Computational physics Final project

Fin analysis using finite element method

정준상, 김영훈

Presentation context

1. Basic of FEM with fin analysis
2. FEM with solid conduction heat transfer
3. FEM with given fin mesh
4. modeling & make a video
5. Result in 2D & 3D
6. Analysis

1. Basic of FEM with fin analysis

a. What is Finite Element Method?

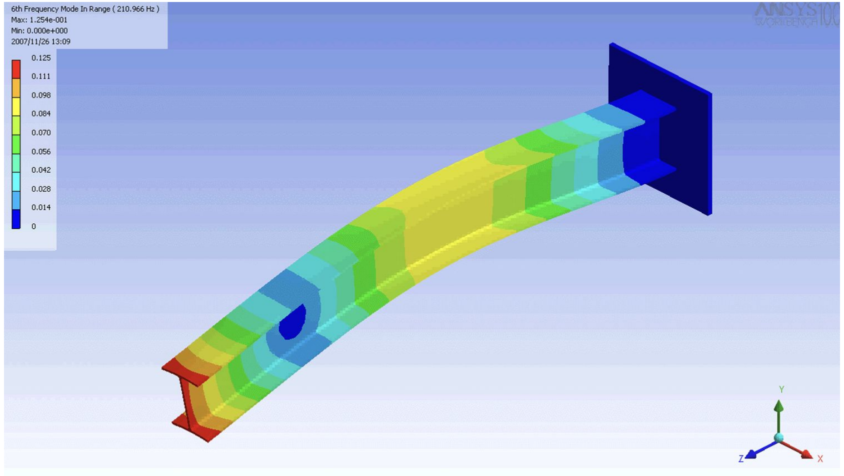
b. What is fin?

c. How can we do this?

1. Basic of FEM with fin analysis -- (a)

What is Finite Element Method?

- FEM is an analytic method to find solution by calculating mesh.

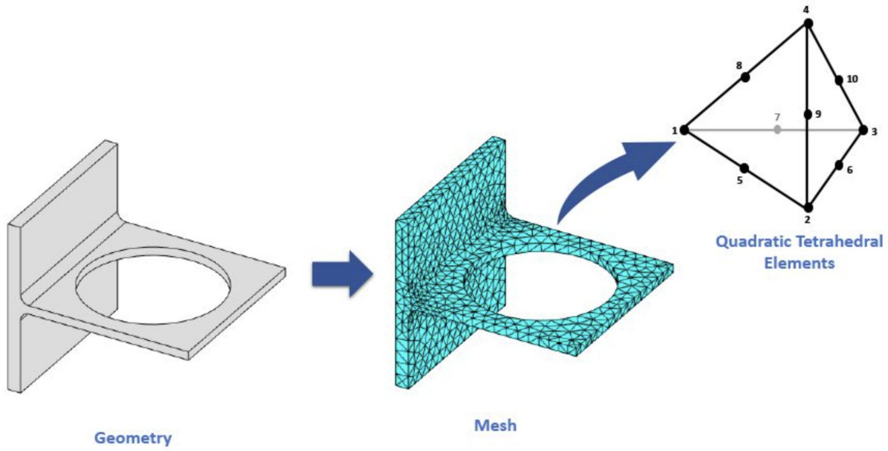


For example, it can be used analyzing critical stress or temperature of material, deflection of beam.

1. Basic of FEM with fin analysis -- (a)

What is Finite Element Method?

- FEM is an analytic method to find solution by calculating mesh.



We slice given geometry to “element”.

We call the set of every elements, “mesh”.

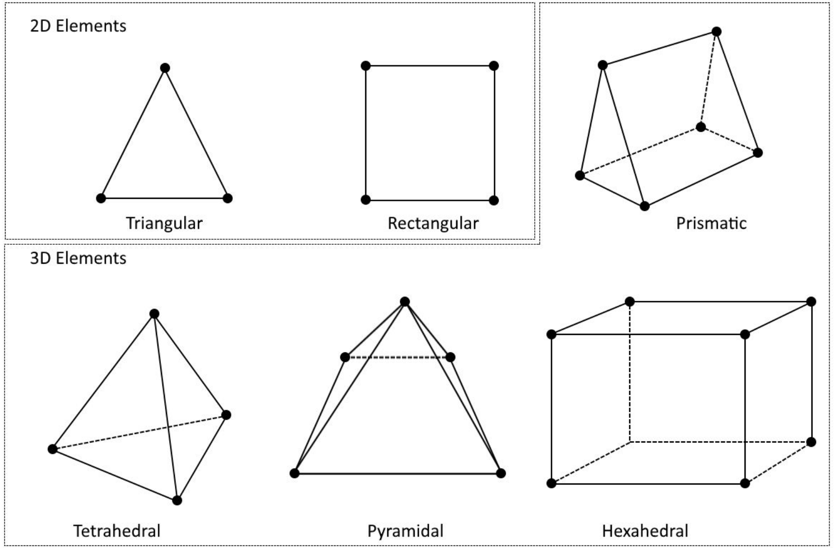
1. Basic of FEM with fin analysis -- (a)

What is Finite Element Method?

- FEM is an analytic method to find solution by calculating mesh.

There could be multiple ways to generating mesh.

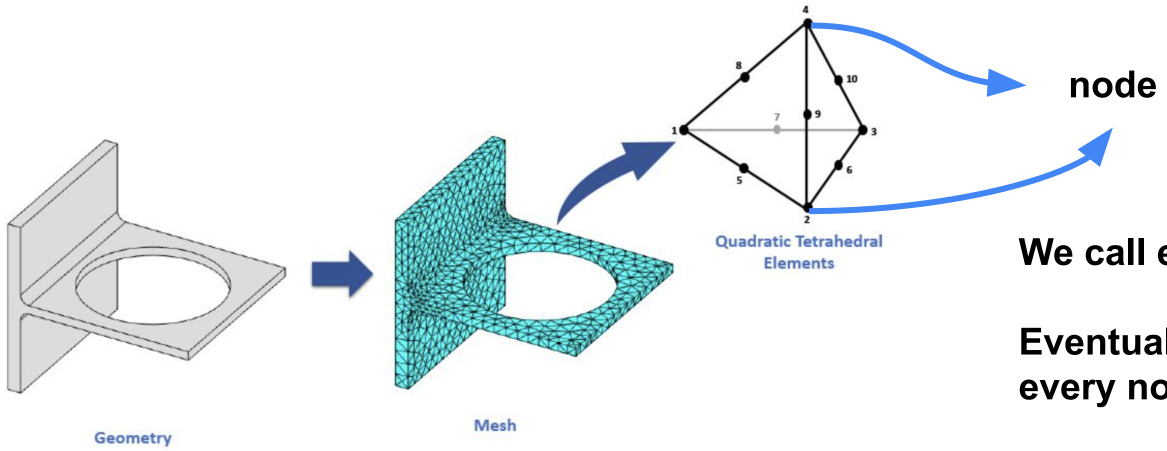
The geometry of mesh may affect the result.



1. Basic of FEM with fin analysis -- (a)

What is Finite Element Method?

- FEM is an analytic method to find solution by calculating mesh.



We call every vertex of mesh “node”.

Eventually, FEM is a way that calculates every nodes via govern equation.

1. Basic of FEM with fin analysis -- (b)

What is fin?

Fin is a surface that extend from an hot object to increase the heat transfer rate.

The amount of conduction, convection and radiation will affect the heat transfer.

In short, if the fin can transfer heat faster, the CPU can cooler.

1. Basic of FEM with fin analysis -- (c)

How can we do this?

- There are several important coefficient about transient fin heat transfer.

1. Biot number (Bi) $$Bi = \frac{hL_c}{k}$$
2. Fourier number (Fo) $$Fo = \frac{\alpha \Delta t}{dx*dy}$$

1. Basic of FEM with fin analysis -- (c)

How can we do this?

- Biot number (Bi)

$$Bi = \frac{hL_c}{k}$$

L_c : characteristic length

h : convection heat transfer coefficient

k : conduction heat transfer coefficient

- Biot number is a coefficient that shows contribution of convection versus conduction in transient heat transfer.
- If Biot number >> 1, convection will be the dominant factor that contributes heat transfer
- If Biot number << 1, conduction will be the dominant factor that contributes heat transfer

1. Basic of FEM with fin analysis -- (c)

How can we do this?

- Biot number (Bi)

$$Bi = \frac{hL_c}{k}$$

L_c : characteristic length

h : convection heat transfer coefficient

k : conduction heat transfer coefficient

- characteristic length is approximated length that describes “outer region VS inner region”

$$L_c \approx \frac{\text{Surface}}{\text{Volume}} \text{ or } \frac{\text{Circumference}}{\text{Surface}}$$

1. Basic of FEM with fin analysis -- (c)

How can we do this?

- Fourier number (Fo)

$$Fo = \frac{\alpha \Delta t}{dx * dy}$$

α : thermal diffusivity

Δt : unit time step

dx and dy : unit length of mesh

- Fourier number is a coefficient that describes “how much heat will travel during the unit time step”.

1. Basic of FEM with fin analysis -- (c)

How can we do this?

- Fourier number (Fo) $$Fo = \frac{\alpha\Delta t}{dx*dy}$$

α : thermal diffusivity

Δt : unit time step

dx and dy : unit length of mesh

- Larger the unit time step becomes, unit mesh length has to be larger too.

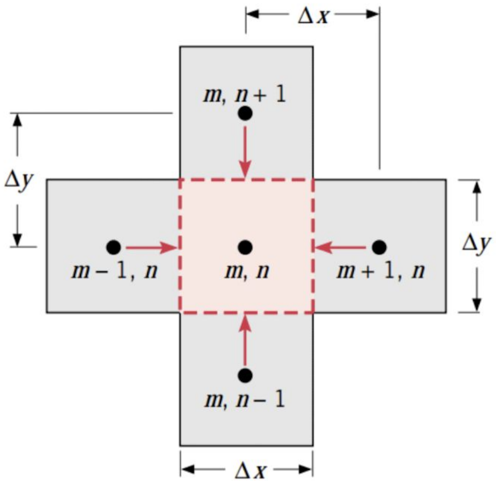
- If we set smaller unit mesh, FEM will collapse, since heat will pass through current mesh element and affect the others.

