



Universidade dos Açores

Cálculo II

Projeto Derivadas

Relatório de Projeto

Junho de 2020

Docente:

Prof. Doutor Paulo Medeiros

Realizado por:

Hélder F. M. de M. Braga

Índice

Introdução.....	6
Ferramentas Utilizadas.....	7
Funcionalidades.....	9
Código.....	18
Possíveis Melhorias.....	20
Conclusão.....	21
Webgrafia.....	22

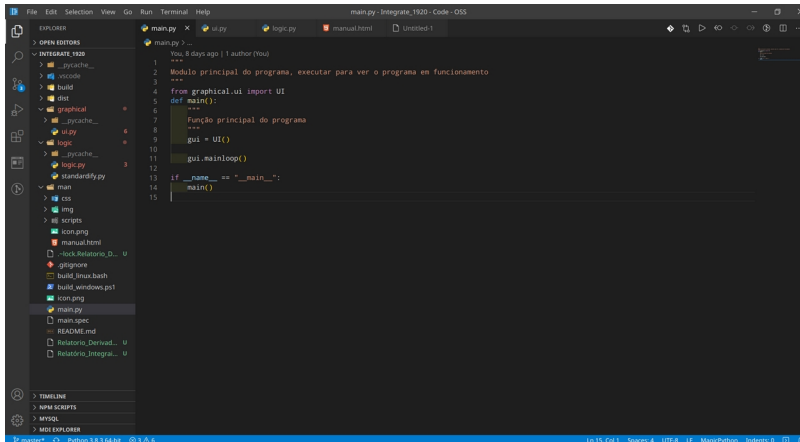
Introdução

Com a realização deste projeto pretendeu-se desenvolver um programa informático na linguagem de programação *Python* capaz de resolver alguns dos problemas relativos ao ramo do calculo de derivadas da disciplina de Calculo II. Além desta aplicação, foi também criado um texto histórico relativo à descoberta das metodologias do calculo diferencial e problemas de derivação.

Ferramentas Utilizadas

Quanto as ferramentas utilizadas, temos as seguintes:

- Programa IDE: *Microsoft Code*



- Texto Histórico e Manual: *Bootstrap, CSS, HTML, Javascript*

ManualDefiniçõesHist. Calc. Dif. e DerivadaHist. Integrar

Est. 2020

INTEGRATE

Manual do programa

Atente não ser mostrados alguns exemplos de funções suportadas pelo programa e qual a formatação a utilizar.
O formato utilizado será (Representação matemática) -> (Representações vetoriais no programa)
Condições
 $\bullet \rightarrow E$
 $\bullet \rightarrow p$
Símbolos
 $\bullet \rightarrow 00$ ou $\bullet \rightarrow 00$
 $\bullet \rightarrow 00$
Polinomial
 $x^2 \rightarrow x^2$ ou x^2 ou x^2
 $x^2(1) \rightarrow x^2(1)$ ou $x^2(1)$
 $x^2(1) \rightarrow x^2(1)$ ou $x^2(1)$
 $x \rightarrow x(1)$ ou $x(1)$
 $x(1) \rightarrow x(1)$ ou $x(1)$

História do Cálculo Diferencial e da Derivada

Introdução

O problema de encontrar a tangente de uma curva foi estudado por vários matemáticos, dos quais se destacam Arquimedes de Siracusa, Zu Chongzhi no período dinastia do Tão Han, Adriano Paavo de LaHaye na Renascença Italiana, e matemáticos do século XIX como Newton e Leibniz. A primeira tentativa elaborada de determinar a tangente de uma curva semelhante à metodologia moderna de Cálculo, veio de Gilles Perreux de Roberval, na década de 1630 a 1640, no entanto, quem na mesma época em que Roberval desenvolveu o seu método, Pierre de Fermat usou a noção de "tangent" e de "inflectional" para encontrar a tangente de uma curva. Alguns historiadores creditam assim Fermat com a descoberta da derivada, mas não foi, no entanto, até Leibniz e Newton terem rigorosamente definido o seu método de tangentes, que uma técnica generalizada tornou-se amplamente aceita.

Método de Roberval de linhas tangentes usando movimento instantâneo

A ideia principal da metodologia de Roberval para determinar a tangente de uma curva consiste, em grande parte, na noção de "movimento instantâneo". Ou seja, considerando-se o desenho de uma curva num ponto em movimento, ou, em algum ponto da curva os vetores que descrevem o movimento desta podem ser decompostos, em x e y , simplesmente, a contribuição da soma destes movimentos.

