MS6021 Scientific Computation

Assignment 4

Dara Corr - 22275193

November 13, 2022

Contents

1 Introduction

In this assignment we will be examining Quadrature and the Calculation of Eigenvalues in MATLAB.

The first section of this assignment is concerned with quadrature which is the name given to Numerical Integration. We will be investigating the performance of the Trapezoidal rule and MATLAB's built in quad function in this assignment. We will also construct a tailored version of the Trapezoidal rule to calculate Integrals with an integrand of the form $x^{\alpha}g(x)$, and compare the performance of this integral to the other 2 numerical integration techniques we will investigate.

In the Second Section of the assignment, we will look at computing Eigenvalues of the differential operator $Lv = -v''$ subject to v(0) = v(1) = 0 Numerically and comparing the numerical eigenvalues to the analytical result. Then we will calculate Eigenvalues for an operator, whose eigenvalues cannot be computed analytically and compare the results to a reference solution we produce.

2 Quadrature

We consider the following Integral in the first part of the assignment:

$$I(\alpha) = \int_0^1 x^\alpha (1 + x^2) dx, \text{ for } \alpha > -1 \tag{1}$$

first, we integrate and evaluate this integral analytically.
Analytical solution:

$$I(\alpha) = \int_0^1 x^\alpha (1 + x^2) dx, \text{ for } \alpha > -1$$

$$\int_0^1 x^\alpha (1 + x^2) dx = \int_0^1 x^\alpha + x^{\alpha+2}$$

$$= \left(\frac{x^{\alpha+1}}{\alpha + 1} + \frac{x^{\alpha+3}}{\alpha + 3} \right) \Big|_{x=0}^{x=1}$$

$$\int_0^1 x^\alpha (1 + x^2) dx = \frac{1}{\alpha + 1} + \frac{1}{\alpha + 3}$$

And then I created a function in MATLAB which calculates an area under a curve f(x) from lower bound a to upper bound b, using the Trapezoidal rule.

The Trapezoidal calculates the area under a curve f(x) by splitting the area under the curve into trapezoids of width h ($h = x_i - x_{i-1} = \frac{b-a}{N}$) with the height at $x = x_i$, $= y(x_i) = y_i$ and the height of the trapezoid at $x = x_{i-1}$, $= y(x_{i-1}) = y_{i-1}$.

This trapezoid can be visualised as a rectangle with a triangular section on top of it. The area of this trapezoid can be written as:

$$\begin{aligned} \text{Trapezoid Area } &= \text{Area of Square } + \text{Area of Triangle} \\ &= (h \cdot y_i) + \frac{1}{2} h (y_{i-1} - y_i) \\ &= \frac{h}{2} (y_i + y_{i-1}) \end{aligned}$$

Then the total Area under the curve (the integrand) is the sum of the areas of all the Trapezoids:

$$\text{Area } = \frac{h}{2}(y_0 + y_1) + \frac{h}{2}(y_1 + y_2) + ... + \frac{h}{2}(y_{n-1} + y_n)$$

$$\text{Area } = \frac{h}{2}[y_0 + 2(y_1 + y_2 + ... + y_{n-1}) + y_n]$$

Here is the code I used to implement the Trapezoidal Rule into MATLAB:

```
function result = trapezoidal(f,a,b,N) %trapezoidal integration for fn of 1
    variable
h = (b-a)/N;
values = [a:h:b];
values(1) = []; values(end) = [];
middle = sum(2.*f(values));
result = (h/2 .* (f(a) + middle + f(b)));
end
```

I also used MATLAB's built in quad function to evaluate the integral using an adapted version of Simpson's Rule so I can compare the precision of that method compared to the Trapezoidal Integal.

Then I computed Integral (1) above for $\alpha = 2.2$, 0.5 and -0.9 using trapezoidal and quad functions and created a table with the error values $|I - I^{num}|$ between the Numerical Methods and the Analytical result.

Table of Errors $|I - I^{num}|$:

	$\alpha = 2.2$	$\alpha = 0.5$	α = -0.9
Trapezoidal (N=10)	5.3322e-03	4.0716e-03	INF
Trapezoidal (N=100)	5.3335e-05	1.8289e-04	INF
Trapezoidal (N=10000)	5.3333e-09	2.0539e-07	INF
quad (tolerence = 10^{-2})	8.2852e-07	1.2636e-03	1.8398e-01
quad (tolerence = 10^{-4})	8.2852e-07	4.4794e-04	2.5066e-01
quad (tolerence = 10^{-8})	8.30561e-09	4.0653e-08	1.0205e-01

