Palmer Edholm
Dr. Joe Koebbe
MATH 4610
September 17, 2021

Tasksheet 2

Task 1 First, I open Git Bash and navigate to the directory that was created for this class. Next, I create a repository where I will put all the files needed for this tasksheet. Next, I create a new python file named bunny.py that I will edit using vim editor. Figure 1 shows everything that I have done thus far. Once I hit enter, I'm taken to the vim editor. I type "a" into the

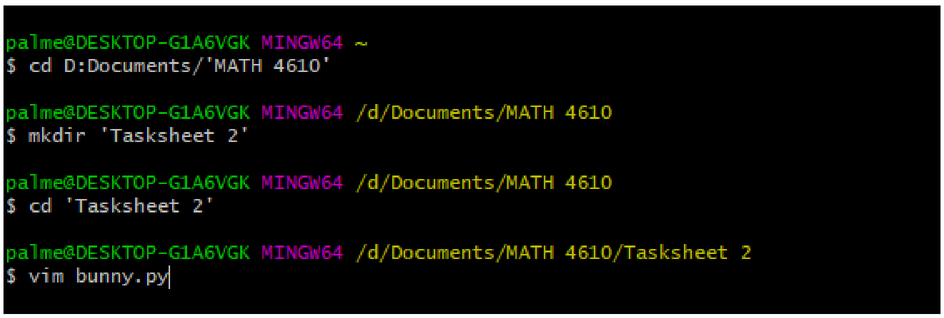

Figure 1: Navigating directories and creating file

vim editor which allows me to edit the python file. I then type the following line of Python code:

```
    print('That\'s the most foul, cruel, and bad tempered rodent\
   you ever set eyes on.')
```

Now I hit escape and type ":x" to save my code and exit out of the vim editor. If I type in the command "ls", I can see the files contained in my repository. I can see that I have a file called bunny.py. To execute this python file, I type in the command "py bunny.py" and Git Bash executes the file. If I did everything correctly, it should display the string inside the print statement in the terminal. Figure 2 shows the past few commands and the output from

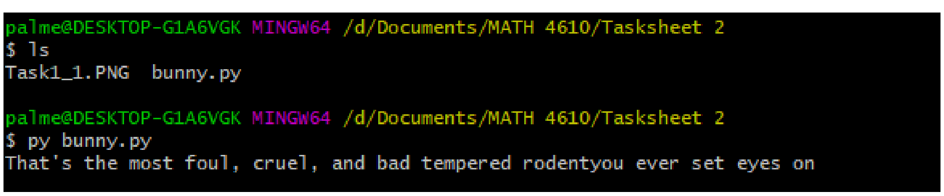

Figure 2: Executing Python file

the bunny.py file. As is evident, I forgot to include a space after splitting the string in the print statement.

Task 2 Figure 3 shows the new home page I created with links to the table of contents for completed homework assignments and the software manual.

