Author: Liam Nester

Created: February 20, 2021

GitHub: https://github.com/LiamEngMan

Backround Information

Perhaps, the most important topic discussed in Chapters 1-3 is the Sturm-Liouville Theory. It has been described by many as the most useful result ever devised by mathematicians. Many of the summed series that you find in tables of series can be traced back to the Sturm-Lioville Theory. Understanding this application is essential to your understanding of the Equations of Mathematical Physics. The linear partial differential equations discussed later in this course are often referred to as the Equations of Mathematical Physics. The following problem has been designed to step you through the process of application of this theory.

This is a specific application of the theory that allows you to see exactly how the theory leads to a solution. At the same time as you are carrying out the process of solution, think about how many other eigenvalue problems could apply to the same series expansion. You should be able to see that there are infinitely many ways to expand the same function into a series of eigenfunctions that result from the solution of eigenvalue problems. It is a remarkable conclusion that all of these series will converge to the same function on the interval in question. These constructions are formal in the sense that any series expansion is just a mathematical artifact until its convergence has been established. The question of convergence of a series of eigenfunctions has been settled by mathematicians, but not discussed to any degree in the book. That subject has been left for students who want to pursue the subject further. In the mean time, we can make a shallow attempt to verify convergence by summing the series pointwise numerically. This should give us some insight into the rate of convergence and its replication of the function on the interval in question.

By solving the eigenvalue problem, we mean find the eigenvalues, λ_n, and the nontrivial eigenfunctions, $y_n(x)$, that satisfy the boundary conditions and the differential equation. As you will see when we turn to the Heat Equation, a Parabolic Partial Differential Equation for which the boundary condition at $x = 2$, has a very significant physical interpretation within the context of a one-dimensional heat flow problem.

After. Expand the function $f(x)$, below, into a series of the eigenfunctions, evaluate the coefficients, c_n, using the orthogonality of the eigenfunctions, $y_n(x)$, relative to the appropriate weight factor on the interval $0 < x < 2$. Then, verify the convergence of the series expansion pointwise across the interval by truncating the series to 10, 25, and 0 terms and evaluating the resulting mathematical expression at enough points on the interval to see that the series converges to $f(x) = 1$.

Instructions:

Solve the eigenvalue problem for $y = y(x)$ given below on the interval $0 < x < 2$.

$$y'' + 4y' + \lambda y \tag{Eq. 1}$$

$$y(0) = 0, y(2) + \frac{1}{2}y'(2) = 0 \tag{Eq. 2}$$

Eigenvalue Problem Solution

$$y'' + 4y' + \lambda y \tag{Eq. 2}$$

$$r^2 + 4r + \lambda = 0 \quad \Rightarrow \quad r = \frac{-4 \pm \sqrt{(4)^2 - (4)(1)(\lambda)}}{2(1)} \quad \Rightarrow \quad r = -2 \pm \sqrt{4 - \lambda} \quad (\text{:}$$

Now, we will analyze three seperate scenarios in which the roots exist.

$$Assume: 4 - \lambda = \omega^2 \quad \Rightarrow \quad r_i = -2 \pm \omega \tag{Eq. 5}$$

Case 1: $\omega^2 > 0$

Assume the solution to be: $y = e^{r_i x}$

$$y(x) = C_1 e^{(-2+\omega)x} + C_2 e^{(-2-\omega)x} \tag{Eq. 6}$$

Take the derivative of equation 6:

$$y'(x) = C_1(-2+\omega)e^{(-2+\omega)x} + C_2(-2-\omega)e^{(-2-\omega)x} \tag{Eq. 7}$$

Apply the first boundary condition to equation 6:

$$y(x=0) = 0 \rightarrow C_1 e^{(-2+\omega)(0)} + C_2 e^{(-2-\omega)(0)} = C_1 + C_2 = 0 \tag{Eq. 8}$$

Apply the second boundary condition to equations 6 and 7:

$$y(2) + \frac{1}{2}y'(2) = 0 \rightarrow C_1 e^{(-2+\omega)(2)} + C_2 e^{(-2-\omega)(2)} + \frac{1}{2}\Big[C_1(-2+\omega)e^{(-2+\omega)(2)} + C_2(-2$$

$$\therefore C_2 = 0 \; and \; C_1 = 0$$

From these results, we can see only trivial solutions exist for this case.