Then I considered a more general form of the integral

$$I_g = \int_0^1 x^\alpha g(x)dx \text{ for } \alpha > -1 \tag{2}$$

To solve this integral numerically, I created a new MATLAB function called *tailored* which evaluates this integral numerically using a tailored version of the Trapezoidal Rule.
The Tailored Integral has the following form:

$$I_g \approx \sum_i \int_{x_{i-1}}^{x_i} x^\alpha g^I(x)dx \tag{3}$$

here $g^I(x)$ is the Linear Interpolant of $g(x)$ on the interval (x_{i-1}, x_i)
We can define the linear interpolant on the interval (x_{i-1}, x_i) as:

$$\frac{y - y_{i-1}}{x - x_{i-1}} = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}$$

and if we solve for y:

$$y = y_{i-1} + (x - x_{i-1})\frac{y_i - y_{i-1}}{x_i - x_{i-1}}$$

$$\vdots$$

$$\text{writing y as g(x): } y^I = g^I(x) = \frac{g(x_{i-1})(x_i - x) + g(x_i)(x - x_{i-1})}{x_i - x_{i-1}}$$

Then,

$$I_g \approx \sum_i \int_{x_{i-1}}^{x_i} x^\alpha g^I(x)dx = \sum_i \int_{x_{i-1}}^{x_i} x^\alpha \frac{g(x_{i-1})(x_i - x) + g(x_i)(x - x_{i-1})}{x_i - x_{i-1}}dx$$

$$= \sum_i \int_{x_{i-1}}^{x_i} x^\alpha \frac{g(x_{i-1})(x_i) - x \cdot g(x_{i-1}) + x \cdot g(x_i) - x_{i-1}g(x_i)}{x_i - x_{i-1}}dx$$

$$= \sum_i \int_{x_{i-1}}^{x_i} x^\alpha \left[\frac{(x_i)g(x_{i-1}) - (x_{i-1})g(x_i)}{x_i - x_{i-1}} + \frac{g(x_i)x - g(x_{i-1})x}{x_i - x_{i-1}}\right]dx$$

$$= \sum_i \left[\int_{x_{i-1}}^{x_i} x^{\alpha+1} \frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}}dx + \int_{x_{i-1}}^{x_i} x^\alpha \frac{(x_i)g(x_{i-1}) - (x_{i-1})g(x_i)}{x_i - x_{i-1}}dx\right]$$

$$= \sum_i \left[\frac{(x_i{}^{\alpha+2} - x_{i-1}{}^{\alpha+2})}{\alpha+2}\left(\frac{g(x_i) - g(x_{i-1})}{x_i - x_{i-1}}\right) + \frac{(x_i{}^{\alpha+1} - x_{i-1}{}^{\alpha+1})}{\alpha+1}\left(\frac{x_i \cdot g(x_{i-1}) - x_{i-1} \cdot g(x_i)}{x_i - x_{i-1}}\right)\right]$$

I implemented the above result into the function *tailored* in Matlab as follows:

```
function result = tailored(g,alpha,N) %trapezoidal integration for fn of  1
    variable
start = 0;
finish = 1;

h = (finish-start)/N;

x_i = [(start+h):h:finish];
x_i_minus_one = [start:h:(finish-h)];

vals = @(x) (g(x_i) - g(x_i_minus_one))./(x_i - x_i_minus_one) .* 1/(alpha +
    2) .* (x.^(alpha + 2)) + (x_i.*g(x_i_minus_one) -(x_i_minus_one).*(g(x_i))
     )./(x_i - x_i_minus_one) .* 1/(alpha + 1) .* (x.^(alpha + 1))

result = sum(vals(x_i) - vals(x_i_minus_one))

end
```

I then Used the Tailored function to calculate the integral and I computed the errors compared to the analytical solution $|I - I^{num}|$ as before and added the results to the table.

	$\alpha = 2.2$	$\alpha = 0.5$	α = -0.9
Trapezoidal (N=10)	5.3322e-03	4.0716e-03	INF
Trapezoidal (N=100)	5.3335e-05	1.8289e-04	INF
Trapezoidal (N=10000)	5.3333e-09	2.0539e-07	INF
quad (tolerence = 10^{-2})	8.2852e-07	1.2636e-03	1.8398e-01
quad (tolerence = 10^{-4})	8.2852e-07	4.4794e-04	2.5066e-01
quad (tolerence = 10^{-8})	8.30561e-09	4.0653e-08	1.0205e-01
Tailored (N=10)	5.2022e-04	1.1124e-03	6.8488e-03
Tailored (N=100)	5.2083e-06	1.1112e-05	8.8679e-05
Tailored (N=10000)	5.2077e-10	1.1110e-09	1.1746e-08

Then I applied the Tailored function to the integral

$$I_g = \int_0^1 x^\alpha e^{1-2x^3}$$

Where $\alpha = -0.9$ and $g(x) = exp(1 - 2x^3)$

and I found the resulting areas for multiple values of N

N	tailored($exp(1 - 2x^3)$), $\alpha = -0.9$
N=10	26.02459368783
N=100	26.03438824788
N=1000	26.03448071414
N=10000	26.03448163415
N=100000	26.03448164334

2.1 Discussion and Conclusions

I plotted the three plots for $f(x) = x^\alpha(1 + x^2), \alpha = (2.2, 0.5, -0.9)$ to gain insight into how the three integration methods performed.

