

AERO 430 HW 2

Judson Upchurch

2024-09-27

Table of Contents

1	Analytical Solution	4
1.1	Abstract	4
1.2	Derivation of General Solution	4
1.2.1	Conservation of Heat Flux	4
1.2.2	Newton's Law of Cooling	4
1.2.3	Fourier's Law of Conduction	4
1.2.4	General Solution	5
1.3	Case 1: Dirichlet Boundary Condition	6
1.4	Case 2: Neumann Boundary Condition	7
1.5	Temperature Results	7
1.5.1	Case 1	8
1.5.2	Case 2	9
1.5.3	Case Comparison	10
1.5.4	Discussion	10
1.6	Heat Flux Results	11
1.6.1	Calculating Heat Flux	11
1.6.2	Case Comparison	11
1.6.3	Discussion	12
2	Finite Difference Method	12
2.1	Abstract	12
2.2	Case 1: Dirichlet Boundary Condition	13
2.3	Case 2: Neumann Boundary Condition	13
2.4	Temperature Results	14
2.4.1	Case 1	14
2.4.2	Case 2	15
2.4.3	Case Comparison	17
2.4.4	Discussion	17
2.5	Heat Flux Results	17
2.5.1	Calculating Heat Flux	17
2.5.2	Case Comparison	18
2.5.3	Discussion	20
3	Finite Element Method	20
3.1	Abstract	20
3.2	Case 1: Dirichlet Boundary Condition	21
3.3	Case 2: Neumann Boundary Condition	21
3.4	Temperature Results	22
3.4.1	Case 1	22
3.4.2	Case 2	23
3.4.3	Case Comparison	25
3.4.4	Discussion	25
3.5	Heat Flux Results	26

3.5.1	Calculating Heat Flux	26
3.5.2	Case Comparison	26
3.5.3	Discussion	27
4	Convergence of FDM and FEM	27
4.1	Abstract	27
4.2	Calculating Rate of Convergence	28
4.3	Temperature At Midpoint Results	28
4.3.1	Case 1	28
4.3.2	Case 2	29
4.4	Heat Flux Results	30
4.4.1	Case 1	30
4.4.2	Case 2	31
4.5	Discussion	32
5	Python Code	33
5.1	main.py	33
5.2	case1.py	52
5.3	case2.py	53
5.4	common.py	54

1 Analytical Solution

1.1 Abstract

Solving for the temperature distribution throughout a rod and the heat flux to the environment depends heavily on the boundary conditions applied to the problem. A comparison between the Dirichlet boundary condition and Neumann boundary condition will be made over a range of specific conditions. This changing condition represents the ratio between heat flux leaving to the surroundings and the heat flux through the rod per dx along the rod.

1.2 Derivation of General Solution

Let us first look at derivation for the general solution for temperature as a function along the rod. To begin, the governing equation will be conservation of heat flux.

1.2.1 Conservation of Heat Flux

In the case of a rod without internal heat generation, conservation of heat flux reduces to the total heat flux entering the system is equal and opposite to the heat flux leaving the system.

$$\dot{Q}_{out}(x) - \dot{Q}_{in}(x) = 0 \quad (1)$$

$\dot{Q}$ is the rate of heat transfer in energy / second units. We will use Joules/second, aka Watts.

1.2.2 Newton's Law of Cooling

Newton's Law of Cooling shows that the rate of heat transfer to the surroundings is proportional to the area of heat transfer and the temperature difference between the object and the surroundings.

$$\dot{Q}_{env}(x) = hA(T(x) - T_{amb}) \quad (2)$$

$\dot{Q}_{env}(x)$ is the rate of heat flux to the environment at a point along the rod in Joules/second. h is the heat transfer coefficient of the environment's fluid in $\frac{W}{m^2 \cdot ^\circ C}$. A is the area exposed to the surroundings in m^2 . $T(x)$ is the temperature of the rod at position x in $^\circ C$.

In both of our cases, the ambient temperature, T_{amb} is $0^\circ C$. This reduces the equation for Newton's Law of Cooling to:

$$\dot{Q}_{env}(x) = hAT(x) \quad (3)$$

1.2.3 Fourier's Law of Conduction

Fourier's Law of Conduction states that the rate of heat transfer through a material via conduction is proportional to the area of heat transfer and the negative temperature gradient.

$$\dot{Q}_c(x) = -kA \frac{dT(x)}{dx} \quad (4)$$

$\dot{Q}_c(x)$ is the rate of heat transfer via conduction through the material in Joules/second. k is the thermal

conductivity of the rod in $\frac{W}{m^{\circ}C}$. A is the cross-sectional area of the rod in m^2 . $\frac{dT(x)}{dx}$ is the temperature gradient of the rod in $\frac{^{\circ}C}{m}$.

1.2.4 General Solution

If we look at an infinitesimal slice of the rod starting at position x and total length dx , we use conservation of heat flux (Equation 1) to get:

$$\dot{Q}(x+dx) - \dot{Q}(x) + \dot{Q}_{lat}(x; dx) = 0$$

From left to right, we have the heat flux leaving the right side of the rod slice, the heat flux entering the left side of the rod slice, and the heat flux leaving the rod slice from the outer circumference to the environment. Replacing $\dot{Q}_{lat}(x; dx)$ with $hPdxT(x)$ from Newton's Law of Cooling (Equation 3) gives:

$$\dot{Q}(x+dx) - \dot{Q}(x) + h(Pdx)T(x) = 0$$

Pdx is the area exposed to the surroundings for a given length dx where P is the perimeter. Dividing the equation by dx gives us:

$$\frac{\dot{Q}(x+dx) - \dot{Q}(x)}{dx} + hPT(x) = 0$$

Noting that $\frac{\dot{Q}(x+dx) - \dot{Q}(x)}{dx} = \dot{Q}'(x)$ as $dx \rightarrow 0$, the equation becomes:

$$\dot{Q}'(x) + hPT(x) = 0$$

Replacing $\dot{Q}(x)$ with $-kAT'(x)$ from Equation 4 gives:

$$-kAT''(x) + hPT(x) = 0$$

Rearranging and letting $\alpha = \sqrt{\frac{hP}{kA}}$ gives:

$$T''(x) - \alpha^2 T(x) = 0 \tag{5}$$

This will be the governing differential equation for both cases. Solving Equation 5 using the characteristic equation $r^2 - \alpha^2 = 0$ gives:

$$T(x) = Ae^{\alpha x} + Be^{-\alpha x}$$

Using the definitions $\sinh(\omega x) = \frac{e^{\omega x} - e^{-\omega x}}{2}$ and $\cosh(\omega x) = \frac{e^{\omega x} + e^{-\omega x}}{2}$, we get:

$$T(x) = C \sinh(\alpha x) + D \cosh(\alpha x) \tag{6}$$

1.3 Case 1: Dirichlet Boundary Condition

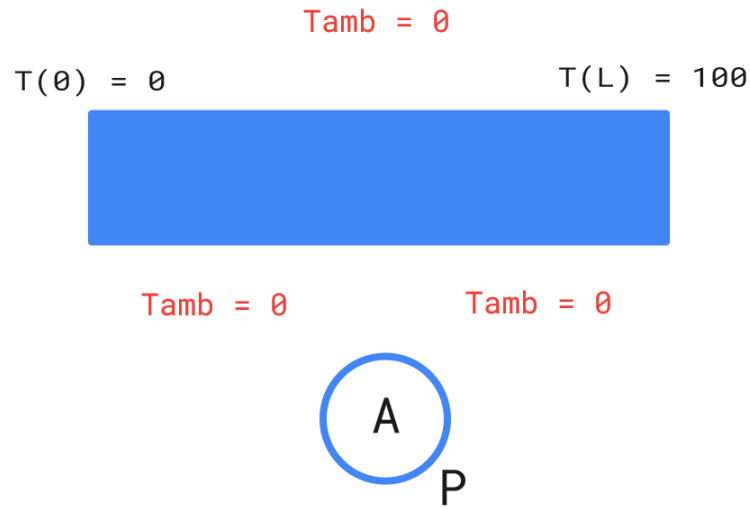


Figure 1: Case 1 Boundary Condition Setup

For the problem, we will let $L = 1$ cm, $r = 0.1$ cm, $P = \frac{\pi}{5}$ cm, $A = \frac{\pi}{100}$ cm², $k = 0.5 \frac{W}{cm \cdot K}$.

From the problem definition, we have boundary conditions $T(0) = 0$ and $T(L) = 100$. Applying these conditions with Equation 6 leads to:

$$T(0) = 0 = D$$

$$T(L) = 100 = C \sinh(\alpha)$$

With these conditions, Equation 6 becomes:

$$T(x) = \frac{100}{\sinh(\alpha)} \sinh(\alpha x) \quad (7)$$

1.4 Case 2: Neumann Boundary Condition

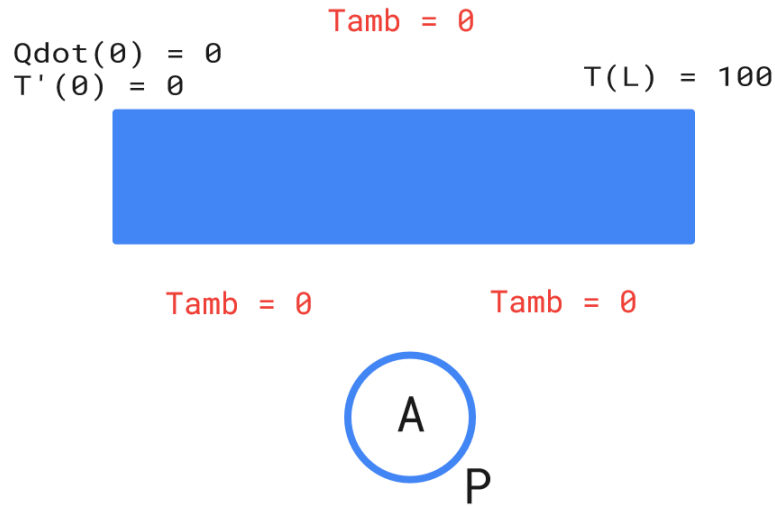


Figure 2: Case 2 Boundary Condition Setup

For the problem, we will let $L = 1$ cm, $r = 0.1$ cm, $P = \frac{\pi}{5}$ cm, $A = \frac{\pi}{100}$ cm².

From the problem definition, we have boundary conditions $T'(0) = 0$ and $T(L) = 100$. Applying these conditions with Equation 6 leads to:

$$T'(0) = 0 = \alpha C$$

$$T(L) = 100 = D \cosh(\alpha)$$

With these conditions, Equation 6 becomes:

$$T(x) = \frac{100}{\cosh(\alpha)} \cosh(\alpha x) \quad (8)$$

1.5 Temperature Results

Using the Python code provided at the end of this report, the analytical values for the temperature at varying positions along the bar for different values of α are shown. Each case will have their values in a table and a following figure. After each case has their values shown, there will be a comparison between each case.

1.5.1 Case 1

Position	$\alpha = 0.29$	$\alpha = 0.50$	$\alpha = 1.35$	$\alpha = 2.75$	$\alpha = 4.75$	$\alpha = 9.15$
0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.125	12.3292	12.0018	9.4243	4.5005	1.0889	0.0300
0.250	24.6746	24.0505	19.1176	9.5381	2.5731	0.1036
0.375	37.0524	36.1931	29.3566	15.7138	4.9914	0.3280
0.500	49.4789	48.4772	40.4336	23.7647	9.2217	1.0305
0.625	61.9705	60.9507	52.6647	34.6515	16.7996	3.2346
0.750	74.5435	73.6624	66.3991	49.6734	30.4760	10.1520
0.875	87.2144	86.6619	82.0288	70.6229	55.2158	31.8622
1.000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000

Table 1: Analytical Case 1 Temperature Distributions with Varying α

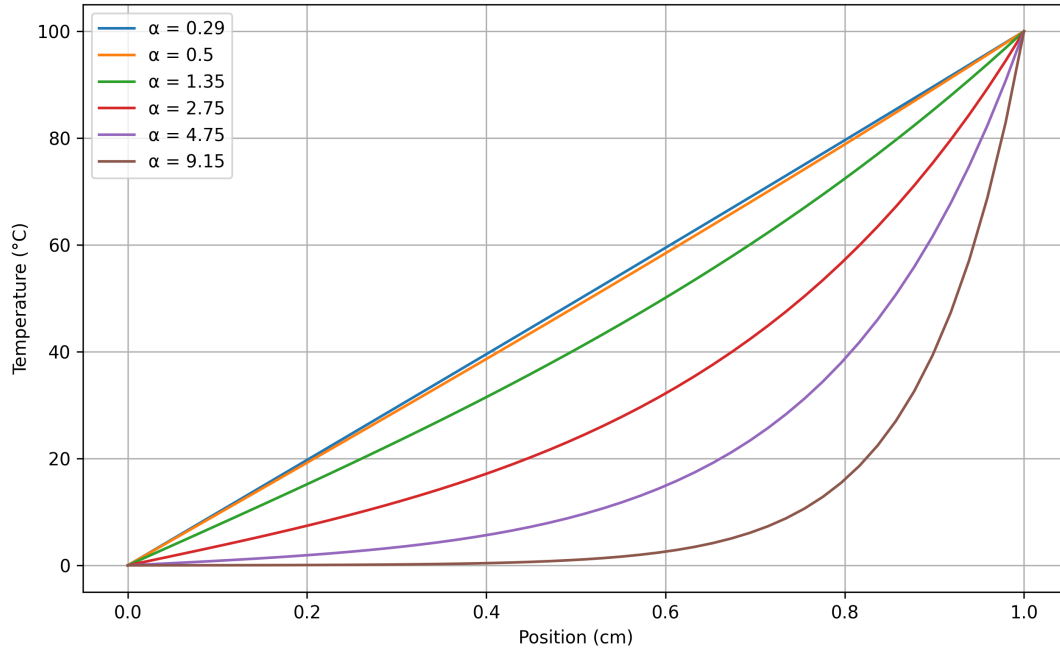


Figure 3: Analytical Case 1 Temperature Distributions with Varying α

1.5.2 Case 2

Position	$\alpha = 0.29$	$\alpha = 0.50$	$\alpha = 1.35$	$\alpha = 2.75$	$\alpha = 4.75$	$\alpha = 9.15$
0.000	95.9375	88.6819	48.5830	12.7335	1.7302	0.0212
0.125	96.0005	88.8552	49.2764	13.4933	2.0443	0.0367
0.250	96.1897	89.3756	51.3763	15.8632	3.1004	0.1057
0.375	96.5053	90.2453	54.9428	20.1262	5.2821	0.3287
0.500	96.9478	91.4677	60.0775	26.7908	9.3812	1.0307
0.625	97.5176	93.0474	66.9271	36.6525	16.8859	3.2347
0.750	98.2157	94.9907	75.6871	50.8879	30.5205	10.1520
0.875	99.0427	97.3053	86.6075	71.1960	55.2347	31.8622
1.000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000

