UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

INTEGRE – UM SISTEMA ACOPLADO PARA O CÁLCULO DE INTEGRAIS COM ANÁLISE DE FUNÇÕES NUMÉRICAS

AMÉDES ELEUTÉRIO DA COSTA JÚNIOR

FORTALEZA – CE
OUTUBRO – 1998

AMÉDES ELEUTÉRIO DA COSTA JÚNIOR

INTEGRE – UM SISTEMA ACOPLADO PARA O CÁLCULO DE INTEGRAIS COM ANÁLISE DE FUNÇÕES NUMÉRICAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre.
Curso de Pós-Graduação em Ciência da Computação, Centro de Ciências, Universidade Federal do Ceará.
Orientador: Mauro Cavalcante Pequeno

FORTALEZA
1998

Aos meus Pais

AGRADECIMENTOS

Pela valiosa colaboração Recebida dos professores Mauro Cavalcante Pequeno e Bitú, através de sugestões em torno da organização dessa dissertação e pela amizade que nasceu na elaboração deste trabalho. A eles, manifesto o meu profundo agradecimento.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

RESUMO

O INTEGRE é um sistema acoplado que trabalha simultaneamente com computação gráfica, numérica e simbólica e foi desenvolvido para ser uma ferramenta inteligente para auxiliar engenheiros com o cálculo de integrais complexas. O sistema trabalha como um especialista, decidindo na escolha do(s) algoritmo(s) para uma dada função integrando, sendo esta escolha baseada nas informações coletadas no processo de análise dessa função. Uma outra característica importante do sistema é a capacidade de auto-refinamento através do uso de técnicas de aprendizado automático.

ABSTRACT

INTEGRE is coupled system that works simultaneously with graphic, numeric and symbolic computation and it was developed to be an intelligent tool to aid engineers with the calculus of complex integral. The system works as an expert, deciding in the choice of the algorithm for a given integrating function, being this choice based on the information collected in the process of analysis of that function. Another important feature of the system is the refinement capacity through the use of techniques of automatic learning.

CAPÍTULO I

Introdução

Neste capítulo será apresentado um pequeno histórico da Análise Numérica onde serão abordados os pontos que motivaram a construção do sistema INTEGRE.

1.1 Visão Geral

A Análise Numérica teve um desenvolvimento acentuado durante os séculos XVIII e XIX no sentido de mecanizar a resolução de problemas numéricos, principalmente no que diz respeito a problemas que envolvessem integrais. Com o desenvolvimento dos métodos numéricos e principalmente com o advento do computador, pôde-se então pensar em automatizá-los, surgindo o que se denominou *software matemático* [01].

O software matemático apenas na década de 70, criou as metodologias de desenvolvimento, manutenção, documentação e distribuição de software numérico de alta qualidade. Como produto surgiram o EISPACK [02], FUNPACK [02], LINPACK [02], ELLPACK [03] e outros pacotes, além das bibliotecas de rotinas numéricas como a NAG [02], IMSL [04] e PORT [02]. Excetuando o ELLPACK, que já foi concebido com uma interface com o usuário, os outros softwares foram desenvolvidos com o objetivo de serem chamados por programas escritos em FORTRAN ou ALGOL, deixando, portanto, o usuário com uma tarefa de programar suas aplicações de utilização dos pacotes.

Na virada da década 70, ficou patente que a tarefa de programar as aplicações pelos usuários era onerosa e, muitas vezes, penosa, desestimulando e restringindo o uso dos pacotes a uma classe de usuários mais ou menos especializada. Alguns pesquisadores chegaram a levantar a questão sobre se as preocupações com a produção de software numérico prejudicaram o atendimento às necessidades de comunicação com o usuário [05]. Surgiu, então, a idéia de facilitar a utilização do software numérico pelas pessoas que tinham problemas para resolver, aliviando-as da tarefa de programar a solução desses problemas.

O ELLPACK foi pioneiro em criar uma linguagem de especificação de problemas bastante flexível, permitindo o usuário descrever um problema numa linguagem parecida com a utilizada em ciência e tecnologia. A linguagem é muito simples, composta de comandos de

declaração e comandos executáveis. A descrição é transformada em um programa FORTRAN convencional que, em seguida, é compilado, ligado e executado.

A linguagem de descrição de problemas no ELLPACK é simples e flexível porque o pacote é restrito a resolver apenas uma classe de problemas, que é a resolução de equações parciais de tipo elíptico. Quando se tenta transportar tal idéia para uma biblioteca como a IMSL, as coisas se complicam, tal como acontece com o PROTRAN, que acompanha o IMSL (em [06] contém a descrição da linguagem e aplicações). Apesar do PROTRAN ser um sistema muito bem concebido, a necessidade de especificar detalhes inerentes a solução de problemas numéricos exige recursos na linguagem que tornam a biblioteca complexa. Como no ELLPACK, os problemas especificados em PROTRAN são convertidos em programas na linguagem FORTRAN, que então são compilados, ligados e executados.

Em vários problemas numéricos, como no caso da integração numérica, equações diferenciais, etc., é exigido do usuário a especificação da função em questão, que deverá ser ligada à rotina de cálculo. Em se tratando de integração numérica, por exemplo, hoje em dia se dispõe de uma quantidade razoável de pacotes possuidores de rotinas que podem solucionar problemas variados. Basicamente, existem duas classes de problemas de resolução de integrais: os que necessitam da avaliação da função integrando e os que não necessitam. Estes últimos resolvem a integral através do fornecimento de uma tabela contendo pontos conhecidos da função integrando.

No primeiro caso, é necessário que o usuário forneça a função integrando através de uma sub-rotina, compile-a e, em seguida, ligue-a à rotina que implementa o método de cálculo. Tal procedimento cria um programa de resolução para aquele problema específico.

Há vários inconvenientes nesse processo. Entre eles pode-se citar:

1. Em primeiro lugar é requerido que o usuário tenha conhecimento da linguagem de programação na qual foi implementada a rotina de cálculo para poder construir a rotina que implementa a função integrando, e para isso, também é requerido conhecimento das peculiaridades das rotinas de cálculo.

2. Outro inconveniente é a necessidade de se ligar o programa de implementação da função, que foi construído pelo usuário, à rotina de cálculo. Isso significa que se o

programa for fornecido na forma de objeto, o usuário deverá utilizar o mesmo compilador ou um compilador compatível com o que foi utilizado pelo fornecedor ao compilar a rotina.

3. Em contraste com a situação anterior, pode ser necessário o fornecimento da rotina em código fonte para ser compilada junto com a função, o que nem sempre é possível, pois não é muito usual o fornecimento de código fonte de rotinas comerciais.

4. Um outro transtorno é que se deve repetir o processo utilizado para a obtenção do programa executável a cada vez que se necessite do valor da aproximação da integral de uma outra função.

Esses e outros inconvenientes fizeram com que se buscassem maneiras de se sanar esses problemas através da automatização desses processos.

Numa das linhas pesquisadas procurou-se uma forma que possibilitasse ao usuário definir a função utilizando uma linguagem próxima da matemática usual. O sistema, então, se encarregaria de interpretar e gerar o código fonte do programa função numa linguagem computacional. Essa foi a maneira adotada também por alguns pesquisadores que estudaram o assunto como BAILEY [07]. Embora seja um trabalho relevante, que constitua um avanço no sentido de aliviar o usuário da carga de programação, resolvendo o primeiro dos inconvenientes anteriormente citados, mesmo assim ainda persistem os demais.

Recentemente, surgiram vários programas que visavam solucionar os problemas descritos acima. Programas como o MATHEMATICA [08], MATHCAD [09] e o MATLAB [10] possuem uma interface com o usuário que sofre um contínuo processo de aprimoramento. A linguagem de especificação dos problemas é feita através de um conjunto de comandos inerentes a estes softwares. Além disso, o MATHEMATICA e o MATHCAD permitem que o usuário possa interagir com a sua interface para "construir" a função que será integrada. A principal vedete desses softwares está no poder de manipulação simbólica que possuem, ocultando em parte, seu poder de manipulação numérica.

Apesar do grande poder desses softwares, existem problemas que não podem ser solucionados pela matemática computacional tradicional, mas com um abordagem inteligente e o emprego de algoritmos de uma forma adequada, consegue-se fazê-lo. Visando a solução desses problemas estabeleceu-se a definição, a motivação, a arquitetura e os fatores mais importantes do que se chamou de *Sistemas Acoplados* [11]. Sistemas Acoplados são sistemas

compostos necessariamente de uma base de conhecimento (computação simbólica) e de algoritmos numéricos, além de estar munido do conhecimento dos processos numéricos envolvidos e as razões sobre as aplicações e/ou uma interpretação dos resultados obtidos por estes processos [12].

Baseado na filosofia acima, o sistema INTEGRE foi construído na tentativa de resolver uma grande gama de integrais, utilizando o melhor algoritmo para uma dada função na expectativa de otimizar o uso do processador. Entre as principais características do sistema pode-se citar:

1. O uso de interface inteligente ajudará o operador na especificação dos problemas, minorando o esforço do usuário e, sobretudo, diminuindo as possibilidades de erros e inconsistência de dados.

2. Possui um método automático de escolha que, de acordo com a especificação da função integrando, o sistema escolhe, de forma automática, o algoritmo de cálculo específico, eliminado do usuário o conhecimento dos algoritmos de integração.

3. Exibe o gráfico da função em questão com vários níveis de ZOOM que possibilita uma análise interativa por parte do usuário.

4. Dispõe de várias rotinas de impressão podendo os dados de processamento serem impressos para análise.

5. Possui um banco com vários algoritmos que podem ser ativados para resolver a integral proposta.

6. Possui técnicas de aprendizado automático que contribuirá para o refinamento do sistema.

Para elaboração do INTEGRE, vários outros softwares foram estudados entre aqueles que aqui já foram citados tal como o EXBITAN [13], FFTEX [14,15] e o SIPREX [16,17]. O INTEGRE deverá contribuir para o desenvolvimento de uma metodologia formal para elaboração de sistemas acoplados, ou mais especificamente na construção de ambientes inteligentes.

1.2 Organização da Dissertação

Este trabalho foi organizado levando-se em conta a evolução da pesquisa desenvolvida que originou o sistema INTEGRE. Assim os capítulos foram dispostos da seguinte maneira:

- O Capítulo 2 expõe de maneira sucinta os tópicos matemáticos relativos aos algoritmos de integração que compõem o sistema INTEGRE.

- O Capítulo 3 descreve os algoritmos de integração numérica referentes a biblioteca NUMERICAL RECIPES [18, 19] e ao pacote QUADPACK [20].

- O Capítulo 4 apresenta o sistema INTEGRE, descrevendo suas principais características: interface com o usuário, Análises Léxica e Sintática, Análise Matemática, Método de Escolha, Método de Aprendizado Automático, Arquivo de Pesos, Arquivo de Ocorrências, etc..

- O Capítulo 5 exibe uma comparação do sistema INTEGRE com alguns softwares já consagrados como o MATHEMATICA, MATHCAD e o MATLAB.

- O Capítulo 6, que apresenta as conclusões, resume as principais contribuições deste trabalho à ciência, resultados sobre a execução paralela dos algoritmos e apresenta propostas para futuros trabalhos que darão prosseguimento à pesquisa realizada.

CAPÍTULO II

Noções sobre Quadratura Numérica

Neste capítulo será apresentado, de forma resumida, um pequeno embasamento matemático necessário a compreensão dos algoritmos de integração numérica que serão descritos nos capítulos posteriores.

2.1 Introdução

A Integração Numérica, ou cálculo numérico de integrais, é o estudo de como obter o valor numérico de uma dada integral. Esse cálculo é um dos problemas mais antigos em matemática e suas origens remontam aos tempos de Arquimedes [20]. Acredita-se que Arquimedes tenha descoberto limitantes superiores e inferiores de π inscrevendo e circunscrevendo polígonos regulares em um círculo. Tal processo foi denominado de quadratura Grega [21] e representa um bom exemplo de como os antigos trabalhavam com Integração Numérica.

O principal objetivo da Integração Numérica é calcular o valor da integral definida de uma dada função. Este valor é dado, no caso unidimensional, pela área abaixo de uma curva. Muitos aspectos da teoria da aproximação são diretamente aplicáveis em integração numérica e resultados das mais diversas áreas (polinômios ortogonais, séries de Fourier e teoria dos números) têm exercido significativa influência no cálculo de integrais [20].

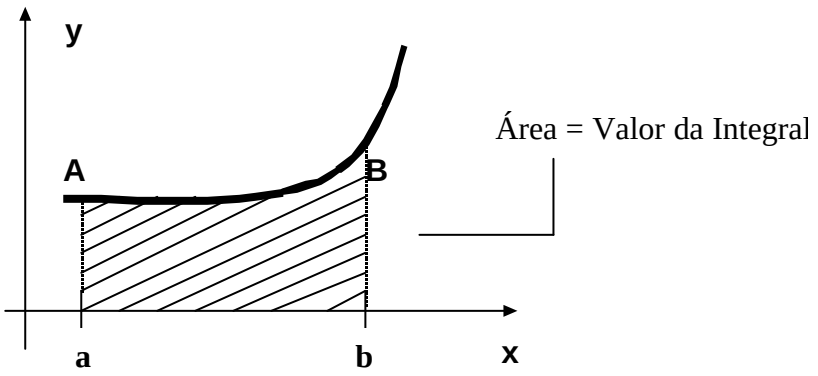


Figura 2.1: Valor da Integral dado pela área abaixo da curva

Apesar do nosso estudo se deter somente a integração numérica, existem outros métodos para o cálculo da área mostrada pela Figura (2.1). Estes são agrupados em quatro classes [22]: Analíticos, Mecânicos, Gráficos e Numéricos ou Algorítmico.

O processo analítico consiste num conjunto de regras que podem ser utilizadas para a solução da integral e, em geral, são estudadas nos cursos de cálculo. No entanto, a maioria das integrais presentes nos trabalhos científicos são incapazes de serem avaliadas por este conjunto de regras [20]. Pode-se citar, o exemplo da função $f(x) = e^{-x^2}$, cuja primitiva não pode ser expressa por uma combinação finita de outras funções algébricas, logarítmicas ou exponenciais. Mesmo quando uma expressão analítica para solução da integral pode ser obtida, em geral são necessárias quantidades excessivas de manipulações simbólicas o que pode levar a erros. Muitas vezes a integração analítica de funções simples pode envolver a avaliação de "funções complexas".

Assim, as regras de integração analítica são usadas apenas no cálculo de integrais de algumas funções elementares: integrais cuja expressão envolve um número finito de combinações de funções algébricas, trigonométricas, logarítmicas ou exponenciais. Para funções avançadas, como a função Erro, Gama, Beta, Elípticas, Hipergeométricas etc., os métodos analíticos não são amplamente utilizados [22]. Resta-nos então, naturalmente, pesquisar métodos que aproximem integrais diretamente para integrandos gerais ao invés de fazer manipulações algébricas específicas do problema.

Os métodos gráfico e mecânico são de uso bastante restrito e não serão abordados neste trabalho. Mais detalhes sobre estes métodos podem ser encontrados em [22].

Porém, se com os métodos analítico, mecânico ou gráfico não se consegue calcular a integral, pode-se então recorrer ao método algorítmico ou numérico. Em algumas situações só é possível o uso do método numérico. Por exemplo, se não é conhecida a expressão analítica de *f*, não é possível utilizar outro método que não o numérico.

Mesmo sendo uma ferramenta poderosa, a Integração Numérica não pode ser utilizada ao acaso, pois qualquer aproximação para uma integral envolve erros e, a menos que algum julgamento desses erros seja disponível, significado algum pode ser atribuído a esta aproximação.

2.2 Por que Integração Numérica ?

O problema de integração numérica a uma variável consiste em resolver numericamente

$$\int_a^b f(x)dx \tag{2.1}$$

Do Teorema Fundamental do Cálculo Integral [23], pode-se concluir que se f(x) é uma função contínua em [a, b], então esta função possui primitiva neste intervalo, ou seja, existe F(x) tal que F'(x) = f(x), sendo o valor dado por :

$$\int_a^b f(x)dx = F(b) - F(a) \tag{2.2}$$

No entanto, pode não ser fácil, ou mesmo possível, expressar esta primitiva por meio de combinações de funções elementares. Para estes casos intratáveis, e, para muitos problemas gerais de integração nos quais apenas possuímos valores da função f(x) em forma de tabelas, algum outro método se torna necessário para o cálculo de (2.2).

2.3 Técnicas Básicas de Integração Numérica

Para calcular o valor aproximado da integral definida utiliza-se uma combinação linear de valores da função f em certos pontos x_i, $a \leq x_i \leq b$, chamados de nós, e de certos valores ω_i, i = 1, 2, 3, ..., n+1 que constituem os pesos, ou seja :

$$\int_a^b f(x)dx \approx \omega_1 f(x_1) + \omega_2 f(x_2) + ... + \omega_n f(x_n) + \omega_{n+1} f(x_{n+1}) \tag{2.3}$$

A expressão (2.3) corresponde a aproximar a integral pela soma da áreas de retângulos de altura $f(x_i)$ e largura ω_i. De acordo com os valores dos pesos e com a escolha de nós, temos no lado direito de (2.3) o que se costuma chamar de *regras de integração*, *quadraturas* ou *soma de quadratura*. Estes termos serão utilizados, indistintamente, para indicar, em geral, qualquer fórmula que forneça uma aproximação da integral. Com a escolha

dos nós x_i e dos pesos ω_i em (2.3), são estabelecidos 2n+2 graus de liberdade na construção da regra de integração.

A determinação dos pesos e dos nós é feita de acordo com várias filosofias que se agrupam em duas grandes categorias: fixas e adaptativas.

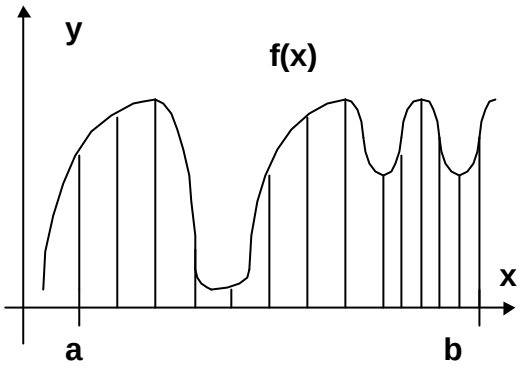


Figura 2.2: Quadratura Adaptativa

Um integrador *fixo* ou *não-adaptativo* utiliza abcissas fixas, de tal modo que o algoritmo se comporta da mesma forma para cada problema e apenas o número de pontos nos quais a função será avaliada é escolhido, dependendo da complexidade do problema.

O procedimento para calcular a seqüência de aproximações é chamado de *adaptativo* se a posição dos pontos da n-ésima iteração depende da informação coletada nas iterações 1,...,n-1. Geralmente isto é feito por uma sucessão de partições do intervalo original de tal forma que mais pontos são tomados na vizinhança dos "pontos difíceis" da função, ocasionando aí uma alta densidade de nós de quadratura. A figura (2.2) mostra uma densidade maior de pontos perto de *b*, onde a função *f* varia com menos suavidade. O principal motivo para o uso da quadratura adaptativa é reduzir o número de avaliações da função *f* necessárias para obter a aproximação da integral, tendo em vista que, em rotinas de quadratura, a eficiência do algoritmo é medida pelo número de avaliações de *f*, o que reflete no custo final do processamento.

Tanto na filosofia fixa como na adaptativa são empregados vários tipos de regras. Os tipos mais importantes são:

a) *Fórmula de Newton-Cötes*: as fórmulas de Newton-Cötes são as de aplicação mais simples e a sua idéia básica consiste em interpolar *f* por polinômios de grau 1,2 ou *m* no intervalo de integração (regras simples) ou em subdivisões $[x_{i+1}, x_i]$ desse intervalo (regras

compostas). Nas regras compostas, $x_o = a$ e x_{n+1}=b, sendo os pontos x_i determinados por $x_i = x_o + ih$, onde $h = x_{i+1} - x_i$ e i = 1,...,n+1, que são pontos igualmente espaçados. Os pesos ω_i são obtidos a partir do grau do polinômio que interpola f nos pontos $(x_i, f(x_i))$, de modo que a regra obtida é exata para qualquer polinômio de grau menor do que ou igual a *n* [24].

b) *Fórmula de Gauss*: determina-se os pontos x_i e os pesos ω_i, de modo que a regra seja exata para qualquer polinômio de grau até $p = 2n + 1$ [24], onde *n* é o número de pontos a serem tomados no intervalo de [a, b]. Os pontos x_i assim obtidos não são igualmente espaçados.

c) *Fórmulas baseadas em métodos de extrapolação ao limite*: as fórmulas newtonianas possuem convergência lenta; uma forma de aumentar a velocidade de convergência é aplicar os métodos de extrapolação ao limite. A idéia básica desses métodos é obter uma melhor aproximação da integral a partir de duas outras aproximações obtidas pelas fórmulas newtonianas. Um ótimo exemplo é a integração de Romberg que utiliza a extrapolação de Richardson como método de extrapolação [24].

d) *Fórmulas especiais*: são obtidas usando o polinômios de Chebyshev [24], Laguerre [24], Hermite [24], as regras de Kronrod [25], Chenshaw Curtis [20, 21] etc..

Cada integrador possui um ou mais *Critérios de Avaliação* que incluem componentes estratégicos para decidir como continuar o procedimento de integração. Uma estratégia é denominada *localmente adaptativa* quando impõe requisitos de precisão independentes para todos os subintervalos. Enquanto a precisão sobre um particular subintervalo não é alcançada ele é considerado *pendente.* Num algoritmo *globalmente adaptativo,* todos os subintervalos permanecem pendentes até que a soma das estimativas do erro se torne menor que a tolerância do erro [20].

Uma soma de quadratura de *n* pontos denominada, Q_n, é dita de grau (*polinômial, algébrico de precisão*) *d* se ela é exata para todas as funções que são polinômios de grau $\leq$ (menor do que ou igual a) *d* e não é exata para todos os polinômios de grau *d+1*. Supõe-se sempre que $d \geq 0$.

As seguintes propriedades de uma soma de quadratura (além de ter grau de precisão *d*) são particularmente desejáveis de um ponto de vista prático [20]:

- Todas as abcissas estão dentro do intervalo de integração;
- Todos os pesos são positivos.

Se a segunda propriedade é satisfeita será usado o termo *regra de quadratura positiva* ou *regra de quadratura com pesos positivos* [20].

a= -1, b=1, w(x) = 1	**Polinômios de Legendre $P_n(x)$** $P_n(x) = \frac{2n-1}{n} xP_{n-1}(x) - \frac{n-1}{n} P_{n-2}(x)$ Quadratura de Gauss-Legendre.
a=-1, b=-1, w(x)= $\frac{1}{\sqrt{1-x^2}}$	**Polinômios de Chebyshev de primeiro tipo $T_n(x)$** $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$ Quadratura de Gauss-Chebyshev de 1^o tipo.
a=-1, b=1, w(x)= $\sqrt{1-x^2}$	**Polinômios de Chebyshev de segundo tipo $U_n(x)$**. $U_{n+1}(x)=2xU_n(x)-U_{n-1}(x)$; $U_0=1$, $U_1=2x$ Quadratura de Gauss-Chebyshev de 2^o tipo.
a=-1, b=1, w(x)=$(1-x)^\alpha(1+x)^\beta$ $\alpha,\beta > -1$	**Polinômios de Jacobi $P_n^{(a,b)}(x)$**. $2(n+1)(n+\alpha+\beta+1)(2n+\alpha+\beta)P_n^{(\alpha,\beta)}(x)=$ $(2n+\alpha+\beta+1)[(\alpha^2-\beta^2)+(2n+\alpha+\beta+2)(2n+\alpha+\beta) \quad x]P_n^{(\alpha,\beta)}(x)-$ $2(n+\alpha)(n+\beta)(2n+\alpha+\beta+2)P_{n-1}^{(\alpha,\beta)}(x)$ $P_0^{(\alpha,\beta)}=1$, $P_1^{(\alpha,\beta)}=(1+ ½ (\alpha+\beta))x+½(\alpha-\beta)$ Quadratura de Gauss - Jacobi.
a=0, b=+∞, w(x) = exp(-x)	**Polinômios de Laguerre** $L_n(x) = (2n-x-1)L_{n-1}(x) - (n-1)^2 L_{n-2}(x)$ Quadratura de Gauss – Laguerre.
a=0, b=+∞, w(x) = x^α exp(-x); $\alpha > -1$	**Polinômios generalizados de Laguerre** $L_n^{(\alpha)}(x)$. $(n+1)L_{n+1}^{(\alpha)}(x)=[(2n+\alpha+1)-x]L_n^{(\alpha)}-(n-\alpha)L_{n-1}^{(\alpha)}(x)$ $L_o^{(\alpha)}=1$, $L_1^{(\alpha)}=1+\alpha-x$ Quadratura de Gauss – Laguerre generalizada.
A=-∞, b=∞, w(x) = exp($-x^2$)	**Polinômios de Hermite**. $H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x)$ Quadratura de Gauss – Hermite.

Tabela 2.1: Classificação das fórmulas especiais

Para um dado conjunto de nós reais, distintos, x_i, i =1,2,...,n geralmente escolhidos como pertencentes ao intervalo de integração, os pesos ω_i, i =1,2,...,n podem ser determinados de tal forma que Q_n seja de grau n-1. Uma soma de quadratura construída de acordo com este princípio é dita ser *interpolatória* [20].

Os Integradores que não exigem do usuário informações explícitas sobre a função integrando serão chamados de Integradores de *propósito geral*. Os integradores para *propósitos especiais* são fornecidos para integrais da forma:

$$I = \int_a^b w(x) f(x) dx , \tag{2.4}$$

onde w(x) é alguma função-peso especificada pela tabela (2.1).

Nos integradores de propósito especiais, o tipo de função-peso e os particulares valores dos parâmetros têm que ser fornecidos explicitamente pelo usuário (via parâmetros das rotinas correspondentes).

A tabela (2.1) exibe a função-peso de algumas integrais especiais. Detalhes sobre sobres estas integrais podem ser encontrados em [20, 21].

No próximo capítulo, serão abordados todos os algoritmos de Integração que fazem parte do sistema INTEGRE.

CAPÍTULO III

Algoritmos de Integração Numérica

Neste capítulo serão descritos os algoritmos de integração numérica utilizados pelo sistema INTEGRE para o cálculo da aproximação da integral.

3.1 Introdução

Na década de 1960, os fabricantes de computadores forneciam bibliotecas de métodos numéricos para os seus equipamentos. Nessas bibliotecas podiam-se encontrar dois defeitos básicos: não tinham confiabilidade e eram difíceis de usar. Esses defeitos faziam com que as pessoas usassem os programas com desconfiança ou, até mesmo, deixavam de usá-los. As causas desses defeitos eram a falta de motivação para produzir programas fáceis de usar, pois, na época, o programador se vangloriava de escrever programas cheios de truques obscuros, como os famigerados "GO TO".

Somente no começo da década de 1970, chegou-se a conclusão de que a elaboração de uma boa biblioteca requer muito esforço técnico, organizacional e recursos financeiros. A partir dessa época, pacotes e bibliotecas de sucesso foram escritas. Porém existem dificuldades em manter uma biblioteca sempre atualizada, fácil de usar e confiável.

Uma biblioteca difere de uma coleção de programas arbitrários nos aspectos abaixo [02]:

- Ela procura resolver um amplo espectro de problemas matemáticos que ocorrem com maior freqüência em ciência e tecnologia.

- Deve possuir um bom acervo literário do assunto tratado (referências) para que o usuário possa se aprofundar no assunto.

- Ela deve resolver a maioria das necessidades dos usuários.

- Os algoritmos selecionados para compor uma biblioteca devem ser capazes de resolver satisfatoriamente uma gama de aplicações. Tais algoritmos devem ser rigorosamente escolhidos de acordo com os seguintes critérios: consagrados na literatura, possuírem boa documentação, serem adaptáveis segundo o conceito de [26] que é requisito fundamental à sua transportabilidade.

- Os programas que compõem uma biblioteca devem ser organizados em estruturas hierárquicas. No topo dessa hierarquia os programas que resolvem os problemas. No segundo nível estão as rotinas primárias que são encarregadas de implementar os algoritmos dos métodos numéricos e finalmente vêm os módulos básicos que implementam operações básicas necessárias para a implementação das rotinas primárias, como as operações com vetores necessárias para as rotinas de álgebra linear.

