HW4

Seong-Min, Han (20221060)

1. Solve the Laplace equation for a circle.
In this time, write a *.contact file (your own format).
By using this file, you can specify the edges belonging to the terminals.

- 구상

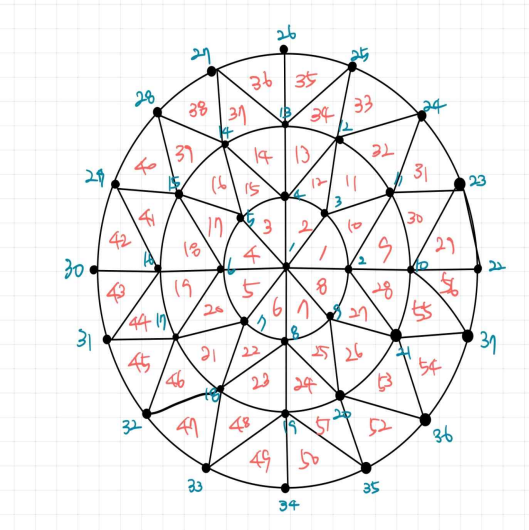


figure1. sketch mesh

위의 그림과 같은 형태로 각과 반지름을 다르게 하여 mesh를 나누었습니다.

가장 안쪽 원부터 $\frac{\pi}{4}, \frac{\pi}{6}, \frac{\pi}{8}$ 순으로 원의 형태로 나누었고, 반지름은 1, 2, 2.5로 구상했습니다.

원의 개수를 numc라고 지정하고 face의 개수를 numf라고 할 때, 일반항은 계차수열을 사용하여

$numf = a_0 + \sum_{i}^{n-1} b_n = 4numc^2 + 8numc - 4$ 로 구할 수 있습니다.

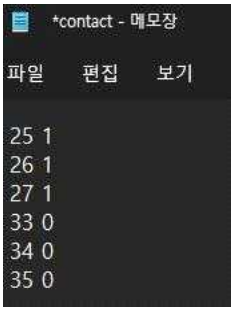
각 원의 vertex 개수를 numv라고 할 때 일반항은 계차수열을 사용하여

$numv = a_0 + \sum_{i}^{n-1} b_n = 2numc^2 + 6numc + 1$ 로 구할 수 있습니다.

각 원의 시작 vertex의 일반항을 계차수열을 사용하여

$a_n = a_0 + \sum_{i}^{n-1} b_n = 2numc^2 + 2numc - 2$로 구할 수 있습니다.

시작 vertex와 각 원의 vertex 개수를 이용하여 각 vertex의 값을 정해줄 수 있습니다.
위 그림에서 top vertex를 기준으로 좌/우측 vertex, 그리고 bottom vertex를 기준으로 좌/우측 vertex에 contact를 지정했습니다.
contact.txt file을 만들어 다음 vertex의 element 정보를 저장하고, MATLAB에서 file을 불러와 해당 vertex의 potential 값에 계산했습니다.



위와 같이 vertex를 지정하여 top vertex에는 1을 bottom vertex는 0을 대입했습니다.
contact edge는 top contact edge 25-26, 26-27이며 bottom contact edge 33-34, 34-35입니다.