Table 2: Analytical Case 2 Temperature Distributions with Varying α

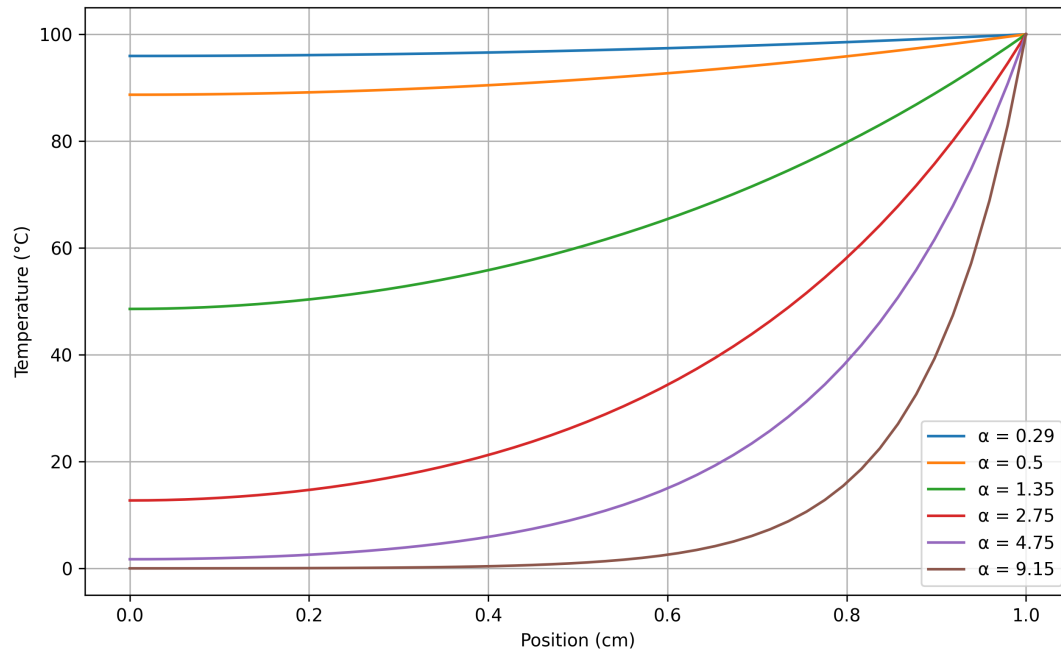


Figure 4: Analytical Case 2 Temperature Distributions with Varying α

1.5.3 Case Comparison

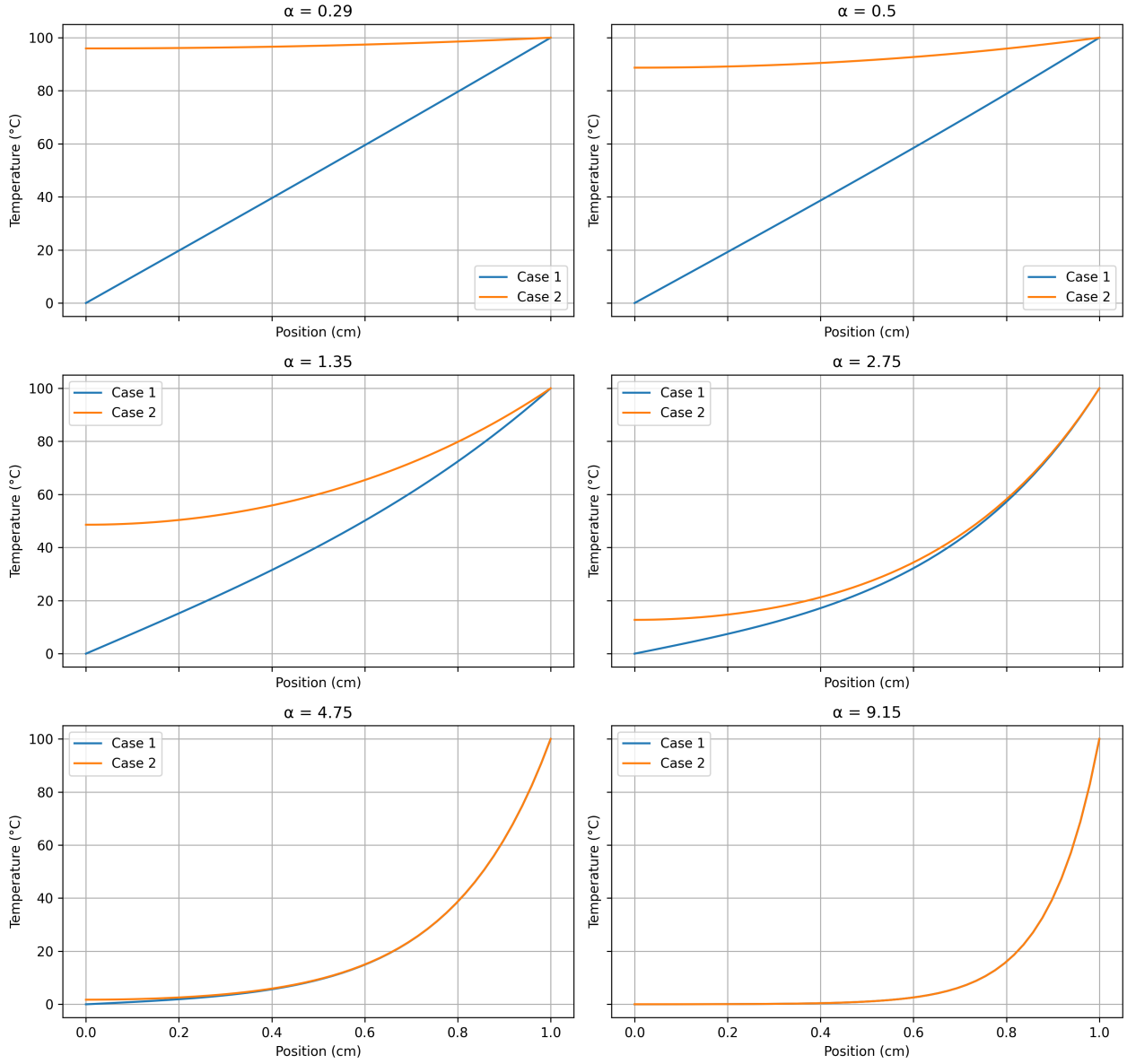


Figure 5: Temperature Distributions of Analytical Case 1 and 2 for Identical α

1.5.4 Discussion

As seen in Figure 3 and Figure 4, there is a strong correlation between an increase in α and the linearity of the temperature distribution. Qualitatively, α is the ratio between heat flux to the environment and thermal conductivity through the bar. With larger values of α , the heat flux to the environment dominates causing a more rapid heat flux in Case 1 or a larger temperature gradient across the bar in Case 2.

Figure 5 shows how the two cases differ from each other. In Case 1, the left hand boundary condition is that $T(0) = 0$. In Case 2, we have $T'(0) = 0$. This subtle difference results in vastly different temperature

distributions, particularly when α is small. For a given real-world problem, it becomes increasingly important to consider the left boundary condition as α becomes smaller.

1.6 Heat Flux Results

1.6.1 Calculating Heat Flux

In addition to the temperature distribution across the bar, the heat flux to the environment at $x = L$ provides a useful quantity for comparison. Using Equation 4 and replacing $\frac{dT(x)}{dx}$ with the derivative of Equation 6, you get:

$$\dot{Q}(1) = -kA[\alpha C \cosh(\alpha) + \alpha D \sinh(\alpha)] \quad (9)$$

For Case 1 this simplifies to become:

$$\dot{Q}(1) = -100kA\alpha \coth(\alpha) \quad (10)$$

For Case 2 this simplifies to become:

$$\dot{Q}(1) = -100kA\alpha \tanh(\alpha) \quad (11)$$

1.6.2 Case Comparison

Using Equation 10 and Equation 11 in Python to calculate the heat flux at the end of the rod for each case and α yields the following table and figure.

α	Case 1 $\dot{Q}$	Case 2 $\dot{Q}$
0.29	-1.6146	-0.1285
0.50	-1.6996	-0.3629
1.35	-2.4261	-1.8535
2.75	-4.3551	-4.2845
4.75	-7.4624	-7.4602
9.15	-14.3728	-14.3728

Table 3: Analytical Heat Flux with Varying α

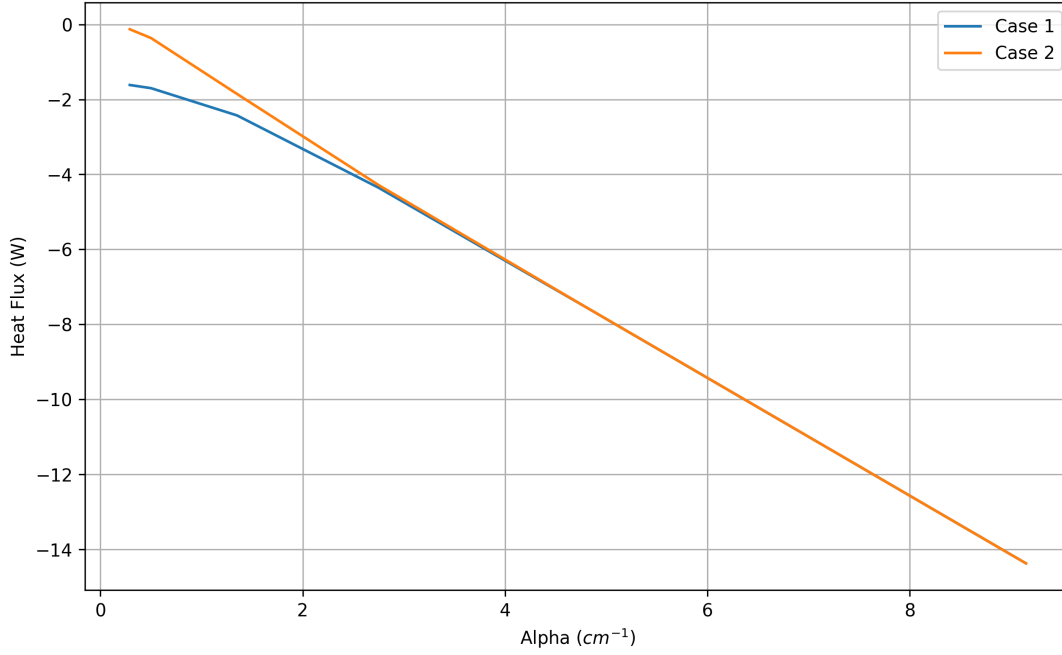


Figure 6: Analytical Heat Flux with Varying α

1.6.3 Discussion

Figure 6 shows larger α values causing a more similar value in heat flux between the two cases. When α becomes smaller than 4, larger differences can be seen between the cases. This indicates that the left boundary condition impacts the heat flux at the right end of the bar primarily for smaller α values and should be considered when modeling a real-world problem.

2 Finite Difference Method

2.1 Abstract

After considering the analytical solution, we will compare the Finite Difference Method (FDM). When solutions to the differential equation are not known, FDM can be applied to approximate the analytical solution.

The method works by selecting N number of points along the rod. At each point, there is an associated T_n , T'_n , and T''_n . Through the governing differential equation (Equation 5), a system of equations can be created for each point along the rod such that the equation is satisfied.

To calculate the values of T_n , T'_n , and T''_n for each point, the finite difference method will be used.

Rewriting the governing differential equation for use with FDM, we have:

$$T''_n - \alpha^2 T_n = 0 \quad (12)$$

Through Taylor expansion and truncating higher order terms, we get that:

$$T'_n = \frac{T_{n+1} - T_{n-1}}{2\Delta x} \quad (13)$$

$$T''_n = \frac{T_{n+1} - 2T_n + T_{n-1}}{\Delta x^2} \quad (14)$$

$$\begin{aligned} \frac{T_{n+1} - 2T_n + T_{n-1}}{\Delta x^2} - \alpha^2 T_n &= 0 \\ -T_{n-1} + kT_n - T_{n+1} &= 0 \end{aligned} \quad (15)$$

Where:

$$k = 2 + \alpha^2 \Delta x^2 \quad (16)$$

2.2 Case 1: Dirichlet Boundary Condition

With the same boundary conditions set up in the analytical solution and Figure 1, we can construct a matrix equation to solve for any N number of temperatures along the length of the bar. Using Equation 15, we get:

$$\begin{bmatrix} k & -1 & \cdots & 0 & 0 \\ -1 & k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & -1 & 0 \\ 0 & 0 & -1 & k & -1 \\ 0 & 0 & 0 & -1 & k \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N-3} \\ T_{N-2} \end{bmatrix} = \begin{bmatrix} T_0 = 0 \\ 0 \\ \vdots \\ 0 \\ T_{N-1} = 100 \end{bmatrix} \quad (17)$$

Solving Equation 24 for varying values of N yields a finite temperature distribution. Any variance in the value of α will change the associated value of k as given in Equation 16.

2.3 Case 2: Neumann Boundary Condition

For Case 2, we have from Equation 13 that:

$$T'_0 = 0 = \frac{T_1 - T_{-1}}{2\Delta x}$$

Therefore:

$$T_1 = T_{-1}$$

Using Equation 15, we can find that:

$$kT_0 - 2T_1 = 0$$

We then get the required condition to setup a similar matrix:

$$-T_1 + k'T_0 = 0$$

Where:

$$k' = \frac{k}{2}$$

This leads to the matrix equation for Case 2:

$$\begin{bmatrix} k' & -1 & \cdots & 0 & 0 \\ -1 & k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & -1 & 0 \\ 0 & 0 & -1 & k & -1 \\ 0 & 0 & 0 & -1 & k \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{N-3} \\ T_{N-2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ T_{N-1} = 100 \end{bmatrix} \quad (18)$$

2.4 Temperature Results

Below are the results calculated using the finite difference method for each case. Each solution used $dx = 0.125$, with $N = 9$ (8 points plus the final $T(L)$).

2.4.1 Case 1

Position	$\alpha = 0.29$	$\alpha = 0.50$	$\alpha = 1.35$	$\alpha = 2.75$	$\alpha = 4.75$	$\alpha = 9.15$
0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.125	12.3292	12.0019	9.4303	4.5385	1.1462	0.0433
0.250	24.6746	24.0508	19.1291	9.6133	2.6966	0.1434
0.375	37.0525	36.1935	29.3726	15.8241	5.1975	0.4309
0.500	49.4790	48.4777	40.4526	23.9047	9.5308	1.2821
0.625	61.9705	60.9512	52.6845	34.8100	17.2241	3.8106
0.750	74.5435	73.6628	66.4167	49.8286	30.9896	11.3241
0.875	87.2145	86.6621	82.0403	70.7351	55.6801	33.6513
1.000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000

Table 4: FDM Case 1 Temperature Distributions with Varying α

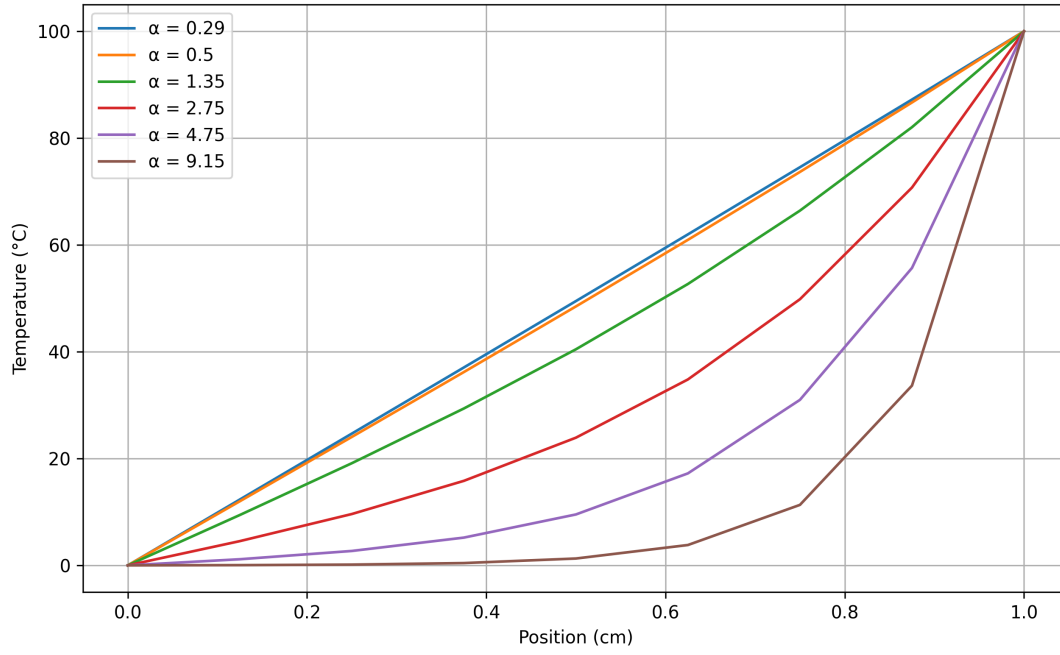


Figure 7: FDM Case 1 Temperature Distributions with Varying α

2.4.2 Case 2

Position	$\alpha = 0.29$	$\alpha = 0.50$	$\alpha = 1.35$	$\alpha = 2.75$	$\alpha = 4.75$	$\alpha = 9.15$
0.000	95.9379	88.6852	48.6508	12.9034	1.8504	0.0329
0.125	96.0009	88.8584	49.3435	13.6658	2.1765	0.0544
0.250	96.1901	89.3788	51.4414	16.0429	3.2700	0.1471
0.375	96.5057	90.2482	55.0041	20.3158	5.5162	0.4322
0.500	96.9481	91.4702	60.1331	26.9892	9.7072	1.2826
0.625	97.5179	93.0495	66.9746	36.8519	17.3203	3.8108
0.750	98.2158	94.9922	75.7232	51.0690	31.0395	11.3241
0.875	99.0428	97.3061	86.6282	71.3207	55.7013	33.6513
1.000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000

