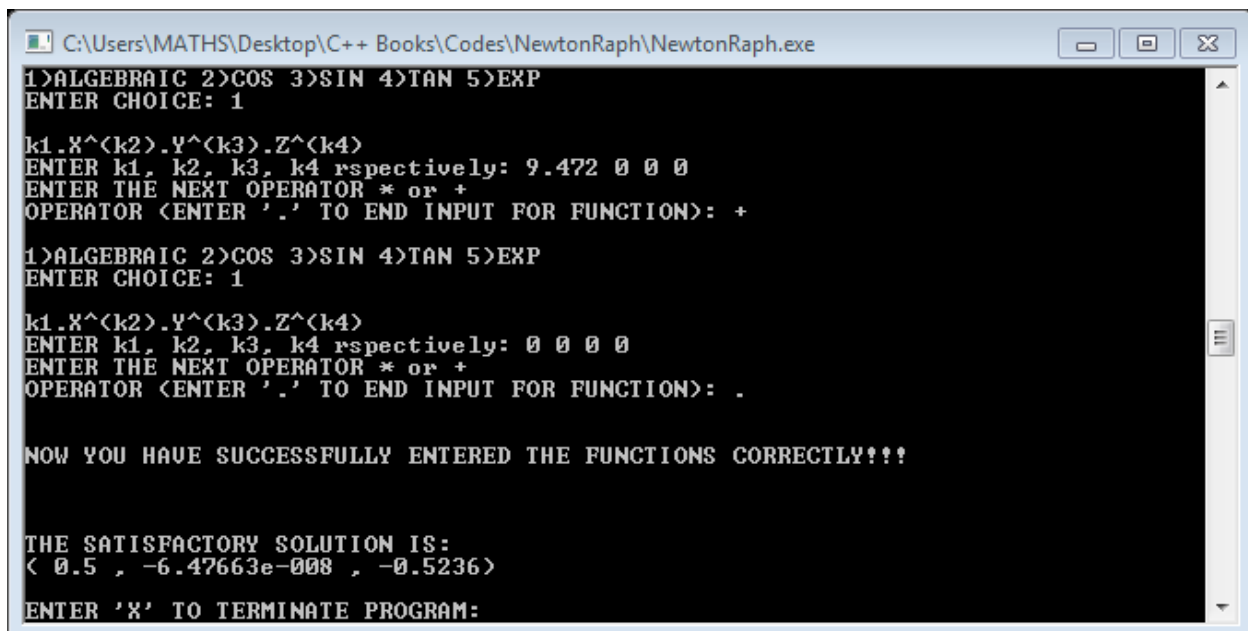


STRUCTURE OF PROGRAMS

1. A method to compute the partial derivative of a vector function (the input is a vector—a three point co-ordinate) was designed. This method is the file partialDiffForThree.h, which was implemented by using it in the program Partial_Differentiation.cpp, to find the partial derivative of some function (at a given point) with respect to each component of the vector.
2. A method was equally created to implement the Newton Raphson's algorithm for solving a system of nonlinear equations. This method made use of the partial derivative method contained in the file partialDiffForThree.h, also another file called matrixClassForThree.h, which contains the necessary class and functions for manipulating the matrix operations involved in the algorithm. Both header files were instrumental in the design of the NewtonRaph.cpp program, which housed the algorithm needed.
3. The Newton Raphson method (NewtonRaph.cpp) above, was used to solve the given system of nonlinear equations:

$$\begin{aligned}3x_1 - \cos(x_2x_3) - \frac{1}{2} &= 0 \\x_1^2 - 81(x_2 + 0.1)^2 + \sin(x_3) + 1.06 &= 0 \\e^{-x_1x_2} + 20x_3 + \frac{10\pi - 3}{3} &= 0\end{aligned}$$

using initial guess of $x=[0.1,0.1,-0.1]^T$. The result of the computation turned out to be $x=[0.5,-6.47663 \times 10^{-8},-0.5236]^T$, as can be seen in the screenshot below.



```
C:\Users\MATHS\Desktop\C++ Books\Codes\NewtonRaph\NewtonRaph.exe
1>ALGEBRAIC 2>COS 3>SIN 4>TAN 5>EXP
ENTER CHOICE: 1
k1.X^(k2).Y^(k3).Z^(k4)
ENTER k1, k2, k3, k4 respectively: 9.472 0 0 0
ENTER THE NEXT OPERATOR * or +
OPERATOR (ENTER '.' TO END INPUT FOR FUNCTION): +
1>ALGEBRAIC 2>COS 3>SIN 4>TAN 5>EXP
ENTER CHOICE: 1
k1.X^(k2).Y^(k3).Z^(k4)
ENTER k1, k2, k3, k4 respectively: 0 0 0 0
ENTER THE NEXT OPERATOR * or +
OPERATOR (ENTER '.' TO END INPUT FOR FUNCTION): .
NOW YOU HAVE SUCCESSFULLY ENTERED THE FUNCTIONS CORRECTLY!!!
THE SATISFACTORY SOLUTION IS:
< 0.5 , -6.47663e-008 , -0.5236 >
ENTER 'X' TO TERMINATE PROGRAM:
```

4. A method was created to implement the Trapezoidal method of numerical solution to a system of differential equation. The method made use of the Newton Raphson method earlier designed. The method is placed in the file called NewtonRaphson.h; this method (for Newton Raphson) has been modified to solve strictly, a system of two equations (with two unknowns), since our design of the trapezoidal method aims at solving a second order ODE, which will be reduced to a system containing two (nonlinear) equations. Also the methods for partial differentiation and for matrix manipulations were equally modified to suit the needs of the modified Newton Raphson method. The methods for partial differentiation and for matrix manipulations are contained in the files partialDiffFunctions.h and matrixFunctions.h respectively. The methods contained in the three header files were instrumental in the design of the Trapezoidal method.cpp program, which housed the algorithm needed.

5. Finally, the Trapezoidal method.cpp was used to solve the given Van Der Pol equation

$$y'' - \mu(1 - y^2)y' + y = 0$$

where $\mu=10$, $y(0) = 2$, $y'(0) = 0$ for range $x = [0,300]$, with a step size(Δx) of 0.1.

The solution generated (the set of points) are saved in the dynamic excel file (value.xls) generated whenever the program is launched. Attached to this document is the excel file **valueHzeroPt1for0To300.xls** housing the already generated solution.

6. The graph of the solution (corresponding x and y values) is plotted using the functionalities in excel spreadsheet. The graph is also contained in the file **valueHzeroPt1for0To300.xls**