- Result

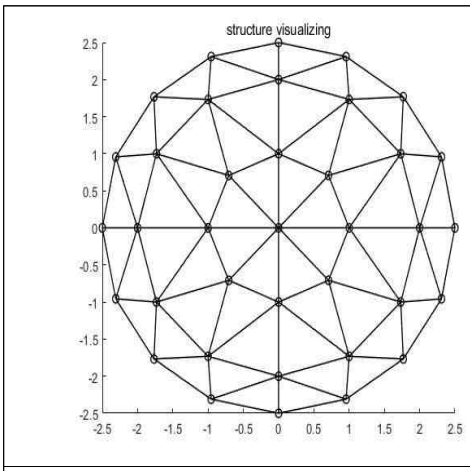


figure2. structure visualizing

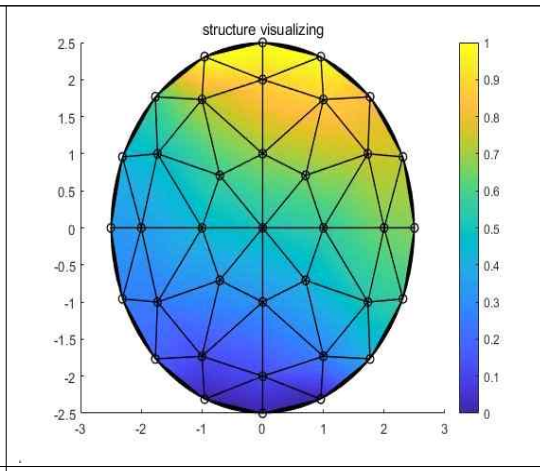


figure3. potential visualizing

21	0.5223
22	0.6666
23	0.7409
24	0.8215
25	1
26	1
27	1
28	0.5936
29	0.4321
30	0.3334
31	0.2591
32	0.1785
33	0
34	0
35	0
36	0.4064
37	0.5679

potential

patch code로 각 vertex에서의 potential 값을 시각화한 결과입니다. 시각화한 potential 결과를 보면 upper contact인 25-26-27, bottom contact인 33-34-35에 값이 올바르게 입력되었다는 것을 알 수 있습니다.

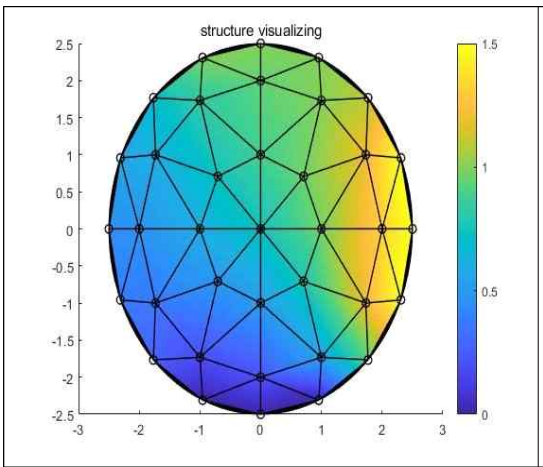
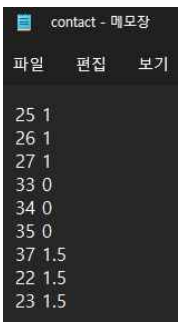


figure4. potential visualizing



contact2

추가로 top, bottom vertex뿐만 아니라 right vertex인 37, 22, 23에 1.5V를 인가한 결과입니다.

2. Write a *.element file (your own format) to specify the yin and yang pattern.
Visualize the read structure.
Calculate the interface edges.