Table 5: FDM Case 2 Temperature Distributions with Varying α

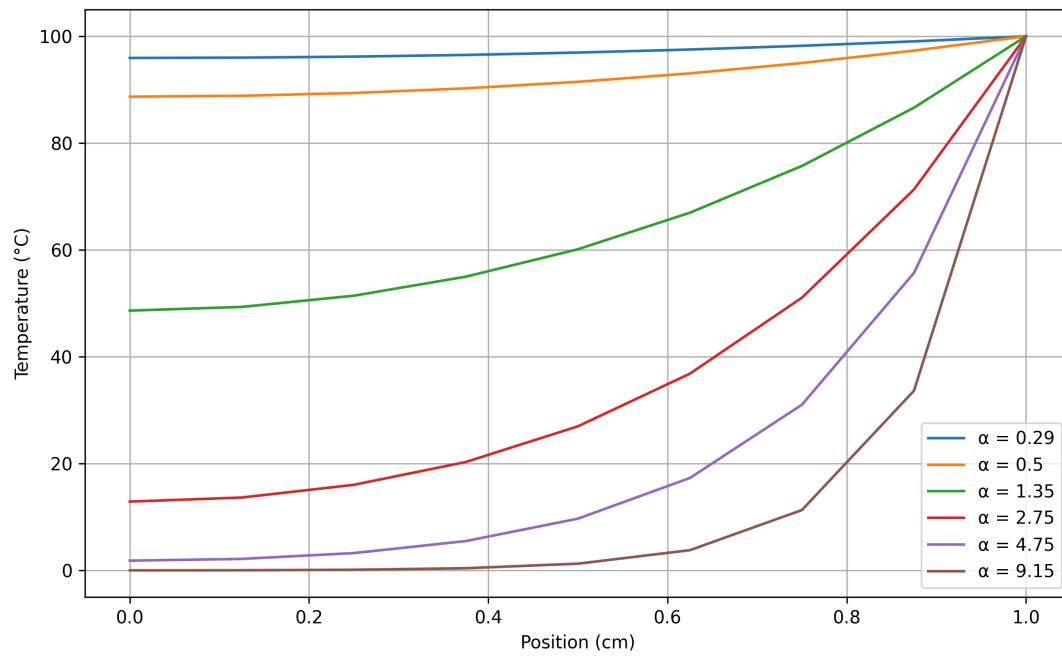


Figure 8: FDM Case 2 Temperature Distributions with Varying α

2.4.3 Case Comparison

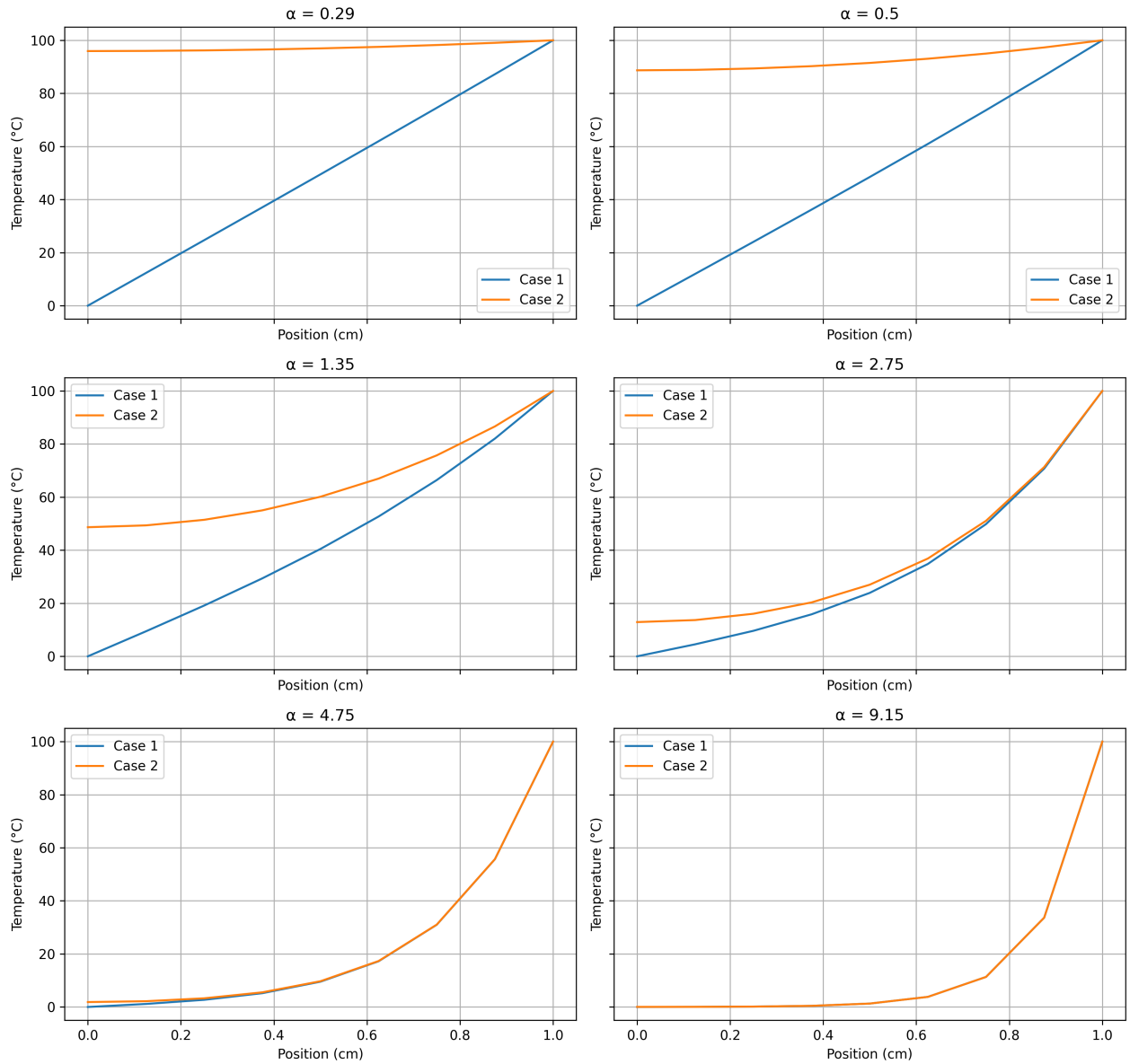


Figure 9: Temperature Distributions of FDM Case 1 and 2 for Identical α

2.4.4 Discussion

The finite difference method provides a straight-forward approach to solving the temperature distribution without having an analytical solution. Similar results to the analytical solution were obtained and show the same trends as α changes.

2.5 Heat Flux Results

2.5.1 Calculating Heat Flux

In order to calculate the heat flux at the end of the bar, $x = L = 1$, we need to calculate the derivative of the temperature at $x = L$ using finite differences. We will start with the 2nd order Taylor Expansion centered around

$x = L$ and applied at $x = L - dx$:

$$T(L - dx) = T(L) - dxT'(L) + \frac{dx^2}{2}T''(L) \quad (19)$$

For a first order extraction, we truncate the last term and only use the first 2. Using $T'_n = \frac{T_n - T_{n-1}}{dx}$ as an approximation. We use this in Equation 4 to get:

$$\dot{Q}(1) = -kA \left[\frac{T_n - T_{n-1}}{dx} \right] \quad (20)$$

For a second order extraction, we use Equation 19 to get $T'_n = \frac{T_n - T_{n-1}}{dx} + \frac{\alpha^2 dx T_n}{2}$ as an approximation. We use this in Equation 4 to get:

$$\dot{Q}(1) = -kA \left[\frac{T_n - T_{n-1}}{dx} + \frac{\alpha^2 dx T_n}{2} \right] \quad (21)$$

2.5.2 Case Comparison

α	1st Order $\dot{Q}(W)$	2nd Order $\dot{Q}(W)$
0.29	-1.6067	-1.6149
0.50	-1.6761	-1.7006
1.35	-2.2569	-2.4358
2.75	-3.6775	-4.4200
4.75	-5.5694	-7.7845
9.15	-8.3376	-16.5571

Table 6: Case 1 FDM 1st and 2nd Order Extraction Heat Flux with Varying α

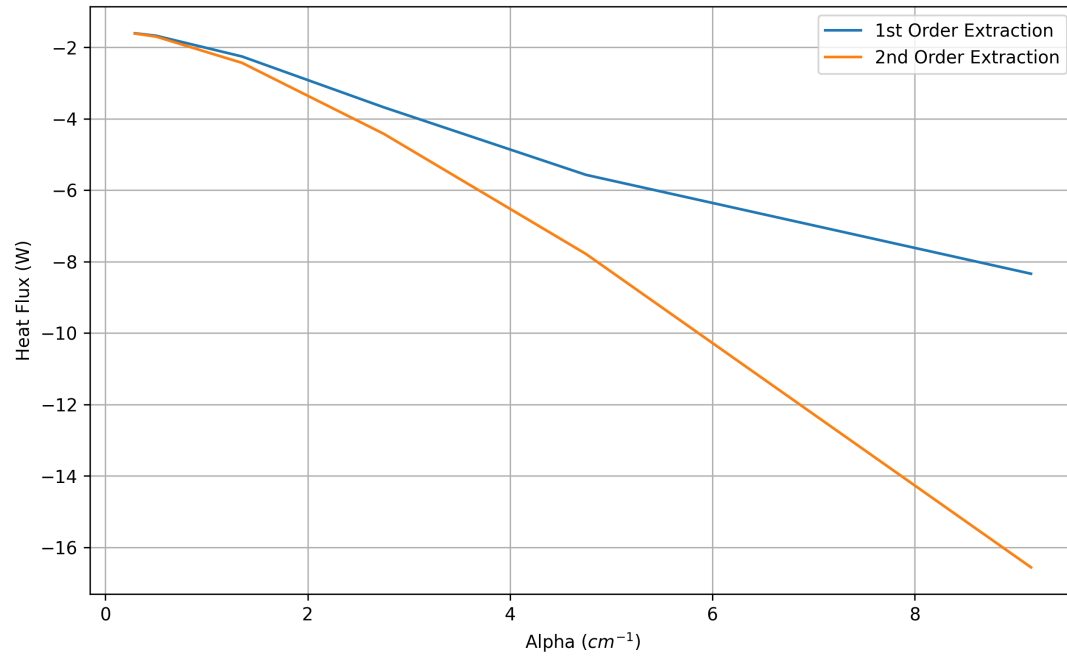


Figure 10: 1st and 2nd Order Extraction of Heat Flux Extraction of FDM Case 1

α	1st Order $\dot{Q}(W)$	2nd Order $\dot{Q}(W)$
0.29	-0.1203	-0.1285
0.50	-0.3385	-0.3631
1.35	-1.6804	-1.8593
2.75	-3.6039	-4.3464
4.75	-5.5667	-7.7818
9.15	-8.3376	-16.5571

Table 7: Case 2 FDM 1st and 2nd Order Extraction Heat Flux with Varying α

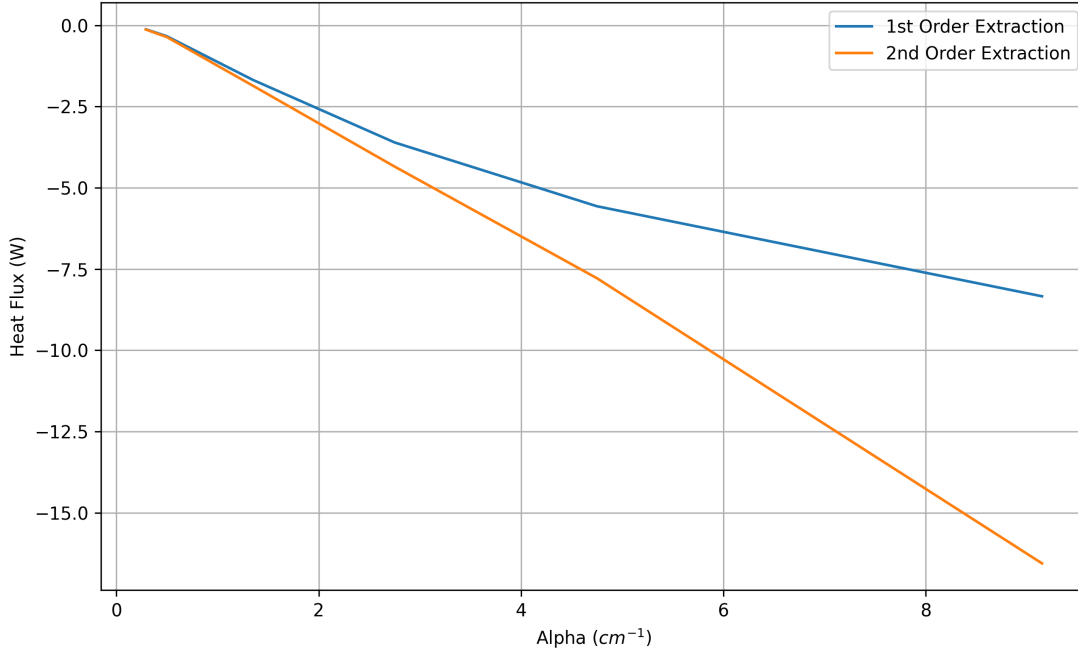


Figure 11: 1st and 2nd Order Extraction of Heat Flux Extraction of FDM Case 2

2.5.3 Discussion

The finite difference method provides a straight-forward approach to solving the heat flux without having an analytical solution. From Figure 10 and Figure 11, it is clear that the 1st and 2nd order extractions vary from each other, giving rise to concerns over the accuracy of the heat flux calculation. In the next section, the convergence of both extraction methods will be analyzed.

