# A DESCRIÇÃO DE UMA INTERFACE GRÁFICA E INTERATIVA PARA MATEMÁTICA COMPUTACIONAL

Álfio Ricardo Martini
Tiaraju Asmuz Diverio, Msc

Instituto de Informática e PGCC da UFRGS
Caixa Postal 1501 - 90 010 -Porto Alegre - RS
E-Mail : DIVERIO@SBU.UFRGS.ANRS.BR

**RESUMO**

Neste artigo é descrita a interface gráfica e interativa para matemática computacional (IMAC). Após anos de utilização de ambientes dirigidos por menus, apresentamos uma interface que utiliza uma linguagem de comando como forma de diálogo com o usuário. Primeiramente é descrito o modelo conceitual do usuário, ou seja, a compreensão conceitual que o usuário tem do programa. Então é introduzido o conceito de linguagens de comando, suas características e atributos desejáveis, a implicação direta destes conceitos no projeto da linguagem de comando de IMAC. Por fim, é apresentado o layout da tela da interface. No apêndice são apresentadas as principais operações (comandos) suportadas pela interface.

**Palavras chaves** : Linguagens de comando, interface usuário-computador, matemática computacional.

**ABSTRACT**

In this article an interactive graphics interface for researching in computational mathematics is described. After many years of using a menu-driven enviromnent, we present an interface that uses a command language as a model of user-computer dialog. Firstly, the conceptual user's model is described, i.e., the conceptual understanding that the user has about the program. Then the concept of command languages is introduced, its characteristics an desirable attributes, the direct implication of this concepts in the design of IMAC's command language. Finally, the interface screen layout is presented. In the appendix, the main operations (commands) supported by the interface are presented.

**Key words** : Command languages, user-computer interface, computational mathematics.

## 1. INTRODUÇÃO

Há alguns anos, no Instituto de Informática e Pós Graduação em Ciência da Computação da Universidade Federal do Rio Grande do Sul, tem-se estudado o problema do desenvolvimento de software de alta qualidade [1]. Como resultado deste estudo, diversas ferramentas foram implementadas, destacando-se **SINAI-16** [2] e **LEPMAC** [3].

Estes sistemas caracterizavam-se por um ambiente dirigido por menus e teclas aceleradoras. Como uma alternativa a este tipo de ambiente, surgiu a idéia do desenvolvimento de uma interface na qual o diálogo usuário-computador pudesse ser efetuado através de uma linguagem de comando. Desta forma, o usuário poderia ter acesso a diversos métodos do cálculo numérico computacional através da simples edição de comandos, os quais seriam, basicamente, os métodos disponíveis e seus respectivos argumentos (parâmetros). Além disso, este tipo de abordagem é mais interessante para usuários mais experientes como professores e pesquisadores da matemática computacional e interessados na resolução de problemas referentes a este assunto, tanto no que diz respeito a utilização do computador quanto ao tipo de tarefa a ser executada.

Como resposta a esta necessidade, foi projetada uma interface gráfica para matemática computacional (IMAC).

**IMAC** foi projetada para ser um ambiente integrado para trabalhar com uma enorme variedade de metodos computacionais como por exemplo, diferenciação e integração numérica, cálculo de zeros, resolução de sistemas de equações diferenciais , ajuste de curvas pelo critério dos mínimos quadrados, interpolação polinomial e resolução de sistemas de equações lineares.

Este artigo tem como objetivo principal descrever a estrutura de diálogo na interface de IMAC. Primeiramente é descrito o modelo conceitual do usuário, ou seja, a compreensão conceitual que o usuário tem do programa. Então é introduzido o conceito de linguagens de comando, suas características e atributos desejáveis e, a implicação direta destes conceitos no projeto da linguagem de comando de IMAC. Após, é apresentado o layout da tela de interface e, no apêndice são apresentadas as principais operações(comandos) suportadas pela interface.

## 2. O MODELO CONCEITUAL DO USUÁRIO EM IMAC

O modelo do usuário é o modelo conceitual formado pelo usuário da informação manipulada e dos processos aplicados sobre esta informação[4]. O modelo permite que o usuário desenvolva, com ou sem conhecimento da tecnologia de computadores, uma ampla compreensão sobre o que o programa está fazendo. Com a ajuda do modelo ele pode antecipar os efeitos de suas ações e pode desenvolver estratégias para operar o programa.

