COMPSCI 1JC3
Introduction to Computational Thinking
Fall 2021

Assignment 1

Dr. William M. Farmer and Curtis D'Alves
McMaster University

Revised: Sept 26, 2021

The purpose of Assignment 1 is to write a program in Haskell that computes approximations to trigonometric functions (i.e., cos, sin, and tan). The requirements for Assignment 1 are given below. Please submit Assignment 1 as a single `Assign_1.hs` file to the Assignment 1 folder on Avenue under Assessments / Assignments. Assignment 1 is due **Sunday, October 3, 2021 before midnight.**

Although you are allowed to receive help from the instructional staff and other students, your submitted program must be your own work. Copying will be treated as academic dishonesty!

1 Background

The **Taylor series** of a function $f : \mathbb{R} \to \mathbb{R}$ provides a method of approximating the value of a function at x using methods from calculus. A Taylor series is constructed at some chosen point a, and requires knowledge of the value of the function and its derivatives at this point. Formally, the Taylor series of f at the point a is:

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{1}$$

Note: $f^{(k)}(a)$ denotes the k^{th} derivative of $f(a)$ with $f^{(0)}(a) = f(a)$, and $k!$ denotes the factorial of k, i.e., $k! = 1 * 2 * \cdots * (k-1) * k$.

For most common functions, we have the following equality for x near a:

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{2}$$

As you can see, a Taylor series is an infinite summation. So for common functions, we can approximate $f(x)$ by the n^{th} **Taylor polynomial**:

$$f(x) \approx \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{3}$$

We can increase the accuracy of the approximation of $f(x)$ by either increasing n (the number of iterations) or choosing a point a closer to x.

For example, the trig function sin has the following derivatives at a:

$$\begin{aligned}
f^{(0)}(a) &= \sin(a) \\
f^{(1)}(a) &= \cos(a) \\
f^{(2)}(a) &= -\sin(a) \\
f^{(3)}(a) &= -\cos(a) \\
f^{(4)}(a) &= \sin(a)
\end{aligned}$$

Notice that the fourth derivative is equal to the zero-th derivative (the original function), so all the other derivatives are repetitions of the first four. Therefore, we obtain the following equation:

$$\begin{aligned}
\sin(x) = &\frac{\sin(a)}{0!}(x-a)^0 \\
&+ \frac{\cos(a)}{1!}(x-a)^1 \\
&+ \frac{-\sin(a)}{2!}(x-a)^2 \\
&+ \frac{-\cos(a)}{3!}(x-a)^3 \\
&+ \frac{\sin(a)}{4!}(x-a)^4 \\
&+ \cdots
\end{aligned} \tag{4}$$

where the right-hand side is the Taylor series for cos at a. When $a = 0$, this simplifies nicely to:

$$\sin(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!} \tag{5}$$

1.0.1 Extra Background

The modulus function returns the remainder of a division. It is often used in computing on integers; however, sometimes, when performing numerical methods like computing Taylor series, there is a need for a modulus operation on floating point numbers.

Definition: Let $\mathrm{mod} : \mathbb{R} \to \mathbb{R} \to \mathbb{R}$ be the modulus function (i.e., the remainder of a division) for two real numbers. For example:

$$\begin{aligned}
\mathrm{mod}(3.1, 1.5) &= 0.1 \\
\mathrm{mod}(4\pi, 2\pi) &= 0.0 \\
\mathrm{mod}\left(\frac{3\pi}{2}, \pi\right) &= \frac{\pi}{2}
\end{aligned}$$

1.1 Aside

Feeling worried because you don't know what a Taylor series is and why they can be used to approximate functions? Don't Worry! As computer scientists, we often work with other disciplines, be it math, physics, biology, etc., and rely on explanations from experts in those fields. Accept the knowledge you've been given and apply the skills you have. You've been given a formula for approximating sin. You know what formulas are, how to interpret them, instantiate their variables, and code them up in Haskell. Don't be scared by something you haven't seen before. You have all you need to complete this assignment.

2 Assignment 1

The purpose of this assignment is to compute approximations for the trig functions sin, cos, and tan. Do so by **carefully** following these requirements:

2.1 Requirements

1. Download from Avenue `Assign1_Project_Template.zip` which contains the Stack project files for this assignment. Modify the `Assign_1.hs` in the `src` folder so that the following requirements are satisfied.

2. Your name, the date, and "Assignment 1" are in comments at the top of your file. `macid` is defined to be your MacID. **Note**: Your MacID is not your student number. Your student number is a number, while your MacID is what you use use to sign into systems like Avenue and Mosaic.

3. The file includes a function `sinTaylor` of type `Double -> Double -> Double -> Double -> Double` that takes the values a, $\cos(a)$, $\sin(a)$ and x as input and computes the 4^{th} Taylor polynomial approximation of $\sin(x)$ at a (i.e., Equation 4 with 5 iterations on the right-hand side).

4. The file includes a function `fmod` of type `Double -> Double -> Double` that computes mod (see the Background section for details).

5. The file includes a function `sinApprox` of type `Double -> Double` that computes an approximation of $\sin(x)$ using `sinTaylor` with appropriate values for a, $\cos(a)$, $\sin(a)$, and y as specified by the following table:

Value of x	**Inputs for `sinTaylor`**			
	a	$\cos(a)$	$\sin(a)$	y
$0 \leq \text{mod}(x, 2\pi) < \frac{\pi}{4}$	0	1	0	$\text{mod}(x, 2\pi)$
$\frac{\pi}{4} \leq \text{mod}(x, 2\pi) < \frac{3\pi}{4}$	$\frac{\pi}{2}$	0	1	$\text{mod}(x, 2\pi)$
$\frac{3\pi}{4} \leq \text{mod}(x, 2\pi) < \frac{5\pi}{4}$	π	-1	0	$\text{mod}(x, 2\pi)$
$\frac{5\pi}{4} \leq \text{mod}(x, 2\pi) < \frac{7\pi}{4}$	$\frac{3\pi}{2}$	0	-1	$\text{mod}(x, 2\pi)$
$\frac{7\pi}{4} \leq \text{mod}(x, 2\pi) < 2\pi$	2π	1	0	$\text{mod}(x, 2\pi)$

6. The file includes a function `cosApprox` of type `Double -> Double` that uses `sinApprox` and the fact that

   $$\cos(x) = -\sin\left(x - \frac{\pi}{2}\right)$$

   to approximate the value of $\cos(x)$.

7. The file includes a function `tanApprox` of type `Double -> Double` that uses `cosApprox`, `sinApprox`, and the fact that

   $$\tan(x) = \frac{\sin(x)}{\cos(x)}$$

   to approximate the value of $\tan(x)$.

8. Your file loads successfully into GHCi and all of your functions perform correctly.

2.2 Testing

We provide you with tests that can be run via the stack tests command. However, when debugging students should be aware of how to make their own test cases, as the tests provided will be difficult to decipher on their own. It is recommended that students compare their implementations of `cosApprox`, `sinApprox`, and `tanApprox` to `Prelude.cos`, `Prelude.sin` and `Prelude.tan` as appropriate.

Note, when dealing with floating point numbers there will always be floating point error, so certain computations that you would expect to return the same result will differ slightly. What's important is that the amount of error is within a reasonable tolerance (i.e. for `sinApprox` the test cases will test if the result is within a tolerance of 1×10^{-2} of `Prelude.sin`) as to pass the test cases.