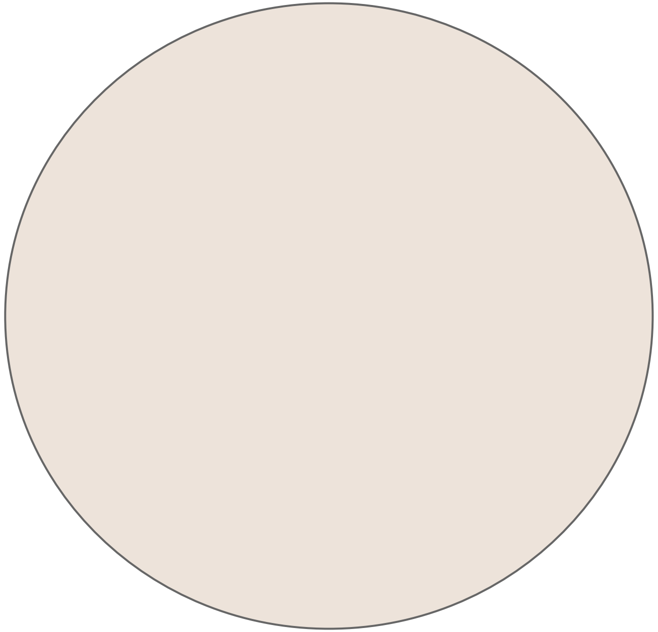
Bilayer Shell Simulation
Final Presentation

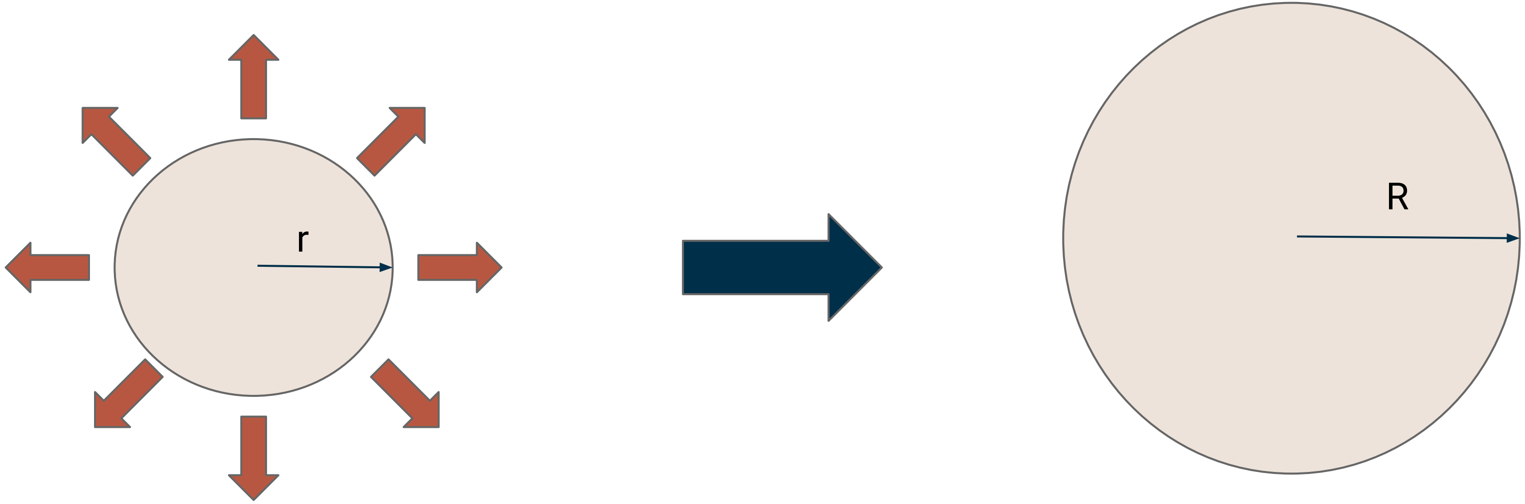
Vishal Kackar, Yunbo Wang, Shyan Shokrzadeh



Substrate Layer

- Circular shape
    - Initial radius = r
- Has the lower Young's Modulus of the two materials
    - E = 116,250 Pa
- Will be stretched
    - Stretched to R
- Sticky

Overview of the system we are trying to simulate - substrate

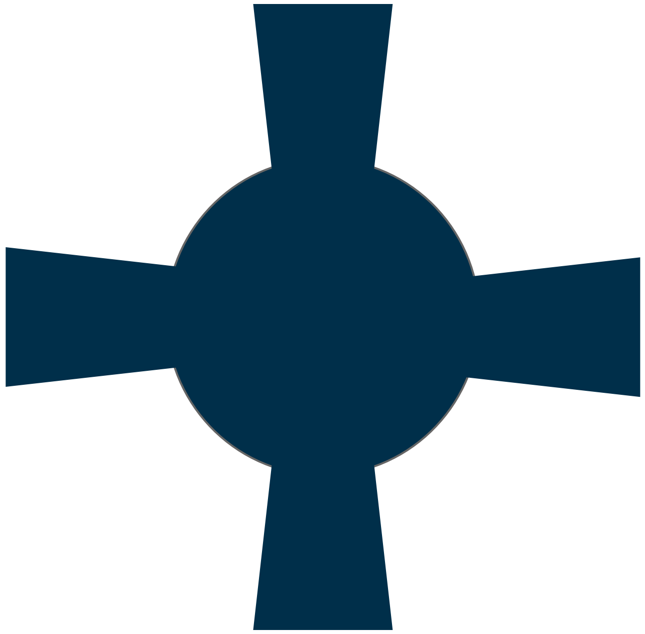


The substrate layer will be stretched radially. We can quantify this stretch using:

