PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

OSCAR REMIGIUS ALBRECHT FILHO

Cartografia do Processamento Numérico em Ponto Flutuante

Porto Alegre
2009

OSCAR REMIGIUS ALBRECHT FILHO

Cartografia do Processamento Numérico em Ponto Flutuante

Proposta referente à disciplina de Trabalho de Conclusão II do curso de Bacharelado em Ciência da Computação da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Profa. Beatriz Regina Tavares Franciosi

Porto Alegre
2009

RESUMO

Os números de ponto flutuante buscam representar um conjunto infinito de números com uma quantidade finita de bits. Contudo, isto não é possível pois não há como representar um conjunto infinito em precisão finita. Sendo assim, quando os números que se deseja representar não podem ser exatamente representados por números de ponto flutuante, armazena-se o valor aproximado deste número obtido a partir da utilização de um dos tipos de arredondamento. Quando isto acontece o número é associado a um erro de arredondamento que, geralmente, não é muito grande mas que pode influenciar no resultado final de operações onde esteja envolvido. Assim, qualquer cálculo com números de ponto flutuante está sujeito à ocorrência do erro de arredondamento.

Este trabalho visa desenvolver um benchmark que, no cenário de avaliação de polinômios, possibilita a comparação de métodos de avaliação polinomial em relação ao tempo necessário para executar o método e a qualidade do resultado obtido. Através da análise dos resultados obtidos com o benchmark, tentaremos fazer apresentar a cartografia do processamento de ponto flutuante e, se esta existir, regras gerais para o comportamento do erro de arredondamento.

Palavras-Chave: Número de ponto flutuante. Benchmark. Processamento numérico. Erro de arredondamento. Exatidão. Precisão.

ABSTRACT

Floating-point numbers aim to represent an infinite set of numbers in a finite number of bits. This is not possible, there is no way to represent all numbers in this set with a limited quantity of bits. Because of that, if the number cannot be represented by a floating point number, an approximated number will be stored instead. This is a type of rounding error. The difference between both numbers can be very small, but once this approximated number is used to perform new calculations, the rounding error may influence the final result.

Any floating-point calculation may produce rounding error. This work aims to propose a benchmark that makes the comparison of evaluation methods possible, in the polynomial evaluation scenario. With the analysis of the benchmark's result, we will try to map the behavior of floating-point processing, if there is any, we will search for rules for the behavior of rounding mode.

Key-words: Floating-point number. Benchmark. Numerical processing. Rounding error. Acurracy. Precision.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

No dia a dia, confiamos cegamente nos computadores para fazer os mais variados tipos de cálculos, desde somas no supermercado e juros bancários até cálculos complexos de previsão do tempo. Estes resultados são interpretados como corretos, mas isto nem sempre é verdade.

Trabalhar com números de ponto flutuante pode levar a resultados inexatos. Um dos casos mais comuns se refere ao arredondamento. Tome, por exemplo, o cálculo do rendimento de uma caderneta de poupança. Muitas vezes o valor do rendimento possui mais do que dois dígitos de mantissa. Então, é arredondado para adequar-se ao sistema monetário nacional. Se o critério de arredondamento adotado for arredondamento para cima (ou arredondamento por excesso), a diferença entre o juro real e o juro arredondado tem que ser paga pelo banco e isso significa prejuízo para o mesmo; se o critério for arredondamento para baixo (ou arredondamento por truncamento) a diferença entre o juro real e o juro arredondado fica para o banco, ou seja, o cliente é prejudicado.

O processamento numérico em ponto flutuante exige cuidados especiais e isso, infelizmente, nem sempre acontece.

Este Trabalho de Conclusão tem por objetivo investigar a qualidade de resultado do processamento numérico em ponto flutuante. Para isto propõe-se um benchmark que tem como contexto a avaliação polinomial. Diferentes formas de avaliação polinomial serão analisadas e comparadas quanto ao tempo de execução e qualidade do resultado. Para tornar a análise mais amigável, serão utilizados recursos gráficos para a exibição dos resultados.

O trabalho está estruturado através dos seguintes itens: fundamentação teórica (capítulo 2), especificação da proposta e desenvolvimento do benchmark (capítulo 3), resultados e análise (capítulo 4) e considerações finais (capítulo 5).

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo reúne conceitos e definições resultantes da pesquisa bibliográfica realizada pelo autor.

2.1. AVALIAÇÃO POLINOMIAL

Um polinômio é uma expressão numérica expressa através de uma soma de termos, onde cada termo é o produto entre uma constante e uma potência não negativa. Para uma variável, a fórmula geral é dada por:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0,$$

onde $a_n \neq 0$, $a_n, a_{n-1}, ..., a_1, a_0$ são constantes e definem os *coeficientes* do polinômio, e *x* a variável.

A maior potência do polinômio determina o grau ou a ordem do mesmo.

Os polinômios são importantes na matemática para resolver diversos problemas e, na computação, em cálculos de *hashs e* criptografia, por exemplo.

Avaliação polinomial consiste em obter o valor numérico resultante da potência da variável do polinômio por um certo valor. Este valor numérico é estimado através da realização das operações aritméticas indicadas na expressão numérica. Por exemplo, $p(1) = 3(1)^2 - 1 + 4 = 6$ é a avaliação polinomial de $p(x) = 3x^2 - x + 4$ quando $x = 1$.

A avaliação polinomial é uma das práticas básicas mais usadas na computação. Mesmo assim, alguns dos que utilizam alguma das diversas bibliotecas de funções matemáticas disponíveis nos computadores, podem não perceber o quanto esta operação está presente.

Um polinômio pode ser expresso de diversas formas, sendo que todas são matematicamente equivalentes. Contudo, os algoritmos para avaliar cada forma podem ter complexidades diferentes. A seguir estão descritas as formas de avaliação polinomial mais utilizadas.

- **Forma das potências**: Esta forma de avaliação consiste em substituir a variável pelo valor desejado e executar o cálculo. Ela requer (2n – 1) multiplicações e n somas.

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

- **Forma de Horner**: Esta forma de avaliação consiste em re-escrever o polinômio através de um conjunto de multiplicações aninhadas em função de x, ou seja,

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = a_0 + x(a_1 + x((\ldots + x(a_n + x))).$$

Ela requer *n* multiplicações e *n* somas e é uma das mais frequentemente utilizadas por possuir baixa complexidade algébrica.

- **Forma do produto de raízes**: Expressa por $p(x) = a_0 \prod_{i=1}^{n} (x - \gamma_i)$, onde $p(\gamma_i) = 0$. Esta forma de avaliação toma por base o produto das raízes da equação algébrica associada ao polinômio. Ela requer *n* multiplicações e *n* somas.

- **Forma de Newton**: Esta forma de avaliação é expressa por

$$p(x) = c_0 (x - \beta_n)(x - \beta_{n-1})\ldots(x - \beta 1) + .. + c_{n-1}(x - \beta_1) + c_n .$$

Ela requer *n* multiplicações e *2n* somas.

- **Forma ortogonal**: Dado por $p(x) = \sum_{k=0}^{n} b_k Q_k(x)$, onde os polinômios ortogonais $Q_i(x)$ satisfazem a seguinte relação:

$$Q_{i+1}(x) = (A_i x + B_i) Q_i(x) - C_i Q_{i-1}(x),$$

onde $A_i \neq 0$, $Q_o(x) = 1$, $Q_{-1}(x) = 0$, e onde A_i, B_i, C_i podem ser dependentes de *i*, mas não de *x*. Geralmente este método requer (*3n - 1*) multiplicações e (*3n - 1*) somas.

2.2. NÚMERO DE PONTO FLUTUANTE

O número de ponto flutuante é a representação de máquina para números não inteiros e ele é representado por expansões de mantissa na forma $* 0, d_1 d_2 \ldots d_n \times b^e$, onde $* \in \{+, -\}$.

2.2.1. ERRO DE ARREDONDAMENTO

Utilizando uma precisão finita de bits, não é possível representar todos os números do conjunto de números reais. Então, o conjunto de números de ponto flutuante F representa um subconjunto próprio do conjunto de números reais $\Re$ cujos elementos são aproximações de números reais e, portanto, incluem um *erro de arredondamento.*

O *erro de arredondamento* é inerente aos valores representados em ponto flutuante pelo fato destes terem precisão finita e, por este motivo, existem diversas alternativas para controlar ou contornar a propagação de erro de arredondamento através das operações aritméticas em ponto flutuante. Contudo, nenhuma é aceita de forma unânime. Uma destas alternativas é a *matemática intervalar* que, em vez de utilizar o número real *x,* utiliza o intervalo $X = [a;b]$ de números reais o qual contém *x*, ou seja, $x \in X$. Seja $x \in [a;b]$ então $f([a;b]) = [c;d]$, onde f é uma função qualquer definida no domínio dos números reais. Assim, então o resultado $[c;d]$ contém todos os possíveis valores da avaliação de *f(x)* onde $x \in [a;b]$. Isto garante que o resultado exato da avaliação de *f(x)* esteja contido no intervalo $[c;d]$.

A *matemática intervalar* pode ser utilizada para estimar valores para os quais valores exatos não podem ser obtidos em precisão finita (Jaulin, et al., 2001). Entretanto, existem algumas operações que aumentam exageradamente o diâmetro do intervalo e isto é um problema, pois quanto maior for o diâmetro do intervalo pior é a aproximação que ele representa. Cita-se, como exemplo, uma situação onde o resultado obtido a partir de uma avaliação intervalar é $[-1000000; 1000000]$. Neste caso, o intervalo é tão grande que se perde a “exatidão” da informação que se quer conhecer.

É fato que o processamento numérico em ponto flutuante está sujeito ao *erro de arredondamento*. Considerando-se, por exemplo, que é realizado um depósito no valor de $100 todos os dias e em uma conta que renda 6% ao ano. Se *n = 365* e *i = 0,06*, o total de capital (em dólares) acumulado no final de um ano é calculado pela seguinte fórmula:

$$\frac{100\left[\left(1+\frac{i}{n}\right)^{n}-1\right]}{\frac{i}{n}}$$

Utilizando base *2* e precisão *24*, o valor obtido a partir da avaliação desta expressão numérica é dado por $37.615,45. Ao comparar o exato de $37.614,05 com o valor calculado, percebe-se que a operação em ponto flutuante está sujeita ao *erro de arredondamento*. A origem deste erro pode ser explicada da seguinte forma: a expressão $1+\frac{1}{n}$ adiciona 1 a 0,0001643836, fazendo com que os bits menos significativos de $\frac{1}{n}$ se percam. Este erro de arredondamento é, então, propagado quando este valor é elevado a n-ésima potência (Goldberg, 1991).

2.2.2. FORMATO DE PONTO FLUTUANTE

Ao longo dos tempos, diversas representações de números reais têm sido propostas, mas na computação a mais utilizada é a representação de ponto flutuante (Goldberg, 1991). Esta representação toma por base $F = F(\beta, p, e_{min}, e_{max})$, onde *β* é a base do sistema de numeração, *p* é a precisão de máquina, $e_{\min}$ e $e_{\max}$ são o maior e menor expoentes aceitos nesta configuração. Considere-se, por exemplo, a configuração $\beta = 10$ e $p = 3$. Nesta configuração, o valor 0,1 é representado por $1{,}00 \times 10^{-1}$. Outro exemplo é: se $\beta = 2$ e $p = 5$ então o número 0,1 é aproximado para 1,0001. A este valor está associado ao *erro de arredondamento*, pois 0,1 não pode ser representado exatamente nesta configuração, tornando necessário o uso de um valor aproximado.

De fato, a representação de ponto flutuante mais utilizada é a de *ponto flutuante normalizado*, ou seja, se $x \in F$ e está escrito na forma normalizada onde $x = 0, d_1 d_2 \ldots d_n * b^e$ sendo que d_1 é sempre diferente de zero e $* \in \{+,-\}$. Esta representação traz benefícios como citado em (Dalcidio):

- Ao utilizar a mantissa normalizada em base 2, o primeiro bit será sempre 1 e assim este bit (que é o mais significativo) não precisa ser representado, obtendo o ganho de um bit (bit implícito).
- O expoente é representado na forma polarizada, o que facilita comparações e evita o uso de um bit de sinal antes do expoente, auxiliando tanto a comparação quando o armazenamento.

Especificam-se números especiais, como +0 (número *subnormal*), -0 (número *unnormalized*), + ∞, - ∞ e NaN (*Not a Number*).

2.2.3. DÍGITO DE GUARDA

Os valores representados em ponto flutuante estão sujeitos ao *erro de arredondamento* e este problema só tende a aumentar se as operações aritméticas básicas – soma, subtração, multiplicação e divisão – introduzirem um *erro de arredondamento* desnecessário.

Para evitar ou reduzir a dimensão deste erro nas operações básicas, adiciona-se um dígito extra ao número, ou seja, um dígito de guarda. Este dígito faz com que a precisão necessária seja maior mas produz resultados de melhor qualidade, assim como apresentado por (Goldberg, 1991) e transcrito a seguir.

Teorema 1: Usando um formato de número de ponto flutuante com os parâmetros β e p, computando a diferença usando p dígitos, o erro relativo do resultado pode ser tão grande quanto β-1.

Quando β=2, o erro relativo pode ser tão grande quanto o resultado, e quando β =10, ele pode ser 9 vezes maior. Um dígito extra (dígito de guarda) é adicionado para evitar este tipo de problema. Neste caso, o menor número de uma subtração teria *p+1* dígitos e, no final, o resultado da subtração seria arredondado para *p* dígitos.

Teorema 2: Se *x* e *y* são números de ponto flutuante em um formato com os parâmetros β e *p*, e se a subtração é feita com *p+1* dígitos (um dígito de guarda), então o arredondamento relativo no resultado é menor que 2ε. A prova de ambos os teoremas se encontram em (Goldberg, 1991).

2.2.4. MODELO DE OPERAÇÕES BÁSICAS E O DÍGITO DE GUARDA

Para a análise de erro de arredondamento em algoritmos executados em computadores digitais, é importante dispor de informações sobre a forma como as operações básicas são realizadas.

A forma como as operações básicas são realizadas nos computadores é definida por

$$fl(x\ op\ y) = (x\ op\ y)(1+\delta) \quad |\delta| \le u \tag{1.3}$$

onde: $op = +,-,*,/$.

Esta forma é válida para a maioria dos computadores digitais, particularmente os que utilizam o padrão IEEE. Entretanto, existem computadores que não seguem este padrão, alguns deles por não utilizarem um dígito de guarda em operações aritméticas. Tome-se, por exemplo, um sistema de ponto flutuante com $\beta = 2$ e $n = 3$. Ao realizar, neste sistema, a operação $(0{,}100\times 2^1 - 0{,}111\times 2^0)$ sem e com dígito de guarda, obtém-se

$$u = \frac{1}{2}\beta^{-n+1} = \frac{1}{2} * 2^{-2} = \frac{1}{8}.$$

Ao realizar esta subtração sem dígito de guarda obtém-se: $fl(0{,}100\times 2^1 - 0{,}111\times 2^0) \Rightarrow fl(0{,}100\times 2^1 - 0{,}011\times 2^1) = 0{,}001\times 2^1 \Rightarrow 0{,}100\times 2^{-1}$.

Considerando $(x-y) = 0{,}100\times 2^{-2}$ e $fl(x-y) = 0{,}100\times 2^{-1}$, tem-se que:

$$0{,}100\times 2^{-1} = 0{,}100\times 2^{-2}(1+\delta)$$
$$\delta = \frac{0{,}100\times 2^{-1} - 0{,}100\times 2^{-2}}{0{,}100\times 2^{-2}} = 1 \Rightarrow |\delta| > u$$

Por sua vez, com o dígito de guarda obtém-se o resultado:

$fl(0{,}100\times 2^1 - 0{,}111\times 2^0) \Rightarrow fl(0{,}100\times 2^1 - 0{,}0111\times 2^1) = 0{,}0001\times 2^1 \Rightarrow 0{,}100\times 2^{-2}$ Co

nsiderando que $(x-y) = 0{,}100\times 2^{-2}$ e $fl(x-y) = 0{,}100\times 2^{-2}$, tem-se:

$$0{,}100\times 2^{-2} = 0{,}100\times 2^{-2}(1+\delta)$$
$$\delta = \frac{0{,}100\times 2^{-2} - 0{,}100\times 2^{-2}}{0{,}100\times 2^{-2}} = 0 \Rightarrow |\delta| < u$$

A forma definida em (1.3) não é válida para computadores que não utilizam o dígito de guarda. Neste caso, as operações aritméticas são definidas por:

$$fl(x\ op\ y) = (x\ op\ y)(1+\delta) \quad |\delta| \le u \tag{1.4}$$

onde: $op = *, /$ e

$$fl(x\ op\ y) = x(1+\alpha) \pm y(1+\gamma) \quad |\alpha|, |\gamma| \leq u \tag{1.5}$$

onde: $op = +, -$.

2.2.5. ERRO DE CANCELAMENTO SUBTRATIVO

Quando números muito próximos são subtraídos, pode acontecer um erro chamado *erro de cancelamento subtrativo*.

Exemplo 1 (Rump, 1983): Se *a* = 10^{+34} e *b* = -2, então o resultado de $(a-a)+b$ é -2 mas, se esta expressão for calculada por $a+(-a+b)$, o resultado será 0. O porquê, nesta última expressão, ocorreu *erro de cancelamento subtrativo*. Este erro ocorre porque a diferença entre *a* e *b* é muito grande e, portanto, quando estes valores são operados, a tendência é que o menor valor desapareça já que a soma de *a* e *b* resulta em 9,999...8 e este valor é arredondado para 10^{+34}.

