Criação de Ferramenta para Cálculo de Derivadas Compostas

José Henrique Ricordi Cruz[1], **Rafael dos Santos Braz**[1]

[1]Centro de Ciências Tecnológicas – Universidade Estadual do Norte do Paraná (UENP)
Bandeirantes – PR – Brasil

henriquericordi@gmail.com, rafinha.santos.braz@gmail.com

1. Introdução

O objetivo do presente trabalho é desenvolver e apresentar uma ferramenta para o auxílio na Disciplina de Cálculo I, que especificamente soluciona funções, que para serem resolvidas usam da ”regra da cadeia” (derivadas compostas). A motivação para o desenvolvimento de tal ferramenta se baseia no fato de ser enfrentada grande dificuldade por docentes e discentes no processo de ensino e aprendizagem deste conteúdo, principalmente pela deficiência de conceitos de matemática básica que são de grande importância para o estudo desse conteúdo, que está presente no ensino superior e que possui um alto índice de reprovação. Além disso, o objetivo de utilizar os recursos tecnológicos é justamente para que os cursistas aprendam a linguagem de tais softwares e resgatem conceitos de Derivadas, mas de uma maneira diferenciada em que há possibilidade de se fazer simulações de maneira rápida e eficiente. [de Figueiredo et al. 2013] [Gasparin and Kestring 2014]

A ferramenta desenvolvida tem como principal objetivo propiciar a relação entre “teoria e prática”, auxiliando o aluno na compreensão da teoria com o uso dos recursos tecnológicos e na prática com uma aplicação do conceito de derivadas em problemas da área de ciência da computação. [de Figueiredo et al. 2013]

O *software* desenvolvido foi para ser executado em ambientes computacionais que utilizam os sistemas operacionais Windows ou distribuições Linux. A aplicação, é executada diretamente por meio do terminal de processamento de comandos e sua utilização é possível pela inserção de linhas de texto contendo a função que deseja-se calcular, e a saída também é uma linha de texto contendo o produto final da derivação.

2. Ferramenta para Cálculo de Derivadas Compostas

As seções e subseções a seguir dedicam-se à descrever e apresentar a ferramenta desenvolvida, bem como os recursos utilizados no processo de codificação e, por fim, apresentar exercícios de teste do *software*.

2.1. Recursos Utilizados

Como recursos foram utilizados a linguagem de programação C/C++, Interface de Desenvolvimento Integrado Atom [GitHub 2016]. A Linguagem C foi criada na década de 1970 por Dennis Ritchie, a linguagem C++, por sua vez, foi desenvolvida por Bjarne Stroustrup, que adaptou a linguagem C para o paradigma orientado a objetos. A popularização da linguagem veio do seu uso para a construção do sistema operacional UNIX, mas outros sistemas usam a linguagem como base, tais como o mundialmente popular Windows, entre outros sistemas operacionais. [Merlin et al. 2013]

Atom é um editor de texto de código aberto e livre, desenvolvido pela *GitHub Inc.* que permite personalização para ser usado como Interface de Desenvolvimento Integrado.

2.2. Estrutura Lógica e Código Fonte

A seguir consta-se o código fonte da aplicação desenvolvida, bem como a descrição de seu funcionamento e estrutura lógica. É importante ressaltar que o código foi desenvolvido em duas partes distintas: a principal que contém o método de execução inicial do *software*; e a biblioteca de funções utilizadas para execução da aplicação.