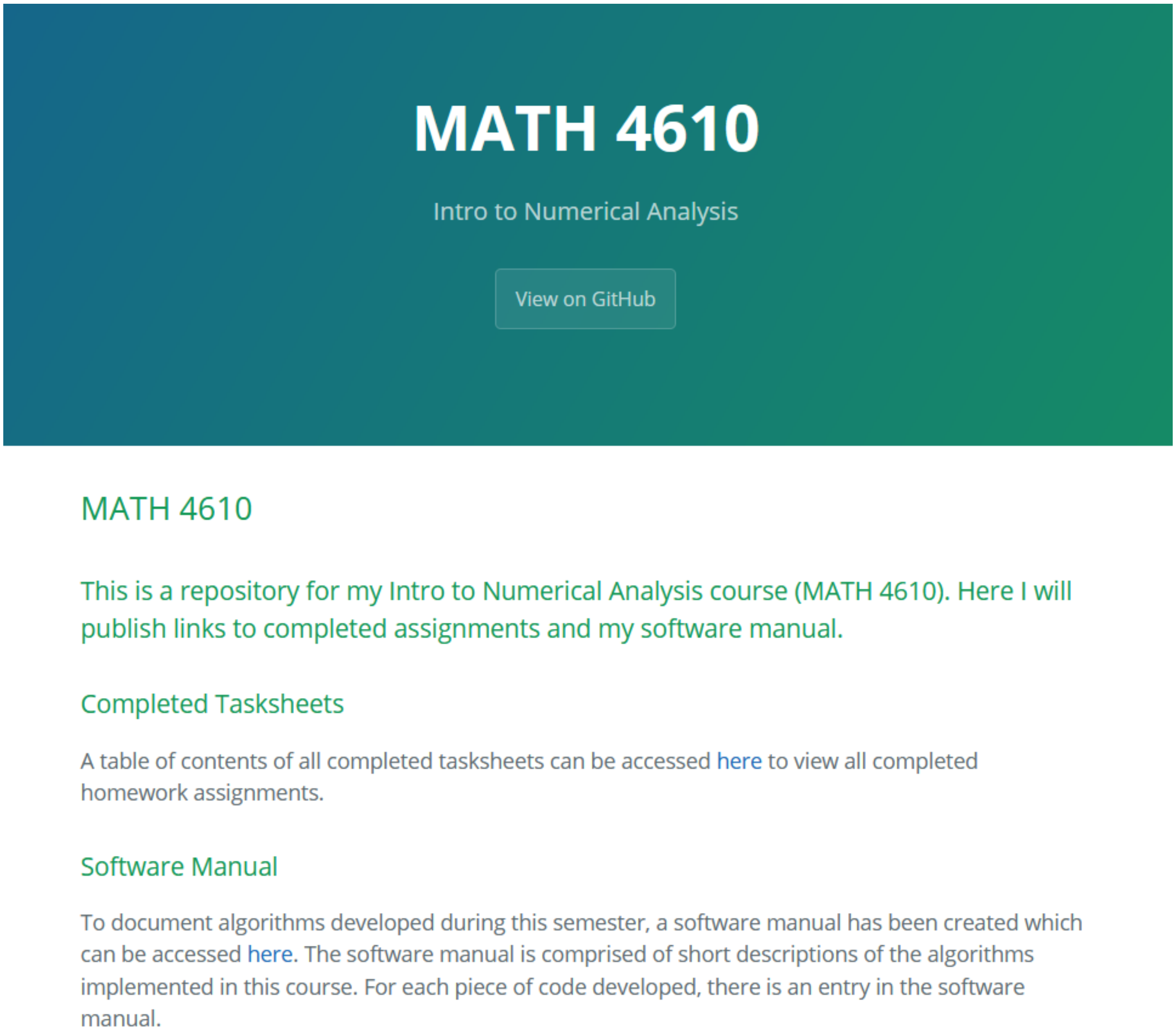

Figure 3: Home page

Task 3 The centered difference approximation for the first derivative is

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h} \tag{1}$$

which is exactly equal to the first derivative as h approaches 0. The Taylor series expansion with remainder is

$$f(a+h) \sim \sum_{k=0}^{N} \frac{f^{(k)}(a)}{k!} h^k + R_f(a, N) \tag{2}$$

where

$$R_f(a, N) = \frac{f^{N+1}(\xi)}{(N+1)!} h^{N+1} \tag{3}$$

and ξ is a point in between a and n. Plugging (1) into (2) and (3), and letting $N = 2$, we get

$$\begin{aligned}
f'(a) &= \frac{1}{2h}\left(f(a) + f'(a)h + \frac{1}{2}f''(a)h^2 + \frac{1}{6}f'''(\xi_1)h^3\right) \\
&\quad - \frac{1}{2h}\left(f(a) + f'(a)(-h) + \frac{1}{2}f''(a)(-h)^2 + \frac{1}{6}f'''(\xi_2)h^3\right) \\
&= \frac{1}{2h}\left(2f'(a)h + \frac{1}{6}(f'''(\xi_1) + f'''(\xi_2))h^3\right) \\
&= f'(a) + \frac{1}{12}h^2 f'''(\xi_3).
\end{aligned}$$

We can therefore see from the previous simplification of the Taylor series with remainder that $|\text{error}| = |\frac{1}{12}h^2 f'''(\xi_3)| \le Ch^2$ and the approximation is second order.

