Assignment 3

Texas A&M University
AERO-430-500 Numerical Simulation
Andrew Hollister

November 9, 2021

Due: November 11, 2021

Contents

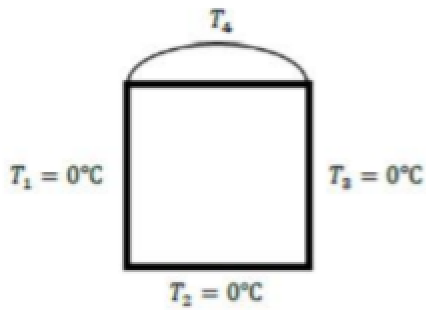
1 Abstract

The purpose of this assignment was to solve the problem of the two-dimensional orthotropic heat conduction problem for the case of an infinite bar under Dirichlet boundary conditions. The analytical solution was derived and was used to investigate the accuracy of the Finite Element Method (FDM). The FDM solution was derived at an accuracy of a second order solution and its convergence was determined at various values of thermal conductivity and number of nodes used.

2 Analytical Solution

2.1 The Orthotropic Diffusion Equation

A visualization of the problem to be solved within this report is given below:



For the first case, the following boundary conditions apply:

$$T_1(0, y) = T_2(x, 0) = T_3(1, y) = 0°C$$

$$T_4(x, 1) = 100 * \sin(\pi x)\, °C$$

Likewise, for the second case, the boundary conditions are:

$$T_1(0, y) = T_2(x, 0) = T_3(1, y) = 0°C$$

$$T_4(x, 1) = \frac{du}{dy} = K\pi * \cosh(K\pi) * 100 * \frac{\sin(\pi x)}{\sinh\,(K\pi)}\, °C$$

Where K is a conductivity constant.

The equations that describe heat flow across the surface are:

$$\nabla Q = 0$$

$$Q = -K_x * \frac{du}{dx}, 0, -K_y * \frac{du}{dx}$$

$$\nabla Q = -K_{xx} * \frac{d^2u}{dx^2} - K_{yy} * \frac{d^2u}{dy^2} = 0$$

Simplifying this expression by expressing $\frac{K_{xx}}{K_{yy}}$ as K^2 yields:

$$K^2 * \frac{d^2u}{dx^2}(x,y) + \frac{d^2u}{dy^2}(x,y) = 0$$

Separation of variables must now take place in order to solve the Dirichlet 2nd order homogenous differential equation. Separating the heat function into two separate functions, one for each variable turns the function $u(x,y)$ into $f(x)g(y)$. Substituting this into the partial differential equation will yield the following:

$$\frac{1}{f} * \frac{d^2f}{dx^2} + \frac{1}{K^2} * \frac{1}{g} * \frac{d^2g}{dy^2} = 0 \; or - \frac{1}{f} * \frac{d^2f}{dx^2} = \frac{1}{K^2} * \frac{1}{g} * \frac{d^2g}{dy^2} = \lambda$$

2.2 Boundary Conditions

Applying the boundary conditions for case 1 produces:

$$f(0)g(y) = f(1)g(y) = 0°C$$

$$f(x)g(0) = 0°C, f(x)g(1) = 100 * \sin(\pi x)°C$$

Applying boundary conditions for case 2 yields:

$$f(0)g(y) = f(1)g(y) = 0°C$$

$$f(x)g(0) = 0°C, f(x)g'^{(1)} = K\pi * \cosh(K\pi) * 100 * \frac{\sin(\pi x)}{\sinh(K\pi)}°C$$

The general solutions for these functions can be derived by utilizing these boundary conditions.

$$f(x) = \sin(\pi x)$$

$$g(y) = 100 * \frac{\sinh(K\pi y)}{\sinh(K\pi)}$$

Multiplying these equations together produces the general solution to the heat equation:

$$u(x,y) = 100 * \frac{\sinh(K\pi y)}{\sinh(K\pi)}$$

3 Numerical Solution

The following section outlines the procedure for deriving the numerical solution utilized within this analysis.

3.1 Second Order Central Difference Scheme FDM

The second order FDM solution derivation commences with an Ordinary Taylor Series Expansion.

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{3!}f''' + \cdots$$
$$f(x) = f(x)$$
$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{3!}f''' + \cdots$$

Adapting this equation for a point in 2D space.

$$U_{i-\Delta x,j} = U_{i,j} - \Delta x U'_{i,j} + \frac{\Delta x^2}{2}U''_{i,j} + \cdots$$
$$U_{i,j} = U_{i,j}$$
$$U_{i+\Delta x,j} = U_{i,j} + \Delta x U'_{i,j} + \frac{\Delta x^2}{2}U''_{i,j} + \cdots$$
$$U_{i,j-\Delta y} = U_{i,j} - \Delta y U'_{i,j} + \frac{\Delta y^2}{2}U''_{i,j} + \cdots$$
$$U_{i,j+\Delta y} = U_{i,j} + \Delta y U'_{i,j} + \frac{\Delta y^2}{2}U''_{i,j} + \cdots$$

Subtracting the Taylor series yields the second order approximation of U':

$$U_{i,j-\Delta y} - U_{i,j+\Delta y} = U_{i,j} - \Delta y U'_{i,j} + \frac{\Delta y^2}{2}U''_{i,j} - \left(U_{i,j} + \Delta y U'_{i,j} + \frac{\Delta y^2}{2}U''_{i,j} + \cdots\right)$$

Dividing both sides of this equation by $2\Delta y$ produces $\frac{dU}{dy}$:

$$U'_{i,j} \approx \frac{U_{i,j-\Delta y} - U_{i,j+\Delta y}}{2\Delta y}$$

Applying a similar process of adding Taylor Series will result in U'':

$$\frac{d^2U}{dx^2} \approx \frac{U_{i+\Delta x,j} + U_{i-\Delta x,j} - 2U_{i,j}}{\Delta x^2}$$
$$\frac{d^2U}{dy^2} \approx \frac{U_{i,j+\Delta y} + U_{i,j-\Delta y} - 2U_{i,j}}{\Delta y^2}$$

3.2 Application Across a 2D Mesh

For the derivation of the FDM solution, a uniform 2D mesh with increments of 0.25 will be utilized. The points are oriented such that the point in the top-left is point 0,0 and the furthest point on the bottom right is 4,4.

Applying the heat conduction equations produces:

$$K^2 * \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = 0$$

$$K^2 * \frac{U_{i+\Delta x,j} + U_{i-\Delta \mathrm{x,j}} - 2U_{i,j}}{\Delta x^2} + \frac{U_{i,j+\Delta y} + U_{i,\mathrm{j}-\Delta \mathrm{y}} - 2U_{i,j}}{\Delta y^2} = 0$$

Since $\Delta x = \Delta y$:

$$-2(K^2+1) * U_{i,j} + K^2\left(U_{i+\Delta x,j} + U_{i-\Delta x,j}\right) + U_{i,j+\Delta y} + U_{i,j-\Delta y} = 0$$

$$U_{\Delta x \Delta y} = \frac{100}{2(K^2+1)}$$

Applying the system of equations to $\Delta x = \Delta y = 0.25$ produces the following set of equations:

$$-2(K^2+1) * T_{1,1} + K^2\left(T_{2,1} + T_{0,1}\right) + \sin(\pi * 0.25) + T_{1,2} = 0$$

$$-2(K^2+1) * T_{2,1} + K^2\left(T_{3,1} + T_{1,1}\right) + \sin(\pi * 0.25) + T_{2,2} = 0$$

$$-2(K^2+1) * T_{3,1} + K^2\left(T_{3,1} + T_{2,1}\right) + \sin(\pi * 0.25) + T_{2,2} = 0$$

Using this pattern, a global matrix can be constructed that can be solved for the temperature at each node:

$$\begin{bmatrix} -2(K^2+1) & K^2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ K^2 & -2(K^2+1) & K^2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & K^2 & -2(K^2+1) & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -2(K^2+1) & K^2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & K^2 & -2(K^2+1) & K^2 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & K^2 & -2(K^2+1) & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2(K^2+1) & K^2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & K^2 & -2(K^2+1) & K^2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & K^2 & -2(K^2+1) \end{bmatrix} \begin{bmatrix} T_{1,1} \\ T_{1,2} \\ T_{1,3} \\ T_{2,1} \\ T_{2,2} \\ T_{2,3} \\ T_{3,1} \\ T_{3,2} \\ T_{3,3} \end{bmatrix} = \begin{bmatrix} -70.71 \\ -100 \\ -70.71 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This matrix pattern can be used to generate the general case to be used in code.

4 Heat Flux Through the Top Boundary

4.1 Analytical Solution

The analytical solution to the total heat flux through the top boundary can be calculated using 1/3 Simpson Integration:

$$\dot{q} = \int_0^x -k * thickness * \frac{dT}{dy} * dx$$

Here,

$$\frac{dT}{dy} = \frac{d}{dy}\left(\frac{100\sinh(K\pi y)}{\sinh(K\pi)} * \sin(\pi x)\right)$$

Integrating yields:

$$\dot{q} = -200 * \frac{K}{\tanh{(K * \pi)}}$$

4.2 Numerical Computation

For the numerical computation of the heat flux through the upper boundary, the same equation used for the analytical solution can be used. However, $\frac{dT}{dy}$ must be derived with the following expression:

$$\frac{dT}{dy} \approx \frac{T_{i,jmax-2} - 4 * T_{i,jmax-1} + 3 * T_{i,jmax}}{2 * \Delta x}$$

The heat flux is then derived by applying Simpson integration with this equation.

5 Performance Analysis

5.1 Error

The errors of the numerical solutions are calculated using the equations below:

$$e_h = u(x) - u_h(x)$$

5.2 Percent Error

The percent error of an estimated quantity $Q_{Estimated}$ (calculated using FDM) against its exact values Q_{Exact} is calculated using the equation below:

$$\%Error = \left| \frac{Q_{Exact} - Q_{Estimated}}{Q_{Exact}} \right| \times 100\%$$

5.3 Extrapolation and Convergence

Richardson's Extrapolation was used to extrapolate an approximate of the exact value from a series of approximated values. In general, error is modeled as:

$$Q_{ex} - Q_h = Ch^{\beta}$$

Where Q is the quantity of interest, Q_h is the approximate value at some mesh size h, C is some constant, and β is the convergence rate. In general, it is rare for the exact value to be known, and it is often difficult or impossible to obtain analytical solutions. In this case it is possible to use Richardson's Extrapolation to obtain reasonably accurate approximate value of the exact solution. If we write this equation at another mesh size, say h/2, the two can be divided and the unknown β can be found.

$$Q_{ex} - Q_h = C(h)^\beta$$

$$Q_{ex} - Q_{\frac{h}{2}} = C\left(\frac{h}{2}\right)^\beta$$

$$\frac{Q_{ex} - Q_h}{Q_{ex} - Q_{\frac{h}{2}}} = \frac{C(h)^\beta}{C(\frac{h}{2})^\beta}$$

$$log\left(\frac{Q_{ex} - Q_h}{Q_{ex} - Q_{\frac{h}{2}}}\right) = log\left(\frac{C(h)^\beta}{C(\frac{h}{2})^\beta}\right)$$

$$log(Q_{ex} - Q_h) - log(Q_{ex} - Q_{\frac{h}{2}}) = \beta log(h) - \beta log\left(\frac{h}{2}\right)$$

$$\boxed{\beta = \frac{log(Q_{ex} - Q_h) - log(Q_{ex} - Q_{\frac{h}{2}})}{log(h) - log(\frac{h}{2})}}$$

Again, Richardson's Extrapolation will be used to derive an expression for an extrapolated value. Here we will have to utilize three mesh sizes rather than the previous two.

$$Q_{ex} - Q_h = C(h)^\beta \approx 2^\beta$$

$$Q_{ex} - Q_{\frac{h}{2}} = C\left(\frac{h}{2}\right)^\beta \approx 2^\beta$$

$$Q_{ex} - Q_{\frac{h}{4}} = C\left(\frac{h}{4}\right)^\beta \approx 2^\beta$$

$$\frac{Q_{ex} - Q_h}{Q_{ex} - Q_{\frac{h}{2}}} \approx 2^\beta \approx \frac{Q_{ex} - Q_{\frac{h}{2}}}{Q_{ex} - Q_{\frac{h}{4}}}$$

$$\frac{Q_{extr} - Q_h}{Q_{extr} - Q_{\frac{h}{2}}} = 2^\beta = \frac{Q_{extr} - Q_{\frac{h}{2}}}{Q_{extr} - Q_{\frac{h}{4}}}$$

$$(Q_{extr} - Q_h)(Q_{extr} - Q_{\frac{h}{4}}) = (Q_{extr} - Q_{\frac{h}{2}})^2$$

$$\boxed{Q_{extr} = \frac{Q_{\frac{h}{2}}^2 - Q_h * Q_{\frac{h}{4}}}{2Q_{\frac{h}{2}} - Q_h - Q_{\frac{h}{4}}}}$$

From here Q_{extra} can be substituted to solve for β, which yields:

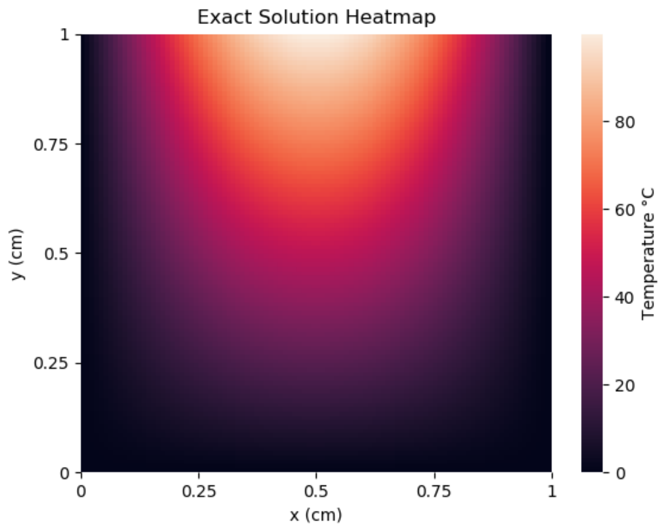
$$\boxed{\frac{log\left(\frac{Q_{extr} - Q_h}{Q_{extr} - Q_{\frac{h}{2}}}\right)}{log(2)} = \beta = \frac{log\left(\frac{Q_{extr} - Q_{\frac{h}{2}}}{Q_{extr} - Q_{\frac{h}{4}}}\right)}{log(2)}}$$

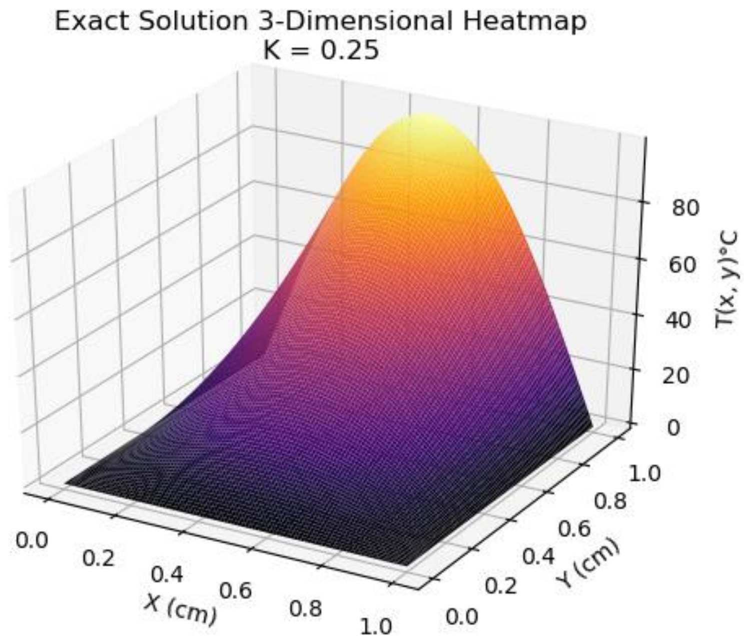
6 Results

6.1 Temperature Analytical Results

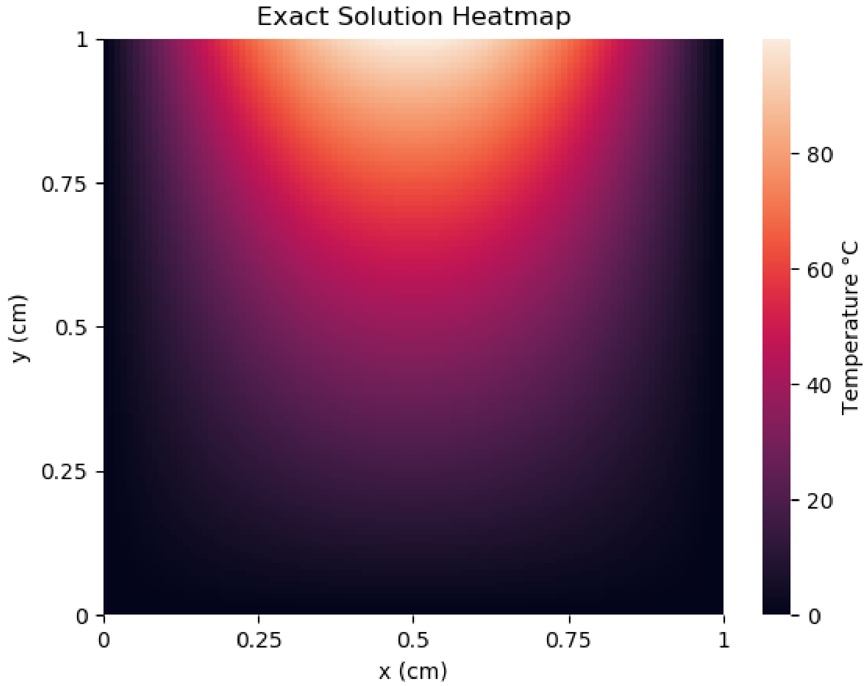
The following 2-dimensional and 3-dimensional heat maps show the temperature distribution of the bar under various thermal conductivity. It can be observed that as the value of K increases, the temperature distribution across the beam decreases as the material prevents more heat transfer:

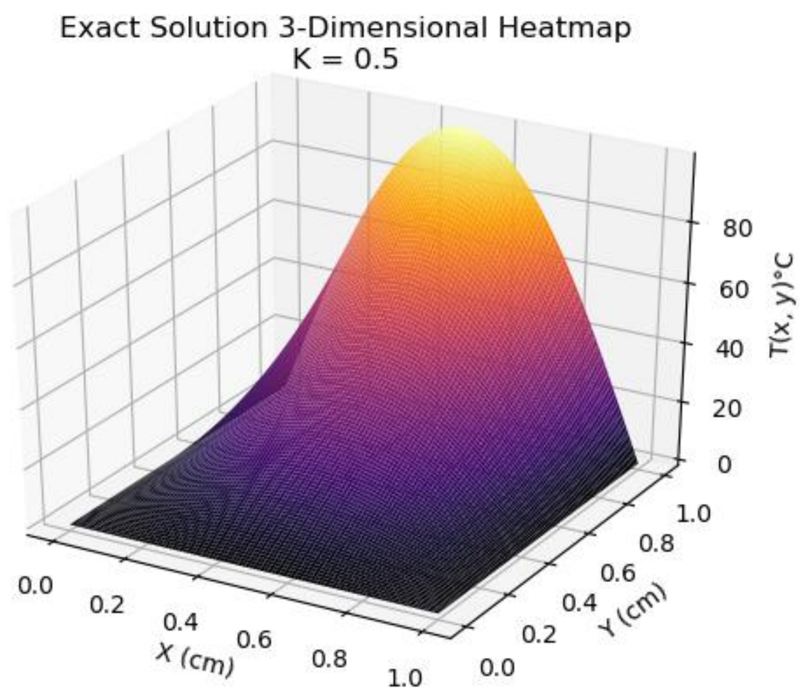
K = 0.25



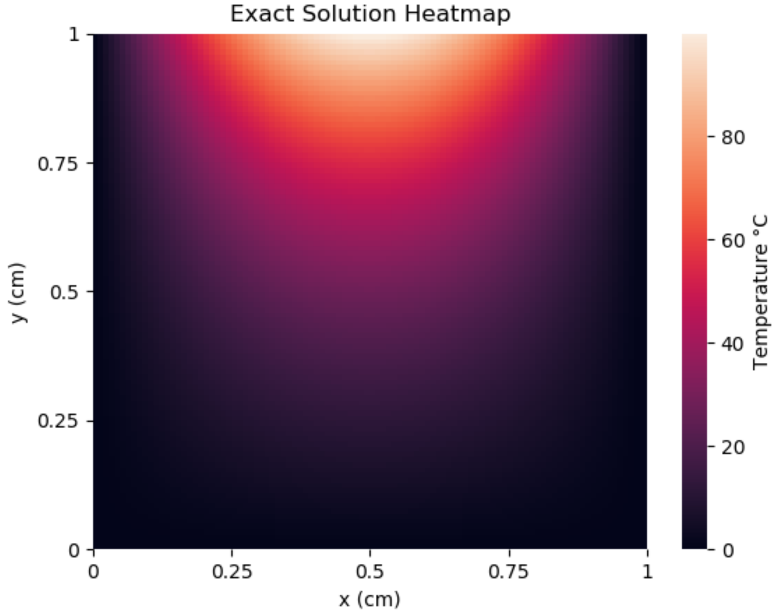


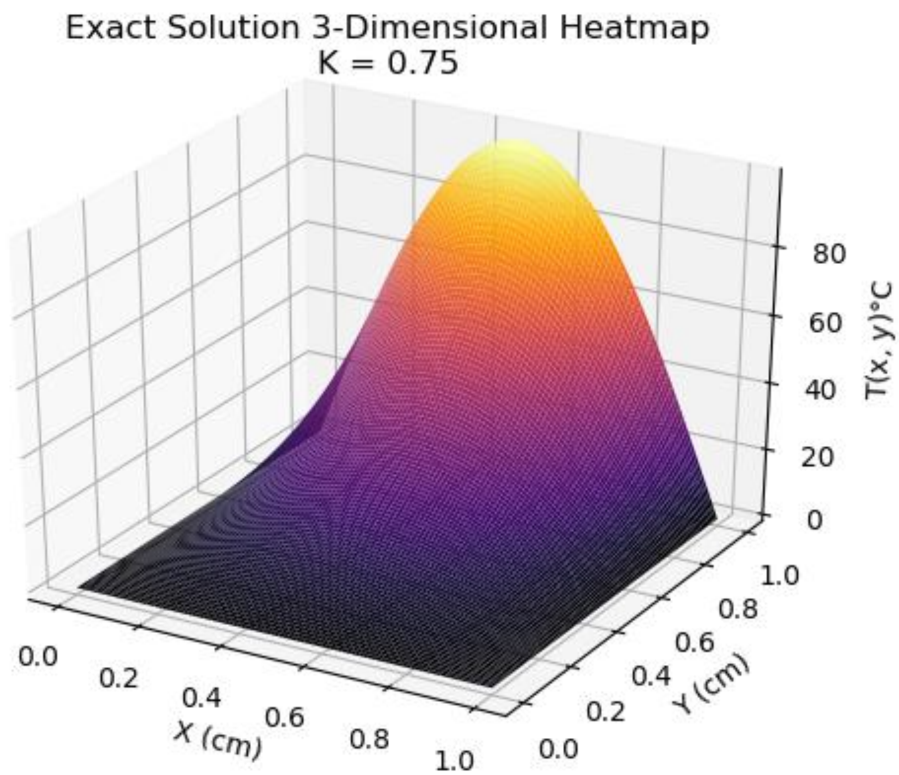
K =0.5


Exact Solution Heatmap
1
0.75
0.5
0.25
0
y (cm)
0
0.25
0.5
0.75
1
x (cm)
80
60
40
20
0
Temperature °C

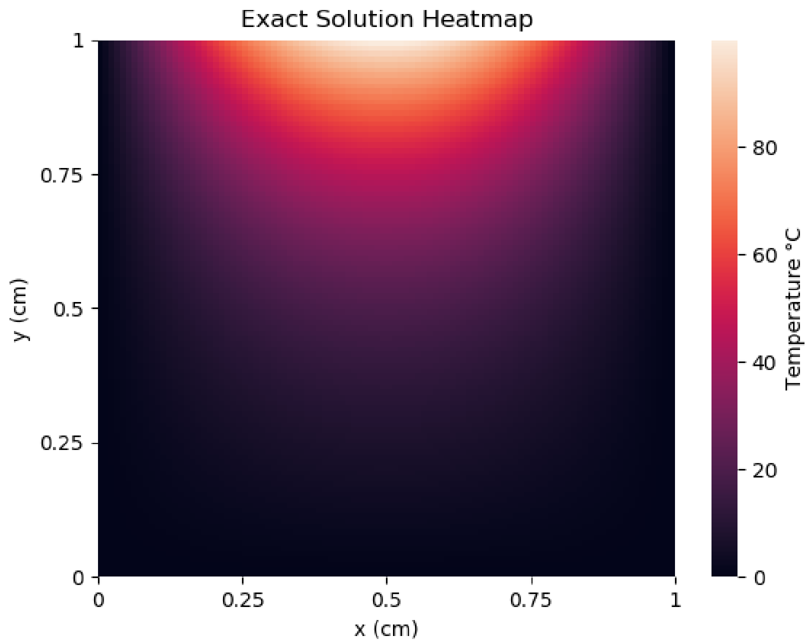

Exact Solution 3-Dimensional Heatmap
K = 0.5
80
60
40
20
0
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
X (cm)
0.0
0.2
0.4
0.6
0.8
1.0
Y (cm)

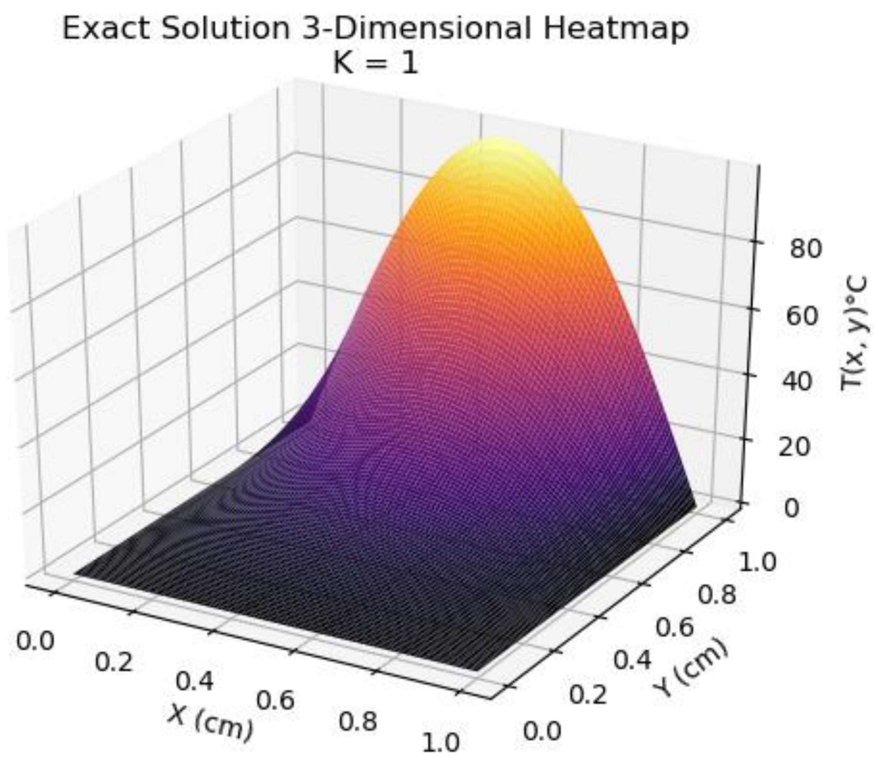
K = 0.75


Exact Solution Heatmap
y (cm)
x (cm)
0
0.25
0.5
0.75
1
Temperature °C
0
20
40
60
80

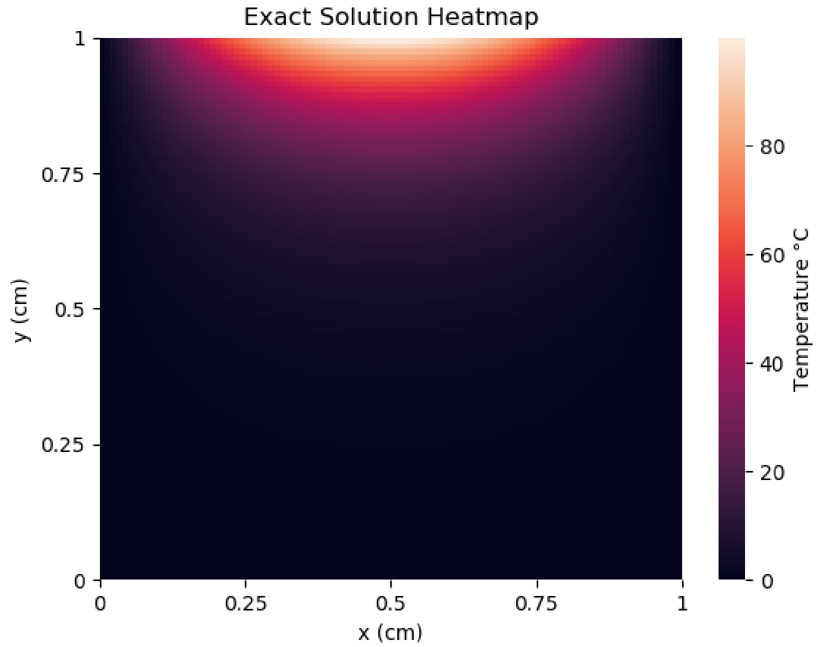

Exact Solution 3-Dimensional Heatmap
K = 0.75
T(x, y)°C
0
20
40
60
80
X (cm)
0.0
0.2
0.4
0.6
0.8
1.0
Y (cm)
0.0
0.2
0.4
0.6
0.8
1.0

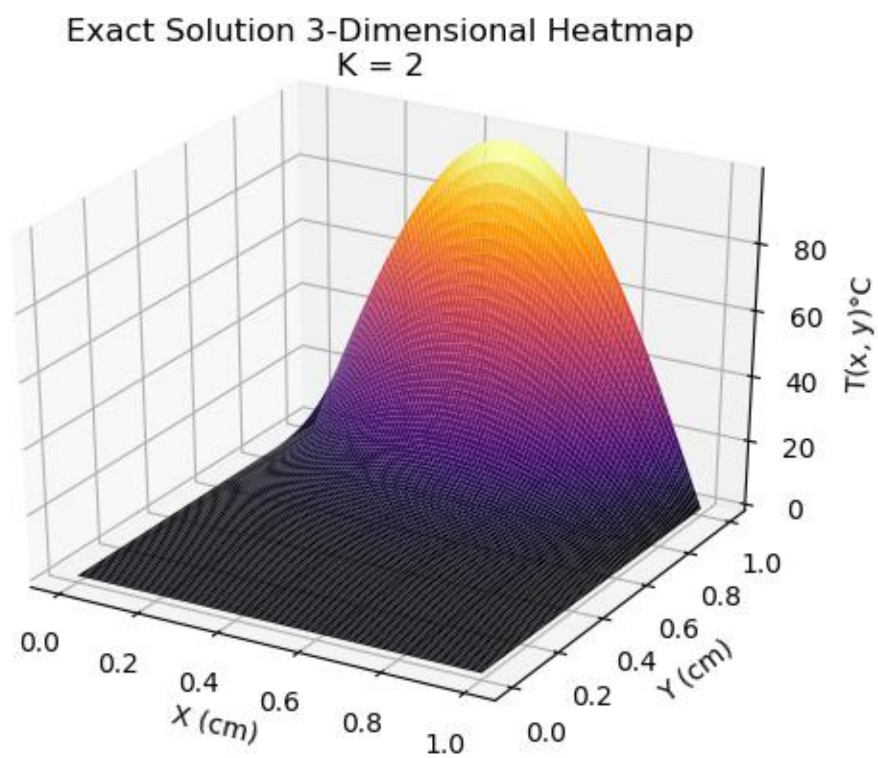
K = 1


Exact Solution Heatmap
1
0.75
0.5
0.25
0
y (cm)
0
0.25
0.5
0.75
1
x (cm)
80
60
40
20
0
Temperature °C

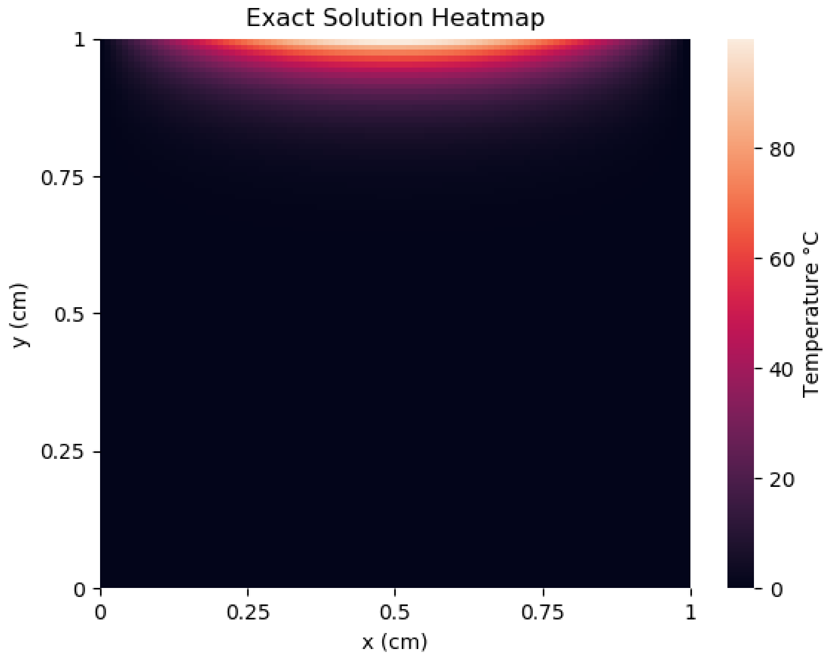

Exact Solution 3-Dimensional Heatmap
K = 1
80
60
40
20
0
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
X (cm)
0.0
0.2
0.4
0.6
0.8
1.0
Y (cm)

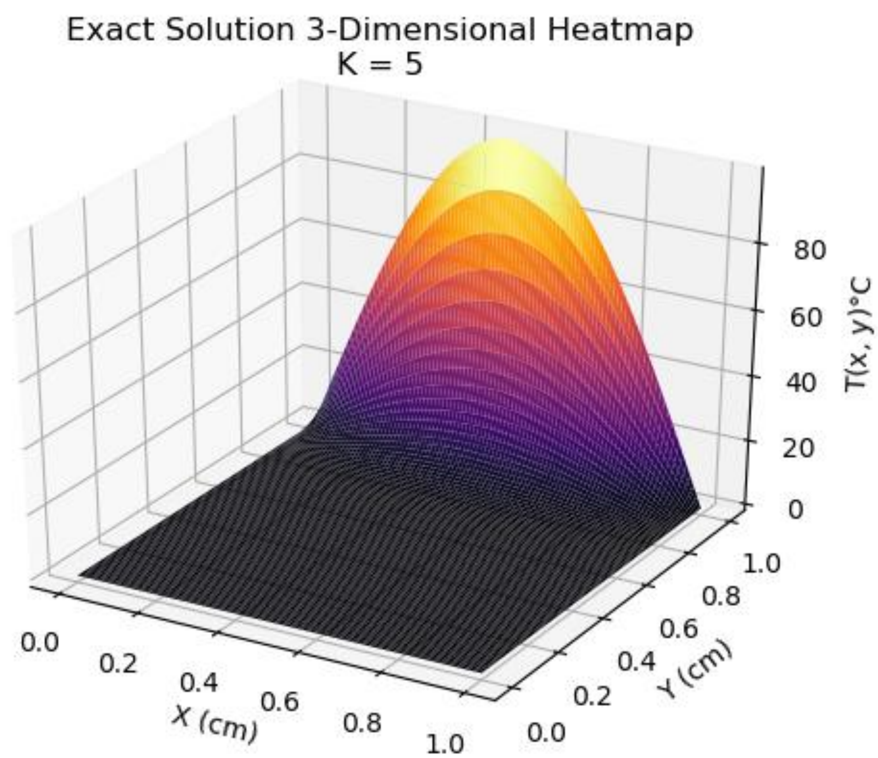
K = 2


Exact Solution Heatmap
1
0.75
0.5
0.25
0
y (cm)
0
0.25
0.5
0.75
1
x (cm)
80
60
40
20
0
Temperature °C

