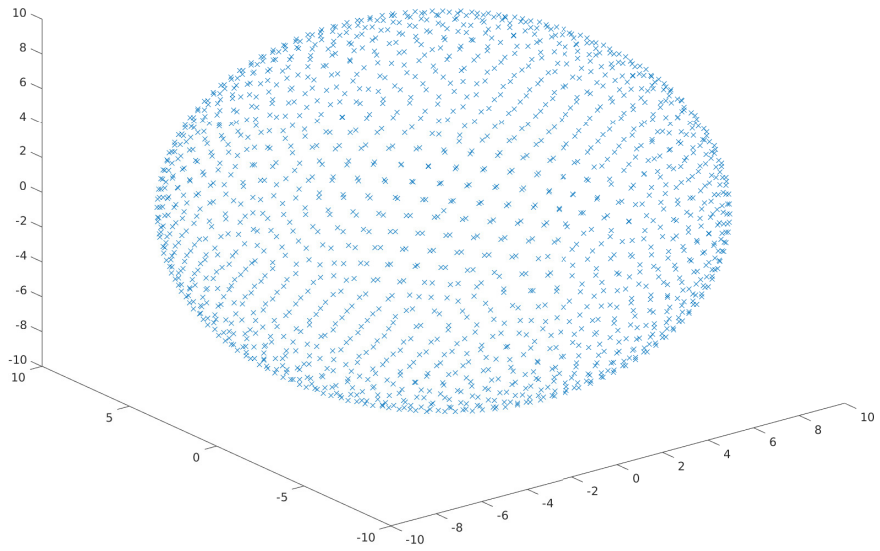
## 4.1 Mesh

Om een 3D-figuur te beschrijven, wordt er een structuur genaamd mesh gebruikt. Onze mesh van een bol en een kubus werden opgesteld in Gmsh opgemaakt. Omdat mesh niet een uniek formaat heeft, waardoor die niet kan worden doorgegeven aan ingebouwde functies in matlab, zijn al onze functies opgebouwd om matrices of argumenten die onderdelen zijn van onze mesh als input te gebruiken. Zo ziet een mesh uit als we die gaan plotten.




Deze mesh wordt beschreven door een verzameling punten, die we als basis gaan gebruiken om onze figuur te vervormen. Deze punten komen overeen

met de hoekpunten op de bovenstaande plot.




In de mesh structuur is de volgende data relevant voor onze berekeningen, naast de coordinaten van onze verzameling punten: het aantal punten dat onze figuur beschrijft, lijnstukken die verschillende punten verbinden en driehoeken die onze figuur beschrijven.

Bij het opstellen van een mesh is het belangerijk om te letten op volgorde van punten in de driehoeken. De volgorde bepaalt wat de orientatie van de driehoek in de ruimte is m.a.w waar de inproduct naartoe wijst. Hierdoor ontstaat er een binnenkant en een buitenkant van de figuur. Conventioneel is het om de volgorde van punten tegen de wijzer in te nemen. Een slecht opgestelde mesh waar geen rekening werd gehouden met volgorde van punten in de driehoek kan een oorzaak zijn van onvoorspelbare gedrag van het figuur in de tijd.

## 4.2 Algoritme

Hier beschrijven we kort het basisalgoritme van ons programma. In ons 2D-geval hadden we een geordende verzameling, waardoor we direct wisten wat de buren waren van een opgegeven punt. Nu hebben we extra informatie nodig om de buren te bepalen van elk punt. Hiervoor maken we gebruik

van de driehoeken en lijnstukken uit de mesh. Voor elke driehoek of lijnstuk wordt er nagegaan welke punten erin zitten en wordt het verband van punt en haar buren in een aparte vector opgeslagen. Op het einde kunnen we voor elk punt meermaals voorkomende buren verwijderen. Hierdoor kunnen we voor elk punt het aantal buren ook tellen.

We weten al dat Laplaciaan gebruik maakt van het aantal buren van elk punt. Na de voorgaande berekening kunnen we een 1D-Laplaciaan opstellen, om die dan 3 keer op de diagonaal te plaatsen om een 3D-Laplaciaan te verkrijgen. De coordinaten van de puntenverzameling worden ook in een kolomvector geplaatst van de vorm $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$.