Se de um lado, biblioteca é uma coleção sistematizada de programas para resolução de diversas classes de problemas, do outro, pacotes são coleções de programas destinados a resolver problemas de uma determinada área, por exemplo, sistema de equações lineares. De uma forma geral, pode-se dizer que uma biblioteca é uma coleção de pacotes.

Neste capítulo serão analisadas as rotinas de integração numérica provenientes da biblioteca *NUMERICAL RECIPES* e as rotinas do pacote *QUADPACK*.

O NUMERICAL RECIPES é uma biblioteca que pode ser comparada a um livro de receitas. O usuário estuda o problema e de acordo com a solução que será dada a este problema ele pode optar por uma das diversas rotinas que esta biblioteca dispõe. O NUMERICAL RECIPES possui algoritmos que implementam soluções nas mais diversas áreas da análise numérica, entre elas pode-se citar: Equações Lineares, Interpolação e Extrapolação, Integração Numérica, Cálculo de Raízes de Funções Não-Lineares, Autovalores, Equações Diferencias Ordinárias, Avaliação de Funções e Análise de Funções Especiais, Números Randômicos e os Métodos de Monte-Carlo, Classificação de Dados, Otimização, Métodos da Transformada de Fourier, etc..

Uma descrição completa desses e de outros tópicos pode ser encontrada em [18]. Neste trabalho, apenas o tópico de integração numérica será estudado. Apesar do NUMERICAL RECIPES ser uma biblioteca, para efeito do nosso estudo, ele será considerado um pacote, pois apenas as rotinas de integração numérica serão analisadas.

O QUADPACK é uma coleção de rotinas em FORTRAN para cálculo de integrais. As rotinas são explicitamente destinadas ao cálculo de integrais unidimensionais (em uma variável), muito embora alguns problemas em dimensões mais altas possam ser tratados através de uma conveniente combinação das mesmas [20]. As rotinas podem ser manipuladas para diferentes classes de problemas, podendo vários programas externos chamá-las, permitindo que usuários com necessidades diferentes acessem o pacote.

Sendo assim, os programas que utilizam os algoritmos de integração numérica que compõem a biblioteca NUMERICAL RECIPES receberão as iniciais **NR** e aqueles que utilizam o pacote QUADPACK receberão as iniciais **QUADP**.

A tabela (3.1) mostra a classificação dos programas por classe de problemas.

Classe	**Programas**
Algoritmos Gerais de integração.	NR1, NR2, NR3, NR10, N11, QUADP1 e QUADP2.
Algoritmos para integrais com singularidade no extremo de intervalo de integração.	NR4, NR5, NR6, NR7, NR9 e QUADP5.
Algoritmos para integrais com intervalo com algum limite de integração no infinito.	NR5, NR8 e QUADP6
Algoritmo de Gauss-Jacobi	NR14
Algoritmo de Gauss-Hermite	NR13
Algoritmo de Gauss-Laguerre	NR12
Algoritmos para funções oscilatórias	QUADP8, QUADP9
Algoritmo para funções que apresentam singularidade(s) no interior do intervalo de integração.	QUADP7
Algoritmo para funções Logarítmicas – Algébricas.	QUADP3
Algoritmo para funções da forma $w(x) = f(x)/(x-c)$.	QUADP4

Tabela 3.1: Algoritmos de integração por classe de problemas

3.2 Os programas NR

Nas próximas seções serão analisadas o conjunto de programas provenientes da biblioteca NUMERICAL RECIPES.

3.2.1 O Programa NR1

O programa NR1 utiliza a rotina TRAPZD [18] que calcula o valor da integral utilizando a regra dos trapézios repetida [24]. O parâmetro *FUNC* é a função que deve ser

integrada entre os limites *A* e *B*. O parâmetro *S* é utilizado para retornar o valor da integral para o programa principal.

A tabela (3.2) mostra a descrição dos parâmetros utilizados pela rotina TRAPZD.

Parâmetro	**Descrição**	
FUNC	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
S	Resultado da Integral no intervalo de [A, B]	Saída
N	Número da iteração da rotina.	Entrada

Tabela 3.2: Parâmetros da rotina TRAPZD

A rotina TRAPZD é chamada através de um novo procedimento, denominado QTRAP [18], que testa se uma determinada precisão *EPS* foi atingida toda vez que for efetuada a chamada da rotina TRAPZD.

A rotina QTRAP retorna, na variável *S*, o valor da integral de *FUNC*, no intervalo de *A* até *B*. O valor da variável *EPS* é a precisão desejada. A variável *JMAX* representa o número máximo de chamadas à rotina de integração TRAPZD.

Parâmetro	**Descrição**	
FUNC	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
S	Resultado da Integral no intervalo de [A, B]	Saída

Tabela 3.3: Parâmetros da rotina QTRAP

3.2.2 O programa NR2

O programa NR2 calcula a integral no intervalo fechado [A, B] utilizando a rotina QSIMP [18].

A rotina QSIMP, retorna na variável *S* o valor da integral de *FUNC* entre o limite inferior *A* e superior *B* e para isso utiliza a extrapolação de Richardson [24] para obter uma melhor aproximação da integral utilizando a rotina TRAPZD.

A tabela (3.4) mostra a descrição dos parâmetros de QSIMP.

Parâmetro	Descrição	
FUNC	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
S	Resultado da Integral no intervalo de [A, B] pela regra de Simpson.	Saída

Tabela 3.4: Descrição dos Parâmetros de QSIMP

3.2.3 O Programa NR3

O programa NR3 calcula a integral no intervalo fechado [A, B] utilizando o método de Romberg [17], que é implementado pela rotina QROMB [18].

Parâmetro	Descrição	
FUNC	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
SS	Resultado da Integral no intervalo de [A, B] pela integração de Romberg.	Saída

Tabela 3.5: Descrição dos Parâmetros de QROMB

Pela tabela (3.5), a variável *EPS* armazena a precisão desejada que é determinada pelo o erro de extrapolação utilizado. O parâmetro *JMAX* indica o número total de passos que deve ser tentado caso a precisão *EPS* ainda não seja atingida. A rotina QROMB chama a rotina TRAPZD que foi apresentada na seção (3.2.1).

3.2.4 O Programa NR4

O programa NR4 calcula a integral no intervalo [A, B] quando os extremos do intervalo não fazem parte da partição utilizando, para isso, utiliza a rotina MIDPNT [18]. Essa rotina utiliza a fórmula (3.1) para generalizar a regra dos trapézios, pois ela não exige que a função seja definida nos limites de integração.

Denotando por f_i o valor da função no ponto x_i, assim f_0, f_1,..., f_n serão, respectivamente, o valor da função *f* nos pontos x_1, x_2,..., x_n. Desta forma, pode-se escrever a regra MIDPNT pela fórmula [18]:

$$\int_{x_1}^{x_N} f(x)dx = h\left[f_{3/2} + f_{5/2} + f_{7/2} + ... + f_{N-3/2} + f_{N-1/2}\right] + O\left(\frac{1}{N^2}\right) \qquad (3.1)$$

A tabela (3.6) mostra a descrição dos parâmetros de MIDPNT.

Parâmetro	Descrição	
FUNC	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
S	Resultado da Integral no intervalo de [A, B] calculado por MIDPNT.	Saída
N	Valor que servirá para o sistema calcular a Quantidade de subintervalos no qual o intervalo original será dividido.	Entrada

Tabela 3.6: Descrição dos Parâmetros de MIDPNT

3.2.5 O programa NR5

O programa NR5 calcula a integral da função *f* quando os limites de integração são infinitos, utilizando, para isso a rotina MIDINF [18] que efetua uma mudança de variável para fazer o mapeamento do intervalo infinito de integração para outro finito.

Por exemplo, a identidade

$$\int_a^b f(x)dx = \int_{1/b}^{1/a} \frac{1}{t^2} f\left(\frac{1}{t}\right)dt, ab > 0 \qquad (3.2)$$

pode ser aplicada quando $b \rightarrow \infty$ e a positivo ou quando a $\rightarrow -\infty$ e b negativo, funciona para qualquer função que decresce rápido [18].

A rotina MIDINF implementa a mudança de variável apresentada pela expressão (3.2) de modo automático, tornando assim, esta mudança transparente ao usuário. A rotina MIDINF é uma versão modificada de MIDPNT, que permite b assumir o valor $+\infty$ e a ser $-\infty$.

A tabela (3.7) mostra a descrição dos parâmetros de MIDINF.

Parâmetro	Descrição	
FUNC	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
S	Resultado da Integral no intervalo de [A, B] calculado por MIDINF.	Saída
N	Valor que servirá para o sistema calcula a quantidade de subintervalos de intervalo original será dividido.	Entrada

Tabela 3.7: Descrição dos Parâmetros de MIDINF

3.2.6 Os Programas NR6 e NR7

Quando a função integrando possui singularidade no seu limite inferior ou superior de integração também é utilizada uma mudança de variável semelhante a apresentada na seção anterior. Se o integrando tiver a forma $(x - a)^{\gamma}$, $0 \leq \gamma < 1$ ele possui uma singularidade perto de $x = a$. Sendo a o limite inferior de integração, pode-se contornar este problema fazendo $x = t^{\gamma/(1-\gamma)} + a$ e aplicando o teorema da mudança de variável, para obter a identidade (3.3):

$$\int_a^b f(x)dx = \frac{1}{1-\gamma} \int_0^{(b-a)^{1-\gamma}} t^{\gamma/(1-\gamma)} f(t^{1/(1-\gamma)} + a)dt, (b > a) \qquad (3.3)$$

Se a singularidade está no limite superior, deve-se proceder da mesma forma, exceto pelo fato que a mudança de variável será $x = b - t^{\gamma/(1-\gamma)}$, obtendo assim a identidade:

$$\int_a^b f(x)dx = \frac{1}{1-\gamma} \int_0^{(b-a)^{1-\gamma}} t^{\gamma/(1-\gamma)} f(b - t^{1/(1-\gamma)})dt, (b > a) \quad (3.4)$$

As equações (3.3) e (3.4) são particularmente simples no caso de singularidades do inverso da raiz quadrada, que ocorre freqüentemente na prática. Para tal, basta substituir o valor de γ por ½ para encontrar as expressões:

$$\int_a^b f(x)dx = \int_0^{\sqrt{b-a}} 2tf(a + t^2)dt, (b > a) \quad (3.5)$$

$$\int_a^b f(x)dx = \int_0^{\sqrt{b-a}} 2tf(b - t^2)dt, (b > a) \quad (3.6)$$

O algoritmo MIDSQL [18], que é componente do programa NR6, funciona de forma análoga ao algoritmo MIDPNT, sendo que o primeiro permite uma singularidade do inverso da raiz quadrada no limite inferior.

A tabela (3.8) mostra a descrição dos parâmetros de MIDSQL.

Parâmetro	Descrição	
FUNK	Função a ser integrada. Esta função é definida pelo usuário	Entrada
AA	Limite inferior de integração.	Entrada
BB	Limite superior de integração.	Entrada
S	Resultado da Integral no intervalo de [A, B] calculado por MIDSQL.	Saída
N	Valor que servirá para o sistema calcula a quantidade de subintervalos de intervalo original será dividido.	Entrada

Tabela 3.8: Descrição dos Parâmetros de MIDSQL

De maneira análoga a rotina MIDSQL, a rotina MIDSQU [18], que é componente do programa NR7, calcula a integral quando existe uma singularidade do inverso da raiz quadrada no limite superior.

A tabela (3.9) mostra a descrição dos parâmetros de MIDSQU.

Parâmetro	Descrição	
FUNK	Função a ser integrada. Esta função é definida pelo usuário.	Entrada
AA	Limite inferior de integração.	Entrada
BB	Limite superior de integração.	Entrada
S	Resultado da Integral no intervalo de [A, B] calculado por MIDSQU.	Saída
N	Valor que servirá para o sistema calcula a Quantidade de subintervalos de intervalo original será dividido.	Entrada

Tabela 3.9: Descrição dos Parâmetros de MIDSQU

3.2.7 O programa NR8

O programa NR8 calcula a integral quando o limite superior de integração é infinito e a função integrando cai exponencialmente, para isso, ele utiliza a rotina MIDEXP [18].

A rotina MIDEXP efetua uma mudança de variável que mapeia e^{-x} dx em (±)dt (a mudança de sinal será efetuada para manter o limite superior da nova variável maior do que o inferior). Fazendo tal substituição, obtém-se:

$$t = e^{-x} \qquad \text{ou} \qquad x = -\log t \tag{3.7}$$

Finalmente tem-se :

$$\int_{x=a}^{x=\infty} f(x)dx = \int_{t=0}^{t=e^{-a}} f(-\log(t))\frac{dt}{t} \tag{3.8}$$

A tabela (3.10) mostra a descrição dos parâmetros de MIDEXP.

Parâmetro	Descrição	
FUNK	Função a ser integrada. Esta função é definida pelo usuário	Entrada
AA	Limite inferior de integração.	Entrada
BB	Limite superior de integração.	Entrada
S	Resultado da Integral no intervalo de [A, B] calculado por MIDEXP.	Saída
N	Valor que servirá para o sistema calcula a quantidade de subintervalos de intervalo original será dividido.	Entrada

Tabela 3.10: Descrição dos Parâmetros de MIDEXP

3.2.8 O programa NR9

O programa NR9 utiliza a rotina QROMO [18] que representa o algoritmo de ROMBERG para um intervalo aberto. O algoritmo QROMO retorna na variável *SS* o valor da integral da função *FUNC* no intervalo de *A* até *B*, usando qualquer uma das sub-rotinas de integração *CHOOSE* e o método de ROMBERG. Normalmente CHOOSE é uma fórmula de integração aberta, pois não fará a avaliação no intervalo de integração em seus pontos finais. As rotinas MIDPNT, MIDINF, MIDSQL, MIDSQU, são possíveis escolhas para CHOOSE.

A descrição dos parâmetros de QROMO é mostrada pela tabela (3.11):

Parâmetro	Descrição	
FUNC	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
SS	Resultado da Integral no intervalo de [A, B] calculado por QROMO.	Saída
CHOOSE	Qualquer uma das rotinas MIDPNT, MIDINF, MIDSQL e MIDSQU.	Entrada

Tabela 3.11: Descrição dos Parâmetros de QROMO

Basicamente não existe diferença entre QROMB e QROMO, contudo a última rotina é excelente para o cálculo de integrais impróprias.

3.2.9 O programa NR10

O programa N10 calcula a integral no intervalo fechado [A, B] utilizando apenas 10 pontos (nós) que são as raízes dos polinômios de Legendre e, para isso, ele utiliza a rotina QGAUSS [18].

A rotina QGAUSS retorna em *SS* o valor da integral da função *FUNC* entre os pontos *A* e *B* utilizando a integração de Legendre em dez pontos. Os valores dos nós (x_i) é dos pesos (w_i) podem ser observados em [18].

A tabela (3.12) mostra a descrição dos parâmetros utilizados pela rotina QGAUSS.

Parâmetro	Descrição	
FUNC	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
SS	Resultado da Integral no intervalo de [A, B] calculado por QGAUSS.	Saída

Tabela 3.12 Descrição dos Parâmetros de QGAUSS

3.2.10 O programa NR11

O programa NR11 calcula a integral no intervalo fechado [A, B] utilizando *N* nós e *N* pesos da quadratura de Gauss-Legendre e, para isso, ele utiliza a rotina GAULEG [18].

Pela seção anterior, percebe-se que a rotina QGAUSS apresenta um inconveniente, caso se deseje que a função seja avaliada em mais nós no intervalo de integração, além daqueles que foram especificados no algoritmo [18] tem-se que modificar o código da rotina QGAUSS. Os novos nós que serão acrescentados são as raízes do polinômio de Legendre de grau *n*, que podem ser encontradas em [24] com os seus respectivos pesos.

O algoritmo GAULEG calcula as abcissas nas quais a função será avaliada com seus respectivos pesos. Tal rotina recebe como parâmetros os limites *X1* e *X2*, é um dado *N* que é o grau do polinômio de Legendre cujas raízes vão ser calculadas retornando assim os vetores *X* e *Y* de tamanho *N* contendo os valores da abcissas (raízes calculadas) e dos pesos da quadratura de Gauss-Legendre.

A tabela (3.13) mostra a descrição dos parâmetros de GAULEG.

Parâmetro	Descrição	
X1	Limite inferior de integração.	Entrada
X2	Limite superior de integração.	Entrada
X	Vetor de tamanho N onde vão ser armazenados os nós calculados por GAULEG.	Saída
W	Vetor de tamanho N onde vão ser armazenados os pesos calculados por GAULEG.	
N	Grau do polinômio de Legendre para o cálculo dos nós.	

Tabela 3.13: Descrição dos Parâmetros de GAULEG

Uma outra rotina, denominada XGAULEG [18], deve interagir com GAULEG para a obtenção do valor da aproximação da integral.

3.2.11 O Programa NR12

O programa NR12 calcula a integral no intervalo [0, $+\infty$] utilizando N nós e N pesos da quadratura de Gauss-Laguerre generalizada e, para isso, ele utiliza a rotina GAULAG [18] para função peso $x^{\alpha} \exp(-x)$ com $\alpha > -1$.

Parâmetro	Descrição	
X	Vetor de tamanho N onde vão ser armazenados os nós calculados por GAULAG.	Entrada
W	Vetor de tamanho N onde vão ser armazenados os pesos calculados por GAULAG.	Entrada
N	Grau do polinômio de Laguerre para o cálculo dos nós.	Saída
ALF	A função-peso generalizada de Laguerre é denotada por $w(x) = x^{\alpha} \exp(-x)$. ALF denota o valor de α que é passado diretamente para a rotina de integração.	Entrada

Tabela 3.14: Descrição dos Parâmetr os de GAULAG

De forma semelhante ao algoritmo GAULEG, GAULAG retorna como resultados os vetores X e W contendo, respectivamente os nós e os pesos que são obtidos através do

polinômio de Laguerre generalizado de grau *N*. O valor de *N* também é dado como entrada no sistema.

A rotina GAULAG também necessita de uma outra rotina, denominada XGAULAG [19], que interaja com ela para obtenção do valor da integral.

A tabela (3.14) mostra a descrição dos parâmetros de GAULAG.

3.2.12 o programa NR13

O programa NR13 calcula a integral no intervalo [-∞, +∞] utilizando *N* nós e *N* pesos da quadratura de Gauss-Hermite e, para isso, ele utiliza a rotina GAUHER [18] para a função-peso $\exp(-x^2)$.

De forma semelhante aos algoritmos GAULEG e GAULAG, GAUHER retorna como resultado os vetores *X* e *W* contendo, respectivamente, os nós e os pesos que são obtidos através do polinômio de Hermite de grau *N*. O valor de *N* também é dado como entrada no sistema.

A tabela (3.15) mostra a descrição dos parâmetros de GAUHER.

Parâmetro	Descrição	
X	Vetor de tamanho N onde vão ser armazenados os nós calculados por GAUHER.	Entrada
W	Vetor de tamanho N onde vão ser armazenados os pesos calculados por GAUHER.	Entrada
N	Grau do polinômio de Hermite para o cálculo dos nós.	Saída

Tabela 3.15: Descrição dos Parâmetros de GAUHER

A rotina GAUHER também necessita de uma outra rotina, denominada XGAUHER [19], que interaja com ela para obtenção do valor da aproximação da integral.

3.2.13 o programa NR14

O programa NR14 calcula a integral no intervalo [-1, 1] utilizando *N* nós e *N* pesos da quadratura de Gauss-Jacobi e, para isso, ele utiliza a rotina GAUJAC [18], para a função-peso $w(x) = (1-x)^{\alpha}(1+x)^{\beta}$ com $\alpha,\beta > -1$.

Parâmetro	Descrição	
X	Vetor de tamanho N onde vão ser armazenados os nós calculados por GAUJAC.	Entrada
W	Vetor de tamanho N onde vão ser armazenados os pesos calculados por GAUJAC.	Entrada
N	Grau do polinômio de Jacobi para o cálculo dos nós.	Saída
ALF	Parâmetro α da Função-Peso	Entrada
BET	Parâmetro β da Função-Peso	Entrada

Tabela 3.16: Descrição dos Parâmetros de GAUJAC

De forma semelhante aos algoritmos GAULEG, GAULAG e GAUHER**,** GAUJAC retorna como resultado os vetores *X* e *W* contendo, respectivamente, os nós e os pesos que são obtidos através do polinômio de Jacobi de grau *N*. O valor de *N* também é dado como entrada no sistema.

3.3 Os programas QUADP

Como foi dito em seções anteriores, o QUADPACK é um conjunto de rotinas desenvolvidas para o cálculo numérico de integrais, também denominadas de integradores numéricos, que objetivam calcular uma aproximação numérica do problema de integração unidimensional,

$$\int_a^b f(x)dx \tag{3.9}$$

para o qual é requerido a precisão absoluta ε_a e/ou uma precisão relativa ε_r. Com este objetivo, todas as rotinas computam uma ou mais seqüências $\{Q_{n_k}, E_{n_k}\}$, k = 1,...,n, onde o primeiro elemento do par é uma aproximação para **I** baseada em n_k aproximações da integral e o segundo item denota a estimativa de erro correspondente. A seqüência de erro é truncada após o cálculo da n-ésima dupla se é suposto pelo integrador que a condição abaixo [20]:

$$|Q_{n_k} - I_w[a, b]| \leq E_{n_k} \leq \text{Máx } \{ \varepsilon_a, \varepsilon_r | I | \} \tag{3.10}$$

é satisfeita ou o integrador supõe a impossibilidade de atingir (3.10). Uma precisão estritamente absoluta é requerida se ε_r é zero e uma precisão estritamente relativa é requerida se ε_a é zero. Em todos os outros casos é suposta a mais fácil das duas tolerâncias

Em geral, as rotinas do QUADPACK calculam a aproximação da integral dada pela fórmula abaixo:

$$I_w[a, b] = \int_a^b w(x)f(x)dx \tag{3.11}$$

sobre o intervalo $[a, b] \subseteq \Re$ (finito ou infinito), onde w(x) é uma função-peso que é integrável em [a, b] e $I_w[a, b]$ representa o valor exato da integral no intervalo especificado. Sendo assim uma *soma de quadratura* fornece a aproximação:

$$Q_n[a,b]f := \sum_{k=1}^{n} P_k f(x_k) \cong I_w[a,b]f \tag{3.12}$$

Em (3.12) os números distintos $x_1, x_2, ...,x_n$ são as *abcissas* ou *nós* e os $P_1, P_2,..., P_n$ são os *pesos* correspondentes. O erro cometido pela fórmula (3.12) é representado pela expressão:

$$E[a, b]f := Q_n[a, b]f - I_w[a, b]f \tag{3.13}$$

3.3.1 Nomenclatura

Nesta seção será exibido algumas noções da nomenclatura utilizadas no restante desse capítulo.

As rotinas do QUADPACK calculam uma aproximação de (3.9) gerando uma ou mais seqüências $\{Q_{n_k}, E_{n_k}\}$, k = 1,...,n. O valor de Q_{n_k} será denominado de RESULT (variável que contém o valor da aproximação da integral), e E_{n_k}, sempre que não houver alguma observação, será chamado de E(J)F para indicar a estimativa de erro da aproximação da integral sobre um subintervalo $J \subseteq [A, B]$.

Quanto a nomenclatura dos programas que usam as rotinas QUADPACK, será feita de forma semelhante ao capítulo anterior, os programas serão denominados por QUADPN, onde N representa o número do programa.

As rotinas do QUADPACK possuem uma nomenclatura própria que denota a classe de problemas na qual ela se destina resolver, tal nomenclatura é apresentada pela tabela (3.17). Essas rotinas possuem em média 2.000 a 2.500 linhas de código, sendo portanto inviável exibir o código completo das rotinas. Para solucionar este problema, será apresentado um pequeno esquema de como o algoritmo funciona. O esquema apresentado pelo quadro (3.1) pode ser considerado de alto nível, e é geral para todas as rotinas de quadraturas apresentadas a seguir, sendo seu refinamento feito nas seções posteriores.

Primeira Letra	
Q	Indica que a rotina é de quadratura
Segunda Letra	
N	Integrador não adaptativo.
A	Integrador Adaptativo.
Terceira Letra	
G	Integrador de Finalidade Geral.
W	Integrador que utiliza função-peso.
Quarta Letra	
S	Integrais com Singularidades.
P	Integrais que apresentam pontos de dificuldades.
I	Integrais com intervalo de integração Infinito.
O	Integrais que apresentam uma função oscilatória (sin(ωx) ou cos(ωx)) com função-peso.
F	Integrais de Fourier [20].
C	Integrais de Cauchy [20].

Tabela 3.17: Nomenclatura das rotinas QUADPACK

Inicialize: RESULT, ABSERR, TOL. While ABSERR > TOL : tente reduzir ABSERR e atualize os valores intermediários. Retorne RESULT

Quadro 3.1: Esquema Geral das Rotinas do Quadpack

Neste esquema a frase *"tente reduzir o erro"*, representa a estratégia básica da rotina para retornar uma melhor aproximação da integral. Para tal, será feito uso de três estratégias básicas: não-adaptativa, globalmente adaptativa e globalmente adaptativa com extrapolação.

3.3.2 O programa QUADP1

O programa QUADP1 calcula a aproximação da integral no intervalo [A, B] para funções bem comportadas utilizando a rotina não-adaptativa QNG.

Considerando as seqüências $\{Q_{n_k}, E_{n_k}\}$ onde Q_{n_1} é aproximação da integral obtida pela quadratura de Gauss-Legendre de 10 pontos. Para tal, $n_1 = 10$ e Q_{n_k}, k = 2,3,4 são os pontos de Patterson [25] relacionados com Q_{10}. Logo, $n_k = 2n_{k-1}+1$, $2 \leq k \leq 4$. Obtém-se Q_{21}, Q_{43} e Q_{87}. As respectivas estimativas de erro são dadas por E_{10}, E_{21}, E_{43} e E_{87}.

A rotina QNG é chamada da seguinte forma:

CALL QNG (F, A, B, ERRABS, ERRREL, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.18):

Parâmetro	**Descrição**	
F	Função a ser integrada. Esta função é definida pelo usuário.	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
ERRABS	Precisão absoluta desejada	Entrada
ERRREL	Precisão Relativa desejada	Entrada
RESULT	Resultado da Integral no intervalo de [A, B] de F.	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.18: parâmetros de QNG

O quadro (3.2) mostra o esquema de funcionamento da rotina QNG.

Início: RESULT = Q_{21}, ERREST = E_{21} , TOL = Máx {ABSERR, ERRREL |RESULT|}, n = 21. Enquanto ERREST > TOL e n < 87 : RESULT = Q_{2n+1} , ABSERR = E_{2n+1} atualize TOL, n = 2n+1.

Quadro 3.2: Esquema de Funcionamento da rotina QNG

3.3.3 O programa QUADP2

O programa QUADP2 calcula a aproximação da integral no intervalo [A, B] utilizando a rotina adaptativa QAG, sendo possível a utilização de 6 (seis) tipos de quadraturas dependendo do comportamento da função integrando no intervalo especificado.

A rotina QAG subdivide o intervalo [A, B], através de uma bisseção, e utiliza a quadratura de Gauss-Kronrod (2k+1) pontos para estimar o valor da integral em cada subintervalo. Os possíveis valores de k são k = 7, 10, 15, 20, 25 ou 30, obtendo assim as quadratura de Gauss-Kronrod de 15, 21, 31, 41, 51 e 61 pontos.

Parâmetro	**Descrição**	
F	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
ERRABS	Precisão absoluta desejada	Entrada
ERRREL	Precisão Relativa desejada	Entrada
IRULE	Escolha da regra de Quadratura. Dependendo do comportamento da função integrando o valor desse parâmetro pode variar de 1 a 6. Vide tabela (3.20).	Entrada
RESULT	Resultado da Integral no intervalo de [A, B] de F.	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.19: parâmetros de QAG

O erro para cada subintervalo é calculado através de comparação com a quadratura gaussiana de k-pontos. Se um determinado subintervalo possuir o erro maior do que o estimado, então este sofre uma bisseção e o mesmo procedimento é aplicado a ambos os subintervalos. O processo de bisseção é continuado até que o critério de erro seja satisfeito ou

os subintervalos fiquem bem pequenos ou o máximo de subintervalos permitido seja alcançado [20].

