**AERO 430 – Assignment Two**



Antonio Diaz ‘22

**Due 02/17/2020**

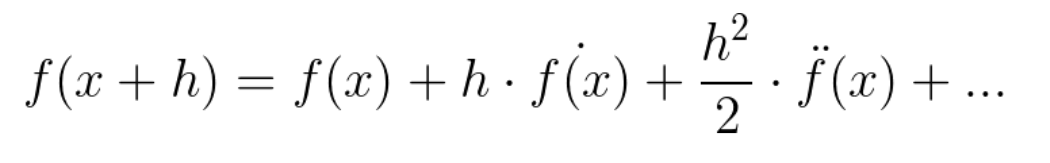
1) Formulation the problem of heat conduction in a rod assuming convective boundary condition at x=0

The heat transfer across the heat rod is described by the conservation of heat as follows:

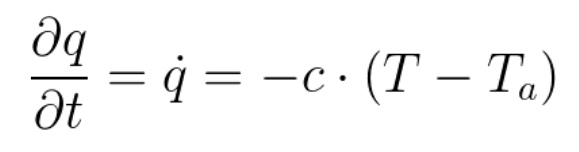


A comprehensive heat equation is then generated from three equations:

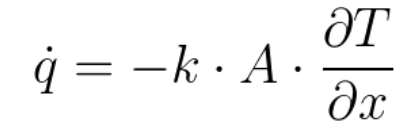
Taylors series expansion:



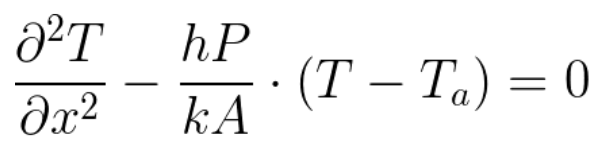
Newton law of cooling:



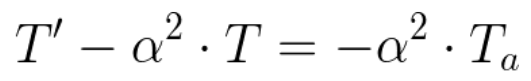
Fourier’s law of heat induction:



Combining these equations leads to:

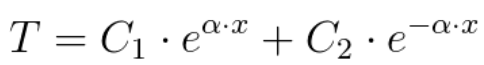


or

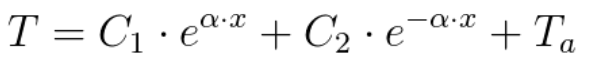


**Analytical Expression for the Exact Solution**

The second order differential equation requires a homogenous solution, taking the form:



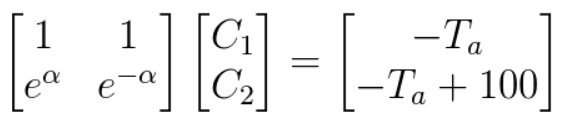
To take ambient temperature into consideration, the equation can be describe by:



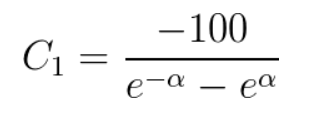
This can be rewritten as:



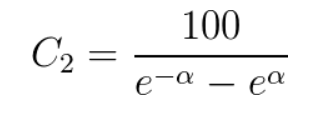
Solving for the coefficients can be done with linear systems of equations using T(0) = 0 and T(L) = 100:



This leads to:

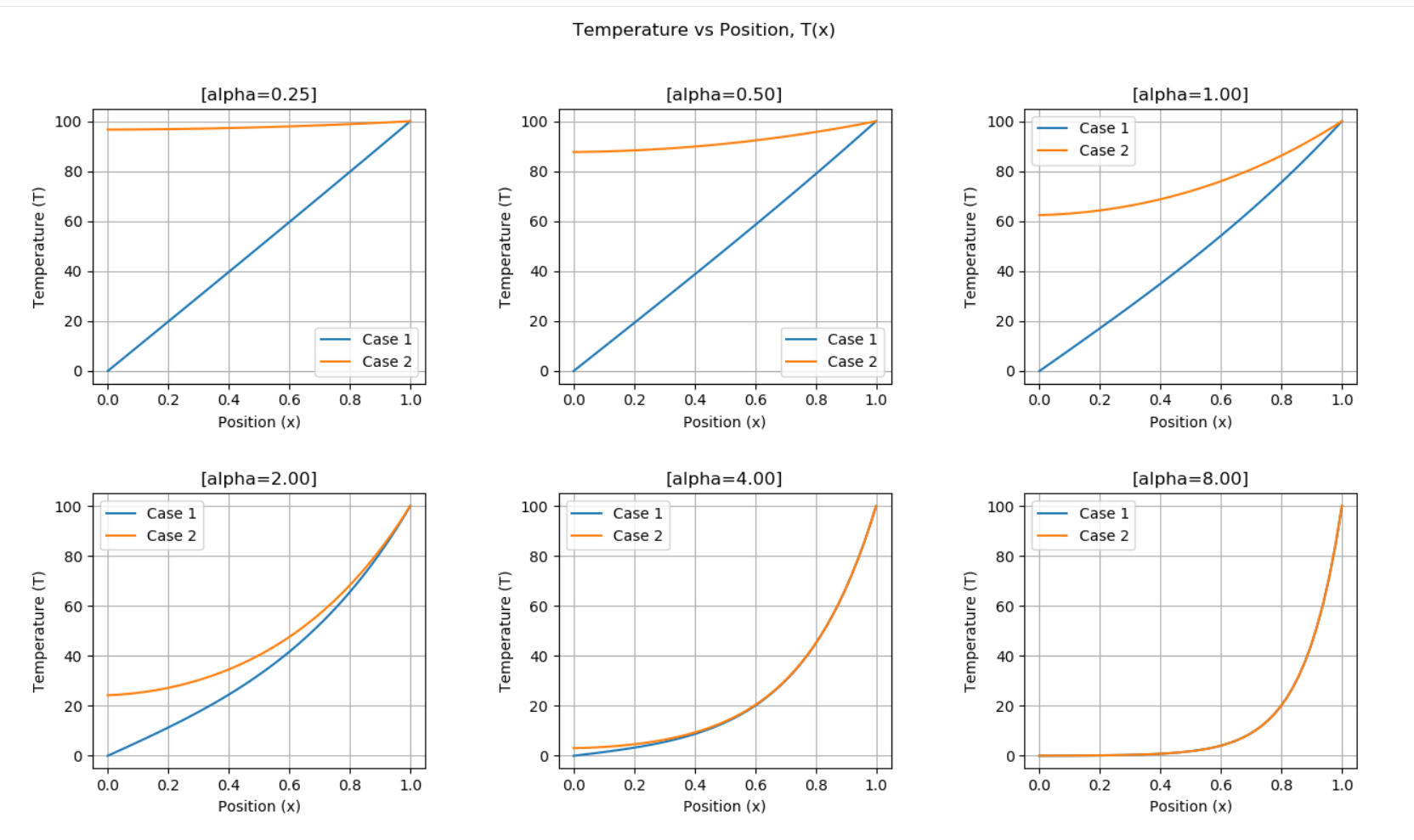


and



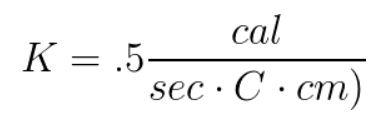
= (+)/2 and = (-+)/2

An analytical solution to the problem with varying levels of alpha for specified boundary temperatures (case 1) and specified root boundary condition with convection from free-end tip (case 2) are shown below:



2) Formulation and solution to the Finite Difference Equations.

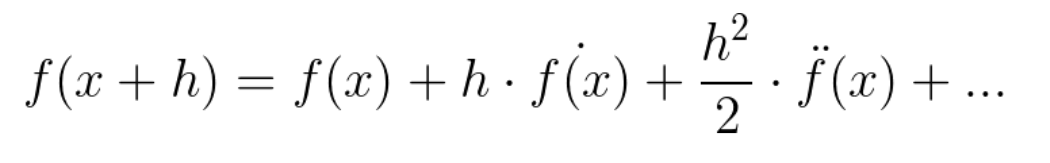
Constants are defined as follows:

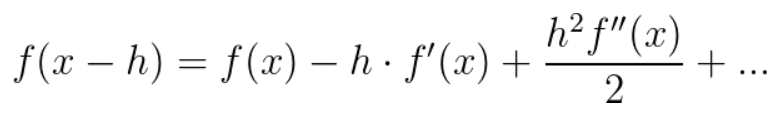


Length = 1 cm

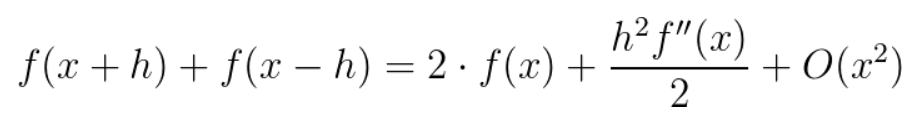
Radius = .1 cm

The finite difference equations come from Taylor series addition:

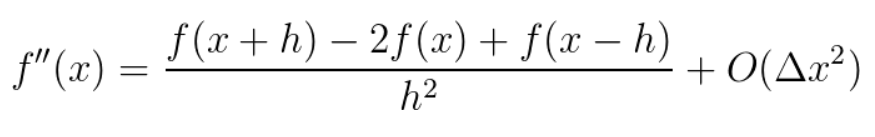




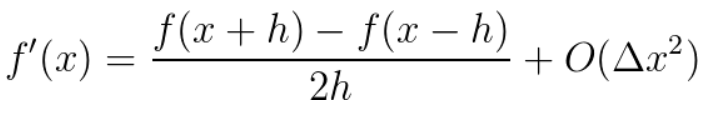
Adding both functions leads to:

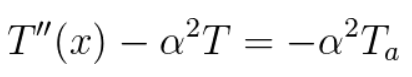


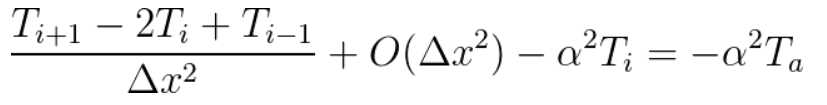
f’’(x) can then be redefined as:



Through a similar process, f’(x) is also shown by:

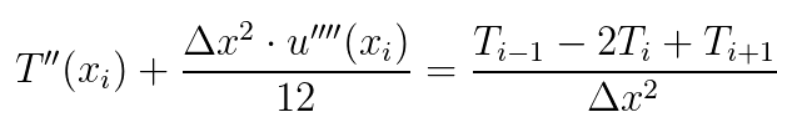


These finite difference elements can then be plugged in to :

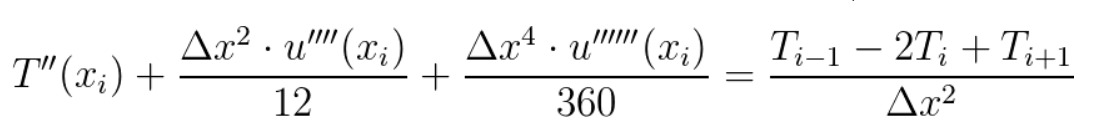


This gives a second order difference. This can be replicated with 4th order, 6th order, 8th order, and 10th order. Taylor series expansion is as follows:

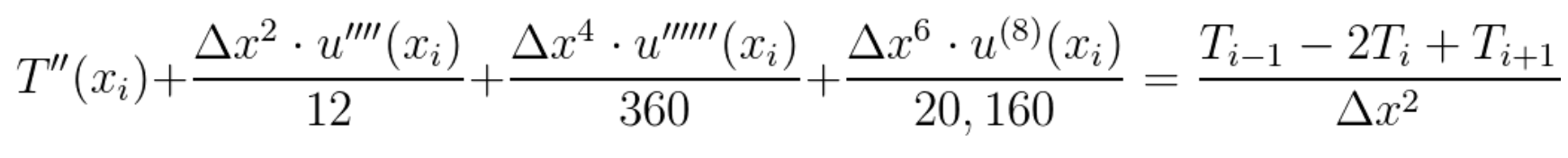
4th order expansion:



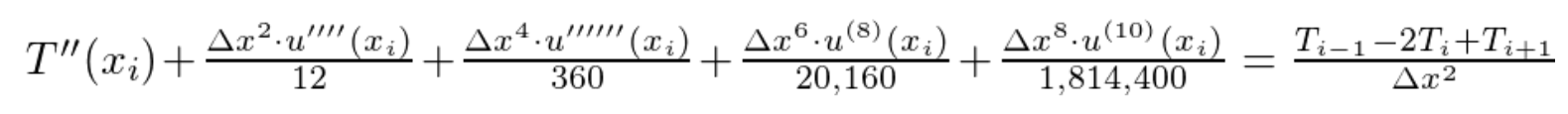
6th order expansion:



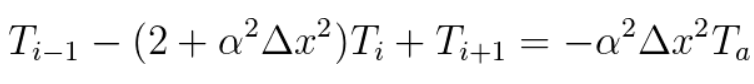
8th order expansion:



10th order expansion:

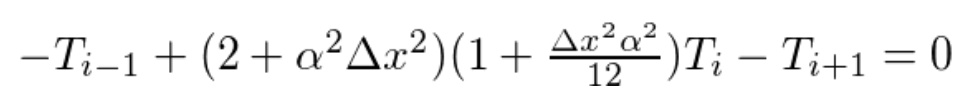


After truncation and multiplying by , the result for is given by:

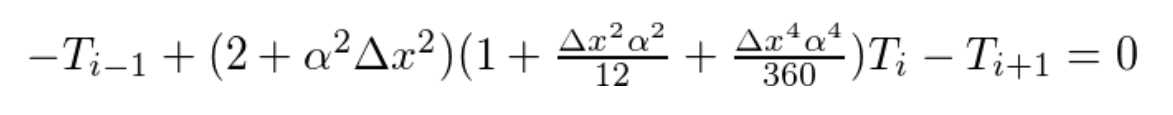


Rearranging, truncating, and multiplying by gives similar systems of equations:

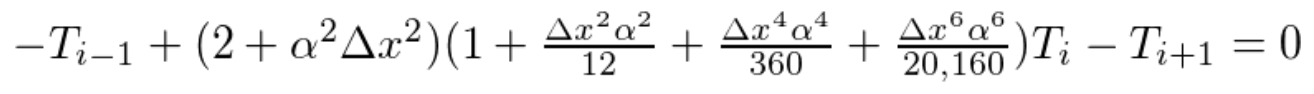
4th order:



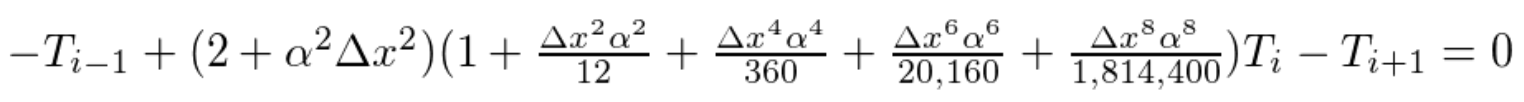
6th order:



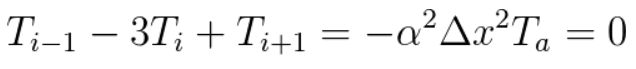
8th order:



10th order:



Using , the equation is written as:



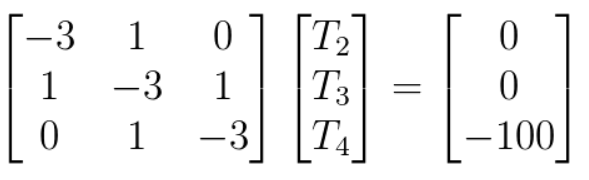
Using = 1/(), the system of equations can be written as:

x = .25cm 

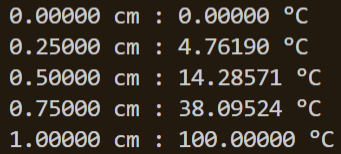
x = .50cm 

x = .75cm 

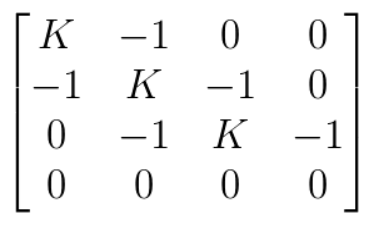
The matrix form of these equations can be rewritten as:



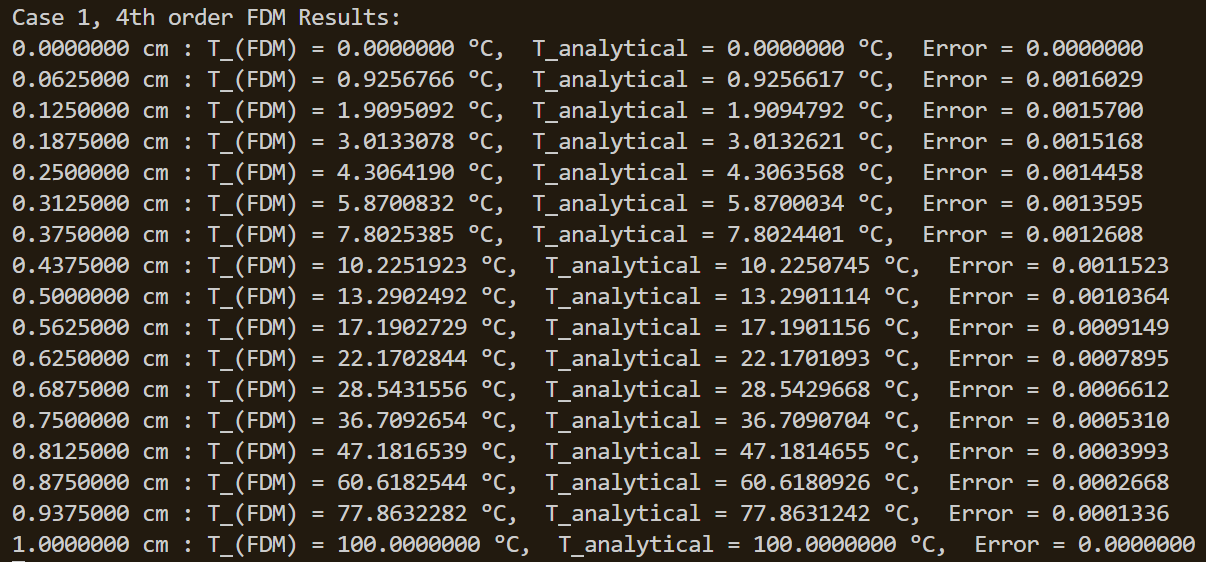
With numpy python packages, this can be programmed and leads to the same results as Bradshaw for case 1, of constrained temperature Dirichlet boundary conditions:

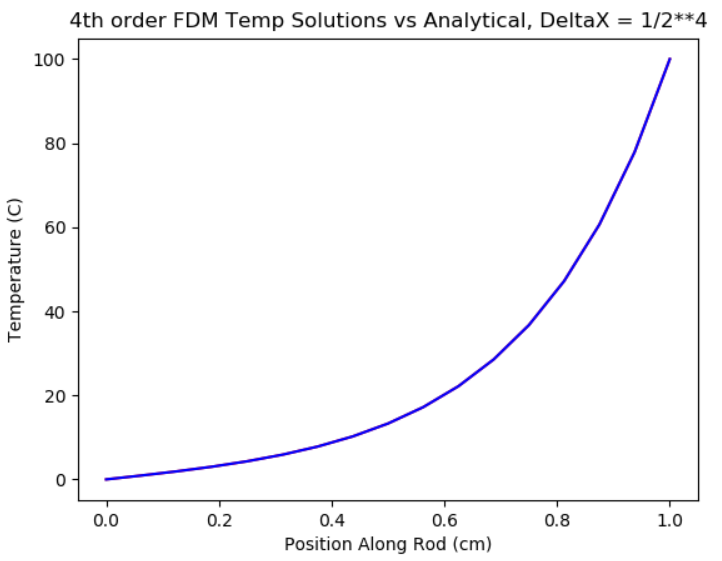


In general, these systems of equations can be written as:

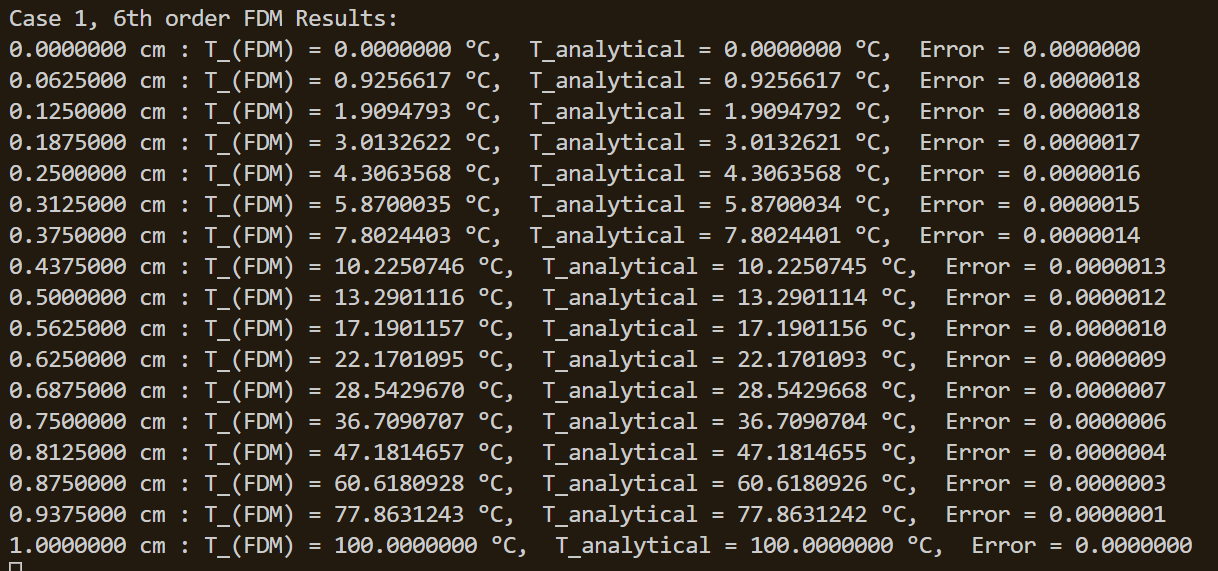


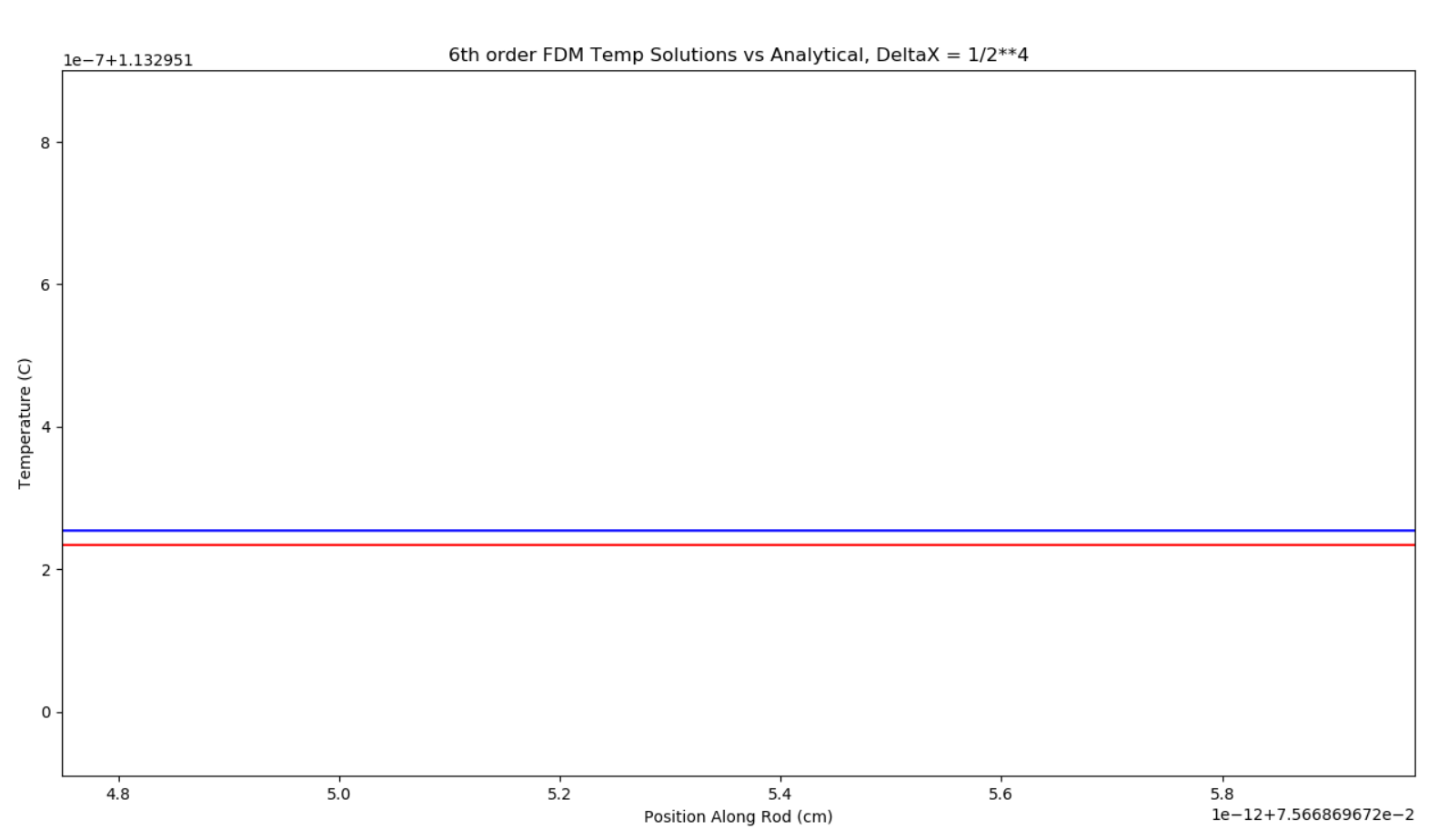
The results of second order FDM temperature are shown below:



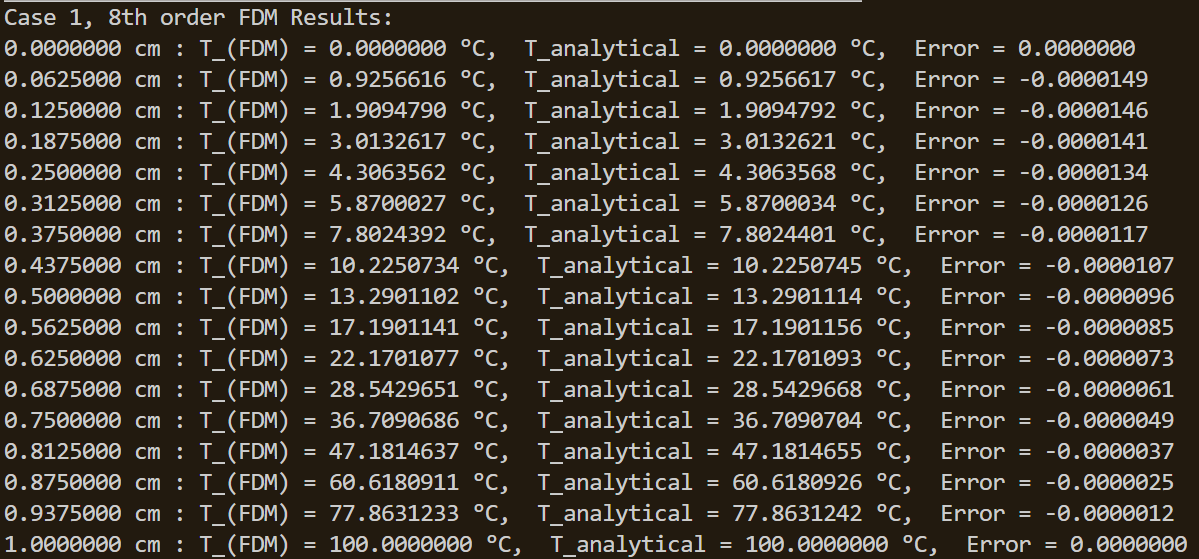


Only after zooming in 10 times can 6th order FDM even show the difference between analytical and FDM solution:

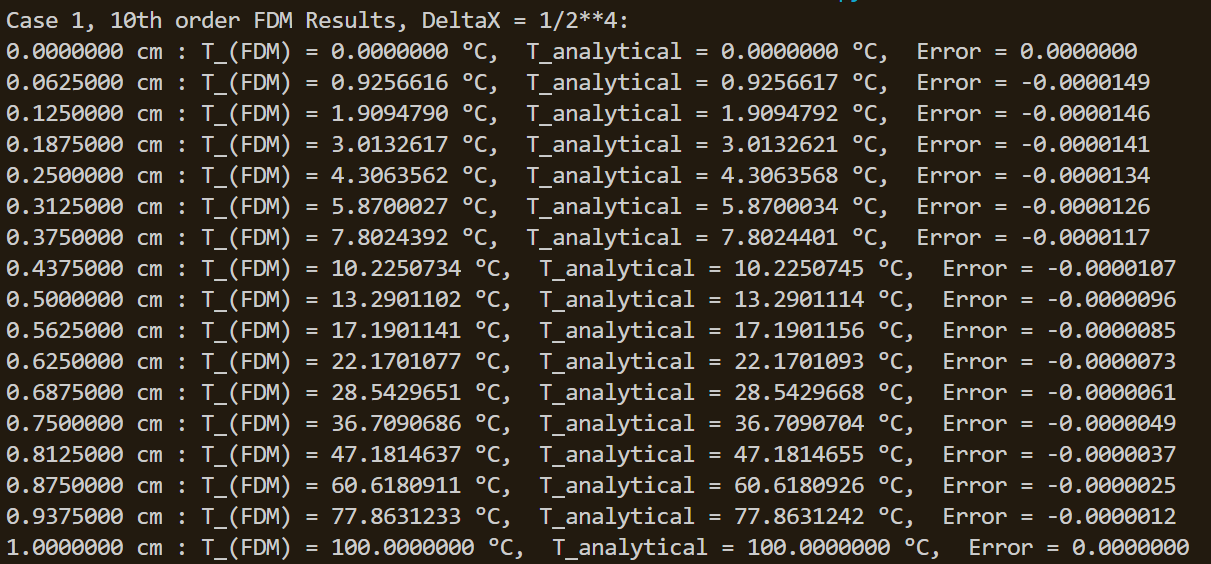




8th order FDM Results:



10th order FDM Results:

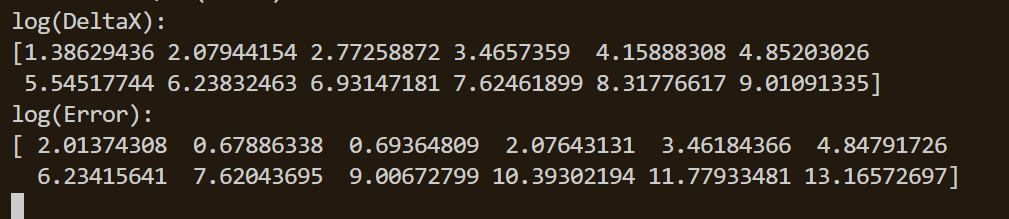


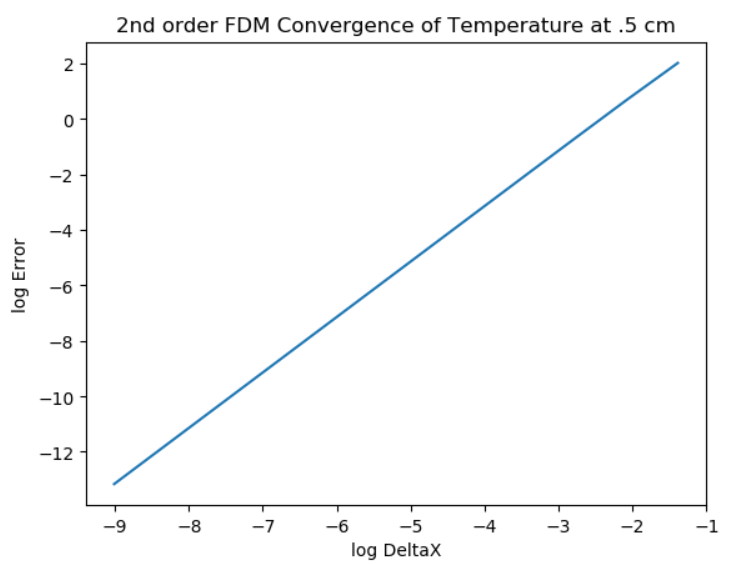
For comparison of Beta Convergence values, the FDM Temperature solution will be taken at .5000 cm for each delta X.

Beta Convergence table and graph for second order FDM:

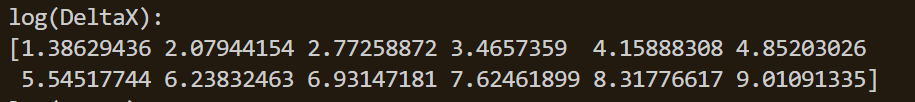
(Delta X array followed by T\_FDM at .5 cm)