Case 2: $\omega^2 = 0$

Again, assume the solution to be: $y = e^{r_i x}$

$$y(x) = C_1 e^{-2x} + C_2 x e^{-2x} \tag{Eq. 10}$$

Take the derivative of equation 10:

$$y(x) = -2C_1 e^{-2x} + C_2 e^{x} + 2C_2 x e^{2x} \tag{Eq. 11}$$

Apply the first boundary condition to equation 10:

$$y(x=0) = 0 \rightarrow y(0) = C_1 e^{-2(0)} + C_2(0)e^{-2(0)} = 0 \quad \Rightarrow \quad C_1 = 0 \tag{Eq. 12}$$

Apply the second boundary condition to equations 10 and 11:

$$y(x=2) + \frac{1}{2}y'(x=2) = 0 \rightarrow 2C_2 e^{2(2)} + \frac{1}{2}\left[C_2 e^2 + 2C_2(2)e^{2(2)}\right] = 2C_2 e^2 \left[e^2 + \frac{1}{4} +$$

$$\therefore C_2 = 0$$

Again, from these results, we can see only trivial solutions exist for this case.

Case 3: $-\omega^2 < 0$

It is important to note that since ω^2 is negative, an imaginary solution exists.

$$y(x) = C_1 e^{-2x} cos(\omega x) + C_2 e^{-2x} sin(\omega x) \tag{Eq. 14}$$

This time, we will use the first boundary condition on equation 14 to make the problem more simple.

$$y(x=0) = 0 \rightarrow y(0) = C_1 e^{-2(0)} cos\left[\omega(0)\right] + C_2 e^{-2(0)} sin\left[\omega(0)\right] = 0 \quad \Rightarrow \quad C_1 = 0 \quad ($$

Take the derivative of equation 14:

$$y(x) = -2C_2e^{-2x}sin(\omega x) + C_2\omega e^{-2x}cos(\omega x) \quad \text{(Eq. 16)}$$

Apply the second boundary condition to equations 14 and 16:

$$y(x=2) + \frac{1}{2}y'(x=2) = 0 \rightarrow C_2e^{-2(2)}sin(2\omega) + \frac{1}{2}\left[-2C_2e^{-2(2)}sin(2\omega) + C_2\omega e^{-2(2)}cos(2\omega)\right]$$

$$\Rightarrow \quad C_2\omega e^{-4}cos(2\omega) = 0 \quad \text{(Eq. 17)}$$

Here, $\omega, e^{-4},\ nor\ C_2$ can be equal to zero, or it will drive to another trivial solution...

The only logical determination is that $cos(2\omega) = 0$.

$$cos(2\omega) = 0 \quad at \quad 2\omega = \frac{2n-1}{2}\pi \quad \Rightarrow \quad \omega = \frac{2n-1}{4}\pi \quad \text{(Eq. 18)}$$

Write the Eigenvalues and Eigenfunctions

$$y_n(x) = e^{-2x}sin\left(\tfrac{2n-1}{4}\pi x\right) \quad \text{(Eq. 19)}$$

...from equation 14 (Note: the constant multiplyer can be left off.

$$\lambda_n = \left[\tfrac{2n-1}{4}\pi\right]^2 + 4 \quad \text{(Eq. 20)}$$

...from equation 5.

Expanding the Eigenfunction

Now, we need to expand the eigenfunction (equation 19) on the scale from the background...

To expand the equation, we will use the following relation:

$$f(x) = \sum_{n=1}^{\infty} C_m y_m(x)$$

To solve for the constants, C_n, we will use the following relation:

$$C_m = \frac{\int_0^2 f(x)w(x)y_m(x)\,dx}{\int_0^2 w(x)y_m^2\,dx}, \quad w(x) = e^{4x} \quad \Rightarrow \quad C_m = \frac{\int_0^2 e^{4x}e^{-2x}sin\left(\frac{2m-1}{4}\pi x\right)\,dx}{\int_0^2 e^{4x}\left[e^{-2x}sin\left(\frac{2m-1}{4}\pi x\right)\right]^2\,dx} \quad \Rightarrow$$

Demonstrating the Expansion Converges to a Value of 1

Assume: $\beta = \frac{2m-1}{4}\pi$

Our matrix (for storing the values) will be in the following order:

| x value | m value | β value | $C_m * y_m(x_1)$ | $C_m * y_m(x_2)$ | ... | |:-----------: |:-----------: |:-----------: |:-----------: |:-----------: |:-----------: | |x_{init} | 1 | β_1 | $solution_1$ | $solution_1$ | ... | |x_{init}+step | 2 | β_2 | $solution_2$ | $solution_1$ | ... |

for values of m from 1 to 0.

Define our Functions

In [103…

```
import math

#------------------------------------------------------------------------------%

def beta_func(mval):

    beta = ((2*mval)-1)*(math.pi/4)

    return(beta)

#------------------------------------------------------------------------------%

def y_func(betaval, xval):

    y = math.exp(-2*xval)*math.sin(betaval*xval)

    return(y)

#------------------------------------------------------------------------------%

def C_func (betaval):

    C = ((-math.exp(4)*betaval*math.cos(2*betaval))+(2*math.exp(4)*math.sin(2*betava

    return(C)

#------------------------------------------------------------------------------%

def f_func (yval, Cval):

    f = Cval*yval

    return(f)
```