Task 4 First, let's look at the Taylor series expansions of $f(x+h)$ and $f(x-h)$ using (2) where:

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \frac{1}{6}f'''(x)h^3 + \frac{1}{24}f^{(4)}(x)h^4 + \dots \tag{4}$$

$$f(x-h) = f(x) - f'(x)h + \frac{1}{2}f''(x)h^2 - \frac{1}{6}f'''(x)h^3 + \frac{1}{24}f^{(4)}(x)h^4 + \dots \tag{5}$$

We can use these expansions to find the central difference approximation for the second derivative. If we add (4) and (5), we get

$$f(x+h) + f(x-h) = 2f(x) + f''(x)h^2 + \frac{1}{12}f^{(4)}(x)h^4 + \dots$$

Let's move $2f(x)$ to the left-hand side and divide through by h^2:

$$\frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = f''(x) + \frac{1}{12}f^{(4)}(x)h^2 + \dots$$

To simplify things, let's consider the expansion where $N = 3$ and use (3) for the remainder:

$$\frac{f(x+h) - 2f(x) + f(x-h)}{h^2} = f''(x) + \frac{1}{12}f^{(4)}(\xi)h^2. \tag{6}$$

From (6), we can see that the left-hand side is the central difference approximation of the second derivative with error of order h^2.

Using (6), we can run the following code to approximate the second derivative of the function $f(x) = \cos(x)$.

```
    from matplotlib import pyplot as plt
    import numpy as np
    from tabulate import tabulate

    #
    # initialize the exact value of the derivative
    # ------------------------------------------------
    #
    list =[]
```

```
    aval = 2.0
    exactVal = -np.cos(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]) - 2*np.cos(aval) +
              np.cos(aval - h[0]) ) / (h[0]**2)
    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:
        # print(dfVal)
        #
        # 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]) - 2*np.cos(aval) + \
                 np.cos(aval - h[l])
        denom = (h[l]**2)
        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
        # ------------------------
        #
        list.append([l,h[l],error[l],dfVal])
        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(2.0)')
    plt.legend()
    plt.show()
    table = tabulate(list, headers=['iter', 'h', 'error', 'dfVal'],
                     tablefmt='orgtbl')
    print(table)
```

The code produces the following table so we can get an idea of what's happening with our error values and difference quotients as h approaches 0.

iter	h	error	dfVal
1	0.5	0.0085978	0.407549
2	0.25	0.00216292	0.413984
3	0.125	0.000541576	0.415605
4	0.0625	0.000135447	0.416011
5	0.03125	3.3865e-05	0.416113
6	0.015625	8.46646e-06	0.416138
7	0.0078125	2.11663e-06	0.416145
8	0.00390625	5.29157e-07	0.416146
9	0.00195312	1.32289e-07	0.416147
10	0.000976562	3.30163e-08	0.416147
11	0.000488281	7.98698e-09	0.416147
12	0.000244141	1.2349e-09	0.416147
13	0.00012207	4.35304e-09	0.416147
14	6.10352e-05	8.07833e-09	0.416147
15	3.05176e-05	3.78807e-08	0.416147
16	1.52588e-05	1.5709e-07	0.416147
17	7.62939e-06	1.34918e-06	0.416148
18	3.8147e-06	2.30286e-06	0.416149
19	1.90735e-06	6.11755e-06	0.416153
20	9.53674e-07	5.18939e-05	0.416199
21	4.76837e-07	0.000112929	0.41626
22	2.38419e-07	0.000845351	0.416992
23	1.19209e-07	0.00182191	0.417969
24	5.96046e-08	0.00572816	0.421875
25	2.98023e-08	0.0213532	0.4375
26	1.49012e-08	0.333853	0.75
27	7.45058e-09	0.583853	1
28	3.72529e-09	3.58385	4
29	1.86265e-09	15.5839	16
30	9.31323e-10	63.5839	64
31	4.65661e-10	255.584	256
32	2.32831e-10	1023.58	1024
33	1.16415e-10	0.416147	0
34	5.82077e-11	16383.6	16384
35	2.91038e-11	65535.6	65536
36	1.45519e-11	0.416147	0
37	7.27596e-12	1.04858e+06	1.04858e+06
38	3.63798e-12	4.1943e+06	4.1943e+06
39	1.81899e-12	1.67772e+07	1.67772e+07
40	9.09495e-13	0.416147	0
41	4.54747e-13	0.416147	0
42	2.27374e-13	1.07374e+09	1.07374e+09
43	1.13687e-13	4.29497e+09	4.29497e+09