A rotina QAG é chamada da seguinte forma:

CALL QAG (F, A, B, ERRABS, ERRREL, IRULE, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.19).

O parâmetro IRULE pode assumir os valores mostrados na tabela (3.20).

Se IRULE = 1, a função apresenta picos e singularidades. IRULE=2 é recomendado para maioria das funções. O valor de IRULE=3 é utilizado para função com pouco oscilatórias e variando até IRULE=6 se a função é muito oscilatória.

Valor de IRULE	VALOR DE K	VALOR DE 2K+1
1	7	30
2	10	21
3	15	31
4	20	41
5	25	51
6	30	61

Tabela 3.20: Valores de IRULE

O quadro (3.3) mostra o esquema de funcionamento da rotina QAG.

Início: RESULT = Q[A, B]F , ERRESULT = E[A,B]F, TOL = máx [ERRABS, ERRREL | RESULT |}.
Enquanto ABSERR > TOL, e o intervalo com o maior erro não é tão pequeno, e o número máximo de subdivisões ainda não foi alcançado e erros de arredondamento não foram detectados
Subdivida o intervalo com a maior estimativa de erro e atualize RESULT, ERREST, TOL.
Retorne RESULT

Quadro 3.3: Esquema de Funcionamento da rotina QAG

3.3.4 O programa QUADP3

O programa QUADP3 calcula a aproximação da integral da forma:

$$\int_a^b w(x)f(x)dx \qquad (3.14)$$

sendo *a* e *b* finitos e com $a < b$ e $w(x) = (x-a)^{\alpha}(b-x)^{\beta}\ v(x)$ com $\alpha, \beta > -1$. A função v(x) pode ser uma das seguintes funções:

$$\begin{cases} v(x) = 1 \\ v(x) = \log(X - A) \\ v(x) = \log(B - X) \\ v(x) = \log(X - A)\mathrm{Log}(B - X) \end{cases}$$

para isso o programa QUADP3 utiliza a rotina adaptativa QAWS.

Parâmetro	**Descrição**	
F	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de Integração.	
IWEIGH	Tipo de função peso a ser utilizada. $\begin{cases} = 1 \ \text{ se } \ (X-A)^{\alpha}(B-X)^{\beta} \\ = 2 \ \text{ se } \ (X-A)^{\alpha}(B-X)^{\beta}\mathrm{Log}(X-A) \\ = 3 \ \text{ se } \ (X-A)^{\alpha}(B-X)^{\beta}\mathrm{Log}(B-X) \\ = 4 \ \text{ se } \ (X-A)^{\alpha}(B-X)^{\beta}\mathrm{Log}(X-A)\mathrm{Log}(B-X) \end{cases}$	Entrada
ALPHA	Valor de Parâmetro α. $\alpha > -1$	Entrada
BETA	Valor do Parâmetro β. $\beta > -1$	
ERRABS	Precisão absoluta desejada	Entrada
ERRREL	Precisão Relativa desejada	Entrada
RESULT	Resultado da Integral no intervalo [A, B] de F.	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.21: Parâmetros de QAWS

A rotina QAWS efetua uma bisseção no intervalo de integração original [A, B] e é aplicado o método de Clenshaw-Curtis modificado (usando 13 e 25 pontos) em ambas as partes. Em todos os passos adicionais o método de Clenshaw-Curtis é utilizado para aqueles

subintervalos que possuem A ou B como extremos, e a quadratura de Gauss-Kronrod é utilizada para todos os outros subintervalos.

A rotina QAWS é chamada da seguinte forma:

CALL QAWS (F, A, B, IWEIGH, ALPHA, BETA, ERRABS, ERRREL, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.21).

3.3.5 O programa QUADP4

O programa QUADP4 calcula a aproximação da integral da forma:

$$\int_a^b w(x)f(x)dx \tag{3.15}$$

sendo *a* e *b* finitos e com $a < b$ e w(x) é da forma:

$$w(x) = \frac{1}{(x - C)} \tag{3.16}$$

com $a < C < b$ e para isso o programa QUADP3 utiliza a rotina adaptativa QAWC.

A rotina QAWC é chamada da seguinte forma:

CALL QAWC (F, A, B, C, ERRABS, ERRREL, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.22).

A estratégia utilizada pela rotina QAWC para o cálculo da aproximação da integral é uma versão modificada de QAG. Em cada passo, o subintervalo com a maior estimativa de erro é subdivido. Assumindo que o intervalo $[C_1, C_2]$, $C_1 < C_2$, as subdivisões ocorrem da seguinte maneira:

- Se $C \notin [C_1, C_2]$ então $[C_1, C_2]$ sofre uma bisseção;
- Se $C_1 < C \leq (C_1 + C_2)/2$ então $[C_1, C_2]$ é subdividido no ponto $(C + C_2)/2$.
- Por outro lado, a subdivisão ocorre em $(C + C_1)/2$.

A integração de Clenshaw-Curtis modificada (utilizando 13 e 25 pontos) é executada sempre que:

$$C_1 - D < C < C_2 + D \qquad (3.17)$$

onde,

$$D = (C_2 - C_1)/20 \qquad (3.18)$$

Por outro lado, a integração de Gauss-Kronrod (utilizando 7 e 15 pontos) é aplicada.

Parâmetro	**Descrição**	
F	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
C	Valor de C que compõe a função integrando que possui a forma $w(x) = \frac{1}{(x - C)}$.	
ERRABS	Precisão absoluta desejada	Entrada
ERRREL	Precisão Relativa desejada.	Entrada
RESULT	Resultado da Integral no intervalo de [A, B] de F.	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.22: Parâmetros de QAWC

3.3.6 O programa QUADP5

O programa QUADP5 calcula a aproximação da integral para funções que apresentam singularidades nos limites de integração. Para isso QUAPD5 utiliza a rotina QAGS.

A rotina QAGS é um integrador de propósito geral que usa o esquema globalmente adaptativo para reduzir o erro absoluto. Esta rotina subdivide o intervalo [A, B] e utiliza a quadratura de Gauss-Kronrod de 21-pontos para calcular a aproximação da integral em cada subintervalo.

A rotina QAGS é chamada da seguinte forma:

CALL QAGS (F, A, B, ERRABS, ERRREL, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.23).

Seja Q(J)F a aproximação da integral em J, onde J pertence a uma partição $\wp$[A, B] de [A, B]. Tal aproximação é obtida através da quadratura de Gauss-Kronrod de 21 pontos.

Parâmetro	**Descrição**	
F	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
ERRABS	Precisão absoluta desejada	Entrada
ERRREL	Precisão Relativa desejada	Entrada
RESULT	Resultado da Integral no intervalo de [A, B] de F.	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.23: Parâmetros de QAGS

Seja,

$$SQ = \sum_{J\in\wp[A,B]} Q(J)F \qquad (3\text{-}19)$$

$$SE = \sum_{J\in\wp[A,B]} E(J)F \qquad (3\text{-}20)$$

Será convencionado que o algoritmo alcançou um nível λ se o menor intervalo da partição de [A, B] é do tamanho $|B\text{-}A|^{-\lambda}$. No nível λ, todos os intervalos da partição são chamados de intervalos "pequenos", caso contrário os intervalos são denominados "grandes". Quando um intervalo é pequeno (com respeito ao nível corrente) e é selecionado para bisseção diz-se que um novo nível foi introduzido. Esta seleção não indica uma imediata bisseção do intervalo corrente, mas este subintervalo será o preferido em caso de uma futura bisseção. A extrapolação utilizada pelo algoritmo pela rotina QAGI é executada pelo

algoritmo ε-algoritmo, que é um método de extrapolação não linear. Detalhes de como funciona o ε-algoritmo pode ser visto em [20].

Início: SQ = Q[A, B]F, SE = E[A,B]F, SBE = SE, TOL = máx [ERRABS, ERRREL |SQ|}, EXE = ∞, EXTOL = TOL, level = 0.
Enquanto SE > TOL e EXE > EXTOL e
O intervalo selecionado para ser subdividido não é pequeno e
O número máximo de subdivisões ainda não foi encontrado e
Nenhum erro de arredondamento ainda não foi detectado e
Cálculos adicionais ainda estão esperando melhora então, selecione o intervalo com a maior estimativa de erro para subdivisão.
- Se nenhum nível é introduzido, subdivida o intervalo corrente e atualize SQ, SE, SBE, TOL;
- Por outro lado:
* Enquanto SBE > EXTOL e nenhum erro de arredondamento é detectado sobre os intervalos grandes, subdivida o intervalo grande com a maior estimativa de erro e atualize SQ, SE, TOL;
* Extrapole e atualize EXQ, EXE, EXTOL, SBE (=SE), level = level + 1.
Se EXE / |EXQ| > SE / |SQ|, atribua RESULT = SQ, ERRABS = SE por outro lado testa se há divergências e atribua RESULT = EXQ, ABSERR = EXE.

Quadro 3.4: Esquema de Funcionamento da rotina QAGS

Onde,

SBE = soma de todas as estimativas de erro sobre os intervalo grandes,

EXQ = aproximação da integral obtida por extrapolação,

EXE = estimativa de erro com respeito a EXQ,

EXTOL = Max {ERRABS, ERRREL |EXQ|}.

O algoritmo QAGS satisfaz a descrição do esquema apresentado pelo quadro (3.4).

3.3.7 O programa QUADP6

O programa QUADP6 calcula a aproximação da integral para funções cujo intervalo de integração é infinito. Para isso ele utiliza a rotina QAGI.

A rotina QAGI transforma um intervalo infinito de integração para o intervalo (0, 1] através da transformação [20]:

$$\int_{A}^{\pm\infty} F(x)dx = \pm\int_{0}^{1} F(A \pm (1-T)/T)T^{-2}dT \qquad (3.21)$$

se A é finito.

Por outro lado, a seguinte substituição é utilizada

$$\int_{-\infty}^{\infty} F(x)dx = \int_{0}^{\infty} (F(x)+F(-x))dx = \int_{0}^{1} (F((1-T)/T)+F((-1+T)/T))T^{-2}dT. \qquad (3.22)$$

A rotina QAGI utiliza a mesma estratégia de QAGS. Contudo existem regras básicas que são diferentes: A aproximação da integral Q(J)F sobre o subintervalo J $\subseteq$ (0,1] é agora obtido através da quadratura de Gauss-Kronrod de 15 pontos. A estimativa de erro E(J)F é obtida utilizando Q junto com quadratura de Gauss de 7 pontos [20].

Parâmetro	**Descrição**	
F	Função a ser integrada. Esta função é definida pelo usuário	Entrada
BOUND	Limitante finito do intervalo de integração	Entrada
INTERV	Flag que indica o intervalo de integração $\begin{cases} = 1 \text{ se } (-\infty, \text{BOUND}) \\ = 1 \text{ se } (\text{BOUND}, +\infty) \\ = 2 \text{ se } (-\infty, +\infty) \end{cases}$	Entrada
ERRABS	Precisão absoluta desejada	Entrada
ERRREL	Precisão Relativa desejada	Entrada
RESULT	Resultado da Integral no intervalo de [A, B] de F.	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.24: Parâmetros de QAGI

A rotina QAGI é chamada da seguinte forma:

CALL QAGI (F, BOUND, INTERV, ERRABS, ERRREL, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.24).

3.3.8 O programa QUADP7

O programa QUADP7 calcula a aproximação da integral para funções que possuem um singularidade interior do intervalo de integração. Para isso ele utiliza a rotina QAGP.

Parâmetro	Descrição	
F	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
NPTS	Quantidade de Singularidades no intervalo de integração.	Entrada
POINTS	Vetor de tamanho NPTS contendo os pontos de singularidade no intervalo [A, B]. Este pontos são representados por P_N em (3-20)	Entrada
ERRABS	Precisão absoluta desejada	Entrada
ERRREL	Precisão Relativa desejada	Entrada
RESULT	Resultado da Integral no intervalo de [A, B] de F.	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.25: Parâmetros de QAGP

A rotina QAGP é similar a QAGS. A definição de nível, contudo, sofre uma sutil diferença. Assumindo que existem N pontos de parada no intervalo [A, B], A < B seja:

$$A < P_1 < P_2 < ..... < P_N < B \tag{3.23}$$

com,

$$P_0 = A,\ P_{N+1}=B \tag{3.24}$$

A primeira aproximação da integral é computada dividindo o intervalo [A, B] e aplicando a quadratura de Gauss-Kronrod de 21 pontos. A partir de agora, todas as subdivisões serão uma bisseção e a quadratura de Gauss-Kronrod é aplicada para o cálculo da aproximação da integral em cada subintervalo. Na rotina QAGP, a definição de subintervalos pequenos e grandes é diferente. Durante a execução do nível $\lambda \geq 0$, o menor intervalo da

partição $[P_k, P_{K+1}]$ tem tamanho $(P_{K+1}-P_K)\ 2^{-\lambda}$. Os subintervalos desse tamanho ou menores são chamados pequenos, e os outros grandes [20].

A rotina QAGP é chamada da seguinte forma:

CALL QAGP (F, A, B, NPTS, POINTS, ERRABS, ERRREL, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.25).

3.3.9 O programa QUADP8

O programa QUADP8 calcula a aproximação da integral da forma:

$$\int_a^b w(x)f(x)dx \tag{3.25}$$

sendo *a* e *b* finitos e com $a < b$ e w(x) é da forma:

$$w(x) = \begin{cases} \cos(\omega x) \\ \sin(\omega x) \end{cases} \tag{3.26}$$

O valor de ω é dado como entrada pelo usuário. Para isso, o programa QUADP8 utiliza a rotina adaptativa QAWO.

A estratégia de extrapolação é uma modificação daquela utilizada em QAGS. Assumindo que o subintervalo possui tamanho:

$$L = |B - A|\ 2^{-\lambda} \tag{3.27}$$

se

$$L\omega > 4 \tag{3.28}$$

a integração sobre este subintervalo é executada através do procedimento de Clenshaw-Curtis modificado com 25 pontos. Se a condição de (3-28) não é satisfeita então a integração é executada pelo método de Gauss-Kronrod 7/15 pontos [20].

Parâmetro	Descrição	
F	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
B	Limite superior de integração.	Entrada
IWEIGH	Tipo de função peso a ser utilizada. $\begin{cases} = 1, & \text{se } \cos(\omega x) \\ = 2, & \text{se } \sin(\omega x) \end{cases}$	Entrada
OMEGA	Valor de Parâmetro ω.	Entrada
ERRABS	Precisão absoluta desejada	Entrada
ERRREL	Precisão Relativa desejada	Entrada
RESULT	Resultado da Integral no intervalo de [A, B] de w(x) f(x).	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.26: Parâmetros de QAWO

No algoritmo QAWO, o nível do subintervalo é iniciado com zero, e um novo nível é introduzido tal como em QAGS, porém somente nos intervalos onde a quadratura de Gauss-Kronrod é utilizada.

A rotina QAWO é chamada da seguinte forma:

CALL QAWO(F, A, B, IWEIGH, OMEGA, ERRABS, ERRREL, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.26).

3.3.10 O programa QUADP9

O programa QUADP9 calcula a aproximação da integral da forma:

$$\int_a^{\infty} w(x) f(x) dx \qquad (3.29)$$

w(x) é da forma:

$$w(x) = \begin{cases} \cos(\omega x) \\ \sin(\omega x) \end{cases}. \qquad (3.30)$$

O valor de ω é dado como entrada pelo usuário, para isso o programa QUADP9 utiliza a rotina adaptativa QAWF.

A rotina QAWF calcula a aproximação da integral no intervalo semi-infinito: [A, ∞) onde A é finito.

Seja C_k definido por:

$$C_k = [A + (k-1)c, A+kc], \; k = 1,2,..... \qquad (3.31)$$

Os intervalos C_k são constantes é o valor de *c* é dado por [20]:

$$c = \frac{2\lfloor|\omega|\rfloor + 1}{|\omega|}\pi \qquad (3\text{-}32)$$

onde [|ω|] representa o maior inteiro que é menor ou igual |ω|. O ε-algoritmo [20] é utilizado como método de extrapolação.

O integrador QAWF trabalha somente com a tolerância absoluta cujo valor é dado como entrada pelo usuário. Sobre um intervalo C_k, QAWF tenta satisfazer o seguinte requisito de precisão [20]:

$$TOL_k = u_k \; ERRABS \qquad (3.33)$$

com $u_k = (1-p)p^{k-1}$, k = 1,2,...., e p = $\frac{9}{10}$ [20].Contudo, quando dificuldades ocorrem durante a integração dentro de cada subintervalo, QAWF relaxa a tolerância C_i para [20]:

$$TOL_i = u_i \quad \max\left\{ERRABS, \max_{1\le k\le i-1} E(C_k)F\right\} \qquad (3.34)$$

A rotina QAWF é chamada da seguinte forma:

CALL QAWF (F, A, IWEIGH, OMEGA, ERRABS, RESULT, ERREST)

onde os parâmetros são apresentados pela tabela (3.27):

Parâmetro	Descrição	
F	Função a ser integrada. Esta função é definida pelo usuário	Entrada
A	Limite inferior de integração.	Entrada
IWEIGH	Tipo de função peso a ser utilizada. $\begin{cases} = 1, \text{ se } Cos(\omega x) \\ = 2, \text{ se } Sin(\omega x) \end{cases}$	Entrada
OMEGA	Valor de Parâmetro ω.	Entrada
ERRABS	Precisão absoluta desejada	Entrada
RESULT	Resultado da Integral no intervalo de [A, B] de w(x) f(x).	Saída
ERREST	Estimativa do erro Absoluto	Saída

Tabela 3.27: Parâmetros de QAWF

No próximo capítulo, todos os módulos do sistema INTEGRE serão analisados de forma sucinta.

CAPÍTULO IV

INTEGRE – Sistema acoplado para o Cálculo de Integrais com análise interativa de Funções Numéricas

Neste capítulo será apresentado o sistema INTEGRE e serão analisados todos módulos que fazem parte dele juntamente com os aspectos envolvidos na sua implementação.

4.1 Introdução

Apesar do grande desenvolvimento que o software numérico teve nas últimas décadas, seu uso permanece restrito a uma classe de usuários mais ou menos especializada. Surgiu, então, a idéia de facilitar o uso de softwares numéricos através da utilização de ambientes interativos.

Ford e ILes [27] apontam como grande esperança da década de 90 se conseguir conceber e implementar sistemas que forneçam um ambiente harmonioso em que a utilização de qualquer um dos métodos de solução não implique em mudanças de procedimento drástico. O que se busca é um sistema capaz de dar assistência a um cientista ou a um engenheiro na solução de seus problemas matemáticos e que se apresente como um sistema amigável.

O trabalho apresentado é um sistema acoplado para otimizar a utilização de algoritmos para o cálculo de integrais. Para isso, foi desenvolvido um ambiente interativo capaz de fornecer, de uma forma amigável, a análise do comportamento de uma determinada função. Esse processo de análise culmina com a representação gráfica da função, com exibição do valor das coordenadas dos pontos visitados pelo mouse e do valor da derivada correspondente. Algumas facilidades foram incorporadas ao sistema de visualização gráfica, como os efeitos de zoom em áreas escolhidas pelo usuário.

Quanto ao processo de escolha dos algoritmos de cálculo a serem utilizados, se dá através de um estudo gráfico da função integrando especificada pelo usuário, levando-se também em consideração outros dois fatores: tempo de processamento e precisão dos resultados. O sistema contém um módulo de aprendizado automático permitindo que ele automodifique os parâmetros, de acordo com o sucesso ou insucesso da solução fornecida.

Nas próximas seções cada módulo do sistema será descrito de forma sucinta.

4.2 Arquitetura do Sistema INTEGRE

O sistema INTEGRE foi desenvolvido em Visual Basic [28]. Ele possui a finalidade de efetuar o cálculo, de forma eficiente, de integrais numéricas de funções integrando com uma variável utilizando um (o mais adequado a uma determinada situação) ou mais algoritmos existentes, a seu dispor, em um banco de algoritmos. O usuário situa facilmente o problema seguindo a orientação fornecida pelo sistema. A escolha do algoritmo é feita levando-se em conta dois fatores: tempo de processamento e precisão dos resultados. O usuário fornece o peso que cada um desses fatores terá na solução de seu problema. Baseado nesses pesos o sistema ativa o algoritmo de cálculo.

Basicamente, o INTEGRE é constituído por duas partes com funções distintas mas acopladas entre si. Uma parte simbólica constituída pelo método de escolha, da interface com o usuário, da rotina das derivadas, da rotina gráfica, do método de aprendizado automático, das análises léxica e sintática, do arquivo de pesos e do arquivo de ocorrências. A outra parte, a parte numérica, é responsável pelo cálculo numérico das integrais. Esquematicamente, o INTEGRE pode se representado pela figura (4.1).

A *Interface do Sistema* é o mecanismo pelo qual o usuário se comunica com o INTEGRE e é através dela que serão acrescentadas informações relativas ao comportamento da função no intervalo especificado. É também através da interface que o usuário tomará contato com o gráfico da função [29, 30].

As *Análises léxica e sintática* são responsáveis pela validação e reconhecimento da função fornecida pelo usuário e geração automática do código da função [32, 33].

A *Análise Matemática* detecta peculiaridades da função dentro do intervalo de integração, e pode ser dividida em: *Análise Gráfica*, que permite a detecção visual, a partir do gráfico, de assíntotas e pontos críticos que possam existir na função e da *Análise da Derivada* que efetua a detecção matemática de assíntotas e pontos críticos dentro do intervalo de integração. Essa detecção pode ser feita de forma interativa com o usuário.

O *Método de Escolha* efetua a seleção do melhor algoritmo para solucionar o problema proposto pelo usuário. Ele também é responsável pela ativação dos algoritmos numéricos previamente escolhidos para o cálculo da aproximação da integral.

O *Arquivo de Pesos* contém o fator de desempenho de cada algoritmo em termos de tempo de processamento e precisão dos resultados. Esse fator é determinado pelo desvio em relação ao valor médio dos desempenhos obtidos em execuções passadas por todos os algoritmos semelhantes, isto é, que são aplicáveis a uma determinada classe de problemas.

O *Arquivo de ocorrências* contém as informações das ocorrências dos algoritmos na resolução do problema proposto.

O *Módulo de Aprendizado Automático* permite que o sistema se adapte à situação particular do usuário. A adaptação é feita através de modificações no arquivo de pesos com base nas informações contidas no arquivo de ocorrências.

Os *Algoritmos Numéricos* calculam o valor da aproximação da integral que será fornecido ao usuário.

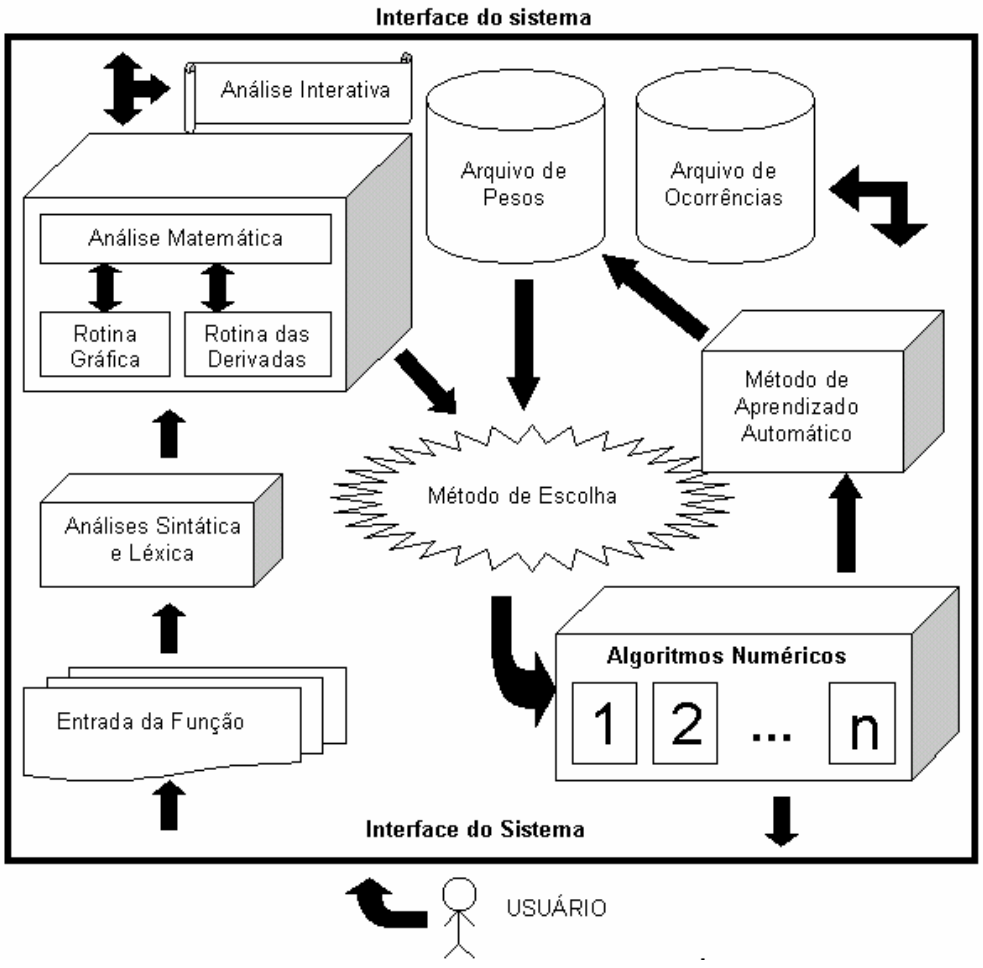


Figura 4.1: Arquitetura do sistema INTEGRE

4.3 Análises Léxica e Sintática

Como foi observado em seções anteriores o cálculo numérico de integrais é definido como uma combinação de valores da função *f* em certos pontos x_i, $a \leq x_i \leq b$, chamados de *nós*, e certos valores de ω_i que consistem os *pesos*, ou seja :

$$\int_a^b f(x)dx \approx \omega_1 f(x_1) + \omega_2 f(x_2) + ... + \omega_n f(x_n) + \omega_{n+1} f(x_{n+1}) \qquad (4.1)$$

A quadratura acima é dita n+1 pontos, e necessita dos valores de $f(x_0)$, $f(x_1)$ ... e $f(x_{n+1})$. Logo a rotina que implementa a Quadratura necessita de uma sub-rotina que avalie f(x) em determinados pontos.

Uma forma de resolver esse problema seria exigir do usuário a declaração da sub-rotina em uma linguagem compatível com a utilizada na implementação da rotina de quadratura, caindo exatamente naqueles casos indesejáveis listados no Capítulo I.

A idéia é exigir do usuário o fornecimento de uma especificação simples de f(x), escrita de uma maneira mais próxima da linguagem matemática usual. Em seguida, o sistema se encarregaria de transformar tal especificação em código de máquina, que seria utilizado pela rotina da Quadratura. A figura (4.2) esquematiza o funcionamento de um sistema auxiliado por uma linguagem intermediária.

Dessa maneira, o usuário especifica a função integrando numa linguagem pseudonatural (matemática), isto é, da maneira como ele escreve uma sentença ou fórmula matemática. Surge, então, a necessidade de se traduzir essa especificação informal da função em linguagem computacional que possa ser transformada em programa executável. Um tradutor faz-se necessário para transformar tal especificação em um programa computacionalmente aplicável [32, 33].

Para tanto, o tradutor foi construído baseando-se nas idéias de construção de compiladores. Detalhes desse processo pode ser visto em [31].