In [103…

```
import numpy as np

MATRIX = np.zeros((1050,6), dtype = float)
```

In [103…

```
MATRIX_step_pos = 0
```

```python
x = 0

while x < 2.1:

    for m in range (1,51):

        BETA = beta_func(m)
        CM = C_func(BETA)
        YM = y_func(BETA, x)
        F = f_func(YM, CM)

        MATRIX[MATRIX_step_pos,0] = x
        MATRIX[MATRIX_step_pos,1] = m
        MATRIX[MATRIX_step_pos,2] = BETA
        MATRIX[MATRIX_step_pos,3] = CM
        MATRIX[MATRIX_step_pos,4] = YM
        MATRIX[MATRIX_step_pos,5] = F

        MATRIX_step_pos += 1

    x += 0.1
```

In [103…

```python
# print(MATRIX)

np.savetxt("test.txt", MATRIX)
```

In [103…

```python
def sum_func(array):

    sums = sum(array)

    return(sums)
```

The next several blocks break the generated matrix into smaller matrices.

In [103…

```python
MATRIX_00 = np.zeros((50,6), dtype = float)
MATRIX_01 = np.zeros((50,6), dtype = float)
MATRIX_02 = np.zeros((50,6), dtype = float)
MATRIX_03 = np.zeros((50,6), dtype = float)
MATRIX_04 = np.zeros((50,6), dtype = float)
MATRIX_05 = np.zeros((50,6), dtype = float)
MATRIX_06 = np.zeros((50,6), dtype = float)
MATRIX_07 = np.zeros((50,6), dtype = float)
MATRIX_08 = np.zeros((50,6), dtype = float)
MATRIX_09 = np.zeros((50,6), dtype = float)
MATRIX_10 = np.zeros((50,6), dtype = float)
MATRIX_11 = np.zeros((50,6), dtype = float)
MATRIX_12 = np.zeros((50,6), dtype = float)
MATRIX_13 = np.zeros((50,6), dtype = float)
MATRIX_14 = np.zeros((50,6), dtype = float)
MATRIX_15 = np.zeros((50,6), dtype = float)
MATRIX_16 = np.zeros((50,6), dtype = float)
MATRIX_17 = np.zeros((50,6), dtype = float)
MATRIX_18 = np.zeros((50,6), dtype = float)
MATRIX_19 = np.zeros((50,6), dtype = float)
MATRIX_20 = np.zeros((50,6), dtype = float)
```

