Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com

Floating Point Arithmetic & Sensitivity Analysis

Section 1: Quadratic Formulas

1.1) Introduction

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:

$$TQF: x_{\pm} = \frac{-b \pm \sqrt[2]{b^2 - 4ac}}{2a}$$

and a second implementation:

$$2ndQF+: x_{+} = -\frac{b + sign(b) * \sqrt[2]{b^2 - 4ac}}{2a}$$

$$2ndQF-: x_{-} = \frac{c}{ax_{+}}$$

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

1.2) Methods

The roots of five quadratic equations were tested using TQF, $2ndQF$, and **roots**. The following table (*Tab 1.2*) shows the values of a, b, and c which define the set of quadratic equations in their standard forms.

Trial	a	b	c
1	1	-2	3
2	3	10^{-14}	-1
3	10^{-14}	9	-3
4	1	2	-10^{-14}
5	$10^8 + 1$	$2 * 10^8$	$10^8 - 1$

Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com

1.3) Results

The following table (*Tab 1.3*) shows the values of the roots determined by each method. Cells for which TQF and/or $2ndQF$ differ from **roots** are highlighted in red.

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

Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com

	2ndQF: 4.999999999999988e-15	**2ndQF:** 0 - 2.000000000000005
	roots: 4.999999999999988e-15	**roots**: 0 - 2.000000000000005
5	**TQF:** -0.999999990000000	**TQF:** - 0.999999990000000
	2ndQF: 0 - 0.999999990000000	**2ndQF:** 0 - 0.999999990000000
	roots: 0 - 0.999999990000000	**roots**: 0 - 0.999999990000000

1.4) Conclusion

The accuracy of $2ndQF$ was greater than TQF because for 15 digits of precision, $2ndQF$ produced results equivalent to **roots**. Trials 2, 3, and 4 show that TQF, when programmed into MATAB, produces slightly inaccurate results. In Trial 2, x_+ is expected to be very slightly larger than zero but TQF produces $x_+ = 0$. In Trial 4, the same discrepancy is present; TQF produces an inaccurate result when x_+ 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

2.1) Introduction

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:

$$f'(x) = \lim_{h \to 0} \left(\frac{f(x+h) - f(x)}{h} \right)$$

Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com

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:

$$f_1(x) = \frac{f(x+h) - f(x)}{h}$$

is just the expression contained within the limit of f'. f_1 uses an h-value which indicates the size of the interval which the approximating secant line depends on. Since f_1 approximates f' at x, the fact that $f(x)$ appears as the second term in the numerator of f_1 means that f_1 is a one-sided approximation whereas the second approximation:

$$f_2(x) = \frac{f(x+h) - f(x-h)}{2h}$$

is a two-sided approximation since the approximating secant line passes through both $f(x+h)$ and $f(x-h)$, which exist on either side of $f(x)$.

2.2) Methods

2.2.a) The specific function chosen to be analyzed was $f(x) = \log(1+x)$. Then, 100 evenly spaced points, $X = \{x_1, x_2, \dots, x_{100}\}$, were constructed on the interval $[0, 1]$. The symbolically attained true derivative, $f'(x) = \frac{1}{1+x}$, was plotted at each x-value along with the two approximations (using $h = 0.3$). Then, the relative error between f_c ($c \in \{1, 2\}$) and f' 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 $h \in H = \{10^{-15}, 10^{-14}, \dots, 10^{-1}\}$ were also tested (using $x = 0$). Then, the relative error between f_c ($c \in \{1, 2\}$) and f' were plotted on a **log-log** scale.