Uma maneira útil de se representar o modelo do usuário, para propósitos de discussão e documentação, é como um conjunto de **objetos** juntamente com um conjunto de **ações** que o usuário pode aplicar a estes objetos. Cada objeto é um item de informação sobre o qual o usuário tem algum controle, ele pode mostrá-lo, pedir informações sobre ele, destruí-lo e criar outros objetos em seu lugar.

**IMAC** tem quatro tipos fundamentais de objetos :

**a) Expressoes Matemáticas (funções algébricas, transcendentes e polinomiais);**

**b) Tabelas de Pontos;**

**c) Sistemas de Equações Diferenciais Ordinárias;**

**d) Sistemas Matriciais;**

## 2.1 Objetos e Ações

**Expressões Matemáticas** são criadas na interface de **IMAC** através da invocação de um comando específico (ver apêndice) seguido da expressão matemática desejada. É permitido trabalhar com até dez funções simultaneamente, ou seja, são sempre mantidas na memória as dez últimas funções criadas pelo usuário. Sempre que o usuário cria uma nova função ela se torna o que em **IMAC** é entendido como **função de trabalho.** É sobre esta função que o usuário pode realizar as seguintes ações:

- Integração numérica;
- Diferenciação numérica;
- Plotagem da função e suas derivadas (primeira e segunda);
- Cálculo de zeros;
- Ampliações;

A **integração numérica** de funções pode ser feita através dos métodos de Simpson, Trapézio, Romberg e Quadratura Gaussiana.

A **diferenciação numérica** pode ser feita de dois modos, **gráfico ou numérico.** No modo gráfico, o usuário pode "caminhar" com um cursor através do gráfico da função enquanto o valor da derivada (primeira e segunda) é automaticamente mostrado na janela de visualização da função. No modo numérico, o usuário edita o comando seguido do valor (parâmetro) numérico sobre o qual ele quer determinar o valor da derivada.

A plotagem das derivadas sempre se refere á **função de trabalho.** O **cálculo** de zeros de funções pode ser feito pelos métodos de **Newton, Jarrat, Pégasus e Miller.**

Ampliação de uma função significa a determinação de uma subjanela de visualização para a função de trabalho. A seleção dos valores para a determinação desta subjanela pode ser feita no modo **gráfico ou numérico.** No modo numérico, o usuário edita o

respectivo comando, seguido dos valores que determinam a subjanela, ou seja, **X1 Y1 X2 Y2.** No modo gráfico, a escolha destes valores pode ser feita pela construção da janela diretamente sobre a área a ser ampliada . Todos os valores iniciais utilizados nos métodos de integração (limites de integração) e cálculo de zeros (aproximação inicial) também podem ser determinados numericamente ou graficamente.

Além disso, a possibilidade de trabalhar com até dez funções ao mesmo tempo permite que se possa plotá-las todas na mesma janela de visualização ( ver layout da interface).

**Tabela de Pontos** é o segundo tipo de objeto fundamental dentro da interface de **IMAC.** Tabelas de Pontos são formadas por uma série de dados (Xi,Yi) i = 0(1)n correspondentes aos valores de argumentos e valores de uma função f, tal que y = f(x), que são obtidos normalmente através de dados experimentais. Tabelas de Pontos podem ser criadas, editadas (deleção, alteração e inserção de pontos) gravadas e carregadas para a interface. Só pode existir uma tabela na interface de cada vez. Os valores são armazenados em memória, podem ser utilizados para ajuste de curvas pelo critério dos mínimos quadrados e interpolação polinomial. É permitido ajuste a curvas do tipo **linear, logarítmica, exponencial, potencial e polinomial** até sexto grau. A função ajustada é automaticamente considerada como **função de trabalho** e todas as operações descritas para as expressões matemáticas também podem ser efetuadas.

A interpolação de valores ( x e y) pode ser feita através do método de **Newton por diferenças divididas.**

O terceiro tipo de objeto que pode ser trabalhado em **IMAC** são os **Sistemas de equações diferenciais ordinárias de primeira ordem.** Para a resolução de equações diferencias ordinárias de ordem ) 1 ( a ordem de uma equação diferencial ordinária é dado pela derivada de maior ordem presente na equação) é necessário transformá-la para um sistema equivalente de equações diferencias

de primeiro grau. O método de redução de uma ou várias equações diferenciais para um sistema de equações de ordem 1 é como segue, e está contido em [6].

**a)** Introduz-se novas variáveis para cada derivada até (m-1)-ésima ordem;

**b)** Substitui-se na equação diferencial original todo $y^{(m)}$ por cada uma das (m-1)-ésimas variáveis correspondentes e esta passa a ser a primeira equação do sistema.

