Math 4610 Fundamentals of Computational Mathematics - Topic 09.

In this section we will see an example of an approximation of a derivative that produces errors in the value of the derivative no matter what we do with the limit approximation. The difference between the exact and approximate value will reduce at first and then will start to grow. This is an easy result to replicate and identifies the problem of finite resource limitations on any computer. The problem arises when the numbers being used are so small, the computer will treat them as a zero or garbage even though analytically the values are nonzero.

Approximation of the First Derivative - the Difference Quotient

One of the most useful tools in mathematics is the derivative. In a calculus course, we say that the derivative of a function, f, at a point a in the domain of the function exists if

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

exists. In this case, the function, is said to be differential at the point a. The definition usually is specified on some open interval that contains the point, a. From a theoretical point of view, showing the limit exists requires the values of the function at points arbitrarily close to the point a and requires checking all the infinitely many points in that neighborhood. In a calculus course, rules are established to allow us to compute a formula for the derivative at the point.
For example, suppose $f(x) = cos(x)$. Then we know through the skill set obtained in a first semester calculus course that

$$f'(x) = -sin(x)$$

The formula is developed using a series of simple theorems that are actually pretty easy to prove. If we have a formula for the derivative, all we really need to do is to evaluate the derivative after computing the symbolic form of the derivative. Note that platforms like Mathematica, Maple, and others make it simple to compute a lot of derivatives of elementary functions. Mathematica is every first semester calculus student's dream come true. Most scientific calculators can compute the same sorts of derivatives.
From a computational point of view, taking the limit of a function is not truely feasible due to the issue of an infinite number of points to be checked to the left and right of the point a in the definition above. It is the calculus formulas that save us on simple elementary functions. But what happens if the function is defined by data points and we do not know a formula that defines the relationship between the inputs and outputs in the data set. Also, it is not unusual that the data sets can be enormous involving petabytes of data. It may even be necessary to process/flatten the data before trying to define a functional relationship if that is even possible.
There are other problems that we cannot avoid which is what we will be investigating in this section. Computationally, we cannot use the limit definition of the derivative to compute approximations. So, we can approximate the derivative using the difference quotient in the definition. That is,

$$f'(a) \approx \frac{f(a+h) - f(a)}{(a+h) - a} = \frac{f(a+h) - f(a)}{h}$$

Given a function, say $f(x) = cos(x)$ as above, we could write an approximation for the derivative at a point, say $a = 2.0$. The approximation will look like

$$-sin(2.0) \approx \frac{cos(2.0+h) - cos(2.0)}{h}$$

From our calculus training, as we take h closer to zero, the values obtained from the difference quotient will approach the derivative value, $-sin(2.0)$.
Unfortunately, things are not as we assumed in calculus. The main problem we will run into is the finite representation of numbers. After we complete this example, we will talk about absolute and relative error. These errors become very important due to the finite precision of number representation on your favorite computer.

The Example

Let's compute the value of the difference quotient at values of $h = 0.1$, $h = 0.01$, $h = 0.001$, and so on and see if the difference quotient actually converges to the value we expect. The following table lists the values of the difference quotient with the exact value and the difference between the approximation and the exact value as h varies. Note that

$$-sin(2.0) = -0.034899497...$$

The values discussed above are shown in the following table with h decreasing as the iteration counter increases.