The code also produces figure 4 so we can visualize what's happening better. As we can see,

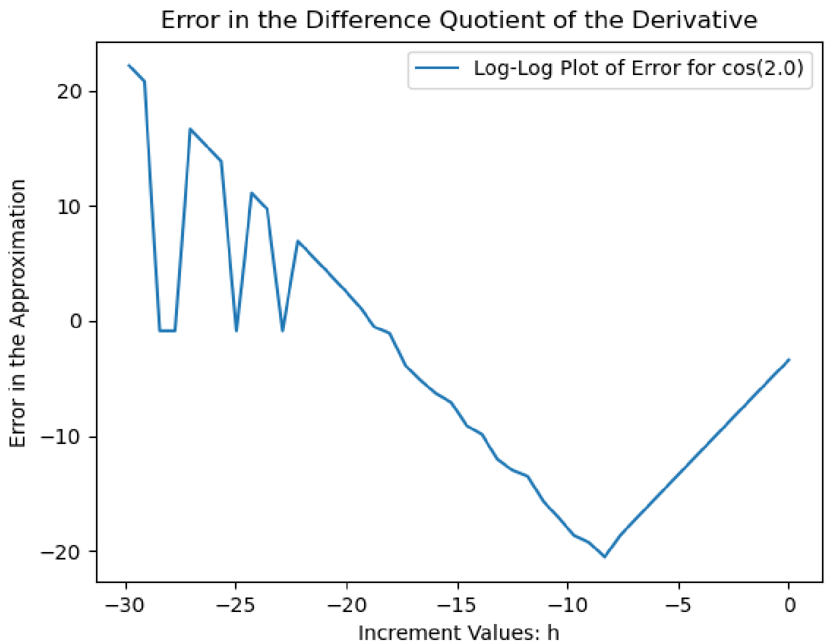

Figure 4: Second derivative approximation

at a point, our approximations start to suffer from catastrophic cancellation.

Task 5 In [1], I found three different finite difference formulas for approximating both first and second derivatives. The first one is called the forward difference approximation and is $O(\Delta x^2)$:

$$f'(x) \approx \frac{-3f(x) + 4f(x+\Delta x) - f(x+2\Delta x)}{2\Delta x} \tag{7}$$

$$f''(x) \approx \frac{2f(x) - 5f(x+\Delta x) + 4f(x+2\Delta x) - f(x+3\Delta x)}{\Delta x^3}. \tag{8}$$

The second one is called the backward difference approximation and is $O(\Delta x^2)$:

$$f'(x) \approx \frac{3f(x) - 4f(x-\Delta x) + f(x-2\Delta x)}{2\Delta x} \tag{9}$$

$$f''(x) \approx \frac{2f(x) - 5f(x-\Delta x) + 4f(x-2\Delta x) - f(x-3\Delta x)}{\Delta x^3}. \tag{10}$$

The last one is the centered difference approximation again, but is $O(\Delta x^4)$ instead of $O(\Delta x^2)$:

$$f'(x) \approx \frac{-f(x+2\Delta x) + 8f(x+\Delta x) - 8f(x-\Delta x) + f(x-2\Delta x)}{12\Delta x} \tag{11}$$

$$f''(x) \approx \frac{-f(x+2\Delta x) + 16f(x+\Delta x) - 30f(x) + 16f(x-\Delta x) - f(x-2\Delta x)}{12\Delta x^2}. \tag{12}$$

References

[1] https://www.dam.brown.edu/people/alcyew/handouts/numdiff.pdf