**c)** Acrescenta-se a equação diferencial para a k-ésima variável, sendo esta a derivada primeira da (k-1)-ésima variável.

Por exemplo, a equação:

$$y''' + xy'' + 3y' - y = x^2 + y^2\sin(x)$$

após aplicarmos o método, gera o seguinte sistema :

$w' = x^2 + y^2\sin(x) - xw - 3z + y$

$y' = z$

$z' = w$

A solução de equações diferenciais em **IMAC** está limitada àquelas que podem ser modeladas pelo problema do **valor inicial.** Para cada equação diferencial do sistema, é necessário que haja uma condição inicial. Os sistemas de equações diferenciais possuem o mesmo comportamento semântico das tabelas de pontos, ou seja, possuem o mesmo tipo de operações básicas (criação, edição, gravação, etc... ) e devem ser primeiramente criados para que depois sejam resolvidos. Novamente, só deve haver um sistema de equações diferenciais na interface de cada vez. Os métodos disponíveis são os de **Runge-Kutta de quarta ordem, Adams-Bashforth e Preditor-Corretor.**

O último tipo de objeto na interface de **IMAC** são os **sistemas matriciais.**Os sistemas matriciais envolvem as matrizes e os sistemas de equações lineares. O comportamento semântico dos

**sistemas matriciais** é idêntico ao da tabela de pontos e sistemas de equações diferenciais. As operações que envolvem a manipulação de matrizes são as de cálculo da matriz inversas, cálculo de normas e determinantes. Para a resolução de **sistemas de equações lineares**, os métodos disponíveis são os Métodos de Eliminação de Gauss com pivotamento simples, o método iterativo de Gauss-Seidel e o método compacto de Barnachevski.

Detalhes e complementação teórica a respeito dos métodos computacionais discutidos neste item podem ser encontradas em [5, 6, 7].

## 3 MODELO DE DIÁLOGO COM O USUÁRIO

### 3.1 Linguagens de comando

Linguagens de comando, as quais se originaram com os sistemas operacionais, são distinguíveis por seu imediatismo e por seu impacto em dispositivos de informação. O usuário edita um comando e observa o que acontece.

Linguagens de comando são distinguíveis de sistemas de seleção por MENUS pelo fato de que os usuários da primeira devem relembrar a notação e iniciar a ação. Usuários de sistemas orientados por MENUS recebem instruções e devem somente reconhecer e escolher entre um número limitado de alternativas visíveis. Eles respondem mais do que tomam iniciativa. Usuários de linguagens de comando são frequentemente exigidos a atingir feitos notáveis de memorização e digitação.

Ao projetarmos uma linguagem de comando, devemos considerar problemas referentes a sua sintaxe e a sua semântica. A sintaxe deve ser consistente e permitir fácil recuperação de erros sintáticos, ao mesmo tempo em que deve ser suficientemente compreensiva e poderosa para satisfazer as necessidades do usuário.

A parte semântica envolve o significado atribuído a cada construção sintática. A semântica de uma linguagem de comando relaciona-se, intimamente, com o **modelo do usuário.** Cada comando corresponde a uma ação, e os operandos (parâmetros) de cada comando são os objetos que a ação afeta.

Os objetos e ações do modelo fornecem um modelo inicial conveniente para o projeto da linguagem de comando. Eles definem, de uma forma abstrata, as operações que o usuário pode aplicar e os operandos nos quais elas podem ser aplicadas. A linguagem de comando de **IMAC** é uma representação concreta do modelo do usuário discutido anteriormente. Existem comandos para cada ação descrita, assim como muitos outros que aumentam o poder e a funcionalidade da interface.

Para a definição da sintaxe da linguagem, diversas qualidades desejáveis a uma boa linguagem de comando [8] foram consideradas, como por exemplo, facilidade de aprendizado, facilidade de uso, resistência a erros semânticos e consistência.

Facilidade de aprendizado pode ser entendida como facilidade de memorização. Facilidade de utilização normalmente significa facilidade de digitação. Resistência a erros semânticos refere-se a probabilidade de que um usuário possa digitar algo que ele não queira, e que assim mesmo é uma construção sintática válida dentro da linguagem. Quanto maior a probabilidade, maior a resistência. Consistência tem a ver com a presença de um estilo único na linguagem.

É sabido, através da experiência no projeto de linguagens de comando, que a presença de todas estas qualidades em uma mesma linguagem é praticamente impossível. Normalmente, uma grande dose de bom senso, experiência e esforço intelectual envolvem a escolha de uma em detrimento de outra característica.