iteration	h	error	difference
01	0.50000	0.22077	-0.37769776210702344
02	0.25000	0.10583	-0.49263505234211946
03	0.12500	0.05156	-0.5469085273524055
04	0.06250	0.02541	-0.5730550005017694
05	0.03125	0.01261	-0.5858578909652437
06	0.01562	0.00628	-0.5921889853481872
07	0.00781	0.00313	-0.595336604818101
08	0.00390	0.00156	-0.5969058904773021
09	0.00195	0.00078	-0.5976893970423589
10	0.0009	0.00039	-0.5980808656044019
11	0.0004	0.00019	-0.5982765286237282
12	0.0002	9.78017e-05	-0.5983743423075794
13	0.1220-e03	4.88994e-05	-0.5984232446917304
14	6.103515e-05	2.44493e-05	-0.5984476947705843
15	3.051757e-05	1.22245e-05	-0.5984599195289775
16	1.525878e-05	6.11226e-06	-0.5984660318426904
17	7.629394e-06	3.05611e-06	-0.598469087984995
18	3.814697e-06	1.52806e-06	-0.5984706160379574
19	1.907348e-06	7.64032e-07	-0.5984713800717145
20	9.536743e-07	3.81957e-07	-0.5984717621468008
21	4.768371e-07	1.91036e-07	-0.5984719530679286
22	2.384185e-07	9.55754e-08	-0.5984720485284925
23	1.192092e-07	4.71466e-08	-0.5984720969572663
24	5.960464e-08	2.38636e-08	-0.5984721202403307
25	2.980232e-08	1.08251e-08	-0.5984721332788467
26	1.490116e-08	3.37452e-09	-0.5984721407294273
27	7.450580e-09	4.07605e-09	-0.5984721481800079
28	3.725290e-09	4.07605e-09	-0.5984721481800079
29	1.862645e-09	3.38783e-08	-0.5984721779823303
30	9.313225e-10	2.57262e-08	-0.5984721183776855
31	4.656612e-10	2.57262e-08	-0.5984721183776855
32	2.328306e-10	2.57262e-08	-0.5984721183776855
33	1.164153e-10	4.51110e-07	-0.5984725952148438
34	5.820766e-11	4.51110e-07	-0.5984725952148438
35	2.910383e-11	4.51110e-07	-0.5984725952148438
36	1.455191e-11	4.51110e-07	-0.5984725952148438
37	7.275957e-12	8.08050e-06	-0.598480224609375
38	3.637978e-12	8.08050e-06	-0.598480224609375
39	1.818989e-12	3.85980e-05	-0.5985107421875
40	9.094947e-13	3.85980e-05	-0.5985107421875
41	4.547473e-13	0.00016	-0.5986328125
42	2.273736e-13	0.00016	-0.5986328125
43	1.136868e-13	0.00016	-0.5986328125

A Python Script to Generate the Values and Errors

A simple program was written in Python to generate the values in the table shown above. The code is shown below.

```
from matplotlib import pyplot as plt
import numpy as np
#
# initialize the exact value of the derivative
# ------------------------------------------
#
aval = 2.5
exactVal = -np.sin(aval)
#
# set up the arrays for plotting the log-log plot we need
# -------------------------------------------------------
#
x = []
y = []
#
# initialize the array for the increment size and error in the finite
# difference approximation
# ------------------------
#
h = []
error = []
#
# append the initial increment with a starting value - in this case, 1.0
# ----------------------------------------------------------------------
#
h.append(1.0)
#
# compute the difference quotient for the increment value
# -------------------------------------------------------
#
dfVal = ( np.cos(aval + h[0]) - np.cos(aval) ) / h[0]
error.append(np.abs(exactVal - dfVal))
#
# append the log-log point for plotting at the end
# ------------------------------------------------
#
x.append(np.log(h[0]))
y.append(np.log(error[0]))
#
# print the exact value for sanity
# --------------------------------
#
print('The exact derivative value is: ', exactVal)
#
# set a loop counter
# -----------------
#
l=1
#
```

```
# the loop over ndiv increments
# -----------------------------
#
ndiv = 44
while l<44:
    #
    # append the next increment of h
    # -----------------------------
    #
    h.append(0.5 * h[l-1])
    #
    # compute the numerator and denominator for the difference approximation
    # and compute the approximation from these
    # ----------------------------------------
    #
    numval = np.cos(aval+h[l])-np.cos(aval)
    denom = h[l]
    dfVal = numval / denom
    #
    # compute the error in the approximation
    # --------------------------------------
    #
    error.append(np.abs( dfVal - exactVal ))
    #
    # append the log-log point to the arrays for plotting below
    # ---------------------------------------------------------
    #
    x.append(np.log(h[l]))
    y.append(np.log(error[l]))
    #
    # update the loop iterator
    # ------------------------
    #
    l += 1
#
# set up a plot for the data generated
# ------------------------------------
#
plt.title('Error in the Difference Quotient of the Derivative')
plt.xlabel('Increment Values: h')
plt.ylabel('Error in the Approximation')
plt.plot(x, y, label='Log-Log Plot of Error for cos(2.5)')
plt.legend()
plt.show()
```

