Mini Project Computational Physics :
Ordinary Differential Equations (ODEs) by Finite Difference method

Ahmad Aiman B. Mohd Nazir
17060060

1 Physics Problem

The physics problem that I'm consider here is related to the spring specifically related to the forced vibration problem. The motion of spring generally can be described from Newton's Second Law of Motion such as $F = ma$. The equation that described the motion of this spring can be written as follow;

$$my'' + by' + ky = f(x) \tag{1}$$

Where
m : mass object that attached to the spring
b : damping force
k : spring constant
$f(x)$: forced that being act on the spring

I obtained this particular problem on webpage (math.libretexts.org, 2021), this forced vibration problem comes with the spring hanging from a support and it began set in a motion by the external force on the system. We could model this system as nonhomogeneous differential equation and it can be written as shown in Equation 1. The question of this problem was written as follow:

> *A mass of 1 slug stretches a spring 2 ft and comes at rest at equilibrium. The system is attached to a dashpot that imparts a damping force equal to 8 times the instantaneous velocity of the mass. The external force equal to* $f(x) = 8\ sin(4x)$ *is applied to the system at time x=0. Find the equation of motion.*

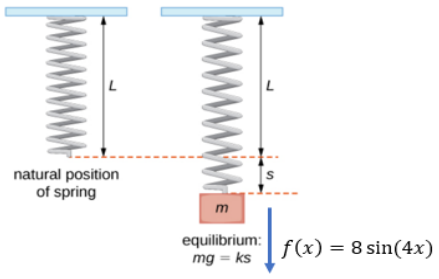


Figure 1: Diagram shows the position of spring for the question that mention above

2 Analytical Solution

By using Equation 1, we then can construct our ODEs from the given question. Our ODEs can be written as follow

$$my'' + by' + ky = f(x) \tag{2}$$
$$y'' + 8y' + 16y = 8\ sin(4x) \tag{3}$$

Where in the given question, the value of mass, $m = 1$, damped force, $b = 8$ and spring constant, $k = 16$. From now, we can obtain the exact solution as

$$y(x) = \frac{1}{4}e^{-4x} + xe^{-4x} - \frac{1}{4}cos(4x) \tag{4}$$

Using Desmos graphic calculator, we can obtain the exact plot from Eyquation 4.

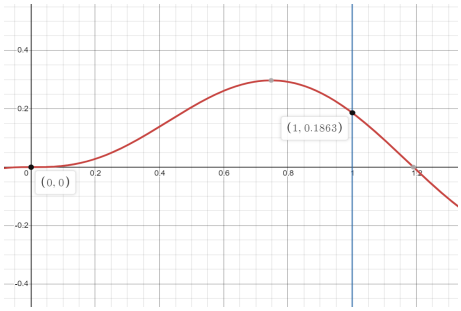


Figure 2: The exact plot that was obtain from Desmos calculator. In this figure, we only specify the boundary between $x = 0$ and $x = 1$, where it will be used in our numerical calculation and for Finite Difference method, we need to specify our boundary which are referred to the starting point and end point

3 Numerical Solution

In order to solve this problem numerically, I have applied Finite Difference method. The program to solve this problem was written in Python Language. The general form for ODEs which I applied in my coding can be written as follow

$$x'' - p(x)x' - q(x)x = f(x) \tag{5}$$

and we can compare Equation 5 with the Equation 3 to obtain their respective values/functions. Then, we can obtain their respective values as follow; $p(x) = -8$, $q(x) = -16$ and $f(x) = 8\ sin(4x)$. The Finite Difference method is a method where we transform our ODEs into tridiagonal matrix ($Ax = B$), A and B is a matrix and x is referred to our solution. The reason why we transform our ODEs in tridiagonal matrix, so that it can be solve numerically. Below is the coding that I write to solve this ODEs. You can also run this code by clicking **here**.

