Fin Heat Transfer Equation Analyze and Simulation

B08611019 林大衛
B08611031 易峻葦
B08611035 柯鉑霆

Outline

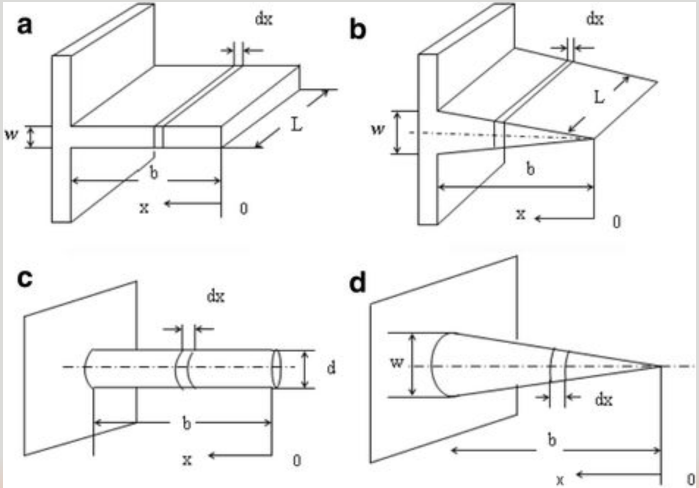
01 **Equation Analysis**

02 **Solidworks Simulation**

03 **Conclusions**

Brief Intro

- Try to design fin in different cross areas (Ac) and profiles (Ac(x))
- Compare the temperature and heat flux between different shape of fins



01 Equation Analysis

Equation derivation

- conservation of energy

$$q_x = q_{x+dx} + dq_{conv}$$

- convection

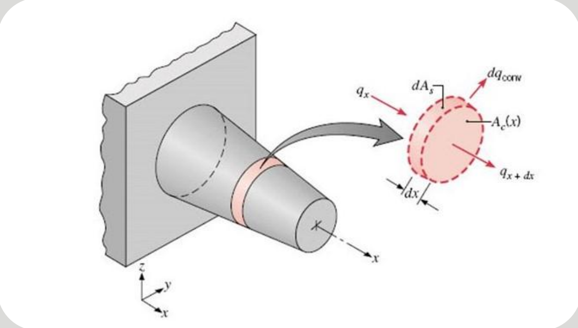
$$dq_{conv} = hdA_s(T - T_\infty) = hPd_x(T - T_\infty)$$

- Fourier's Law

$$q_x = -kA_c\frac{dT}{dx}$$

$$q_{x+dx} = q_x + \frac{dq_x}{dx}dx$$

$$= -kA_c\frac{dT}{dx} - k\frac{d}{dx}(A_c\frac{dT}{dx})dx$$



$$\frac{d}{dx}(A_c\frac{dT}{dx}) - \frac{hP}{k}(T - T_\infty) = 0$$

$$\Rightarrow \frac{d^2T}{dx^2} + (\frac{1}{A_c}\frac{dA_c}{dx})\frac{dT}{dx} - \frac{hP}{kA_c}(T - T_\infty) = 0$$

Drive the general equation in state space form due to the scipy applied

The general fin equation is given by

$$\frac{d^2\theta}{dx^2} + \left(\frac{1}{A_c}\frac{dA_c}{dx}\right)\frac{d\theta}{d_x} - \frac{hP}{kA_c}\theta = 0$$

where $\theta(x) \equiv T(x) - T_\infty$

The state-space form,

$$\frac{dy}{dx} = \begin{bmatrix} y_0' \\ y_1' \\ y_2' \\ y_3' \\ y_4' \end{bmatrix} = \begin{bmatrix} \theta' \\ \theta'' \\ A_c \\ A_c' \\ A_c'' \end{bmatrix} = \begin{bmatrix} y_1 \\ -\frac{y_4}{y_3}y_1 + \frac{hP}{ky_3}y_0 \\ y_3 \\ y_4 \\ \dots \end{bmatrix}$$