3 Finite Element Method

3.1 Abstract

From our governing equation of heat conduction Equation 5, we can integrate to have:

$$\int_0^1 (T''v - \alpha^2 T v) dx = 0$$

Where v is a test function. Splitting the integral into the two parts yields:

$$\int_0^1 (T''v) dx - \int_0^1 (\alpha^2 T v) dx = 0$$

Noting that $T''v = (T'v)' - T'v'$, we get:

$$\int_0^1 ((T'v)' - T'v') dx - \int_0^1 (\alpha^2 T v) dx = 0$$

This leads to:

$$T'(1)v(1) - T'(0)v(0) + \int_0^1 (-T'v') dx - \alpha^2 \int_0^1 (T v) dx = 0 \quad (22)$$

Let ϕ be the test function v . $T(x)$ can be generalized as an interpolation:

$$T_{FEM}(x) = \sum_{n=0}^N T_n \phi_n(x)$$

Discretizing Equation 22 and considering the interior nodes gives us:

$$T_{i-1} \left[\frac{-1}{dx} + \frac{\alpha^2 dx}{6} \right] + T_i \left[\frac{2}{dx} + \frac{4\alpha^2 dx}{6} \right] + T_{i+1} \left[\frac{-1}{dx} + \frac{\alpha^2 dx}{6} \right] \quad (23)$$

3.2 Case 1: Dirichlet Boundary Condition

Similarly to the FDM, we can construct a matrix equation to solve for any N number of temperatures along the length of the bar. Using Equation 23, we get a new matrix where $k = \frac{\frac{2}{dx} + \frac{4\alpha^2 dx}{6}}{\frac{1}{dx} - \frac{\alpha^2 dx}{6}}$:

$$\begin{bmatrix} k & -1 & \cdots & 0 & 0 \\ -1 & k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & -1 & 0 \\ 0 & 0 & -1 & k & -1 \\ 0 & 0 & 0 & -1 & k \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{N-3} \\ T_{N-2} \end{bmatrix} = \begin{bmatrix} T_0 = 0 \\ 0 \\ \vdots \\ 0 \\ T_{N-1} = 100 \end{bmatrix} \quad (24)$$

Solving Equation 24 for varying values of N yields a finite temperature distribution. Any variance in the value of α will change the associated value of k as given in Equation 16.

3.3 Case 2: Neumann Boundary Condition

For Case 2, we have from Equation 13 that:

$$T'_0 = 0 = \frac{T_1 - T_{-1}}{2\Delta x}$$

Therefore:

$$T_1 = T_{-1}$$

Using Equation 15, we can find that:

$$kT_0 - 2T_1 = 0$$

We then get the required condition to setup a similar matrix:

$$-T_1 + k'T_0 = 0$$

Where:

$$k' = \frac{k}{2}$$

This leads to the matrix equation for Case 2:

$$\begin{bmatrix} k' & -1 & \cdots & 0 & 0 \\ -1 & k & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & -1 & 0 \\ 0 & 0 & -1 & k & -1 \\ 0 & 0 & 0 & -1 & k \end{bmatrix} \begin{pmatrix} T_0 \\ T_1 \\ \vdots \\ T_{N-3} \\ T_{N-2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ T_{N-1} = 100 \end{pmatrix} \quad (25)$$

3.4 Temperature Results

3.4.1 Case 1

Position	$\alpha = 0.29$	$\alpha = 0.50$	$\alpha = 1.35$	$\alpha = 2.75$	$\alpha = 4.75$	$\alpha = 9.15$
0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.125	12.3292	12.0016	9.4183	4.4617	1.0299	0.0182
0.250	24.6746	24.0502	19.1061	9.4612	2.4455	0.0669
0.375	37.0524	36.1927	29.3405	15.6011	4.7771	0.2276
0.500	49.4789	48.4767	40.4145	23.6216	8.8980	0.7691
0.625	61.9704	60.9502	52.6448	34.4893	16.3516	2.5971
0.750	74.5434	73.6619	66.3814	49.5144	29.9296	8.7700
0.875	87.2144	86.6616	82.0174	70.5078	54.7177	29.6142
1.000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000

Table 8: FEM Case 1 Temperature Distributions with Varying α

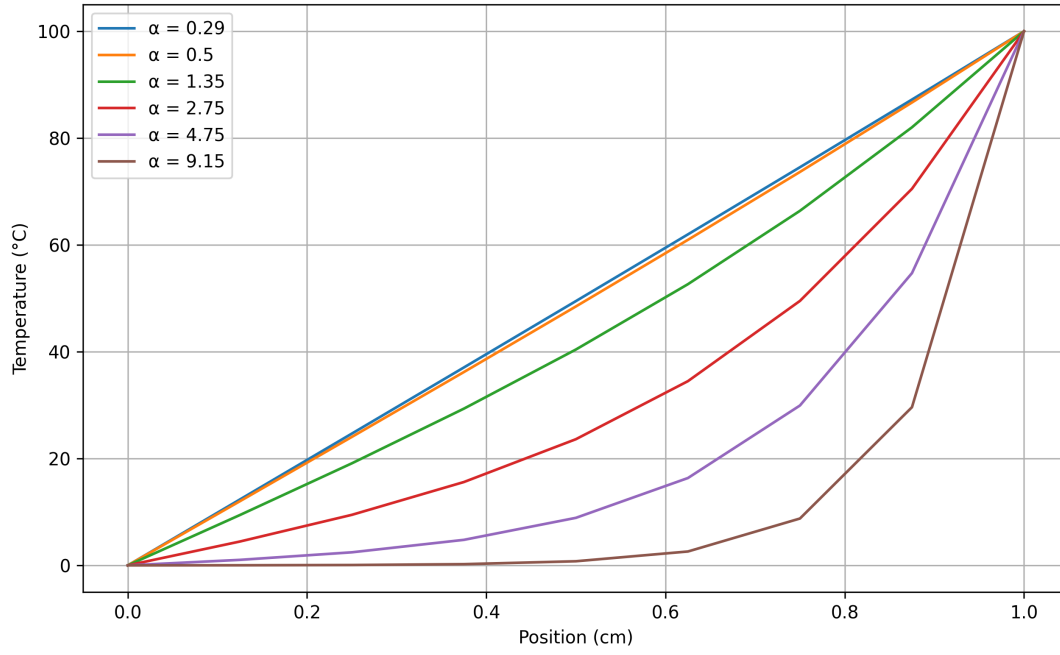


Figure 12: FEM Case 1 Temperature Distributions with Varying α

3.4.2 Case 2

Position	$\alpha = 0.29$	$\alpha = 0.50$	$\alpha = 1.35$	$\alpha = 2.75$	$\alpha = 4.75$	$\alpha = 9.15$
0.000	95.9370	88.6786	48.5148	12.5614	1.6090	0.0118
0.125	96.0001	88.8519	49.2089	13.3184	1.9103	0.0217
0.250	96.1893	89.3725	51.3109	15.6809	2.9271	0.0680
0.375	96.5050	90.2424	54.8811	19.9335	5.0402	0.2279
0.500	96.9475	91.4651	60.0215	26.5888	9.0412	0.7692
0.625	97.5174	93.0453	66.8793	36.4491	16.4285	2.5972
0.750	98.2155	94.9893	75.6507	50.7028	29.9690	8.7700
0.875	99.0426	97.3045	86.5866	71.0682	54.7342	29.6142
1.000	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000

Table 9: FEM Case 2 Temperature Distributions with Varying α

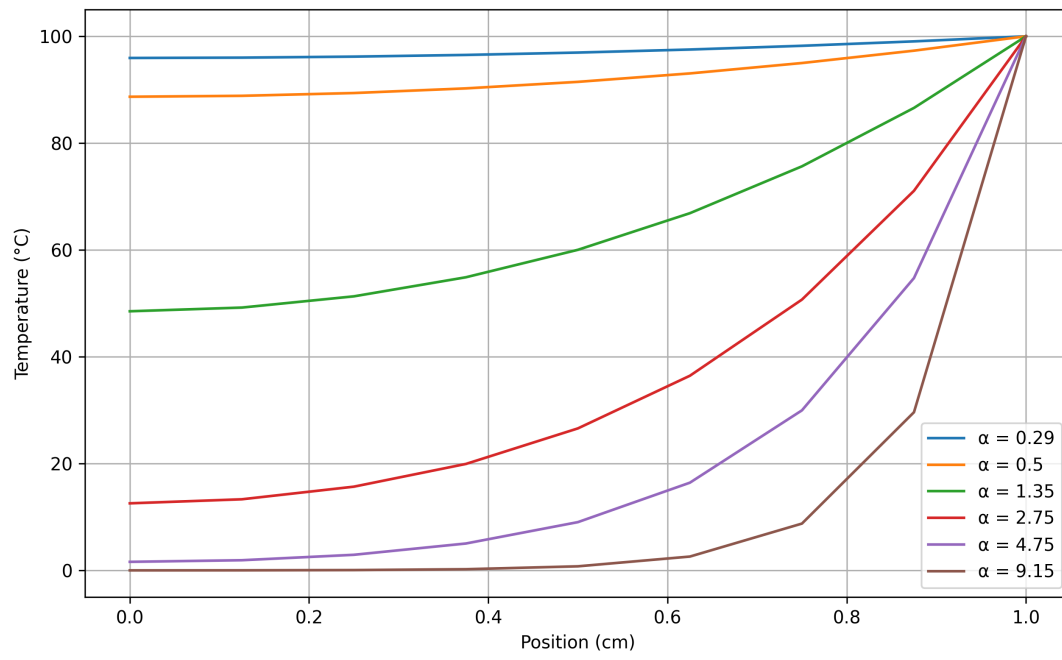


Figure 13: FEM Case 2 Temperature Distributions with Varying α

3.4.3 Case Comparison

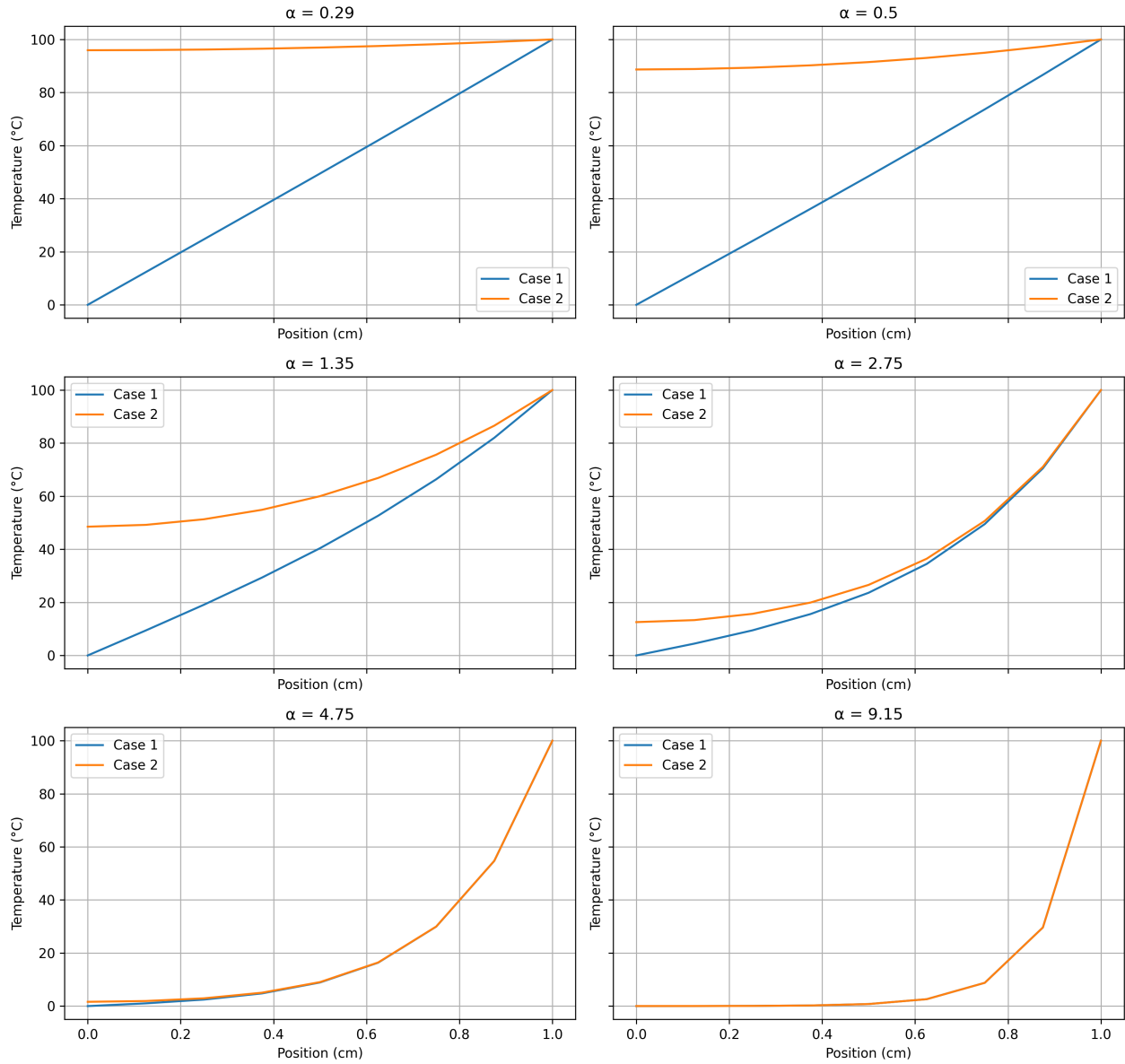


Figure 14: Temperature Distributions of FEM Case 1 and 2 for Identical α

3.4.4 Discussion

The figures generated through the use of FEM show a strong similarity to the figures generated by the FDM. The discussion follows in suit, with case 1 showing larger values of α causing heat to leave the bar when closer to $x = L$, while smaller values of α tend to more linear solutions.

In case 2, large values of α drive the solution to become similar to case 1, nearing the result of a boundary condition of $T(0) = 0$.

3.5 Heat Flux Results

3.5.1 Calculating Heat Flux

Heat flux is calculated using the using the following equation:

$$\dot{Q}(1) = -kA \left[\left(\frac{-1}{dx} + \frac{\alpha^2 dx}{6} \right) T_{N-1} + \left(\frac{1}{dx} + \frac{2\alpha^2 dx}{6} \right) T_N \right] \quad (26)$$

3.5.2 Case Comparison

α	$\dot{Q}(W)$
0.29	-1.6146
0.50	-1.6996
1.35	-2.4280
2.75	-4.3756
4.75	-7.5711
9.15	-15.1359

Table 10: Case 1 FEM Extraction Heat Flux with Varying α

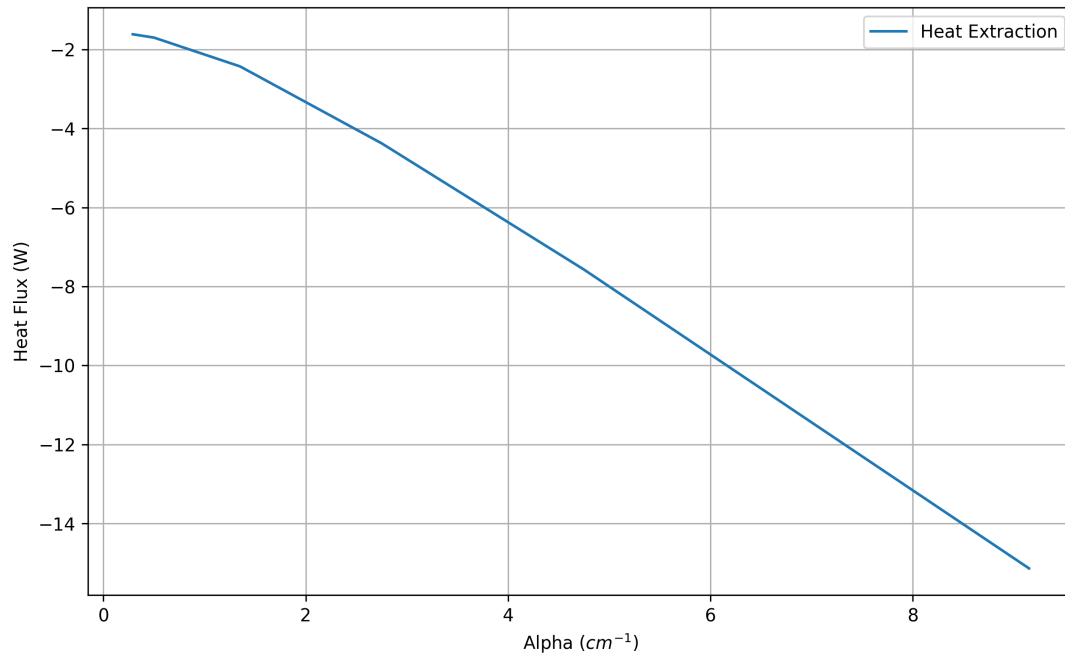


Figure 15: Extraction of Heat Flux Extraction of FEM Case 1

α	$\dot{Q}(W)$
0.29	-0.1285
0.50	-0.3631
1.35	-1.8565
2.75	-4.3065
4.75	-7.5691
9.15	-15.1359