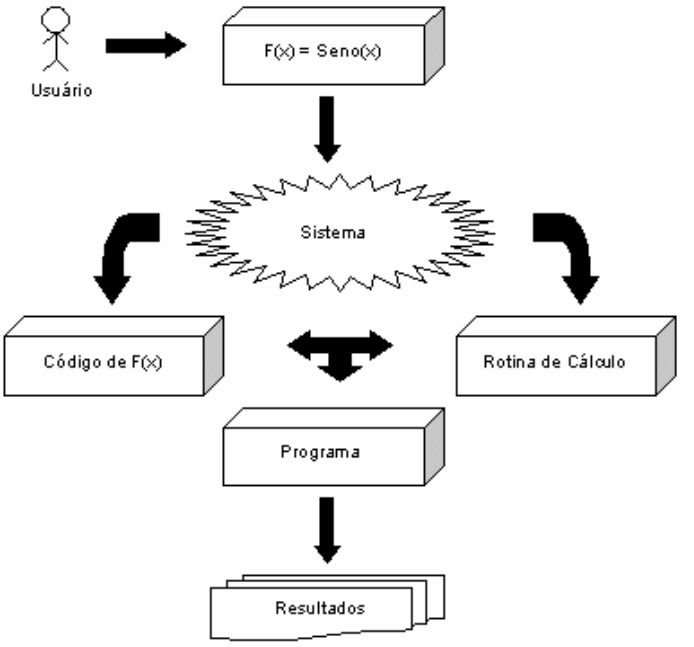


Figura 4.2: Sistema de Cálculo Numérico auxiliado por Linguagem Intermediária

4.4 Análise Matemática

Uma vez validada a função, o INTEGRE fará a análise matemática verificado a existência de pontos críticos da função integrando no intervalo de integração. Esta verificação é feita através da visualização do gráfico da função no intervalo especificado, com a vantagem do usuário poder interagir com o sistema através de ZOOM em áreas específicas juntamente com a exibição da derivada primeira.

O resultado dessa análise é utilizado diretamente pelo método de escolha para determinar o melhor algoritmo para resolução da integral proposta.

4.4.1 As Bibliotecas de Ligação Dinâmica (DLLs)

As DLLs (Dynamic-Link libraries – bibliotecas de ligação dinâmica) são módulos executáveis que contém código que podem ser utilizados por diversos outros aplicativos Windows. A principal característica de uma DLL é prover serviços a um programa Windows através das diversas funções que podem ficar encapsuladas dentro dela.

Quando um programa utiliza um procedimento ou uma função, uma cópia daquele procedimento ou função é ligada estaticamente ao arquivo executável. Se dois programas necessitam usar a mesma função deve existir uma cópia de cada função para esses procedimentos. A figura (4.3) exemplifica o processo descrito acima.

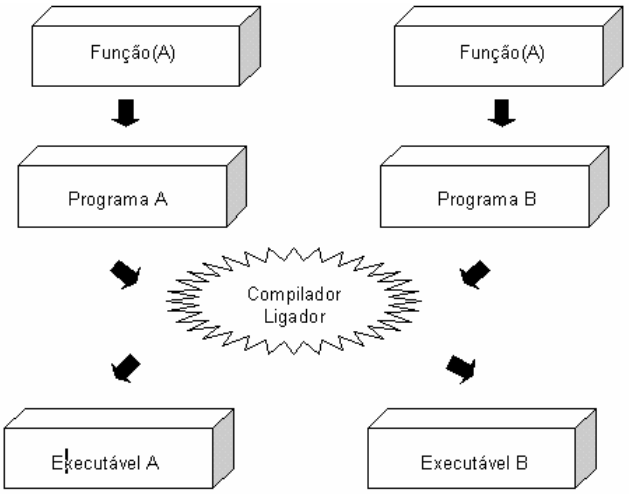


Figura 4.3: Função ligada estaticamente a um programa executável

Suponha que a Função(A) tenha 256KB de tamanho, o Programa A e B tenham 300KB, sendo assim, o tamanho de cada executável é de 556KB, fazendo um total de 1112KB.

Em contraste com a caso acima, o código contido dentro da DLL é ligado aos programas em tempo de execução, ou seja, a DLL é chamada diretamente pelo programa executável sem a necessidade de compilações/ligações intermediárias. A figura (4.4) exemplifica o caso descrito acima.

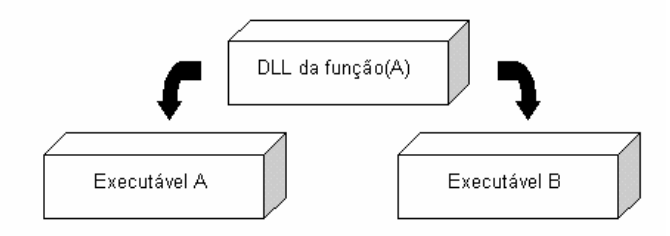


Figura 4.4: DLL sendo chamada por dois executáveis

Supondo os dados anteriormente mencionados, tem-se 256KB da DLL e 600KB dos dois executáveis, fazendo um total de 856KB de espaço total. Os números acima, embora sejam fictícios, dão uma boa idéia de como esse processo funciona.

4.4.2 Rotina das derivadas

O programa das derivadas foi escrito em Visual C++ [34], sendo incorporado ao sistema através de uma DLL. A rotina possui uma hierarquia de classes mostrada pela figura (4.5).

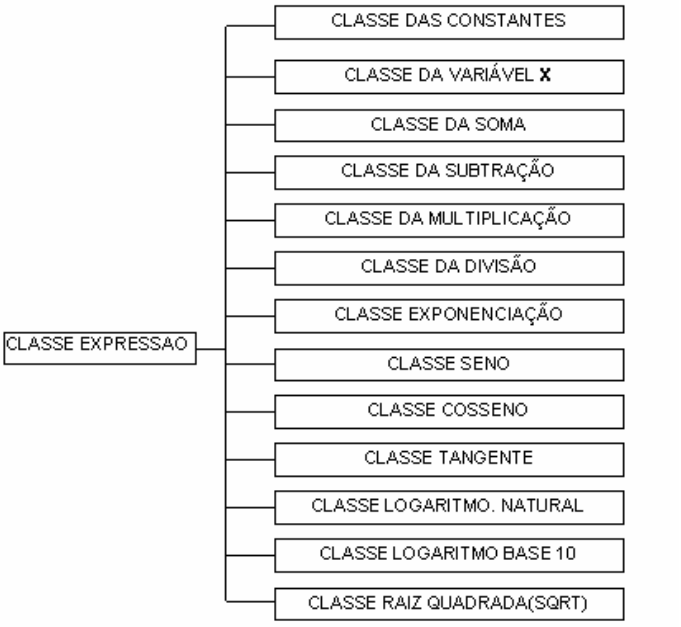


Figura 4.5: Hierarquia de Classes do programa das Derivadas

A classe expressão foi criada unicamente para que seus métodos fossem herdados e sobrecarregados pelas classes filhas. Cada classe filha utiliza os métodos herdados de acordo com suas especificações. Isso se dá graças ao poder da programação orientada a objetos [35]. Os métodos da classe expressão são apresentados pelo quadro (4.1).

O método *EXPR()* e *~EXPR()* são os métodos construtor e destrutor, ou seja, o método construtor é aquele que é executado quando a classe é inicializada e o destrutor é executado quando a classe é finalizada. Todas as outras classes possuem seus métodos construtores e destrutores. O método *DERIVA* cuida de derivar a função especificada pelo usuário. Cada classe deriva a função seguindo um conjunto de regras que serão apresentadas na seção (4.4.3). O método *IS_CONST* verifica se um dada expressão é constante. O método *COPIA* cria um cópia da expressão que vai ser derivada, esta será enviada diretamente para a rotina de derivação.

```
CLASS EXPR
 {
   PUBLIC:
   EXPR();
   VIRTUAL ~EXPR();
   VIRTUAL EXPR *DERIVA();
   VIRTUAL INT IS_CONST();
   VIRTUAL EXPR *COPIA();
   VIRTUAL VOID PRINT(FILE *);
   VIRTUAL INT APARENTAR();
 };
```

Quadro 4.1: Métodos da classe EXPRESSÃO

```
EXPR *MULT::DERIVA()
  {
   RETURN NEW SOMA(NEW MULT(EXPR1 ->DERIVA(),EXPR2 ->COPIA()) ,
                           NEW MULT(EXPR1 ->COPIA(),EXPR2 ->DERIVA()) );
  }
```

Quadro 4.2: Copia dos parâmetros para derivação

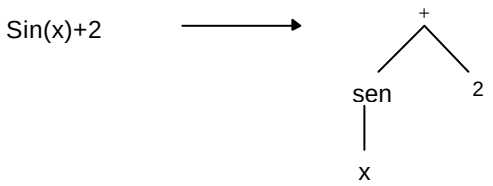
No quadro (4.2) mostra o exemplo da derivação do produto de duas expressões. Como pode-se observar pelo quadro (4.2), a derivada está sendo definida como a soma de dois produtos. Cada produto é definido como sendo a derivada de uma expressão vezes a copia da outra expressão. Note que, EXPR1 e EXPR2 vão ser utilizadas duas vezes, logo existe a necessidade de uma cópia para cada expressão.

O método *PRINT* imprime o resultado para um arquivo (se for especificado STDOUT, ou seja, saída padrão, o resultado será colocado na tela) e o método *APARENTAR* coloca parênteses para evitar ambigüidades na derivada da expressão.

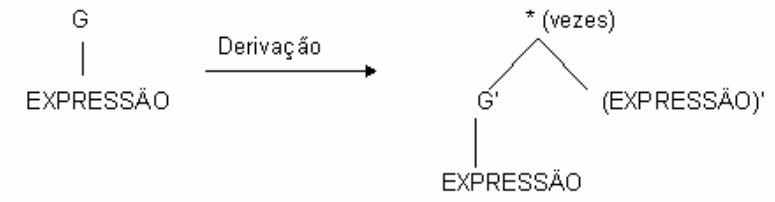
4.4.3 Cálculo das Derivadas

Depois de passar pela filtragem das análises léxica e sintática a função fornecida pelo usuário vai para análise matemática, e conseqüente execução do sistema de cálculo numérico. Esta expressão deve ser transformada e armazenada em uma estrutura de dados conveniente. Tal estrutura conterá toda a significação da função numérica e terá a ordem de precedência entre operadores e seus respectivos operandos. A função é decomposta numa árvore para que cada parte integrante dessa árvore seja analisada para cálculo da derivada.

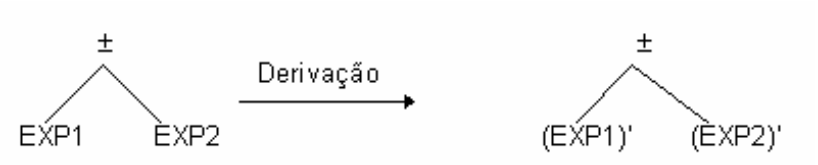
Suponhamos que o usuário forneça a seguinte expressão: sen(x) + 2 , esta é decomposta na seguinte árvore:



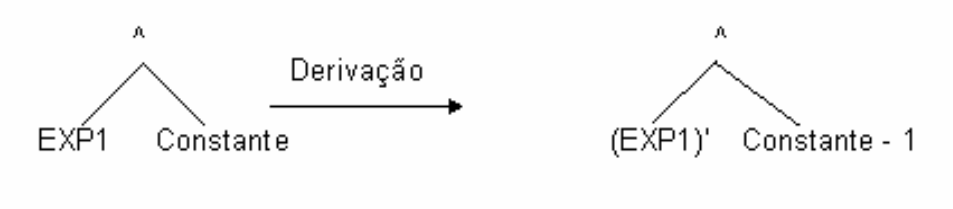
O cálculo das derivadas se utiliza desta árvore para gerar as derivações. Após gerar a árvore para derivação da função fornecida pelo usuário o programa a deriva utilizando basicamente oito regras. A primeira é que a derivada de uma expressão do tipo G(EXPRESSÃO) é igual a derivada de G vezes a derivada de EXPRESSÃO.



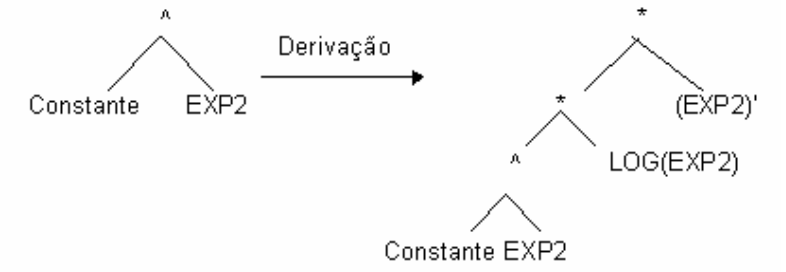
A segunda regra é que a derivada de expressões do tipo EXP1 ± EXP2 é igual à soma das derivadas de cada uma:



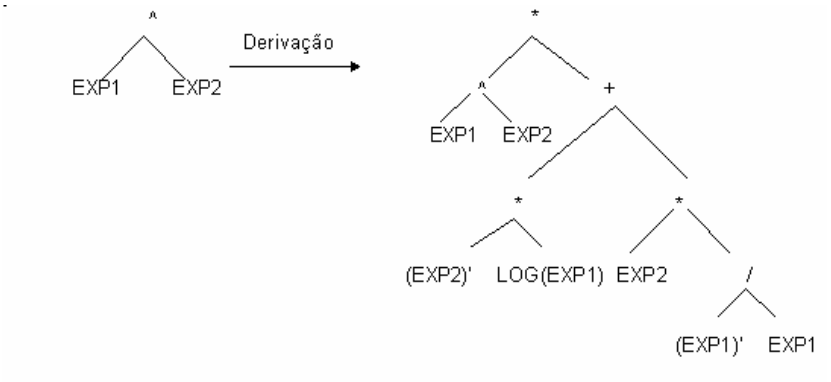
A terceira regra é que a derivada de expressões exponenciais é igual a seguinte derivação:



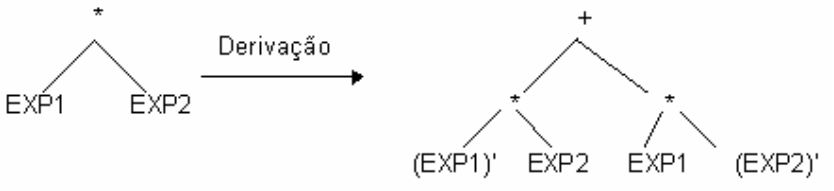
A quarta regra é que a derivada de expressões do tipo CONSTANTE ^ EXP2 é igual a seguinte derivação:



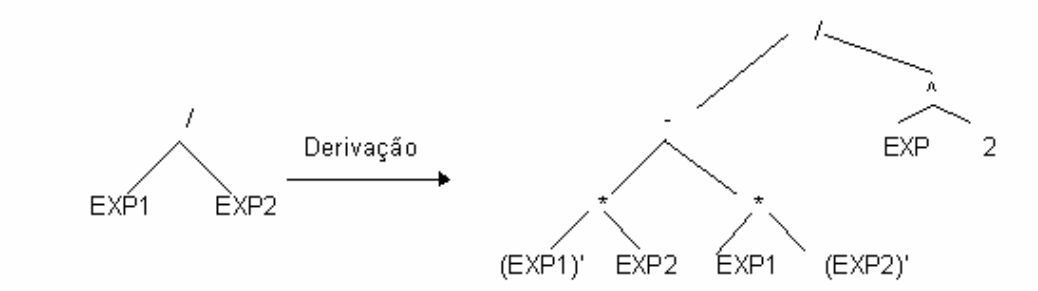
A quinta regra é que a derivada de expressões do tipo EXP1 ^ EXP2 é igual a seguinte derivação:



A sexta regra é que a derivada de expressões do tipo EXP1 * EXP2 é igual a seguinte derivação:



A sétima regra é que a derivada de expressões do tipo EXP1 / EXP2 é igual a seguinte derivação:



A oitava regra é que a derivada de constantes é ZERO.

Com base em cada uma das regras dá-se a derivação e o conteúdo da derivada é depositado em uma estrutura de dados idêntica àquela que armazena a função fornecida pelo usuário. Depois disto, as duas estruturas são disponibilizadas para serem utilizadas pelo sistema INTEGRE.

4.4.4 Análise Gráfica

O projeto para se desenhar gráficos de funções começa fixando o tamanho do gráfico em um número de linhas e de colunas, por exemplo 50 linhas e 100 colunas. Fixado o tamanho do gráfico conhece-se o número de pontos distintos que pode caber na área e cada

ponto dessa área tem coordenadas inteiras bem definidas. Associando-se, no caso y = f(x), um par (x, y) será mapeado num par (x', y') de inteiros que pode ser representado na tela do computador. A construção do mapeamento envolve a determinação do valores máximos e mínimos de x e y.

Sejam X_m e X_M os valores mínimos e máximo de x e sejam Y_m e Y_M os valores mínimo e máximo de y. Sejam *c* e *l* o número de colunas e de linhas disponíveis na superfície do gráfico. Então as unidades utilizadas para x e y são dadas por :

$$\mathbf{ux = (X_M - X_m)/(c - 1)}$$

$$\mathbf{uy = (Y_M - Y_m)/(l - 1)}.$$

Agora é simples obter o mapeamento :

$$\mathbf{Mx = (X_M - X_i) / ux}$$

$$\mathbf{My = (Y_M - Y_i) / uy}$$

onde $\mathbf{X_i}$ e $\mathbf{Y_i}$ são pontos do gráfico em coordenadas euclidianas e **Mx** e **My** são os pontos devidamente mapeados para a tela do computador.

4.5 Método de Escolha

A escolha de um algoritmo numérico para resolução de um problema se baseia em dois fatores: aplicabilidade e eficiência.

Levando-se em conta a aplicabilidade, os algoritmos foram agrupados de acordo com as características da integral. Estas características levam em conta os limites de integração e peculiaridades da função integrando. Essa função passa pelo processo de análises léxica e sintática para construção da árvore sintática, parte integrante do processo de avaliação da função. Logo após este passo, o gráfico da função integrando é construído para a melhor visualização do comportamento da função dentro do intervalo de integração especificado. Neste ponto é processada a análise da função com detecção de pontos críticos como assíntotas e pontos de máximo e mínimo. Esta pode ser feita de forma interativa através da exibição dos valores correspondentes da função e de sua derivada primeira em qualquer ponto solicitado pelo usuário.

Essa análise é fundamental para que se possa classificar o problema e por conseguinte orientar o sistema na escolha do algoritmo de resolução.

Uma vez determinada a classe na qual se encaixa o problema especificado pelo usuário, a decisão por um dos algoritmos dentro de uma classe é feita com base na eficiência dos mesmos. Esta eficiência é medida em função de dois parâmetros: tempo de processamento e precisão dos resultados. O usuário fornece pesos (percentuais) a serem atribuídos a cada um desses parâmetros, em função dos quais o sistema calcula o **coeficiente de produtividade** de cada algoritmo da classe e decide-se pelo de melhor rendimento. O coeficiente de produtividade é calculado como:

CP = Ptemp * Pesot + Ppres * Pesop

- **Ptemp** e **Pprec** são os pesos atribuídos ao tempo de processamento e precisão dos resultados.

- **Pesot** e **Pesop** são os fatores de desempenho dos algoritmos correspondentes a tempo e precisão dos resultados obtidos com base no comportamento médio relativo dos algoritmos na resolução de problemas anteriores. Este fatores são extraídos do **Arquivo de Pesos**, descrito na próxima seção.

4.6 O Arquivo de Pesos

Os algoritmos para a resolução de integrais são classificados em duas classes bastante distintas: os de finalidade geral e aqueles que apresentam funções-peso específicas ou tratamento para singularidades. Deste modo, um algoritmo que se mostra eficiente para um determinado tipo de função não pode sê-lo para outro tipo. Até mesmo a mudança de ambiente computacional (hardware + sistema operacional + compilador) pode influir no desempenho dos algoritmos [36]. De tal forma, não se tem uma regra geral para classificar o desempenho dos algoritmos, principalmente no que diz respeito ao tempo de processamento e a precisão dos resultados. Observando esse comportamento é que se procurou dar ao sistema desenvolvido a capacidade de se ajustar ao ambiente a que se destina, baseando esse ajuste na observação do comportamento dos algoritmos em casos passados, devidamente registrados pelo próprio sistema, traduzido sob forma de pesos no Arquivo de Pesos.

4.7 Método de Aprendizagem Automática

O arquivo de pesos é composto de registros agrupados por classe de problemas. Cada registro corresponde a um algoritmo aplicável a uma classe de problemas e contém os pesos relativos ao respectivo algoritmo em termos de tempo de processamento e precisão dos resultados.

Devido a falta de maiores informações sobre o comportamento dos algoritmos e pela diversidade de aplicações a que se destina o sistema, procurou-se gerar informações a partir da simulação de casos reais. Desta forma, todos algoritmos partem inicialmente com o mesmo Arquivo de Pesos. Com a utilização eles vão se diferenciando, e as informações contidas no Arquivo de Pesos originais vão sendo pouco a pouco substituídas por novas informações extraídas do Arquivo de Ocorrências, o qual é gerado pelo sistema para ali armazenar as informações sobre o desempenho dos algoritmos nas aplicações particulares dos usuários. A atualização do Arquivo de Pesos é feita automaticamente pelo Módulo de Aprendizado após constatar, em exame completo do Arquivo de Ocorrências, a existência de um número X (determinado pelo usuário) de registros de casos semelhantes.

Seja a criação do Arquivo de Pesos original a partir das informações oriundas dos testes simulados, ou na atualização deste a partir das informações obtidas com o processamento de casos reais, o processo é feito da mesma forma segundo o algoritmo abaixo:

1. Identificar todos os casos semelhantes (Algoritmos que resolvem mais eficientemente a integral de uma determinada função) no Arquivo de Ocorrências.

2. Para cada algoritmo, calcular a média dos tempos de processamento gastos e precisão dos resultados.

3. Calcular a média global de tempo e precisão dos resultados entre todos os algoritmos da mesma classe.

4. Calcular o desvio relativo de cada algoritmo, dividindo as médias de tempo, precisão dos resultados referentes ao algoritmo, pela média global respectiva. O resultado serão os novos pesos do algoritmo.

5. Identificar no Arquivo de Pesos o registro correspondente à classe de problema e ao algoritmo, substituindo os valores existentes pelos novos pesos.

6. Apagar as informações processadas no Arquivo de Ocorrências, colocando em seu lugar as médias obtidas por cada algoritmo. Tais médias representarão um caso a mais no próximo ajuste.

4.8 Interface do Sistema

O sistema INTEGRE possui um ambiente integrado que possibilita ao usuário um fácil acesso à todas as ferramentas das quais ele necessita. O menu principal do integre é formado pelos seguintes submenus: *Função*, *Gráfico*, *Algoritmos* e *About*.

No menu *Função* é realizado a entrada e o arquivamento das funções que serão introduzidas no sistema. No menu *Gráfico*, o usuário poderá modificar as configurações do gráfico da função como as cores, o espaçamento dos pontos, redesenhar e limpar o gráfico. No menu *Algoritmos* estão concentradas todas as informações referentes aos algoritmos de integração numérica, entre elas pode-se citar: parâmetros dos algoritmos (erro absoluto, erro relativo, etc.), ver pontos críticos, imprimir os relatórios e solicitar que o sistema calcule a integral.

Há ainda na janela principal do INTEGRE, uma barra de ferramentas que possibilita o acesso rápido a todas as funções do sistema.

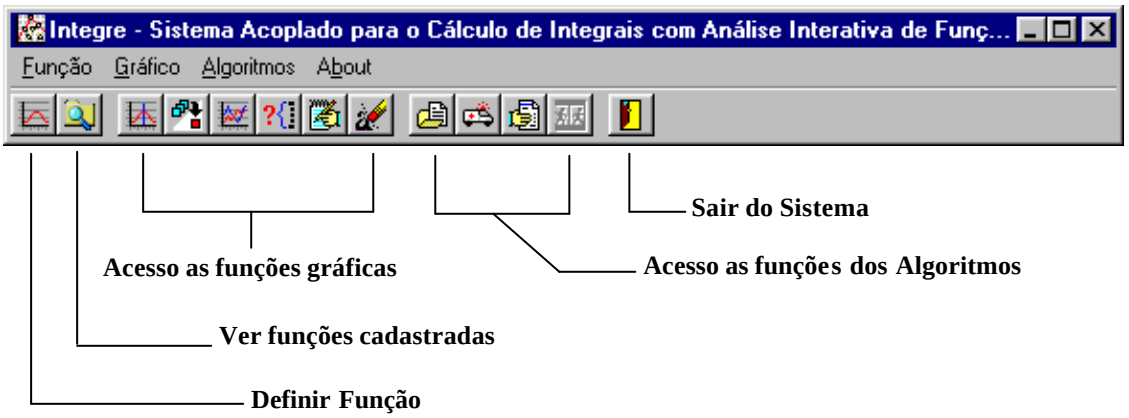


Figura 4.6: Janela principal do sistema INTEGRE

4.8.1 Definindo uma nova Função

Para definir uma nova função basta clicar no menu *Funções* e depois selecionar *Definir Função*, ou então, através do botão  . Após isso, aparecerá a tela apresentada pela figura (4.7) onde deverá ser escolhido o tipo da função.

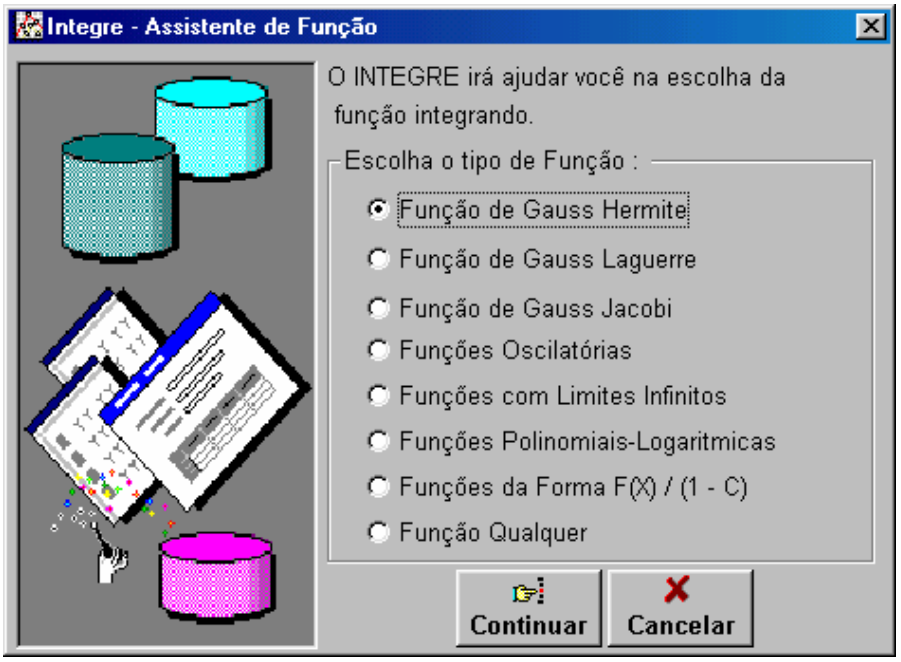


Figura 4.7: Assistente de função do sistema INTEGRE

Para cada tipo de função, o sistema evoca *o assistente de função* que exibe um conjunto de informações acerca daquela função. Fazem parte desse conjunto: a função-peso, os limites de integração e a forma final da integral que será calculada pelo sistema.

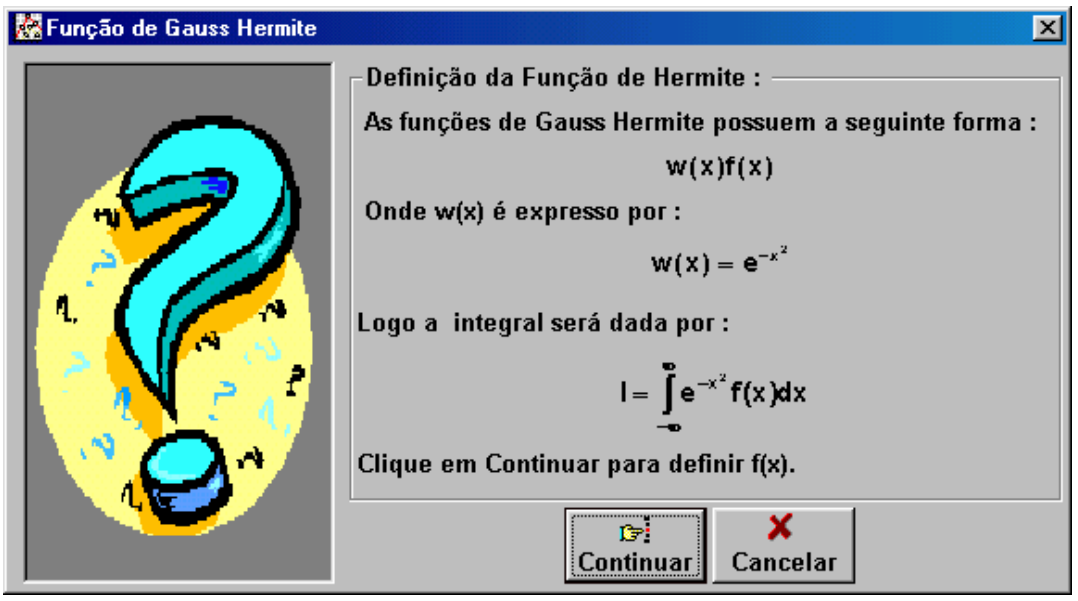


Figura 4.8: Assistente de função → Função de Gauss-Hermite

A figura (4.8) mostra as informações da função de Hermite. Clicando no botão *continuar* o sistema exibe a tela mostrada pela figura (4.9).

Pela figura (4.9), a função-peso já aparece no canto superior permitindo que o usuário entre apenas com o restante da função integrando.

Como foi observado a função de Hermite já possui os limites de integração pré-definidos, ou seja, -∞ a +∞. Os limites que são pedidos na tela da figura (4.9) servem somente para construção de gráfico. Na parte inferior da figura (4.9) o usuário entra com os pesos relativos a precisão dos resultados e ao tempo de processamento. Em caso de existir mais de um algoritmo para resolver a integral, aquele que tive o melhor coeficiente de produtividade será acionado. Os botões auxiliares de entrada (.....) servem para que o usuário possa entrar com os valores dos limites de integração e com os pesos. Acionando qualquer botão auxiliar de entrada o sistema expõem a tela apresentada pela figura (4.10).

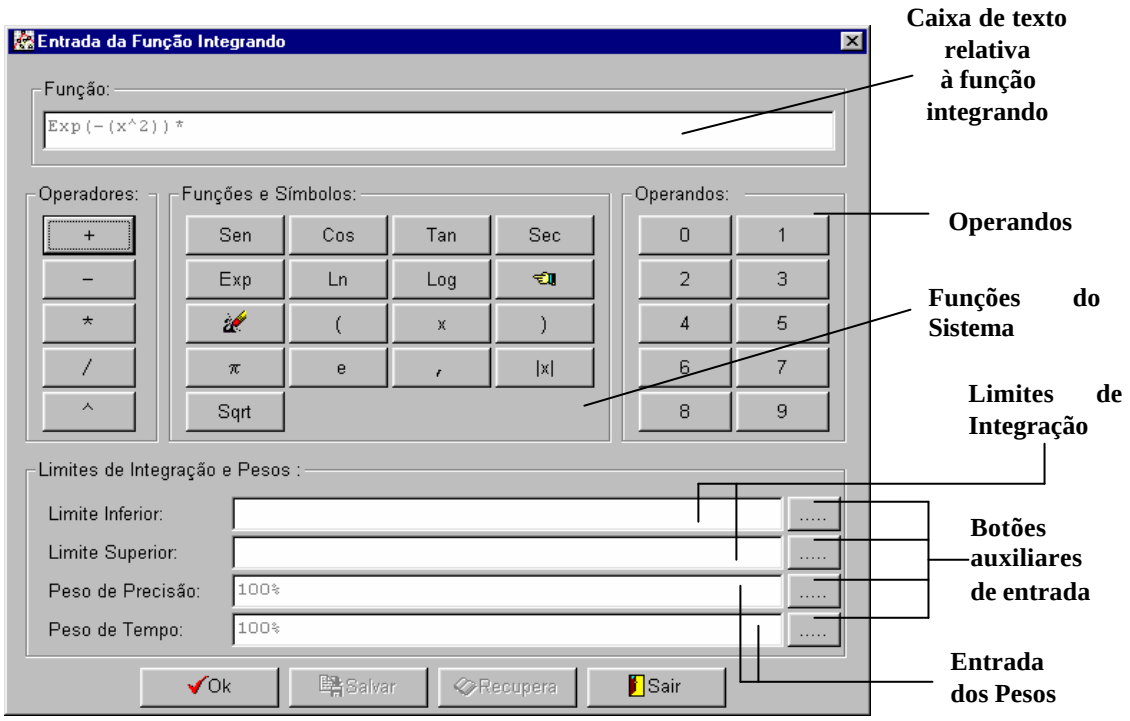


Figura 4.9: Entrada da função integrando

A tela mostrada pela figura (4.10) é uma constante no sistema INTEGRE, pois todas as vezes que existe a necessidade da entrada de parâmetros auxiliares, o sistema possibilita ao usuário a opção de evocá-la através dos botões auxiliares de entrada. Como esta tela pode ser acionada em diversos contextos dentro do sistema, a diferença ficará apenas nos rótulos dos títulos.

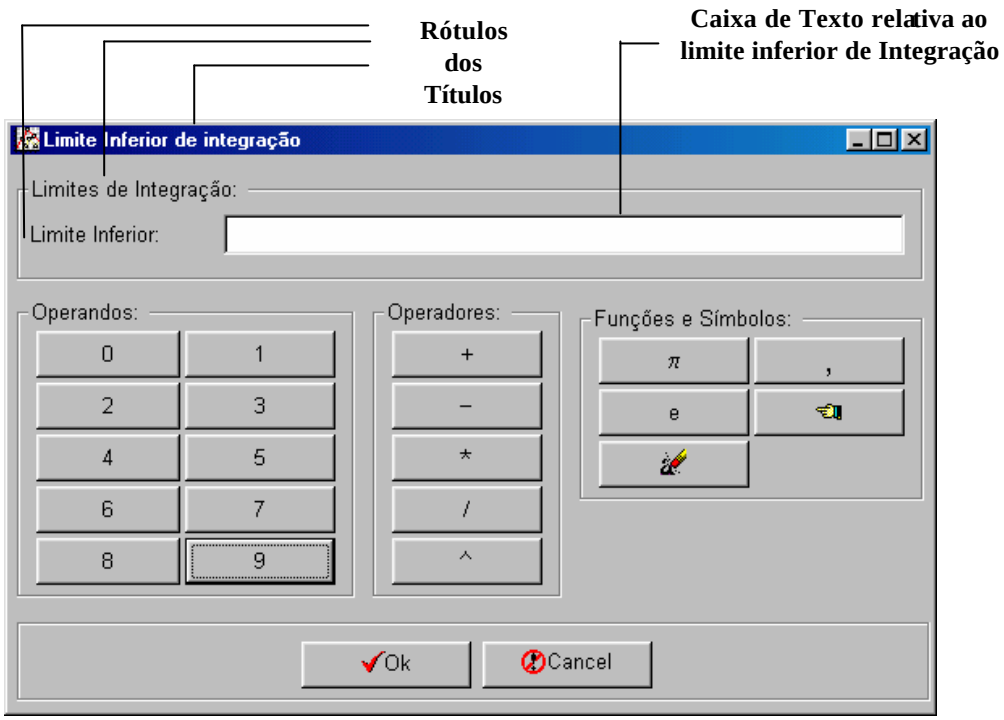


Figura 4.10: Tela para entrada de valores acionada pelos botões auxiliares de entrada

Os botões exibidos pelas figuras (4.11) e (4.12) e presentes nas figuras (4.9) e (4.10) servem respectivamente para apagar o último caractere fornecido e para limpar todo o conteúdo das respectivas caixas de texto. Se houver alguma função-peso pré-definida, esta não será apagada.

Figura 4.11: Apagar o último caractere fornecido

Figura 4.12: Limpa o conteúdo da caixa de texto

De forma análoga a função de Gauss-Hermite, todas os outros assistentes seguem esta filosofia.

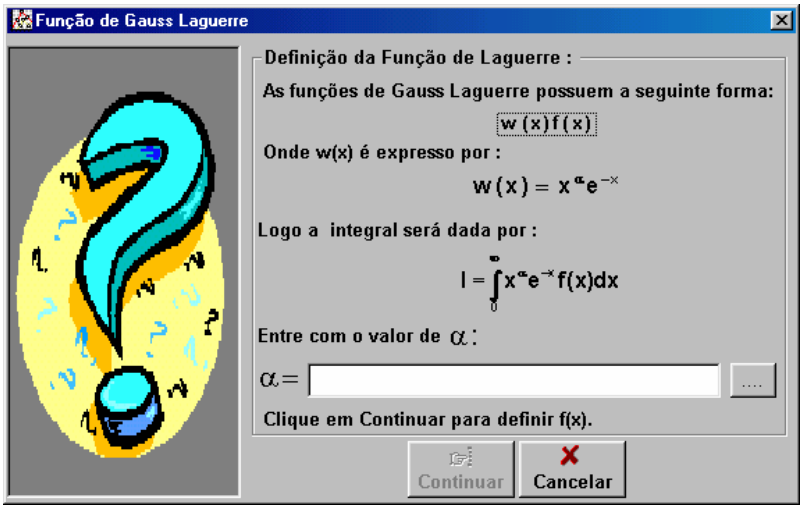


Figura 4.13: Assistente de função → Função de Gauss-Laguerre

A definição da função de Laguerre pode ser vista pela figura (4.13). Pode-se observar a existência de um botão auxiliar de entrada (.....) para que o valor de α possa ser inserido. Logo após a entrada de α, o botão *continuar* ficará habilitado permitindo que o usuário possa definir o restante da função integrando. Uma tela igual a mostrada pela figura (4.9) aparecerá. A única diferença reside no fato de que a função-peso será x^{α} Exp(-x) ao invés de Exp($-x^2$) e que o limite inferior de integração já estará definido (seu valor será 0 (zero)). O sistema também exige a entrada do limite superior de integração, sendo este, porém, utilizado apenas para a construção do gráfico, pois a integral será calculada de $[0, +\infty)$.

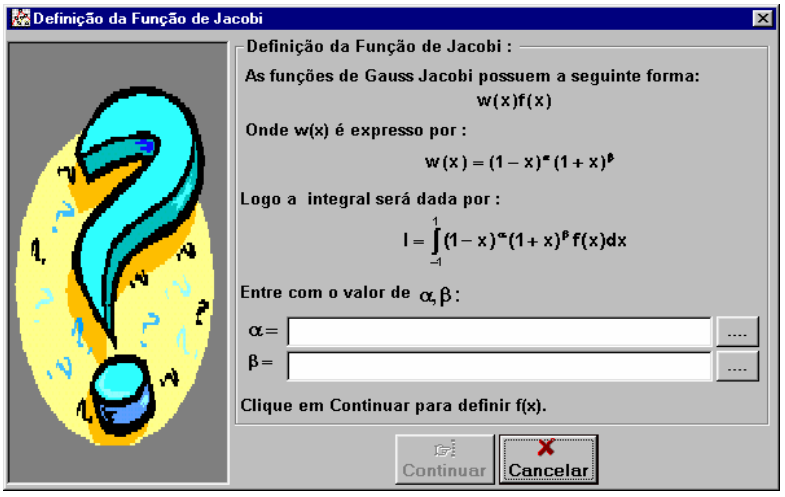


Figura 4.14: Assistente de função → Função de Gauss-Jacobi

O assistente da função de Gauss-Jacobi (apresentado pela figura (4.14)) necessita da entrada de α e β. Observe a existência dos botões auxiliares de entrada que orientam o usuário na entrada de α e β. Logo após a entrada desses valores, o botão *continuar* ficará

habilitado e o usuário poderá entrar com o restante da função integrando através da tela semelhante a apresentada em (4.9). A função integrando já terá $(1-x)^{\alpha}$ $(1+x)^{\beta}$ como parte integrante e os limites de integração já estarão definidos [-1, 1].

A figura (4.15) mostra a tela de entrada para funções oscilatórias. Nesta tela o usuário escolhe a função-peso, o intervalo de integração e entra com o valor de ω. Todos estes valores serão passados como parâmetros para a respectiva rotina de integração. Após a entrada de ω, o botão *continuar* ficará habilitado para a entrada dos parâmetros restantes. Novamente, uma tela semelhante a apresentada pela figura (4.9) aparecerá com a função-peso escolhida. Se o intervalo de integração for]0,+∞[, o limite inferior da tela (4.9) ficará 0 (zero) e o sistema necessitará que o limite superior seja especificado. Vale lembrar que, este limite só servirá para construção do gráfico, pois a integral será calculada de 0 (zero) a + ∞. Se o usuário selecionar a opção *intervalo qualquer* da figura (4.15), o sistema exigirá que ele entre com os limites de integração.

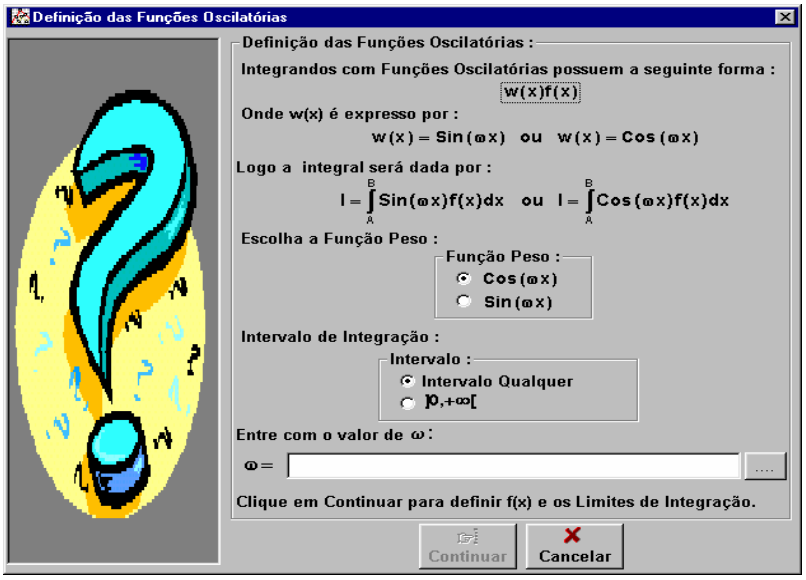


Figura 4.15: Assistente de função → Funções Oscilatórias

A figura (4.16) mostra a tela para entrada de funções com os limites infinitos. Clicando em *continuar*, a tela da figura (4.9) será mostrada com a caixa de texto relativa à função integrando vazia, pois não existe nenhuma função-peso pré-definida. Se o usuário escolher a opção]-∞, B], o sistema INTEGRE necessitará da entrada do parâmetro B (limite superior de integração) para enviá-lo à rotina de integração. O limite inferior também será pedido sendo o seu uso apenas para a construção do gráfico. De forma semelhante ocorre com a segunda opção, [A, +∞[, sendo apenas o valor A (limite inferior de integração) passado para

a rotina de integração. Na terceira opção os dois limites são pedidos e usados apenas para a construção do gráfico, sendo a integral calculada de $-\infty$ a $+\infty$.

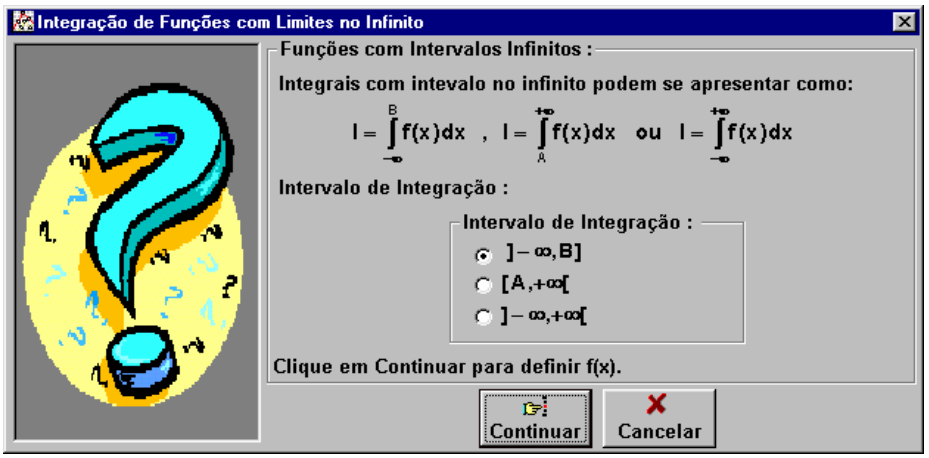


Figura 4.16: Assistente de função → Funções com limites no infinito

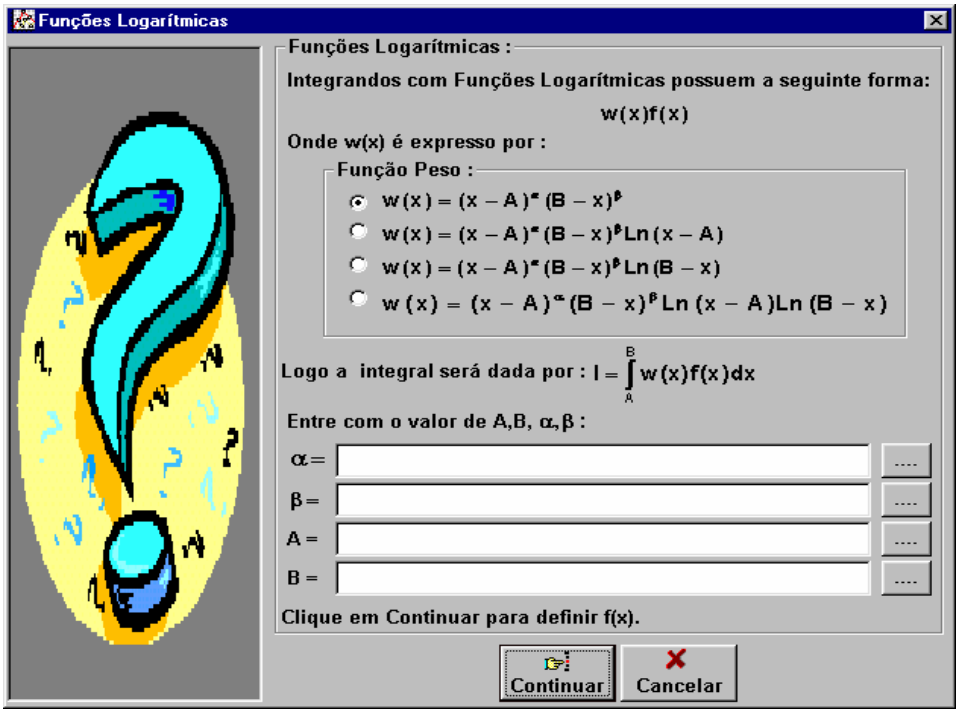


Figura 4.17: Assistente de função → Funções Polinômiais-Logarítmicas

A figura (4.17), mostra a tela de entrada das funções logarítmicas. O usuário escolhe a função-peso e entra com os valores de α, β, A, B. Observe que A e B são, respectivamente, os limites inferior e superior de integração. Clicando em *continuar* é exibida a tela mostrada pela figura (4.18).

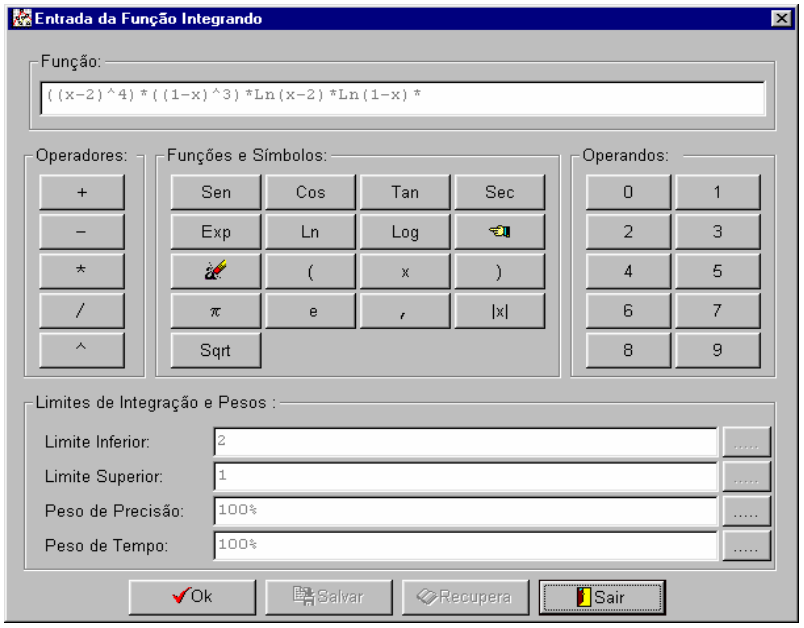


Figura 4.18: Entrada da função Integrando

Na figura (4.18), os limites de integração já ficam definidos e a função-peso já está composta na parte superior, restando apenas ao usuário especificar o seu complemento.

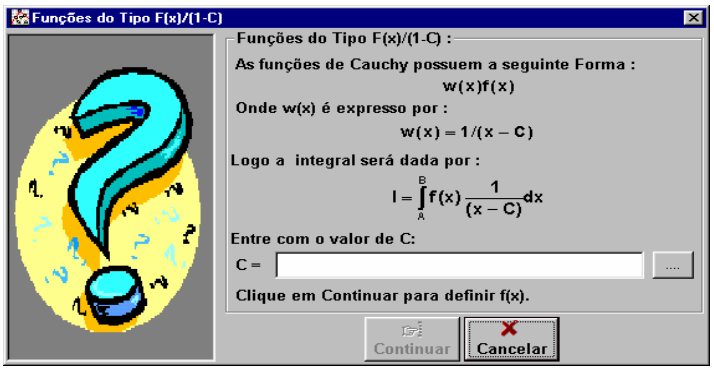


Figura 4.19: Funções da forma F(x)/(x - C)

Na figura (4.19), o usuário deve entrar com o valor de C (ponto de singularidade) e depois clicar em *continuar* para definir o restante da função integrando e os limites de integração.

Se o usuário desejar entrar com uma função qualquer, este deve selecionar o último item da tela exibida na figura (4.7). Logo após clicar no botão *continuar* a tela apresentada pela figura (4.20) aparecerá.

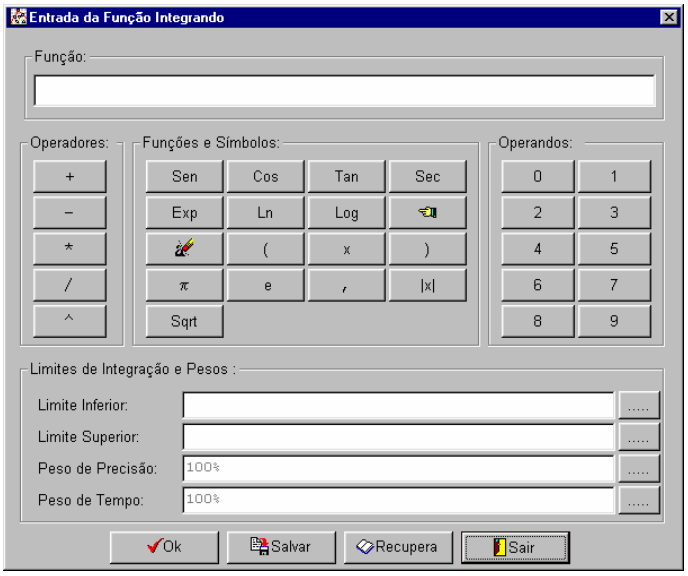


Figura 4.20: Entrada de uma função qualquer

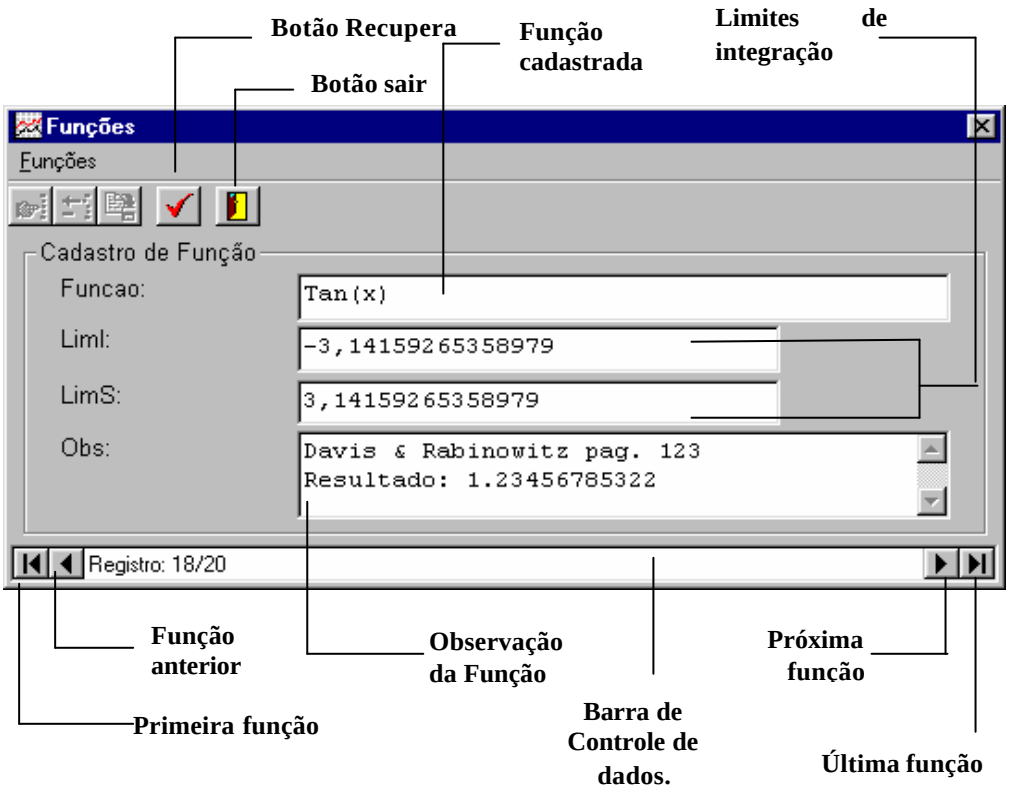


Figura 4.21: Tela de recuperação de função

O usuário poderá entrar com a função integrando, os limites de integração e os pesos da forma que desejar, ou recuperar uma função previamente cadastrada através do botão *recupera*. Ao entrar com a função, o usuário possui a opção de salvá-la no banco de dados. As figuras (4.21) e (4.22) mostram, respectivamente, as telas de recuperação e salvamento.

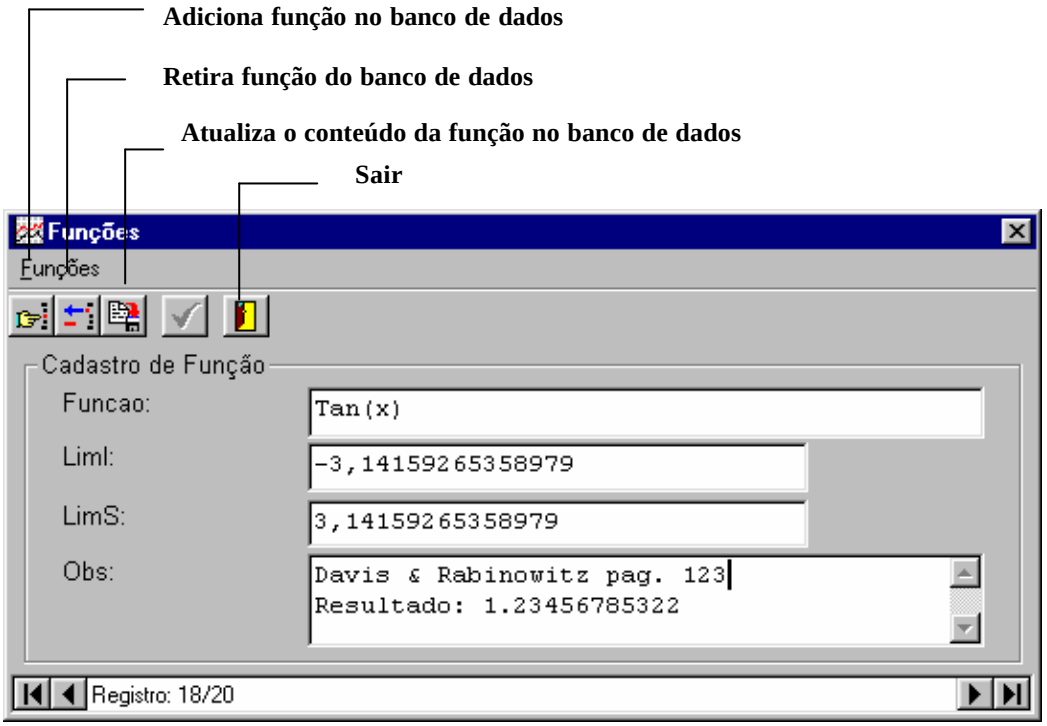


Figura 4.22: Tela de cadastro de função

Na parte superior da figura (4.21), percebe-se que a maioria dos botões ficam desabilitados e somente serão utilizados no caso do usuário desejar cadastrar uma função. Na barra de controle de dados o usuário pode navegar pelas diversas função já cadastradas no sistema.

Na figura (4.22), o campo da função e os limites de integração são obtidos pelo sistema direto da tela mostrada pela figura (4.20). O conteúdo do campo observação é editado pelo o usuário no momento de adicionar a função no banco de dados, ou senão, este campo pode ser editado a qualquer momento bastando o usuário clicar em  para que esta informação seja atualizada.

Se o usuário desejar visualizar todas as função cadastradas no sistema basta clicar no menu *função* e depois em *Consultar funções*, ou através da barra de ferramentas pelo botão  .

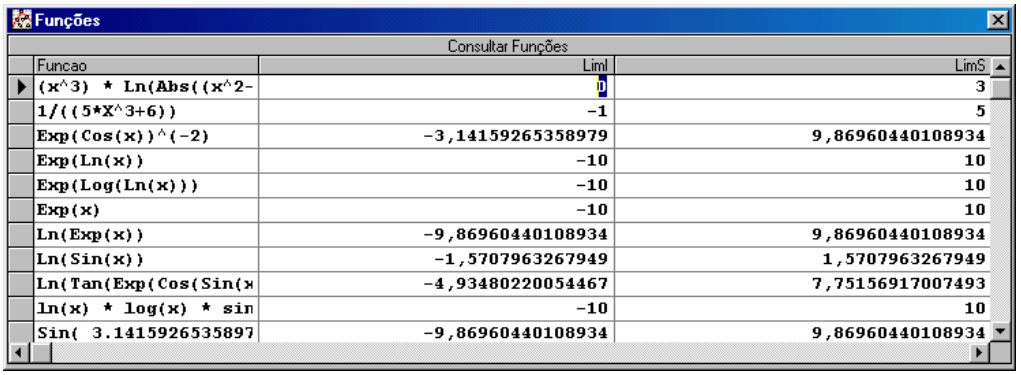


Figura 4.23: Funções cadastradas no sistema

4.8.2 Construção do gráfico da função e a análise interativa com o usuário.

Logo após a confirmação da entrada da entrada da função, o usuário terá a oportunidade de interagir com o sistema através do gráfico que é exibido. A figura (4.24) mostra o gráfico da função $Ln(abs(x^2-3))$.

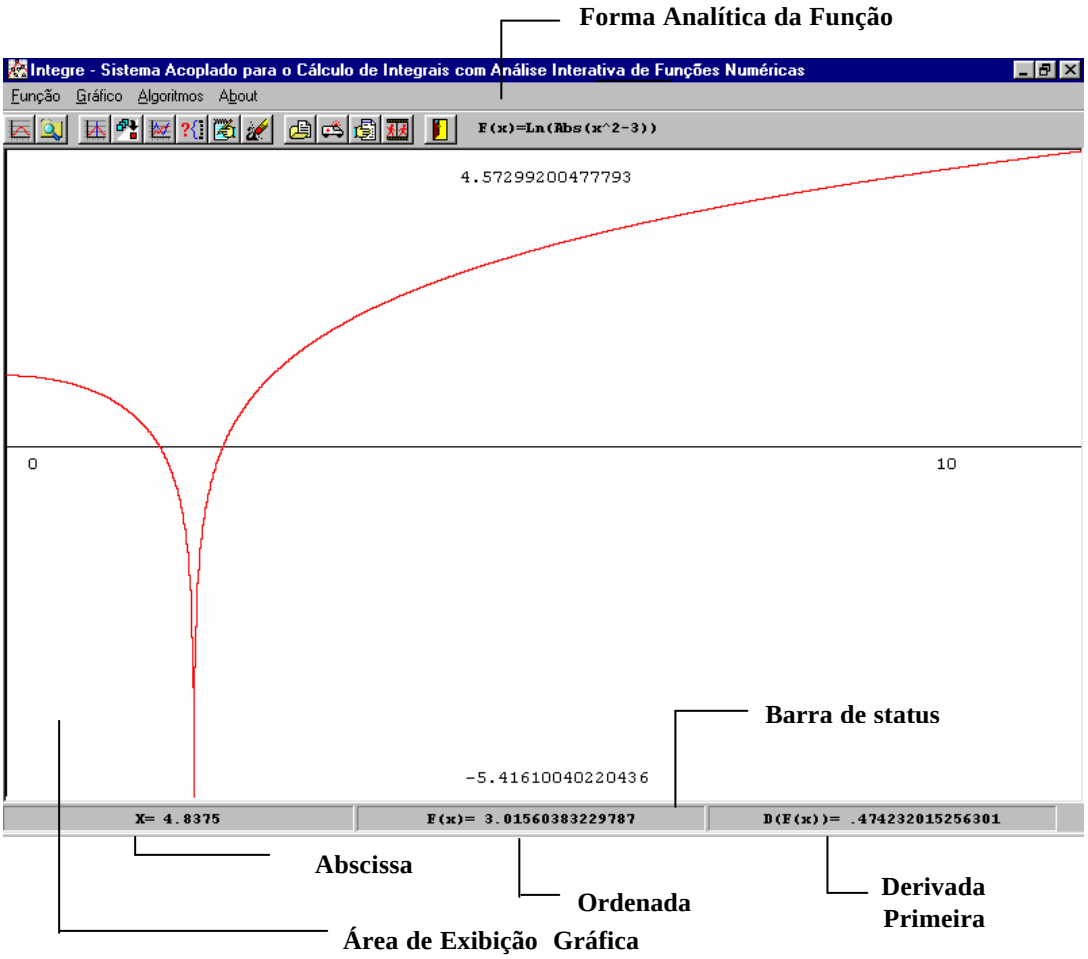


Figura 4.24: Visualização gráfica no sistema Integre

A cada ponto da área de exibição gráfica que o mouse passar, o sistema exibe na barra de status a abscissa, a ordenada e o valor da derivada primeira para uma análise por parte do usuário. No canto superior da figura (4.24) é mostrada a forma analítica da função.

O usuário pode ampliar (ZOOM) uma determinada área conforme o seu desejo para uma análise mais aprofundada. Para isso basta clicar no botão esquerdo do mouse e um retângulo irá aparecer com a extremidade inferior seguindo a ponta do mouse, indicando assim, a área de ZOOM.

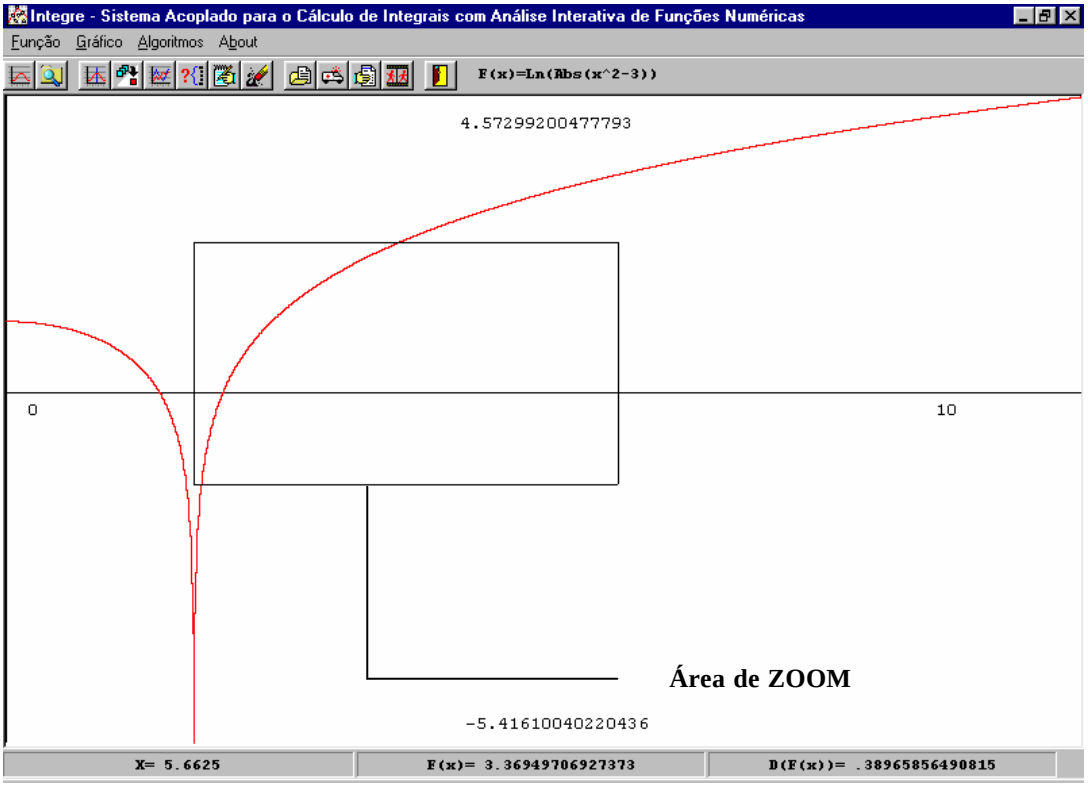


Figura 4.25: ZOOM no sistema Integre

O usuário pode alterar os parâmetros de configuração do gráfico através do menu *gráfico*. Como pode-se observar pela figura (4.25), a maior ordenada que o sistema calculou foi 4.57299200477793. O sistema INTEGRE permite que este valor seja modificado conforme o desejo do usuário. Para isso basta acessar o menu *gráfico* e logo após o submenu *Definir ordenada máxima*, ou então, através do botão  .

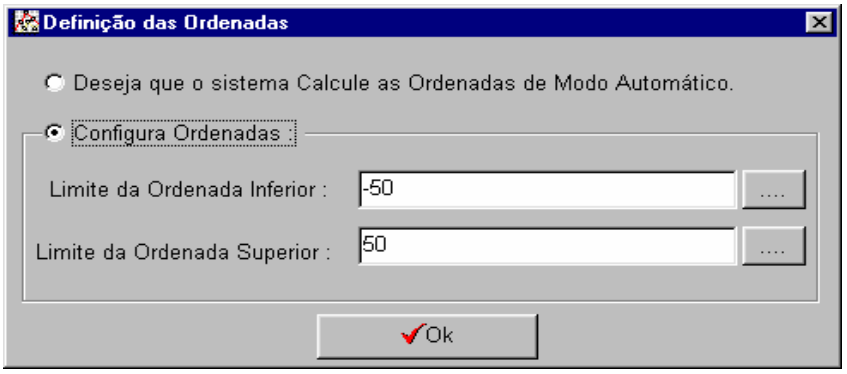


Figura 4.26: Entrada dos limites da ordenada

A figura (4.26) mostra a tela de configuração dos limites da ordenada. Se a opção *deseja que o sistema calcula as ordenadas de modo automático* estiver selecionada o sistema calcula de modo automático as ordenadas sem limitantes. Caso contrário, o usuário pode definir limites para esse cálculo e o sistema exibirá o gráfico dentro dessa especificação, ou seja, se a ordenada superior ou inferior ultrapassar os limites estabelecidos pelo usuário, o sistema INTEGRE ignora estes valores. Na figura (4.26), pode-se observar a presença dos botões auxiliares de entrada, que ajudam o usuário na especificação desses valores.

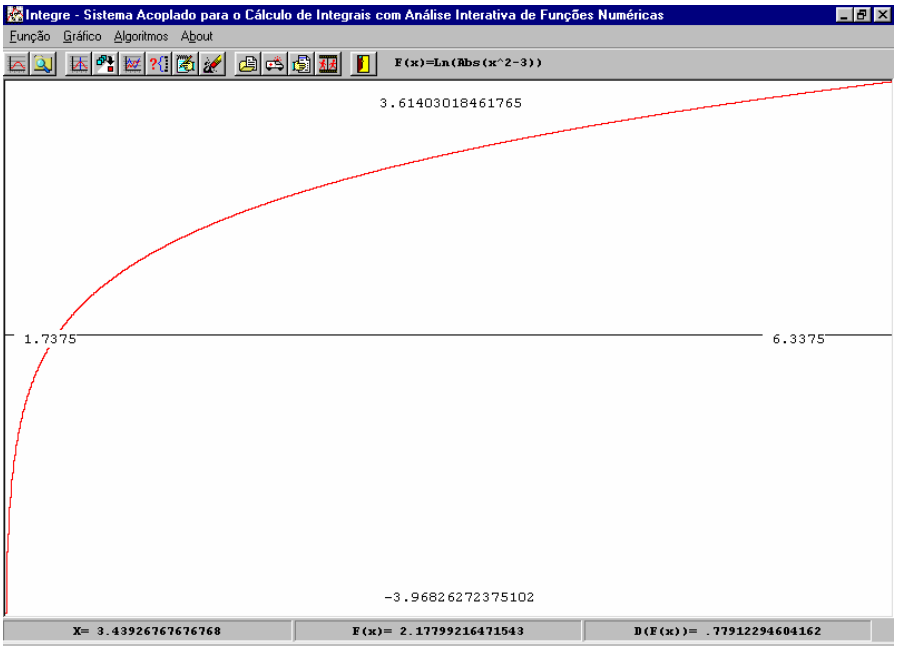


Figura 4.27: ZOOM da figura (4.25)

Pelas figuras (4.25) e (4.27), pode-se perceber que o gráfico é construído através de pontos que são plotados na área de exibição gráfica. O usuário pode alterar o espaçamento desses pontos, e também, pode escolher se estes pontos vão ser ligados ou não. Para definir o espaçamentos dos pontos, bastar clicar no submenu *gráfico* e logo após em *Definir*

Espaçamento de Pontos, ou através do botão  . Depois disso, a tela apresentada pela figura (4.28) aparecerá.

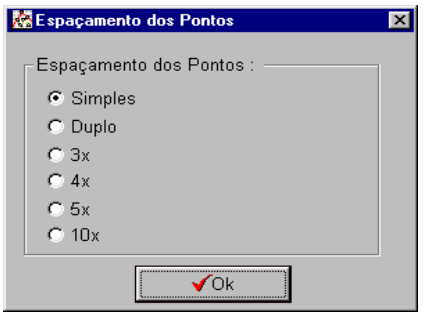


Figura 4.28: Definir espaçamento de Pontos

Na figura (4.28), o sistema exibe uma série de opções sobre o espaçamento dos pontos. A opção *Simples* é a padrão. Se o usuário marcar a opção *Dupla*, o sistema, na hora de construir o gráfico, avaliará a função no dobro de pontos da opção *Simples*, fazendo com que eles fiquem mais próximos para o intervalo especificado. Vale lembrar que, diminuindo o espaçamento entre os pontos o sistema construirá o gráfico de forma mais lenta. A outras opções avaliam a função em 3, 4, 5 e 10 vezes mais pontos que na opção *Simples.* O usuário confirma o seu desejo através do botão *OK*.

O usuário ainda pode configurar se os pontos vão ser ou não interligados. Para isso, bastar clicar no submenu *Gráfico* e logo após em *Ligar Pontos*, ou através do botão  . A tela apresentada pela figura (2.29) aparecerá.

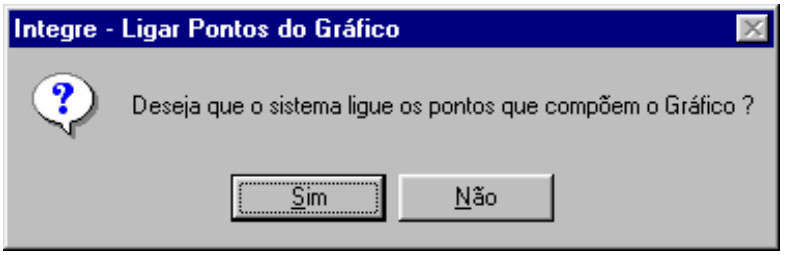


Figura 4.29: Ligar Pontos do Gráfico

O efeito dessas mudanças pode ser visualizado solicitando-se que o sistema redesenhe o gráfico da função. Para isso, bastar clicar no submenu *Gráfico* e logo após em *Redesenha Gráfico*, ou através do botão  .

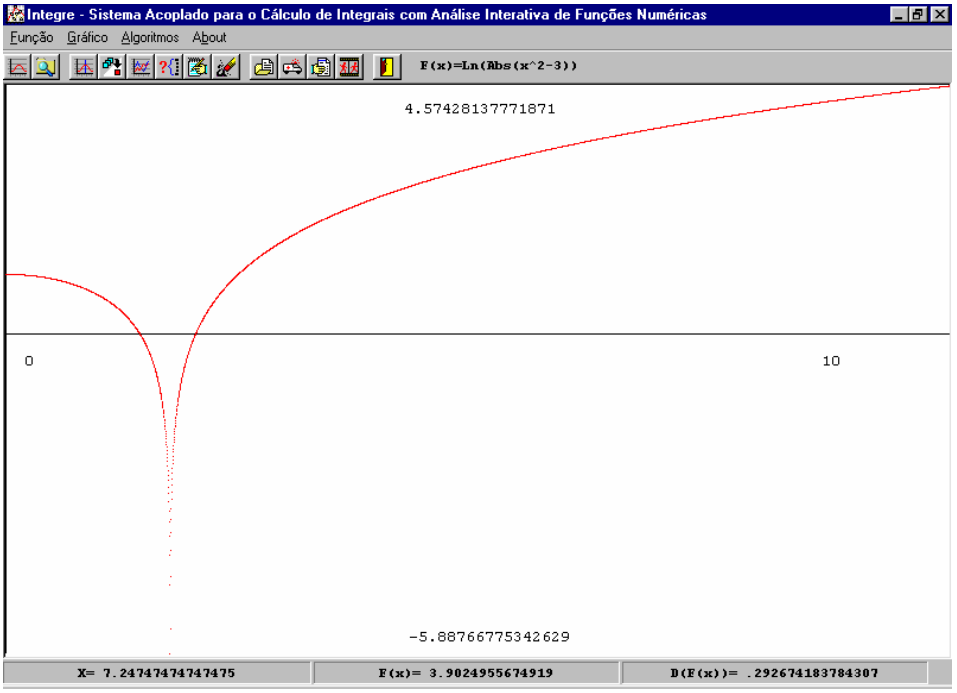


Figura 4.30: Gráfico da Função Ln(Abs(x^2-3)) com espaçamento duplo e os pontos sem estarem ligados

As cores dos objetos que compõem o gráfico da função podem ser alteradas clicando no submenu *Gráfico* e depois em *Configurar Cores* ou através do botão  . A tela apresentada pela figura (4.31) aparecerá.

O sistema possibilita ao usuário mudar as cores do fundo da área de exibição gráfica, a cor dos eixos, a cor do gráfico e da linha de ZOOM. Observe pela figura (4.31) que os botões auxiliares de entrada fornecem um auxílio ao usuário na escolha da cor apropriada. Clicando em algum desses botões, a tela (4.32) aparecerá.

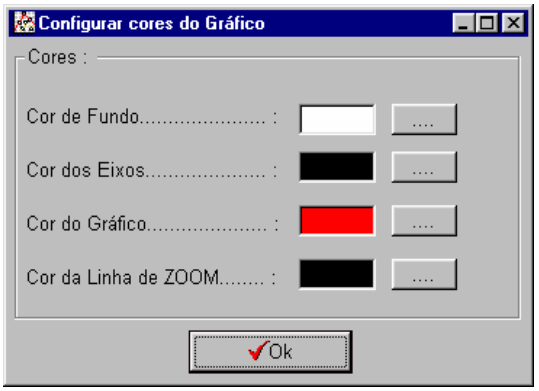


Figura 4.31: Tela de Mudança das cores do Gráfico no Sistema INTEGRE

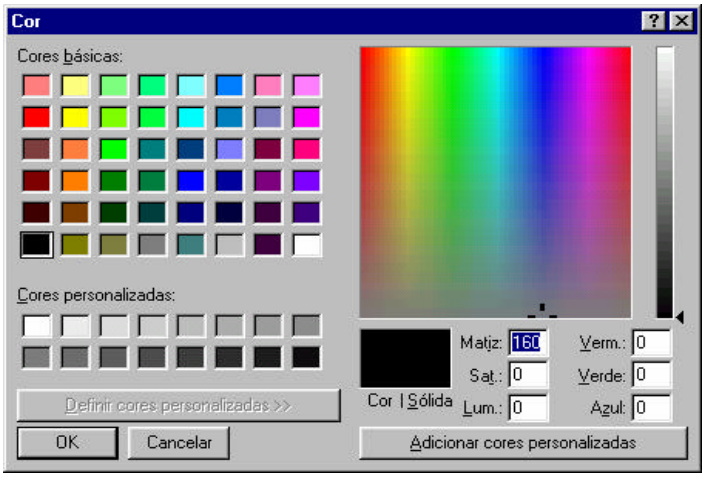


Figura 4.32: Cores disponíveis para mudança

O usuário pode escolher a cor desejada clicando no quadro da respectiva cor, ou então é possível personalizar uma conforme o seu desejo.

Finalmente, o usuário pode limpar a área de exibição gráfica clicando no submenu *Gráfico* e depois em *Limpar Gráfico*. Depois de efetuar estas operações a área de exibição gráfica ficará limpa para definição de uma nova função.

4.8.2.1 A utilização das Derivadas (D(f(x))) na análise matemática preliminar.

Uma análise matemática se faz necessária ao módulo do sistema de cálculo numérico que realiza a escolha da rotina numérica para melhor solução do problema. Dado que, para cada problema existe um algoritmo que é o mais adequado à obtenção da solução, é imprescindível que o sistema tome conhecimento das características e do comportamento da função no intervalo solicitado, que são os fatores críticos no processo de escolha.

O cálculo da derivada da função integrando é de vital importância para a realização da análise matemática preliminar. É a partir da derivada que se pode detectar assíntotas verticais no intervalo dado. Isto é feito através do cálculo das raízes da seguinte equação: $\text{ArcTan}(D(f(x))) = \pi/2$. Dado que o cálculo genérico das raízes de tal equação é um trabalho complicado, optou-se por percorrer o intervalo numérico analisado com um incremento (espaçamento dos pontos que compõem o gráfico da função) e verificar se os valores de $\text{ArcTan}(D(f(x)))$ aproximavam-se de $\pi/2$. Se tal fato ocorre, as coordenadas do ponto avaliado como assintóticas (singularidades) seriam inseridas na tabela de pontos críticos da análise matemática preliminar. Logicamente, tal procedimento leva a aproximações na determinação destes pontos que podem ser confirmados pela análise gráfica interativa.

Mas por que não se utiliza f(x) em vez de ArcTan(D(f(x))) para detectar assíntotas verticais? Tem-se que os valores da imagem de ArcTan(D(f(x))) estão no intervalo $[-\pi/2, \pi/2]$, portanto pode-se ter certeza que uma determinada avaliação de arctan(D(f(x))) não extrapolará os limites da máquina quando submetida a valores muito grandes de x. Isso poderia ocorrer com f(x), pois para detecção de uma assíntota vertical era necessária a verificação de que $f(x) \rightarrow \pm\infty$, causando um erro conhecido como OVERFLOW.

A interface do sistema INTEGRE possibilita a fácil iteração com o usuário para determinação dos pontos assíntoticos. Clicando no botão direito do mouse, em cima do ponto solicitado, o sistema exibe a tela apresentada pela figura (4.33):

Figura 4.33: Detecção de assíntotas no sistema INTEGRE

O INTEGRE exibe os valores da abscissa, da ordenada, da derivada e do arco tangente. Logo após, ele mostra o quanto ArcTan(D(f(x)) se aproxima de $\pi/2$, ou seja, $\text{ArcTan}(D(f(x)) - \pi/2 \rightarrow 0$. Na figura acima esse valor é -1.96598754500366E-02. O usuário pode ampliar a área onde está localizada a assíntota para uma análise mais aprimorada dos resultados e obter assim uma melhor aproximação para $\text{ArcTan}(D(f(x)) - \pi/2$. A abscissa observada pode ser cadastrada como um ponto de assíntota, bastando para isso o usuário clicar no botão *Adicionar*. O botão *exibir* exibe todos os pontos cadastrados como assíntotas e o botão *sair* volta para a tela principal do sistema.

Clicando no botão *exibir* o sistema mostra a tela apresentada pela figura (4.34). Nesta tela, o usuário pode visualizar todas as informações acerca do ponto cadastrado podendo, conforme o seu desejo, excluir este ponto.

O usuário também poderá acessar esta tela (4.34) através do submenu *algoritmos*, e logo depois clicando em *Ver Pontos Críticos*, ou então através do botão  .

Esta parte do sistema é fundamental importância para o módulo de escolha, pois a existência ou não de uma singularidade no interior do intervalo de integração faz com que o método de escolha mude o algoritmo para o cálculo da aproximação da integral.

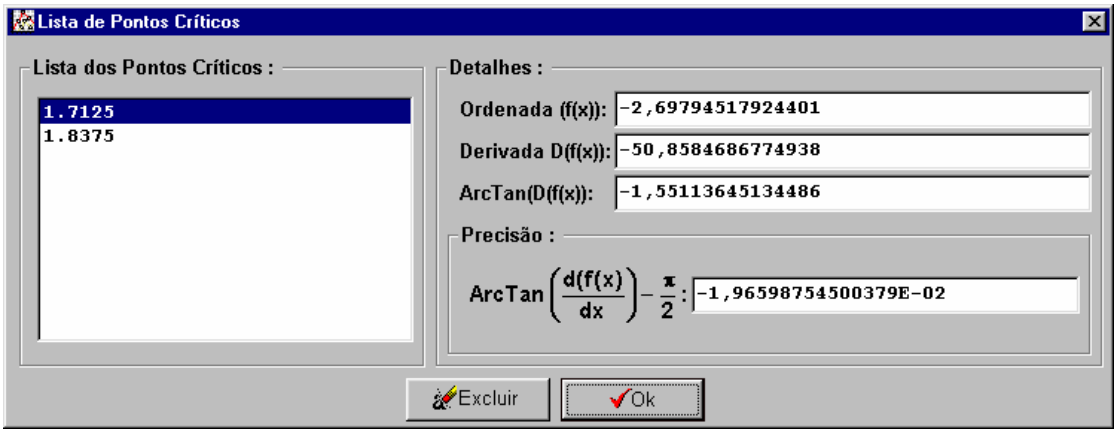


Figura 4.34: Pontos cadastrados como assíntotas

4.8.3 O Cálculo da integral

Nesta parte será mostrado como o usuário deverá interagir com o INTEGRE para obtenção de relatórios referentes ao arquivo de pesos e ao processo de atualização e, principalmente, como obter o valor da aproximação da integral pelo algoritmo escolhido.

4.8.3.1 Definição de Parâmetros

Como foi observado nos capítulos anteriores, os vários algoritmos para o cálculo da integral necessitam de alguns parâmetros para efetuarem o cálculo de maneira correta. A maioria desses parâmetros são coletados pelo o método de escolha, a medida que o usuário vai compondo a função integrando através dos assistentes de função. Alguns outros parâmetros são entrados no sistema pelo próprio usuário. A figura (4.35) exibe uma lista de parâmetros necessários ao cálculo da integral que podem ser modificados.

O usuário pode visualizar o valor desses parâmetro e conforme o seu desejo alterá-los. Para isso, bastar clicar no submenu *Algoritmos* e logo após em *Definir Parâmetro*, ou através do botão  .