2.2.1. Arquivo Principal

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <cstdio>
4  #include "bib.h"
5
6  using namespace std;
7
8  int main(){
9      char op;
10     do{
11         limpatela();
12         cabecalho();
13         derivada(inserir_funcao());
14         printf("Opcoes:\n1 - Novo Calculo\n2 - Sair\n3 - Ver
               Exemplos\n\nDigite a Opcao: ");
15         cin >> op;
16         cin.ignore();
17         op = tolower(op);
18         primaria = "";
19         if (op == '3'){
20             limpatela();
21             cabecalho();
22             cout << "\nDigite:\nf(x)=[Funcao Trigonometrica]([
                   Expressao1*x]^[Expressao2])\n\n[Expressao
                   Trigonometrica], podem ser -sen, -cos, sen ou cos
                   \n[Expressao1] e [Expressao2], podem ser
                   constituidos de inteiros,\nfracoes(numero1/
                   numero2), pi(pi) e euler(e), sendo que nao podem
                   conter x e sao separados por *\n\nExemplo 1: \nf(
                   x)=sen(x^2)\nf'(x)=cos(x^2)*2*x\n\nExemplo 2: \nf
                   (x)=-cos(1/2*x^3*pi)\nf'(x)=sen(1/2*x^3*pi)*(1/2)
                   *3*pi*x^3*pi-1\n";
23             string c;
24             cout << "\nPressione qualquer tecla para continuar
                   ...\n";
25             getline(cin,c);
26         }
27     } while (op != '2');
28     return 0;
29 }
```

Este arquivo contém a função principal da ferramenta, por onde o aplicativo é iniciado e guia todas as operações a serem executadas até a obtenção do resultado final, sua estrutura baseia-se em um laço de repetição onde o usuário informa a operação desejada bem como a função a ser derivada, tendo como resultado uma única linha de texto com o resultado do processo de derivação.

2.3. Biblioteca de Funções

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <sstream>

using namespace std;

void cabecalho(void);
string inttostring (int);
void limpatela(void);
string inserir_funcao(void);
void derivada(string);
string copiar(string);
void derivada_dentro(string);
void simplificacao(string);

string primaria = "";
int x, y;

void cabecalho(void){
    cout << "
       +-------------------------------------------------------------+\
       n";
    cout << "|Software para Calculo de Derivadas Compostas (
       Regra da Cadeia)|\n";
    cout << "|UENP/CLM - CCT - Ciencia da Computacao -
       Professora: Caroline |\n";
    cout << "|Alunos: Rafael dos Santos Braz  || Jose Henrique
       Ricordi Cruz |\n";
    cout << "
       +-------------------------------------------------------------+\
       n\n";
}

void limpatela(void){
    #ifdef _WIN32
    system("cls");
    #endif
    #ifdef __linux__
    system("clear");
    #endif
}
```

```
36
37 string inserir_funcao(void){
38     string funcao;
39     cout << "f(x)=";
40     getline(cin, funcao);
41     return funcao;
42 }
43
44 void derivada(string funcao){
45     for (int cont = 0; cont < funcao.length(); cont++){
46         funcao[cont] = tolower(funcao[cont]);
47     }
48     cout << "f'(x)=";
49     if (funcao.find("-sen") != -1){
50         primaria += "-cos(" + copiar(funcao) + ")*";
51         derivada_dentro(copiar(funcao));
52     } else if (funcao.find("-cos") != -1 ){
53         primaria += "sen(" + copiar(funcao) + ")*";
54         derivada_dentro(copiar(funcao));
55     } else if (funcao.find("sen") != -1 ){
56         primaria += "cos(" + copiar(funcao) + ")*";
57         derivada_dentro(copiar(funcao));
58     } else if (funcao.find("cos") != -1 ){
59         primaria += "-sen(" + copiar(funcao) + ")*";
60         derivada_dentro(copiar(funcao));
61     }
62     cout << endl;
63 }
64
65 string copiar(string funcao){
66     string f = "";
67     for (int c = funcao.find('(') + 1; c < funcao.find(')'); c
       ++){
68         f += funcao[c];
69     }
70     return f;
71 }
72
73 string inttostring (int n){
74   stringstream s;
75   s << n;
76   return s.str();
77 }
78
79 void derivada_dentro(string funcao){
80     string f1 = "", f2 = "", mostrar = "";
81     string f1_1 = "", f1_2 = "";
82     string f2_1 = "", f2_2 = "";
83     for (int c = 0; c < funcao.find('^'); c++){
```

```
84          f1 += funcao[c];
85      }
86      for (int c = funcao.find('^') + 1; c < funcao.length(); c++)
            {
87          f2 += funcao[c];
88      }
89      int statusf1 = f1.find('*');
90      int statusf2 = f2.find('*');
91      if (statusf1 != -1){
92          for (int c = 0; c < statusf1; c++){
93              f1_1 += f1[c];
94          }
95          for (int c = statusf1 + 1; c < f1.length(); c++){
96              f1_2 += f1[c];
97          }
98      }
99      if (statusf2 != -1){
100         for (int c = 0; c < statusf2; c++){
101             f2_1 += f2[c];
102         }
103         for (int c = statusf2 + 1; c < f2.length(); c++){
104             f2_2 += f2[c];
105         }
106     }
107     if (statusf1 == -1 && statusf2 == -1){
108         if (f1[0] != 'x'){
109             mostrar += '0';
110         } else{
111             if (isdigit(f2[0]) && f2.length() == 1){
112                 if (f2.length() == 1){
113                     if (f2[0] != '1'){
114                         mostrar += f2 + '*' + f1;
115                         int n = atoi(f2.c_str()) - 1;
116                         mostrar += '^' + inttostring(n);
117                     } else{
118                         mostrar += f2 + '*' + f1;
119                         int n = atoi(f2.c_str()) - 1;
120                         mostrar += '^' + inttostring(n);
121                     }
122                 } else{
123                     mostrar += f2 + '*' + f1;
124                     int n = atoi(f2.c_str()) - 1;
125                     mostrar += '^' + inttostring(n);
126                 }
127             } else if (f2.find('/') == -1){
128                 mostrar += f2 + '*' + f1 + '^' + f2 + "-1";
129             } else if (f2.find('/') != -1){
130                 mostrar += f2 + '*' + f1 + '^' + '(' + f2 + ')'
                        + "-1";
```

```
131             }
132         }
133     } else if (statusf1 != -1 && statusf2 == -1){
134         if (((f1_1.find('/') == -1 && f2.find('/') == -1) && (
                isdigit(f1_1[0]) || f1_1[0] == '-')) && (isdigit(f2
                [0]) || f2[0] == '-')){
135             int n1 = atoi(f1_1.c_str());
136             int n2 = atoi(f2.c_str());
137             int aux = n1 * n2;
138             int exp = n2 - 1;
139             mostrar += inttostring(aux) + '*' + f1_2 + '^' +
                    inttostring(exp);
140         }  else if (f2.find('/') == -1 && f1_1.find('/') == -1){
141             mostrar += f1_1 + '*' + f2 + '*' + f1_2 + '^' + f2 +
                     "-1";
142         } else if (f2.find('/') != -1 && f1_1.find('/') == -1){
143             mostrar += f1_1 + '*' + '(' + f2 + ')' + '*' + f1_2
                    + '^' + '(' + f2 + ')' + "-1";
144         } else if (f2.find('/') != -1 && f1_1.find('/') != -1){
145             mostrar += '(' + f1_1 + ')' + '*' + '(' + f2 + ')' +
                     '*' + f1_2 + '^' + '(' + f2 + ')' + "-1";
146         } else if (f2.find('/') == -1 && f1_1.find('/') != -1){
147             mostrar += '(' + f1_1 + ')' + '*' + f2 + '*' + f1_2
                    + '^' + f2 + "-1";
148         }
149     } else if (statusf1 == -1 && statusf2 != -1){
150         if (((isdigit(f2_1[0]) || f2_1[0] == '-') && f2_1.find('
                /') == -1) && ((isdigit(f2_2[0]) || f2_2[0] == '-')
                && f2_2.find('/') == -1)){
151             int n1 = atoi(f2_1.c_str());
152             int n2 = atoi(f2_2.c_str());
153             int aux = n1 * n2;
154             int exp = aux - 1;
155             mostrar += inttostring(aux) + '*' + f1 + '^' +
                    inttostring(exp);
156         } else if ((f2_1.find('/') == -1) && (f2_2.find('/') ==
                -1)){
157             mostrar += f2 + '*' + f1 + '^' + f2 + "-1";
158         } else if ((f2_1.find('/') == -1) && (f2_2.find('/') !=
                -1)){
159             mostrar += f2_1 + '*'+ '(' + f2_2 + ')' + '*' + f1 +
                     '^' + f2_1 + '*'+ '(' + f2_2 + ')' + "-1";
160         } else if ((f2_1.find('/') != -1) && (f2_2.find('/') ==
                -1)){
161             mostrar += '(' + f2_1 + ')' + '*' + f2_2 + '^' + '('
                     + f2_1 + ')' + '*' + f2_2 + "-1";
162         } else if ((f2_1.find('/') != -1) && (f2_2.find('/') !=
                -1)){
163             mostrar += '(' + f2_1 + ')' + '*' + '(' + f2_2 + ')'
```

```
                + f1 + '^' + '(' + f2_1 + ')' + '*' + '(' + f2_2
                + ')' + "-1";
164         }
165     } else if (statusf1 != -1 && statusf2 != -1){
166         if (((isdigit(f2_1[0]) || f2_1[0] == '-') && f2_1.find('
            /') == -1) && ((isdigit(f2_2[0]) || f2_2[0] == '-')
            && f2_2.find('/') == -1) && ((isdigit(f1_1[0]) ||
            f1_1[0] == '-') && f1_1.find('/') == -1)){
167             int n1 = atoi(f2_1.c_str());
168             int n2 = atoi(f2_2.c_str());
169             int base = atoi(f1_1.c_str());
170             int aux = n1 * n2;
171             int exp = aux - 1;
172             int aux2 = aux * base;
173             mostrar += inttostring(aux2) + '*' + f1_2 + '^' +
                    inttostring(exp);
174         }
175         else if ((f1_1.find('/') == -1) && (f2_1.find('/') ==
            -1) && (f2_2.find('/') == -1)){
176             mostrar += f1_1 + '*' + f2 + '*' + f1_2 + '^' + f2 +
                 "-1";
177         }
178         else if ((f1_1.find('/') == -1) && (f2_1.find('/') ==
            -1) && (f2_2.find('/') != -1)){
179             mostrar += f1_1 + '*' + f2_1 + '*' + '(' + f2_2 + ')
                ' + '*' + f1_2 + '^' + f2_1 + '*' + '(' + f2_2 +
                ')' + "-1";
180         }
181         else if ((f1_1.find('/') == -1) && (f2_1.find('/') !=
            -1) && (f2_2.find('/') == -1)){
182             mostrar += f1_1 + '*' + '(' + f2_1 + ')' + '*' +
                    f2_2 + '*' + f1_2 + '^' + '(' + f2_1 + ')' + '*'
                    + f2_2 + "-1";
183         }
184         else if ((f1_1.find('/') == -1) && (f2_1.find('/') !=
            -1) && (f2_2.find('/') != -1)){
185             mostrar += f1_1 + '*' + '(' + f2_1 + ')' + '*' + '('
                 + f2_2 + ')' + '*' + f1_2 + '^' + '(' + f2_1 + '
                )' + '*' + '(' + f2_2 + ')' + "-1";
186         }
187         else if ((f1_1.find('/') != -1) && (f2_1.find('/') ==
            -1) && (f2_2.find('/') == -1)){
188             mostrar += '(' + f1_1 + ')' + '*' + f2 + '*' + f1_2
                    + '^' + f2 + "-1";
189         }
190         else if ((f1_1.find('/') != -1) && (f2_1.find('/') ==
            -1) && (f2_2.find('/') != -1)){
191             mostrar += '(' + f1_1 + ')' + '*' + f2_1 + '*' + '('
                 + f2_2 + ')' + '*' + f1_2 + '^' + f2_1 + '*' + '
```

```
                (' + f2_2 + ')' + "-1";
192         }
193         else if ((f1_1.find('/') != -1) && (f2_1.find('/') !=
               -1) && (f2_2.find('/') == -1)){
194             mostrar += '(' + f1_1 + ')' + '*' + '(' + f2_1 + ')'
                    + '*' + f2_2 + '*' + f1_2 + '^' + '(' + f2_1 + '
                   )' + '*' + f2_2 + "-1";
195         }
196         else if ((f1_1.find('/') != -1) && (f2_1.find('/') !=
               -1) && (f2_2.find('/') != -1)){
197             mostrar += '(' + f1_1 + ')' + '*' + '(' + f2_1 + ')'
                    + '*' + '(' + f2_2 + ')' + '*' + f1_2 + '^' + '(
                   ' + f2_1 + ')' + '*' + '(' + f2_2 + ')' + "-1";
198         }
199     }
200     simplificacao(mostrar);
201 }
202
203 void simplificacao(string funcao){
204     x = (funcao.find("0*"));
205     string simplificada = funcao;
206     if (funcao.find("*0") != -1 || x != -1){
207         if((x == 0) || x != 0 && isdigit(x - 1) == false){
208             simplificada = "0";
209             primaria = "";
210         }
211         else{
212             simplificada = funcao;
213         }
214     }
215     else {
216         simplificada = funcao;
217     }
218     x = simplificada.find("^1");
219     if (x != -1){
220         if (x + 1 == simplificada.length() - 1){
221             simplificada.erase(x, 2);
222         }
223         else {
224             if(isdigit(x + 2) == false){
225                 simplificada.erase(x, 2);
226             }
227         }
228     }
229     x = simplificada.find("^0");
230     if (x != -1){
231         if (x + 1 == simplificada.length() - 1){
232             if (simplificada.find_last_of("*", x)){
233                 simplificada.erase(simplificada.find("*") + 1,x
```

```
                    + 1);
234                 simplificada += '1';
235             } else{
236                 simplificada.erase(0,x + 1);
237                 simplificada += '1';
238             }
239         }
240     }
241     x = simplificada.find("*1");
242     if(x != -1){
243         if (x + 1 == simplificada.length() - 1){
244             simplificada.erase(x, 2);
245             if (simplificada.length() == 1 && simplificada[0] ==
                    '1'){
246                 simplificada = "";
247                 primaria.erase(primaria.length() - 1 ,1);
248             }
249         }
250     }
251     x = simplificada.find("^");
252     y = simplificada.find("-");
253     if(x && y && isdigit(simplificada[y + 1]) && isdigit(
            simplificada[y - 1])){
254         string simp1 = "", simp2 = "";
255         for(int i = x + 1; i < y ; i++){
256             simp1 += simplificada[i];
257         }
258         for(int i = y + 1; i < simplificada.length(); i++){
259             simp2 += simplificada[i];
260         }
261         int n1 = atoi(simp1.c_str());
262         int n2 = atoi(simp2.c_str());
263         int aux = n1 - n2;
264         simplificada.erase(x + 1);
265         simplificada += inttostring(aux);
266     }
267     cout << primaria << simplificada << endl;
268 }
```

Este arquivo é subdividido em várias funções, que executam tarefas diferentes:

- cabecalho (linhas 20–26): Mostra uma apresentação da ferramenta, dos autores e demais pessoas envolvidas por meio de códigos de saída de texto;
- inttostring (linhas 73–77): Converte um dado de texto (string) em um dado numérico inteiro (int);
- limpatela (linhas 28–35): Apaga da janela de visualização todo o texto exibido anteriormente, por meio de código verificador de sistema operacional, que define o comando adequado;
- inserir_funcao (linhas 37–42): Permite ao usuário informar a função a ser derivada;

- derivada (linhas 44–63): Realiza a primeira etapa do processo de derivação: ”derivada da de fora”(-sen, -cos, sen, cos para sua respectiva derivada), utiliza estrutura de comparação e condição para a escolha;
- copiar (linhas 65–71): Responsável por identificar a função interna da função composta por meio de uma estrutura de repetição, comparação e condição;
- derivada_dentro (linhas 79–201): A maior seção do código, que é responsável por efetivamente realizar a derivação da função interna, retornando um resultado que pode ainda não estar simplificado, utiliza uma estrutura de repetição, condição, comparação e as demais funções presentes na biblioteca;
- simplificacao (linhas 203–268): Encontra no retorno da função anterior possíveis simplificações, como por exemplo: $*0$, ^1, ^0 , $*1$ e após esse processo ela realiza as conversões e correções necessárias, fornecendo a função final já derivada.

2.4. Exercícios

Consta-se, abaixo, dez exercícios de teste da ferramenta desenvolvida:

1. Função de Entrada: $f(x) = sen(x^2)$;
   Função de Saída: $f'(x) = cos(x^2) * 2 * x$;
   Processo de Integração:
   $\int cos(x^2) * 2x \, \mathrm{d}x$
   $u = x^2$
   $du = 2xdx$
   $\int cos(u) \, \mathrm{d}u \rightarrow |sen(u) \, \mathrm{d}u$
   $f(x) = sen(x^2) + C$

2. Função de Entrada: $f(x) = cos(2 * x^2)$;
   Função de Saída: $f'(x) = -sen(2 * x^2) * 4 * x$;
   Processo de Integração:
   $\int -sen(2x^2) * 4x \, \mathrm{d}x$
   $u = 2x^2$
   $du = 4xdx$
   $\int -sen(u) \, \mathrm{d}u \rightarrow |cos(u) \, \mathrm{d}u$
   $f(x) = cos(2x^2) + C$

3. Função de Entrada: $f(x) = -sen(pi * x^{2*e})$;
   Função de Saída: $f'(x) = -cos(pi * x^{2*e}) * pi * 2 * e * x^{2*e-1}$;
   Processo de Integração:
   $\int -cos(\pi x^{2e}) * 2\pi e * x^{2e-1} \, \mathrm{d}x$
   $u = \pi x^{2e}$
   $du = 2\pi e x^{2e-1} dx$
   $\int -cos(u) \, \mathrm{d}u \rightarrow |-sen(u) \, \mathrm{d}u$
   $f(x) = -sen(\pi x^{2e}) + C$