1. Basic of FEM with fin analysis -- (c)

How can we do this?

- FEM is based on energy equilibrium.

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



(Heat flows to (m,n)) + (Heat that generated by (m,n)) = (Heat that (m,n) emit)

Assume there's no heat generated by material, (q = 0) and apply this condition to Heat diffusion equation

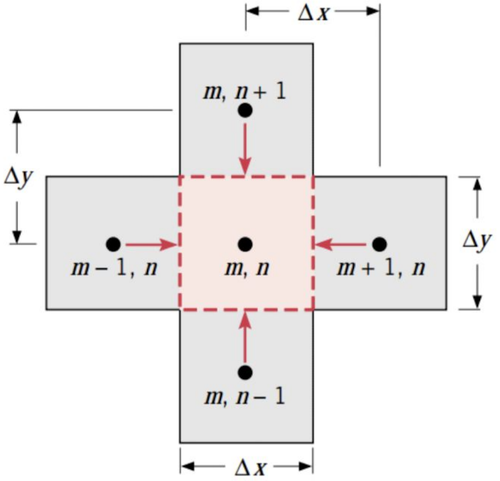
$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{\dot{q}}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t}$$

We can use finite difference approximation to calculate second derivative.

1. Basic of FEM with fin analysis -- (c)

How can we do this?

- Eventually, we can describe the node like below



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

$T_{m,n}^{p}$: temperature of node (m,n) in time (p)

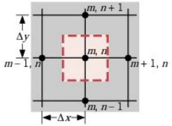
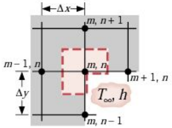
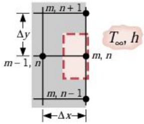
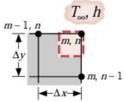
Note that in only applies 2-dimensional, closed mesh (4 nodes are adjacent with node (m,n))

1. Basic of FEM with fin analysis -- (c)

- Other mesh case can be calculated like this

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

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

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

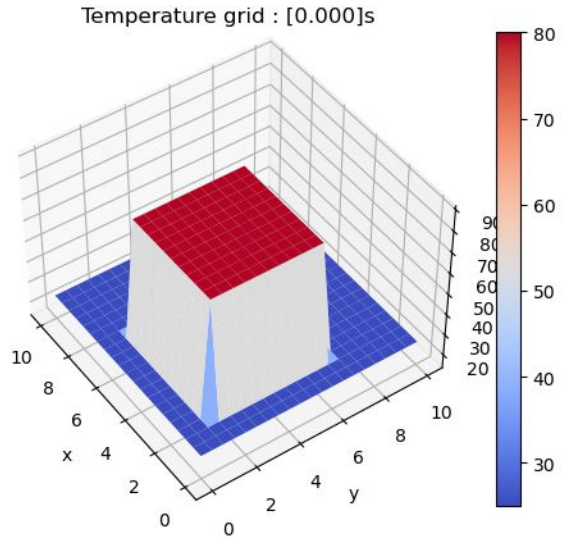
[a]To obtain the finite-difference equation and/or stability criterion for an adiabatic surface (or surface of symmetry), simply set *Bi* equal to zero.

2. FEM with solid conduction heat transfer

We made first model using numpy, matplotlib, and OpenCV to export animation

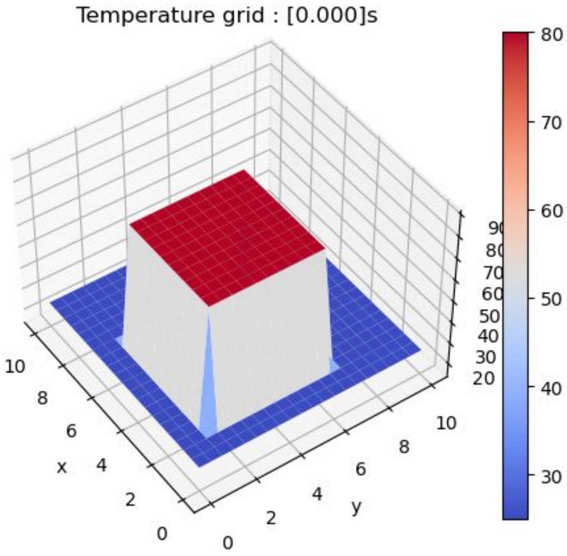
```python
import numpy as np
import matplotlib.pyplot as plt
import cv2
from matplotlib.cm import coolwarm
from matplotlib.backends.backend_agg import FigureCanvasAgg

class FEM_2d :
    def __init__(self, alpha, X, Y, T, dx, dy) :…

    def return_index(self, x, y) : …

    def set_ic_rect(self, T, pt1 : list, pt2 : list) :                # pt1[x0, y0] ~ pt2[x1, y1]…

    def set_bc(self, T, X_axis : list=None, Y_axis : list=None) : …

    def return_plot(self, elev=45, azim=145) : …

    def plot_status(self, elev=45, azim=145) : …

    def print_status(self) : …

    def build(self, dt) : …

    def compute(self, t_end, verbose=False, save_gif=False, elev=45, azim=145) : …

    def save_animation(self, t_end) : …
```

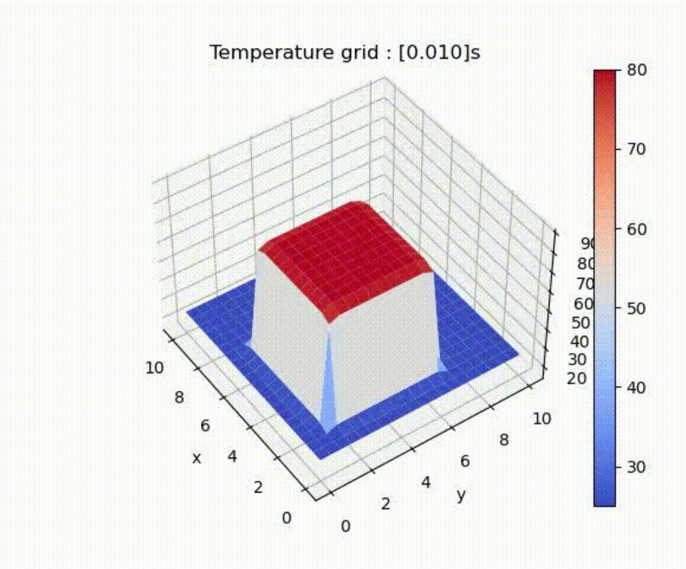
✓ 2.8s   Python

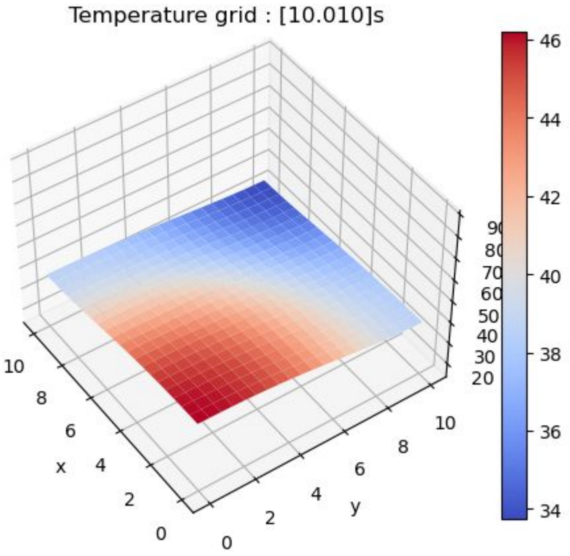


2. FEM with solid conduction heat transfer

It’s a model for rectangular fin, only affected by conduction and boundary condition.

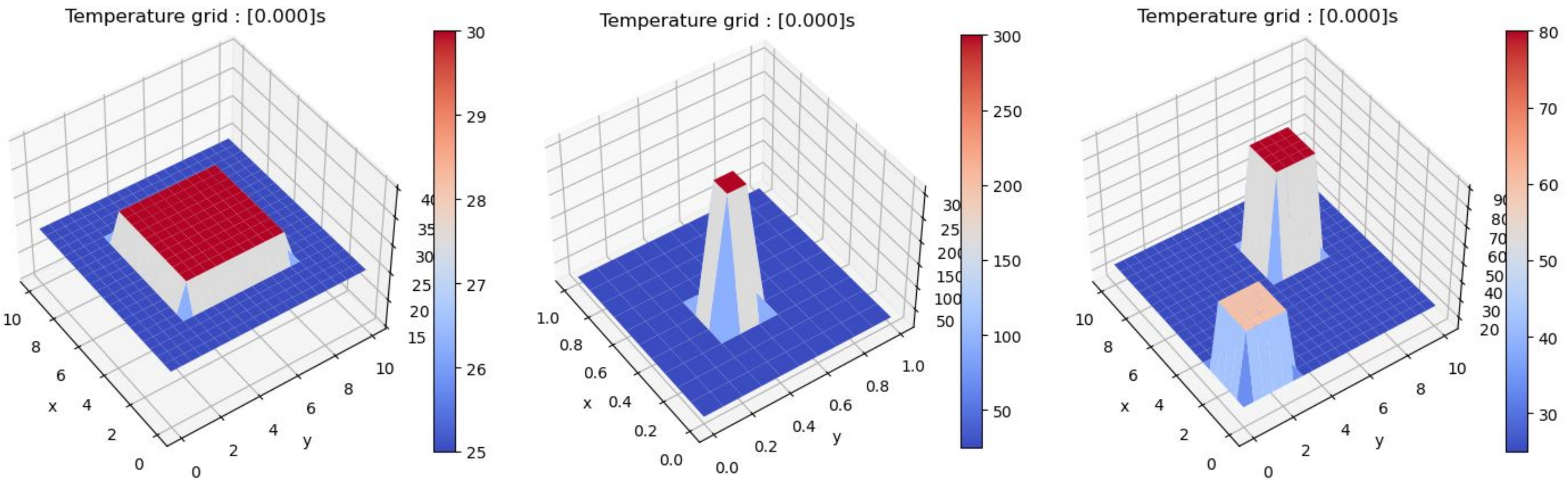






2. FEM with solid conduction heat transfer

You can adjust mesh size, initial conditions and boundary conditions



2. FEM with solid conduction heat transfer

There are disadvantage with this model

1. There’s too many meshes to calculate.
2. Using OpenCV to animate process wasn’t good idea.
3. There’s no utils for computing mesh until steady-state is reached.

```
test.compute(t_end=10, verbose=True, save_gif=True)
[9]  50.2s                                                    Python
t : [0.000] --> [10.000]
Computing mesh... [-]  [>                              ][20/400000]
```