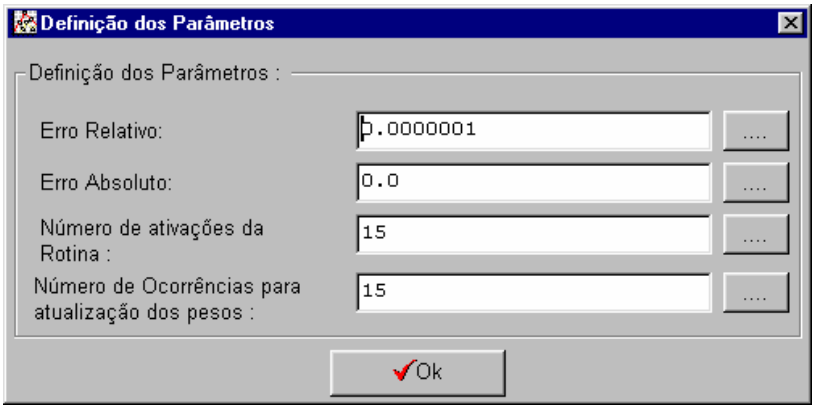


Figura 4.35: Parâmetros para os Algoritmos de Integração

O parâmetro *Número de Ativações da Rotina* possui um significado diferente dependendo do algoritmo de integração utilizado. Nos algoritmos referentes ao NUMERICAL RECIPES que utilizam as fórmulas de Newton-Cotes, tanto abertas quanto fechadas, esse parâmetro limita o número de chamadas que deverão ser efetuadas ao integrador. Como foi visto anteriormente, o integrador é chamado N vezes (N =1,2,3,...) e o valor máximo que ele pode assumir será, no caso da figura (4.35), 15. A cada chamada seqüencial de N, o valor da integral será calculado adicionando mais pontos no interior do intervalo de integração e o valor da aproximação obtido da chamada de M-1, onde M < N, será aproveitado na chamada M para assim se obter uma melhor aproximação para a integral.

No caso das quadraturas Gaussianas, esse parâmetro indica o grau do polinômio ortogonal (Legendre, Laguerre, Hermite e Jacobi) de onde vão ser calculados os pesos e os nós. Os algoritmos pertencentes ao QUADPACK não utilizam este parâmetro.

Outro parâmetro importante para o sistema INTEGRE é o *Número de Ocorrências para Atualização dos Pesos*. Como foi visto na seção (4.7) depois de um número X de registros de casos semelhantes o sistema deve atualizar o arquivo de pesos. Assim, X é determinado pelo valor desse parâmetro.

4.8.3.2 Relatórios do Sistema

O sistema INTEGRE emite alguns relatórios que servem para o usuário se manter informado sobre os valores atuais do arquivo de pesos, do arquivo de ocorrências e da classificação dos algoritmos por classe de problema. Para tal, bastar clicar no submenu *Algoritmos* e logo após em *Imprimir Relatórios*, ou através do botão  . Através do submenu *Algoritmos* o sistema exibe as opções de relatórios apresentada pela figura (4.36):

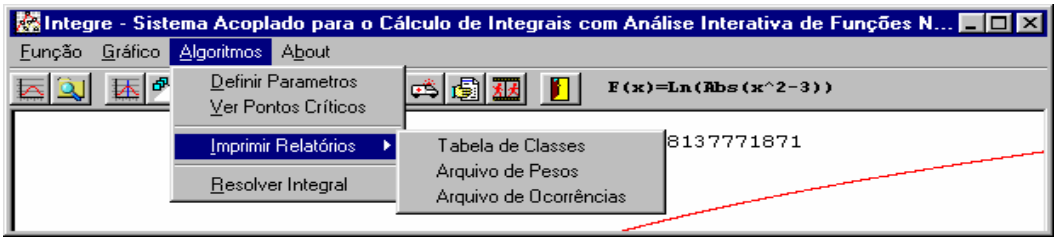


Figura 4.36: Relatórios do sistema INTEGRE

Caso o usuário clique no botão  , o sistema abre a janela exibida pela figura (4.37):

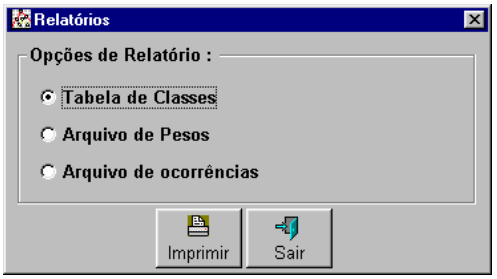


Figura 4.37: Opções de Relatórios do sistema INTEGRE

Para impressão de alguns desses relatórios, basta o usuário selecioná-lo com um clique em cima da descrição do relatório, e logo após clicar em *Imprimir*. A figura (4.38) mostra a tela do relatório *Tabela de Classes*.

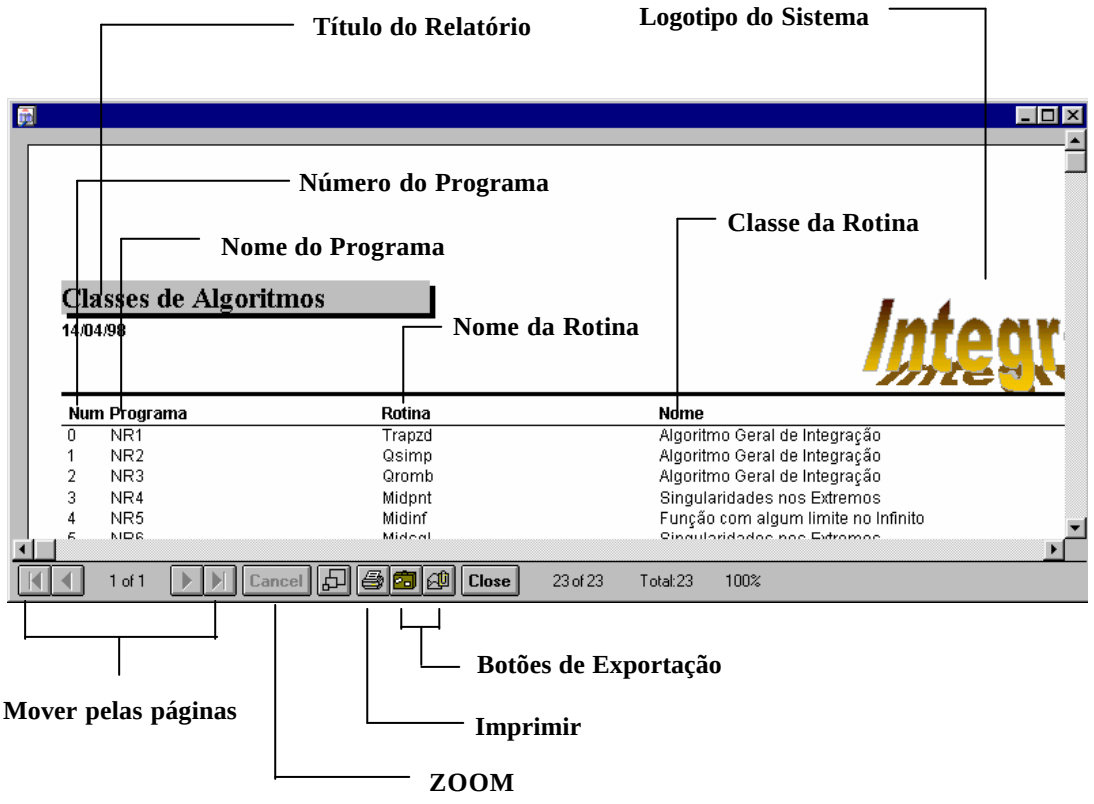


Figura 4.38: Relatório da Classe dos Algoritmos

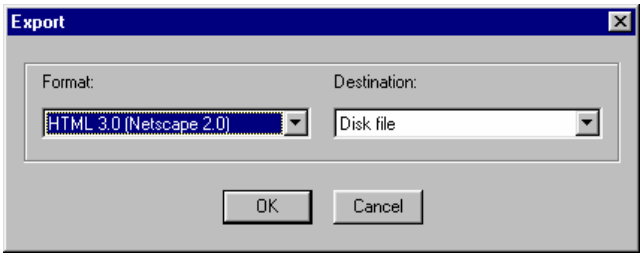


Figura 4.39: Exportação do Relatório para Formato HTML 3.0

As telas dos relatórios *Arquivo de Pesos* e *Arquivo de ocorrências* são mostradas pelas figuras (4.40) e (4.41):

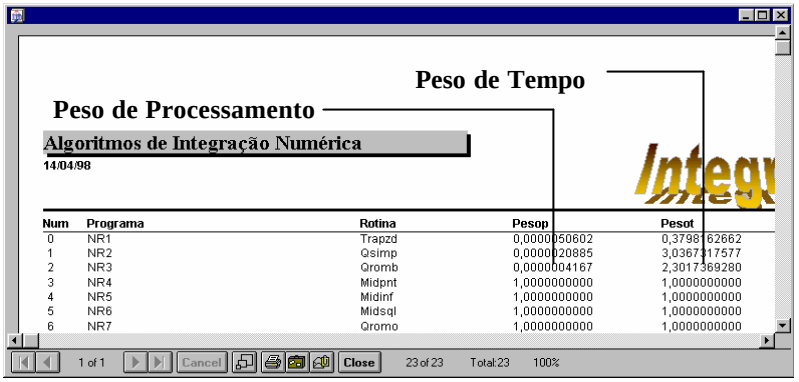


Figura 4.40: Relatório do Arquivo de Pesos

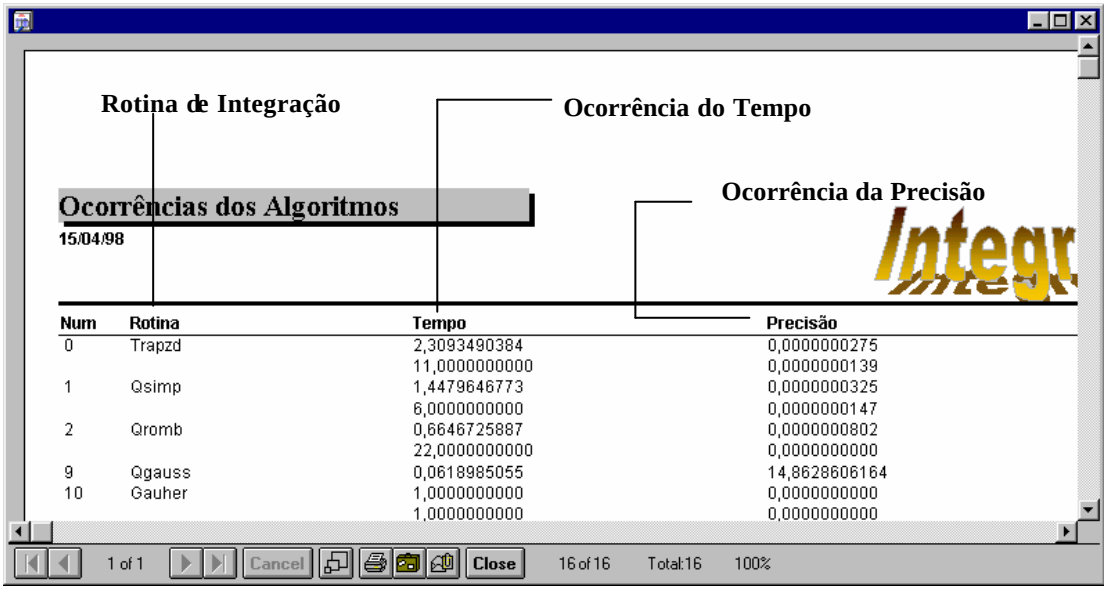


Figura 4.41: Relatório do Arquivo de Pesos

4.8.3.3 Obtendo o valor da integral

O objetivo principal do sistema INTEGRE é a obtenção do valor da integral por um método (o melhor para uma dada situação). Depois de especificar a função integrando e efetuar, junto com o sistema, uma análise iterativa para obtenção pontos críticos, o usuário, finalmente, pode obter o valor da integral clicando no submenu *Algoritmos* e logo depois em

Resolver a Integral ou através do botão  . Efetuando este procedimento a tela apresentada pela figura (4.42) aparecerá:

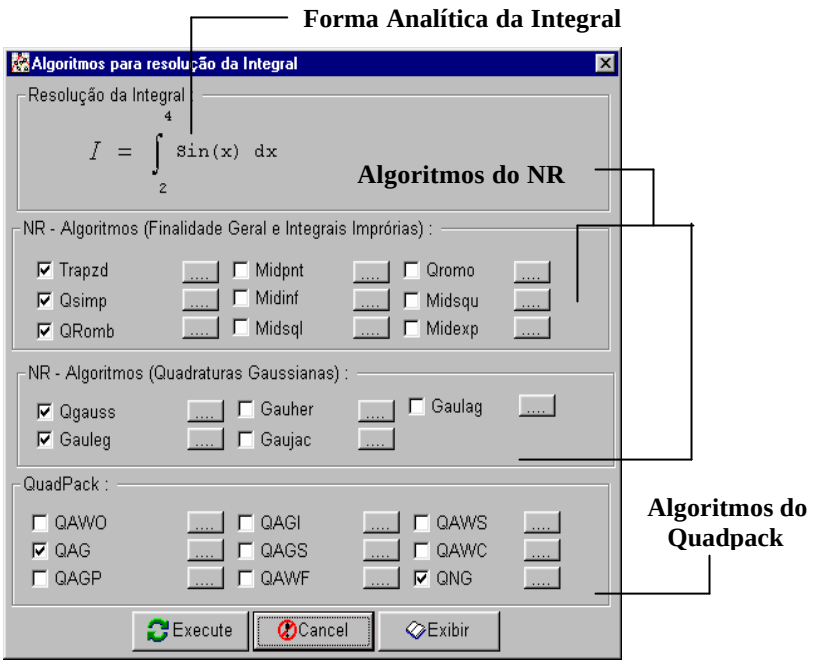


Figura 4.42: Algoritmos para resolução da Integral

Na figura (4.42), os algoritmos escolhidos para resolução da integral estão marcados. Para executar o algoritmo (aquele que tiver o melhor coeficiente de produtividade) o usuário deve clicar no botão *Execute.* O botão *Cancel* volta para a tela principal e o botão *exibir* mostra os resultados dos algoritmos que já resolveram a integral proposta. Nota-se também, a presença dos botões auxiliares de entrada, que nesta parte do sistema, exibem as informações do respectivo algoritmo.

Figura 4.43: Informações dos Algoritmos que estão na base de dados do INTEGRE

A tela apresentada pela figura (4.43) exibe uma série de informações dos algoritmos que estão na base de dados do sistema INTEGRE, entre elas pode-se citar: a classe

que o algoritmo pertence, o programa que utiliza a rotina de integração, o nome da rotina de integração e os pesos relativos ao tempo de processamento e a precisão dos resultados.

Os botões *Próximo* e *Anterior* exibem as informações do restante dos algoritmos que estão na base de dados. Os botões *Imprimir* e *Classes* imprimem, respectivamente, os relatórios mostrados pelas figuras (4.38) e (4.40). O botão *Sair* volta para tela da figura (4.42).

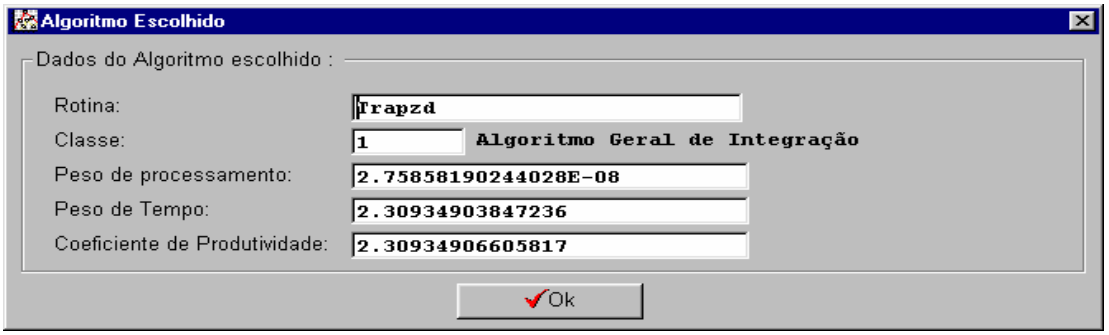


Figura 4.44: Informações do Algoritmo de integração

Clicando no botão execute (figura (4.42)), o sistema exibe uma série de informações do algoritmo escolhido (figura (4.44)), entre elas: o nome da rotina, a classe na qual ela pertence, o pesos de processamento e tempo e o coeficiente de produtividade. Clicando em *OK* será exibida a tela da figura (4.45):

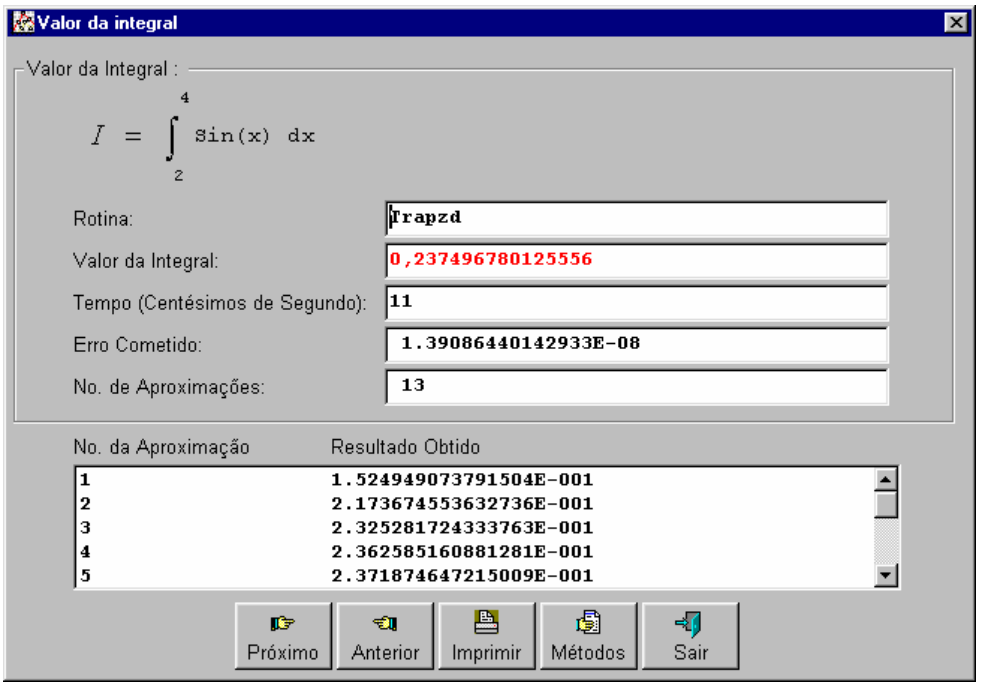


Figura 4.45: Resultado da Integral

A figura (4.45) mostra o resultado da integral calculado pela rotina TRAPZD. Observe que, além do valor da integral, o sistema exibe o tempo (em centésimos de segundo),

o erro cometido, o número de aproximações que a rotina efetuou até que precisão desejada fosse alcançada, além de mostrar o valor de todas as aproximações. Clicando em *Anterior* ou *Próximo* o sistema mostra, se existir, o resultado da integral obtido por outros algoritmos. O botão *Imprimir* imprime todos os resultados mostrados pela tela da figura acima. Se vários métodos já foram utilizados para o cálculo da integral, o botão *Métodos* imprime o resultado de todos esses outros métodos. O botão *Sair* volta para a tela da figura (4.42). Se o usuário desejar que o sistema calcule a integral por outro método de integração basta clicar no botão *Execute*, da figura (4.42) e um outro método será escolhido.

Diferentemente do botão *Execute*, o botão *Exibir* mostra a tela da figura (4.45) sem efetuar a cálculo da integral por outro método de integração.

4.8.3.4 Atualização do Arquivo de Pesos

Uma outra característica importante do INTEGRE e a capacidade de modificar os valores dos pesos de acordo com o sucesso ou insucesso da solução fornecida pelo o método de integração. Quando um número X (determinado pelo usuário) de ocorrências de uma mesma classe for encontrada pelo sistema, ele efetuará a mudança do arquivo de pesos através do módulo de aprendizado automático. No início todos os pesos são iguais a 1 (um) e, pouco a pouco, através do aprendizado automático, eles vão se diferenciando.

Quando o sistema INTEGRE vai efetuar uma atualização no arquivo de pesos a tela apresentada pela figura (4.46) irá aparecer:

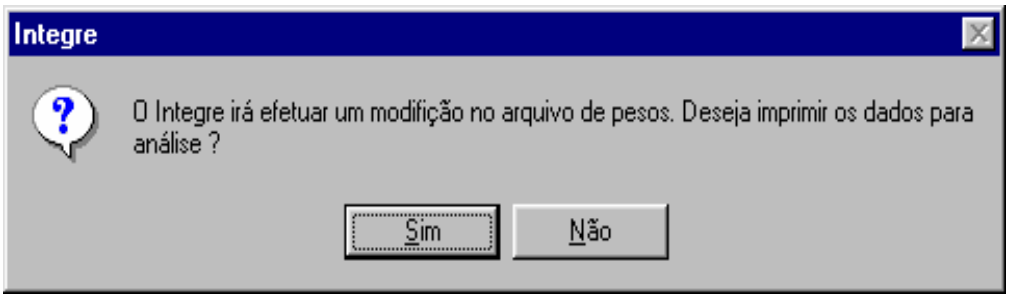


Figura 4.46: Aviso do Sistema ao efetuar uma atualização no Arquivo de Pesos

Se o usuário desejar imprimir os dados, o sistema INTEGRE mostra uma lista de relatórios que o usuário pode, conforme seu desejo, imprimir.

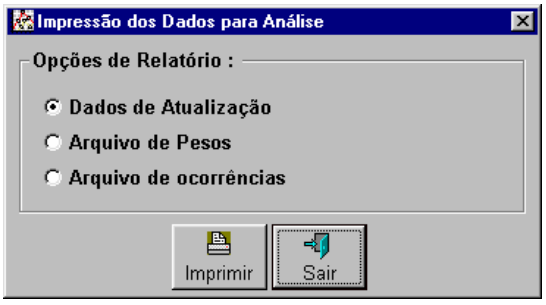


Figura 4.47: Opções de relatórios

Os relatórios *Arquivo de Pesos* e *Arquivo de Ocorrências* são iguais aos apresentados pelas figuras (4.40) e (4.41). O relatório *Dados da Atualização* é mostrado pela figura (4.48):

Figura 4.48: Relatório Dados da Atualização

A principal característica do relatório acima é mostrar os tempos e a precisão de cada algoritmo da mesma classe, sendo que, no final de cada algoritmo, o sistema expõem a média obtida por ele em relação ao tempo de processamento e a precisão dos resultados. Observe que os algoritmos são divididos por classes e no final de cada classe também é apresentada as médias de tempo de processamento e precisão dos resultados da respectiva classe. A figura (4.49) mostra o resto do relatório da figura (4.48).

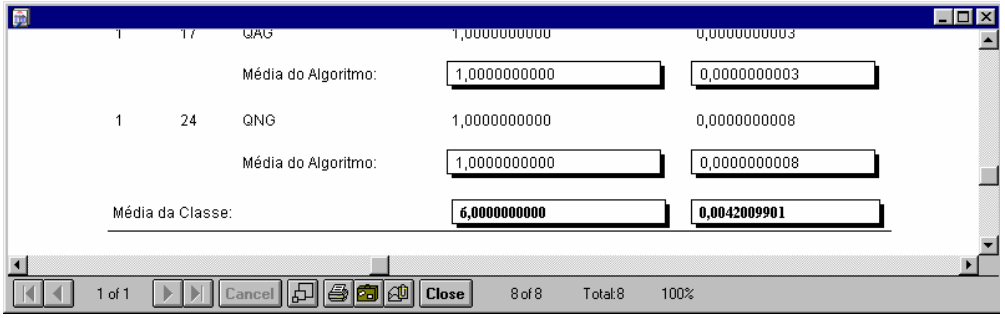


Figura 4.49: Média da Classe

Todos os dados apresentados pelo relatório acima são utilizados pelo método de aprendizado automático para a atualização do arquivo de pesos.

Depois que o usuário imprimir todos os relatórios necessários a sua análise o sistema exibirá a tela apresentada pela figura (4.50):

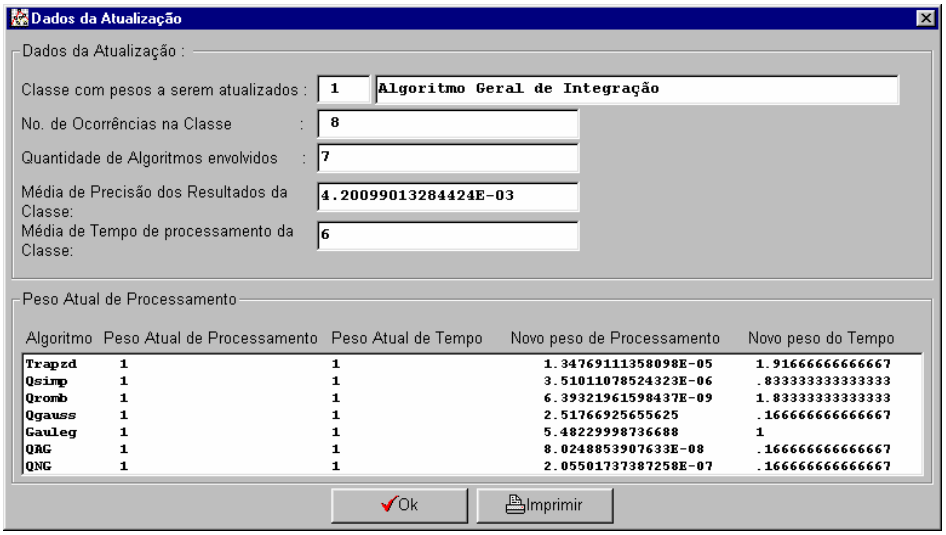


Figura 4.50: Dados da Atualização do Arquivo de Pesos

A figura (4.50) mostra a classe onde os pesos vão ser atualizados, o número de ocorrências encontradas, a quantidade de algoritmos que serão envolvidos no processo de atualização, a média da precisão dos resultados e do tempo de processamento. Logo após é exibido os algoritmos envolvidos mostrando os atuais pesos e os pesos que foram calculados pelo o método de aprendizado automático. O usuário pode imprimir todos estes dados através do botão *Imprimir*.

No próximo capítulo, será feita uma pequena comparação do INTEGRE com alguns softwares consagrados.

CAPÍTULO V

Comparação com Softwares Consagrados

5.1 Introdução

Vários softwares estão disponíveis no mercado para atender a crescente gama de usuários que necessitam utilizar métodos numéricos para resolver os mais variados problemas matemáticos. Os softwares que aqui serão estudados, se apresentam como sistemas gerais para elaboração de cálculos matemáticos e se caracterizam por efetuarem manipulações simbólicas complexas, o que esconde em parte, o seu real poder de manipulação numérica.

Para que essa manipulação simbólica seja efetuada com êxito, tais softwares foram escritos em LISP ou PROLOG, que em geral, são linguagens inadequadas para o processamento numérico, e mais, possuem dificuldade de comunicação com o FORTRAN.

Uma solução que vem sendo praticada é a transposição de todo o código do software para a linguagem C, pois com essa linguagem é possível, embora com mais trabalho, a computação de manipulações simbólicas e no processamento numérico está se tornando competitiva com o FORTRAN. A alternativa de transposição torna-se inviável, tendo em vista o grande acervo de softwares que já foram escritos em FORTRAN.

Neste capítulo será feito o estudo do MATHEMATICA, MATHCAD e do MATLAB e uma pequena comparação com o INTEGRE.

5.2 O MATHEMATICA, o MATHCAD e o MATLAB

O MATHEMATICA, o MATHCAD e o MATLAB são softwares de finalidade geral que podem ser utilizados de diversas maneiras, entre as quais pode-se destacar:

1. Efetuar cálculos complexos: além das manipulações numéricas convencionais estes softwares trabalham com computação simbólica e geram gráficos.

2. Permitem a criação de NOTEBOOKS que são textos agregados com gráficos que são geralmente utilizados para elaborar projetos de estudo assistido por computador.

3. São possuidores de uma interface que auxilia o usuário na especificação dos parâmetros de entrada, facilitando assim o seu manuseio.

4. Possuem uma linguagem de programação própria capaz de representar o conhecimento matemático de forma eficiente. Tal linguagem pode efetuar tanto manipulações simbólicas como numéricas.