Table 11: Case 2 FEM Extraction Heat Flux with Varying α

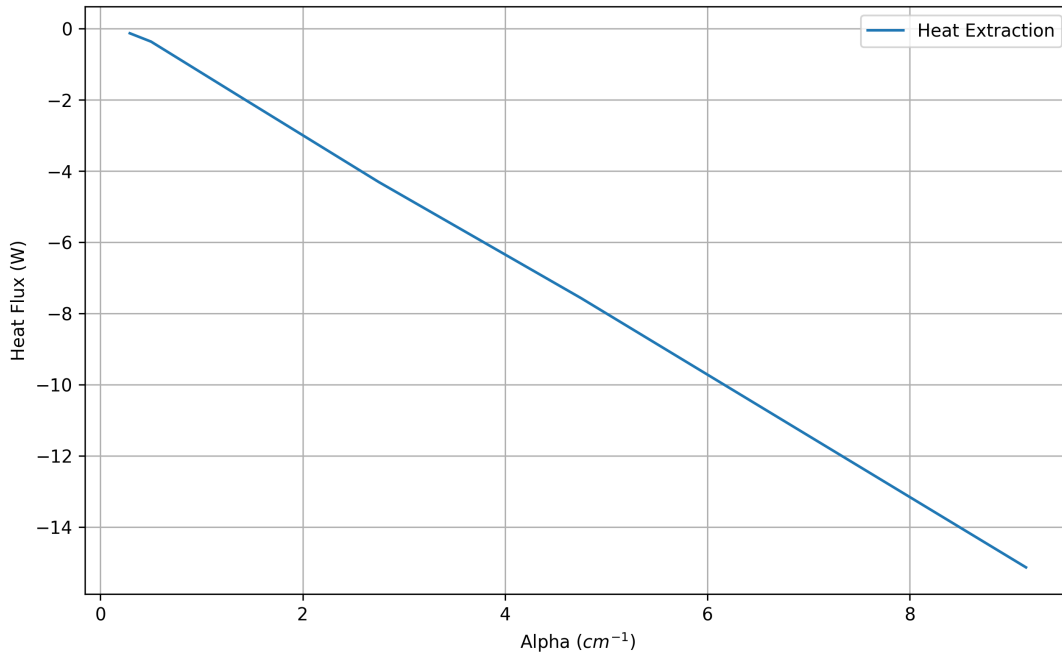


Figure 16: Extraction of Heat Flux Extraction of FEM Case 2

3.5.3 Discussion

The results obtained through extraction of the heat flux using FEM follow the results obtained using FDM. Larger values of α result in larger heat flux values at $x = L$.

4 Convergence of FDM and FEM

4.1 Abstract

It is necessary to compare the results of the FDM and FEM solutions to the analytical solutions. To do this, we will take the heat flux calculated at $T(L)$ in both cases at a constant $\alpha = 2.75$ and compare the analytical values to the FDM and FEM values with varying Δx . Specifically, Δx values of 0.25, 0.125, 0.0625, 0.03125, 0.015625, and 0.0078125 as these are computed from $\frac{1}{2^n}$ from $n = [2, 7]$.

The rate of convergence will be calculated for the heat flux extracted with a first order Taylor expansion and the second order Taylor expansion.

4.2 Calculating Rate of Convergence

The error in the finite methods solution compared to the analytical solution is given as follows:

$$E = |\dot{Q}_{analytical} - \dot{Q}_{FDM}| \propto \Delta x^\beta$$

Using 2 values for Δx to give us 2 $\dot{Q}_{FDM}$ values gives:

$$\beta = \frac{\log\left(\frac{|\dot{Q}_{analytical} - \dot{Q}_{FDM,1}|}{|\dot{Q}_{analytical} - \dot{Q}_{FDM,2}|}\right)}{\log\left(\frac{\Delta x_1}{\Delta x_2}\right)} \quad (27)$$

The error and convergence rate of the temperature at the midpoint can be calculated similarly.

4.3 Temperature At Midpoint Results

4.3.1 Case 1

Δx	FDM $T(0.5)$	FEM $T(0.5)$	True $T(0.5)$	FDM % Error	FEM % Error	FDM β	FEM β
0.2500000	24.3071	23.1721	23.7647	0.0228	0.0249	0.0000	0.0000
0.1250000	23.9047	23.6216	23.7647	0.0059	0.0060	1.9539	2.0497
0.0625000	23.8000	23.7292	23.7647	0.0015	0.0015	1.9880	2.0122
0.0312500	23.7736	23.7559	23.7647	0.0004	0.0004	1.9970	2.0030
0.0156250	23.7669	23.7625	23.7647	0.0001	0.0001	1.9992	2.0008
0.0078125	23.7653	23.7642	23.7647	0.0000	0.0000	1.9998	2.0002

Table 12: Case 1 FDM and FEM Midpoint Temperature of Varying Δx

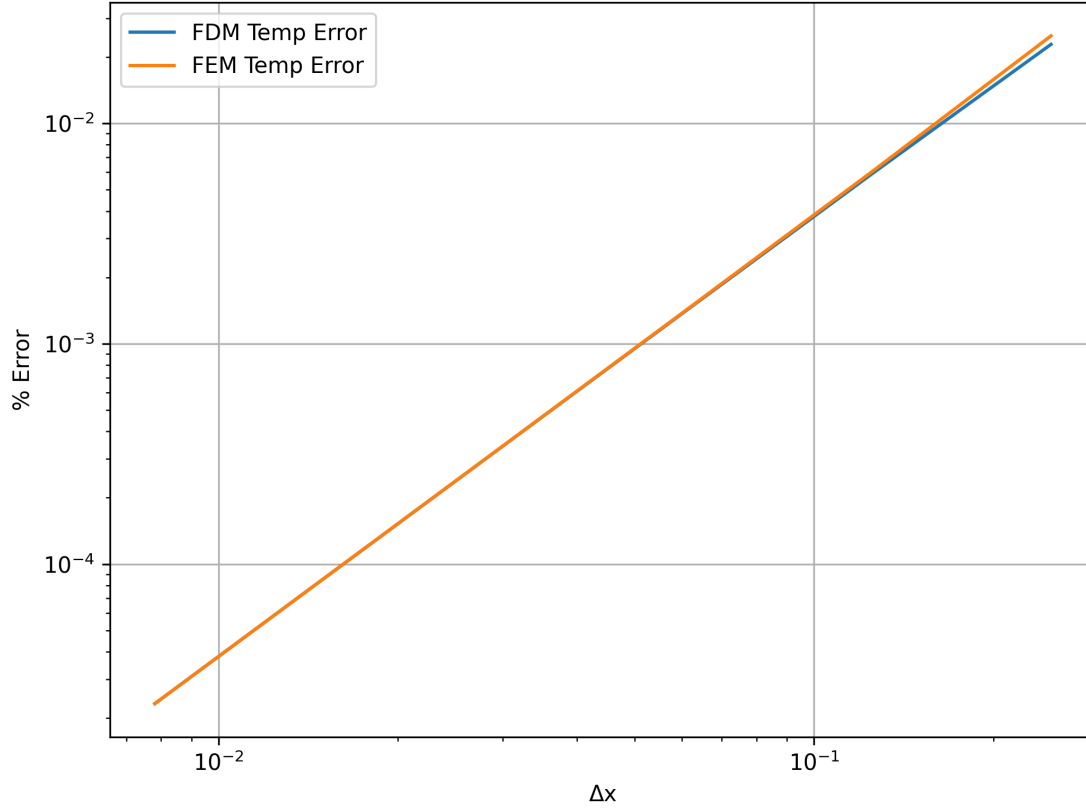


Figure 17: Case 1 Temp Convergence

4.3.2 Case 2

Δx	FDM $T(0.5)$	FEM $T(0.5)$	True $T(0.5)$	FDM % Error	FEM % Error	FDM β	FEM β
0.2500000	27.5643	25.9599	26.7908	0.0289	0.0310	0.0000	0.0000
0.1250000	26.9892	26.5888	26.7908	0.0074	0.0075	1.9627	2.0401
0.0625000	26.8408	26.7407	26.7908	0.0019	0.0019	1.9903	2.0099
0.0312500	26.8033	26.7783	26.7908	0.0005	0.0005	1.9976	2.0025
0.0156250	26.7939	26.7877	26.7908	0.0001	0.0001	1.9994	2.0006
0.0078125	26.7916	26.7900	26.7908	0.0000	0.0000	1.9998	2.0002

Table 13: Case 2 FDM and FEM Extracted $T(0.5)$ of Varying Δx

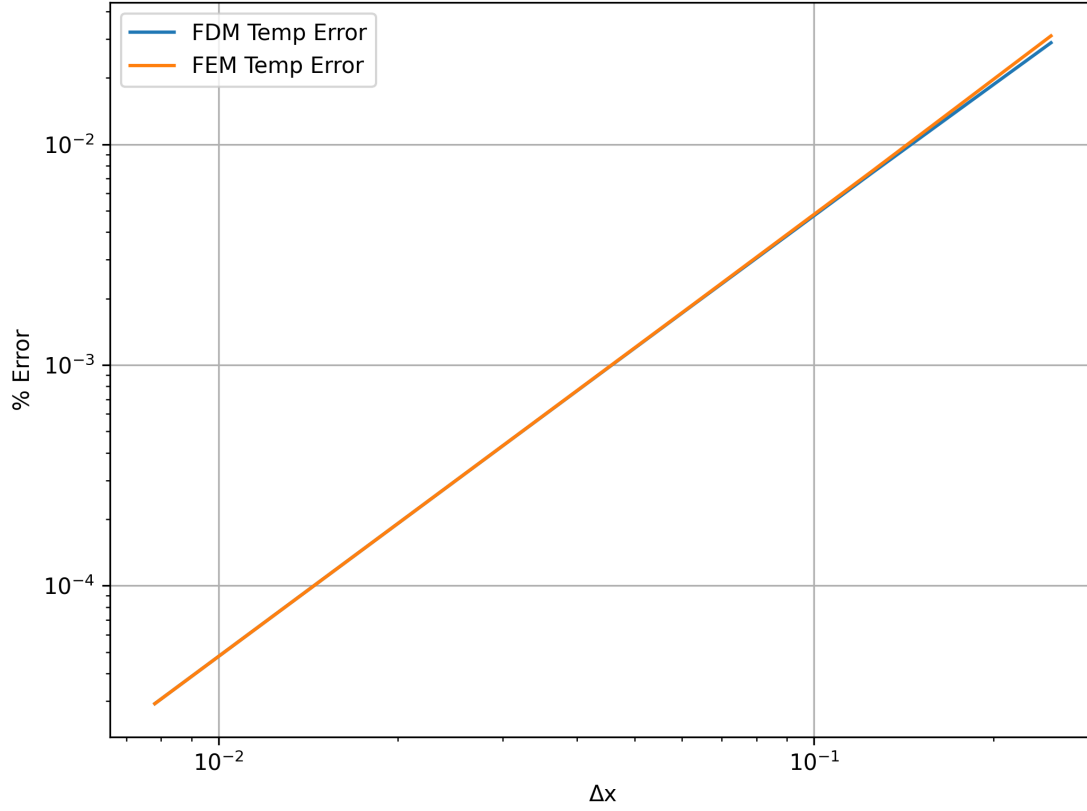


Figure 18: Case 2 Temp Convergence

4.4 Heat Flux Results

4.4.1 Case 1

Δx	FDM $\dot{Q}$	FEM $\dot{Q}$	True $\dot{Q}$	FDM % Error	FEM % Error	FDM β	FEM β
0.2500000	-4.6094	-4.4362	-4.3551	5.84E-2	1.86E-2	0.0000	0.0000
0.1250000	-4.4200	-4.3756	-4.3551	1.49E-2	4.69E-3	1.9711	1.9882
0.0625000	-4.3714	-4.3603	-4.3551	3.74E-3	1.18E-3	1.9925	1.9970
0.0312500	-4.3592	-4.3564	-4.3551	9.37E-4	2.94E-4	1.9981	1.9992
0.0156250	-4.3562	-4.3555	-4.3551	2.34E-4	7.3E-5	1.9995	1.9998
0.0078125	-4.3554	-4.3552	-4.3551	5.9E-5	1.8E-5	1.9999	1.9999

Table 14: Case 1 FDM and FEM Extracted $\dot{Q}$ of Varying Δx

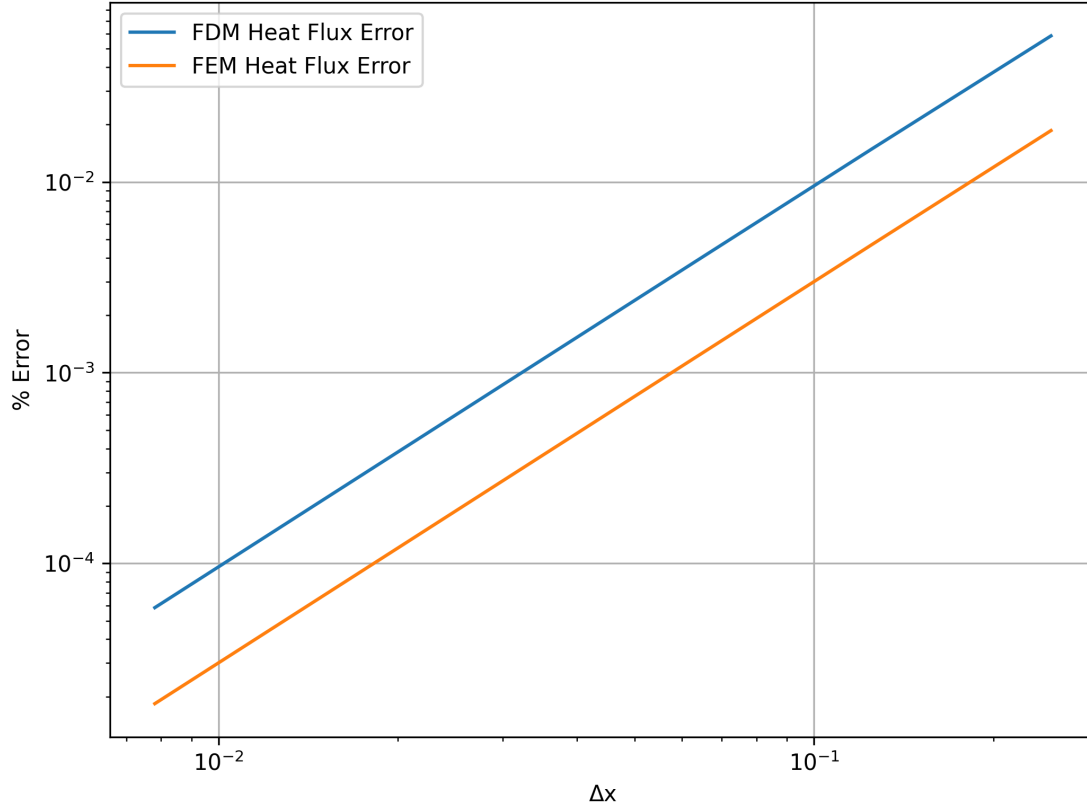


Figure 19: Case 1 Heat Convergence

4.4.2 Case 2

Δx	FDM $\dot{Q}$	FEM $\dot{Q}$	True $\dot{Q}$	FDM % Error	FEM % Error	FDM β	FEM β
0.250000	-4.5266	-4.3720	-4.2845	5.65E-2	2.04E-2	0.0000	0.0000
0.125000	-4.3464	-4.3065	-4.2845	1.44E-2	5.13E-3	1.9682	1.9908
0.062500	-4.3001	-4.2900	-4.2845	3.63E-3	1.29E-3	1.9918	1.9977
0.031250	-4.2884	-4.2859	-4.2845	9.09E-4	3.22E-4	1.9979	1.9994
0.015625	-4.2855	-4.2849	-4.2845	2.27E-4	8.00E-5	1.9995	1.9999
0.0078125	-4.2848	-4.2846	-4.2845	5.7E-5	2.0E-5	1.9999	2.0000

