**Solutions to Tasksheet 2**

# **Task 1:**

I have chosen to use the C++ Programming Language for this class. The program I wrote for Task 1 is under the [math4610/hw\_toc/Tasksheet\_02/src/](https://github.com/jpoll962/math4610/tree/master/hw_toc/Tasksheet_02/src) folder (GitHub may not let that be a clickable link). The code is as follows:

//LanguageDeclaration.cpp

#include <iostream>

int main()

{

  std::cout << "Hello World!" << std::endl;

  std::cout << "I have decided to use the C++ Programming Language for this class." << std::endl;

  return 0;

}

* The code was compiled with the command: g++ LanguageDeclaration.cpp
* The compilation resulted in the creation of an executable named a.out
* I executed the file a.out with the following command: ./a.out
* That resulted in the printing of the following message to the screen:

Hello World!

I have decided to use the C++ Programming Language for this class.

# **Task 2:**

I have edited my README.md file to contain an introduction for the repository. In addition, there are links to my hw\_toc and my software manual.

# **Task 3:**

I have created a hw\_toc folder to hold the table of contents for the class tasksheets. In addition, I have a local version of my math4610 repository up and running. I have cloned it and updated it with the git pull command.

# **Task 4:**

Taylor series expansion of a function, f(x), around a point x = a, looks like

When rearranging the terms, you get

This is the approximation of the derivative of the equation f(x). In this case, it is known as the forward difference approximation.

In the above case, a + h is used. If we replace that with a-h during times of h<0, we get the backward difference approximation

The forward difference approximation and the backward difference approximation have a first order approximation.

When you subtract the backward difference approximation from the forward difference approximation, like this:

you get:

The right-hand side demonstrates the order of accuracy with h2. After rearranging to find the first derivative, you get the centered difference approximation:

The centered difference approximation is a second order approximation because if h is decrease by a factor of 2, the error will decrease by a factor of 22.

# **Task 5:**

The order of accuracy of the given central difference approximation of the second derivative can be identified through Taylor series expansions:

Adding these expansions together to get closer to the central difference approximation:

Then, with some manipulation:

From here, we can identify the order of approximation through the h2 from the right-hand side of the equation. This central difference approximation of the second derivative is of second-order accuracy. The equation in question can be found from the previous equation through further manipulation:

The source code can be found in the [math4610/hw\_toc/Tasksheet\_02/src/SecDerivAppr.cpp](https://github.com/jpoll962/math4610/blob/master/hw_toc/Tasksheet_02/src/SecDerivAppr.cpp) and is typed out below:

//SecDerivAppr.cpp

#include <iostream>

#include <cstdio>

#include <cmath>

int main()

{

    //Assign Values

    int iter [18] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18};

    long double error [18];

    long double x = 2.0L;

    long double h [18] = {1, 0.5, pow(10,-1), pow(10,-2), pow(10,-3), pow(10,-4), pow(10,-5), pow(10,-6), pow(10,-7), pow(10,-8), pow(10,-9),

        pow(10,-10), pow(10,-11), pow(10,-12), pow(10,-13), pow(10,-14), pow(10,-15), pow(10,-16)};

    long double apprVal [18];

    long double exactVal = -cos(x);

    // Setting range of decimals to show

    std::cout.precision(16);

    // Printing Exact Value

    std::cout << "The Exact Value = " << exactVal << std::endl << std::endl;

    for(int i = 0; i < 18; i++)

    {

        apprVal[i] = ( cos((x) + (h[i])) - 2.0 \* cos(x) + cos((x) - (h[i])) ) / ( pow(h[i],2) );

        error[i] = std::abs(exactVal - apprVal[i]);

    }

    std::cout << std::endl;

    std::cout << "| iteration |     h     |" << std::endl;

    for(int i = 0; i < 18; i++)

    {

        i <= 8 ? std::cout << "|    0" : std::cout << "|    ";

        std::cout << iter[i] << "     | " << h[i] << "       |" << std::endl;

    }

    std::cout << std::endl;

    std::cout << "| iteration |   approximation   |" << std::endl;

    for(int i = 0; i < 18; i++)

    {

        i <= 8 ? std::cout << "|    0" : std::cout << "|    ";

        std::cout << iter[i] << "     | " << apprVal[i] << " |" << std::endl;

    }

    std::cout << std::endl;

    std::cout << "| iteration |         error         |" << std::endl;

    for(int i = 0; i < 18; i++)