In [103…

```python
for i in range(0, 50):
```

```
    MATRIX_00[i, 0] = MATRIX[i, 0]
    MATRIX_00[i, 1] = MATRIX[i, 1]
    MATRIX_00[i, 2] = MATRIX[i, 2]
    MATRIX_00[i, 3] = MATRIX[i, 3]
    MATRIX_00[i, 4] = MATRIX[i, 4]
    MATRIX_00[i, 5] = MATRIX[i, 5]

#-----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_01[i, 0] = MATRIX[i+50, 0]
    MATRIX_01[i, 1] = MATRIX[i+50, 1]
    MATRIX_01[i, 2] = MATRIX[i+50, 2]
    MATRIX_01[i, 3] = MATRIX[i+50, 3]
    MATRIX_01[i, 4] = MATRIX[i+50, 4]
    MATRIX_01[i, 5] = MATRIX[i+50, 5]

#-----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_02[i, 0] = MATRIX[i+100, 0]
    MATRIX_02[i, 1] = MATRIX[i+100, 1]
    MATRIX_02[i, 2] = MATRIX[i+100, 2]
    MATRIX_02[i, 3] = MATRIX[i+100, 3]
    MATRIX_02[i, 4] = MATRIX[i+100, 4]
    MATRIX_02[i, 5] = MATRIX[i+100, 5]

#-----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_03[i, 0] = MATRIX[i+150, 0]
    MATRIX_03[i, 1] = MATRIX[i+150, 1]
    MATRIX_03[i, 2] = MATRIX[i+150, 2]
    MATRIX_03[i, 3] = MATRIX[i+150, 3]
    MATRIX_03[i, 4] = MATRIX[i+150, 4]
    MATRIX_03[i, 5] = MATRIX[i+150, 5]

#-----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_04[i, 0] = MATRIX[i+200, 0]
    MATRIX_04[i, 1] = MATRIX[i+200, 1]
    MATRIX_04[i, 2] = MATRIX[i+200, 2]
    MATRIX_04[i, 3] = MATRIX[i+200, 3]
    MATRIX_04[i, 4] = MATRIX[i+200, 4]
    MATRIX_04[i, 5] = MATRIX[i+200, 5]

#-----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_05[i, 0] = MATRIX[i+250, 0]
    MATRIX_05[i, 1] = MATRIX[i+250, 1]
    MATRIX_05[i, 2] = MATRIX[i+250, 2]
    MATRIX_05[i, 3] = MATRIX[i+250, 3]
    MATRIX_05[i, 4] = MATRIX[i+250, 4]
```

```
    MATRIX_05[i, 5] = MATRIX[i+250, 5]

#---------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_06[i, 0] = MATRIX[i+300, 0]
    MATRIX_06[i, 1] = MATRIX[i+300, 1]
    MATRIX_06[i, 2] = MATRIX[i+300, 2]
    MATRIX_06[i, 3] = MATRIX[i+300, 3]
    MATRIX_06[i, 4] = MATRIX[i+300, 4]
    MATRIX_06[i, 5] = MATRIX[i+300, 5]

#---------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_07[i, 0] = MATRIX[i+350, 0]
    MATRIX_07[i, 1] = MATRIX[i+350, 1]
    MATRIX_07[i, 2] = MATRIX[i+350, 2]
    MATRIX_07[i, 3] = MATRIX[i+350, 3]
    MATRIX_07[i, 4] = MATRIX[i+350, 4]
    MATRIX_07[i, 5] = MATRIX[i+350, 5]

#---------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_08[i, 0] = MATRIX[i+400, 0]
    MATRIX_08[i, 1] = MATRIX[i+400, 1]
    MATRIX_08[i, 2] = MATRIX[i+400, 2]
    MATRIX_08[i, 3] = MATRIX[i+400, 3]
    MATRIX_08[i, 4] = MATRIX[i+400, 4]
    MATRIX_08[i, 5] = MATRIX[i+400, 5]

#---------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_09[i, 0] = MATRIX[i+450, 0]
    MATRIX_09[i, 1] = MATRIX[i+450, 1]
    MATRIX_09[i, 2] = MATRIX[i+450, 2]
    MATRIX_09[i, 3] = MATRIX[i+450, 3]
    MATRIX_09[i, 4] = MATRIX[i+450, 4]
    MATRIX_09[i, 5] = MATRIX[i+450, 5]

#---------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_10[i, 0] = MATRIX[i+500, 0]
    MATRIX_10[i, 1] = MATRIX[i+500, 1]
    MATRIX_10[i, 2] = MATRIX[i+500, 2]
    MATRIX_10[i, 3] = MATRIX[i+500, 3]
    MATRIX_10[i, 4] = MATRIX[i+500, 4]
    MATRIX_10[i, 5] = MATRIX[i+500, 5]

#---------------------------------------------------------------------------%

for i in range(0, 50):
```

```
    MATRIX_11[i, 0] = MATRIX[i+550, 0]
    MATRIX_11[i, 1] = MATRIX[i+550, 1]
    MATRIX_11[i, 2] = MATRIX[i+550, 2]
    MATRIX_11[i, 3] = MATRIX[i+550, 3]
    MATRIX_11[i, 4] = MATRIX[i+550, 4]
    MATRIX_11[i, 5] = MATRIX[i+550, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_12[i, 0] = MATRIX[i+600, 0]
    MATRIX_12[i, 1] = MATRIX[i+600, 1]
    MATRIX_12[i, 2] = MATRIX[i+600, 2]
    MATRIX_12[i, 3] = MATRIX[i+600, 3]
    MATRIX_12[i, 4] = MATRIX[i+600, 4]
    MATRIX_12[i, 5] = MATRIX[i+600, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_13[i, 0] = MATRIX[i+650, 0]
    MATRIX_13[i, 1] = MATRIX[i+650, 1]
    MATRIX_13[i, 2] = MATRIX[i+650, 2]
    MATRIX_13[i, 3] = MATRIX[i+650, 3]
    MATRIX_13[i, 4] = MATRIX[i+650, 4]
    MATRIX_13[i, 5] = MATRIX[i+650, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_14[i, 0] = MATRIX[i+700, 0]
    MATRIX_14[i, 1] = MATRIX[i+700, 1]
    MATRIX_14[i, 2] = MATRIX[i+700, 2]
    MATRIX_14[i, 3] = MATRIX[i+700, 3]
    MATRIX_14[i, 4] = MATRIX[i+700, 4]
    MATRIX_14[i, 5] = MATRIX[i+700, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_15[i, 0] = MATRIX[i+750, 0]
    MATRIX_15[i, 1] = MATRIX[i+750, 1]
    MATRIX_15[i, 2] = MATRIX[i+750, 2]
    MATRIX_15[i, 3] = MATRIX[i+750, 3]
    MATRIX_15[i, 4] = MATRIX[i+750, 4]
    MATRIX_15[i, 5] = MATRIX[i+750, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_16[i, 0] = MATRIX[i+800, 0]
    MATRIX_16[i, 1] = MATRIX[i+800, 1]
    MATRIX_16[i, 2] = MATRIX[i+800, 2]
    MATRIX_16[i, 3] = MATRIX[i+800, 3]
```

```python
    MATRIX_16[i, 4] = MATRIX[i+800, 4]
    MATRIX_16[i, 5] = MATRIX[i+800, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_17[i, 0] = MATRIX[i+850, 0]
    MATRIX_17[i, 1] = MATRIX[i+850, 1]
    MATRIX_17[i, 2] = MATRIX[i+850, 2]
    MATRIX_17[i, 3] = MATRIX[i+850, 3]
    MATRIX_17[i, 4] = MATRIX[i+850, 4]
    MATRIX_17[i, 5] = MATRIX[i+850, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_18[i, 0] = MATRIX[i+900, 0]
    MATRIX_18[i, 1] = MATRIX[i+900, 1]
    MATRIX_18[i, 2] = MATRIX[i+900, 2]
    MATRIX_18[i, 3] = MATRIX[i+900, 3]
    MATRIX_18[i, 4] = MATRIX[i+900, 4]
    MATRIX_18[i, 5] = MATRIX[i+900, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_19[i, 0] = MATRIX[i+950, 0]
    MATRIX_19[i, 1] = MATRIX[i+950, 1]
    MATRIX_19[i, 2] = MATRIX[i+950, 2]
    MATRIX_19[i, 3] = MATRIX[i+950, 3]
    MATRIX_19[i, 4] = MATRIX[i+950, 4]
    MATRIX_19[i, 5] = MATRIX[i+950, 5]

#----------------------------------------------------------------------------%

for i in range(0, 50):

    MATRIX_20[i, 0] = MATRIX[i+1000, 0]
    MATRIX_20[i, 1] = MATRIX[i+1000, 1]
    MATRIX_20[i, 2] = MATRIX[i+1000, 2]
    MATRIX_20[i, 3] = MATRIX[i+1000, 3]
    MATRIX_20[i, 4] = MATRIX[i+1000, 4]
    MATRIX_20[i, 5] = MATRIX[i+1000, 5]

#----------------------------------------------------------------------------%

# print(MATRIX_20)
```

