

# 전산물리 기말 프로젝트 보고서

## <유한요소 해석법을 이용한 핀 유용도 해석>

정준상, 김영훈

유한요소 해석법을 이용한 핀 유용도 해석을 진행하였다. 본 보고서는 유한요소 해석 모델 개발 과정, 모델 아이디어에 대한 간략한 설명, 그리고 분석 결과를 포함하고 있다.

시중에는 여러 모양의 핀 (fin) 이 있다. 핀이란 전열면적을 넓혀 대류열전달을 촉진시키기 위한 구조물로, 열교환기, 컴퓨터 heat sink 등에 사용된다. 우리는 이 중 어떤 모양의 핀이 제일 효율적인지 알고 싶었고, 이를 진행하기 위해 유한요소 해석법을 사용하였다.

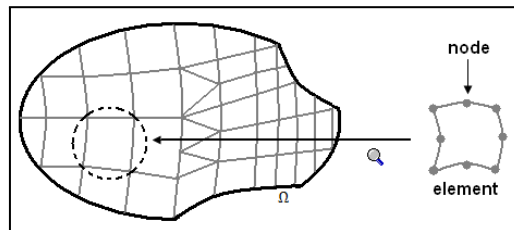
이를 통해 분석한 결과 [~~~~~] 하다는 결과를 얻었고, [~~~~~] ... [~~~~~].

### 1. 모델 개발 과정 및 결과

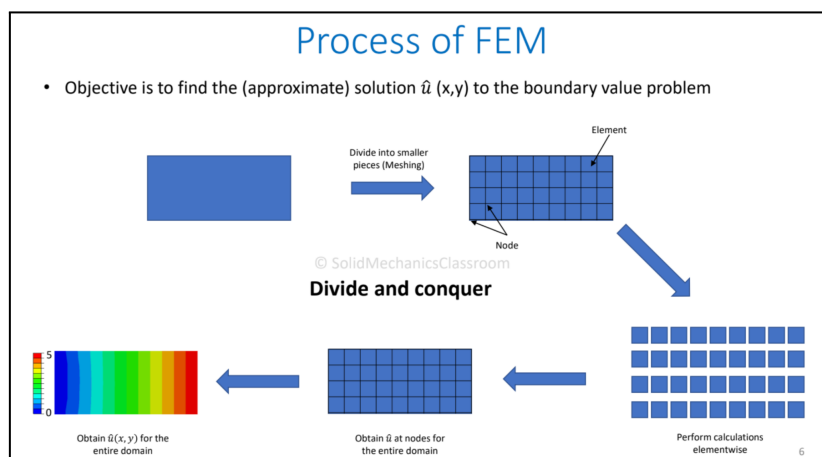
#### a) Background : Finite element method (FEM) and transient heat transfer

모델 개발 과정을 설명하기에 앞서 유한요소 해석법에 대해 설명하고자 한다. 유한요소 해석법이란 “여러 메쉬 (mesh) 를 계산함으로써 해를 얻을 수 있는 수치해석적 방법” 이다. 유한요소 해석법은 보 (beam) 의 처짐, 좌굴하중 (critical load), 유체 해석 등, 지배방정식은 알지만 엄밀해를 구하기 힘든 경우에 많이 쓰인다.

유한요소 해석법은 주어진 모형을 “여러개의 요소 (element)” 로 나눠 계산이 이뤄지는데, 이 요소들의 집합을 메쉬 (mesh) 라 부른다. 또한 요소는 다수의 노드 (node) 로 이뤄지는데, 결국 유한요소 해석법의 계산은 각 노드간 상호작용임을 암시한다.



정리하자면 유한요소 해석법은 주어진 기하요소를 메쉬화 시켜 여러개의 요소로 나눠, 요소 속 노드간 계산을 통해 해를 얻는 방법이다. 이를 그림으로 나타내면 다음과 같다.



우리는 이 유한요소 해석법으로 각기 다른 핀의 효율을 계산하고자 한다. 그런데 이를 어떻게 적용해야 할까?