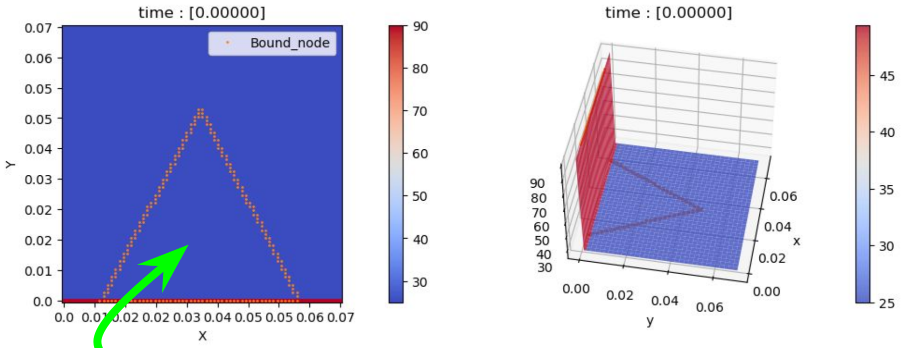
3. FEM with given fin mesh

We’ve made a second model that decrease computation, and uses FuncAnimation of matplotlib to animate process.

```python
import numpy as np
import matplotlib.pyplot as plt
import time, sys

from numpy import pi, sin, cos
from math import sqrt
from functools import partial
from matplotlib.cm import coolwarm
from matplotlib.animation import FuncAnimation

class Bound_node : ...

class Mesh : ...

class FEM_2D() : ...
```

[1]

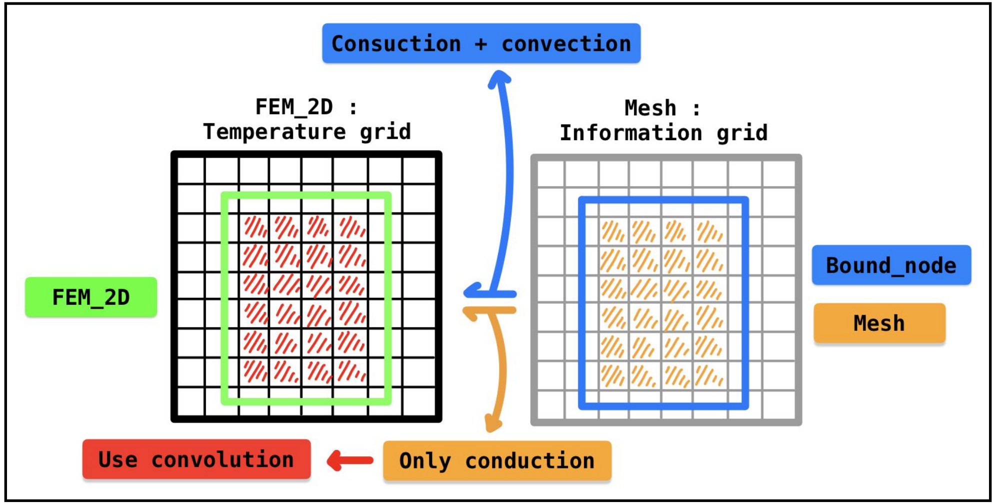
To decrease computation, we only compute the mesh inside the fin.



we only compute inside of the fin

3. FEM with given fin mesh

We've made a second model that decrease computation, and uses FuncAnimation of matplotlib to animate process.



To decrease computation, we use convolution to compute inner temperature of fin.

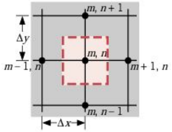
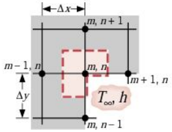
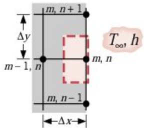
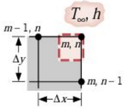
Because inside the fin, every nodes have 4 adjacent nodes, which means only conduction occurs, makes computation easier.

Details at final report - [2. model idea]

$$T^{p+1}_{m,n} = Fo(T^p_{m+1,n} + T^p_{m-1,n} + T^p_{m,n+1} + T^p_{m,n-1}) + (1 - 4Fo)T^p_{m.n}$$

3. FEM with given fin mesh

TABLE 5.3 Transient, two-dimensional finite-differenc

Configuration	(a) Explicit Meth — Finite-Difference Equation	(b) Implicit Method
	$T^{p+1}_{m,n} = Fo(T^p_{m+1,n} + T^p_{m-1,n} + T^p_{m,n+1} + T^p_{m,n-1}) + (1-4Fo)T^p_{m,n}$ (5.76) 1. Interior node	$(1+4Fo)T^{p+1}_{m,n} - Fo(T^{p+1}_{m+1,n} + T^{p+1}_{m-1,n} + T^{p+1}_{m,n+1} + T^{p+1}_{m,n-1}) = T^p_{m,n}$ (5.92)
	$T^{p+1}_{m,n} = \frac{2}{3}Fo(T^p_{m+1,n} + 2T^p_{m-1,n} + 2T^p_{m,n+1} + T^p_{m,n-1} + 2Bi\,T_\infty) + (1-4Fo-\frac{4}{3}Bi\,Fo)T^p_{m,n}$ (5.85) 2. Node at interior corner with convection	$(1+4Fo(1+\frac{1}{3}Bi))T^{p+1}_{m,n} - \frac{2}{3}Fo \cdot (T^{p+1}_{m+1,n} + 2T^{p+1}_{m-1,n} + 2T^{p+1}_{m,n+1} + T^{p+1}_{m,n-1}) = T^p_{m,n} + \frac{4}{3}Bi\,Fo\,T_\infty$ (5.95)
	$T^{p+1}_{m,n} = Fo(2T^p_{m-1,n} + T^p_{m,n+1} + T^p_{m,n-1} + 2Bi\,T_\infty) + (1-4Fo-2Bi\,Fo)T^p_{m,n}$ (5.87) 3. Node at plane surface with convection[a]	$(1+2Fo(2+Bi))T^{p+1}_{m,n} - Fo(2T^{p+1}_{m-1,n} + T^{p+1}_{m,n+1} + T^{p+1}_{m,n-1}) = T^p_{m,n} + 2Bi\,Fo\,T_\infty$ (5.96)
	$T^{p+1}_{m,n} = 2Fo(T^p_{m-1,n} + T^p_{m,n-1} + 2Bi\,T_\infty) + (1-4Fo-4Bi\,Fo)T^p_{m,n}$ (5.89) 4. Node at exterior corner with convection	$(1+4Fo(1+Bi))T^{p+1}_{m,n} - 2Fo(T^{p+1}_{m-1,n} + T^{p+1}_{m,n-1}) = T^p_{m,n} + 4Bi\,Fo\,T_\infty$ (5.97)

[a]To obtain the finite-difference equation and/or stability criterion for an adiabatic simply set Bi equal to zero.

computation, and uses FuncAnimation

To decrease computation, we use convolution to compute inner temperature of fin.

Because inside the fin, every nodes have 4 adjacent nodes, which means only conduction occurs, makes computation easier.

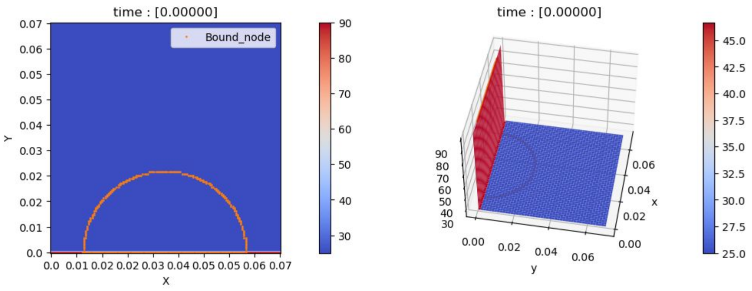
Details at final report - [2. model idea]

$$T^{p+1}_{m,n} = Fo(T^p_{m+1,n} + T^p_{m-1,n} + T^p_{m,n+1} + T^p_{m,n-1}) + (1-4Fo)T^p_{m.n}$$

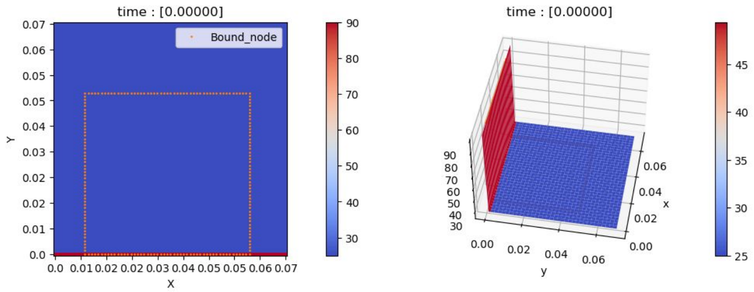
3. FEM with given fin mesh

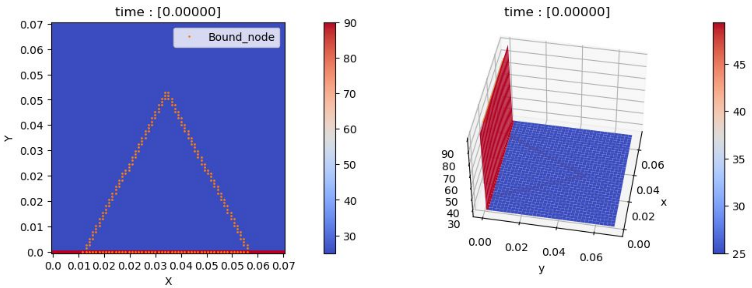
There are three geometry of second model

Half-circle mesh



Rectangular mesh



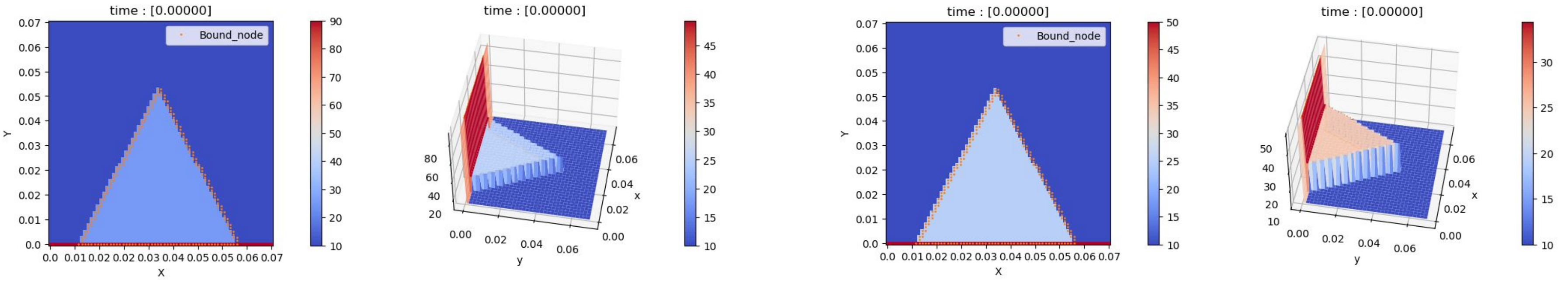


Triangular mesh

3. FEM with given fin mesh

Model can adjust boundary condition, initial condition, and mesh size.