$$\lambda = \frac{\text{New radius}}{\text{Original radius}} = \frac{R}{r}$$

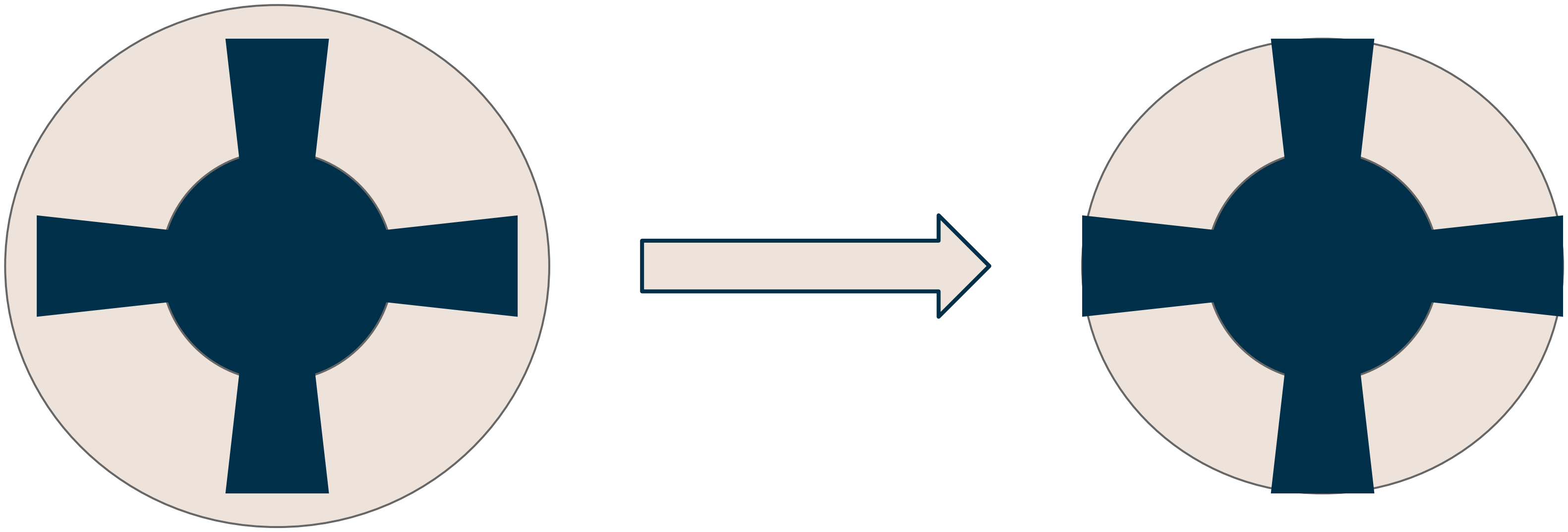
A λ value of 1 corresponds to no stretch
(Similar to strain)

Overview of the system we are trying to simulate - substrate



Kirigami Layer

- Has a special shape determined by kirigami principles
  - Similar to origami
  - Has radius = r
- Has the higher Young's Modulus of the two materials
  - E = 368,988 Pa
- Will be bending
  - Not stretched
- Also sticky