Table 15: Case 2 FDM and FEM Extracted $\dot{Q}$ of Varying Δx

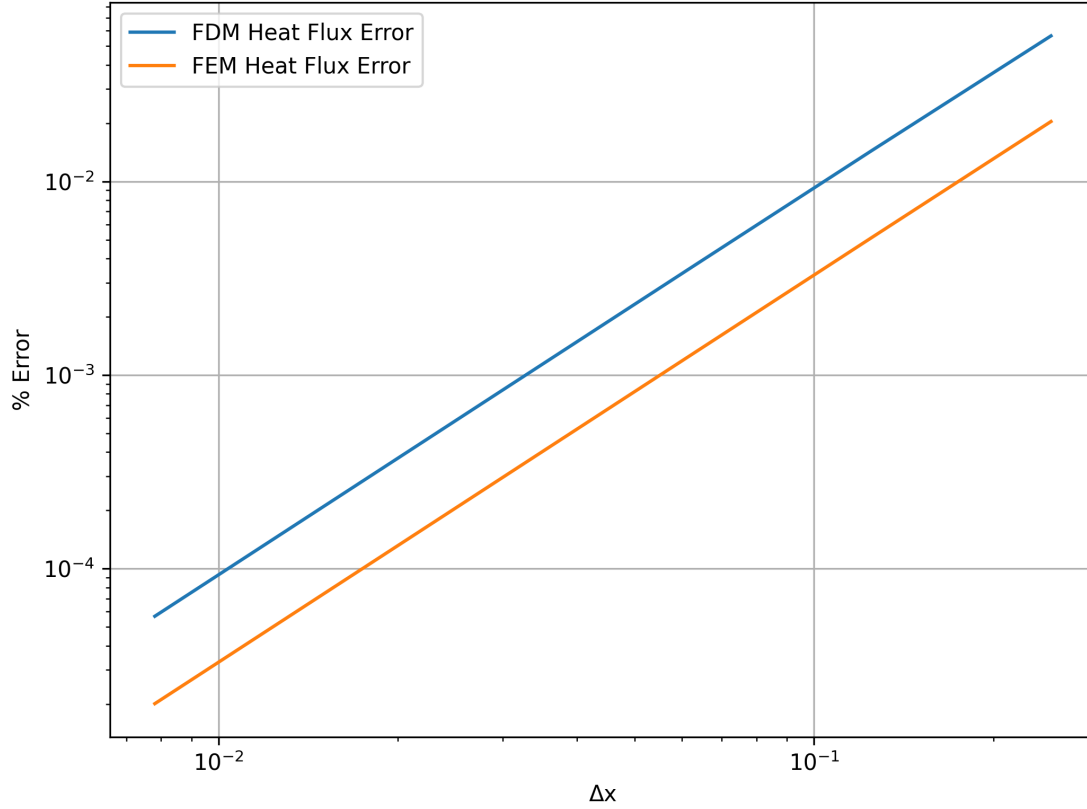


Figure 20: Case 2 Heat Convergence

4.5 Discussion

Temperature convergence will be discussed first. Using Figure 17 and Figure 18, it can be seen that both FDM and FEM converge with $\beta = 2$ as the mesh count increases. There is a negligible increase in % error going from FDM to FEM, as seen in Table 12 and Table 13.

Regarding the heat flux convergence, Using Figure 19 and Figure 20, it can be seen that both FDM and FEM converge with $\beta = 2$ as the mesh count increases. There is a noticeable decrease in % error going from FDM to FEM, with FEM resulting in approximately 2-3 times less error than FDM as seen in Table 14 and Table 15.

5 Python Code

5.1 main.py

```
# main.py

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import case1
import case2
import common

data_dict = {}
data_dict["a_values"] = np.array([0.29, 0.5, 1.35, 2.75, 4.75, 9.15])

def prob1():
    '''Analytical solutions'''
    print("Problem 1")
    # Problem 1 - Analytical solution
    data_dict["prob1"] = {}
    data_dict["prob1"]["x_values"] = np.array([0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75,
        0.875, 1.0])
    data_dict["prob1"]["x_values_fine"] = np.linspace(0, 1)

    # Case 1 temperature calculations
    results = []
    results_fine = []
    for a_value in data_dict["a_values"]:
        result = case1.analytical(a=a_value, x=data_dict["prob1"]["x_values"])
        result_fine = case1.analytical(a=a_value, x=data_dict["prob1"]["x_values_fine"])
        results.append(result)
        results_fine.append(result_fine)

    data_dict["prob1"]["case1_temp_results"] = np.array(results_fine)

    df_case1 = pd.DataFrame(results, index=[f'a = {a}' for a in data_dict["a_values"]],
        columns=[f'x = {x:.3f}' for x in data_dict["prob1"]["x_values"]])

    # Pretty print DataFrame for Case 1 temperature
    print("Analytical Case 1 Temperature Results:")
    print(df_case1)
    print("\n" * 2)

    # Plotting Case 1 temperature
    plt.figure(figsize=(10, 6))
    for idx, a_value in enumerate(data_dict["a_values"]):
        plt.plot(data_dict["prob1"]["x_values_fine"],
            data_dict["prob1"]["case1_temp_results"][idx], label=f' = {a_value}')
```

```

plt.xlabel('Position (cm)')
plt.ylabel('Temperature (C)')
plt.legend()
plt.grid(True)
plt.savefig('prob1_case1_analytical_temperature_vs_position.png', dpi=300)
#plt.show()
plt.close()

# Case 2 temperature calculations
results = []
results_fine = []
for a_value in data_dict["a_values"]:
    result = case2.analytical(a=a_value, x=data_dict["prob1"]["x_values"])
    result_fine = case2.analytical(a=a_value, x=data_dict["prob1"]["x_values_fine"])
    results.append(result)
    results_fine.append(result_fine)

data_dict["prob1"]["case2_temp_results"] = np.array(results_fine)

df_case2 = pd.DataFrame(results, index=[f'a = {a}' for a in data_dict["a_values"]],
    columns=[f'x = {x:.3f}' for x in data_dict["prob1"]["x_values"]])

# Pretty print DataFrame for Case 2
print("Analytical Case 2 Temperature Results:")
print(df_case2)
print("\n" * 2)

# Plotting Case 2
plt.figure(figsize=(10, 6))
for idx, a_value in enumerate(data_dict["a_values"]):
    plt.plot(data_dict["prob1"]["x_values_fine"],
        data_dict["prob1"]["case2_temp_results"][idx], label=f' = {a_value}')
plt.xlabel('Position (cm)')
plt.ylabel('Temperature (C)')
plt.legend()
plt.grid(True)
plt.savefig('prob1_case2_analytical_temperature_vs_position.png', dpi=300)
#plt.show()
plt.close()

# Comparison plots between cases for temperature distribution
fig, axes = plt.subplots(3, 2, figsize=(12, 12), sharex=True, sharey=True)

for idx, a_value in enumerate(data_dict["a_values"]):
    ax = axes[idx // 2, idx % 2]
    ax.plot(data_dict["prob1"]["x_values_fine"],
        data_dict["prob1"]["case1_temp_results"][idx], label='Case 1')
    ax.plot(data_dict["prob1"]["x_values_fine"],
        data_dict["prob1"]["case2_temp_results"][idx], label='Case 2')
    ax.set_title(f' = {a_value}')

```

```

ax.set_xlabel('Position (cm)')
ax.set_ylabel('Temperature (C)')
ax.legend()
ax.grid(True)

plt.tight_layout(rect=[0, 0, 1, 0.95]) # Adjust layout to make room for the main title
plt.savefig('prob1_comparison_cases.png', dpi=300)
#plt.show()
plt.close()

# Case 1 heat flux calculations
results = []
for a_value in data_dict["a_values"]:
    result = case1.analytical_heat_loss(a=a_value)
    results.append(result)

data_dict["prob1"]["case1_heat_flux_results"] = np.array(results)

# Create a DataFrame with alpha as the index and heat flux as the data
df_case1_heat_flux = pd.DataFrame(results, index=[f'a = {a}' for a in
    data_dict["a_values"]], columns=['Heat Flux'])

# Pretty print DataFrame for Case 1 heat flux
print("Analytical Case 1 Heat Flux Results:")
print(df_case1_heat_flux)
print("\n" * 2)

# Case 2 heat flux calculations
results = []
for a_value in data_dict["a_values"]:
    result = case2.analytical_heat_loss(a=a_value)
    results.append(result)

data_dict["prob1"]["case2_heat_flux_results"] = np.array(results)

# Create a DataFrame with alpha as the index and heat flux as the data
df_case2_heat_flux = pd.DataFrame(results, index=[f'a = {a}' for a in
    data_dict["a_values"]], columns=['Heat Flux'])

# Pretty print DataFrame for Case 2 heat flux
print("Analytical Case 2 Heat Flux Results:")
print(df_case2_heat_flux)
print("\n" * 2)

# Plotting heat flux vs alpha for both cases
plt.figure(figsize=(10, 6))
plt.plot(data_dict["a_values"], data_dict["prob1"]["case1_heat_flux_results"],
    label='Case 1')
plt.plot(data_dict["a_values"], data_dict["prob1"]["case2_heat_flux_results"],
    label='Case 2')

```

```

plt.xlabel('Alpha ( $\text{cm}^{-1}$ )')
plt.ylabel('Heat Flux (W)')
plt.legend()
plt.grid(True)
plt.savefig('prob1_heat_flux_comparison.png', dpi=300)
#plt.show()
plt.close()

def prob2():
    '''FDM with dx = 0.125 and alpha varying'''
    print("Problem 2")
    data_dict["prob2"] = {}
    data_dict["prob2"]["num_point"] = 8 # This does not consider the final T(L) point. So
        actually we have 9 points
    data_dict["prob2"]["dx"] = 0.125

    # Case 1
    results = []
    # num_point does not consider T(L). So "4" points is T0, T1, T2, T3, and then we add
        back T(L)
    for a_value in data_dict["a_values"]:
        fdm_array = case1.fdm_array(data_dict["prob2"]["num_point"], a = a_value)
        fdm_array_inverse = np.linalg.inv(fdm_array)

        right_array = case1.right_array(data_dict["prob2"]["num_point"])

        unknown_temps_array = fdm_array_inverse @ right_array
        solution = [0] # First point is 0 from BC
        solution[1:] = unknown_temps_array
        solution.append(100) # T(L) BC

        results.append(solution)

    data_dict["prob2"]["case1"] = {}
    data_dict["prob2"]["case1"]["temp_results"] = np.array(results)

    data_dict["prob2"]["x_values"] = np.array([0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75,
        0.875, 1.0])
    df_case1 = pd.DataFrame(results, index=[f'a = {a}' for a in data_dict["a_values"]],
        columns=[f'x = {x:.3f}' for x in data_dict["prob2"]["x_values"]])

    # Pretty print DataFrame for Case 1 temperature
    print("FDM Case 1 Temperature Results:")
    print(df_case1)
    print("\n" * 2)

    # Plotting Case 1 temperature
    plt.figure(figsize=(10, 6))
    for idx, a_value in enumerate(data_dict["a_values"]):

```

```

plt.plot(data_dict["prob2"]["x_values"],
         data_dict["prob2"]["case1"]["temp_results"][idx], label=f' = {a_value}')
plt.xlabel('Position (cm)')
plt.ylabel('Temperature (C)')
plt.legend()
plt.grid(True)
plt.savefig('prob2_case1_fdm_temperature_vs_position.png', dpi=300)
#plt.show()
plt.close()

# Case 2
results = []
# num_point does not consider T(L). So "4" points is T0, T1, T2, T3, and then we add
  back T(L)
for a_value in data_dict["a_values"]:
    fdm_array = case2.fdm_array(data_dict["prob2"]["num_point"], a = a_value)
    fdm_array_inverse = np.linalg.inv(fdm_array)

    right_array = case2.right_array(data_dict["prob2"]["num_point"])

    unkown_temps_array = fdm_array_inverse @ right_array

    solution = unkown_temps_array.tolist()
    solution.append(100) # T(L) BC

    results.append(solution)

data_dict["prob2"]["case2"] = {}
data_dict["prob2"]["case2"]["temp_results"] = np.array(results)

data_dict["prob2"]["x_values"] = np.array([0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75,
    0.875, 1.0])
df_case2 = pd.DataFrame(results, index=[f'a = {a}' for a in data_dict["a_values"]],
    columns=[f'x = {x:.3f}' for x in data_dict["prob2"]["x_values"]])

# Pretty print DataFrame for Case 2 temperature
print("FDM Case 2 Temperature Results:")
print(df_case2)
print("\n" * 2)

# Plotting Case 2 temperature
plt.figure(figsize=(10, 6))
for idx, a_value in enumerate(data_dict["a_values"]):
    plt.plot(data_dict["prob2"]["x_values"],
             data_dict["prob2"]["case2"]["temp_results"][idx], label=f' = {a_value}')
plt.xlabel('Position (cm)')
plt.ylabel('Temperature (C)')
plt.legend()
plt.grid(True)
plt.savefig('prob2_case2_fdm_temperature_vs_position.png', dpi=300)

```

```

plt.show()
plt.close()

# Plot the comparisons similar to problem 1
# Comparison plots between cases for temperature distribution
fig, axes = plt.subplots(3, 2, figsize=(12, 12), sharex=True, sharey=True)

for idx, a_value in enumerate(data_dict["a_values"]):
    ax = axes[idx // 2, idx % 2]
    ax.plot(data_dict["prob2"]["x_values"],
            data_dict["prob2"]["case1"]["temp_results"][idx], label='Case 1')
    ax.plot(data_dict["prob2"]["x_values"],
            data_dict["prob2"]["case2"]["temp_results"][idx], label='Case 2')
    ax.set_title(f' = {a_value}')
    ax.set_xlabel('Position (cm)')
    ax.set_ylabel('Temperature (C)')
    ax.legend()
    ax.grid(True)

plt.tight_layout(rect=[0, 0, 1, 0.95]) # Adjust layout to make room for the main title
plt.savefig('prob2_comparison_cases.png', dpi=300)
plt.show()
plt.close()

# Heat flux extraction
# Case 1
results = []
for i in range(len(data_dict["a_values"])):
    a = data_dict["a_values"][i]
    (t_0, t_1) = data_dict["prob2"]["case1"]["temp_results"][i][-2:]
    dx = data_dict["prob2"]["dx"]
    first_order = common.taylor_extraction(t_0, t_1, dx, a, 1)
    second_order = common.taylor_extraction(t_0, t_1, dx, a, 2)
    results.append([first_order, second_order])

data_dict["prob2"]["case1"]["heat_results"] = np.array(results)

# Create a DataFrame with alpha as the index and heat flux as the data
df_case1_heat_flux = pd.DataFrame(results, index=[f'a = {a}' for a in
    data_dict["a_values"]], columns=['1st Order', '2nd Order'])

# Pretty print DataFrame for Case 1 heat flux
print("FDM Case 1 Heat Flux Results:")
print(df_case1_heat_flux)
print("\n" * 2)

# Plotting heat flux vs alpha for both extractions