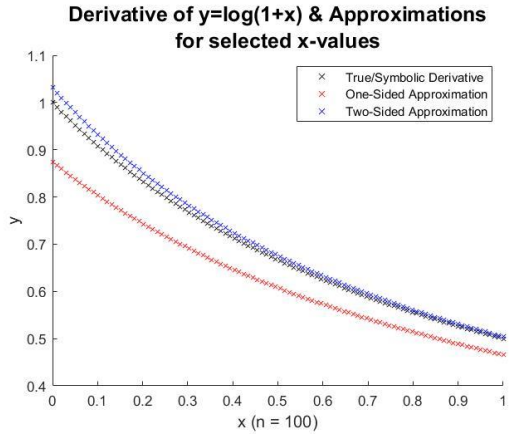
2.3) Results

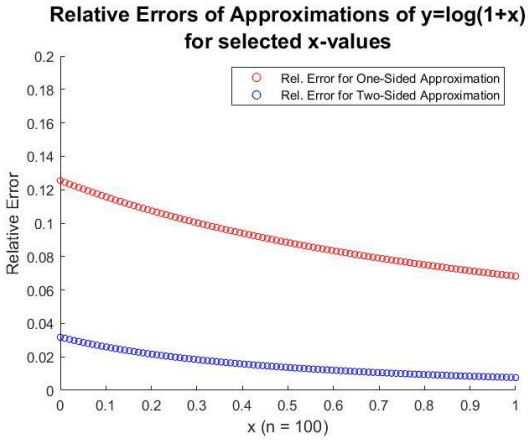
Vinayak Ravichandran

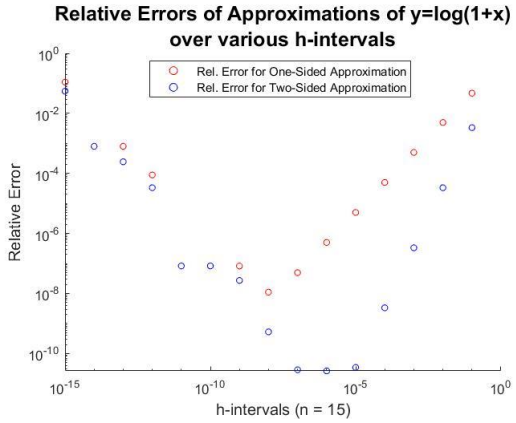
Github: @vravich01

Email: vinayak1ravichandran@gmail.com

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*

2.4) Conclusion

Fig 2.3.a appears to show that the two-sided approximation f_2 is more accurate than f_1 for all $x \in X$ and *Fig 2.3.b* confirms that since the relative error of f_2 is less than f_1 for all $x \in X$. Note that using relative error as a measure of accuracy over the interval of X is valid since the true derivative, $f'(x)$, 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 f_1 for $h \in H$ is the same for f_2. This is because h plays the same role in both equations: it determines the interval on which the approximating secant line which f_c ($c \in \{1, 2\}$) represents (h for f_1 and $2h$ for f_2). When h is a value near the minimum and maximum of H, the

relative errors of f_c ($c \in \{1, 2\}$) increase (according to *Fig 2.3.c)*. The reason behind the inaccuracy for smaller h-values is because both denominators of f_c ($c \in \{1, 2\}$) contain a multiple of h 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, $f(x) = \log(1 + x)$, 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 (h for f_1 or $2h$ for f_2).

Section 3: Polynomial Evaluations

3.1) 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.

3.2) Methods

For this analysis, the function of focus was:

$$f_1(x) = (x - 2)^{13}$$

and a second, but algebraically equivalent, function:

$$f_2(x) = x^{13} - 26x^{12} + 312x^{11} + 2288x^{10} + 11440x^9 + 41184x^8 + 109824x^7 + 219648x^6 + 329472x^5 + 366080x^4 + 292864x^3 + 159774x^2 + 53248x - 8192$$

was also used.