$$y_0 = \theta(x)$$

$$y_1 = \theta'(x) = \frac{d\theta}{dx}$$

$$y_2 = V(x) = \int_0^x A_c\,dx$$

$$y_3 = A_c(x)$$

$$y_4 = A_c'(x) = \frac{dA_c}{dx}$$

where for a polygons, $P = k\sqrt{A_c}$

Parameter Setup

```python
k: Final[float] = 3e-2        # W mm-1 K-1
h: Final[float] = 25e-6        # W mm-2 K-1 Aluminum Alloy
r: Final[float] = 5e-1       # mm
L: Final[float] = 5e0        # mm
T_b: Final[float] = 398.0    # K
T_inf: Final[float] = 298.0  # K
```

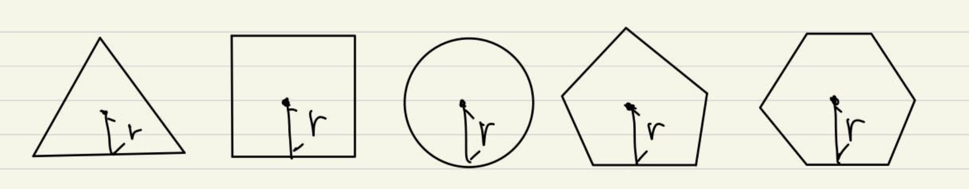
- r :
  The radius for circle area (We what to let heat sinks have **same volumes**、Ac different P)。
- k: alluminum Alloy (300 W/mk)
- h: natural convection air
- T(base) : 398 K
- T(surr): 298 K

Parameter Setup

Pin Geomtry condition of fin

- **Cross Section Area(Ac)**:
  Circle
  EquilateralTriangle
  Square
  RegularPentagon
  RegulareHexagon

- **Axial profile**(Ac(x))
  1. Uniform : r =constant
  2. Linear : $r = r_0(1 - \frac{x}{L})$
  3. Parabolic : $r = r_0(1 - \frac{x^2}{L^2})$
  4. Cosine : $r = r_0 cos(\frac{\pi x}{2L})$



Code Implementation

Model and solving state deriving

State-space form

```python
def deriv(self, x: np_arr_f64, y: np_arr_f64) -> np_arr_f64:
    y0, y1, _, y3, y4 = y
    return np.vstack(
        [
            y1,
            -y4 * y1 / y3
            + (self.h * self.cross_section.P(np.abs(y3))) / (self.k * y3) * y0,
            y3,
            y4,
            self.profile.d2Ac_dx2(x, y),
        ]
    )
```

solving with scipy

```python
@final
def solve(
    self,
    bc: Callable[[np_arr_f64, np_arr_f64], np_arr_f64],
) -> PPoly:
    x = np.linspace(0, self.L, 5)
    y = np.ones((5, x.size))

    self.res = scipy.integrate.solve_bvp(
        fun=self.deriv, bc=bc, x=x, y=y, max_nodes=100000
    )
    return self.res.sol
```

Code Implementation

Boundary Conditions ya(x=0) / yb(x=L)

- thermal : uniform / linear
- geometry: fin volume constant / the cross section area on tip

```
def bc_linear(ya, yb):
    return np.array(
        [
            # Temperature BCs
            ya[0] - (T_b - T_inf),  # θ(0) = T_b - T_inf
            yb[1],  # θ'(L) = 0; adiabatic tip
            # Geometry BCs
            ya[2],  # V(0) = 0
            yb[2] - np.pi * r**2 * L,  # V(L) = πr²L
            yb[3] - 1e-12,  # Ac(L) = 0; reduces to a point
        ]
    )
```

```
def bc_uniform(ya, yb):
    return np.array(
        [
            # Temperature BCs
            ya[0] - (T_b - T_inf),  # θ(0) = T_b - T_inf
            h * yb[0] + k * yb[1],  # θ'(L) = 0; adiabatic tip
            # Geometry BCs
            ya[2],  # V(0) = 0
            ya[3] - np.pi * r**2,  # Ac(0) = πr²
            yb[3] - np.pi * r**2,  # Ac(L) = πr²
        ]
    )
```