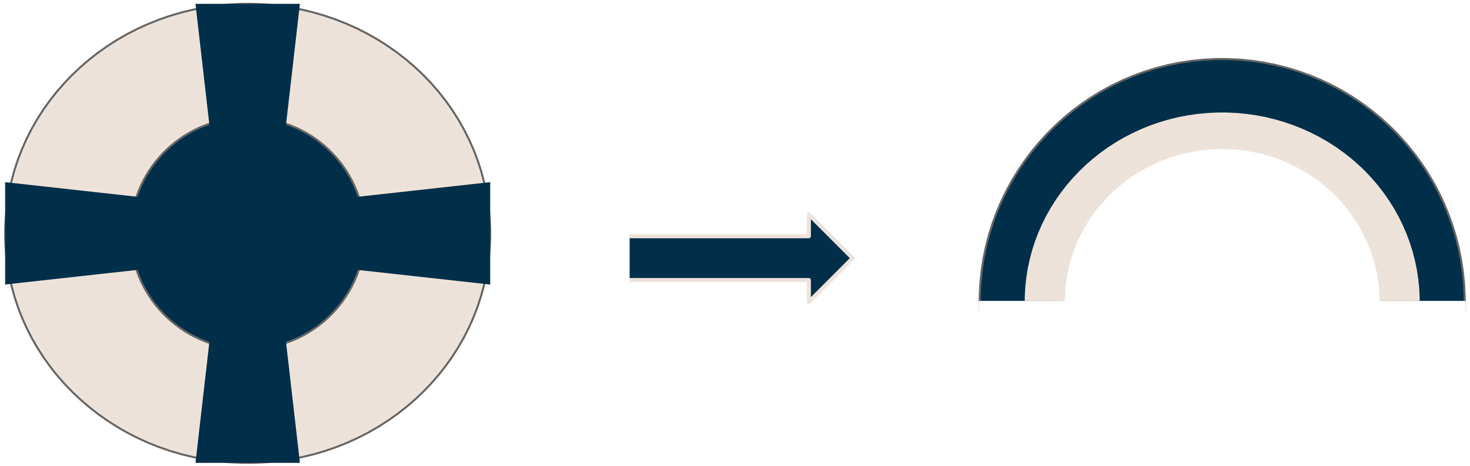


1) Put kirigami on top of stretched substrate layer

2) Cut off excess substrate layer

3) Final 2D shape

Overview of the system we are trying to simulate – creating the shell



- Previous research has found that this specific kirigami shape will yield a hemispherical shape with the appropriate amount of lambda
    - (Analog to origami, but we are cutting instead of folding)

- The shell starts to take its 3D shape as soon as the substrate layer is released

- Caused by a mismatch in stretch between the two layers. The substrate wants to shrink, but the kirigami doesn’t want to change its shape

Overview of the system we are trying to simulate

Simulation

- The goal of the simulation is to predict what shape the shell will take depending on the parameters we give it
    - Ex: Young's Moduli, thickness, lambda
- Compare the results of the simulation against FEA results

Simplifying Assumptions

- The substrate only region doesn't have a natural curvature
- Thickness is constant
    - Could use poisson's ratio + λ to find the changing thickness
- All of the stretching energy goes into the bending energy

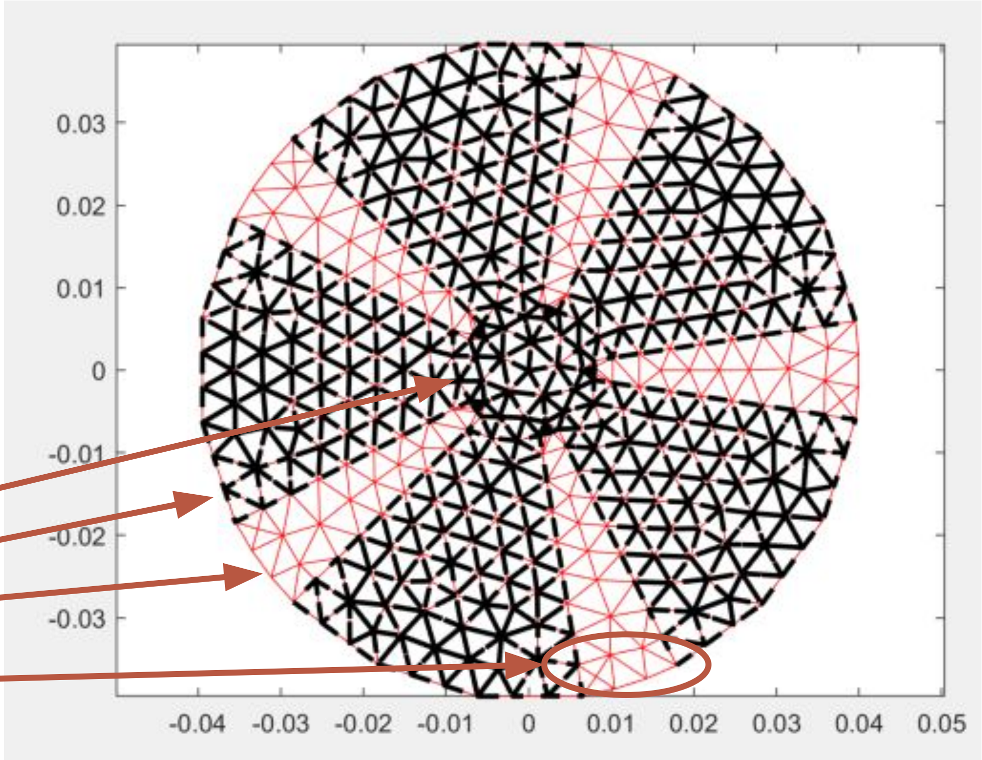
Initial Conditions

We first want to create a mesh in MATLAB and note which sections are just the substrate and which sections are the substrate and kirigami.

Relevant Parameters

- Inner radius (= 0.008)
- Outer radius (= 0.04)
- Number of strips (= 5)
- dTheta (= 0.3)

We feed these parameters into a function and get the following mesh



The point closest to (0,0,0) is fixed

Extracting Relevant Vectors from the Mesh

- The mesh function gives us the nodes, edges, and kirigami bending element indices.