This model includes the convection heat transfer via surrounding air with given coefficient and temperature

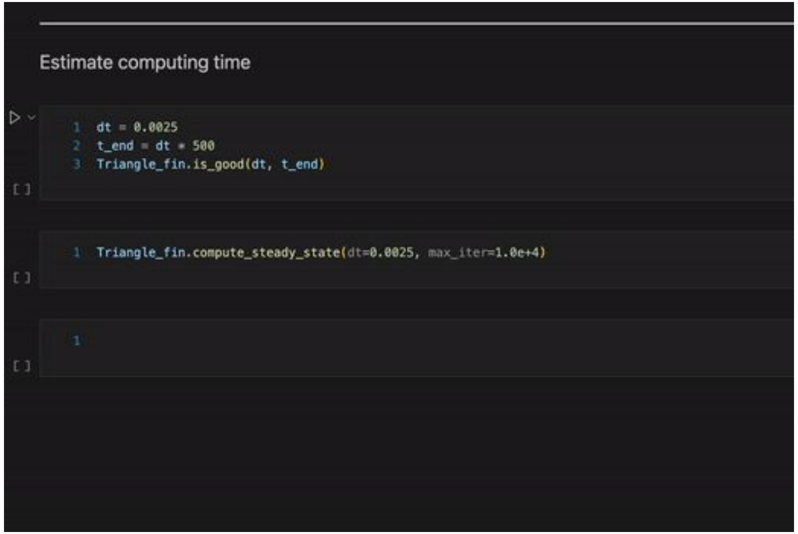


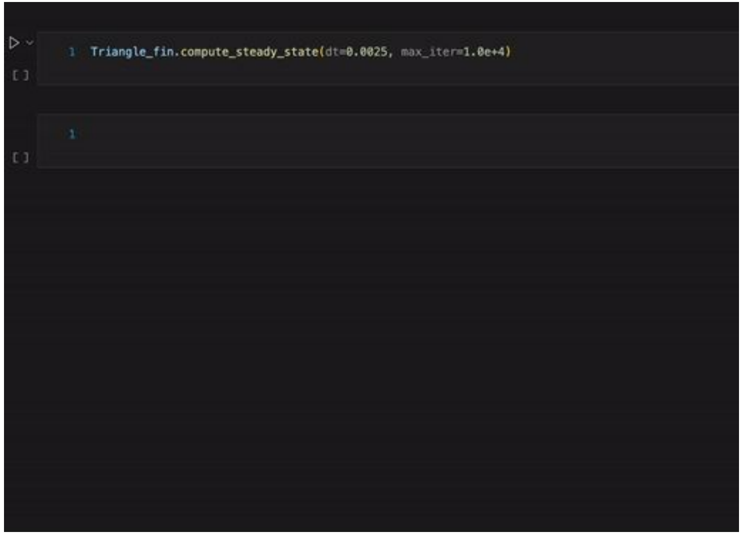
You can see both the air and boundary temperature are different

3. FEM with given fin mesh

We made a utils that compute mesh until steady-state.

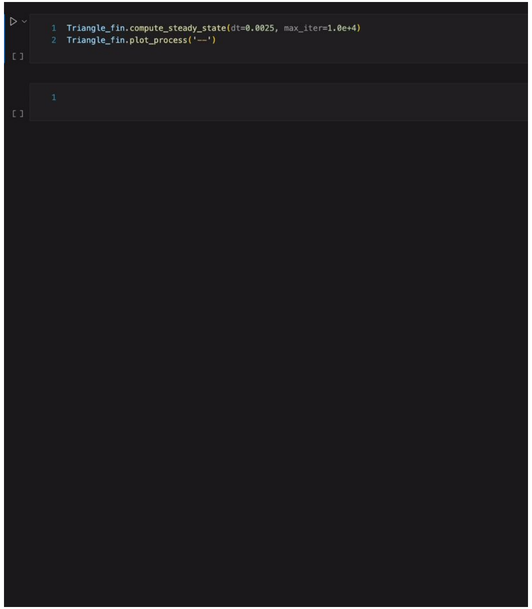
If the model reaches steady-state in max iteration, it will shows the precision of convergence and average temperature. Otherwise, it will stop

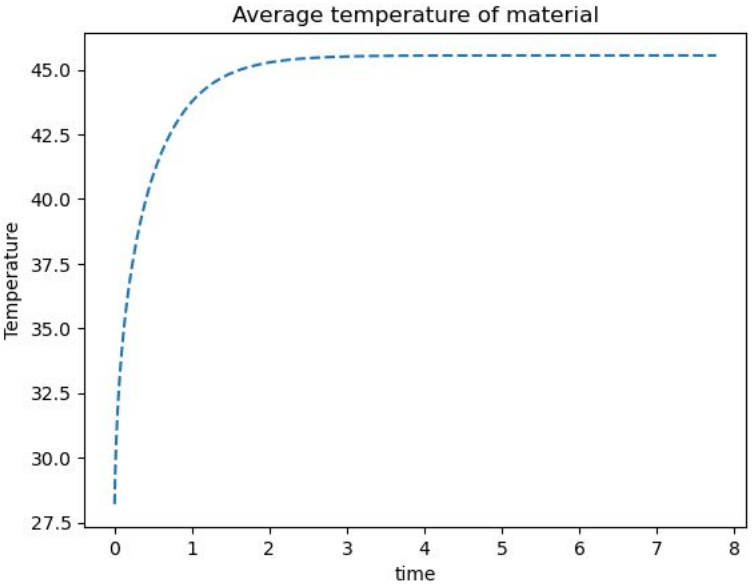




3. FEM with given fin mesh

You can watch the average temperature difference via time.

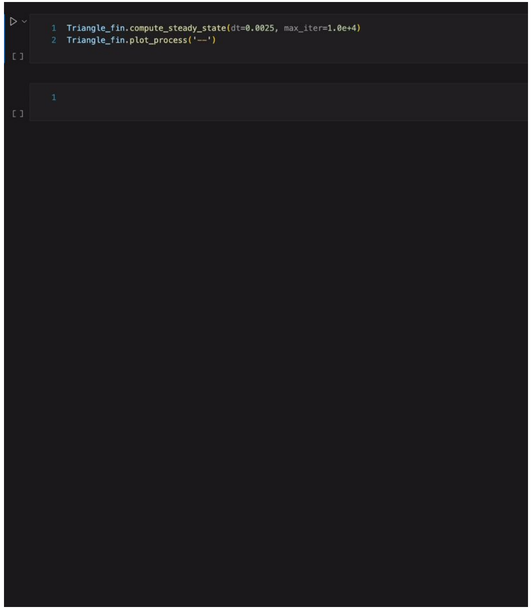


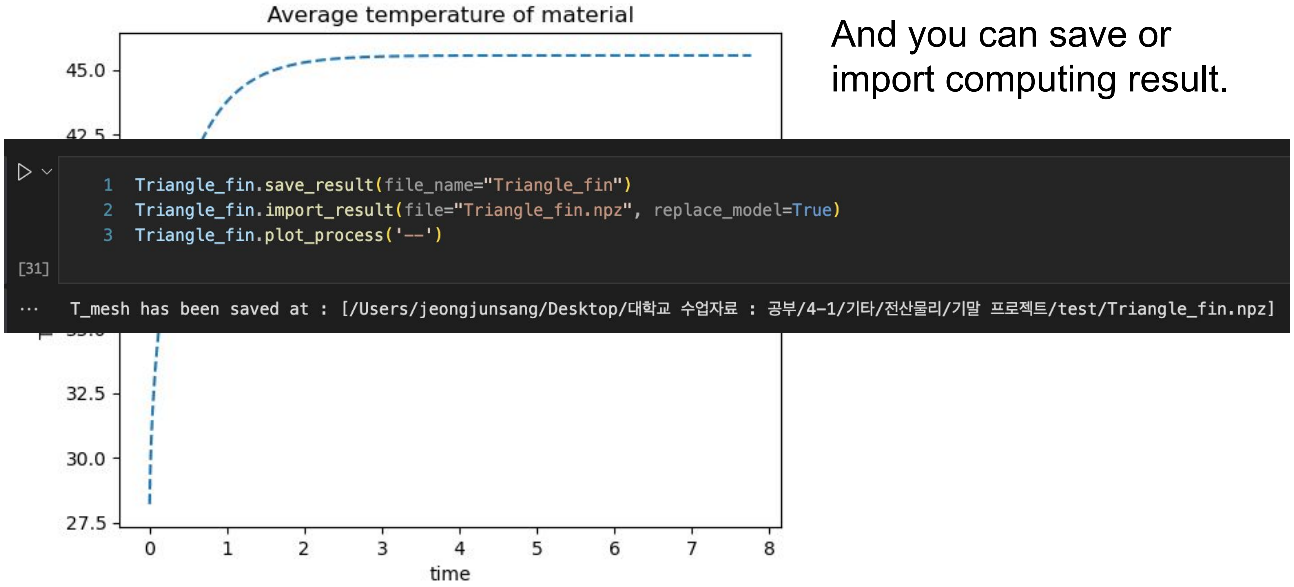


And you can save or import computing result.