Nu kunnen we beginnen aan ons iteratieve gedeelte in ons algoritme. We beginnen met het zoeken van een punt binnen in onze figuur. We gaan ervan uit dat we met convexe lichamen werken, daarom volstaat het om de massamiddelpunt te vinden. We hebben deze massamiddelpunt nodig om het volume te berekenen van ons figuur. Hierbij sommeren we gewoon het volume van alle tetraeders die gevormd worden door onze driehoeken en de massamiddelpunt.

De oppervlakte van een figuur berekenen wordt snel gedaan door de som van oppervlaktes van alle driehoeken. Oppervlakte van elke driehoek apart kan via volgende formule worden gevonden:

$$A_{\triangle} = \frac{\det\left(\begin{bmatrix} v_1 \\ v_2 \\ v_1 \times v_2 \end{bmatrix}\right)}{2 \cdot \|v_1 \times v_2\|_2}$$

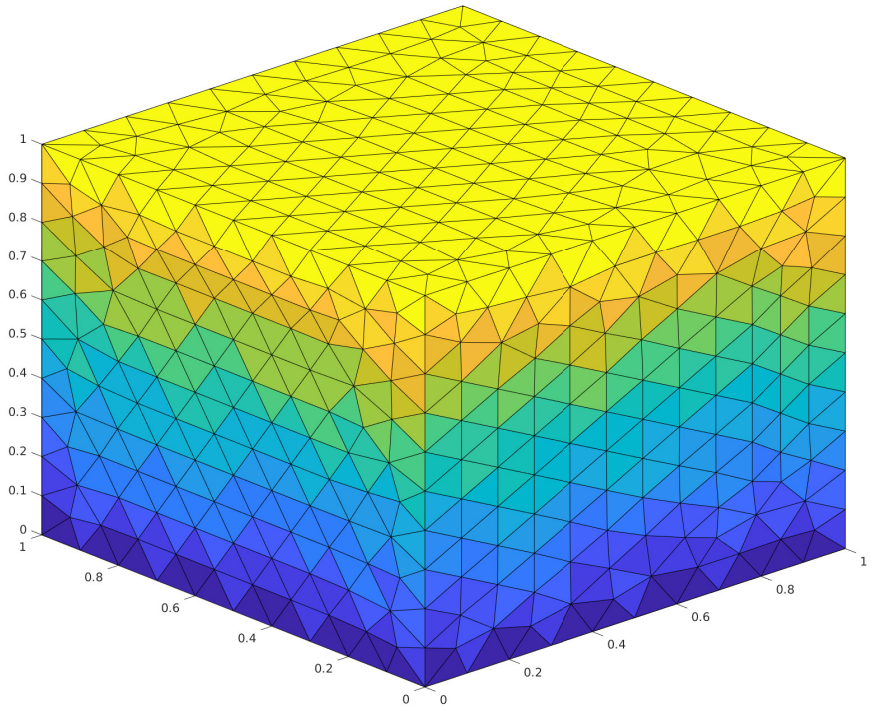
Het principe voor het vinden van een krachtvector op elk punt verschilt weinig van 3D-geval. Het enige verschil is dat elke geschaalde orthogonale vector op een driehoek over 3 punten wordt verdeelt in plaats van 2. Hierdoor zal elke krachtvector in elk hoekpunt van de driehoek een derde bedragen van de totale kracht op het oppervlak. Het aantal vectoren dat er moet worden gesommeerd hangt af van het aantal buren van het punt. Op het einde sommeren we al deze vectoren met hun overeenkomstige vectoren gecreerd door 3D-Laplaciaan.