Goldberg (Goldberg, 1991) apresenta algumas formas de minimizar e/ou evitar o *erro de cancelamento subtrativo.*

2.2.6. TIPOS DE ARREDONDAMENTO

Pelo fato de haver um número máximo de dígitos na representação de números em computadores, deve-se definir uma abordagem para armazenar um número cuja precisão requerida está além da precisão de máquina. Assim, o valor de entrada é arredondado sempre que sua precisão exceder a precisão de máquina. Segundo o padrão IEEE-754, existem quatro tipos de arredondamento (Dalcidio):

- **Arredondamento para o número de máquina mais próximo:** é o arredondamento mais recomendado pelo padrão e consiste em verificar se o primeiro dígito excedente da precisão é maior ou igual a cinco. Se for, soma-se um ao valor na posição do dígito *n*. Caso contrário, os dígitos excedentes são desconsiderados. Caso o primeiro dígito excedente seja 5, arredonda-se para o número par. Exemplo:

$$X = +0{,}6582 \Rightarrow x' = \pm 0{,}658$$

$$Y = \pm 0{,}6588 \Rightarrow Y' = \pm 0{,}659$$

$$Z = \pm 0{,}6575 \Rightarrow Z' = \pm 0{,}658$$

$$T = \pm 0{,}6585 \Rightarrow T' = \pm 0{,}658$$

- **Arredondamento para zero:** sempre que o valor a ser representado exceder a precisão de máquina, escolhe-se a representação mais próxima do valor em direção a zero:

$$X = + 0{,}65821 \Rightarrow X' = + 0{,}658$$

$$Y = - 0{,}65821 \Rightarrow Y' = - 0{,}658$$

- **Arredondamento por excesso (+∞):** sempre que o valor a ser representado exceder a precisão de máquina, soma-se um ao valor contido na posição do dígito *n*. Exemplo:

$$X = + 0{,}65821 \Rightarrow X' = + 0{,}659$$

$$Y = - 0{,}65821 \Rightarrow Y' = - 0{,}659$$

- **Arredondamento por truncamento (-∞):**Sempre que o valor a ser representado exceder a precisão de máquina, desconsidera-se todos os valores que excederam a precisão. Exemplo:

$$X = + 0{,}65821 \Rightarrow X' = + 0{,}658$$

$$Y = - 0{,}65821 \Rightarrow Y' = - 0{,}658$$

2.3. PADRAO IEEE

O padrão IEEE é recomendado pelos institutos ANSI (*American National Standard Institute*) e IEEE (*Institute of Electrical and Electronic Engineers*). Refere-

se às normas a serem seguidas pelos fabricantes de computadores e construtores de compiladores de linguagem para computação científica ou de bibliotecas de funções matemáticas, na utilização da aritmética binária para números de ponto flutuante. As recomendações são relativas ao armazenamento de dados numéricos, métodos de arredondamento, tratamento de casos de *underflow* e *overflow*, formas de realizar as quatro operações aritméticas básicas e implementação de funções nas linguagens de programação.

Existem dois padrões IEEE para aritmética binária de ponto flutuante: IEEE-754 e o IEEE-854. O primeiro é um padrão binário em que a precisão é de 24 bits, para precisão simples, e 53 bits para precisão dupla, sendo que o *layout* dos bits é definido para ambas as precisões. O padrão IEEE-854 é um padrão tanto para binário quanto para decimal que, ao contrário do IEEE 754, não especifica como os números de ponto flutuantes são codificados em bits e também não define um valor fixo para a precisão. O padrão especifica apenas as restrições para valores em precisão simples ou dupla.

2.3.1. FORMATOS E OPERAÇÕES

Assim como apresentado no *item 2.2.2* – Formato de Ponto Flutuante – o sistema de ponto flutuante é definido a partir da configuração da base, precisão, expoentes mínimo e máximo. A seguir estes parâmetros são discutidos à luz do padrão IEEE.

- **Base**: Faz muito sentido a norma 854 da IEEE permitir base 10, pois esta base é naturalmente utilizada pelas pessoas para fazer cálculos do dia a dia. Utilizar base 10 é especialmente adequado em calculadoras, onde o resultado de cada operação é apresentado como um número decimal (Goldberg, 1991). Mas utilizar base 2 é importante em algumas situações. Uma delas é quando o erro de arredondamento excede um valor previamente fixado. Outra é porque bases maiores podem perder exatidão. Geralmente, a base 16 pode perder até 3 bits de precisão e, por isto, a precisão de *n* dígitos hexadecimais pode resultar numa precisão efetiva de (*4n – 3*), contra *4n* em base dois (Goldberg, 1991).

- **Precisão**: O padrão IEEE define quatro possibilidades de precisão: simples, dupla, simples estendida e dupla estendida. Na norma 754, a precisão simples é definida a partir de uma única palavra de 32 bits e a dupla utiliza duas palavras de 32 bits. A precisão estendida – simples e dupla – é um formato que oferece mais precisão e um intervalo maior de números de máquina (Goldberg, 1991).

Parameter	Format			
	Single	Single Extended	Double	Double Extended
p	24	≥ 32	53	≥ 64
E_{max}	+127	≥ +1023	+1023	≥ +16383
E_{min}	–126	≤ –1022	–1022	≤ –16382
Exponent *bias*	+127	unspecified	+1023	unspecified
Exponent width in bits	8	≥ 11	11	≥ 15
Format width in bits	32	≥ 43	64	≥ 79

Tabela 2.3.1: Padrão IEEE 754 para números de ponto flutuante (IEEE).

O padrão IEEE especifica apenas um limite inferior para o número de bits extras que a precisão estendida disponibiliza. O menor formato possível para a precisão dupla estendida é também conhecido como *80-bit format*, sendo que na *tabela 2.3.1* este formato é apresentado utilizando 79 bits. O motivo para isto é que normalmente, as implementações de hardware para precisão estendida utilizam um bit escondido e, então, o tamanho da palavra de armazenamento seria 80 bits ao invés de 79 (Goldberg, 1991).

Uma das situações onde se justifica o uso de precisão estendida são as calculadoras que normalmente apresentam 8 ou 10 dígitos, mas utilizam mais do que isto para os cálculos (Goldberg, 1991).

- **Expoente**: Como o expoente pode ser positivo ou negativo, alguma outra forma de representação do valor pode ser utilizada em lugar da representação sinal-magnitude – onde um bit é utilizado para conter o sinal e o resto para representar a magnitude do número – visando incluir o sinal na representação do número. A forma de representação mais utilizada é o complemento de dois. Neste caso, um número é representado pelo menor número não negativo que seja congruente à sua representação sem sinal. O

padrão IEEE para aritmética binária não usa nenhuma destas formas de representação para representar o expoente, mas sim a representação polarizada onde, se em precisão simples o expoente é armazenado em 8 bits, o *bias* é 127 (na precisão dupla é 1023). Isto quer dizer que se *k* é o valor dos bits do expoente como um *unsigned integer*, então o expoente do número de ponto flutuante é *k-127*. A vantagem de uma representação polarizada é que números de ponto flutuante não negativos podem ser tratados como inteiros em uma comparação (Goldberg, 1991).

- **Operações**: O padrão IEEE requer que o resultado de operações aritméticas básicas seja *exatamente arredondado*, ou seja, deve ser computado exatamente e então arredondado para o número de máquina mais próximo. Realizar este tipo de computação para obter a soma/diferença exata entre dois valores pode ser muito caro quando os valores envolvidos nesta operação possuem magnitudes muito diferentes (um é muito grande e o outro é muito pequeno). Por sua vez, utilizar apenas um dígito de guarda pode não ser o suficiente, mas o problema do custo da operação pode ser diminuído, sem perda de exatidão, ao utilizar dois dígitos de guarda e um sticky bit (Goldberg, 1991).

Ainda não existe consenso sobre quais operações o padrão de ponto flutuante deve cobrir, além das 4 operações básicas. O padrão IEEE também especifica a raiz quadrada, a função “mod” (resto da divisão) e a conversão entre número inteiro e de ponto flutuante. O padrão também define que a conversão entre formatos – interno e decimal – seja corretamente arredondada (com exceção de números muito grandes) (Goldberg, 1991).

A principal vantagem da utilização do padrão IEEE é de possibilitar a portabilidade de softwares numéricos. Se duas máquinas realizam a computação utilizando aritmética binária de ponto flutuante segundo o padrão IEEE, e produzem um resultado diferente para uma mesma implementação, só pode ser um problema do programa. Outro motivo muito importante para utilização do padrão IEEE consiste no auxílio para a realização de provas matemáticas sobre números de ponto flutuante. Ou seja, provas com uma única aritmética já são difíceis o bastante, imaginem como esta dificuldade aumenta quando existem diversos casos provenientes de diversas aritméticas (Goldberg, 1991).

Ao mesmo tempo em que a regularidade de comportamento entre diferentes arquiteturas ajuda, ela também atrapalha: os usuários obterão os mesmos resultados em diversas máquinas diferentes. Aqueles que tiverem menos experiência podem vir a crer que a resposta está correta, o que nem sempre é verdade. Os usuários devem estar informados de que erros podem acontecer e, devido à padronização, os resultados serão os mesmos, mas isto não garante que estão corretos.

Mesmo o padrão IEEE apresenta alguns problemas associados à forma de especificação das operações aritméticas básicas. Kulisch e Miranker (Kulish, et al., 1986) notaram que, quando produtos internos são computados segundo a aritmética do padrão IEEE, podem produzir um resultado errado. Por exemplo, ao avaliar a expressão $((2*10^{-30} + 10^{30}) - 10^{30}) -10^{-30}$, cuja resposta exata é 10^{-30}, obteve-se o resultado -10^{-30} em uma máquina com aritmética IEEE (Goldberg, 1991).

2.3.2. FORMATOS ESPECIAIS

Em algumas implementações de hardware, todas as combinações de bits representam um número de ponto flutuante válido. Todavia, isto representa um problema pois, em algumas situações, como no caso da raiz quadrada de um número negativo, não haveria outra opção a não ser abortar a computação. Para evitar este problema, a aritmética binária, segundo o padrão IEEE, utiliza valores especiais que permitem o tratamento destes acontecimentos sem que a computação seja abortada. São eles: ± 0, números desnormalizados, $\pm\infty$ e NaNs (Not a Number). No caso de uma raiz quadrada de um número negativo, a aritmética IEEE retorna um NaN, possibilitando que a computação continue. Entretanto, é importante destacar que existe mais de um tipo de NaN, mas que não será tratado neste trabalho. Maiores informações podem ser obtidas em (Goldberg, 1991).

A *tabela 2.3.2* apresenta os valores especiais do padrão IEEE.

Exponent	Fraction	Represents
$e = e_{min} - 1$	$f = 0$	± 0
$e = e_{min} - 1$	$f \neq 0$	$0.f \times 2^{e_{min}}$
$e_{min} \leq e \leq e_{max}$	—	$1.f \times 2^{e}$
$e = e_{max} + 1$	$f = 0$	∞
$e = e_{max} + 1$	$f \neq 0$	NaN

Tabela 2.3.2: Valores especiais do padrão IEEE (Goldberg, 1991)

2.4. EFICIÊNCIA E COMPLEXIDADE

Escrever algoritmos é, na maioria das vezes, simples. Contudo, desenvolver bons algoritmos pode não ser tão fácil assim. Um bom algoritmo é um procedimento efetivo[1] que sempre termina, qualquer que seja a entrada, consome poucos recursos computacionais e fornece um resultado de qualidade. E é isto que este trabalho se propõe a buscar, ou seja, bons algoritmos para realizar a avaliação polinomial.

Muitas vezes, para alterar o tempo de execução de um algoritmo, basta alterar o tamanho da entrada de dados, uma vez que a eficiência de algoritmos é dependente do tamanho da entrada.

Certamente, não é correto medir a eficiência de um algoritmo com apenas uma execução. O que se deve fazer é testar uma quantidade representativa de entradas (James Madison University).

O estudo da complexidade de algoritmos se baseia nesta ideia: em lugar de determinar o desempenho "real" do algoritmo, ele estabelece limites de desempenho. Por exemplo, a complexidade do pior caso determina um limite teórico para o desempenho do algoritmo. Mesmo que isto não diga exatamente como o algoritmo se comporta, temos uma estimativa pessimista do seu comportamento (James Madison University).

Geralmente, o custo para obtenção de uma resposta cresce proporcionalmente com o aumento do tamanho da entrada *(n)*. Se *n* for suficientemente pequeno, até um algoritmo ineficiente pode não ter um custo muito alto. Consequentemente, a escolha do algoritmo para resolução de problemas

[1] Descrição finita não-ambígua de um conjunto de operações efetivas, estritamente mecânicas, e realizadas em uma certa ordem de execução.

pequenos não é vital (a menos que este seja resolvido diversas vezes). Mas existe a situação onde *n* cresce tanto que o algoritmo se torna incapaz de resolvê-lo em tempo aceitável. Este crescimento é demonstrado pelas *tabelas 2.4.1 e 2.4.2*(James Madison University).

Complexidade de tempo	**Tamanho do problema *n***			
	10	10^2	10^3	10^4
$\log_2(n)$	3,3	6,6	10	13,3
n	10	10^2	10^3	10^4
$n\log_2(n)$	$0{,}33 * 10^2$	$0{,}7 * 10^3$	10^4	$1{,}3 * 10^5$
n^2	10^2	10^4	10^6	10^8
n^3	10^3	10^6	10^9	10^{12}
2^n	1024	$1{,}3 * 10^{30}$	$>10^{100}$	$>10^{100}$
n!	$3 * 10^6$	$>10^{100}$	$>10^{100}$	$>10^{100}$

Tabela 2.4.1: Comparação do crescimento de algumas funções de complexidade comuns (James Madison University).

Função da complexidade de tempo	**Tamanho máximo do problema executado em**		
	1s	1 min	1 hora
$\log_2(n)$	2^{10^6}	2^{6*10^7}	$2^{3{,}6*10^9}$
n	10^6	$6 * 10^7$	$3{,}6 * 10^9$
$n\log_2(n)$	62746	$2{,}8 * 10^6$	$1{,}3 * 10^8$
n^2	10^3	7746	60000
n^3	10^2	$3{,}9 * 10^2$	$1{,}5 * 10^3$
2^n	20	25	32
n!	9	11	12

Tabela 2.4.2: Limite do tamanho dos problemas que podem ser resolvidos dentro de um dado intervalo de tempo (assume-se uma operação por microssegundo) (James Madison University).

2.4.1. TIPOS DE ALGORITMOS

Os algoritmos podem ter diversas classificações, porém julgou-se apropriado utilizar a classificação apresentada por Lydia em (Kronsjö, 1979). Ela separa os algoritmos entre numéricos e não-numéricos, onde os numéricos são aqueles que realizam operações aritméticas, produzindo um valor numérico que representa a solução computada do problema numérico correspondente. Por sua vez, os algoritmos não-numéricos manipulam valores e não-numéricos não realizam operações aritméticas. Cita-se, como exemplo, o algoritmo para ordenação de valores em ordem crescente ou decrescente. Embora manipule números, este

algoritmo realiza apenas operações de comparação, portanto não se inclui na categoria de algoritmo numérico.

Segundo essa mesma autora, a análise de algoritmos deve ser realizada através da análise da *complexidade algébrica* e da análise de *complexidade analítica*. A *complexidade analítica* busca descobrir a quantidade de computação necessária para obter um resultado com um certo grau de exatidão e tem como foco aqueles processos que só terminam depois de um certo número de execuções repetidas. Um bom exemplo disto são os processos iterativos. Já a análise da *complexidade algébrica* tem por objetivo identificar o número de operações aritméticas realizadas pelo algoritmo.

2.5. BENCHMARKS

Benchmarks são testes ou conjuntos de testes para comparar dois ou mais objetos (Standard Performance Evaluation Corporation , 2007).

Na computação, benchmark é o ato de avaliar o desempenho de computadores visando conhecer a relação custo-benefício entre hardware e compiladores e ferramentas para detectar gargalos de sistemas. Mas nem sempre somente a avaliação de desempenho é suficiente para estabelecer esta relação custo-benefício. No processamento numérico para computação científica, a exatidão do resultado é tão importante quanto o tempo necessário para obter este resultado. Neste caso, as métricas que geralmente são utilizadas não suprem as necessidades (Ossama, 2000). A seguir esta questão será discutida mais detalhadamente. Neste momento, apresentar-se-á uma forma como os benchmarks podem ser classificados.

2.5.1. TIPOS DE BENCHMARKS

Segundo Ossama (2000), os benchmarks mais importantes da computação estão classificados em:

- **Sintético**: O código não executa nada de útil computacionalmente, não representa nenhuma computação real. O que o benchmark faz é apenas

testar os componentes do computador. Normalmente é determinada a freqüência média de instruções que é recriada em um programa. Os benchmarks deste tipo mais conhecidos são Whetstone e Dhrystone.

- **Kernel**: Baseia-se no fato de que a maior parte da computação de um programa está centrada em uma pequena parte do código. Esta parte, chamada de *core* (*kernel*), é extraída do programa e usada como um benchmark. Este tipo de benchmark é muito interessante por ser pequeno e simples, mas ele não avalia completamente o desempenho do computador. Um exemplo deste tipo de benchmark é o Livermore Loops.
- **Algoritmo**: Normalmente são implementações de métodos numéricos bem conhecidos através de algoritmos numéricos bem definidos. Um exemplo deste tipo de benchmark é dado pelo método numérico para resolução de sistemas lineares que faz parte da biblioteca numérica Linpack.
- **Aplicação** - Programas completos que procuram soluções para problemas científicos bem definidos. Um exemplo são alguns dos benchmarks desenvolvidos pela organização SPEC.

2.5.2. PADRÕES DE BENCHMARKS

A padronização de benchmarks da computação é um assunto bastante discutido atualmente, mas é muito difícil que esta discussão resulte na definição de um padrão devido a grande variedade de arquiteturas e de velocidade dos processadores.