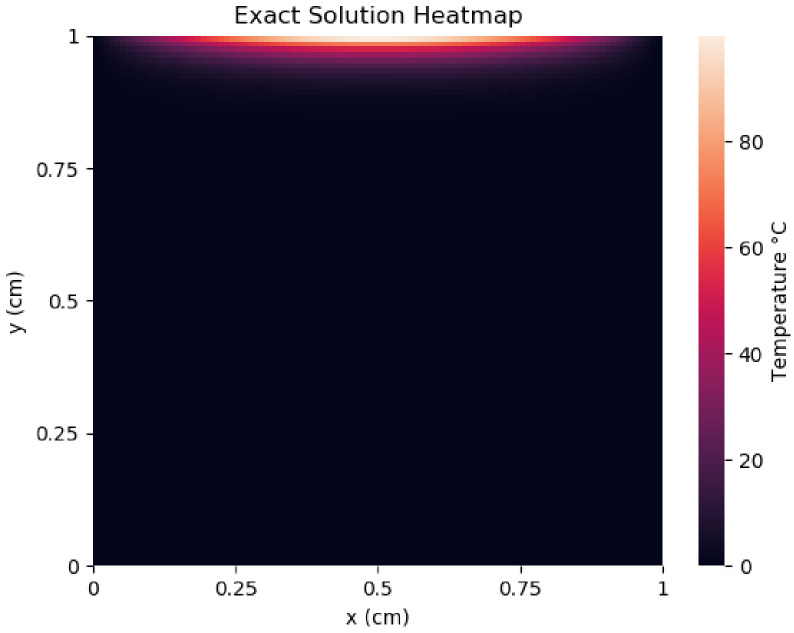

Exact Solution 3-Dimensional Heatmap
K = 2
80
60
40
20
0
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
X (cm)
0.0
0.2
0.4
0.6
0.8
1.0
Y (cm)

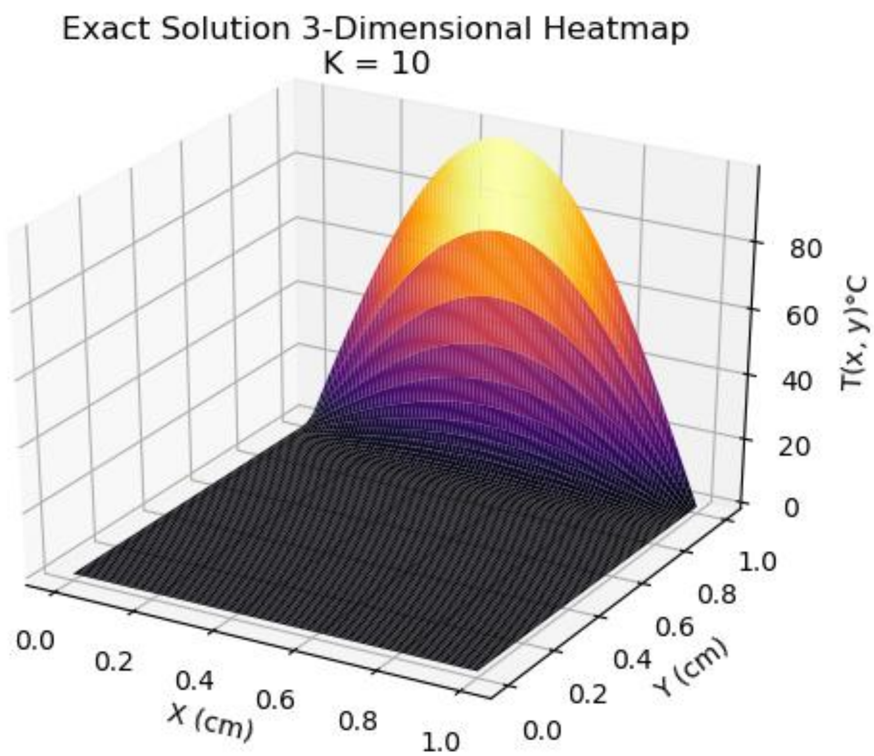
K = 5


Exact Solution Heatmap
y (cm)
x (cm)
Temperature °C
1
0.75
0.5
0.25
0
0
0.25
0.5
0.75
1
80
60
40
20
0


Exact Solution 3-Dimensional Heatmap
K = 5
X (cm)
Y (cm)
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
80
60
40
20
0

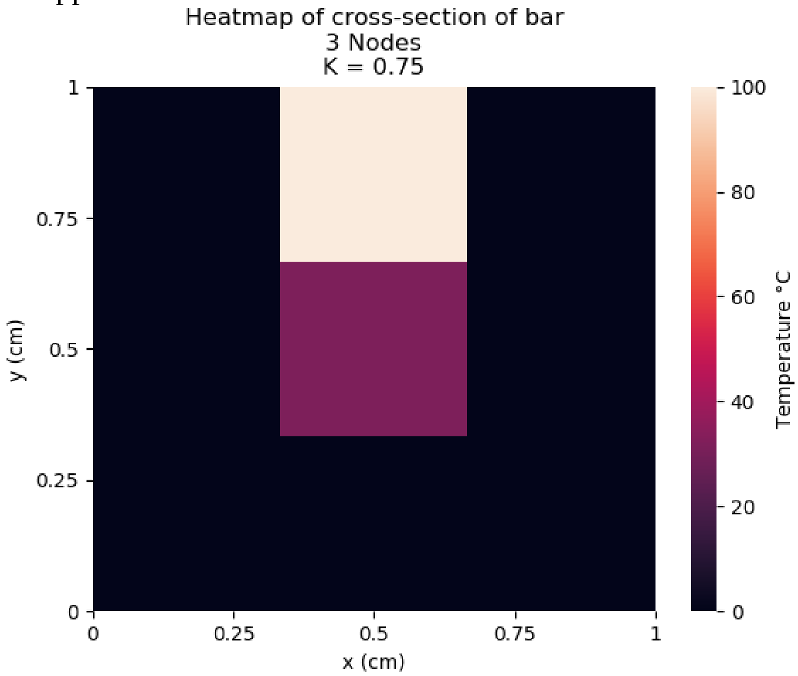
K = 10

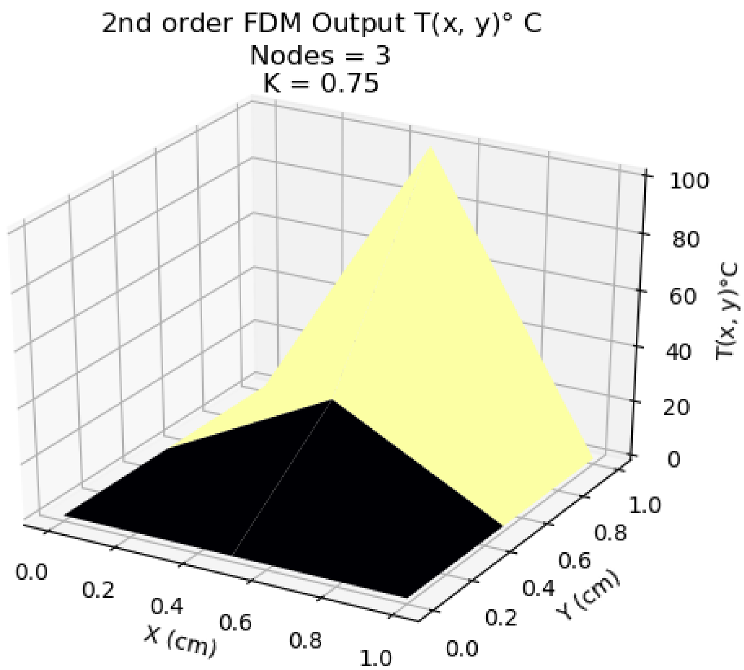

Exact Solution Heatmap
1
0.75
0.5
0.25
0
y (cm)
0
0.25
0.5
0.75
1
x (cm)
80
60
40
20
0
Temperature °C

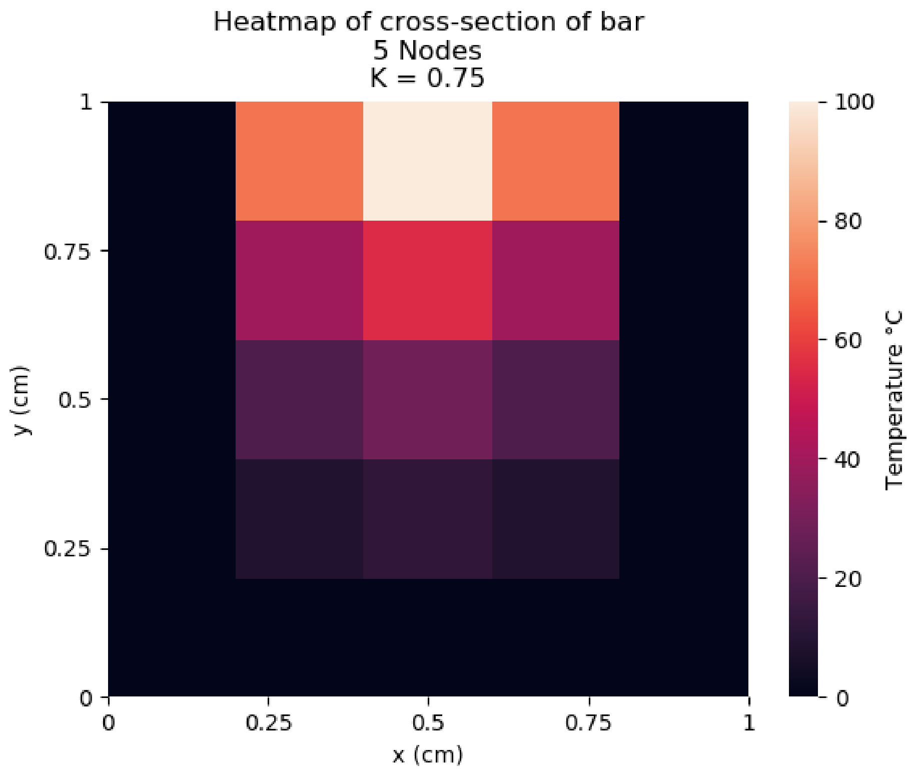

Exact Solution 3-Dimensional Heatmap
K = 10
80
60
40
20
0
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
X (cm)
0.0
0.2
0.4
0.6
0.8
1.0
Y (cm)

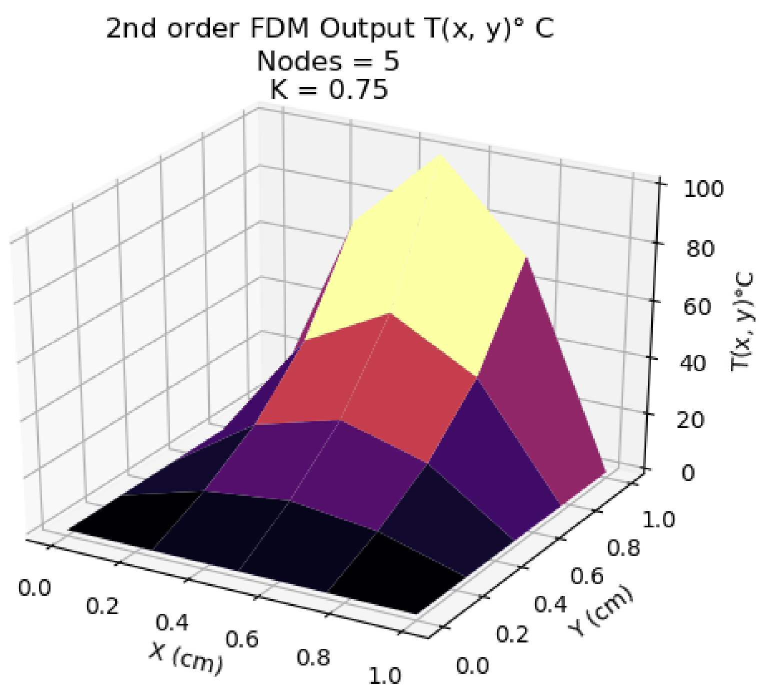
6.2 Temperature FDM Results

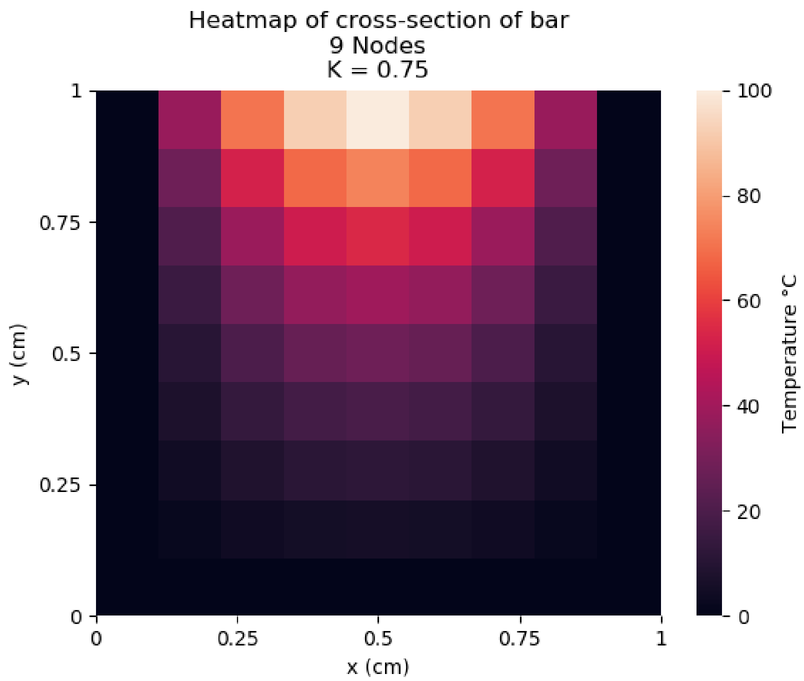
The following section outlines the results of the 2^{nd} Order FDM solution of the infinite bar at various mesh sizes and a thermal conductivity value of K = 0.75. Additional charts and graphs performed at different thermal conductivity values are generated by the code referenced in Appendix A: Code.

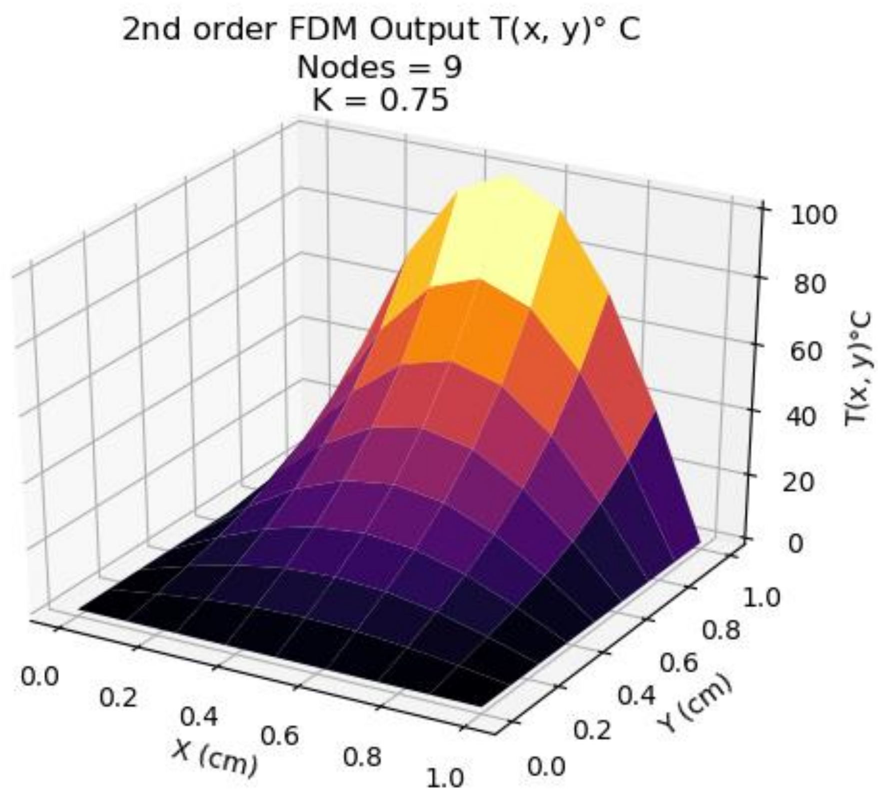


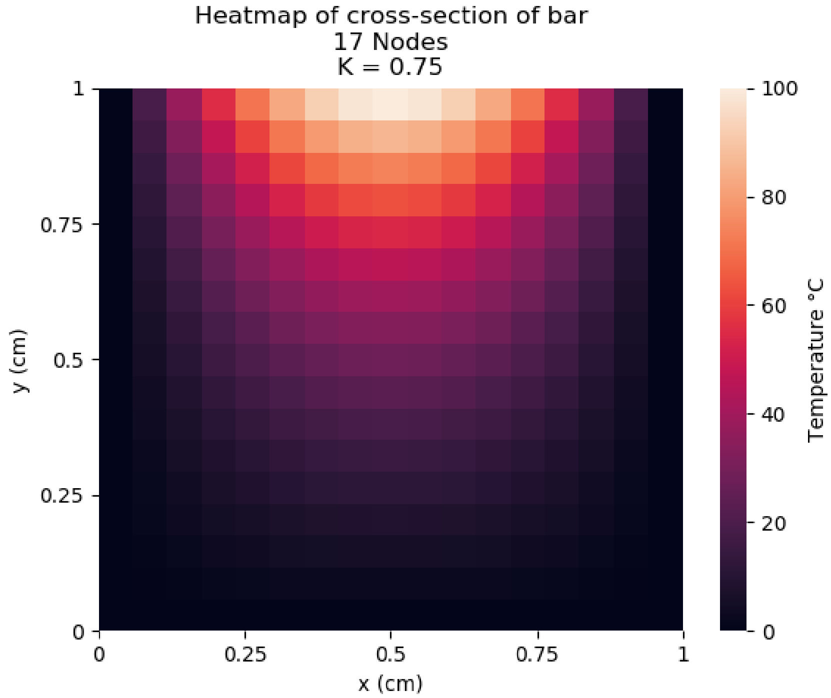


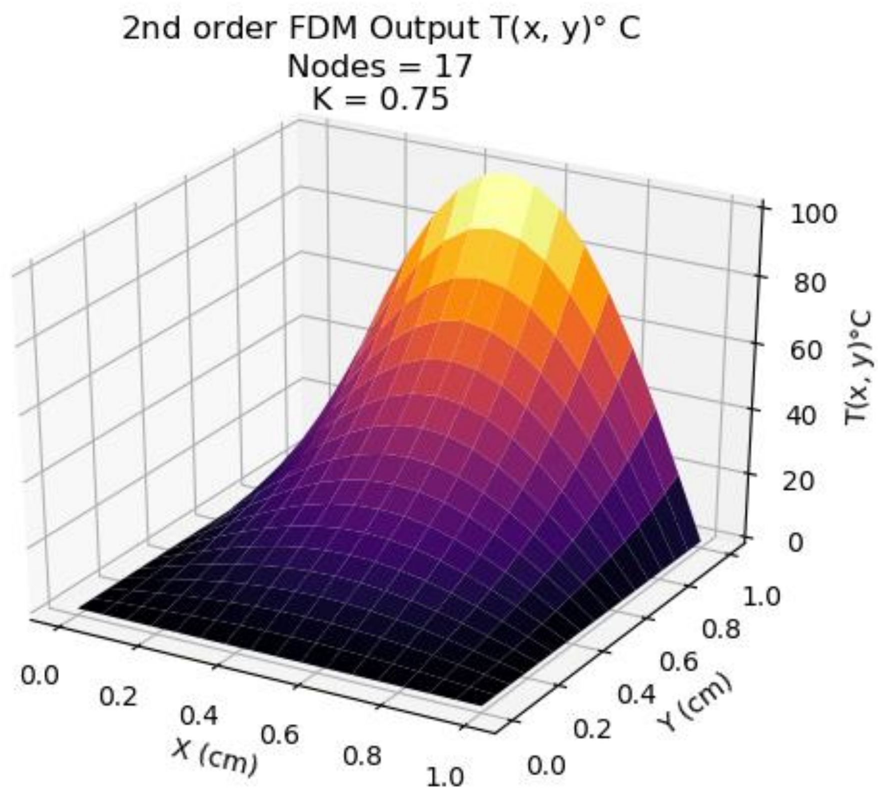

Heatmap of cross-section of bar
5 Nodes
K = 0.75
y (cm)
x (cm)
Temperature °C
0
0.25
0.5
0.75
1
20
40
60
80
100

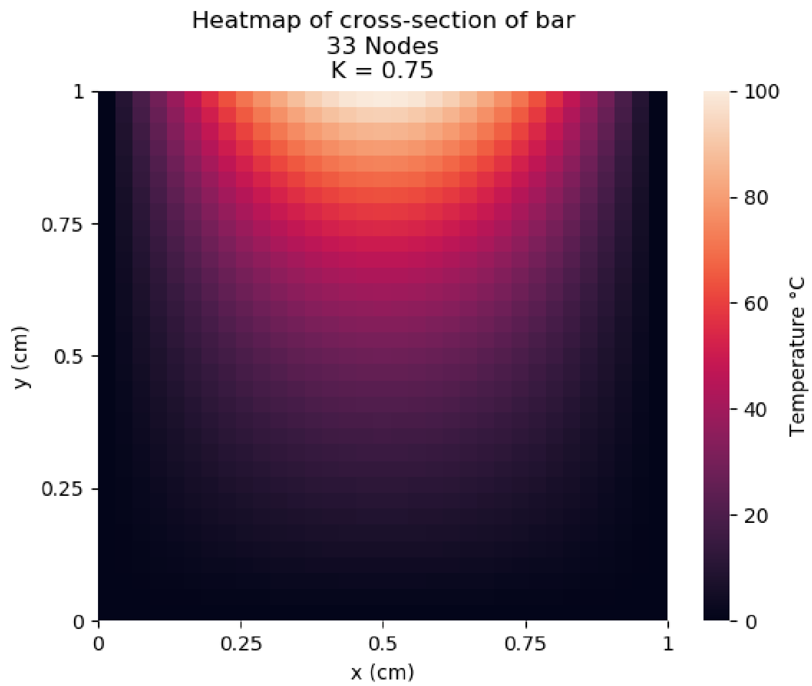

2nd order FDM Output T(x, y)° C
Nodes = 5
K = 0.75
X (cm)
Y (cm)
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

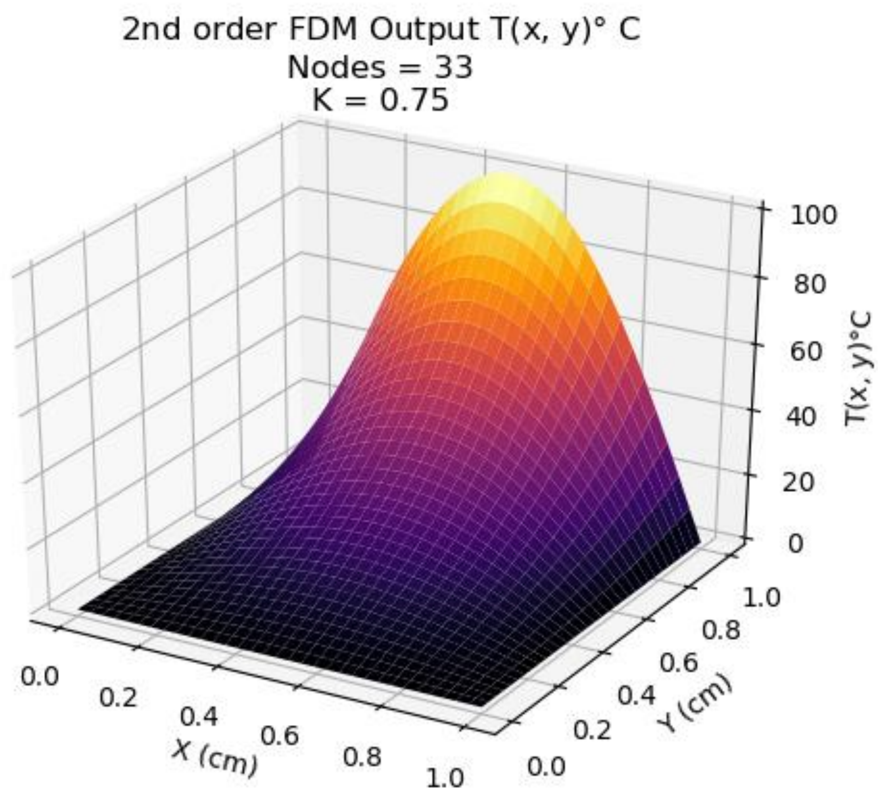

Heatmap of cross-section of bar
9 Nodes
K = 0.75
y (cm)
x (cm)
0
0.25
0.5
0.75
1
Temperature °C
0
20
40
60
80
100

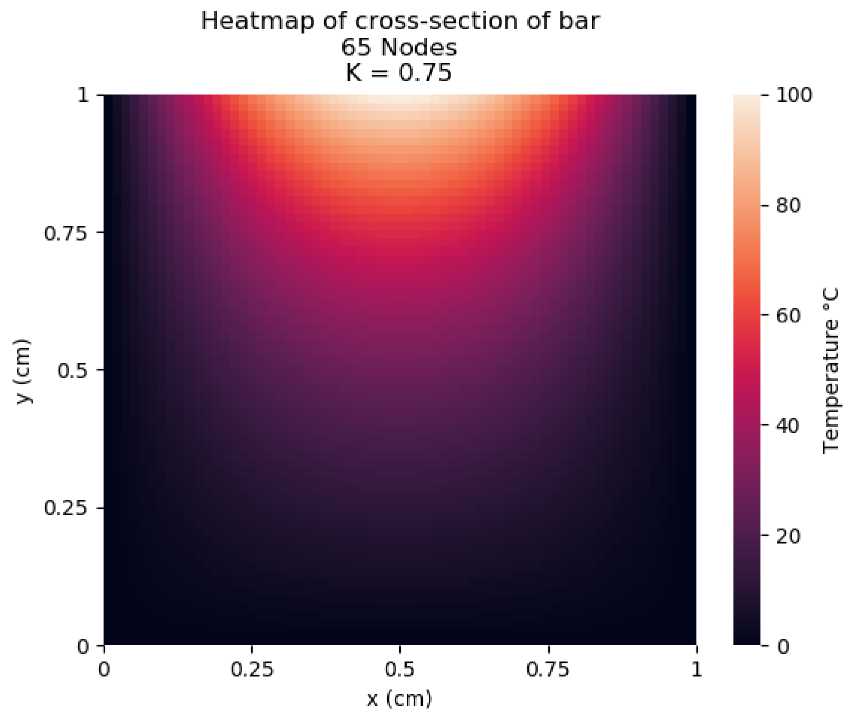

2nd order FDM Output T(x, y)° C
Nodes = 9
K = 0.75
X (cm)
Y (cm)
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

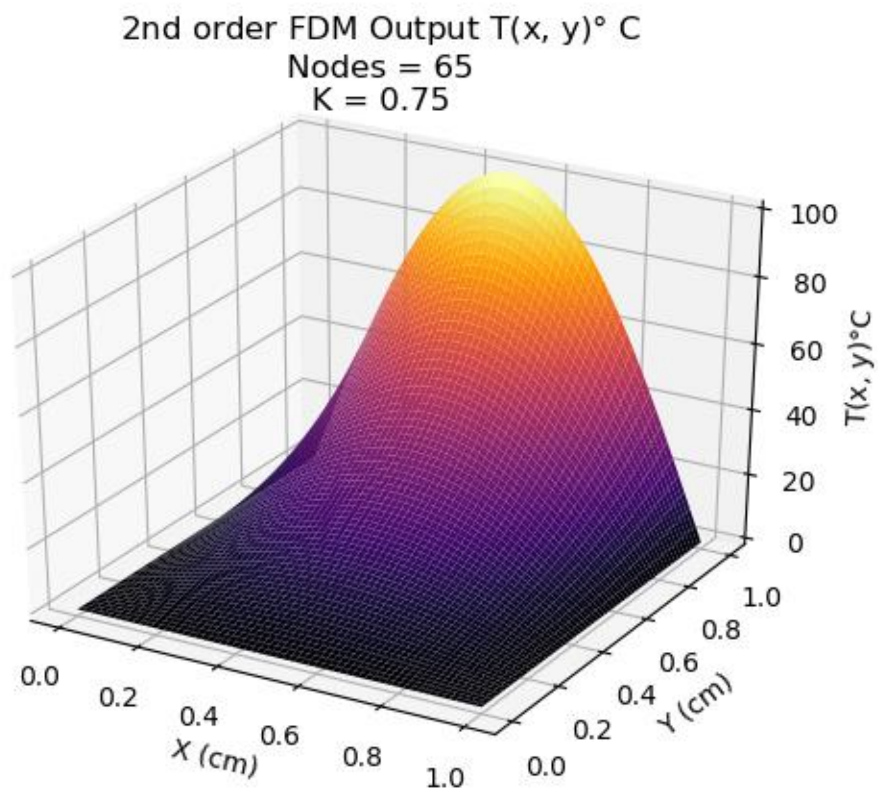

Heatmap of cross-section of bar
17 Nodes
K = 0.75
y (cm)
x (cm)
Temperature °C
0
0.25
0.5
0.75
1
20
40
60
80
100


2nd order FDM Output T(x, y)° C
Nodes = 17
K = 0.75
X (cm)
Y (cm)
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


Heatmap of cross-section of bar
33 Nodes
K = 0.75
y (cm)
x (cm)
Temperature °C
0
0.25
0.5
0.75
1
100
80
60
40
20


2nd order FDM Output T(x, y)° C
Nodes = 33
K = 0.75
X (cm)
Y (cm)
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
100
80
60
40
20
0


Heatmap of cross-section of bar
65 Nodes
K = 0.75
y (cm)
x (cm)
Temperature °C
0
0.25
0.5
0.75
1
0
20
40
60
80
100


2nd order FDM Output T(x, y)° C
Nodes = 65
K = 0.75
X (cm)
Y (cm)
T(x, y)°C
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

6.3 Heat Flux FDM Results

The following section presents the results of the heat flux through the upper boundary of the bar at various mesh sizes and thermal conductivity values. In addition, the convergence of these solutions is provided.

Convergence of Heat Flux (K = 0.25):

Num. Elements	dx	Exact Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-76.2434	-74.5098	2.27381	n/a
4	0.25	-76.2434	-74.922	1.73313	0.39172944097167783
8	0.125	-76.2434	-75.859	0.504218	1.781259498021684
16	0.0625	-76.2434	-76.1432	0.131412	1.9399456525572434
32	0.03125	-76.2434	-76.218	0.0333101	1.98007139942149
64	0.015625	-76.2434	-76.237	0.00837129	1.992437330814078

Convergence of Heat Flux (K = 0.5):

Num. Elements	dx	Exact Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-109.033	-93.3333	14.3991	n/a
4	0.25	-109.033	-102.761	5.75292	1.3236135797928712
8	0.125	-109.033	-107.211	1.67154	1.783112953893871
16	0.0625	-109.033	-108.549	0.444083	1.9122814752475343
32	0.03125	-109.033	-108.909	0.114042	1.9612630475079167
64	0.015625	-109.033	-109.002	0.0288698	1.9819344537559005

Convergence of Heat Flux (K = 0.75):

Num. Elements	dx	Exact Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-152.719	-114.667	24.9168	n/a
4	0.25	-152.719	-137.009	10.2869	1.276310140243971
8	0.125	-152.719	-147.869	3.17583	1.6956012524728339
16	0.0625	-152.719	-151.386	0.873308	1.862572167244878
32	0.03125	-152.719	-152.371	0.228281	1.9356766230712397
64	0.015625	-152.719	-152.63	0.0583083	1.9690388825278446

Convergence of Heat Flux (K = 1):

Num. Elements	dx	Exact Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-200.748	-133.333	33.5819	n/a
4	0.25	-200.748	-170.541	15.0476	1.1581500585742317
8	0.125	-200.748	-190.786	4.96237	1.600430811455807
16	0.0625	-200.748	-197.91	1.41408	1.8111642612147045
32	0.03125	-200.748	-199.993	0.376345	1.9097356905830358
64	0.015625	-200.748	-200.554	0.0969932	1.956102251124672

Convergence of Heat Flux (K = 2):

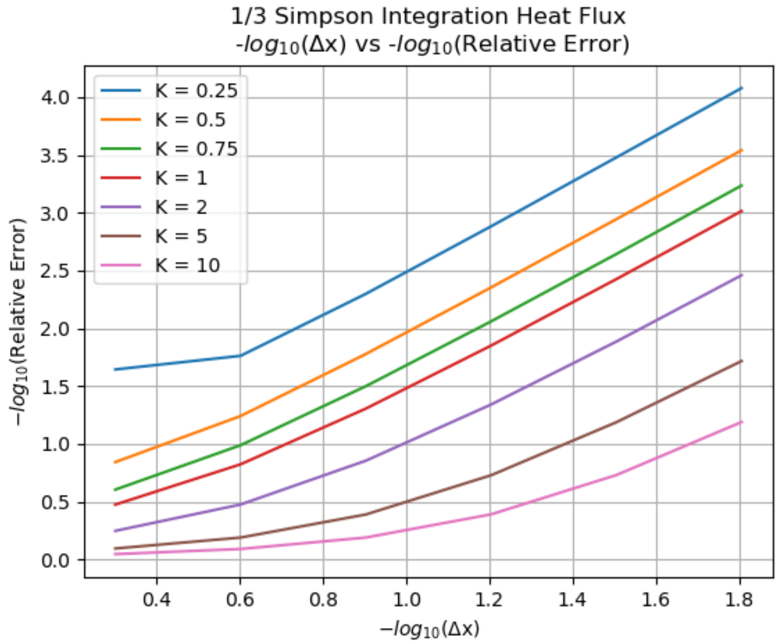
Num. Elements	dx	Exact Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-400.003	-173.333	56.667	n/a
4	0.25	-400.003	-265.909	33.5232	0.75734769137366
8	0.125	-400.003	-343.985	14.0043	1.2592882588058265
16	0.0625	-400.003	-381.704	4.57473	1.61411253483227
32	0.03125	-400.003	-394.781	1.30555	1.8090320490738938
64	0.015625	-400.003	-398.609	0.348365	1.905984022430321

Convergence of Heat Flux (K = 5):

Num. Elements	dx	Exact Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-1000	-194.872	80.5128	n/a
4	0.25	-1000	-352.527	64.7473	0.31439911677174426
8	0.125	-1000	-591.087	40.8913	0.6630243806977204
16	0.0625	-1000	-812.802	18.7198	1.1272280368091916
32	0.03125	-1000	-934.863	6.51369	1.5230201904232348
64	0.015625	-1000	-980.757	1.92425	1.7591784712191902

Convergence of Heat Flux (K = 10):

Num. Elements	dx	Exact Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-2000	-198.68	90.066	n/a
4	0.25	-2000	-374.449	81.2776	0.1481256540869807
8	0.125	-2000	-705.565	64.7218	0.3286059404502064
16	0.0625	-2000	-1184.24	40.7878	0.6661120414729824
32	0.03125	-2000	-1626.94	18.6528	1.1287498517795977
64	0.015625	-2000	-1870.22	6.48882	1.523361198648693



6.4 Heat Flux Richardson Extrapolation

The exact solution is not always available in modern problems. As such, the Richardson Extrapolation was also performed to be compared against the approximate solution. The following section presents this data at various values of thermal conductivity.

Convergence of Extrapolated Heat Flux (K = 0.25):

Num. Elements	dx	Extrapolated Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-74.186	-74.5098	0.436534	1.18454
4	0.25	-76.267	-74.922	1.76352	1.72089
8	0.125	-76.2447	-75.859	0.505934	1.92606
16	0.0625	-76.2435	-76.1432	0.131542	1.9759

Convergence of Extrapolated Heat Flux (K = 0.5):

Num. Elements	dx	Extrapolated Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-111.189	-93.3333	16.0591	1.08301
4	0.25	-109.125	-102.761	5.83188	1.73338
8	0.125	-109.041	-107.211	1.67876	1.89496
16	0.0625	-109.034	-108.549	0.444835	1.95419

Convergence of Extrapolated Heat Flux (K = 0.75):

Num. Elements	dx	Extrapolated Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-158.14	-114.667	27.4906	1.04078
4	0.25	-153.07	-137.009	10.4921	1.62685
8	0.125	-152.754	-147.869	3.19784	1.83579
16	0.0625	-152.723	-151.386	0.875793	1.92405

Convergence of Extrapolated Heat Flux (K = 1):

Num. Elements	dx	Extrapolated Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-214.953	-133.333	37.9709	0.877954
4	0.25	-201.776	-170.541	15.4803	1.50705
8	0.125	-200.854	-190.786	5.01234	1.77368
16	0.0625	-200.76	-197.91	1.4199	1.89328

Convergence of Extrapolated Heat Flux (K = 2):

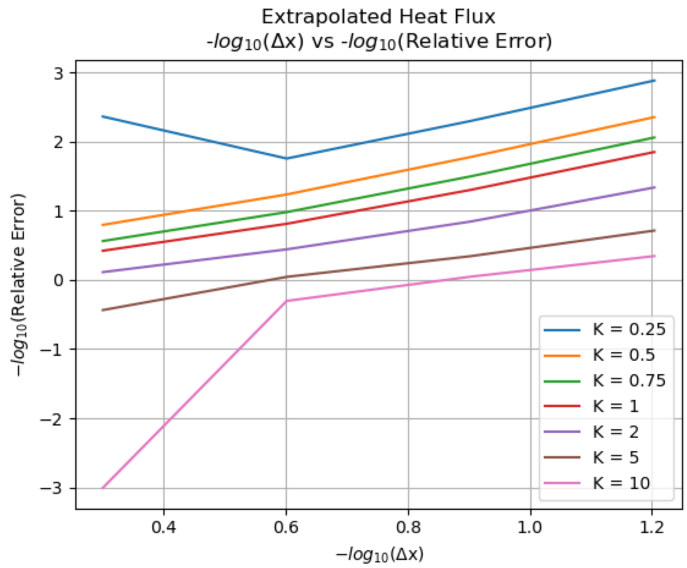
Num. Elements	dx	Extrapolated Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	-764.403	-173.333	77.3243	0.245752
4	0.25	-416.956	-265.909	36.2261	1.0496
8	0.125	-401.72	-343.985	14.372	1.52826
16	0.0625	-400.194	-381.704	4.62043	1.77206

Convergence of Extrapolated Heat Flux (K = 5):

Num. Elements	dx	Extrapolated Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	112.348	-194.872	273.454	0.597573
4	0.25	-3731.14	-352.527	90.5518	0.105642
8	0.125	-1084.37	-591.087	45.4903	0.8611
16	0.0625	-1008.41	-812.802	19.3978	1.41122

Convergence of Extrapolated Heat Flux (K = 10):

Num. Elements	dx	Extrapolated Heat Flux	Approx. Heat Loss	Percent Error	Beta
2	0.5	0.196153	-198.68	101388	0.913655
4	0.25	368.54	-374.449	201.603	0.531722
8	0.125	-7074.43	-705.565	90.0265	0.112723
16	0.0625	-2167	-1184.24	45.3511	0.863724



6.5 Convergence of FDM Solution

The following section presents the convergence of 2nd Order FDM solution by comparing the temperature of a point within the middle of the bar derived from the FDM solution with the analytical solution.

Convergence of 2nd Order FDM Temperature Solution (K = 0.25):

Num. Elements	dx	Exact Midpoint Temp	Approx. Midpoint Temp	Percent Error	Beta
2	0.5	46.3778	47.0588	1.46845	n/a
4	0.25	46.3778	46.5596	0.391918	1.9056726734563847
8	0.125	46.3778	46.424	0.0996266	1.975947176276953
16	0.0625	46.3778	46.3894	0.0250112	1.9939545513403978
32	0.03125	46.3778	46.3807	0.00625937	1.9984865746513485
64	0.015625	46.3778	46.3785	0.00156525	1.9996215153853052

Convergence of 2nd Order FDM Temperature Solution (K = 0.5):

Num. Elements	dx	Exact Midpoint Temp	Approx. Midpoint Temp	Percent Error	Beta
2	0.5	37.747	40	5.96873	n/a
4	0.25	37.747	38.3548	1.61033	1.890064341048055
8	0.125	37.747	37.9021	0.410845	1.9706929406809564
16	0.0625	37.747	37.786	0.103244	1.9925398771150318
32	0.03125	37.747	37.7567	0.0258445	1.998126274717085
64	0.015625	37.747	37.7494	0.00646322	1.9995310196004619

Convergence of 2nd Order FDM Temperature Solution (K = 0.75):

Num. Elements	dx	Exact Midpoint Temp	Approx. Midpoint Temp	Percent Error	Beta
2	0.5	28.1211	32	13.7937	n/a
4	0.25	28.1211	29.1835	3.77806	1.8682863441294335
8	0.125	28.1211	28.3936	0.969265	1.9626833381613211
16	0.0625	28.1211	28.1897	0.243947	1.9903227976580664
32	0.03125	28.1211	28.1383	0.0610901	1.9975574278793857
64	0.015625	28.1211	28.1254	0.015279	1.999387876879413

Convergence of 2nd Order FDM Temperature Solution (K = 1):

Num. Elements	dx	Exact Midpoint Temp	Approx. Midpoint Temp	Percent Error	Beta
2	0.5	19.9268	25	25.4589	n/a
4	0.25	19.9268	21.3388	7.08589	1.8451504875388924
8	0.125	19.9268	20.2915	1.8301	1.9530246988447173
16	0.0625	19.9268	20.0188	0.461496	1.9875349860890625
32	0.03125	19.9268	19.9499	0.115627	1.9968339145515912
64	0.015625	19.9268	19.9326	0.0289228	1.9992052803820966

Convergence of 2nd Order FDM Temperature Solution (K = 2):

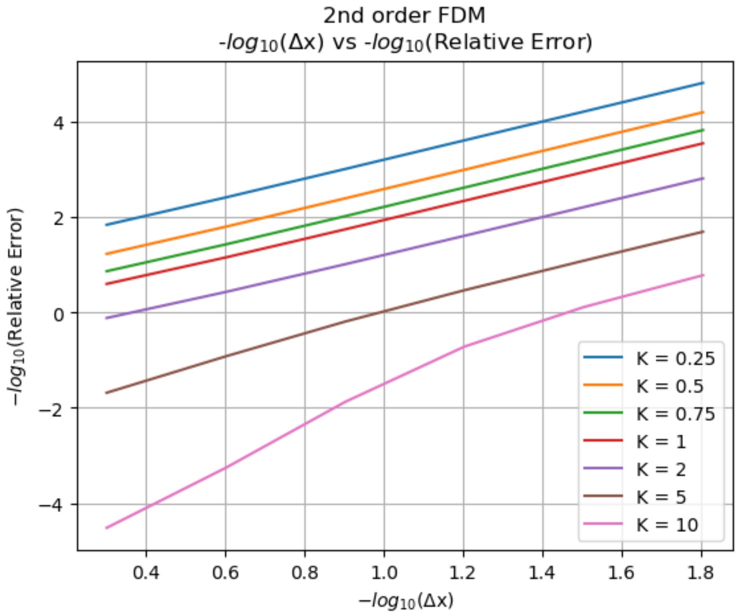
Num. Elements	dx	Exact Midpoint Temp	Approx. Midpoint Temp	Percent Error	Beta
2	0.5	4.31334	10	131.839	n/a
4	0.25	4.31334	5.93017	37.4846	1.814409081914657
8	0.125	4.31334	4.73837	9.85402	1.9275126560263582
16	0.0625	4.31334	4.4212	2.50066	1.9784039340079103
32	0.03125	4.31334	4.34041	0.627635	1.9943110148969143
64	0.015625	4.31334	4.32011	0.157066	1.9985580156492015

Convergence of 2nd Order FDM Temperature Solution (K = 5):

Num. Elements	dx	Exact Midpoint Temp	Approx. Midpoint Temp	Percent Error	Beta
2	0.5	0.0388203	1.92308	4853.79	n/a
4	0.25	0.0388203	0.363577	836.564	2.5365633634160085
8	0.125	0.0388203	0.0996494	156.694	2.4165267143029645
16	0.0625	0.0388203	0.0522322	34.5485	2.1812534660952503
32	0.03125	0.0388203	0.0420473	8.31268	2.0552396238446007
64	0.015625	0.0388203	0.0396189	2.05722	2.0146146625571846

Convergence of 2nd Order FDM Temperature Solution (K = 10):

Num. Elements	dx	Exact Midpoint Temp	Approx. Midpoint Temp	Percent Error	Beta
2	0.5	1.50702e-05	0.49505	3.28486e+06	n/a
4	0.25	1.50702e-05	0.0272645	180817	4.183230529085337
8	0.125	1.50702e-05	0.0011517	7542.26	4.583390150663314
16	0.0625	1.50702e-05	9.40808e-05	524.285	3.846573806219785
32	0.03125	1.50702e-05	2.68226e-05	77.9849	2.7490846449177506
64	0.015625	1.50702e-05	1.75987e-05	16.7786	2.216569741450256



7 Discussion

This report investigated the application of FDM to solve a 2-dimensional boundary condition problem. Furthermore, this report investigated the effect that varying the number of nodes and the value of the thermal conductivity had on the resulting temperature distribution and the convergence and accuracy of the results.

It was observed that the temperature distribution within the bar varied drastically as the value of K increased. This result is expected as highly values of K represent a greater ability of the material to resist a change in temperature. In addition, the convergence of the results consistently approximated to a value of 2. This result is also expected as the FDM solution was derived to a second order. One of the limiting factors experienced with the code was the number of nodes that could be used in calculations. For nodes used beyond 129, the performance of the code significantly decreased. Utilizing a greater number of nodes beyond this value quickly become impractical due to the computation time required.

8 References

The following reports and codes were used to assist in the creation of this report and the code utilized within.