```

```

plt.figure(figsize=(10, 6))
plt.plot(data_dict["a_values"], [i[0] for i in
    data_dict["prob2"]["case1"]["heat_results"]], label='1st Order Extraction')
plt.plot(data_dict["a_values"], [i[1] for i in
    data_dict["prob2"]["case1"]["heat_results"]], label='2nd Order Extraction')
plt.xlabel('Alpha ( $\text{cm}^{-1}$ )')
plt.ylabel('Heat Flux (W)')
plt.legend()
plt.grid(True)
plt.savefig('prob2_case1_heat_flux.png', dpi=300)
#plt.show()
plt.close()

```

```

# Case 2
results = []
for i in range(len(data_dict["a_values"])):
    a = data_dict["a_values"][i]
    (t_0, t_1) = data_dict["prob2"]["case2"]["temp_results"][i][-2:]
    dx = data_dict["prob2"]["dx"]
    first_order = common.taylor_extraction(t_0, t_1, dx, a, 1)
    second_order = common.taylor_extraction(t_0, t_1, dx, a, 2)
    results.append([first_order, second_order])

data_dict["prob2"]["case2"]["heat_results"] = np.array(results)

# Create a DataFrame with alpha as the index and heat flux as the data
df_case2_heat_flux = pd.DataFrame(results, index=[f'a = {a}' for a in
    data_dict["a_values"]], columns=['1st Order', '2nd Order'])

# Pretty print DataFrame for Case 1 heat flux
print("FDM Case 2 Heat Flux Results:")
print(df_case2_heat_flux)
print("\n" * 2)

# Plotting heat flux vs alpha for both extractions
plt.figure(figsize=(10, 6))
plt.plot(data_dict["a_values"], [i[0] for i in
    data_dict["prob2"]["case2"]["heat_results"]], label='1st Order Extraction')
plt.plot(data_dict["a_values"], [i[1] for i in
    data_dict["prob2"]["case2"]["heat_results"]], label='2nd Order Extraction')
plt.xlabel('Alpha ( $\text{cm}^{-1}$ )')
plt.ylabel('Heat Flux (W)')
plt.legend()
plt.grid(True)
plt.savefig('prob2_case2_heat_flux.png', dpi=300)
#plt.show()
plt.close()

```

```

def prob3():
    '''FEM with dx = 0.125 and alpha varying'''
    print("Problem 3")
    data_dict["prob3"] = {}
    data_dict["prob3"]["num_point"] = 8 # This does not consider the final T(L) point. So
        actually we have 9 points
    data_dict["prob3"]["dx"] = 0.125

    # Case 1
    results = []
    # num_point does not consider T(L). So "4" points is T0, T1, T2, T3, and then we add
        back T(L)
    for a_value in data_dict["a_values"]:
        fdm_array = case1.fem_array(data_dict["prob3"]["num_point"], a = a_value)
        fdm_array_inverse = np.linalg.inv(fdm_array)

        right_array = case1.right_array(data_dict["prob3"]["num_point"])

        unkown_temps_array = fdm_array_inverse @ right_array
        solution = [0] # First point is 0 from BC
        solution[1:] = unkown_temps_array
        solution.append(100) # T(L) BC

        results.append(solution)

    data_dict["prob3"]["case1"] = {}
    data_dict["prob3"]["case1"]["temp_results"] = np.array(results)

    data_dict["prob3"]["x_values"] = np.array([0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75,
        0.875, 1.0])
    df_case1 = pd.DataFrame(results, index=[f'a = {a}' for a in data_dict["a_values"]],
        columns=[f'x = {x:.3f}' for x in data_dict["prob3"]["x_values"]])

    # Pretty print DataFrame for Case 1 temperature
    print("FEM Case 1 Temperature Results:")
    print(df_case1)
    print("\n" * 2)

    # Plotting Case 1 temperature
    plt.figure(figsize=(10, 6))
    for idx, a_value in enumerate(data_dict["a_values"]):
        plt.plot(data_dict["prob3"]["x_values"],
            data_dict["prob3"]["case1"]["temp_results"][idx], label=f' = {a_value}')
    plt.xlabel('Position (cm)')
    plt.ylabel('Temperature (C)')
    plt.legend()
    plt.grid(True)
    plt.savefig('prob3_case1_fem_temperature_vs_position.png', dpi=300)
    #plt.show()
    plt.close()

```

```

# Case 2
results = []
# num_point does not consider T(L). So "4" points is T0, T1, T2, T3, and then we add
  back T(L)
for a_value in data_dict["a_values"]:
    fdm_array = case2.fem_array(data_dict["prob3"]["num_point"], a = a_value)
    fdm_array_inverse = np.linalg.inv(fdm_array)

    right_array = case2.right_array(data_dict["prob3"]["num_point"])

    unkown_temps_array = fdm_array_inverse @ right_array

    solution = unkown_temps_array.tolist()
    solution.append(100) # T(L) BC

    results.append(solution)

data_dict["prob3"]["case2"] = {}
data_dict["prob3"]["case2"]["temp_results"] = np.array(results)

data_dict["prob3"]["x_values"] = np.array([0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75,
    0.875, 1.0])
df_case2 = pd.DataFrame(results, index=[f'a = {a}' for a in data_dict["a_values"]],
    columns=[f'x = {x:.3f}' for x in data_dict["prob3"]["x_values"]])

# Pretty print DataFrame for Case 2 temperature
print("FEM Case 2 Temperature Results:")
print(df_case2)
print("\n" * 2)

# Plotting Case 2 temperature
plt.figure(figsize=(10, 6))
for idx, a_value in enumerate(data_dict["a_values"]):
    plt.plot(data_dict["prob3"]["x_values"],
        data_dict["prob3"]["case2"]["temp_results"][idx], label=f' = {a_value}')
plt.xlabel('Position (cm)')
plt.ylabel('Temperature (C)')
plt.legend()
plt.grid(True)
plt.savefig('prob3_case2_fem_temperature_vs_position.png', dpi=300)
#plt.show()
plt.close()

# Plot the comparisons similar to problem 1
# Comparison plots between cases for temperature distribution
fig, axes = plt.subplots(3, 2, figsize=(12, 12), sharex=True, sharey=True)

for idx, a_value in enumerate(data_dict["a_values"]):

```

```

ax = axes[idx // 2, idx % 2]
ax.plot(data_dict["prob3"]["x_values"],
        data_dict["prob3"]["case1"]["temp_results"][idx], label='Case 1')
ax.plot(data_dict["prob3"]["x_values"],
        data_dict["prob3"]["case2"]["temp_results"][idx], label='Case 2')
ax.set_title(f' = {a_value}')
ax.set_xlabel('Position (cm)')
ax.set_ylabel('Temperature (C)')
ax.legend()
ax.grid(True)

plt.tight_layout(rect=[0, 0, 1, 0.95]) # Adjust layout to make room for the main title
plt.savefig('prob3_comparison_cases.png', dpi=300)
plt.show()
plt.close()

# Heat flux extraction
# Case 1
results = []
for i in range(len(data_dict["a_values"])):
    a = data_dict["a_values"][i]
    (t_0, t_1) = data_dict["prob3"]["case1"]["temp_results"][i][-2:]
    dx = data_dict["prob3"]["dx"]
    result = common.fem_heat_extraction(t_0, t_1, dx, a)
    results.append([result])

data_dict["prob3"]["case1"]["heat_results"] = np.array(results)

# Create a DataFrame with alpha as the index and heat flux as the data
df_case1_heat_flux = pd.DataFrame(results, index=[f'a = {a}' for a in
    data_dict["a_values"]], columns=['Heat Extraction'])

# Pretty print DataFrame for Case 1 heat flux
print("FEM Case 1 Heat Flux Results:")
print(df_case1_heat_flux)
print("\n" * 2)

# Plotting heat flux vs alpha for both extractions
plt.figure(figsize=(10, 6))
plt.plot(data_dict["a_values"], [i[0] for i in
    data_dict["prob3"]["case1"]["heat_results"]], label='Heat Extraction')
plt.xlabel('Alpha (cm-1)')
plt.ylabel('Heat Flux (W)')
plt.legend()
plt.grid(True)
plt.savefig('prob3_case1_heat_flux.png', dpi=300)
plt.show()
plt.close()

```

```

# Case 2
results = []
for i in range(len(data_dict["a_values"])):
    a = data_dict["a_values"][i]
    (t_0, t_1) = data_dict["prob3"]["case2"]["temp_results"][i][-2:]
    dx = data_dict["prob3"]["dx"]
    result = common.fem_heat_extraction(t_0, t_1, dx, a)
    results.append([result])

data_dict["prob3"]["case2"]["heat_results"] = np.array(results)

# Create a DataFrame with alpha as the index and heat flux as the data
df_case2_heat_flux = pd.DataFrame(results, index=[f'a = {a}' for a in
    data_dict["a_values"]], columns=['Heat Extraction'])

# Pretty print DataFrame for Case 1 heat flux
print("FEM Case 2 Heat Flux Results:")
print(df_case2_heat_flux)
print("\n" * 2)

# Plotting heat flux vs alpha for both extractions
plt.figure(figsize=(10, 6))
plt.plot(data_dict["a_values"], [i[0] for i in
    data_dict["prob3"]["case2"]["heat_results"]], label='Heat Extraction')
plt.xlabel('Alpha (cm-1)')
plt.ylabel('Heat Flux (W)')
plt.legend()
plt.grid(True)
plt.savefig('prob3_case2_heat_flux.png', dpi=300)
plt.show()
plt.close()

def prob4():
    '''Convergence analysis of the cases'''
    print("Problem 4")
    data_dict["prob4"] = {}
    data_dict["prob4"]["num_points"] = [2**i for i in range(2, 8)] # This does not
        consider the final T(L) point. So actually we have n + 1 points
    data_dict["prob4"]["dx_values"] = [1 / i for i in data_dict["prob4"]["num_points"]]

    # Case 1
    # Get fdm temp results
    data_dict["prob4"]["case1"] = {}
    data_dict["prob4"]["case1"]["fdm"] = {}
    data_dict["prob4"]["case1"]["fdm"]["temp_results"] = []
    # num_point does not consider T(L). So "4" points is T0, T1, T2, T3, and then we add
        back T(L)

```

```

for num_point in data_dict["prob4"]["num_points"]:
    fdm_array = case1.fdm_array(num_point)
    fdm_array_inverse = np.linalg.inv(fdm_array)

    right_array = case1.right_array(num_point)

    unkown_temps_array = fdm_array_inverse @ right_array
    solution = [0] # First point is 0 from BC
    solution[1:] = unkown_temps_array
    solution.append(100) # T(L) BC

    data_dict["prob4"]["case1"]["fdm"]["temp_results"].append(solution)

# Get fem temp results
data_dict["prob4"]["case1"]["fem"] = {}
data_dict["prob4"]["case1"]["fem"]["temp_results"] = []
# num_point does not consider T(L). So "4" points is T0, T1, T2, T3, and then we add
# back T(L)
for num_point in data_dict["prob4"]["num_points"]:
    fdm_array = case1.fem_array(num_point)
    fdm_array_inverse = np.linalg.inv(fdm_array)

    right_array = case1.right_array(num_point)

    unkown_temps_array = fdm_array_inverse @ right_array
    solution = [0] # First point is 0 from BC
    solution[1:] = unkown_temps_array
    solution.append(100) # T(L) BC

    data_dict["prob4"]["case1"]["fem"]["temp_results"].append(solution)

# We have all of the temperature results for varying dx for both fdm and fem
# For each dx, get the midpoint temperature from fdm and fem
results = []
for i in range(len(data_dict["prob4"]["num_points"])):
    a = 2.75
    num_points = data_dict["prob4"]["num_points"][i]
    midpoint_index = int(num_points / 2)
    t_fdm = data_dict["prob4"]["case1"]["fdm"]["temp_results"][i][midpoint_index]
    t_fem = data_dict["prob4"]["case1"]["fem"]["temp_results"][i][midpoint_index]

    results.append([t_fdm, t_fem])

data_dict["prob4"]["case1"]["midpoint_temp_results"] = np.array(results)
data_dict["prob4"]["case1"]["midpoint_temp_true"] = case1.analytical(a = 2.75, x = 0.5)

# Calculate % error and beta values for midpoint temperature
errors = []
betas = []
t_exact = data_dict["prob4"]["case1"]["midpoint_temp_true"]

```

```

t_results = data_dict["prob4"]["case1"]["midpoint_temp_results"]

for i in range(0, len(t_results)):
    # % Error calculation
    t_fdm, t_fem = results[i]
    fdm_error = common.calc_error(t_exact, t_fdm)
    fem_error = common.calc_error(t_exact, t_fem)

    # Beta (convergence rate) calculation
    if i != 0:
        t_fdm_prev, t_fem_prev = results[i-1]
        dx_1 = data_dict["prob4"]["dx_values"][i-1]
        dx_2 = data_dict["prob4"]["dx_values"][i]
        fdm_beta = common.calc_beta(t_exact, t_fdm, t_fdm_prev, dx_2, dx_1)
        fem_beta = common.calc_beta(t_exact, t_fem, t_fem_prev, dx_2, dx_1)
    else:
        fdm_beta = 0
        fem_beta = 0

    errors.append([fdm_error, fem_error])
    betas.append([fdm_beta, fem_beta])

data_dict["prob4"]["case1"]["temp_errors"] = errors
data_dict["prob4"]["case1"]["temp_betas"] = betas

# Combine dx values, results, errors, and betas into a DataFrame for display
table_data = []
for i in range(len(data_dict["prob4"]["dx_values"])):
    table_data.append([
        results[i][0], # fdm result
        results[i][1], # fem result
        data_dict["prob4"]["case1"]["midpoint_temp_true"], # True T
        errors[i][0], # fdm % error
        errors[i][1], # fem % error
        betas[i][0], # fdm beta
        betas[i][1] # fem beta
    ])

columns = ['FDM T', 'FEM T', 'True T', 'FDM % Error', 'FEM % Error', 'FDM ', 'FEM ']
df_case1 = pd.DataFrame(table_data, index=[f'dx = {i}' for i in
    data_dict["prob4"]["dx_values"]], columns=columns)

print("FDM and FEM Case 1 Midpoint Temperature Convergence Results:")
print(df_case1)
print("\n" * 2)
# Plotting
plt.figure(figsize=(8, 6))
plt.plot(data_dict["prob4"]["dx_values"], [err[0] for err in
    data_dict["prob4"]["case1"]["temp_errors"]], label='FDM Temp Error')