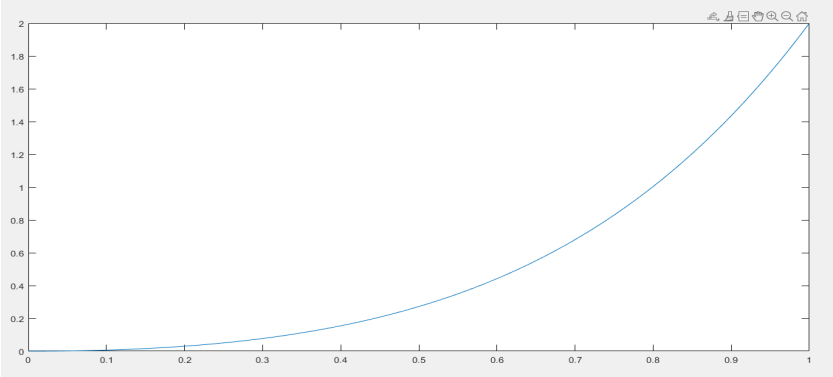

Figure 1: Plot of $f(x) = x^\alpha(1 + x^2)$ for $\alpha = 2.2$

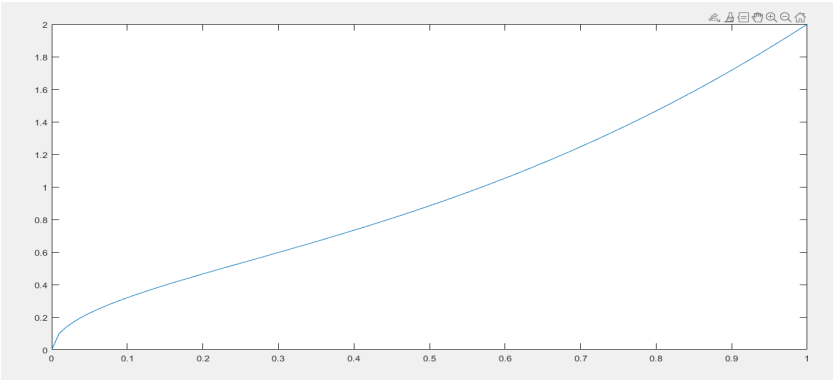

Figure 2: Plot of $f(x) = x^{\alpha}(1 + x^2)$ for $\alpha = 0.5$

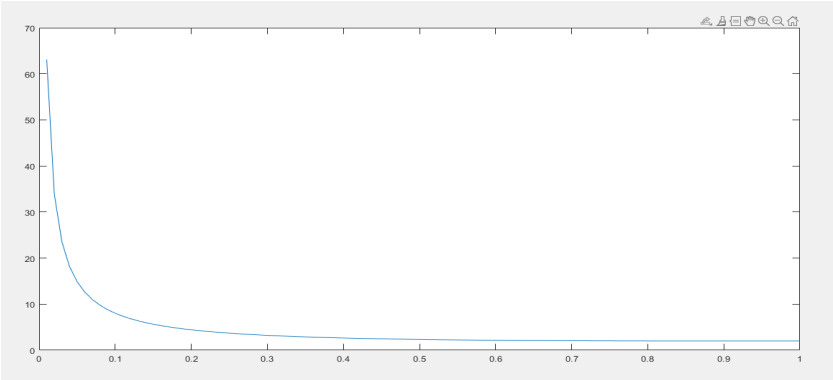

Figure 3: Plot of $f(x) = x^{\alpha}(1 + x^2)$ for $\alpha = -0.9$

From the tables, we see that for $\alpha = 2.2$, the quad function performs better than the Trapezoidal function and quad also performs better than Tailored for small N, but Tailored performs best for large N. All functions perform well for large N or large tolerence.