Este método foi aplicado por Roberval para encontrar as tangentes de curvas para as quais ele foi capaz de determinar os vetores referentes ao movimento num ponto. Para uma parábola, Roberval conseguiu determinar estes vetores, exemplificados abaixo na figura.

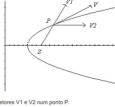


Gráfico de uma parábola, representando o movimento dos vetores V_1 e V_2 num ponto P.

Roberval determinou que num ponto P de um parábola, existem dois vetores que indicam o valor do seu movimento instantâneo, V_1 , que está na mesma direção da semi-retta SP, focando-se na parábola no ponto S e a ponto P, sendo V_2 , por sua vez, o indicador do movimento instantâneo perpendicular ao eixo x e a direção da interseção da parábola. A tangente do gráfico no ponto P é simplesmente a soma dos vetores $V = V_1 + V_2$.

Desde então, Roberval foi capaz de encontrar as tangentes de curvas muito mais complexas incluindo elipses e hipérbolas, no entanto, encontrar os vetores de movimento instantâneo num ponto mostrou-se difícil para um grande número de curvas, além disso, Roberval nunca foi capaz de explicar o funcionamento deste método logicamente e, por isso mesmo, este foi historicamente sempre visto como o precursor do método de encontrar tangentes com o auxílio das infinitesimais, desenvolvidas por Edward mais tarde.

- Programação:
 - Bootstrap
 - CSS
 - JavaScript
 - HTML
 - *Python 3.8.3*
 - Modulos:
 - *Matplotlib* → Criação de gráficos
 - *Numpy* → Lógica matemática
 - *Sympy* → Lógica matemática calculo diferencial
 - *PyInstaller* → Criação de executavel da aplicação
 - *Tkinter* → Interface gráfica
- Sistema(s) Operativo(s):
 - *Windows 10*
 - *Linux*
- Criação de executáveis:
 - *Modulo PyInstaller*
- Criação Setup:
 - *Nullsoft Scriptable Install System (NSIS)*
- Controlo de versões:
 - *Git*
 - *Github*

Funcionalidades

Irei agora referir quais as funcionalidades implementadas no programa, assim como proceder a demonstração de exemplos do uso das mesmas de forma a facilitar o seu entendimento.

Pontos Desenvolvidos

- Mostrar o gráfico de uma função $f(x)$
- Mostrar a reta tangente de $f(x)$ num determinado ponto
- Calculo e representação da primeira derivada $f'(x)$ de uma função $f(x)$
- Mostrar o gráfico da primeira derivada $f'(x)$ de uma função
- Mostrar a reta tangente da primeira derivada $f'(x)$ de $f(x)$ num determinado ponto
- Calculo e representação da segunda derivada $f''(x)$ de uma função $f(x)$
- Mostrar o gráfico da segunda derivada $f''(x)$ de uma função $f(x)$
- Mostrar a reta tangente da segunda derivada de $f''(x)$ num determinado ponto
- Interface de utilizador gráfica com teclado
- Introdução de funções em modo de texto pelo utilizador
- Manipulação/customização de gráficos pelo utilizador
- Resumo da história da derivação e do calculo diferencial em formato de página web
- Manual de programa em formato web

Exemplos de uso

Função $f(x)$

Uma das funcionalidades do programa é mostrar o gráfico de uma função introduzida, como podemos ver na imagem com o exemplo da função $f(x)=x^3$.

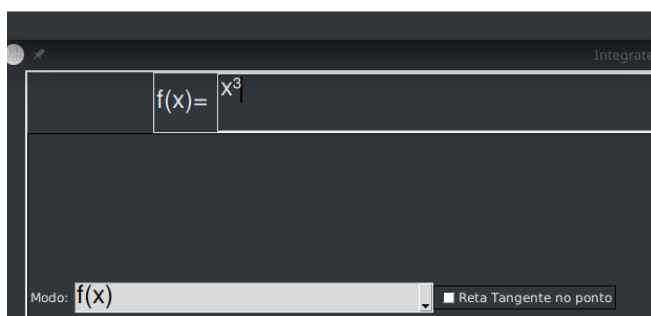
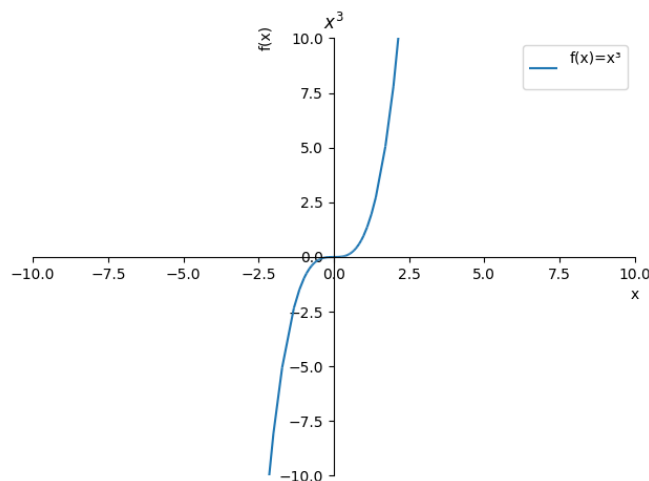