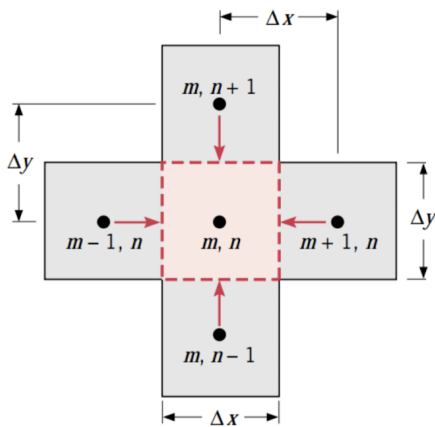
이를 진행하기 위해 우선 열전달이 어떻게 이뤄지는지 알아야 한다. 다음 식은 공간과 시간에 따른 열확산 방정식이다.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{\dot{q}}{k} = \frac{1}{\alpha} \frac{\partial T}{\partial t}$$

$\alpha$ : thermal diffusivity,  $k$ : thermal conductivity,  $\dot{q}$ : generative heat

열확산 방정식은 (해를 구하는 것과 다르게) 생각 외로 매우 단순하다. 에너지 평형법에 기초해 있기 때문이다. 위 식에서 앞선 세항 ( $\nabla^2 T$ ) 는 검사체적에 들어오는 열,  $\frac{\dot{q}}{k}$  는 검사체적에서 발생하는 열, 마지막으로  $\frac{1}{\alpha} \frac{\partial T}{\partial t}$  는 체적이 방출하는 열이다. 즉, 열확산 방정식은 “들어오는 열 = 나가는 열” 인 것이다.

이를 유한요소 해석법에 적용시켜 다음과 같은 메쉬를 생각해보자.



앞서 열확산 방정식은 들어오는 열과 나가는 열이 같다 하였다. 이를 식으로 나타내면 다음과 같다.

$$\dot{E}_{in} + \dot{E}_{generate} = \dot{E}_{out}$$

이 때 메쉬에서 발생되는 열,  $\dot{E}_{generate}$  을 0 이라 가정하면, 노드 (m,n) 에 들어오는 열은 방출하는 열과 같다.

이에 그치지 않고 앞서 열확산 방정식에 유한 차분법 (finite difference method) 를 적용하자. 이를 적용하는 이유는 애초에 메쉬가 등간격으로 나뉘어져 있고, 이를 통해  $\frac{\partial^2 T}{\partial x^2}$  을  $\frac{T_{(m+1,n)} - 2T_{(m,n)} + T_{(m-1,n)}}{dx^2}$  처럼 나타낼 수 있기 때문이다.

결국 위 과정을 거쳐 식을 정리하면 열확산 방정식은 다음처럼 나타낼 수 있다.

$$T_{(m,n)}^{p+1} = Fo(T_{(m+1,n)}^p + T_{(m-1,n)}^p + T_{(m,n+1)}^p + T_{(m,n-1)}^p) + (1 - 4Fo)T_{(m,n)}^p$$

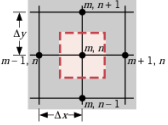
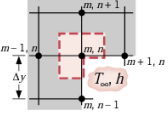
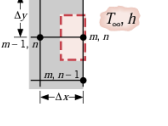
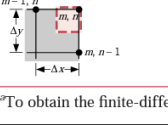
$Fo$ : fourier number =  $\frac{\alpha \Delta t}{dx^2 dy}$

여기서 푸리에 수 (fourier number) 는 쉽게말해 “단위 시간동안 열이 이동하는 거리” 를 의미한다.  $\alpha \Delta t$  는 단위 시간  $\Delta t$  동안 열확산이 일어난 면적을 의미하고,  $dx * dy$  는 우리가 생성한 요소의 크기를 뜻한다. 때문에 만약  $Fo \leq 0$  이라면 유한요소 해석법을 진행할 수 없다. 애초에 단위 시간동안 확산된 열이 인접한 노드를 지나 다른 노드에 까지 영향을 미치지 때문이다.

또한 위 식은 매우 한정된 상황에서만 맞는다. 위 메쉬를 보면 노드 (m,n) 에는 인접한 4 개의 노드가 있고, 그 4 개는 모두 전도를 통해 열이 전달된다. 때문에 위 식은 정확히 말하자면 “전도를 통한 열확산 상황에서의 유한요소 해석법” 이다.

하지만 우리는 더 많은 상황에도 적용시키고 싶었고, 조사 결과 다음과 같은 참고 자료를 찾을 수 있었다.