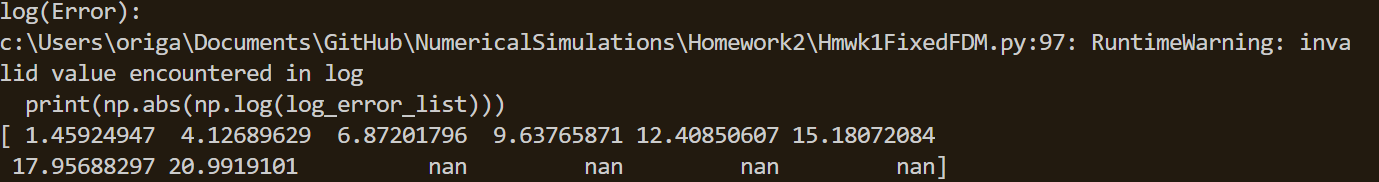
Beta= 2



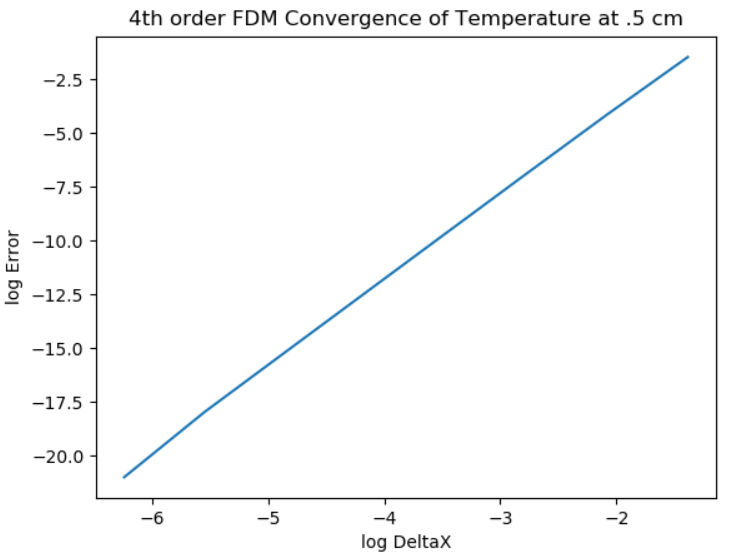


Beta = 4

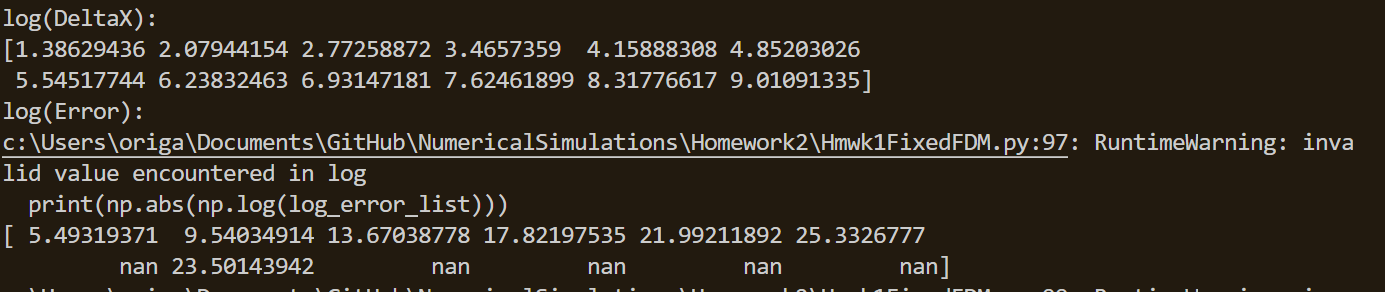


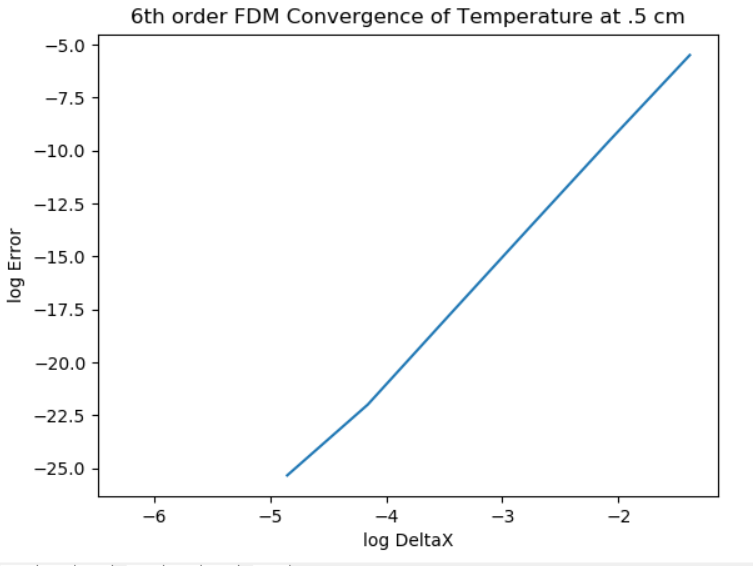


\*nan values from negative differenes between analytical and FDM solution

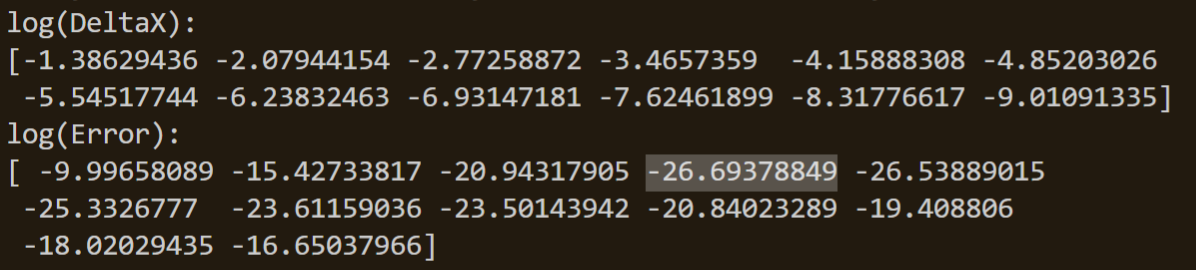


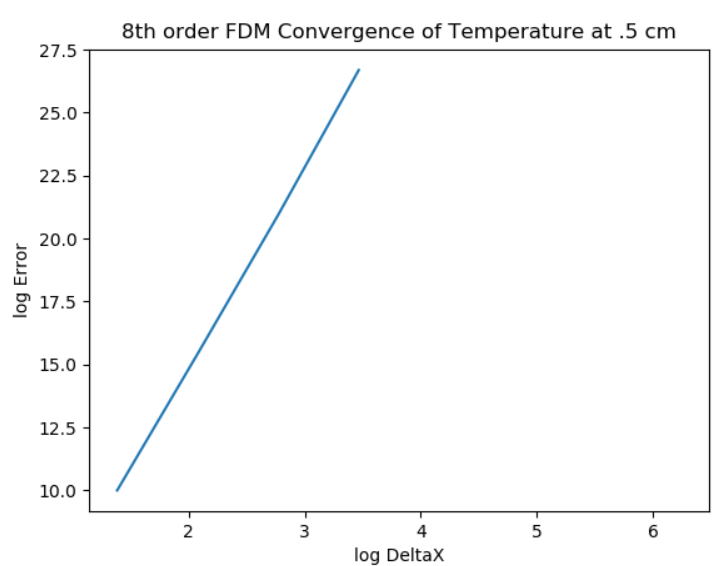
Beta = 6



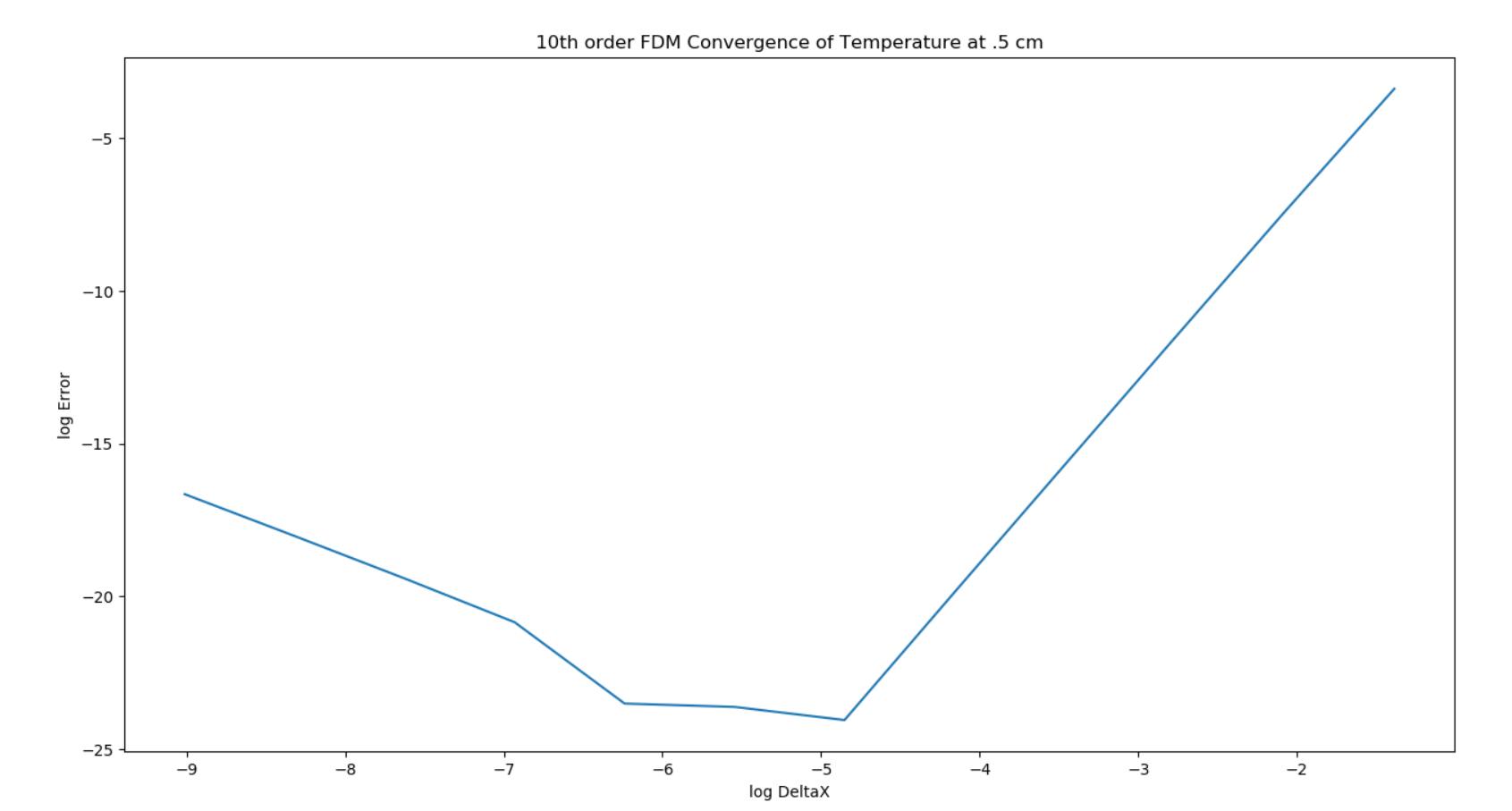


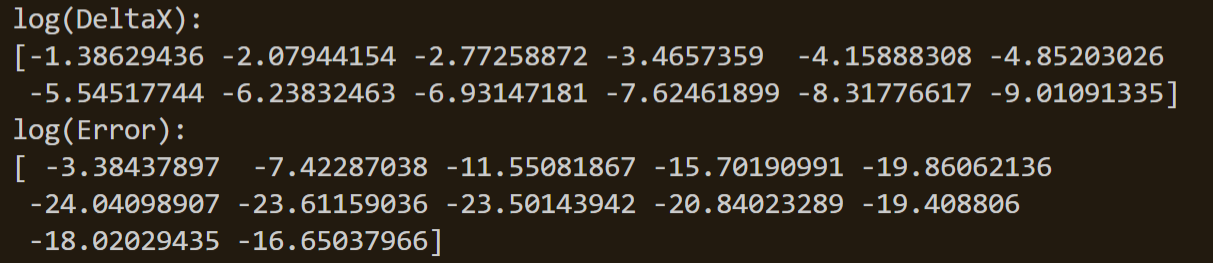
Beta = 8





Beta = 10

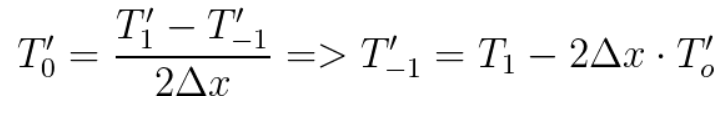




Case 2 with free, convecting tip leads to the issue of the -1 (“ghost”) index:

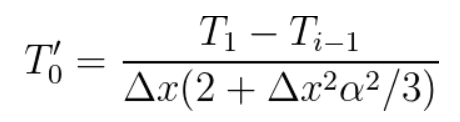


Using the first order taylor series shown earlier, can be shown to be:

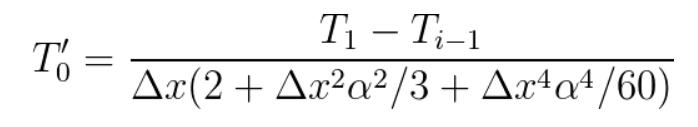


This can be extended to the 4th, 6th, 8th, and 10th order form as follows:

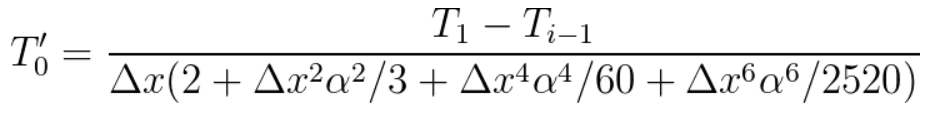
4th order form:



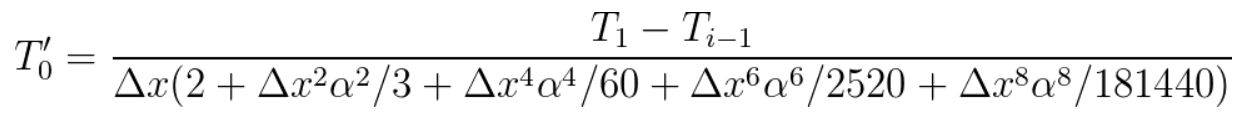
6th order form:



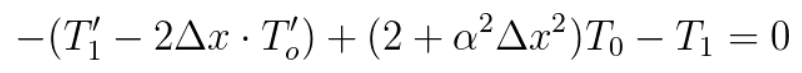
8th order form:



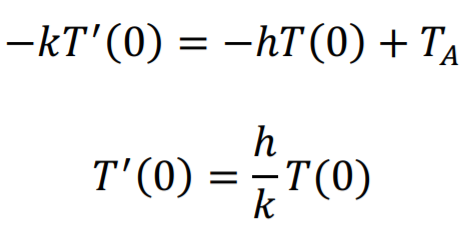
10th order form:

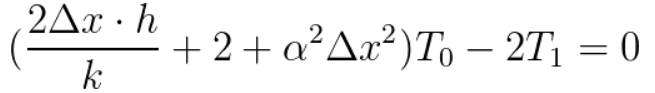


This can be replaced in our initial second order differential equation as follows:

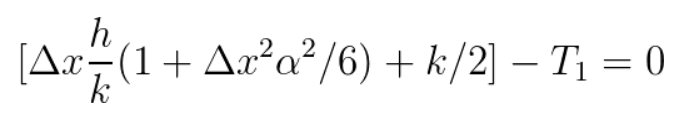


Our boundary conditions can also help simplify this further:

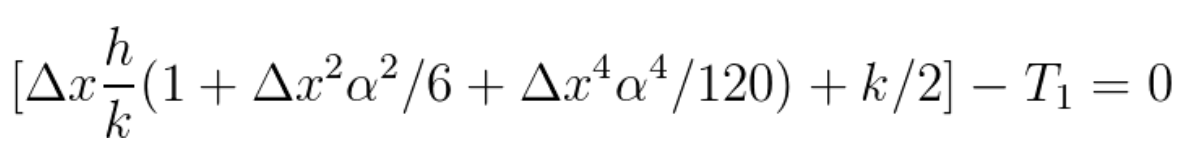




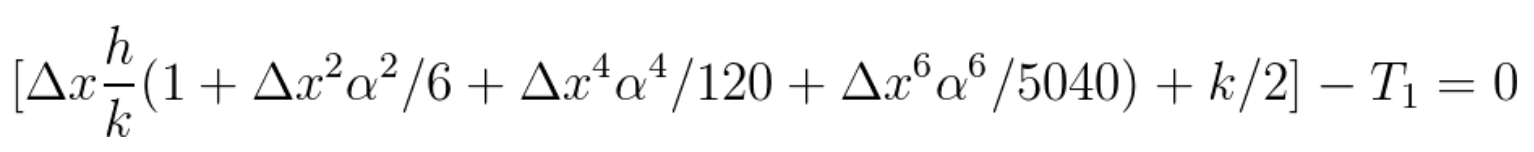
Adding the 4th order terms:



6th order terms:



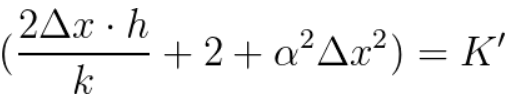
8th order terms:

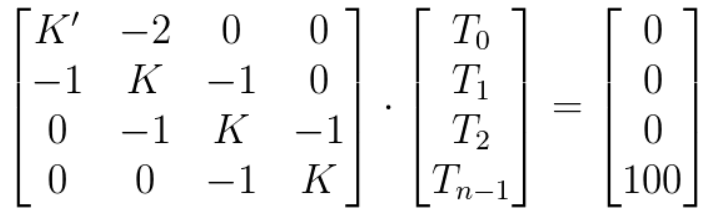


10th order terms:



This can be replaced by another matrix:



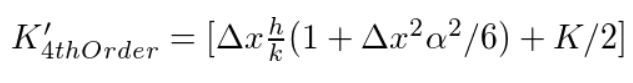


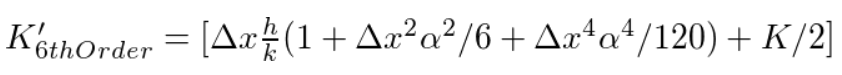
(n representing the divisions)

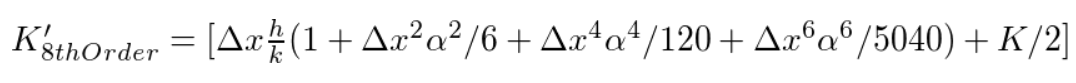
(this matrix is also equivalent to K’/2 with -1 instead of -2.)

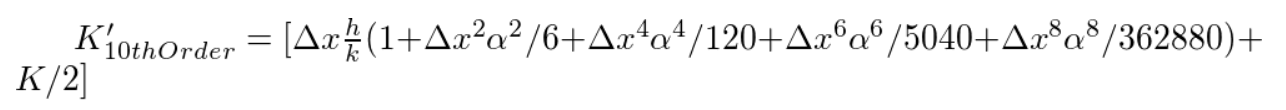
The systems of equations can also be defined with higher order values of K’:

\*k is the heat transfer coefficient while K is from the previously derived value K

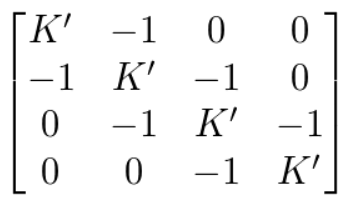




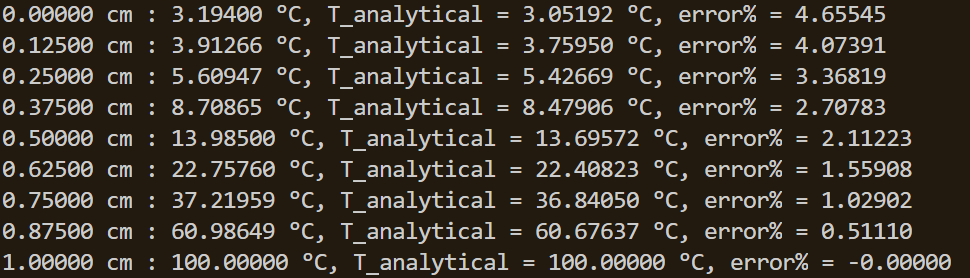




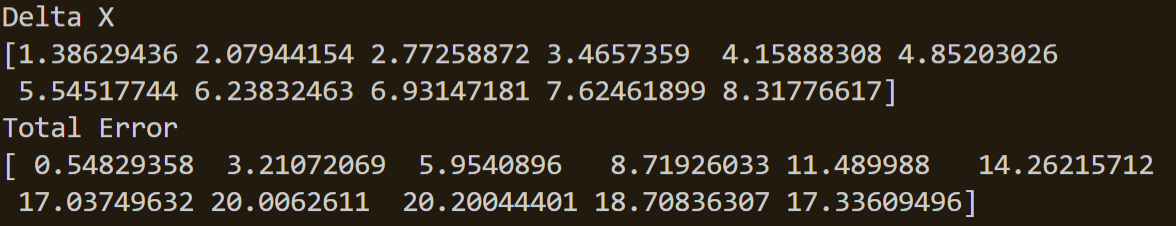
This matrix can also be represented by



alpha = 4, with a deltaX of .125 cm, below are the python code outputs:

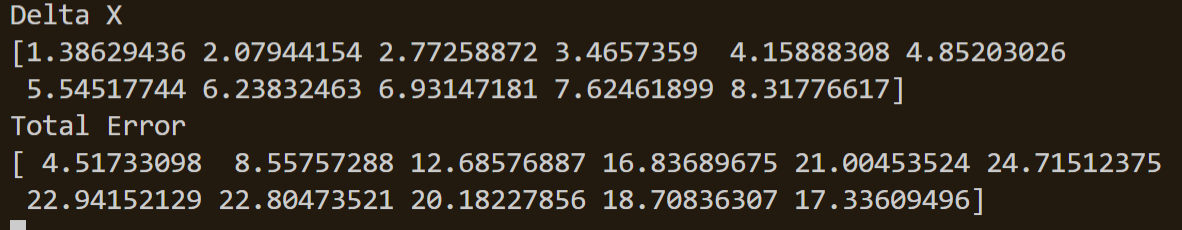


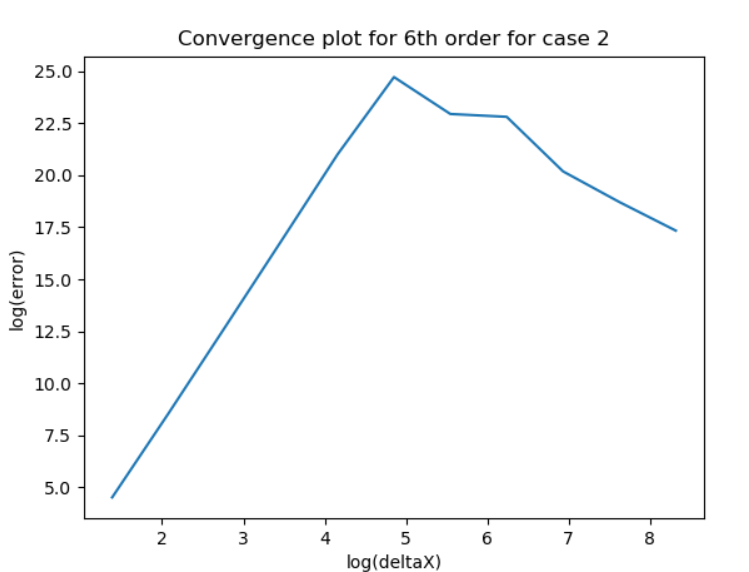
4th order data table and convergence plot with free end:



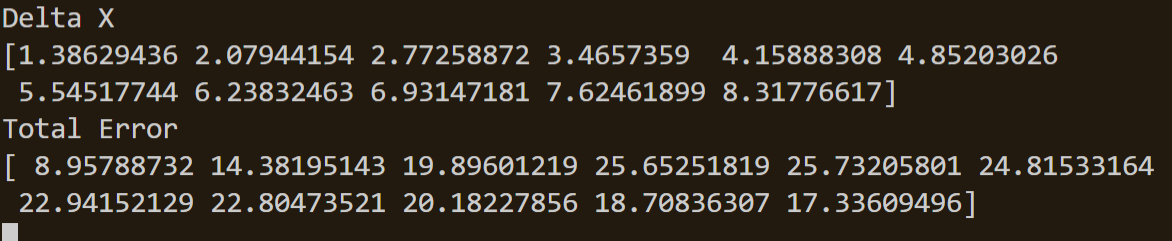


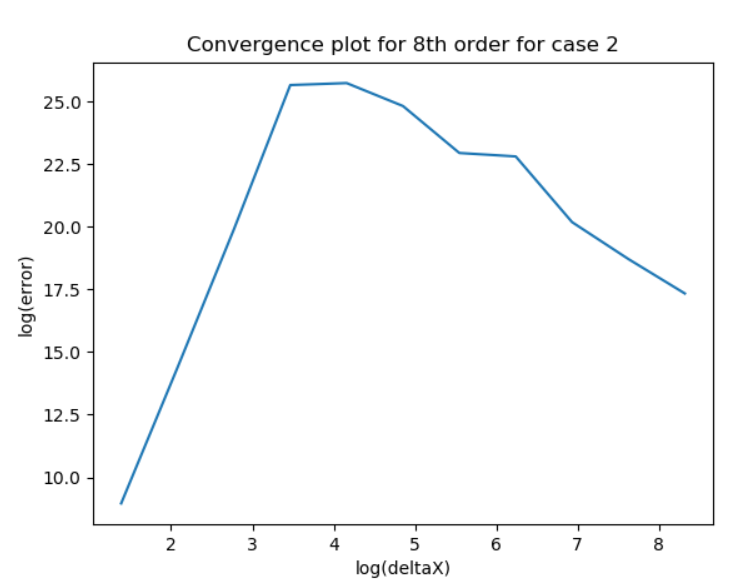
6th order data table and convergence plot:





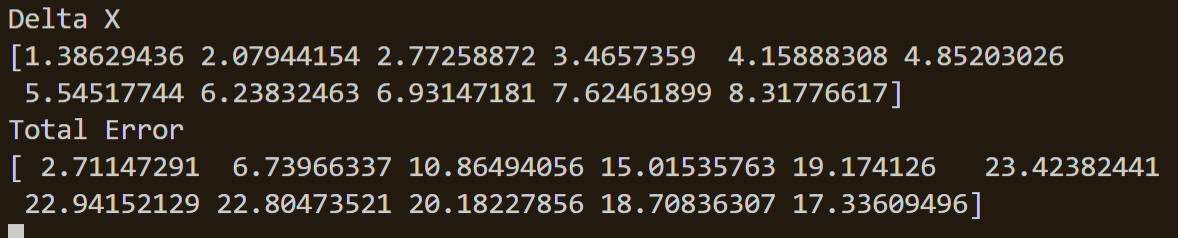
8th order data table and convergence plot