- We need to get the unique edges, substrate elements, and combined elements
  - Unique edges and bending elements are found from the code the professor gave us

```
%% create the flag vector for kirigami regions
flagVec = flagKirigami(cutOutElements, ElementToNode, bendingElements);
```

- Create a flag vector that is as long as the bending elements matrix

- Compare the elements in there to the elements described by cutOutElements

- 1 = combined elements (has a natural curvature)
- 0 = only substrate (no natural curvature)

Initial Conditions

- Set up the quantities we usually set up for shell/plate simulations
- Also need to compute the I of the combined section
    - Done using composite beam theory

```
rho = 1000; %density

Yk = 368988; %kirigami
Ys = 1.1625e5;  %substrate
tk = 0.001;
ts = 0.001;

lambda = 1.24;

%combined section moment of inertia
It = CompositeBeamI(Yk,Ys,tk,ts);

kb = zeros(2,1);
kb(1) = 2/sqrt(3) * Ys * ts^3 / 12; % bending stiffness of substrate
kb(2) = 2/sqrt(3) * Ys * It; % bending stiffness of special region

%simulation time
t0 = 0;
dt = 0.001;
tf = 0.5;
steps = (tf-t0)/dt+1;

bc_time = 2; % have l_bar reach l_inf after 2 seconds

maxIter = 100;
tol = 1e-6;

visc = 1;
```

Initial Conditions

We give the simulation the relevant material properties and the amount of pre-stretch applied to the substrate layer.

We then calculate the total energy of the system using the stretching energy of the substrate layer before the pre-stretch is released.

The initial stretch we put into the system is “captured” when we put the kirigami on top.

$$\text{Energy Initial} = \tfrac{1}{2} * E_s * A_s * (\lambda-1)^2 * L$$

Where: $A_s = w * t_s$

L = reference length

Reference Length

- Since the simulation starts after the substrate has been stretched, we need to find the original length of the substrate when calculating the reference length (L_{inf}).
- However, immediately setting reference length to L_{inf} would make the simulation fail to converge, so we need to slowly step down our reference length.
- Start with the length at t = 0 (L_0) and step down to the length at t = inf (L_{inf})
- We then set a boundary condition time (bc_time) that we will use to step from L_0 down to L_{inf}

$$L_{inf} = \frac{L_0}{\lambda}$$

* Apply this to all of the edges

```
if(dt*(i-1) < bc_time)
    lk = lk_0 - (lk_0 - lk_inf)*(dt*(i-1)/bc_time);

else
    lk = lk_inf;
end
```

Bending Energy

- We need to calculate the natural curvature of the system before we start the simulation
- Use the bending energy formula from class in order to solve for **theta bar**

$$\text{Bending Energy} = \tfrac{1}{2} * k_b * \theta^2$$

$$k_b = 2/\sqrt{(3)} * E * I_{eff}$$

```
kb = zeros(2,1);
kb(1) = 2/sqrt(3) * Ys * ts^3 / 12; % bending stiffness of substrate
kb(2) = 2/sqrt(3) * Ys * It; % bending stiffness of special region
```

From Stretching to Bending

- As mentioned earlier, we are assuming that all of the stretching energy we apply goes into the bending energy
- Thus, we can equate the stretching and bending energies in order to solve for **theta bar**

- **Since reference length is a function of time, so is theta bar**

```
%calculate the max theta bar once, step through it later
thetaBarInf = findTBarInfinity(bendingElements, Nodes);
```

Similar to L_{inf}, we first find thetaBar_{inf}

```
theta_bar = computeThetaBar(x0, x1, x2, x3, flagVec(h), c_time, h);
```

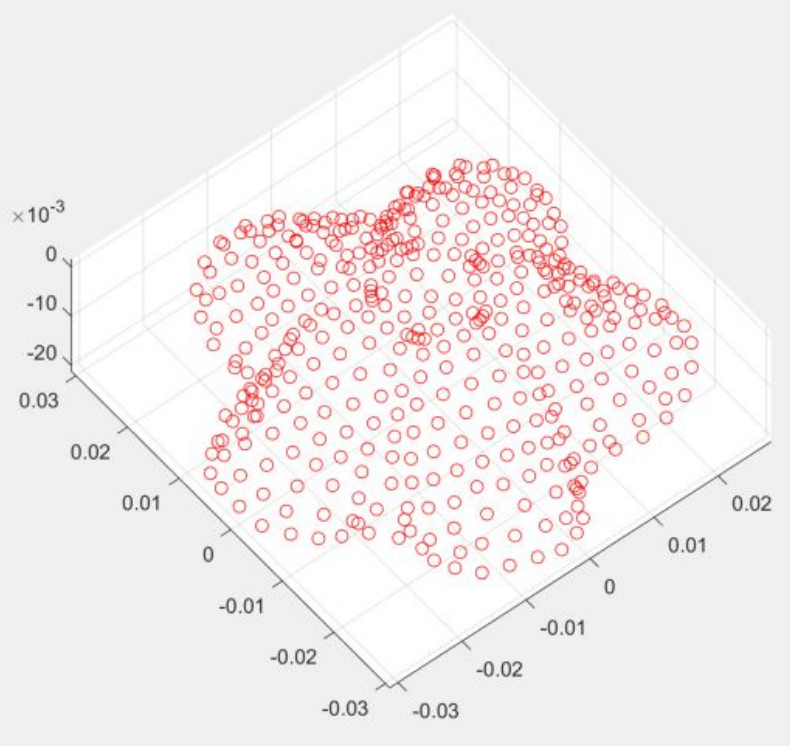
computeThetaBar finds the theta bar for the current moment in time because our reference length is changing. It also fixes the sign of theta bar

```
ang = getTheta(x0, x1, x2, x3);

if ang < 0
    thetaBarGlobal = -thetaBarGlobal;
end


% need to also slowly step theta
if c_time <= bc_time
    thetaBarGlobal = thetaBarGlobal * c_time/bc_time;
end

thetaBar = thetaBarGlobal * flag;
```