No próximo item, serão discutidas as considerações principais envolvidas no projeto da linguagem de comando de **IMAC.**

## 3.2 Atributos de uma linguagem de comando e sua relação com a linguagem de comando de IMAC

A análise de uma linguagem de comando pode ser baseada em qualquer de seus atributos, mas apenas dois serão discutidos aqui; **estilo e estrutura**, pois são os que mais interessam dentro do contexto (modelo) para o qual a linguagem foi projetada.

**Estilo** refere-se ao fato de os comandos poderem ser extensos ou abreviados. Comandos extensos são menos eficientes quanto a digitação ( utilização mais difícil ), entretanto sua redundância faz com que a mesma seja menos sujeita a erros semânticos.

**Estrutura** se refere ao fato da relação entre os comandos e suas opções (argumentos) ser **linear** ou **hierárquica**. Uma estrutura linear fornece todos as funções da linguagem como ações distintas, ou seja, cada ação é representada por um comando. Uma estrutura hierárquica mostra um mesmo comando com várias opções. Uma estrutura linear normalmente é de utilização e memorização mais fácil do que uma estrutura hierárquica, ao passo que a última possui uma maior resistência a erros semânticos.

A linguagem de comando **IMAC** possui simultaneamente uma estrutura linear e hierárquica, sendo que, aproximadamente, metade dos comandos possui uma estrutura e o restante a outra (ver apêndice), além de permitir a edição de comandos extensos e abreviados.

Existem abreviações para praticamente todos os comandos da linguagem (ao menos para todos cujo comando extenso possui no mínimo quatro caracteres). A estratégia de abreviação adotada foi a de eliminação de vogais (a exceção do primeiro caracter) com

truncagem simples dos dois primeiros. Em alguns casos, em que a abreviação resultante foi considerada semanticamente insatisfatória ou então conflitante com alguma outra abreviação, foram utilizados o primeiro e o último caracteres.

| Comando | Abrev. | Opção | Parâmetros |
|---|---|---|---|
| **Cria** | **Cr** | **Fun** | X^2 - COS(X*SIN(X^2)) |
| **SelFun** | **Sl** | - | - |
| **Plot** | **Pl** | **Di** | - |
| **Jarrat** | **Jr** | - | 0.4857 8 |
| **Diret** | **Dr** | - | A:*.Exe |

**Figura 1. Exemplos da linguagem de comando**

Na Figura 1 vemos um subconjunto de comandos sintaticamente válidos na linguagem de **IMAC**, juntamente com suas abreviações correspondentes. O caractere "!" serve para indicar uma opção entre o comando extenso ou sua abreviação. O primeiro comando faz com que a expressão matemática dada por **X^2 - COS(X*SIN(X^2))** torne-se a **função de trabalho.** O segundo permite que o usuário escolha outra **função de trabalho** através da seleção de uma nova função entre todas as outras já criadas anteriormente na interface. O terceiro comando causa a plotagem da primeira derivada da **função de trabalho.** O quarto permite o cálculo do zero da **função de trabalho** utilizando **0.4857** como aproximação inicial e **8** dígitos significativos exatos como precisão do resultado final. O últ mo permite mostrar o diretório do drive **A** correspondente ao parâmetro ***.Exe.**

Com base nas considerações feitas anteriormente, pode-se concluir que a linguagem de comando de **IMAC** é de fácil utilização, uma vez que além de existirem abreviações para praticamente todos os comandos, as regras utilizadas na estratégia de abreviação podem ser facilmente memorizadas pelo

usuário (somente duas). O estilo adotado na linguagem possibilita uma maior resistência a erros semânticos (comandos extensos) ao mesmo tempo que permite a acomodação de usuários mais experimentados que desejem executar suas tarefas com maior rapidez (comandos abreviados). A estrutura linear e hierárquica da linguagem facilita o aprendizado (linear) ao mesmo tempo em que contribui para aumentar a resistência a erros semânticos.

Na linguagem de comando de **IMAC**, um string representando um comando sintaticamente válido digitado pelo usuário é representado por um nome de comando **Nome**, seguido por um ou mais espaços em branco, **eb**, seguido por zero ou mais opções, seguido por zero ou mais parâmetros. Portanto, a sintaxe da linguagem de comandos de **IMAC** pode ser representado pelo seguinte conjunto de expressões regulares:

Comando = (String_Comando)$^{+}$ Delimitador
String_Comando = Nome eb (Opção eb )$^{*}$ (Param)$^{*}$
Opção = Opção(eb Opção)$^{*}$
Param = Nome (eb Nome)$^{*}$
Nome = Letra(Letra|Dígito)$^{*}$
Opção = Letra(Letra|Dígito)$^{*}$
Letra = "A".."Z","a".."z"
Dígito = "0".."9"
Delimitador = EOL
eb = " "

## 4.0 LAYOUT DA INTERFACE DE IMAC

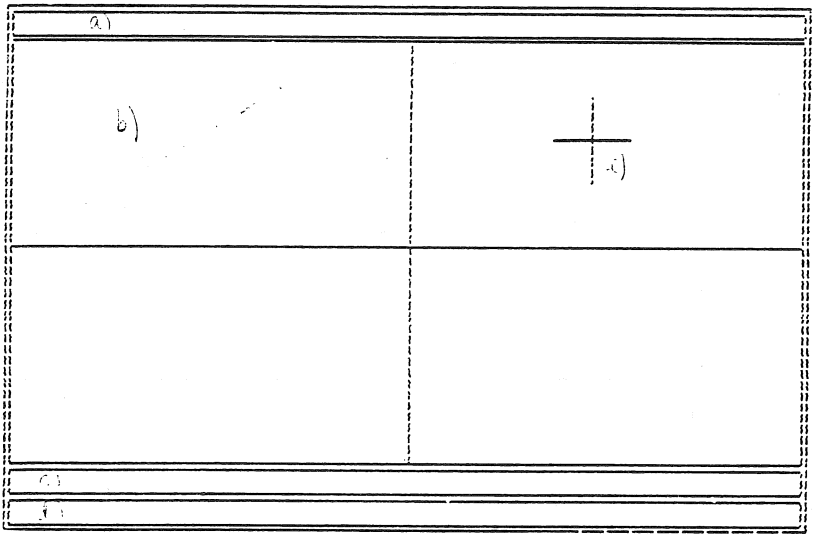

Figura 2. Layout da tela da interface

A Figura 2 apresenta o layout da tela da interface de **IMAC**. Nela vemos quatro áreas e o cursor. As áreas são: área da função atual, área de edição de comandos e área de teclas de funções especiais.

**a) A área da função atual de trabalho** serve para mostrar e expressão matemática da função que é correntemente a função de trabalho.

**b) A área de visualização gráfica da função** é utilizada para mostrar o gráfico da função atual de trabalho e de suas derivadas, assim também como a apresentação de janelas pop-up de resultados, diretórios , seleção de valores iniciais para métodos de integração e cálculo de zeros e outras atividades.

**c) A área de edição de comandos** é a área da interface em que o usuário interage com o computador através da edição de comandos.

**d) A área de teclas de funções especiais** serve para dois propósitos. Primeiramente como uma lembrança visual ao usuário de uma série de funções importantes atribuidas as teclas. Todas as funções são disponíveis também através da linguagem de comando, mas dada a sua importância e a frequência que as mesmas podem vir a serem utilizadas, a atribuição direta destas funções a algumas teclas facilitam e motivam a sua utilização, como por exemplo:

**F1 = Help  F2 = Imprimir  F3 = Lista  F4 = Ampliar**

Em segundo lugar, a área de teclas de funções especiais serve para dar feedback ao usuário das atividades realizadas pelo programa, como por exemplo :

**Gravando C:\IMAC\EXP001.EXP...**
**Integrando função pelo método de SIMPSON. Aguarde...**

**e) O cursor** é utilizado para seleção de valores numéricos e determinação de subjanelas de visualização para plotagem de determinadas regiões do gráfico da **função de trabalho.**

## 5. CONCLUSÃO

Neste artigo, descrevemos uma interface gráfica e interativa para trabalho com métodos computacionais, cuja estrutura de diálogo usuário-computador é determinada através de uma linguagem de comando.

Além disso, mostramos que o projeto desta linguagem deve basear-se em dois aspectos fundamentais :

- Em um **modelo do usuário** bem definido, o que determina praticamente toda a definição da semântica (ações) desta linguagem.

- E que o projeto da sintaxe desta linguagem deve levar em conta fatores cognitivos (fatores humanos), ou seja, que ela tenha características tais que permitam um fácil aprendizado pelas pessoas que realmente vão utilizá-la e, como consequência, colabore para o sucessc do sistema.