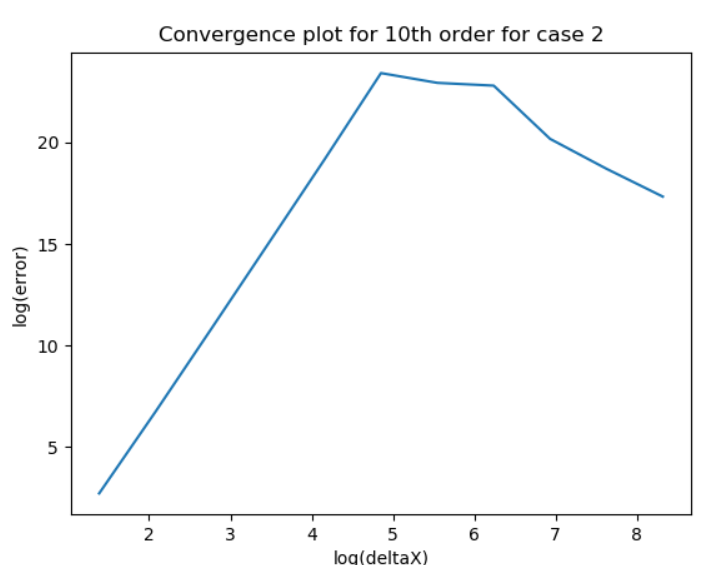




10th order data table and convergence plot

\*This plot and data did not indicate a Beta value of 10.

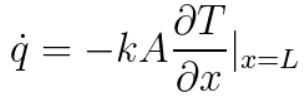




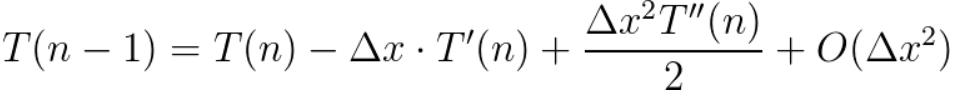
3) Compute the Heat-Loss in two ways:

1. By computing the heat entering the domain at x=L;

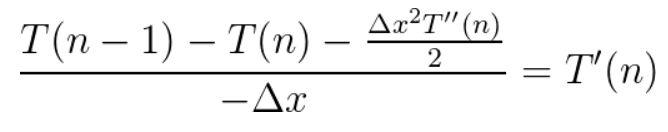
Heat loss entering the domain at x= L is found by using the definition of heat transfer across the bar:



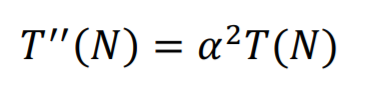
This can be approximated using Taylor series as follows:



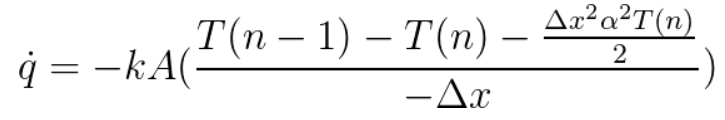
Rearranging for T’(n) leads to



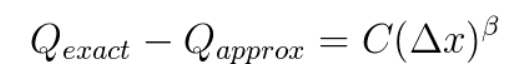
From our original second order differential equation, the following becomes true:



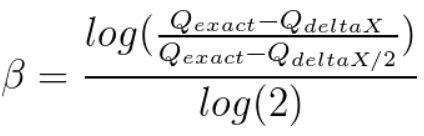
Solving for T’(n) and replacing this value in our heat loss equation leads to:



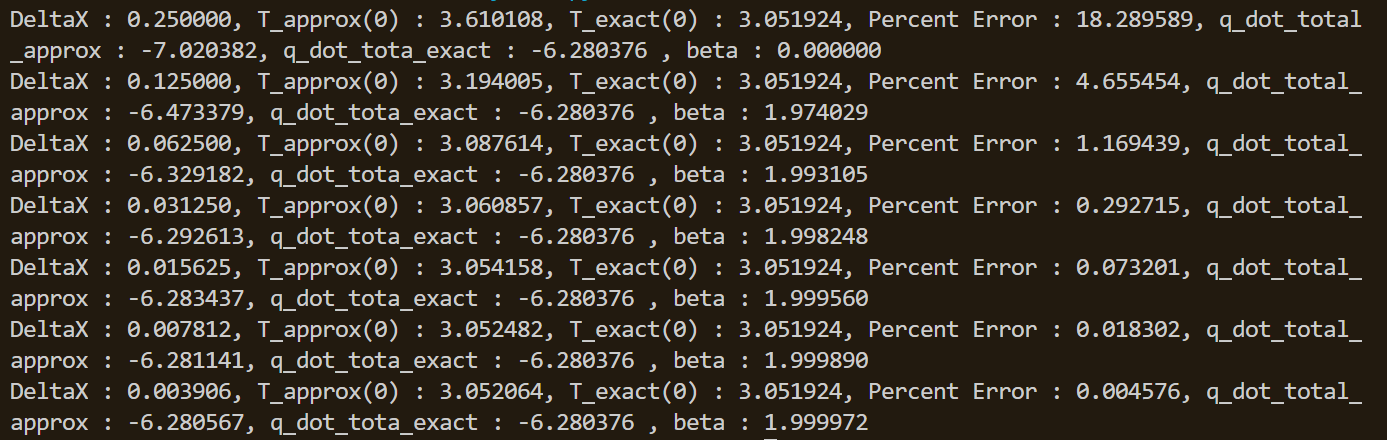
The rate of convergence between the exact solution can be defined by



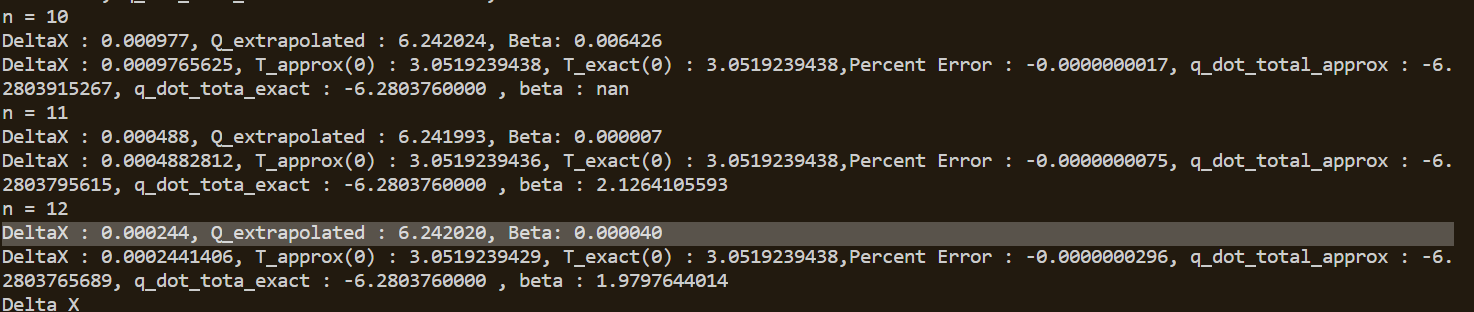
This can be rearranged with a successive approximation with:



The result of using these equations in python leads to the following results the agree with Austin Bradshaw’s results:

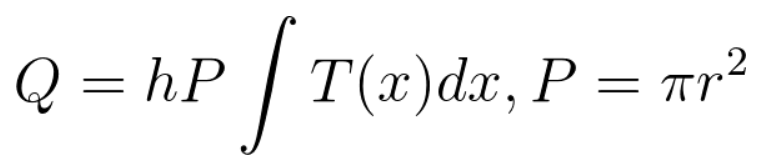


Heat entering using 10th order convergence using the same method of finding dq/dt: This data also indicates a convergence of 2 with this method.

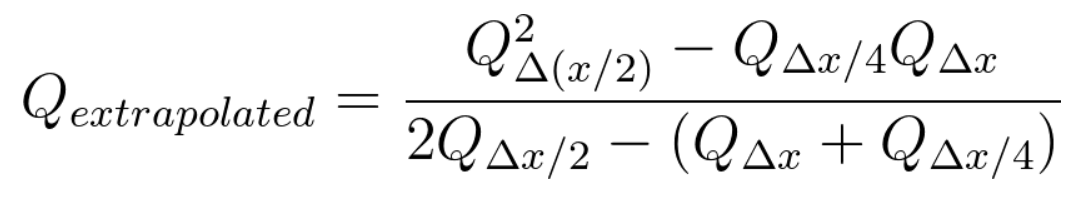


1. By computing the total heat flux exiting the domain through Newton Cooling from the lateral surface and the cross-section at x=0. Present your results in the same way as Bradshaw using Tables & Graphs and report the Convergence Rates