To sum it up...

- Given a stretch amount, λ, and the relevant material properties we can calculate initial stretching energy, L_{inf}, L_0, k_b, and $theta_{inf}$
- We extract the relevant matrices and vectors from the mesh and make sure to note which bending elements are in the combined region
- We start iterating through time
- Slowly decrease L from L_0 to L_{inf}
- Same with increasing theta from $theta_0$ to $theta_{inf}$

Results

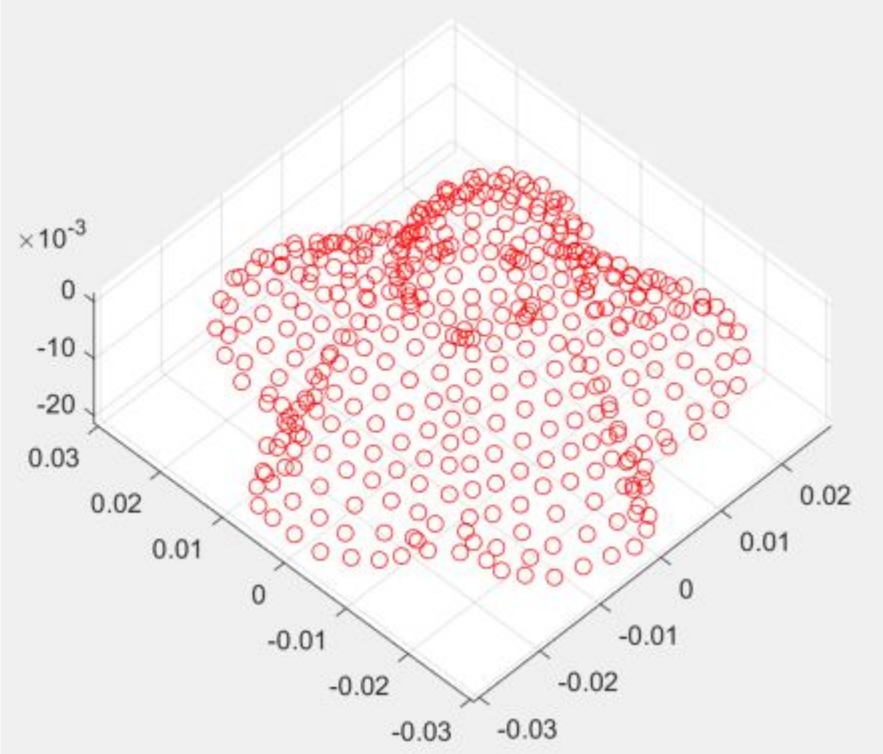


λ = 1.09

Run for 2 seconds

Needed about 20 iterations to reduce error down to acceptable amount

Results

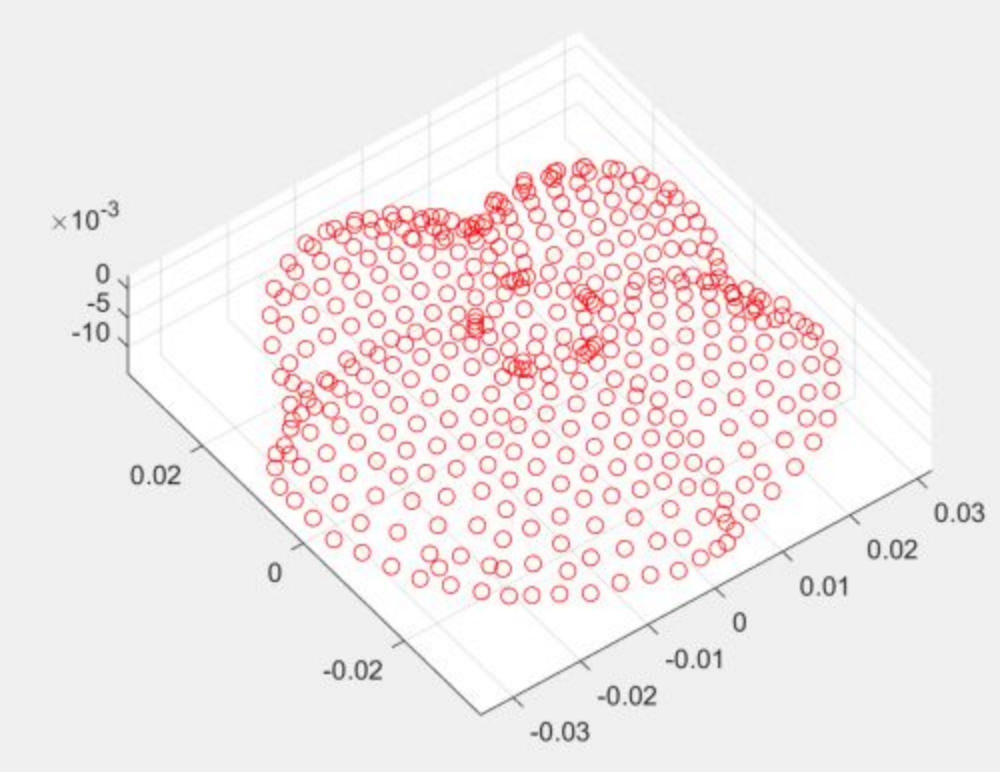


$\lambda = 1.12$

Run for 1.75 seconds

Needed about 40 iterations to reduce error down to acceptable amount

Results



λ = 1.12

Run for 0.7 seconds