For $\alpha = 0.5$, Tailored performs the best overall with the lowest error values. quad performs next best and Trapezoidal gives the largest errors for $\alpha = 0.5$. I believe the reason for the change in the errors compared to when $\alpha = 2.2$, is the section of the graph where $x < 0.5$ and there is a relatively large jump in y for a small change in x. The size of the errors here is indicative of how well these integration methods deal with functions that have a large change in y for an interval where there is a relatively small change in x. The use of the Interpolant of g(x) in Tailored, deals with this issue the best here.

Trapezoidal Rule tends to underestimate when a curve is concave down and the opposite occurs for curves that are concave up, Trapezoidal Rule will overestimate the area. Simpson's is more accurate than Trapezoidal because it matches a parabola to a curve rather than a straight line like Trapezoidal rule essentially does. Simpson's Rule (quad uses adapted version of Simpson's Rule) is proportional to the 4th derivative of the function that we are

integrating, at some point in the interval, so it tends to be quite precise. The result using Simpson's rule is exact when the integrand is a linear or quadratic function.

For $\alpha = -0.9$, the Trapezoidal function gives INF values for the error as it evaulates f(0) = +INF and this means the result here using Trapezoidal is an INF value. This is because of a vertical asymptote of $+\infty$ at x = 0, as we can see from the plot. The max errors given by the quad function are quite high and are given to 1 decimal place which is not very good for applications where precision is desirable. This is probably an issue with poor fit of the Lagrange Polynomial Interpolant used in Simpson's rule with the asymptotic curve we have in the plot in figure 3.

The Tailored function is able to overcome the problem with the asymptote and give reasonably small error values since it avoids calculating $y_0 = y(x_0) = (\frac{1}{x^{0.9}} + x^{1.1})\big|_{x=0}$ which gives some values that tend to ∞ using Trapezoidal rule. Instead, using the Linear Interpolant, it evaulates $g(x_i) = 1 + {x_i}^2$ which is quite straightforward to calculate instead.

When I evaluated the integral $I_g = \int_0^1 x^\alpha e^{1-2x^3}$ where $\alpha = -0.9$ using *tailored*, for a range of N values of different magnitudes, I found that the Tailored function gives a result that tends towards the analytical result as $N \to \infty$. WolframAlpha gives the result = 26.0345 which is equivalent to the result obtained from *tailored* using N = 1000.

In conclusion, This tailored method I created generates very precise results for our problem and often outperforms Simpson's Rule.

3 Eigenvalues

Now we will calculate Eigenvalues Numerically and Analytically for differential operators and compare the results:

We will calculate eigenvalues for the differential operator L $=-\frac{d^2}{dx^2}$ i.e. $Lv = -v''$ with boundary conditions $v(0) = v(1) = 0$.

We recognise that this differential Operator L is equivalent to the Matrix A we used in Assignment 1 when it is discretized. We must make one minor change here, because in Assignment 1 part I, we had $-u'' + 4u = exp(x) \rightarrow -u'' + a(x)u = f(x)$. But here we can set $a(x) = 0$ and we are uninterested in f(x), we are only interested in the differential operator applied on u (or now in our case, v).

We can use A (with a(x) = 0) from Part I because AU describes $-\frac{d^2u}{dx^2}$ discretized using the centred finite difference method. so thus A is the same as our differential Operator L

$$L = A = \begin{bmatrix} 2N^2 & -N^2 & 0 & 0 \\ -N^2 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & -N^2 \\ 0 & 0 & -N^2 & 2N^2 \end{bmatrix}$$

Where A is a $(N-1) \times (N-1)$ Symmetric Matrix

For this case, it is possible to calculate the eigenvalues analytically, and once we have calculated the analytical eigenvalues, we can then compute the Numerical eigenvalues and make comparisons between them.

We seek to find Eigenvalues λ for $v(x) \neq 0$ such that $Lv(x) = \lambda v(x)$ is satisfied subject to boundary conditions.
I.e. We are trying to find the values of λ that satisfy the following matrix equation:

$$\begin{bmatrix} 2N^2 & -N^2 & 0 & 0 \\ -N^2 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & -N^2 \\ 0 & 0 & -N^2 & 2N^2 \end{bmatrix} \begin{bmatrix} v_2 \\ \vdots \\ \vdots \\ v_N \end{bmatrix} = \lambda \begin{bmatrix} v_2 \\ \vdots \\ \vdots \\ v_N \end{bmatrix}$$

We must consider 3 cases here:

- $\lambda > 0$
- $\lambda = 0$
- $\lambda < 0$

case (1) $\lambda > 0$:

$$Lv = \lambda v$$
$$\text{Take } +\lambda = \kappa^2$$
$$-\frac{d^2}{dx^2}v = \kappa^2 v$$
$$\frac{d^2}{dx^2}v + \kappa^2 v = 0$$

Notice how the equation now has the form of the Simple Harmonic (SHM) Equation.