## 6. REFERÊNCIAS


[1] DIVÉRIO, T. A. **Software Numérico Aplicativo e Instrucional.** Porto Alegre, PGCC da UFRGS, 1986 (dissertação de mestrado).

[2] DIVÉRIO, T.A. e Rombaldi, V. **SINAI-16. Um exemplo de software instrucional.** In: CNMAC, X, Gramado, set. 20-25, 1987.

[3] LEPMAC. Manual do usuário. Porto Alegre, PGCC da UFRGS, 1990.

[4] NEWMAN, M. W. and SPROULL F.R. **Principles of Interactive Computer Graphics.** McGraw-Hill, New York, 1979.

[5] RALSTON, A. **A First Course in Numerical Analisis.** McGraw-Hill, Tokyo, 1965.

[6] RICE, J.R. **Numerical Methods, Software and Analysis.** McGraw-Hill, New York, 1975.

[7] HORNBECK, R.W. **Numerical Methods.** Quantum Publishers, New York, 1975.

[8] HARDY, Jr. T. The Syntax of Interctive Command Languages: A framework for Design. **Software-Practice and Experience**, vol. 12, 67-75 (1982).

## APÊNDICE

### Os comandos suportados por IMAC

Este item apresenta a declaração de todos os comandos suportados na interface de **IMAC**. Os comandos e suas abreviações aparecem em negrito. Os parâmetros dos comandos devem ser sempre ser separados por no mínimo um espaco em branco. Para maior clareza e entendimento dos comandos, todos os parâmetros numéricos são acompanhados por seu tipo (inteiro ou real). O símbolo ":" é utilizado para separar um parâmetro numérico de seu respectivo tipo . A linguagem de comando de **IMAC** não é sensível ao caso, ou seja, pode-se misturar livremente caracteres maiúsculos e minúsculos. O símbolo reservado **Gr** da linguagem de comando indica que o valor numérico correspondente pode ser selecionado via cursor. Os comandos serão divididos em comandos computacionais e comandos de uso geral.

### Comandos Computacionais

### Comandos para integração numérica de funções

**Rm** | **Romberg** X1:Real|Gr X2:Real|Gr Digse:Integer
**Sm** | **Simpson** X1:Real|Gr X2:Real|Gr Digse:Integer
**Tr** | **Trapezio** X1:Real|Gr X2:Real|Gr Digse:Integer
**Qd** | **QuadGauss** X1:Real|Gr X2:Real|Gr Digse:Integer

Onde :

X1,X2 : Parâmetros que definem o limite de integração
Digse : Dígitos significativos exatos requeridos na operação

### Comandos para o cálculo de zeros de funções

**Nw** | **Newton** X1:Real|Gr Digse:Integer
**Pg** | **Pegasus** X1:Real|Gr X2:Real|Gr Digse:Integer
**Ml** | **Miller** X1:Real|Gr X2:Real|Gr X3:Real|Gr Digse:Integer
**Jr** | **Jarrat** X1:Real|Gr Digse:Integer

Onde, X1,X2,X3 : Valores que fornecem a aproximação inicial para o método em questão.

**Comando para interpolação de valores dada uma tabela de pontos**

**Nt** | **NewtonInt** X|Y Valor|Gr

Onde, X,Y : Representa o parâmetro que se quer interpolar Valor : Representa o valor do parâmetro a ser interpolado

**Comando para a resolução de equações diferenciais ordinárias**

**Edo** X1:Real X2:Real h:Real positivo Método

Onde, X1,X2 : Intervalo sobre o qual se quer procurar a solução da equação diferencial

h : Passo

Método : Método a ser utilizado na resolução da equação diferencial. Os métodos possíveis são os de Runge-Kutta4, Adams-Bashforth e Preditor e Corretor.

Para invocar este comando é preciso, inicialmente, ter criado um sistema de equações diferenciais com o comando **CriaEdo** Grau.

**Comandos para resolução de sistemas de equações lineares**

**Gs** | **Gauss** -> Permite a resolução do sistema de equações lineares correntemente na memória através do método de eliminação de Gauss.

**Gl** | **Gauss-Seidel** -> Permite a resolução do sistema de equações lineares correntemente na memória através do método iterativo de Gauss-Seidel.

**Br** | **Barnac** -> Permite a resolução do sistema de equações lineares correntemente na memória através do método de Barnachevsky.

**Sl** ! **Salva** Opção Nome_Arquivo

Onde Opção -> **Edo** ! **Tab** ! **Mat** !**Sis**!**Fun**

**Edo** -> Permite salvar um sistema de equações diferenciais previamente criado com o comando **Cria Edo.** O sistema é gravado com o nome de Nome_Arquivo. A extensão **.EDO** é automaticamente verificada pelo programa.

