CSC120 2025S Lab No.6: Taylor Expansions

Instructor: Mitsu Ogihara

Abstract

The goal of this lab is to write a class, `Taylor`, that attempts to compute an approximate value for $\sqrt{x+\delta}$, $\sqrt[3]{x+\delta}$, and $\log(1+\delta)$. The program will use Taylor expansions.

1 Functions and Their Derivatives

The Taylor expansion of a function f at $a+x$ with small x approximates the value of $f(a+x)$ using the value of x and the values of $f(a)$ and the derivatives of f at a. For the approximation to work, the absolute. value of x must be less than 1.

In calculus, a function $f(x)$ is derivative at x is defined as:

$$\lim_{\delta \to 0} \frac{f(x+\delta) - f(x)}{\delta}.$$

This quantity is the "ultimate value of the fraction $\frac{f(x+\delta)-f(x)}{\delta}$ when the value of δ gradually decreases to 0 from positive value." For example, the derivative of $f(x) = x^2$ is:

$$\lim_{\delta \to 0} \frac{(x+\delta)^2 - x^2}{\delta} = \lim_{\delta \to 0} 2x + \delta.$$

The ultimate value of this quantity is $2x$.

We call the process of applying the derivative calculation "differentiation." We can imagine applying differentiation multiple times iteratively; i.e., we can think of the derivative of the derivative, the derivative of the derivative of the derivative, etc. The iterative application is expressed with a superscript of (n), where n is the application number. In other words, for all functions $f(x)$ and all $n \geq 1$, the n-th derivative of $f(x)$ is $f^{(n)}(x)$.

Some functions have a property that for all x in its domain and all integers $n \geq 1$, $f^{(n)}(x)$ exists. Some functions do not have this property.

2 The Taylor Expansions

Let $f(x)$ be a function for all $n \geq 1$, its n-th derivative $f^{(n)}(x)$ exists. Let a and δ be two real numbers, where δ has a small absolute value. For each $n \geq 1$, the n-th Taylor expansion of $f(a+\delta)$ is given by the following formula:

$$f(a+\delta) \approx f(a) + f^{(1)}(a)\frac{\delta}{1!} + f^{(2)}(a)\frac{\delta^2}{2!} + \cdots + f^{(n)}(a)\frac{\delta^n}{n!}.$$

The symbol $\approx$ means "approximately equal to." Using the summation symbol, $\sum$, the formula is simplified as:

$$f(a+\delta) \approx f(a) + \sum_{k=1}^{n} f^{(k)}(a) \cdot \frac{\delta^k}{k!}.$$

If $|\delta|$ is small, the fraction appearing in the summation decreases quickly as the value of n increases, and the approximation becomes very close to the actual value.

3 Using a Taylor Expansion for Calculation

3.1 The Square Root

We know the square root function's first derivative is $(1/2)x^{-1/2}$, its second dervative is $(1/2)(-1/2)x^{-3/2}$, its third derivative is $(1/2)(-1/2)(-3/2)x^{-5/2}$, etc. We notice the following relations:

- Let $c_0 = 1$, $e_0 = 1/2$, and $p_0 = \sqrt{x}$.
- For each $i \geq 1$, define $c_i = c_{i-1} * e_{i-1}$, $e_i = e_{i-1} - 1$, $p_i = p_{i-1}/x$.
- Then for all $i \geq 1$, the i-th derivative is $c_i * p_i$.

Based on these relations, we design the following algorithm for computing the n-th approximation of $\sqrt{a+\delta}$:

- We represent the x by a `double` variable `base`, $\sqrt{x}$ by a `double` variable `square_root`, and and δ by a `double` variable `delta`.
- We represent the approximate value of the square root with a `double` variable named `approx`. We initialize the variable with `Math.sqrt(a)`.
- We use `double` variables `coefficent`, `exponent`, and `power` for c_i, e_i, and p_i.
- We use `long` variable `factorial` for the factorial used in each term. We must use `long` instead of `int` because the factorial value grows very quickly.
- We also use a `double` variable `delta_power` for computing the power series $\delta, \delta^2, \ldots$.
- We initialize `coefficient` with the value of `1.0`, `exponent` with the value of `0.5`, and `power` with `square_root`. We also initialize `factorial` with the value of `1` and `delta_power` with `1.0`.
- Let us use an `int` variable `num_terms` to represent the number of terms and an `int` variable `round` to represent the present round number. The latter variable goes from `1` to `num_terms`.
- At each round, we execute the following computation:
  - We update `coefficient` by multiplying it by `exponent`.
  - We update `exponent` by subtracting `1` from it.
  - We update `power` by dividing it by `base`.

- We update `factorial` by multiplying it by `round`.
- We update `delta_power` by multiplying it by `delta`.
- We update `approx` by adding

  `coefficient * power * delta_power / factorial`.