    {

        i <= 8 ? std::cout << "|    0" : std::cout << "|    ";

        std::cout <<  iter[i] << "     | " << error[i] << " |" << std::endl;

    }

}

The output is:

|  |
| --- |
| The Exact Value = 0.4161468365471424 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| | iteration | h | |
|  |

|  |
| --- |
| |-----------------------| |
|  |

|  |
| --- |
| | 01 | 1 | |
|  |

|  |
| --- |
| | 02 | 0.5 | |
|  |

|  |
| --- |
| | 03 | 0.1 | |
|  |

|  |
| --- |
| | 04 | 0.01 | |
|  |

|  |
| --- |
| | 05 | 0.001 | |
|  |

|  |
| --- |
| | 06 | 0.0001 | |
|  |

|  |
| --- |
| | 07 | 1e-05 | |
|  |

|  |
| --- |
| | 08 | 1e-06 | |
|  |

|  |
| --- |
| | 09 | 1e-07 | |
|  |

|  |
| --- |
| | 10 | 1e-08 | |
|  |

|  |
| --- |
| | 11 | 1e-09 | |
|  |

|  |
| --- |
| | 12 | 1e-10 | |
|  |

|  |
| --- |
| | 13 | 9.999999999999999e-12 | |
|  |

|  |
| --- |
| | 14 | 1e-12 | |
|  |

|  |
| --- |
| | 15 | 1e-13 | |
|  |

|  |
| --- |
| | 16 | 1e-14 | |
|  |

|  |
| --- |
| | 17 | 1e-15 | |
|  |

|  |
| --- |
| | 18 | 1e-16 | |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| | iteration | approximation | |
|  |

|  |
| --- |
| |--------------------------------| |
|  |

|  |
| --- |
| | 01 | 0.3826034823619792 | |
|  |

|  |
| --- |
| | 02 | 0.4075490368602161 | |
|  |

|  |
| --- |
| | 03 | 0.415800163092389 | |
|  |

|  |
| --- |
| | 04 | 0.4161433686711291 | |
|  |

|  |
| --- |
| | 05 | 0.4161468019070469 | |
|  |

|  |
| --- |
| | 06 | 0.41614681700608 | |
|  |

|  |
| --- |
| | 07 | 0.4161471167662966 | |
|  |

|  |
| --- |
| | 08 | 0.4160005673270462 | |
|  |

|  |
| --- |
| | 09 | 0.4385380947269369 | |
|  |

|  |
| --- |
| | 10 | 1.110223024625156 | |
|  |

|  |
| --- |
| | 11 | 55.51115123125782 | |
|  |

|  |
| --- |
| | 12 | 5551.115123125782 | |
|  |

|  |
| --- |
| | 13 | 555111.5123125783 | |
|  |

|  |
| --- |
| | 14 | 0 | |
|  |

|  |
| --- |
| | 15 | 5551115123.125782 | |
|  |

|  |
| --- |
| | 16 | -1665334536937.735 | |
|  |

|  |
| --- |
| | 17 | 277555756156289.1 | |
|  |

|  |
| --- |
| | 18 | 0 | |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| | iteration | error | |
|  |

