Derivatives

Ayda Ghalkhanbaz

Spring Term 2024

Introduction

The first assignment is mostly about implementing a program that calculates the derivative of a function, but also gives an introduction into the Elixir programming language. In this report, different parts of the program are discussed and as an example of the program's functionality, the result of a test run is presented in a later section.

Method

The assignment instructions have already provided a set of derivative rules. Given that the implementation of these rules was thoroughly explained in the lecture videos, this section will focus on discussing the implementation of the remaining rules.

The remaining derivative rules that complete the program are as follows:

1. $\frac{d}{dx} f^n = n \cdot f^{n-1}$
2. $\frac{d}{dx} \ln(f) = \frac{1}{f} \cdot \frac{df}{dx}$
3. $\frac{d}{dx} \left(\frac{1}{f}\right) = -\frac{1}{f^2} \cdot \frac{df}{dx}$
4. $\frac{d}{dx} \sqrt{f} = \frac{1}{2\sqrt{f}} \cdot \frac{df}{dx}$
5. $\frac{d}{dx} \sin(f) = \cos(f) \cdot \frac{df}{dx}$
6. $\frac{d}{dx} \cos(f) = -\sin(f) \cdot \frac{df}{dx}$

The program's skeleton began to take shape by first defining a function `derive\2` which takes two arguments: a function to be derived and the variable of the differentiation. This single function utilises pattern matching, i.e. using the `case...do...end` construct.

The first argument of the function goes through a set of pattern checking till it hits a match, then the program follows the corresponding rule. As an

example, the following code illustrates the derivative rules mentioned earlier in this section.

```
def derive(func, var) do
    case func do
    {:div, e1, e2} -> {:div, {:sub, {:mul, derive(e1, var), e2},
        {:mul, e1, derive(e2, var)}}, {:exp, e2, {:num, 2}}}

      {:exp, e, {:num,n}} -> {:mul,
      {:mul, n, {:exp, e, {:num, n-1}}}, derive(e,var)}

      {:ln, e} -> {:div, derive(e, var), e}

      {:sqrt,e} -> {:div, derive(e, var),
      {:mul, {:num, 2}, {:sqrt, e}}}

      {:sin, e} -> {:mul, derive(e, var), {:cos, e}}

      {:cos, e} -> {:mul, {:mul, {:num, -1}, derive(e, var)},
      {:sin, e}}
       end
  end
```

The next step was to code several functions in order to simplify the expressions since they had a weird and complex format. The `simplify\1` functions employ some helper functions to make the formatting of the expressions short and concise. This may include removing unnecessary 0's and 1's in certain arithmetical operations.

Finally, several `print\1` functions were implemented in order to print the expressions in a readable formatting.

Result

The following function was executed to test the program:

$$f(x) = 2x^2 + \frac{1}{\sqrt{x^3}} + \ln(\cos(x)) \cdot \sin(4 + x^3)$$

The result of executing the test function is shown below.

```
iex(64)> Derivative.test()
expression: 2 * x ^ (2) + 1 / sqrt(x ^ (3)) +
```

```
ln(cos(x)) * sin(4 + x ^ (3))
derivative: 0 * x ^ (2) + 2 * 2 * x ^ (1) * 1 + 0 * sqrt(x ^ (3))
- 1 * 3 * x ^ (2) * 1 / 2 * sqrt(x ^ (3)) / sqrt(x ^ (3)) ^ (2)
+ -1 * 1 * sin(x) / cos(x) * sin(4 + x ^ (3)) + ln(cos(x)) * 0
+ 3 * x ^ (2) * 1 * cos(4 + x ^ (3))

simplified: 2 * 2 * x +
3 * x ^ (2) / 2 * sqrt(x ^ (3)) / sqrt(x ^ (3)) ^ (2)
+ -1 * sin(x) / cos(x) * sin(4 + x ^ (3)) +
ln(cos(x)) * 3 * x ^ (2) * cos(4 + x ^ (3))
```

The derivative of the function is:

$$
\begin{aligned}
&0 \cdot x^2 + 2 \cdot 2 \cdot x^1 \cdot 1 + 0 \cdot \sqrt{x^3} \\
&- \frac{1 \cdot 3 \cdot x^2 \cdot 1}{2 \cdot \sqrt{x^3}} - \frac{\sin(x)}{\cos(x)} \cdot \sin(4 + x^3) + \ln(\cos(x)) \cdot 0 + 3 \cdot x^2 \cdot 1 \cdot \cos(4 + x^3) \\
&= 4x - \frac{3x^2}{2\sqrt{x^3}} - \tan(x) \cdot \sin(4 + x^3) + 3x^2 \cdot \cos(4 + x^3)
\end{aligned}
$$

which may seem complicated but it is actually the derivative of the function.

Below is the simplified version which is accurate compared to the outcomes of the run test:

$$
4x - \frac{3x^2}{2\sqrt{x^3}} + 3x^2 \cdot \cos(4 + x^3) - \tan(x) \cdot \sin(4 + x^3)
$$