PUC

Monografias em Ciência da Computação
nº 8/92

Tópicos Especiais sobre os Sistemas de Computação Algébrica

Paulo Roberto Trales

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA

Monografias em Ciência da Computação, Nº 8/92
Editor: Carlos J. P. Lucena

Março, 1992

Tópicos Especiais sobre os Sistemas de Computação Algébrica*

Paulo Roberto Trales

* Trabalho patrocinado pela Secretaria de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

In charge of publications:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC Rio - Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
Brasil

Tel.:(021)529-9386 Telex:31078 Fax:(021)511-5645
E-mail:rosane@inf.puc-rio.br

Abstract

In this work a critical study is performed on the computer algebra systems implemented in Lisp and C most widely in use. Besides disseminating the area, the motivation is contributing to the best use of available software capabilities and potentialities. Generations and families of this kind of software are presented, together with specific habilities, sintax forms, equipments, problems and bugs among other comparative analyses. Moreover topics about other systems and sugestions for their applications according to users interest are considered.

Keywords: Computer algebra systems, Reduce, Macsyma, Derive, Mathematica, Maple, Capabilities, Habilities, Users, Alghoritms, Sintax, Software.

Resumo

O trabalho consiste em um estudo crítico sobre a performance dos sistemas de computação algébrica implementados em LISP e C mais divulgados, e tem como objetivo difundir a área e otimizar o uso de suas capacidades e potencialidades. Nele são apresentadas gerações e famílias desse tipo de software, habilidades específicas, modos de sintaxe, equipamentos, problemas e bugs e outras análises comparativas. Tópicos sobre outros sistemas e sugestões para aplicações dos programas de acordo com interesses do usuário são também considerados.

Palavras-chave: Sistemas de computação algébrica, Reduce, Macsyma, Derive, Mathematica, Maple, Capacidades, Habilidades, Usuários, Algoritmos, Sintaxe, Software.

PREFÁCIO

A idéia de fazermos o presente estudo surgiu através das muitas dificuldades encontradas na divulgação desse tipo de software, na procura de fontes de consulta e na utilização efetiva de alguns sistemas de computação algébrica. Tais sistemas, embora difundidos, ainda têm seu acesso restrito pela falta de detalhamento, às vezes básico mas necessário, nos textos existentes sobre o assunto.

Tratamos mais detidamente de sistemas que usam as linguagens de implementação LISP e C por serem mais comumente usados, e por fornecerem praticamente todas as ferramentas exigidas pela comunidade acadêmica usuária.

Esse estudo não pretende vir a ser um documento completo sob nenhuma hipótese, visto que essa matéria continua em evolução permanente, e com certeza apresentará consideráveis progressos a curto prazo, pois pesquisas nessa área se desenvolvem de maneira acelerada em vários centros científicos em todos os continentes.

Tivemos por objetivo reunir e comparar informações não encontradas facilmente na literatura usual, elaborar mais um auxílio às pesquisas e à divulgação dessa nova disciplina emergente e tentar ampliar o acesso ao uso dessa tecnologia no nosso país. Finalmente, nos propusemos a fornecer esclarecimentos com relação ao melhor aproveitamento desse ou daquele sistema, de acordo com o tipo de trabalho desejado pelo usuário.

SUMÁRIO

INTRODUÇÃO

A computação científica usando computadores pode ser dividida em duas áreas: computação numérica e computação simbólica. Em computação numérica o computador é utilizado como um triturador de números, são calculados números e somente números, em geral somente resultados aproximados podem ser obtidos.

Na computação simbólica, também conhecida como computação algébrica, lápis e papel são trocados por programas interativos de computador, os chamados sistemas de computação algébrica. Esses softwares, permitem a seus usuários operar não somente com números, mas também com símbolos, equações, polinômios, séries de Taylor e outras fórmulas, com a vantagem fundamental de enfatizar a exatidão dos cálculos.

Os sistemas de computação algébrica podem ser caracterizados como sistemas espertos, que fornecem a seus usuários acesso a várias técnicas matemáticas. No topo do conhecimento já construído, novos algoritmos vem sendo acrescentados por pesquisadores da área, aumentando dessa forma o escopo dos programas; podemos dizer que os sistemas de computação algébrica são realmente indispensáveis em pesquisa moderna pura e aplicada e em educação.

Computação algébrica é o nome que se dá a tecnologia de manipulação de fórmulas matemáticas pelo computador. A utilização dessa tecnologia era limitada a pesquisadores, mas a rápida evolução do hardware e do software veio a torná-la accessível a todo cientista, matemático, físico e engenheiro que trabalhe com métodos matemáticos.

Um dos primeiros passos dados na direção de se automatizar efetivamente um software - para se trabalhar em algo parecido com computação algébrica - já tem algum tempo, mais precisamente, esses passos foram dados por Kahrimanian e Nolan em 1953.

De 1960 em diante, vários programas apareceram com a intenção de mostrar que no campo científico, só se poderia ir adiante na área puramente numérica com o uso dos computadores, e já se vislumbrou que tal tipo de software teria que ser provido de um sistema completo, que incluisse um método com uma estrutura muito especial para representação de dados não numéricos, uma linguagem que tornasse possível sua manipulação e uma biblioteca de funções para as operações básicas necessárias.

E daí apareceram sistemas que funcionaram bem, e os mais conhecidos, utilizados e divulgados são descritos ou citados nesse trabalho.

Escrevendo, desenvolvendo e mesmo usando esses sistemas, gradualmente emergiu uma disciplina científica autônoma chamada computação algébrica, obviamente baseada na ciência da computação.

Por um longo tempo também, a utilização desses sistemas se limitou a um tipo de máquina, e obviamente essa limitação não ajudou a expansão do conhecimento deles, o que tornou suas

possibilidades e potencialidades por tanto tempo ignoradas pelo mundo científico e industrial.

A linguagem LISP data desse período e abriu caminho para as primeiras demonstrações espetaculares das seguintes possibilidades: integração formal e provas de teoremas. No mesmo filão algumas pessoas viram muito rapidamente que poderiam perguntar à máquina como realizar operações algébricas tediosas, que seriam muito úteis para cálculos numéricos futuros: a expansão de expressões polinomiais, diferenciação formal etc.

O aparecimento de sistemas em tempo compartilhado contribuiu de grande maneira para a generalização de programas de cálculos algébricos, e tiveram gradualmente genuínos sub-sistemas que podem ser usados numa linguagem bem próxima as linguagens algoritmicas usuais.

LISP e sua grande quantidade de dialetos e variações não puderam evitar um sistema tal como o REDUCE de estar disponível a um grande número de computadores e work-stations em todo o mundo. Com uma extensão menor, MACSYMA vem sendo adaptado para uso comercial em certos computadores, e já está longe de ser considerado um sistema inaccessível.

Recentemente vários sistemas para micro-computadores apareceram, e ainda que limitados, no mínimo nos dão a impressão que muito poderá ser feito nessa matéria: muMATH, DERIVE, MathCAD, Mathlab, TKSolver!, GAUSS e MATHPERT são alguns exemplos.

O que é notável é que a maioria dos sistemas de computação algébrica existentes: REDUCE, MACSYMA, MUMATH , a família LAM, SHEEP, RIEMANN II, o tão esperado a nível de usuário comum, SCRACTHPAD e mais de uma dezena, além desses, foram escritos em LISP.

A mais intuitiva, embora algumas vezes restrita aproximação dos sistemas de computação algébrica é dizer que são feitos através da manipulação de fórmulas científicas e de engenharia utilizadas sistematicamente no dia a dia.

Uma fórmula matemática descrita em uma das linguagens usuais (FORTRAN, PASCAL, BASIC, ...) só pode ser calculada numericamente, uma vez que às variáveis e aos parâmetros tenham sidos dados valores numéricos, enquanto que numa linguagem que permita manipulações algébricas, a mesma fórmula pode ser calculada numericamente, mas acima de tudo pode ser o objeto de transformações formais: diferenciação, desenvolvimento em séries, várias expansões e mesmo integração.

Embora existam diferenças entre os sistemas, sintaxe não é o principal problema de computação algébrica. Em geral, algumas horas e um pouco de treino são perfeitamente adequados. A sintaxe é dita ser maior que a de PASCAL. É claro que existe uma instrução de transferência, a idéia de funções de chamada (COMANDOS), um conjunto bastante rico de estruturas de controle (IF, DO, WHILE, REPEAT, ETC.), possibilidades de declaração de procedimentos...; de um modo geral, todo o arsenal de linguagens de programação exigido para se escrever algoritmos.

Nesse trabalho também apresentamos dois sistemas de computação algébrica conhecidos pela comunidade usuária que utilizam a linguagem C como linguagem de implementação.

A linguagem C é uma década mais "jovem" do que LISP e vem prestando serviços fantásticos

em várias áreas; C é uma linguagem de propósito geral, e embora tenha sido criada com o objetivo de ser uma linguagem de programação de sistemas operacionais e utilitários, pelas suas facilidades de acesso a nível de máquina e eficiência do código gerado, tem-se mostrado extremamente útil para escrever um grande número de aplicações numéricas, de processamento de textos e de programas para banco de dados.

Com todo esse poderio C também colaborou sobremaneira com os sistemas de computação algébrica com o MATHEMATICA e com o MAPLE que já se tornaram um sucesso, além de outros ainda em fase de desenvolvimento.

Antes do estudo de cada módulo de sistemas que utilizam uma das duas linguagens acima citadas, apresentamos um breve resumo sobre aquelas linguagens, chamadas no texto de tópicos sobre LISP e tópicos sobre C.

É possível também classificarmos os sistemas de computação algébrica em três grupos, que refletem seu desenvolvimento histórico, de modo análogo ao das linguagens de programação convencionais, que são geralmente designadas por primeira, segunda, terceira, quarta ou quinta geração.

Os sistemas do primeiro grupo são os descendentes das primeiras tentativas de se escrever programas em computação algébrica, e tinham como tendência serem concebidos para problemas específicos em campos tais como física matemática e química teórica. Como eram de propósito específico, operavam com cálculos padrão o que os tornava bastante rápidos. Embora eles tenham tido um impacto significante em cada um dos ramos escolhidos, muitos deles não eram simples de serem utilizados. A seguir apresentamos alguns softwares desse grupo com suas principais áreas de atuação.

ASHMEDAI e SCHOONSHIP - eletrodinâmica quântica

CAMAL - teoria lunar e relatividade geral

SCHOONSHIP - física de alta energia

SHEEP - manipulação indicial de tensores

TRIGMAN - mecânica celeste

ALTRAN - equações limitadas a funções racionais e polinomiais

ALDES-SAC/2 - um programa de propósito específico que pode auxiliar no desenvolvimento de novos programas de computação algébrica.

Nesse grupo podemos ainda incluir os sistemas SAC I, SIN, ALPAK, FORMAC, SAINT, CAYLEY, MATHLAB, CLAM, POLYNOM, ADC, ORTOCARTAN e outros menos conhecidos.

Os sistemas do segundo grupo são de propósito mais geral e não foram escritos apenas para determinação de soluções de problemas particulares e especializados, eles dão ao investigador uma vasta demonstração das capacidades e possibilidades matemáticas. Tais sistemas podem executar as quatro operações aritméticas básicas, fazem integração definida e indefinida e resolvem equações, incluindo equações diferenciais ordinárias que expressam algebricamente como

a mudança em uma variável depende de mudanças em outras. Os programas podem também resolver sistemas de equações algébricas lineares e não lineares, diferenciações, simplificações, calculam somas finitas e infinitas, como séries de Taylor e operam todas as funções dos sistemas de propósito específico, embora não tenham tanta velocidade e poder. Obviamente, estamos apresentando questões matemáticas tratadas pelos sistemas do segundo grupo de um modo geral, um maior aprofundamento nas potencialidades de quatro desses sistemas, é apresentado ao longo do nosso trabalho. As principais características desse grupo são de sistemas não restritos a aplicações específicas, serem interativos, portáteis e de fácil utilização.

Alguns dos membros mais conhecidos desse grupo são:

MACSYMA, REDUCE, SCRATCHPAD, MAPLE, SMP e MATHEMATICA.

Os sistemas do terceiro e último grupo tem as seguintes características; operam em microcomputadores, requerem pouca memória, são simples de serem usados, têm um grande ramo de funções internas construídas e são mais lentos e menos compreensíveis que os sistemas gerais desenhados para computadores maiores, pois o tamanho, memória e velocidade dos micros não permitem que eles ataquem problemas mais complexos, mas mesmo assim podem executar cálculos mais rapidamente do que muitos matemáticos.

Existem dois membros a se destacar nesse grupo:

muMATH e DERIVE.

Estes dois softwares, mesmo não tendo a flexibilidade e a abrangência dos sistemas gerais mais poderosos , já entraram para a "história", pois se tornaram bastante conhecidos por serem usados em ensino na maioria das universidades. muMATH e DERIVE estão apenas disponíveis para computadores MS-DOS; e muMATH não faz uso de mais do que 256Kb (Kilobytes) de memória principal. Entretanto tais sistemas - ou seus descendentes mais elaborados - encontram espaço em PC's com pequena quantidade de memória, e talvez até o fim de 1992 já tenhamos outros sistemas de computação algébrica de bolso. A Soft Warehouse, responsável pelas pesquisas e distribuição do DERIVE, vem trabalhando fortemente essa idéia, tendo se associado nesse sentido à Hewlett Packard, e de modo pioneiro lançou no segundo semestre de 1991 o primeiro sistema de computação algébrica de bolso em uma HP-95-LX.

Nesse estudo analisaremos comparativamente e com detalhes os sistemas:

REDUCE, MACSYMA, DERIVE, MATHEMATICA e MAPLE.

Essa análise não pretende vir a ser completa e nem sonha em ser definitiva, pois obviamente envolve uma série de parâmetros como: versão estudada, equipamento disponível, desenvolvimento atual de cada sistema e etc. Tentamos mostrar no trabalho, a maneira que nos pareceu mais simples de se tratar esse problema delicado que é o de comparar pacotes desse tipo. Para cada um dos cinco sistemas acima citados, apresentamos um sumário dividido de 10 a 15 seções - dependendo naturalmente do programa onde são mostradas capacidades, modos de uso da sintaxe e habilidades específicas - que vão da introdução às informações adicionais sobre cada um dos softwares. Sempre recomendamos a utilização dos manuais de cada um dos sistemas

para o usuário efetivo desse ou daquele software, com o objetivo de aprofundamento e domínio de mais capacidades de cada um deles, pois não é possível se tratar com abrangência e profundidade em 30 ou 40 páginas, o que alguns manuais trazem em mais de 1000.

No final do trabalho são também apresentadas tabelas, considerações gerais, tópicos sobre outros sistemas, além de um postscriptum, que dão uma visão global de aplicações desses e de outros sistemas de computação algébrica de nossa época.

Voltando ao problema das linguagens de implementação dos softwares de computação algébrica, poderíamos finalizar essa introdução colocando algumas questões para reflexão.

Será coincidência que dois dos grandes sistemas de computação algébrica mais recentes usaram C como linguagem de implementação, e C virá a tomar o lugar de LISP ?

LISP voltará a ocupar novamente o top, como linguagem de implementação do promissor SCRATCHPAD ? [1]

Como para muitos pesquisadores o domínio da linguagem LISP é mais "difícil" do que o da linguagem C, podemos afirmar que a linguagem de implementação dos sistemas de computação algébrica dos anos 90 será o C^{++} ?

Certamente o tempo nos responderá essas questões.

[1]SCRATCHPAD foi rebatizado pela NAG, sua distribuidora atual, de AXIOM - e é um software com estrutura totalmente diferente de seus antecessores, mas ainda não totalmente disponível, e que segundo vários autores será o protótipo da nova geração de sistemas de computação algébrica.

Capítulo 1

TÓPICOS SOBRE A LINGUAGEM LISP

1.1 Histórico

A linguagem LISP foi pela primeira vez descrita no texto " Recursive Functions of Simbolic Expressions and Their Computation by Machine " por John McCarthy em 1960, e implementada no Massachusets Institute of Tecnology. LISP é um acrônimo de LISt Processing.

1.2 Objetivos da Linguagem

LISP ainda é a linguagem de programação mais utilizada pela comunidade de I. A., e vários ambientes em LISP vêm sendo usados para a construção de sistemas especialistas e até para desenvolvimento de ambientes de engenharia de software em geral.

A linguagem LISP é designada primariamente para processamento de dados simbólicos. LISP especializa-se na manipulação de expressões simbólicas, e, foi usada para Cálculos Simbólicos em Cálculo Diferencial e Integral, Teoria dos Circuitos Elétricos, Lógica Matemática, Jogos, Robótica, Processamento de Linguagem Natural, Verificação de Programas e Computação Simbólica.

A computação simbólica, ou como chamamos no nosso estudo computação algébrica, tem em LISP sua viga mestra. Os sistemas mostrados com mais detalhamento nos capítulos seguintes, o REDUCE, o MACSYMA e o DERIVE, foram implementados nessa linguagem que contribuiu sobremaneira com a área. Mais de 60 % dos softwares de computação algébrica usam LISP, ou

algum de seus dialetos, como linguagem base.

Existem diferentes implementações (dialetos LISP), contudo não existe um LISP padrão. Como exemplo podemos citar: Common LISP, muLISP, InterLISP, PSL, RLISP e outros; alguns de suma importância na construção de softwares de computação algébrica.

1.3 Características da Linguagem

Uma importante técnica de gerenciamento de armazenamento de dados chamada "Garbage Collection", foi implementada, pela primeira vez, em LISP.

A estrutura de dados básica em LISP é formada por átomos e listas, que definiremos posteriormente. Aquela diferença citada no início do trabalho se apresenta de três maneiras.

A primeira é na natureza dos dados. Em LISP todos os dados vêm na forma de expressões simbólicas, referidas em algumas bibliografias como S-expressões, e em outras, inclusive no nosso texto, como expressões, que são de comprimento indefinido e possuem um tipo de ramificação de árvore. Por essa razão sub-expressões significantes podem ser lidas isoladas. No sistema de programação LISP, grande quantidade de memória acessível é usada para armazenar expressões na forma de listas. Esse tipo de organização de memória liberta o programador da necessidade de alocação de armazenamento para diferentes seções do seu programa.

A segunda é a própria origem da linguagem LISP que sempre especifica de que maneira a expressão deve ser processada. Isso consiste de funções recursivas das expressões. A *meta linguagem* é chamada sempre que a notação para a escrita de funções recursivas das expressões estejam, elas mesmas, fora da notação das expressões (chamadas primeiramente por John McCarthy de M-expressões).

A terceira é que LISP pode interpretar e executar programas escritos na forma de expressões, daí, como linguagem de máquina, e ao contrário de muitas outras linguagens de alto nível pode ser usada para gerar programas de execução suplementar.

Programas em LISP são eles próprios listas. É simples construir programas LISP que tomam programas LISP como entrada para produzir versões transformadas destes. A grande vantagem de tratar dados e funções uniformemente como listas é que podemos escrever funções que modificam outras funções, tratando essas últimas como se fossem dados. Programas em LISP podem ser assim auto-modificáveis durante a execução. Além disso, o interpretador de LISP - que é um programa para analisar e executar programas escritos nessa linguagem - é usualmente escrito na própria linguagem LISP.

LISP é altamente interativo. Expressões LISP ou programas podem ser imediatamente executados, sem qualquer pré-processamento pelo interpretador.

LISP é uma linguagem recursiva; qualquer programa pode ser definido em termos de si mesmo,

a programação em LISP usa a recursividade como principal estrutura de controle, ao passo que a maior parte das LP's utilizam as iterações.

O gerenciamento da memória é automático; listas que não estão correntemente em uso são alocadas para o sistema.

A estrutura de dados do LISP se baseia na equivalência entre dados e programas que permitem que dados possam ser executados como programas e que programas possam ser modificados como dados.

1.4 Procedimentos e Construção de Expressões

Notemos que (+ 3.14 2.71) tem três elementos: + , 3.14 e 2.71. Em LISP, o que se deve fazer (o procedimento) é sempre especificado primeiramente seguido pelos objetos com os quais o procedimento será executado, objetos esses chamados argumentos.

As expressões em LISP são escritas na forma polonesa ou pré-fixada: (Função Arg1 Arg2 ...), daí no exemplo fornecido 3.14 e 2.71 são os argumentos dados para o procedimento denominado + .

Conceitos Elementares:

- Um *procedimento* é a entidade básica em LISP que especifica como alguma coisa deve ser feita.
- Um procedimento como + , fornecido pelo LISP, é chamado uma *primitiva.*
- Uma coleção de procedimentos que trabalham juntos é chamado um *programa.*
- Uma descrição abstrata de um procedimento ou programa, não expressa em uma linguagem de programação é chamado um *algoritmo.* Tal descrição pode, por exemplo, ser dada em inglês ou como um diagrama de blocos.
- Um procedimento ou um programa pode ser considerado uma implementação de um algoritmo em uma linguagem de programação particular.

1.5 Expressões Simbólicas

O tipo mais elementar de uma expressão simbólica é o *átomo.*

Elementos Básicos:

- Objetos indivisíveis como 27, 3.14 e + que têm significados óbvios, assim como FOO, B27 e HYPHENATED-OBJECT são chamados *átomos.*
- Átomos como 27 e 3.14 são chamados *átomos numéricos* ou abreviadamente, *números.*
- Átomos como FOO , B27, HYPHENATED-OBJECT, FIRST e + são chamados *átomos simbólicos* ou simplesmente *símbolos.*
- Uma *lista* consiste de um parênteses à esquerda seguido por zero ou mais átomos ou listas, seguido de um parênteses à direita.

Exemplos:

(A (B C D) E (F) G) -> lista de dados, não avaliável

sub-listas

(CAR X) -> função em LISP

func. arg.

A função QUOTE é usada para indicar quando uma lista deve ser ou não avaliada. Assim toda lista de dados (não avaliável) deve vir precedida por QUOTE ou somente " ' " dependendo do dialeto em que estivermos trabalhando.

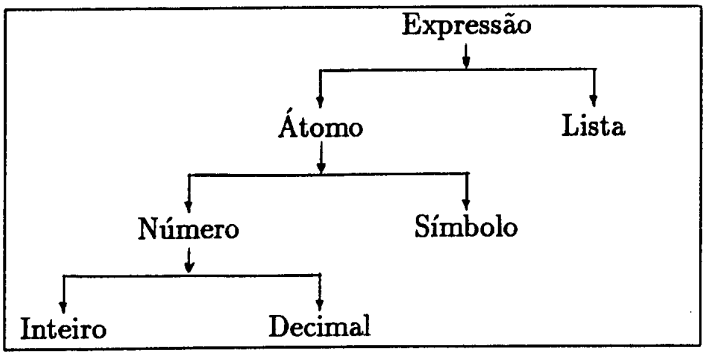
Ambos, átomos e listas, são chamados *expressões simbólicas* ou simplesmente *expressões.*

Uma expressão é chamada uma *forma*, se esta é esperada ser calculada. Se é uma lista, o primeiro elemento é geralmente o nome do procedimento que pode ser usado no processo de cálculo.

O processo de computar o valor de uma forma é chamado *avaliação.* O resultado será sempre referido como o *valor retornado* pelo procedimento específico na forma.

LISP é uma linguagem de programação funcional, o que significa que as expressões são compostas de chamadas de funções. Não há programa principal como nas outras linguagens do tipo PASCAL, ou seja, não existe estrutura de bloco, nem declarações de variáveis. Toda e qualquer expressão é avaliada e o resultado é impresso imediatamente após sua avaliação.

A principal ferramenta do LISP está em usar listas encadeadas por ponteiros como tipo básico de dados para operações, o que dá a LISP o poder de simular problemas em tempo real com grande desenvoltura. Números e caracteres podem também ser manipulados com facilidade.



As relações existentes entre os vários tipos de objetos em LISP estão no diagrama acima. Uma expressão pode ser uma lista ou um átomo; um átomo pode ser um símbolo ou um número; um número pode ser inteiro ou decimal.

O propósito desse capítulo, é apresentar como é feita a manipulação de alguns símbolos básicos primitivos do LISP, para posteriormente acompanharmos como é construída a implementação de um "pedaço" de um software de computação algébrica que utiliza como linguagem de implementação o LISP.

Para fazer isso, algumas primitivas são introduzidas, pois o LISP "puro" utiliza apenas cinco funções primitivas:

CAR, CDR, CONS, ATOM e EQ

e duas formas;

COND e LAMBA .

Tudo o mais, inclusive o próprio interpretador LISP, pode ser construído com essas funções.

Exemplos da aritmética são simples, mas aritmética não mostra o talento de LISP em manipular expressões. Suponhamos uma expressão como (COMPUTADORES NOVOS SÃO EFICIENTES). Poderíamos querer tirar o primeiro elemento deixando (NOVOS SÃO EFICIENTES); ou ainda poderíamos querer colocar um novo primeiro elemento produzindo algo como (PEQUENOS COMPUTADORES NOVOS SÃO EFICIENTES). Para se fazer tais manipulações, é necessário começar com técnicas básicas de dissecar e construir listas.

Alguns exemplos ajudarão a explicar como as primitivas básicas CAR e CDR funcionam. O valor retornado por CAR é o primeiro elemento da lista dada como seu argumento:

(CAR ' (COMPUTADORES NOVOS SÃO EFICIENTES)) e [Enter] produz

COMPUTADORES

No próximo exemplo, o argumento dado ao CAR é a lista com dois elementos ((AB) C). O primeiro elemento é por si mesmo uma lista, (AB). Consequentemente (AB) é retornado por CAR.

(CAR ' ((AB) C)) e [Enter] produz

(AB)

O CDR faz o complementar do que CAR faz. Este faz retornar a lista contendo todos menos o primeiro elemento.

Exemplos:

(CDR ' (COMPUTADORES NOVOS SÃO EFICIENTES)) e [Enter] produz

(NOVOS SÃO EFICIENTES)

(CDR ' (ABC)) e [Enter] produz

(BC)

(CDR ' ((AB) C)) e [Enter] produz

(C)

Note que CDR , ao contrário de CAR , sempre retorna uma lista.Lembrando o seguinte diagrama pode ajudar a guardar a assimetria de CAR e CDR :

$$(ABC)\begin{cases}\boxed{\text{CAR}} \longrightarrow A \\ \boxed{\text{CDR}} \longrightarrow (BC)\end{cases}$$

Note também que quando CDR é aplicado a uma lista com somente um elemento, retorna a lista vazia, às vezes denotada por ( ) ou NIL.

Usaremos os fatos de que as primitivas CAR e CDR podem ser postas juntas, assim como as primitivas aritméticas. Para retirar o segundo elemento de alguma lista , a primeira primitiva a se usar é CDR e a segunda é CAR. Então se queremos o segundo elemento de (ABC), parece-nos correto escrever:

(CAR (CDR (ABC)))

Há um problema, entretanto. Queremos que CDR tome a lista (ABC) e retorne (BC), e então CAR retornaria certamente B , o segundo elemento da lista original. Mas como LISP pode saber onde a especificação de o que fazer termina, e os dados a serem manipulados começam ?

Observe a seguinte lista: (ABC)

LISP pode pensar legitimamente que A é algum tipo de procedimento , talvez algum definido pelo usuário. Similarmente, a seguinte expressão é certamente uma lista (CDR (ABC)), e seu primeiro elemento é com certeza, CDR ! Então a seguinte expressão pode muito bem resultar em uma resposta de CDR :

(CAR (CDR (ABC)))

Até onde o processo de avaliação vai em uma expressão ?

LISP precisa de ajuda ao fazer essa decisão, o usuário especifica onde parar com a avaliação

colocando um sinal, na forma de um plique, que a inibe ('). Por exemplo, a expressão seguinte retornará B:

(CAR (CDR '(ABC)))

Note que B foi retornado, pois o plique previniu LISP de pensar em (ABC) como uma forma na qual A é o nome do procedimento a ser aplicado a B e C; ao contrário, (ABC) é dado ao CDR que então retorna (BC) ao CAR, resultando finalmente B. Observe agora, que se movermos o plique alteramos o resultado; senão vejamos:

(CAR '(CDR (ABC)))

Nessa expressão LISP não tenta tirar o CDR de nada, simplesmente fornece a expressão (CDR (ABC)) ao CAR como uma lista a ser trabalhada, resultando em CDR pois CDR é o primeiro elemento.

Se deixarmos de lado o plique, teríamos como resultado uma tentativa de usar A como um procedimento. Não há procedimento por LISP, e se nada tiver sido definido pelo usuário, LISP irá responder que houve um erro de procedimento indefinido.

Às vezes é extremamente útil sabermos de algum outro caminho equivalente de parar com a avaliação, o que é feito do seguinte modo; a expressão a ser protegida contra avaliação é simplesmente colocada entre parênteses, com um parenteses à esquerda e antecedida pelo símbolo QUOTE, e à direita por outro parênteses. Por exemplo, para fazer parar a avaliação da lista (ABC) : (QUOTE (ABC)) fornece (ABC).

(CAR (CDR (QUOTE (ABC)))) e [Enter] produz:

B

Note então que '(ABC) = (QUOTE (ABC))

Nota: Alguns autores dizem que QUOTE é uma função do LISP - assim como CAR e CDR - e " ' " é uma abreviação.

CONS toma uma expressão e uma lista e retorna uma nova lista cujo primeiro elemento é a expressão e cujos outros elementos são aqueles da antiga lista. Por exemplo;

(CONS ' A ' (BC)) e [Enter] produz

(ABC)

CAR, CDR e CONS são operações básicas para selecionar componentes de listas e construir listas, com o uso do CONS qualquer lista pode ser construída elemento por elemento. Por exemplo, (CONS A (CONS B (CONS C NIL))) constrói uma lista dos três elementos referidos por A, B e C. Se L = (ABC) é uma lista, então (CAR L) é A e (CAR (CDR L)) é B e temos que (CAR (CDR (CDR L))) é C. É bastante conveniente a abreviação do uso múltiplo de CAR's e de CDR's para tornar a escrita menos pesada.

Pelo uso apropriado de CAR, CDR e CONS, qualquer lista pode ser desmembrada nos elemen-

tos que a constituem, e novas listas podem ser construídas com esses e com outros elementos.

$$(ABC)\begin{cases}\boxed{\text{CAR}} \longrightarrow A \\ \boxed{\text{CDR}} \longrightarrow (BC)\end{cases} \Longrightarrow \boxed{\text{CONS}} \longrightarrow (ABC)$$

A primitiva LIST pode ser usada para substituir uma longa sequência de operações CONS.

LIST toma qualquer número de argumentos e constrói uma lista desses argumentos onde cada um passa a ser um elemento da nova lista. Por exemplo:

(LIST '(AB) ' (CD)) e [Enter] produz

((AB) (CD))

Definições mais complicadas requerem o uso de funções chamadas *predicados*. Um predicado é um procedimento que retorna um valor que assinala Verdadeiro ou Falso; Falso é sempre assinalado por NIL e Verdadeiro pelo símbolo especial T.

A primitiva EQUAL testa dois argumentos para ver se seus valores são a mesma expressão e trabalha tanto com átomos quanto com listas. Se os valores de dois argumentos são iguais então o valor retornado por EQUAL é T; caso contrário o valor retornado é NIL. Por exemplo,

(EQUAL (+ 2 3) 3) e [Enter] produz

NIL

Existem outros predicados que determinam se seus argumentos são iguais. Por exemplo,

Nome	Propósito
EQUAL	São dois valores de argumentos a mesma expressão?
EQL	São dois valores de argumentos o mesmo símbolo ou número?
EQ	São dois valores de argumentos o mesmo símbolo?
=	São dois valores de argumentos o mesmo número?

EQUAL primeiramente testa para ver se seus argumentos satisfazem EQL; se não, tenta ver se são listas cujos elementos satisfazem EQUAL.

EQL primeiramente testa para ver se seus argumentos satisfazem EQ; se não, tenta ver se são números do mesmo tipo e valor.

EQ testa para ver se seus argumentos são representados pela mesma amostra da memória do computador, eles serão se forem símbolos iguais.

= testa para ver se seus argumentos representam os mesmos números, mesmo que não aconteça de ser o mesmo tipo de número.

LISP tem muitos predicados que testam objetos para ver se eles pertencem a um particular tipo de dados, por exemplo,

Nome	Propósito
ATOM	É um átomo?
LISTP	É uma lista?

ATOM é um predicado que testa seu argumento para ver ele é um átomo; e é Verdadeiro se seu argumento é um símbolo atômico e Falso se seu argumento é composto. Nos exemplos a seguir o valor PI é a aproximação da constante matemática π.

(ATOM ' PI) e [Enter] produz

T

(ATOM ' NIL) e [Enter] produz

NIL

LISTP testa seus argumentos para ver se formam uma lista. Por exemplo,

(LISTP ' (ISTO É UMA LISTA A QUAL II PERTENCE)) e [Enter] produz

T

(LISTP NIL) e [Enter] produz

T

Nota : NIL e a lista vazia são completamente equivalentes.

Comentário: O que apresentamos aqui sobre a linguagem LISP é, de fato, apenas uma conceituação preliminar; entretanto para maior aprofundamento destas e de outras questões sugerimos a leitura da bibliografia recomendada ao final deste capítulo, pois até algumas das implementações que virão a seguir podem utilizar , eventualmente, alguma noção não enfocada nesta seção.

1.6 Implementando Algumas Funções

Após essa introdução sumária de alguns conceitos fundamentais da linguagem LISP (a bibliografia utilizada com mais ênfase trabalhava com Common Lisp), vamos apresentar, a título de ilustração, a implementação de algumas funções. Trataremos de funções que operam tarefas bastante triviais com listas, escritas em um dos dialetos LISP, mais precisamente em MuLisp-86, e que servem dar uma idéia do que pode ser feito com manipulação de listas pela linguagem

e também para ratificar que LISP é uma poderosa ferramenta para se trabalhar em matemática de um modo geral, tanto simbolica quanto numericamente. Obviamente a implementação das funções dessa secção, e também da diferenciação podem ser feitas de maneira mais rápida e eficiente. Tente isso.

Nota: $ pode ser encarado como um superparênteses.

I) Vamos chamar de REVERSE1 uma função que recebe uma lista e a devolve na ordem inversa da entrada.

```
(DEFUN REVERSE1 (LIST1 LIST2)
((EQ LIST1 NIL) LIST2)
(REVERSE1 (CDR LIST1) (CONS (CAR LIST1) LIST2) ) )
```

Após essa implementação, basta chamar a função segundo a sintaxe abaixo, senão vejamos:

$(REVERSE1 ' (A B C D E)) e [Enter] produz

(E D C B A)

$(REVERSE1 ' (A B C (F G H))) e [Enter] produz

((F G H) C B A)

$(REVERSE1 ' (LISP EM PROGRAMA) e [Enter] produz

PROGRAMA EM LISP

Nota : Nesta implementação usamos uma lista, (LIST2) para acumular os valores; a função REVERSE1 necessitou da lista vazia (LIST2) como segundo argumento para reverter LIST1.

II) Vamos chamar de APPEND1 a função que ao receber um elemento e uma lista, devolva a lista com o elemento ao final.

```
(DEFUN APPEND1 (ATM LST)
(LIST (CONS ATM (LIST LST ))))
(DEFUN LIST (LST1 LST2)
((EQ LST1 NIL) LST2)
(LIST (CDR LST1) (CONS (CAR LST1) LST2)))
```

Exemplos:

$(APPEND1 'A '(X Y Z)) e [Enter] produz

(X Y Z A)

$(APPEND1 '0 '((3 4) (2) 1 0)) e [Enter] produz

((3 4) (2) 1 0 0)

III) Vamos chamar de SUBST1 uma função que recebe dois elementos e uma lista e substitui o primeiro pelo segundo.

(DEFUN SUBST1 (ATM1 ATM2 LIST)

((EQ LIST NIL) NIL)

((EQ ATM1 LIST) ATM2)

((LISTP LIST) (CONS (SUBST1 ATM1 ATM2 (CAR LIST))

(SUBST1 ATM1 ATM2 (CDR LIST)))) (CAR LIST))

Exemplos:

$(SUBST1 '1 '0 (2 (3 4) 1 (3 4))) e [Enter] produz

(2 (3 4) 0 (3 4))

$(SUBST1 'X 'A (SUBST1 'Y 'B '(SQRT (PLUS (TIMES X X) (TIMES Y Y))))) e [Enter] produz

(SQRT (PLUS (TIMES A A) (TIMES B B)))

IV) Vamos chamar de LENGTH1 a função que ao receber uma lista devolva o nũmero de elementos do primeiro nível e em cada nível separadamente.

(DEFUN LENGTH1 (LIST N)

((EQ LIST NIL) N)

((LISTP (CAR LST))

(CONS (LENGTH1 (CAR LIST) 0) (LENGTH1 (CDR LIST) (+N 1))))

(LENGTH1 (CDR LIST) (+ N 1)))

Exemplos:

$ LENGTH1 '(A B C) 0) e [Enter] produz

3

$LENGTH1 '((2 3) 5 ((4 7))) 0) e [Enter] produz

(2 (2 . 1) . 3)

Nota : Quando a lista tem mais de um nível, esta implementação fornece por último o número de elementos do primeiro nível, o número de elementos de cada nível separadamente, e seu primeiro elemento fornece o número de elementos do seu maior nível.

V) Vamos chamar de POSITION1 a função que recebe uma lista e um elemento e devolve a

posição do mesmo e em que nível se encontra.

```
(DEFUN POSITION1 (LIST ATM NIV POS)
((EQ LIST NIL) NIL)
((EQ ATM (CAR LIST)) (CONS NIV POS))
((LISTP (CAR LIST)) (CONS (POSITION1 (CDR LIST) ATM NIV (+ POS 1))
(POSITION1 (CAR LIST) ATM (+ NIV 1) 1)))
(POSITION1 (CDR LIST) ATM NIV (+ POS 1)))
```

Exemplos:

$POSITION1 '(A B C) 'B 1 1) e [Enter] produz

(1 . 2)

$POSITION1 '((2 3) 5 ((4 7))) '4 1 1) e [Enter] produz

(NIL NIL 3. 1)

Nota : Esta implementação fornece por último a posição do elemento, dando posteriormente o nível em que este elemento se encontra.

VI) Vamos chamar de MISMATH1 a função que ao receber duas listas, devolve T (Verdadeiro) se não são iguais e NIL (Falso) caso contrário.

```
(DEFUN MISMATH1 (LIST1 LIST2)
((AND (EQ LIST1 NIL) (EQ LIST2 NIL) NIL)
((EQ (CAR LIST1) (CAR LIST2)) (MISMATH1 (CDR LIST1) (CDR LIST2)))
((AND (LISTP (CAR LIST1)) (LISTP (CAR LIST2)))
((NOT (MISMATH1 (CAR LIST1) (CAR LIST2)))
(MISMATH1 (CDR LIST1) (CDR LIST2))
T)
T)
```

Exemplos:

$MISMATH1 '((A B) (C D)) '((A B) (C G))) e [Enter] produz

T

$MISMATH1 '((O L K I) ( ) A) '((O L K I) ( ) A) e [Enter] produz

NIL

1.7 Diferenciação Simbólica

Não podemos pensar na arte dos computadores tendo em mente um número reduzido de linguagens de computação. É fato que toda linguagem conhecida tem um ponto forte, pois caso contrário, cairia no ostracismo. No ramo de matemática simbólica LISP e seus descendentes continuam firmes como veículo de desenvolvimento de códigos simbólicos matemáticos.

Vamos apresentar nessa seção o que poderia ser chamado de um "pedaço" de um sistema de computação algébrica implementado em LISP. Com certas alterações e devidas conexões com outros "componentes", do sistema - tarefa bem mais complexa - os programas a seguir (Lista1, Lista2, ...) poderiam certamente fazer parte de um software desse tipo; nas considerações gerais que seguem essa seção falamos mais sobre esse tema.

Antes de enfocarmos os programas de diferenciação simbólica, discutiremos um mecanismo chamado a lista de associações (A-list) usado na maioria dos sistemas LISP. O A-list existe como um software que empilha variáveis "ligadas".

Uma maneira comum, utilizada com frequência, é criar uma lista de pares, cada uma contendo o nome e o valor de uma variável. Quando o valor de uma variável é requerido, o A-list é procurado de cima para baixo, e o valor da primeira ocorrência da variável é retornado. Existem várias vantagens distintas para esse tratamento: uma delas é que variáveis podem ser tornadas completamente locais com respeito a uma função. Isso é completado em virtude do fato de que a função "liga" suas variáveis antes de operar com elas e remove as "ligações" quando a função está completa. A definição modular de funções, característica que permite recursão - uma função pode chamar a si mesma um número qualquer de vezes. Existem dois tipos de funções definidas pelo usuário em LISP. Uma é a função do tipo EXPR, a qual tem seus argumentos calculados, enquanto as funções do tipo FEXPR não tem seus argumentos calculados. Dos dois argumentos passados para a função FEXPR, o primeiro é uma lista consistindo de argumentos e o segundo é o A-list. A passagem de A-list como um argumento permite a função FEXPR preservar a chamada do ambiente.

Uma lista de propriedades está associada a cada átomo. As definições das funções FEXPR e EXPR estão contidas na lista de propriedades, daí o interpretador pode propriamente calcular uma função.

O programa de diferenciação simbólica aqui mostrado foi escrito em Common LISP, em 1981 por Ronald L. Nicol - Químico e Pesquisador do Laboratório Naval de Pesquisas e da Escola Naval Nuclear dos EUA - e apresentado no artigo "Symbolic Differentiation à la LISP".