Code Implementation

Cross section and Profile models

- Define the Relation between Suface Area **Ac** and **P**
- For RegularPolygon: $P = 2\sqrt{Ac * ntan(\frac{\pi}{n})}$

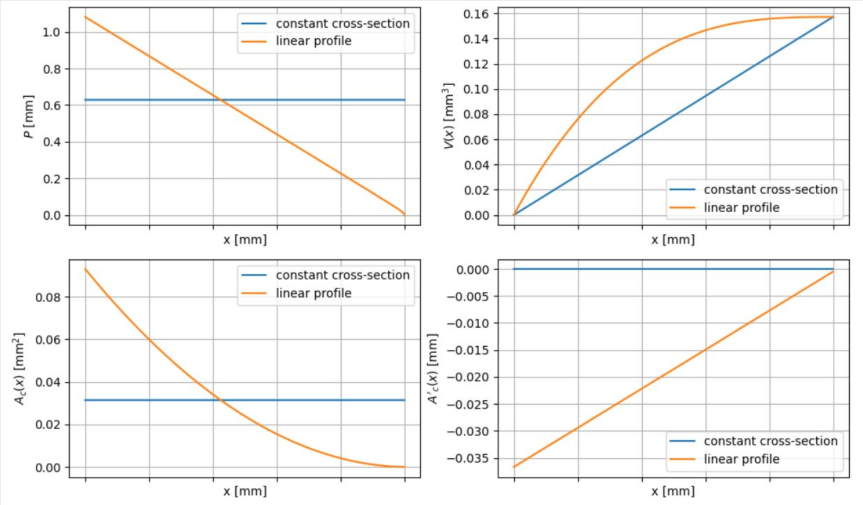
```python
class CircleCrossSection(CrossSection):
    def P(self, Ac: np_arr_f64) -> np_arr_f64:
        return 2 * np.sqrt(np.pi * np.abs(Ac))


class RectangleCrossSectionMeta(type):
    def __init__(self, name: str, w: float) -> None:
        super().__init__(self)

    def __new__(cls, name: str, w: float) -> type:
        kls = super().__new__(cls, name, (), {"w": w, "P": RectangleCrossSectionMeta.P})
        return kls

    @staticmethod
    def P(self, Ac: np_arr_f64) -> np_arr_f64:
        return np.full_like(Ac, 2 * self.w)
```