**TABLE 5.3** Transient, two-dimensional finite-difference equations ( $\Delta x = \Delta y$ )

Configuration	(a) Explicit Method		(b) Implicit Method
	Finite-Difference Equation	Stability Criterion	
 <p>1. Interior node</p>	$T_{m,n}^{p+1} = Fo(T_{m+1,n}^p + T_{m-1,n}^p + T_{m,n+1}^p + T_{m,n-1}^p) + (1 - 4Fo)T_{m,n}^p \quad (5.76)$	$Fo \leq \frac{1}{4} \quad (5.80)$	$(1 + 4Fo)T_{m,n}^{p+1} - Fo(T_{m+1,n}^{p+1} + T_{m-1,n}^{p+1} + T_{m,n+1}^{p+1} + T_{m,n-1}^{p+1}) = T_{m,n}^p \quad (5.92)$
 <p>2. Node at interior corner with convection</p>	$T_{m,n}^{p+1} = \frac{2}{3}Fo(T_{m+1,n}^p + 2T_{m-1,n}^p + 2T_{m,n+1}^p + T_{m,n-1}^p + 2BiT_{\infty}) + (1 - 4Fo - \frac{4}{3}BiFo)T_{m,n}^p \quad (5.85)$	$Fo(3 + Bi) \leq \frac{3}{4} \quad (5.86)$	$(1 + 4Fo(1 + \frac{1}{3}Bi))T_{m,n}^{p+1} - \frac{2}{3}Fo \cdot (T_{m+1,n}^{p+1} + 2T_{m-1,n}^{p+1} + 2T_{m,n+1}^{p+1} + T_{m,n-1}^{p+1}) = T_{m,n}^p + \frac{4}{3}BiFoT_{\infty} \quad (5.95)$
 <p>3. Node at plane surface with convection<sup>a</sup></p>	$T_{m,n}^{p+1} = Fo(2T_{m-1,n}^p + T_{m,n+1}^p + T_{m,n-1}^p + 2BiT_{\infty}) + (1 - 4Fo - 2BiFo)T_{m,n}^p \quad (5.87)$	$Fo(2 + Bi) \leq \frac{1}{2} \quad (5.88)$	$(1 + 2Fo(2 + Bi))T_{m,n}^{p+1} - Fo(2T_{m-1,n}^{p+1} + T_{m,n+1}^{p+1} + T_{m,n-1}^{p+1}) = T_{m,n}^p + 2BiFoT_{\infty} \quad (5.96)$
 <p>4. Node at exterior corner with convection</p>	$T_{m,n}^{p+1} = 2Fo(T_{m-1,n}^p + T_{m,n-1}^p + 2BiT_{\infty}) + (1 - 4Fo - 4BiFo)T_{m,n}^p \quad (5.89)$	$Fo(1 + Bi) \leq \frac{1}{4} \quad (5.90)$	$(1 + 4Fo(1 + Bi))T_{m,n}^{p+1} - 2Fo(T_{m-1,n}^{p+1} + T_{m,n-1}^{p+1}) = T_{m,n}^p + 4BiFoT_{\infty} \quad (5.97)$

<sup>a</sup>To obtain the finite-difference equation and/or stability criterion for an adiabatic surface (or surface of symmetry), simply set  $Bi$  equal to zero.

Bergman, T. L., and Frank P. Incropera. Fundamentals of Heat and Mass Transfer. Seventh edition. Wiley, 2011, 330-334

이를 통해 우리는 전도와 대류로 인한 과도 열전달 (transient heat transfer) 상황을 유한요소 해석법에 적용할 수 있었고, 최종 결과물에서 이를 구현하였다.