O primeiro passo para desenvolver um sistema simbólico matemático LISP utilizável é definir as funções de definição. A lista 1 abaixo contém as funções DEF e DFF. A função DEF é a função definida EXPR e DFF é a função definida FEXPR. Ambas as funções operam trocando-se a definição de função sobre a propriedade da lista do nome da função dada, sob os indicadores atômicos apropriados. Note que ambas as funções existem como funções FEXPR sobre os átomos DEF e DFF respectivamente. Funções definidas usando DEF tem seus argumentos

calculados enquanto que aquelas definidas com DFF não tem essa característica.

```
        Lista 1

(PUTPROP(QUOTE DEF)
  (QUOTE(LAMBDA(S A)
      (PUTPROP(CAR S)
        (CONS LAMBDA(CDR S))
        EXPR)))
  FEXPR)
(PUTPROP(QUOTE DFF)
  (QUOTE(LAMBDA(S A)
      (PUTPROP(CAR S)
        (CONS LAMBDA(CDR S))
        FEXPR)))
  FEXPR)
```

Quando for, ocasionalmente, necessário se usar caracteres LISP especiais num contexto diferente dos seus significados em LISP, a haspa-dupla é usada. O uso dessa característica é ilustrado no que se segue:

(QUOTE -) = 0

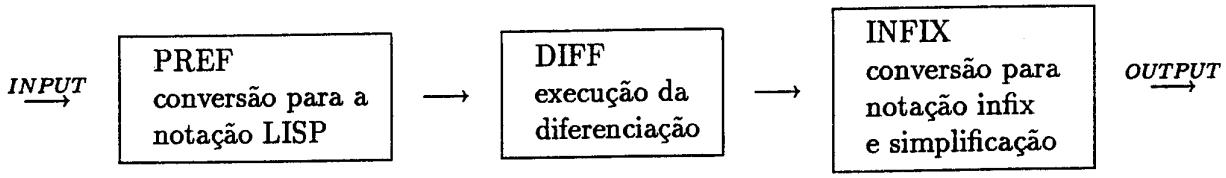
(QUOTE " - ") = -

(QUOTE +) = 0

(QUOTE " + ") = +

Além disso, as funções ADD e SUB são, respectivamente, as duas funções argumentos casos especiais das funções LISP padrão PLUS e DIFFERENCE.

O diferenciador simbólico consiste de três seções: o analisador de input, a função diferenciação e o analisador de output. Esse processo é mostrado simbolicamente na figura abaixo através de uma estrutura esqueleto de um sistema de diferenciação simbólica, onde cada bloco representa uma função recursiva que executa o que está denotado em seu interior.



O propósito do analisador de input é o de traduzir o input do usuário de notação matemática infix normal para notação prefix de Cambridge, a notação usada pelo LISP. A função diferenciação diferencia essa saída, o qual é subsequentemente convertida novamente para notação infix pelo analisador de output.

A função de input, PREF, consiste de um analisador descendente recursivo sem "backtracking" escrita por Michael Tucker e Ronald L.Nicol, que fizeram essa função em não mais de meia hora, um ponto que ratifica a clareza e a lucidez da programação LISP.

Um analisador descendente recursivo determina qual entre dois operadores tem maior prioridade; divide a entrada em duas seções, e chama a si próprio em cada um dos pedaços.

A característica "no-backtracking" significa que o analisador não tem que examinar minuciosamente para checar prioridades.

PREF foi desenhado com a flexibilidade de mudar precedência de operadores. Cada operador binário tem um poder de "ligação" à esquerda LB e um poder de "ligação" à direita RB.

Ao comparar os poderes de ligação esquerda e direita dos operadores competindo para um argumento, o analisador decide qual operador ganha o argumento. Se os poderes de ligação esquerdo e direito dos operadores são iguais, o operador à esquerda do argumento ganha.

Os valores dos poderes de ligação esquerda e direita são armazenados na lista de propriedades do operador sobre os indicadores LB e RB, respectivamente. A lista de propriedades de cada operador deve conter o nome da função equivalente LISP, sob o indicador OP.

Por exemplo, a lista de propriedades do operador multiplicação deve aparecer como se segue.

((OP,TIMES) (LB,60) (RB,80))

Precedência de operadores é inicialmente setada pela execução da função SETUP, que é apresentada na lista 2 abaixo. O usuário, entretanto, pode alterar a qualquer momento a precedência de operadores, trocando o valor LB ou RB de um operador pelo significado da função LISP PUTPROP. Na lista 2 é apresentada a definição da função SETUP a qual atribui potências esquerda e direita aos operadores binários e troca as propriedades apropriadas no indicador OP para operadores unários. A função auxiliar MAPCAR mapeia o segundo argumento em CARs sucessivos do primeiro argumento.

Lista 2

```
(DEF SETUP( )
  (PROG( )
    (MAPCAR(QUOTE((| 90 100 EXPT)
                  (* 60 80 TIMES)
                  (/ 50 70 DIV)
```

```
                ("+" 20 40 ADD)
                ("-" 10 30 SUB)))
    (QUOTE(LAMBDA(L)
                 (PROG( )
                   (PUTPROP(CAR L)
                     (CAR(CDR L))
                     (QUOTE LB))
                   (PUTPROP(CAR L)
                     (CAR(CDR(CDR L)))
                     (QUOTE RB))
                   (PUTPROP(CAR L)
                     (CAR(CDR(CDR(CDR L))))
                     (QUOTE OP))))))
(PUTPROP(QUOTE COS)
  (QUOTE COS)
  (QUOTE OP))
(PUTPROP(QUOTE SIN)
  (QUOTE SIN)
  (QUOTE OP))
(PUTPROP(QUOTE EXP)
  (QUOTE EXP)
  (QUOTE OP))
(PUTPROP(QUOTE LOG)
  (QUOTE LOG)
  (QUOTE OP))))
```

Operadores unários e funções operadores simples são tratados colocando-se a função dentro de parenteses.

A lista 3 apresenta a definição do analisador de input recursivo PREF. O analisador é definido sem "backtracking" o que minimiza tanto o uso de memória quanto exigências de tempo.

Lista 3

```
(DEF PREF(E)
  (COND((ATOM E)E)
    ((NULL(CDR E))(PREF(CAR E)))
    ((COND((ATOM(CAR E))(GET(CAR E)(QUOTE OP))))
      (COND((EQ(CAR E)(QUOTE"_"))
```

```
      (LIST(QUOTE SUB)
        0
        (PREF(CDR E))))
     (T(LIST(GET(CAR E)(QUOTE OP))
         (PREF(CDR E))))))
((EQ(CAR(CDR E))(QUOTE |))
  (LIST(QUOTE FAC)
    (PREF(CAR E))))
((NULL(CDR(CDR(CDR E))))
  (LIST(GET(CAR(CDR E))(QUOTE OP))
    (PREF(CAR E))
    (PREF(CDR(CDR E)))))
((GREATER(GET(CAR(CDR(CDR(CDR E))))(QUOTE LB))
     (GET(CAR(CDR E))(QUOTE RB)))
  (LIST(GET(CAR(CDR E))(QUOTE OP))
    (PREF(CAR E))
    (PREF(CDR(CDR E)))))
(T(LIST(GET(CAR(CDR(CDR(CDR E))))(QUOTE OP))
    (PREF(LIST(CAR E)
       (CAR(CDR E))
       (CAR(CDR(CDR E)))))
    (PREF(CDR(CDR(CDR(CDR E)))))))))
```

O poder dessa função reside na sua habilidade de recursivamente chamar a si própria. A tabela seguinte contém exemplo de input e output da função PREF, que converte notação infix em notação prefix.

Input	Output
(PREF(QUOTE(A * B "+" C)))	(ADD(TIMES A B)C)
(PREF(QUOTE(A * (B "+" C))))	(TIMES A(ADD B C))
(PREF(QUOTE(A "-" B^C)))	(SUB A(EXPT B C))
(PREF(QUOTE(1/(N !))))	(DIV 1(FAC N))
(PREF(QUOTE("-" Y)))	(SUB 0 Y)
(PREF(QUOTE(X * (COS X/2))))	(TIMES X(COS(DIV X 2)))

Ao invés de outras áreas da matemática, tais como integração ou provas de teoremas, a diferenciação pode geralmente ser caracterizada por um conjunto finito de regras definidas rigorosamente, as quais podem ser facilmente definidas como funções recursivas LISP.

A tabela abaixo exibe um resumo das regras de diferenciação usadas por DIFF. Os termos u e v representam funções de x; e a representa um número real.

$\frac{d}{dx}(a)=0$
$\frac{d}{dx}(x)=1$
$\frac{d}{dx}(u+v)=\frac{du}{dx}+\frac{dv}{dx}$
$\frac{d}{dx}(uv)=u\frac{dv}{dx}+v\frac{du}{dx}$
$\frac{d}{dx}(\sin u)=\frac{du}{dx}(\cos u)$
$\frac{d}{dx}(\cos u)=-\frac{du}{dx}(\sin u)$
$\frac{d}{dx}(u/v)=\frac{(v\frac{du}{dx}-u\frac{dv}{dx})}{v^2}$
$\frac{d}{dx}(u-v)=\frac{du}{dx}-\frac{dv}{dx}$
$\frac{d}{dx}(u^v)=vu^{v-1}\frac{du}{dx}+(\log u)u^v\frac{dv}{dx}$
$\frac{d}{dx}(e^u)=e^u\frac{du}{dx}$
$\frac{d}{dx}(\log u)=\frac{1}{u}\frac{du}{dx}$

A função de diferenciação DIFF recebe dois argumentos: a expressão a ser diferenciada, e a variável em relação à qual a diferenciação ocorrerá. A definição da função DIFF aparece na lista 4 a seguir. Esta consiste de uma sentença COND na qual cada componente da sentença verifica uma particular regra de diferenciação. Note que a recursão é usada extensivamente.

Nota: Para aqueles que não dominam LISP vamos tentar clarear aqui um fato importante; dentro desse programa sobre diferenciação simbólica que estamos apresentando, é na verdade na lista 4 (função DIFF), onde a "grosso modo" são feitas as diferenciações das funções. Sugerimos também ao leitor interessado, tentar fazer à mão - método "chinês" - algumas derivações de funções, usando caso necessário, a tabela com o resumo das regras de diferenciação, e a teoria inicial - sobre as primitivas e as formas do LISP - apresentadas em qualquer das bibliografias sugeridas no final desse capítulo. Isso é realmente interessante, pois permite que o leitor veja por si próprio que LISP, quando escrita com correção, é bem simples de ser entendida e manipulada.

Lista 4

```
(DEF DIFF(E X)
  (COND((ATOM E)(COND((EQ E X)1)(T 0)))
    ((EQ(CAR E)(QUOTE ADD))
      (LIST(QUOTE ADD)
```

```
    (DIFF(CAR(CDR E))X)
    (DIFF(CAR(CDR(CDR E)))X)))
((EQ(CAR E)(QUOTE TIMES)
  (LIST(QUOTE ADD)
    (LIST(QUOTE TIMES)
      (CAR(CDR E))
      (DIFF(CAR(CDR(CDR E)))X))
    (LIST(QUOTE TIMES)
      (CAR(CDR(CDR E)))
      (DIFF(CAR(CDR E))X))))
((EQ(CAR E)(QUOTE SIN))
  (LIST(QUOTE TIMES)
    (LIST(QUOTE COS)
      (CAR(CDR E)))
    (DIFF(CAR(CDR E))X)))
((EQ(CAR E)(QUOTE COS))
  (LIST(QUOTE TIMES)
    (LIST(QUOTE SUB)
      0
      (LIST(QUOTE SIN)
        (CAR(CDR E))))
      (DIFF(CAR(CDR E)X)))
((EQ(CAR E)(QUOTE DIV))
  (LIST(QUOTE DIV)
    (LIST(QUOTE SUB)
      (LIST(QUOTE TIMES)
        (CAR(CDR(CDR E)))
        (DIFF(CAR(CDR E))X))
      (LIST(QUOTE TIMES)
        (CAR(CDR E))
        (DIFF(CAR(CDR(CDR E)))X)))
    (LIST(QUOTE TIMES)
      (CAR(CDR(CDR E)))
      (CAR(CDR(CDR E))))))
((EQ(CAR E)(QUOTE SUB))
  (LIST(QUOTE SUB)
    (DIFF(CAR(CDR E))X)
    (DIFF(CAR(CDR(CDR E)))X)))
((EQ(CAR E)(QUOTE EXPT))
  (LIST(QUOTE ADD)
    (LIST(QUOTE TIMES)
      (DIFF(CAR(CDR E))X)
```

```
        (LIST(QUOTE TIMES)
          (CAR(CDR(CDR E)))
          (LIST(QUOTE EXPT)
            (CAR(CDR E))
            (LIST(QUOTE SUB)
              (CAR(CDR(CDR E)))
              1))))
    (LIST(QUOTE TIMES)
      (DIFF(CAR(CDR(CDR E)))X)
      (LIST(QUOTE TIMES)
        (LIST(QUOTE LOG)
          (CAR(CDR E)))
        (LIST(QUOTE EXPT)
          (CAR(CDR E))
          (CAR(CDR(CDR E)))))))))
((EQ(CAR E)(QUOTE EXP))
  (LIST(QUOTE TIMES)
    E
    (DIFF(CAR(CDR E))X)))
((EQ(CAR E)(QUOTE LOG))
 (LIST(QUOTE TIMES)
   (LIST(QUOTE DIV)
     1
     (CAR(CDR E)))
   (DIFF(CAR(CDR E))X)))))
```

A função DIFF primeiramente testa se a expressão é ou não um átomo. Se for, determina se a expressão é a variável de interesse - nesse caso o valor da função é 1 - pois a derivada de uma variável com respeito a ela mesma é a unidade. Por outro lado, se o argumento para DIFF é algum outro átomo, 0 é retornado. As cláusulas adicionais para o enunciado DIFF COND procura por regras de diferenciação mais complexas que possam ser satisfeitas. As regras são aplicadas recursivamente, o que permite uma representação LISP simples.

A modularidade dessa função LISP é óbvia. É facilmente extendida de de modo a englobar regras adicionais pela inclusão de componentes adicionais ao COND. Considerando DIFF com uma caixa preta, a tabela a seguir lista exemplarmente o input e o output da função DIFF que diferencia seu 1^o argumento em relação ao 2^o.

Input	Output
(DIFF(QUOTE(ADD A A))(QUOTE A))	(ADD 1 1)
(DIFF(QUOTE(TIMES X X))(QUOTE X))	(ADD(TIMES X 1)(TIMES X 1))
(DIFF(QUOTE(LOG X))(QUOTE X))	(TIMES(DIV 1 X)1
(DIFF(QUOTE(COS X))(QUOTE X)	(TIMES(SUB 0(SEN X))1)
(DIFF(QUOTE A)(QUOTE B))	0
(DIFF(QUOTE B)(QUOTE B))	1

Uma vez que DIFF representou seus inputs em uma derivada apropriada, é necessário traduzir o output de notação LISP para a notação infix. Esse processo é conseguido pelo analisador output INFIX cuja definição é mostrada na lista 5 que se segue. INFIX converte seus inputs de notação prefix de Cambridge para a notação matemática infix. INFIX também calcula simplificação algébrica.

Lista 5

```
(DEF INFIX(E)
  (COND((ATOM E)E)
    (T(PROG(FIRST SECOND)
        (SETQ FIRST(INFIX(CAR(CDR E)))
        (COND((CDR(CDR E))
           (SETQ SECOND(INFIX(CAR(CDR(CDR E))))))
        (RETURN
          (COND((EQ(CAR E)(QUOTE ADD))
             (COND((EQ FIRST 0)SECOND)
               ((EQ SECOND 0)FIRST)
               ((TWONUM)(EVAL E NIL))
               ((EQUAL FIRST SECOND)
                 (LIST 2
                   (QUOTE *)
                   FIRST))
               ((MATCH(LIST(QUOTE ?)(QUOTE *)FIRST)SECOND)
                 (INFIX(LIST(QUOTE TIMES)
                           (LIST(QUOTE ADD)
                             (CAR SECOND)
                             1)
                           FIRST)))
               ((MATCH(LIST(QUOTE ?)(QUOTE *)SECOND)FIRST)
                 (INFIX(LIST(QUOTE TIMES)
                           (LIST(QUOTE ADD)
```

```
                    (CAR FIRST)
                    1)
                   SECOND)))
        (T(LIST FIRST
            (QUOTE "+")
            SECOND))))
    ((EQ(CAR E)(QUOTE TIMES))
      (COND((EQ FIRST 1)SECOND)
           ((EQ SECOND 1)FIRST)
           ((EQ FIRST 0)0)
           ((EQ SECOND 0)0)
           ((EQUAL FIRST SECOND)
             (LIST FIRST
               (QUOTE 1)
               2))
           ((TWONUM)(EVAL E NIL))
           ((MATCH(LIST FIRST(QUOTE 1)(QUOTE ?))SECOND)
             (INFIX(LIST(QUOTE EXPT)
                   FIRST
                   (LIST(QUOTE ADD)
                     (CAR(CDR(CDR SECOND)))
                     1))))
           ((MATCH(LIST SECOND(QUOTE 1)(QUOTE ?))FIRST)
             (INFIX(LIST(QUOTE EXPT)
                   SECOND
                   (LIST(QUOTE ADD)
                   (CAR(CDR(CDR FIRST)))
                   1))))
           ((MATCH(LIST(QUOTE ?)(QUOTE *)FIRST)SECOND)
             (INFIX(LIST(QUOTE TIMES)
                   (CAR SECOND)
                   (LIST(QUOTE EXPT)
                     FIRST
                     2))))
           ((MATCH(LIST(QUOTE ?)(QUOTE *)SECOND)FIRST)
             (INFIX(LIST(QUOTE TIMES)
                   (CAR FIRST)
                   (LIST(QUOTE EXPT)
                     SECOND
                     2))))
(T(LIST FIRST
    (QUOTE *)
```

```
         SECOND))))
((EQ(CAR E)(QUOTE DIV))
  (COND((EQ SECOND 1)FIRST)
        ((EQUAL FIRST SECOND)1)
        ((EQ FIRST0)0)
        ((TWONUM)(EVAL E NIL))
        ((MATCH(LIST(QUOTE ?)(QUOTE *)SECOND)FIRST)
          (INFIX(LIST(QUOTE TIMES)
                (LIST(QUOTE SUB)
                  (CAR FIRST)
                  1)
                SECOND)))
        ((MATCH(LIST(QUOTE ?)(QUOTE *)FIRST)SECOND)
          (INFIX(LIST(QUOTE DIV)
                1
                (LIST(QUOTE TIMES)
                  (LIST(QUOTE SUB)
                    (CAR SECOND)
                    1)
                  FIRST))))
        ((MATCH(LIST FIRST(QUOTE 1QUOTE ?))SECOND)
          (INFIX(LIST(QUOTE DIV)
                1
                (LIST(QUOTE EXPT)
                  FIRST
                  (LIST(QUOTE SUB)
                    (CAR(CDR(CDR SECOND)))
                    1)))))
        (T(LIST FIRST
           (QUOTE /)
           SECOND))))
((EQ(CAR E)(QUOTE EXPT))
  (COND((EQ SECOND 1)FIRST)
        ((EQ SECOND 0)1)
        ((TWONUM)(EVAL E NIL))
        (T(LIST FIRST
           (QUOTE 1)
           SECOND))))
((EQ(CAR E)(QUOTE SUB))
  (COND((EQ SECOND 0)FIRST)
        ((EQ FIRST 0)(LIST(QUOTE "-")
                     SECOND))
```

```
      ((TWONUM)(EVAL E NIL))
      ((EQUAL FIRST SECOND)0)
      (T(LIST FIRST
         (QUOTE "-")
         SECOND))))
((EQ(CAR E)(QUOTE LOG))
  (COND((EQ FIRST 1)0)
       (T(LIST(QUOTE LOG)
          FIRST))))
     ((EQ(CAR E)(QUOTE SIN))
       (LIST(QUOTE SIN)
         FIRST))
     ((EQ(CAR E)(QUOTE COS))
       (LIST(QUOTE COS)
         FIRST))
     ((EQ(CAR E)(QUOTE EXP))
       (COND((EQ FIRST 0)1)
          (T(LIST(QUOTE EXP)
             FIRST))))
((EQ(CAR E)(QUOTE FAC))
  (COND((EQ FIRST 1)1)
       ((EQ FIRST 0)1)
       (T(LIST FIRST
           (QUOTE !)))))
(T E)))))))
```

Essa função (lista 5) usa amplamente outra função - MATCH - mostrada a seguir na lista 6 que foi descrita por Willian A. Kornfeld. Match é um modelo de correspondência usado por INFIX para simplificar expressões algébricas por casamento de padrões, como as identidades multiplicativas e aditivas.

Lista 6

```
(DEF MATCH(X Y)
  (COND((EQ X(QUOTE ?))T)
       ((AND(ATOM X)(EQ X Y))T)
       ((OR(ATOM X)(ATOM Y))NIL)
       (T(AND(MATCH(CAR X)(CAR Y))
        (MATCH(CDR X)(CDR Y))))))
```

MATCH recebe dois argumentos e determina se eles seguem ou não o mesmo modelo. Se um ponto de interrogação - ? - é encontrado no primeiro argumento, qualquer átomo do segundo argumento se igualará a ele. Uns poucos exemplos do uso desse "coringa" são apresentados a seguir:

```
(MATCH(QUOTE( A B C))
  (QUOTE(A B C))) = T
(MATCH(QUOTE(A B D))
  (QUOTE(A B C))) = NIL
(MATCH(QUOTE(A B ?))
  (QUOTE(A B C))) = T
```

Uma vez que modelos são reconhecidos por INFIX a correta operação de análise começa. INFIX é também configurado para reconhecer as identidades aditivas e multiplicativas assim como muitas simplificações algébricas.

O centro nervoso de INFIX é uma função PROG que chama a si mesma recursivamente no primeiro, e caso necessário, no segundo argumento da expressão de entrada. O uso das variáveis FIRST e SECOND previne INFIX de recomputá-los a cada encontro.

INFIX usa a função TWONUM (lista 7), cuja definição é apresentada abaixo. TWONUM verifica se as variáveis FIRST e SECOND são números. Se forem, retorna T (verdadeiro), e senão retorna NIL. E daí (EVAL E NIL) é executado, que permite INFIX de fato calcular a expressão. Note que a A-List é NIL, desde que nenhuma variável de ligação for requerida por EVAL.

Lista 7

```
(DEF TWONUM()
  (COND((NUMBER FIRST)(NUMBER SECOND))))
```

Agora que os elementos da diferenciação estão "a postos" é necessário desenvolver um programa de modo que as derivações possam ser calculadas. A função DERIV, mostrada abaixo na lista 8 serve para esse propósito. DERIV conecta PREF, INFIX e DIFF. A função CHR é análoga à função CHR$ em BASIC. A sequência (CHR 28)(CHR 31) limpa a tela.

Lista 8

```
(DEF DERIV()
  (PROG()
```

```
    (CHR 28)
    (CHR 31)
    (PRINT(QUOTE(DERIV "..." UM DIFERENCIADOR SIMBOLICO)))
    (TERPRI)
    (TERPRI)
LOOP(PRINT(QUOTE > ))
  (PRINT(INFIX(DIFF(PREF(READ))(PREF(READ)))))
  (GO LOOP)))
```

A lista 9 mostra exemplos do funcionamento da função DERIV. Um input do usuário é precedido pelo prompt >.

Lista 9

```
>(DERIV)
(DERIV---UM DIFERENCIADOR SIMBOLICO)

>(A/B)B
((-A)/(B↑2))

>(X * X * X)X
(3 *(X↑2))

>(SIN 4 * X)X
((COS(4 * X)) * 4)

>(LOG X)X
(1/X)

>(X ↑ 6)X
(6 *(X↑5))

>(X ↑ N)X
(N *(X↑(N-1)))
```

1.8 Considerações Gerais

Obviamente o programa apresentado pode ser simplificado, levemente alterado ou até totalmente modificado por quantos queiram escrever seu próprio diferenciador simbólico usando

LISP. A sintaxe de LISP é extremamente simples, como dissemos anteriormente, um programa LISP pode usar um outro programa LISP como se fosse um dado, é rápida para cálculos matemáticos, permite ampliar a precisão dos cálculos, com fixação do número de dígitos maior do que o usualmente usado por outras linguagens, possui um mecanismo interno que constantemente libera a memória não mais usada, além de permitir que programas pequenos executem cálculos extensos. Outro detalhe, que não pode deixar de ser mencionado, é que esse ou outro qualquer programa não pode trabalhar sozinho em um sistema de computação algébrica. Qualquer sistema contém um grande conjunto de comandos conectados ao conceito de diferenciação; essas operações são "linkadas" aos comandos de simplificação - uma das partes mais importantes e difícies de serem bem programadas em softwares desse tipo - pois sem boas ferramentas de simplificação existem sérios riscos de termos fórmulas imcompreensíveis muito rapidamente. Aqueles que desejam construir um software de computação algébrica devem ter em mente a abrangência da sua tarefa. Não devem se sentir desestimulados, mas por outro lado não podem esquecer que no mundo inteiro há vários grupos de pesquisadores subsidiados por órgãos de pesquisa - e apoiados até mesmo por governos que aplicam muita verba neste tipo de desenvolvimento - para criar softwares ou aprimorar os já existentes.

1.9 Publicações Pertinentes

A seguir apresentamos uma lista de publicações sobre LISP, tanto a nível de conceituação como de programação propriamente dita, que certamente servirá para o aprofundamento do leitor nessa linguagem. Nunca é demais salientar que LISP foi, e ainda é, a responsável pelas maiores contribuições na construção dos softwares de computação algébrica da nossa época.

1. WINSTON P.H. e HORN B.K.P., *LISP Second Edition*, Addison-Wesley Publishing Company, U.S.A., 1984.
2. McCarthy J., *LISP Programmer's Manual*, Massachussets Institute of Tecnology, U.S.A., 1981.
3. SIKLÓSSY L., *Let's Talk LISP*, Prentice-Hall, U.S.A., 1976.
4. FURTADO A.L., *Paradigmas de Linguagem de Programação*, Editora da Unicamp, 1986.
5. LUCENA C.J.P. *Inteligência Artificial e Engenharia de Software*, Publicações Acadêmico-Científicas PUC-RJ / IBM Brasil, 1987.
6. OAKEY S., *LISP para Micros* Editora Campus, 1986.

Capítulo 2

REDUCE

2.1 Introdução

Nos últimos 25 anos foram feitos progressos substanciais no campo da computação algébrica. Neste período foram concebidos diversos softwares utilizando diferentes linguagens de implementação e que foram desenvolvidos em vários centros científicos e em distintos países.

Entre os maiores sistemas algébricos em serviço atualmente, temos que destacar aquele que ocupa uma posição privilegiada, o REDUCE; até hoje o sistema mais amplamente difundido, utilizado em mais de 1000 universidades e usados por engenheiros, matemáticos e cientistas em máquinas que vão desde o IBM PC até ao Cray X-MP.

Em 1963 John McCarthy, o criador da linguagem LISP, se encontrou com Anthony C. Hearn, que trabalhava com problemas de cálculos elementares em Física Teórica, e sugeriu a utilização de LISP para automatizar cálculos manuais, segundo o próprio Hearn depois de alguns experimentos concluiu que McCarthy estava certo, e desde então começou a trabalhar no ramo da computação algébrica.

A primeira publicação de Anthony Hearn nessa área foi apresentada em agosto de 1966 no artigo "Computation of Algebraic Properties of Elementary Particle Reactions Using a Digital Computer", da ACM.

A primeira versão do REDUCE, desenvolvido na Universidade de Stanford, veio a público em 1967 e continua até os dias atuais com a colaboração da sua comunidade de usuários "avançados" - no sentido de que contribuições, mesmo vindo de centros científicos de outros países, são bem vindas por Hearn - contribuindo com vários módulos e facilidades para o sistema, que já apresenta versões recentes ampliadas tanto em velocidade de processamento como em abrangência em relação a outros softwares de computação algébrica.

REDUCE roda em vários sistemas de computação, existem versões para PC's, Workstations, Minicomputadores, Minimainframes e Mainframes. Vamos apresentar alguns equipamentos para os quais o software está disponível:
VAX / VMS ; VAX / UNIX ; SUN-3 / UNIX ; Apollo / UNIX ; ATARI ST; IBM 43xx series; IBM-PC / MS-DOS ; IBM 390 series ; MacIntosh / MacOp Sys; Cray-1; Cray X-MP; Symbolics 3600; Cyber series; Burroughs series; DEC VAX; Xerox Series, HP9000 serie 200.

2.1.1 Instalação

Vamos apresentar a seguir os procedimentos passo a passo para instalação do REDUCE em dois tipos de microcomputadores, que na realidade atendem um número expressivo de usuários do software, ou seja, os que não utilizam o sistema em computadores de grande porte.

A versão utilizada no texto foi a do REDUCE 3.3 , que no caso de instalação em PC's - AT / 286 era composta de 11 disquetes 5 1/4" - A até K - de instalação no disco rígido, no nosso caso drive C, bastante fácil, bastando seguir os seguintes passos:
Passo 0 : Com o cursor no prompt do MS-DOS do drive C, digitar;

copy a:HDLOAD.BAT c: \ e dar enter

Passo 1 : Coloque o primeiro disquete no drive A, e com cursor voltando para o C:>, digitar;

HDLOAD e dar enter

Passo 2 : O sistema faz alguns comentários, diz que vai criar o diretório REDUCE e para continuar a instalação, dar enter novamente.
Passo 3 : Todos os arquivos do primeiro disquete - disquete A - são copiados, o sistema pede então para o segundo disquete - disquete B - seja colocado.
Passo 4 : O sistema vai pedindo de modo análogo todos os outros disquetes, listando todos os arquivos copiados até o final do último disquete - disquete K - REDUCE está instalado quando aparecer echo off na tela.

Nota : Os próprios distribuidores do software admitem que a versão do REDUCE para PC's 286 não é satisfatória, concordamos com essa observação e recomendamos fortemente que caso um usuário esteja interessado no uso do sistema que o faça "no mínimo" num 386. Para PC's - 386, com ou sem coprocessador numérico, a versão do REDUCE 3.3 se mostrou realmente muito boa, pois permitiu que fossem feitos trabalhos de alto nível e só apresentou limitações, quando dele foram exigidos cálculos realmente extensos.

Vamos apresentar a seguir, passo a passo, os procedimentos para instalação dos discos de distribuição do REDUCE para MS-DOS 386, seus testes e sua manutenção nos computadores 80386 com MS-DOS Versão 3.3 ou posterior. O sistema também roda nos 80387, os que vem com o chip coprocessador matemático.

Nota : Essa versão não rodará em máquinas 8086, 8088, 80186 ou 80286 mesmo com grande quantidade de memória disponível.

REDUCE é baseado em Standard LISP, e essa versão requer a disponibilidade de PSL - Portable Standard LISP - versão 3.4. Os arquivos PSL necessários para rodar o REDUCE são incluídos nos discos do sistema. Isso não é, entretanto um PSL completo e em particular não inclui os fontes do PSL. Um PSL completo caso necessário é disponível separadamente no Konrad-Zuse-Zentrum cujo endereço fornecemos a seguir.

```
Konrad-Zuse-Zentrum fur Informationstechnik
Berlin
-Symbolik-
Heilbronner Strasse 10
D1000 Berlin 31
Germany
Eletronic Mail: melenik @ sc. ZIB-Berlin.dbp.de
Facsimile: (+49) 30 89604 125
```

Essa versão do REDUCE requer aproximadamente 2 megabytes para o programa sozinho, sem levar em conta na computação exigências de espaço de trabalho. O menor tamanho de memória instalada para a execução do REDUCE é 2.5 megabytes, embora para cálculos mais avançados ainda se necessite mais memória.

REDUCE opera em modo protegido no PSL, com acesso uniforme a toda memória disponível (não existe fronteira de 640k durante a execução do REDUCE); isso é possível por causa do DOS ECLIPSE Extender OS 386. A licença para o REDUCE 386 inclui uma licença do uso do OS386, mas somente para esse produto (a menos que uma pessoa tenha ela própria a licença do ECLIPSE completo) e a essa pessoa, não é permitido além disso distribuir esse material.

Descrição dos Discos Distribuidos

O material é distribuido como um conjunto de discos de 3 $\frac{1}{2}$" ou de 5 $\frac{1}{4}$" com 1.44 ou 1.2 megabytes respectivamente; no caso da nossa instalação, 3 discos do primeiro tipo. Todos os dados são comprimidos para o programa de domínio público ZOO.ZOO.EXE residentes no primeiro disco. Os arquivos comprimidos gerados por ZOO são:

EXE.ZOO material básico executável e de instalação

RINI.ZOO binários básicos do REDUCE

RED.ZOO módulos de carregamento do REDUCE

PSL.ZOO compilador PSL e módulos de suporte

DOC.ZOO Documentação

SRC.ZOO Código fonte do REDUCE

XMPL.ZOO Programas exemplo do REDUCE, testes e lições iterativas;
Por exemplo, para extrair um arquivo "abcd" do arquivo ZOO "A:XYZ.ZOO", entrar com;
ZOO - EA:XYZ.ZOO' abcd

Para extrair todos os arquivos, entrar com:
ZOO - EA:XYZ.ZOO*

Instalando o REDUCE

Para instalar o sistema REDUCE no disco rígido seguir os seguintes passos:

Passo 0 : Fazer cópias dos discos de instalação e usar essas cópias para a instalação. O processo de cópia assegurará que seus discos não serão avariados durante o carregamento.

Passo1 : Criar um diretório para o sistema de arquivo REDUCE. Isso é identificado por "$REDUCE" daqui por diante. É assumido que se tem acesso a esse diretório e que o nome de variável REDUCE no ambiente de pontos desse diretório. Esse diretório residirá no disco rígido, o que é consumado pela entrada dos seguintes comandos.

```
CD

MKDIR REDUCE

SET REDUCE=REDUCE

CD REDUCE
```

Se o investigador desejar colocar o REDUCE em algum outro diretório modificar os comandos acima convenientemente.
O SET deverá ser incluído no seu arquivo AUTOEXEC.BAT, por exemplo, se o diretório for mudado para RED33 - ao invés do REDUCE - o arquivo AUTOEXEC.BAT deverá ter a linha:

```
SET REDUCE=RED33
```

Passo 2 : Inserir o disco 1 no drive - A ou B - e descarregar os binários entrando com os seguintes comandos. Estamos assumindo que seu drive é o drive A, se não for o caso fazer as alterações necessárias.

```
COPY A:ZOO.EXE

ZOO -E A:EXE.ZOO *
```

Nota : Para completar esse comando aguardar entre 1 e 2 minutos.

Passo 3: Converter o carregador binário do PSL para seu BIOS local entrando com:

TUNE -k PSLL.EXE

Aqui a customização do Eclipse DOS386 toma lugar e em alguns casos requer vários ciclos de reboots. Se a máquina parar nessa fase, não recomeçar o procedimento completo de instalação; reboot a máquina e rodar esse passo novamente.

Passo 4 : Caso o passo anterior tenha sido bem sucedido, deletar TUNE.EXE:

DEL TUNE.EXE

Passo 5 : Agora descarregar os binários básicos do REDUCE. Novamente assumimos que estamos no drive A.

ZOO -E A:RINI.ZOO *

Passo 6 : Se sua máquina estiver equipada com o coprocessador numérico 80387, editar o arquivo MKREDUCE.SL e remover o sinal de comentário (o sinal %) que estava na frente do:

(load arith387)

Passo 7: Então rode o Script:

MKREDUCE

Isso gera o arquivo imagem do REDUCE, REDUCE.IMG que pode mais tarde ser carregado e executado pelo carregador do PSL, o PSLL.EXE.

Passo 8 : Agora todos os binários intermediários podem ser deletados:

DEL *.b

DEL BPSL.IMG

com a finalidade de salvar espaço.

Passo 9 : Para completar o conjunto de módulos, descarregar os arquivos do RED.ZOO, os quais são requeridos pelo REDUCE quando está rodando.

ZOO -E A:RED.ZOO

Nota : Nesse momento são copiados os famosos pacotes, Gentran, Groebner, Excalc etc.

Passo 10 : Se desejar usar o compilador PSL se equipar agora do disco 2 e entrar com:

ZOO -E A:PSL.ZOO

Passo 11 : Para compatibilidade com instalações do REDUCE em UNIX, recomendamos que sejam criados os subdiretórios SRC, XMPL e DOC sob o $ REDUCE e que sejam descarregados os arquivos SRC.ZOO, XMPL.ZOO e DOC.ZOO dentro desses diretórios, esses arquivos não

são necessários pelo REDUCE enquanto rodando. Verificar se o disco 2 está pronto e então fazer:

MKDIR XMPL

CD XMPL

..\ ZOO -E A:XMPL.ZOO

CD..

MKDIR DOC

CD DOC

..\ ZOO -E A:DOC.ZOO

CD..

Ir para o Disco 3 e ler no código fonte do REDUCE:

MKDIR SRC

CD SRC

..\ ZOO -E A:SRC.ZOO

CD..

Passo 12 : O script REDUCE.BAT é usado para executar o REDUCE. Esse script contém uma chamada para o PSLL com dois parâmetros:

PSLL $< img >< size >$

onde:

$< img >$ é o nome do arquivo imagem a ser carregado; esse nome deve ser inteiramente qualificado. Note que PSLL não é capaz de expandir o nome do arquivo para variáveis do ambiente.

$< size >$ é o tamanho da memória a ser usada como um número decimal de bytes. Se $< size >$ for omitido, são assumidos 2.5 Megabytes. Se o tamanho for muito alto, a quantidade máxima de memória disponível é tomada.

Editar REDUCE.BAT de modo que os atributos sejam adequados. Por exemplo, uma instalação no C:\ REDUCE\ usando toda a memória disponível requererá uma linha em REDUCE.BAT:

C:\ REDUCE\ PSLL C:\ REDUCE\ REDUCE.IMG 9999999

Imprimindo Documentos

Os documentos distribuídos estão no arquivo ZOO, DOC.ZOO. Eles podem ser impressos usando as facilidades padrão do MS-DOS. Os documentos são paginados e formatados com os caracteres de controle padrão ASC II. Um máximo de 60 linhas de impressão por página é assumida. A margem esquerda tem que ser fornecida pelo usuário.

Testando o REDUCE

Para testar a instalação do REDUCE entrar com:

REDUCE

in " ($ reduce \ xmpl \ reduce.tst \ ) ";

Isso requer cerca de um minuto num sistema MS-DOS de 16 Megahertz e 4 Megabytes e 80386SX com disco rígido.

Outros programas para testar o REDUCE podem ser encontrados no arquivo comprimido

XMPL.ZOO.

Rodando programas no REDUCE

Para rodar o REDUCE entrar com:

REDUCE

REDUCE responde com uma linha, mostrada em detalhes posteriormente, e aparece então o prompt para a primeira linha de input:

REDUCE 3.3, 15 - Jan - 88 ..

1:

Nota : Instruções para o uso da implementação do REDUCE para MS-DOS 386 são disponíveis no arquivo comprimido DOC.ZOO e no arquivo OPER. É aconselhavel editar esses arquivos para dirimir dúvidas sobre sua implementação específica. Instruções independentes para uso do REDUCE são encontradas no REDUCE User's Manual.

2.1.2 Inicialização / Entrada / Saída

A inicialização do REDUCE para PC's - AT / 286 é feita também de modo simples, embora mais demorada, bastando seguir a orientação abaixo:

Passo 0 : Com o prompt do MS-DOS no drive C, digitar;

cd \ REDUCE e [Enter], para mudar para o diretório REDUCE.

Passo 1 : Com o prompt do MS-DOS no REDUCE, digitar;

copy treduce.sl uoinit.sl/V e [Enter], para iniciar uma configuração teste do REDUCE.

Passo 2 : Com o prompt do MS-DOS no REDUCE, digitar;

uolisp i e [Enter], para começar o processo de inicialização, onde é lido o arquivo uoinit.sl e criada uma seção teste de configuração freeze chamada treduce.frz.

Passo 3 : Ainda com o prompt do MS-DOS no REDUCE, digitar;

uolisp treduce.frz, para começar a configuração do teste do REDUCE, vai aparecer na tela:

REDUCE 3.3 date...

1:

Passo 4 : Nesse 1: digitar;

IN" $ RUTILS \ REDUCE.TST" $; e [Enter], para o REDUCE ler o arquivo teste. Esse arquivo para ser lido demora de 5 a 10 minutos, dependendo do equipamento. A instrução $ é dada ao REDUCE para não imprimir o que leu no arquivo, quando termina essa leitura o REDUCE emite:

2:

Nota : A partir desse ponto, REDUCE está pronto para trabalhar. Todas as dúvidas e esclarecimentos para operação dos cálculos desejados estão apresentadas de modo didático no manual do usuário da versão 3.1 em diante. REDUCE muda, a cada cálculo pedido pelo usuário, esse número entre colchetes, por exemplo; [3], [4], etc.

Passo 5 : A saída do REDUCE é dada quando é digitado BYE com enter em seguida, no número que estiver entre colchetes após o término dos cálculos, o que faz com que voltemos para o diretório REDUCE.

Comentário : Para voltarmos para a raiz, digitar cd \ e [Enter].

A inicialização do REDUCE para PC's 386 também é feita facilmente, bastando para tal seguir a orientação dada na seção 2.4 (Apresentando o REDUCE)

2.2 O Porquê da Utilidade do REDUCE

- REDUCE é um sistema algébrico de propósito geral, interativo, para cálculos matemáticos com aplicações em engenharia, física, matemática, etc. A versão utilizada nesse estudo foi a do REDUCE 3.3 instalada em um PC-386, o que representa uma grande vantagem pois o número de usuários desse tipo de equipamento vem crescendo a cada dia.

