PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

AVALIAÇÃO DE UM ALGORITMO PARA O CÁLCULO DA DERIVADA EM TEMPO CONTÍNUO

ADALBERTO ROMEIRO DE LIMA

GOIÂNIA
2021/1

ADALBERTO ROMEIRO DE LIMA

AVALIAÇÃO DE UM ALGORITMO PARA O CÁLCULO DA DERIVADA EM TEMPO CONTÍNUO

Trabalho de Conclusão de Curso apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Talles Marcelo Gonçalves de Andrade Barbosa.

Co-Orientador: Prof. Dr. Marco Antonio Figueiredo Menezes.

GOIÂNIA
2021/1

ADALBERTO ROMEIRO DE LIMA

AVALIAÇÃO DE UM ALGORITMO PARA O CÁLCULO DA DERIVADA EM TEMPO CONTÍNUO

Este trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, em ___/___/___.

_____

Prof. Ma. Nome do coordenador(a) de TCC

Coordenador(a) de Trabalho de Conclusão de Curso

Banca examinadora:

_____

Orientador: Talles Marcelo Gonçalves de Andrade Barbosa.

_____

Co-Orientador: Marco Antonio Figueiredo Menezes.

_____

Prof. Me. Daniel Corrêa da Silva.

RESUMO

Este trabalho apresenta a avaliação de um algoritmo para o cálculo da derivada em tempo contínuo. O algoritmo utilizado, realiza o cálculo da derivada por meio de duas aproximações, definidas como sendo diferença à frente e diferença central. Para isto, o cálculo utiliza o ajuste dinâmico no decréscimo da hipotenusa. O objetivo deste trabalho é avaliar o desempenho do algoritmo de derivada que utiliza este ajuste dinâmico. Foi realizada uma revisão bibliográfica, para conhecimento, aprofundamento e acompanhamento do objeto de estudo. Como segunda atividade foi feita a implementação do algoritmo. A terceira atividade foi a realização de testes onde fez-se o cálculo de erro relativo e absoluto da solução da derivada e também a medição do tempo computacional no controlador Arduino, no software Matlab e no simulador ModelSim-Altera. Chegou-se à conclusão de que para o uso do algoritmo deve-se observar o grau de exatidão que o problema envolve para o cálculo da derivada, pois em plataformas diferentes o resultado do cálculo da derivada pode ter variações.

Palavras-chave: Desempenho de algoritmos, decréscimo da hipotenusa, implementação de algoritmo.

ABSTRACT

This work presents the evaluation of an algorithm to calculate the derivative in continuous time. The algorithm used performs the calculation of the derivative by means of two approximations, defined as being forward difference and central difference. For this, the calculation uses the dynamic adjustment in the decrease of the hypotenuse. The objective of this work is to evaluate the performance of the derivative algorithm that uses this dynamic fit. A bibliographical review was carried out, for knowledge, deepening and monitoring of the object of study. The second activity was the implementation of the algorithm. The third activity was the realization of tests where the calculation of relative and absolute error of the solution of the derivative was done and also the measurement of the computational time in the Arduino controller, in the Matlab software and in the ModelSim-Altera simulator. It was concluded that, for the use of the algorithm, the degree of accuracy that the problem involves for the calculation of the derivative must be observed, since on different platforms the result of the calculation of the derivative may vary.

Keywords: Algorithm performance, hypotenuse decrease, algorithm implementation.

SUMÁRIO

1. INTRODUÇÃO

1.1. Apresentação do Tema

Conforme Cormen (2012), o algoritmo é uma sequência de etapas computacionais que transformam a entrada na saída, ou seja, qualquer procedimento computacional bem definido que toma um valor ou um conjunto de valores como entrada e produz como saída um valor ou um conjunto de valores é considerado um algoritmo.

No cenário da computação, o algoritmo é tido como uma ferramenta para resolução de problemas computacionais específicos.

Segundo Cormen (2009), para que uma solução proposta por intermédio de um algoritmo seja eficiente, dois aspectos devem ser considerados em termos computacionais: o primeiro refere-se à memória requerida e o segundo o tempo de execução gasto. Com a evolução do hardware, em muitas das aplicações o aspecto da memória tem apresentado um desafio menor, quando comparado com o tempo de execução.

O desempenho do algoritmo depende de diversos fatores, não somente de sua complexidade teórica. Para cada grupo problema-algoritmo é possível medir o tempo computacional, o número de falhas, o erro da solução dentre outros, analisar os resultados obtidos não é uma tarefa fácil. A quantidade de informações obtidas pode ser enorme, o que pode dificultar a apresentação e interpretação dos resultados Araújo (2015).

Segundo Linder (2008) a avaliação de desempenho de um algoritmo quando executado por um computador pode ser feita de duas maneiras: a primeira é a posteriori e a segunda é a priori. A primeira maneira a posteriori envolve a execução própria do algoritmo, medindo-se o tempo de execução do algoritmo. Só pode ser exata se e somente se detalhes da arquitetura da máquina, da linguagem de programação, do código gerado pelo compilador dentre outros fatores forem conhecidos. A segunda maneira a priori é feita de forma analítica sem a execução do algoritmo, desde de que dois itens sejam considerados: a entrada (os dados fornecidos) e o número de instruções executadas pelo algoritmo.

A derivada é um operador matemático, cuja definição está intimamente relacionada à taxa de variação instantânea da função sobre a qual é aplicada. Disto decorrem várias aplicações, como, por exemplo, o estudo da taxa de variação de

temperaturas, do crescimento de certa população, da velocidade de corpos ou objetos, dentre outras.

Conforme Santana (2010) o entendimento da metodologia da derivação é de suma importância em razão das diversas áreas de aplicações em diferentes campos da ciência. Com o auxílio de diversos matemáticos, o estudo da derivada foi desenvolvido no decurso de 2500 anos, o que era como foco do estudo a reta tangente, se transformou em uma ferramenta poderosa para solução de problemas.

O cálculo da derivada, conforme (LANG, 1968), pode ser formalizado da seguinte maneira. Dado I = [a,b] um intervalo da reta com a≠b, uma função real f: I $\subset$ R→R é diferenciável em x $\in$ I. Assim, define-se a derivada em um ponto x' $\in$ I como:

$$f'(x') = \lim_{h \to 0} \frac{f(x'+h) - f(x')}{h} \quad (1)$$

Contudo, em muitas circunstâncias, pode ser difícil de se obter valores de derivadas de uma função, uma vez que, elas podem ser expressões matemáticas difíceis ou intratáveis de derivar analiticamente, sendo necessário o uso de métodos numéricos para resolução desses problemas. Dentre os métodos encontrados na literatura, destacam-se a diferença para frente, também chamado de diferença progressiva, e a diferença central.

Conforme Gilat (2008) a diferença para frente e diferença central é definida como a inclinação da reta que conecta os pontos os respectivos pontos: (xi, f(xi)) e (xi+1, f(xi+1)), para diferença para frente e (xi-1, f(xi-1)) e (xi+1, f(xi+1)) para diferença central, conforme mostrado na Figura 1 e 2.

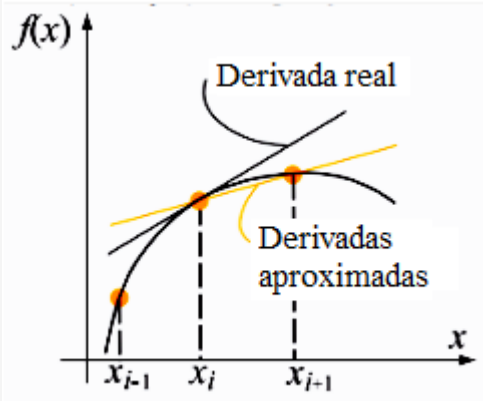

Figura 1 - Aproximação da derivada por diferença progressiva (GILAT, 2008, p. 256).

A Figura 1 mostra a aproximação da derivada por diferença para frente, ela avalia a derivada no ponto xi com os valores nesse ponto e naquele imediatamente a sua direita, representado como xi+1.

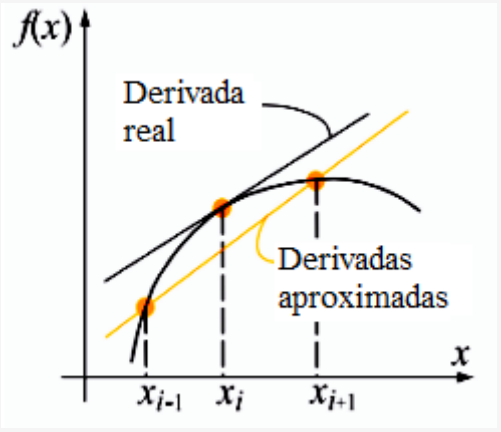

Figura 2 - Aproximação da derivada por diferença central (GILAT, 2008, p. 256).

A Figura 2 mostra a aproximação da derivada por diferença central, essa aproximação avalia a derivada em um dado ponto xi com os valores nesse ponto e naqueles imediatamente a sua esquerda e a sua direita, representado como xi-1 e xi+1.

Conforme o algoritmo M, proposto por Menezes, Maculan e Bueno (2018, no prelo), o cálculo da derivada é realizado por meio de duas aproximações, definidas neste texto como sendo diferença à frente e diferença central. Para isto, o cálculo utiliza o ajuste dinâmico no decréscimo da hipotenusa. Isto contrasta com os métodos encontrados na literatura, pelo cálculo das diferenças finitas de forma progressiva ou central, fixando dois pontos.

Segundo Menezes, Maculan e Bueno (2018, no prelo) a derivada pode ser definida com a tangente do ângulo α formado pela reta tangente ao gráfico da função f, denotado por G(f). Conforme mostrado na Figura 3. Considerando a aproximação linear de G(f) no ponto $\bar{x}$, quando $h \rightarrow 0$, tem-se que $\bar{x} + h \rightarrow \bar{x}$ e que $o(\bar{x}, h)$ tende para zero. Contudo, verifica-se que o tamanho da hipotenusa também tende para zero, quando $h \rightarrow 0$.

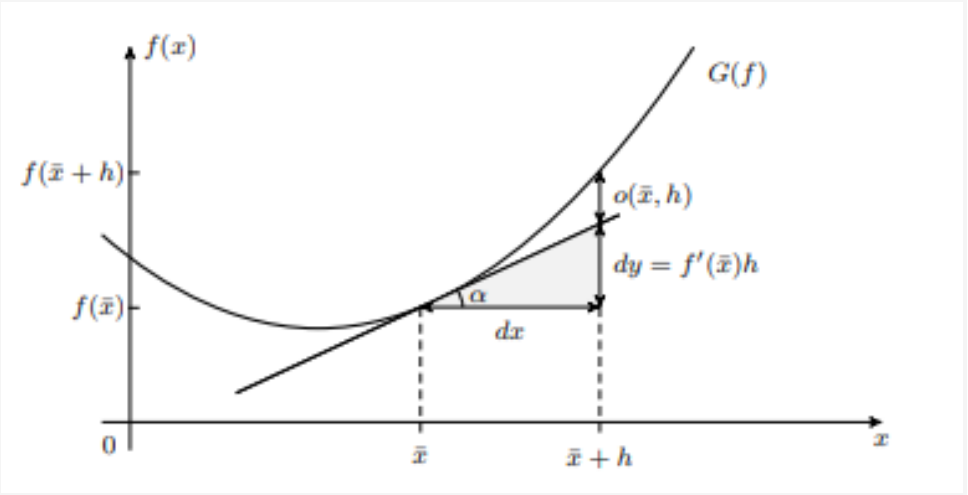

Figura 3 - Aproximação linear de G(f) no ponto $\overline{x}$ (MENEZES; MACULAN; BUENO, 2018).

A Figura 3 mostra a aproximação linear de G(f) no ponto $\bar{x}$, observa-se o decréscimo da hipotenusa quando $h \rightarrow 0$.

Este trabalho visa responder a seguinte questão de pesquisa: **Qual o desempenho do algoritmo M em formas de implementação diferentes?**

1.2. Objetivos

1.2.1. Geral

Avaliar o desempenho do algoritmo M para o cálculo da derivada que utiliza o ajuste dinâmico no decréscimo da hipotenusa, proposto por Menezes e Bueno (2018).

1.2.2. Específicos

Implementar o algoritmo M para derivada, utilizando o matlab, o controlador arduino e o simulador ModelSim-Altera 10.1d.

Realizar testes para avaliar o desempenho do algoritmo M utilizando o matlab, o controlador arduino e o simulador ModelSim-Altera 10.1d.

2. TRABALHOS RELACIONADOS

Silva (2017) enfatiza a fórmula de Taylor, indicando que ela é um pilar fundamental do Cálculo Numérico, pois ela ensina como aproximar funções complicadas por funções mais simples, contudo, fornece uma aproximação polinomial local cada vez melhor de uma função (muitas vezes) diferenciável à medida que calcula-se as suas derivadas.

Segundo Silva (2017), a primeira aplicação da fórmula de Taylor é a aproximação de derivadas. Uma vez que a função tenha uma fórmula muito complexa ou quando a função f não possui uma fórmula explícita para ela. A fórmula de Taylor fornece as opções:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{2}$$

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{3}$$

Em (2) está representada a fórmula da aproximação diferença para frente oriunda da fórmula de Taylor. A aproximação da diferença central está representada em (3) Também oriunda da fórmula de Taylor.

Silva (2017) realizou a implementação destas aproximações e utilizou como função de entrada f(x) = sin(x) + cos(x) e o ponto x = 1. Com o uso da diferença Progressiva e um h = $\sqrt{\epsilon mac}$ o erro relativo para aproximação = 5.363255086786758e-8. Com o uso da diferença centrada e um h = $\sqrt[3]{\epsilon mac}$ o erro relativo para aproximação = 1.790070602380168e-11.

Neto (2012) Afirma que o tempo de processamento de um algoritmo pode ser calculado ou estimado com o uso de dois métodos, são eles: o experimental e o teórico. O experimental é realizado a partir de testes de implementação do algoritmo. Já o teórico, parte-se da hipótese de que toda e qualquer instrução que compõem o algoritmo é executada em um tempo constante, contudo busca-se definir uma função t = f(n) onde t é o total de tempo de processamento e n é o tamanho da entrada.

Segundo Neto (2012) o método experimental possui como vantagem o fato de considerar condições reais. Já o analítico possui a garantia da neutralidade em relação aos efeitos que o tradutor e o hardware podem causar.

3. MATERIAIS E MÉTODOS

Para atingir os objetivos propostos foram realizadas as seguintes atividades.

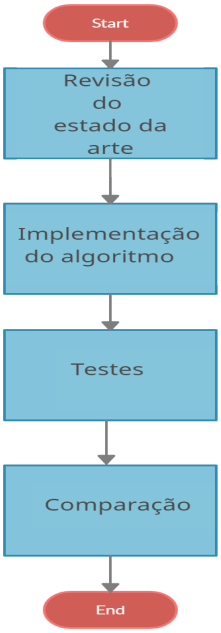

Figura 4 - Atividades propostas.

A Figura 4 mostra um diagrama de atividades que foram realizadas na elaboração do trabalho.

Na atividade de revisão do estado da arte foi feita uma revisão bibliográfica para conhecimento, aprofundamento e acompanhamento do objeto de estudo.

Para a segunda atividade do projeto realizou-se a implementação do algoritmo M com o cálculo da derivada pela diferença para a frente e também com o cálculo da derivada pela diferença central.

A terceira atividade do projeto foi a realização de testes para avaliar a exatidão dos algoritmos e o tempo de execução gasto pelos mesmos. Para a exatidão do algoritmo foi investigado o erro absoluto e o erro relativo em torno do valor exato da derivada da função de entrada.

Para o tempo de execução foi investigado os recursos disponíveis em cada plataforma, e feito a execução do algoritmo dez vezes em cada plataforma a fim de obter a média aritmética destes valores e assim poder representar o tempo de execução o mais próximo do real.

Conforme Justo (2018) Seja x um número real e $\bar{x}$, sua aproximação. O erro absoluto da aproximação $\bar{x}$ é definido como:

$$|x - \overline{x}| \tag{4}$$

Justo (2018) afirma também que o erro relativo da aproximação $\overline{x}$ é definido como:

$$\frac{|x - \overline{x}|}{|x|}, \quad x \neq 0 \tag{5}$$

Para a quarta atividade, foi feita a análise dos resultados obtidos com algoritmo M diferença para a frente e também com o cálculo da derivada pela diferença central.

Para realização da segunda atividade foram utilizadas as ferramentas de cada plataforma para implementação e teste de desempenho do algoritmo. As plataformas utilizadas foram:

- **Arduino UNO;**
- **Software Matlab;**
- **Simulador ModelSim-Altera 10.1d.**

3.1. Arduino UNO

A placa escolhida para o projeto foi a UNO, por se tratar de uma placa com custo mais baixo e de boa dinâmica, oferecendo as ferramentas necessárias para os objetivos propostos no trabalho. As principais características desta placa são:

- Microcontrolador ATmega328P;
- Pinos de I/O Digitais 14;
- Memória flash 32 KB (ATmega328P);
- Eeprom 1 KB (ATmega328P);
- Velocidade do Clock 16 MHz.

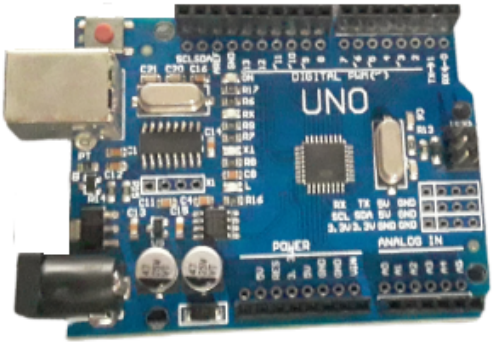

Figura 5 - Arduino Uno.

A Figura 5 mostra a placa Arduino utilizada na implementação e teste de desempenho do algoritmo M.

Conforme Rodrigues (2012), o Arduino foi se originou na Itália, através do professor Massimo Banzi, ointuito do professor era ensinar a programação de computadores de forma que este conhecimento conseguisse ser aplicado a projetos de arte, robótica e automação.

O Arduino tem sido usado em milhares de projetos e aplicativos diferentes. O aumento do seu uso se dá pelos fatores: o primeiro é por que ele é barato, as placas Arduino são relativamente baratas em comparação com outras plataformas de microcontroladores. O segundo é a questão dele ser Multiplataforma, o Software Arduino (IDE) é executado nos sistemas operacionais Windows, MacOS e Linux. O terceiro fator é o Ambiente de programação (IDE) simples e claro. O quarto fator é o código aberto Arduino (2018).