- **구상**

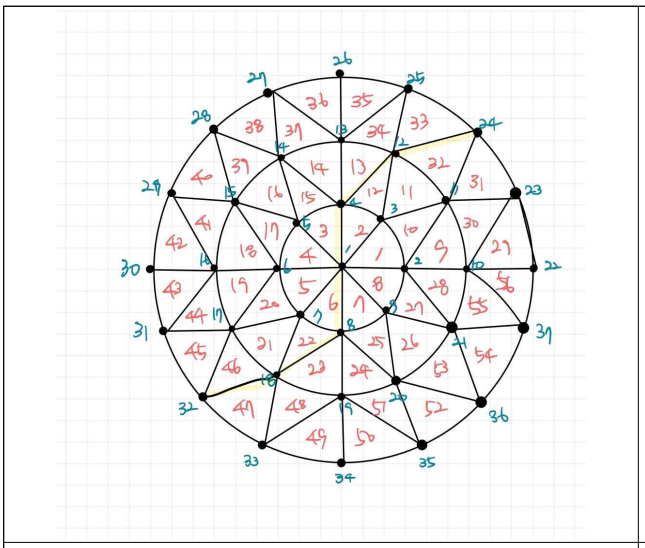


figure5. Describe region & interface

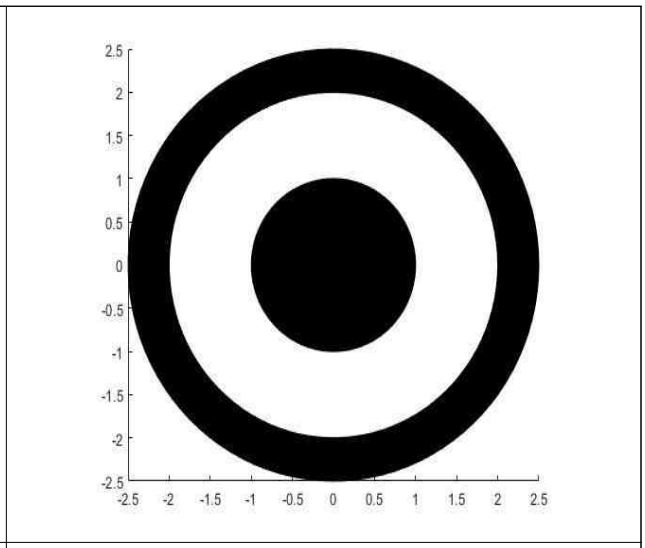


figure6. Describe region & interface

먼저 해당 영역을 figure 6처럼 원으로 분할하고 각과 반지름을 증가시키면서 meshing 하여 vertex와 전체영역에 대한 element를 설정했습니다.

- **Result**

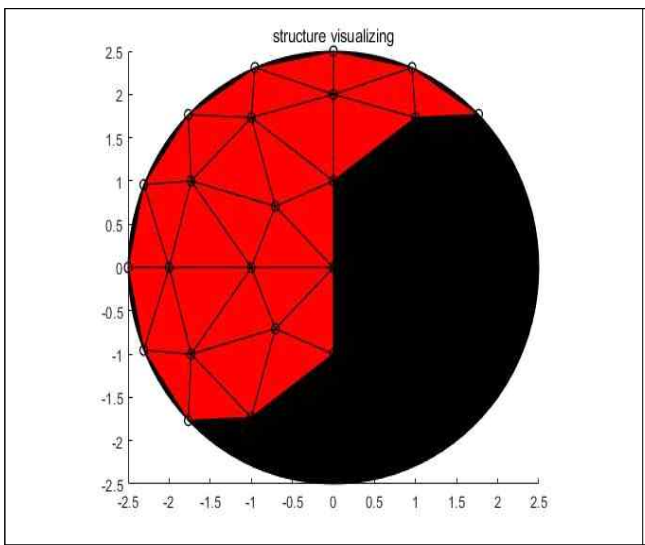


figure7. region1 (oxide)

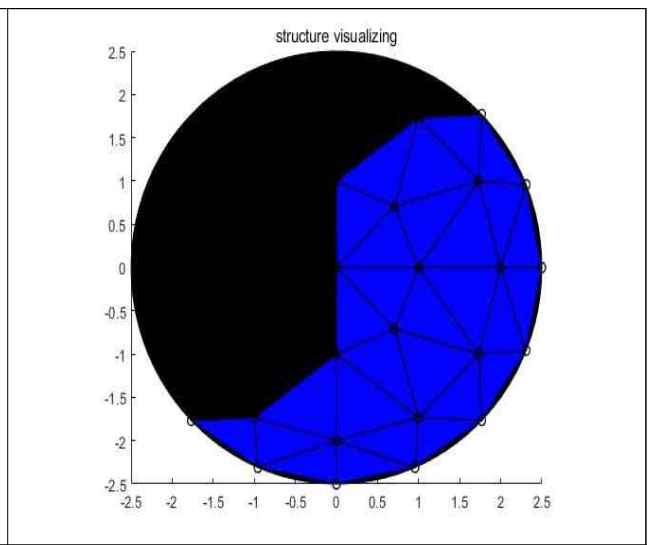


figure8. region2 (silicon)

다음으로, oxide, silicon 각 영역의 element를 따로 저장했습니다. 각 영역의 element file을 통해 region을 구성하고 이를 red/blue color로 구분했습니다.
다음으로, interface에 해당하는 edge를 구하기 위해 각 element file을 비교했습니다. 반복문을 사용하여 oxide element를 기준으로 silicon element를 비교해가면서 같은 vertex가 있는 경우 해당 vertex의 좌우 vertex를 비교하여 또 같은 vertex가 있으면 inter_edge에 저장하도록 설계했습니다.

