Bradley Green

Math 4610

September 22, 2021

Dr. Koebbe

TaskSheet 3

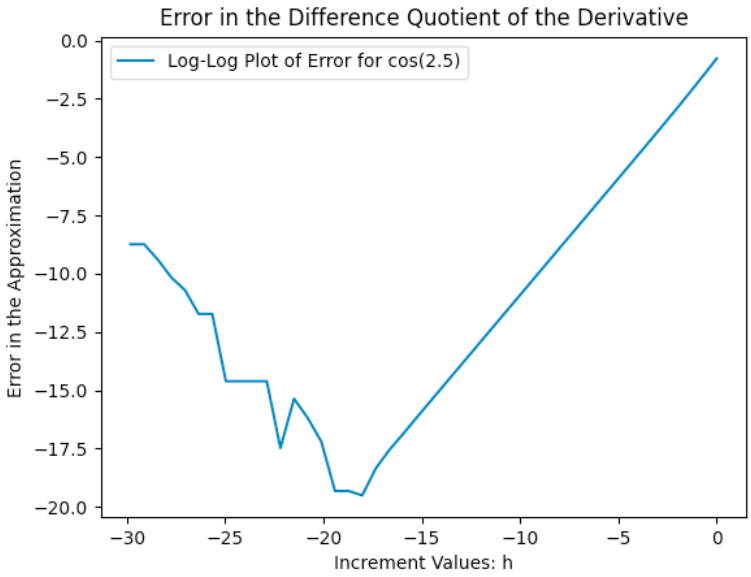
Task 1:

Using the code from task 4 in tasksheet 2, I produced the following output.

Iter	h	Approx	Error
1	1	0.38260348236197916	0.03354335418516324
2	0.5	0.4075490368602161	0.00859779968692631
3	10e-1	0.41580016309240014	0.0003466734547422634
4	10e-2	0.416143368670574	3.4678765684081903e-06
5	10e-3	0.4161468019070469	3.4640095514237856e-08
6	10e-4	0.41614681700608	1.9541062379335727e-08
7	10e-5	0.4161471167662966	-2.802191542139454e-07
8	10e-6	0.41600056732704616	0.0001462692200962512
9	10e-7	0.4385380947269369	-0.022391258179794482
10	10e-8	1.1102230246251563	-0.694076188078014
11	10e-9	55.51115123125782	-55.09500439471068
12	10e-10	0.0	0.4161468365471424
13	10e-11	555111.5123125783	-555111.0961657417
14	10e-12	0.0	0.4161468365471424
15	10e-13	0.0	0.4161468365471424
16	10e-14	-1665334536937.7349	1665334536938.1511
17	10e-15	222044604925031.28	-222044604925030.88
18	10e-16	0.0	0.4161468365471424

Task 2:

The log -log plot shows that the best approximation of the derivative comes at an incremented value h around -18. As h gets smaller than -18 is when the approximation begins to fail due to the finite precision of arithmetic. The approximation can also be seen to be second order because the positive linear slope shown in the graph, has a slope of about 2.



Task 3:

The following is my code for single and double precision machine epsilon

```python
import numpy

def maceps():

    x = numpy.float32(1)
    epsilon = 0.5

    for i in range(100):
        xapprox = x + epsilon
        error = numpy.float32(abs(x - xapprox))

        if error == 0:
            break
        else:
            epsilon = epsilon / 2
            print(i, error)

maceps()
```

```python
def dmaceps():
    x = 1
    epsilon = float(0.5)
    for i in range(100):
        xapprox = float(x + epsilon)
        error = float(abs(x - xapprox))
        if error == 0:
            break
        else:
            epsilon = float(epsilon / 2)
            print(i, error)

dmaceps()
```

Task 4:

I copied the template for creating software manuals from git and used it to write my own documentation for the two routines above. Below is a screenshot of one of the routines.

main math4610 / software_manual / smaceps.md Go to file ···

bgreen55 Update smaceps.md ✓ Latest commit 294064b 9 minutes ago History

1 contributor

51 lines (28 sloc) | 1.47 KB Raw Blame

Routine Name: smaceps

Author: Bradley Green

Language: Python. The code was built using Pycharm. Can be run in terminal by:

```
python3 maceps.py
```

Description/Purpose: This routine will compute the single precision value for the machine epsilon or the number of digits in the representation of real numbers in single precision. This is a routine for analyzing the behavior of any computer. This usually will need to be run one time for each computer.

Input: There are no inputs needed in this case. Even though there are arguments supplied, the real purpose is to return values in those variables.

Output: This routine returns a single precision value for the number of decimal digits that can be represented on the computer being queried.

Usage/Example:

The routine has no arguments. It runs an iteration dividing 1 in half each iteration and adding that number, epsilon, to 1. Once epsilon is so that epsilon plus 1 equals 1 the iteration breaks. Each iteration before hand prints out including the size of epsilon in 32 bit size.

Implementation/Code: The following is the code for maceps()

```
if __name__ == '__main__':

x = numpy.float32(1)
epsilon = 0.5

for i in range(100):
    xapprox = x + epsilon
    error = numpy.float32(abs(x - xapprox))

    if error == 0:
        break
    else:
        epsilon = epsilon / 2
        print(i, error)
```

Last Modified: September/2021

Back

Task 5:

I created the shared library in my math4610 folder and was given the following output in the terminal from the last two commands.

```
Bradleys-MacBook-Air:math4610lib Bradley$ ranlib mylib.a
warning: /Library/Developer/CommandLineTools/usr/bin/ranlib: warning for library
: mylib.a the table of contents is empty (no object file members in the library
define global symbols)
Bradleys-MacBook-Air:math4610lib Bradley$ ar tv mylib.a
rw-r--r--     501/20               8 Sep 27 17:48 2021 __.SYMDEF SORTED
rw-r--r--     501/20             264 Sep 27 17:39 2021 dmaceps.py
rw-r--r--     501/20             288 Sep 27 17:39 2021 maceps.py
```

Task 6:

Shared libraries are beneficial to programmers in that they provide precompiled code that can be reused in other programs by other programmers. Shared libraires can make creating new routines easier and also take up less memory. However, if there is a bug in the shared library, the programmer using the library will have to rewrite their project and have to find a way to do it without the library because they have no access to the library to fix the bugs. The people who maintain the library also will have to reship the entire project back out when they have fixed what bugs were found in the library.

https://www.geeksforgeeks.org/working-with-shared-libraries-set-1/