Floating Point Arithmetic & Sensitivity Analysis

**Section 1: Quadratic Formulas**

* 1. **Introduction**

Blank

The first section of the project explores the inaccuracy of machine floating point arithmetic (MATLAB) under a sequence of operations by comparing the tradition quadratic formula:

and a second implementation:

against the MATLAB **roots** function which computes the roots using a symbolic formula of , , and .

* 1. **Methods**

Blank

The roots of five quadratic equations were tested using , , and **roots**. The following table (*Tab 1.2*) shows the values of , , and which define the set of quadratic equations in their standard forms.

|  |  |  |  |
| --- | --- | --- | --- |
| Trial |  |  |  |
| 1 | 1 | -2 | 3 |
| 2 | 3 |  | -1 |
| 3 |  | 9 | -3 |
| 4 | 1 | 2 |  |
| 5 |  |  |  |

Blank

Blank

* 1. **Results**

Blank

The following table (*Tab 1.3*) shows the values of the roots determined by each method. Cells for which and/or differ from **roots** are highlighted in red.

|  |  |  |
| --- | --- | --- |
| Trial |  |  |
| 1 | |  | | --- | | **TQF:** 1.000000000000000 + 1.414213562373095i | | **2ndQF:** 1.000000000000000 + 1.414213562373095i | | **roots**: 1.000000000000000 + 1.414213562373095i | | |  | | --- | | **TQF**: 1.000000000000000 - 1.414213562373095i | | **2ndQF:** 1.000000000000000 - 1.414213562373095i | | **roots**: 1.000000000000000 - 1.414213562373095i | |
| 2 | |  | | --- | | **TQF:** 0 | | **2ndQF:** 1.000000000000000e-14 | | **roots**: 1.000000000000000e-14 | | |  | | --- | | **TQF:** 0 -3.333333333333333e+13 | | **2ndQF:** 0 -3.333333333333333e+13 | | **roots**: 0 -3.333333333333333e+13 | |
| 3 | |  | | --- | | **TQF:**  0.355271367880050 | | **2ndQF:** 0.333333333333333 | | **roots**: 0.333333333333333 | | |  | | --- | | **TQF:** 0 -9.000000000000004e+14 | | **2ndQF:** 0 -9.000000000000004e+14 | | **roots**: 0 -9.000000000000004e+14 | |
| 4 | |  | | --- | | **TQF:** 4.884981308350689e-15 | | **2ndQF:** 4.999999999999988e-15 | | **roots**: 4.999999999999988e-15 | | |  | | --- | | **TQF:** 0 -2.000000000000005 | | **2ndQF:** 0 -2.000000000000005 | | **roots**: 0 -2.000000000000005 | |
| 5 | |  | | --- | | **TQF:** -0.999999990000000 | | **2ndQF:** 0 -0.999999990000000 | | **roots**: 0 -0.999999990000000 | | |  | | --- | | **TQF:** -0.999999990000000 | | **2ndQF:** 0 -0.999999990000000 | | **roots**: 0 -0.999999990000000 | |

* 1. **Conclusion**

1. Blank

The accuracy of was greater than because for 15 digits of precision, produced results equivalent to **roots**. Trials 2, 3, and 4 show that , when programmed into MATAB, produces slightly inaccurate results. In Trial 2, is expected to be very slightly larger than zero but produces . In Trial 4, the same discrepancy is present; produces an inaccurate result when is expected to be very slightly larger than zero.

The overall takeaway from this section of the project is that applying formulas to machines will introduce error to some extent. Therefore, developing algorithms which circumvent error-inducing operations is an important breadth of mathematics.

**Section 2: Derivative Approximations**

* 1. **Introduction**

Blank

In this section of the project, the accuracy of two formulaic approximations of the derivative of a certain function were analyzed. The two approximations are based on the symbolic limit definition of the derivative:

We cannot accurately compute the numerical value of a derivative at any given x-value since computers must round off the value to some degree of precision. The limit definition of the derivative, along with “derivative rules,” allow us to compute the symbolic derivative but when we require a numerical result, we must turn to approximations. The first approximation:

is just the expression contained within the limit of . uses an h-value which indicates the size of the interval which the approximating secant line depends on. Since approximates at , the fact that appears as the second term in the numerator of means that is a one-sided approximation whereas the second approximation:

is a two-sided approximation since the approximating secant line passes through both and , which exist on either side of .