Depicting the Results Graphically

The results in the table are graphed on a log-log plot to see how the error varies as h is reduced. In looking at this type of graph you should move your eyes right to left since as h decreases we expect the error to go to zero. The error reduces for awhile, but then starts to grow due to finite precision issues with numbers and arithmetic operations. Another point to make about the graph is that the error is not changing smoothly as it increases. This, as we will find is due to the fact that the numbers being used are random garbage. We will look into these problems over the next few topics in the course.

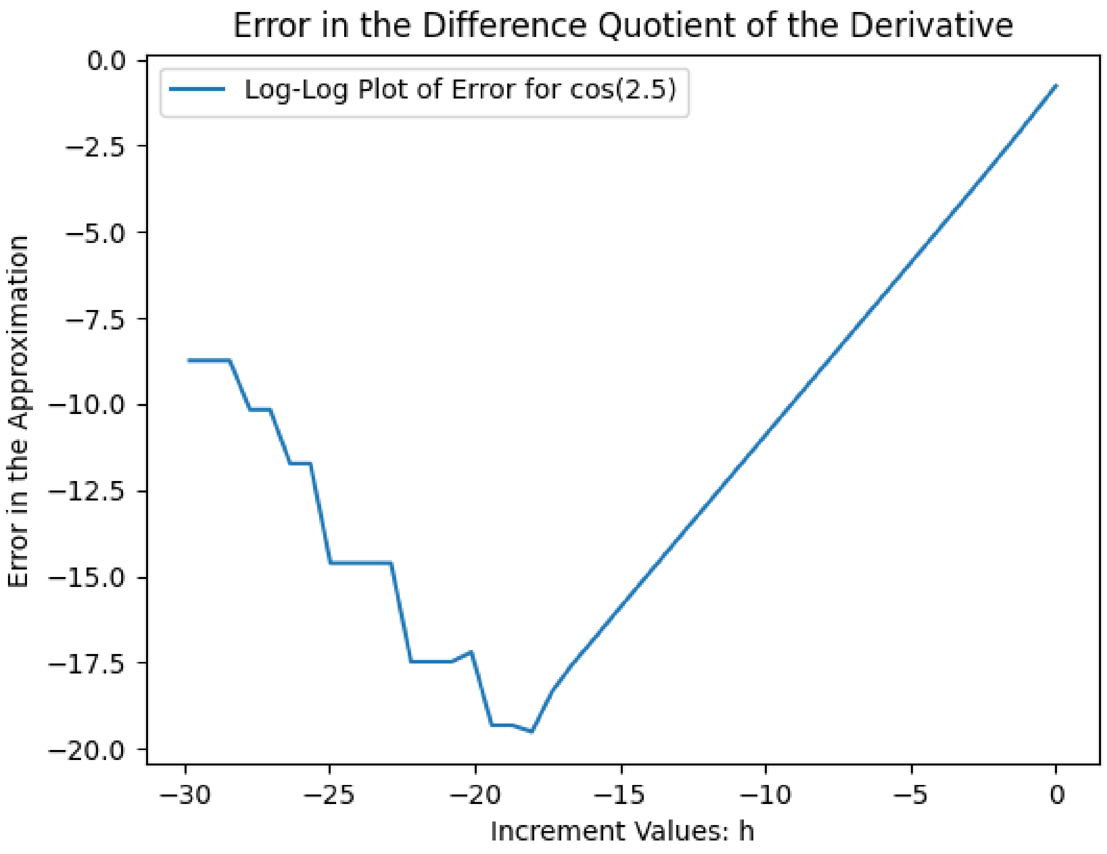

Figure 1: The figure shows how the error is reduced as h approaches zero until the value is too small. The plot is a log-log graph of the data generated by the python script described in this section.