5.3 Teste dos Sofwares Consagrados

Como foi dito anteriormente, o nosso enfoque estará voltado para os resultados provenientes de integração numérica. Um ótimo exemplo para o teste dos softwares anteriormente citados é a integral (5.1) que pode ser encontrada em [20]:

$$\int_0^3 x^3 \ln\left[\left|(x^2-1)(x^2-2)\right|\right]dx \qquad (5.1)$$

O gráfico da função a ser integrada pode ser observado na figura (5.1):

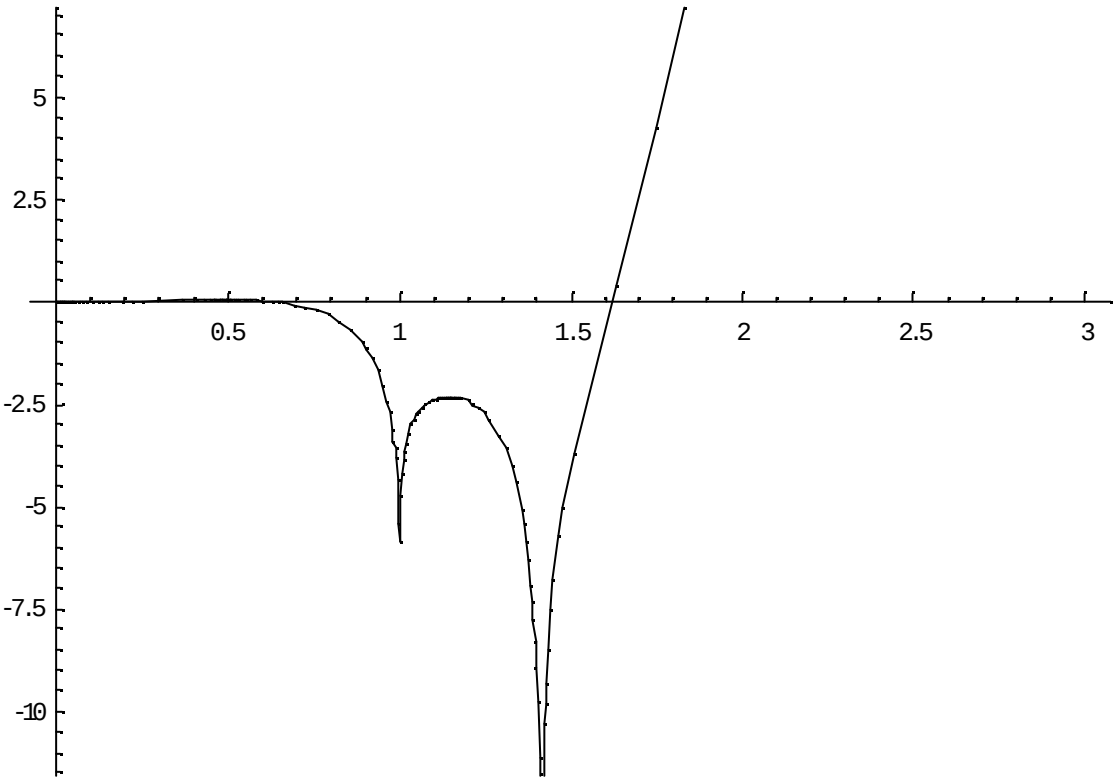


Figura 5.1: Gráfico da função $x^3 \ln\left[\left|(x^2-1)(x^2-2)\right|\right]$

Segundo [20], o resultado exato para este problema de integração é:

$$\int_0^3 x^3 \ln\left|(x^2-1)(x^2-2)\right| dx = 61\ln(2) + 77\ \frac{\ln(7)}{4} - 27 \qquad (5.2)$$

Pela expressão acima, o valor da integral será (5.3):

$$\int_0^3 x^3 \ln\left|(x^2-1)(x^2-2)\right| dx = 52.740,748,383,471,44 \qquad (5.3)$$

A Tabela (5.1) exibe uma comparação dos resultados obtidos pelo MATHEMATICA, MATHCAD, MATLAB e o INTEGRE.

Software	**Valor da Aproximação**	**Erro Relativo**
MATHEMATICA	52.751,096,238,611,30	1.961637933189292e-004
MATLAB	52.740,576,819,312,22	3.252982229778950e-006
MATHCAD	52.749,770,538,411,61	1.710368566180176e-004
INTEGRE	52.740,748,383,471,43	2.694473467057510e-016

Tabela 5.1: Valores da aproximação da Integral $\int_0^3 x^3 \ln\left|(x^2-1)(x^2-2)\right| dx$

Para obtenção desses valores foram utilizados os comandos: NINTEGRATE do MATHEMATICA [08], QUAD8 do MATLAB [10] e o operador $\int$ do MATHCAD [09].

5.4 Análise com funções complexas

Algumas funções, encontradas em [20, 21], foram testadas nos softwares anteriormente apresentados. Para efeito de apresentação, será exibida a integral proposta e um quadro contendo o valor das aproximações que foram obtidas com cada um dos softwares em questão, juntamente com o valor exato da integral.

5.4.1 Estudo de Caso 1

$$\int_0^1 \sqrt{x}\,\ln(x)\,dx \qquad (5.5)$$

Software	Valor da Aproximação	Erro Relativo
MATHEMATICA	-0.444,444,444,822,03	8.495675405949442e-010
MATLAB	-0.444,444,065,475,76	8.526802670384939e-007
MATHCAD	-0.444,291,362,290,62	3.445535223331494e-004
INTEGRE	-0,444,444,444,616,99	3.882275749369978e-010
VALOR EXATO	$-\frac{4}{9}$ =-0.444,444,444,444,44	-

Tabela 5.2: Aproximações para integral da função $\sqrt{x}\ln(x)$ no intervalo (0, 1]

5.4.2 Estudo de Caso 2

$$\int_0^1 \frac{\ln(x)}{\sqrt{x}} dx \tag{5.6}$$

Software	Valor da Aproximação	Erro Relativo
MATHEMATICA	-3.999,999,999,999,91	2.253752739989118e-014
MATHCAD	-3.999,199,588,261,21	2.001429838957360e-004
MATLAB	-4.115,332,989,646,97	0.028025190169820e-000
INTEGRE	-4,000,000,000,094,99	2.374744845156324e-011
VALOR EXATO	-4.000,000,000,000,00	-

Tabela 5.3: Aproximações para integral da função $\frac{\ln(x)}{\sqrt{x}}$ no intervalo (0, 1]

5.4.3 Estudo de Caso 3

$$\int_0^\infty \frac{x}{1+x^2} \mathrm{Sin}(x) dx \tag{5.7}$$

SOFTWARE UTILIZADO	VALOR DA APROXIMAÇÃO	ERRO RELATIVO
MATHEMATICA	0.577,863,674,895,46	1.729128799008524e-015
MATHCAD	Não Resolveu	-
MATLAB	Não Resolveu	-
INTEGRE	0.577,863,674,895,53	1.212311413526943e-013
VALOR EXATO	0.577,863,674,895,46	-

Tabela 5.4: Aproximações para integral da função $\frac{x}{1+x^2}\mathrm{Sin}(x)$ no intervalo [0, ¥)

5.4.4 Estudo de Caso 4

$$\int_0^{\infty} \frac{\cos(\pi x/2)}{\sqrt{x}}dx \tag{5.8}$$

SOFTWARE UTILIZADO	VALOR DA APROXIMAÇÃO	ERRO RELATIVO
MATHEMATICA	1.000,000,000,036,14	3.613997989629200e-011
MATHCAD	Não Resolveu	-
MATLAB	Não Resolveu	-
INTEGRE	0.999,999,999,385,69	6.143100476529004e-010
VALOR EXATO	1.000,000,000,000,00	-

Tabela 5.5: Aproximações para integral da função $\frac{\cos(\pi x/2)}{\sqrt{x}}$ no intervalo (0, ¥)

5.4.5 Estudo de Caso 5

$$\int_{-1}^{5} \frac{1}{x(5x^3+6)}dx \tag{5.9}$$

SOFTWARE UTILIZADO	VALOR DA APROXIMAÇÃO	ERRO RELATIVO
MATHEMATICA	0.638,340,656,716,59	1.14090283301139e-000
MATHCAD	Não Resolveu	-
MATLAB	1.268,264,292,596,93	1.07091897760012e-000
INTEGRE	-0.089,944,006,957,71	1.110914174228914e-013
VALOR EXATO	-0.089,944,006,957,72	-

Tabela 5.6: Aproximações para integral da função $\frac{1}{x(5x^3+6)}$ no intervalo (-1, 5]

5.4.6 Estudo de Caso 6

$$\int_0^{2\pi} \frac{x\ \mathrm{Sin}(10x)}{\sqrt{1-(x^2/4\pi^2)}}\,dx \tag{5.10}$$

SOFTWARE UTILIZADO	VALOR DA APROXIMAÇÃO	ERRO RELATIVO
MATHEMATICA	Não Resolveu	-
MATHCAD	-4.387,611,013,858,86	-7.819673137476792e-010
MATLAB	-0.125,043,347,552,57	-34.08872005722028e-000
INTEGRE	-4.387,611,092,208,54	-1.707505780176577e-008
VALOR EXATO	-4.387,611,017,289,83	-

Tabela 5.7: Aproximações para integral da função $\frac{xsin(10x)}{\sqrt{1-(x^2/4p^2)}}$ no intervalo [0, 2p]

5.4.8 Estudo de Caso 7

$$\int_0^1 \ln(x)\sin(\pi 10x)\,dx \tag{5.11}$$

SOFTWARE UTILIZADO	VALOR DA APROXIMAÇÃO	ERRO RELATIVO
MATHEMATICA	-0.128,136,848,399,15	1.171853888686294e-013
MATHCAD	-0.128,137,055,844,98	1.618936978282528e-006
MATLAB	-0.128,135,318,887,58	1.193669005761811e-005
INTEGRE	-0.128,136,847,859,68	4.210217607418592e-009
VALOR EXATO	-0.128,136,848,399,17	-

Tabela 5.8: Aproximações para integral da função ln(x) sin(p 10 x) no intervalo (0, 1]

CAPÍTULO VI

Conclusões

Neste capítulo será apresentada a conclusão desse trabalho e sua contribuição em ciência e tecnologia, bem como uma série de ramificações que podem ser explorados em trabalhos futuros.

6.1 Introdução

O sistema EXIBITAN apresentou inovações no seu domínio de aplicação, cálculo numérico básico, como controle de ativação dos algoritmos e a introdução de alguma análise dos resultados obtidos. A partir do conhecimento adquirido com a construção do EXIBITAN, os estudos foram conduzidos, resultando na construção do protótipo, o FFTEX, com a aplicação do método das transformadas rápidas de Fourier. O FFTEX apresentou uma evolução em relação ao EXIBITAN e aos sistema acoplados em geral em diversos aspectos [11]:

a) Foi introduzida a técnica de simulação como esforço ao processo tradicional de aquisição de conhecimento.

b) A escolha de um determinado algoritmo é feita por base em valores percentuais de graus de satisfação fornecidos pelo usuário.

c) Uma técnica original de aprendizado automático baseado no estudo estatístico de casos passados, que torna o sistema mais realista e confiável.

O sistema FFTEX representou um grande avanço na nascente área dos sistemas acoplados, servindo como base para construção de um outro sistema denominado SIPREX [11]. O sistema SIPREX incorpora os resultados das experiências obtidas com a construção desses dois protótipos. Diversas outras inovações foram acrescentadas, que podem ser constatadas na sua estrutura de funcionamento e nos módulos que o compõem, como por exemplo no emprego da interface inteligente com a finalidade de direcionar e facilitar o processo de especificação do problema e no uso de um aprendizado seletivo, onde somente os usuários classificados têm a permissão de desencadear o processo de aprendizado que resulta na alteração da base de conhecimento do sistema [11].

Na época em que a pesquisa para o desenvolvimento do SIPREX avançava, eram divulgados vários trabalhos que integravam técnicas de IA para concepção de sistema voltados a engenharia, mostrado assim a relevância do SIPREX.

Através do estudo da limitações dos softwares acima foi possível desenvolver o sistema INTEGRE. O sistema INTEGRE foi desenvolvido com a finalidade de escolher o melhor algoritmo para uma determinada situação. Para efeito de comparação, considere um sistema para diagnósticos clínicos. Com as informações obtidas de maneira interativa com o usuário, este sistema pode chegar a um possível diagnóstico e, em alguns casos, aplicar uma receita visando a cura do paciente. No INTEGRE também chega-se a um "diagnóstico", que no nosso caso é a escolha de um determinado algoritmo. A partir desta escolha o controle passa à parte numérica, com a ativação da rotina que implementa o algoritmo escolhido. Uma vez executados os cálculos numéricos o sistema expõem os resultados obtidos para uma análise final, podendo o usuário solicitar um nova tentativa utilizando um outro método. É como se o sistema de diagnósticos obtivesse do paciente a resposta sintomática e instantânea dos efeitos da receita prescrita para, a partir daí, processar a análise dessa resposta.

O INTEGRE além de ser um sistema mais completo, do ponto de vista de acoplamento e inteligência, que os anteriores, aborda e apresenta soluções de problemas que nem sequer foram mencionadas pelos demais. Um desses problema é especificação dos parâmetros de entrada (função integrando, limites de integração, função peso, etc..) que é realizado através de sua interface gráfica inteligente que auxilia o usuário na entrada desses parâmetros, e permite que o mesmo possa interagir com o sistema podendo assim mudar a escolha do algoritmo para a resolução da integral. Caso sejam enviados parâmetros inválidos de entrada (intervalo no qual a função não esteja definida) o sistema ajusta, de forma automática, esses parâmetros para que estes possam ser envidados à parte numérica.

O segundo problema abordado pelo sistema INTEGRE é o do gerenciamento e acoplamento entre as diversas partes que compõem o sistema, o que não é uma tarefa trivial, principalmente em se tratando de um sistema acoplado. As informações fornecidas pelo usuário devem satisfazer as necessidades da parte simbólica e da parte numérica do sistema, lembrando que cada um dos métodos que compõem a parte numérica possui uma nomenclatura própria e exige o fornecimento de um conjunto de parâmetros distintos. Para a solução deste problema, o projeto da interface do sistema foi de fundamental importância. A interface do sistema INTEGRE é auto-explicativa, o usuário entra com todos o parâmetros

pedidos, estes são encapsulados em dois grupos: parâmetros para as rotinas do NR e parâmetros para as rotina do QUADPACK. Apesar de existirem diferenças entre os parâmetros das rotinas, estes foram assim encapsulados devido algumas similaridades o que resultou na construção de uma estrutura comum para estes dois grupos.

Um terceiro problema abordado no INTEGRE é o do auto-refinamento do sistema através do uso de técnicas de aprendizado automático. Embora seja um atributo desejável a todos os sistemas inteligentes, raramente esta capacidade está presente, constituindo-se uma ferramenta no refino de sistemas pouco explorada. A técnica particular de aprendizado adotada pelo INTEGRE é feita de duas etapas, análise e alteração. Na fase de análise, tenta-se refinar a solução com ajuda do usuário, que é levado a opinar sobre o resultado final fornecido. Uma nova tentativa de cálculo pode resultar desta análise. Se bem ou mal sucedido, o sistema registra este fato no arquivo ocorrências. Depois de um certo número X de ocorrências de uma mesma classe, o sistema altera (fase de alteração) o arquivo de pesos que refletirá o sucesso ou insucesso da solução fornecida. Tal informação será utilizada posteriormente pelo método de escolha para a seleção de um dado algoritmo e aquele que tiver um melhor coeficiente de produtividade será escolhido.

Com relação a alguns softwares já consagrados, o INTEGRE apresentou uma melhor estabilidade no cálculo de algumas integrais como foi apresentado no capítulo V, mostrando assim uma grande diversidade de integrais que ele consegue resolver.

Com relação aos algoritmos envolvidos no cálculo da aproximação da integral, os algoritmos pertencentes ao pacote QUADPACK obtiveram, no geral, um melhor desempenho do aqueles provenientes do NR. Esta conclusão foi possível graças ao método de aprendizagem automática que de acordo com a precisão dos resultados e os tempos de processamento efetua, de forma automática a classificação dos melhores algoritmos.

6.2 Os algoritmos de Integração e a Paralelização

No capítulo III foram apresentados os algoritmos que compõem o sistema INTEGRE. Estes algoritmos foram classificados, segundo a origem, em duas grandes categorias: algoritmos provenientes do NUMERICAL RECIPES e algoritmos provenientes do QUADPACK.

Uma pequena análise feita com os algoritmos gerais de integração do NUMERICAL RECIPES mostra, pela tabela (6-1), o tempo de processamento para o cálculo da integral (6-1) a uma precisão de 10^{-8}.

$$\int_0^{10} Sin(Cos(x))dx \qquad (6\text{-}1)$$

Algoritmo	**Tempo de Processamento (Centésimos de Segundo)**
TRAPZD	16
QSIMP	11
QROMB	6
QGAUSS	6
GAULEG	5

Tabela 6.1: Análise dos algoritmos de integração para o pior caso

Tais algoritmos foram escolhidos por serem os mais lentos dente aqueles que aqui foram estudados. Os tempos de processamento, apresentados pela tabela (6-1) podem variar dependendo da máquina que esteja sendo utilizada (os tempos acima foram obtidos num Pentium de 266 MHz). O tempo total de processamento foi de 50 centésimos de segundo.

Utilizando paralelismo funcional (vários processos sendo disparados em diferentes processadores), os algoritmos pertencentes ao NUMERICAL RECIPES foram simulados no ambiente UNIX e foi obtido os tempos de processamento apresentados pela tabela (6-2).

Estes testes foram executados no computador IBM SP-2. Vale ressaltar que devido a fatores externos estes tempos oscilam com muita freqüência, para tal foi tomada uma média dos tempos de processamento. Duas abordagem podem ser utilizadas para obtenção do valor da aproximação da integral:

1) O sistema dispara todos os algoritmos e espera o primeiro retornar a aproximação. A aproximação retornada é exibida para o usuário. Caso o usuário rejeite esta aproximação, o sistema dispara os algoritmos restantes pegando, novamente a primeira

aproximação. Tal abordagem, para o pior caso, seria semelhante a execução serial desses algoritmos não obtendo assim ganho de tempo, além de invalidar o método de aprendizagem automática, pois não haveria necessidade de se selecionar um determinado algoritmo já que todos estariam previamente selecionados.

Algoritmo	**Tempo de Processamento (Centésimos de Segundo)**
TRAPZD	24,6
QSIMP	27,5
QROMB	24,7
QGAUSS	29,7
GAULEG	23,6

Tabela 6.2: Algoritmos de Integração executados em diversos processadores

2) O sistema dispara todos os algoritmos e espera o último algoritmo retornar com o valor da aproximação. Neste caso, o tempo de processamento seria o máximo do tempos. Caso o usuário rejeitasse um determinado valor para a aproximação, um outro valor seria exibido sem ônus de processamento. No caso acima, o tempo processamento seria de 29,7 centésimos de segundo. Tal procedimento, novamente, invalidaria o método de escolha do melhor algoritmo através do aprendizado automático, pois novamente, todos os algoritmos seriam executados. Se um determinado algoritmo revolver o problema de forma satisfatória em menos tempo o sistema terá que esperar o algoritmo mais lento para depois exibir os resultados obtidos na tela.

O sistema INTEGRE, através do método de escolha inteligente, obteve melhores resultados do que as duas alternativas acima, pois o auto-refinamento que o sistema adota dos parâmetros de saída dos algoritmos (precisão dos resultados e tempo de processamento) garante que apenas um, o melhor algoritmo, é selecionado para o cálculo da integral otimizando assim o uso do processador.

Um outro aspecto a se levar em consideração é que a passagem dos parâmetros de integração, no ambiente multiprocessamento, corresponde a uma boa parte do tempo total de processamento.

6.3 Perspectiva de Trabalhos Futuros

Embora o INTEGRE represente um grande avanço em relação ao estado de arte na área de sistemas acoplados, acredita-se que ele seja apenas mais um passo na formalização de uma metodologia sólida para concepção de outros sistemas. Entre os pontos levantados, em torno dos quais futuros trabalhos serão desenvolvidos, destaca-se os seguintes:

1. Reformulação a Ampliação do número de funções numéricas

Dotar o sistema de maleabilidade nas especificações das funções numéricas. O usuário dispõe de um conjunto fixo de funções que o INTEGRE lhe oferece. O sistema poderia receber a definição de outras funções como é o caso das funções hiberbólicas e incorporar estas informações para um uso posterior do usuário.

2. Ampliação do Módulo Numérico

Na perspectiva de ampliação do escopo do sistema, outros algoritmos poderiam ser adicionados no sistema.

3. Elaboração de uma Versão do INTEGRE para Unix

A plataforma INTEL já está possui a hegemonia do mercado e na perspectiva de uma versão comercial do INTEGRE ele foi desenvolvido nesta plataforma para o sistema operacional WINDOWS 95/98/NT. Na tentativa de explorar melhor os recursos do paralelismo pode ser feita uma versão do sistema em ambiente UNIX.

4. Ampliação do estudo sobre paralelismo

Apesar do paralelismo não atender, nesse primeiro momento, as necessidades de redução de tempo do sistema INTEGRE devido a troca de parâmetros entre as rotinas. O estudo sobre paralelismo pode ser ampliado na tentativa de redução do tempo de processamento.

REFERÊNCIAS

[01] Rice, J. R., *Announcement and call for papers, mathematical software*, SIGNUM Newsletter 4, p. 7, 1969.

[02] Hattori, M. T., *Métodos e Softwares Numéricos*, Trabalho não Publicado, Departamento de Sistemas e Ciência da Computação, Universidade Federal da Paraíba, Campus II, Campina Grande, Paraíba, 1992.

[03] Rice, J. R., and Boisvert, R. F., *Solving Elliptic Problem Using ELLPACK*, Springer-Verlag, New York,1985.

[04] IMSL – International Mathematical and Statistical Libraries, *Fortran Subroutines for Mathematical Applications*, v. 1 e 2, Visual Numerics, Inc., USA, 1994.

[05] Cody, W. J., *Observations on the Mathematical Software Effort*, in Cowell [1984], pp. 1-19, 1984.

[06] Rice, J. R., *Numerical Methods, Software, and Analysis*, McGraw-Hill International Student Edition Edition, Singapore, 1983.

[07] Bailey, D. H., *Multiprecision Translation and Execution of FORTRAN Programs*, ACM Trans. Math. Softw. 19, pp. 288-319, 1993.

[08] Wolfram, S., *MATHEMATICA: A System for Doing Mathematics by Computer*, Addison-Wesley, London, 1988.

[09] Rowell, J. W., *Mathematical Modeling With Mathcad: Explorations in the Calculus and Points Beyond/Book and Disk*, Addison-Wesley, [S. l.], 1991.

[10] Moler, C. B., Litter, J. and Bangert, S., *PC-MATLAB for MS-DOS Personal Computers*, The Math Works , Inc. Sherborn, MA (User's Guide, Tutorial & Reference), 1987.

[11] Pequeno, M. C., *SIPREX – Um Sistema Especialista para o Processamento Digital de Sinais*, Tese (Doutorado em Engenharia Elétrica) - Curso de Pós-Graduação em Engenharia Elétrica, Universidade Federal da Paraíba, 1991.

[12] Pequeno, M. C., De Carvalho. J. M. e Hattori, M. T., *Uma metodologia Para Construção do Módulo Numérico de Sistemas Especialistas Acoplados*, XV Congresso Nacional de Matemática Aplicada e Computacional, S. Carlos, São Paulo, 1993.

[13] Pequeno, M. C., De Carvalho, J. M. e Hattori, M. T., *EXBITAN – Um Sistema Especialista para o Cálculo Numérico*, XVI Conferência Latino Americana de Informática, Clei Painel'90 Expodata, Assunção, Paraguai, 1990.

[14] Pequeno, M.C., De Carvalho, J. M., Hattori, M. T., e Nicolletti, P.S., *FFTEX – Um Sistema Especialista para a Computação de Transformadas Rápidas de Fourrier*, XII Congresso Nacional de Matemática Aplicada e Computacional, Águas de Lindóia, São Paulo, 1990.

[15] Pequeno, M.C., De Carvalho, J. M., Hattori, M. T., e Nicolletti, P.S., *FFTEX – Uma técnica de Aprendizado Automático para um Sistema Especialista Acoplado (FFTEX)*, XIV Congresso Nacional de Matemática Aplicada Computacional, Nova Friburgo, Rio de Janeiro, 1991.

[16] Pequeno, M.C., De Carvalho, J. M., Hattori, M. T. e Silva Filho, A. M., *SIPREX: An Expert System for Digital Filter Design*, Thirteenth GRETSI Symbosium on Signal and Image Processing, Juan-les-Pins, França, 1991.

[17] Pequeno, M.C., De Carvalho, J. M. e Hattori, M. T., *Methodology for Building Numerical Structures for SIPREX a Coupled Expert Systems for Digital Filter Design*, 38th Midwest Symposium on Circuits and Systems, Rio de Janeiro, 1995.

[18] Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T., *NUMERICAL RECIPES – The Art of Scientific Computing (Fortran Version)*, Cambridge University Press, Cambridge, England, p. 102-130, 1990.

[19] Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T., *NUMERICAL RECIPES – Example Book (Fortran)*, 2 ed., Cambridge University Press, New York, p. 42-55, 1992.

[20] Piessens, R., Doncker-Kapenga, E., Überhuber, W. and Kahaner, D. K., *QUADPACK – A Subroutine Package for Automatic Integration*, Springer-Verlag, Berlin Heidelberg, 1983.

[21] Davis, P. J. and Rabinowitz, P., *Methods of Numerical Integration*, Academic Press, New York, 1975.

[22] Cláudio, D. M. e Marins, J. M., *Cálculo Numérico Computacional - Teoria e Prática*. 2 ed., Editora Atlas, São Paulo, p.260-350, 1994.

[23] Swokowski, E. W., *Cálculo com Geometria Analítica*. McGraw-Hill do Brasil, São Paulo, 1983.

[24] Carnahan, B., Luther, H. A. and Wilkes, J. O., *Applied Numerical Methods*, John Wiley & Sons, New York, 1969.

[25] Kahaner, D., *Sources of Information on Quadrature Software*, In Sources and Development of Mathematical Sofware, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, p. 134-164 , [198-?].

[26] Smith, B. T., *Portability and Adaptability - What Are the Issues, in Jacobs*, D. A. H. (ed), Numerical Software - Needs and Availability, Academic Press, London, p. 21-38, 1978.

[27] Ford, B., and Iles, R. M. J., *The what and why of problem solving environments for scientific computing*, Problem Solving Environments for Scientific Computing, B. Ford & F. Chatelin, eds., North-Holland, Amsterdam, The Netherlands, p. 3-21, 1987.

[28] Roman, S., *Concepts of Object-Oriented Programming with Visual Basic*, Springer-Velarg, New York, 1997.

[29] Da Costa Jr., A.E. e Pequeno, M. C., *GRAFPLUS – Um Ambiente para Análise Automática de Funções*, INFOSOL 95, Congresso e Feira de Informática e Telecomunicações, Fortaleza, Ceará, Junho, 1995.

[30] Pequeno, M. C., Hattori, M. T., Da Costa Jr., A.E., *ANAFU – Um Ambiente amigável para Análise de Funções*, III Congresso Ibero Americano de Informática na Educação, Barraquilla, Colômbia, Julho,1996. (Selecionado para publicação).

[31] Pequeno, M. C., Hattori, M. T. e Farias, M. S. A. T., *Avaliador Automático de Funções Matemáticas para o Cálculo Numérico*, XVII Congresso Nacional de Matemática Aplicada e Computacional, Vitória, Espirito Santo, Setembro, 1994.

[32] Pequeno, M. C., Hattori, M. T. e Farias, M. S. A. T., *Geração Automática do Código Fonte de Funções para Programas de Cálculo Numérico*, XV Congresso Ibero Latino-Americano sobre Métodos Computacionais para Engenharia, Belo Horizonte, MG, Dezembro, 1994.

[33] Pequeno, M. C., Hattori, M. T. e Farias, M. S. A. T., *Automatic Programming and Evaluation of Mathematical Functions for Numerical Calculos*, International Congress on Industrial and Applied Mathematics, Hamburg, Alemanha, Julho, 1995.

[34] Zaratian, Z., *Microsoft Visual C++ Owner's Manual Version 5.0*, Microsoft Press, USA, 1997.

[35] Schildt, H., *C Completo e Total*, Makron, McGraw Hill, São Paulo, p. 841-860, 1990.

[36] Pequeno, M. C, Hattori, M. T e Carvalho, J. M., *FFT - Problemática de Testes*, XII CNMAC, S. J. do Rio Preto, São Paulo, 1989.