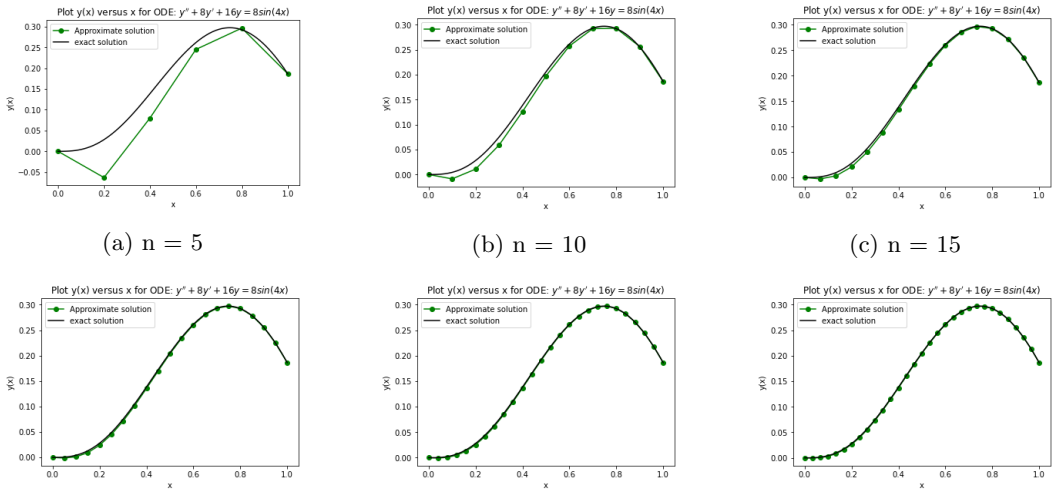
```
#Author: Ahmad Aiman B. Mohd Nazir (17060060)
from scipy.linalg import solve  #import the relevent libraries
import  numpy as np
import matplotlib.pyplot as plt
x = [0]       #set the initial value of x and define all parameters
f = []        #f(x)
n = 10        #number of iterations for n = 5,10,15,20,25,30 or mesh points
h = 1/n       #step size or increment
p = -8        #p(x)
q = -16       #q(x)
u = 0         #y(0)
u_1 = 0.1863   #y(1.0)

#define matrix zeros with size (n-1)x(n-1)
A = np.zeros((n-1,n-1))

#define matrix zeros with size (n-1)x(1)
B = np.zeros((n-1,1))
```

```
19
20 #append x values into array x by using loop
21 for i in range (0,n):
22     x.append(x[i]+h)
23     i+=1
24 #calculate the values of f(x) for each mesh points
25 for i in range (0,n):
26   f.append(8*np.sin(4*x[i]))
27   i+=1
28 #calculate the coefficient for tridiagonal matrix
29 a = 2+ q*h**2
30 b = -(1-1/2*p*h)
31 c = -(1+1/2*p*h)
32 #input the values obtain from a,b and c into matrix A
33 for i in range(0,n-1):
34     for j in range(0,n-1):
35         if i == j:
36             A[i][j] = a
37         elif i<j and j-i==1:
38             A[i][j] = b
39         elif i>j and i-j==1:
40             A[i][j] = c
41         else:
42             A[i][j] = 0
43     j+=1
44 i+=1
45 #input the values into matrix B
46 for i in range(0,n-1):
47     for j in range(0,1):
48         if i == j:
49             B[i][j] = (1+(1/2)*p*h)*u - h**2*f[i+1]
50         elif i > j and i<= n-3:
51             B[i][j] = -(h**2)*f[i+1]
52         elif i > j and i == n-2:
53             B[i][j] = (1-(1/2)*p*h)*u_1 - (h**2)*f[i+1]
54     j+=1
55 i+=1
56 X = solve(A,B)               #solve matrix AX = B
57
58 #define new array for x, ranging from 0 to 1 with increment = h
59 xnew = np.arange(0,1.01,h)
60 y1 = np.insert(X,0,0)            #insert value y = 0 into y1[0]
61 ynew = np.append(y1,0.1863)      #append value y = 0.1863 in y1 array
62
63 #plot the graph ynew vs xnew
64 plt.plot(xnew,ynew,'-og',label='Approximate solution')
65
66 #define the exact solution
67 def exact(x):
68     return 1/4*np.exp(-4*x) + x*np.exp(-4*x) - 1/4*np.cos(4*x)
69 #list of values of x with increment of 0.01
70 xexact = list(np.arange(0.0,1.01,0.01))
71
72 #calculate the yexact values for each xexact
73 yexact = []
74 for i in xexact:
75   yexact.append(exact(i))
76
77 #plot the graph between yexact and xexact
78 plt.plot(xexact,yexact,'-k',label='exact solution')
79 plt.legend()
80 plt.xlabel('x')
81 plt.ylabel('y(x)')
82 plt.title('Plot y(x) versus x for ODE: '"$y''+8y'+16y = 8sin(4x)$")
83 plt.show()
```

