ME140A - Homework 1

October 8, 2022

Due by 11:59PM, Oct 6th, by email to ameiburg@ucsb.edu. Collaboration is encouraged!

1 Problem 1 - Precision

1.1 (a)

We can use the following code:

```
for m = -7:-2
    for n = 3:6
        x=10.0^m;
        y=10.0^n;
        f = ((x+y)^2-2*x*y-y^2)/(x^2);
        disp(['m=',num2str(m),' n=',num2str(n),' f=',num2str(f)])
    end
end
```

which prints (the exact output may vary depending on your version of MATLAB):

```
m=-7 n=3 f=-11641.5322
m=-7 n=4 f=1490116.1194
m=-7 n=5 f=0
m=-7 n=6 f=0
m=-6 n=3 f=0
m=-6 n=4 f=14901.1612
m=-6 n=5 f=-1907348.6328
m=-6 n=6 f=0
m=-5 n=3 f=0
m=-5 n=4 f=-149.0116
m=-5 n=5 f=0
m=-5 n=6 f=-1220703.125
m=-4 n=3 f=1.0012
m=-4 n=4 f=0
m=-4 n=5 f=190.7349
```

```
m=-4 n=6 f=-12207.0312
m=-3 n=3 f=1
m=-3 n=4 f=0.99838
m=-3 n=5 f=1.9073
m=-3 n=6 f=122.0703
m=-2 n=3 f=1
m=-2 n=4 f=1
m=-2 n=5 f=0.99182
m=-2 n=6 f=1.2207
```

We see that we get our most accurate results when n is small and m is large, or in other words, when m and n are closer. When $n - m \geq 8$, the results are pretty bad. This is because, when n and m are farther apart, x and y are very different scales. Then adding and subtracting them causes more rounding erorrs, and the result does not cancel out to 1 accurately.

1.2 (b)

You should get outputs of 0.0 and 2.1783. The first result is very far from the expected value of 2.1783. This is also because we added a very large number (e^{500}) to a very small number (e^1), and the e^1 terms gets lost. The second calculation is accurate because $e^{500} - e^{500}$ cancels out to zero, and then we add e^1 and get an accurate answer.

When computing a sum from a sorted list, it's more accurate to start from the small end: then we add many smaller numbers into the sum first. By the end, when we're adding larger terms in, our running total has grown larger as well, so we lose less precision to rounding.

2 Problem 2 - Taylor Series

The fastest way to solve this problem is to plug one polynomial into another. To compute $f(g(x))$, you can use, say,

this WolframAlpha query to "Expand 1 + 3/2*(4x – 5xˆ2) – 1/10*(4x–5xˆ2)ˆ2".

We get

$$1 + 6x - \frac{91}{10}x^2 + 4x^3 - \frac{5}{2}x^4).$$

Our original series is only accurate to order $O(x^3)$, so we need to truncate this to that accuracy:

$$\boxed{1 + 6x - \frac{91}{10}x^2 + O(x^3)}$$

An alternate route is to use the definition of a Taylor series. Writing $F = f \circ g$ as our desired function $f(g(x))$, we say

$$F(x) = F(0) + F'(0)x + \frac{F''(0)}{2}x^2 + O(x^3)$$

Then we can work out the three values we need,

$$F(0) = f(g(0)) = f(0) = \boxed{1}$$

$$F'(x) = f'(g(x))g'(x), \quad F'(0) = f'(g(0))g'(0) = f'(0) \cdot 4 = \frac{3}{2} \cdot 4 = \boxed{6}$$

$$F''(x) = (f'(g(x))g'(x))' = (f'(g(x))'g'(x) + f'(g(x))g''(x)$$
$$= (f''(g(x))g'(x))(g'(x)) + f'(g(x))g''(x) = f''(g(x))(g'(x))^2 + f'(g(x))g''(x)$$

$$F''(0) = f''(g(0))(g'(0))^2 + f'(g(0))g''(0) = f''(0) \cdot 4^2 + f'(0) \cdot (-10) = \frac{-1}{5} \cdot 16 + \frac{3}{2} \cdot (-10) = \boxed{\frac{-91}{5}}$$