## 4.3 Verloop

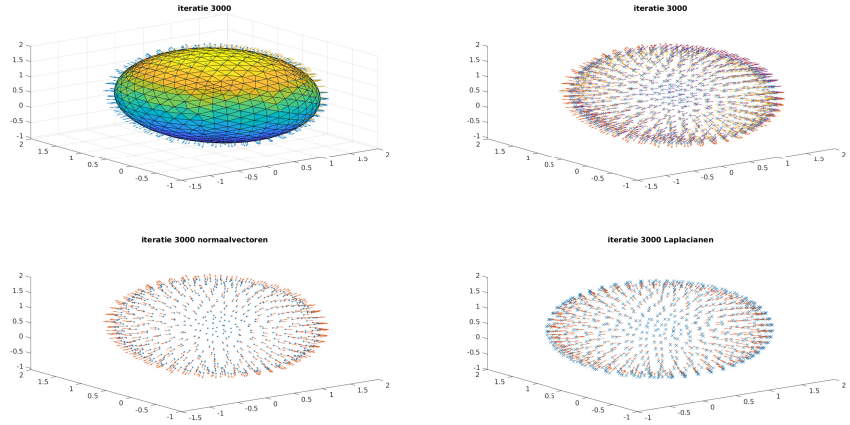
Laat ons terug een testprobleem beschouwen. Deze keer nemen we een kubus in plaats van een vierkant. Uit onze voorgaande testprobleem, kunnen we

veronderstellen dat onze kubus de vorm van een bol moet aannemen en naar een vaste volume convergeren (omdat er geen druksverandering is).

Zo ziet onze kubus uit op iteratie 0.

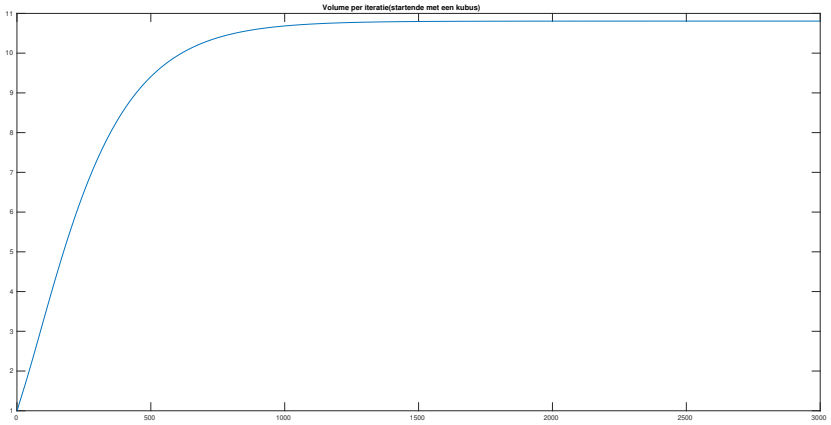


Als we onze berekeningen erop toepassen, dan krijgen we volgende resultaten na 3000 iteraties.




We zien dat we uiteindelijk een bol krijgen, zoals we het ervoor had-

den voorspeld. Nu interesseert ons de convergentie van onze bol naar een gefixeerd volume. Hieronder is de volume gegeven per iteratie.




We kunnen weer zien dat deze waarden convergeren en omdat er geen drukverandering is in de tijd, veranderd het gefixeerde volume ook niet. Dus ook hier kunnen we een stopcriterium plaatsen om geen overbodige berekeningen te maken, wetende dat de oplossing stabiel is.

# 5 Uitbreidingen

Met deze uitwerking eindigt onze code. We hebben tijdens dit project geleerd hoe we een fysisch model kunnen versimpelen en discretiseren, zodat we het kunnen programmeren. Daarnaast hebben we gezien hoe mesh's in elkaar zitten en hoe structuren kunnen worden toegepast in code.

Laat ons nu kijken welke mogelijke uitbreidingen er bestaan. Om te beginnen, kan ons programma worden omgezet naar een functiefile. Zo zou er een systeem van soft robots gecreerd worden. Daarnaast kan er naar gekeken worden om randvoorwaarden toe te voegen aan onze probleemstelling om interactie met andere objecten te simuleren. De code kan worden geraadpleegd via github of een persoonlijke website.

# 6 Bijlagen

- https://github.com/vladomeare/soft_robotics_simulation
- http://www.vladomeare.com/projects/bachelorproef_soft_robotics

# Referenties

[1] Roi Poranne. Shape modeling and geometry processing: Smoothing (continued) deformations.