Needed about 30 iterations to reduce error down to acceptable amount

Would fail to converge if run for much longer

Interpreting the Results

- All of the λ values gave the same shape
- What changed was how long it took to get to that shape
  - Lower λ = longer simulation time
  - Higher λ = shorter simulation time
- The amount of error was also influenced by the λ value
  - Lower λ = lower error throughout the simulation
    - Only need 3 or 4 iterations initially
    - Number of iterations grows slowly
  - Higher λ = higher error throughout the simulation
    - Need 4 or 5 iterations initially
    - Number of iterations grows quickly

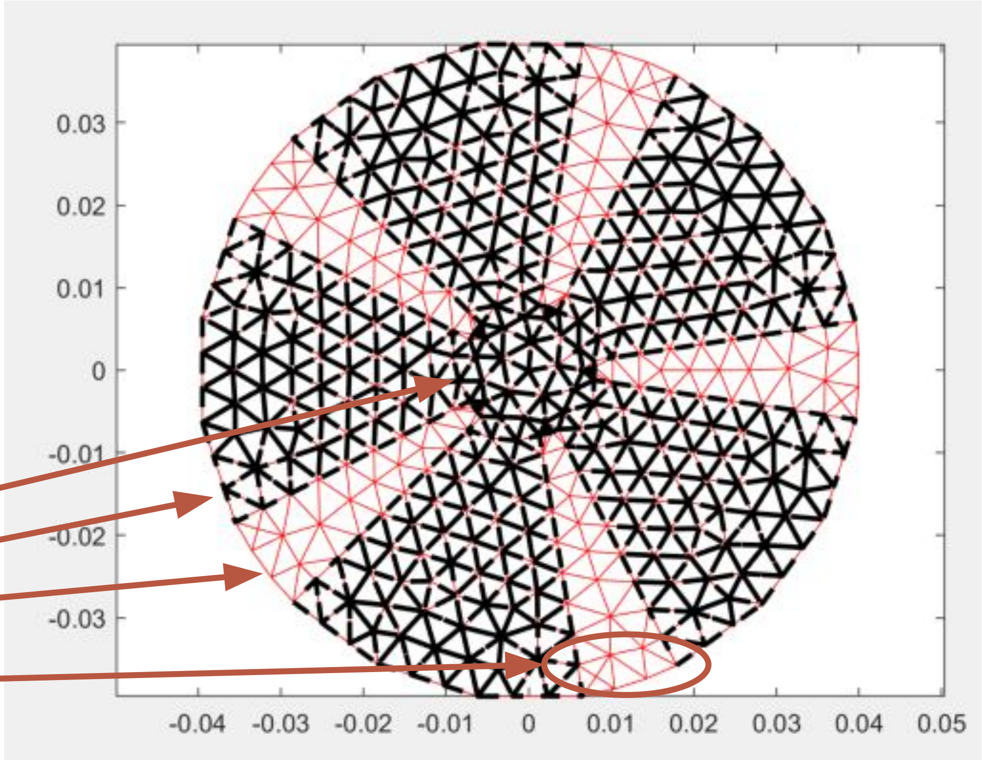
Data Comparison

We first want to create a mesh in MATLAB and note which sections are just the substrate and which sections are the substrate and kirigami.