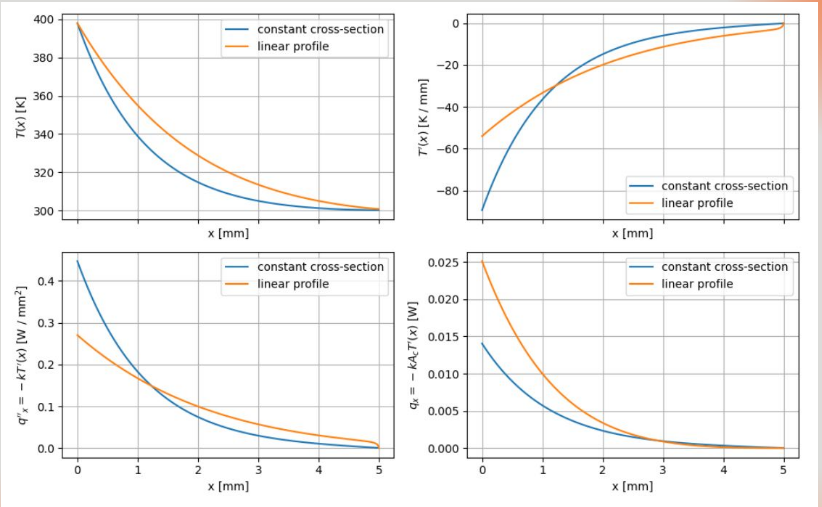
```python
class RegularPolygonCrossSectionMeta(type):
    def __init__(self, name: str, n: int) -> None:
        super().__init__(self)

    def __new__(cls, name: str, n: int) -> type:
        kls = super().__new__(
            cls, name, (), {"n": n, "P": RegularPolygonCrossSectionMeta.P}
        )
        return kls

    @staticmethod
    def P(self, Ac: np_arr_f64) -> np_arr_f64:
        return 2 * np.sqrt(self.n * np.tan(np.pi / self.n) * Ac)


EquilateralTriangleCrossSection = RegularPolygonCrossSectionMeta(
    "EquilateralTriangleCrossSection", 3
)
SquareCrossSection = RegularPolygonCrossSectionMeta("SquareCrossSection", 4)
RegularPentagonCrossSection = RegularPolygonCrossSectionMeta(
    "RegularPentagonCrossSection", 5
)
RegulareHexagonCrossSection = RegularPolygonCrossSectionMeta(
    "RegulareHexagonCrossSection", 6
)
```

Code Implementation

Plotting the results

```python
x_plot = np.linspace(0, L, 100001)
sol_uniform = CircularUniformPinFin(k, h, L, T_b, T_inf).solve(bc_uniform)(x_plot)
linear = CircularLinearPinFin(k, h, L, T_b, T_inf)
sol_linear = linear.solve(bc_linear)(x_plot)
```

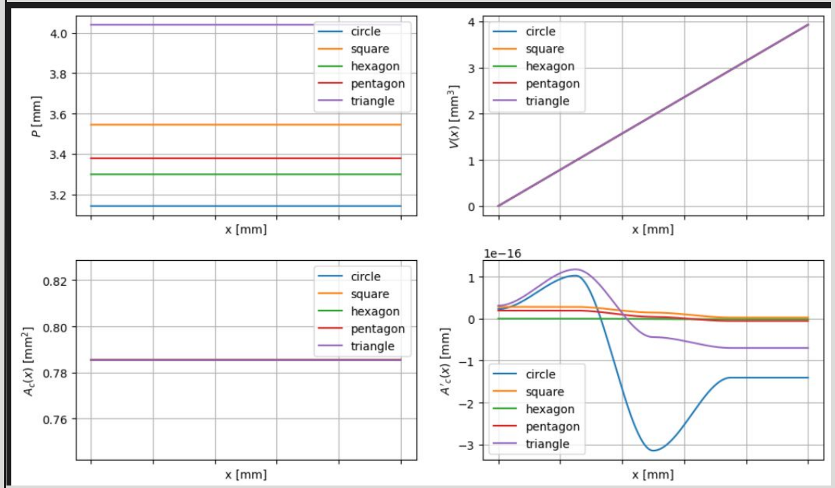


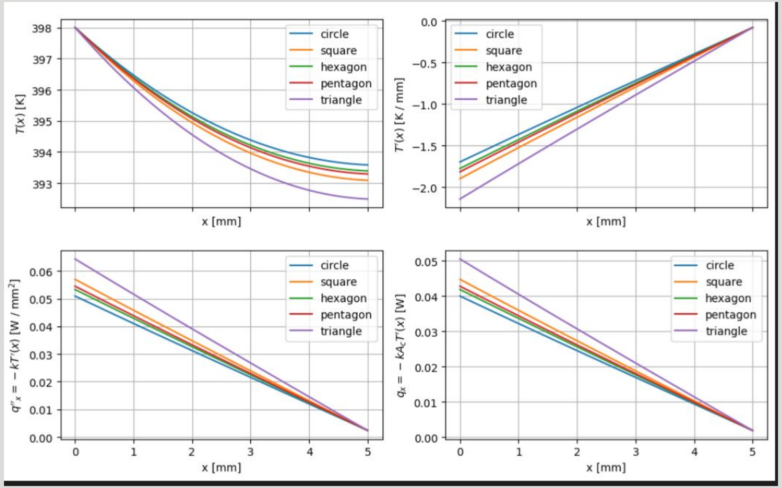


Case Analysis

Case1: Different cross section in uniform profile (volume constant)