## b) First model

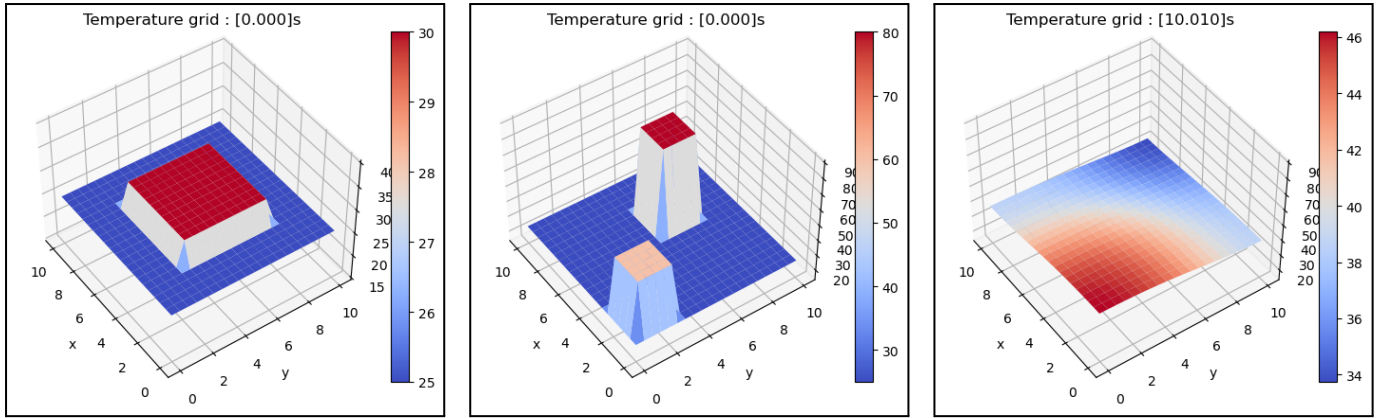
일단 시험삼아 대류 열전달을 고려하지 않은 유한요소 해석 모델을 만들었다.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import cv2
4 from matplotlib.cm import coolwarm
5 from matplotlib.backends.backend_agg import FigureCanvasAgg
6
7 class FEM_2d :
8     def __init__(self, alpha, X, Y, T, dx, dy) :-
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

해당 모델은 메쉬 크기, boundary condition, initial condition 등을 자유롭게 조절할 수 있고, 계산 결과를 plot 시키거나 계산 과정을 avi 파일로 저장할 수 있다.



이를 구현하며 몇가지 문제점과 개발 방향을 되잡을 수 있었다.

가장 큰 문제점은 계산량이었다. 우리가 더 높은 정확도와 (온도 grid 의) 해상도를 원할수록 더 작은 메쉬 크기 ( $dx * dy$ )를 설정해야 한다. 하지만 메쉬 크기가 작아지면 해석 안정성을 위해 단위시간  $dt$  또한 작아져야 한다. ( $1 - 4Fo \geq 0$ ) 때문에 정확성이 높을수록 계산해야될 요소 수도 많아지고, 정상상태 (steady state) 에 도달하기 위한 iteration 또한 많아져, 계산량이 급절로 늘어났다. ( $dt$ 가 작아지므로 시간  $T$  까지 도달하기까지 더 많은 반복이 필요하다.)

두번째 문제로 계산 과정을 저장하는데 또한 오래 걸린다는 것이었다. 해당 모델이 계산 과정을 저장하는 방식은, plot 된 figure 의 RGB 값을 OpenCV 로 읽어들여 frame 마다 저장하는 방식이다. 즉, 한 iteration 이 완료될 때마다 figure 의 RGB 를 추출해 저장하는 것이다. 이는 당연히 느릴 수밖에 없다.

(matplotlib 의 figure 는 자체적으로 plot 의 RGB 값을 보여주는 기능이 없다. 때문에 이를 가능케하기 위해선 matplotlib 의 backend 를 이용해야 한다. 즉, 복잡하고 추출하는데도 오래 걸린다.)

하지만 단점만 있던것은 아니었다. 우선 각 iteration 별 계산이 어떻게 이뤄질지 구체화 되었고 (iteration 별 boundary 에 있는 노드의 온도 설정, 메모리 누수 방지 등), 문제점을 파악했으므로 이를 보완할 여지가 있었다. 또한 여러 편의 기능을 확장시킬 필요성과 이들을 각기 관리할 필요성을 느꼈다.

### c) Second model

앞서 인지한 문제점을 상기하며 두번째 모델을 만들었다. 이에 대한 전체 소스코드와 사용법은 github repository 에서 확인할 수 있다. ([https://github.com/jbw9964/FEM\\_2d.git](https://github.com/jbw9964/FEM_2d.git))