We can implement this idea as a method that receives `base`, `square_root`, `delta`, and `num_terms`, produces the value of `approx` after an update at each round, and reports the final value of `approx` and the value of `Math.sqrt(base + delta)`.

3.2 The Cubic Root

You can write a code for computing the cubic root by copying the code to approximate the square root, changing the method's name, and giving the following two updates.

- The initial value of `approx` is `Math.pow(a, 1.0/3.0)`.
- The initial value of `coefficient` is `1.0/3.0`.

3.3 The Log Function

The Taylor expansion of the logarithmic function only for $\log(1+\delta)$ with a small δ. The approximation is:

$$\log(1+\delta) \approx \frac{\delta}{1!} - \frac{\delta^2}{2!} + \frac{\delta^3}{3!} - \frac{\delta^4}{4!} + \cdots (-1)^n \frac{x^n}{n!}.$$

We can compute the factorial and the power of δ in the same manner as before, but we do not need to calculate the derivative. We must, however, compute the sign. Alternatively, we can compute the term without the sign, add it to the approximation if the round number is odd, and subtract it otherwise.

4 The Code Design

You must write the code in which the user can select the function to approximate among the three functions: the square root, the cubic root, and the logarithmic function. In the case of the first two, your program must receive three parameters: the base value a, the difference value δ, and the number of terms n. You can use some meaningful variable names for them. In the case of log, we need only δ.

During computing, your program must report the present value of the approximation.

5 Execution Examples

```
% java Taylor
Which approximation do you want to compute?
1. Square Root
2. Cubic Root
3. Logarithm
Enter your choice: 1
```

```
Enter the value of base: 16
Enter the value of delta: 0.05
Enter the number of terms: 20
---- the square root of 16.00000 + 0.05000 -----
Round=01,approx=4.006250000000000
Round=02,approx=4.006245117187500
Round=03,approx=4.006245124816894
Round=04,approx=4.006245124801994
Round=05,approx=4.006245124802026
Round=06,approx=4.006245124802026
Round=07,approx=4.006245124802026
Round=08,approx=4.006245124802026
Round=09,approx=4.006245124802026
Round=10,approx=4.006245124802026
Round=11,approx=4.006245124802026
Round=12,approx=4.006245124802026
Round=13,approx=4.006245124802026
Round=14,approx=4.006245124802026
Round=15,approx=4.006245124802026
Round=16,approx=4.006245124802026
Round=17,approx=4.006245124802026
Round=18,approx=4.006245124802026
Round=19,approx=4.006245124802026
Round=20,approx=4.006245124802026
-----------------------------------------------
  Expected     =4.006245124802026
  Approximation=4.006245124802026


% java Taylor
Which approximation do you want to compute?
1. Square Root
2. Cubic Root
3. Logarithm
Enter your choice: 2
Enter the value of base: 27
Enter the value of delta: 0.2
Enter the number of terms: 20
---- the cubic root of 27.00000 + 0.20000 -----
Round=01,approx=3.007407407407408
Round=02,approx=3.007389117512574
Round=03,approx=3.007389192779631
Round=04,approx=3.007389192407942
Round=05,approx=3.007389192409961
Round=06,approx=3.007389192409950
Round=07,approx=3.007389192409950
```

```
Round=08,approx=3.007389192409950
Round=09,approx=3.007389192409950
Round=10,approx=3.007389192409950
Round=11,approx=3.007389192409950
Round=12,approx=3.007389192409950
Round=13,approx=3.007389192409950
Round=14,approx=3.007389192409950
Round=15,approx=3.007389192409950
Round=16,approx=3.007389192409950
Round=17,approx=3.007389192409950
Round=18,approx=3.007389192409950
Round=19,approx=3.007389192409950
Round=20,approx=3.007389192409950
-----------------------------------------------
  Expected     =3.007389192409950
  Approximation=3.007389192409950


% java Taylor
Which approximation do you want to compute?
1. Square Root
2. Cubic Root
3. Logarithm
Enter your choice: 3
Enter the value of delta: 0.1
Enter the number of terms: 20
---- the logarithm of 1.00000 + 0.10000 -----
Round=01,approx=0.100000000000000
Round=02,approx=0.095000000000000
Round=03,approx=0.095333333333333
Round=04,approx=0.095308333333333
Round=05,approx=0.095310333333333
Round=06,approx=0.095310166666667
Round=07,approx=0.095310180952381
Round=08,approx=0.095310179702381
Round=09,approx=0.095310179813492
Round=10,approx=0.095310179803492
Round=11,approx=0.095310179804401
Round=12,approx=0.095310179804318
Round=13,approx=0.095310179804326
Round=14,approx=0.095310179804325
Round=15,approx=0.095310179804325
Round=16,approx=0.095310179804325
Round=17,approx=0.095310179804325
Round=18,approx=0.095310179804325
Round=19,approx=0.095310179804325
```

```
Round=20,approx=0.095310179804325
-----------------------------------------------
  Expected     =0.095310179804325
  Approximation=0.095310179804325
```