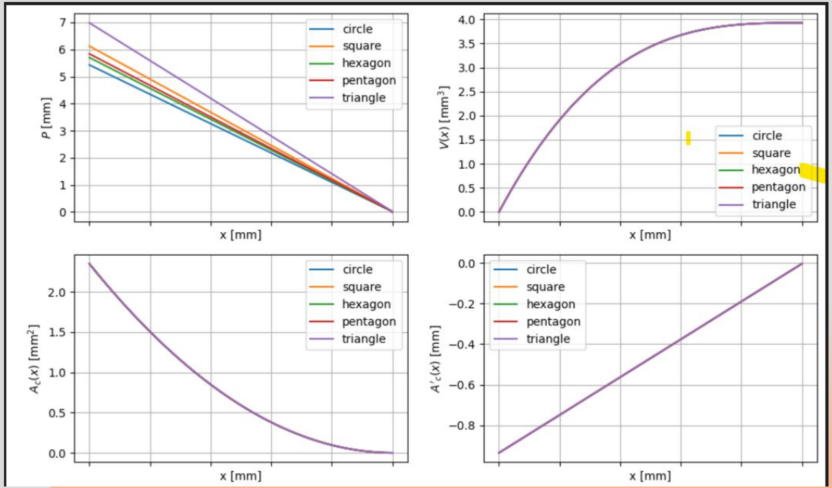
- Testing on circle, square, triangle, pentagon and hexagon cross section

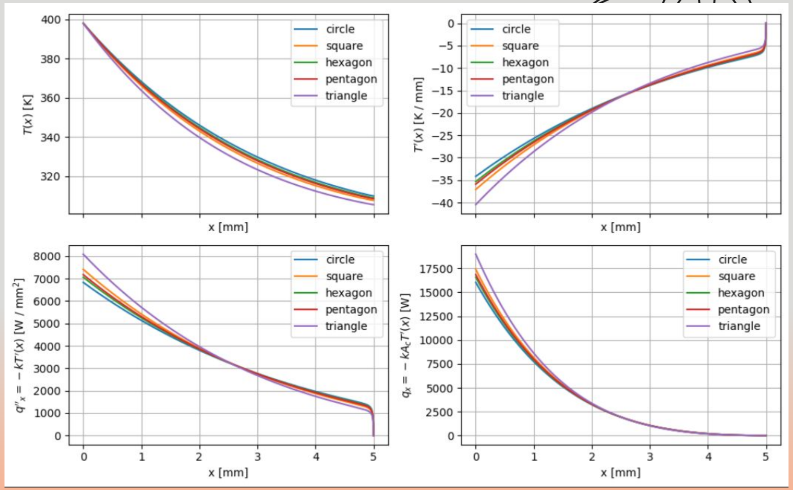




Case2: Different cross section in linear profile (volume constant)