|  |
| --- |
| ------------------------------------- |
|  |

|  |
| --- |
| | 01 | 0.03354335418516324 | |
|  |

|  |
| --- |
| | 02 | 0.008597799686926311 | |
|  |

|  |
| --- |
| | 03 | 0.0003466734547533656 | |
|  |

|  |
| --- |
| | 04 | 3.467876013296678e-06 | |
|  |

|  |
| --- |
| | 05 | 3.464009551423786e-08 | |
|  |

|  |
| --- |
| | 06 | 1.954106237933573e-08 | |
|  |

|  |
| --- |
| | 07 | 2.802191542139454e-07 | |
|  |

|  |
| --- |
| | 08 | 0.0001462692200962512 | |
|  |

|  |
| --- |
| | 09 | 0.02239125817979448 | |
|  |

|  |
| --- |
| | 10 | 0.694076188078014 | |
|  |

|  |
| --- |
| | 11 | 55.09500439471068 | |
|  |

|  |
| --- |
| | 12 | 5550.698976289234 | |
|  |

|  |
| --- |
| | 13 | 555111.0961657417 | |
|  |

|  |
| --- |
| | 14 | 0.4161468365471424 | |
|  |

|  |
| --- |
| | 15 | 5551115122.709635 | |
|  |

|  |
| --- |
| | 16 | 1665334536938.151 | |
|  |

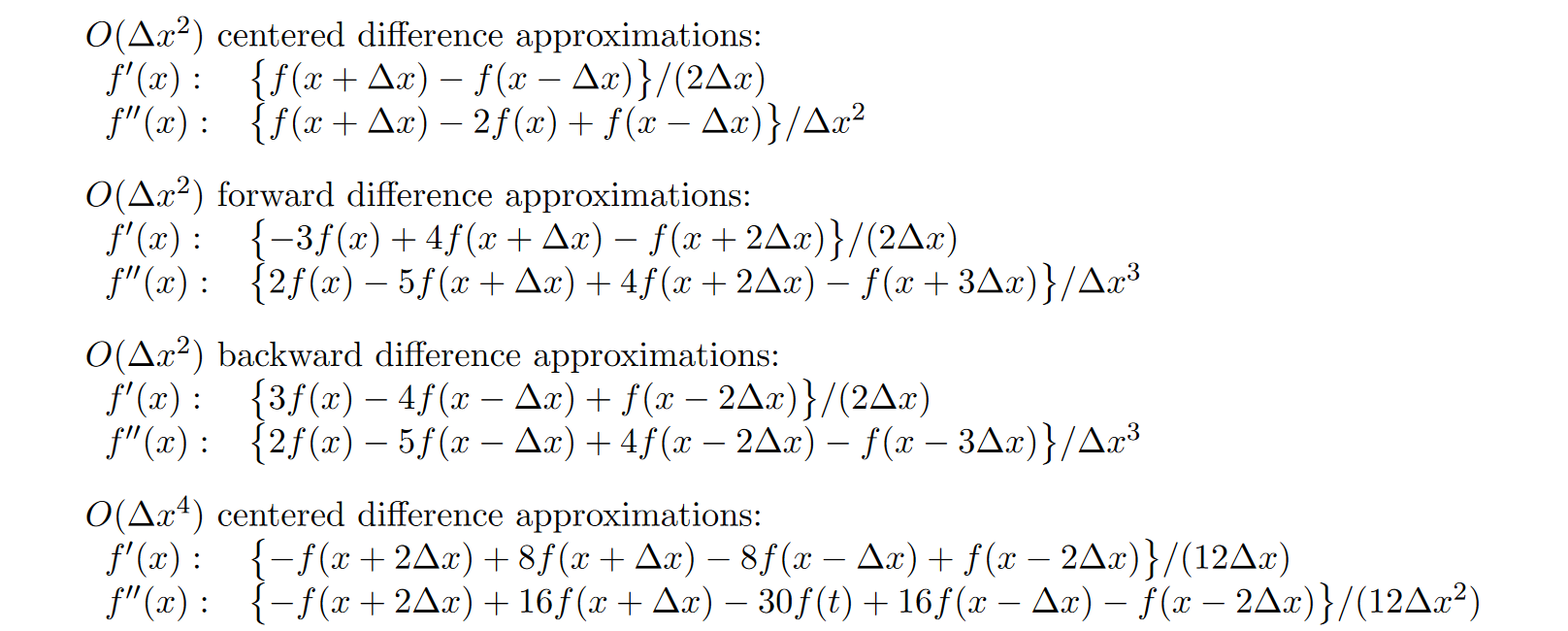
|  |
| --- |
| | 17 | 277555756156288.7 | |
|  |

| 18 | 0.4161468365471424 |

# **Task 6:**

There are three finite difference approximations mentioned in my findings. Those are the forward difference approximation, the backward difference approximation, and the central/centered difference approximation ([Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems – Chapter 1](http://www.siam.org/books/ot98/sample/OT98Chapter1.pdf)). Big O notation is often used in Taylor Series Approximations to express the order of accuracy, such as O(h) for first order and O(h2) for second order. In addition, higher-order approximations can usually be found using similar manipulation techniques as with the first and second orders ([Fundamentals of Engineering Numerical Analysis – Chapter 2, pg 14](https://www.researchgate.net/profile/Parviz_Moin/publication/245378079_Fundamentals_of_Engineering_Numerical_Analysis/links/57150bae08ae8b7c0481ac53/Fundamentals-of-Engineering-Numerical-Analysis.pdf)~).

Some examples of finite difference approximations of different orders ([Numerical Differentiation: Finite Differences](http://www.dam.brown.edu/people/alcyew/handouts/numdiff.pdf)):



For the purpose of continuity: