Contents

1 Setup

```
using Pkg;
Pkg.activate("~/EEL-USP/Symbolics/")
Pkg.add("Plots")
Pkg.add("Symbolics")
Pkg.add("StaticArrays")
```

```
using Symbolics
using Plots
using StaticArrays
```

2 Escrevendo as derivadas simbólicas

Vamos criar duas funções, a `derivative` e a `derivative_symb` as quais computam tanto o valor numérico, quanto retornam a expressão simbólica (literal) e uma função chamável em Julia.

Se não ficou claro, acompanhe o tutorial.

2.1 Funções ajudantes

Criaremos as funções ajudantes, utilizando do pacote `Symbolics.jl`. Os detalhes do porquê essas funções são definitas dessa maneira vão além do escopo do tutorial, mas deixamos aqui como documentação.

```
function derivative(exp, ξ)
      @variables y
      Dy=Differential(y)
      ϕe = exp(y) # (
      return first(substitute.(expand_derivatives(Dy(ϕe)),(Dict(y => ξ),)))
  end
```

```
function derivative_symb(exp)
    @variables ξ
    Dξ=Differential(ξ)
    ϕe = exp(ξ) # (
    return (eval(build_function(expand_derivatives(Dξ(ϕe)), ξ)),
    expand_derivatives(Dξ(ϕe)))
end
```

2.2 Como avaliar uma derivada

Consideraremos a função $f(x)$, a seguir, como exemplo,

$$f(x) = 0.1\ \sin(x) + 2\ \sin(x)^2 - x^2 \tag{1}$$

Em Júlia,

```
f(x) = -x^2 + 0.1*sin(x) + 2*sin(x)^2
```

2.2.1 Avaliando numericamente a derivada

Avaliemos, como exemplo, a derivada em $x = 1.3$. Isto é,

$$\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{(x=1.3)} = g(1.3) \tag{2}$$

Em que g é uma função derivada.
De forma literal, a função é equivalente a:

$$
\begin{aligned}
\left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{(x=1.3)} &= \left.\left[\frac{\mathrm{d}(-x^2)}{\mathrm{d}x} + \frac{\mathrm{d}(0.1\ \sin(x))}{\mathrm{d}x} + \frac{\mathrm{d}(2\ \sin(x)^2)}{\mathrm{d}x}\right]\right|_{(x=1.3)} \\
\Leftrightarrow &= \left.\left[-2x + 0.1\ \cos(x) + 2.\frac{\mathrm{d}(\ \sin(x)^2)}{\mathrm{d}\sin(x)}.\frac{\mathrm{d}(\sin(x))}{\mathrm{d}x}\right]\right|_{(x=1.3)} \qquad (3) \\
\Leftrightarrow \left.\frac{\mathrm{d}f}{\mathrm{d}x}\right|_{(x=1.3)} &= \left.[-2x + 0.1\ \cos(x) + 2.(2\sin(x)).\cos(x)]\right|_{(x=1.3)}
\end{aligned}
$$

Ou seja, gostaríamos de obter a função g, $g(x) = -2x + 0.1\ \cos(x) + 4\sin(x).\cos(x)$ com ferramentas computacionais.

Em `Julia`, temos,

```
derivative(f, 1.3)
```

2.2.2 Avaliando a derivada simbolicamente

A vantagem de avaliarmos a derivada simbolicamente é que podemos obter uma expressão literal para derivada e ao mesmo tempo definir uma função em `Julia` a qual é equivalente a função derivada.

Isto é, definiremos a função g em Sec. numérica, e saberemos qual é sua expressão simbólica.

A função `derivative_symb` nos retorna dois valores em uma tupla:

```
derivative_symb(f)
```

O primeiro valor da tupla é como `Julia` representa uma função anônima (explícita, mas sem nome) que pode ser passado para uma função com nome, como por exemplo g.

O segundo valor é a expressão da derivada literal, utilizando-se da regra da cadeia. Note que o símbolo ξ foi utilizado para denotar a variável, na expressão (em nossa f a variável independente é x).

1. Definindo a função-derivada internamente em Julia

   Tudo que precisamos fazer, agora, é definir a função-derivada g, passando o primeiro argumento retornado de `derivative_symb`.

```
g = first(derivative_symb(f))
```

Avaliano $g(1.3)$, temos:

```
g(1.3)
```

2. Obtendo uma expressão, em LaTeX, de g Existe um pacote em `Julia` chamado `Latexify.jl`. Podemos utilizá-lo para renderizar o segundo termo retornado de `derivative_symb`.

```
Pkg.add("Latexify")
```

```
using Latexify
```

```
latexify(last(derivative_symb(f)))
```

Essa expressão, quando renderizada, fica:

$$- 2\xi + 0.1 \cos\left(\xi \right) + 4 \cos\left(\xi \right) \sin\left(\xi \right) \tag{4}$$

3 Tangente à uma curva $\mathbb{R}^2$ (plot)

4 Outros materiais

4.1 CalculusWithJulia

`CalculusWithJulia.jl` utiliza SymPy (interoperabilidade com a biblioteca em Python) para derivar todos seus resultados.

Esse link `https://docs.juliahub.com/CalculusWithJulia/AZHbv/0.0.5/` é um ótimo recurso para se aprender cálculo, utilizando-se de Julia.

4.2 Manim

Biblioteca de animação em Python utilizado para explicar matemática, em vídeo. Inicialmente, desenvolvido por 3Blue1Brown.