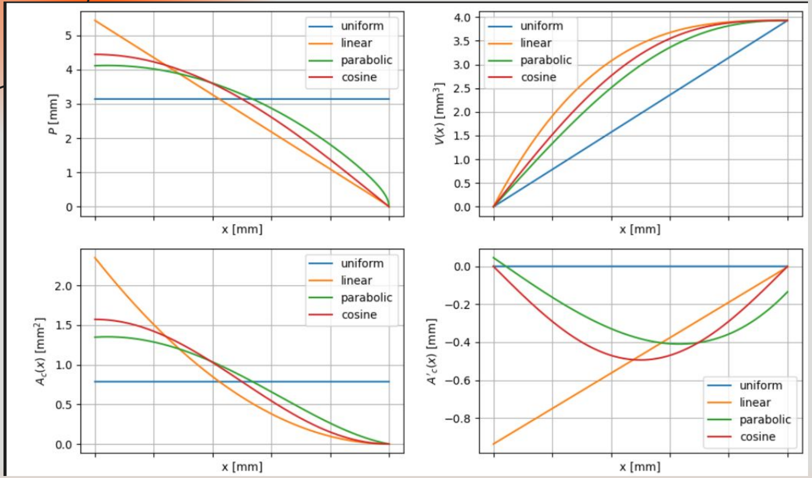
- Testing on circle, square, triangle, pentagon and hexagon cross section