In [104…

```python
MATRIX_00_sum = np.zeros((1,3), dtype = float)
MATRIX_01_sum = np.zeros((1,3), dtype = float)
MATRIX_02_sum = np.zeros((1,3), dtype = float)
MATRIX_03_sum = np.zeros((1,3), dtype = float)
MATRIX_04_sum = np.zeros((1,3), dtype = float)
MATRIX_05_sum = np.zeros((1,3), dtype = float)
MATRIX_06_sum = np.zeros((1,3), dtype = float)
MATRIX_07_sum = np.zeros((1,3), dtype = float)
```

```
MATRIX_08_sum = np.zeros((1,3), dtype = float)
MATRIX_09_sum = np.zeros((1,3), dtype = float)
MATRIX_10_sum = np.zeros((1,3), dtype = float)
MATRIX_11_sum = np.zeros((1,3), dtype = float)
MATRIX_12_sum = np.zeros((1,3), dtype = float)
MATRIX_13_sum = np.zeros((1,3), dtype = float)
MATRIX_14_sum = np.zeros((1,3), dtype = float)
MATRIX_15_sum = np.zeros((1,3), dtype = float)
MATRIX_16_sum = np.zeros((1,3), dtype = float)
MATRIX_17_sum = np.zeros((1,3), dtype = float)
MATRIX_18_sum = np.zeros((1,3), dtype = float)
MATRIX_19_sum = np.zeros((1,3), dtype = float)
MATRIX_20_sum = np.zeros((1,3), dtype = float)

MASTER_MATRIX = np.zeros((21,4), dtype = float)
```

In [104…

```
for k in range (0,21):

    MASTER_MATRIX[k,0] = k/10
```

The next several lines sum the columns of the matrices and store them into appropriate matrices.

In [104…

```
## Matrix_00

sum10 = sum(MATRIX_00[:10,5])
sum25 = sum(MATRIX_00[:25,5])
sum50 = sum(MATRIX_00[:,5])

MATRIX_00_sum[0,0] = sum10
MATRIX_00_sum[0,1] = sum25
MATRIX_00_sum[0,2] = sum50

MASTER_MATRIX[0,1] = sum10
MASTER_MATRIX[0,2] = sum25
MASTER_MATRIX[0,3] = sum50

# print(MATRIX_00_sum)
```

In [104…

```
## Matrix_01

sum10 = sum(MATRIX_01[:10,5])
sum25 = sum(MATRIX_01[:25,5])
sum50 = sum(MATRIX_01[:,5])

MATRIX_01_sum[0,0] = sum10
MATRIX_01_sum[0,1] = sum25
MATRIX_01_sum[0,2] = sum50

MASTER_MATRIX[1,1] = sum10
MASTER_MATRIX[1,2] = sum25
MASTER_MATRIX[1,3] = sum50

# print(MATRIX_01_sum)
```