Dentre as funções disponíveis no arduino, este trabalho, como parte dos testes, fez o uso da função micros() do arduino. A função micros() retorna o número de microssegundos decorridos desde que a placa do Arduino começou a executar o programa atual – ou seja, o tempo em execução da aplicação. Esse número é incrementado no decorrer de 70 minutos, sendo zerado após passado esse tempo Arduino (2018).

3.2. Software Matlab

O software Matlab foi escolhido neste trabalho por se tratar de um software interativo de alta performance voltado para o cálculo numérico. O Matlab integra análise numérica, cálculo com matrizes e construção de gráficos.

Conforme Marchetto (2016), o software Matlab foi criado por volta de 1970, por Cleve Moler, então presidente do departamento de ciências da computação da

Universidade do Novo México. O software Matlab, permite a solução de problemas numéricos de maneira mais simples em comparação com outras linguagens de programação. Este software é voltado para o cálculo numérico, cálculo de matrizes, geração de gráficos e construção de algoritmos. O ambiente de trabalho do Matlab possui vantagens em seu uso, como por exemplo, os comandos são mais semelhantes com as expressões algébricas.

Segundo Chapman (2017), o Matlab (abreviatura do inglês *MATrix LABoratory* - Laboratório de Matrizes), possui como elemento de dados básico as matrizes (ou arranjos). Este software é especializado e otimizado para realização de cálculos científicos. Inicialmente o software foi projetado para cálculos com matrizes, no decorrer do tempo, o software se transformou em um sistema computacional capaz de resolver qualquer problema técnico. O matlab possui uma ampla biblioteca de funções predefinidas e que tornam as tarefas de programação mais fáceis e eficientes.

O software Matlab é do tipo interpretador, ou seja, o programa conversor recebe a primeira instrução do programa fonte, confere para ver se a sintaxe está correta, converte-a em linguagem de máquina e então ordena ao computador que execute esta instrução, e assim sucessivamente. Ao contrário de um compilador, que a partir de um programa de entrada produz outro, que equivale ao original, mas em uma linguagem executável como por exemplo a linguagem de máquina.

Este trabalho a fim de verificar o tempo de execução do algoritmo, como parte dos testes, utilizou-se a função tic,toc do Matlab. O matlab possui em sua biblioteca de funções a função *tic, toc*. O comando tic possui o papel de iniciar um cronômetro que está interno no software. O comando toc lê o tempo decorrido desde que a função iniciou o temporizador do cronômetro. Contudo, esta ferramenta do Matlab é capaz de ler o tempo interno no momento da execução da função e exibe o tempo decorrido desde a última chamada de uma função Mathworks (2021).

3.3. Simulador ModelSim-Altera 10.1d

A fim de atingir os objetivos propostos no trabalho foi utilizado o simulador ModelSim-Altera 10.1d. Para implementação do algoritmo foi utilizada a linguagem VHDL.

O ModelSim é um simulador de HDL (*Hardware Description Language*) desenvolvido pela Mentor Graphics. Ele é conhecido por oferecer alto desempenho e facilidade de uso. Ele conta com uma interface gráfica que permite identificar e depurar rapidamente problemas, auxiliados por janelas atualizadas dinamicamente. O ModelSim suporta os padrões atuais de linguagem VHDL e Verilog (MICROCHIP, 2020).

A linguagem VHDL foi criada por volta da década de 1980 como parte de um programa de pesquisa do Departamento de Defesa dos Estados Unidos, o programa chamado de Very High Speed Integrated Circuits, visava a construção de circuitos eletrônicos de alta velocidade, contudo, em um dos seus resultados foi a especificação, padronização, e posterior implementação da linguagem VHSIC-HDL, ou seja, linguagem de Descrição de Hardware de Circuitos Integrados com Altíssima Velocidade Hexsel (2021).

O VHDL possui suas particularidades, ao contrário de linguagens de programação como C++, C ou assembly, nas quais as instruções são executados na ordem em que aparecem no código fonte, ou seja, sequencialmente, em VHDL a computação das expressões e as atribuições de valores ocorrem em paralelo, com um efeito que emula a propagação de sinais num circuito. Nesta linha, o código VHDL é declarativo, ou seja, o programa descreve o resultado desejado sem listar explicitamente os comandos ou etapas que devem ser executados. Ao invés de imperativo, como em C Hexsel (2021).

O software ModelSim disponibiliza uma janela Wave com todos os recursos de alto desempenho. A janela Wave fornece cursores para marcar pontos interessantes no tempo e medir a distância de tempo entre os cursores. O conteúdo da janela do Wave pode ser formatado de maneira flexível por definições e agrupamentos de sinais. As comparações de formas de onda podem ser facilmente realizadas entre dois resultados de simulação Siemens (2019).

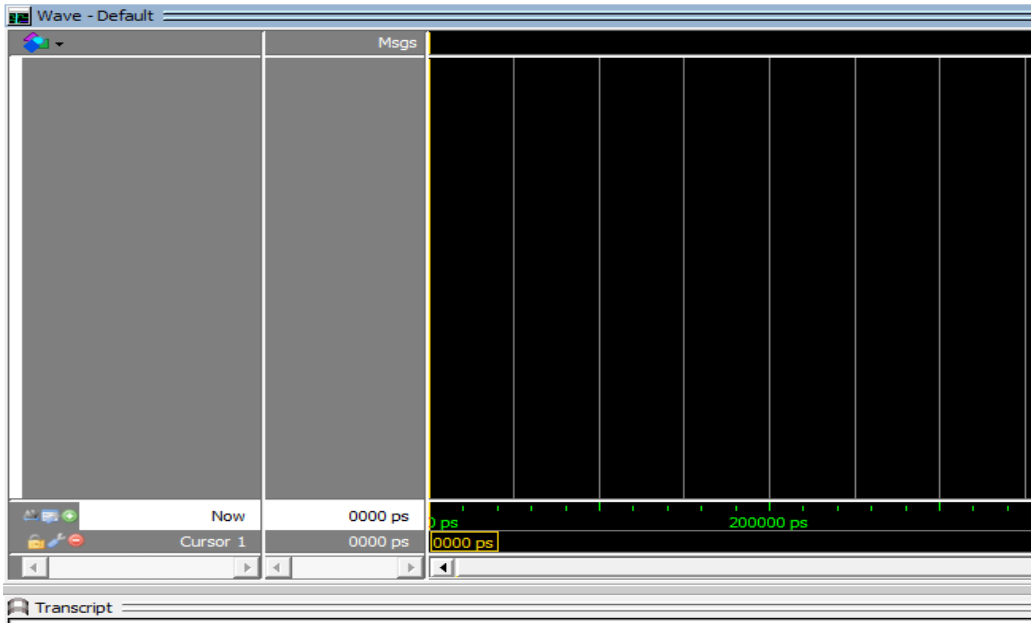

Figura 6 - Janela Wave.

A Figura 6 mostra a janela Wave do software ModelSim.

4. RESULTADOS E DISCUSSÃO

A seguir é mostrado a interface gráfica de cada plataforma supracitada neste trabalho, com a resposta do algoritmo M frente e central.

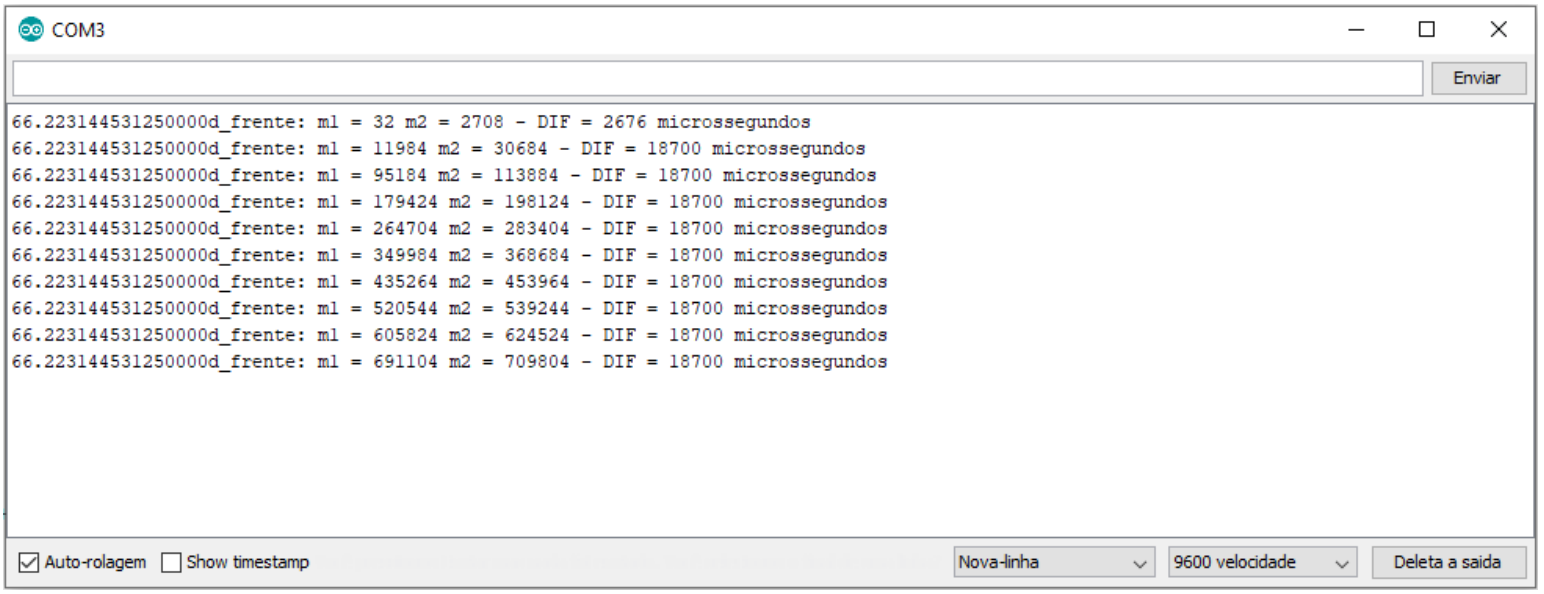

Figura 7 - Resposta do algoritmo M frente.

A Figura 7 mostra a interface gráfica do Arduino com a resposta do algoritmo M frente e o tempo de execução gasto. A função utilizada foi: $5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$.

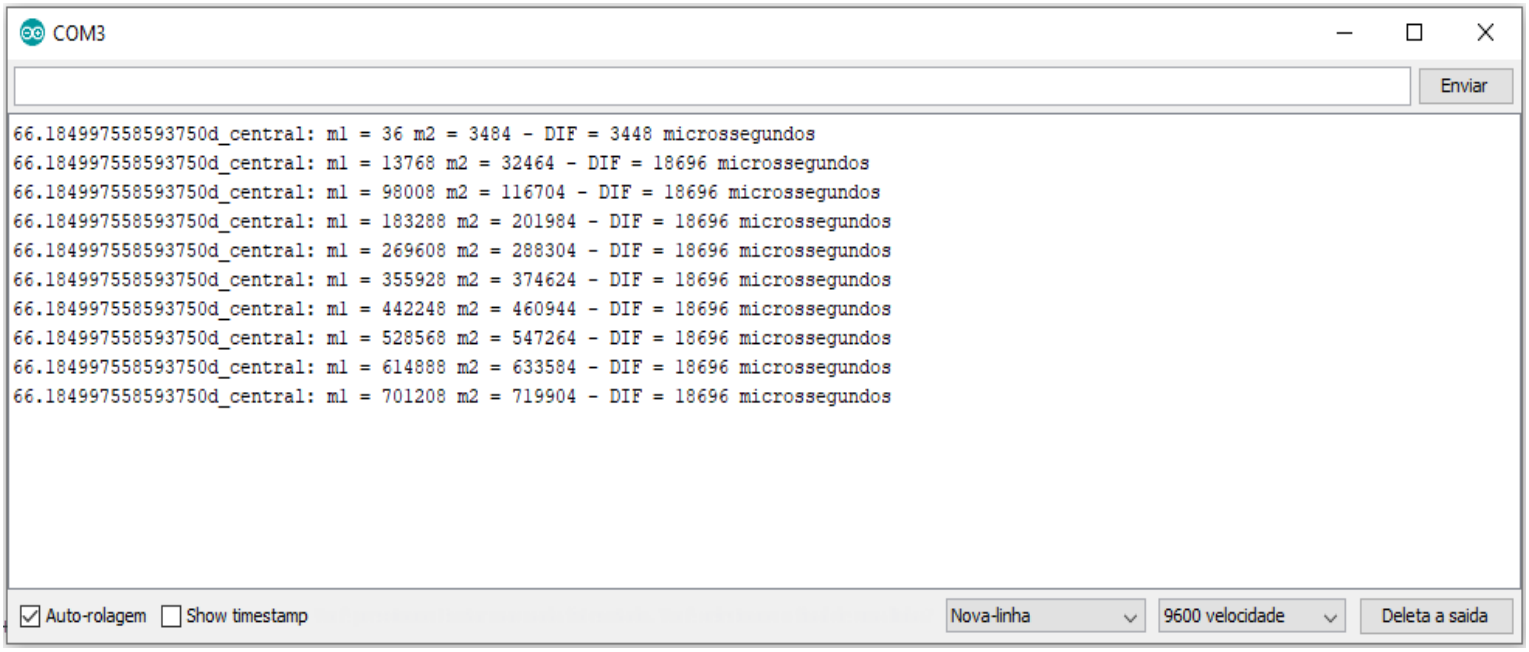

Figura 8 - Resposta do algoritmo M central.

A Figura 8 mostra a interface gráfica do Arduino com a resposta do algoritmo M central e o tempo de execução gasto. A função utilizada foi: $5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$.

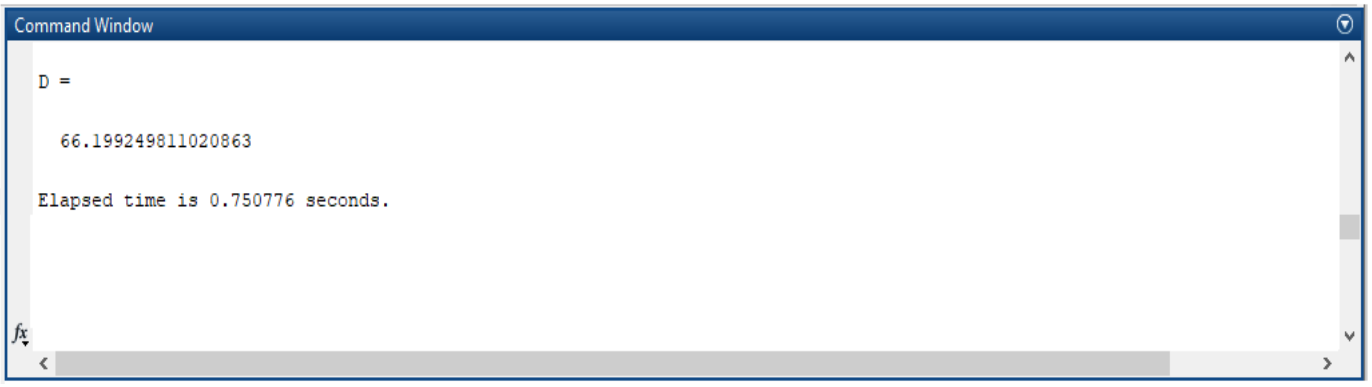

Figura 9 - Resposta do algoritmo M frente.

A Figura 9 mostra a interface gráfica do Matlab com a resposta do algoritmo M frente e o tempo de execução gasto. A função utilizada foi: $5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$.

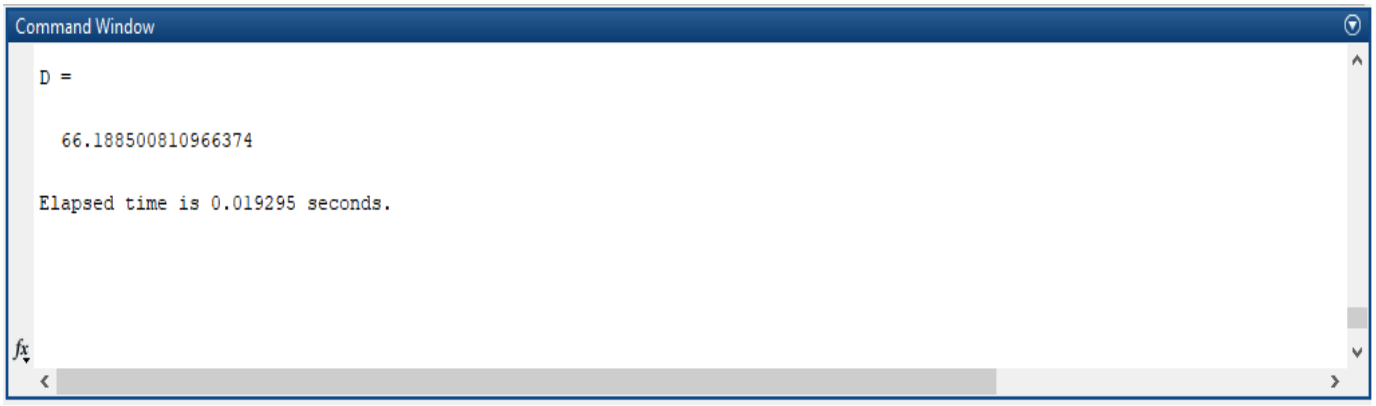

Figura 10 - Resposta do algoritmo M central.

A Figura 10 mostra a interface gráfica do Matlab com a resposta do algoritmo M central e o tempo de execução gasto. A função utilizada foi: $5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$.

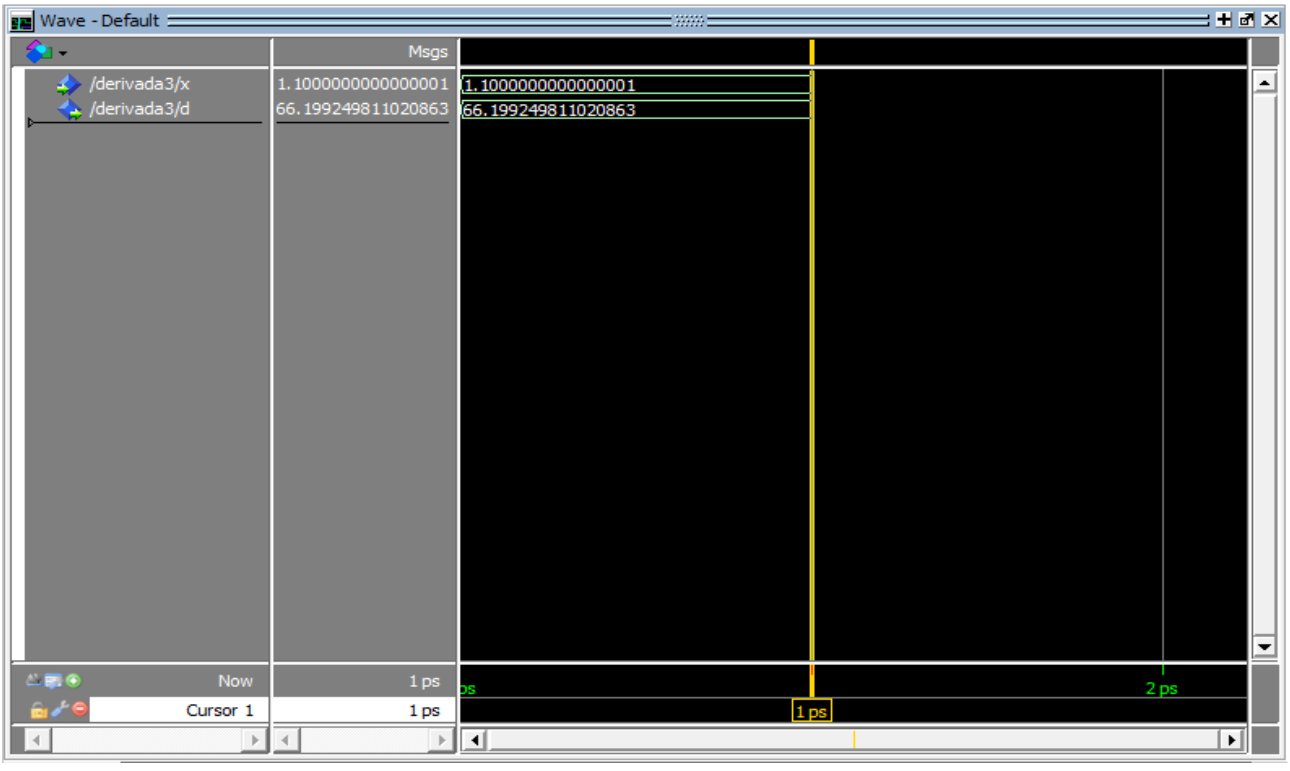

Figura 11 - Resposta do algoritmo M frente.

A Figura 11 mostra a interface gráfica do simulador ModelSim com a resposta do algoritmo M frente e o tempo de execução gasto. A função utilizada foi: $5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$.

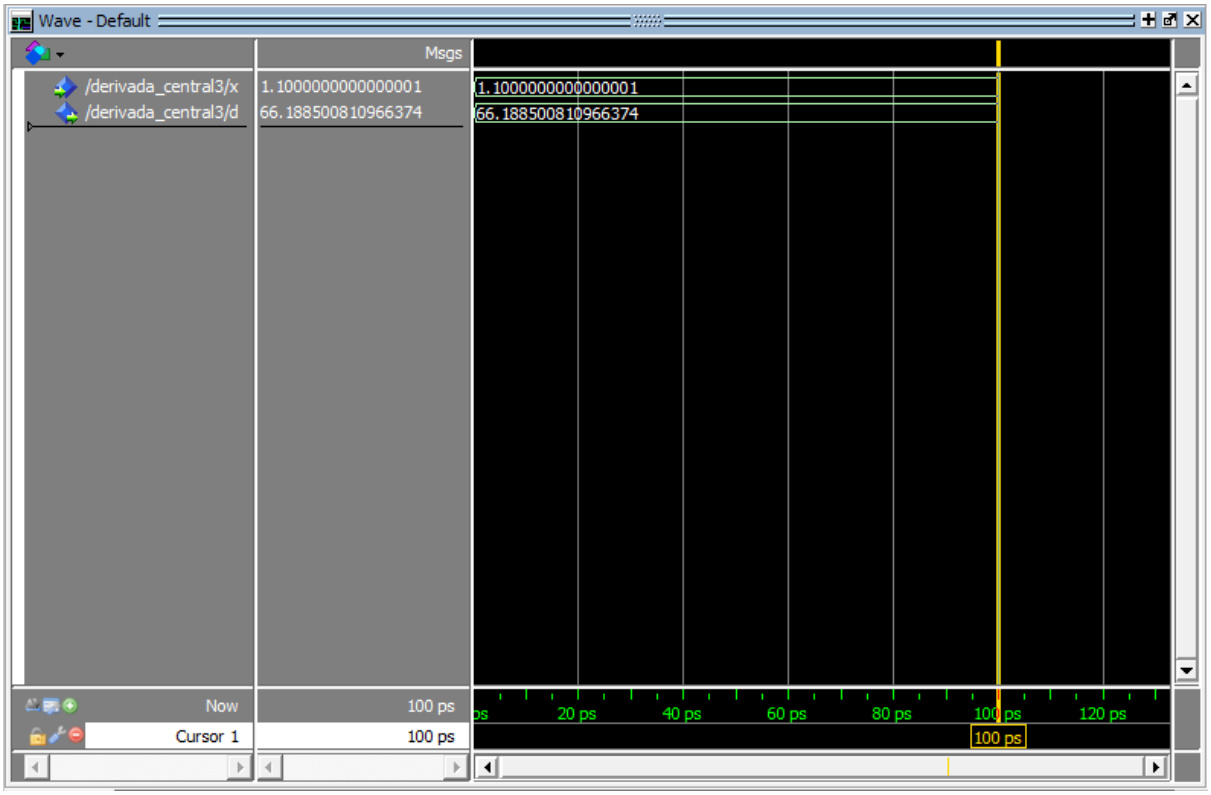

Figura 12 - Resposta do algoritmo M central.

A Figura 12 mostra a interface gráfica do simulador ModelSim com a resposta do algoritmo M central e o tempo de execução gasto. A função utilizada foi: $5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$.

4.1. Gráficos referentes ao tempo de execução

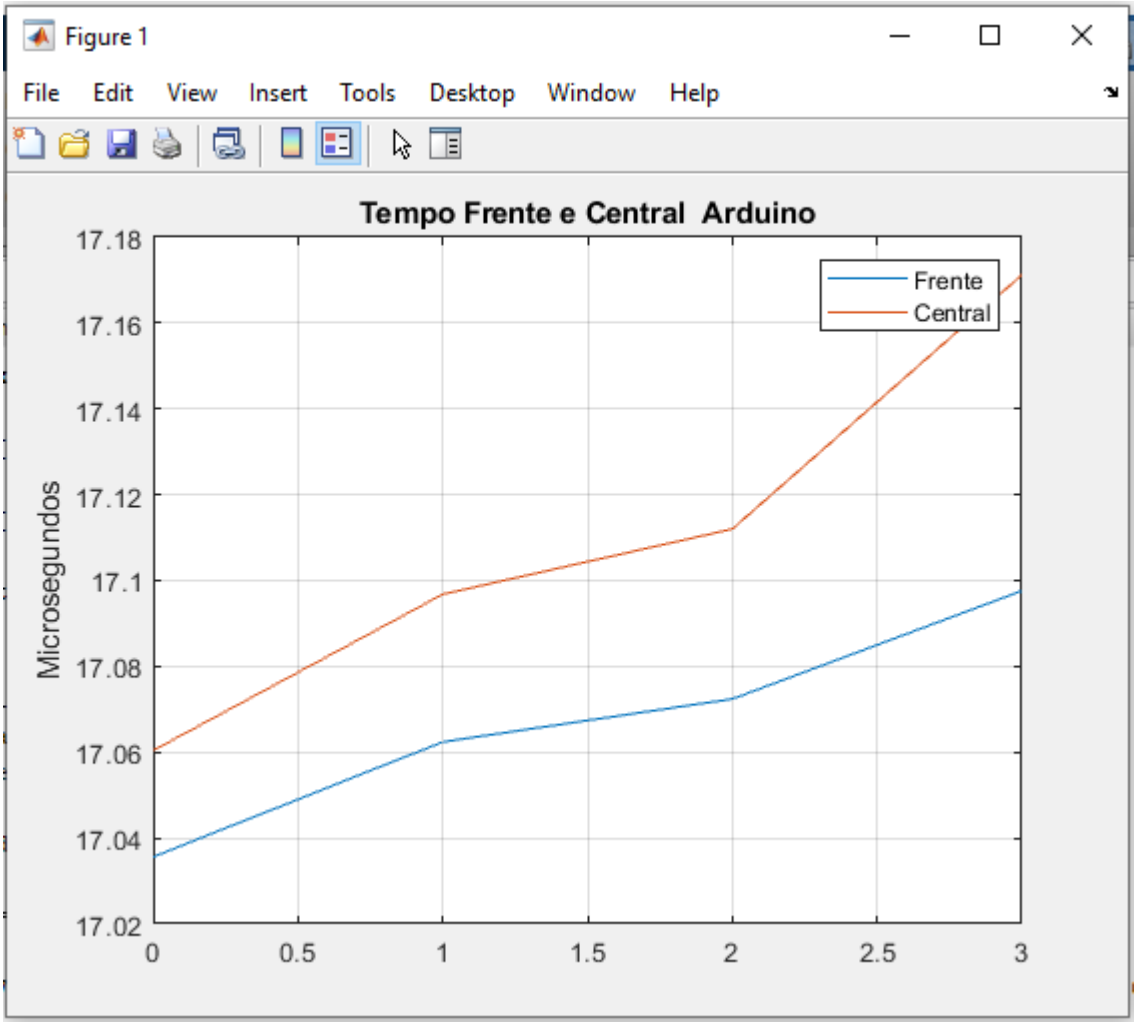

Figura 13 - Gráfico da plataforma Arduino.

A Figura 13 mostra o gráfico do tempo de execução do algoritmo M frente e central, para as funções testadas neste trabalho para a plataforma Arduino. As funções testadas no trabalho serão mostradas nas tabelas a seguir.

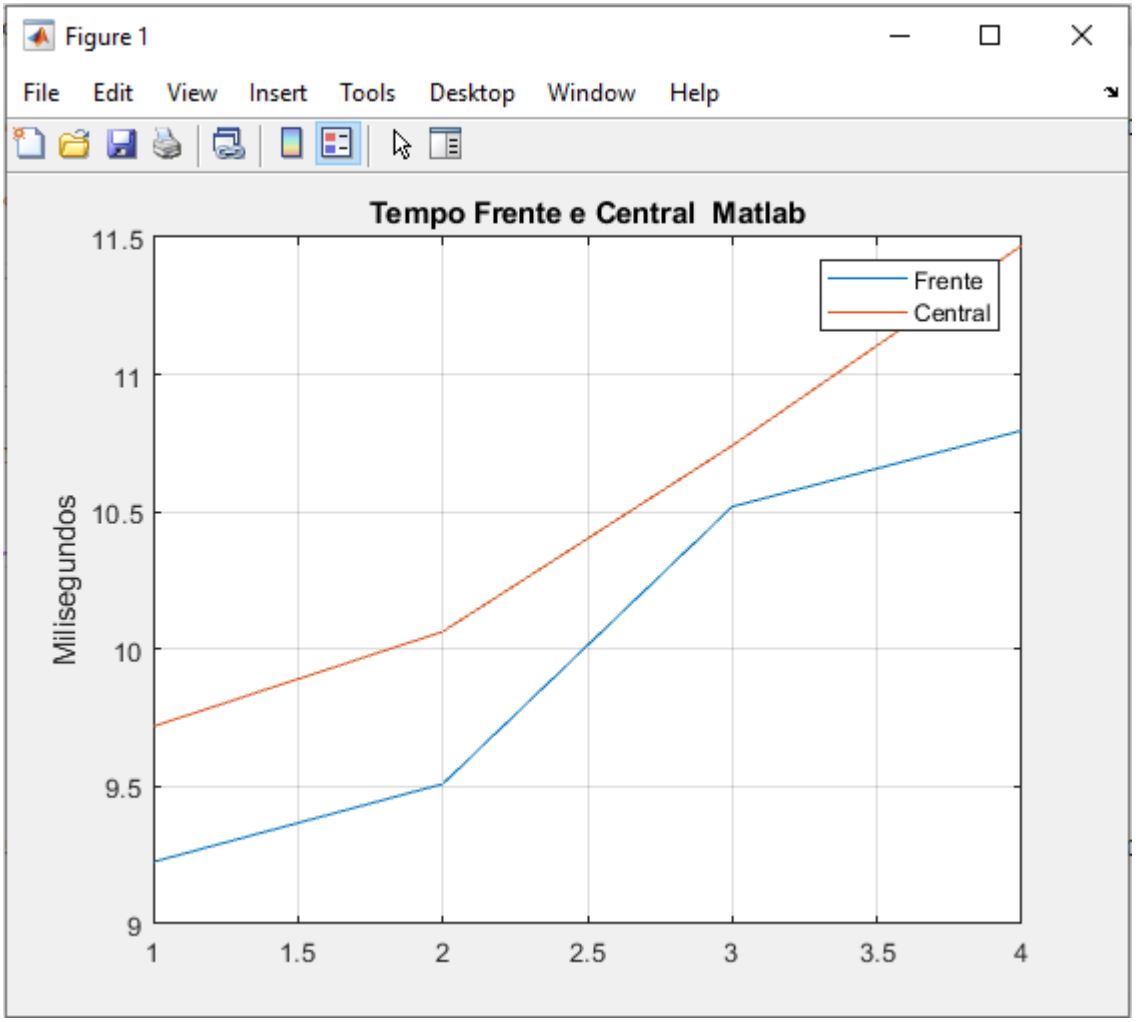

Figura 14 - Gráfico do software Matlab.

A Figura 14 mostra o gráfico do tempo de execução do algoritmo M frente e central, para as funções testadas neste trabalho, referente ao software Matlab.

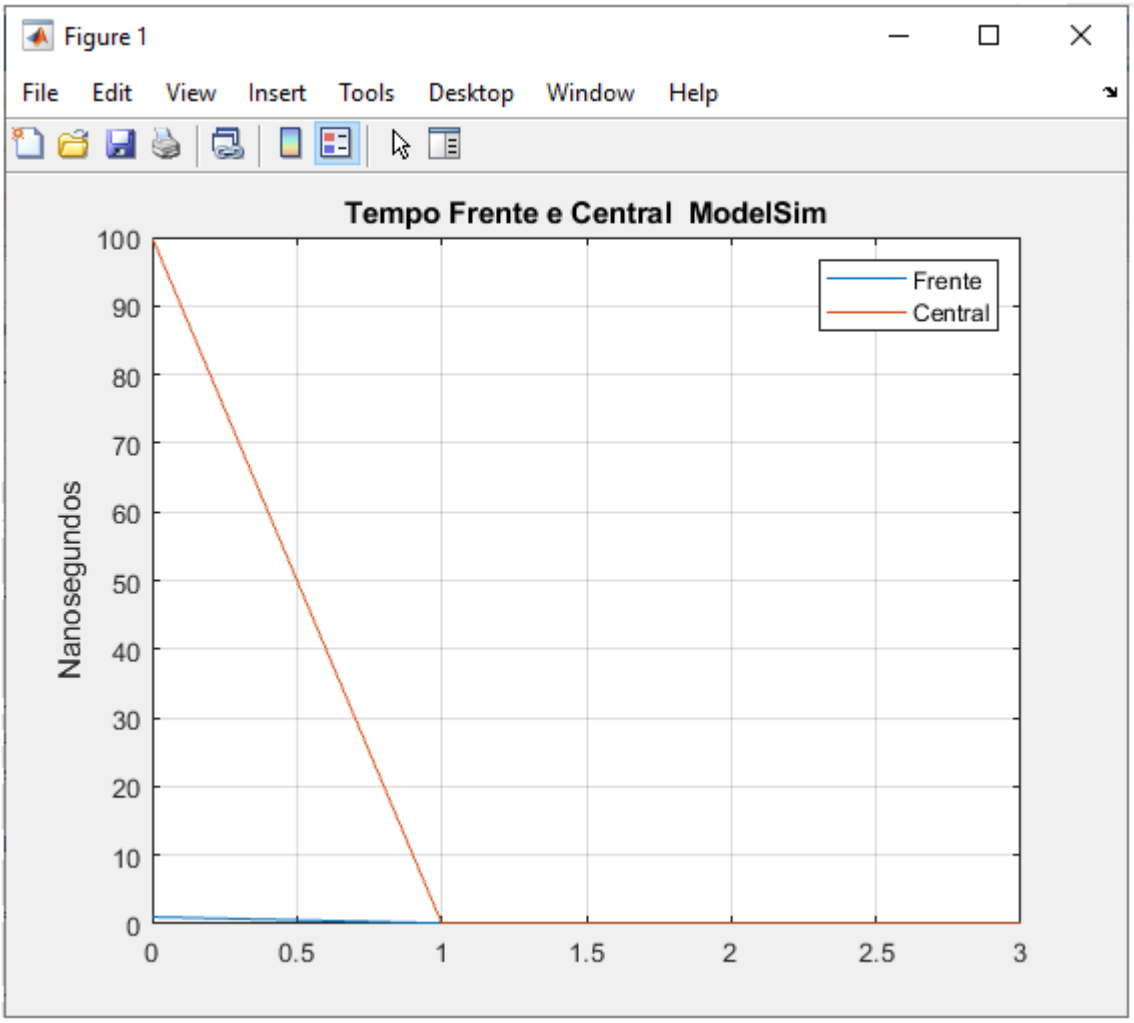

Figura 15 - Gráfico do simulador ModelSim.

A Figura 15 mostra o gráfico do tempo de execução do algoritmo M frente e central, para as funções testadas neste trabalho, referente ao simulador ModelSim.

A seguir são apresentados os resultados do algoritmo M para frente e central no arduino, matlab e no simulador modelsim.

As Tabelas 1,2 e 3 mostram quatro funções de entrada para o algoritmo M, com seus respectivos graus, ordenados em ordem crescente. Logo após é mostrado a derivada de cada função com um ponto x onde se quer a derivada. Em seguida é calculado o valor exato da derivada da função de entrada.

Posteriormente é apresentado o tempo de execução e a saída do algoritmo M frente e central, para respectivas funções. Com a resposta do algoritmo M frente e central foi possível calcular o erro absoluto, relativo e relativo percentual com base no valor exato da derivada das respectivas funções.

Tabela 1 - Resultados do algoritmo M no controlador Arduino.

	Função de Entrada			
	$2x^2 + 3x + 1$	$x^3 + 7x^2 - 6x + 1$	$3x^4 + 2x^3 + x^2 + 2x + 1$	$5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$
	Derivada da Função de Entrada			
	$4x + 3$ para x = 3.5; **17.000000000000000**	$3x^2 + 14x - 6$ para x = 2.1; **36.630000000000003**	$12x^3 + 6x^2 + 2x + 2$ para x = 1.5; **59.000000000000000**	$25x^4 + 16x^3 + 9x^2 + 4x - 7$ para x = 1.1; **66.188500000000033**
	Algoritmo M (Frente)			
Tempo (μ)	**17.035,6**	**17.062,4**	**17.072,4**	**17.097,6**
Resposta	**17.002105712890625**	**36.563873291015625**	**57.010646820068359**	**66.223144531250000**
Erro absoluto	**0.002105712890625**	**0.066126708984378**	**1.989353179931641**	**0.034644531249967**
Erro Relativo	**1.238654641544118 e-04**	**0.001805260960534**	**0.033717850507316**	**5.234222145836056e-04**
Erro Relativo Percentual	**0.012386546415441 %**	**0.180526096053448 %**	**3.371785050731595 %**	**0.052342221458361 %**
	Algoritmo M (Central)			
Tempo (μ)	**17.060,4**	**17.096,8**	**17.112**	**17.171,2**
Resposta	**17.000198364257812**	**36.573410034179687**	**57.001110076904296**	**66.184997558593750**
Erro absoluto	**1.983642578125000 e-04**	**0.056589965820315**	**1.998889923095703**	**0.003502441406283**
Erro Relativo	**1.166848575367647 e-05**	**0.001544907611802**	**0.033879490221961**	**5.291616226811302e-05**
Erro Relativo Percentual	**0.001166848575368 %**	**0.154490761180200 %**	**3.387949022196100 %**	**0.005291616226811 %**

Tabela 2 - Resultados do algoritmo M no software Matlab.

	Função de Entrada			
	$2x^2 + 3x + 1$	$x^3 + 7x^2 - 6x + 1$	$3x^4 + 2x^3 + x^2 + 2x + 1$	$5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$
	Derivada da Função de Entrada			
	$4x + 3$ para x = 3.5; **17.000000000000000**	$3x^2 + 14x - 6$ para x = 2.1; **36.630000000000003**	$12x^3 + 6x^2 + 2x + 2$ para x = 1.5; **59.000000000000000**	$25x^4 + 16x^3 + 9x^2 + 4x - 7$ para x = 1.1; **66.188500000000033**
	Algoritmo M (Frente)			
Tempo (*ms*)	**9.2255**	**9.5093**	**10.519**	**10.7964**
Resposta	**17.001999999997960**	**36.631330010052906**	**57.005050200018538**	**66.199249811020863**
Erro absoluto	**0.001999999997960**	**0.001330010052904**	**1.994949799981462**	**0.010749811020830**
Erro Relativo	**1.176470587035294 e-04**	**3.630931075358995 e-05**	**0.033812708474262**	**1.624120658547934e-04**
Erro Relativo Percentual	**0.011764705870353 %**	**0.003630931075359 %**	**3.381270847426200 %**	**0.016241206585479 %**
	Algoritmo M (Central)			
Tempo (*ms*)	**9.7186**	**10.0639**	**10.7413**	**11.4699**
Resposta	**16.999999999999456**	**36.630000010067207**	**57.000000199991497**	**66.188500810966374**
Erro absoluto	**5.435651928564766 e-13**	**1.006720395935190 e-08**	**1.999999800008503**	**8.109663411914880e-07**
Erro Relativo	**3.197442310920451 e-14**	**2.748349429252498 e-10**	**0.033898301695059**	**1.225237527956499e-08**
Erro Relativo Percentual	**3.197442310920451 e-12 %**	**2.748349429252498 e-08 %**	**3.389830169505900 %**	**1.225237527956499e-06 %**

Tabela 3 - Resultados do algoritmo M no simulador ModelSim.

	Função de Entrada			
	$2x^2 + 3x + 1$	$x^3 + 7x^2 - 6x + 1$	$3x^4 + 2x^3 + x^2 + 2x + 1$	$5x^5 + 4x^4 + 3x^3 + 2x^2 - 7x + 48$
	Derivada da Função de Entrada			
	$4x + 3$ para x = 3.5; **17.000000000000000**	$3x^2 + 14x - 6$ para x = 2.1; **36.630000000000003**	$12x^3 + 6x^2 + 2x + 2$ para x = 1.5; **59.000000000000000**	$25x^4 + 16x^3 + 9x^2 + 4x - 7$ para x = 1.1; **66.188500000000033**
	Algoritmo M (Frente)			
Tempo (*ns*)	**1**	**0.1**	**0.1**	**0.001**
Resposta	**17.00199999999796**	**36.631330010052906**	**57.005050200018538**	**66.199249811020863**
Erro absoluto	**0.001999999997960**	**0.001330010052904**	**1.994949799981462**	**0.010749811020830**
Erro Relativo	**1.176470587035294 e-04**	**3.630931075358995 e-05**	**0.033812708474262**	**1.624120658547934e-04**
Erro Relativo Percentual	**0.011764705870353 %**	**0.003630931075359 %**	**3.381270847426200 %**	**0.016241206585479 %**
	Algoritmo M (Central)			
Tempo (*ns*)	**100**	**0.1**	**0.1**	**0.1**
Resposta	**16.999999999999456**	**36.630000010067207**	**57.000000199991497**	**66.188500810966374**
Erro absoluto	**5.435651928564766 e-13**	**1.006720395935190 e-08**	**1.999999800008503**	**8.109663411914880e-07**
Erro Relativo	**3.197442310920451 e-14**	**2.748349429252498 e-10**	**0.033898301695059**	**1.225237527956499e-08**
Erro Relativo Percentual	**3.197442310920451 e-12 %**	**2.748349429252498 e-08 %**	**3.389830169505900 %**	**1.225237527956499e-06 %**

O tempo de execução do algoritmo M no controlador arduino foi da ordem de microsegundos, no software matlab milissegundos e no simulador Modelsim, nanosegundos.

Como mostrado nas Figuras 13,14 e 15, o tempo de execução do algoritmo M central foi maior em cada plataforma de teste. Contudo ao se realizar os cálculos de erros para cada plataforma supracitada, observa-se que o algoritmo M central teve um erro relativo percentual menor, do que o algoritmo M frente, esse indicativo mostra neste que, em comparação com o algoritmo M frente, o algoritmo M central conseguiu ser melhor em termos de exatidão acerca do valor exato da derivada da função de entrada.

Contudo, para a exatidão do algoritmo M central, exige-se da plataforma de teste maior processamento, com isso o tempo de execução acaba sendo maior conforme observado nas Figuras 13, 14 e 15.

5. CONCLUSÃO

5.1. Considerações finais

Na computação a busca por procedimentos computacionais bem definidos que resolvam problemas com maior desempenho tem se tornado cada vez mais frequente.

Em termos computacionais, a derivada pode ser implementada na forma de algoritmo para resolver diversos problemas do cotidiano. A derivada possui diversas aplicações em diversas áreas como: física, biologia, administração. Contudo, para implementação é preciso o uso de plataformas que oferecem suporte para o cálculo da derivada como o matlab por exemplo.

Portanto este trabalho, visou realizar a implementação e avaliação do desempenho do algoritmo M para o cálculo da derivada, haja vista que, não se foi encontrado na literatura um algoritmo que realiza-se o cálculo da derivada considerando o decréscimo da hipotenusa.

Os objetivos gerais e específicos do trabalho foram cumpridos, e com os resultados deste trabalho, pode-se indicar qual plataforma utilizar para determinados tipos de problemas, que envolvem o cálculo da derivada.

Caso o problema a ser analisado envolva como um requisito uma precisão não tão exata, pode-se utilizar o arduino que é uma plataforma de baixo custo e que atende as expectativas. Caso o problema solicite uma precisão mais alta pode-se utilizar o matlab ou o modelsim em um hardware com maior capacidade de processamento.

5.2. Proposta para trabalho futuro

Como foi realizado o trabalho com o algoritmo em tempo contínuo, uma sugestão seria:

- Realizar aplicações do algoritmo para controle analógico.

6. REFERÊNCIAS

Araújo, Miscelânea. Métricas de Avaliação de Algoritmos de Otimização. Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 1, 2015. Disponível em: <https://proceedings.sbmac.org.br/sbmac/article/view/757/763/> Acesso em: 20 mar. 2021.

ARDUINO, 2018. disponível em: <www.arduino.cc>, acesso em: 22 mar. 2021.

CHAPMAN, Stephen J. Programação em Matlab para Engenheiros, Cengage Learning, 2017.

CORMEN, T. H. et al. Introduction to Algorithms. [S.l.]: MIT Press, 2009.

GILAT, A. Métodos Numéricos para Engenheiros e Cientistas: Uma introdução com aplicações usando MATLAB, Bookman, 2008.

HEXSEL, Roberto. A. Uma Breve Introdução à VHDL. UFPR, 2021. p. 275 - p.340.

JUSTO, Dagoberto A. R. et al. Cálculo Numérico. Creative Commons Atribuição-CompartilhaIgual 3.0, 2018.

LINDER, Marcelo Santos. Noções de complexidade de algoritmos.22 jun. 2008. Notas de Aula.

MARCHETTO, Raquel. Utilização do software MATLAB como recurso tecnológico de aprendizagem na transformação de matrizes em imagens. REVEMAT. Florianópolis (SC), v.11, n. 1, p. 120-130, 2016. Disponível em: <http: www.periodicos.ufsc.br/>. Acesso em: 24 mar. 2021.

MENEZES, Marco A. F.; MACULAN, Nelson; BUENO, Elivelton F. Um Algoritmo para Diferenciação Numérica. 02 feb. 2018, 28 jul. 2018. Notas de aula.

MICROCHIP. ModelSim. 2020. Disponível em: <https://www.microsemi.com/product-directory/dev-tools/4900-modelsim#documents /> Acesso em: 11 mai. 2021.

NETO, Alberto Costa. Complexidade de Algoritmos. 10 de dez. de 2012. albertocn.sytes.

RODRIGUES, L et al. Intodução ao Arduino. Fundação Universidade Federal de Mato Grosso do Sul - UFMS. Campo Grande: 2012.

S. Lang Analysis I. Second printing, Addison-Wesley Publishing Company, 1968.

SANTANA, Anderson, Marcolino de. Aplicação das Derivadas. UNIR. JI-Paraná: 2010. Disponível em: < http://www.dmejp.unir.br/menus_arquivos/1787_anderso_marcolino.pdf>. Acesso em 21 mar. 2021.

SILVA, Paulo J. S. Fórmula de Taylor e Aproximação de Derivadas. 12 de set. de 2017. 7p. Ime.unicamp.

Tic Toc. Mathworks, 2021. Disponível em: <https://www.mathworks.com/help/matlab/ref/tic.html/>. Acesso em: 22 mar. 2021.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Algoritmos: Teoria e Prática. 3a edição. Elsevier, 2012. ISBN 9788535236996

Apêndice

Arduino frente

```
void setup() {
  Serial.begin(9600);
conta_tempo(frente, "d_frente");            // chamada a função para cálculo do tempo
}
void conta_tempo(void(*p)(), String descricao)
{
  long m1, m2;                              // Variáveis auxiliares
  m1 = micros();                            // tempo inicial
  (*p)();                                   // passagem do bloco a ser analisado
  m2 = micros();                            // tempo final

  Serial.print(descricao + ": m1 = ");
  Serial.print(m1);
  Serial.print(" m2 = ");
  Serial.print(m2);
  Serial.print(" - DIF = ");
  Serial.print(m2 - m1);
  Serial.println(" microssegundos" );
}
void frente()
{
  float x =  //ponto, h = 1.0;
  float ep = 0.001;
  float t = //f(x);                          //variáveis auxiliares
  float k = 0.0;
  float y;
  float delta, D;
                                  // inicialização das variáveis na repetição
  y = f(x+h);
  delta = (h * h) + (y - t) * (y - t);      //cálculo do delta inicial

  while (delta > ep){
    h = 0.1 * h;
```

```
    y = f(x+h);
    delta = (h * h) + (y - t) * (y - t);
   k = k + 1;
  }
  D = (y - t) / h;                    // saída com o valor da derivada no ponto x
  Serial.print(D,15);
}

void loop() {
  conta_tempo(frente, "d_frente");
}
```

Arduino Central

```
void setup() {
  Serial.begin(9600);
conta_tempo(central, "d_central");          // chamada a função para cálculo do tempo

}
void conta_tempo(void(*p)(), String descricao)
{
  long m1, m2;                            // Variáveis auxiliares
  m1 = micros();                          // tempo inicial
  (*p)();                                 // executa a função
  m2 = micros();                          // tempo final

  Serial.print(descricao + ": m1 = ");
  Serial.print(m1);
  Serial.print(" m2 = ");
  Serial.print(m2);
  Serial.print(" - DIF = ");
  Serial.print(m2 - m1);
  Serial.println(" microsssegundos" );
}

void central()
```

```
{
  float x =  //ponto, h = 1.0;
  float ep = 0.001;                                    //variáveis auxiliares
  float k = 0.0;
  float y_p, y_n;
  float delta, D;

  y_p = //f(x+h)                                       // ponto a direita
  y_n = //f(x-h)                                       // ponto a esquerda
  delta = (h * h) + 0.25 * (y_p - y_n) * (y_p - y_n);  //cálculo do delta inicial

  while (delta > ep){
    h = 0.1 * h;
     y_p = //f(x+h)
     y_n = //f(x-h)
     delta = (h * h) + 0.25 * (y_p - y_n) * (y_p - y_n);
    k = k + 1;
  }
  D = (y_p - y_n) / (2*h);                             // saída com o valor da derivada no ponto x
  Serial.print(D, 15);
}

void loop() {
  conta_tempo(central, "d_central");
}
```

Matlab frente

```
tic                          % início do cronômetro
x = //ponto;                 % ponto inicial onde se quer a derivada
h = 1;                       % Incremento inicial do h
eps = 1.e-3;                 %  tolerância

t = //f(x)                   %  função de entrada a ser derivada

k = 0;                       % variável auxiliar
```

```
y = //f(x+h)                    % função de entrada com o incremento inicial
delta = (h*h) + (y-t)*(y-t);    % cálculo do delta inicial

      while (delta > eps)
            h = h * 0.1;
            y = //f(x+h)
            delta = (h*h) + (y-t)*(y-t);
            k = k+1;
      end
      format long
      D = (y-t) / h             % saída com o valor da derivada no ponto x
G = toc;                        % final cronômetro
```

Matlab Central

```
tic                             % início do cronômetro
x = //ponto                     % ponto inicial onde se quer a derivada
h = 1;                          % Incremento inicial do h
eps = 1.e-3;                    %  tolerância

k = 0;                          % variável auxiliar
                                % iniciação das variáveis na repetição
y_p = //f(x+h)                  % ponto a direita
y_n = //f(x-h)                  % ponto a esquerda

delta = (h*h) + 0.25 * (y_p - y_n) * (y_p - y_n);          % cálculo do delta inicial

while  (delta > eps)

      h  = h * 0.1;
      y_p = //f(x+h)
      y_n = //f(x-h)
      delta = (h*h) + 0.25 * (y_p - y_n) * (y_p - y_n);
      k = k + 1;
end
format long
```

```
D = (y_p - y_n) / (2*h)                    % saída com o valor da derivada no ponto x
G = toc;                                   % final cronômetro
```

ModelSim frente

```
                                                    -- “--” comentário
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;                          -- Declaração de bibliotecas
use IEEE.std_logic_unsigned.all;

entity derivada is                                 -- Declaração de entidade
 port( x : in real; d : out real);        -- Declaração de entradas e saídas para o cálculo
end entity derivada;                               -- Fim da declaração da entidade

architecture hardware of derivada is               -- Declaração da arquitetura
begin
derivativee : process(x) is                        -- início do processo
        variable eps   : real;
        variable t     : real;                     -- Declaração de variáveis auxiliares
        variable h     : real;
        variable y     : real;
        variable delta  : real;
        variable k     : natural;
  begin
        h := 1.0;                                  -- Incremento inicial do h
        eps := 0.001;                              -- tolerância
        t := f(x);                                 -- função de entrada

                                                   -- iniciação das variáveis na repetição
        k := 0;
        y := f(x+h);                       -- função de entrada com o incremento inicial
        delta := (h*h) + (y-t)*(y-t);      -- cálculo do delta inicial

        while delta > eps loop
                h := h * 0.1;
                y := f(x+h);
```

```
            delta := (h*h) + (y-t)*(y-t);
            k := k+1;
        end loop;
        d <= (y-t) / h;                    -- saída com o valor da derivada no ponto x
   end process derivativee;                -- final do processo
end hardware;                              -- fim da arquitetura
```

ModelSim Central

```
                                           -- "--" comentário
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;                  -- Declaração de bibliotecas
use IEEE.std_logic_unsigned.all;

entity derivada_central is                 -- Declaração de entidade
   port( x: in real; d : out real);        -- Declaração de entradas e saídas para o cálculo
end entity derivada_central;               -- Fim da declaração da entidade

architecture hardware of derivada_central is    -- Declaração da arquitetura
begin
derivativee_central : process(x) is        -- início do processo
            variable h      : real;
```

```
        variable eps   : real;
        variable y_p   : real;                  -- Declaração de variáveis auxiliares
        variable y_n   : real;
        variable delta  : real;
        variable k      : natural;
begin
        h := 1.0;                               -- Incremento inicial do h
        eps := 0.001;                           -- tolerância
        k := 0;
                                -- iniciação das variáveis na repetição
        y_p := f(x+h);                  --ponto a direita
        y_n := f(x-h);                  --ponto a esquerda
        delta := (h*h) + 0.25*(y_p - y_n)*(y_p - y_n);      -- cálculo do delta inicial

        while delta > eps loop
                h := h * 0.1;
                y_p := f(x+h);
                y_n := f(x-h)
                delta := (h*h) + 0.25*(y_p - y_n)*(y_p - y_n);
                k := k+1;
        end loop;
        d <= (y_p - y_n) / (2.0*h);   -- saída com o valor da derivada no ponto x
  end process derivativee_central;          -- final do processo
end hardware;                               -- fim da arquitetura
```