- REDUCE é compatível com um grande número de processadores e sistemas operacionais diferentes, dispondo em todas as implementações dos mesmos recursos, sendo apenas limitado pela velocidade e memória do equipamento. O código é portátil para um grande número de máquinas; um programa que rode num PC também rodará na maioria dos Mainframes científicos mais conhecidos.

- O investigador pode se concentrar no conteúdo intelectual de um problema deixando detalhes computacionais para o computador, e pode testar facilmente e sem trabalho conjecturas e resultados matemáticos dos quais duvide. REDUCE pode ser usado para checar esses resultados utilizando técnicas algébricas, numéricas ou combinação delas.

- Não é necessário que o usuário conheça LISP para trabalhar no REDUCE, pois este oferece uma linguagem externa muito próxima a ALGOL para a definição de dados e programas. Um interpretador interno converte essa linguagem de alto nível para os procedimentos internos em LISP e RLISP. O investigador pode resolver problemas razoavelmente complexos, simplesmente escrevendo expressões a serem manipuladas, com poucas horas de uso.

- RLISP constitui-se num ambiente apropriado não só para a implementação de algoritmos algébricos, mas também para outras aplicações, tais como jogos, demonstrações de teoremas, tradução de linguagem natural e inteligência artificial em geral. Esse ambiente é também chamado de modo simbólico do REDUCE.

- REDUCE é distribuído com os arquivos fontes. Dessa forma o investigador interessado pode desenvolver suas próprias extensões do sistema; os módulos de fatoração, integração, solução de equações, cálculos com números de precisão arbitrária, solução de equações diferenciais (a partir da versão 3.4) e outros foram desenvolvidos dessa forma.

- REDUCE dispõe de várias chaves, programáveis pelo usuário para controlar o processo de simplificação (por exemplo permitindo ou inibindo a expansão de expressões ou o cancelamento de divisores comuns), fornecendo uma variedade de formas de apresentação, agrupando ou separando partes da expressão etc., e provê uma variedade de formatos de saída; particularmente o GENTRAN, fornece a saída de expressões nos formatos FORTRAN, Código C e outros, para uso direto em cálculos com ponto flutuante.

- A linguagem provê estruturas de blocos, estruturas de controle tais como FOR e WHILE, definições de procedimentos e uma variedade de tipos algébricos e simbólicos. Baseado em LISP, REDUCE permite ainda pleno uso de definições recursivas tanto para procedimentos como para relações. A capacidade simbólica da linguagem é suficientemente completa para permitir o programa inteiro ser escrito na sua linguagem fonte. O programa usa frequentemente Standard LISP como uma linguagem intermediária.

- O código é desenhado para ser modular, é possível acrescentar novas capacidades com pouco conhecimento do sistema básico. Essas novas capacidades podem também ser compiladas e carregadas no código baseado da máquina.

2.3 Capacidades e Usos do REDUCE

2.3.1 Capacidades do REDUCE

Seria difícil listar todas as capacidades do REDUCE, o próprio manual do sistema tem centenas de páginas e ainda assim não pode ser considerado um documento completo. Entretanto, algumas das mais importantes capacidades da versão utilizada nesse estudo incluem facilidades para prover ferramentas para:

Aritmética inteira exata e real de precisão arbitrária.
Expansão e ordenação de polinômios e de expressões algébricas.
Substituições em uma grande variedade de formas algébricas.
Simplificações automáticas e sob controle do usuário.
Operações com matrizes simbólicas.
Aceitação de novas funções e extensões da sintaxe.
Integração e diferenciação analítica.
Fatoração de polinômios.
Resolução de sistemas de equações lineares e não lineares com coeficientes algébricos.
Cálculo com formas exteriores.
Produção de programas FORTRAN a partir de expressões REDUCE.
Saídas de expressões numa variedade de formatos.

Geração de programas numéricos otimizados a partir de entradas simbólicas.
Linguagens nos modos Algébrico, Simbólico e LISP.

2.3.2 Usos do REDUCE

Da divulgação do uso do REDUCE em artigos, conferências, revistas e publicações, apresentamos a seguir algumas das áreas do conhecimento onde ele foi empregado. Muitos pesquisadores não revelam na realidade como e de que forma cálculos foram executados, mesmo assim aqui estão alguns usos e campos de aplicação do sistema, fornecidos através da literatura aberta, que dão uma idéia da contribuição, poder e abrangência que o software forneceu até o momento, como instrumento auxiliar a várias áreas e linhas de pesquisa.

Análise de Circuitos Elétricos
Análise Numérica
Análise das Deformações
Astrofísica
Desenho Auxiliar de Computadores
Desenho de Antenas
Dinâmica dos Fluidos
Estatística
Eletrodinâmica Quântica
Física Teórica
Física das Partículas
Física do Estado Sólido
Geometria Algébrica
Engenharia Eletrônica
Física de Plasmas
Mecânica Celeste
Mecânica Estrutural
Mecânica dos Fluidos
Mecânica Quântica
Método dos Elementos Finitos
Processamento de Imagens
Projetos de Cascos de Navios
Ótica
Relatividade Geral
Teoria das E.D.O.'s
Termodinâmica
Química
Teoria da Otimização

2.4 Apresentando o REDUCE

2.4.1 Preliminares

O propósito deste parágrafo é apresentar uma pequena mostra sobre a utilização do REDUCE e também tem a intenção de ser intelegível para o leitor que não está familiarizado com o sistema. No final do capítulo listamos outras fontes para aprofundamento, informações adicionais sobre o software e alguns exercícios elementares. Sugerimos ler essa apresentação em frente a um equipamento que rode o REDUCE, para que o investigador possa executar nossas sugestões como passo inicial.

Para entrar no REDUCE na maioria das máquinas digitar:

cd REDUCE e [Enter]

REDUCE e [Enter] novamente.

Aparecerá então na tela uma mensagem como abaixo, ou outras similares em versões mais recentes, e o prompt aguardando o primeiro pedido de cálculo.

```
C:\REDUCE>psll reduce.img
Copyright (c) 1989 Eclipse Computer Solutions, Inc.
Loading: reduce. img
System memory size is :1900000
REDUCE 3.3, 15-JAN-88 ...
1:
```

Vamos apresentar uma capacidade trivial do sistema, usando a sintaxe para acionar o MMC de denominadores de expressões fracionárias quando isto for desejado pelo usuário:

1: OFF MCD; A1:= 1/x + 1/(x-2) - 2/(2x-3); e [Enter] produz

A1:= $-(2*(2*X-3)^{-1}-(X-1)^{-1}-X^{-1})$

3: ON MCD; A2:=A1; e [Enter] produz

A2:= $\frac{2*(X^2-3*X+3)}{X*(2*X^2-7*X+6)}$

5:

Nota: O prompt 5: aguarda o próximo pedido de entrada de cálculo.

2.4.2 Símbolos e Chaves Especiais

- Para informarmos ao REDUCE que terminamos o pedido de algum cálculo não esquecer o (;). Observe que [Enter] sozinha não sinaliza ao software o que deve ser calculado.
- Para sairmos do REDUCE voltando a seguir a entrar no sistema digitar Bye e [Enter] e para sairmos do sistema encerrando uma seção digitar QUIT e [Enter].
- As quantidades padrão e (base natural de log), i (raiz quadrada de -1) e π 3.14159, são respectivamente referidos por E, I e PI.
- O comando que terminar com o sinal de dolar $; e [Enter] será obedecido mas o resultado não será impresso na tela.

- ws (workspace) é um nome reservado, geralmente é usado para nos referirmos ou pedirmos a execução de algum cálculo na proposição anterior (última proposição "ordinária" executada).
- Comentários explicativos começam com o sinal de porcentagem % que deve ser repetido a cada linha de comentário do usuário.
- Para abortar a execução de um cálculo do programa usar as teclas Ctrl e C.
- Todos os comandos dados durante uma seção do REDUCE podem ser salvos. Suponhamos que um comando numerado de 46 seja uma expressão extensa envolvendo a variável P. Se quisermos alterar o valor de P e recalcular a expressão, entrar com o comando INPUT 46; ou INPUT (46); e na expressão seguinte usar o ws para trabalhar com aquela expressão.

2.5 Entrada das Expressões

A entrada de expressões no REDUCE é feita de maneira simples, de modo a permitir a manipulação de expressões, o que é um dos principais propósitos do sistema.

As operações aritméticas comuns no REDUCE são:

+ Adição

- Subtração

* Multiplicação

/ Divisão

* ou^Exponenciação

REDUCE sabe que algumas funções matemáticas podem tomar expressões escalares arbitrárias como seu único argumento.

SQRT	ACOS
EXP	ATAN
LOG	SINH
SIN	COSH
COS	TANH
TAN	ASINH
COT	ACOSH
ASIN	ATANH

REDUCE conhece as mais elementares identidades e propriedades dessas funções (com excessão em ON NUMVAL), por exemplo:

exp(1) = e	log(e**x) = x
e**x = exp(x)	log(e) = 1
e**(i* pi) = -1	log(1) = 0
e**(i* pi) = -1	sin(0) = 0
sin(pi) = 0	x **(1/2) = sqrt(x)
sin(pi/2) = 1	sin(-x) = - sin(x)
e**(i* pi/2) = i	cos(-x) = cos(x)

Nota: Os vários tipos de expressões possíveis no REDUCE são expressões escalares, expressões equacionais, expressões inteiras e expressões booleanas; maiores detalhes sobre estes diferentes tipos de expressões são encontradas nos manuais do sistema, e no livro REDUCE Software for Algebraic Computation de Gerhard Rayna.

2.6 Sintaxe de Algumas Capacidades do REDUCE

2.6.1 Simplificação

Mostraremos a seguir, de maneira breve, algumas habilidades do REDUCE para operar simplificação de expressões. A simplificação de expressões é certamente um dos maiores problemas da computação algébrica, onde pesquisas na melhoria e em construções de rotinas para esse fim, se desenvolvem em larga escala em todos os sistemas. Algumas operações (por exemplo, redução de termos comuns) são sempre feitos pelo software, outras são controladas por várias chaves que são "ligadas" e "desligadas" através das proposições:

- ON < Nome da Chave > ; e [Enter] "liga" essa chave.
- OFF < Nome da Chave > ; e [Enter] "desliga" essa chave.

Existem várias dessas chaves no REDUCE; as que controlam a simplificação de expressões são:

EXP ⟶ faz com que expressões sejam expandidas durante o cálculo.

GCD ⟶ faz com que o sistema cancele o maior divisor comum durante o cálculo de expressões racionais.

LCM $\longrightarrow$ faz com que o sistema compute mínimos múltiplos comuns.

MCD $\longrightarrow$ faz com que o sistema coloque uma soma de expressões racionais sobre um denominador comum.

FLOAT $\longrightarrow$ faz com que o sistema faça cálculos em ponto flutuante de acordo com a precisão construída da máquina.

BIGFLOAT $\longrightarrow$ faz com que o sistema faça cálculos em ponto flutuante em "precisão arbitrária" com o número de dígitos decimais a ser usado especificado pelo investigador.

Nota: Vamos exemplificar apenas o uso da chave BIGFLOAT; recomendamos a consulta ao manual do sistema para maiores esclarecimentos sobre as chaves de simplificação.

O número de dígitos decimais a ser usado é setado pela proposição PRECISION, como mostrado abaixo.

1: on bigfloat; e [Enter]

2: precision 48; e [Enter] produz

48

3: 5/31 + 4/173; e [Enter] produz

0.184411709863882155509975759835912735409285847473

4: off bigfloat; e [Enter]

2.6.2 Fatoração

REDUCE pode fatorar expressões polinomiais com coeficientes inteiros em uma ou mais variáveis. A fatoração do sistema sempre observa o domínio que o investigador deseja para seus trabalhos que podem ser modulares, complexas e de ponto flutuante. Existem duas sintaxes possíveis:

- FACTORIZE (< expressão >); que retorna uma lista de todos os fatores.
- FACTORIZE (< expressão > , < primo >) ; que usa o dado número primo como uma "sugestão" para o processo de redução modular.

Nota: A opção de controle on ifactor; fatoram também inteiros como produto de primos. Também é possível se fatorar inteiros módulo p onde p é primo e se obter uma descrição dos passos seguidos pelo algoritmo. Ver o manual para maiores detalhes.

Exemplos:

1: factorize($145 * x\hat{}4 - 145 * x\hat{}3 - 580 * x\hat{}2 + 290 * x + 580$); e [Enter] produz

{ $145, X-2, X+1, X^2-2$ }

2: on ifactor; factorize ($51*(a\hat{}3*b\hat{}3-a\hat{}3*c+a\hat{}2*b\hat{}4-a\hat{}2*b*c-a*b\hat{}5-a*b\hat{}3*c\hat{}2+a*b\hat{}2*c+a*c\hat{}3-b\hat{}6+b\hat{}4*c\hat{}2+b\hat{}3*c-b*c\hat{}3)$); e [Enter] produz

{ $3, 17, B^3-C, A+B-C, A+B+C, A-B$ }

2.6.3 Substituição

A função SUB fornece o resultado algébrico da substituição de qualquer ocorrência de uma variável VAR na expressão EXPRN 1 pela expressão EXPRN, que pode ser outra variável ou uma expressão.

Sintaxe:

SUB([VAR:Kernel=EXPRN:algebraic,]EXPRN1:algebraic):algebraic

Exemplo:

1: SUB(X = A + B ,$X**2+Y**2$); e [Enter] produz

$A^2+2*A*B+B^2+Y^2$

SUB pode ser usada para fazer qualquer número de substituições simultaneamente.Os pares "variável = trocada por" são escritos primeiramente e a expressão nas quais as trocas devem ser feitas posteriormente; todos separados por vírgulas.

Exemplo:

2: MM :=$X**3+Y**2$; e [Enter] produz

MM := X^3+Y^2

3: F := SUB (X = A+B , Y = Y+1 , MM); e [Enter] produz

F := $A^3+3*A^2*B+3*A*B^2+B^3+Y^2+2*Y+1$

Nota : SUB primeiramente simplifica a expressão; então troca nela as variáveis que ocorreram na lista de substituições e finalmente simplifica novamente o resultado. As substituições são feitas somente uma vez : dois loops infinitos não ocorrem. Nenhuma atribuição como X := A+B na realidade, toma lugar definitivo, daí X permanece livre.

Exemplo:

4: EE := $2*X**2+3*Y**2$; e [Enter] produz

EE := $2*X^2+3*Y^2$

5: EE := SUB (X = Y , Y = X , EE); e [Enter] produz

EE := $3*X^2+2*Y^2$

Aqui SUB foi usado para permutar X e Y numa expressão.

2.6.4 Equações e Sistemas de Equações

O operador SOLVE tenta resolver uma simples equação em um termo desconhecido ou um sistema de equações lineares em várias variáveis. São retornadas todas as soluções encontradas através de uma lista a qual é o valor do operador SOLVE.

Sintaxe:

SOLVE(equação, variável);

SOLVE(lista de equações, lista de variáveis);

Exemplos:

1: SOLVE($X**5 - 13*X**4 + X**3 + 293*X**2 - 2*X - 280$, X); e [Enter] produz

{ X = 7, X = -1, X = -4, X = 10, X = 1 }

2: SOLVE({ X+Y+Z=2, X-Y-Z=-3, 2X+Y+2Z=1, 3X+2Y+3Z=3 }, { X,Y,Z }); e [Enter] produz

{ { X = - $\frac{1}{2}$, Y = 3, Z = -$\frac{1}{2}$ } }

Nota: Se o segundo argumento for omitido, as equações são resolvidas em todos os termos desconhecidos que aparecerem.

2.6.5 Matrizes

Uma característica muito poderosa do sistema REDUCE é a facilidade de cálculos com matrizes. Para estender nossa sintaxe a essa classe de cálculos, precisaremos acrescentar outro operador de prefixo para construção de matrizes, o MAT, que é usado para representar matrizes n x m. MAT tem n argumentos interpretados como linhas da matriz, cada um sendo uma lista com m expressões representando elementos naquela linha.
Uma matriz como

$$\begin{pmatrix} 0 & 1 & 0 \\ -1 & 5 & 6 \\ P & Q & R \\ A & B & C \end{pmatrix}$$

pode ser escrita na forma

```
MAT ( (0,1,0) ; (-1,5,6) ; (P,Q,R) ; (A,B,C) );
```

ou ainda, numa maneira mais próxima à visão natural de uma matriz, pois o REDUCE aceita a divisão de uma expressão em linhas.

```
MAT(
( 0,1,0),
(-1,5,6),
( P,Q,R),
( A,B,C));
```

Nota : Matrizes também podem ser definidas através de seus elementos por atribuições individuais como A(1,1) :=...$
Não recomendamos usar o MAT para entrar com matrizes muito grandes ou complicadas pois um simples erro anularia um grande esforço de digitação.

Uma matriz no REDUCE é superficialmente como um array 2-dimensional. O símbolo A é declarado para representar uma matriz de 3 linhas e 4 colunas pela declaração

MATRIX A(3,4)$

Isso permite o uso de 12 símbolos, A(1,1) até A(3,4), o mesmo modo que se usa posições de array. O comando CLEAR A faz com que a declaração de matriz (e as entradas na matriz) sejam abandonadas.
Várias matrizes podem ser declaradas do seguinte modo:

MATRIX A(3,4) , B(3,3) , C(5,5)$

que declara A como sendo uma matriz 3 x 4, B como uma matriz 3 x 3 e C como sendo uma matriz 5 x 5.

Se quisermos imprimir uma matriz grande com muitos zeros, a impressão dos zeros pode ser suprimida setando-se o modo ON NERO$:

```
MATRIX B (3,3) $
FOR I : 1 : 3  DO  B (I,I) := 1$
ON NERO$
B ;
   MAT (1,1) := 1
   MAT (2,2) := 1
   MAT (3,3) := 1
OFF NERO$
```

Nota: O propósito do comando final é o de restituir ao REDUCE a seu modo de operação normal, no qual todas as entradas das matrizes, incluindo os zeros, são impressos.

Expressões matriciais

Os símbolos + , − , * , / , ** quando aplicados às matrizes, representam as operações comuns da álgebra das matrizes.
A + B e A - B são definidos somente se A e B são matrizes de mesma ordem.

A * B é definido somente se A e B são compatíveis para a multiplicação.

X * A é também definido se X é uma variável ordinária (não uma matriz) e A é uma matriz. Também pode ser escrito com a matriz primeiro; A * X significa que todo elemento de A é multiplicado por X.

A ** N é definido se A é uma matriz quadrada e N é um inteiro (ou uma variável ou expressão cujo valor é um inteiro). Por exemplo, A **4 significa A * A * A * A. Se N for negativo, A ** N representa uma potência da inversa de A. Por exemplo, A ** (-3) significa o cubo de A ** (-1), ou seja, o cubo da inversa de A.

A / B é definida se B é uma matriz quadrada e A tem o mesmo número de colunas que o lado de B; é interpretada como A*(B**(-1)). Observe a ordem dessa multiplicação.

X / B é também definida se X é uma variável ordinária (não uma matriz) e B é uma matriz quadrada. É interpretado como X * (B ** (-1)). Em particular, 1 / B é a inversa de B.

Como um exemplo final; se A, B e C são matrizes 2 por 2, então:
B ** 2 * A - 3 * B ** (-2) * (A + C) + MAT((1,X),(Y,Z))/2
é uma expressão matricial expressiva (desde que B tenha inversa).

Outras operações com matrizes

As funções de matrizes DET, TP e TRACE são definidas da seguinte maneira:

O operador DET é usado para representar o determinante de uma expressão de uma matriz quadrada.

Sintaxe:

DET(expressão matricial):número

Nota : DET(A) pode também ser escrito DET A.

Exemplos:

DET ($C * *2$) é uma expressão escalar cujo valor é o determinante do quadrado da matriz C

DET MAT((A,-B,2 * C),(0,sqrt 2,1),(0,-0.1,1)); é uma expressão escalar cujo valor é o determinante da matriz

$$\begin{pmatrix} A & -B & 2C \\ 0 & \sqrt{2} & 1 \\ 0 & -0.1 & 1 \end{pmatrix}$$

O operador TP pega uma matriz simples como argumento e retorna sua transposta.

Sintaxe:

TP(matriz-expressão):matriz

Exemplo:

TP MAT((1,2,3,4,5)) define a matriz coluna como a transposta define uma matriz linha.

O operador TRACE é usado para representar o traço de uma matriz quadrada.

Sintaxe:

TRACE(matriz-expressão):número

Exemplos triviais com matrizes

Vamos pedir ao REDUCE para demonstrar algumas de suas capacidades com a matriz dada acima.

```
1: M:=MAT((A,-B,2C),(0,SQRT(2),1),(0,-0.1,1)); e [Enter] produz
*** 0.09999 represented by 1/10
M(1,1) := A
M(1,2) := -B
M(1,3) := 2*C
M(2,1) := 0
M(2,2) := SQRT(2)
M(2,3) := 1
M(3,1) := 0
M(3,2) := - 1
            10
M(3,3) := 1

2: DET(M); e [Enter] produz
A*(10*SQRT(2) + 1)
        10

3: TRACE(M); e [Enter] produz
SQRT(2) + A + 1
```

4: 1/M; e [Enter] produz

$$MAT(1,1) := \frac{1}{A}$$

$$MAT(1,2) = \frac{2*(5*B - C)}{A*(10*SQRT(2) + 1)}$$

$$MAT(1,3) := -\frac{10*(2*SQRT(2)*C + B)}{A*(10*SQRT(2) + 1)}$$

$$MAT(2,1) := 0$$

$$MAT(2,2) := \frac{10}{10*SQRT(2) + 1}$$

$$MAT(2,3) := -\frac{10}{10*SQRT(2) + 1}$$

$$MAT(3,1) := 0$$

$$MAT(3,2) := \frac{-1}{10*SQRT(2) + 1}$$

$$MAT(3,3) := \frac{10*SQRT(2)}{10*SQRT(2) + 1}$$

5: M*M; e [Enter] produz

$$MAT(1,1) := A^2$$

$$MAT(1,2) := -\frac{5*SQRT(2)*B + 5*A*B + C}{5}$$

$$MAT(1,3) := 2*A*C - B + 2*C$$

```
MAT(2,1) := 0
```

$$\texttt{MAT(2,2)} := \frac{19}{10}$$

```
MAT(3,1) := 0
```

$$\texttt{MAT(3,2)} := -\frac{\texttt{SQRT(2)} + 1}{10}$$

$$\texttt{MAT(3,3)} := \frac{9}{10}$$

Podemos também ilustrar o uso de matrizes com outro exemplo para achar a solução de um sistema de equações lineares; não estamos afirmando que a inversão de matrizes provê um modo eficiente de resolver sistemas de equações lineares.

Suponhamos um sistema de n equações lineares com n termos desconhecidos.

$1 * X(1) + 5 * X(2) + 3 * X(3) = 19,$

$2 * X(1) + 2 * X(2) + 2 * X(3) = 10,$

$7 * X(1) - 9 * X(2) + 13 * X(3) = -35.$

Convertendo para um problema matricial, temos:os coeficientes das n equações são armazenados em uma matriz A n por n ; as "constantes" formam uma matriz C n por 1; então os termos desconhecidos podem ser calculados como entradas da matriz n por 1.
X:=(1/A) * C pois A * X = C.
Aqui 1/A é claramente a notação REDUCE da inversa de A.

```
MATRIX  A,C,X;
 A:= MAT (1,5,3) ,
         (2,2,2) ,
         (7.-9,13) ) $
 C:= TP MAT ((19,10,-35)) $
e ao darmos
 X:= (1/A) * C ; e [Enter] temos
 X(1,1) := 2       % valor de X1
 X(2,1) := 4       % valor de X2
 X(3,1) := -1      % valor de X3.
```

Para achar a solução de um sistema de equações

$(P+Q)*X(1)+(U-V)*X(2)=Y1$

$(P-Q)*X(1)+(U-V)*X(2)=Y2$

podemos simplesmente escrever

```
1: MATRIX X$ e [Enter]
2: X := 1/MAT ((P+Q,U-V),(P-Q,U-V))*TP MAT ((Y1 , Y2)); e [Enter] produz

  X(1,1) :=  U*Y1 - U*Y2 - V*Y1 + V*Y2
                    2*Q*(U - V)

  X(2,1) := - P*Y1 - P*Y2 -Q*Y1 - Q*Y2
                    2*Q*(U - V)
```

Entre os softwares utilizados pelo autor na resolução de grandes sistemas lineares, o REDUCE teve o melhor aproveitamento. O desenvolvimento de várias outras habilidades não mostradas explicitamente no texto estão disponíveis, e são tratadas com mais abrangência no livro REDUCE Software for Algebraic Computation / Gerhard Rayna - Springer - Verlag. 1987.

2.6.6 Diferenciação

O operador DF é usado para representar diferenciações parciais com respeito a uma ou mais variáveis.

Sintaxe:

DF (EXPR: algébrica, [var: Kernel , NUM: integer]): algebraic

O primeiro argumento é a expressão algébrica a ser diferenciada. Os argumentos restantes especificam as variáveis de diferenciação e a ordem de derivação. O número NUM pode ser omitido caso seja 1.

DF (Y,X) = dY/dX

DF (Y,X,2) = $\frac{d^2Y}{dX^2}$

DF (Y,X,2,U,W,2) que se escreve em cálculo $\frac{\partial^5 Y}{\partial X^2 \partial U \partial W^2}$

A função DF faz diferenciação de expressões e conhece a maioria das regras de derivação de funções contínuas, incluindo muitas formas trigonométricas.

EE:=4 * X **3; e [Enter] produz

EE := $4*X^3$

S:=DF(EE,X); e [Enter] produz

S := $12 * X^2$

O símbolo DF(EE,X) é lido como "derivada de EE com respeito a X". DF pode ser usado para se achar derivadas de graus mais altos em um simples passo, pela introdução do valor inteiro como terceiro argumento; para se achar a derivada segunda de EE:

J:=DF(EE,X,2); e [Enter] produz

J := 24*X

A operação DF é na realidade diferenciação parcial; variáveis diferentes daquela que está no segundo argumento são normalmente tratadas como se fossem constantes.

DF(X * Y,X); e [Enter] produz

Y

Se uma variável, Y deve ser tratada como dependente de outra X, a declaração DEPEND Y,X$ deve ser dada; com isso o resultado se apresentará bastante diferente e na resposta aparecerá DF(Y,X).

K:=DF(X*Y,X); e [Enter] produz

K := DF(Y,X)*X + Y

Se um valor vier a ser atribuído a Y o símbolo será calculado.

Se posteriormente numa seção do REDUCE, Y deixar de ser tratado como dependente de X, o comando NODEPEND Y,X$ deverá ser digitado para anular a declaração DEPEND.

Nota : Se Z depender tanto de X quanto de Y então DF(Z,X) e DF(Z,Y) não precisarão de dois comandos DEPEND e podemos escrever;

DEPEND Z,X,Y $

Observe que DEPEND Z,X,Y significa que Z depende de X e Y e não que Z e X dependem de Y. Para se expressar essa última hipótese dois comandos DEPEND são necessários.

2.6.7 Integração

INT é um operador em REDUCE para integração analítica usando uma combinação do algoritmo de Risch Norman e do algoritmo padrão. Em algumas implementações do sistema, o comando LOAD"INT"$ ou LOAD"INTEG"$ deve ser digitado antes do primeiro uso do INT em uma seção.

Sintaxe:

INT(EXPR: algébrica, VAR: Kernel): algébrica

Isto é, o primeiro argumento é o integrando(que é uma expressão algébrica) e o segundo argumento é a variável de integração.

Irá retornar corretamente a integral indefinida de expressões abrangendo polinômios, funções logarítmicas, exponenciais, TAN e ATAN. A constante arbitrária não é representada. Se a integral não puder ser feita em termos fechados, será retornada uma integral formal como resposta em um dos seguintes modos:

I) A mesma entrada inalterada, INT(... , ...).

II) Uma expressão envolvendo o INT com alguma outra expressão, na maioria das vezes mais complicada do que a original.

1: INT(2 * sin(X) - X^3),X); e [Enter] produz

$-\frac{8*COS(X)+X^4}{4}$

que é o resultado da integral indefinida da expressão dada com respeito a variável X.

Como se sabe, nem todo problema de integração tem uma resposta que pode ser expressa em termos de combinações de funções conhecidas. Por exemplo, os cálculos INT(E**(X**2),X) e INT(SIN(X)/X,X) não são bem sucedidos. Mesmo algumas outras integrais em a resposta exista e possa ser representada em termos de funções padrão, não são sempre manuseadas pelo operador INT do REDUCE; uma dessas limitações é por exemplo, uma integral tal como INT(1/SQRT(1 - X**2),X) que pode ser calculada de fato em termos da função ASIN (arcoseno). Esse problema resulta de certas falhas e lacunas da teoria, nas quais o método de integração do REDUCE se baseou, o que certamente virá a ser reparado em versões futuras; segundo informações de bibliografia especializada, o REDUCE 3.4 estará disponível para a comunidade usuária em 1992 onde esperamos essas críticas já tenham se tornado obsoletas.

O exemplo abaixo mostra a habilidade da operação integração para fazer fatoração e decomposição em frações parciais quando necessário.

2: A:=(X + 1)*($X + 3$) * *2* (X - 2); e [Enter] produz

$X^4 + 5 * X^3 + X^2 - 21*$ X - 18

O valor de A é armazenado nessa forma multiplicada; os fatores serão recuperados pelo INT por fatoração.

3: INT(X/A,X); e [Enter] produz

$(8 * LOG(X - 2) * X + 24 * LOG(X - 2) - 33 * LOG(X + 3) * X - 99 * LOG(X + 3) + 25 * LOG(X + 1) * X + 75 * LOG(X + 1) - 30 * X)/(300 * (X + 3))$

O operador DF conhece que DF de um INT é igual a mesma expressão com a qual se começou, por exemplo, consideremos;

4: A:= INT(SIN(10 * X^2) / X,X); e [Enter] produz

A:= INT($\frac{SIN(10*X^2)}{X}$,X)

Nota : Esse é um exemplo do caso (I) acima citado, a resposta é o problema sem alterações, pois SIN(10*X^2)/X não pode ser integrado, mas podemos derivar A para obter $\frac{SIN(10*X^2)}{X}$ de volta.

2.6.8 Listas

Uma habilidade muito poderosa do REDUCE é sua capacidade de manipulação de listas. Listas são as estruturas fundamentais de dados sobre os quais o próprio REDUCE foi construído, e nas quais se apoiam vários dos principais e maiores operadores do sistema. Uma lista é construída colocando-se os elementos dela entre chaves, separadas por vírgulas. Listas também podem ser vistas como sequências de expressões (que podem ser elas mesmas também listas). Os objetos colocados no interior de uma lista são calculados antes de serem armazenados, e daí podem limitar variáveis ou expressões que serão trocadas por seus valores tão bem quanto variáveis ilimitadas ou constantes. Portanto, não há necessidade de um operador especial para construir listas de objetos que devam ser primeiramente calculados, isto é feito automaticamente. Essa é a principal diferença entre listas no modo algébrico do REDUCE e as listas de LISP.

Partes de uma lista podem ser selecionadas com os operadores LAST, FIRST, SECOND, THIRD e REST. Um novo elemento pode ser acrescentado na frente de uma lista com o operador infix "." , (equivalente à função CONS do LISP).

```
lista = first(lista) . rest(lista)
second(lista) = first(rest((lista)))
third(lista) = first(first(rest(lista)))
```

Existem também no REDUCE as funções APPEND (que toma precisamente dois argumentos), REVERSE, etc. exemplificadas a seguir.

Exemplos:

1: U:= { 3, { $x**2, b$}, { { a },2b }, { 9,0 } }; e [Enter] produz

U:={ 3,{X^2, B},{{A},$2*B$ },{ 9,0 } }

2: P:={ { w },y,{ { },4,6 } }; e [Enter] produz

P:={ { w },y, { { },4,6 } }

3: append(U,P); e [Enter] produz

{ 3,{ X^2, B },{ { A },2 * B },{ 9,0 },{ W },Y,{ { },4,6 } }

4: length(P); e [Enter] produz

3

5: first(P); e [Enter] produz

{ W }

6: rest(P); e [Enter] produz

{ Y, { { },4,6 } }

7: reverse(U); e [Enter] produz

{ { 9, 0 },{ { A }, 2 * B },{ X^2,B },3 }

8: second(reverse(U)); e [Enter] produz

{ { A },2 * B }

Nota: A lista vazia, chamada NIL em LISP, é denotada por { } no REDUCE.

2.7 Programando no REDUCE

Muitas linguagens de programação permitem aos usuários contruir pacotes usando sequências de proposições e comandos. Esses pacotes são chamados "procedimentos" ou "subrotinas" ou "funções". REDUCE além de ser uma poderosa ferramenta quando utilizado no "modo calculadora" (digitando os comandos e examinando os resultados obtidos), também é uma das maiores linguagens entre os softwares de computação algébrica para a criação de procedimentos criados pelo usuário.

Procedimentos no REDUCE podem ser escritos no próprio REDUCE ou em LISP (RLISP). Características do REDUCE tais como NUM, DEN, INT, SUB, PART e outras, foram procedimentos escritos em RLISP, que manipulam a representação interna de expressões. Nosso texto não tem a intenção de ensinar essas formas internas e de como programar em LISP, mas para o leitor que conheça programação em LISP, temos a dizer que procedimentos escritos em LISP executam seu trabalho mais rapidamente que aqueles escritos em REDUCE, embora com um custo maior de programação e debugagem.

As definições de procedimentos escritos em RLISP começam com as palavras SYMBOLIC PROCEDURE, ou equivalentemente LISP PROCEDURE. As definições de procedimentos escritos em REDUCE começam com a palavra PROCEDURE ou ALGEBRAIC PROCEDURE, as quais tratamos nessa seção. Para permitir o conhecimento do investigador sobre o fácil acesso ao programa fonte do REDUCE, podemos dizer que procedimentos do sistema não estão protegidos de redefinições. Quando qualquer procedimento é redefinido, a mensagem

* * < procedure name > REDEFINED

é apresentada. Se isto ocorrer e o usuário não redefinir seu próprio procedimento, será avisado

para renomeá-lo e possivelmente começar de novo, saindo do REDUCE e entrando novamente como fez no começo da sessão, pois ele definiu algum procedimento interno cujo funcionamento correto pode ser usado no seu trabalho.

Procedimentos feitos pelo usuário carecem de muitas outras informações, que são encontradas na maioria das publicações pertinentes sobre o software encontradas no final desse capítulo; sugerimos fortemente a leitura dessa poderosa habilidade do sistema para construção dos seus próprios programas. Mostraremos a título de ilustração a construção de dois procedimentos escritos na linguagem do REDUCE. Um exemplo matemático interessante é o procedimento chamado TR, que mudará uma fração com o número complexo i aparecendo no denominador para uma fração equivalente onde não aparece i no denominador. Recordemos que o REDUCE tem uma regra pré-definida LET I** 2 = -1 de modo que qualquer I que não seja local ao procedimento (ou a uma proposição FOR ou a uma expressão FOR) funciona como raiz quadrada de -1.

Nota: Observe que ao digitar o procedimento seguido do seu nome com (;) e [Enter], REDUCE continua com o mesmo número de entrada com o qual começou, até que o investigador sinalize com o fim do programa (END;) ou (END$) e então muda para o número seguinte, aguardando o pedido de execução.

```
1: PROCEDURE TR C;
1: BEGIN SCALAR P, Q, R, S;
1:          P:=NUM C; Q:=DEN C;
1:          R:=SUB(I=-I, Q);
1:          P:=P*R; Q:=Q*R;
1:             ON GCD;
1:             S:=P/Q;
1:             OFF GCD;
1:          RETURN S;
1: END;
```

Nota: Aqui foi usada a conhecida técnica de se multiplicar o numerador e o denominador, pelo conjugado (chamado de R no programa) do denominador.

2: N:= (4+2 * i)/(1-3 * i); e [Enter] produz

$$-\frac{4+2I}{3I-1}$$

3: TR N; e [Enter] produz

$$\frac{7I-1}{5}$$

Se as construções IF...THEN...ELSE, WHILE, REPEAT, FOR e outras, não forem suficientes para descrever a lógica desejada de um procedimento, pode-se usar GO TO's no corpo desse

tal procedimento. O exemplo a seguir mostra o uso do GO TO no procedimento chamado de FATOL, que fornece uma definição para cálculo de fatoriais. Naturalmente é possível construir programas mais simples para esse fim, tente isso.

```
4: PROCEDURE FATOL N;
4: BEGIN SCALAR M, S;
4:          M:=1; S:=N;
4:     A:   IF S=0 THEN RETURN M;
4:          M:=M*S;
4:          S:=S-1;
4:          GO TO A;
4: END$
```

Nota: O estilo de programação usando GO TO pode manusear uma larga escala de programação usando recursão.

5: FATOL 48; e [Enter] produz

124139155925360726708622890473733750385214863546777600000000000

6: 3*FATOL 7; e [Enter] produz

15120

2.8 Formatos de Entrada e Saída

2.8.1 Geração de Programas em Outras Linguagens

O propósito dessa seção é mostrar que é possível gerar programas e saídas em outros códigos, a partir de entradas no REDUCE, fornecidas pelo usuário. Enfocaremos a geração de código FORTRAN, para o qual um suporte razoável é construído dentro do sistema. Programas em outras linguagens podem também ser gerados sem dificuldade; para isso existe um pacote no software, o GENTRAN, com a finalidade de gerar programas em FORTRAN, C, Ratfor, PASCAL etc. Maiores detalhes sobre o uso da versão corrente do GENTRAN são encontrados no manual do software e no livro Algebraic Computing with REDUCE.

2.8.2 Geração FORTRAN

O comando ON FORT; e [Enter] (chave FORT ligada), faz com que a saída seja compatível com FORTRAN: expressões começam na coluna 7, colunas 1 a 6 são deixadas em branco (exceto para

continuação de linhas na coluna 6); expoentes são escritos com **; coeficientes (não expoentes) tem pontos decimais; sinais como " = " são usados no lugar de " := ", e assim por diante. Se a expressão editada resultar de uma atribuição a uma variável, esta é editada como sendo o nome da expressão; senão recebe o nome default ANS. Erros ocorrem se identificadores e números estiverem fora dos limites permitidos pelo FORTRAN.

Nota: A saída gerada pelo REDUCE é automaticamente direcionada para uma variável denominada ANS, mas se outro nome default for desejado, o comando VARNAME deve ser usado antes de se escrever o arquivo.

VARNAME RESP; e [Enter] muda o default para " RESP= "

O modo de saída ON FORT está pronto para fazer vinte linhas continuadas. Esse limite é controlado pelo valor de variável CARDNO! *, que o investigador pode alterar, se sua versão do FORTRAN permitir um número maior ou menor de linhas de continuação. Se a expressão for demasiadamente longa, REDUCE quebra a expressão em pedaços chamados ANS1, ANS2, etc.

Exemplo:

1: X:=$(u+2*w-3*z\hat{}4)\hat{}3$

2: X; e [Enter] produz

3: $U^3+6*U^2*W-9*U^2*Z^4+12*U*W^2-36*U*W*Z^4+27*U*Z^8+8*W^3-36*W^2*Z^4+54*W*Z^8-27*Z^{12}$

4: ON FORT; e [Enter]

5: X; e [Enter] produz

$$\text{ANS} = U**3+6.*U**2*W-9.*U**2*Z**4+12.*U*W**2-36.*U*W*Z**4+27.*U*Z**8+8.*W**3-36.*W**2*Z**4+54.*W*Z**8-27.*Z**12$$

Nota: Quando não necessitarmos mais saídas em FORTRAN "desligar" a chave FORT, ou seja digitar OFF FORT; e [Enter].

2.9 Exercícios

1. Calcular :
   a) $\int cos^3 x dx$ b) $\int x^2 e^x dx$ c) $\int \frac{dx}{(x-1)(2x+1)}$

   d) $\frac{dy}{dx}$ onde $y = ln\sqrt{\frac{1+x}{1-x}}$ e) $\int \sqrt{\frac{1+x}{1-x}} dx$ f) $\int \sqrt{4-x^2} dx$

   Se o REDUCE falhar na última integral tente algint, e se falhar novamente use a substi-

tuição $x = sin2\theta$.

2. Seja $M = \begin{pmatrix} -1 & -4 & -2 & -2 \\ -4 & -1 & -2 & -2 \\ 2 & 2 & 1 & 4 \\ 2 & 2 & 4 & 1 \end{pmatrix}$. Calcule M^{-1} , o determinante de M , o traço de M , os autovalores e os autovetores de M e M^{178}.

3. Achar a solução geral da equação 7 sin x + 24 cos x = 15.
Sugestão: Aplicar a fórmula da soma trigonométrica.

4. Mostrar que a função $z = (x^2 + xy + y^2)e^{-x^2}$ tem valores estacionários nos pontos de coordenadas (0,0) , (-1, 1/2) , (1,-1/2), e determinar se eles são pontos de máximo , de mínimo ou de sela.

5. Achar os valores de π , e , $\sqrt{2}$, $2^{1/3}$, log 2 com 100 figuras significantes. Checar os valores aplicando aos resultados uma função inversa conveniente.

2.10 Publicações Pertinentes

Apresentamos agora uma pequena lista de publicações sobre o software, que embora restrita, em virtude do sistema ser o que tem maior número de artigos e livros sobre seu uso e poder, oferecem entretanto uma vasta gama de informações úteis e variadas para seus usuários.