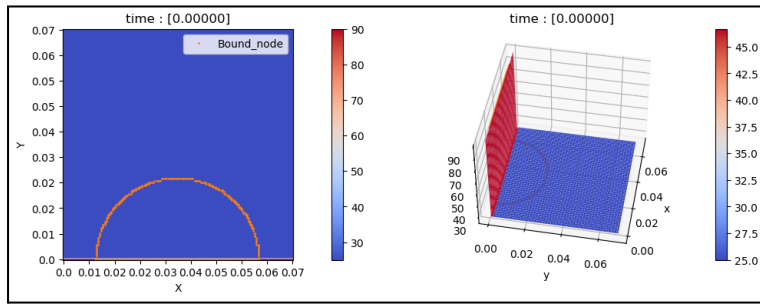
```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  import time, sys
4
5  from numpy import pi, sin, cos
6  from math import sqrt
7  from functools import partial
8  from matplotlib.cm import coolwarm
9  from matplotlib.animation import FuncAnimation
10
11 > class Bound_node : ...
13
14 > class Mesh : ...
246
247 > class FEM_2D() : ...
952

```

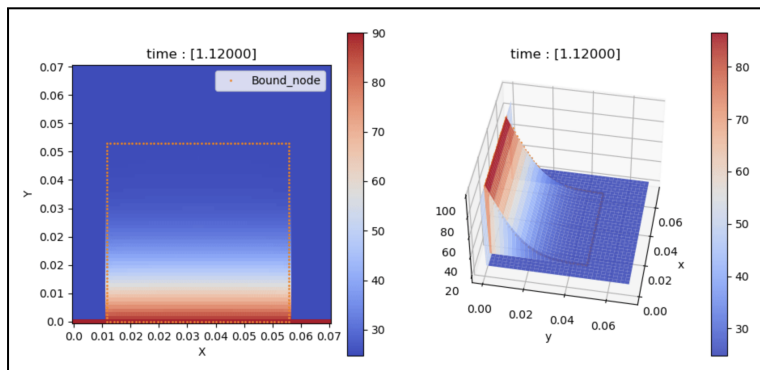
[1]

두번째 모델은 각 iteration 별 계산량을 줄이고 matplotlib 의 FuncAnimation 을 이용해 계산 과정을 gif 로 볼 수 있다. 또한 핀을 세가지 기하형태 (직사각형, 반원, 삼각형) 로 나타낼 수 있고, 전도와 대류로 인한 열전달을 포함해 계산하였다.

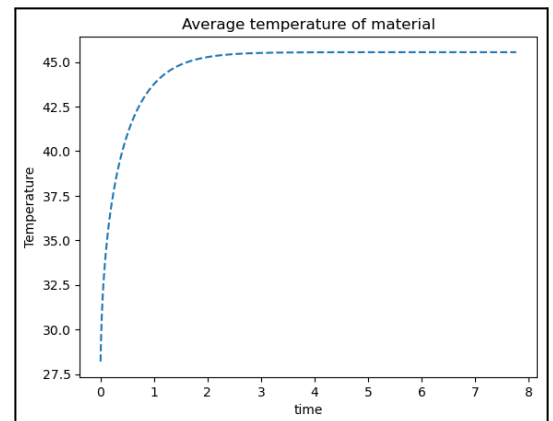
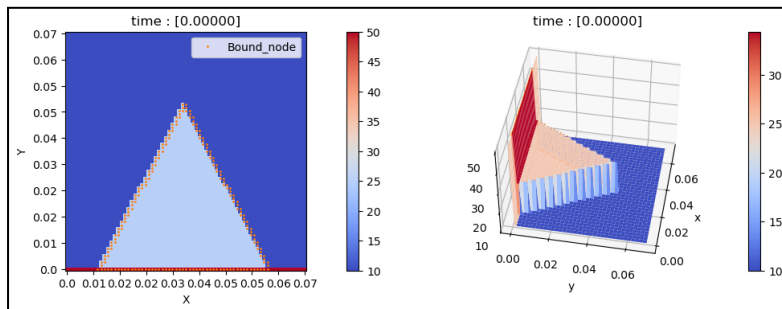


그림을 보면 핀의 경계면이 Bound\_node 로 구분되는 것을 볼 수 있고, 각 기하 형태는  $y = 0$  축에 고정되어 있다.

또한 세번째 그림을 보면 fin 의 초기 온도와 boundary temperature 가 조절 가능함을 알 수 있다.



이외에도 계산 과정을 한 plot 에 보여주는 기능, 핀이 정상상태에 도달할 때까지 자동 계산하는 기능, 계산 과정을 저장, 불러오는 기능 등을 추가하였다.