As duas organizações mais conhecidas por seu trabalho com benchmarks para a computação são o Comitê PARKBENCH[2] (*Parallel Kernels and Benchmarks*) e a SPEC[3] (*Standard Performance Evaluation Corporation*) [8].

Os benchmarks do comitê PARKBENCH incluem benchmarks de aplicação (ou benchmarks de baixo nível), benchmarks de kernel e um tipo criado por este comitê, chamado de “aplicações compactas”. Os benchmarks de baixo nível visam

[2] Esta organização tem contribuições de importantes universidades, laboratórios e indústrias.

[3] Esta é uma organização – sem fins lucrativos – que busca produzir “benchmarks para computadores que sejam justos, imparciais e significativos”. Seus benchmarks se caracterizam pela permanente atualização da documentação.

medir a arquitetura básica (hardware) e os compiladores do computador. Eles são complexos e normalmente não têm versões para máquinas paralelas. Benchmarks do tipo *kernel* são simples, porém são muito específicos em um trecho de código, sem testar o problema como um todo. Os benchmarks do tipo “aplicação compacta” solucionam este problema: são programas completos, porém mais simples do que os benchmarks de baixo nível.

2.5.3. UNIDADES DE MEDIDA

Usualmente, para comparar objetos, é necessário dispor de uma unidade de medida e um valor numérico. Mas isto pode ser contraditório uma vez que, em muitas situações, um único número não é suficiente para avaliar o desempenho de um computador. Entretanto, reunir a informação transportada por diversas variáveis em um único, valor possibilita uma forma de análise mais simples e imediata e por isto esta é a forma mais utilizada nos benchmarks (Ossama, 2000).

Segundo Ossama (2000), as unidades de medida mais utilizadas nos benchmarks são:

- MIPS (*Millions of Instructions per Second)*: Medida muito utilizada no passado, mas que perdeu seu significado quando surgiram as máquinas RISC (*Reduced Instruction Set Computer*). O problema é que, com esta unidade, quando se compara uma máquina CISC (*Complex Instruction Set Computer*) com uma RISC, mesmo que o tempo de execução seja igual, o valor MIPS é mais alto na máquina RISC. Isto acontece porque a arquitetura RISC precisa de mais instruções que a CISC para completar a mesma tarefa, mas não necessariamente significa que demore mais para realizá-la.
- MFLOPS (*Millions of Floating Point Operations per Second*): Esta unidade é principalmente utilizada na computação científica onde são realizados muitos cálculos em ponto flutuante. O problema desta unidade é similar ao do MIPS, ou seja, uma arquitetura pode usar mais operações de ponto flutuante para realizar uma tarefa, mas o número de operações não significa necessariamente que a tarefa demorou mais para ser executada. Outro ponto é que esta unidade considera apenas operações de ponto flutuante, e, para algumas aplicações, isto não é tão relevante.

2.5.4. BENCHMARKS MAIS CONHECIDOS

O Linpack e o SPEC são dois dos benchmarks mais conhecidos. O primeiro é usado na lista dos quinhentos computadores mais poderosos do mundo, e o segundo é mais uma suíte de benchmarks do que um benchmark em si:

- **SPEC**: Os benchmarks desenvolvidos por esta organização são benchmarks de aplicação que visam medir o desempenho da maior parte possível do sistema com foco na arquitetura, processador, memória e compilador.
- **Benchmarks do PARKBENCH** – o comitê PARKBENCH tem benchmarks do tipo aplicação (chamada por eles de “baixo-nível”), kernel e uma classificação criada por eles, “aplicações compactas” (Ossama, 2000).

Depois destes dois, os mais conhecidos são (Ossama, 2000):

- **Whetstone**: Primeiro benchmark sintético da literatura. Foi publicado em 1976 por H. J. Curnow e B. A. Wichmann do Laboratório Nacional de Física do Reino Unido. Ele visa medir o desempenho e sua unidade é a MWIPS (*Mega Whetstone Instructions per second*).
- **Dhrystone**: Este benchmark foi publicado em 1984 por Reinhol P. Weicker e dá ênfase a funções com strings. Sua principal aplicação é na análise da eficiência da combinação entre hardware e compilador em computadores de pequeno e médio porte. Sua unidade de medida é Dhrystone por segundo.
- **Livermore Loops** (ou Livermore Fortran Kernels): Benchmark do tipo kernel que é a síntese de vários programas feitos por Poe Frank H. McMahon, do *Lawrence Livermore National Laboratory*, em 1970.
- **NAS[4] Parallel Benchmarks**: Conjunto de programas desenvolvido pelo NASA Ames Research Center e publicado em 1995. Ele explora a capacidade de comunicação e memória do computador. Sua unidade de medida é o MFLOP.
- **Linpack**: Este é um dos benchmarks mais famosos e é usado no teste dos 500 computadores mais rápidos do mundo (Top500) (Dongarra). Foi

[4] *Numerical Aerodynamics Simulation*

publicado em 1976 por Jack Dongarra, da Univerisadade do Tennessee, sem a intenção de fazer dele um benchmark. É um benchmark do tipo algoritmo que dá os resultandos usando MFLOPS, GFLOPS (Giga FLOPS) ou até TFLOPS (Tera FLOPS). Sua aplicação é em máquinas que usam softwares para cálculos científicos ou de engenharia.

2.5.5. QUALIDADE DO RESULTADO NOS BENCHMARKS

Ambos os benchmarks, Linpack e SPEC, têm testes sobre cálculos com números de ponto flutuante, mas sua ênfase está no desempenho. Existe, porém, uma qualidade mínima necessária a ser atingida, a qual não influi no resultado do benchmark. No caso do Linpack, o resultado deve satisfazer a fórmula que pode ser vista em (Dongarra). Para o SPEC, esta tolerância varia de benchmark para benchmark.

2.6. O PRINCÍPIO DO OBSERVADOR

O Princípio do Observador (*observer principle*) diz que se o processo de observação acrescenta ou retira energia de um sistema, esta observação pode influenciar o estado do sistema. Por conseguinte, não se tem certeza de que o que é observado é a mesma coisa que existe no sistema quando não há observação. Por exemplo, na física, para realizar a detecção de um elétron, é necessário que um fóton interaja com ele, entretanto esta interação altera sua trajetória. Este princípio pode ser aplicado intencionalmente e tem grande importância, por exemplo, na criptografia quântica, onde o ato de ler o estado quântico de um fóton destrói o estado do mesmo (Denning, 2007).

De acordo com a especificação do padrão IEEE-754 para números de ponto flutuante e a configuração do sistema de ponto flutuante remetida pelo Matlab, o menor número de ponto flutuante normalizado é dado por $2.2251e^{-308}$, considerando uma representação em precisão dupla. Entretanto, em sua primeira versão, o algoritmo do Matlab retornava o valor zero como resultado. Na segunda versão deste mesmo algoritmo, a qual imprimia os resultados intermediários, o resultado obtido era $8.0948e^{-320}$. Esta diferença tem origem no fato dos processadores

modernos utilizarem registradores de 80 bits para o processamento e de 64 bits para o armazenando dos resultados. Quando não era realizada a exibição de resultados, o cálculo era totalmente executado em registradores de 80 bits, resultando em *zero* no momento da conversão para 64 bits, decorrente de underflow. Ao imprimir os resultados, os parciais eram convertidos para 64 bits, resolvendo o problema de underflow, mas produzindo um resultado incorreto. Então, foi desenvolvida uma terceira versão do algoritmo que apresentava a resposta correta quando executado diretamente porém, quando executado a partir de um arquivo, a resposta obtida era $4.9407e^{-324}$. O erro que ocorreu nesta versão do algoritmo também foi decorrente da diferença de tamanho dos registradores, ou seja, o arquivo compilado gerava código otimizado para o processador, armazenando variáveis em registradores de 80 bits (Gander, 2006).

Este efeito pode acontecer em situações onde os cálculos estão sendo monitorados e é extremamente prejudicial tanto para o cálculo em si quanto para os resultados da monitoração.

2.7. VISUALIZAÇÃO GRÁFICA DE INFORMAÇÕES

Para que o usuário possa fazer uma análise dos dados, é necessário que ele os entenda muito bem. A partir disto, ele pode inferir relações e realizar comparações. Para isto, utiliza-se a capacidade do ser humano de perceber padrões, exceções, tendências e relações (Wong, 1999). Usualmente, a percepção visual a partir de figuras é mais desenvolvida nas pessoas do que a capacidade de compreender números ou tabelas e é aqui que entra a mineração visual de dados.

A visualização gráfica de dados consiste em transformar informações em gráficos, imagens e figuras para que o entendimento das mesmas seja facilitado. Ela se baseia em uma representação visual e em mecanismos de interação que permitem ao usuário manipular esta representação. O nível da representação é mais alto porque frequentemente não há relação direta entre os dados e uma entidade física ou geométrica, ou o usuário não está interessado em dados brutos, mas sim em observar características ou padrões no conjunto de dados (Freitas, 2001). Se comparada com a apresentação textual, a representação gráfica de informação possibilita a compreensão de grandes volumes de dados (Keim, 2002).

Existem diversas técnicas de visualização de dados. Os dados representados descrevem fenômenos (também chamados de processos) ou entidades que são objeto de estudo ou análise (Freitas, 2001). Eles podem ser classificados por diferentes critérios e esta classificação é a primeira coisa que deve ser analisada no momento da escolha de uma técnica de visualização. Infelizmente, devido ao fato de que sua classificação está estritamente ligada à classificação do conhecimento, ainda não existe consenso quanto a melhor forma de fazer esta organização (Ware, 2000). Os seguintes critérios são geralmente utilizados para a classificação de dados:

- Classe (ou tipo) representada pelos dados: este deve ser um dos primeiros critérios analisados. Os atributos podem enquadrar as entidades descritas dentro de uma classe. Neste caso são chamados de característica, categoria, atributo nominal ou ordinal. Também podem representar uma propriedade: atributos com valores escalares, vetoriais ou tensoriais, que assumem valores inteiros ou reais dentro de um intervalo. Atributos podem indicar a existência de um relacionamento (hierarquia ou ligação) (Freitas, 2001). É importante lembrar que existem especialistas que dizem que a relação não é um tipo de dado, como Bertin (Ware, 2000). Contudo, a fim de simplificação, este trabalho segue a corrente que diz que relacionamento é um atributo.
- Tipo de dado: Este critério é utilizado no sentido de tipo primitivo, ou seja, se o dado pode assumir valores alfanuméricos, inteiros, reais ou simbólicos (Freitas, 2001).
- Dimensão e natureza do domínio: os dados podem estar definidos em um domínio unidimensional, bidimensional, tridimensional ou n-dimensional, e este domínio pode ser contínuo ou discreto (Freitas, 2001).

As representações visuais correspondem às figuras ou imagens empregadas para representar o conjunto sob análise. Existem diversos modos para apresentar dados. Os mais utilizados são gráficos de barras, linhas, circulares e algumas outras representações mais ou menos complexas (Freitas, 2001). Na *Tabela 2.7.1* apresenta-se um resumo dos critérios utilizados para classificar a informação. Já na *Tabela 2.7.2* há uma proposta de classificação para os tipos das representações gráficas.

Critério	Classe	Exemplo
Classe de informação	Categoria	Gênero
	Escalar	Temperatura
	Vetorial	Grandezas físicas associadas
	Tensorial	a dinâmica de fluidos
	Relacionamento	Link num hiperdocumento
Tipo dos valores	Alfanumérico	Gênero
	Numérico (inteiro, real)	Temperatura
	Simbólico	Link num hiperdocumento
Natureza do domínio	Discreto	Marcas de automóveis
	Contínuo	Superfície de um terreno
	Contínuo-discretizado	Anos (tempo discretizado)
Dimensão do domínio	1D	Fenômeno ocorrendo no tempo
	2D	Superfície de um terreno
	3D	Volume de dados médicos
	n-D	Dados de uma população

Tabela 2.7.1: Resumo dos critérios utilizados (Freitas, 2001)

Classe	Tipo	Utilização
Gráficos 2D, 3D	Pontos Circulares Linhas Barras Superfícies (para 3D)	Representação da distribuição dos elementos no espaço domínio, representação da dependência/correlação entre atributos.
Ícones Glifos Objetos geométricos	Elementos geométricos 2D ou 3D diversos	Representação de entidades num contexto, representação de grupos de atributos de diversos tipos.
Mapas	de pseudo-cores	Representação de campos escalares ou de categorias.
	de linhas	Representação de linhas de contorno de regiões, isovalores.
	de superfícies	Idem, no espaço 3D.
	de ícones, símbolos diversos	Representação de grupos de atributos (categoriais, escalares, vetoriais, tensoriais).
Diagramas	Nodos e arestas	Representação de relacionamentos diversos: É-um, É-parte-de, Comunicação, Seqüência, Referência, etc.

Tabela 2.7.2: Classes de representações visuais (Freitas, 2001)

Uma vez que uma representação estática nem sempre é suficiente para que o usuário compreenda a informação transportada por grandes conjuntos de dados, é interessante dispor de um sistema de visualização que possibilite ao usuário explorar e observar os dados segundo diferentes aspectos. Um nível mais básico de exploração visual são as funções de navegação e reposicionamento do observador, seja pelo deslocamento de uma *scrollbar* em uma representação plana, como no deslocamento de uma câmera virtual (ou rotação do conjunto de dados) no espaço

em uma representação 3D (Freitas, 2001). Já em um segundo nível, estão as funções de seleção de elementos de dados de interesse. Estas podem provocar tanto um reposicionamento do conjunto de dados para uma melhor visualização (zooming semântico) de uma parte desse conjunto e, eventualmente, a supressão de outra parte. Operações de poda (prunning), agrupamento (clustering) e expansão são muito importantes no apoio ao processo de navegação e exploração em diagramas que representam grandes hierarquias e grafos, e são encontradas na maioria das técnicas (Freitas, 2001).

Em um nível superior, estão funções que permitem representar visualmente apenas parte do conjunto de dados, dependendo da satisfação de certos critérios. Tais funções podem apenas estabelecer filtros com base nos valores de atributos ou corresponder a consultas dinâmicas apoiadas em processos de mineração de dados (Freitas, 2001). A visualização também tem alguns problemas, pois existem alguns aspectos, como a oclusão de objetos, desordem e desorientação visual, que dificultam a interpretação das informações. A oclusão de objetos ocorre quando existe muita informação e alguns elementos se sobrepõem a outros. A desordem visual é decorrente da dificuldade de reconhecimento e interpretação de muitos elementos, estando eles sobrepostos ou não, o que força o seu sistema cognitivo. Para resolver estes dois problemas, podem ser utilizadas sombras e transparências, o que permite a localização do objeto. Também podem ser fornecidos mecanismos básicos de manipulação geométrica como rotação, mudança de escala e translação dos objetos para minimizar estes problemas. A desorientação visual ocorre quando o usuário tem dificuldades na manutenção da atenção em uma troca de ponto de vista ou retorno a situações anteriores. Algumas técnicas tentam minimizar este problema, evitando que elementos desapareçam (eles diminuem de tamanho) e fazendo uma transição animada e gradual entre dois momentos no processo interativo (Freitas, 2001).

2.8. AMBIENTES DE PROCESSAMENTO MATEMÁTICO

Para a realização deste trabalho, avaliamos quatro ambientes de processamento matemático visando a utilização de um destes ambientes para geração dos dados utilizados na representação gráfica.

Os ambientes analisados foram o Maple e Matlab – ambientes proprietários - e o Maxima e Octave – ambientes de domínio público. A primeira vista, os ambientes são muito parecidos, pois todos utilizam programação funcional via linha de comando. Porém, uma análise um pouco mais detalhada permite perceber diferenças que serão essenciais para a tomada de decisão sobre qual dos ambientes utilizar.

O Maple é a ferramenta matemática líder no mercado (Maplesoft, a division of Waterloo Maple Inc. 2009). Em termos de visualização, oferece a possibilidade de gráficos 2D e 3D, e permite a interação do usuário com estes gráficos. Além disto, é possível construir gráficos apenas "arrastando equações" para uma área gráfica gerando, portanto, um gráfico "sob demanda". Outra possibilidade interessante do uso do Maple consiste na criação de *scripts* que podem ser utilizados dentro do próprio ambiente. O Maple também converte o código criado no seu ambiente para C, Java, VisualBasic, Fortran e para a linguagem do Matlab (também chamada de M-code). Também é possível incluir as bibliotecas do Maple em programas destas linguagens. Como já foi dito, o Maple é proprietário, então testamos um software livre com o mesmo intuito, o Maxima.

O Maxima teve origem no projeto Macsyma, desenvolvido no período 1968-1982 pelo grupo de computação algébrica do Massachusetts Institute of Technology (MIT) (Maxima). Em 1982, este projeto passou a ser comercializado e então um grupo dissidente criou o Maxima como uma alternativa pública. O Maxima permite a visualização de gráficos 2D e 3D, e a interação do usuário com estes gráficos. Contudo, depois que o gráfico é apresentado na tela, não é mais possível alterá-lo. Ele permite salvar o estado da sessão, porém não permite a criação de *scripts*. O Maxima foi implementado em Lisp e por isto não é possível o reaproveitamento de código por outros programas.

Outro ambiente analisado foi o Matlab. Este ambiente permite o uso de diversos *add-ons* que incluem funcionalidades no programa, como processamento de imagens e computação distribuída. Aqui analisaremos apenas o ambiente, sem estes *add-ons*. O Matlab fornece gráficos necessários para análise de dados científicos e de engenharia (The MathWorks, Inc.) como gráficos 2D, 3D e de volume. O usuário tem muitos meios de interagir com o gráfico, principalmente os gráficos 3D. É possível definir o ângulo no qual se vê o gráfico, a perspectiva, os efeitos de luz, os locais da fonte de luz e a transparência (The MathWorks, Inc.).

Assim como o Maple, ele também permite que se altere o gráfico após este estar pronto, arrastando conjuntos de dados sobre a figura do gráfico, e que o usuário crie scripts para utilizar dentro do próprio ambiente. O ambiente também permite que suas funções sejam utilizadas em programas desenvolvidos pelo usuário, dando suporte para diversas linguagens. O Matlab fornece um editor para que estes *scripts* sejam feitos com mais facilidade.

Por último, testou-se o Octave, que é o clone público do Matlab. O Octave permite a visualização de gráficos em 2D e 3D e que o usuário interaja com o gráfico 3D através da rotação. Uma vez desenhado, o gráfico não pode mais ser alterado, diferente do que acontece no Maple e Matlab. O Octave permite que o usuário defina seus *scripts* e os utilize dentro do ambiente. Também que o carregue módulos escritos em C, C++, Fortran entre outras linguagens (University of Wisconsin - Department of Chemical Engineering). Na instalação é fornecido um editor de código da linguagem utilizada no Octave, que é a mesmo do Matlab, o M-code.

A *Tabela 2.8.1* apresenta um quadro comparativo dos ambientes de processamento matemático analisados, levando em conta alguns pontos considerados relevantes para a escolha do ambiente a ser utilizado no desenvolvimento do trabalho.

	Maple	Maxima	Matlab	Octave
Proprietário / Público	Proprietário	Público	Proprietário	Público
Permite salvar o estado do programa	SIM	SIM	SIM	SIM
Visualização de gráficos 2D e 3D	SIM	SIM	SIM	SIM
Interação com o gráfico	Apenas rotação	Apenas rotação	SIM	Apenas rotação
Alteração do gráfico sob demanda	SIM	NÃO	SIM	NÃO
Criação de scripts pelo usuário	SIM	NÃO	SIM	SIM
Fornece meios de utilizar suas funções em outros programas	SIM	NÃO	SIM	SIM

Tabela 2.8.1: Comparação dos Ambientes de Processamento Matemático avaliados

Ao realizar este levantamento, observou-se que os ambientes analisados não forneciam controle suficiente para que o benchmark fosse implementado como planejado. Então, ficou decidido que seria desenvolvida uma implementação própria para a avaliação polinomial onde o usuário poderia implementar suas próprias

bibliotecas de avaliação polinomial. Quanto à parte gráfica, decidiu-se que seria utilizada uma biblioteca externa sem ligação com nenhum ambiente de processamento matemático.

3. ESPECIFICAÇÃO DA PROPOSTA

3.1. OBJETIVO GERAL

Considerando que todo e qualquer cálculo realizado em precisão finita está sujeito ao erro de arredondamento, a realização de cálculos em ponto flutuante, em computadores, não está isenta de erro. Todo e qualquer cálculo com números de ponto flutuante e executado em um computador está sujeito ao erro de arredondamento. Mesmo com a padronização da aritmética de ponto flutuante, pode acontecer que computadores diferentes gerem resultados diferentes para o mesmo cálculo.

Alguns fatores influenciam o erro de arredondamento, entre eles a maneira como o cálculo é realizado e o tipo de dado utilizado. Estes fatores também influenciam o desempenho do cálculo. Normalmente, quanto melhor a qualidade do resultado, mais tempo é consumido para obtê-lo. Achar o ponto de equilibro entre qualidade do resultado e tempo necessário é fundamental para o cálculo científico.

A proposta deste Trabalho de Conclusão é prospectar possíveis pontos de equilíbrio, mantendo a vinculação com cenários de ocorrência de erro de arredondamento delineados a partir da aritmética de ponto flutuante segundo o padrão IEEE-754. Será proposto um benchmark para o processamento numérico que pondera a relação entre qualidade do resultado e o tipo de dado utilizado na declaração de variáveis que correspondem aos operandos de operações aritméticas em ponto flutuante.

3.2. OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral deste trabalho, foram definidos dois objetivos específicos: propor uma metodologia de benchmarks e implementar a metodologia proposta de forma funcional. O primeiro objetivo, a criação de uma metodologia de benchmarks, se justifica na medida em que os benchmarks existentes não consideram a qualidade do resultado obtido a partir da realização de cálculos em ponto flutuante. Por sua vez, o segundo objetivo possibilitará a análise de resultados através de sua exploração visual.

3.3. ARQUITETURA (PROTÓTIPO)

O cenário de desenvolvimento do trabalho se configura a partir do problema da qualidade do resultado proveniente da avaliação polinomial, segundo diferentes formas de avaliação – Horner, potências, produto de raízes - e diferentes formatos de números de ponto flutuante, assim como especificados no padrão IEEE-754. A metodologia de benchmark utilizada observa os critérios especificados pelo PARKBENCH Committee.

Assim, desenvolveu-se uma ferramenta que realiza a avaliação polinomial, segundo diferentes formas de avaliação e formatos de números de ponto flutuante. Os resultados obtidos são exibidos graficamente, possibilitando sua análise. Cada resultado é aplicado ao benchmark que calcula um valor de referência para a comparação entre qualidade do resultado e tempo para sua obtenção. Para tanto, teve-se o cuidado de utilizar números de multiprecisão que podem ter um grande número de dígitos e, quem sabe, possibilitar a obtenção de resultados menos sujeitos ao erro de arredondamento.

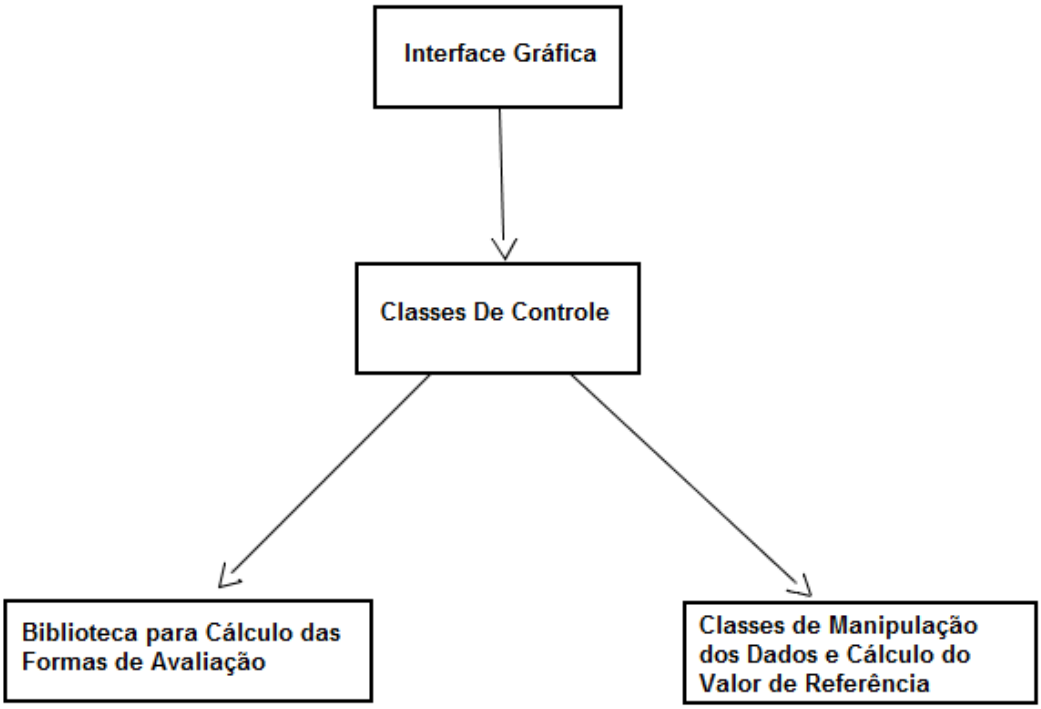

Figura 3.3.1: Modelagem do Protótipo da Ferramenta

A ferramenta foi divida em 4 módulos. No primeiro, chamado de “interface gráfica”, ficam as classes de interação com o usuário. Estas classes terão o acesso limitado ao módulo seguinte, chamado de “classes de controle”, que é responsável pelas operações de alto nível e fluxo do programa. Para controlar o fluxo do programa, ele acessará os módulos “classes para cálculo das rotinas matemáticas” e “bibliotecas do ambiente de processamento matemático”. As classes são responsáveis, respectivamente, pela execução dos algoritmos matemáticos, produção de dados e geração dos gráficos.

3.4. O NOVO BENCHMARK

O benchmark proposto pelo autor e apresentado neste trabalho tem por foco medir tanto a exatidão quanto o desempenho e estabelecer uma relação entre estas medidas tendo a avaliação polinomial como cenário. Como estes dois parâmetros dependem de diversas variáveis, o benchmark levará em consideração: o polinômio a ser avaliado; a faixa de valores em que o mesmo será avaliado; o método utilizado na avaliação e a precisão utilizada nos cálculos.

Destaca-se que as formas de avaliação polinomial serão analisadas com relação à eficiência e desempenho, sendo que a qualidade do resultado é o mais importante.

A estratégia adotada por este benchmark consiste em realizar a avaliação polinomial através de diferentes formas de avaliação, a partir de diferentes valores, e utilizando possibilidades de precisão definidas pelo padrão IEEE. Para isto foi utilizada uma bateria de polinômios de teste que inclui polinômios de natureza diversa, um conjunto de valores pré-definidos que possibilitarão a avaliação polinomial, um conjunto de formas de avaliação polinomial escolhidas por sua complexidade algébrica. O resultado de cada avaliação polinomial é armazenado e, a seguir, utilizado no cálculo da pontuação das formas de avaliação, tomando por base estes valores.

O cálculo da pontuação, proposto pelo autor, considera uma estimativa do *erro de arredondamento*, a exatidão e o desempenho da avaliação. Estes fatores são ponderados na seguinte formulação:

$$\frac{1}{T\varepsilon^2},$$

onde T é o tempo de execução e ε é o *erro de arredondamento*.

3.4.1. METODOLOGIA

O benchmark proposto consiste em calcular, para cada valor contido no intervalo de avaliação, um valor de referência que é o valor com o qual os resultados serão comparados com relação à exatidão. Este valor é gerado através da aplicação da forma de Horner, usando aritmética de multiprecisão e, por isto, considera-se que este valor tem uma exatidão confiável. Após obter os valores de referência, o sistema avalia o polinômio utilizando as técnicas de avaliação que serão analisadas para cada valor do intervalo de avaliação. Então são obtidos o tempo de execução, o resultado e a precisão utilizada. Com estas informações, o sistema compara os valores de referência aos valores obtidos através da aplicação das diferentes formas de avaliação polinomial. Calcula-se o erro absoluto, o erro relativo e a exatidão, utilizando DIGSE (medida de precisão que calcula os dígitos significativos exatos) e obtém-se o tempo de execução. Por fim, o ranking é calculado, a partir da fórmula:

$$\frac{1}{tempo + (erroRelativo)^2}.$$

Segundo esta fórmula, quanto maior o tempo e o erro relativo, menor a pontuação atribuída. O erro relativo tem um peso maior que o tempo, devido ao fato de que o benchmark está focado mais na qualidade do que no tempo para obter a resposta. Os valores das variáveis de tempo e erro relativo são obtidos, respectivamente, somando-se o tempo de todas as avaliações e todos os erros relativos dentro do intervalo.

Estas informações são apresentadas ao usuário a fim de que este possa analisar os dados da melhor forma possível.

3.4.2. IMPLEMENTAÇÃO DO BENCHMARK

Na seção anterior está descrito o funcionamento do benchmark. Nesta seção será apresentado o detalhamento da estrutura do programa implementado para a viabilização do benchmark, bem como detalhes e decisões de implementação.

O diagrama da *figura 3.4.1* apresenta a estrutura do sistema que implementa o benchmark.

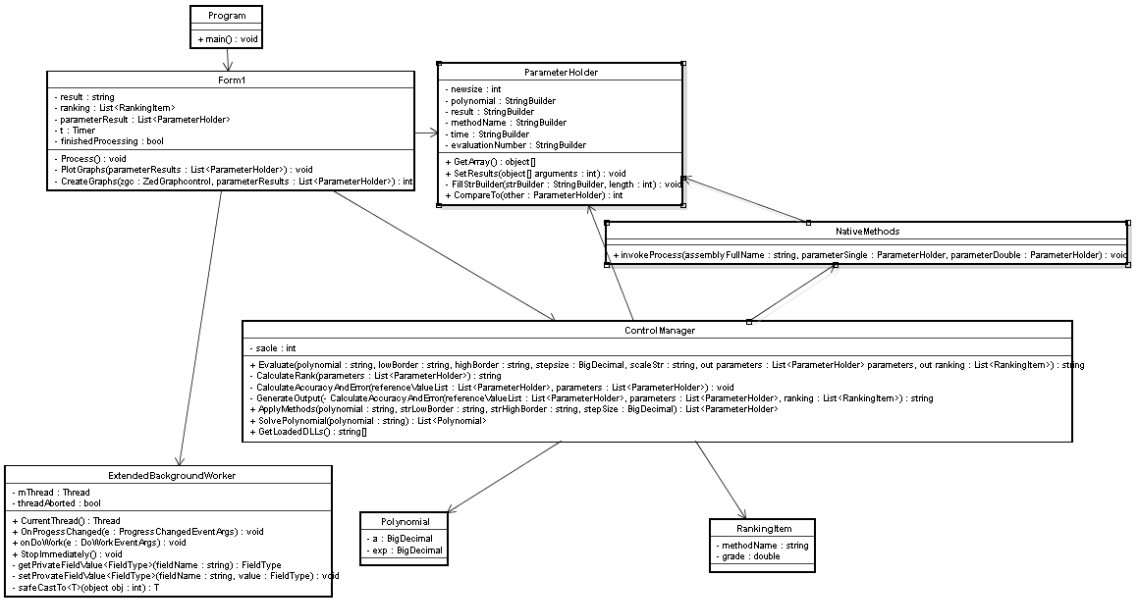

Figura 3.4.1: Diagrama de classes do sistema.

Na figura 3.4.1 é possível observar que existe uma classe que gerencia a execução do benchmark. Esta classe é chamada de ‘ControlManager’. Ela é responsável pela execução dos passos explicados na seção anterior. A classe ‘Form1’ implementa a interface com o usuário. A ‘ParameterHolder’ é a classe que representa os parâmetros de entrada e saída de cada chamada dos métodos de avaliação. No final da execução é apresentada uma lista de ‘ParameterHolder’ que contém todos os valores de x e para todos os métodos de avaliação. A classe ‘NativeMethods’ chama por ‘reflection’ as DLLs que executam os métodos de avaliação. A ‘Polynomial’ é a classe que representa um termo do polinômio que é representado por uma lista desta classe, e a ‘RankingItem’ é a classe que representa uma método e sua pontuação no ranking calculado pelo sistema. Existe também a classe ‘ExtendedBackgroundWorker’, que gerencia o trabalho em segundo plano na execução do benchmark (Osherove).

Para implementação do benchmark, optou-se por tomar o tempo de execução de maneira a levar em conta apenas a execução do método de avaliação em si e isto é realizado pela DLL que calcula e retornar este valor. Caso a tomada de tempo não fosse realizada desta forma, o tempo incluiria todo o tempo de acesso à DLL, o que alteraria os valores e diminuiria os significados dos mesmos. Outra decisão importante foi quanto ao uso de 'reflection'. Com este processo, um tipo particular de metaprogramação (Osherove), um programa pode observar sua estrutura e a de outros programas, assim como modificar sua própria estrutura em tempo de execução. Isto faz com que o programa passe a utilizar automaticamente as DLLs, que estão no diretório de bibliotecas dos métodos de avaliação sem alteração de código, facilitando a comparação e possibilitando que os usuários codifiquem seus próprios métodos. As desvantagens de utilizar este recurso são que este processo é mais custoso que o acesso direto a uma biblioteca e que as DLLs devem ser desenvolvidas utilizando o .NET Framework.

Os parâmetros das formas de avaliação são, em sua maioria, strings mesmo quando são valores numéricos. Isto acontece porque não se sabe qual a precisão que a forma de avaliação irá utilizar internamente. Então, não seria seguro definir um tipo de dado que corresse o risco de ser de uma precisão menor do que a utilizada pela forma de avaliação, pois isto provavelmente geraria um arredondamento. Para contornar esta situação, os valores são retornados como strings e armazenados internamente em números de precisão infinita, evitando ao máximo o *erro de arredondamento*.

Um detalhe observado é que, muitas vezes, o tempo de execução da avaliação retornava como *zero*. Como a execução obrigatoriamente consumiu algum tempo, acreditou-se que esta anormalidade aconteceu pelo fato do tempo ser muito pequeno. Para isto, dentro da DLL dos métodos de avaliação, executa-se n vezes o método, medindo-se o tempo, e depois divide-se o tempo por n para obter a média de tempo de uma execução.

3.5. FERRAMENTA

A qualidade da resposta da avaliação polinomial depende muito do polinômio utilizado e do valor da variável para o qual ele será avaliado. Como já vimos, existem

algumas situações que são mais críticas do que outras (como resultados intermediários muito próximos a zero) e, às vezes, a dupla método-precisão pode ser melhor em uma situação e pior em outra. Assim, a forma de Horner tem melhor desempenho em precisão simples do que em precisão dupla, porém a chance de haver *erro de arredondamento* em precisão simples é maior. Então, é preferível utilizar a precisão simples, caso a qualidade do resultado seja aceitável, e utilizar precisão dupla somente quando o erro de arredondamento for inaceitável. Para verificar esta situação particularmente interessante, o benchmark proposto possui uma ferramenta para a execução deste teste.

A ferramenta para análise de Métodos de Avaliação Polinomial - MAP é um sistema que busca estabelecer uma comparação entre métodos de avaliação polinomial através de um benchmark. Para facilitar o entendimento pelo usuário desta comparação, o sistema MAP possibilita a visualização gráfica dos resultados individuais dos métodos e também sua pontuação final. Esta pontuação é calculada através de uma fórmula matemática que leva em conta a velocidade para obtenção do resultado com qualidade (exatidão). Para verificar qual a melhor dupla técnica de avaliação polinomial-precisão para uma dada situação, o sistema MAP possibilita a definição do polinômio a ser avaliado e do valor da avaliação, bem como a escolha dos métodos utilizados.

O método ideal é aquele que avalia o polinômio com a qualidade necessária e no menor tempo possível, ou seja, a exatidão necessária do resultado é quem determina a escolha da forma de avaliação mais adequada. Não tem valor um resultado obtido extremamente rápido, mas com nenhum dígito significativo exato, da mesma forma que não tem necessidade de gastar horas para obter um resultado com muitos dígitos significativos exatos quando se necessita de apenas uma exatidão de dois ou três dígitos. Então, o principal objetivo desta ferramenta é ajudar o usuário a tomar a decisão sobre qual método e precisão utilizar em diferentes situações. Em especial, o sistema MAP possibilita comparar diferentes formas de avaliação polinomial.

3.5.1. MODO DE USO

A *figura 3.5.1* ilustra a tela principal do sistema MAP.

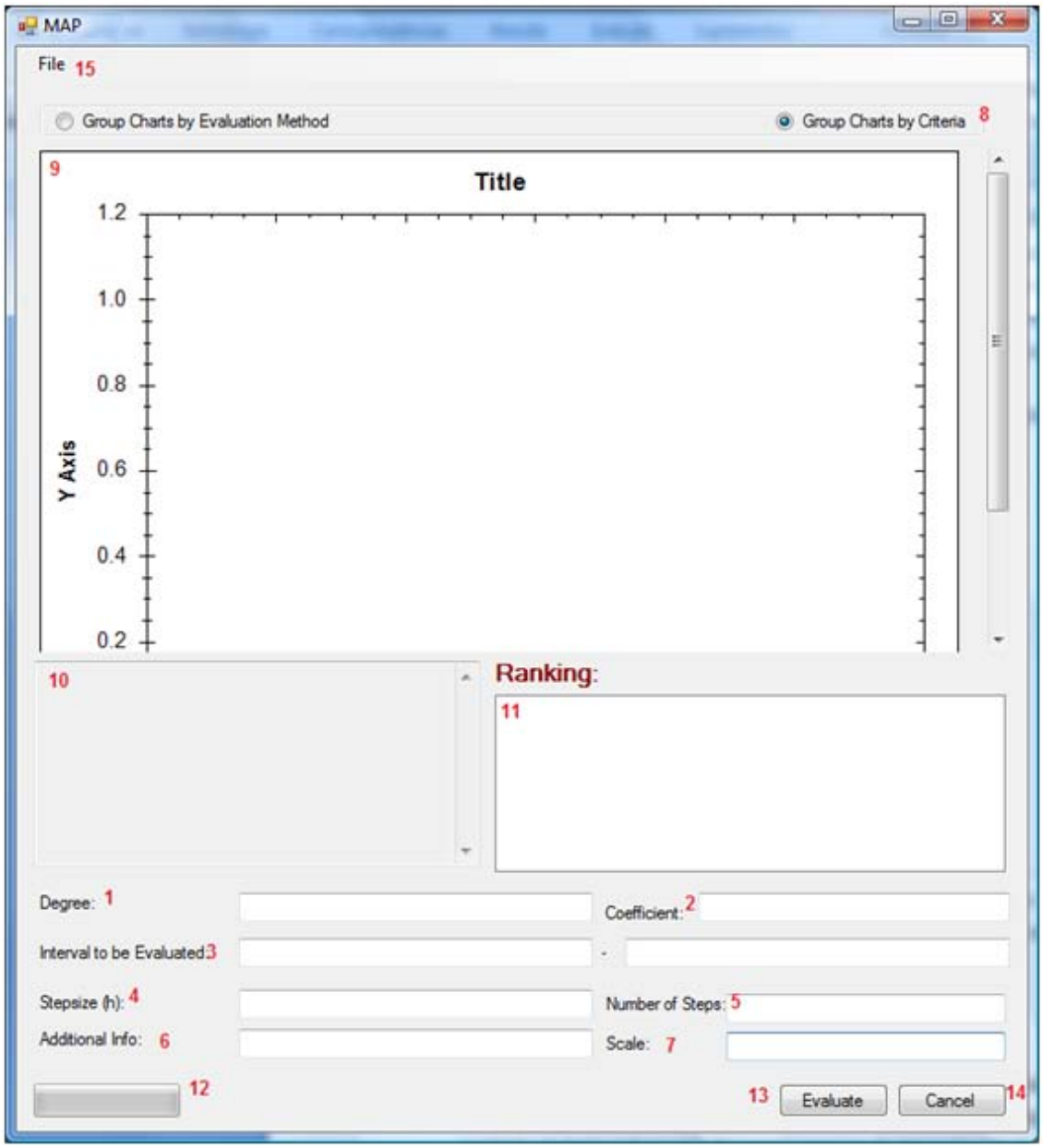

Figura 3.5.1: 1Tela inicial do sistema

Na tela apresentada na *figura 3.5.1,* as informações necessárias para o funcionamento da ferramenta são informadas nos campos identificados nos itens 1 a 7 e os resultados são mostrados ao usuário nos itens 9 a 11. Os demais itens são para a utilização da ferramenta, como chamar funções e iniciar a execução da mesma:

- **Item 1**: Informar o grau do polinômio utilizado para avaliação.
- **Item 2**: Informar os coeficientes do polinômio, separados por ponto-e-vírgula.
- **Item 3**: Informar os limites inferior e superior do intervalo de avaliação.

- **Itens 4 e 5**: Pelo menos um deles deve ser informado. No item 4 é informado o incremento que deve ser adicionado a cada avaliação, partindo do limite inferior até chegar no superior do intervalo. No item 5 é informado o número de avaliações a ser realizado dentro deste intervalo.
- **Item 6**: Informação adicional para a execução dos métodos como, por exemplo, as raízes para o método do produto de raízes.
- **Item 7**: Valor que define a precisão dos números de multiprecisão utilizados no benchmark. Quanto maior este valor, maior a precisão.
- **Item 8**: Permite ao usuário selecionar como os gráficos serão organizados: se serão agrupados por método, ou seja, um gráfico para cada método de avaliação e o tempo, erro relativo e absoluto e exatidão em um único gráfico, ou se as informações serão agrupadas por estes critérios, gerando quatro gráficos (tempo, erro relativo e absoluto e exatidão) com as informações de todos os métodos dentro de cada gráfico.
- **Item 9**: Apresentação, para o usuário, da representação gráfica do valor do erro absoluto, erro relativo e exatidão das formas de avaliação.
- **Item 10**: Apresentação, para o usuário, de uma saída de texto para cada valor de avaliação de cada método, com as seguintes informações: nome do método, valor avaliado, resultado, resultado do valor de referência, tempo de execução, erro absoluto, erro relativo e exatidão.
- **Item 11**: Apresentação, para o usuário, do ranking das formas de avaliação. Através da fórmula proposta pelo autor, gera-se uma pontuação para cada forma de avaliação a partir da qual elas são classificadas.
- **Item 12**: Barra de progresso - a execução do benchmark pode demorar muito tempo. Para verificar que o sistema está funcionando observa-se o movimento da barra de progresso.
- **Item 13**: Botão para iniciar a execução do benchmark.
- **Item 14**: Botão para interromper a execução do benchmark.
- **Item 15**: Menu - a única opção presente no momento é a de abrir um arquivo já salvo.

Uma vez informados estes dados, clicar no botão “Evaluate” e aguardar enquanto o sistema MAP realiza a execução do benchmark.

Adicionar novo Método de Avaliação Polinomial

Para adicionar uma nova forma de avaliação, basta codificá-la utilizando o .NET framework, desenvolvido pela Microsoft, para gerar uma DLL. Dentro desta DLL devem constar as formas de avaliação com as seguintes assinaturas:

```
static void xxxSingle (String^ degree, String^ coefficients, char* evaluationNumber, char* answer, int answerMaxLength, char* methodName, int nameMaxLength, char* time, int timeMaxLength, char* precision, int precisionMaxLength,String^ additionalInfo)

static void xxxDouble (String^ degree, String^ coefficients, char* evaluationNumber, char* answer, int answerMaxLength, char* methodName, int nameMaxLength, char* time, int timeMaxLength, char* precision, int precisionMaxLength,String^ additionalInfo)
```

É importante destacar que é necessário observar o seguinte cuidado na realização desta ação: o nome da classe deve ser o nome da DLL mais o sufixo “Class”, e os nomes dos métodos devem ser o nome da DLL mais os sufixos “Single” e “Double”. Exemplo:

```
DLL: Horner.dll
Classe: HornerClass
Métodos:HornerSingle e HornerDouble
```

A princípio, os sufixos ‘Single’ e ‘Double’ significavam a avaliação utilizando precisão simples e dupla respectivamente mas, como o sistema permite que se utilize internamente qualquer precisão, estes sufixos não têm mais um significado. Ainda assim é importante que a DLL tenha ambos os métodos. Os parâmetros são:

- **degree** : string que contém o grau do polinômio. (Parâmetro de entrada)
- **coefficients**: string com os coeficientes do polinômio. (Parâmetro de entrada)
- **evaluationNumber**: string que contém o valor com o qual o polinômio deve ser avaliado. (Parâmetro de entrada)
- **answer**: string onde o resultado deve ser informado. (Parâmetro de saída)
- **answerMaxLength**: número máximo de caracteres que são aceitos na resposta. (Parâmetro de entrada)
- **methodName**: string onde o nome do método deve ser informado. (Parâmetro de saída)

- **nameMaxLength**: número máximo de caracteres que são aceitos no nome do método. (Parâmetro de entrada)
- **time**: string onde o tempo deve ser informado. (Parâmetro de saída)
- **timeNameMaxLength**: número máximo de caracteres que são aceitos no tempo. (Parâmetro de entrada)
- **precision**: string onde a precisão utilizada deve ser informada. (Parâmetro de saída)
- **precisionNameMaxLength**: número máximo de caracteres que são aceitos na precisão. (Parâmetro de entrada)
- **additionalInfo**: informação adicional que o método possa precisar. Por exemplo: as raízes no método do produto de raízes.

Após compilar a DLL, deve-se copiá-la para a pasta “EvaluationMethods”. A próxima vez que o botão “Evaluate” for acionado, a DLL será incluída na análise.

4. RESULTADOS E ANÁLISES

Neste capítulo, iniciamos apresentando os casos de teste e os resultados obtidos a partir da execução destes testes no sistema MAP e, a seguir, são apresentadas as conclusões do trabalho.

4.1. CASOS DE TESTE

Definiu-se um conjunto de polinômios utilizado no benchmark e este conjunto foi dividido em dois grupos: o grupo A é constituído por polinômios que possuem raízes complexas; o grupo B é constituído por polinômios que não possuem raízes complexas.

	Polinômio	Raízes	Grupo
p1	$x^5 + x^3 - 10^{-15}$	• **1** • 0,000005 + 0,00000866 • −0,00001 • **−1** • 0,000005 − 0,00000866	A
p2	$64x^7 - 80x^5 + 8x^4 + 4x^3 - 2x^2 + 3x - 1$	• 1,0350822**29** • 0,3507431080 + 0,123 • −0,03091855376 + 0,41 • −0.5564529712 • −1.118278367 • −0,03091855376 − 0.41 • 0,3507431080 • −0,1230360295 *I*	A
p3	$x^2 + x - 10^{-16}$	• **0** • **−1**	B

p4	$8118x^4 - 11{,}482x^3 + x^2 + 5{,}741x - 203$	• 0,0007071938901 + 0,7 • 0,0007071938901 − 0,7 • 0,707106781**2** • −0,707106781**2**	A
p5	$x^3 + x + 0{,}0001$	• 0,0000499999995 + 1,0 • −0,000099999999 • 0,0000499999995 − 1,0	A
p6	$31{,}5x^5 - 40x^3 + 0{,}75x^2 + 13{,}5x + 0{,}75$	• 0,8158079556 + 0,1654 • −0,056257627**92** • −0,689361146**2** • −0,885997137**1** • 0,8158079556 − 0,1654	A
p7	$x^2 - 4x + 3$	• **1** • **3**	B
p8	$x^3 - 13x^2 + 20x + 100$	• **−2** • **5** • **10**	B
p9	$x^2 - 4x + 4$	• **2** • **2**	B

Tabela 4.1.1: Polinômios que constituem a bateria de testes

Para efetuar os testes, escolheu-se, para cada polinômio, uma raiz aleatoriamente e, para ela, um intervalo com o limite inferior 0,5 abaixo da raiz e 0,5 acima da mesma. Para este intervalo, definiram-se 100 avaliações, além das avaliações nas duas extremidades do mesmo, totalizando 102 avaliações. Para o método do produto de raízes, é necessário informar as raízes dos polinômios. Para obtê-las, foi utilizado o ambiente de processamento matemático Maple, e as raízes

foram inseridas no sistema MAP do mesmo modo que o Maple as retornou, sem alteração, mesmo que houvesse erro de arredondamento.

O tempo necessário para execução do benchmark não é muito grande, principalmente, quando considerando a quantidade de métodos e do número de avaliações realizadas em cada método. Com 6 métodos e 5.000 avaliações para cada, necessita-se de aproximadamente 1 hora e meia. Já com 100 avaliações, necessita-se de aproximadamente 10 minutos.

4.2. RESULTADOS

Os resultados a seguir foram obtidos através da execução do benchmark para cada um destes polinômios. O resultado de *p1* será mostrado de forma ampliada para que seja possível explicar como os gráficos são interpretados.

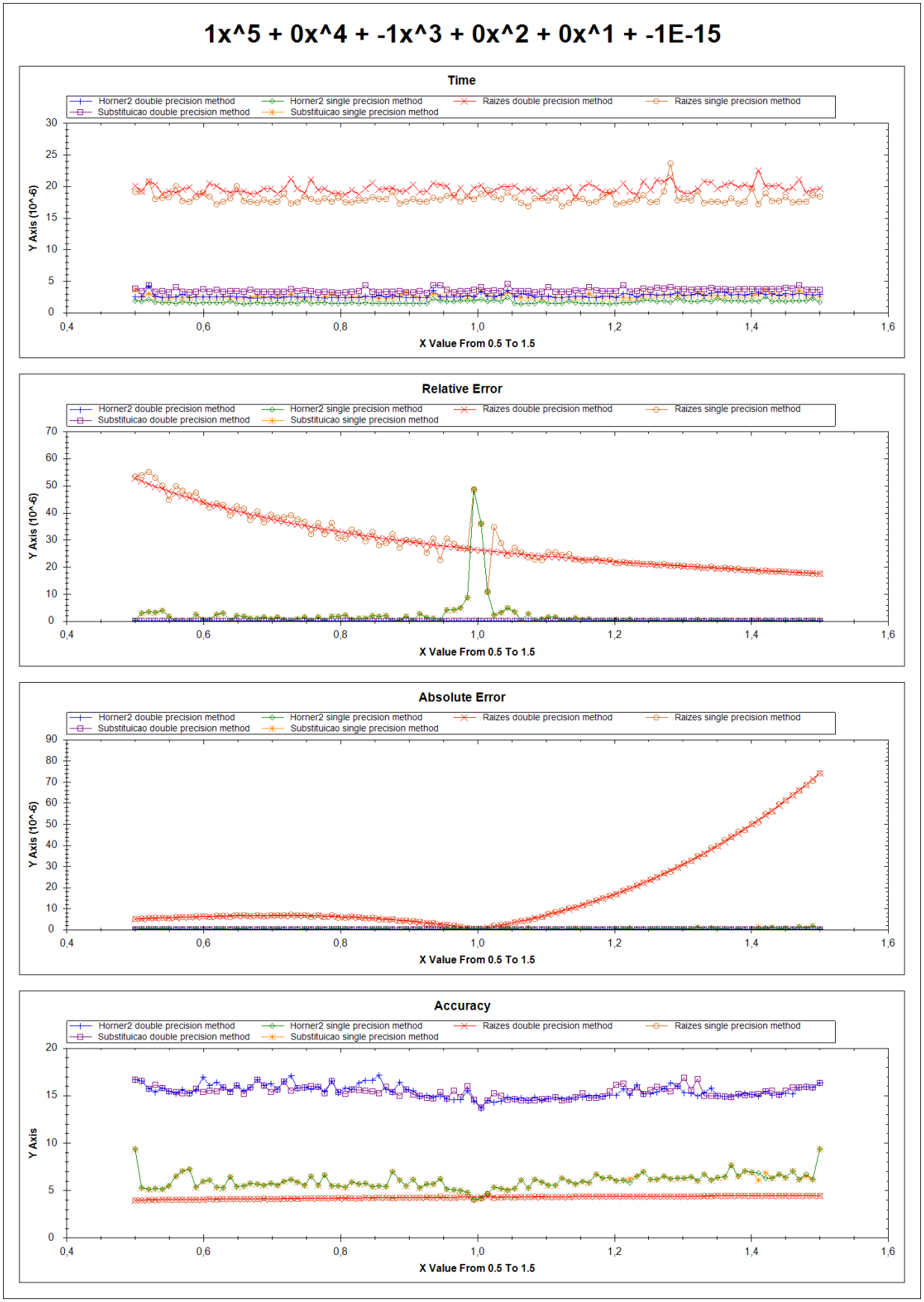

Figura 4.2.1: Polinômio *p1*

Nestes gráficos é possível verificar o comportamento dos métodos de avaliação para *p1* sob vários pontos de vista no intervalo em que o benchmark foi executado (de 0,5 a 1,5 neste caso).

No primeiro gráfico, analisa-se o tempo que cada método demorou para avaliar o polinômio para cada ponto. Neste caso, é possível notar que os métodos do produto das raízes (vermelho para precisão dupla e marrom para simples) precisaram de mais tempo para a avaliação. Já os métodos de Horner (azul para precisão dupla e verde para simples) e da potência (roxo para precisão dupla e amarelo para simples) precisaram de muito menos tempo para a avaliação do polinômio.

No segundo gráfico, analisa-se o erro relativo. Nele também é possível perceber que os métodos do produto de raízes tem um erro relativo maior neste polinômio. Para os outros métodos, com precisão simples ocorreu um crescimento no erro relativo próximo à raiz, enquanto que, com a dupla, este crescimento só é visível com a utilização de zoom.

O erro absoluto também pode ser analisado. Pode-se ver que o erro absoluto para o método do produto cai próximo à raiz. O comportamento dos outros métodos só pode ser melhor analisado com o uso do zoom.

No último gráfico, pode-se observar o comportamento da exatidão. Vemos que os métodos com precisão dupla têm maior exatidão, enquanto os simples têm uma exatidão mediana. O método do produto de raízes apresenta uma exatidão inferior.

Devido à disposição dos pontos nos gráficos, muitas vezes estes podem ficar confusos, com muitos pontos próximos uns dos outros, dificultando a análise. Também pode acontecer que, por um valor e as alterações deste ao longo das avaliações serem muito pequenas, não seja possível detectar estas alterações. Para solucionar este problema, é possível utilizar a ferramenta de zoom. Esta permite que seja selecionada uma área na tela que será expandida, dando ênfase a esta e escondendo a área que não é interessante no momento.

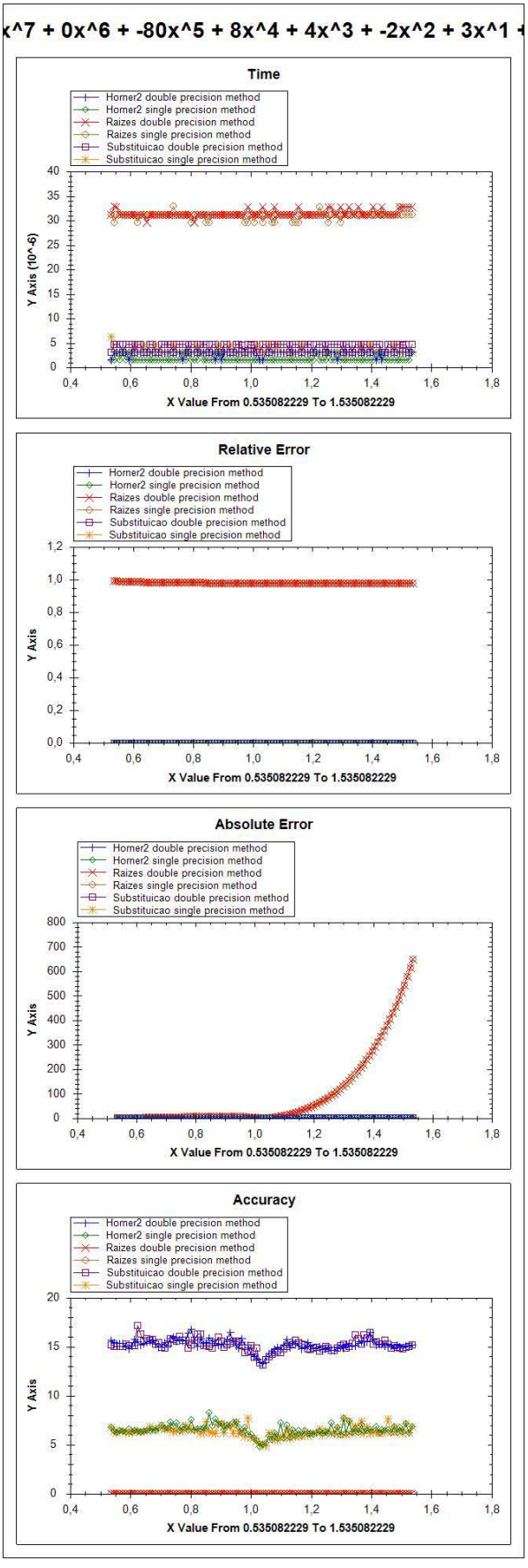

Figura 4.2.2: Polinômio *p2*

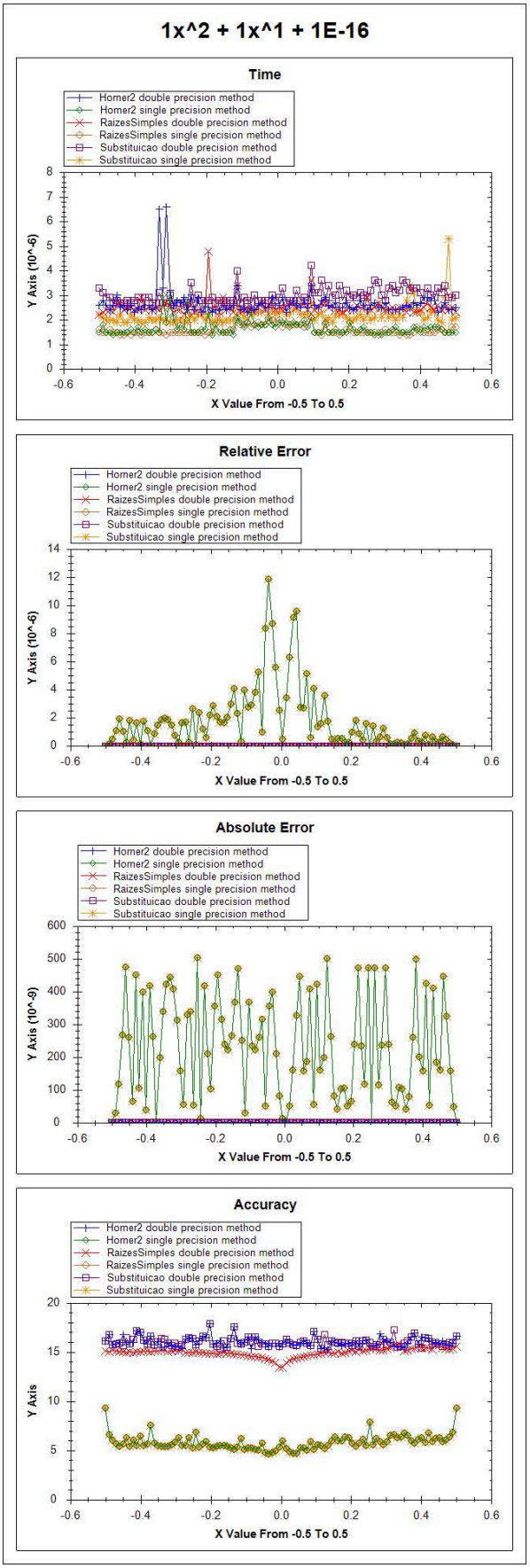

Figura 4.2.3: Polinômio *p3*

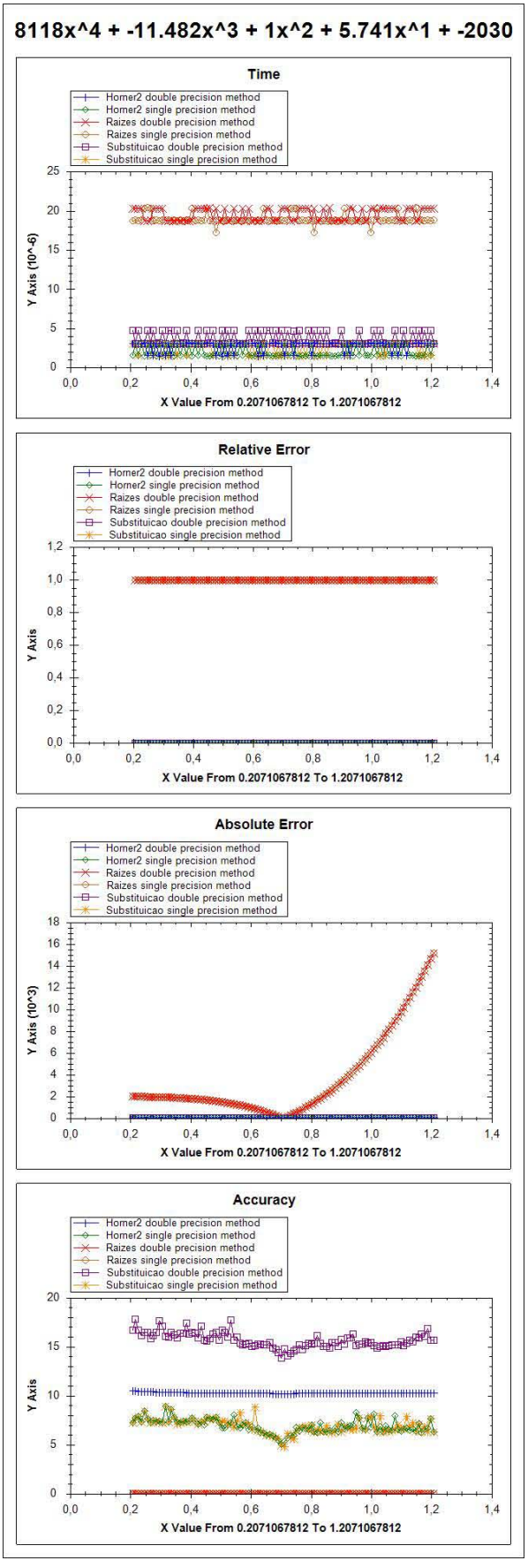

Figura 4.2.4: Polinômio *p4*

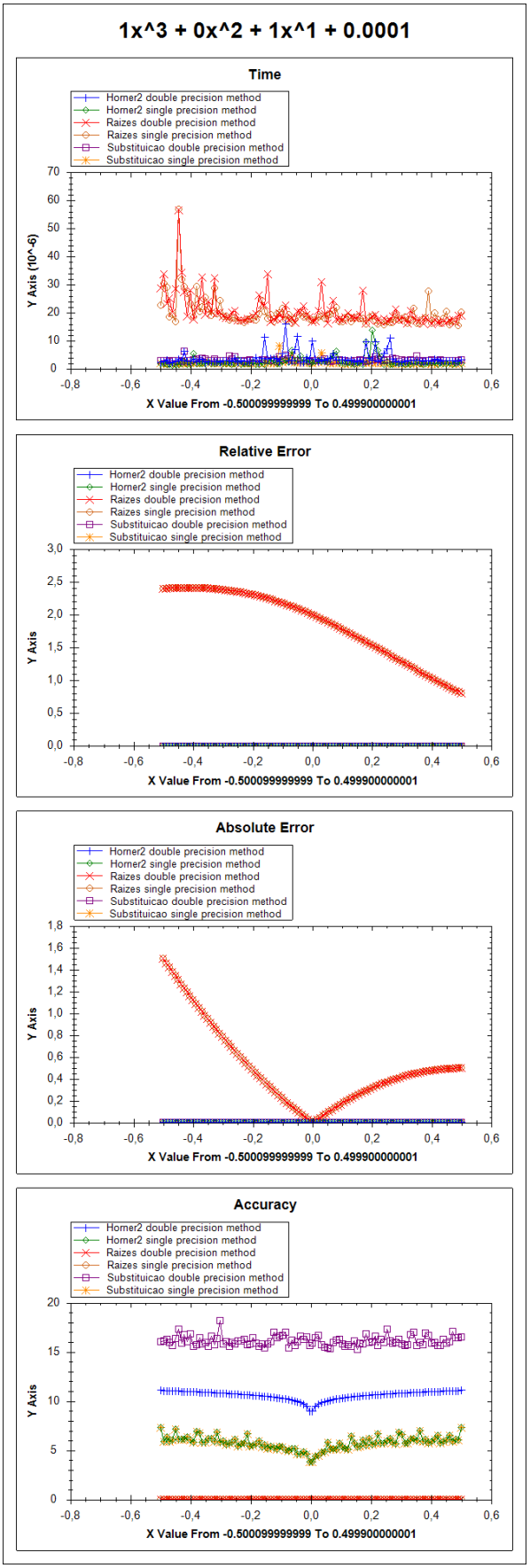

Figura 4.2.5: Polinômio *p5*

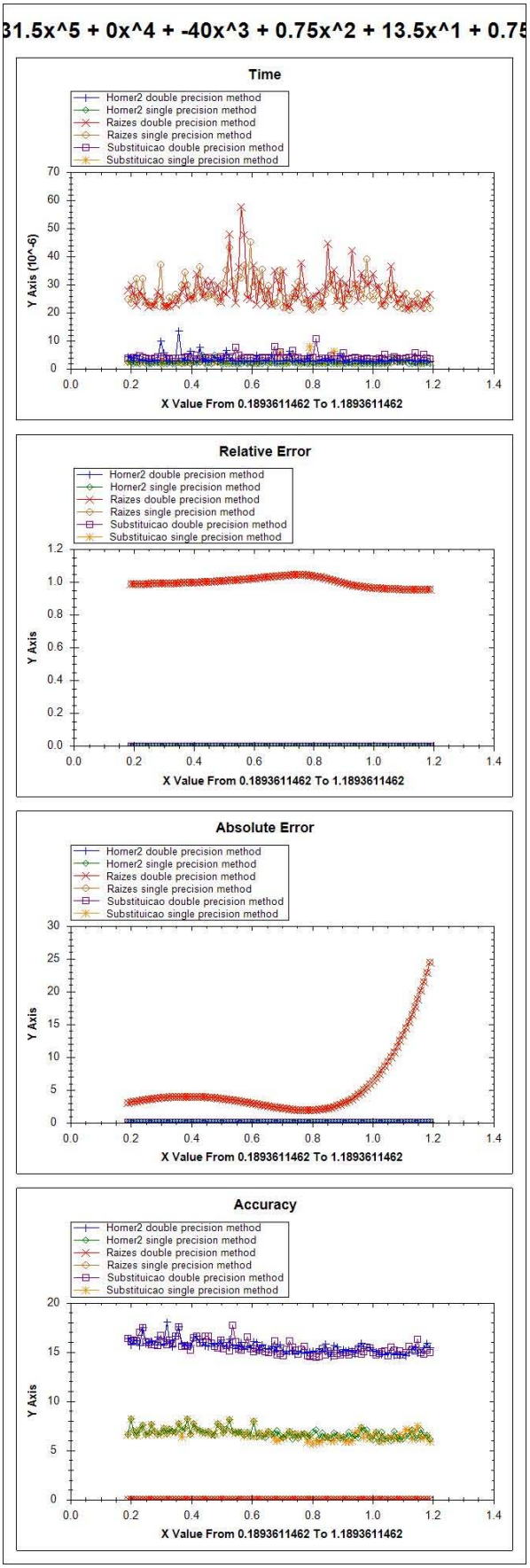

Figura 4.2.6: Polinômio *p6*

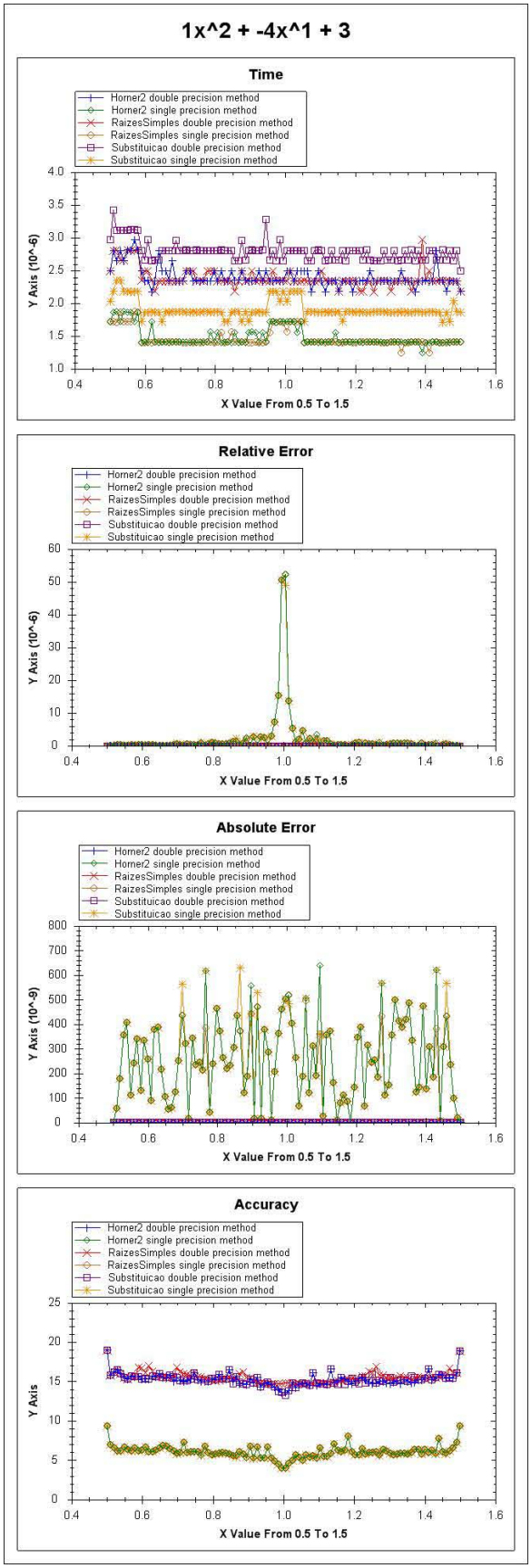

Figura 4.2.7: Polinômio *p7*

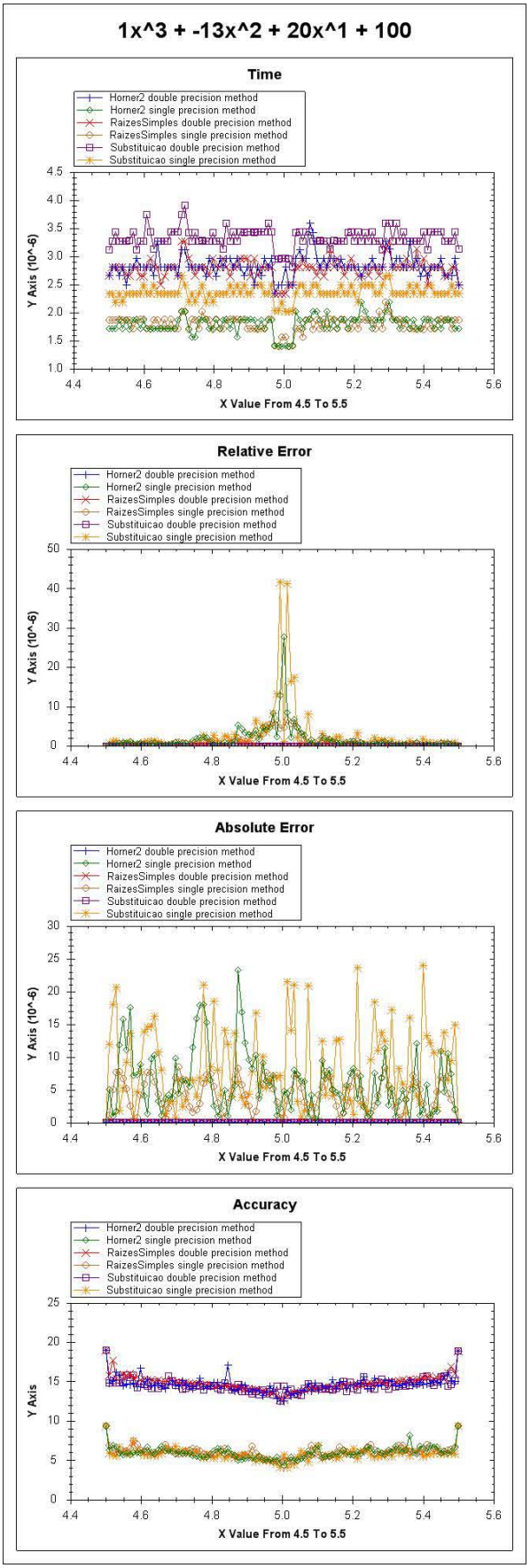

Figura 4.2.8: Polinômio *p8*

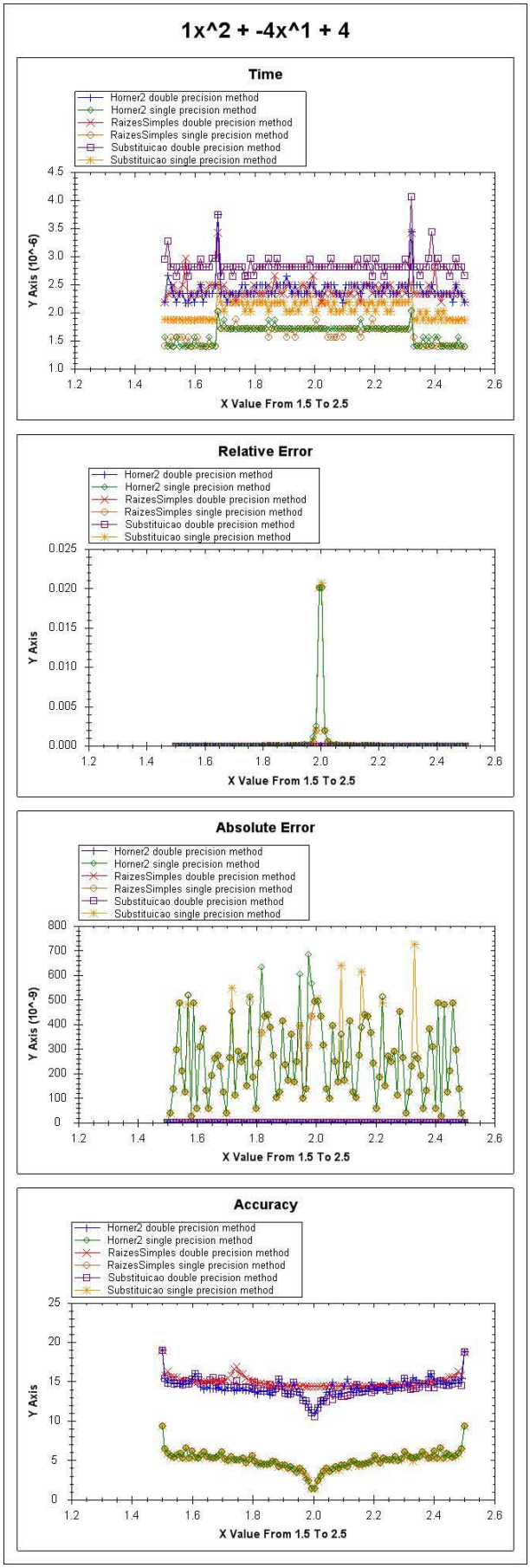

Figura 4.2.9: Polinômio *p9*

4.3. ANÁLISE DOS RESULTADOS

Os resultados foram analisados comparando os comportamentos dos gráficos dos polinômios entre si com a literatura. É importante destacar que não é válido comparar os valores dos gráficos entre si, como comparar o tempo do método de Horner utilizando precisão dupla para *p1* e *p2*. O benchmark foi executado em máquinas e momentos diferentes para os polinômios, então esta comparação não é válida.

O primeiro critério avaliado foi o tempo. Com ele foi possível notar alguns comportamentos:

- No grupo A, a avaliação através da forma do produto de raízes, teve um tempo muito superior ao das outras formas. Isto aconteceu porque, para realizar esta avaliação, foi necessário utilizar a aritmética complexa e, assim, manipular um tipo de dados especial. Concluiu-se que a implementação do método, com a necessidade de instanciar um tipo de dados mais pesado do que tipos de dados primitivos, causou esta anomalia.
- No grupo B, o comportamento foi mais regular. Como pode ser observado na *figura 4.2.8*, relaciona-se, por ordem decrescente de tempo:
    - Método das potências com precisão dupla
    - Método de Horner e do produto de raízes com precisão dupla
    - Método das potências com precisão simples
    - Método de Horner e do produto de raízes com precisão simples.
- Excluindo o método do produto de raízes do grupo A, podemos verificar que esta ordem continua inalterada, como mostra a *figura 4.3.1*, onde foi feito o zoom do gráfico de *p6*, o que excluiu o método do produto de raízes do gráfico, mas possibilitou a visualização mais detalhada do comportamento do tempo dos outros quatro métodos de avaliação.

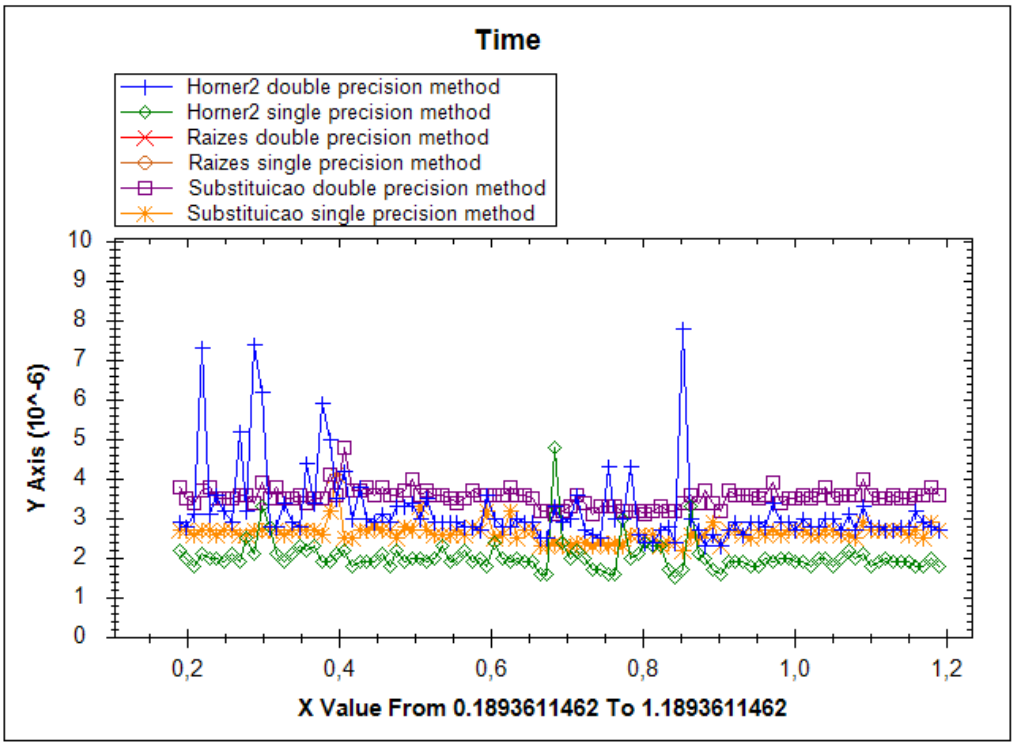

Figura 4.3.1: Zoom do gráfico de *p6*, excluindo os métodos do produto de raízes

Mais testes foram realizados para verificar se há algum motivo específico para estes picos no método de precisão dupla de Horner, mas, como os picos não eram constantes (ver *figura 4.3.2*), julgou-se que eram efeitos colaterais da utilização do processador.

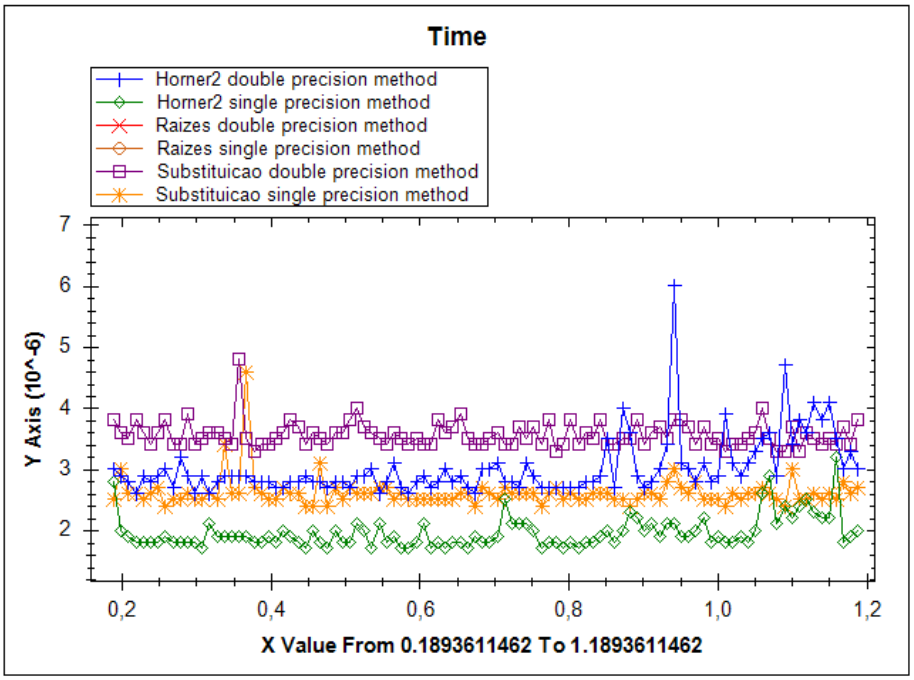

Figura 4.3.2: Nova avaliação do polinômio *p6*, que não apresenta os picos em Horner

Como é possível ver nas imagens, geralmente os métodos da potência são mais demorados. Isto acontece porque eles realizam $(2n-1)$ multiplicações e n somas. Já os métodos do produto de raízes e de Horner precisam de n multiplicações e n somas e, por isto, tendem a ser mais rápidos.

O erro relativo dos resultados também foi analisado, sendo que a seguir estão listadas as principais constações:

- No grupo A há um erro relativo muito grande no método do produto de raízes. Isto não se deve às raízes complexas, mas sim ao fato deste método exigir a entrada das raízes por parte do usuário. Com o ambiente de processamento matemático que utilizamos (Maple), obtivemos algumas raízes com erros de arredondamento, como foi o caso de *1* e *-1* para *p1*. Isto degradou a qualidade da saída do método de avaliação polinomial. Como este é um problema inerente ao método de avaliação, julgamos estes testes válidos. Os polinômios do grupo B foram escolhidos por terem raízes não complexas e por estas não conterem erros de arredondamento no seu cálculo (os polinômios foram gerados a partir de raízes pré-estabelecidas). Isto evitou o problema citado acima.
- No grupo B, é possível notar um comportamento mais uniforme, onde os métodos com precisão simples têm um erro relativo maior que os com precisão dupla.
- Alguns casos especiais foram identificados. No grupo B, as entradas para *p3* foram inseridas como $\mathbf{0}$ e $\mathbf{-1}$. Isto está incorreto, visto que $p3(-1) = p3(0) = 10^{-16}$ e acarretou um erro relativo maior quando foi utilizado o método do produto de raízes, principalmente próximo às raízes. Como é possível ver na *figura 4.3.3*, este método utilizando precisão dupla tem um erro relativo superior quando comparado aos dois outros com o mesmo tipo de dado. Ao contrário de outras situações, o erro inserido não foi grande a ponto da exatidão ficar inferior aos métodos que utilizam precisão simples, mas existe uma perda de exatidão que pode ser verificada também no gráfico de exatidão (*figura 4.3.4*).

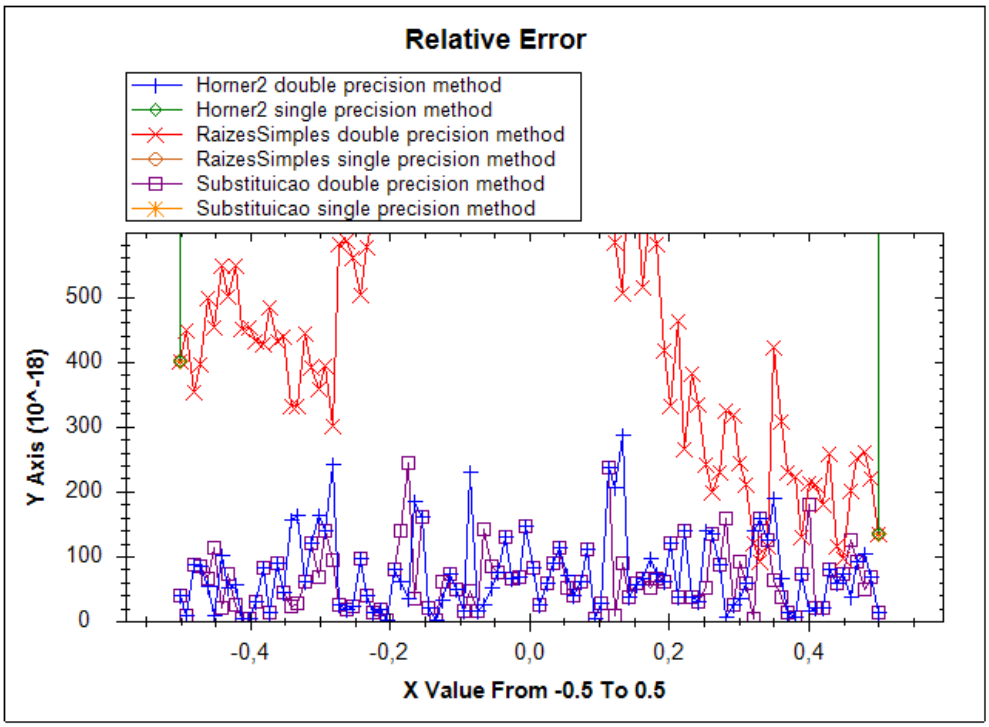

Figura 4.3.3: Zoom do método dos produtos de raízes, deixando claro o erro relativo superior deste método comparado aos outros dois.

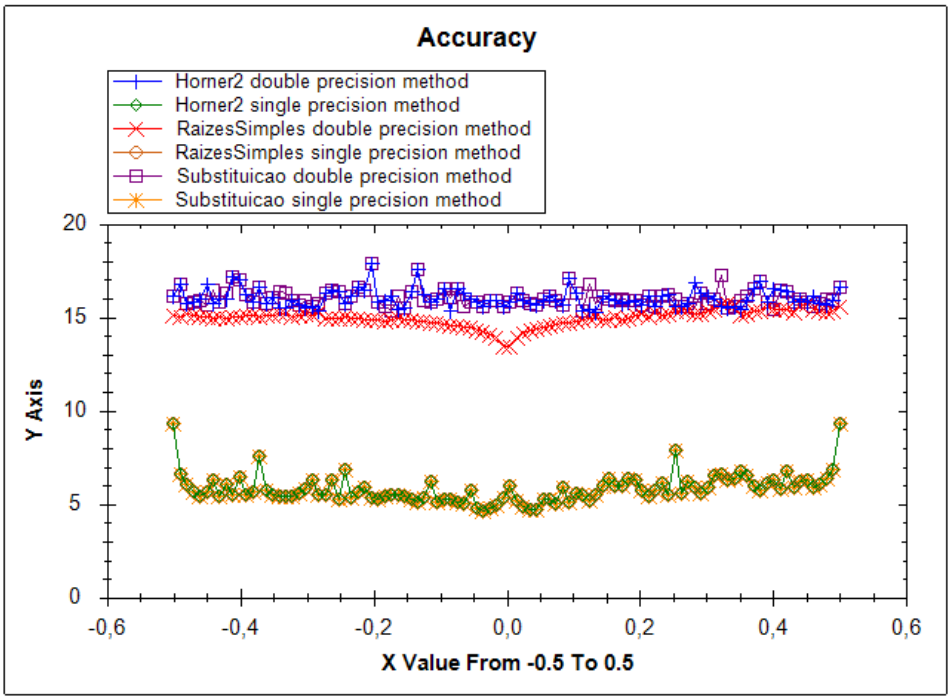

Figura 4.3.4: Gráfico da exatidão de *p3*. Pode-se notar uma queda na exatidão do método do produto de raízes.