ex)

oxide.element	1	3	4	-> 방향	<- 방향
silicon.element	1	2	3	<- 방향	-> 방향

oxide element의 1과 silicon element의 1이 같을 경우, 각각 right hand method를 이용하여 비교했습니다. oxide의 1을 기준으로 우측 vertex, silicon을 기준으로 좌측 vertex를 비교했습니다. 만약 일치하는 vertex가 없을 경우, 반대로 oxide 1의 좌측 vertex, silicon 1의 우측 vertex를 비교하여 일치하는 vertex가 있는 경우 inter_edge에 저장했습니다. 각 영역의 element가 중복 vertex가 항상 존재하는 것은 아니었기 때문에 inter_edge는 중간중간 0이 존재하는 배열입니다.
따라서 $N = \mathrm{nnz}(\mathrm{inter\_edge})$ 코드를 통해 inter_edge에서 0이 아닌 요소의 개수를 구하고, 이 값의 절반으로 save_edge 배열을 만들어 0이 아닌 요소를 저장해주었습니다. 요소의 개수는 edge의 양 끝 vertex 2개가 있으므로, 절반의 크기를 행으로 가지는 배열을 만들었습니다.
해당 mesh의 경우, 0이 아닌 요소는 12개이고, 절반인 6의 값으로 save_edge의 행을 구성했습니다.
해당 vertex를 patch code를 통해 기존의 region과 같이 plot 하였고, 결과는 다음과 같습니다.

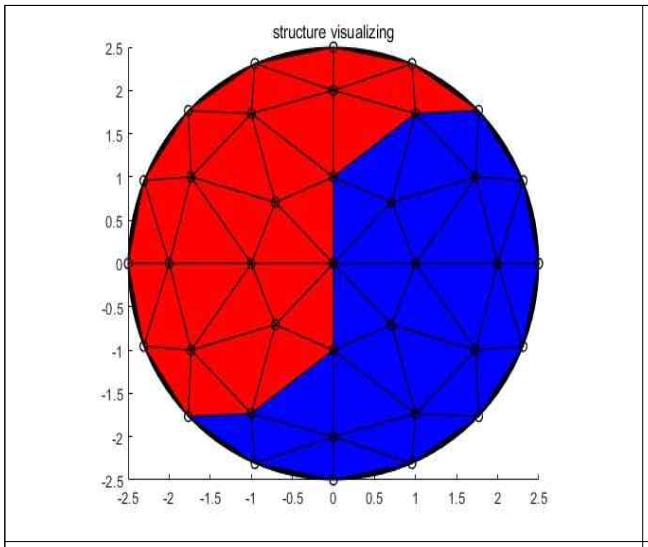


figure9. Visualize region

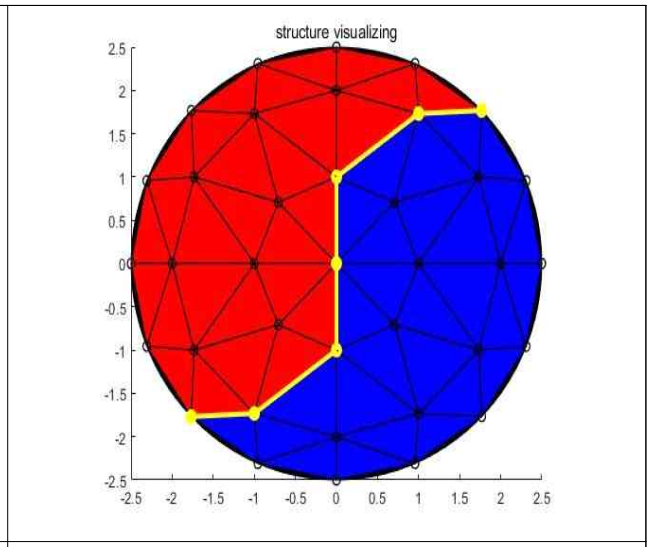


figure10. Region + Interface

결과를 보면 yin and yang pattern을 형성하기 위해 좌측 그림처럼 노란색 선인 interface edge을 기준으로 영역이 나누어진 것을 볼 수 있습니다.
interface edge를 확인하기 위해 save_edge를 확인해보면 다음과 같이 interface edge가 잘 저장된 것을 확인할 수 있습니다.

6x2 double

	1	2
1	1	4
2	1	8
3	12	4
4	18	8
5	24	12
6	32	18