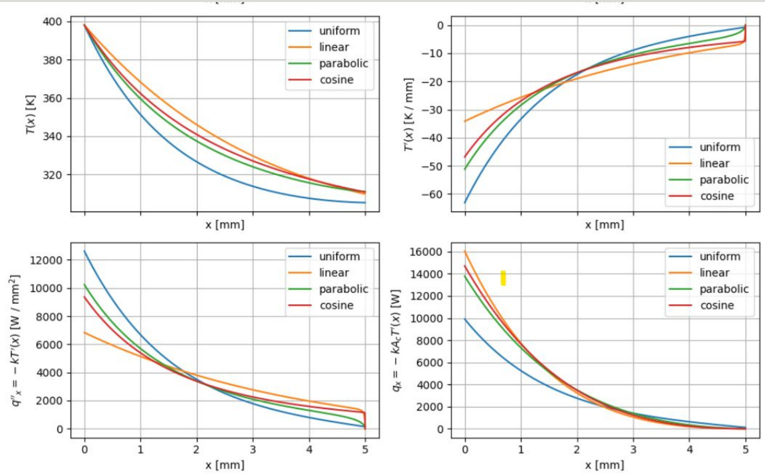




Case3: Different profile in circle cross section

- Testing on uniform, linear, parabolic amd cosine profile

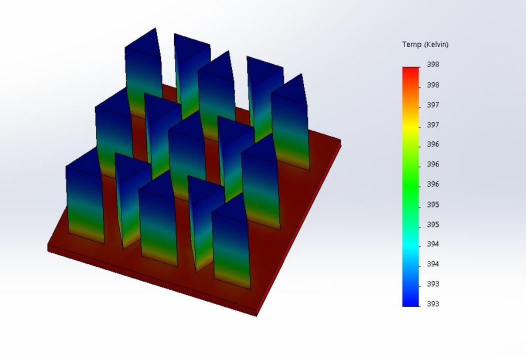


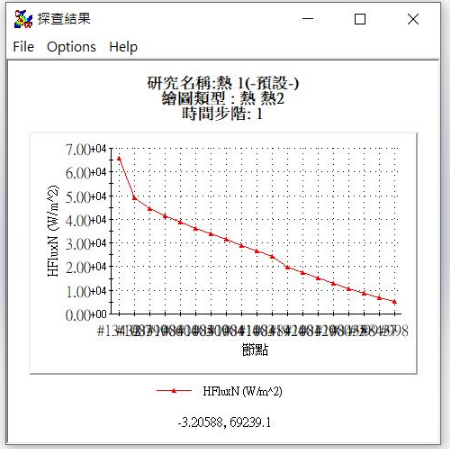


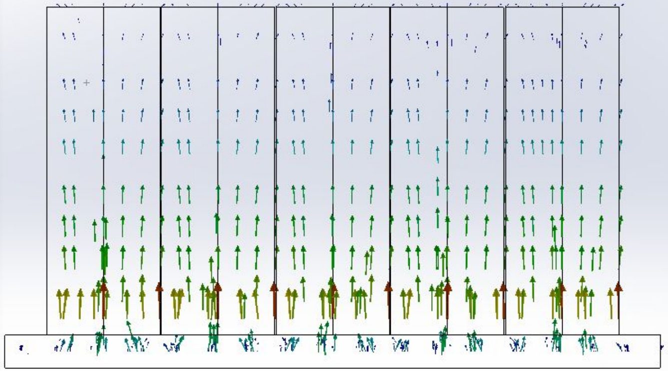
02 Solidworks Simulation

Case: Uniform Profile simulation

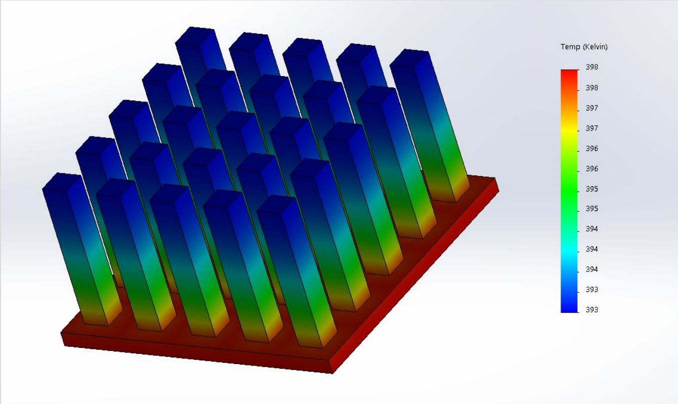
Triangular Cross section

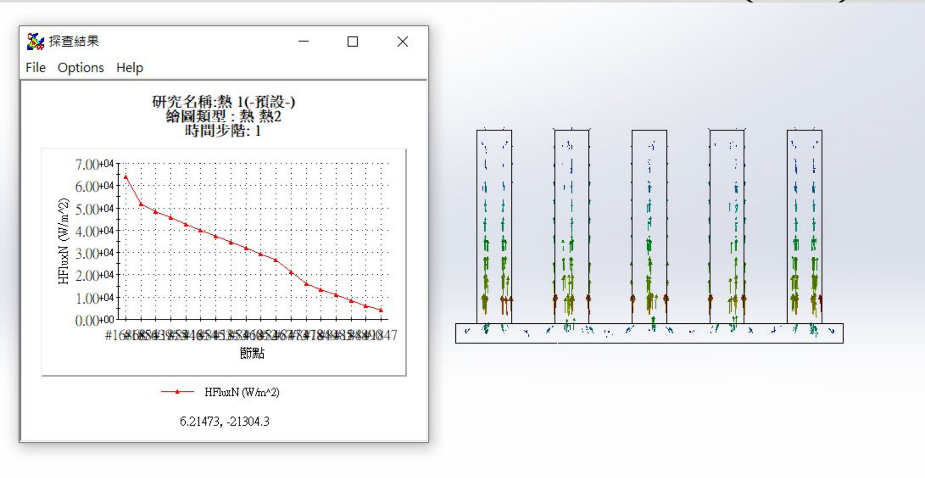




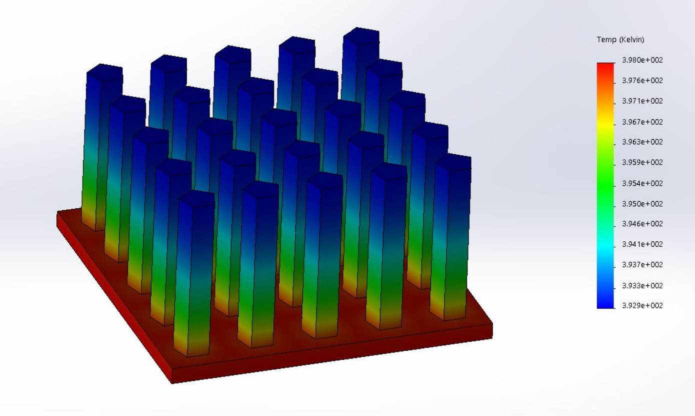


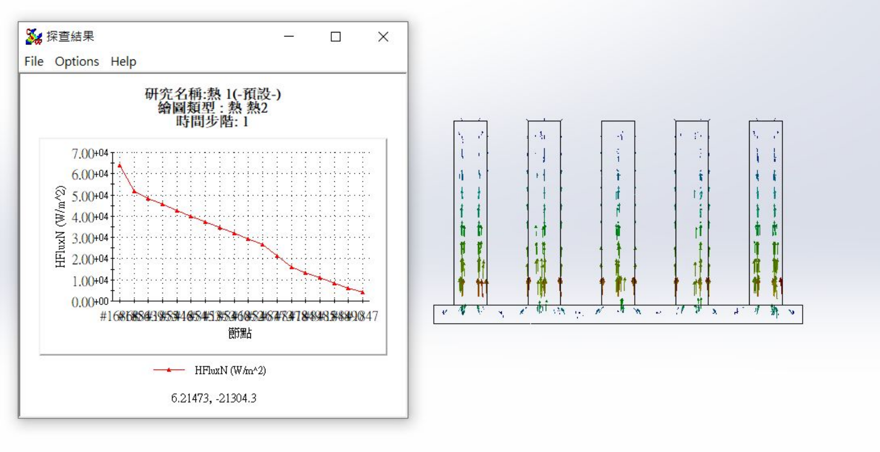