* 1. **Methods**

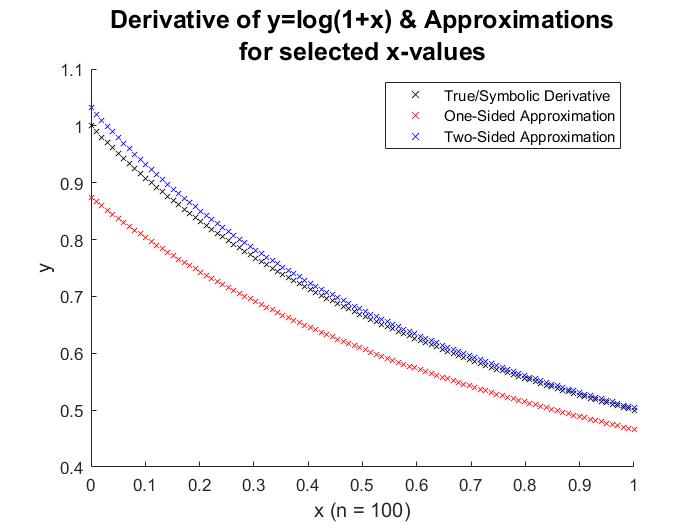
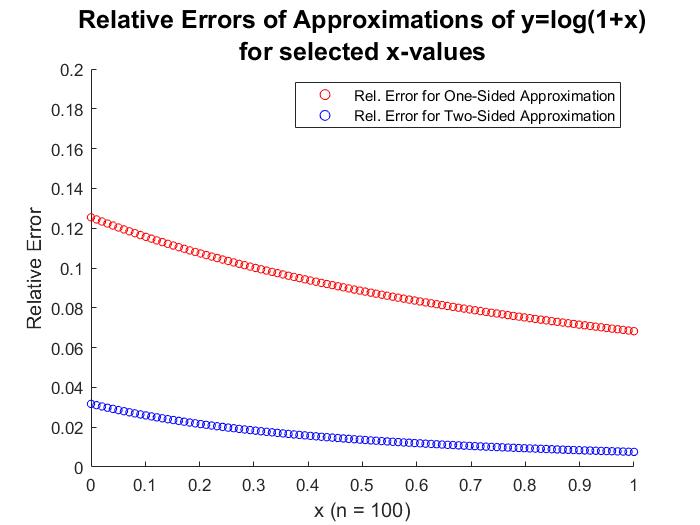
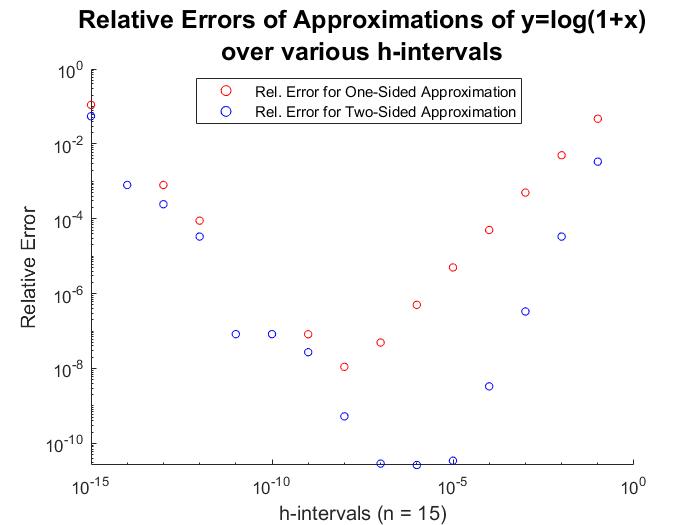
Blank

**2.2.a)** The specific function chosen to be analyzed was . Then, 100 evenly spaced points, , were constructed on the interval . The symbolically attained true derivative, , was plotted at each x-value along with the two approximations (using ). Then, the relative error between () and were plotted.

**2.2.b)** In addition to testing the accuracy of the approximations over a range of x-values, a range of h-values such that were also tested (using ). Then, the relative error between () and were plotted on a **log-log** scale.

* 1. **Results**

Blank

The following two figures (*Fig 2.3.b* and *Fig 2.3.c*) show the relative errors for the two methods described in *Section 2.2*, whereas *Fig 2.3.a* shows the true derivative and the approximations themselves.

**TOP RIGHT:** *Fig 2.3.a*

**TOP LEFT:** *Fig 2.3.b*

**BOTTOM:** *Fig 2.3.c*

* 1. **Conclusion**

Blank

*Fig 2.3.a* appears to show that the two-sided approximation is more accurate than for all and *Fig 2.3.b* confirms that since the relative error of is less than for all . Note that using relative error as a measure of accuracy over the interval of is valid since the true derivative, , is never equal to zero (a concept explored in *Section 3*).

Another interesting thing of notice is the fact that the relative error trend of for is the same for . This is because plays the same role in both equations: it determines the interval on which the approximating secant line which () represents ( for and for ). When is a value near the minimum and maximum of , the relative errors of () increase (according to *Fig 2.3.c)*. The reason behind the inaccuracy for smaller h-values is because both denominators of () contain a multiple of and dividing by a very small value yields a very large result. The reason behind the inaccuracy for larger h-values is because the actual function itself, , changes more between the endpoints of the interval on which the approximating secant line exists. A larger numerator yields a larger result.

It should be noted that these conclusions were developed using data from a specific function. Deciding which derivative approximation (one-sided or two-sided) depends on the selected h-value and how much the function changes within the respective interval ( for or for ).

**Section 3: Polynomial Evaluations**

* 1. **Introduction**

The focus of this section of the project was the inaccuracies of evaluating polynomials with different forms. As expressed in *Section 1*, inaccuracy (due to loss of exact precision) is inherent with numerical calculations using a computing machine so it is useful to rely on alternative algorithms which minimize error-introducing operations. However, in the absence of an evaluation algorithm, a reformulated version of a function may be used.