1. FITCH J. P., *Solving Algebraic Problems with REDUCE* , J. Symbolic Computation, 1,211-277, U.S.A., 1985.

2. FLATAU P. J., BOYD J. P., COTTON W. R., *Symbolic Algebra in Applied Mathematics and Geophysical Fluid Dynamics - REDUCE Examples* , Dept. of Atmospheric Sciences, Colorado State University Report, U.S.A., 1986.

3. GRISS M. L., *The REDUCE System for Computer Algebra* , Proc. ACM 75, ACM 4-5, New York, U.S.A., 1975.

4. HEARN A. C., BRADFORD R. J., PADGET J. A., *Extending the REDUCE Domain of Computation* , Proc. of SYMSAC, 100-106, New York, U.S.A., 1986.

5. HEARN A. C., *The Personal Algebra Machine* , Info. Processing 80 (Proc. IFIP Congress 80), North-Holland, 621-628, U.S.A., 1980.

6. RAYNA G., *REDUCE Software for Algebraic Computation* , Springer-Verlag, New York, U.S.A., 1987.

7. WRIGHT F. J., MacCallum M. A. H., *Algebraic Computation with REDUCE* , Queen Mary College, University of London, U. K., England, 1991.

2.11 Informações Adicionais

Para informações complementares e adicionais sobre o REDUCE, tanto para problemas e bugs, quanto sobre a aquisição do sistema, fornecemos um dos endereços de distribuição do software; outros endereços são encontrados no capítulo 8 do nosso texto.

Northwest Computer Algorithms

P. O. Box 1747

Novato CA 94948

U.S.A.

Office Phone: (415) 897 - 1302

Fax Phone: (415) 898 - 2382

Observação : Referêcias sobre o software também podem ser obtidas através da REDUCE Network Library.
Endereço Eletrônico:

`redlib@rand.org`

Capítulo 3

MACSYMA

3.1 Introdução

MACSYMA é tanto um grande sistema interativo de computação algébrica, como também uma linguagem de programação com a finalidade de resolver problemas matemáticos. MACSYMA é utilizado como ferramenta auxiliar em pesquisas acadêmicas e científicas nos cinco continentes por engenheiros, físicos e matemáticos. O sistema foi desenvolvido no Massachussets Institute of Tecnology entre 1969 e 1982 por um grupo de pesquisadores liderados por J. Moses e R. Pavelle.

A partir de 1982, o MACSYMA passou a ser controlado e distribuído pela Symbolics, Inc. através do Computer Aided Mathematics Group - embora existam atualmente outras distribuidoras - sempre oferecendo novas versões. Em cada uma dessas novas versões, maiores habilidades e facilidades são acrescentadas tanto a nível de software quanto de hardware.

MACSYMA roda em alguns sistemas de computação entre os quais:

PC's $\longrightarrow$ PC 386 / 486 DOS

Workstations $\longrightarrow$ Apollo DN3000, 4000; Sun 3, 4; Symbolics 36xx; VAX 2xxx e MicroVAX II

Minicomputadores $\longrightarrow$ VAX 78x, 3xxx, 82xx, 83xx, 6xxx e 85xx

Minimainframe $\longrightarrow$ VAX 86xx+

MACSYMA já está em uso em centenas de Centros Científicos e Universidades, e juntamente com o REDUCE forma a dupla dos mais citados em artigos, conferências e publicações técnicas. A maioria dos artigos coloca o sistema como um dos melhores softwares já produzidos em computação algébrica.

3.2 O Porquê da Utilidade do MACSYMA

- As respostas são exatas e podem ser checadas por procedimentos independentes. Por exemplo, podemos calcular uma integral indefinida e checar a resposta por diferenciação; o algoritmo de diferenciação é independente do algoritmo de integração. Como são fornecidas respostas exatas, a análise do erro estatístico associado com computação numérica é desnecessário pois as respostas obtidas são confiáveis.

- O usuário pode gerar expressões FORTRAN, que permitem computadores numéricos rodarem com mais rapidez e eficiência. Isso salva os ciclos da CPU o que torna a computação mais econômica. O usuário pode gerar expressões FORTRAN de expressões MACSYMA, o que é uma característica extremamente importante pois combina capacidades numéricas e simbólicas. O encaminhamento é claro, e provavelmente em poucos anos já teremos poderosos desktops ou computadores de bolso que unirão capacidades simbólicas, gráficas e numéricas numa workstation científica.

- O investigador pode explorar problemas extremamente complexos que não podem ser resolvidos de outro modo. Essa capacidade é o que normalmente se pensa como um dos maiores usos em um sistema de computação algébrica, entretanto não podemos perder de vista que MACSYMA é mais geralmente utilizado como um calculador avançado para resolver problemas simbólicos e numéricos do dia a dia.

- Uma grande distribuição de conhecimento é dada na base do MACSYMA, o usuário tem acesso a técnicas matemáticas que não estão disponíveis em qualquer outro tipo de software. O investigador resolve problemas mesmo não conhecendo as técnicas usadas pelo sistema para se atingir a resposta.

- Um usuário pode testar facilmente e sem trabalho conjecturas matemáticas. Frequentemente encontramos resultados matemáticos na literatura e questionamos sua validade, MACSYMA pode ser usado para checar esses resultados usando técnicas algébricas, numéricas ou uma combinação delas. Da mesma forma podemos usar o sistema para mostrar que alguns problemas não tem solução.

- MACSYMA é facilmente utilizável; mesmo quem não tem experiência em computação pode aprender a resolver problemas razoavelmente difícies em poucas horas de uso. Embora MACSYMA tenha sido escrito em um dos dialetos LISP, o investigador não precisa

sequer conhecer sua linguagem base. MACSYMA é uma linguagem matemática completa, cuja sintaxe se assemelha a ALGOL.

- O investigador pode se concentrar no conteúdo intelectual de um problema deixando detalhes computacionais para o computador. Isso normalmente resulta em descobertas acidentais e devido ao poder do software essas ocorrências acontecem obviamente em intervalos de tempo fantasticamente menores do que quando cálculos são feitos à mão.

- Mas, talvez a mais importante razão para a utilização do MACSYMA esteja de comum acordo com a citação de R. W. Hamming, "O propósito da computação é perspicácia, não números". Isso exemplifica o maior benefício do uso do MACSYMA e a validade dessa afirmação é constatada quando vemos que ao conhecermos as potencialidades do sistema e o utilizarmos plenamente ganhamos perspicácia até para a construção da teoria.

3.3 Capacidades e Usos do MACSYMA

3.3.1 Capacidades do MACSYMA

Não é possível listar por completo todas as capacidades do MACSYMA em tão poucas linhas; o próprio manual de referência tem centenas de páginas, entretanto, algumas das mais importantes capacidades incluem (acrescentando as operações básicas aritméticas) facilidades para prover ferramentas analíticas para;

Limites	Simplificação
Derivação	Séries de Poisson
Integração Indefinida	Transformações de Laplace
Integração Definida	Somatório Indefinido
Equações Diferenciais Ordinárias	Manipulação Matricial
Sistemas de Equações (Não Lineares)	Manipulação Vetorial
Séries de Taylor (Várias variáveis)	Manipulação Tensorial
Fatorização	Geração Fortran
Substituição	Geração C
Plotagens em 2D e 3D	Programação

Existem outras rotinas para cálculos em teoria dos números, combinatória, teoria dos conjuntos, aritmética complexa etc., também existe uma biblioteca, chamada SHARE, contendo cerca

de 80 subrotinas onde algumas fazem computação tal como análise assintótica e otimização, manipulação das funções transcedentes mais elevadas etc.

Além disso, podemos calcular expressões numericamente com precisão arbitrária em muitos estágios da computação. MACSYMA também prove capacidades gráficas extensivas ao usuário.

Para colocar as capacidades do MACSYMA em perspectiva, podemos dizer que MACSYMA conhece uma larga porcentagem de técnicas matemáticas usadas em ciência e engenharia. Obviamente não estamos dizendo que MACSYMA faz tudo, é razoavelmente fácil apresentar exemplos que MACSYMA não manipula, apresentaremos também no trabalho alguns desses exemplos, mas podemos afirmar que o software é uma das ferramentas mais úteis e poderosas para qualquer engenheiro, matemático, físico ou cientista.

3.3.2 Usos do MACSYMA

Novamente é difícil listar os usos e campos de aplicação do sistema, pois os usuários geralmente não descrevem as ferramentas que os auxiliaram a trabalhar nas suas pesquisas, entretanto, a partir da literatura aberta da área e de conferências sobre computação algébrica, podemos listar alguns dos campos em que MACSYMA foi utilizado;

Acústica	Dinâmica de Fluidos
Geometria Algébrica	Relatividade Geral
Teoria de Antenas	Teoria dos Números
Mecânica Celeste	Análise Numérica
Desenho Auxiliar de Computadores	Física das Partículas
Teoria do Controle	Física do Plasma
Análise das Deformações	Física do Estado Sólido
Econometria	Mecânica Estrutural
Matemática Experimental	Termodinâmica
Análise dos Elementos Finitos	Robótica
Modelagem de Polímeros	Análise spectral
Ressonância Magnética Nuclear	Teoria do Controle de Otimização
Ondas de choque Submarinas	Química da Emulsão

Naturalmente não mostramos várias usos, áreas ou campos de aplicação do sistema, mas essas apresentadas por notícias de pesquisadores já dão uma idéia do poder e abrangência do software.

3.4 Apresentação do MACSYMA

3.4.1 Preliminares

O propósito desse parágrafo é apresentar uma pequena mostra sobre a utilização do MACSYMA, e tem também a intenção de ser intelegível para o leitor que não está familiarizado com o sistema. No final do parágrafo listamos outras fontes para aprofundamento e maiores informações sobre o software.

Sugerimos ler essa apresentação em frente a uma máquina que rode o MACSYMA, para que o leitor possa executar nossas sugestões como input inicial, inclusive fazendo alguns exercícios elementares, encontrados no final do parágrafo, para posteriomente trabalhar em seus objetivos próprios.

A versão utilizada no nosso estudo foi a do MACSYMA da Symbolics, operando em uma estação de trabalho SUN no LNCC (Laboratório Nacional de Computação Científica).

Para entrar no MACSYMA na maioria das máquinas, bater:

MACSYMA e [Enter]

Será fornecida uma mensagem, como abaixo, ou outras similares em versões mais recentes através da tela.

```
This is MACSYMA 417.100 for SPARC Series Computers

Copyright (c) 1982 Massachusetts Institute of Technology.

All Rights Reserved.

Enhancements (c) 1986, 1991 Symbolics, Inc. All Rights Reserved.

Type DESCRIBE(TRADE\_SECRET) ; to see important legal notices.

Type " HELP( ); " for more information.


(C1)
```

O (C1) é um rótulo, cada linha de entrada ou de saída é rotulada. Os rótulos (Ci) significam comandos de entrada e terminam com " ; " ou com " $ " e cada rótulo (Di) significa o display de saída que é de responsabilidade da máquina. Nunca usar nomes de variáveis tais como C1 ou D5 por exemplo, pois pode haver confusão com as linhas rotuladas.

Vamos mostrar agora um exemplo trivial que demonstra uma das habilidades do MACSYMA para resolver equações. Na linha de entrada (C1) escrevemos uma expressão numa sintaxe parecida com ALGOL finalizando com um ponto e vírgula e em (D1) o computador apresenta a saída em uma maneira similar a notação manual.

(C1) X ^ 3 - 3*B*X ^2 + A*X ^2 - 3*A*B*X - 5*X ^2 + 15*B*X - 5*A*X + 15*A*B = 0 ;
e [Enter] produz,

(D1) X^3 - 3BX^2 + AX^2 - 5X^2 - 3ABX + 15BX - 5AX + 15AB = 0

Em (C2) pedimos para resolver a expressão (D1) em X e então as três raízes aparecem numa lista em (D2).

(C2) SOLVE(D1,X); e [Enter] produz,

(D2) [X = 3B , X = - A , X = 5]

Observe que MACSYMA obteve as raízes analiticamente, as aproximações numéricas não foram feitas. Isso demonstra uma diferença fundamental entre um sistema de computação algébrica e um resolvedor numérico de equações ordinárias, mostrando as habilidades para obtenção de soluções sem aproximações. Note que poderíamos ter dado também para o MACSYMA uma equação numérica cúbica em X especificando valores numéricos para A e B, MACSYMA teria então resolvido a equação e dado as raízes numéricas aproximada ou exatamente dependendo do comando específico.

MACSYMA também pode resolver equações quadráticas, cúbicas e quárticas tão bem como classes de equações de grau mais alto, entretanto claramente não resolve equações analiticamente em forma fechada quando métodos ainda não são conhecidos. MACSYMA pode entretanto achar soluções reais aproximadas de equações polinomiais de qualquer grau com precisão arbitrária.

3.4.2 Símbolos e Chaves Especiais

- Para terminarmos uma seção do MACSYMA bater QUIT( ): Em algumas máquinas ^C ou ^Y ou ^Z também funcionam. Aqui ^ é entendida como a chave de controle, de modo que^C significa apertar simultâneamente a chave de controle marcada e a chave C.

- Para abortarmos uma computação sem sair do MACSYMA bater^G ou^C ou apertar a chave ABORT. Esse passo difere de uma máquina para outra, é importante saber como

isso é feito na sua máquina, pois as vezes precisamos desse detalhe numa computação que leve muito tempo em execução.

- Para informarmos ao MACSYMA que terminamos um comando, usar ponto e vírgula (;). Em algumas máquinas basta bater (:) com return. Observe que a tecla return sozinha não sinaliza ao software o que deve ser calculado.

- Um finalizador alternativo que pode ser utilizado em vez do ponto e vírgula (;) é o sinal dolar ($) que informa ao MACSYMA para não imprimir o resultado. Ele é útil quando se está computando alguns resultados intermediários extensos e não se pode perder tempo com os displays de saída aparecendo na tela.

- Se quisermos apagar completamente a entrada corrente (e começar tudo novamente do início) bater uma interrogação dupla (??).

- Se quisermos repetir um comando que já tenha sido dado, digamos na linha (C5), não precisamos bater tudo novamente, basta bater ("), isto é "C5 . Observe que se entrarmos somente com C5 esse trabalho não será feito.

- Se precisarmos nos referir ao resultado imediatamente anterior computado pelo MACSYMA, podemos usar o seu próprio rótulo D ou usar o símbolo especial (%).

- As quantidades padrão e (base natural de log.), i (raíz quadrada de -1) e $\pi = 3.14159...$ são respectivamente referidas como % E , % I e % PI. Note que o uso de % aqui, como um prefixo, é completamente diferente do uso de % que faz referência ao resultado anterior computado.

- Ao invés de atribuir um valor para uma variável, MACSYMA usa o sinal (:) não com o sinal de igual. O sinal de igual é usado para representar equações.

- Existem outros símbolos especiais, por exemplo, no sintaxe dos limites, quando o limite for indefinido é retornado und e se for indefinido mas limitado é retornado ind.

3.5 Entrada das Expressões

A entrada de expressões no MACSYMA não é muito diferente da maioria dos softwares de computação algébrica estudados no nosso trabalho, quando estudarmos a sintaxe de entrada do sistema veremos esse fato, mas, nesse momento gostaríamos de colocar uma observação que nos parece no mínimo curiosa e pertinente.

O MACSYMA é um dos programas de computação algébrica mais conhecidos e respeitados

pela comunidade usuária e, realmente faz um trabalho muito abrangente na construção da matemática, e embora não cheguemos a níveis profundos de estudo sobre a programação do MACSYMA podemos criticar a nomenclatura do seu superconjunto, ou seja, "Os Sistemas de Computação Algébrica" pois o MACSYMA, como também outros softwares dessa área, faz muito mais do que álgebra.

As operações aritméticas comuns no MACSYMA são:

+ adição

- subtração

* multiplicação comutativa

/ divisão

^ ou ** exponenciação

. multiplicação não comutativa

! fatorial

SQRT(X) raiz quadrada de X

A saída no MACSYMA é caracterizada por aritmética (racional) exata, por exemplo;

(C1) 5/10000+5/10001; e [Enter] produz,

(D1) $\frac{20001}{200200}$

Se números racionais são envolvidos em uma computação, eles não retornam na forma simbólica;

(C2) (5+SQRT(5))^5; e [Enter] produz,

(D2) $(SQRT(5)+5)^5$

(C3) EXPAND(%); e [Enter] produz,

(D3) 4400 SQRT(5) + 10000

Nota : Algumas vezes omitiremos o [Enter] nos comandos (Ci) para simplificar a escrita.

Entretanto é geralmente útil se expressar um resultado numa notação decimal. Para fazer isso na expressão que queremos expandir, bater em seguida " . NUMER" daí;

(C4) % , NUMER;

(D4) 19838.69910099907

Note que a utilização de % aqui se referiu ao resultado prévio. Nessa versão do MACSYMA, NUMER apresenta 16 figuras significantes.

MACSYMA pode manusear números muito grandes sem aproximação;

(C5) 99!

(D5) 933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000

3.6 Sintaxe de Algumas Capacidades do MACSYMA

Mostraremos agora a sintaxe correta de algumas das capacidades do sistema, escolhidas aleatoriamente, mas que de qualquer forma auxiliam o investigador a dar os primeiros passos no software, embora recomendemos fortemente o uso do manual do MACSYMA para um maior aprofundamento tanto no uso da sintaxe, quanto na amplitute de tratamento dado para essas e outras capacidades.

3.6.1 Expansão e Substituição Polinomial

A importância do MACSYMA como uma ferramenta computacional para facilitar o cálculo se torna evidente quando vemos quanto facilmente ele faz álgebra. Aqui está um exemplo onde um polinômio é expandido;

(C1) (X + 3*Y + X^2*Y) ^ 3;

(D1) $(X^2Y + 3Y + X)^3$

(C2) EXPAND(%);

(D2) $X^6Y^3 + 9X^4Y^3 + 27X^2Y^3 + 27Y^3 + 3X^5Y^2 + 18X^3Y^2 + 27XY^2 + 3X^4Y + 9X^2Y + X^3$

Suponhamos agora que quisessemos substituir 5/Z por X na expressão acima;

(C3) D2,X=5/Z;

(D3) $\frac{135Y^2}{2} + \frac{675Y^3}{Z^2} + \frac{225Y}{Z^2} + \frac{2250Y^2}{Z^3} + \frac{125}{Z^3} + \frac{5625Y^3}{Z^4} + \frac{1875Y}{Z^4} + \frac{9375Y^2}{Z^5} + \frac{15625Y^3}{z^6} + 27Y^3$

A função RATSIMP do MACSYMA colocará esse resultado com um denominador comum;

(C4) RATSIMP(%);

(D4) $(27Y^3Z^6 + 135Y^2Z^5 + (675Y^3 + 225Y)Z^4 + (2250Y^2 + 125)Z^3 + (5625Y^3 + 1875Y)Z^2 + 9375Y^2Z + 15625Y^3)/Z^6$

Expressões podem também ser fatoradas;

(C5) FACTOR(%);

(D5) $\frac{(3YZ^2+5Z+25Y)^3}{Z^6}$

3.6.2 Equações, Inequações e Sistemas

MACSYMA pode obter soluções exatas para sistemas de equações não lineares, no exemplo seguinte resolvemos três equações em três termos desconhecidos A,B e C;

(C1) eq1: A+B*C=13/30;

(D1)B*C + A = $\frac{13}{30}$

(C2) eq2: A*C+B*C= -1/6;

(D2) BC + AC = $\frac{-1}{6}$

(C3) eq3: A + B^2 = 11/18;

(D3)$A + B^2 = \frac{11}{18}$

(C4)Solve([eq1, eq2,eq3],[A, B, C]);

```
(D4) [[A = -1.61272770286255, B = 1.491254186825456, C = 1.372040586245772],

      [A = 0.4896971712526473 - 0.1817316029713937 % I,
       B = -0.2203909444979289 % I - 0.4122937160267582,
       C = -0.3996594350270231 % I - 0.07692933753504208],

      [A = 0.1817316029713937 % I + 0.4896971712526473,
       B = 0.2203909444979289 % I - 0.4122937160267582,
       C = 0.3996594350270231 % I - 0.07692933753504208],

      [A = 1/2 , B = 1/3 , C = -1/5]]
```

3.6.3 Limites

No MACSYMA existem dois comandos básicos para determinação de limites, embora existam outros que apresentam mais detalhes sobre limites indefinidos, limites bilaterais, etc. Exemplos

são encontrados no MACSYMA User's Guide.

- *limit* calcula o limite de uma expressão quando uma dada variável real tende a algum valor, inclusive infinito.
- *tlimit* é semelhante a limit, exceto quando este usa série de Taylor quando possível.

Sintaxe para cálculo de limites de algumas funções:

lim(expressão, variável, valor, direção)

Acha o limite da "expressão" quando a "variável" real tende ao dado "valor" em alguma direção.

Se não for especificada a direção, o limit calcula o limite bilateral.
Exemplos:

(C1) limit((x + exp(x) + exp(2*x))^(1/x), x, inf) ;

(D1) % E^2

(C2) expr:1/(exp(1/x) + 1) ;

(D2) $\frac{1}{1\%E^{1/x}+1}$

Para calcular o limite da expressão expr quando x tende a zero pela direita, fazemos então:

(C3) limit(expr, x, 0, plus);

(D3) 0

Para calcular o limite da expressão expr quando x tende a zero pela esquerda, procedemos da seguinte forma:

(C4) limit(expr, x, 0, minus);

(D4) 1

Note que o limite bilateral da expressão expr quando x tende a zero é indefinido, pois:

(C5) limit(expr, x, 0);

(D5) UND

(C6) limit(SQRT(X^2 + X) - SQRT(X^2 - X), X, INF);

(D6) 1

3.6.4 Derivadas

Qualquer software de computação algébrica contém um conjunto mais ou menos completo de comandos conectados com o conceito de diferenciação. Essas operações são linkadas aos comandos de simplificação, pois sem boas ferramentas de simplificação existem riscos de termos fórmulas incompreensíveis muito rapidamente; o MACSYMA entretanto trabalha muito bem com esse tipo de tarefa. Observe nos exemplos a seguir que podemos pedir derivadas de funções de modos distintos.

Sintaxe da diferenciação de algumas funções

(C1) DIFF(SIN(X^4) - log(X + Y), X) ; e [Enter] produz,

(D1) $4X\text{^}3COS(X\text{^}4) - \frac{1}{Y+X}$

(C2) f1:2 * (log(x ^ 2 - x + 1) / 6 + atan((2 * x - 1) / sqrt(3)) / sqrt(3) - log(x + 1) / 3 ; e [Enter], produz na saída seguinte a mesma fórmula na notação matemática usual,

(D2) $2\ (\frac{LOG(X^2-X+1)}{6} + \frac{ATAN(\frac{2X-1}{SQRT(3)})}{SQRT(3)} - \frac{LOG(X+1)}{3}\ )$

(C3) df1: diff(f1, x);

(D3) $2\left(\frac{2}{3(\frac{(2X-1)^2}{3}+1)} + \frac{2X-1}{6(X^2-X+1)} - \frac{1}{3(X+1)}\right)$

3.6.5 Integrais

O mecanismo de procura de integrais sempre foi uma das partes mais importantes na computação algébrica pelas ferramentas de trabalho que fornece ao investigador; os pesquisadores mais conhecidos na construção de algoritmos para esse fim e com relação ao aproveitamento de uso de memória das máquinas na integração foram; J. Moses - que por ser o criador do sistema permitiu que o MACSYMA fosse um dos primeiros softwares da área a ser beneficiado com um método avançado de integração - J. Slagle e Risch.

Sintaxe de Integração de algumas funções

A sintaxe de integração indefinida é bastante simples, o comando no MACSYMA é integrate que pode ser usada de maneiras distintas, aqui estão alguns exemplos triviais;

(C1) integrate(x^ 3 *cos(x^ 2),x) ; e [Enter] produz,

(D1) $\frac{X^2SIN(X^2)+COS(X^2)}{2}$

Nota : Se,por exemplo, uma expressão for digitada em (C2), em (D2) ela aparecerá no formato usual de matemática, para calcularmos a integral na variável x basta digitarmos

em (C3) integrate (D2, x) que teremos a solução em (D3).

A sintaxe da integração definida é um pouco mais difícil de codificar do que a integração indefinida pois o número de técnicas conhecidas é muito maior, mas o MACSYMA opera fantasticamente bem com integração definida. O símbolo inf é o que o MACSYMA reserva para inf. Se o comando integrate necessitar de informações ele colocará essas questões para o investigador.

(C4) integrate(1/sqrt((1 + x)^ 3 * (1 + k*x)), x, 0, inf) ;
Is k positive, negative or zero ?
positive;
Is k - 1 zero or nonzero ?
nonzero;
Is k - 1 positive or negative ?
positive;

(D4) $\frac{2SQRT(K)}{K-1}$ - $\frac{2}{K-1}$

Comentário : A integral abaixo teve resposta errada no MAPLE e não foi feita pelo DERIVE.

(C5) integrate(sqrt((1+x)/(1-x)),x,0,1);

(D5) $\frac{\%PI}{2} + 1$

3.6.6 Desenvolvimento em Séries de Taylor

MACSYMA faz desenvolvimentos em Séries de Taylor através de uma função interna chamada taylor.

Sintaxe para Séries de Taylor

taylor(função, variável, inicial, ordem)

O símbolo " /T/ " na frente do resultado significa que estamos lidando com uma expressão truncada. Operações em resultados dessa espécie obviamente levam em conta a ordem fixada.

(C1) h: (1 - sqrt(1 - x^ 2))/x ;

(D1) $\frac{1-SQRT(1-X^2)}{X}$

(C2) taylor(h, x, 0, 10) ;

(D2) /T/ $\frac{X}{2} + \frac{X^3}{8} + \frac{X^5}{16} + \frac{5X^7}{128} + \frac{7X^9}{256} + \ldots$

(C3) g: sin(2 - 3 * x);

(D3) -sin(3X - 2)

(C4) taylor(g,x,0,5);

(D4) /T/ -SIN(-2) -3 COS(-2) X + $\frac{(9SIN(-2))X^2}{2}$ + $\frac{(9COS(-2))X^3}{2}$ - $\frac{(27SIN(-2))X^4}{8}$ - $\frac{(81COS(-2))X^5}{40}$ + ...

3.6.7 Somatórios

Modernamente algoritmos são descobertos de um modo geral, combinando-se os sistemas de computação algébrica com os procedimentos da matemática experimental; isso significa que se ganha experiência empírica com um método heurístico que geralmente trabalha bastante, até que alguém tenha uma idéia de como as respostas para um problema particular se parecem. Um exemplo dessa técnica é o algoritmo para somatório indefinido (finito) criado por Gosper com o MACSYMA. As restrições para os procedimentos são que ambos, o somando e as respostas têm que ser expressos como produto de potências, fatoriais, binomiais e/ou funções racionais.

Mesmo com essas restrições é possível manusear uma grande classe de somas indefinidas. A história do problema mais geral para desenvolvimento de somatórios indefinidos contém os nomes de Newton, Euler, Bernoulli e Boole mas, a despeito dos muitos anos que esses matemáticos devotaram a esses problemas, os resultados surpreendentemente foram em um número bem restrito. Uma razão para se acreditar fortemente na existência de um algoritmo satisfatório foi em virtude do sucesso alcançado na resolução de problemas de integrais indefinidas e os paralelos conceituais entre os dois. A aproximação de Gosper envolvia testar muitos casos com o MACSYMA até emergir um modelo que funcionasse bem para todos os casos e conduzisse a descoberta do algoritmo.

Recentemente, usando técnicas algébricas abstratas semelhantes àquelas que Risch usou para integração, Karr estendeu o trabalho de Gosper. Algumas partes do algoritmo de Gosper foram implementadas no MACSYMA, aqui estão exemplos do seu poder.

Sintaxe para Somatório Indefinido

(C1) sum((A ^ N * N ^ 4),N,0,M); com [Enter] produz,

(D1) $\sum_{N=0}^{M} A^N N^4$

A expressão (D1) é a soma indefinida que queremos calcular. O comando NUSUM, que faz essa computação é digitado em (C2).

(C2) NUSUM(A ^$N * N$^4, $N, 0, M$); e [Enter] produz,

(D2) $A^{M+1}(A^4M^4 - 4A^3M^4 + 6A^2M^4 - 4AM^4 + M^4 - 4A^3M^3 + 12A^2M^3 - 12AM^3 + 4M^3 + 6A^3M^2 - 6A^2M^2 - 6AM^2 + 6M^2 - 4A^3M - 12A^2M + 12AM + 4M + A^3 + 11A^2 + 11A + 1)/(A - 1)^5 - \frac{A(A+1)(A^2+10A+1)}{(A-1)^5}$

Nota : Em uma workstation Sun (SPARC STATION 1) o MACSYMA levou 1 minuto e 12 segundos para fazer essa computação.

É possível se checar a resposta chegando ao somando inicial, de modo análogo a maneira que se checa uma integral indefinida diferenciando-a.

(C3) UNSUM(D2, N); e [Enter] produz, (D3) $A^N N^4$

Sintaxe para Somatório Finito

Apresentaremos a seguir um exemplo com sintaxe para somatório finito, baseado na série geométrica definida em (D1) a seguir para um inteiro arbitrário M, cuja expressão é identicamente nula. Para gerar um problema mais complicado substituiremos 1 + 1/X por X em (D1) resultando (D2).

(C1) sum(x ^ n - x * (x ^ n - 1)/ (x-1), n,1,m);

(D1) $\sum_{N=1}^{M} X^N - \frac{X(X^N-1)}{X-1}$

(C2) % = 0;

(D2) $\sum_{N=1}^{M} X^N - \frac{X(X^N-1)}{X-1} = 0$

(C3) D2, X = 1 + 1/X;

(D3) $\sum_{N=1}^{M}(\frac{1}{X}+1)^N - ((\frac{1}{X}+1)^N - 1)(\frac{1}{X}+1)X = 0$

Nota : Essa expressão (D2) também tem que se anular para qualquer inteiro M

Em (C3) é pedido ao MACSYMA para fazer o somatório de (D2) de 1 a 1000. A expressão truncada (D3) tem um grande número de termos. A função NTERMS do MACSYMA fornece um manuseamento de quão grande uma expressão poderia ficar se ela fosse totalmente expandida sem nenhuma simplificação de termos. MACSYMA respondeu que existiriam 503504 termos antes de ser feita qualquer simplificação. Do teorema binomial não é difícil de ser mostrado que esse número é precisamente $(M\hat{}2 + 7 * M + 8)/2$ para M = 1000. MACSYMA simplifica então (D3) através do comando RATSIMP (D3); e realmente a resposta esperada será retornada diminuindo consideravelmente o número de termos.

(C3) D2,M = 1000, SUM;

(D3) $(\frac{1}{X}+1)^{1000} + \ldots + (\frac{1}{X}+1) + 1 - ((\frac{1}{X}+1)^{1000} - 1)(\frac{1}{X}+1)X = 0$

(C4) NTERMS(LHS(D3));

(D4) 503504

3.6.8 Vetores e Matrizes

Vetores e matrizes, constantemente utilizados em várias áreas de conhecimento, são obviamente necessários em qualquer sistema de computação algébrica. MACSYMA tem capacidades poderosas para manipulação de matrizes e faz todas as operações comuns, ou seja, pode calcular determinantes, inversas, autovalores e autovetores de matrizes com elementos simbólicos, isto é, elementos que envolvam variáveis algébricas. Existem várias maneiras de entrar com matrizes no MACSYMA; através de declaração com atribuição interativa, através de declaração explícita, via uma função geradora dos elementos etc.

Sintaxes mais comuns para entrada de matrizes

A tela de entrada ao usarmos declaração com atribuição interativa se apresenta conforme abaixo, de acordo com a sintaxe de entrada em (C1),

(C1) M : ENTERMATRIX(3, 3) ; e [Enter] produz,

Is the matrix 1. Diagonal 2. Symmetric 3. Antisymmetric 4. General
Answer 1, 2, 3 or 4
Selection : 4;

Row 1 Column 1 : 1;

Row 1 Column 2 : -2;

Row 1 Column 3 : -2;

Row 2 Column 1 : 0;

Row 2 Column 2 : 1;

Row 2 Column 3 : 0;

Row 3 Column 1 : 0;

Row 3 Column 2 : 2;

Row 3 Column 3 : A;

Matrix Entered

(D1)

```
[ 1   -2   -2 ]
[             ]
[ 0    1    0 ]
[             ]
[ 0    2    A ]
```

Depois achamos sua transposta, seu determinante e sua inversa:

(C2) TRANSPOSE(M); e [Enter] produz

(D2)

```
[  1  0  0 ]
[          ]
[ -2  1  2 ]
[          ]
[ -2  0  A ]
```

(C3) DETERMINANT(M); e [Enter] produz

(D3) A

(C4) INVERT(M), DETOUT; e [Enter] produz

(D4)

```
[ A  2A - 4  2 ]
[              ]
[ 0     A    0 ]
[              ]
[ 0    -2    1 ]
----------------
        A
```

Em (C4), o modificador mantém o determinante fora da inversa. Como um teste, multiplicamos M por sua inversa(note o uso do período para representar multiplicação de matrizes):

(C5) RATSIMP(M.D4);

(D5)

```
[ 1  0  0 ]
[         ]
[ 0  1  0 ]
[         ]
[ 0  0  1 ]
```

Para achar os autovalores e autovetores de M, usa-se a função EIGENVECTORS:

(C7)EIGENVECTORS(M):

(D7) [[[A , 1],[1 , 2]],[1 , 0 , $-\frac{A-1}{2}$],[1 , 0 , 0]]

A primeira lista em (D7) é [[A , 1],[1 , 2]] o que significa que os autovalores de M são A e 1 sendo A de multiplicidade 1, e 1 de multiplicidade 2. A segunda lista em (D7) é [1 , 0 , $-\frac{A-1}{2}$] que é base para o autoespaço associado ao autovalor A. A terceira lista em (D7) é [1 , 0 , 0] que é um autovetor (base) do autoespaço correspondente ao autovalor 1 de multiplicidade 2.

Para extrair desta expressão um dos autovetores devemos usar a função PART:

(C8) PART(%,2);

(D8) [1 , 0 , $-\frac{A-1}{2}$]

3.6.9 Simplificação

Uma importante característica do MACSYMA é sua habilidade para simplificar expressões. Num estudo sobre métricas das ondas planas, um pesquisador chegou a um cálculo particular que tinha produzido uma expressão com centenas de milhares de termos. Por argumentos geométricos ele concluiu que ela poderia ser simplificada e o MACSYMA transformou aquelas centenas de folhas de saída em um pequeno número de folhas. A expressão abaixo ocorria repetidamente e causava o colapso da expressão maior durante a simplificação.

(D1) $\frac{(SQRT(R^2+A^2)+A)(SQRT(R^2+B^2)+B)}{R^2} - \frac{SQRT(R^2+B^2)+SQRT(R^2+A^2)+B+A}{SQRT(R^2+B^2)+SQRT(R^2+A^2)-B-A}$

(C2) RATSIMP(D1);

(D2) 0

Nota: Quando o simplificador canônico RATSIMP é chamado acima em (D1) retornou o valor zero. Embora o MACSYMA seja um sistema extremamente confiável fizemos esse cálculo à mão em exatamente 48 minutos e confirmamos a resposta.

3.7 Tratamento Gráfico

Embora o tratamento gráfico oferecido pelo software seja bom, aconselhamos ao investigador a consulta do manual de refência do sistema, pois em algumas versões, nem todas as habilidades e características de plotagem estão disponíveis e, também porque nessa seção apresentamos apenas uma limitada introdução a essas habilidades.

Plotagens Bidimensionais

plot ⟶ plota uma expressão na direção y enquanto o eixo x toma valores no domínio que tiver sido especificado pelo usuário.

Paramplot ⟶ plota uma expressão em coordenadas de x contra outra expressão em coordenadas de y num domínio especificado pelo usuário.

graph ⟶ plota pontos especificados por uma lista de abscissas e ordenadas.

plotnum ⟶ faz a plotagem para o número de pontos especificados pelo usuário, por default o valor do plotnum é 20.

A sintaxe plot(exp, var, bound1, bound2) produz uma plotagem bidimensional da expressão exp, onde var é a variável a ser plotada e bound1 e bound2 são os limites dados para os valores de plotagem de x.

O exemplo seguinte plota a expressão, chamada em (C1) de plota_graf para x entre -3 e 3. O resultado é mostrado no gráfico abaixo. O comando nameplot é descrito no final dessa seção.

(C1) plota_graf: -2 * x * exp(-x ^ 2); e [Enter] produz,

(D1) $-2X\%E^{-X^2}$

(C2) plot(plota_graf,x,-3,3); e [Enter] produz,

(D2)DONE

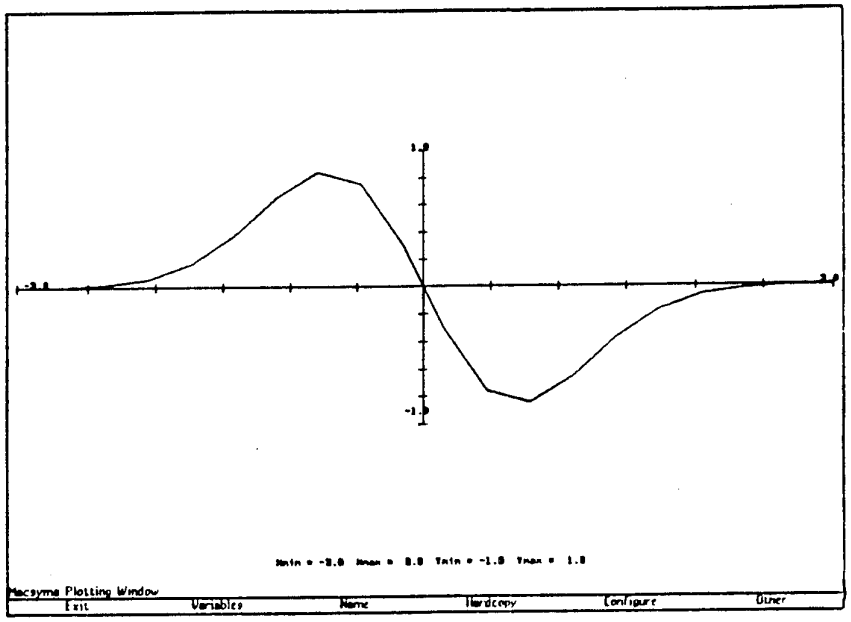


FIGURA 1

Ao fazer as plotagens abaixo fica fácil entender os comentários teóricos apresentados na sintaxe anterior.

(C4) plot(plota_graf,x,[-3,-2,-1,0,1,2,3]); e [Enter] produz a plotagem desejada em (D4).

O comando paramplot(x_expr, y_expr, var, bound1, bound2) plota a expressão ou lista de expressões x_expr como coordenadas de x enquanto que a expressão ou lista de expressões y_expr como coordenadas de y; bound1 e bound2 dão o domínio de plotagem.

Ao fazer a plotagem a seguir, paramplot plota a expressão x_expr contra a expressão y_expr.

(C5) x_expr: cos(2 * x); e [Enter] produz,

(D5) COS(2X)

(C6) y_expr: sin(x);

(D6) SIN(X)

(C7) paramplot(x_expr,y_expr,x,-5* % pi,5* % pi); e [Enter] produz a plotagem pedida em (D7).

Para uma representação mais refinada pode ser feita a plotagem com um maior número de pontos.

(C8) paramplot(x_expr,y_expr,x,-5* % pi,5* % pi),plotnum:100; e [Enter] produz a plotagem em (D8).

Nota : É possível plotar novamente a figura anterior em coordenadas polares segundo a sintaxe.

(C9)paramplot(x_expr,y_expr,x,-5* % pi,5* % pi,polar),plotnum:100; e [Enter] produz a plotagem das expressões definidas em coordenadas polares em (D9).

O comando graph(x-list, y-list) plota pontos pela lista - ou lista de listas - x-list e y-list.

Plotagens Tridimensionais

plot3d ⟶ plota tridimensionalmente uma expressão com respeito às variáveis especificadas x e y.

contourplot ⟶ calcula as mesmas plotagens, como em plot3d, sendo que os pontos de display saem em plotagens de contorno.

graph3d ⟶ produz uma plotagem tridimensional especificada por listas de pontos x, y e z.

A sintaxe plot3d(exp, xvar, xlim1, xlim2, yvar, ylim1, ylim2) produz uma plotagem tridimensional da expressão exp onde xvar e yvar são as variáveis a serem plotadas, xbound1 e xbound2 são os valores dados sobre os valores de plotagem de x e yboud1 e ybound2 são os limites dados sobre os valores de plotagem de y.

O exemplo seguinte plota a expressão chamada em (C10) de p2_expr para x entre 0 e 2 e para y entre -2 e 2. O resultado é mostrado no gráfico abaixo.