Relevant Parameters

- Inner radius (= 0.008)
- Outer radius (= 0.04)
- Number of strips (= 5)
- dTheta (= 0.3)

We feed these parameters into a function and get the following mesh



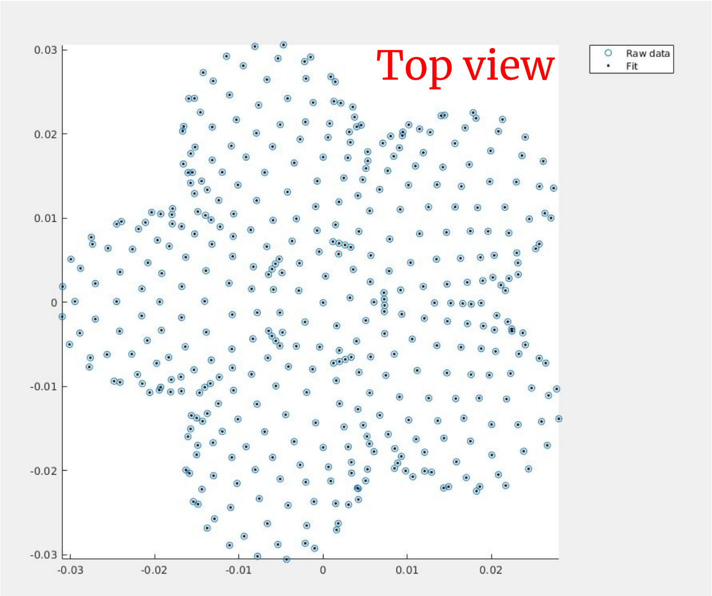
The point closest to (0,0,0) is fixed

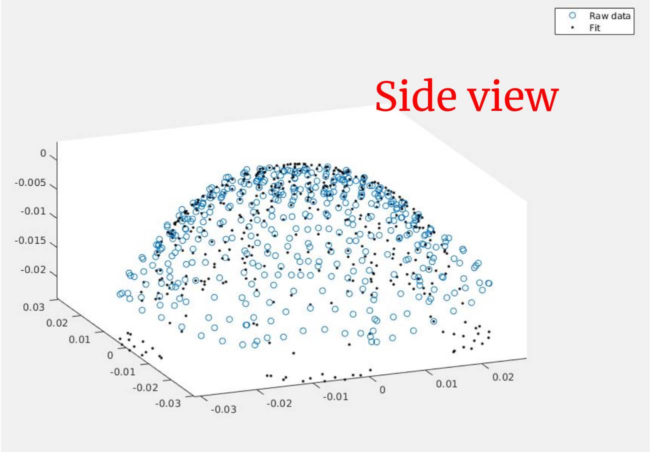
Hemisphere Shape

Quality (Dimensionless quantity):

R
X_range/2R
Y_range/2R
Z_range/2R

The more close to 0,
The better the result.