```

```

plt.plot(data_dict["prob4"]["dx_values"], [err[1] for err in
    data_dict["prob4"]["case1"]["temp_errors"]], label='FEM Temp Error')
plt.xscale('log')
plt.yscale('log')
plt.xlabel('x')
plt.ylabel('% Error')
plt.legend()
plt.grid(True)
plt.savefig("prob4_case1_temp_convergence.png", dpi=300)
#plt.show()
plt.close()

# For each dx, calculate the 2nd order heat flux for fdm and the heat flux for fem
results = []
for i in range(len(data_dict["prob4"]["dx_values"])):
    a = 2.75
    dx = data_dict["prob4"]["dx_values"][i]
    (t_0_fdm, t_1_fdm) = data_dict["prob4"]["case1"]["fdm"]["temp_results"][i][-2:]
    (t_0_fem, t_1_fem) = data_dict["prob4"]["case1"]["fem"]["temp_results"][i][-2:]

    fdm = common.taylor_extraction(t_0_fdm, t_1_fdm, dx, a, 2)
    fem = common.fem_heat_extraction(t_0_fem, t_1_fem, dx, a)
    results.append([fdm, fem])

data_dict["prob4"]["case1"]["heat_results"] = np.array(results)
data_dict["prob4"]["case1"]["heat_results_true"] = case1.analytical_heat_loss(a=2.75)

# Calculate % error and beta values for heat loss
errors = []
betas = []
q_exact = data_dict["prob4"]["case1"]["heat_results_true"]
q_results = data_dict["prob4"]["case1"]["heat_results"]

for i in range(0, len(q_results)):
    # % Error calculation
    q_fdm, q_fem = results[i]
    fdm_error = common.calc_error(q_exact, q_fdm)
    fem_error = common.calc_error(q_exact, q_fem)

    # Beta (convergence rate) calculation
    if i != 0:
        q_fdm_prev, q_fem_prev = results[i-1]
        dx_1 = data_dict["prob4"]["dx_values"][i-1]
        dx_2 = data_dict["prob4"]["dx_values"][i]
        fdm_beta = common.calc_beta(q_exact, q_fdm, q_fdm_prev, dx_2, dx_1)
        fem_beta = common.calc_beta(q_exact, q_fem, q_fem_prev, dx_2, dx_1)
    else:
        fdm_beta = 0

```

```

    fem_beta = 0

    errors.append([fdm_error, fem_error])
    betas.append([fdm_beta, fem_beta])

data_dict["prob4"]["case1"]["heat_errors"] = errors
data_dict["prob4"]["case1"]["heat_betas"] = betas

# Combine dx values, results, errors, and betas into a DataFrame for display
table_data = []
for i in range(len(data_dict["prob4"]["dx_values"])):
    table_data.append([
        results[i][0], # fdm result
        results[i][1], # fem result
        q_exact, # True Q
        errors[i][0], # fdm % error
        errors[i][1], # fem % error
        betas[i][0], # fdm beta
        betas[i][1] # fem beta
    ])

columns = ['FDM Q', 'FEM Q', 'True Q', 'FDM % Error', 'FEM % Error', 'FDM ', 'FEM ']
df_case1 = pd.DataFrame(table_data, index=[f'dx = {i}' for i in
    data_dict["prob4"]["dx_values"]], columns=columns)

print("FDM and FEM Case 1 Heat Loss Convergence Results:")
print(df_case1)
print("\n" * 2)

# Plotting
plt.figure(figsize=(8, 6))
plt.plot(data_dict["prob4"]["dx_values"], [err[0] for err in
    data_dict["prob4"]["case1"]["heat_errors"]], label='FDM Heat Flux Error')
plt.plot(data_dict["prob4"]["dx_values"], [err[1] for err in
    data_dict["prob4"]["case1"]["heat_errors"]], label='FEM Heat Flux Error')
plt.xscale('log')
plt.yscale('log')
plt.xlabel('x')
plt.ylabel('% Error')
plt.legend()
plt.grid(True)
plt.savefig("prob4_case1_heat_convergence.png", dpi=300)
plt.show()
plt.close()

# Case 2
# Get fdm temp results

```

```

data_dict["prob4"]["case2"] = {}
data_dict["prob4"]["case2"]["fdm"] = {}
data_dict["prob4"]["case2"]["fdm"]["temp_results"] = []
# num_point does not consider T(L). So "4" points is T0, T1, T2, T3, and then we add
    back T(L)
for num_point in data_dict["prob4"]["num_points"]:
    fdm_array = case2.fdm_array(num_point)
    fdm_array_inverse = np.linalg.inv(fdm_array)

    right_array = case2.right_array(num_point)

    unkown_temps_array = fdm_array_inverse @ right_array
    solution = []
    solution[0:] = unkown_temps_array
    solution.append(100) # T(L) BC

    data_dict["prob4"]["case2"]["fdm"]["temp_results"].append(solution)

# Get fem temp results
data_dict["prob4"]["case2"]["fem"] = {}
data_dict["prob4"]["case2"]["fem"]["temp_results"] = []
# num_point does not consider T(L). So "4" points is T0, T1, T2, T3, and then we add
    back T(L)
for num_point in data_dict["prob4"]["num_points"]:
    fdm_array = case2.fem_array(num_point)
    fdm_array_inverse = np.linalg.inv(fdm_array)

    right_array = case2.right_array(num_point)

    unkown_temps_array = fdm_array_inverse @ right_array
    solution = []
    solution[0:] = unkown_temps_array
    solution.append(100) # T(L) BC

    data_dict["prob4"]["case2"]["fem"]["temp_results"].append(solution)

# We have all of the temperature results for varying dx for both fdm and fem
# For each dx, get the midpoint temperature from fdm and fem
results = []
for i in range(len(data_dict["prob4"]["num_points"])):
    a = 2.75
    num_points = data_dict["prob4"]["num_points"][i]
    midpoint_index = int(num_points / 2)
    t_fdm = data_dict["prob4"]["case2"]["fdm"]["temp_results"][i][midpoint_index]
    t_fem = data_dict["prob4"]["case2"]["fem"]["temp_results"][i][midpoint_index]

    results.append([t_fdm, t_fem])

data_dict["prob4"]["case2"]["midpoint_temp_results"] = np.array(results)
data_dict["prob4"]["case2"]["midpoint_temp_true"] = case2.analytical(a = 2.75, x = 0.5)

```

```

# Calculate % error and beta values for midpoint temperature
errors = []
betas = []
t_exact = data_dict["prob4"]["case2"]["midpoint_temp_true"]
t_results = data_dict["prob4"]["case2"]["midpoint_temp_results"]

for i in range(0, len(t_results)):
    # % Error calculation
    t_fdm, t_fem = results[i]
    fdm_error = common.calc_error(t_exact, t_fdm)
    fem_error = common.calc_error(t_exact, t_fem)

    # Beta (convergence rate) calculation
    if i != 0:
        t_fdm_prev, t_fem_prev = results[i-1]
        dx_1 = data_dict["prob4"]["dx_values"][i-1]
        dx_2 = data_dict["prob4"]["dx_values"][i]
        fdm_beta = common.calc_beta(t_exact, t_fdm, t_fdm_prev, dx_2, dx_1)
        fem_beta = common.calc_beta(t_exact, t_fem, t_fem_prev, dx_2, dx_1)
    else:
        fdm_beta = 0
        fem_beta = 0

    errors.append([fdm_error, fem_error])
    betas.append([fdm_beta, fem_beta])

data_dict["prob4"]["case2"]["temp_errors"] = errors
data_dict["prob4"]["case2"]["temp_betas"] = betas

# Combine dx values, results, errors, and betas into a DataFrame for display
table_data = []
for i in range(len(data_dict["prob4"]["dx_values"])):
    table_data.append([
        results[i][0], # fdm result
        results[i][1], # fem result
        data_dict["prob4"]["case2"]["midpoint_temp_true"], # True T
        errors[i][0], # fdm % error
        errors[i][1], # fem % error
        betas[i][0], # fdm beta
        betas[i][1] # fem beta
    ])

columns = ['FDM T', 'FEM T', 'True T', 'FDM % Error', 'FEM % Error', 'FDM ', 'FEM ']
df_case2 = pd.DataFrame(table_data, index=[f'dx = {i}' for i in
    data_dict["prob4"]["dx_values"]], columns=columns)

print("FDM and FEM Case 2 Midpoint Temperature Convergence Results:")
print(df_case2)
print("\n" * 2)

```

```

# Plotting
plt.figure(figsize=(8, 6))
plt.plot(data_dict["prob4"]["dx_values"], [err[0] for err in
    data_dict["prob4"]["case2"]["temp_errors"]], label='FDM Temp Error')
plt.plot(data_dict["prob4"]["dx_values"], [err[1] for err in
    data_dict["prob4"]["case2"]["temp_errors"]], label='FEM Temp Error')
plt.xscale('log')
plt.yscale('log')
plt.xlabel('x')
plt.ylabel('% Error')
plt.legend()
plt.grid(True)
plt.savefig("prob4_case2_temp_convergence.png", dpi=300)
plt.show()
plt.close()

# For each dx, calculate the 2nd order heat flux for fdm and the heat flux for fem
results = []
for i in range(len(data_dict["prob4"]["dx_values"])):
    a = 2.75
    dx = data_dict["prob4"]["dx_values"][i]
    (t_0_fdm, t_1_fdm) = data_dict["prob4"]["case2"]["fdm"]["temp_results"][i][-2:]
    (t_0_fem, t_1_fem) = data_dict["prob4"]["case2"]["fem"]["temp_results"][i][-2:]

    fdm = common.taylor_extraction(t_0_fdm, t_1_fdm, dx, a, 2)
    fem = common.fem_heat_extraction(t_0_fem, t_1_fem, dx, a)
    results.append([fdm, fem])

data_dict["prob4"]["case2"]["heat_results"] = np.array(results)
data_dict["prob4"]["case2"]["heat_results_true"] = case2.analytical_heat_loss(a=2.75)

# Calculate % error and beta values for heat loss
errors = []
betas = []
q_exact = data_dict["prob4"]["case2"]["heat_results_true"]
q_results = data_dict["prob4"]["case2"]["heat_results"]

for i in range(0, len(q_results)):
    # % Error calculation
    q_fdm, q_fem = results[i]
    fdm_error = common.calc_error(q_exact, q_fdm)
    fem_error = common.calc_error(q_exact, q_fem)

    # Beta (convergence rate) calculation
    if i != 0:
        q_fdm_prev, q_fem_prev = results[i-1]
        dx_1 = data_dict["prob4"]["dx_values"][i-1]
        dx_2 = data_dict["prob4"]["dx_values"][i]

```

```

        fdm_beta = common.calc_beta(q_exact, q_fdm, q_fdm_prev, dx_2, dx_1)
        fem_beta = common.calc_beta(q_exact, q_fem, q_fem_prev, dx_2, dx_1)
    else:
        fdm_beta = 0
        fem_beta = 0

    errors.append([fdm_error, fem_error])
    betas.append([fdm_beta, fem_beta])

data_dict["prob4"]["case2"]["heat_errors"] = errors
data_dict["prob4"]["case2"]["heat_betas"] = betas

# Combine dx values, results, errors, and betas into a DataFrame for display
table_data = []
for i in range(len(data_dict["prob4"]["dx_values"])):
    table_data.append([
        results[i][0], # fdm result
        results[i][1], # fem result
        q_exact, # True Q
        errors[i][0], # fdm % error
        errors[i][1], # fem % error
        betas[i][0], # fdm beta
        betas[i][1] # fem beta
    ])

columns = ['FDM Q', 'FEM Q', 'True Q', 'FDM % Error', 'FEM % Error', 'FDM ', 'FEM ']
df_case2 = pd.DataFrame(table_data, index=[f'dx = {i}' for i in
    data_dict["prob4"]["dx_values"]], columns=columns)

print("FDM and FEM Case 2 Heat Loss Convergence Results:")
print(df_case2)
print("\n" * 2)

# Plotting
plt.figure(figsize=(8, 6))
plt.plot(data_dict["prob4"]["dx_values"], [err[0] for err in
    data_dict["prob4"]["case2"]["heat_errors"]], label='FDM Heat Flux Error')
plt.plot(data_dict["prob4"]["dx_values"], [err[1] for err in
    data_dict["prob4"]["case2"]["heat_errors"]], label='FEM Heat Flux Error')
plt.xscale('log')
plt.yscale('log')
plt.xlabel('x')
plt.ylabel('% Error')
plt.legend()
plt.grid(True)
plt.savefig("prob4_case2_heat_convergence.png", dpi=300)
#plt.show()
plt.close()

```

```

if __name__ == "__main__":
    print("Homework 1 by Judson Upchurch")
    print("Please note that some of the tables are transposed compared to the LaTeX
          equivalent")
    print("")

    prob1() # Analytical temperature distribution

    prob2() # FDM with dx = 0.125 and varying alpha

    prob3() # FEM with dx = 0.125 and varying alpha

    prob4() # Convergence analysis

```

5.2 case1.py

```

# case1.py

import numpy as np

k = 0.5          # W / (cm K)
area = np.pi / 100 # cm^2

def analytical(a, x):
    return 100 * np.sinh(a*x) / np.sinh(a)

def analytical_heat_loss(a):
    return -100*k*area*a/np.tanh(a)

def fdm_array(n, a = 2.75):
    dx = 1 / n
    kap = 2 + ((a**2)*(dx**2))

    left_FDM_array = np.zeros((n-1, n-1))
    left_FDM_array[0][0:2] = [kap, -1]
    row_FDM = [-1, kap, -1]

    for i in range(1, n-2):
        left_FDM_array[i][i-1:i+2] = row_FDM

    left_FDM_array[n-2][n-3:n-1] = [-1, kap]

    return left_FDM_array

def fem_array(n, a = 2.75):
    dx = 1 / n
    kap = (2/dx + 4*a**2*dx/6) / (1/dx - a**2*dx/6)

```

```

left_FDM_array = np.zeros((n-1, n-1))
left_FDM_array[0][0:2] = [kap, -1]
row_FDM = [-1, kap, -1]

for i in range(1, n-2):
    left_FDM_array[i][i-1:i+2] = row_FDM

left_FDM_array[n-2][n-3:n-1] = [-1, kap]

return left_FDM_array

def right_array(n):
    array = [0 for _ in range(n-2)]
    array.append(100)
    return array

```

5.3 case2.py

```

# case2.py

import numpy as np

k = 0.5          # W / (cm K)
area = np.pi / 100 # cm^2

def analytical(a, x):
    return 100 * np.cosh(a*x) / np.cosh(a)

def analytical_heat_loss(a):
    return -100*k*area*a*np.tanh(a)

def fdm_array(n, a = 2.75):
    dx = 1 / n
    kap = 2 + ((a**2)*(dx**2))
    kap_prime = (kap/2)

    left_FDM_array = np.zeros((n, n))
    left_FDM_array[0][0:2] = [kap_prime, -1]
    row_FDM = [-1, kap, -1]

    for i in range(1, n-1):
        left_FDM_array[i][i-1:i+2] = row_FDM

    left_FDM_array[n-1][n-2:n] = [-1, kap]

    return left_FDM_array

```

```

def fem_array(n, a = 2.75):
    dx = 1 / n
    kap = (2/dx + 4*a**2*dx/6) / (1/dx - a**2*dx/6)
    kap_prime = (kap/2)

    left_FDM_array = np.zeros((n, n))
    left_FDM_array[0][0:2] = [kap_prime, -1]
    row_FDM = [-1, kap, -1]

    for i in range(1, n-1):
        left_FDM_array[i][i-1:i+2] = row_FDM

    left_FDM_array[n-1][n-2:n] = [-1, kap]

    return left_FDM_array

def right_array(n):
    array = [0 for _ in range(n-1)]
    array.append(100)
    return array

```

5.4 common.py

```

# common.py

import numpy as np

k = 0.5          # W / (cm K)
area = np.pi / 100  # cm^2

def taylor_extraction(t_0, t_1, dx, a=0, order=1):
    if order == 1:
        return -1 * k * area * (t_1 - t_0) / dx
    elif order == 2:
        return -1 * k * area * (((t_1 - t_0) / dx) + (a**2 * dx * t_1 / 2))

def calc_error(q_exact, q_1):
    return np.abs((q_exact - q_1) / q_exact)

def calc_beta(q_exact, q_1, q_2, dx_1, dx_2):
    return np.log(np.abs((q_exact - q_1)/(q_exact - q_2))) / np.log(dx_1 / dx_2)

def fem_heat_extraction(t_1, t_2, dx, a = 2.75):
    term_1 = (-1/dx + a**2*dx/6) * t_1
    term_2 = (1/dx + 2*a**2*dx/6) * t_2
    return -1 * k * area * (term_1 + term_2)

```