which we can put in to get the Taylor series,

$$F(x) = 1 + 6x - \frac{91}{10}x^2 + O(x^3)$$

For $(g \circ f)(x)$, plugging one polynomial into the other gives

$$-1 - 9x - \frac{213x^2}{20} + \frac{3x^3}{2} - \frac{x^4}{20}$$

which truncates to

$$\boxed{-1 - 9x - \frac{213x^2}{20} + O(x^3)}$$

3 Problem 3 - Implementing an Integration Rule

We can implement the simple Boole's rule with $f(x) = sin(exp(x))$, and then the composite Boole's rule that calls that function on n subintervals.

After implementing the composite Boole's rule, we can try a few values of n and double it until the values converge to 10^{-5}. Doubling is a more reliable way of estimating error than increasing n one-by-one and seeing how the value changes. (I will provide some brief comments to this effect in class.)

```
composite_boole(0, 1.8, 2)
    0.473551740926354
composite_boole(0, 1.8, 2)
    0.473551740926354
composite_boole(0, 1.8, 4)
    0.4764701050137082
composite_boole(0, 1.8, 8)
    0.4764988606526055
composite_boole(0, 1.8, 16)
    0.47649899501603005
```

Going from $n = 8$ to $n = 6$, the value changed by only 1.3×10^{-7}, so we are within error tolerance. Our final answer is $\boxed{0.4764990}$.

The integration converges quickly because the function is reasonably smooth over this interval. If we look at the function from $x = 0$ to $x = 5$, on the other hand, it has a very high frequency around $x = 5$, and will probably converge much more slowly.

4 Problem 4 - Undetermined Coefficients

We want an integration rule that is exact on the four gives integrals. First, we need their true values:

$$\int_{-1}^{1} 1\,dx = 2, \qquad \int_{-1}^{1} e^x\,dx = e - \frac{1}{e} = 2.3504$$

$$\int_{-1}^{1} e^{2x}\,dx = \frac{1}{2}e^2 - \frac{1}{2e^2} = 3.62686, \qquad \int_{-1}^{1} e^{3x}\,dx = \frac{1}{3}e^3 - \frac{1}{3e^3} = 6.67858$$

Our integration rule will be $c_1 f(x_1) + c_2 f(x_2)$, so write down a system of equations for these constraints:

$$c_1 \cdot 1 + c_2 \cdot 1 = 2$$

$$c_1 e^{x_1} + c_2 e^{x_2} = 2.3504$$

$$c_1 e^{2x_1} + c_2 e^{2x_2} = 3.62686$$

$$c_1 e^{3x_1} + c_2 e^{3x_2} = 6.67858$$

This can be solved numerically (with any numerical solver – here's an example Wolfram Alpha query, it will give you an expression with awful square roots, but you can click on them to get the decimal value) or some fancy algebra (substitute $p = e^{x_1}$ and $q = e^{x_2}$, now it's a system of polynomials you can start to solve). We find the solution:

$$\boxed{c_1 = 0.6800643, \quad c_2 = 1.31994}$$

$$\boxed{x_1 = 0.737758, \quad x_2 = -0.352052}$$

Note that some solvers, like WolframAlpha, will find two solutions: the other solution corresponds to swapping c_1 with c_2, and x_1 with x_2. This produces the same integration rule of course, just with different ordering of labels on our variables.

We can check that it indeed provides exact answers on our four desired integrals. To try it on some *different* integrals, we can try

$$f(x) = x, \qquad \int_{-1}^{1} f(x)\,dx = 0$$

$$\int_{-1}^{1} f(x)\,dx \approx c_1 f(x_1) + c_2 f(x_2) = c_1 \cdot x_1 + c_2 \cdot x_2 = 0.0370285$$

So, on this simple linear integral, we get an error of 0.037. This makes it "worse" than the trapezoidal rule in some regards, but we know it performs well on our exponential integrals that we designed it for.