3. FEM with given fin mesh

You can watch the average temperature difference via time.

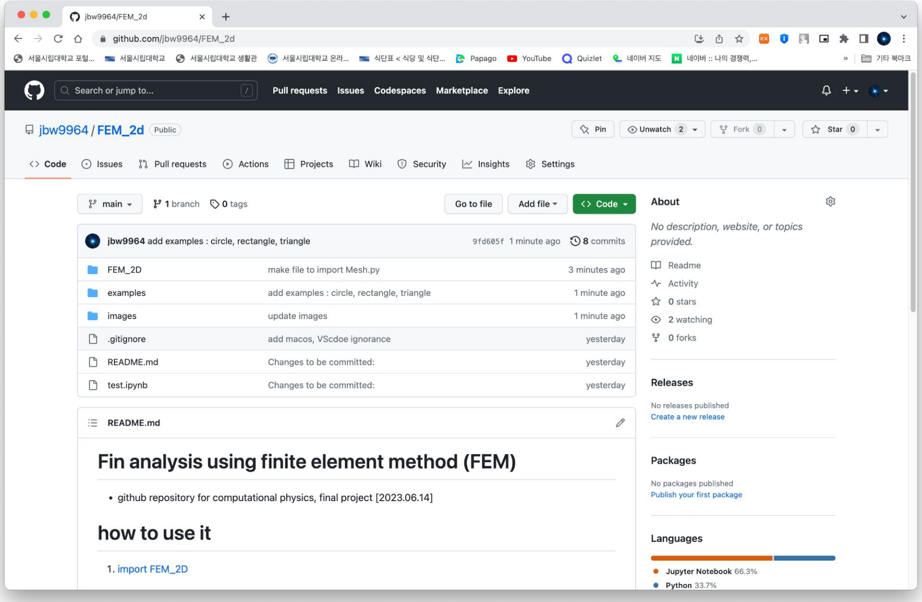




And you can save or import computing result.

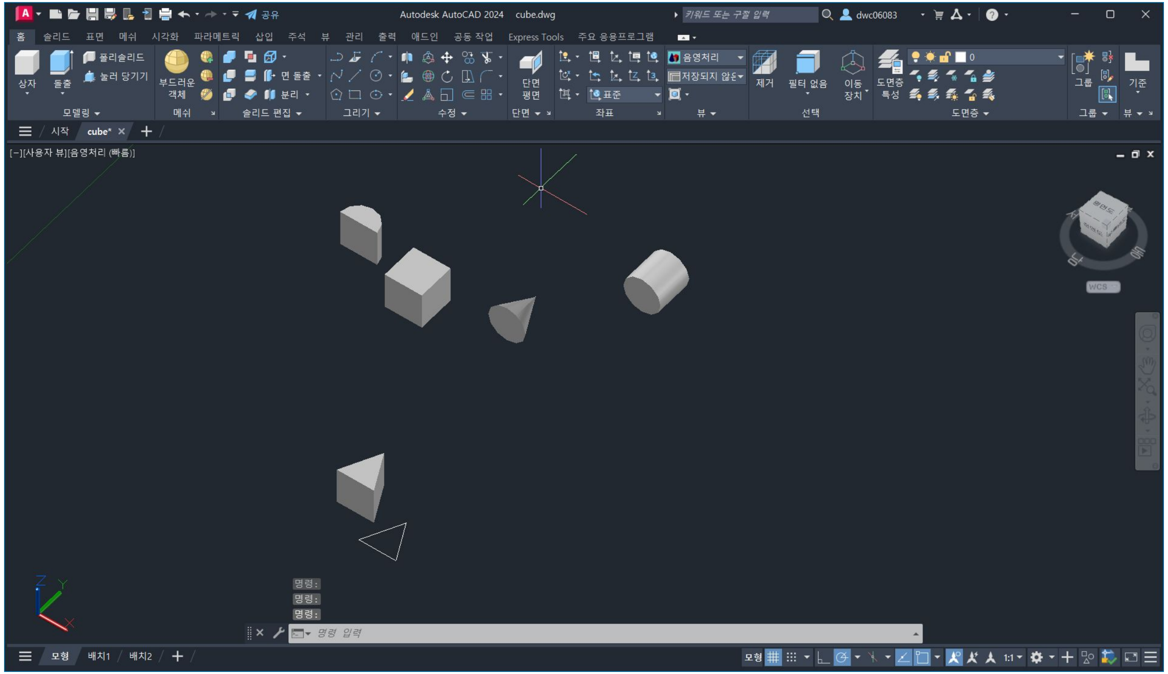
3. FEM with given fin mesh

More details at github repository.



https://github.com/jbw9964/FEM_2d.git

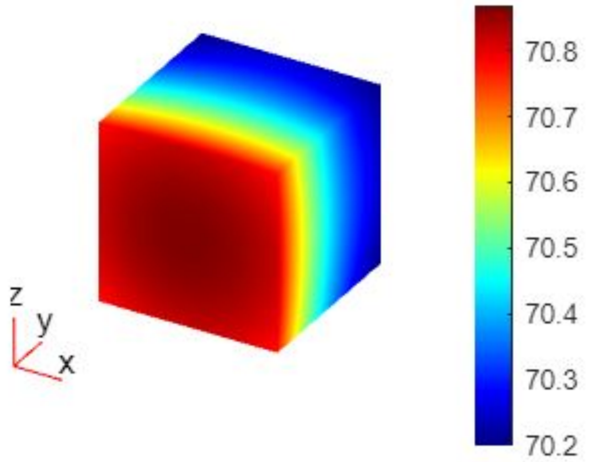
4. Modeling & make a video



Auto CAD를 이용한
structure 생성

매트랩 코드, cad
캡처처

```matlab
tmodel = createpde('thermal', 'steadystate');
importGeometry(tmodel, ['small_cube.stl']);
pdegplot(tmodel, 'FaceLabels', 'on', 'FaceAlpha', 0.5);
msh= generateMesh(tmodel, 'Hmax', 0.005);

% Material properties
kappa = 237; % In 2/m/K
thermalProperties(tmodel, 'ThermalConductivity', kappa);

% BC of air
thermalBC(tmodel, 'Face', [1 2 4:6], ...
    'ConvectionCoefficient', 20, ...
    'AmbientTemperature', 25);

% BC of contact surface
thermalBC(tmodel, 'Face', [3], ...
    'ConvectionCoefficient', 237, ...
    'AmbientTemperature', 90);

% solve and plot
Rt = solve(tmodel);
pdeplot3D(tmodel, 'ColorMapData', Rt.Temperature);

% calculate temperature
S = sum(Rt.Temperature, 'all');
sz = size(Rt.Temperature);
tot_sz = sz(1,1)*sz(1,2);
avg = S/tot_sz;

% [] operator를 이용해 여러 문자형 벡터 결합
% num2str로 숫자형값 -> 문자로 변환
x = ['Average Temperature: ', num2str(avg)];
disp(x)
%disp('average Temperature:')
%disp(avg)
```



Average Temperature: 70.4851

matlab을 이용한
steady state 구현

```matlab
tmodel = createpde('thermal', 'transient');
importGeometry(tmodel, ['small_cube.stl']);
pdegplot(tmodel, 'FaceLabels', 'on', 'FaceAlpha', 0.5);
msh= generateMesh(tmodel, 'Hmax', 0.01);

% Material properties
kappa = 237; % In 2/m/K

thermalProperties(tmodel, 'ThermalConductivity', ...
    kappa, 'MassDensity', 2700, 'SpecificHeat', 8.97);

% BC of air
thermalBC(tmodel, 'Face', [1 2 4:6], ...
    'ConvectionCoefficient', 20, ...
    'AmbientTemperature', 25);

% specify the stefan-boltzmann constant
tmodel.StefanBoltzmannConstant = 5.670367e-8;

% BC of contact surface
thermalBC(tmodel, 'Face', [3], ...
    'ConvectionCoefficient', kappa, ...
    'AmbientTemperature', 90);

% IC of model
thermalIC(tmodel, 25);

% solve and plot, choose time interval
Rt = solve(tmodel, 0:0.05:50);
%pdeplot3D(tmodel, 'ColorMapData', Rt.Temperature);

% cylinder.avi 생성
v = VideoWriter('cube');
open(v)

```

```matlab
for i = 1:length(Rt.SolutionTimes)
    figure
    pdeplot3D(tmodel, 'ColorMapData', Rt.Temperature(:,i));
    title({['Time=' num2str(Rt.SolutionTimes(i)) 's']})

    frame = getframe(gcf);
    writeVideo(v, frame);


end

close(v)

% temp 계산
S = sum(Rt.Temperature);
%sz = size(Rt.Temperature);
%tot_sz = sz(1,1)*sz(1,2);
l = length(Rt.Temperature);
avg = S/l;

x = ['Average Temperature: ', num2str(avg)];
disp(x)

%disp('average Temperature:')
%disp(avg)

```

matlab을 이용한
transient video 제작

5. Result

핀 유용도(Fin effectiveness)

핀이 없을 때보다 있을 때 얼마나 더 많은 열을 방출하는지에 대한 척도

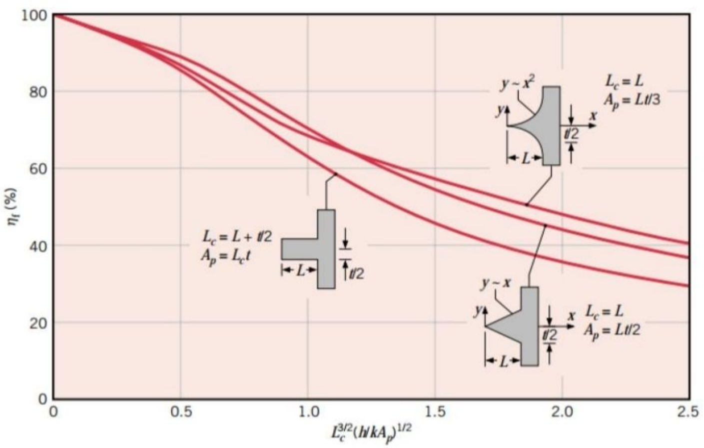
$$\varepsilon_f = \frac{q_f}{hA_c(T_b - T_\infty)}$$