Figure 1: Gráfico da função x^3

A escala pode ser ajustada caso necessário, por exemplo, se quisermos outra “view” da mesma função, para tal basta definir as coordenadas mínimas e máximas de x e y como demonstrado na imagem.

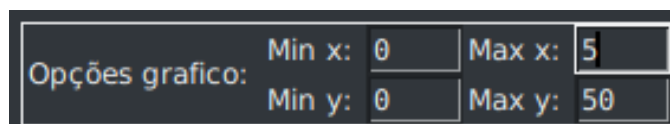


Figure 2: Opções do gráfico

Ficamos assim com a mesma função anterior, mas agora “cortada” para os valores de “0” a “5” para as abcissas e “0” a “50” para ordenadas

Além do ajuste da “*viewport*”, também é possível traçar a reta tangente num ponto da função, para tal, basta introduzir os valores do ponto na reta.

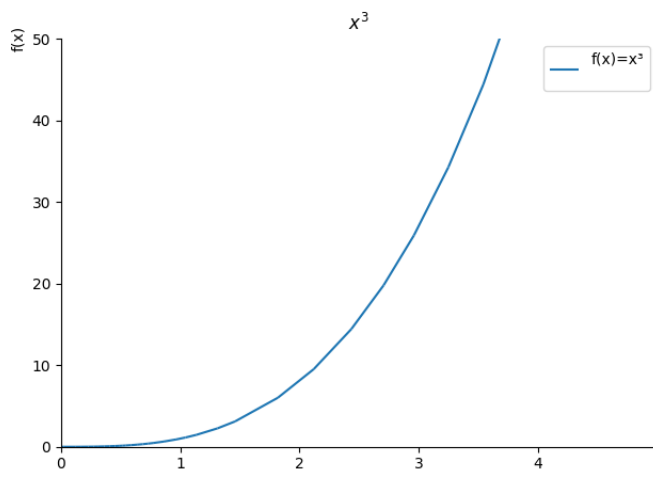


Figure 3: Gráfico função x^2 com “*viewport*” customizada

Obtemos agora um dos pontos da função como demonstrado na imagem. Neste exemplo, usaremos o ponto $P(2.005, 8.465)$.

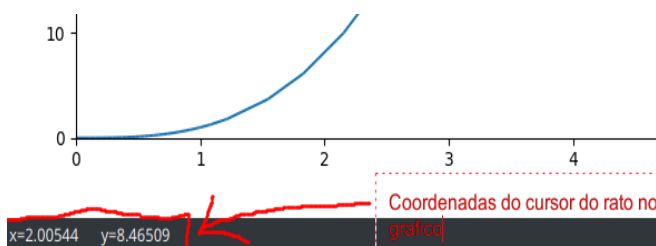


Figure 4: Coordenadas de um dos pontos de x^3

Introduzimos agora o ponto no campo relativo a reta tangente como pode ser visto a direita.

	x: 2.005
<input checked="" type="checkbox"/> Reta Tangente no ponto	y: 8.465
Margem erro:	

Figure 5: Opções Reta Tangente

O que nos dará a seguinte representação gráfica...