(C10) p2_expr: y ^ 2 + cos(4 * x) + 2 * x; e [Enter] produz,

(D10) $Y^2 + COS(4X) + 2X$

(C11) plot3d(p2_expr,x,0,2,y,-2,2); e [Enter] produz em (D11) a figura abaixo:

(D11)DONE

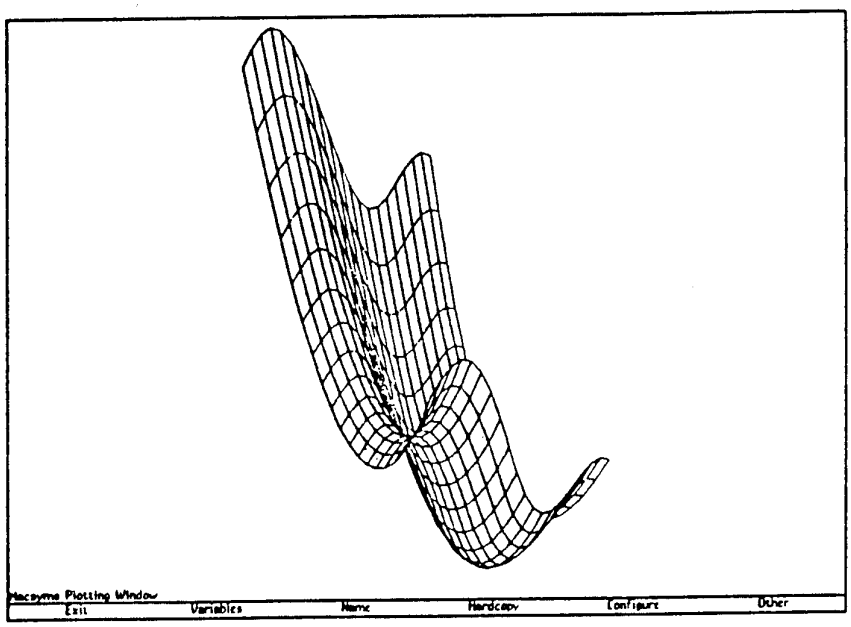


FIGURA 2

O comando equalscale, discutida com detalhes no manual, indica se a escala da plotagem é ou não a mesma em ambas as direções.

(C12) plot3d(p2_expr,x,0,2,y,-2,2),equalscale:false; e [Enter] produz em (D12) a figura abaixo:

(D12)DONE

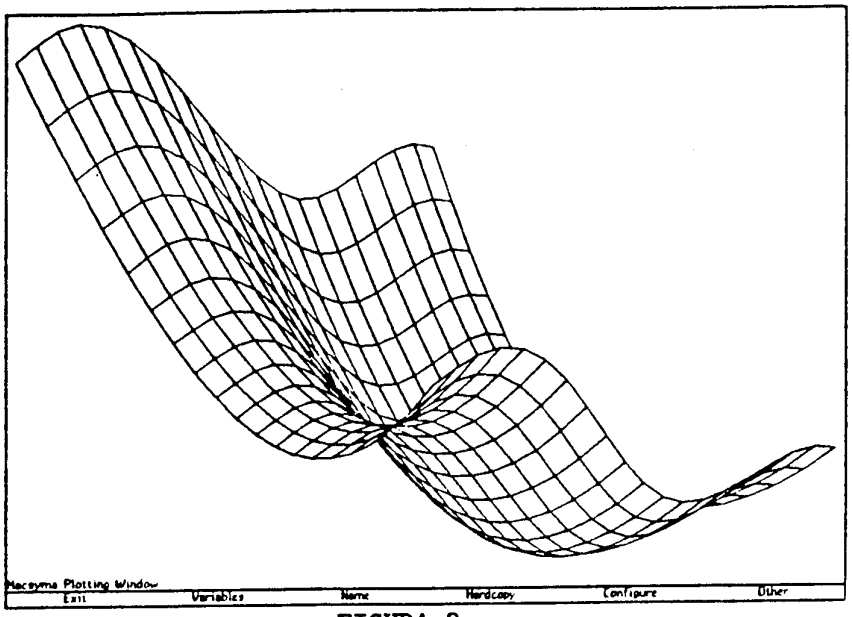


FIGURA 3

O comando contourplot(exp, xvar, xlim1, xlim2, yvar, ylim1, ylim2) calcula os mesmos pontos que plot3d, mas produz uma plotagem de contorno da expressão.

O próximo exemplo produz uma plotagem de contorno da expressão em (C10).

(C13) contourplot(p2_expr,x,0,2,y,-2,2), equalscale:false; e [Enter] produz em (D13) a plotagem das curvas de contorno da expressão.

Nota : Existem vários comandos para dar nomes a plotagens, mudar escalas de plotagens etc., detalhados no manual do MACSYMA; alguns desses comandos apresentamos a seguir:

- nameplot dá nome à mais recente plotagem feita.
- saveplots salva as plotagens cujos nomes foram especificados pelo usuário.
- loadplots carrega plotagens já salvas.
- killplots mata plotagens correntemente carregadas para aumentar o espaço livre armazenado.

3.8 Equações Diferenciais

Outro poderoso feito do MACSYMA é o seu programa ODE, uma coleção de algoritmos para resolução de equações diferenciais ordinárias, construídos por E. L. Lafferty, J. P. Golden, R. A. Bogen e B. Kuipers que são descritas no manual de referência do MACSYMA. Esse pacote incorpora uma técnica em computação algébrica que faz a procura de modelos para um dado

problema e quando existe um, o computador gera a resposta por meio de um procedimento (algoritmo). Muitos matemáticos usam na verdade essa técnica, só que, às vezes, erram.

ODE tem, atualmente, capacidades para resolver uma larga classe de equações analiticamente; talvez uma boa maneira de medir as capacidades do software nessa área, é dizer que virtualmente, todos os métodos conhecidos para solução de equações diferenciais ordinárias de primeira e segunda ordem estão embutidos no programa, entretanto como sabemos, resolver equações diferenciais é uma espécie de arte científica. Dada uma equação diferencial cuja forma ou estrutura não sejam facilmente reconhecidas implica numa procura de transformações para conversão do problema dado em um outro já conhecido. ODE, na realidade, tem uma capacidade heurística modesta nessa área.

O desafio está feito, será que nos próximos anos uma incrível expressão reconhecedora será acrescentada ao sistema? Essa expressão virá a simplificar e transformar problemas numa maneira pelo menos próxima àquela que matemáticos especializados nessa área conheçam? Serão todas as regras conhecidas para que o controle do usuário seja razoavelmente rápido para a convergência para uma solução da equação, ou mesmo será fornecida para o investigador uma decisão que o problema não pode ser resolvido em termos de funções conhecidas?

Naturalmente muito ainda há para ser feito no campo das equações diferenciais nos sistemas de computação algébrica, e como o MACSYMA continua em desenvolvimento esperamos que capacidades como essas venham ao encontro do mundo científico muito brevemente.

MACSYMA possui um resolvedor simbólico de equações diferenciais e trabalha com alguns comandos entre os quais os principais são:

- ode → resolve equações diferenciais ordinárias de primeira e segunda ordem.
- ic1 → coloca uma condição inicial para problemas de valor inicial de primeira ordem.
- ic2 → coloca uma condição inicial para problemas de valor inicial de segunda ordem.
- bc2 → coloca uma condição de contorno para problemas de contorno de segunda ordem.

Ao resolver uma EDO, MACSYMA indica vários sistemas de variáveis que você pode usar para obter informações sobre a solução:

- method → fornece o método de solução usado.
- intfactor → fornece qualquer fator integrante usado.
- odeindex → fornece o índice para métodos de Bernoulli ou para o método homogêneo generalizado.
- yp → fornece a solução particular para a técnica de variação de parâmetros.

Sintaxe exemplo para resolução de algumas equações diferenciais

Usando o operador haspa podemos escrever equações diferenciais:

(C1) 'DIFF(Y,X) * X + 2 * Y = SIN(X)/X ; e [Enter] produz na saída seguinte,

(D1) $X\frac{dY}{dX} + 2Y = \frac{SIN(X)}{X}$

Nota : A função ODE do MACSYMA pode resolver algumas E.D.O. de primeira e segunda ordens.

No rótulo de entrada seguinte digitar;

(C2) ODE2(D1, Y, X) ; e [Enter], que fornece a solução procurada

(D2) $Y = \frac{\%C - COS(X)}{X^2}$

Nota : O símbolo % C que aparece no resultado acima representa uma constante arbitrária para soluções de equações diferenciais de 1^a ordem.

Resolveremos agora uma equação diferencial ordinária de 2^a ordem com valor inicial e perguntaremos ao sistema alguns detalhes da resolução:

(C3) 'diff(y,x,2) + 4 * y = x ^ 2 + 3 * exp(x);

(D3) $\frac{d^2Y}{dX^2} + 4Y = 3\%E^X + X^2$

(C4) sol: ode(%,y,x);

(D4) $Y = \%K_1 SIN(2X) + \%K_2 COS(2X) + \frac{24\%E^X + 10X^2 - 5}{40}$

(C5) method;

(D5) VARIATIONOFPARAMETERS

(C6) ic2(sol,x=0,y=0,'diff(y,x)=2);

(D6) $Y = \frac{7SIN(2X)}{10} - \frac{19COS(2X)}{40} + \frac{24\%E^X + 10X^2 - 5}{40}$

Apresentamos agora outro exemplo:

(C7) 1 + (1+ x * y) * exp(x*y) + (1 + x ^2 * exp(x * y)) * 'diff(y,x);

(D7) $(X^2\ \%E^{XY} + 1)\frac{dY}{dX} + (XY + 1)\%E^{XY} + 1$

(C8) ODE(%,y,x);

(D8) $X\%E^{XY} + Y + X = \%C$

(C9) method;

(D9) EXACT

(C10) intfactor;

(D10) 1

Podemos também encontrar soluções numéricas aproximadas.

(C11) EQ: 'DIFF(Y,T,2)=-(Y+E * Y^3)-SIN(2 * T);

(D11) $\frac{d^2Y}{dT^2} = -EY^3 - Y - SIN(2T)$

(C12) IC:['AT(Y,T=0)=A,'AT('DIFF(Y,T), T=0)=0];

```
        !              !
(D12) [Y!     = A, dY  !     = 0]
       !T=0          --- !
                     dT  !T=0
```

(C13) RUNGE_KUTTA(EQ,'Y,'T,IC,0,.3,.1), A=1, E=0.1;

```
(D13) [T = [0.0, 0.1, 0.2, 0.3],

Y = [1.0, 0.99417317, 0.9754558, 0.9421899],

dY
--- = [0.0, -0.11971783, -0.25740242, -0.4101274],
dT
 2
d Y
--- = [-1.1, -1.2911047, -1.4576902, -1.5904726]]
  2
dT
```

3.9 Programando no MACSYMA

Acabamos de usar o MACSYMA no modo interativo como um calculador, entretanto para computações que envolvam uma sequência repetitiva de comandos, como algumas que apresentamos nesse estudo, é melhor executarmos um programa. Não somente MACSYMA possui capacidades matemáticas poderosas assim como é uma linguagem de programação completa. MACSYMA aceita muitas estruturas de programação clássicas, incluindo sentenças condicionais e loops. Podemos escrever procedimentos para uma ampla variedade de propósitos como por exemplo, técnicas numéricas e pesquisa combinatória. MACSYMA e LISP - a linguagem na

qual MACSYMA foi escrito - são linguagens funcionais. Isto é, um programa no MACSYMA consiste de um número de funções, e possivelmente variáveis as quais foram criadas para desempenhar determinada tarefa. Em geral, elas são salvadas em um arquivo de disco e carregadas no computador com o comando LOADFILE. Uma outra forma é a que mostramos no exemplo PONTOCRIT a seguir onde o programa é escrito fora do MACSYMA. Apresentamos agora um programa modelo que calcula os pontos críticos de uma função f de duas variáveis x e y. O programa deixa que o usuário entre com uma função f, computa as derivadas parciais f_x e f_y e então usa o comando SOLVE do MACSYMA para obter soluções para $f_x = f_y = 0$. O programa aqui exemplificado foi escrito fora do MACSYMA, com um editor de textos, e então carregado no MACSYMA com o comando BATCH.

Aqui está o programa listado:

```
PONTOCRIT():=(
PRINT("PROGRAMA PARA ENCONTRAR PONTOS CRITICOS"),
F:READ("ENTER F(X,Y)"),
PRINT("F =".F),
EQS:[DIFF(F,X),(F,Y)],
UNK:[X,Y],
SOLVE(EQS,UNK))$
```

O programa (que é atualmente uma função sem argumento) é chamado PONTOCRIT. Cada linha é um comando válido MACSYMA, que poderia ser executado do teclado, o qual é separado do comando seguinte por uma vírgula. As derivadas parciais são armazenadas em uma variável chamada EQS e os valores desconhecidos são armazenados em UNK.

A seguir apresentamos um modelo que foi rodado:

```
(C1) BATCH(PONTOCRIT)$
[program is loaded in...]

(C4) PONTOCRIT();
```

PROGRAMA PARA ENCONTRAR PONTOS CRITICOS

ENTER F(X,Y)

% E^(X^3+Y^2)*(X+Y);

F = (y + x) % $e^{y^2+x^3}$

(D4) [[x = 0.4588955685487001 %i + 0.3589790871086935,

y = 0.4942017368275118 %i - 0.1225787367783657],

[x = 0.3589790871086935 - 0.4588955685487001 %i,

y = -0.4942017368275118 %i - 0.1225787367783657],

[x = 0.4187542327234816 %i - 0.6923124204420268,

y = 0.455912070111699 - 0.869726269281412 %i],

[x = -0.4187542327234816 %i - 0.6923124204420268,

y = 0.869726269281412 %i + 0.455912070111699]]

3.10 Formatos de Entrada e Saída

3.10.1 Geração FORTRAN

Sugerimos novamente o uso do manual para um maior aprofundamento nessa habilidade do software, e apresentamos a seguir um exemplo para gerar expressões FORTRAN no MACSYMA.

(C1) FORTRAN(EXPAND((2 * X + Y + 3 * Z)^ 9)); com [Enter] produz,

19683*Z**9+59049*Y*Z**8+118098*X*Z**8+78732*Y**2*Z**7+314928*X*Y*Z**7+314928*X**2*Z**7+61236*Y**3*Z**6+367416*X*Y**2*Z**6+734832*X**2*Z**6+489888*X**3*Z**6+30618*Y**4*Z**5+244944*X*Y**3*Z**5+734832*X**2*

$Y**2*Z**5+979776*X**3*Y*Z**5+489888*X**4*Z**5+10206*Y**5*Z**4+102060*X*$
$Y**4*Z**4+408240*X**2*Y**3*Z**4+816480*X**3*Y**2*Z**4+816480*X**4*Y*Z*$
$*4+326592*X**5*Z**4+2268*Y**6*Z**3+27216*X*Y**5*Z**3+136080*X**2*Y**4*Z*$
$*3+362880*X**3*Y**3*Z**3+544320*X**4*Y**2*Z**3+435456*X**5*Y*Z**3+145152*$
$X**6*Z**3+324*Y**7*Z**2+4536*X*Y**6*Z**2+27216*X**2*Y**5*Z**2+90720*X*$
$*3*Y**4*Z**2+181440*X**4*Y**3*Z**2+217728*X**5*Y**2*Z**2+145152*X**6*Y*Z*$
$*2+41472*X**7*Z**2+27*Y**8*Z+432*X*Y**7*Z+3024*X**2*Y**6*Z+12096*X**3*$
$**5*Z+30240*X**4*Y**4*Z+48384*X**5*Y**3*Z+48384*X**6*Y**2*Z+27648*X**7*$
$Y*Z+6912*X**8*Z+Y**9+18*X*Y**8+144*X**2*Y**7+672*X**3*Y**6+2016*X**4*$
$Y**5+4032*X**5*Y**4+5376*X**6*Y**3+4608*X**7*Y**2+2304*X**8*Y+512*X**9$
DONE

Nota: Exemplos de saídas em C e TEX são também encontradas no MACSYMA User's Guide.

3.11 Lista Parcial de Funções do MACSYMA

Apresentamos a seguir uma lista de funções fundamentais do MACSYMA, em ordem alfabética, que certamente deve ser complementada no Manual de Referência do MACSYMA para problemas mais específicos. Informações ON LINE sobre qualquer função estão disponíveis através do comando DESCRIBE (nome da função).

ALLROOTS(A) acha todas as raízes (geralmente complexas) da equação polinomial A, e as lista então no formato numérico (para 9 figuras significantes).

APPEND(A,B) junta a lista B a lista A resultando em uma única lista simples.

BATCH(A) carrega e roda um programa BATCH com nome de arquivo A.

COEFF(A,B,C) fornece o coeficiente de B elevado ao expoente C numa expressão A. C pode ser omitido se for a unidade.

CONCAT(A,B) cria o símbolo AB.

CONS(A,B) acrescenta A à lista B como seu primeiro elemento.

DEMOIVRE(A) transforma todas as exponenciais complexas em A em seus equivalentes trigonométricos.

DENOM(A) fornece o denominador de A.

DEPENDS(A,B) declara A como sendo uma função de B. Isso é útil para escrever derivadas não calculadas como em equações diferenciais especificadas.

DESOLVE(A,B) tenta resolver um sistema linear A de equações diferenciais ordinárias para o

valor desconhecido B usando transformadas de Laplace. Existem outros detalhes dessa função para os quais recomendamos a leitura do manual.

DETERMINANT(A) retorna o determinante de uma matriz quadrada A.

DIFF(A,B1,C1,B2,C2,....,Bn,Cn) fornece as derivadas parciais mistas de A com respeito a cada Bi, Ci vezes. Por comodidade, DIFF(A,B,1) pode ser escrito como DIFF(A,B). 'DIFF(....) representa a derivada não calculada, útil para especificar uma equação diferencial.

EIGENVALUES(A) retorna duas listas, a primeira sendo a dos autovalores da matriz quadrada A, a segunda sendo a de suas respectivas multiplicidades.

EINGENVECTORS(A) faz tudo o que EIGENVALUES faz e acrescenta uma lista dos autovetores de A.

ENTERMATRIX(A,B) deixa que o usuário entre com uma matriz AxB, elemento por elemento.

EV(A, B1, B2,...,Bn) calcula A sujeito às condições Bi. Em particular os Bi podem ser equações, listas de equações (tais como as retornadas pelo SOLVE) ou atribuições, e nesses casos EV conecta Bi em A. Os Bi's podem também ser palavras tais como NUMER (e neste caso o resultado é dado no formato numérico), DETOUT (e neste caso qualquer inversas de matrizes em A são calculadas com o determinante fatorado) ou DIFF (neste caso todas as diferenciações em A são calculadas, isto é, 'DIFF em A é substituído por DIFF). Para brevidade em um comando manual (isto é, não dentro de uma função definida pelo usuário) , o EV pode ser omitido, abreviando a sintaxe para A, B1, B2, ...,Bn.

EXPAND(A) expande algebricamente A. Em particular, a multiplicação é distribuída pela adição.

EXPONENTIALIZE(A) transforma todas as funções trigonométricas em A para suas equivalentes exponenciais complexas.

FACTOR(A) fatora A.

FREEOF(A, B) é verdadeira se a variável A não é parte da expressão B.

GRIND(A) fornece uma variável ou função A num formato compacto. Quando usado com WRITEFILE e um editor fora do MACSYMA, oferece um esquema para produzir arquivos BATCH que incluem expressões geradas pelo MACSYMA.

IDENT(A) retorna uma matriz identidade A x A.

IMAGPART(A) retorna a parte imaginária de A.

INTEGRATE(A, B) tenta encontrar a integral indefinida de A com respeito a B.

INTEGRATE(A, B, C, D) tenta encontrar a integral definida de A com respeito a B tomado de B = C a B = D. Os limites de integração C e D podem ser tomados como INF (infinito positivo) ou MINF (infinito negativo).

INVERT(A) computa a inversa de uma matriz quadrada A.

KILL(A) remove a variável A juntamente com todas as suas atribuições e propriedades do ambiente corrente do MACSYMA.

LIMIT(A, B, C) fornece o limite da expressão A quando a variável B se aproxima do valor C. O último pode ser tomado como INF ou MINF, como na integração.

LHS(A) dá o lado esquerdo da equação A.

LOADFILE(A) carrega um arquivo de disco como o nome de arqivo A do corrente diretório defaut.O arquivo de disco deve estar no formato próprio, isto é, criado por um comando SAVE.

MAKELIST(A, B, C, D) cria uma lista de A's (cada uma delas presumivelmente dependendo de B), concatenadas de B = C a B = D.

MAP(A, B) mapeia a função A dentro das subexpressões de B.

MATRIX(A1 ,A2 ,...,An) cria uma matriz consistindo das linhas Ai ,onde cada linha Ai é uma lista de m elementos ,[B1 ,B2 ,...,Bn]

NUM(A) dá o numerador de A.

ODE2(A, B, C) tenta resolver equações diferenciais ordinárias de primeira e segunda ordens A para B como uma função de C.

PART(A, B1, B2,..., Bn) primeiramente toma a B1-ésima parte de A ,então a B2-ésima parte dessa ,e assim por diante.

PLAYBACK(A) fornece último label A (um inteiro) e suas expressões associadas. Se A é omitido, todas as linhas são played back. Ver o manual para outras opções.

RATSIMP(A) simplifica A e retorna um quociente de 2 polinômios.

REALPART(A) retorna a parte real de A.

RHS(A) dá o lado direito da equação A.

SAVE(A, B1, B2, ..., Bn) cria um arquivo no disco com nome A no diretório defaut corrente, de variáveis, funções, ou arrays Bi. O formato do arquivo o permite de ser recarregado no MACSYMA usando o comando LOADFILE. Tudo (incluindo labels) pode ser salvo tomando-se B1 igual a ALL.

SOLVE(A, B) tenta resolver a equação algébrica A para o desconhecido B. Uma lista de soluções da equação é retornada. Brevemente, se A é uma equação da forma C = 0, ela pode ser abreviada simplesmente pela expressão C.

SOLVE([A1, A2, ..., An], [B1, B2, ..., Bn]) tenta resolver o sistema de n equações polinomiais Ai para os n desconhecidos Bi. Uma lista de soluções da equação é retornada.

STRINGOUT(A, B1, B2, ..., Bn) cria um arquivo no disco com nome A no diretório defaut corrente, de variáveis (labels). O arquivo fica num formato texto e não é recarregável no MAC-

SYMA. Entretanto, as expressões podem ser incorporadas dentro de um programa FORTRAN ou BASIC com um mínimo de edição.

SUBST(A, B, C) substitui A por B em C.

SUM(A, B, C, D) soma expressões A com B variando entre C e D.

TAYLOR(A, B, C, D) expande A em uma série de Taylor em B sobre B = C incluindo o termo (B - C)^D. MACSYMA também aceita expansões em mais do que uma variável independente. Ver manual para maiores detalhes.

TRANSPOSE(A) dá a transposta da matriz A.

TRIGEXPAND(A) é uma função trigonométrica de simplificação que usa as fórmulas de soma de ângulos para simplificar os argumentos de senos e cossenos individualmente. EX: TRIGEXPAND (Sin(X + Y)) fornece Cos(X) Sin(Y) + Sin(X) Cos(Y).

TRIGREDUCE(A) é uma função trigonométrica de simplificação que usa identidades para converter produtos e potências de seno e cosseno numa soma de termos, cada uma delas contendo somente um único seno ou cosseno. EX: TRIGEXPAND ($Sin(X)^2$) fornece (1 - Cos(2X))/2.

TRIGSIMP(A) é uma função trigonométrica de simplificação que troca TAN ,SEC ,etc... por seus equivalentes em Seno e Cosseno. Usa-se a identidade $SIN()^2 + COS()^2 = 1$.

3.12 Exercícios

1. Faça a plotagem em 3D da expressão z = cos x . cos y na região onde $-2 \leq x \leq 2$ e $-2 \leq y \leq 2$.
2. Mostre que $\int \frac{x}{1-x} dx = -x - log(1-x)$ considerando x < 1.
3. Calcule o limite da expressão $\frac{log(cos(x))}{log(1-sin(x))}$ quando x tende a $\pi/2$.
4. Calcule $\sum_{n=1}^{(q-5)/2} x^{4nl}.(-1)^n$.
5. Resolva a equação diferencial $(xy + x^2)\frac{dy}{dx} + y^2 + 3xy = 0$.
6. Calcular o determinante, a inversa, a transposta, os autovalores e os autovetores da matriz 4x4 definida pela lei de formação $a_{ij} = i^2 - j^2$.

3.13 Publicações Pertinentes

Apresentamos a seguir uma pequena lista de publicações sobre o MACSYMA que certamente contribuirão para o aprofundamento do investigador em várias facilidades do sistema não enfocadas ou tratadas superficialmente no nosso estudo, mas que de qualquer forma deve ser uma companhia frequente para o usuário do software.

1. COMPUTER AIDED MATHEMATICS GROUP , *Macsyma User's Guide* , Simbolics , U.S.A. , 1988.
2. DAVENPORT J.H., SIRET Y. e TOURNIER E., *Computer Algebra Systems and Algorithms for Algebraic Computation*, Academic Press, 1988.
3. PAVELLE R., *MACSYMA: Capabilities and Aplications to Problems in Engineering and the Sciences*, Symbolics Inc. Cambridge, MIT, 1980.
4. HELLER, *MACSYMA for Statisticians*, Wiley, 1988.

3.14 Informações Adicionais

Para informações adicionais e complementares tanto para instalação do sistema no equipamento disponível do usuário, problemas e bugs do software, procedimentos para sua aquisição e outros esclarecimentos, fornecemos a seguir um dos endereços de distribuição do MACSYMA; outros endereços são dados no capítulo 8 do nosso texto.

1. Symbolics, Inc., Eleven Cambridge Center, Cambridge, MA 02142, U.S.A.

2. Computer Aided Mathematics Group
   Symbolics, Inc.
   8 New England Executive Park , East
   Burlington , MA 01803 , U.S.A.

Capítulo 4

DERIVE

4.1 Introdução

DERIVE - A Mathematical Assistant, um dos mais recentes sistemas de computação algébrica; foi desenvolvido na Universidade do Hawaii por um grupo de pesquisadores liderados por David Stoutemyer, Albert Rich e Joan Rich, estando disponível a partir de 1988 para a comunidade acadêmica usuária, e nesse curto período já tendo sido objeto de dezenas de publicações em artigos, revistas e livros da área, em virtude do sucesso alcançado em todo o mundo.

DERIVE não tem a robustez e a abrangência de alguns de seus concorrentes, e é muitas vezes tratado por pesquisadores em computação algébrica como um "softwarezinho", idéia que não compactuamos de forma alguma, pois nesse sistema existe alguma interação extra, que ainda não está totalmente clara para nós, mas que "apaixona" os usuários, que torcem pelo sistema, quase que de modo fanático como se fosse por um time de futebol ou algo parecido; não fale que o DERIVE não faz isso ou aquilo perto de um desses "torcedores", pode ser perigoso.

DERIVE não foi o primeiro trabalho realizado pelos pesquisadores acima descritos, antes do DERIVE, de 1978 em diante eles tinham concebido um outro software de computação algébrica, o muMATH que também conseguiu atrair um grande número de usuários e pode ser considerado como o "pai" do DERIVE. No capítulo 9 do nosso estudo fornecemos mais alguns detalhes sobre o muMATH.

DERIVE roda nos Intel 8036, 8088, 80186, 80286 e nos computadores pessoais 80386. Apresentamos a seguir o hardware mínimo e o software (DOS) requerido para rodar o DERIVE. Correntemente existem duas versões para o DERIVE:

IBM PC compatível (XT, 286, 386, 486) rodando o software de sistema operacional da Microsoft Corporation, MS-DOS versão 2.1 ou posterior, Computador Pessoal Sistema/2 rodando o software de sistema operacional PC-DOS ou PS/2, com pelo menos 512 Kilobytes de memória,

com pelo menos um drive de 5 1/4 (360k) ou um de 3 1/2 (720k) e com um monitor compatível adaptado ao computador (MDA, CGA, EGA, VGA, Hercules etc.).

NEC PC-9801 ou compatível rodando o software de sistema operacional da Microsoft Corporation MS-DOS, versão 2.1 ou posterior, com pelo menos 512 Kilobytes de memória, com pelo menos um drive para disquete de 5 1/4 (640k) e com monitor monocromático ou colorido.

4.2 O Porquê da Utilidade do DERIVE

- DERIVE é o software mais usado em microcomputadores e certamente o mais indicado para estudantes e iniciantes em utilização de sistemas de computação algébrica.
- O modo pelo qual o DERIVE interage com o usuário é o mais tutorial entre todos os sistemas de computação algébrica existentes.
- A grande maioria das Universidades e Centros Científicos já utilizam o sistema como instrumento auxiliar de ensino.
- De todos os sistemas conhecidos o DERIVE é o de mais fácil instalação (apenas 1 disquete de 5 1/4 ou de 3 1/2) e o de domínio mais imediato pelo investigador para uso como ferramenta auxiliar, e, talvez essas sejam as maiores razões para o seu sucesso.
- DERIVE tem o menor preço entre os sistemas o que veio a transformá-lo no maior ponto de venda de softwares desse área.
- O investigador pode testar rapidamente e com pouco trabalho conjecturas matemáticas das quais duvide da validade bem como verificar que alguns problemas são insolúveis.
- Atualmente, a criação de softwares para várias áreas é precedida por sondagens e pesquisas entre seus reais futuros usuários, para que digam em que tipo de software gostariam de trabalhar, quais seriam suas características mais importantes, qual deveria ser o tipo de interação com o pacote etc., na opinião da maioria dos usuários de sistemas de computação algébrica, o DERIVE tem o perfil ideal e se encaixa perfeitamente nessa idéia.
- Em várias oportunidades ouvimos pesquisadores em sistemas de computação algébrica discordarem sobre o sistema "ideal"; teria que ter a abrangência deste, teria que ter a confiabilidade daquele, teria que ter o tratamento gráfico daquele outro, além de outras considerações; o ponto comum dessas discussões era que um sistema de "consenso" teria que ter o lay-out e a interação com o investigador que o DERIVE tem.

4.3 Capacidades e Usos do DERIVE

4.3.1 Capacidades do DERIVE

Apresentamos abaixo uma lista das mais significativas capacidades do DERIVE e sugerimos a leitura do manual para estudo detalhado de outras habilidades do sistema.

Derivação	Limites
Equações e Inequações	Manipulação Matricial
Expressões Algébricas	Operações com Complexos
Fatorização	Séries de Taylor
Funções Transcendentes	Simplificação
Gráficos em 2D e 3D	Sistemas de Equações
Integração	Somatórios

Para colocar as capacidades do DERIVE em perspectiva, podemos dizer que o sistema conhece uma larga porcentagem de técnicas matemáticas usadas em ciência e engenharia; obviamente pela pequena quantidade de memória exigida pelo software, seu tempo de processamento para o display de saída é mais lento quando comparado com a maioria dos softwares desse tipo.

4.3.2 Usos do DERIVE

DERIVE é bastante usado no meio acadêmico, principalmente na graduação de cursos que exijam base matemática por professores e alunos. Embora seus usos sejam variados, vamos listar alguns deles obtidos através de palestras, artigos e publicações específicas sobre o sistema.

Álgebra Linear e Abstrata	Geometria Algébrica
Análise Numérica	Matemática Aplicada e Computacional
Arquitetura	Matemática Experimental
Engenharia	Química
Física das Partículas	Teoria das E.D.O.'s
Física do Estado Sólido	Teoria dos Números
Mecânica Celeste	Teoria da Otimização
Mecânica Estrutural	Termodinâmica

4.4 Apresentação do DERIVE

4.4.1 Preliminares

A finalidade desse parágrafo é apresentar uma demonstração sobre a utilização do DERIVE e tem também a intenção de ser intelegível para o leitor que não está familiarizado com o sistema. No final do parágrafo apresentamos também uma lista de outras fontes para um maior aprofundamento no software.

Sugerimos que a leitura dessa apresentação seja feita em frente a um computador que rode o DERIVE, de modo que o usuário possa executar nossas sugestões como ponto de partida, fazendo inclusive alguns exercícios elementares ao final do parágrafo, para posteriormente poder trabalhar em seus próprios objetivos.

Para entrar no DERIVE na maioria dos computadores, seguir a seguinte orientação:

Digitar DERIVE ao lado do prompt A, B, C etc. do sistema operacional e pressionar a tecla [Enter]. Após alguns segundos aparecerá a 1ª tela com a tela e menu principais do DERIVE como apresentado abaixo.

```
                              DERIVE
                      A Mathematical Assistant

                              Version x.xx


            Copyright (C) 1988 and 1990 by Soft Warehouse, Inc.
                            Honolulu, Hawaii, USA



                            Press H for help

------------------------------------------------------------------------------
COMMAND: Author Build Calculus Declare Expand Factor Help Jump soLve Manage
         Options Plot Quit Remove Simplify Transfer moVe Window approX
Enter option
                                              Free: 100%  Insert   Derive Algebra
```

Para se movimentar nas opções desse menu pressionar a tecla TAB ou a barra de espaços. Para selecionar o ítem que tivermos destacado pressionar [Enter]. Um outro modo de selecionar um ítem é pressionar a letra que estiver maiúscula na opção; por exemplo para selecionar o ítem "soLve" do menu basta pressionar a letra L.

A opção Author é selecionada por default pelo software. Teclar [Enter] e o programa apresenta:

AUTHOR expression:

Aguardando sua expressão de entrada.

Como exemplo vamos mostrar passo a passo uma das habilidades do sistema para o cálculo de uma expansão em série de Taylor de e^x.

Primeiramente digitar x e [Enter], aparecerá então:

1 : x

Escolha no menu principal a opção Build e tecle [Enter]. O sistema apresentará agora:

BUILD first expression : # 1

Tecle [Enter], "caminhe" até a opção EXP

Tecle [Enter] e aparecerá a linha seguinte como saída na tela.

2 : EXP (x)

Novamente no menu principal selecionar a opção Calculus e teclar [Enter], surgirá então as opções de cálculo realizadas pelo DERIVE.

CALCULUS : Differentiate Integrate Limit Product Sum Taylor

Selecionar Taylor, teclar [Enter] e será mostrado agora.

TAYLOR expression: # 2

Teclar [Enter] e informar ao sistema qual é a variável do seu problema, no nosso caso é x, teclar então [Enter].

Devemos em seguida informar qo sistema qual o grau da derivada que se quer e em que ponto a série de Taylor será expandida. Vamos escolher nesse exemplo;

TAYLOR: Degree: 5 Point: 1

Para se chegar a esses valores digitar 5 (default), teclar [TAB], ou $\leftrightarrows$ e então digitar 1 no ponto e [Enter], aparecerá então a linha seguinte;

3 : TAYLOR (EXP(x), x, 1, 5)

Selecionar agora a opção Expand para calcular efetivamente o que foi pedido na linha 3, e surgirá então na linha seguinte, linha 4, a resposta pedida.

Na última linha teríamos então:

$$4: \quad \frac{\hat{e}\, x^5}{120} + \frac{\hat{e}\, x^3}{12} + \frac{\hat{e}\, x^2}{6} + \frac{3.\hat{e}\, x}{8} + \frac{11\, \hat{e}}{30}$$

Apresentamos a seguir a tela cumulativa numerada do DERIVE para esse desenvolvimento do nosso problema exemplo qua sai ao pedirmos a impressão:

```
1:  x
2:  EXP(x)
3:  TAYLOR(EXP(x), x, 1, 5)

          5          3          2
       #e x       #e x       #e x       3 #e x       11 #e
4:   ------- + ------- + ------- + --------- + --------
       120        12         6          8          30
```

4.4.2 Símbolos e Chaves Especiais

- Para terminarmos uma seção do DERIVE, selecionar a opção Quit do menu e pressionar Y para abandonar as expressões digitadas. Se o monitor estiver com baixa luminosidade, pode não ser perceptível por parte do investigador que o software aguarda a confirmação de um Y para terminar.

- Quando a expressão para a qual desejamos um cálculo estiver destacada na tela, selecionar a opção conveniente do menu principal. Por exemplo, o cálculo de limites, derivadas, integrais etc., é feito através da opção Expand.

- Para desistirmos de alguma seleção feita errada acidentalmente pressionar a tecla Esc.

- Para tomarmos conhecimento de outras funções do DERIVE selecionar primeiramente a expressão Help do menu principal e posteriormente a expressão Functions; uma outra maneira de fazer esse procedimento seria digitar H e depois F a partir do menu principal e aí aparecerá a 1ª tela com as opções Next, para mudar para a próxima tela do Help; Previous para voltar a tela anterior e Resume para sair do Help.

- As quantidades padrão pi (3.14159...), i (raiz quadrada de -1), e (base natural de log. / 2,718...) e mais infinito, são respectivamente referidas por pi, #i, #e e inf.

- Algumas letras gregas podem ser geradas pressionando-se simultaneamente a tecla Alt e alguma letra, por exemplo, Alt+A = alfa, Alt+P = pi, Alt+S = Sigma etc., para outras letras consultar o manual do software.

4.5 Entrada das Expressões

Após a seleção da opção Author do menu principal e a digitação sintaticamente correta por parte do investigador de alguma expressão, DERIVE a apresenta na tela usando o formato usual de matemática em vídeo reverso. A sintaxe usada pelo sistema é parecida com a da linguagem BASIC. As expressões digitadas vão recebendo numeração de acordo com a ordem de entrada e ficam guardadas na memória.

Para nos referirmos a uma expressão já digitada, usar # antes do número de ordem da expressão. Por exemplo, # 7 é a forma de se referir à sétima expressão digitada. É permitido também pelo software "caminhar" nas expressões já digitadas usando as teclas de setas. Uma outra maneira de se deslocar rapidamente para uma expressão já digitada anteriormente é feita através do comando Jump, que pode ser utilizado digitando-se apenas a letra J no teclado. O software vai então perguntar para que expressão o investigador deseja ir - o que é feito digitando-se ao lado do Jump o número da expressão desejada - para então operarmos os cálculos naquela expressão ou em expressões próximas aquela.

As operações aritméticas comuns do DERIVE são:

+ Adição

- Subtração

$\star$ multiplicação escalar (pode ser omitido)

$\hat{}$ potenciação (acento circunflexo gerado pelas teclas [Shift] e 6)

SQRT(x) raiz quadrada de x

x! fatorial de x (ou GAMMA (x) se x $\notin$ N)

As principais funções matemáticas do DERIVE são:

EXP(x) ou # ex	exponencial de base e
LOG(x) ou LN(x)	logaritmo natural de x
LOG(x , a)	logaritmo de x na base a
SIN(x)	seno de x
COS(x)	cosseno de x
TAN(x)	tangente de x
SEC(x)	secante de x
COT(x)	cotangente de x
CSC(x)	cossecante de x
ASIN(x)	arco-seno de x
ACOS(x)	arco-cosseno de x
ATAN(x)	arco-tangente de x
SINH(x)	seno hiperbólico de x
COSH(x)	cosseno hiperbólico de x
TANH(x)	tangente hiperbólica de x
ASINH(x)	arco-seno hiperbólico de x
ACOSH(x)	arco-cosseno hiperbólico de x
ATANH(x)	arco-tangente hiperbólico de x
ABS(x)	valor absoluto de x
SIGN(x)	sinal de x
MAX(x, y, ...)	máximo de x, y, ...
MIN(x, y, ...)	mínimo de x, y, ...
GAMMA(x)	gama de x
PERM (m , p)	arranjo de m elementos p a p
COMB (m , p)	combinação de m elementos p a p

4.6 Sintaxe de Algumas Capacidades do DERIVE

Mostraremos a seguir a sintaxe correta de algumas das capacidades do sistema, obviamente nos permitimos escolher aleatoriamente essas habilidades e sugerimos fortemente a utilização do manual para determinação de outras funções, construções de gráficos etc.

4.6.1 Limites

Sintaxe: LIM(u, v, p, l)

onde

u é a expressão que define a função

v é a variável

p é o ponto para o qual v tenderá

l é um número positivo, se o limite for lateral à direita, negativo se o limite for lateral à esquerda. Observe que l pode ser omitido.

Selecione a opção Author do menu, digite a expressão desejada e finalmente não se esqueça de selecionar Expand para que o cálculo indicado seja executado.

Exemplos :

Digitar na opção AUTHOR:

LIM(SIN(3x)/(x ^ 2 + 2x), x, 0, -1) $\longrightarrow$ limite de $\frac{sen(3x)}{x^2+2x}$ quando x tende a 0 pela esquerda.

LIM(sqrt(x ^ 2 + x) + x, x, -inf) $\longrightarrow = -\frac{1}{2}$

4.6.2 Derivadas

Sintaxe : DIF(u, v, n)

onde

u é a expressão que define a função

v é a variável com relação a qual se quer derivar

n é a ordem da derivada

Se a função tiver mais do que uma variável então deverá ser solicitada uma ordenação das variáveis, ou seja, DERIVE pergunta qual a primeira, segunda, terceira, etc. ... variável. Quando não necessário pressionar [Enter] para dispensar essa ordenação. Observe também que o parâmetro n pode ser omitido.

Exemplos :

DIF(ATAN(x ^ 2/(3x+1)), x , 2) $\longrightarrow$ derivada de segunda ordem de arctan$\frac{x^2}{3x+1}$ com relação a x.

DIF(Ln(x ^ 3 + 5 * Sin(6x), x) $\longrightarrow = \frac{3(10\,cos(6x)+x^2}{5\,sin(6x)+x^3}$

Usar a expressão Expand para verificar os resultados. Se uma expressão de número n já estiver na tela, então sua derivada (entre outras possíveis operações), pode ser feita expandindo uma expressão da forma DIF(#n, ...). Por exemplo DIF(#7 , x) dá a derivada da expressão 7 com relação a x.

4.6.3 Integrais

Sintaxe : INT(u, x, a, b)

onde

u é a expressão que define a função

x é a variável de integração

a é o limite inferior do intervalo de integração

b é o limite superior do intervalo de integração

Observe que a e b podem ser omitidos. É possível também o cálculo de integrais impróprias e integrais múltiplas iteradas.