**Tab** -> Permite salvar uma tabela de pontos previamente criada com o comando **Cria Tab.** A tabela é gravada com o nome de Nome_Arquivo. A extensão **.TAB** é automaticamente verificada pelo programa.

**Mat** -> Permite salvar uma matriz previamente criada com o comando **Cria Mat.** A matriz é gravada com o nome de Nome_Arquivo. A extensão **.MAT** é automaticamente verificada pelo programa.

**Sis** -> Permite salvar um sistema de equaçoes lineares previamente criado com o comando **Cria Sis.** O sistema é gravado com o nome de Nome_Arquivo. A extensão **.SIS** é automaticamente verificada pelo programa.

**Fun** -> Permite salvar uma função de trabalho previamente criada com o comando **Cria Fun.** A função é gravada com o nome de Nome_Arquivo. A extensão **.EXP** é automaticamente verificada pelo programa.

**Co** ! **Car Edo** Nome_Arquivo -> Permite carregar um sistema de equações diferenciais para a memória que tenha sido gravado com o comando **SalvaEdo.**

**Cr** ! **Carrega** Opção Nome_Arquivo -> Permite carregar a memória do programa um arquivo previamente gerado pelo sistema.

Onde Opção -> **Tab** ! **Edo** ! **Fun** ! **Sis** ! **Mat**

**Ed** ! **Edit** Opção -> Permite editar um **objeto** da interface de **IMAC** definido pelo parâmetro opção.

## Comandos de Uso Geral

**Lm** ! **Limpa** -> Este comando permite limpar a área de visualização gráfica da função.

**Cr** ! **Cria** Opção

Onde Opção -> **Tab** ! **Fun** ! **Edo** ! **Sis**!**Mat**

**Tab** N:Integer -> Esta opção permite criar uma tabela de pontos (pares ordenados) onde o usuário pode efetuar operações de ajuste e interpolação. O valor de N deve ser <= 50.

**Fun** função -> Esta opção possibilita a criação de uma função de trabalho. O parâmetro função significa uma expressão matemática. A invocação deste comando também causa a plotagem automática da função na área de visualização gráfica da função.

**Edo** N:Integer -> Permite a criação de um sistema de equações diferenciais para ser resolvido pelo comando **Edo.** O número de equações do sistema está limitado a um máximo de 10, ou seja, é possível resolver equações diferenciais ordinárias de até décima ordem. Deve-se entrar uma condição inicial para cada equação após a respectiva equação ter sido digitada. O símbolo ";" serve como delimitador da equação e de sua respectiva condição inicial.

Exemplo :

$$y' = 3x + 4 \; ; \; y(0) = 2$$

**Mat** Ordem -> Permite e entrada de uma matriz de ordem dada pelo parâmetro **Ordem.** A ordem máxima permitida é de 20.

**Sis** Ordem -> Permite a entrada de um sistema de equações lineares cuja ordem é dada pelo parâmetro **Ordem.** A ordem máxima permitida é de 20.

**Sn** ! **Salva Fun** Nome_arquivo -> Este comando permite gravar a função atual de trabalho em disco. Nome_arquivo pode também indicar o drive e o path. A extensão **.FUN** é automaticamente verificada pelo programa.

Onde Opção -> **Tab** | **Edo** | **Fun** | **Sis** | **Mat**

**Sn** | **SelFun** -> Este comando permite ao usuário escolher uma nova função de trabalho a partir de uma janela que mostra as últimas (até um máximo de dez) funções criadas na interface.

**Pl** | **Plot** D1 | D2 | Fun -> Este comando permite ao usuário plotar a função atual de trabalho e/ou sua derivada primeira e segunda.

**Cl** | **Calc** D1 | D2 Gr | Num_Real -> Permite calcular o valor da derivada primeira ou segunda de uma **função de trabalho.** Este cálculo pode ser feito através da especificação de um número real ou então de maneira gráfica, onde a medida que o usuário "caminha" com o cursor sobre a função, o valor da derivada (primeira ou segunda) é automaticamente mostrado.

**Im** | **Imprime** Lan | Port -> Este comando permite ao usuário fazer uma cópia gráfica da tela no modo **Landscape** ou no modo **Portrait.** No modo Portrait o eixo C da tela é plotado ao longo da largura da folha do papel e o eixo X ao longo do comprimento da folha do papel, produzindo uma imagem que ocupa metade da folha. Por outro lado, no modo Landscape, o eixo X da tela é plotado de acordo com o comprimento da folha e o eixo Y de acordo com a largura, produzindo, desta forma, uma imagem do tamanho de uma folha.