4. Função de Entrada: $f(x) = -cos(25 * x^{25*25})$;
   Função de Saída: $f'(x) = sen(25 * x^{25*25}) * 15625 * x^{624}$;
   Processo de Integração:
   $\int sen(25x^{625}) * 15625x^{624}\,\mathrm{d}x$
   $u = 25x^{625}$
   $du = 15625x^{624}dx$
   $\int sen(u)\,\mathrm{d}u \rightarrow |-cos(u)\,\mathrm{d}u$
   $f(x) = -cos(25x^{625}) + C$

5. Função de Entrada: $f(x) = sen(x^0)$;
   Função de Saída: $f'(x) = 0$;
   Processo de Integração:
   $\int 0\,\mathrm{d}x$
   $0\,|xdx$
   $f(x) = 0 + C$

6. Função de Entrada: $f(x) = -cos(x^1)$;
   Função de Saída: $f'(x) = sen(x^1)$;
   Processo de Integração:
   $\int -cos(x)\,\mathrm{d}x$
   $0\,|sen(x)dx$
   $f(x) = sen(x) + C$

7. Função de Entrada: $f(x) = -cos(2x^1)$;
   Função de Saída: $f'(x) = sen(2x^1) * 2$;
   Processo de Integração:
   $\int sen(2x) * 2\,\mathrm{d}x$
   $u = 2x$
   $du = 2dx$
   $\int sen(u)\,\mathrm{d}u \rightarrow |-cos(u)\,\mathrm{d}u$
   $f(x) = -cos(2x) + C$

8. Função de Entrada: $f(x) = sen(x^e)$;
   Função de Saída: $f'(x) = cos(x^e) * e * x^{e-1}$;
   Processo de Integração:
   $\int cos(x^e) * e * x^{e-1}\,\mathrm{d}x$
   $u = x^e$
   $du = ex^{e-1}dx$
   $\int cos(u)\,\mathrm{d}u \rightarrow |sen(u)\,\mathrm{d}u$
   $f(x) = sen(x^e) + C$

9. Função de Entrada: $f(x) = cos(x^{-pi})$;
   Função de Saída: $f'(x) = -sen(x^{-pi}) * -pi * x^{-pi-1}$;
   Processo de Integração:
   $\int -sen(x^{-\pi}) * -\pi * x^{-\pi-1} \, \mathrm{d}x$
   $u = x^{-\pi}$
   $du = -\pi x^{-\pi-1} dx$
   $\int -sen(u) \, \mathrm{d}u \rightarrow |cos(u) \, \mathrm{d}u$
   $f(x) = cos(x^{-\pi}) + C$

10. Função de Entrada: $f(x) = -sen(x^{\frac{1}{2})}$;
    Função de Saída: $f'(x) = -cos(x^{\frac{1}{2}}) * \frac{1}{2} * x^{\frac{1}{2}-1}$;
    Processo de Integração:
    $\int -cos(x^{\frac{1}{2}}) * \frac{1}{2} * x^{-\frac{1}{2}} \, \mathrm{d}x$
    $u = x^{\frac{1}{2}}$
    $du = \frac{1}{2} * x^{-\frac{1}{2}} dx$
    $\int -cos(u) \, \mathrm{d}u \rightarrow |-sen(u) \, \mathrm{d}u$
    $f(x) = -sen(x^{\frac{1}{2}}) + C$

Referências

de Figueiredo, E. B., Siple, I. Z., Longen, L. G., and Boeing, F. K. (2013). Integral definida: Um recurso tecnolÓgico para o professor. In ULBRA, editor, *Congresso Internacional de Ensino da Matemática*.

Gasparin, P. P. and Kestring, F. B. F. (2014). CÁlculo diferencial e integral na ponta da tecnologia: Geogebra e winplot como recurso de ensino aprendizagem. In EPREM, editor, *Encontro Paranaense de Educação Matemática*.

GitHub, I. (2016). Atom.

Merlin, J. R., Costa, E. B. B., and Mura, W. D. (2013). Programação i - notas de aula. *Universidade Estadual do Norte do Paraná*, 1(1):1 – 81.