- Antonio Diaz’s Code and Report
- Valentina Musu’s Report

9 Appendix B: Code

```
"""
AERO 430 Assignment 3
Andrew Hollister
UIN: 127008398
"""

import numpy as np
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
import tabulate as tbl


"""
Data Structure
"""


class DataStructure:
    def __init__(self):
        self.data = []

    def add_data(self, nodes, k, x_vals, y_vals, temp_mesh):
        self.data.append((nodes, k, x_vals, y_vals, temp_mesh))

    def return_data(self, nodes, k):
        x_vals = next(item[2] for item in self.data if item[0] == nodes and item[1] == k)
        y_vals = next(item[3] for item in self.data if item[0] == nodes and item[1] == k)
        temp_mesh = next(item[4] for item in self.data if item[0] == nodes and item[1] == k)
        return x_vals, y_vals, temp_mesh


"""
Analytical Solution
"""


def exact_temp(x, y, k):
    temp = 100*np.sinh(k*np.pi*y)*np.sin(np.pi*x)/np.sinh(k*np.pi)
    return temp


"""
2nd Order FDM Solution
"""


def bc_temp_func(x):
    return -100*np.sin(np.pi*x)  # deg c


def FDM_2nd_order(n_nodes, k):

    # Generating A Matrix
    dim = n_nodes**2
    A = np.zeros([dim, dim])
    A += (-2*(k**2+1))*np.eye(dim)
    A += np.eye(dim, k=-n_nodes)
    A += np.eye(dim, k=n_nodes)
    A += k**2*np.eye(dim, k=1)
    A += k**2*np.eye(dim, k=-1)
    for i in range(dim):
        if i % n_nodes == 0 and i != 0:
            A[i-1][i] = 0
            A[i][i-1] = 0

    # Generating B Matrix
    B = np.zeros([dim, 1])
    dx = length/(n_nodes+1)
    for i in range(n_nodes):
        x = dx*(i+1)
        B[i][0] = bc_temp_func(x)

    # Solving Matrix
```

```
    temps = np.reshape(np.linalg.solve(A, B), [n_nodes, n_nodes])
    temps = np.row_stack((-bc_temp_func(np.linspace(0, length, n_nodes+2))[1:-1], temps))
    temps = np.row_stack((temps, np.zeros(n_nodes)))
    temps = np.column_stack((temps, np.zeros(n_nodes+2)))
    temps = np.column_stack((np.zeros(n_nodes+2), temps))

    return np.flip(temps)


"""
Heat Flux Functions
"""


# Exact Heat Transfer Function
def exact_heat_transfer(k):
    return -200*k/np.tanh(k*np.pi)


# Approximate Heat Flux Function
def approx_heat_transfer(lower, upper, nodes, fdm_temp, k):
    dTdy = []
    dx = length/nodes
    for i in range(len(fdm_temp)):
        col = fdm_temp[:, i]
        dTdy.append((col[-3]-4*col[-2]+3*col[-1])/(2*dx))

    sum = 0
    for i in range(nodes + 1):
        summand = dTdy[i]
        if (i != 0) and (i != nodes):
            summand *= (2 + (2 * (i % 2)))
        sum += summand

    val = ((upper - lower) / (3 * nodes)) * sum
    return -val


"""
Richardson Extrapolation Function
"""


def q_rich_extr(q, q2, q4):
    q_extr = (q2**2 - q*q4)/(2*q2-q-q4)
    beta = abs(np.log((q_extr - q2)/(q_extr-q4))/np.log(2))
    perc_err = abs(abs(q_extr-q)/q_extr)*100
    return q_extr, beta, perc_err


"""
Convergence Rate Function
"""


def get_beta(exact, approx, approx_2, h, h_2):
    A = np.log(abs(exact - approx))
    B = np.log(abs(exact - approx_2))
    C = np.log(h) - np.log(h_2)
    return -(A - B) / C


"""
Data Generation
"""

# FDM Solution
length = 1
FDM_Data = DataStructure()
node_list = range(1, 7)
k_list = [0.25, 0.5, 0.75, 1, 2, 5, 10]
re_log_err = []
re_dx_err = []
hf_log_err = []
hf_dx_err = []
tc_log_err = []
tc_dx_err = []
for k in k_list:
    for n in node_list:
```

```
        n_nodes = 2**n-1
        x_vals = np.linspace(0, length, n_nodes+2)
        y_vals = np.linspace(0, length, n_nodes+2)

        sol = FDM_2nd_order(n_nodes, k)
        FDM_Data.add_data(n_nodes+2, k, x_vals, y_vals, sol)

    """
    Post-Processing
    """

    # Exact Solution Plotting
    x_vals = np.linspace(0, length, 100)
    y_vals = np.linspace(0, length, 100)
    X1, Y1, = np.meshgrid(x_vals, y_vals)
    Z1 = exact_temp(X1, Y1, k)
    ax = plt.axes(projection='3d')
    ax.plot_surface(X1, Y1, Z1, rstride=1, cstride=1,
                    cmap='inferno', edgecolor='none')
    plt.title('Exact Solution 3-Dimensional Heatmap\nK = '+str(k))
    ax.set_xlabel('X (cm)')
    ax.set_ylabel('Y (cm)')
    ax.set_zlabel(r'T(x, y)$\degree$C')
    plt.show()

    # 2-Dimensional Heat Maps
    ax = sns.heatmap(Z1, xticklabels=False, yticklabels=False, cbar_kws={'label': u'Temperature \N{DEGREE
SIGN}C'})
    plt.xticks(100 * np.array([0, 0.25, 0.5, 0.75, 1]), labels=[0, 0.25, 0.5, 0.75, 1])
    ax.set_xlabel('x (cm)')
    plt.yticks(100 * np.array([0, 0.25, 0.5, 0.75, 1]), labels=[0, 0.25, 0.5, 0.75, 1])
    ax.set_ylabel('y (cm)')
    ax.set_title('Exact Solution Heatmap')
    plt.gca().invert_yaxis()
    plt.show()

    # Additional Post-Processing
    heat_flux_data = []
    temp_error_data = []
    for n in node_list:
        n_nodes = 2**n-1
        x_vals, y_vals, Z = FDM_Data.return_data(n_nodes+2, k)
        X, Y = np.meshgrid(x_vals, y_vals)

        # Heat Flux Calculations and Error Calculations
        q_exact = exact_heat_transfer(k)
        q_approx = approx_heat_transfer(0, 1, n_nodes + 1, Z, k)
        error_q = abs((q_exact-q_approx)/q_exact*100)
        heat_flux_data.append([n_nodes + 1, length / (n_nodes + 1), q_exact, q_approx, error_q])

        # Temperature Error Calculations
        midpoint_index = int(len(Z) / 2)
        midpoint = length / (n_nodes + 1) * midpoint_index
        approx_t = Z[midpoint_index][midpoint_index]
        exact_t = exact_temp(midpoint, midpoint, k)
        error_t = abs(exact_t-approx_t)/exact_t*100
        temp_error_data.append([n_nodes + 1, length / (n_nodes + 1), exact_t, approx_t, error_t])

        # 2-Dimensional Heat Maps
        ax = sns.heatmap(Z, xticklabels=False, yticklabels=False, cbar_kws={'label': u'Temperature \N{DEGREE
SIGN}C'})
        plt.xticks((n_nodes+2)*np.array([0, 0.25, 0.5, 0.75, 1]), labels=[0, 0.25, 0.5, 0.75, 1])
        ax.set_xlabel('x (cm)')
        plt.yticks((n_nodes+2)*np.array([0, 0.25, 0.5, 0.75, 1]), labels=[0, 0.25, 0.5, 0.75, 1])
        ax.set_ylabel('y (cm)')
        ax.set_title('Heatmap of cross-section of bar\n'+str(n_nodes+2)+' Nodes'+'\nK = '+str(k))
        plt.gca().invert_yaxis()
        plt.show()

        # 3-Dimensional Heat Maps
        ax = plt.axes(projection='3d')
        ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
                        cmap='inferno', edgecolor='none')
        plt.title(r'2nd order FDM Output T(x, y)$\degree$ C'+'\nNodes = '+str(n_nodes+2)+'\nK = '+str(k))
        ax.set_xlabel('X (cm)')
        ax.set_ylabel('Y (cm)')
        ax.set_zlabel(r'T(x, y)$\degree$C')
        plt.show()
```

```
    # Richardson Extrapolation and Convergence
    heat_flux_rc_data = []
    for i in range(len(heat_flux_data)-2):
        q_extr, q_b, perc_err = q_rich_extr(heat_flux_data[i][3], heat_flux_data[i+1][3],
heat_flux_data[i+2][3])
        heat_flux_rc_data.append([heat_flux_data[i][0], heat_flux_data[i][1], q_extr, heat_flux_data[i][3],
                                  perc_err, q_b])
    print('\nConvergence of Extrapolated Heat Flux (K = '+str(k)+'):')
    print(tbl.tabulate(heat_flux_rc_data, headers=['Num. Elements', 'dx', 'Extrapolated Heat Flux', 'Approx.
Heat Loss',
                                                   'Percent Error', 'Beta']))

    re_log_err.append([-np.log10(item[4]/100) for item in heat_flux_rc_data])
    re_dx_err.append([-np.log10(item[1]) for item in heat_flux_rc_data])

    # Heat Flux Convergence
    heat_flux_data[0].append('n/a')
    for i in range(len(heat_flux_data)-1):
        heat_flux_data[i+1].append(get_beta(heat_flux_data[i][2], heat_flux_data[i][3], heat_flux_data[i+1][3],
                                            heat_flux_data[i][0], heat_flux_data[i+1][0]))
    print('\nConvergence of Heat Flux (K = '+str(k)+'):')
    print(tbl.tabulate(heat_flux_data, headers=['Num. Elements', 'dx', 'Exact Heat Flux', 'Approx. Heat Loss',
                                                'Percent Error', 'Beta']))
    hf_log_err.append([-np.log10(item[4]/100) for item in heat_flux_data])
    hf_dx_err.append([-np.log10(item[1]) for item in heat_flux_data])

    # Temperature Convergence
    temp_error_data[0].append('n/a')
    for i in range(len(temp_error_data)-1):
        temp_error_data[i+1].append(get_beta(temp_error_data[i][2], temp_error_data[i][3],
temp_error_data[i+1][3],
                                             temp_error_data[i][0], temp_error_data[i+1][0]))
    print('\nConvergence of 2nd Order FDM Temperature Solution (K = '+str(k)+'):')
    print(tbl.tabulate(temp_error_data, headers=['Num. Elements', 'dx', 'Exact Midpoint Temp', 'Approx.
Midpoint Temp',
                                                 'Percent Error', 'Beta']))
    tc_log_err.append([-np.log10(item[4]/100) for item in temp_error_data])
    tc_dx_err.append([-np.log10(item[1]) for item in temp_error_data])

# Plotting Convergence Graphs
for i in range(len(k_list)):
    plt.plot(re_dx_err[i], re_log_err[i], label='K = '+str(k_list[i]))
plt.title('Extrapolated Heat Flux\n' + r'-$log_{10}$($\Delta$x) vs -$log_{10}$(Relative Error)')
plt.xlabel(r'$-log_{10}$($\Delta$x)')
plt.ylabel(r'$-log_{10}$(Relative Error)')
plt.grid()
plt.legend()
plt.show()

for i in range(len(k_list)):
    plt.plot(hf_dx_err[i], hf_log_err[i], label='K = '+str(k_list[i]))
plt.title('1/3 Simpson Integration Heat Flux\n '
          + r'-$log_{10}$($\Delta$x) vs -$log_{10}$(Relative Error)')
plt.xlabel(r'$-log_{10}$($\Delta$x)')
plt.ylabel(r'$-log_{10}$(Relative Error)')
plt.grid()
plt.legend()
plt.show()

for i in range(len(k_list)):
    plt.plot(tc_dx_err[i], tc_log_err[i], label='K = '+str(k_list[i]))
plt.title('2nd order FDM\n' + r'-$log_{10}$($\Delta$x) vs -$log_{10}$(Relative Error)')
plt.xlabel(r'$-log_{10}$($\Delta$x)')
plt.ylabel(r'$-log_{10}$(Relative Error)')
plt.grid()
plt.legend()
plt.show()
```