Listing 1: Python code using Finite Difference Method to solve ODEs

4 Result and Conclusion

Based on Figure 3, 6 difference plots were obtained for different numbers of iterations or mesh points, denoted as n. We can see as we increase the number of mesh points, n the approximate plot (green line) will be same as the exact plot (black line). Thus, we can conclude that Finite Difference method is one



(a) n = 5 (b) n = 10 (c) n = 15

(d) n = 20 (e) n = 25 (f) n = 30

Figure 3: A few plots that represents the output of code

of the methods used in numerical calculation where we can solve the ordinary differential equations and it can give us a good approximation to the exact solution if the number of iterations or mesh points that we inputted into our coding is good enough or sufficient to approximate our solution. In Figure 4, we calculate the error for each mesh points between exact and approximate values for $n = 20$. Thus, we can conclude the error between approximate solution with exact solution is low, and the accuracy is increase if the number of iteration, n is large.

n	X	Yexact	Yapproximate	Error
0	0	0	0	0
1	0.05	-0.00123379	0.000602581	0.001836371
2	0.1	0.00129597	0.004346768	0.003050798
3	0.15	0.00942958	0.013190751	0.003761171
4	0.2	0.02394837	0.028021357	0.004072987
5	0.25	0.04478523	0.048864144	0.004078914
6	0.3	0.07120814	0.075067378	0.003859238
7	0.35	0.10198379	0.105466393	0.003482603
8	0.4	0.1355256	0.138532617	0.003007017
9	0.45	0.17002884	0.172509745	0.002480905
10	0.5	0.20359417	0.205538172	0.001944002
11	0.55	0.23433954	0.235767806	0.001428266
12	0.6	0.26050007	0.261458689	0.000958619
13	0.65	0.28051482	0.281068409	0.000553589
14	0.7	0.29309917	0.293325145	0.000225975
15	0.75	0.29730191	0.297285193	1.67175E-05
16	0.8	0.29254567	0.292374008	0.000171662
17	0.85	0.27865041	0.278410145	0.000240265
18	0.9	0.25583955	0.255611885	0.000227665
19	0.95	0.22472898	0.224586854	0.000142126
20	1	0.1863	0.186305454	5.45383E-06

Figure 4: The error calculation for $n = 20$, error can be calculated by using $Y_{exact} - Y_{approximate}$ and we could see the range of our error in range of $10^{-3} - 10^{-6}$ which means our approximate value is quite accurate with the exact value

References

math.libretexts.org. (2021, Jan). *17.3: Applications of second-order differential equations.* Libretexts. Retrieved from `https://math.libretexts.org/Bookshelves/Calculus/Book:_Calculus_(OpenStax)/17:_Second-Order_Differential_Equations/17.3:_Applications_of_Second-Order_Differential_Equations`