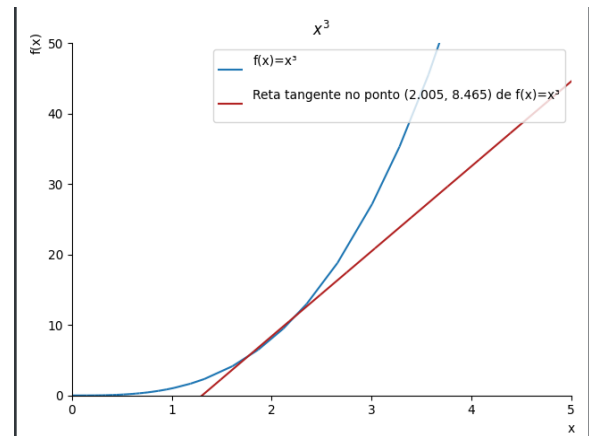


Figure 6: Gráfico da função x^3 com reta tangente num ponto

A margem de erro por predefinição é de 0.05, no entanto, podemos ajustá-la caso se necessário, quanto maior o valor introduzido na margem de erro, menos precisa será a reta tangente demonstrada no gráfico, no entanto, isto permite que no caso de existirem pontos não inteiros, por exemplo, $P(2.00544, 8.4509)$, o utilizador possa introduzir um ponto aproximado, como, $P(2.005, 8.4509)$ ou $P(2.0, 8.4)$ que terá uma representação gráfica semelhante, sem ser necessário uma precisão ao nível das milésimas.

Caso queira-mos grande precisão relativamente ao ponto para a reta tangente, podemos fazer como demonstrado na figura, no entanto, se o ponto não estiver dentro da margem de erro, surge uma mensagem de erro.

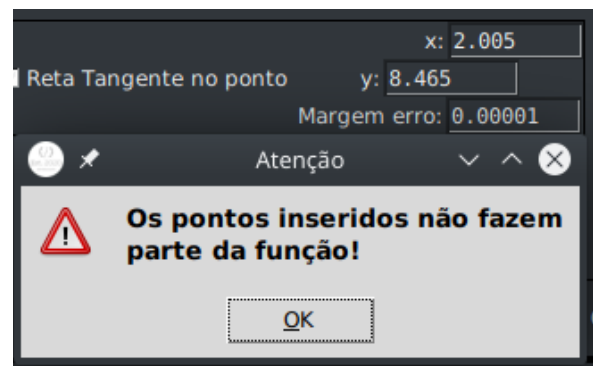


Figure 7: Reta tangente com grande precisão

Alternativamente, se precisão do ponto da reta tangente não for crucial, podemos por exemplo introduzir.

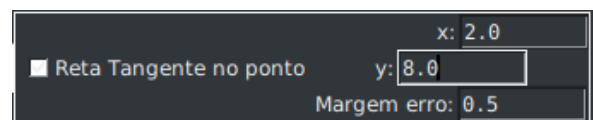


Figure 8: Reta tangente com pouca precisão

O que nos dará no gráfico...

Neste caso específico, a diferença entre o caso da “Figura 6” e da “Figura 9” é mínima, mas, tal pode nem sempre ser verdade, pelo que é aconselhável manter a margem de erro o mais pequena possível para evitar resultados incorretos.

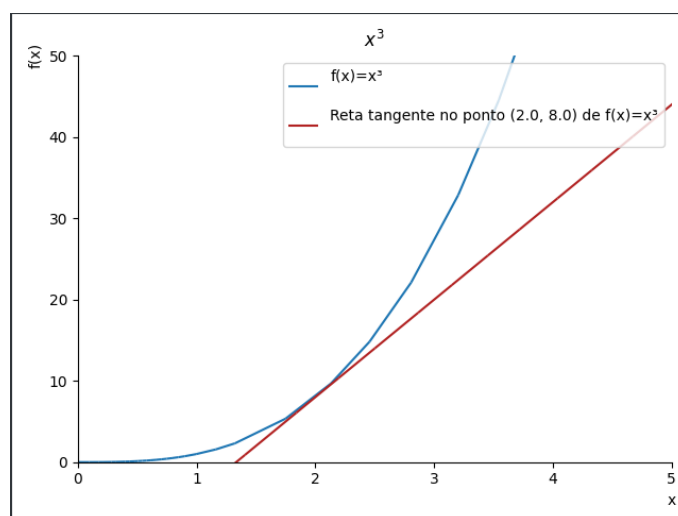


Figure 9: Reta tangente no ponto $P(2, 8)$

Um exemplo “extremo” de uma margem de erro demasiado generosa pode ser visto na “Figura 10”, o que nos dá resultados erróneos.

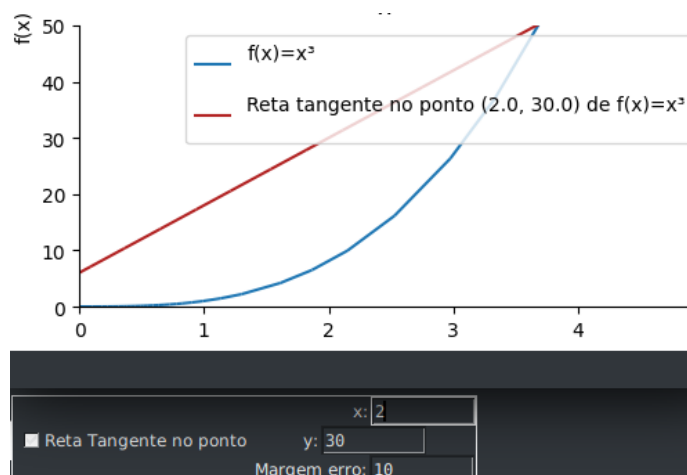


Figure 10: Reta tangente com margem de erro exagerada

Agora que já mostramos a manipulação de diversos aspetos dos gráficos da aplicação assim como expandi-mos no funcionamento da funcionalidade da tangente, passemos à primeira derivada da função x^3 .

Função $f'(x)$

Para obter a primeira derivada da função, começa-mos por seleccionar o modo “ $f'(x)$ ” no programa.

Ao clicar agora no botão “Resolver” o programa irá calcular a derivada de “ x^3 ” e mostrar o gráfico da mesma.

Como podemos ver na “*Figura 12*”, a função derivada de “ x^3 ” é “ $3x^2$ ”.

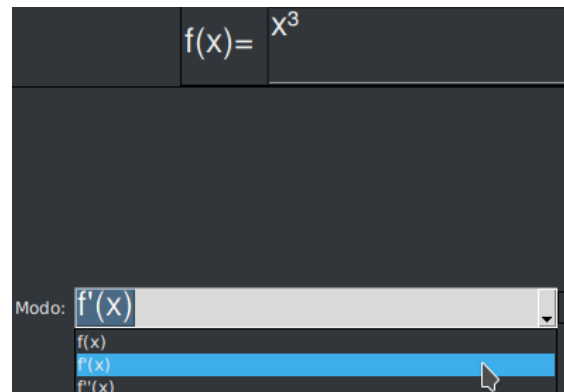


Figure 11: Modo $f'(x)$

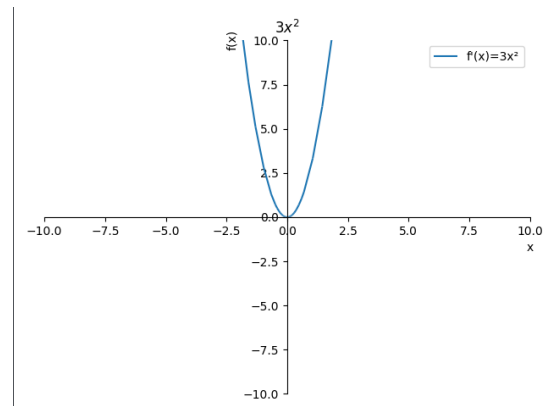


Figure 12: $f'(x)$ da função x^3

Caso queira-mos novamente a reta tangente num ponto para a função derivada, repetimos o processo que efetuamos na secção da função “ $f(x)$ ”.

Usando o ponto $P(1.295, 5.004)$ como exemplo obtemos a figura...

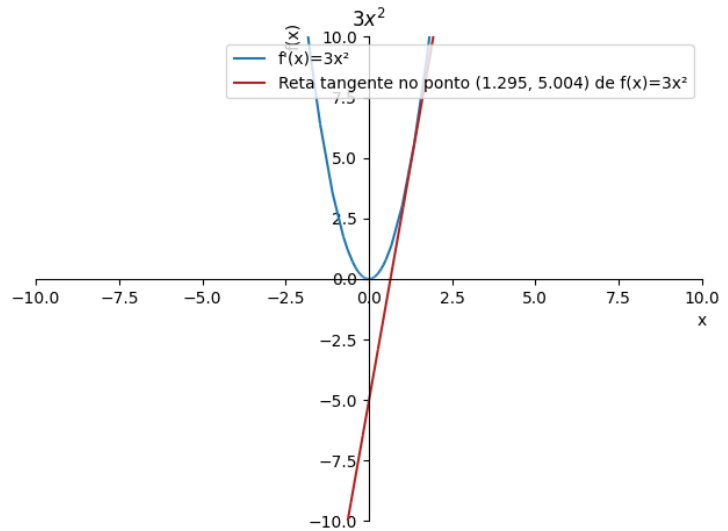


Figure 13: Função derivada com reta tangente

Função $f''(x)$

Para obter a segunda derivada, procedemos de forma semelhante ao que fizemos anteriormente no caso da primeira derivada.

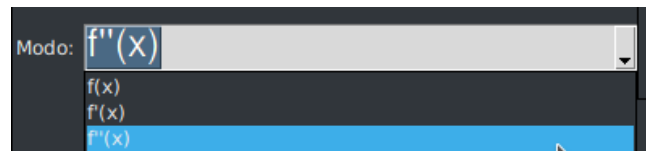


Figure 14: Modo segunda derivada $f''(x)$

Novamente, observado o gráfico obtido, vemos que a segunda derivada de $f(x) = x^3$ é igual a $f''(x) = 6x$

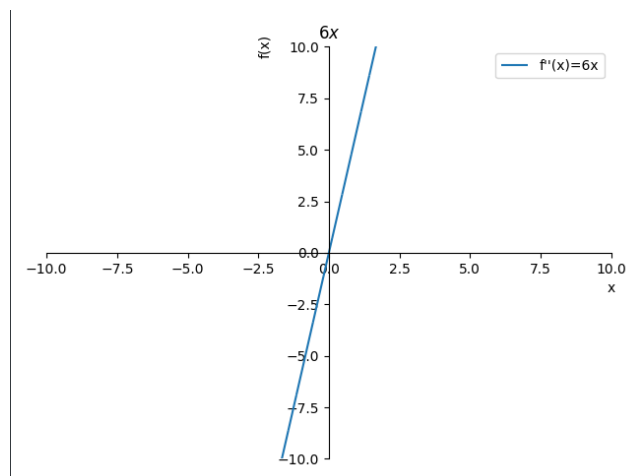


Figure 15: $f''(x)$ de x^3

Uma vez mais, usando a funcionalidade de reta tangente, com o ponto P(0.270, 1.626) como exemplo obtemos.

Como podemos observar, a reta da função e a sua reta tangente estão sobrepostas, pois a segunda derivada corresponde a uma função do primeiro grau.

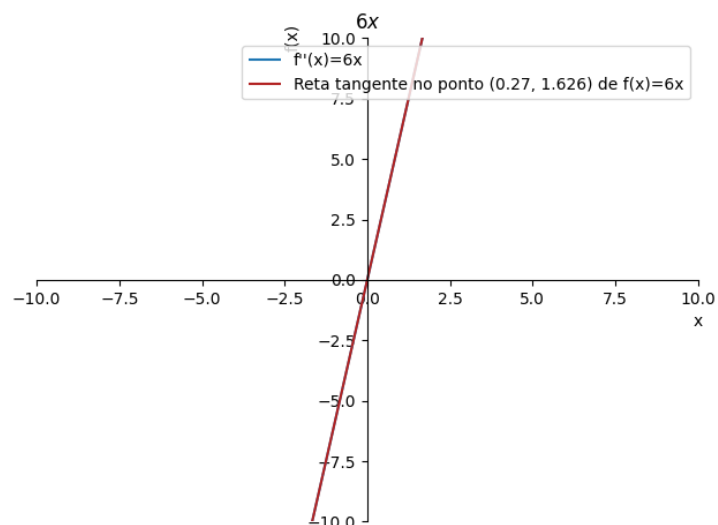


Figure 16: Reta tangente no ponto em $f'(x)$

No entanto, usando por exemplo a função

$f(x) =$

$$\frac{x^2}{\sqrt{x} + 3}$$

e obtendo a sua segunda derivada

$f''(x) =$

$$-\frac{7\sqrt{x}}{4(\sqrt{x} + 3)^2} + \frac{x}{2(\sqrt{x} + 3)^2} + \frac{2}{\sqrt{x} + 3}$$

obtemos a reta tangente no ponto P(0.1013, 0.4619) como demonstrado.