3.2.a) Evaluations of f_c ($c \in \{1, 2\}$) at $x \in X = \{x_1, x_2, \dots, x_{100}\}$ where X spans the interval $[1.75, 2.25]$ were compared against each other. Then, the absolute error ad relative error between f_1 and f_2 were separately plotted since absolute error was plotted

Vinayak Ravichandran

Github: @vravich01

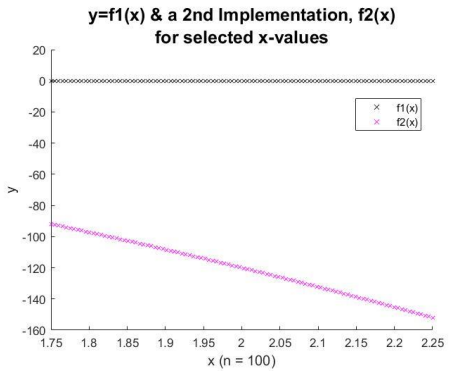
Email: vinayak1ravichandran@gmail.com

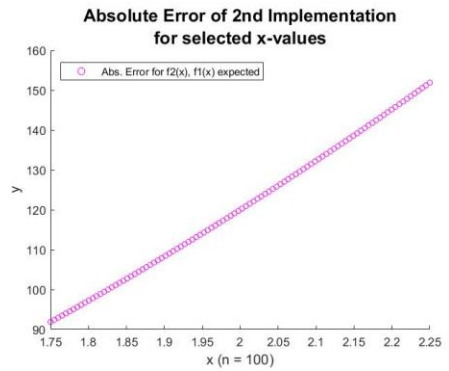
on a **lin-lin** scale and relative error was plotted on a **lin-log** scale; f_1 is assumed to be highly accurate, so it was used as the expected value.

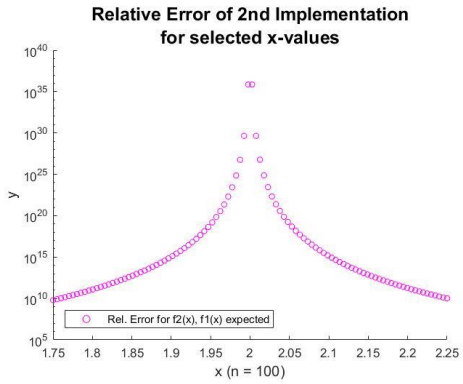
3.2.a) Apart from evaluations of f_c ($c \in \{1, 2\}$) at x, a [provided] algorithm (**bisection**) was used to compute a root, x_r, of f_c ($c \in \{1, 2\}$) such that $x_r \in [a, b]$. **bisection** checks the given interval and determines whether x_r is in $[a, c]$ or $[c, b]$ where $c = \frac{a+b}{2}$, the midpoint of $[a, b]$. . The process repeats by recursively replacing either a (if $x_r \in [c, b]$) or b (if $x_r \in [a, c]$) with c and updating c until $p = x_r = c$ where $p \in \{a, b\}$. The results were recorded in a table.

3.3) Results

The two implementations (f_1 and f_2) 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 f_c ($c \in \{1, 2\}$) for various values of a and b 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	x_r for f_1	x_r for f_2
$[-5, 5]$	2	2
$[-10, 10]$	2	2

Vinayak Ravichandran

Github: @vravich01

Email: vinayak1ravichandran@gmail.com

$[-50, 50]$	2	2
$[100, -100]$	2	2
$[-1000, 1000]$	2	2

3.4) Conclusion

Although f_1 and f_2 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, f_2 is linearly growing away from f_1. When looking at the relative error on a **lin-log** scale, the relative error rises around $x = 2$ since $f_1(2) = 0$. 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 f_1 and f_2 since $f_1(x)$ when x is near 2 is very close to zero.

However, despite the inaccuracy between the implementations, **bisection** was able to accurately find x_r such that it was the same for f_1 and f_2 for all intervals tested. Since it is known that $f_1(2) = 0$ and observed error at $x = 2$, it can be inferred that $f_2(2) \neq 0$. However, **bisection** does not compute the root traditionally but depends on an algorithm that circumvents error-inducing operations; even though the plot of f_2 shows $f_2(2) \neq 0$ the algorithm can still identify $x_r = 2$. This result reaffirms the overall takeaway from *Section 1*.