In [104…

```
## Matrix_02

sum10 = sum(MATRIX_02[:10,5])
sum25 = sum(MATRIX_02[:25,5])
```

```
sum50 = sum(MATRIX_02[:,5])

MATRIX_02_sum[0,0] = sum10
MATRIX_02_sum[0,1] = sum25
MATRIX_02_sum[0,2] = sum50

MASTER_MATRIX[2,1] = sum10
MASTER_MATRIX[2,2] = sum25
MASTER_MATRIX[2,3] = sum50

# print(MATRIX_02_sum)
```

In [104…

```
## Matrix_03

sum10 = sum(MATRIX_03[:10,5])
sum25 = sum(MATRIX_03[:25,5])
sum50 = sum(MATRIX_03[:,5])

MATRIX_03_sum[0,0] = sum10
MATRIX_03_sum[0,1] = sum25
MATRIX_03_sum[0,2] = sum50

MASTER_MATRIX[3,1] = sum10
MASTER_MATRIX[3,2] = sum25
MASTER_MATRIX[3,3] = sum50

# print(MATRIX_03_sum)
```

In [104…

```
## Matrix_04

sum10 = sum(MATRIX_04[:10,5])
sum25 = sum(MATRIX_04[:25,5])
sum50 = sum(MATRIX_04[:,5])

MATRIX_04_sum[0,0] = sum10
MATRIX_04_sum[0,1] = sum25
MATRIX_04_sum[0,2] = sum50

MASTER_MATRIX[4,1] = sum10
MASTER_MATRIX[4,2] = sum25
MASTER_MATRIX[4,3] = sum50

# print(MATRIX_04_sum)
```

In [104…

```
## Matrix_05

sum10 = sum(MATRIX_05[:10,5])
sum25 = sum(MATRIX_05[:25,5])
sum50 = sum(MATRIX_05[:,5])

MATRIX_05_sum[0,0] = sum10
MATRIX_05_sum[0,1] = sum25
MATRIX_05_sum[0,2] = sum50

MASTER_MATRIX[5,1] = sum10
MASTER_MATRIX[5,2] = sum25
MASTER_MATRIX[5,3] = sum50
```

```
# print(MATRIX_05_sum)
```

In [104…

```
## Matrix_06

sum10 = sum(MATRIX_06[:10,5])
sum25 = sum(MATRIX_06[:25,5])
sum50 = sum(MATRIX_06[:,5])

MATRIX_06_sum[0,0] = sum10
MATRIX_06_sum[0,1] = sum25
MATRIX_06_sum[0,2] = sum50

MASTER_MATRIX[6,1] = sum10
MASTER_MATRIX[6,2] = sum25
MASTER_MATRIX[6,3] = sum50

# print(MATRIX_06_sum)
```

In [104…

```
## Matrix_07

sum10 = sum(MATRIX_07[:10,5])
sum25 = sum(MATRIX_07[:25,5])
sum50 = sum(MATRIX_07[:,5])

MATRIX_07_sum[0,0] = sum10
MATRIX_07_sum[0,1] = sum25
MATRIX_07_sum[0,2] = sum50

MASTER_MATRIX[7,1] = sum10
MASTER_MATRIX[7,2] = sum25
MASTER_MATRIX[7,3] = sum50

# print(MATRIX_07_sum)
```

In [105…

```
## Matrix_08

sum10 = sum(MATRIX_08[:10,5])
sum25 = sum(MATRIX_08[:25,5])
sum50 = sum(MATRIX_08[:,5])

MATRIX_08_sum[0,0] = sum10
MATRIX_08_sum[0,1] = sum25
MATRIX_08_sum[0,2] = sum50

MASTER_MATRIX[8,1] = sum10
MASTER_MATRIX[8,2] = sum25
MASTER_MATRIX[8,3] = sum50

# print(MATRIX_08_sum)
```

In [105…

```
## Matrix_09

sum10 = sum(MATRIX_09[:10,5])
sum25 = sum(MATRIX_09[:25,5])
sum50 = sum(MATRIX_09[:,5])

MATRIX_09_sum[0,0] = sum10
```

```
MATRIX_09_sum[0,1] = sum25
MATRIX_09_sum[0,2] = sum50

MASTER_MATRIX[9,1] = sum10
MASTER_MATRIX[9,2] = sum25
MASTER_MATRIX[9,3] = sum50

# print(MATRIX_09_sum)
```

In [105…

```
## Matrix_10

sum10 = sum(MATRIX_10[:10,5])
sum25 = sum(MATRIX_10[:25,5])
sum50 = sum(MATRIX_10[:,5])

MATRIX_10_sum[0,0] = sum10
MATRIX_10_sum[0,1] = sum25
MATRIX_10_sum[0,2] = sum50

MASTER_MATRIX[10,1] = sum10
MASTER_MATRIX[10,2] = sum25
MASTER_MATRIX[10,3] = sum50

# print(MATRIX_10_sum)
```