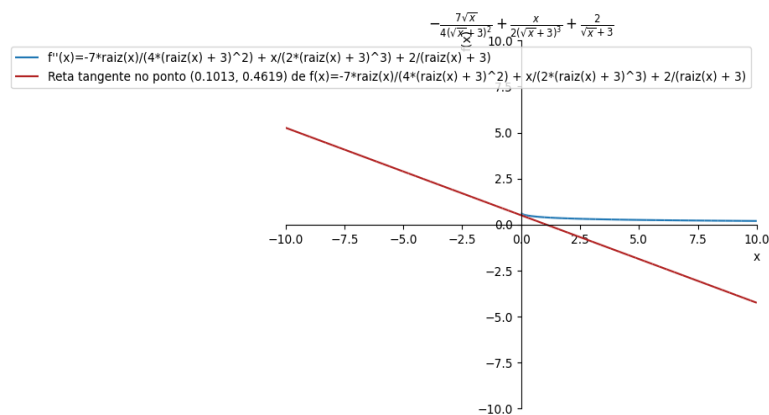


Figure 17: Exemplo alternativo de reta tangente no ponto em $f''(x)$

A introdução de funções no programa por sua vez pode ser feita de duas formas, utilizando os botões fornecidos na própria aplicação como demonstrado na imagem.

Operações:	+	-	*	/
	sin(x)	cos(x)	tan(x)	cotg(x)
	x^	√	()
	x	√(a,x)	log(x)	log_a(x)
	e	e^	ln(x)	π
	sec(x)	cosec(x)	arcsen(x)	arccos(x)
	arctan(x)	arccotan(x)	arcsec(x)	arccosec(x)

Figure 18: Painel de inserção de operações comuns da aplicação

Alternativamente, podemos introduzir diretamente a função desejada utilizando sintaxe do programa, por exemplo temos a seguinte função.

Figure 19: Caixa de inserção da função

Quanto aos restante elementos, temos o botão “Apagar” que elimina o texto da função.

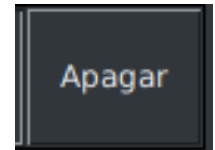


Figure 20:
Botão Apagar

Finalmente, temos o botão “*Resolver*”, que executa as operações seleccionadas na aplicação, “*Info*” que mostra-nos o manual e a história do calculo diferencial e derivadas e o botão “*Sair*” que fecha a aplicação.

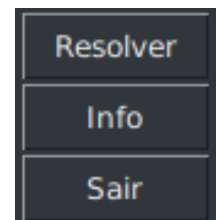


Figure 21:
Botões
execução
aplicação

Código

Explicação Geral

```
def inFuncao(self, funcao, coord_x, coord_y, margem_erro=0.05):
    """Verifica se um ponto inserido faz parte de uma função tendo em conta uma margem de erro dada

    Args:
        funcao (string): Função a avaliar
        coord_x (float): Coordenada x da reta tangente a função
        coord_y (float): Coordenada y da reta tangente a função
        margem_erro (float, optional): #Margem de erro para considerar um ponto dentro da função. Defaults to 0.00001.

    Returns:
        [boolean]: True se faz parte da função, False caso não faça
    """
    #Substitui x por valor da coordenada x
    func_point = funcao.replace('x', "("+str(coord_x)+")")

    #Resolve a função
    func_point = s.sympify(func_point)

    #Verifica se o ponto está na reta
    return (coord_y*(1-margem_erro) < func_point and func_point < coord_y*(1+margem_erro))
```

Código que verifica se um determinado ponto faz ou não parte de uma função

```
def DecliveValor(self, funcao, coord_x):
    """Calcula o declive da reta num ponto de uma função recorrendo a derivada

    Args:
        funcao (string): Função para analisar
        coord_x (float): Coordenada do ponto x

    Returns:
        float: declive
    """
    x = s.Symbol('x')
    funcao_deriv = s.diff(funcao)

    m = funcao_deriv.subs(x, coord_x)
    return m
```