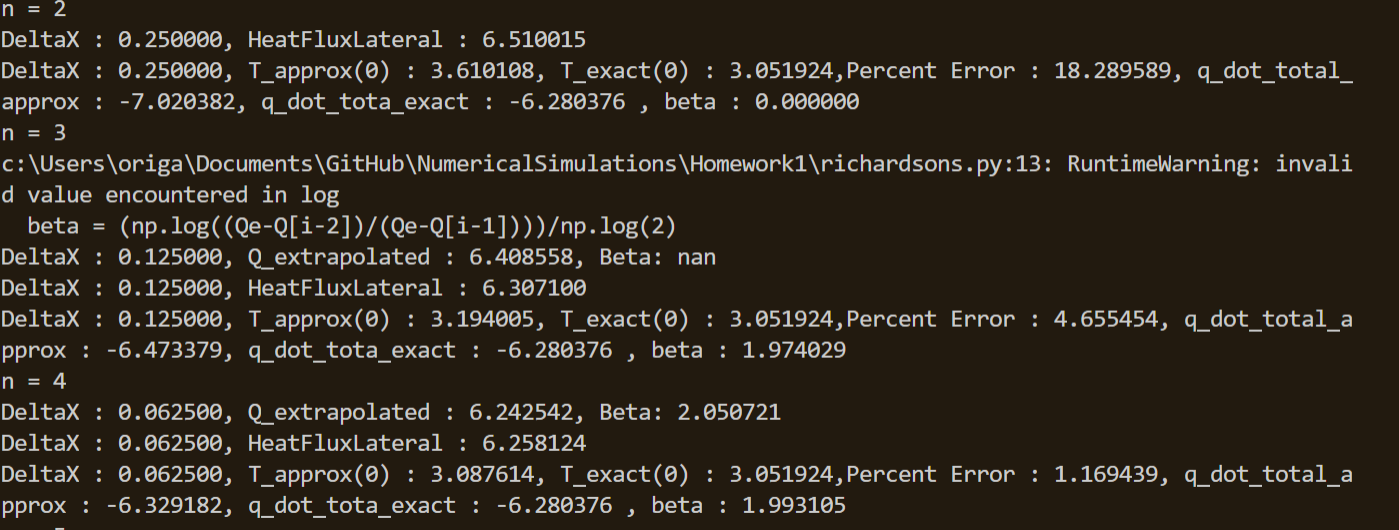
Total heat flux can be calculated by

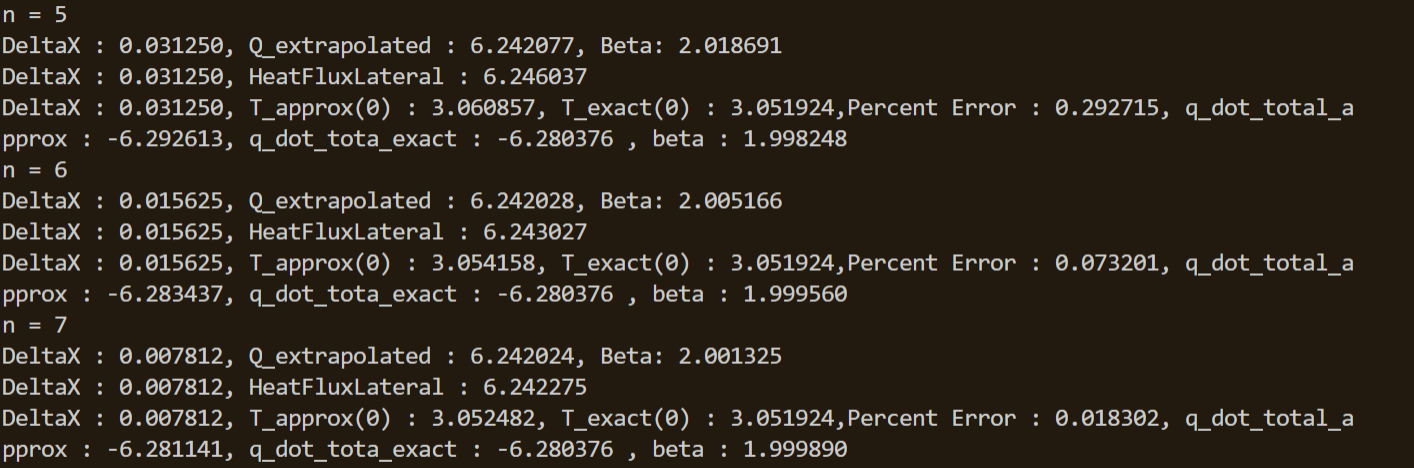


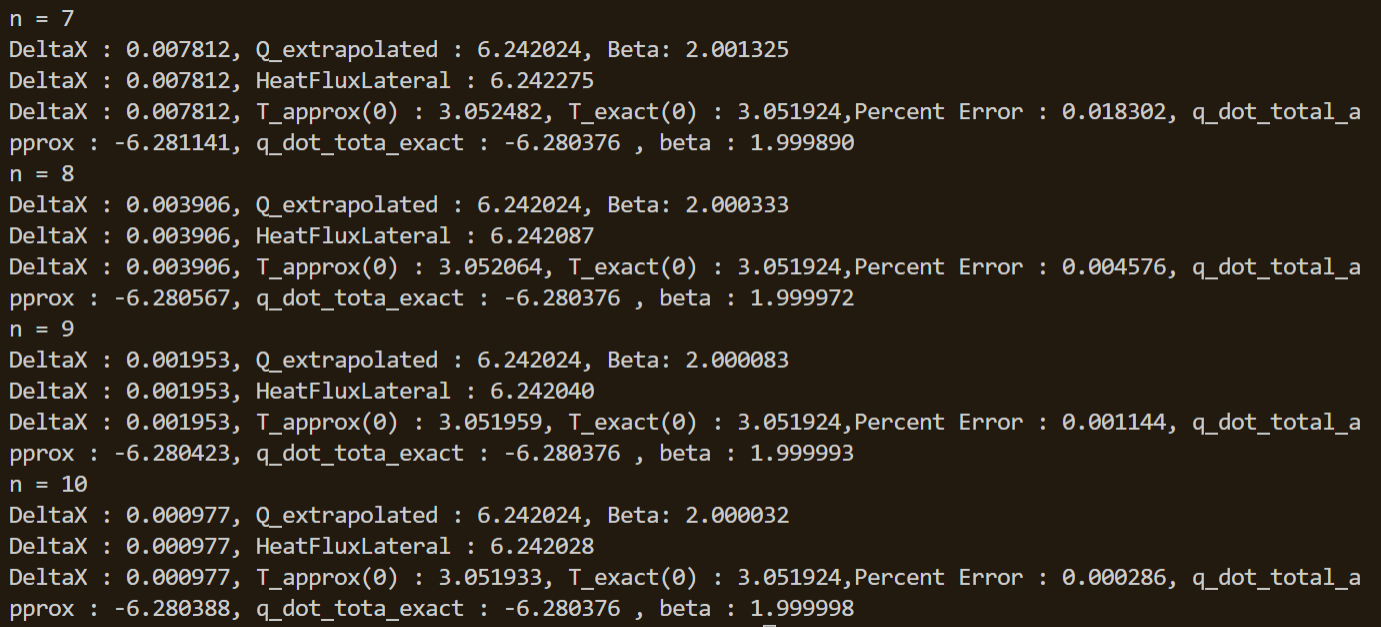
Using Simpsons rule, the total lateral heat flux can found and programmed in python. The lateral heat can also be extrapolated using Richardson extrapolation with the following relationship:



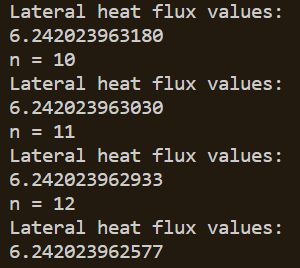
Convergence for case 2, with heat flux, extrapolated values, and Beta values are given from the code below:



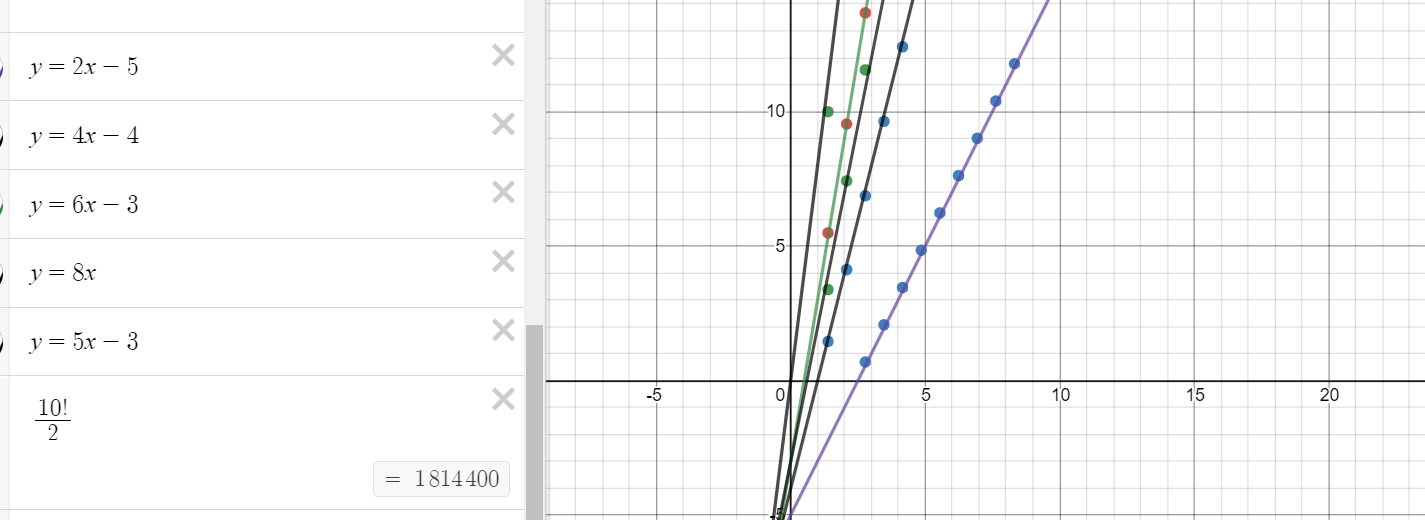




Results of calculating total heat flux using Simpson’s method on a 10th order convergence:



Overall convergence values of graphs, graphed on Desmos:



The last approximate fit on the left demonstrates the issue of convergence from 10th order.

Conclusion:

Using finite difference models, the Temperature as a function of position on the heat rod can be estimated. The current models used in this homework assignment make use of second order finite difference models, 3rd order FDM, 4th order FDM, 6th order FDM, 8th order FDM, and 10th order FDM. The values of convergence reach their expected values for all orders except 10th order FDM. The code I created in Python for the class uses the same models and ideas, however it is inefficient in solving the temperatures. The code does not use the Thomas algorithm, instead just using the numpy packages to find the inverse.

Homework 2 was useful in seeing how quickly the accuracy can improve with higher orders of FDM convergence. ­­