## 2. 모델 아이디어

앞서 두번째 모델은 계산량을 줄여 속도를 빠르게 했다 하였다. 이를 어떤 방식으로 해결하였는지 설명하도록 하겠다. 우선 두번째 모델의 구조를 보자.

```

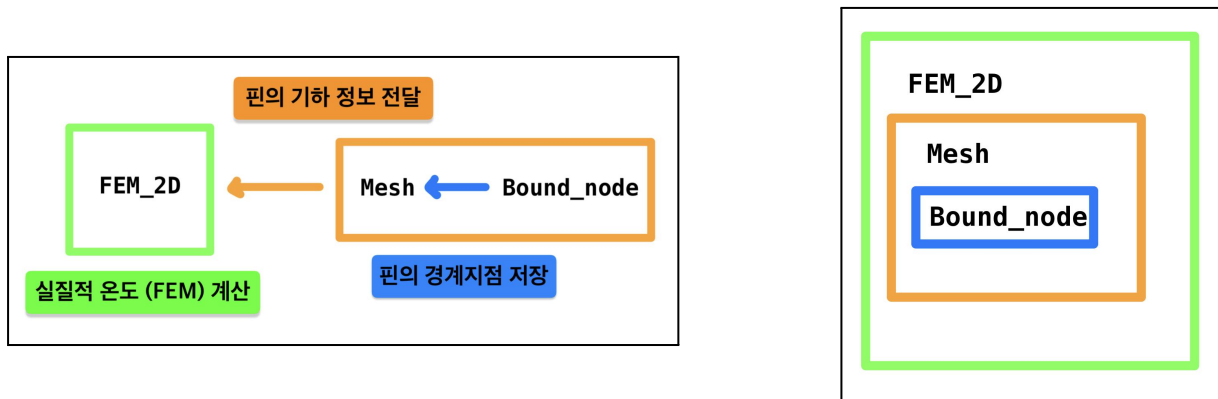
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import time, sys
4
5 from numpy import pi, sin, cos
6 from math import sqrt
7 from functools import partial
8 from matplotlib.cm import coolwarm
9 from matplotlib.animation import FuncAnimation
10
11 >class Bound_node : ...
12
13
14 >class Mesh : ...
15
16
17 >class FEM_2D() : ...
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

```

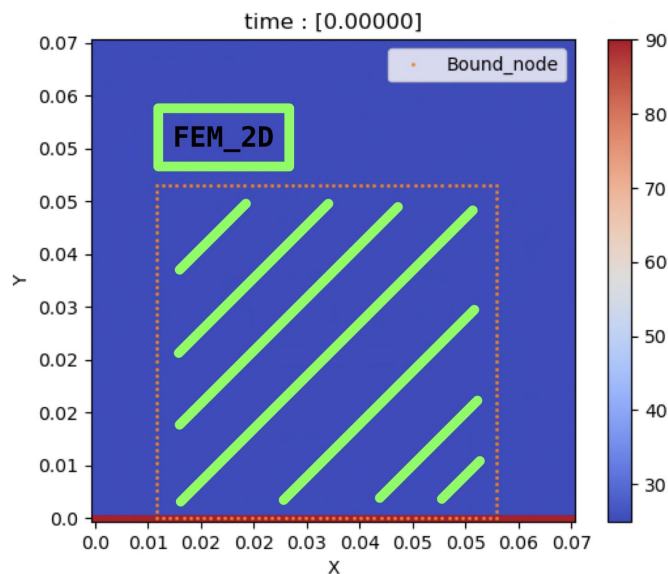
위 코드 중, Bound\_node class 는 핀의 경계면을 표시할 empty class 이다. 그리고 Mesh class 는 핀의 기하 형태를 만들고 경계면 위치를 저장하는 class 이다. 마지막으로 FEM\_2D class 는 위 Mesh class 의 정보를 이용해 온도 grid 를 계산하는 class 이다.

이들을 기능적인 측면에서 바라보면 Bound\_node 는 Mesh 의 subset, Mesh 는 FEM\_2D 의 subset 임을 알 수 있다. 그리고 소스코드를 보면 Bound\_node 는 Mesh class 내부 member 로 사용되고, Mesh 는 FEM\_2D class 내부 member 로 사용됨을 확인할 수 있다.

때문에 각 class 들을 기능적으로, 공간적으로 나타내면 다음 그림과 같다.