Calcula o declive da reta num determinado ponto dada a abcissa deste ponto e a função

```
def retaTangentePonto(self, funcao, coord_x=0.0, coord_y=0.0, show_tangente=False, legenda="f(x)", plot_eixo_x_limites=(-10, 10), plot_eixo_y_limites=(-10, 10)):
    """Mostra o grafico de uma função, suporta mostrar a reta tangente num ponto (x, y) dadas as coordenadas
    You, 10 days ago · LaTeX on graphs, fixes, rewrites and improvements
    Args:
        funcao (string): Função a mostrar em formato grafico
        coord_x (float, optional): Ponto x da reta tangente. Defaults to 0.0.
        coord_y (float, optional): Ponto y da reta tangente. Defaults to 0.0.
        show_tangente (bool, optional): Mostrar reta tangente da função, True para mostrar, False para omitir. Defaults to False.
        legenda (str, optional): Identificador do tipo de função, exemplos; f(x), f'(x), f''(x), etc. Defaults to "f(x)".
    """
    m, x, y = s.symbols('m x y')

    funcao_pretty = self.handler.pretty_ready(funcao)

    funcao_plot = s.plot(funcao, show=False)
    funcao_plot.legend=True
    funcao_plot[0].label=legenda+ funcao_pretty
    funcao_plot.xlim = plot_eixo_x_limites
    funcao_plot.ylim = plot_eixo_y_limites

    #Reta tangente no ponto (x,y)
    if show_tangente == True:
        m = self.DecliveValor(funcao, coord_x)
        y=s.sympify(m*(x-coord_x)+coord_y)

        rt = s.plot(y, show=False)

        rt.xlim = plot_eixo_x_limites
        rt.ylim = plot_eixo_y_limites

        funcao_plot.append(rt[0])

        reta_tangente_pretty = "Reta tangente no ponto (" +str(coord_x)+", " +str(coord_y)+") de f(x)="+self.handler.pretty_ready(funcao)
        funcao_plot[1].label=reta_tangente_pretty
        funcao_plot[1].line_color = 'firebrick'

    funcao_latex = self.sympyLatexify(funcao)
    funcao_plot.title = f"${funcao_latex}$"
    funcao_plot.show()
```

Código responsável o gráfico de uma função e se selecionado, mostrar também a reta tangente num ponto.

```
def derivate(self, funcao, ordem=1):
    """Calcula a derivada de ordem n e retorna a sua expressao
    Args:
        funcao (string): Função a derivar
        ordem (int, optional): Order da derivada, 1 para f'(x), 2 para f''(x), etc... Defaults to 1.
    Returns:
        [Derivative]: Expressao da derivada de ordem n
    """
    derivada_ordem_n = s.diff(funcao)
    for i in range(ordem-1):
        try:
            if(str(derivada_ordem_n) == "0"):
                return derivada_ordem_n
            derivada_ordem_n = s.diff(derivada_ordem_n)
            derivada_ordem_n = str(derivada_ordem_n)
        except Exception as e:
            messagebox.showerror("Erro!",
                                "Não foi possivel calcular a derivada de ordem " + str(ordem) + " da função inserida!\n\n" + str(e))
    return derivada_ordem_n
```

Função que efetua a derivada de ordem “n” de uma determinada função

Possíveis Melhorias

Passando agora a possíveis melhorias que poderiam ter sido implementadas no projeto, em termos de funcionalidade, a criação de um subsistema para permitir a atualização dos gráficos mostrados pela aplicação em tempo real (gerar o gráfico para novas coordenadas da “*viewport*” quando o utilizador o arrasta com o rato) poderia mostrar-se uma mais valia. Adicionalmente, a implementação de uma interface gráfica que permitisse o calculo de derivadas de ordem “ n ” poderia mostrar-se também útil caso fosse desejado fazer uma análise da 3^o derivada ou 10^o derivada de uma função por exemplo.

Conclusão

Em suma, creio que os objetivos propostos pelo docente para a realização deste projeto foram concluídos, no entanto, como qualquer trabalho realizado, existem sempre possíveis melhorias ou funcionalidades que poderiam ser adicionadas.

Webgrafia

História Cálculo Diferencial e Derivada

<https://en.wikipedia.org/wiki/Derivative>

https://en.wikipedia.org/wiki/History_of_calculus

[https://en.wikipedia.org/wiki/Fermat%27s_theorem_\(stationary_points\)](https://en.wikipedia.org/wiki/Fermat%27s_theorem_(stationary_points))

<http://www.math.wpi.edu/IQP/BVCalcHist/calc2.htm>

https://en.wikipedia.org/wiki/Numerical_differentiation

https://en.wikipedia.org/wiki/Symmetric_derivative

Software

Code IDE

<https://code.visualstudio.com/>

Python

<https://www.python.org/>

Módulo Numpy

<https://numpy.org/>

Módulo Matplotlib

<https://matplotlib.org/>

Módulo PyInstaller

<https://www.pyinstaller.org/>

Módulo Scipy

<https://www.scipy.org/>

Módulo Sympy

<https://www.sympy.org/en/index.html>

NSIS

https://nsis.sourceforge.io/Main_Page

Repositório do projeto no Github:

https://github.com/hfmmb/Integrate_1920

Repositório do projeto no Github (Versão executável):

https://github.com/hfmmb/Integrate_1920/releases