Square Cross section



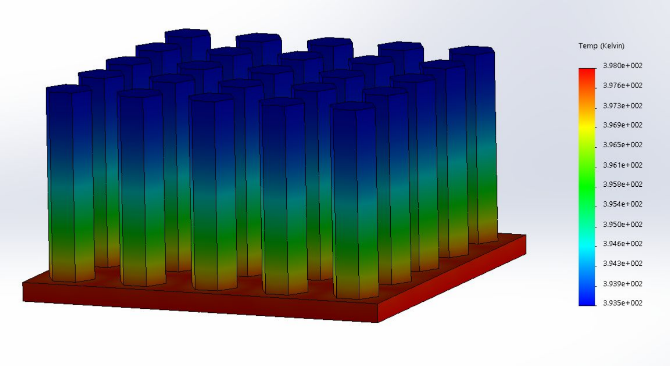


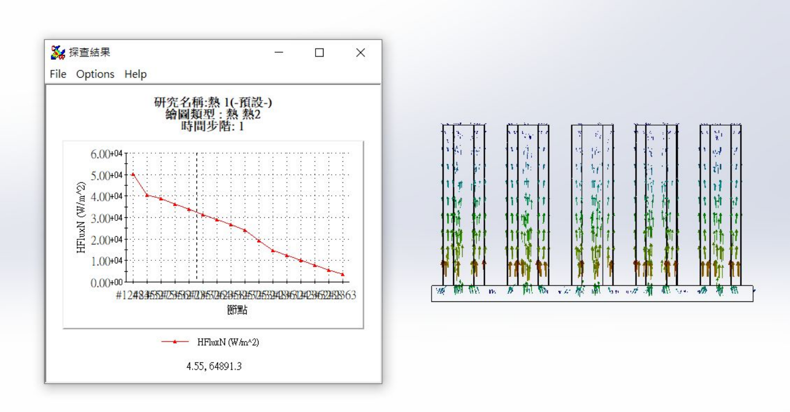
Pengaton Cross section





Hexagon Cross section





03 Conclusion

	triangle	square	circle	hexagon	pengaton
qx(w)	17620	17551	16524	16642	16742

- As the table shown above, we find that the **triangular** cross section fin perform best in linear profile and constant volume

- As x become longer, the efficiency of heat fin becomes worse

github link:
https://github.com/dvnatanael/fin-equation-analysis/tree/david-scipy

Thanks