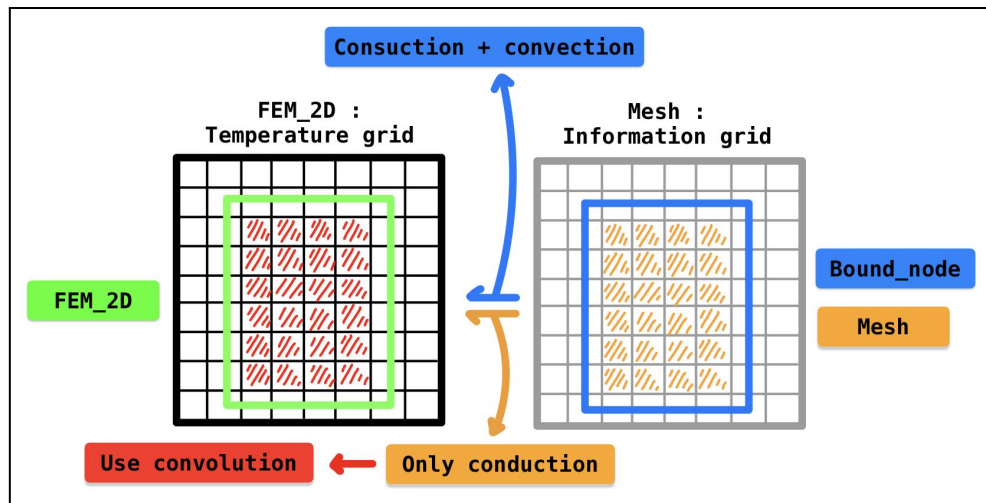


이처럼 구조를 설계하면 2 가지 이점이 있다. 첫째로 유한요소 해석법을 이용해 온도 계산을 진행할 때, 오직 핀 내부에 존재하는 메쉬만 계산하면 된다. 한번 메쉬 그림을 자세히 보자.



앞서 Bound\_node 를 통해 핀의 경계면을 표시한다 하였다. 그런데 우리가 원하는 것은 결국 (테두리를 포함한) 핀의 내부 온도 변화일 뿐이다. 즉, Bound\_node 바깥은 계산할 필요가 없다는 것이다. 이전 첫번째 모델의 경우 전체 메쉬를 계산하여 시간이 오래걸렸지만, 위 구조를 가진 두번째 모델은 필요한 영역만 계산하여 계산량을 한번 줄일 수 있다.

두번째 이점은 각 class 별 기능을 다르게 만들어 관리하기 편하고, 궁극적으로 2 개의 메쉬를 통해 온도계산과 정보처리를 빠르게 한다는 점이다. 이를 그림을 통해 알아보자.



앞서 FEM\_2D 는 Mesh 의 정보를 이용해 온도 grid 를 계산한다 하였다. 두번째 모델에서 핀의 기하 형태가 만들어졌을때, 모델은 2 개의 메쉬를 생성한다. 하나는 온도 grid 를 나타내는 메쉬, 다른 하나는 기하 형태에 따른 정보를 저장할 메쉬이다.

여기서 중요한 것은 정보 메쉬인데, 기하 형태가 선언되면 시간이 조금 걸리더라도 핀의 테두리 위치와 내부의 위치를 나눠서 각기 따로 저장해둔다. 이렇게 둘을 나누는 이유가 있는데, 핀 내부는 오직 전도로 인한 열전달, 테두리는 전도와 대류로 인한 열전달이 일어나기 때문이다.

앞서 전도로 인한 열전달은 대류가 있을때에 비해 식이 간단했었다. 그래서 계산을 convolution 형태로 진행할 수 있고, 이를 통해 계산 시간을 줄일 수 있다.

반면 핀의 테두리는 인접한 노드가 몇개인지, exterior, interior 노드인지에 따라 계산방식이 달라진다. 때문에 각 테두리 node 마다 이를 상황에 맞게 계산한다.

결국 정리하자면, 외부 메쉬는 계산에서 제외하여 계산량을 줄이고, 핀 내부 온도계산은 convolution 을 통해 계산 속도를 높였다. 이러한 방식으로 첫번째 모델에 비해 더 빠른 계산 속도를 보였다.

### 3. 핀 분석 결과

이제 Python 으로 구현한 유한요소 해석 모델을 통해 핀을 분석해보자.