핀 효율(Fin efficiency) : 핀이 얼마나 효율적인지에 대한 척도

$$\eta_f \equiv \frac{q_f}{q_{\max}} = \frac{q_f}{hA_f(T_b - T_\infty)}$$

q_{max} : κ = inf인 경우

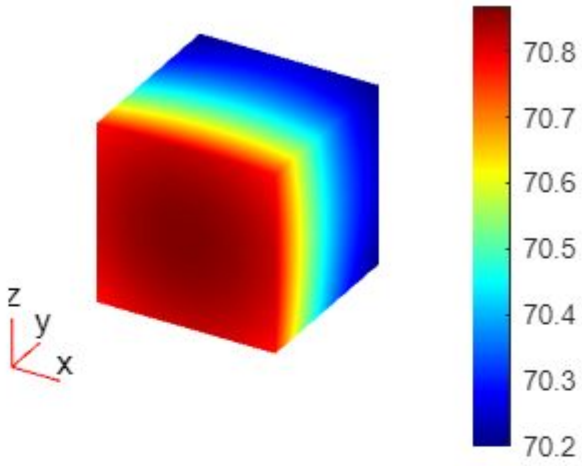
Efficiency of straight fins (rectangular, triangular, and parabolic profile)



3D results

steady state

transient video



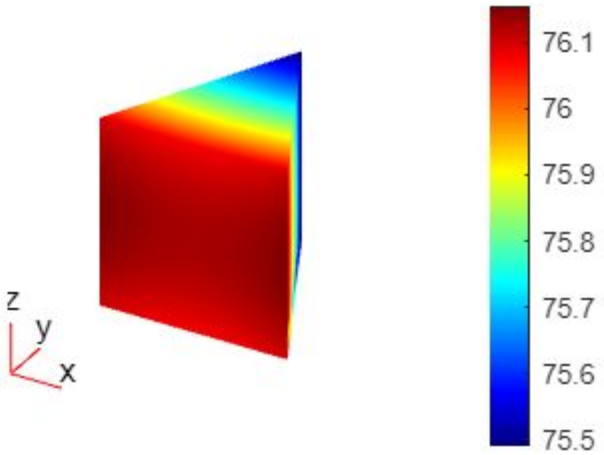
Average Temperature: 70.4851

η_f = 69.977%

ε_f = 3.499

steady state

transient video



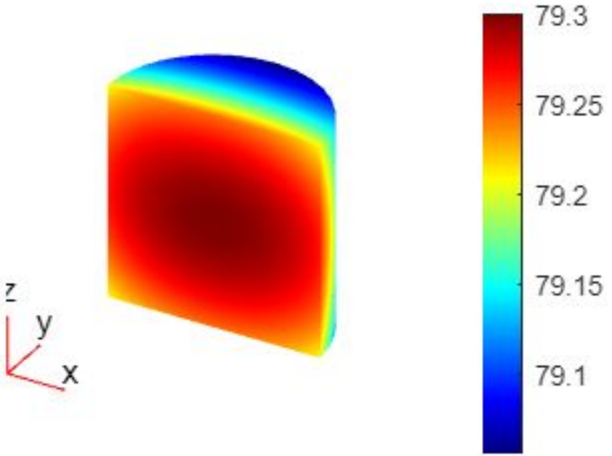
Average Temperature: 75.8988

η_f = 78.306%

ε_f = 5.068

steady state

transient video



Average Temperature: 79.1791

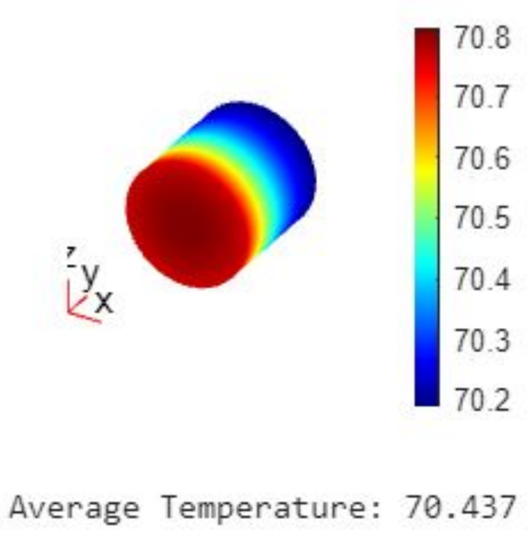
$\eta_f = 83.352\%$

$\varepsilon_f = 2.501$

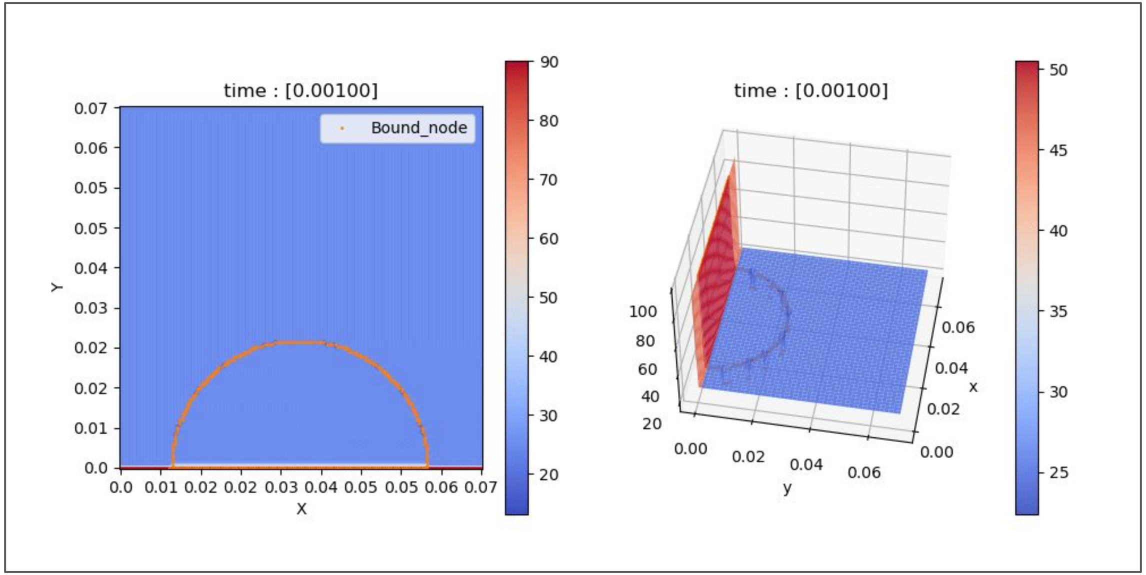
run-on experiment

steady state

transient video

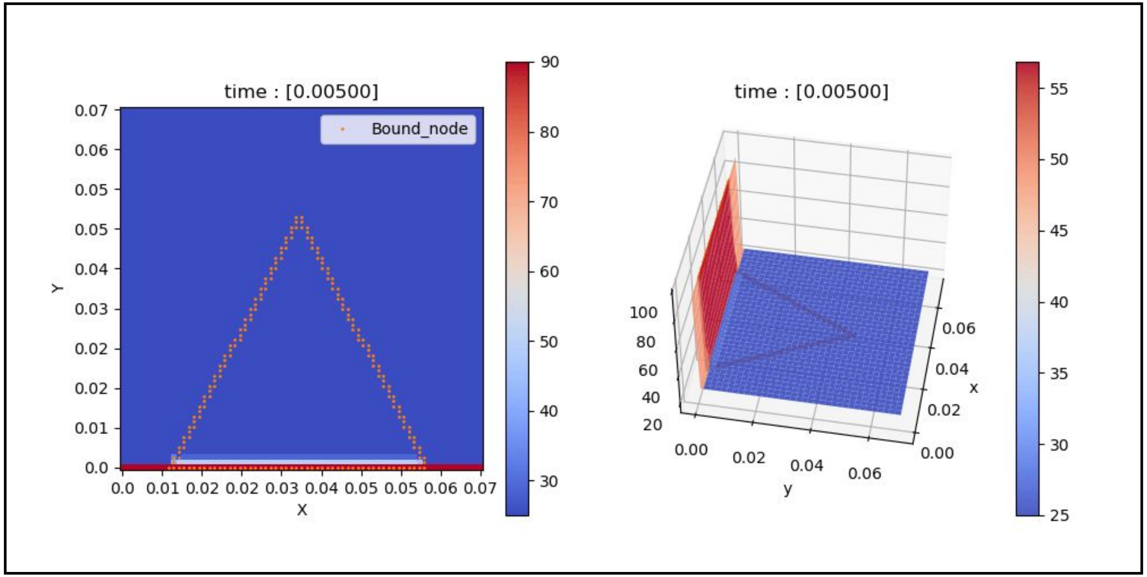


Half circle



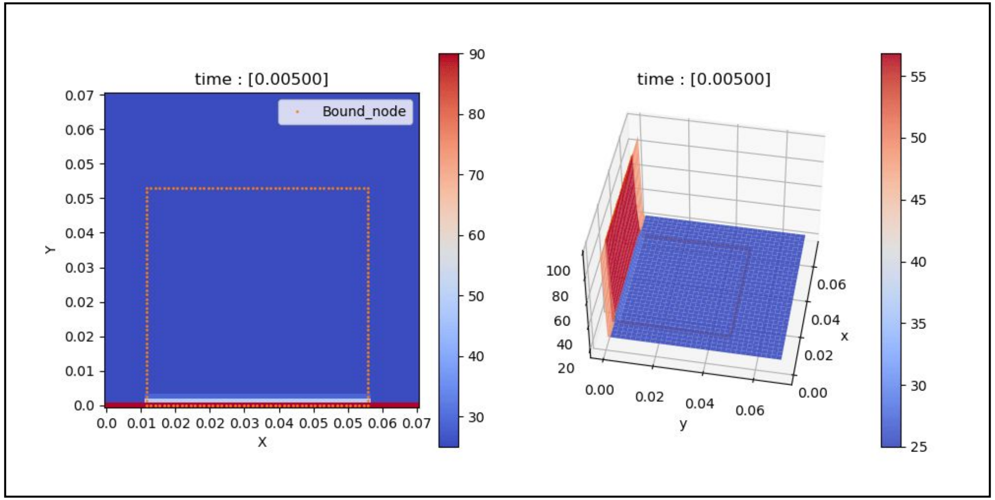
$T_f = 40.2091$

Triangle



T_f = 45.5461

Rectangle

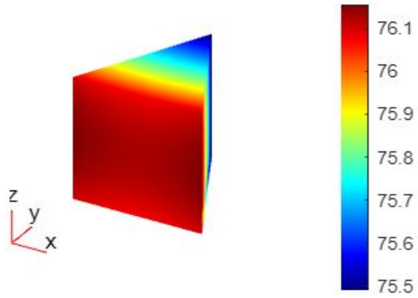


η_f = 91.567%

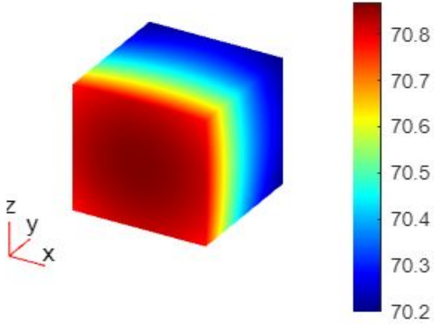
ε_f = 2.747

comparison of 2D & 3D

3D

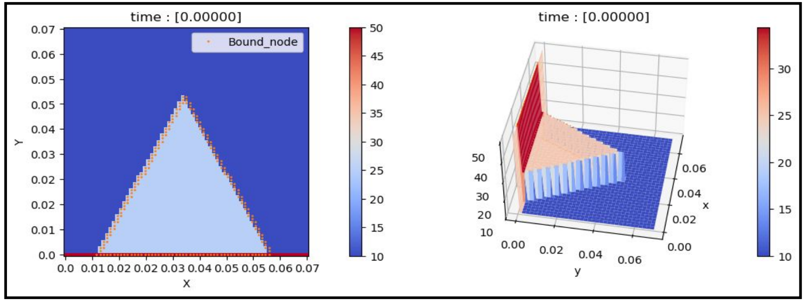


Average Temperature: 75.8988

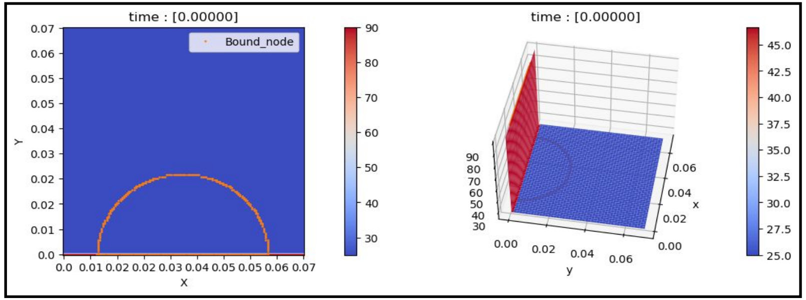


Average Temperature: 70.4851

2D



$T_f = 45.5461$



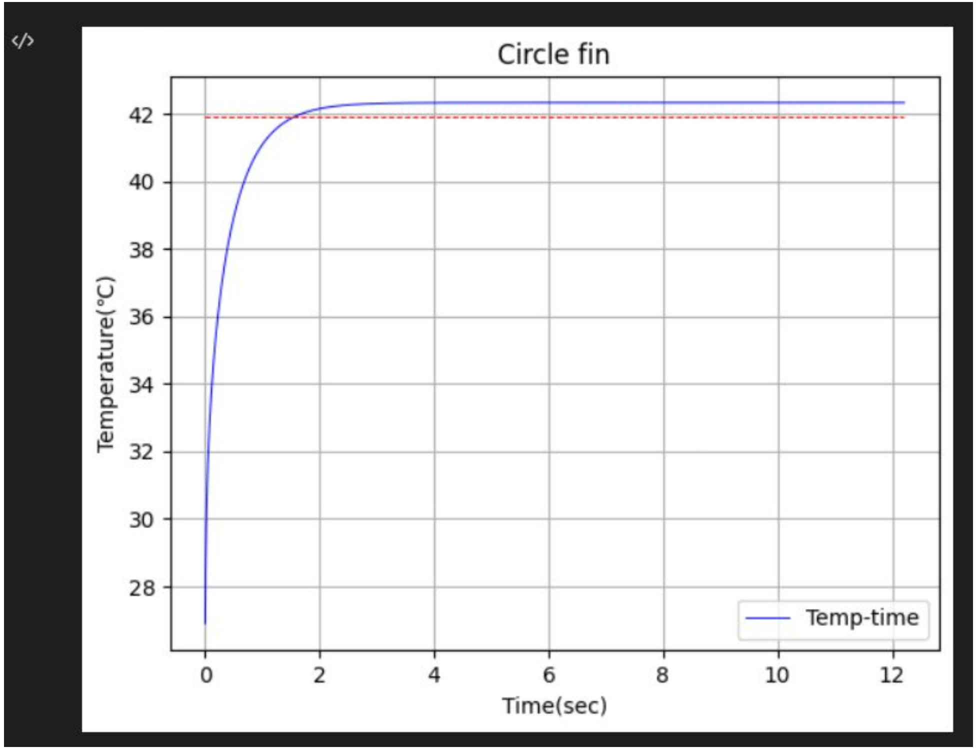
$T_f = 40.2091$

How to compare these?

```python
Circle = np.load('Circle_fin.npz')

for i in Circle:
    print(i)

print('process time:\n', Circle['process_time'])
print('process temp:\n', Circle['process_avg'])
print('t_mesh:\n', Circle['T_mesh'])

x_circle = Circle['process_time']
y_circle = Circle['process_avg']

l = y_circle[-1]*0.99
print('T_final =', y_circle[-1], '\nl =', l)