In [105…

```
## Matrix_11

sum10 = sum(MATRIX_11[:10,5])
sum25 = sum(MATRIX_11[:25,5])
sum50 = sum(MATRIX_11[:,5])

MATRIX_11_sum[0,0] = sum10
MATRIX_11_sum[0,1] = sum25
MATRIX_11_sum[0,2] = sum50

MASTER_MATRIX[11,1] = sum10
MASTER_MATRIX[11,2] = sum25
MASTER_MATRIX[11,3] = sum50

# print(MATRIX_11_sum)
```

In [105…

```
## Matrix_12

sum10 = sum(MATRIX_12[:10,5])
sum25 = sum(MATRIX_12[:25,5])
sum50 = sum(MATRIX_12[:,5])

MATRIX_12_sum[0,0] = sum10
MATRIX_12_sum[0,1] = sum25
MATRIX_12_sum[0,2] = sum50

MASTER_MATRIX[12,1] = sum10
MASTER_MATRIX[12,2] = sum25
MASTER_MATRIX[12,3] = sum50

# print(MATRIX_12_sum)
```

In [105…

```
## Matrix_13
```

```
sum10 = sum(MATRIX_13[:10,5])
sum25 = sum(MATRIX_13[:25,5])
sum50 = sum(MATRIX_13[:,5])

MATRIX_13_sum[0,0] = sum10
MATRIX_13_sum[0,1] = sum25
MATRIX_13_sum[0,2] = sum50

MASTER_MATRIX[13,1] = sum10
MASTER_MATRIX[13,2] = sum25
MASTER_MATRIX[13,3] = sum50

# print(MATRIX_13_sum)
```

In [105…

```
## Matrix_14

sum10 = sum(MATRIX_14[:10,5])
sum25 = sum(MATRIX_14[:25,5])
sum50 = sum(MATRIX_14[:,5])

MATRIX_14_sum[0,0] = sum10
MATRIX_14_sum[0,1] = sum25
MATRIX_14_sum[0,2] = sum50

MASTER_MATRIX[14,1] = sum10
MASTER_MATRIX[14,2] = sum25
MASTER_MATRIX[14,3] = sum50

# print(MATRIX_14_sum)
```

In [105…

```
## Matrix_15

sum10 = sum(MATRIX_15[:10,5])
sum25 = sum(MATRIX_15[:25,5])
sum50 = sum(MATRIX_15[:,5])

MATRIX_15_sum[0,0] = sum10
MATRIX_15_sum[0,1] = sum25
MATRIX_15_sum[0,2] = sum50

MASTER_MATRIX[15,1] = sum10
MASTER_MATRIX[15,2] = sum25
MASTER_MATRIX[15,3] = sum50

# print(MATRIX_15_sum)
```

In [105…

```
## Matrix_16

sum10 = sum(MATRIX_16[:10,5])
sum25 = sum(MATRIX_16[:25,5])
sum50 = sum(MATRIX_16[:,5])

MATRIX_16_sum[0,0] = sum10
MATRIX_16_sum[0,1] = sum25
MATRIX_16_sum[0,2] = sum50

MASTER_MATRIX[16,1] = sum10
```

```python
MASTER_MATRIX[16,2] = sum25
MASTER_MATRIX[16,3] = sum50

# print(MATRIX_16_sum)
```

In [105…

```python
## Matrix_17

sum10 = sum(MATRIX_17[:10,5])
sum25 = sum(MATRIX_17[:25,5])
sum50 = sum(MATRIX_17[:,5])

MATRIX_17_sum[0,0] = sum10
MATRIX_17_sum[0,1] = sum25
MATRIX_17_sum[0,2] = sum50

MASTER_MATRIX[17,1] = sum10
MASTER_MATRIX[17,2] = sum25
MASTER_MATRIX[17,3] = sum50

# print(MATRIX_17_sum)
```

In [106…

```python
## Matrix_18

sum10 = sum(MATRIX_18[:10,5])
sum25 = sum(MATRIX_18[:25,5])
sum50 = sum(MATRIX_18[:,5])

MATRIX_18_sum[0,0] = sum10
MATRIX_18_sum[0,1] = sum25
MATRIX_18_sum[0,2] = sum50

MASTER_MATRIX[18,1] = sum10
MASTER_MATRIX[18,2] = sum25
MASTER_MATRIX[18,3] = sum50

# print(MATRIX_18_sum)
```

In [106…

```python
## Matrix_19

sum10 = sum(MATRIX_19[:10,5])
sum25 = sum(MATRIX_19[:25,5])
sum50 = sum(MATRIX_19[:,5])

MATRIX_19_sum[0,0] = sum10
MATRIX_19_sum[0,1] = sum25
MATRIX_19_sum[0,2] = sum50

MASTER_MATRIX[19,1] = sum10
MASTER_MATRIX[19,2] = sum25
MASTER_MATRIX[19,3] = sum50

# print(MATRIX_19_sum)
```