* 1. **Methods**

For this analysis, the function of focus was:

and a second, but algebraically equivalent, function:

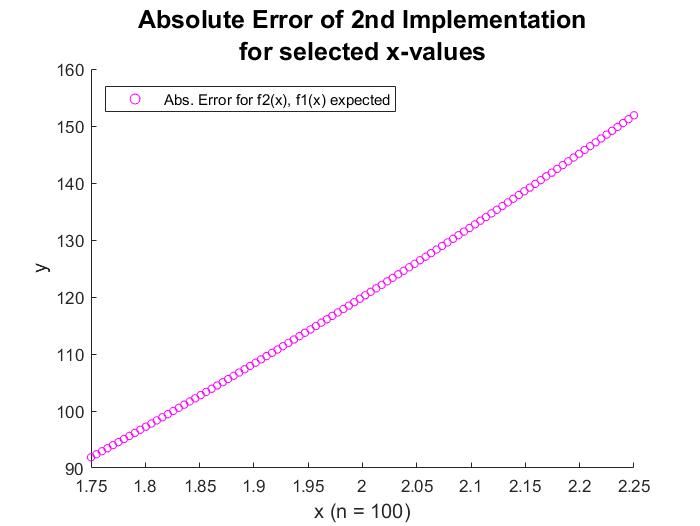
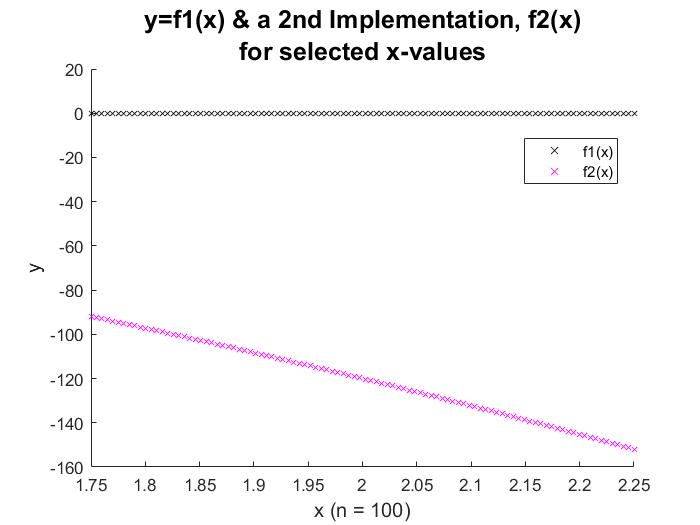
was also used.

**3.2.a)** Evaluations of () at where X spans the interval were compared against each other. Then, the absolute error ad relative error between and were separately plotted since absolute error was plotted on a **lin-lin** scale and relative error was plotted on a **lin-log** scale; is assumed to be highly accurate, so it was used as the expected value.

**3.2.a)** Apart from evaluations of () at , a [provided] algorithm (**bisection**) was used to compute a root, , of () such that . **bisection** checks the given interval and determines whether is in or where , the midpoint of . . The process repeats by recursively replacing either (if ) or (if ) with and updating until where The results were recorded in a table.

* 1. **Results**

The two implementations ( and ) of the function are shown in *Fig 3.3.a.i*. The absolute error is shown in *Fig 3.3.a.ii* and relative error in *Fig 3.3.a.iii*. Lastly, the results of **bisection** on () for various values of and are shown in *Tab 3.3.b****.***



**TOP:** *Fig 3.3.a.i*

**BOTTOM LEFT:** *Fig 3.3.a.ii*

**BOTTOM RIGHT:** *Fig 3.3.a.iii*

**NEXT PG:** *Tab 3.3.b*

|  |  |  |
| --- | --- | --- |
| Interval | for | for |
|  | 2 | 2 |
|  | 2 | 2 |
|  | 2 | 2 |
|  | 2 | 2 |
|  | 2 | 2 |

* 1. **Conclusion**

Although and are algebraically (therefore symbolically) equivalent, MATLAB shows some discrepancy between the evaluation of those functions over X. The absolute error appears to take on a linear form with a positive slope, meaning that on X, is linearly growing away from . When looking at the relative error on a **lin-log** scale, the relative error rises around since . Any deviation away from zero is considered infinitely inaccurate in the relative sense.

It is clear from this portion of *Section 3* that the mathematical result of an error calculation may not adhere to the system which is explains. For example, if someone is measuring rainfall and collects 0.002cm of water when the forecast said 0cm, the relative error would be infinite, but it may be logical to accept 0.002cm of water as being consistent with the forecast since the water could have come from condensation. In this case, absolute error seems to be a more reliable measure of error as it directly visually correlates with the gap between and since when is near 2 is very close to zero.

However, despite the inaccuracy between the implementations, **bisection** was able to accurately find such that it was the same for and for all intervals tested. Since it is known that and observed error at , it can be inferred that . However, **bisection** does not compute the root traditionally but depends on an algorithm that circumvents error-inducing operations; even though the plot of shows the algorithm can still identify . This result reaffirms the overall takeaway from *Section 1.*