plt.title('Circle fin')
plt.plot(x_circle, y_circle, color = 'blue', label = 'Temp-time', linewidth = 0.7)
plt.hlines(xmin = min(x_circle), xmax = max(x_circle), y = l, color = 'red',
           linestyles = '--', linewidth = 0.7)
plt.xlabel('Time(sec)')
plt.ylabel('Temperature(℃)')
plt.grid()
plt.legend()
plt.show()
```

[33] ✓ 0.1s Python

```
process_time
process_avg
T_mesh
process time:
 [5.00000e-04 1.00000e-03 1.50000e-03 ... 1.22090e+01 1.22095e+01
 1.22100e+01]
process temp:
 [26.89526888 27.10389247 27.28106995 ... 42.32534788 42.32534788
 42.32534788]
t_mesh:
 [[90. 25. 25. ... 25. 25. 25.]
 [90. 25. 25. ... 25. 25. 25.]
 [90. 25. 25. ... 25. 25. 25.]
 ...
 [90. 25. 25. ... 25. 25. 25.]
 [90. 25. 25. ... 25. 25. 25.]
 [90. 25. 25. ... 25. 25. 25.]]
T_final = 42.32534788063744
l = 41.902094401831064
```



```
l = 41.902094401831064
```

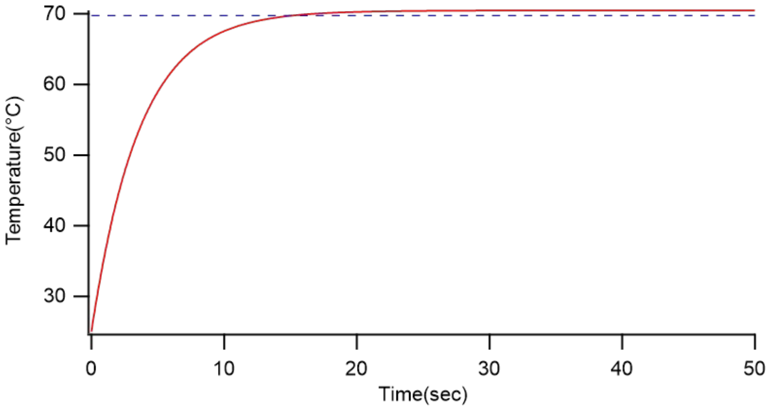
```python
df = DataFrame({'Circle_time': x_circle, 'Circle_temp': y_circle})
#df2= DataFrame({'Circle_temp': y_circle})

circle_Writ = pd.ExcelWriter('circle_fin.xlsx', engine = 'xlsxwriter')
df.to_excel(circle_Writ, sheet_name = 'Sheet1')
#df2.to_excel(circle_Writ, sheet_name = 'Sheet1')

workbook = circle_Writ.book
worksheet = circle_Writ.sheets['Sheet1']

chart = workbook.add_chart({'type':'column'})
#chart.add_series({''})

circle_Writ.close()
```

[105]

3102	**3100**	1.5505	41.89982
3103	**3101**	1.551	41.90023
3104	**3102**	1.5515	41.90064
3105	**3103**	1.552	41.90105
3106	**3104**	1.5525	41.90146
3107	**3105**	1.553	41.90187
3108	**3106**	1.5535	41.90228
3109	**3107**	1.554	41.90269
3110	**3108**	1.5545	41.90309
3111	**3109**	1.555	41.9035
3112	**3110**	1.5555	41.90391