In [106…

```python
## Matrix_20

sum10 = sum(MATRIX_20[:10,5])
sum25 = sum(MATRIX_20[:25,5])
sum50 = sum(MATRIX_20[:,5])
```

```
MATRIX_20_sum[0,0] = sum10
MATRIX_20_sum[0,1] = sum25
MATRIX_20_sum[0,2] = sum50

MASTER_MATRIX[20,1] = sum10
MASTER_MATRIX[20,2] = sum25
MASTER_MATRIX[20,3] = sum50

# print(MATRIX_20_sum)
```

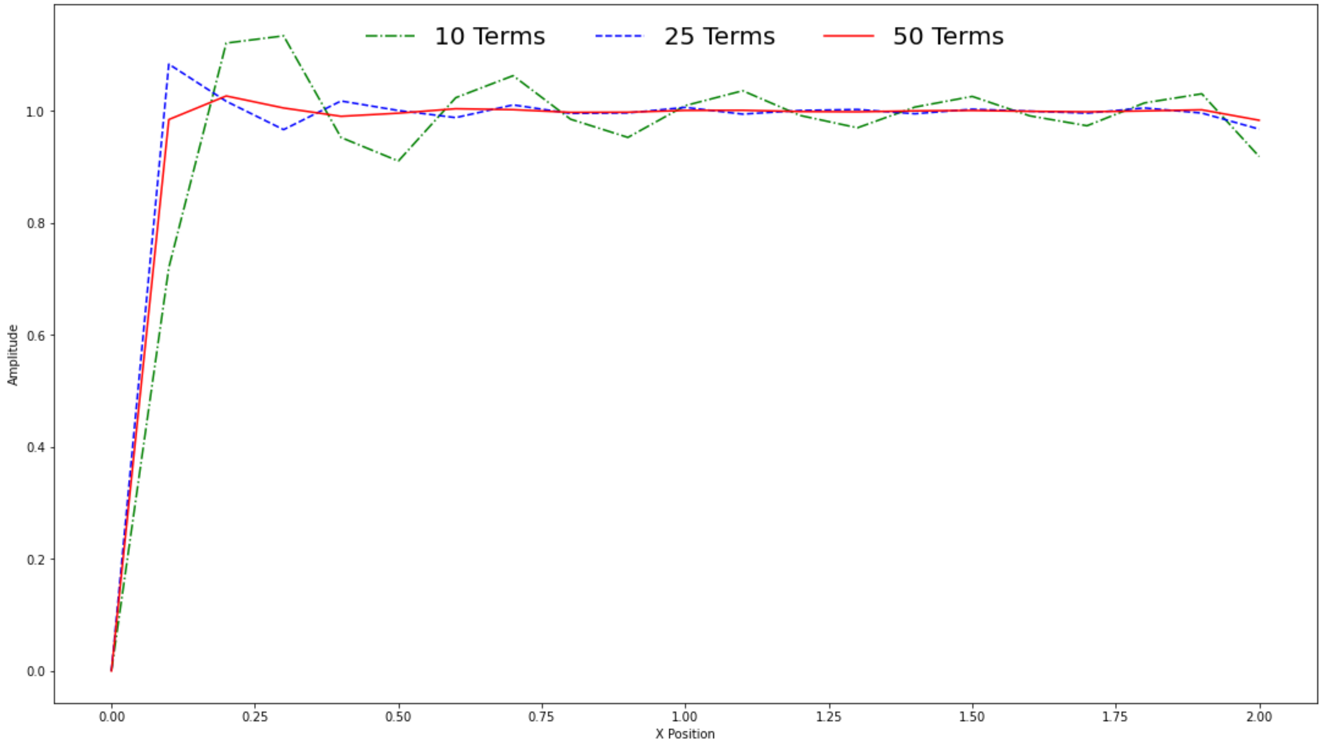
In [109…

```
# 10 terms

# print(MASTER_MATRIX)
```

Plotting the summed columns from the matrices (based on number of values used to sum).

In [109…

```
import matplotlib.pyplot as plt
import pylab as plot
params = {'legend.fontsize': 20,'legend.handlelength': 2}
plot.rcParams.update(params)

fig = plt.gcf()
fig.set_size_inches(18.5, 10.5)

plt.plot(MASTER_MATRIX[:,0], MASTER_MATRIX[:,1], '-.g', label='10 Terms')
plt.plot(MASTER_MATRIX[:,0], MASTER_MATRIX[:,2], '--b', label='25 Terms')
plt.plot(MASTER_MATRIX[:,0], MASTER_MATRIX[:,3], '-r', label='50 Terms')

plt.xlabel('X Position')
plt.ylabel('Amplitude')

plt.legend(frameon=False, loc='upper center', ncol=3)

plt.show()
```



Conclusions

As you can see, the more terms used to sum the expansion of the eigenfunction, the more accurate the solution will converge to, as shown in the plot above.