Exemplos :

INT(u ^ 2 * sin(u), u, 0, pi) $\longrightarrow$ integral da função $u^2\, sen(u)$ no intervalo $[0\ ,\ \pi]$

INT(INT(x ^ 3 * Exp(x) * y, x, 0, 2y), y, 0, 1) $\longrightarrow = 2\hat{e} - 12$

Sempre é bom lembrar, usar a opção Expand para a obtenção dos resultados.

4.6.4 Desenvolvimento em Séries de Taylor

Sintaxe : TAYLOR(u, x, a, n) $\longrightarrow$ desenvolvimento de Taylor de ordem n da função definida pela expressão u, em torno do ponto x = a.

Exemplos :

TAYLOR(EXP(x), x, 1, 5) $\longrightarrow$ Exemplo dado no início do parágrafo

TAYLOR(Ln(x ^ 2 + x), x, 1, 4) $\longrightarrow = ln(2) - \frac{17x^4}{64} + \frac{23x^3}{16} - \frac{107x^2}{32} + \frac{79x}{16} - \frac{177}{64}$

4.6.5 Somatórios e Produtórios

Sintaxe : SUM(u, n, k, m) $\longrightarrow$ Somatório de u com n variando de k até m

Sintaxe : PRODUCT(u, n, k, n) $\longrightarrow$ produto de u com n variando de k a m

Observe que tanto k quanto m podem ser ilimitados (-inf ou +inf)

Exemplos :

SUM(3 / (10 ^ n), n, 1, inf) $\longrightarrow = \frac{1}{3}$

PRODUCT((-1) ^(n + 1) * 3 / 2 ^ n, n, 0, 5) $\longrightarrow -\frac{729}{32768}$

4.6.6 Vetores e Matrizes

Os vetores devem entrar no formato usual, entre colchetes com todas as coordenadas separadas por vírgulas; por exemplo, [4, -SQRT(12), j]. Uma matriz deve entrar como um "vetor de vetores", deste modo a entrada correta da matriz identidade 3x3 será; [[1, 0, 0], [0, 1, 0], [0, 0, 1]]. O default das matrizes é 3x3 para alterá-lo basta usar a tecla Tab para o número de colunas.

Uma maneira conveniente de trabalhar com vetores é declará-los com a opção Author antes de usá-los, assim eles ficaram disponíveis até o encerramento da seção do DERIVE; neste caso usar o sinal de atribuição (:=) na declaração, por exemplo, v := [1, 0, 0]. O procedimento para matrizes é análogo.

As operações permitidas com vetores são : soma, produto por escalar, produto interno - todas com notação usual - e o produto vetorial (CROSS). As operações com matrizes são: soma, produto, produto por escalar, determinantes, inversas e potências - todas com notação usual - e C' para a obtenção da matriz transposta de C. Note que a transposta é indicada com um acento grave, que não deve ser confundido com o apóstrofo.

Exemplos :

Use a opção Author para declarar o seguinte:

v := [2, c, -SQRT(80)]	Pressione [Enter] e A
w := [-9, 3, d]	Pressione [Enter] e A\
M := [[3, -7], [7, 45]]	Pressione [Enter] e A
N := [[a, h+4], [-5, 6p]]	Pressione [Enter] e A

Agora usando as opções AUTHOR e EXPAND é possível fazer várias operações tais como:

v + w	soma dos vetores v e w
(3v) . w	produto interno de 3v com w
CROSS (v, w)	produto vetorial v x w
M + 4N	matriz M mais o produto do escalar 4 por N
N.M	produto de N por M
M^-1	matriz inversa de M
DET(M)	determinante da matriz M
(N')^2	o quadrado da matriz transposta de N

Note que é possível usar a opção AUTHOR para definir a expressão que define o produto misto dos vetores e, f e g ;

PROD_ MISTO (e, f, g) := e . CROSS (f, g)

Agora se entrarmos com a expressão PROD_ MISTO(v, w, [1, 2, -8]) através da opção Author e posteriormente a opção Expand chegaríamos ao resultado desejado.

4.6.7 Polinômio Característico e Autovalores de Matrizes

Sintaxe : CHARPOLY (M, v) $\longrightarrow$ polinômio característico de M na variável v

Sintaxe : EIGENVALUES (N) $\longrightarrow$ conjunto de autovalores da matriz N

Exemplo :

Vamos apresentar passo a passo um exemplo trivial de cálculo do polinômio característico e dos autovalores de uma matriz no DERIVE;

Entrar primeiramente na opção AUTHOR e pressionar [Enter].

Em AUTHOR digitar, M := [[5, -7], [4h, -3z]] , teclar [Enter] e a matriz aparecerá na tela em [1];

Entrar no AUTHOR e pressionar [Enter], novamente.

Em AUTHOR digitar, CHARPOLY (M , x) e teclar [Enter] e essa linha aparecerá na tela em [2];

Caminhar com a barra de espaços até EXPAND e pressionar [Enter], o software perguntará se é para expandir a expressão [2]; pressionar [Enter] e o polinômio característico aparecerá na tela em [3];
3: $28h + x^2 + 3xz - 5x - 15z$

Em AUTHOR, pressionar [Enter], digitar a expressão EIGENVALUES (M), pressionar [Enter] e essa linha aparecerá na tela em [4];

Ir até o EXPAND, de novo o sistema pergunta se é para expandir a expressão [4]; pressionar [Enter] e os autovalores serão mostrados em [5];

5: $[w = -\frac{\sqrt{-112h+9z^2+30z+25}}{2} - \frac{3z}{2} + \frac{5}{2}, w = \frac{\sqrt{-112h+9z^2+30z+25}}{2} - \frac{3z}{2} + \frac{5}{2}]$

Nota: A cada cálculo o DERIVE fornece o tempo de computação; a título de ilustração, num PC-386 o último cálculo acima foi feitos em 1.2 segundos.

Nota: É bastante interessante que o investigador conheça os arquivos modelo de apresentação e mostra do sistema; MATRIX.MTH é um deles. Para carregar esse arquivo escolher no menu principal TRANSFER e dentro desse a opção DEMO, onde se digita MA-

TRIX.MTH e várias capacidades são apresentadas pelo software. Para sair teclar ESC. Para maiores detalhes ver a seção deste capítulo intitulada ARQUIVOS.MTH.
Para imprimir qualquer tela do DERIVE teclar PrintScreen; note ainda nessa saída num formulário de 80 colunas que quando um resultado fornecido pelo software não cabe na largura da página, ele continua na linha seguinte com sinais superpostos.

4.6.8 Expressões Algébricas

As expressões algébricas têm a sintaxe usual; são possíveis diversos tipos de fatoração e simplificação, por exemplo; entrar na opção Author, digitar (x + 2y + 5)^4 e expandir essa expressão em relação a x e posteriormente em relação a y, é feito de maneira bem tutorial pelo software do seguinte modo:

Pressione E X [Enter] para expandir em relação a x

Pressione E Y [Enter] para expandir em relação a y

É sempre necessário responder qual o número da expressão a ser expandida, para fazer isso, ver na tela qual é o número da expressão desejada e digitar # seguido daquele número, quando o DERIVE perguntar qual o número da expressão.

Vejamos mais algumas possibilidades de se operar com expressões embora não tenhamos nesse parágrafo esgotado o que pode ser feito pelo sistema em expressões algébricas.

A expressão (x ^ 2 + 2xy + y ^ 2)/(x ^ 2 - y ^ 2) é simplificada usando-se a opção Simplify do menu.

A expressão x ^ 4 - 3x ^ 3 + 3x ^ 2 - 3x + 2 é fatorada usando-se a opção Factor do menu. Observe que Factor tem um sub-menu formado pelas opções Trivial, Squarefree, Rational, raDicals e Complex, usar cada uma dessas opções comparativamente é a melhor receita para entendê-las.

É interessante também carregar o arquivo ALGEBRA.MTH - digitar T L ALGEBRA [Enter] Y - usando as setas para selecionar em vídeo reverso alguma expressão a ser calculada, pressionando-se S (para simplificar) ou E (para expandir). Faça isso.

4.6.9 Equações, Inequações e Sistemas

As equações e inequações também tem sua entrada no formato usual onde, /= é usado como sinal de diferente, <= como menor ou igual e >= como maior ou igual.

Os sistemas entram de modo análogo aos vetores; as equações separadas entre si por vírgulas e

os colchetes envolvendo todas as equações. Por exemplo, o sistema;

$$\begin{cases} 3x + 2y = 32 \\ 2x - 8y = 2 \end{cases}$$

tem sua entrada correta escrita na forma, [3x + 2y = 32 , 2x - 8y = 2], e uma vez que o sistema a resolver esteja na tela, basta usar a opção soLve para a obtenção da solução. Os sistemas indeterminados terão os símbolos @1, @2, ... na solução identificando as variáveis livres. Por exemplo, a solução do sistema [x + y = 1 , 2x + 2y = 2] será [x = @1 , y = 1 - @1] ou seja, y = 1 - x.

4.6.10 Números Complexos

É possível a simplificação de expressões que envolvam números complexos, bem como o cálculo de funções com argumentos complexos. A unidade imaginária deverá entrar como #i ou através das teclas [Alt] + i. Na tela, o i aparecerá com um acento circunflexo.

Exemplos :

Note que (1 + i)^22 é tratado como um polinômio de grau 22 na variável i.

(1 - #i)^22 $\longrightarrow$ 2048i

EXP (2 + 5 #i) $\longrightarrow$ e^2 cos(5) + e^2 i sin(5)

4.6.11 Gradiente, Divergente, Rotacional e Laplaciano

Sugerimos novamente a leitura do manual ou uma das fontes complementares sobre o DERIVE, listadas no final do parágrafo, para um melhor entendimento sobre as habilidade acima descritas, entretanto apresentamos a sintaxe básica necessária para trabalharmos com esses cálculos matemáticos.

Sintaxe : GRAD (expressão) $\longrightarrow$ gradiente

Sintaxe : DIV (vetor) $\longrightarrow$ divergente

Sintaxe : CURL (vetor) $\longrightarrow$ rotacional

Sintaxe : LAPLACIAN (expressão) $\longrightarrow$ laplaciano

Exemplos :

GRAD (SIN(xy) - LN(3x + z)) $\longrightarrow$ = [y cos(xy) - $\frac{3}{3x+z}$, x cos(xy), - $\frac{1}{3x+z}$]

DIV ([x/y , y^2 + cos(z) , x*z^4]) $\longrightarrow = 4xz^3 + 2y + \frac{1}{y}$.

CURL ([SINH(x) + y + z , x^2 + y , z] $\longrightarrow$ = [0 , 1 , 2x - 1].

LAPLACIAN (LN(x^2/y)) $\longrightarrow = \frac{1}{y^2} - \frac{2}{x^2}$.

4.7 Tratamento Gráfico

O tratamento gráfico dado pelo DERIVE pode ser avaliado de duas maneiras: Uma pelo lado positivo, que é a de se elogiar a programação do sistema, que com apenas 1 disquete de 5 1/4 (360k) ou com um disquete de 3 1/2, faz plotagens em 2D e 3D em microcomputadores, muitas vezes mais rapidamente do que alguns dos "grandes" softwares da área, dependendo do equipamento no qual está rodando. De outro lado, fazendo essa avaliação de um modo mais frio, não podemos deixar de observar e citar que a plotagem de algumas equações - como por exemplo, z = sin (x) * sin (y) em 3D, deixa a desejar pois mais parecem planos, onde deveriam existir claramente curvaturas. Após ler essa apresentação sobre gráficos, faça a plotagem da equação acima, se possível também no GAUSS, no MACSYMA e no MATHEMATICA, e tire suas próprias conclusões.

Vamos agora passar aos procedimentos para a construção de gráficos pelo sistema.

Nota: Caso seu monitor seja colorido, VGA, Super VGA, etc. consulte o manual do sistema para alterações nas cores das malhas, dos eixos, do sombreado e outras possibilidades que o DERIVE oferece.

O modo de entrada para a construções de gráficos no DERIVE é bastante simples, normalmente a opção Author é usada para a entrada da função a ser plotada. Por exemplo, se quisermos construir o gráfico de f(x) = x^3 cos(x) em 2D, podemos entrar de vários modos equivalentes no sistema para esse tipo de construção;

y = x^3 * cos(x) , w = v^3 cos(v) ou x^3 cos(x)

Quando a função já tiver sido definida, é necessário colocar a tela no modo gráfico, o que é feito através do comando Options. No sub-menu de Options deverá ser escolhido a opção Display, que por sua vez apresenta três opções; Mode, Resolution e Adapter. Dentro de cada sub-menu usar a barra de espaços para se mover e caso seja necessário mudar de sub-menu usar a tecla Tab. Escolher em Mode a opção Graphics, em Resolution a opção High e em Adapter escollher CGA, VGA etc. Lembramos novamente uma maneira rápida de se fazer isso, basta pressionar as teclas O (Options), D (Display), G , H, C e [Enter]. Estando então definidas, a função e a tela no modo gráfico usar a opção Plot para ter acesso ao menu abaixo.

COMMAND : Algebra Center Delete Help Move Options Plot Quit Scale Ticks Window Zoom

Usar Plot para dar início à construção do gráfico e usar Algebra para entrar com novas equações.

Usar a opção Scale para alterar a escala do gráfico, utilizando a tecla Tab.

Nota: Facilmente podemos alterar a rede utilizada na construção do gráfico proposto com a opção Grids, assim como podemos efetuar uma translação do gráfico através da opção Center. No exemplo de plotagem em 3D que mostraremos a seguir, observe que o centro se encontra em x = 0 e y = 0.

Cada vez que se for entrar com uma nova equação para construir um gráfico, serão construídos todos os gráficos anteriores, para que só seja construído o gráfico da última equação fornecida devem ser pressionadas as teclas D e B (de Delete Butlast).

Gráficos em equações paramétricas podem ser construídos por um processo análogo, embora a equação tenha que ter sua entrada na forma de um vetor. Por exemplo, [cos(2a), cos(a)].

Gráficos em coordenadas polares são feitos usando-se a opção, Options Type Polar e uma vez feito esse ajuste, as equações entram de modo semelhante ao das funções y = f(x). Os gráficos de funções z = f (x, y) também são construídos de forma semelhante.

Os arquivos Plot2D , Plot3D e Plotpara contém diversas equações que podem ser usadas como exemplos na construção de gráficos.

A opção Options Display permite que a tela volte a ser colocada no modo texto.

Vamos mostrar passo a passo um exemplo de plotagem em 3D no DERIVE;

Entrar primeiramente na opção Author e pressionar [Enter]

Em AUTHOR expression digitar z = 1 / (x^(2) + y^(2) + 9), pressionar [Enter] e essa equação aparecerá na tela, no formato usual de matemática em [1];

Entrar agora na opção Options - ganhe tempo, é só teclar O -, que coloca a tela no modo gráfico e aparecerá então;

OPTIONS: Color Display Execute Input Mute Notation Precision Radix

Dentro do sub-menu do Display existem três opções; Mode, Resolution e Adapter

Escolher em Mode a opção Graphics, em Resolution a opção High e em Adapter a que corresponder a seu monitor - CGA, EGA, VGA, Hércules etc. - e pressionar [Enter].

No menu principal novamente ir para o Plot e pressionar [Enter], no sub-menu ir para o Plot e pressionar [Enter] de novo - não se esqueça, ganhe tempo é só digitar P duas vezes seguidas - e a plotagem é feita na tela.

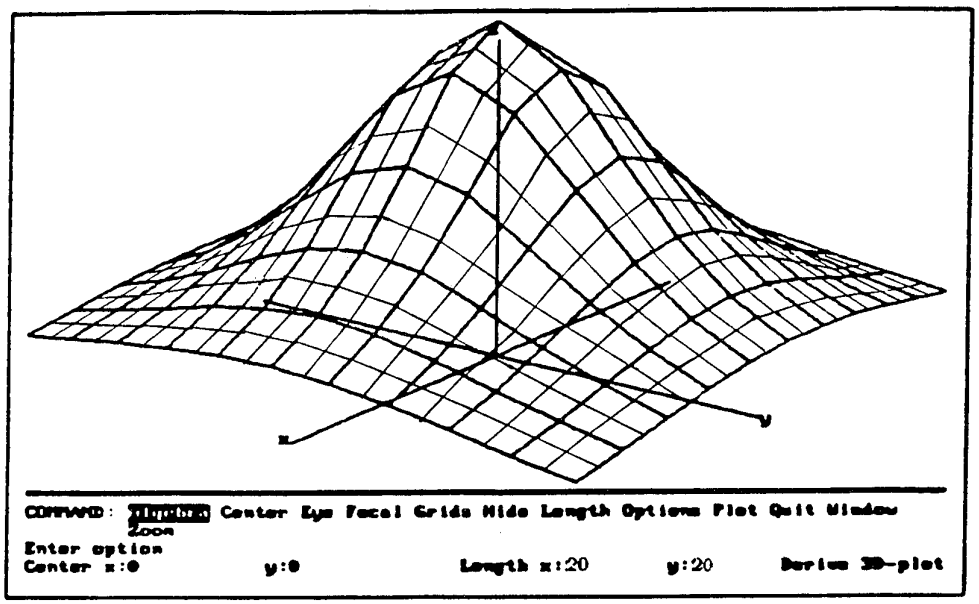


FIGURA 1

Use a imaginação, construa outros gráficos, tente conhecer mais e vasculhe o interior de todos os sub-menus apresentados pelo sistema, ainda há muito o que ser visto e explorado.

4.8 Os Arquivos * . MTH

O disquete do sistema de computação algébrica DERIVE contém 17 arquivos com extensão . MTH , por exemplo, Coord . MTH, Integral . MTH, Deriv . MTH, Plot3D . MTH etc. que por sua vez contém diversas fórmulas e exemplos interessantes, o investigador pode aumentar seus conhecimentos sobre o software lendo e testando esses arquivos, e também tirando dúvidas com as instruções na tela do Help. Todos os arquivos . MTH, inclusive o Help - DERIVE . HLP - podem ser impressos bastando para isso ligar a impressora e digitar;

A> COPY Nome-do-arquivo . extensão PRN : e pressionar [Enter].

É possível ainda carregar os arquivos . MTH através da opção Transfer do menu, que contém um sub-menu com as opções;

TRANSFER : Merge Clear Demo Load Save Print sTate

Para caminhar no Transfer, usar Tab ou a barra de espaços até a opção desejada e pressionar a letra maiúscula da opção ou [Enter].

A opção Merge permite que se mescle expressões do investigador com expressões de um arquivo . MTH do disco cujo nome será solicitado.

A opção Load carrega um arquivo . MTH do disco retirando da memória todas as expressões do investigador digitadas antes do Load.

A opção Save permite que se crie um arquivo próprio de fórmulas e funções prediletas, onde deverá ser fornecido o nome do arquivo a ser criado informando o que será usado no arquivo, se for o próprio DERIVE, deverá ser criado um arquivo . MTH, se for o BASIC, deverá ser criado um arquivo . BAS, se for o Pascal, deverá ser criado um arquivo . PAS e se for o FORTRAN, deverá ser criado um arquivo . FOR, e cada arquivo que venha a ser criado deverá usar a sintaxe daquela respectiva linguagem.

Vamos usar a título de demonstração, Transfer seguido de Merge - basta pressionar T e M - para carregar do disco o arquivo Coord . MTH, - não é necessário digitar o . MTH -, que habilita o DERIVE a calcular jacobianos de transformações. Quando todas as expressões do Coord . MTH tiverem aparecido na tela, usar a opção Author para entrar com a expressão;

JACOBIAN ([w^4 , w * h] , [w , h]) $\longrightarrow = \begin{pmatrix} 4w^3 & 0 \\ h & w \end{pmatrix}$

que como sempre, para ser calculado necessitou do uso da opção Expand.

Nota: sem o arquivo - Coord . MTH - não poderíamos calcular esse jacobiano, pois a fórmula que possibilita sua determinação se encontra naquele arquivo.

Existe uma maneira de carregar o arquivo . MTH e verificar a execução automática das operações sendo feitas à medida que se pressiona uma tecla, basta usar a opção Demo do sub-menu do Transfer.

Sugerimos a seguir que o investigador carregue - com, Transfer Merge, Transfer Load ou Transfer Demo - outros arquivos . MTH para conhecer outras fórmulas e funções neles contidas.

O usuário pode ainda acrescentar linhas de comentários às suas fórmulas, usando a opção Author com o seu texto digitado entre aspas.

Nota: Obviamente deixamos de apresentar várias habilidades do sistema, sugerimos fortemente o uso do manual do software, pois o DERIVE tem muito a oferecer como auxílio à pesquisa e ao ensino a comunidade acadêmica de um modo geral. Ao terminarmos esse trabalho várias novas capacidades eram introduzidas nas novas versões, o que mostra que o nome "softwarezinho" realmente não deve voltar a ser aplicado ao programa.

4.9 Exercícios

1. Desenvolver a série de Taylor de f(x) = $\frac{1}{cos(ax)}$ em torno de x = 0 de ordem 7.
2. Resolver a equação $(x + LNa)^2$ = x em x.
3. Encontrar:
   - $\int \frac{(x+a)}{x^2+x+1} dx$
   - $\int_0^1 \sqrt{4 - x^2} dx$
4. Faça a plotagem em 3D de $z = \frac{3cos((x^2+y^2)/3)}{3+x^2+y^2}$
5. Encontre a derivada de $sin(x + 2e^y).arctg(z^2)$ em relação a z.
6. Encontre o determinante, a sua inversa, a transposta, o polinômio característico e os autovalores de

$$\begin{pmatrix} -1 & \sqrt{2} & 2 & -1 & 0 \\ 1/3 & -5 & \sqrt{3} & 2 & 0 \\ 1/4 & -6 & -\sqrt{2} & \sqrt{5} & 1 \\ 0 & 0 & -1 & 1 & 0 \end{pmatrix}$$

4.10 Considerações Complementares

A partir do segundo semestre de 1991 a Soft Warehouse começou a distribuir a família das Versões 2 do DERIVE que trazem algumas melhorias de peso. No capítulo 8 fornecemos maiores detalhes sobre esses acréscimos no software, que segundo seu principal responsável, David Stoutemyer, apresenta " 2000 anos de conhecimento matemático em um disco". Entre as principais capacidades das novas versões podemos destacar; Transformadas de Laplace, Funções para resolução de equações diferenciais ordinárias de primeira ordem e de segunda ordem, Funções de Bessel, Funções hipergeométricas, Funções especiais, além da inclusão de Programação, com definições de funções usando recursão ou interação, programas usando construções do tipo if-then-else, etc.

4.11 Publicações Pertinentes

Apresentamos agora uma pequena lista de publicações sobre o software, que certamente contribuirão para o aprofundamento do investigador em várias nuances não abordadas, enfocadas de maneira introdutória ou mostradas de modo pouco abrangente no nosso texto, e que tem sido bastante utilizado pela comunidade de usuários do DERIVE em todo o mundo.

1. STOUTEMYER D., RICH A., DERIVE User Manual, Soft Warehouse, Honolulu, Havaii, U.S.A., 1988.
2. GLYNN J., Exploring Math from Algebra to Calculus with DERIVE, A Mathematical Assistant, Mathware, U.S.A., 1989.
3. GILLIGAN L., MARQUARDT J., Calculus and the DERIVE Program, Gilmar Publishing Company, Cincinnati, U.S.A., 1990.
4. EVANS B., JOHNSON J., Uses of Tecnology in the Mathematics Curriculum, CipherSystems, U.S.A., 1990.
5. ELLIS W., LODI E., DERIVE for the Calculus Student: A Tutorial, Brooks/Cole Publishing, U.S.A., 1991.

4.12 Informações Adicionais

Para informações complementares e adicionais tanto para problemas e bugs do sistema, quanto sobre sua aquisição, fornecemos a seguir um dos endereços de distribuição do software. Outros endereços são dados no capítulo 8 do nosso texto.

Soft Warehouse, Inc., 3615 Harding Ave, Suite 505, Honolulu, Hawaii, 96816, U.S.A.

Soft Warehouse, Inc., Tel: (808) 734 - 5801

æ

Capítulo 5

TÓPICOS SOBRE A LINGUAGEM C

5.1 Histórico

A linguagem C foi desenvolvida no início dos anos 70, no Centro de Pesquisas da Bell Laboratories por Dennis Ritchie. Sua primeira implementação foi feita num PDP-11 da DEC com sistema operacional UNIX, tendo sido em seguida, utilizada para reescrever o UNIX.

O desenvolvimento de C foi fortemente influenciado por B, desenvolvida por Ken Thompson, que por sua vez teve suas origens em uma linguagem relativamente "antiga", BCPL, desenvolvida por Martin Richards. Embora muitas caracteríticas de C sejam comuns com BCPL, C não é um dialeto dessa última.

Nas linguagens BCPL e B, o único tipo de dados embutido é a palavra de máquina, sendo o acesso a qualquer outro tipo de objeto feito através de operadores e chamadas de funções especiais.

Devido a sua concepção simples e flexível, a linguagem C veio a se tornar rapidamente, uma ferramenta poderosa em aplicações técnicas (CAD / CAM), aplicações comerciais (Planilhas de Cálculo), criação de compiladores, criação de sistemas operacionais, editores, em controle de processos e como linguagem de implementação de dois dos maiores softwares de computação algébrica da nossa época, o MATHEMATICA e o MAPLE, descritos com detalhes nos capítulos 6 e 7 do nosso texto.

5.2 Objetivos da Linguagem

C é uma linguagem de programação de finalidade geral que permite economia de expressão , modernos fluxos de controle e estruturas de dados e um rico conjunto de operadores. C, com frequência, é denominada uma linguagem de computador de nível médio. Nível médio não tem uma conotação negativa; não significa que C é menos poderosa, mas difícil de utilizar ou menos desenvolvida que uma linguagem de alto nível; nem significa que C seja similar a uma linguagem de baixo nível que não passa de uma representação simbólica do código de máquina. C é uma linguagem de nível médio porque combina elementos de uma linguagem de alto nível com a funcionalidade do assembly.

Pensamos com frequência nas linguagens de nível médio como linguagens de construção, porque a programação é feita primeiramente se criando as rotinas que vão executar todas as funções necessárias ao programa, para em seguida reuni-las.

A linguagem vem sendo largamente utilizada no desenvolvimento de sistemas de programação e em aplicações onde são necessários eficiêcia, confiabilidade, simplicidade e rapidez na implementação de programas.

C é também bastante indicada para escrever sistemas complexos de programas tais como compiladores, analisadores léxicos, formatadores de documentos e banco de dados. A própria linguagem através das funções, induz a utilização do método de projeto "top-down", onde problemas grandes são divididos sucessivamente em pequenos pedaços, onde podem ser utilizados os diagramas de bloco estruturados, nos quais são feitos refinamentos sucessivos para se chegar a um ponto onde é possível codificar diretamente em linguagem C.

A disseminação da linguagem C foi muito rápida devido a algumas facilidades e alguns fatores técnicos, dentre os quais podemos citar:

- Geração de códigos executáveis mais rápidos e de tamanhos menores.
- Conjunto de comandos pequeno e bem definido.
- Facilidade de migração de softwares entre os vários sistemas de hardware.
- Poderosos recursos de acesso à máquina, podendo substituir com vantagens grandes módulos escritos em ASSEMBLY.
- Padronização bem definida a nível de linguagem e a nível de funções oferecidas pelos fabricantes.
- Recursos de definição de tipos complexos de dados.
- Disponibilidade de comandos estruturados.

- Total interação com vários sistemas operacionais.
- Implementação de outras linguagens.

Com essas e outras vantagens e facilidades, C vem substituindo sistematicamente linguagens como BASIC, FORTRAN, PASCAL, ASSEMBLY e outras, na escrita de sistemas profissionais em diversas áreas.

5.3 Características da Linguagem

C não provê operações para manipular diretamente objetos compostos tais como cadeias de caracteres, conjuntos, listas ou vetores considerados como um todo. A linguagem não define nenhuma facilidade para alocação de memória outra que a definição estática e a disciplina de pilha fornecidas pelas variáveis locais de funções; não há monte (heap) ou coleta de lixo.

C é relativamente pequeno, um compilador C pode ser simples e compacto apresentando um alto grau de portabilidade. Podemos inclusive esperar que um programador conheça, entenda e até mesmo use regularmente a linguagem inteira. C não é fortemente tipada, sendo bastante permissiva quanto a conversão de dados, embora não converta dados de forma tão liberal quanto PL/I. A linguagem provê apenas os tipos embutidos, caractere, inteiros de diversos tamanhos, reais de precisão simples e dupla e ponteiros. Adicionalmente possui construtores para criação dos seguintes tipos agregados: arrays, struct (produto cartesiano) e union (união discriminada).

Com relação a operações sobre os tipos elementares, C possui uma grande variedade, o que faz com que seja considerada uma "linguagem de expressões". Outra característica peculiar é sua sensibilidade para o tipo de letra , maiúsculas e minúsculas , nomes iguais, escritos com tipos de letras distintas, representam identificadores diferentes.

C é uma linguagem estruturada, assim como o são ADA e PASCAL; COBOL e FORTRAN são linguagens não estruturadas. A principal característica de uma linguagem estruturada é a utilização de blocos. Um bloco é um conjunto de instruções que estão ligadas logicamente. Por exemplo, imagine uma instrução IF que, sendo bem sucedida, executará cinco instruções discretas, se estas instruções puderem ser agrupadas e facilmente referenciadas, formarão um bloco. A nível de unidades de programação, as únicas unidades previstas são os blocos e as funções. Uma linguagem estruturada fornece uma variedade de possibilidades de programação.

A linguagem oferece estruturas de controle a nível de cláusulas, comuns à maioria das linguagens: sequenciamento, seleção (if-else, else-if e switch), e repetição (while, do-while e for).

Uma linguagem estruturada permite que se utilize sub-rotinas compiladas separadamente sem

que pertençam ao programa propriamente dito, isto significa que se pode criar uma biblioteca de sub-rotinas, formadas por funções úteis e testadas, que podem ser acessadas por qualquer programa escrito.

Uma linguagem estruturada permite que se insira instruções, e não exige o conceito restrito de campo como em FORTRAN.

As linguagens estruturadas são mais modernas e, de fato uma das características de uma linguagem de computação "antiga" é de não ser estruturada. Em virtude de sua clareza, as linguagens estruturadas não são apenas mais fáceis de serem utilizadas em sua programação como também mais fáceis de serem mantidas.

Em C as funções podem ser chamadas recursivamente, mas não podem ser aninhadas dentro de outras funções ou blocos, embora seja permitido o aninhamento de blocos.

C não provê facilidades de entrada e de saída; não há comandos READ ou WRITE, nem métodos de acesso a arquivos. Todos esses mecanismos devem ser fornecidos por funções explicitamente chamadas. A maior parte das implementações de C incluem uma coleção razoavelmente padronizadas dessas funções. De forma semelhante, C oferece somente construções simples de fluxo de controle; testes, laços, agrupamentos e sub-programas, mas não multiprogramação, operações paralelas, sincronização ou sub-rotinas. Embora a falta de algumas dessas facilidades possa parecer uma grave deficiência,(tenho que chamar uma função para comparar duas cadeias de caracteres!?), a manutenção da linguagem em dimensões modestas tem trazido benefícios fantásticos e reais.

Por muitos anos, a definição de C foi o manual de referência da primeira edição original do livro de Kernigham e Ritchie. Em 1983, o American National Standards Institute, (ANSI), criou um comitê para prover uma definição moderna e abrangente de C . A definição resultante, o padrão ANSI ou ANSI C foi aprovado em 1988. A maioria das características do padrão já tem suporte por compiladores modernos. O padrão é baseado no manual de referência original. A linguagem foi relativamente pouco alterada; uma das finalidades do padrão foi assegurar que a maioria dos padrões existentes continuassem válidos, ou, se isso falhasse, que os compiladores pudessem produzir avisos quanto ao novo comportamento.

Para a maior parte dos programadores, a mudança mais importante é uma nova sintaxe para declarar e definir funções. Uma declaração de função pode agora incluir uma descrição dos argumentos da função; a sintaxe da definição se altera de forma correspondente. Esta informação extra facilita bastante a detecção pelos compiladores de argumentos sem correspondência o que é um acréscimo muito útil à linguagem.

Uma segunda contribuição significativa do padrão é a definição de uma biblioteca acompanhando o C . Ela especifica funções para acesso ao sistema operacional - por exemplo, para ler e gravar arquivos - , entrada e saída formatada, alocação de memória, manipulação de cadeias de caracteres e coisas desse tipo. Um conjunto de cabeçalhos-padrão dá um acesso uniforme às declarações de funções e tipos de dados. Os programas que usam esta biblioteca para interagirem com um sistema "da casa" possuem comportamento compatível assegurado. A maior

parte da biblioteca é intimamente moldada na "biblioteca de E / S padrão" do sistema UNIX. Esta biblioteca foi descrita na primeira edição, e tem sido amplamente utilizada também em outros sistemas.

Como os tipos de dados e estruturas de controle de C são suportados diretamente pela maioria dos computadores, a biblioteca em tempo de execução necessária para implementar programas autocontidos é pequena. As funções da biblioteca-padrão são chamadas somente de forma explícita, de modo que podem ser evitadas se não forem necessárias. A maior parte delas pode ser escrita em C, e, exceto pelos detalhes do sistema operacional que elas ocultam são portáteis.

Embora C equipare-se com as capacidades de muitos computadores, ela é independente de qualquer arquitetura de máquina em particular. Com um pouco de cuidado, é muito fácil escrever programas portáteis, isto é que possam ser rodados sem alterações em uma série de hardware diferentes. O padrão torna explícitos os problemas de portabilidade, e prescreve um conjunto de constantes que caracterizam a máquina em que o programa é rodado.

Os compiladores avisarão sobre a maior parte dos erros de digitação, e não existe conversão automática de tipos de dados incompatíveis. Apesar disso, C retém a filosofia básica de que os programadores sabem o que estão fazendo; ela só pede que eles digam suas intenções explicitamente.

C, como qualquer outra linguagem tem suas falhas. Alguns operadores tem a precedência errada; algumas partes da sintaxe podiam ser melhores etc. No entanto, C tem provado ser uma linguagem extremamente expressiva e efetiva para um grande variedade de aplicações.

5.4 Elementos Básicos de Programação

Para ser possível elaborar os primeiros programas podemos citar os elementos fundamentais de programação na linguagem C; tipos básicos de dados, operadores da linguagem, recursos de entrada e saída, comandos de seleção, comandos de iteração e as funções.

Tipos de dados ⟶ char, int, float e double.

Operadores da linguagem ⟶ de atribuição, de binários unários, de auto incremento / decremento, de bits, de endereço.

Entrada ⟶ printf, puts e putchar.

Saída ⟶ scanf, getchar.

Seleção ⟶ if com operadores relacionais e lógicos.

Iteração ⟶ while, for e do ... while.

Funções ⟶ Ver bibliografia.

Nota: Uma função em C pode ser "comparada" a um procedimento em PASCAL ou PL/1.

Como em algumas outras linguagens de programação, o ciclo de desenvolvimento de programas é composto das seguintes etapas: Especificação (Definição do problema), Detalhamento (Implementação de funções especificadas na etapa anterior), Codificação (Tradução da solução para o vocabulário do computador), Depuração e Testes (Problemas e bugs, testes dos programas implementados e verificação se atingem as metas especificadas) e Documentação Final (Descrição da forma de utilização do sistema).

5.5 Esboçando um Programa

A única forma de aprender uma linguagem de programação é escrever programas nessa linguagem. Este é o obstáculo básico, para passar por ele deve-se criar o texto do programa em algum lugar, compilá-lo co sucesso, carregá-lo, executá-lo e ver como saiu a resposta. Uma vez aprendidos esses detalhes mecânicos, tudo o mais é relativamente fácil, desde que alguns procedimentos para se escrever um programa sejam seguidos com certo rigor.

Existem três maneiras possíveis de se escrever um programa: de-cima-para-baixo (top-down), de-baixo-para-cima (botton-up) e ad-hoc. No processo top-down se inicia com a rotina de nível mais elevado e se dirige às rotinas de nível inferior. Uma boa maneira de se iniciar a codificação de qualquer programa, é definir exatamente o que o programa vai fazer, no nível mais elevado. C como uma linguagem estruturada permite seguir o método top-down. Esse método poderá fornecer um código limpo, legível, e de fácil manutenção; poderá ainda ajudar a esclarecer a estrutura geral e a operação do programa, antes de codificar as funções de nível inferior. O método top-down assemelha-se a um esboço, onde se inicia com a idéia geral, e se define progressivamente e de maneira melhor a cada nível subsequente; é por nós considerado o melhor modo de se escrever um programa.

O processo botton-up opera em sentido inverso; se inicia com diversas rotinas que são construídas progressivamente em estruturas mais complexas, terminando na rotina mais elevada. O processo ad-hoc não apresenta um método pré-determinado.

5.6 Organização de Programas

Um programa em C é organizado através de um título e do corpo do programa, sendo este iniciado por uma chave ({) e terminado com outra (}). O título do programa usado pelo compilador como referêcia é "main( )". C permite que programas extensos sejam divididos em

pequenas seções ou funções, que podem ser chamadas pelo programa principal.

Ao escrevermos programas, devemos nos preocupar com seu layout, pois um programa bem organizado é mais simples de ser lido e entendido. A estrutura da linguagem faz com que programas em C sejam formados por funções como apresentado no exemplo genérico abaixo.

```
nome_da_funcao
{
     instrucao;
     instrucao;
     {
         instrucao;
             --
      }
      instrucao;
 }
```

Nota: Observe que cada vez que abrimos uma chave, deslocamos o código do programa para a direita e quando fechamos cada chave deslocamos o código para a esquerda, note também que letras minúsculas são utilizadas na formação de nomes de variáveis, funções e comandos; as constantes devem ser escritas em letras maiúsculas.

Todos os programas devem ser bem documentados, isto é, devem conter comentários como guia de compreensão para quem vier a tentar acompanhar o código. Um comentário em C é iniciado com o par de caracteres "/ * " e finalizado pelo par " * /" conforme apresentado abaixo.

```
/* progi.c
     Programa ilustrativo para mostrar as fases de compilacao,
     linkedicao e execucao.
                  Rio de Janeiro, 20 / 12 / 91
*/
main( )
{
      printf("Programa Inicial");
}
```

Nota: A instrução printf( ) corresponde a uma função que faz parte do compilador C. Essa função escreve na tela o texto entre aspas duplas, e é um programa separado, que pertence a uma biblioteca que acompanha o compilador.

5.7 Programas Exemplo

A finalidade dessa seção é apresentar dois pequenos programas, que fornecem uma pequena mostra de um dos usos da linguagem em aplicações matemáticas. Esses exemplos apresentados aqui foram testados em um compilador Turbo C, versão 2.0 e serão compilados sem erro na maioria dos compiladores; recomendamos a consulta do manual do usuário da linguagem, caso ocorram pequenas variações entre compiladores diferentes.

Primeiro Exemplo

PROGRAMA CÁLCULO.C ⟶	Programa comentado para simular uma calculadora simples, só com as quatro operações elementares.

```
#include <stdio.h>

main() /* Programa emulador de calculadora */
{                                        \
float n1, n2;
char op;

do
  {
  printf("Entre com dois numeros reais e uma das operacoes elementares.");
  printf("De um espaco em branco entre cada um de seus dados de entrada.\n");
  printf("Exemplo:  92.435 b 6.18 b *  e [Enter] produz 571.2483.\n");
  scanf("%f %f %c",&n1,&n2,&op);
  switch(op)
    {
    case '+' : printf(%f\n",n1 + n2);
               break;
    case '-' : printf(%f\n",n1 - n2);
               break;
    case '*' : printf(%f\n",n1 * n2);
               break;
    case '/' : printf(%f\n",n1 / n2);
               break;
    default  : printf("Nao tratamos dessa operacao neste programa.\n");
    }
  } while(op != 'q');
}
```

Segundo Exemplo

PROGRAMA CONVERSAO.C ⟶	Programa para converter numeros complexos de forma polar para forma retangular e vice-versa.

```
#include <stdio.h>

#define PI 3.141592654

double getnumd(),

       x,y,

       modulo,
       fase,

       pow(),

       sqrt(),

       gr(),

       rg(),

       cos(),

       atan(),

       sin();

main()
{
   char tecla;

   clrscr();

   linha();

   puts("\t\tConversao de numeros complexos");

   while()
```

```
    {
       linha();

       printf("(0 = P -> R) (1 = R -> P) (2 = fim) ---> ");

       tecla=getchar();

       getchar();        /* limpa o buffer do teclado */

       if (tecla=='2') break;

       linha();

       switch (tecla)
       {
          case '0': po_re();

                    break;
          case '1': re_po();

                    break;
          default : puts("\t\tOpcao nao valida ...");
       }
    }
    clrscr();

    puts("\t\tT C H A U ...");
}
po_re()
{
    printf("Modulo = ");

    modulo=getnumd();

    printf("Fase   = ");

    fase=getnumd();

    linha();

    x=modulosin(gr(fase));
```

```
    y=modulocos(gr(fase));

    printf("Z = %.2f + j%.2f\n",x,y);
}
re_po()
{
    printf("x = ");

    x=getnumd();

    printf("y = ");

    y=getnumd();

    linha();

    modulo=sqrt(pow(x,2.0)pow(y,2.0));

    fase=atan(y/x);

    fase=rg(fase);

    printf("Modulo = %.2f    Fase = %.2f\n",modulo,fase);
}
double gr(q)

double q;
{
   return(q*PI/180);
}
double rg(q)

double q;
{
   return(q*180/PI);
}
```

Nota: Naturalmente esses programas podem ser melhor documentados e escritos de maneira mais concisa por quem já programe com desenvoltura em C. Caso tenha esse perfil, sugerimos tentar fazer isso.