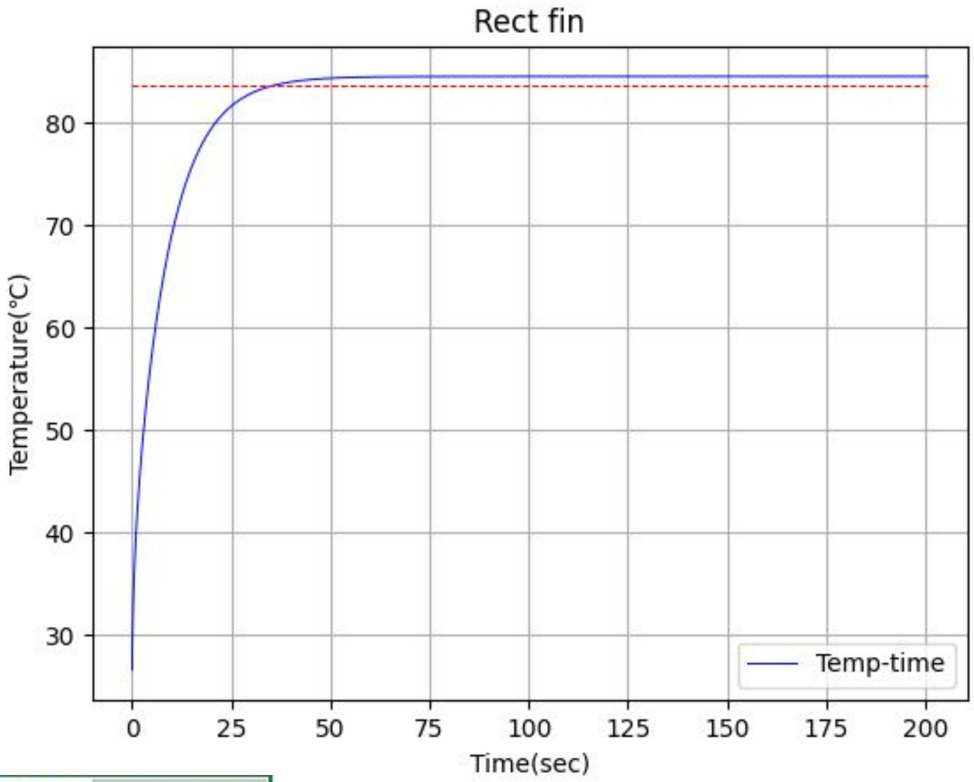
Cube(rectangle)

3D



t = 15.2sec

2D

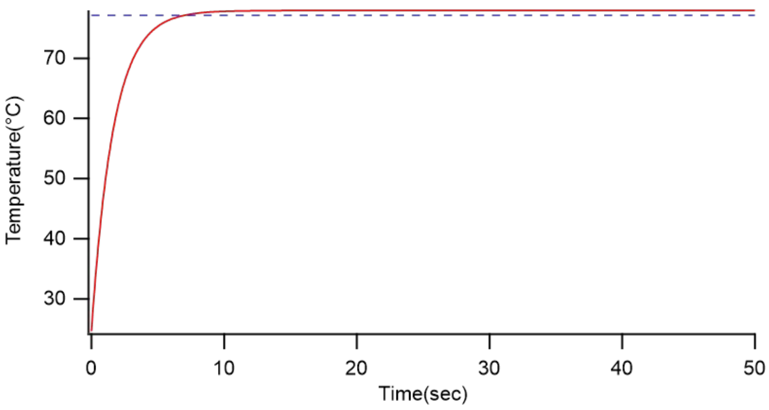


36.0475	83.67328
36.05	83.67351

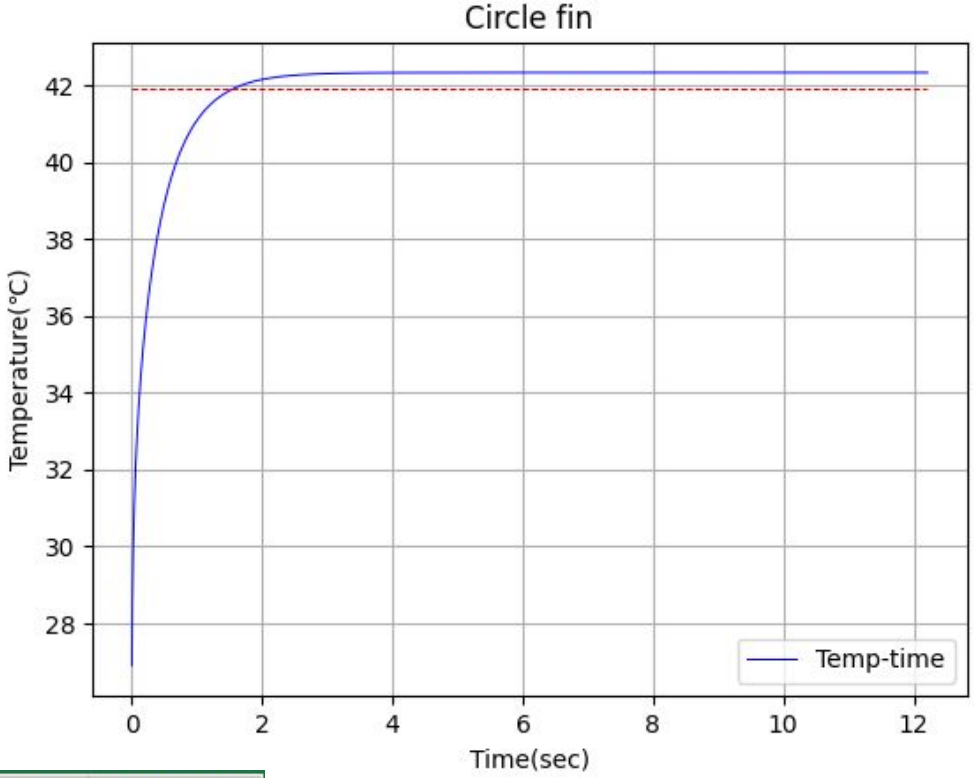
t = 36.0475sec

Half cylinder(circle)

3D



2D



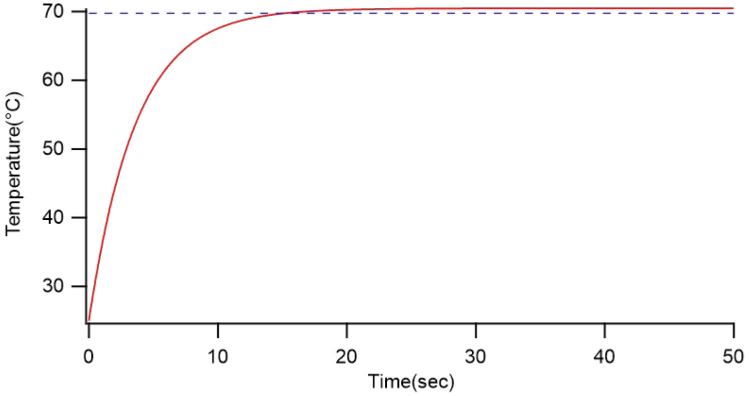
t = 7.05sec

1.553	41.90187
1.5535	41.90228

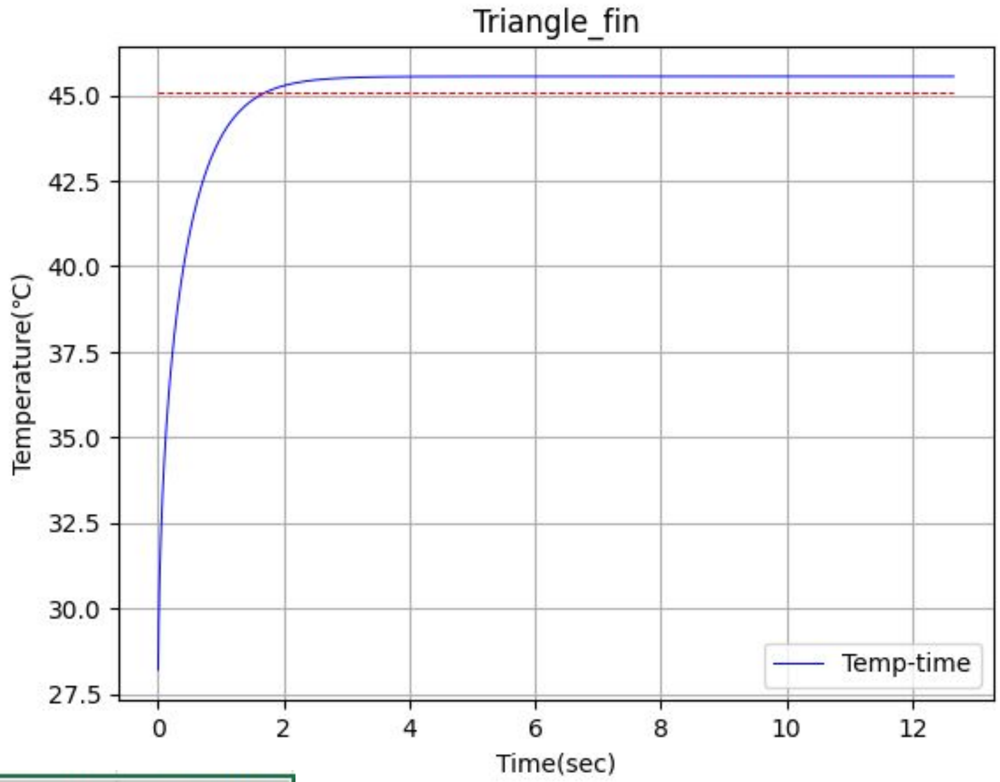
t = 1.553sec

Sandwich(triangle)

3D



2D

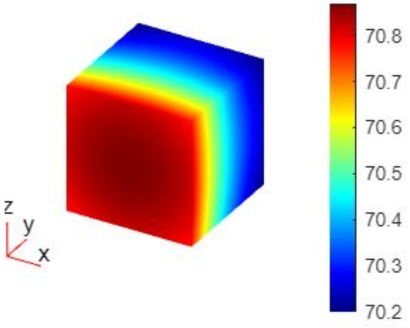


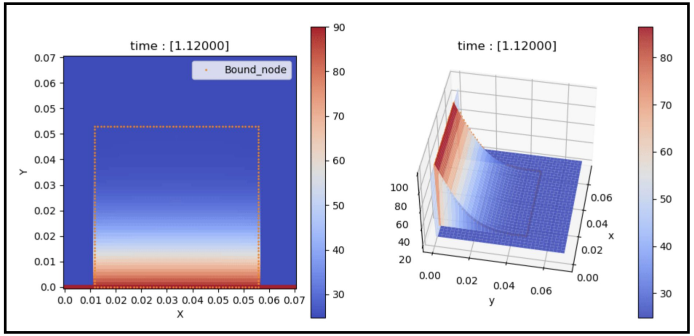
t = 8.55sec

1.725	45.08938
1.7275	45.09152

t = 1.725sec

6. Analysis





in 3D	error	in 2D
$\eta_f = 69.977\%$	30.853%	$\eta_f = 91.567\%$
$\varepsilon_f = 3.499$	21.492%	$\varepsilon_f = 2.747$
$T_f = 70.4851$	19.910%	$T_f = 84.51859351$

Q&A