General solution of SHM equation $= Acos(\kappa x) + Bsin(\kappa x)$ for constants A and B.

$$v(0) = 0 \rightarrow A = 0$$
$$v = Bsin(\kappa x)$$
$$v(1) = 0 \rightarrow Bsin(\kappa) = 0$$
$$B = 0 \text{ gives trivial solution}$$
$$\text{We want } sin(\kappa) = 0$$
$$\kappa = n\pi \rightarrow sin(n\pi) = 0 \text{ for natural number n= 1,2,3...}$$
$$\lambda = \kappa^2$$
$$\lambda = n^2\pi^2$$

case (2) $\lambda = 0$:

$$Lv = \lambda v$$
$$-\frac{d^2}{dx^2}v = 0 \cdot v$$
$$\frac{d^2}{dx^2}v = 0$$
$$v = Ax + B \text{ Where A and B are unknown constants}$$
$$v(0) = B = 0$$
$$v(1) = A = 0$$
$$v = 0, \lambda = 0$$
$$\text{trivial case, we ignore it}$$

case (3) $\lambda < 0$:

$$Lv = \lambda v$$
$$\text{Take } -\lambda = +\kappa^2$$
$$-\frac{d^2}{dx^2}v = -\kappa^2 v$$
$$\frac{d^2}{dx^2}v - \kappa^2 v = 0$$

Equation now has the form of the Hyperbolic SHM Equation.

General solution of Hyperbolic SHM equation = $Acosh(\kappa x) + Bsinh(\kappa x)$ for constants A and B.

$$v(0) = 0 \rightarrow A = 0$$
$$v = Bsinh(\kappa x)$$
$$v(1) = 0 \rightarrow Bsin(\kappa) = 0$$
$$B = 0 \text{ gives trivial solution}$$
$$\text{We want } sinh(\kappa) = 0$$
$$\kappa = 0 \text{ not a possible value for } \kappa \text{ since } \lambda < 0, \kappa \text{ must} \neq 0$$
$$\kappa = in\pi \rightarrow \text{this satisfies } sinh(\kappa) = 0$$
$$-\lambda = +\kappa^2$$
$$-\lambda = -n^2\pi^2$$
$$\lambda = n^2\pi^2$$

So therefore the Analytical Eigenvalues for L are $n^2\pi^2$ for n = 1,2,...$+\infty$

3.1 Computing Eigenvalues Numerically

We can use MATLAB's *eig* and *eigs* functions to find the Eigenvalues of our differential operator L.

Here are 3 plots of the Numerical Eigenvalues plotted against the Analytical ones for N = 40, N = 80 and N = 160