- Outro ponto que influencia a qualidade do resultado é o arredondamento do valor de avaliação. Suponha-se que se deseja avaliar um intervalo de $diâmetro = 1$ em *150 pontos*. Dividir o diâmetro do intervalo pela quantidade de pontos, para obter o incremento, resulta em $0,\overline{6666}$. Os

métodos de avaliação receberão este valor e, quando ele for armazenado nas variáveis de tipo numérico, ocorrerá um arredondamento. Calcular a avaliação polinomial a partir de um valor arredondado introduz um erro. Acredita-se também que isto causa a oscilação do erro relativo, pois ela depende do ponto que está sendo avaliado.

Para o cálculo da exatidão, foi utilizada a fórmula de DIGSE, apresentada a seguir (Camponogara):

$$DIGSE(Valor_{Aprox}, Valor_{Ex}) = -\left(0{,}3 + \log\left(\frac{|Valor_{Ex} - Valor_{Aprox}|}{|Valor_{Aprox}|}\right)\right)$$

onde $Valor_{Ex}$ é o valor exato e $Valor_{Aprox}$ é o valor aproximado do qual se deseja calcular o erro.

O erro relativo a ser calculado por:

$$ER(Valor_{Aprox}, Valor_{Ex}) = \frac{|Valor_{Ex} - Valor_{Aprox}|}{|Valor_{Aprox}|}$$ (Camponogara)

Então o DIGSE pode ser descrito como:

$$DIGSE(Valor_{Aprox}, Valor_{Ex}) = -\left(0{,}3 + \log\left(ER(Valor_{Aprox}, Valor_{Ex})\right)\right)$$

(Camponogara)

Como o DIGSE está diretamente relacionado ao erro relativo, as alterações que ocorrem no erro relativo influenciam diretamente a exatidão do resultado. Nestes resultados, foi possível observar que:

- Nos gráficos do grupo A, com exceção de *p1*, a exatidão do método de produto das raízes é menor do que 30% de um dígito e isto é simbolizado no gráfico com a exatidão igual a zero. Em *p1*, a exatidão deste método não é tão pequena, ela fica entre 4 ou 5 dígitos. Mesmo assim, ele ainda é pior do que os outros métodos, até nas situações onde se utiliza precisão simples.
- Nos gráficos do grupo B e os do grupo A, excetuando os métodos com precisão dupla, pode-se notar facilmente a diferença de precisão dos dois tipos de dados. Os métodos de avaliação que utilizaram tipos simples ficaram muito abaixo dos métodos com precisão dupla.

- Os valores nos gráficos de exatidão podem parecer estranhos quando os valores double superam os 15 dígitos e os single superam os 6 dígitos corretos. Isto acontece quando o resultado tem um número de dígitos significativos menor do que o tamanho máximo do tipo de dado. Exemplo:
    - *P1* avaliado em 0,75 resulta em *-0,184570312500001*, que tem um número de dígitos significativos suficientes para ser representado em precisão dupla, causando erro relativo *0* e exatidão 18,9659197224948.
    - *P9 avaliado em 1.5 resulta em 2,25, valor que pode ser representado em precisão simples. O erro relativo é 0 e a exatidão 9,3329598612474.*

O gráfico de erro absoluto foi utilizado como auxiliar na análise dos resultados de erro relativo e exatidão. Ele não foi diretamente analisado e, por isto, a sua análise não consta neste texto.

4.4. ANÁLISE DO RESULTADO DO RANKING

A utilização de um único valor (de acordo com Ossama (2000)), mesmo não sendo unanimidade, é uma prática comum dos benchmarks.

Neste trabalho, foi adotada a prática de gerar um único valor – esquema de pontuação –, onde se utiliza duas variáveis para gerar esta pontuação (tempo e exatidão).

A fórmula utilizada para chegar a esta pontuação é:

$$\frac{1}{t\varepsilon^2}$$

onde ε é o erro relativo. Esta fórmula dá mais valor ao erro relativo do que ao tempo. Mesmo assim, para um usuário que deseja exatidão de apenas dois dígitos após a vírgula os resultados podem não ter significado. Considera-se que o os métodos com precisão dupla obtiveram mais pontos devido ao seu menor erro relativo. Estes métodos tendem a levar mais tempo para avaliar o polinômio. O usuário tem como resposta que os métodos com precisão dupla são melhores. Isto não é verdade, uma vez que todos os métodos obtiveram a exatidão que o usuário

necessita, cabendo a ele, então, escolher o método que precisa de menos tempo para fazer as avaliações.

Estas análises mais aprofundadas do comportamento geral dos métodos podem ser obtidas através da análise dos gráficos. Esta análise requer mais tempo e mais conhecimento por parte do usuário, tanto da sua necessidade quanto de processamento numérico com ponto flutuante.

Abstraindo estas limitações, este ranking foi concebido para que seja possível resumir o desempenho do método durante as avaliações e detectar os limites de cada método. Foram encontrados diversos cenários para os rankings.

Caso Número	Ranking	Polinômios
1	1. Horner precisão simples 2. Potência precisão simples 3. Horner precisão dupla 4. Potência precisão dupla 5. Produto de raízes precisão simples 6. Produto de raízes precisão dupla	• *p1* • *p4* • *p6*
2	1. Horner precisão simples 2. Horner precisão dupla 3. Potência precisão simples 4. Potência precisão dupla 5. Produto de raízes precisão simples 6. Produto de raízes precisão dupla	• *p2*
3	1. Produto de raízes precisão simples 2. Horner precisão simples 3. Potência precisão simples 4. Produto de raízes precisão dupla 5. Horner precisão dupla 6. Potência precisão dupla	• *p3* • *p7* • *p8*
4	1. Potência precisão simples 2. Horner precisão simples 3. Potência precisão dupla 4. Horner precisão dupla 5. Produto de raízes precisão dupla	• *p5*

	6. Produto de raízes precisão simples	
5	1. Produto de raízes precisão dupla 2. Horner precisão dupla 3. Potência precisão dupla 4. Produto de raízes precisão simples 5. Potência precisão simples 6. Horner precisão simples	• *p9*

Tabela 4.4.1: Casos obtidos no ranking

Como é possível notar na *tabela 4.4.1*, não há um padrão que se segue em nenhum dos dois grupos. Ocorre no grupo A um comportamento padrão que é o do método de produto das raízes ter a menor pontuação. Isto já foi discutido na análise do erro relativo, e é causado pela entrada de dados com erro de arredondamento.

Normalmente, no contexto de avaliação polinomial, sempre se imagina, como “melhor alternativa”, o método de Horner com precisão dupla. Com a análise do ranking, é possível ver que isto não é verdade, pois existem casos em que outros métodos são superiores ao Horner.

Nos casos 1, 3 e 4, temos as precisões simples com mais pontuação. Isto se deve ao fato de que a diferença entre o tempo para a avaliação a favor da precisão simples é maior do que a diferença entre os erros relativos a favor da precisão dupla, que não acontece nos casos 2 e 5. No caso 2, o método de Horner com ambas as precisões é melhor que os outros métodos e, entre eles, a precisão simples tem mais pontos. No caso 5, os métodos com precisão dupla têm mais pontuação.

O método de Horner também não é sempre o melhor entre os métodos. Nos casos 3, 4 e 5, houve algum método com pontuação maior do que Horner.

O caso 5 é especial devido ao fato de que, em todos os outros, a pontuação segue um comportamento: se um método é melhor que outro em uma das precisões, vai ser na outra também. Se no caso 3 o produto das raízes é melhor do que o Horner, que é melhor do que o da potência em precisão simples, esta mesma ordem vai se repetir na precisão simples.

No caso 5, este comportamento não acontece. O método do produto de raízes é melhor do que Horner, que é melhor do que potência em precisão dupla,

mas potência tem mais pontuação do que Horner com precisão simples. Este caso prova que, se este comportamento for padrão, há algumas exceções que devem ser observadas e não se pode assumir este comportamento como uma verdade para todos os casos.

5. CONSIDERAÇÕES FINAIS

O erro de arredondamento é uma constante nos cálculos de ponto flutuante e detectá-lo é tão importante quanto difícil. Ele pode acontecer em qualquer cálculo: dos complexos, como cálculos de computadores de bordo de aviões de passageiros, até os mais simples, como cálculos de juros.

Como o arredondamento pode ter efeitos colaterais muito fortes, é extremamente necessário evitá-lo ou minimizá-lo ao máximo, seja utilizando técnicas (como as apresentadas em Goldberg (Goldberg, 1991)) ou métodos apropriados de cálculo.

No contexto de avaliação polinomial, este trabalho apresentou uma iniciativa pioneira de criação de benchmarks de avaliação polinomial com exatidão como o critério principal. Este benchmark, como todo o trabalho, ainda pode ser melhorado, mas já serviu para o aprendizado tanto na sua época de desenvolvimento teórico, quanto da implementação e, principalmente, utilização e análise de suas saídas.

O impacto do erro de arredondamento ficou muito claro quando, nos métodos do produto de raízes, foram informadas raízes com um erro de arredondamento embutido. Por menor que fosse o mesmo, os valores obtidos já destoavam dos outros métodos, muitas vezes fazendo com que os valores obtidos não tivessem nem 30% de um dígito significativo correto. Isto já fez com que o método fosse totalmente descartado para alguns fins. Exatamente nestes cenários, quando as raízes não puderam ser calculadas sem erro de arredondamento, era preferível utilizar um método com tipo de dados de precisão simples, que acabava por ser mais rápido e mais exato do que o método das raízes, mesmo que este utilizasse precisão dupla.

O número de operações aritméticas necessárias associadas a cada método foi outro ponto que teve sua importância ressaltada durante a execução deste trabalho. Nos dias de hoje, onde os recursos são relativamente baratos e até os computadores mais simples são extremamente poderosos comparados com os de alguns anos atrás, fica um pouco difícil entender que existe uma diferença importante entre um método que precisa de n adições e n multiplicações e um com $(2n - 1)$ multiplicações e n adições. Esta diferença se dá tanto em termos de tempo de execução quanto de erro de arredondamento. Atualmente, o padrão IEEE define que o cálculo seja exatamente arredondado, ou seja, deve ser calculado exatamente

e depois arredondado no final da operação, caso necessário (Goldberg, 1991). Quanto mais operações são realizadas, obviamente mais tempo é consumido, e mais arredondamentos podem acontecer.

A representação gráfica de resultados não era o foco inicial do trabalho, mas ela teve uma importância destacada no momento da análise dos resultados. Os gráficos apresentavam os resultados de maneira simples e flexível em um pequeno espaço e permitindo um entendimento fácil tanto do comportamento geral dos métodos de avaliação quanto do comportamento em algum ponto específico. A mesma informação que foi demonstrada em 4 gráficos, quando exportada para o Word, durante a fase de testes, foi apresentada em mais de 200 páginas, de maneira muito menos intuitiva. Isto mostrou a importância de um modo efetivo de apresentação dos resultados, pois todo o trabalho poderia ter se perdido, uma vez que a análise dos resultados fosse complexa e demorada.

Por fim, este trabalho pode auxiliar tanto a iniciantes na área de computação numérica, para que estes entendam onde, quando e por que o arredondamento acontece, quanto a pessoas já experientes na área, auxiliando a tomar decisões sobre a escolha de método de avaliação ou realizar estudos mais específicos, uma vez que o benchmark possibilita testes extensos e flexíveis, permitindo que o usuário teste os mais diversos cenários de um modo simples e fácil.

6. TRABALHOS FUTUROS

O trabalho obteve resultados extremamente satisfatórios, porém surgiram contratempos e não foi possível implementar tudo o que foi inicialmente planejado e nem do modo que era desejado.

Algumas coisas ficaram em aberto, tanto para serem analisadas quanto para serem melhoradas.

Em um trabalho futuro, ainda devem ser analisadas as seguintes questões relacionadas à saída do benchmark para os polinômios propostos neste artigo:

- Em relação ao tempo, descobrir o motivo para os picos ou quedas dos polinômios *p7*, *p8*, *p9*, quando todos os métodos têm este comportamento no mesmo ponto da avaliação.
- Em relação ao erro relativo, verificar a hipótese do método do produto de raízes ser o método mais exato para o grupo B. Caso seja, verificar o motivo

Em relação a melhorias:

- O sistema gráfico não suporta muitos pontos. O teste máximo dele foi com 4 gráficos, cada gráfico com 6 linhas, cada linha com 5000 pontos, totalizando 120000 pontos. Deste modo, o componente de gráficos ficou pesado e a apresentação dos pontos ruim. Seria muito interessante achar outro componente que comportasse mais pontos e tivesse uma melhor interação com o usuário.
- Aprender a utilizar o sistema gráfico com eficiência também é um pouco demorado. Um sistema com uma curva de aprendizado mais rápida seria melhor.
- Seria interessante uma funcionalidade de abrir e salvar arquivos melhor projetada.
- Possibilidade de importação de arquivos externos
- Logs mais eficientes para computação mais demorada
- Uso melhor da memória e utilização de técnicas para evitar o consumo de toda a memória disponível, possibilitando assim um cálculo com mais pontos.

7. REFERÊNCIAS

CAMPONOGARA, Eduardo. **Resolução de Equações Não-Lineares** [Online]. - 3/6/2009. - http://www.das.ufsc.br/~camponog/Disciplinas/DAS-5103/Slides/l5-nlneq-bisection.pdf.

DALCIDIO, Claudio M**. Sistema de Ponto Flutuante** [Online]. - 26/3/2009. - http://www.inf.pucrs.br/~dalcidio/disciplinas/metodos_computacionais/capitulo1.ppt

DENNING, Peter J**. The Choice Uncertainty Principle** [Artigo] // Communications of The ACM. - 2007. - 11 : Vol. 50.

DONGARRA, J**.** Frequently Asked Questions on the Linpack Benchmark and Top500 [Online]. - 12/12/2008. - http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html.

FREITAS, Carla Maria Dal Sasso et al. **Introdução à Visualização de Informações** [Artigo] // RITA. - 2001. - 2 : Vol. VIII. - Disponível em: http://www.inf.ufrgs.br/cg/publications/carla/Freitas-RITA2001.pdf, acessado em: 9/8/2008.

GANDER, Walter. **ETH Computer Science** [Online]. - 30/5/2006. - 4/9/2008. - http://www.inf.ethz.ch/news/focus/res_focus/april_2005/.

GOLDBERG, David. **What Every Computer Scientist Should Know About Floating-Point Arithmetic** [Artigo] // ACM Computing Surveys. - 1991. - 1 : Vol. 23. - http://perso.ens-lyon.fr/jean-michel.muller/goldberg.pdf.

IEEE. **IEEE** [Online] // IEEE Standard Fr Binary Floating-Point Arithmetic. - 12/11/2008. - http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=30711&isnumber=1316.

James Madison University [Online] // **Efficiency and Complexity**. - 12/4/2009. - https://users.cs.jmu.edu/bernstdh/Web/common/lectures/algorithms_efficiency.php.

JAULIN, L. [et al.] **Applied Interval Analysis**: With examples in parameter estimation robust control and robotics [Livro]. - Londres : Springer, 2001. - 1-85233-219-0.

KEIM, Daniel A. **Information Visualization and Visual Data Mining** [Artigo] // IEEE Transactions on Visualization and Computer Graphics. - 2002. - 1 : Vol. 7. - Disponível em: http://www.ailab.si/blaz/predavanja/ozp/gradivo/2002-Keim-Visualization%20in%20DM-IEEE%20Trans%20Vis.pdf acessado em: 1º/8/2008.

KRONSJÖ, Lydia L**. Algoeithms**: their complexity and efficiency [Livro]. - N.Y. : John Wiley & Sons Ltd., 1979. - 0471997528.

KULISH, U. W. ; MIRANKER, W. L**. The Arithmetic of the Digital Computer**: A new approach [Livro]. - 1986.

Maplesoft, a division of Waterloo Maple Inc. 2009 **Maple Features** [Online] // Math ¨Engineering Software. - 6/8/2008. - http://www.maplesoft.com/products/Maple/features/index.aspx.

Maxima SourceForge.net [Online] // Main Page - maxima. - 6/8/2008. - http://apps.sourceforge.net/mediawiki/maxima/index.php/History%20and%20other%20context/?PHPSESSID=32b2ec524613d84ad39755b0ac17cbda.

OSHEROVE, Roy **ISerializable - Roy Osherove's Blog** [Online] // Creating a better BackgroundWorker: CancelImmediately and other goodies. - 22/3/2009. - http://weblogs.asp.net/rosherove/pages/BackgroundWorkerEx.aspx.

OSSAMA, Cesar. **Benchmarks** [Online]. - 11/5/2000. - 21/8/2008. - http://www.inf.ufrgs.br/procpar/disc/cmp134/trabs/T1/001/benchmarks/Benchmarks_p2.htm.

RUMP, S. M**. How Reliable are Results of Computers?** [Seção do Livro] // Jahrbuch Überblicke Mathematik. - 1983.

Standard Performance Evaluation Corporation **SPEC Glossary** [Online]. - 2007. - 21/8/2008. - http://www.spec.org/spec/glossary/#benchmark.

The MathWorks, Inc**. MATLAB** [Online] // Visualizing Data. - 6/8/2008. - http://www.mathworks.com/products/matlab/description4.html.

University of Wisconsin **- Department of Chemical Engineering** Octave [Online] // About Octave. - 6/8/2006. - http://www.gnu.org/software/octave/index.html.

WARE, C**. Information Visualization:**Perception for Design [Livro]. - San Francisco : Morgan Kaufmann, 2000. - 1-55860-819-2.

WONG, Pak Chung **PNNL**: InfoViz [Online] // Visual Data Mining. - 1999. – 1º/8/2008. - http://infoviz.pnl.gov/pdf/visual_data_mining.pdf.