**Cn** | **Const** -> Este comando permite ao usuário criar uma tabela de constantes, que podem ser utilizadas tanto nas funções quanto nos sistemas de equações diferenciais. Cada constante tem um nome (máximo de dez caracteres) e um valor associado. A invocação deste comando causa o aparecimento de uma janela pop-up que contém todas as constantes já definidas pelo usuário. É permitido a alteração, deleção e inserção de constantes. Entretanto, existem algumas constantes que são pré-definidas pela interface, e que não podem ser alteradas nem deletadas.

**Ls ! Lista** -> Causa o aparecimento de uma janela que contém os últimos comandos (até um máximo de dez) editados pelo usuário e que são sintaticamente válidos de acordo com a linguagem de comando. A seleção de algum comando retira a janela de apresentação dos comandos e deixa o comando selecionado na área de edição, pronto para modificações ou até invocação imediata.

**Hp ! Help** -> Causa o aparecimento de uma janela que permite (através de scrolling vertical) a visualização de todos os comandos da interface. A seleção de algum item causa o aparecimento de uma outra janela (em offset em relação a anterior) com uma breve descrição do comando e também a respeito de sua declaração.

**Al ! AltDir** S:String -> Permite ao usuário alterar o diretório corrente para um path especificado por S. Se S especificar um drive, então o drive corrente também é selecionado.

**Dr ! Diret** Path + Máscara -> Permite visualizar todos os arquivos que coincidam com o argumento Path + Máscara. É permitido a navegação vertical e horizontal entre os arquivos, de tal maneira que o usuário possa selecionar algum arquivo do sistema (funções, tabela de pontos, sistema de equações diferenciais, sistemas de equações lineares e matrizes) para a memória.

**Ex ! Eixo** X!Y On!Off -> Este comando permite que retire o eixo X e/ou o eixo Y da janela de visualização da função.

**Es ! Escala** X!Y V1:Real V2:Real -> Este comando permite que o usuário altere os valores da escala X e Y para os correspondentes valores numéricos de V1 e V2.

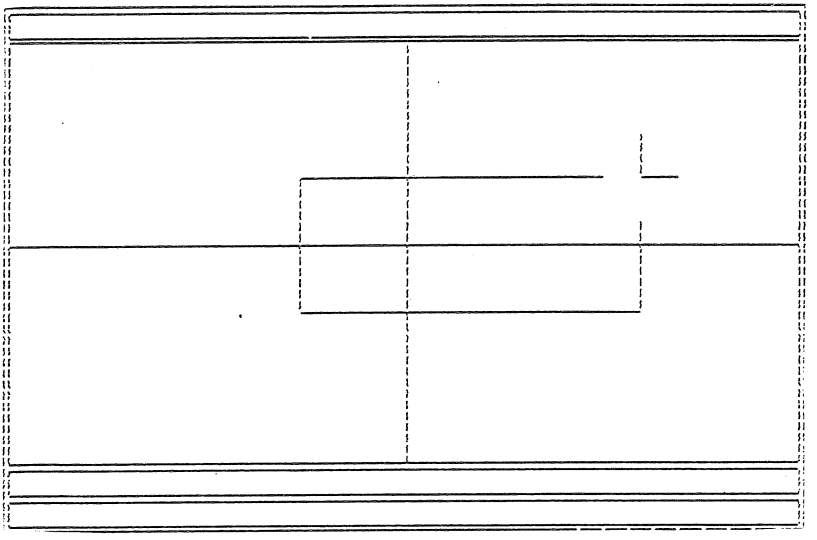

Figura 3. Criação de uma subjanela da função de trabalho

**Am** ! **Amplia** Gr ! X1:Real Y1:Real X2:Real Y2:Real -> Este comando permite ao usuário selecionar uma subjanela de visualização (Figura 3) para a função atual de trabalho. As coordenadas desta subjanela podem ser determinadas diretamente pelos parâmetros X1,Y1,X2 e Y2. As coordenadas também podem ser determinadas graficamente através de um cursor que "caminha" sobre a função. O símbolo **Gr** identifica esta opção.

**Ms** ! **MostraEsc** -> Este comando causa o aparecimento de uma janela com os valores das coordenadas atuais da janela de visualização da função.