5.8 Programação de Grandes Sistemas

A programação de grandes sistemas está intimamente ligada ao problema da decomposição modular, ou seja, projetos de grandes softwares, como por exemplo os sistemas de computação algébrica, necessitam ser subdivididos em pequenos "pedaços", definidos em módulos.

Os sistemas estudados nos capítulos 6 e 7 do presente texto se encaixam bem nessa idéia, como por exemplo podemos citar o MATHEMATICA versão 2.0, tem aproximadamente 200.000 linhas de código e dezenas de rotinas para cálculos distintos.

C não possui o conceito de compilação em separado, em outras palavras, em C cada arquivo-fonte é compilado individualmente, sem verificações cruzadas entre os vários arquivos no que tange a tipos de dados e passagem de parâmetros; podemos concluir como consequência imediata, que certas vezes, unidades corretas quando combinadas, podem gerar programas com erro, o que aconteceu em versões anteriores daquele sistema nas rotinas de integração.

5.9 Considerações Gerais

Obviamente apresentamos conceitos bastante triviais sobre C, pois esse tema foge ao escopo do nosso estudo. Foram mostrados apenas considerações superficiais sobre a linguagem e sobre seu poderio como ferramenta para a construção de sistemas de computação algébrica. Vamos fornecer a título de ilustração, para o leitor que conheça relativamente bem conceitos mais profundos, tanto a nível de domínio da linguagem, quanto a nível de programação, o "coração" de um algoritmo simbólico de diferenciação, e a "alma" de um algoritmo para determinação do M.D.C. entre dois inteiros a e b. Eventuais tentativas de começar a pensar em escrever um software de computação algébrica sempre começaram dessa forma ou seja, primeiramente, através de implementações de algoritmos claros, corretos e eficientes.

Uma das mais importantes operações em computação matemática é a diferenciação. Sistemas simbólicos modernos podem calcular derivadas muito bem hoje em dia; a dificuldade mais observada atualmente nessa tarefa, é a de simplificar ao máximo o algoritmo utilizado ou a implementação feita.

Considere diferenciar uma função elementar arbitrária g(x) com respeito a x.

```
                    Diferenciacao Simbolica
                      [ diff(g(x),x) ]

Algoritmo de diferenciacao : diff(g,x)
Dado: Uma funcao elementar g(x) e a variavel x
Resultado: A derivada de g(x) com respeito a x

PASSO 1 : Se g = x entao 1
PASSO 2 : Se g e' uma constante entao 0
PASSO 3 : Se g = a + b entao diff(a,x) + diff(b,x)
PASSO 4 : Se g = a * b entao b*diff(a,x) + a*diff(b,x)
PASSO 5 : Se g = a**b entao a**b*(log(a)*diff(b,x) + b*diff(a,x)/a)
PASSO 6 : Se g = f(x) onde f e' sen, cos, tan, log, exp  e  assim  por
          diante entao o resultado e' recuperado de uma tabela onde
          as derivadas dessas funcoes sao armazenadas.
PASSO 7 : Se g = a(b(x)) entao se aplica a regra da cadeia.
```

Um outro algoritmo interessante é o que computa o máximo divisor comum de dois inteiros (MDC).

```
                    O Algoritmo Euclideano
                   [ para MDC entre a e b ]

PASSO 1 : Se b = 0 entao a e' a resposta (PARE)
PASSO 2 : Se a = 0 entao b e' a resposta (PARE)
PASSO 3 : Faca r o resto da divisao de a por b.
PASSO 4 : Coloque b no lugar de a , e r no lugar de b.
PASSO 5 : Volte ao passo 1
```

5.10 Publicações Pertinentes

A seguir apresentamos uma lista de publicações, sobre a linguagem C, que certamente servirá como ponto de partida para o usuário interessado em ampliar seus conhecimentos a nível de conteúdo e de programação. Nunca é demais ressaltar que C é uma das linguagens que ainda tem muito a oferecer em termos de contribuição à ciência, em geral, e, em particular, aos softwares de computação algébrica.

1. KERNIGHAN B.W., RITCHIE D.M., *C - A Linguagem de Programação (Padrão ANSI)*, Editora Campus, 1990.

2. SCHILDT H., *Linguagem C, Guia do Usuário*, McGraw-Hill, 1986.

3. SILVA J.C.G., ASSIS F.S.G., *Linguagens de Programação, Conceitos e Avaliação*, McGraw-Hill, 1988.

4. KELLY-BOOTLE S., *Dominando o Turbo C*, Editora Ciência Moderna, 1989.

Capítulo 6

MATHEMATICA

6.1 Introdução

O sistema de computação algébrica MATHEMATICA foi concebido por um grupo de pesquisadores liderados por Stephen Wolfran que sempre esteve ligado ao desenvolvimento de pesquisas nessa área. Stephen Wolfran é atualmente diretor do Centro de Pesquisas em Sistemas Complexos, professor de Física, Matemática, e Ciência da Computação da Universidade de Illinois.

Em 1981 Wolfran fazia parte de um grupo que trabalhava no desenvolvimento do sistema SMP, e juntamente com mais sete pesquisadores começou então a desenvolver outro software, o MATHEMATICA que já é considerado um grande sistema e vem sendo amplamente utilizado pela comunidade acadêmica.

MATHEMATICA é um programa escrito em linguagem C com aproximadamente 150.000 linhas de tamanho compreendidas só no seu núcleo. A parte em que o usuário interage com o programa para um computador como o Macintosh tem 50.000 linhas adicionais.

Os autores, Daniel Grayson, Roman Maeder, Henry Cejtin, Theodore Gray, Stephen Omohundro, David Ballman, Jerry Keiper e inclusive o próprio Wolfran, admitem explicitamente que foram influenciados por vários e distintos softwares fonte, e que idéias vindas deles aparecem em vários lugares no MATHEMATICA: linguagens numéricas interativas tais como BASIC, sistemas numéricos interativos tais como MathCAD, MATLAB e TK!Solver, sistemas algébricos tais como MACSYMA, MAPLE, SCHOONSCHIP, SCRATCHPAD e obviamente o SMP, linguagens gráficas interpretadas tais como o PostScript, linguagens de manipulação de listas numéricas e simbólicas tais como APL e LISP e linguagens estruturadas de programação tais como C e PASCAL.

MATHEMATICA roda em vários computadores entre os quais; Apollo DN 3000, 3500, 4000

e 4500; Ardent Titan; Hewlett-Packard 9000, 300 e 800; IBM AIX/RT; MIPS M/120; NeXT; Silicon Graphics Iris 4D; Sony NFWS; Stellar GS 1000; Sun 3, 4 e 386i; DEC e Cray. A versão standard roda no Macintosh Plus, SE e II, a versão Macintosh II roda somente no Macintosh II. O sistema requer 2.5 MB mas 4 MB de RAM são recomendadas pelos construtores do software. MATHEMATICA roda também na versão 386 (sem coprocessador numérico), versão 386/7 (com coprocessador numérico 287 ou 387), com processador 80386, sistema operacional MS-DOS com versões maiores que 3.0 inclusive, 640 Kilobytes de RAM mais 1 megabyte de memória extendida, com 6 megabytes de espaço no disco rígido e além de outros, com os monitores padrão CGA, EGA, VGA, MCGA e Hercules.

6.2 O Porquê da Utilidade do MATHEMATICA

- A versão utilizada no nosso estudo foi a do MATHEMATICA - 1.2 instalado num computador PC - 386/DX o que já representa uma grande vantagem sobre alguns softwares de computação algébrica, pois permite uma maior divulgação para o investigador interessado devido a portabilidade, a facilidade de instalação e a acessibilidade do seu uso. Como uma grande quantidade de memória é geralmente necessária para a computação simbólica, MATHEMATICA parece ter chegado a uma articulação auspiciosa, um grande número de Macintosh II, Intel 80386, PC's - 386 e Workstations com memória e velocidades adequadas já executam um trabalho sério e eficaz operando com o sistema.

- O manual do MATHEMATICA, embora extenso, foi, por nós, considerado o mais abrangente, claro, didático e que satisfaz, a nível de profundidade, desde o investigador que já tenha prática nesse tipo de programa até o iniciante em utilização desse tipo de sistema; o que é bastante positivo, pois não é o que se encontra normalmente na literatura similar de outros sistemas.

- A qualidade das plotagens para computação algébrica e discreta no MATHEMATICA é uma das melhores quando comparadas com seus "concorrentes" mais diretos, e com display de saída para o tratamento gráfico num formato que pode ser manipulado. A parte gráfica do sistema é o maior feito já conseguido em softwares de computação algébrica e onde daremos ênfase especial no decorrer do parágrafo e na avaliação comparativa desse ítem com similares de outros programas.

- Embora o MATHEMATICA tenha sido escrito em linguagem C , não é necessário por parte do usuário o conhecimento da linguagem de implementação do software, e com algumas horas de uso e consulta no manual já é possível resolver problemas razoavelmente difícies e complexos.

- O investigador pode gerar expressões FORTRAN de expressões fornecidas pelo MATHEMATICA o que é uma característica extremamente importante pois combina capacidades

numéricas e simbólicas, além disso, a interface de saída, que faz com que sejam gerados displays em C e em TEX também é parte de suas habilidades, que o é facilmente gerenciado pelo sistema.

- O MATHEMATICA fornece um arquivo com textos, gráficos, tipos de entradas e saídas do software, que também organiza e documenta os cálculos do investigador, além de outras facilidades; esses objetos são chamados Notebooks.

- MATHEMATICA é um sistema de computação algébrica que faz cálculos matemáticos simbólicos e iterativos e é também um excelente instrumento para exposições sobre matemática em ensino.

- MATHEMATICA provê facilidades para computação numérica de precisão exata e arbitrária e tem também um armazenamento de informações sobre simplificação e manipulação de funções matemáticas, além de operar com objetos poliédricos.

- O sistema vem galgando posições e alcançando sucesso crescente na comunidade usuária principalmente nas Universidades e Centros Científicos; MATHEMATICA já se encontra entre os cinco primeiros softwares de computação algébrica na preferência dos investigadores "avançados" .

- MATHEMATICA é uma linguagem de programação que unifica idéias de programações procedurais, funcionais, baseadas em regras, orientadas a objetos e baseadas em restrições (constraint-based programming).

- O sistema também provê funcionalidade construída para manipulação algébrica de fórmulas compreendendo polinômios, funções racionais, as funções de cálculo elementar, funções avançadas da Física, ("funções especiais"), funções da teoria dos números, funções combinatórias, composições de estruturas de dados (listas, matrizes, funções literais e argumentos, etc.) e debugagem.

- MATHEMATICA fornece ainda operações internas construídas para manipulação algébrica tais como diferenciação simbólica, integração (na forma fechada), somatórios, aproximação em séries de Taylor etc. (tão bem quanto operações que possam ser extendidas por programação).

6.3 Capacidades e Usos do MATHEMATICA

6.3.1 Capacidades do MATHEMATICA

Tratamento Gráfico

O ponto alto de qualquer discussão sobre o MATHEMATICA é sobre sua habilidade gráfica, pois não somente plotagens bidimensionais são possíveis com também uma grande variedade de plotagem 3D e plotagens de contorno são disponíveis. É possível escolher a densidade, a "iluminação" ou "colorido" da plotagens, os pontos de vista para plotagens tridimensionais e muitos outros parâmetros.Como exemplo;

In[1] := Plot[Release[Table[BesselJ[n,x], {n,4}]], {x,0,10}] e [Enter]

fornece a plotagem das funções de Bessel J_nx com n variando de 1 a 4.

Nota : O Release diz para o MATHEMATICA primeiramente fazer uma tabela de funções e somente então completá-la para valores particulares de x

Note que colchetes são geralmente usados para se definir listas. Muitos comandos Plot esperam uma lista de 3 elementos como segundo argumento. O primeiro ítem do segundo argumento é a variável a ser plotada e os outros dois os pontos finais do seu domínio no caso do R . Observe as seguintes plotagens:

In[2] := DensityPlot[Sin[x + Sin[y]], {x, 0, 8Pi}, {y, 0, 8Pi}, Mesh -> False, PlotPoints -> 100] e [Enter] fornece a plotagem da figura 1.

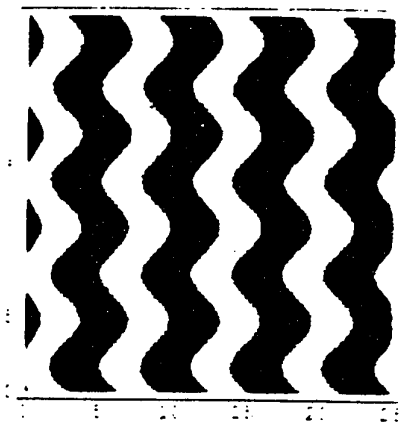


FIGURA 1

In[3] := Plot3D[Sin[x] Sin[y], {x, -10, 10}, {y, -10, 10}, PlotPoints -> 40] e [Enter] fornece a plotagem da figura 2.

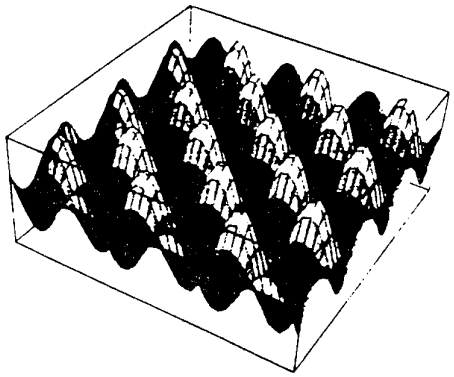


FIGURA 2

Ao observar a apresentação gráfica tente entender segundo sua entrada a delimitação da figura no $\mathcal{R}^3$, digite agora;

In[4] := Show[%, Lighting -> True]

E a figura 2 aparece colorida, sombreada e iluminada.

Os criadores do MATHEMATICA escolheram a linguagem gráfica interpretada PostScript para essas construções, pois ela assegura um display de saída para a maior parte de tipos e modelos de impressoras disponíveis. Daremos mais detalhes sobre a parte gráfica do software na seção *Tratamento Gráfico.*

Cálculos Aritméticos e Numéricos

MATHEMATICA pode computar resultados numéricos para precisão arbitrária, reconhece indeterminações e infinitos diretos. Pode gerar números pseudorandômicos. Possui funções numérico-teóricas para K módulo n, quociente de m e n, MDC e MMC, fatora inteiros, procura o k-ésimo número primo, trabalha com módulo de potências, a função de Euler, a função m de Mobius, a função divisor, o símbolo de Jacobi, o MDC extendido, além de várias outras habilidades e capacidades não mostradas aqui.

Funções combinatórias incluem o fatorial, o fatorial duplo, os coeficientes binomiais e multinomiais, números e polinômios de Bernoulli, números e polinômios de Euler, números de Stirling de primeira e segunda espécie e duas funções de partição.

Funções Transcedentais e Funções Especiais

Todas as funções elementares usuais; exponenciais, trigonométricas, funções hiperbólicas e suas inversas, são construídas no MATHEMATICA. (MATHEMATICA conhece funções de valores complexos.)

Reconhece polinômios de Legendre, funções harmônicas esféricas, polinômios de Chebyshev, polinômios de Gegenbauer, polinômios de Hermite, polinômios de Laguerre, polinômios de Jacobi, funções de Airy, funções de Bessel, funções beta, funções de erro, integrais exponenciais, funções gama, todas as espécies de funções hipergeométricas, funções de Legendre, funções transcedentais de Lerch, a integral logaritmica, o símbolo de Pochhammer, a função digama, a n - ésima derivada da função digama, a função polilogarítmica, a função zeta de Riemann e a função zeta generalizada. MATHEMATICA conhece funções elípticas e integrais elípticas incluindo o sentido aritmético-geométrico dos dois números.

Funções Polinomiais e Funções Racionais

MATHEMATICA pode expandir ou fatorar qualquer expressão polinomial e coeficientes particulares dessas expressões. O software faz divisão polinomial e uma grande variedade de operações adicionais sobre funções racionais.

Manipulação de Equações

MATHEMATICA pode resolver uma equação ou um sistema de equações. Se possível a solução será na forma fechada. Por outro lado, elas são numéricas sempre que a precisão for exigida. MATHEMATICA pode geralmente também resolver um sistema de equações sujeito a alguma condicão.

Cálculo

MATHEMATICA faz derivadas (parciais, múltiplas, totais,...) simbolicamente. Se possível MATHEMATICA integrará uma função e dará o resultado na forma fechada, mas valores numéricos de integrais definidas também podem ser determinados. O sistema também achará a expansão em séries de potências de funções de uma especificada ordem, pode manipular séries e inverter séries.

Álgebra Linear

MATHEMATICA constrói, faz multiplicação e inversão de matrizes, determinantes e também lista matrizes menores de matrizes dadas, resolve sistemas lineares e retorna os autovalores e autovetores de um sistema e conhece tensores.

Operações Numéricas Sobre Dados e Sobre Funções

MATHEMATICA pode prover curvas para listas de dados, pode trabalhar com transformadas de Fourier de dados ou inverter uma transformada.

Como acréscimo a esse repertório, MATHEMATICA reconhece e atende a modelos e conjuntos de regras. É como se construíssemos regras e pedíssemos ao MATHEMATICA para as ler quando necessário. Essas coleções de regras são chamadas pacotes. Por exemplo, tais pacotes capacitam o MATHEMATICA para trabalhar com transformadas de Laplace, equações diferenciais, métodos numéricos para resolução de equações diferenciais, entre outras coisas. (Entretanto existem lacunas surpreendentes, MATHEMATICA não tem facilidades prontas

para fazer geometria analítica, embora essas regras sejam facilmente criadas.)

6.3.2 Usos do MATHEMATICA

Os campos de atuação do sistema são amplos e as informações que fornecemos a seguir dão apenas uma pequena idéia de que tipos de pesquisas usaram o software como ferramenta auxiliar. As fontes nas quais nos baseamos para listar essas áreas de aplicação do programa - conferências, artigos, revistas especializadas e a literatura de computação algébrica de um modo geral - sempre enfatizam que o sistema pode oferecer mais contribuição para o investigador do que aquelas que foram listadas.

Análise Numérica	Mecânica Estrutural
Relatividade Geral	Termodinâmica
Mecânica Celeste	Análise Espectal
Análise dos Elementos Finitos	Acústica
Dinâmica de Fluidos	Econometria
Desenho Auxiliar de Computadores	Análise das Deformações
Desenho de Aerofólios	Geomeria Algébrica
Desenho de Microscópios	Teoria dos Números
Desenho de Circuitos Integrados	Física das Partículas
Robótica	Física do Estado Sólido
Análise das Deformações	Matemática Experimental

6.4 Apresentando o MATHEMATICA

6.4.1 Preliminares

Esse parágrafo apresenta uma pequena demonstração sobre o uso do MATHEMATICA e tem também a intenção de ser de fácil entendimento para o usuário que não está familiarizado com o sistema.

Como de hábito sugerimos que o usuário leia essa apresentação em frente a um computador que rode o MATHEMATICA para que possa executar nossas sugestões iniciais, fazendo posteriormente alguns exercícios elementares encontrados no final do parágrafo, que certamente servirão como embrião para trabalhar em seus objetivos próprios.

Para entrar no MATHEMATICA na maioria das máquinas, bater;

cd MATH e [Enter]

MATH e novamente [Enter]

O computador fornecerá então uma mensagem, conforme abaixo, ou outras similares em versões mais recentes.

```
Mathematica (MS-DOS 386/7) 1.2 (September 27, 1989) [With pre-loaded data]
by S. Wolfram, D. Grayson, R. Maeder, H. Cejtin,
S. Omohundro, D. Ballman and J. Keiper
with I. Rivin, D. Withoff and T. Sherlock
Copyright 1988,1989 Wolfram Research Inc.
In[1]:=
```

Logo que o MATHEMATICA é inicializado, opera um prompt para uma linha de operação a qual examina (para sintaxe) e calcula, retornando os resultados dos cálculos; MATHEMATICA aceita uma rica variedade de comandos de matemática como prompt. MATHEMATICA roda numa grande quantidade de máquinas, como citado na introdução, e para fazer isso é organizado em duas partes - o núcleo e a parte em que o usuário interage com o programa.

O prompt inicial é,

In[1] :=

onde após a entrada do seu pedido de acordo com a sintaxe do software, é apresentada a resposta no prompt de saída;

Out[1] :=

e assim por diante.

Aqui temos um prompt típico após terem sido efetuados 36 "trabalhos" pelo sistema.

In[37] :=

MATHEMATICA numera e lembra dos seus inputs, nessa seção exemplo, o prompt é 37 - em outras palavras é o trigésimo sétimo pedido de uma determinada seção - usamos uma linguagem natural unidimensional como input para entrar com algum cálculo matemático. Após esse prompt poderíamos entrar, por exemplo, com as instruções para cálculo, no sistema, da integral indefinida $\int x^3\sqrt{x^2+1}\,dx$ e teríamos então a tela;

In[37] := Integrate[x ^3 Sqrt[x ^2 + 1], x]

e após teclarmos [Enter], o MATHEMATICA responderia com;

Out[37] := $\frac{\frac{-2(1+x^2)^{3/2}}{3}+\frac{2(1+x^2)^{5/2}}{5}}{2}$

e já teríamos então no monitor o sistema pronto para outro trabalho, o que é apresentado através do prompt de numeração seguinte, ou seja;

In[38] :=

Esse exemplo resolvido pelo sistema apresenta uma amostra da sintaxe do MATHEMATICA, todas as palavras chave reservadas começam com maiúsculas - para distingui-las das variáveis e dos procedimentos que viermos a criar - e são sempre escritas fora dos separadores - podemos definir abreviações se quisermos - colchetes são reservados para argumentos e argumentos múltiplos são listados em uma sequência. Parênteses são usados somente para agrupamentos (grupar) e para multiplicações. Os dois argumentos do Integrate especificam o integrando e a variável de integração. Multiplicações podem tanto ser entendidas pelo sistema tanto através de um espaço em branco entre os fatores, quanto através do tradicional asterisco, naturalmente temos algumas dezenas de colocações sobre sintaxe do software das quais mostramos algumas no exemplos usados na seção 6.6 do texto.

Nota : Quando o MATHEMATICA não resolve um pedido do investigador ele repete o que foi dado como entrada, no prompt de saída, na notação usual de matemática.

6.4.2 Símbolos e Chaves Especiais

- Para terminarmos uma seção do MATHEMATICA, basta digitarmos Quit e [Enter].
- As quantidades padrão $\pi \cong 3.14159...$, e $\cong 2.71828...$, mais infinito, i (raiz quadrada de -1) e $\pi/180$ (fator de conversão de graus para radianos), são respectivamente referidos por Pi, E, Infinity, I e Degree.
- % fornece o último resultado gerado; % n fornece o resultado da linha de Out[n].
- MATHEMATICA fornece resultados numéricos aproximados como uma calculadora, bastando para isso terminar seu input por // N. A sintaxe é expr // N ou N [expr], N sempre em letra maiúscula.
- Uma outra sintaxe permitida é N [expr , n] que apresenta o valor numérico da expressão com n dígitos decimais.
- O modo pelo qual se interrompe um cálculo no MATHEMATICA é feito pressionando-se CONTROL + C , CONTROL_ ou através do botão de parada.
- Quando expressões grandes não cabem em uma linha, o sistema usa o símbolo > nas linhas abaixo significando a continuação da expressão.

6.5 Entrada das Expressões

A entrada de expressões no MATHEMATICA é feita da maneira mais trivial possível que pouco difere de outros softwares de computação algébrica, como veremos nos exemplos apresentados

no texto.

Operações aritméticas no sistema são grupadas de acordo com as convenções matemáticas padrão.

Por exemplo, 5^3 + 7 significa (5^3) + 7 e não 5^(3 + 7).

As operações aritméticas comuns no MATHEMATICA são:

x + y + z	Adição
-x	Subtração
x y z ou x*y*z	Multiplicação
x^y	Potenciação
x / y	Divisão

Algumas funções matemáticas do MATHEMATICA são:

Sqrt[x]	raiz quadrada de x
Exp[x]	exponencial de base e
Log[x]	logaritmo natural de x
Log[b, x]	logaritmo de x na base b
Sin[x], Tan[x]	funções trigonométricas, argumentos em radianos
ArcSin[x], ArcCos[x]	funções trigonométricas inversas
n!	fatorial, produto de inteiros 1,2,...,n
Abs[x]	valor absoluto
Round[x]	inteiro mais próximo de x
Mod[n, m]	n módulo m, resto da divisão de n por m
Random[ ]	número pseudorandômico entre 0 e 1
Max[x, y, ...], Min[x, y, ...]	máximo e mínimo de x, y, ...
FactorInteger	fatores primos de n

6.6 Sintaxe de Algumas Capacidades do MATHEMATICA

Mostraremos agora a sintaxe correta de entrada em algumas das capacidades do sistema, escolhidas aleatoriamente, mas que de algum modo auxiliam o investigador nos primeiros passos para obtenção de resultados triviais; como de hábito enfatizamos que a utilização do manual do MATHEMATICA é indispensável por sua didática e clareza até em problemas mais sofisticados, caso venhamos a nos tornar usuários do software para trabalhos mais complexos.

6.6.1 Diferenciação

Embora o sistema opere com poder e abrangência na derivação, existem alguns momentos em que o investigador demora a entender certas "respostas" do software. Como exemplo, pedimos ao MATHEMATICA para integrar $x^2\sqrt{1-x^2}$ e então derivar o resultado, obtemos então uma expressão confusa, que após simplificada (Simplify) ainda apresentava um resultado não satisfatório, $(x^2 - x^4)/\sqrt{1-x^2}$ e ao aplicarmos finalmente a operacão (Factor), para a fatoração, obtemos $ix^2\sqrt{-1+x}\ \sqrt{1+x}$ que é matematicamente equivalente a função original mas visualmente diferente.

Sintaxe da diferenciação :

D [f , x]	derivada parcial
D [f , x_1 , x_2 , ...]	derivada múltipla
D [f , { x , n }]	derivada repetida
D_t [f]	derivada total
D_t [f , x]	derivada total

Exemplos:

In[1] := D[ArcCos[x] + x , x] e [Enter] produz,

Out[1] = 1 - $\frac{1}{Sqrt[1-x^2]}$

In[2] := $D[x\hat{}n, \{x, 3\}]$ e [Enter] produz,

Out[2] = $(-2+n)(-1+n)nx^{-3+n}$

6.6.2 Integração

Tanto o MATHEMATICA quanto seu predecessor filosófico o SMP, tiveram dificuldades ao integrar expressões com parâmetros ou singularidades. Para tratar de alguns desses problemas, o MATHEMATICA 1.2 incluiu um pacote para integração definida (DefiniteInteger), o qual, usando integração por partes corrigiu a resposta para muitos problemas, como um desses casos, temos Integrate[1 / x ^2,{ x, -1, 1 }] que fornece ∞, ao invés da resposta -2 dada previamente. Não estamos dizendo com isso que a versão 1.2 conseguiu corrigir todos os problemas do software com integração, o professor W. Kahan do Departamento de Ciência da Computação da Universidade da California, Berkeley, "coleciona" erros cometidos pelo MATHEMATICA e também por outros sistemas de computação algébrica; alguns desses, não restritos apenas a integração, são apresentados em [5] das publicações pertinentes do MATHEMATICA.

Sintaxe da Integração:

Integrate[f, x]	a integral indefinida $\int$ f dx
Integrate[f, x, xmin, xmax]	a integral definida $\int_{xmin}^{xmax}$f dx
Integrate[f, x, xmin, xmax, y, ymin, ymax]	a integral múltipla $\int_{xmin}^{xmax}$ dx $\int_{ymin}^{ymax}$ f dy

Exemplos:

In[1] := Integrate[x^n,x] e [Enter] produz,

Out[1] := $\frac{x^{1+n}}{1+n}$

que é a integral $\int x^n$ dx.

In[2] := Integrate[Log[x], { x, a, b }] e [Enter] produz,

Out[2] := a - b - aLog[a] + bLog[b]

que é a integral definida $\int_a^b$ log(x) dx.

In[3] := Integrate[x^2 + y^2, {x, 0, 1}, {y, 0, x}] e [Enter] produz,

Out[3] := $\frac{1}{3}$

que é o valor do cálculo da integral múltipla $\int_0^1$ dx $\int_0^x$ $(x^2 + y^2)$ dy . As variáveis aparecem no Integrate na mesma ordem como na notação matemática usual com o limite inferior vindo primeiro.

Nota : Exemplos mais complicados, que o usuário queira computar, podem ser perfeitamente executados com o auxílio de exemplos completos mostrados no manual.

6.6.3 Somas e Produtos

Sintaxe de Somatórios e Produtórios:

Sum[f, i, imin, imax]	a soma $\sum_{i=imin}^{imax} f$
Sum[f, i, imin, imax, di]	a soma com i sendo incrementado nos passos de di
Sum[f, i, imin, imax, j, jmin, jmax]	a soma $\sum_{i=imin}^{imax} \sum_{j=jmin}^{jmax} f$
Product[f, i, imin, imax]	o produto $\prod_{i=imin}^{imax} f$

Exemplos:

In[1] := Sum[x^i/i, { i, 1, 7 }] e [Enter] produz,

Out[1] := x + $\frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \frac{x^5}{5} + \frac{x^6}{6} + \frac{x^7}{7}$

Nota : Se o limite inferior for igual a 1, como no exemplo acima, pode ser omitido que o resultado da soma não é alterado.

In[2] := Product[x + i, {i, 1, 4}]

Out[2] := (1 + x) (2 + x) (3 + x) (4 + x)

6.6.4 Equações, Sistemas e Inequações

MATHEMATICA resolve equações algébricas em uma variável quando o maior expoente é menor do que cinco facilmente, senão vejamos;

Exemplos:

In[1] := x ^4 - 3 x ^3 + 2 a x ^2 == 0 e [Enter] produz,

Out[1] = 2 a $x^2 - 3x^3 + x^4 == 0$

No input de entrada seguinte, no nosso caso é o segundo, basta digitar;

In[2] := Solve[%, x] e [Enter], que serão apresentados os valores de x procurados em;

Out[2] = $\{ \{ x \rightarrow 0\}, \{x \rightarrow 0\}, \{x \rightarrow \frac{3+Sqrt[9-8a]}{2}\}, \{x \rightarrow \frac{3-Sqrt[9-8a]}{2}\}\}$

Nota : Se forem necessários valores numéricos para soluções que não tenham outros termos, literais, digitar no input de entrada seguinte N[%].

6.6.5 Números Complexos

Sintaxe nas Operações com Números Complexos:

x + I y	o número complexo x + iy
Re[z]	parte real
Im[z]	parte imaginária
Conjugate[z]	conjugado complexo $\overline{z}$
Abs[z]	valor absoluto $\|z\|$
Arg[z]	o argumento ϕ em $\|z\|e^{i\phi}$

Exemplos:

In[1] := (2+3i) (2-3i)-Sqrt[-25] e [Enter] produz,

Out[1] = 13-5i

In[2] := Conjugate[(7-3i)/(2+5i)+401/29] e [Enter] produz,

Out[2] = -$(\frac{1}{29})$ + $\frac{1}{29}$

6.6.6 Vetores e Matrizes

Vetores e matrizes no MATHEMATICA são simplesmente representados por listas e listas de listas respectivamente, o design do software nesse particular é bastante interessante, pois nunca permite que sejam feitas confusões entre "vetores linha" e "vetores coluna".

Vamos mostrar algumas maneiras simples de uso da sintaxe para cálculos com vetores e matrizes, por exemplo;

Sintaxe da representação de vetores e matrizes por listas:

{a, b, c} $\longrightarrow$ vetor (a, b, c)

{{a, b}, {c, d}} $\longrightarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Uma matriz 2x2

In[1] := m = {{a,b}, {c,d}}

Out[1] = {{a,b}, {c,d}}

Um vetor de 2 componentes

In[2] := v = {f,g}

Out[2] = {f,g}

Uma matriz n, 3x3, com elementos n_{ij} = i+j

In[3] := Table[i+j, {i,3}, {j,3}]

Out[3] = {{2,3,4}, {3,4,5}, {4,5,6}}

Algumas funções para vetores e matrizes:

Table[f,{i,n}]	Constrói um vetor de comprimento n calculando f com i=1, i=2,..., i=n.
Array[a,n]	Constrói um vetor de comprimento n da forma {a[1], a[2], ...}.
list[[i]]	Fornece o i-ésimo elemento na lista do vetor.
ColummForm[list]	Fornece os elementos da lista em uma coluna.
Table[f,{i,m},{j,n}]	Constrói uma matriz m × n calculando f com i variando de 1 a m e j variando de 1 a n.
Array[a,{m,n}]	Constrói uma matriz m × n cujo i,j - ésimo elemento é a[i,j]
IdentityMatrix[n]	Gera uma matriz identidade n × n.
DiagonalMatrix[list]	Gera uma matriz quadrada com elementos numa lista na diagonal.
list[[i]]	Fornece a i-ésima linha na lista da matriz.
list[[i,j]]	Fornece o i,j-ésimo elemento na lista da matriz.
MatrixForm[list]	Fornece a lista na forma usual das matrizes.

Algumas operações matemáticas com matrizes:

cm	Multiplicação por escalar.
m.n	Produto de matrizes.
Inverse[m]	Matriz inversa.
Det[m]	Determinante.
Transpose[m]	Transposta.
Eigenvalues[m]	Autovalores.
Eigenvectors[m]	Autovetores.
[*Eigenvalues*[*N*[*m*]]	Autovalores Numéricos.
[*Eigenvectors*[*N*[*m*]]	Autovetores Numéricos.

A seguir apresentamos, a título de ilustração, uma pequena secção mostrando o uso do sistema com manipulação matricial.

Exemplo:

In[1] := m = Table[1/(i+j), {i,4}, {j,4}] e [Enter] produz,

Out[1] = $\{\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}\}, \{\frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}\}, \{\frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}\}, \{\frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}\}\}$

que é a matriz pedida segundo sua lei de formação.

In[2] := MatrixForm[m] e [Enter] produz,

$$\begin{matrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{matrix}$$

$$\begin{matrix} \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \end{matrix}$$

Out[2] / / MatrixForm =

que é a matriz m apresentada numa forma mais usual.

In[3] := Det[m]

Out[3] = $\frac{1}{423360000}$

In[4] := Transpose[m]

Out[4] = $\{\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}\}, \{\frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}\}, \{\frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}\}, \{\frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}\}\}$

In[5] := Inverse[m]

Out[5] = {{200, -1200, 2100, -1120}, {-1200, 8100, -15120, 8400},
> {2100, -15120, 29400, -16800}, {-1120, 8400, -16800, 9800}}

In[6] :=Inverse[m].m e [Enter] produz,

Out[6] = {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
que é o produto da inversa de m pela matriz m.

In[7] := mn = N[m] e [Enter] produz,

Out[7] = {{0.5, 0.333333, 0.25, 0.2}, {0.333333, 0.25, 0.2, 0.166667},
> {0.25, 0.2, 0.166667, 0.142857}, {0.2, 0.166667, 0.142857, 0.125}}
que é a aproximação numérica da matriz m.

In[8] := Eigenvalues[mn] e [Enter] produz,

Out[8] = {0.977556, 0.0622678, 0.00182126, 0.0000213066}
que são as aproximações numéricas dos autovalores

Nota : Naturalmente esse exemplo trivial dá apenas uma idéia das potencialidades dessa capacidade do sistema; só no capítulo 3 do manual do MATHEMATICA, mais de 30 páginas são dedicadas exclusivamente a Álgebra Linear e várias possibilidades oferecidas pelo software, inclusive tópicos avançados como decomposição de valores singulares e tensores, são tratados com clareza e abrangência.

6.6.7 Solução de Sistemas Lineares

Um sistema de equações lineares pode ser representado na forma a.x = b onde x é um vetor de variáveis.

Com o comando LinearSolve[a, b] , MATHEMATICA resolve o sistema a.x = b e nos dá como resposta o vetor solução x.

Exemplo:

```
In[1]:= m = {{1,5},{2,1}}

Out[1] = {{1,5}, {2,1}}

In[2]:= LinearSolve[m, {a,b}]

Out[2] = {-( a/9 - 5b/9), -(-2a/9 +b/9)}
```

Com o comando NullSpace[a] , MATHEMATICA resolve o sistema linear homogêneo a.x = 0 e produz uma lista de vetores que formam uma base para o espaço solução do tal sistema.

Exemplo:

```
In[1]:= m = {{1,2}, {1,2}}

Out[1]= {{1,2}, {1,2}}

In[2]:= NullSpace[m, {a,b}]

Out[2]= {{-2, 1}}
```

Nota: Podemos também resolver sistemas de equações lineares com o comando Solve conforme apresentado abaixo.

Exemplo:

```
In[1]:= Solve[{X+5*Y+3*Z==19, X+2*Y+2*Z==10, X-9*Y+13*Z==-35},{X,Y,Z}] e [Enter]
```

$$\text{Out[1]}=\{\{\ X \to \frac{56}{11}\ ,\ Y \to \frac{36}{11}\ ,\ Z \to -\left(\frac{9}{11}\right)\ \}\}$$

6.7 Equações Diferenciais

Uma importante característica da versão 2.0 do MATHEMATICA é a melhoria da habilidade para manipulação de equações diferenciais com as funções DSolve e NDSolve. DSolve é um resolvedor simbólico de equações diferenciais que aceita como entrada uma equação diferencial ordinária - uma equação com uma função desconhecida de uma variável y(x) e suas derivadas - que tenta encontrar a função desconhecida.

Nota: Na versão 1.2 já existe o pacote ODE que tenta resolver equações diferenciais ordinárias de primeira e segunda ordens. Vamos mostrar dois exemplos:

In[1]:= DSolve[y"[x] == 3/(4 x^2) y[x], y[x], x]

Out[1]= { { y[x] -> $\frac{2C[1]+x^2C[2]-C[1]^2C[2]}{2Sqrt[x]}$ } }

DSolve pode ser utilizado para vários fins; por exemplo resolve problemas com condições de contorno, aceita sistemas de equações diferenciais lineares, etc. Vamos exemplificar a solução de um sistema com o uso do DSolve; e sugerimos a consulta do manual para outros esclarecimentos.

In[2]:= DSolve[{ x'[t] == 3 x[t] - 4 y[t] + E^t,
y'[t] == x[t] - y[t] - E^t }, { x[t], y[t] }, t]

Out[2]= { { x[t] -> $E^t(t + 3t^2 + 2C[1] + C[2] + 2tC[2])$,
y[t] -> $\frac{E^t(-2t+3t^2+2C[1]+2tC[2])}{2}$ } }

6.8 Formatos de Entrada e Saída

MATHEMATICA usualmente imprime expressões numa aproximação da notação padrão usada em matemática; aqui temos uma expressão impressa na saída padrão do sistema que envolve expoentes e frações.

In[1] := x^3/5 + 1/y^2 e [Enter] produz,

Out[1] = $\frac{x^3}{5} + y^{-2}$

A função InputForm permite imprimir expressões num formato apropriado para entradas subsequentes no MATHEMATICA.

In[2] := InputForm[%] e [Enter] produz,

Out[2] / / InputForm = x ^3 / 5 + y ^(-2)

Vamos apresentar um resumo das saídas TEX, C e FORTRAN do sistema; no manual do MATHEMATICA são fornecidos maiores detalhes sobre essas e outras formas de saídas, como fazer arquivos externos com elas e outros esclarecimentos.

Sintaxe para TEX, C e FORTRAN:

TeXForm ⟶ entrada no formato TEX

CForm ⟶ entrada no formato da linguagem C

FortranForm ⟶ entrada no formato FORTRAN

Aqui temos novamente uma expressão na forma de saída padrão do MATHEMATICA.

In[3] := x^5/7 + y^3 - Log[y]

Out[3] = $\frac{x^5}{7} + y^3$ - Log[y]

Aqui a mesma expressão numa forma conveniente de entrada no TEX.

In[4] := TeXForm[%]

Out[4] / / TeXForm = { { { x ^5 } } \ over 7 } + { y ^3 } -\ log(y)

Aqui a mesma expressão na forma da linguagem C; objetos como Power são definidos no arquivo mdefs.h

In[5] := CForm[%]

Out[5] / / CForm = Power (x, 5)/7 + Power(y, 3) - Log(y)

Aqui a mesma expressão na forma FORTRAN

In[6] := FortranForm[%]

Out[6] / / FortranForm = x**5 / 7 + y**3 - Log(y)

Quando uma expressão de saída muito extensa é gerada pelo MATHEMATICA, e só se quer a priori ter uma idéia geral da estrutura da expressão, usar a função Short que fornece essa visão segundo a sintaxe.

Short[expr] ⟶ imprime uma linha de saída da expressão

Short[expr, n] ⟶ imprime a linha n da saída da expressão

Por exemplo, In[1] := t = Expand[(1 + x + y) ^12] gera uma expressão extensa; se toda a expressão fosse impressa ocuparia 16 linhas.

A entrada abaixo fornece a saída; o << 86 >> indica que 86 termos foram omitidos.

In[2] := Short[t] e [Enter] produz

Out[2] / / Short = $1 + 12x + 66x^2 +$ << 86 >> $+y^{12}$

Aqui temos uma versão de t em três linhas onde mais termos são apresentados.