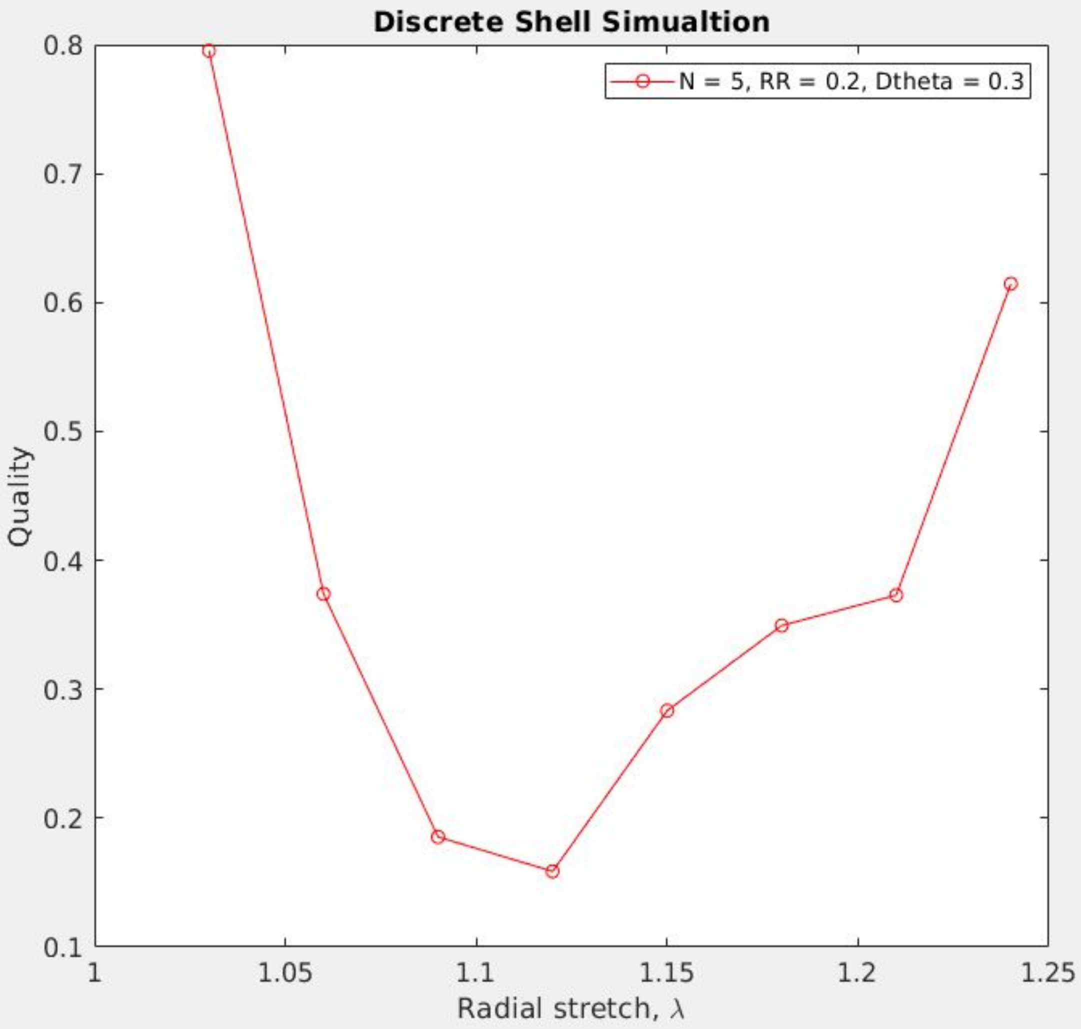




Data from Vishal

The Best result :

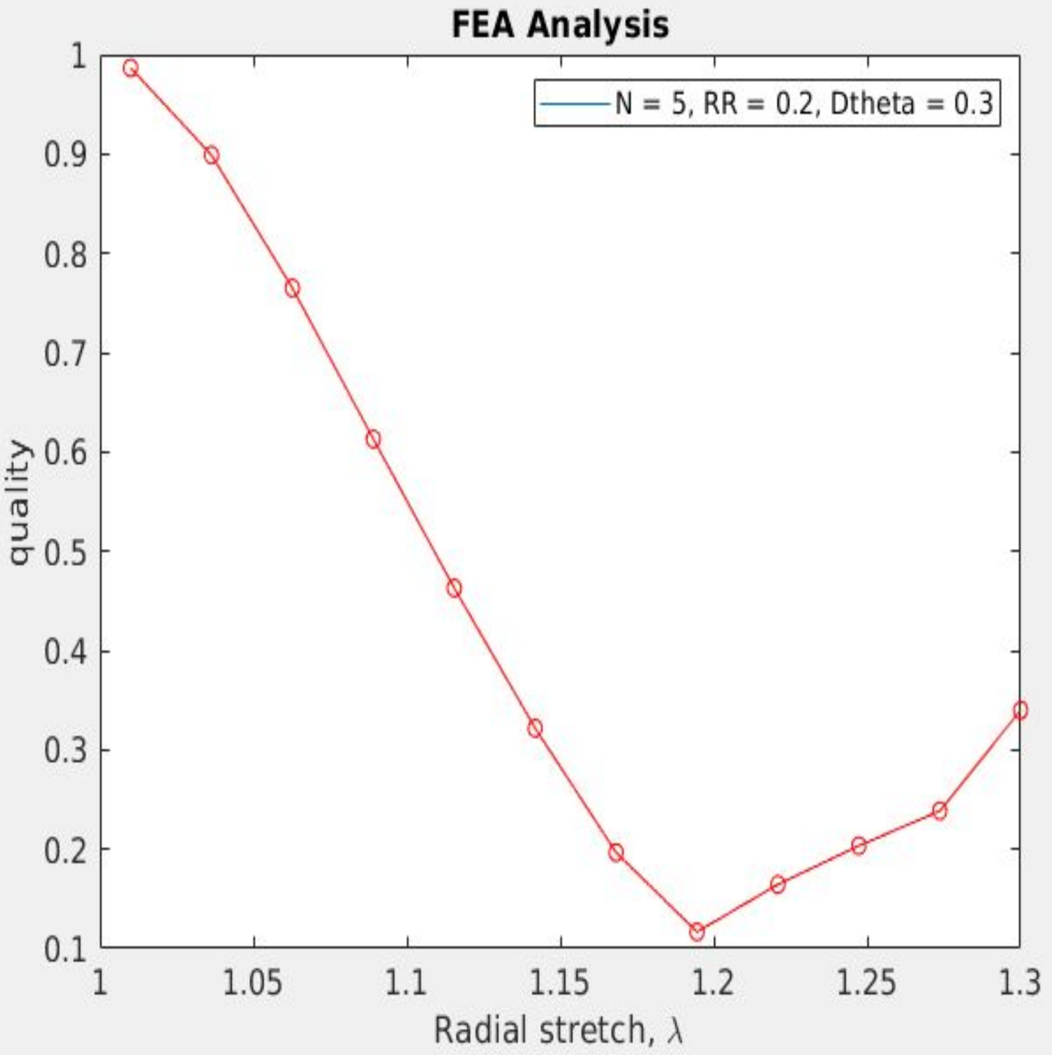
Stretch = 1.21
Quality = 0.15



Data from Vishal

The Best result :

Stretch = 1.19
Quality = 0.1167



Data Comparison

FEA using Mooney–Rivlin model

E are the same