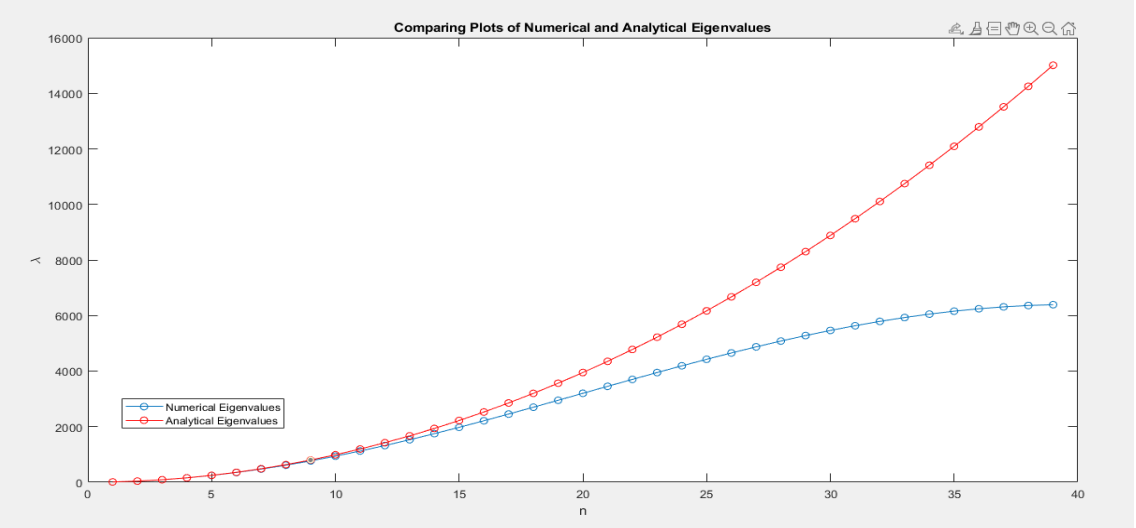

Figure 4: Plots of Numerical and Analytical Eigenvalues for N = 40

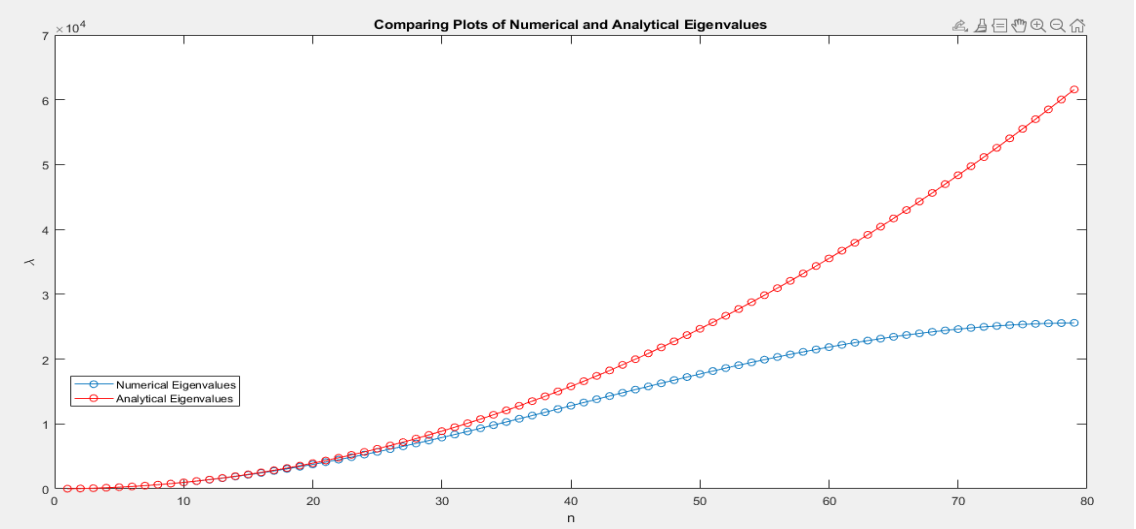

Figure 5: Plots of Numerical and Analytical Eigenvalues for N = 80

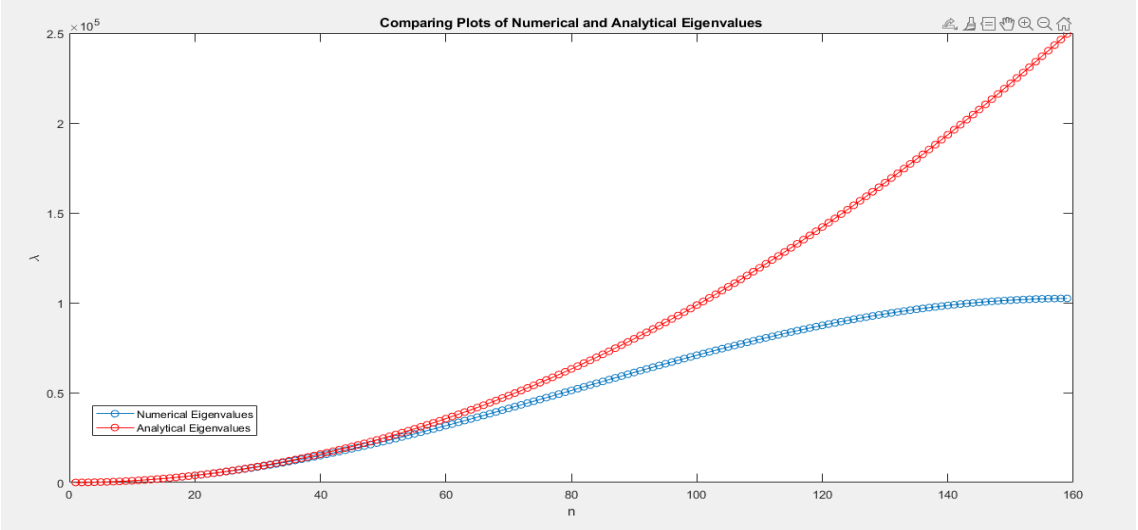

Figure 6: Plots of Numerical and Analytical Eigenvalues for N = 160

```
%Q1 -> take N = 40,80,160
N=40

x = linspace(0,1,N+1)

%Matrix A from assign 1 for equation: -u'' = f
b = -(N^2)*ones(N-1,1);
c = (2*N^2 + 0)*ones(N-1,1);
A = spdiags([b c b], [1 0 -1], N-1, N-1); %
A;

lambdas = eig(A) %numerical lambda values corresponding to theoretical values
    for lambda >0

n = 1:N-1;

analytic_lam = (n.^2 .*pi.^2) %lambda >0 as in q1
```

From the plots, it appears that the Numerical Eigenvalues start to diverge greatly from the Analytical Eigenvalues for values of N $>$ 30. However when N is increased, it appears that the Numerical Eigenvalues appear to converge towards the analytical eigenvalues for the first 30-40 eigenvalues.

I then plotted the first 30 smallest Numerical Eigenvalues against the first 30 Analytical Eigenvalues for increasing values of N to see if the Numerical Eigenvalues become more precise as N increases.
And then I computed the error between the 30th smallest Numerical Eigenvalue and the 30th Analytical Eigenvalue for different values of N:

```
%plot and compare the 30 smallest eigenvalues

```

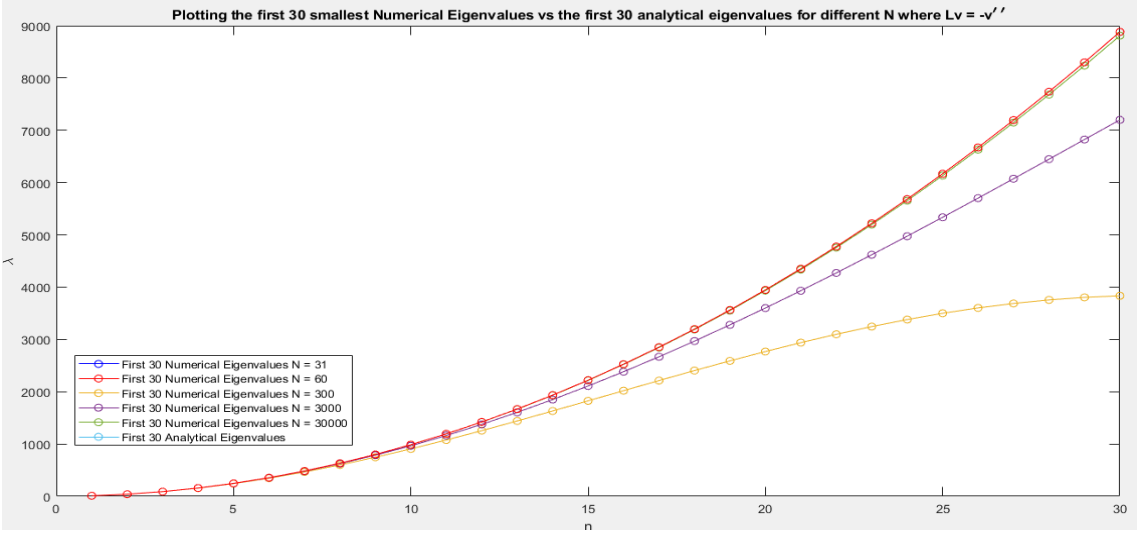

Figure 7: Plots of first 30 smallest Numerical Eigenvalues against first 30 analytical eigenvalues

```
N = 30000 %interested in N = 31,60,300,3000,30000

x = linspace(0,1,N+1)

b = -(N^2)*ones(N-1,1);
c = (2*N^2 + 0)*ones(N-1,1);
A = spdiags([b c b], [1 0 -1], N-1, N-1); %
A;

eigenv_numeric_30 = eigs(A,30, 'smallestabs')

n = 1:30
eigenv_analytical_30 = [(n.^2 .*pi.^2)]

plot(n,eigenv_numeric_30,'o-','color','blue')
hold on
plot(n,eigenv_analytical_30,'o-','color','red')
legend("First 30 smallest Numerical Eigenvalues", "First 30 Analytical
    Eigenvalues",'Location','southwest')
title('Plotting the first 30 smallest Numerical Eigenvalues vs the first 30
    analytical eigenvalues')
xlabel('n')
ylabel('\lambda')

maxerror = abs(eigenv_analytical_30(30)-eigenv_numeric_30(30))./
    eigenv_analytical_30(30)
```

N	$\lvert\lambda_{30} - \lambda_{30}^{num}\rvert/\lambda_{30}$
N=31	0.5684
N=60	0.1894
N=300	8.1977e-03
N=3000	8.2244e-05
N=30000	8.2247e-07

From these results, we see that the Numerical Eigenvalues tend towards the analytical eigenvalues as N is increased towards $+\infty$.

We also consider a different operator

$$Lv = -v'' + a(x)v$$

where $a(x)$ is from assignment 1 part II

Unlike the previous operator, we are unable to calculate the Eigenvalues of this operator Analytically.

This Operator has the following matrix form:

$$L = A = \begin{bmatrix} 2N^2 + a(x_2) & -N^2 & 0 & 0 \\ -N^2 & \ddots & -N^2 & 0 \\ 0 & -N^2 & \ddots & -N^2 \\ 0 & 0 & -N^2 & 2N^2 + a(x_N) \end{bmatrix}$$

$$= \begin{bmatrix} 2N^2 + exp(x_2^3 - x_2) & -N^2 & 0 & 0 \\ -N^2 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & -N^2 \\ 0 & 0 & -N^2 & 2N^2 + exp(x_N^3 - x_N) \end{bmatrix}$$

Now I will plot the first 30 smallest Eigenvalues for this operator for different values of N, like I did with the previous operator except now there is no set of Analytical Eigenvalues to compare the Numerical Eigenvalues with.

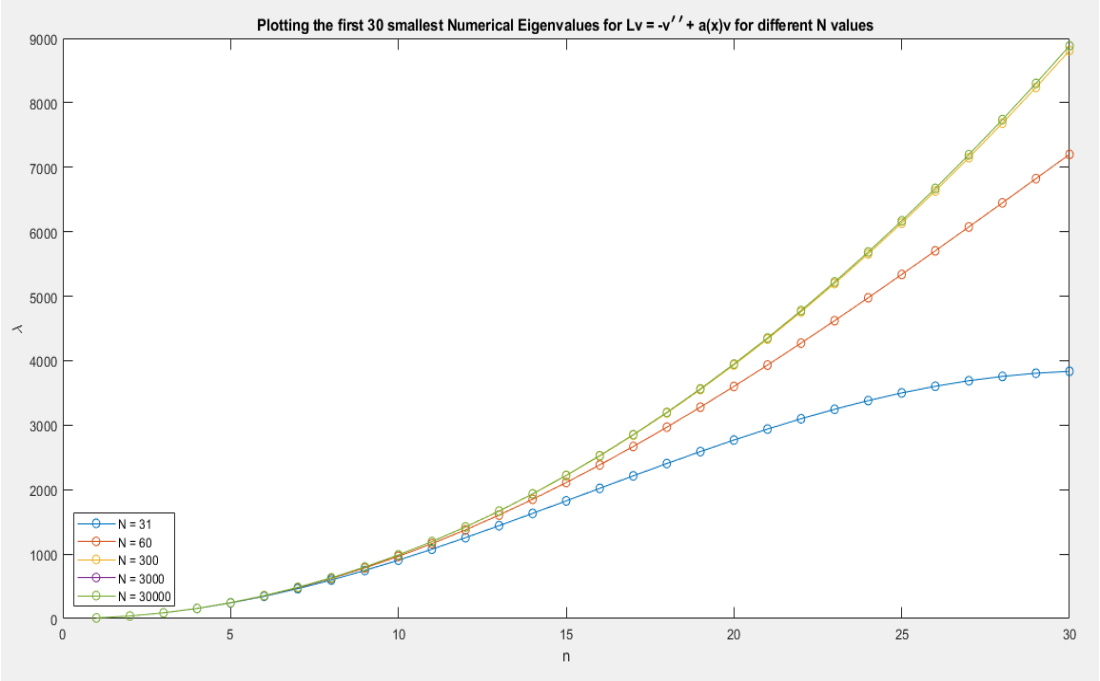

Figure 8: Plots of first 30 smallest Numerical Eigenvalues for the second differential operator for different N values

Then I created a reference solution with N = 300000 and found errors of the first 30 eigenvalues for different N values compared to this reference solution since we do not have an analytical solution to compare with in this case.

N	$\lvert\lambda_{30}^{ref} - \lambda_{30}^{num}\rvert/\lambda_{30}^{ref}$
N=31	0.56831
N=60	0.18941
N=300	8.1969e-03
N=3000	8.2228e-05
N=30000	8.1385e-07

These results are very similar to what we saw for our first operator. I can see from this data and the observations made from the plots that as N tends to $+\infty$ the Numerical Eigenvalues converge towards the Analytical Eigenvalues with very high precision. The error values reduce quickly as N is increased.

We also see here that this is a good method to determine eigenvalues for a problem where Analytical Eigenvalues cannot be determined and we have shown that for large N, the Numerical Values of these eigenvalues are very precise.

3.2 Conclusions

There is a large difference between the Numerical Eigenvalues and the Analytical Eigenvalues as seen in the plots I made. This is because we have discretized the problem using the finite difference method, so we are finding the eigenvalues of the discrete version of L. But however, as we increase the value of N, the first 30 Numerical Eigenvalues tend towards the value of the Analyical ones. This process can be used to get the first few numerical eigenvalues to a good precision, for problems where the analytical eigenvalues cannot easily be computed.