In[3] := Short[t, 3]

Out[3] / / Short =
$1 + 12x + 66x^2 + 220x^3 + 495x^4 + 792x^5 +$
$924x^6 + 792x^7 + 495x^8 + 220x^9 +$ << 76 >> +
$132xy^{10} + 66x^2y^{10} + 12y^{11} + y^{12}$

Short pode ser usado com outros formatos de saída, InputForm, por exemplo;

In[4] := Short[InputForm[t]]

Out[4] / / Short = 1 + 12 * x + 66 * x^2+ << 86 >> +12 * x * y^11 + y^12

É possível gerar saídas em forma tabular segundo a sintaxe abaixo.

TableForm[expr] ⟶ imprime em forma tabular

MatrixForm[expr] ⟶ imprime como uma matriz

A função TableForm fornece uma lista de saída das expressões em colunas

In[5] := TableForm[{ 5!, 10!, 15! }

```
                          120
                          3628800
Out[5] / / TableForm =    1307674368000
```

Aqui temos uma matriz com saída impressa como uma lista de listas no formato padrão de saída do MATHEMATICA.

In[6] := m = Table[i j ^ 2, { i, 3 }, { j, 3 }]

Out[6] = { { 1, 4, 9 }, { 2, 8, 18 }, { 3, 12, 27 } }

MatrixForm fornece um array onde cada elemento é colocado em uma área de mesma largura e altura.

In[7] := MatrixForm[m]

```
                          1   4   9
                          2   8   18
Out[7] // MatrixForm =    3   12  27
```

Nota : Existem outras formas que permitem até que o investigador defina seus próprios formatos de saída, maiores detalhes no manual do software.

6.9 Tratamento Gráfico

Uma das partes mais cuidadosamente bem feitas e bem programadas em sistemas de computação algébrica é a parte de gráficos matemáticos do MATHEMATICA, pois em nenhum outro pacote são vistos gráficos de uma e duas variáveis tão bem feitos e tão simples de serem dados como entrada para o usuário.

Em qualquer discussão sobre o MATHEMATICA temos que destacar sua habilidade gráfica. MATHEMATICA conhece como fazer gráficos com singularidades, portanto se pedirmos ao software a entrada abaixo teremos:

In[1] := Plot[1/Sin[x], {x, -5, 5}] e [Enter] produz,

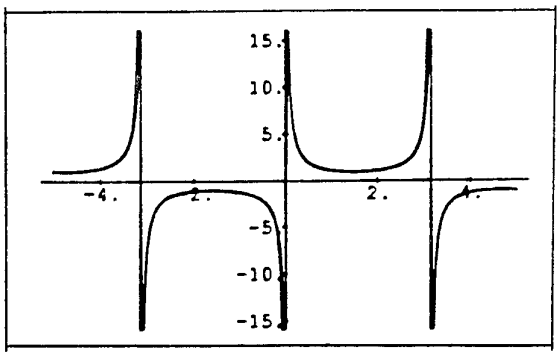


FIGURA 3

Juntamente com um aviso (não mostrado aqui) sobre os problemas que o software teve ao tratar de singularidades.

Nota : Aqui também está embutida uma lição sobre sintaxe, pois colchetes são geralmente usados para se definir listas. Muitos comandos de plotagem esperam uma lista de 3 elementos como segundo argumento. O primeiro item é a variável a ser plotada e as duas finais são os extremos do domínio.

No exemplo Plot[Sqrt[4 - x ^ 2] / x ^ 2 / (1 - x ^ 2), {x, -3, 3}] e [Enter], tente entender mais sobre a sintaxe, veja como o software fez uma escolha sobre os limites dos eixos verticais, observe como o gráfico se comporta próximo das assíntotas , compare a resolução com outros softwares e tire suas conclusões.

No segundo exemplo, o comportamento do gráfico, próximo a singularidade, novamente é mostrado de modo bastante claro.

In[2] := Plot3D[(x ^ 2 - y ^ 2) / (x ^ 2 + y ^ 2), {x, -1, 1}, {y, -1, 1}] e [Enter] produz,

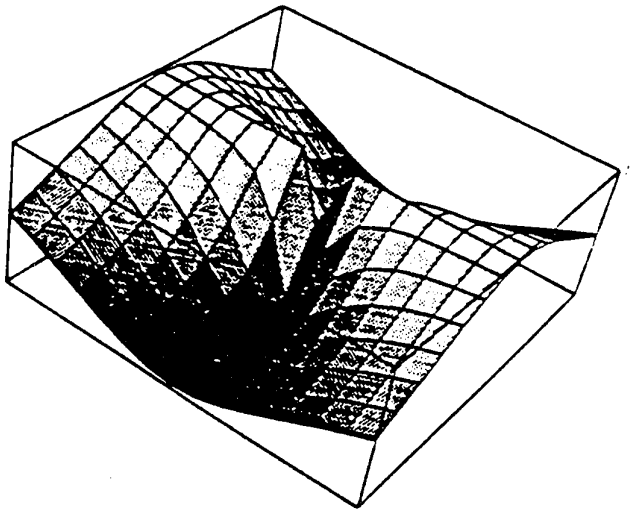


FIGURA 4

Note que é possível se ver uma linha radical de pontos limite, acima e abaixo da origem. Além disso, a caixa delimitadora e a superfície sombreada ajudam o investigador a perceber a orientação da superfície no espaço tridimensional e a posição relativa dos pedaços da superfície (pedaços iluminados mais altos do que pedaços mais escuros).

Desta forma os designers fizeram um bom trabalho ao evitar os perigos que todos os programas gráficos de funções tropeçam mais cedo ou mais tarde - escala defaut desapropriada na direção vertical, falta de detalhes quando a variável dependente muda rapidamente, e também de deixar o investigador desorientado no espaço.

Rotinas de plotagem do MATHEMATICA trabalham melhor pois seus algoritmos de plotagem adaptativos plotam mais pontos quando a variável dependente muda rapidamente.

Realmente não precisamos mais enfatizar como o MATHEMATICA pode ser usado em pesquisa, ensino etc. com seu potencial gráfico. Os resultados dizem mais que dezenas de palavras elogiosas. Para finalizar a apresentação de plotagens feitas pelo sistema vamos mostrar uma figura que a nosso ver não seria feita com tanta perfeição por qualquer outro software até hoje implementado.

In[3] := DensityPlot[Sin[1/(x y)], {x, -2, 2}, {y, -2, 2}, PlotPoints->500, Mesh->False];

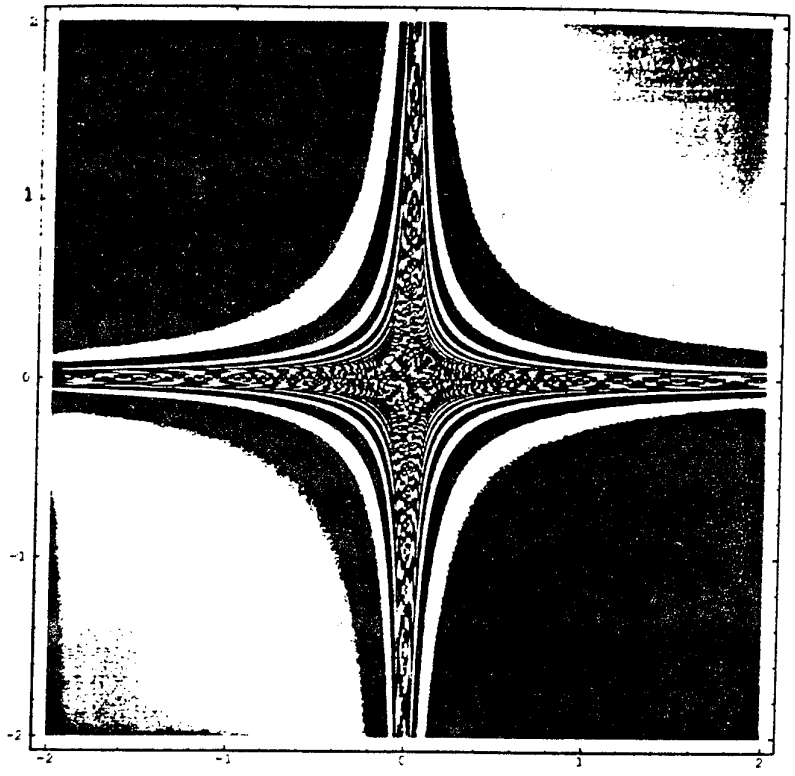


FIGURA 5

Além disso, a velocidade na construção de gráficos - mesmo bastante elaborados - é geralmente boa.

Nota : Se o gráfico não tiver tão bom quanto se deseja, os comandos Plot e Plot3D tem um grande número de opções, uma característica que a maioria dos comandos do MATHEMATICA traz para o usuário.

Por exemplo, no Plot podem ser mudados o número de pontos exemplo a serem plotados, especificar o domínio da variável dependente e mudar o raio de aspecto.

Pode-se também acrescentar alguns detalhes como labels, marcas do riscado e composição. Em Plot3D, é possível mudar a iluminação simulada, "colorido" sobre a superfície e o ponto de vista do qual a plotagem é mostrada.

Para funções de duas variáveis, MATHEMATICA tem também comandos para desenhar contornos e "densidade". Uma plotagem de densidade é como uma família de contornos, exceto que sombreia o plano melhor que desenha curvas nele e pontos altos são indicados por sombreamento brilhante, o que permite distinguir máximos locais, mínimos locais e pontos de sela.

Geralmente, projeções de superfícies em um plano não são tão úteis na saída de aspectos cruciais de comportamentos de funções para gráficos planares de contornos e pontos críticos.

Existe um número de outros comandos gráficos, mas os mais intrigantes são uma coleção de rotinas de animação acrescentadas ultimamente.

Primeiramente usar um comando em um dos pacotes externos para criar uma família de objetos gráficos relatados, tais como, gráficos de funções, contornos ou densidades; então selecionar um item do menu que anima o grupo todo colocando suas saídas numa rápida sucessão.

A animação é tão rápida que se tem uma ilusão de movimento suave mesmo para objetos bem complicados como superfícies.

Por exemplo, é possível demonstrar vários vistos manifestados de uma superfície ao criar uma animação que nos dê a ilusão de "voar" pela superfície. Fly By é um comando em um pacote externo de animação.

Ainda não foram mencionados como gráficos são gerados no MATHEMATICA o que é também digno de distinção; eles são gerados pelo PostScript, uma linguagem de descrição de página gráfica, o que é facilmente observável, pois ao se dar um comando gráfico, uma pausa indica que o código PostScript está sendo gerado e por conseguinte a figura começa também a ser gerada. Uma vantagem do uso do PostScript é que impressoras modernas, principalmente à laser, reconhecem PostScript e geram saída de alta qualidade, outra é que o MATHEMATICA permite que o investigador modifique o código PostScript gerado e escreva seu próprio código.

Por exemplo, pode ser dado um comando para gerar um gráfico, ser selecionada a célula contendo o gráfico (em cima do qual muda imediatamente de figura para o código PostScript), ser editado o código e novamente ser reformatada a célula para obter seu novo gráfico. Existe uma desvantagem para esse método de geração de gráficos, ele não permite que a própria pessoa faça usos iterativos.

Para finalizar queremos sugerir aos investigadores que façam suas análises pessoais com relação ao tratamento gráfico do sistema, inclusive comparando com outros softwares, pois certamente concordarão com o autor desse estudo que diz que as plotagens e gráficos do MATHEMATICA são os melhores já produzidos por softwares de computação algébrica até o momento.

6.10 Programando no MATHEMATICA

Embora do ponto de vista de alguns matemáticos, as linguagens de programação dos sistemas de computação algébrica ainda não tenham atingido um nível desejável; uns, inclusive como J. R. Kundera, chegam a afirmar que "linguagens de programação matemáticas são horríveis de serem usadas"; enquanto nenhum incrível gênio de design de softwares dessa área criar uma linguagem de programação que nos permita comunicar matemática para um computador num

estilo próximo ao que escrevemos matemática, temos que aprender a escrever matemática num estilo que seja reconhecido pelos sistemas de computação algébrica, e especialmente sobre a linguagem de programação do MATHEMATICA, podemos fazer algumas colocações, que para a nossa época parecem pertinentes.

A linguagem do MATHEMATICA é bastante similar às linguagens de programação de outros sistemas, contém todos os aspectos usuais de qualquer linguagem de alto nível tais como, declarações de decisões, declarações de loopings, operações de input-output extensivas etc., e é suficientemente rica, de modo que em contraste com as idéias colocadas no início do parágrafo, achamos que é até agradável resolver problemas com a linguagem do MATHEMATICA, nunca se esquecendo que "a prática leva á perfeição".

Se cometermos um erro em algum comando, basta editar os caracteres incorretos com a ajuda do mouse e então pressionar [Enter], para a execução do comando correto. Se quisermos executar o mesmo comando várias vezes, com variações, podemos fazer isso editando, podemos mesmo reformatar uma expressão de uma forma de output bidimensional para uma forma de input linear, e converter isso simplesmente editando. Outro fato agradável é que quando o editor detecta um erro, ele não emite apenas um beep, ele move o cursor para próximo do erro.

Aqui estão alguns exemplos, embora sempre recomendemos fortemente o uso do manual do MATHEMATICA para um maior conhecimento de outras habilidades na programação do sistema.

Todo símbolo começa com uma letra maiúscula e não traz outras letras maiúsculas, a menos que esta seja a concatenação de duas palavras - prudentemente, símbolos são escritos fora como palavras completas tais como Integrate, Simplify e WorkingPrecision, portanto não devemos conjecturar com abreviações -, existem quatro tipos de separadores, ( ), [ ], { } e [[ ]], e nenhum deles pode ser usado de modo alternado com os outros.

Existem pelo menos cinco espécies de sinais de igualdade, embora somente = , := e = = sejam usados geralmente. As quatro proposições seguintes, embora parecidas na sua aparência tem significados diferentes;

f[x] = x ^2 ; f[x] : = x ^2

f[x_] = x ^2 : f[x_] := x ^2

O sublinhado do lado do x indica uma variável muda; os dois pontos com o igual indicam que a computação do valor f e é para ser referida até quando necessária. Podem-se definir um novo comando escrevendo-se um procedimento que o defina, então ele é executado como se tivesse um comando interno construído (built-in). Podem-se definir novas funções tanto explicitamente quanto recursivamente. Pode-se salvar suas próprias definições e esconder o que não for preciso e também entrelacar elas dentro da sua seção corrente.

Por exemplo, as duas proposições seguintes definem a função fatorial recursivamente:

f[0] = 1

f[n_] : = n f(n - 1)

Aqui f[0] é computado imediatamente como tendo o valor 1 - já que igual foi usado -, mas outros valores de f não são computados até serem necessários - já que dois pontos com igual foi usado para eles -, além disso n é uma variável muda, já que está seguida de um sublinhado (chamado uma "lacuna").

A linguagem de programação tem também uma coleção extensiva de construções de loops e condicionais emprestados de outras linguagens, de modo que é possível encontrar muitas coisas que facilitam o seu uso, os pacotes externos que existem no MATHEMATICA, são de fato agendas de funções e comandos escritos nessa linguagem de programação.

Os comandos e a linguagem de programação se encaixam bem, juntas, porque são atualmente uma linguagem; uma linguagem árvore-estruturada parecida com LISP que é especialmente semelhante àquelas construídas em outros sistemas de computação algébrica. A linguagem do MATHEMATICA parece particularmente bem definida e geral, e é baseada em três conceitos fundamentais:

Expressões, Regras de Transformações e Modelos.

Todos os objetos no MATHEMATICA são chamados expressões e são definidos por uma sintaxe precisa. Todas as ações no MATHEMATICA podem ser descritas como regras de transformação que mudam uma expressão de uma forma para outra, e, finalmente regras de transformação produzem não somente expressões individuais, mas classes inteiras de expressões, chamadas modelos.

Por exemplo, na segunda linha de fatorial da expressão acima, n_ é um modelo que espera qualquer inteiro positivo, e a equação é uma transformação agindo nesse modelo mas cuja execução é deferida até ser requerida, entretanto, um modelo pode ser uma classe de expressões de qualquer tipo, não apenas de expressões numéricas, daí a linguagem ser uma linguagem natural para sistemas de computação algébrica e foi explicitamente desenhada para produzir classes gerais de expressões simbólicas por regras de transformação gerais.

Não existem tipos requeridos de dados na linguagem do MATHEMATICA, mas tipos de dados como Integer, Real e String são reconhecidos e tomados na computação. Particularmente podemos declarar uma variável modelo para ser um tipo de dado particular.

Por exemplo, na definição de fatorial acima podemos restringir n para ser um inteiro digitando n_ Integer no lugar de n_, podemos também restringir n para ser um inteiro positivo, o que é feito acrescentando-se uma condição de lado à definição.

MATHEMATICA tem uma ótima documentação, tanto on-line quanto impressa, de dentro do programa é possível serem encontradas todas as opções que um comando permita, o significado de cada escolha de menu e as convenções que governam a célula da agenda, entretanto não se pode perguntar pela sintaxe de um comando ou de um exemplo tabalhado on-line.

Mensagens de erro são normalmente claras e específicas embora para um usuário iniciante, o software não forneça mensagens muito detalhadas. Como já dissemos anteriormente o manual

do MATHEMATICA é bastante didático e abrangente, além disso, o guia de referência descreve cada comando, tem um capítulo rápido em termos tutoriais, um capítulo de pesquisa aprofundado e capítulos mais avançados sobre a estrutura da linguagem, os comandos matemáticos disponíveis e técnicas para se escrever programas.

Conceitos e convenções são explicados em sentenças completas organizadas em seções e parágrafos coerentes. Existem vários exemplos anotados que são realçados em fontes distintas. Sumários de convenções são apresentados em vídeo reverso num último plano cinza e, finalmente o índice é bastante completo e usa uma variedade de fontes que transmitem informações extras.

Programas no MATHEMATICA podem ser escritos como em linguagem C. MATHEMATICA é um interpretador. Os programas podem ser rodados tão rapidamente quanto o tempo que se levou para digitá-los. O exemplo seguinte define uma função f que constrói uma tabela dos n primeiros números primos.

In[1] := f[n_] := Table[Prime[i], { i, n }] que pode ser usada imediatamente.

In[2] := f[12] e [Enter] produz,

Out[2] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 }

O exemplo seguinte é um programa simples que gera e expande produtos.

In[3] := exprod[n_] := Expand[Product[x+i, { i, 1, n }]]

In[4] := exprod[4] e [Enter] produz,

Out[4] = $24 + 50x + 35x^2 + 10x^3 + x^4$

6.11 Considerações Complementares

Ao apresentarmos esse pequeno estudo sobre o MATHEMATICA, temos que colocar para o leitor que o software apresentou em suas primeiras versões e ainda apresenta em menor escala alguns problemas e bugs. Alguns desses problemas seriam; o software tem um pequeno número de rotinas de computação algébrica quando comparado com o MACSYMA e com o MAPLE; seu integrador simbólico manuseia um conjunto muito menor de funções; o sistema, até a versão 1.2 não tinha um resolvedor simbólico de E.D.O.'s e também não opera com séries infinitas ou séries finitas de tamanho indeterminado simbolicamente.

Outra possível deficiência da computação algébrica do MATHEMATICA é a lentidão de certas rotinas chave, quando comparadas com diferentes pacotes em diferentes máquinas. Um outro aspecto da computação algébrica do sistema, que também deixa a desejar, é a qualidade das suas rotinas de simplificação. O comando Integrate, por exemplo, ás vezes apresenta resultados longos e complicados, para pegar um caso extremo, Integrate[x ^4 * Sqrt[1 - x^2], x] produziu uma soma de 4 termos extensos, enquanto que outros sistemas e até métodos manuais produ-

zem resultados mais compactos. É claro que se poderia aplicar o comando Simplify, o qual embora lento, faz um bom trabalho para muitas expressões algébricas, mas quando esse trabalho não é feito, as outras rotinas para transformar expressões algébricas não são suficientemente poderosas para ajudar. Por exemplo, derivando o resultado do comando Integrate acima foi apresentada uma soma de 7 termos, que ao ser simplificado forneceu um termo diferente do original.

Uma pequena parte do sistema - A integração - produziu muitas respostas erradas. Em testes randômicos feitos por vários pesquisadores (J. Barwise, W. Kahan, E. A. Herman) entre outros, o sistema foi criticado violentamente; E. A. Herman em 1988 afirmou inclusive, que integrais impróprias são calculadas incorretamente mais vezes do que calculadas corretamente no artigo "Computers and Mathematics". Algumas integrais claramente convergentes, foram calculadas "indeterminadas", "infinito complexo" e várias integrais divergentes calculadas como uma constante; existiram também resultados tais como $\int_0^1 \frac{1}{x} dx$ calculadas como Log[x]. Algumas integrais indefinidas, como o exemplo trivial, $\int \frac{1}{(x^3+x)}$ dx também apresentaram problemas que são mostrados no mesmo artigo acima citado.

Não estamos falando só de fatos negativos e depreciativos do software, de fato, o MATHEMATICA tem algumas construções muito gerais e poderosas. Expressões no MATHEMATICA têm uma estrutura detalhada e uma sintaxe precisa, as regras de transformação para expressões são ricas em variedade e poder. A interface gráfica do sistema auxilia bastante até o investigador mais exigente, que pode selecionar parte da figura com a ajuda do mouse, trabalhar em parte delas etc. Outro ponto forte do sistema é sua rica coleção de funções inteiras, a qual inclui o totient de Euler e funções de Moebius, o símbolo de Jacobi e números de Bernoulli e Stirling.

Na verdade, muito poucos pacotes foram tão poderosos nos seus "primeiros anos"; e como o sistema está em pleno desenvolvimento para aprimorar suas capacidades e facilidades, tanto a nível de software quanto de hardware - inclusive bem receptivo à criticas segundo vários autores - é de se esperar que muito brevemente novas versões do software já atendam aos desejos implícitos nas críticas, e coloquem o MATHEMATICA no topo dos sistemas de computação algébrica. A última versão do software, distribuída a partir de 1991 - MATHEMATICA versão 2.0 - já apresenta uma série de alterações tanto a nível de acréscimos de capacidades como por exemplo funções para DSolve e NDSolve para manipular equações diferenciais, mudanças para agilizar e otimizar computações numéricas e outras melhorias no software , quanto a nível de suporte e divulgação, e esperamos que muitas das críticas apresentadas já tenham se tornado antigas e obsoletas, pois o MATHEMATICA realmente está aquinhoando aficcionados e usuários constantes e a cada versão são observados progressos pela comunidade científica em todo o mundo.

6.12 Exercícios

1. Integrar a expressão $\frac{1-Cos[t].x}{x^2-2.Cos[t].x+1}$ com t variando de -π a π , se possível tanto na versão 1.1 quanto na versão 1.2, e na versão 2.0; dar o valor de $\frac{1}{2}$ para x. Tentar fazer esse mesmo trabalho em outros softwares de computação algébrica.

2. Verificar se o sistema faz a derivada da função f definida por um conjunto de regras tais como:
$$x = \begin{cases} x^2 + x & \text{se } x > 0 \\ x & \text{se } x \leq 0 \end{cases}$$

3. Fazer um programa no MATHEMATICA que forneça uma lista na forma tabular dos fatoriais dos números de 1 a 15.

4. Gerar a matriz 5x5 definida pela lei$a_{ij} = \frac{1}{i+j+1}$ e calcular sua inversa, seu determinante e seus autovalores.

5. Construir a plotagem tri-dimensional z $= \frac{Cos(x^2+y^2)}{\frac{\sqrt{(2)}}{2}(x^2+y^2)}$ para x $\in$ [-5, 5] e y $\in$ [-5, 5].

6.13 Publicações Pertinentes

A seguir apresentamos uma pequena lista de publicações sobre o software que certamente virão a contribuir para o aprofundamento do investigador em vários detalhes não enfocados ou mostrados de modo pouco abrangente no nosso texto e que vem sendo largamente utilizado pele comunidade dos usuários do MATHEMATICA.

1. WOLFRAM S., GRAYSON D., MAEDER R., CETJIN H., GRAY T., OMOHUNDRO S., BALLMAN D., KEIPER J., MATHEMATICA, A System for Doing Mathematics by Computer - User Manual - Addison-Wesley Publishing Company, Inc., U.S.A., 1988.

2. MAEDER R., Programming in MATHEMATICA, Addison-Wesley, U.S.A., 1989.

3. DAVIS C., MATHEMATICA, A Program for Various Workstations and Personal Computers, Artigo, (Review), The Mathematical Intelligencer Vol 12 no.2, Springer-Verlag, U.S.A., 1990, 69 - 74.

4. HERMAN E. A., Review of MATHEMATICA (também com discusssões de BARWISE J., UHL J. J. e ZORN P., Notices of the A.M.S. 35, no.9, Nov. 1988, 1334 - 1349.

5. FATEMAN R. J., A Review of MATHEMATICA, Artigo, Computer Science Division, University of California, Berkeley, U.S.A., 1991.

6. The MATHEMATICA Journal, Addison-Wesley,Publishing Company, U.S.A., 1991.

6.14 Informações Adicionais

Para informações adicionais e complementares sobre a aquisição do software, instalação no equipamento disponível do usuário, problemas e bugs do sistema e outras comunicações necessárias com as distribuidoras do software, fornecemos a seguir um dos seus endereços. Outros endereços são dados no capítulo 8 do nosso texto.

Wolfram Research, Inc.

P. O. Box 6059

Champaign, IL 61821-9902

Illinois, U.S.A.

Observação : Existe um grupo de usuários do MATHEMATICA, chamado MathGoup que planeja escrever boletins, estudar pacotes, publicar um jornal sobre o sistema e organizar uma conferência anual para os investigadores com interesse em intercâmbios sobre o sistema. Interessados em conhecer ou participar do MathGroup contactar:
Steven Christensen
MathGroup
National Center for Supercomputing Applications
258 Computing Applications Building
University of Illinois
605 East Springfield Avenue
Champaign, IL 61820
Endereço Eletrônico:

steve@ncsa.uiuc.edu.

Capítulo 7

MAPLE

7.1 Introdução

MAPLE é um sistema para computação matemática simbólica desenvolvido na Universidade de Waterloo. O projeto foi primeiramente discutido em dezembro de 1980, quando Keith Geddes pediu uma reunião do seu departamento para colocar algumas idéias que tinha sobre computação simbólica. O principal objetivo naquela época, era o de se fazer um software para utilização em ensino universitário, e após alguns encontros e maior detalhamento do sistema alguns pesquisadores concluíram que poderia vir a ser criado um poderoso instrumento para uso acadêmico.

Gaston Gonnet e Keith Geddes começaram então a criar o design e a implementação a partir de um sistema piloto em que Gaston havia trabalhado. Em fevereiro de 1981, MAPLE já começava a ser usado como instrumento auxiliar por estudantes de graduação.

Em setembro deste mesmo ano, Bruce Char também entrou para o "team". Em 1982 MAPLE estava sendo usado para pesquisas em outros centros científicos e, em 1984, embora os criadores do sistema ainda não o considerassem um produto acabado, cerca de 50 universidades já utilizavam o software.

A versão 3.3 distribuida através da WATCOM, continuou sendo aprimorada e desenvolvida provendo um número maior de facilidades ao sistema, o que inclusive continua ocorrendo até o presente momento.

MAPLE consiste de um sistema núcleo, escrito em linguagem C, e uma biblioteca de procedimentos matemáticos. O núcleo atende vantagens básicas tais como aritmética de polinômios e a programação própria do MAPLE usando programação do tipo PASCAL, linguagem de comando usadas durante as seções interativas do MAPLE e para programação usual. A biblioteca

matemática é escrita nessa linguagem.

O núcleo da versão 4.2 ocupa aproximadamente 167 Kbytes em um VAX 11/780. Funções da biblioteca são automaticamente carregadas de modo que as exigências de memória cresçam até uma medida que depende das aplicações do usuário. O código da biblioteca consiste de mais de 1400 arquivos, isso ocuparia cerca de 2.4 megabytes se fosse totalmente carregada na memória principal. Existem ainda 1100 Kbytes de documentação on-line.

O "team" que desenvolveu o MAPLE é composto pelos pesquisadores abaixo sob a liderança dos três primeiros.

Keith O. Geddes, Doutor pela Universidade de Toronto,(1973). Seus trabalhos profissionais incluem pesquisas em algoritmos numéricos; ele era analista numérico até começar a trabalhar no projeto MAPLE. É Co-Diretor do Grupo de Computação Simbólica e Professor Associado do Departamento de Ciência da Computação da Universidade de Waterloo.

Gaston H. Gonnet, Doutor pela Universidade de Waterloo. Trabalhou como Professor Visitante na PUC/RJ, tendo retornado a Universidade de Waterloo em 1978. É Co-Diretor do Grupo de Computação Simbólica e Professor Associado do Departamento de Ciência da Computação.

Bruce W. Char, Doutor pela Universidade da California, Berkeley. Trabalhou com MACSYMA no Doutorado e completou seu primeiro ano de Pós-Doutorado no Argonney National Laboratory em Argonne, Illinois. É membro do Grupo de Computação Algébrica e Professor Assistente do Departamento de Ciência da computação desde 1981.

Greg J. Fee, Programador de Sistemas e Aplicações do Grupo de Computação Simbólica.

Michael B. Monaghan, Doutor pela Universidade de Waterloo, Professor do Departamento de Ciência da Computação e membro do Grupo de Computação Simbólica.

Benton L. Leong, Programador Senior de Sistemas do Grupo de Computação Simbólica.

A WATERLOO MAPLE SOFTWARE distribui o MAPLE (4.2 e 4.3) para os seguintes sistemas:

Amdhal S/370 family/UTS V 1.1; Amiga 1000 / Amiga DOS; Apollo 3xxx, 4xxx/Unix 4.2; ATT 7300,3B1/Unix V2S; Convex/Convex Unix; Cray 2/Unicos; Gould NP1/Unix; Gould DN/9082/Unix; IBM RT/AIX 1.1; MacIntosh Plus, SE/MPW 1.OB2; MacIntosh Plus,SE,II/Find (Exceto para a América do Norte); Masscomp 5xxx/Unix; NCR Tower/Unix; PCS Cadmus/Munix; Pyramid 9600/Unix; Sequent Balance/Dynix; Sequent Simetry/Dynix; Sun 3/Sun Unix; Sun 4/Sun Unix e Sun 386i Sun Unix.

MAPLE está também disponivel para os sistemas: DEC Microvax/Ultrix 32m; DEC VAX 11, 8xxx, 3xxx/4.2 ou 4.2 BSD Unix; DEC VAX 11, 8xxx, 3xxx/VMS 4.4 ou 4.5; IBM 43xx/VM/SP CMS Rel4 e IBM S370 family/Amdhal UTS/V 1.1, que necessitam de informações adicionais para instalação.

7.2 O Porquê da Utilidade do MAPLE

- A versão utilizada no nosso estudo foi a do MAPLE - 4.2 instalado em um computador ATARI no CBPF (Centro Brasileiro de Pesquisas Físicas); já a partir de 1991, vêm sendo comercializadas as versões 5.0, 5.1, etc. para PC's 386 o que certamente virá a ampliar fantasticamente o universo de usuários do software.
- MAPLE usa notação matemática padrão e é provido de alguns mecanismos de programação que fazem a seleção dos melhores algoritmos de acordo com o tipo de problema proposto.
- MAPLE tem uma larga cobertura para resolução de problemas matemáticos com mais de 1000 funções matemáticas implementadas e o seu poder de resolução e confiabilidade, pode ser avaliado pelos problemas submetidos sistematicamente pelo American Mathematical Montly.
- O suporte do produto e de desenvolvimento e apoio ao usuário é considerado pelos investigadores de um modo geral, comunidade acadêmica usuária, industrias etc. como um dos melhores; esse trabalho é feito pela Waterloo MAPLE Software e pelo Grupo de Computação Simbólica de Waterloo.
- No MAPLE a poderosa integração de Risch pode achar uma integral na forma fechada ou provar que ela não existe.
- MAPLE pode produzir arquivos LATEX, gerar códigos FORTRAN e mesmo otimizá-los para computações numéricas mais rápidas.
- Códigos fonte na linguagem MAPLE de alto nível são incluídos para cada rotina de biblioteca; embora o MAPLE tenha sido escrito em linguagem C, não é necessário por parte do investigador conhecer essa linguagem para resolver problemas razoavelmente complexos com poucas horas de uso.

7.3 Capacidades e Usos do MAPLE

7.3.1 Capacidades do MAPLE

Apresentamos a seguir uma pequena lista das capacidades do sistema; voltamos a sugerir a utilização do manual do usuário do MAPLE ou do livro MAPLE - First Leaves para conhecimento de outras habilidades do software.

Álgebra Simbólica	Manipulação Matricial
Aritmética com Frações e Polinômios	Manipulação Tensorial
Divergentes	Polinômios de Chebyschev, Legendre e Laguerre
Eliminação Gaussiana	Plotagens
Equações Diferenciais Ordinárias	Problemas em Otimização Linear
Funções de Bessel	Rotacional
Funções Gamma	Sistemas de Equações Lineares e Não Lineares
Jacobianos	Séries de Taylor
Laplacianos	Séries de Fourier
Teoria dos Grupos	Transformadas de Laplace
Teoria dos Números	Variedade de Funções da Trigonometria
Somatórios	Programação

7.3.2 Usos do MAPLE

MAPLE é bastante utilizado no meio acadêmico em centenas de Universidades e Centros Científicos, e da divulgação do seu uso em artigos, publicações em revistas especializadas e conferências de pesquisadores podemos apresentar algumas áreas do conhecimento onde foi empregado.

Análise dos Elementos Finitos	Imagens Para Pesquisas Médicas
Desenho a Laser	Matemática Avançada
Desenho de Cascos de Navio	Mecânica Estrutural
Dinâmica dos Fluidos	Ondas de Choque Submarinas
Dinâmica de Hélices de Helicópteros	Programação Linear
Econometria	Química
Engenharia Elétrica	Robótica
Estatística	Sistemas de Orientação de Satélites
Física	Teoria Espectral

7.4 Apresentando o MAPLE

7.4.1 Preliminares

Esse parágrafo apresenta uma pequena demonstração sobre a utilização do MAPLE e tem também a intenção de ser intelegível para o usuário que não está familiarizado com o sistema.

Como em todos os sistemas apresentados nesse estudo, sugerimos que a leitura dessa apresentação seja feita em frente a um computador que rode o MAPLE, de modo que o usuário possa executar nossas sugestões iniciais, fazendo inclusive alguns exercícios elementares encontrados no final do parágrafo, que certamente servirão como passo inicial para futuros trabalhos.

Para entrar no MAPLE na maioria das máquinas, digitar:

maple e [Enter]

Será então fornecida a mensagem a seguir com o logotipo do MAPLE, ou outras similares em versões mais recentes.

```
                    Watnum at Univ. of Waterloo

MAPLE               Version 4.x - - - Oct 1987

                    For on-line help, type help();
```

e o MAPLE apresentará seu prompt, como abaixo, aguardando algo para ser computado.

```
>
```

nesse prompt digitar o pedido, seguido de ponto e vírgula e [Enter]; serão então fornecidas "as respostas" - os números e fórmulas geradas pelo software de acordo com a entrada do usuário.

7.4.2 Símbolos e Chaves Especiais

- Linhas começando com o símbolo > são linhas de entrada; as saídas são apresentadas por linhas deslocadas para a direita, de responsabilidade do sistema.
- A menos que seja indicado, todos os comandos do MAPLE são escritos em letras minúsculas.
- As constantes matemáticas e(base natural de log), i(raiz quadrada de -1), π e a constante γ de Euler são respectivamente referidas por exp(1) ou E, I, Pi e gamma.
- Para terminarmos uma seção do MAPLE bater quit e [Enter]; o MAPLE ainda imprime uma mensagem final dando um sumário dos recursos usados durante aquela secção.
- Para informar ao MAPLE que seu pedido de entrada está completo não esquecer o ponto e vírgula antes de teclar [Enter].

- O símbolo ' " ' se refere a expressão digitada na linha anterior para não ser necessário digitá-la novamente em algum pedido de cálculo.
- Ao terminarmos de digitar uma expressão com ' : ', MAPLE retorna o rótulo de entrada para algum pedido de cálculo sobre aquela expressão.
- O símbolo " # " é usado para indicar comentários.
- Para se obter uma descrição mais detalhada de um comando especificado, digitar help (comando); e [Enter].

7.5 Entrada das Expressões

A entrada de expressões no MAPLE é feita de modo simples que pouco difere de outros softwares de computação algébrica, em especial os estudados nesse trabalho.

As operações aritméticas comuns no MAPLE são:

+ , -	adição, subtração
* , /	multiplicação, divisão
^ ou * *	exponenciação
!	fatorial

Algumas funções matemáticas do MAPLE são:

f:= < y / x >;	define uma função f para a expressão y com respeito a variável x, onde y é uma expressão em termos de x. Se não houver ambiguidade como variável, usar f:= < expressão > ;como por exemplo f:= < sin(x) + x^4 >; Se tiver mais do que uma variável, usar f:= < expressão / x, y, ... > como por exemplo; f:= < a * x^2 + b * y^3 / x, y >;
y:= expr;	Atribui a expressão algébrica expr, para y.
exp	Função exponencial
ln ou log	logaritmo natural
abs	valor absoluto
sqrt	raiz quadrada
sin, cos, tan, sec, csc, cot	funções trigonométricas
arcsin, arccos, arctan, arcsec, arccsc, arccot	funções trigonométricas inversas
sinh, cosh, tanh sech, csch, coth	funções hiperbólicas
arcsinh, arccosh arcsech, arccsch, arctanh, arccoth	funções hiperbólicas inversas.

7.6 Sintaxe de Algumas Capacidades do MAPLE

Mostraremos agora a sintaxe correta de entrada de pedidos para algumas capacidades do sistema; obviamente nos permitimos escolher um número bem restrito de habilidades do software, embora apresentados com clareza e correção. Voltamos a enfatizar que para um trabalho profundo e abrangente o uso do manual do usuário do MAPLE juntamente com o livro First Leaves: A Tutorial Introduction to MAPLE é indispensável. As páginas seguintes contém o transcrito de seções interativas do sistema semelhantes as que são vistas nos terminais.

7.6.1 Séries de Taylor e Séries Assintóticas

Sintaxe:

taylor(expressão, variável = ponto de expansão, ordem de truncamento)

asympt(expressão, variável, ordem de truncamento)

Nota : O(x^n) é acrescentado no final das séries se os termos até x^{n-1} não são a expressão completa. Como uma regra de manuseio, o O(x^n) sempre aparece, exceto quando a expressão é um polinômio de grau menor do que n na variável das séries. O valor de Order controla a ordem de truncamento(i.e. o valor n acima)usada durante o cálculo; esse valor é inicialmente setado como 6, mas pode ser alterado como exemplificado abaixo:

Um terceiro parâmetro especifica a ordem para anular o defaut 6.

> t1 := taylor (sin(x ^ 2 + x), x = 0 , 4); e [Enter] produz,

t1 := $x + x^2 + \frac{1}{6}x^3 + O(x^4)$

Séries assintóticas com x = infinity(ordem de defaut O(x^(-6))

> asympt(exp(1/x) / (1+x ^ 3), x); e [Enter] produz

$\frac{1}{x^3} + \frac{1}{x^4} + 1/2\frac{1}{x^5} + O(\frac{1}{x^6})$

7.6.2 Limites de Funções Reais e Complexas

Sintaxe:

limit(expr, x = a); → limite real bidirecional

limit(expr, x = a, direção); → limite direcional

Exemplos:

> limit(sin(1/x^2), x = 0); e [Enter] produz

-1 .. 1

> q :=(x^2 - 1) / (3 * x - 5): e [Enter] produz,

q:= $\frac{x^2-1}{3x-5}$

Nesse rótulo, digitar:

> limit(sqrt(q), x = 1); e [Enter] produz,

0

Um outro exemplo de limite direcional:

> limit(sin(x)/sqrt(x), x = 0 , right);

0

> limit(I * x^2, x = infinity, complex);

infinity

7.6.3 Diferenciação

Sintaxe:

> diff(cos(ln(x ^ 2)) - exp(sin(x)),x); e [Enter] produz,

$-2\frac{sin(ln(x\hat{}2))}{x}$ - cos(x) exp(sin(x))

Derivadas parciais de ordem mais altas podem ser obtidas por diff com uma sequência de variáveis. O exemplo a seguir calcula a derivada parcial $\frac{\partial^2}{\partial t \partial s}$ de uma expressão em s e t.

Calcular uma derivada parcial de segunda ordem de uma expressão.

> (t^2 - s) / (s^3 - 1) e [Enter] produz,

$\frac{t^2-s}{s^3-1}$

> diff(", t, s); e [Enter] produz,

$-6\frac{ts^2}{(s^3-1)^2}$

7.6.4 Integração Definida e Indefinida

MAPLE tentará calcular uma integral definida ou indefinida quando o comando int for dado, entretanto, ao invés da diferenciação, onde o código de computação de "como derivar" é direto, não é garantido o sucesso de todas as integrações. Quando o software falha ao resolver um problema de integração, o pedido para o cálculo, eventualmente um pouco simplificado, é deixado como a resposta simbólica.

Sintaxe:

int(integrando, variável de integração) → integração indefinida

int(integrando, variável de integração = inf .. sup) → integração definida

Exemplos:

> int(x ^ 2 * cos(x)^2, x); e [Enter] produz

$x^2(1/2cos(x)sin(x) + 1/2x) + 1/2xcos(x)^2 - 1/4cos(x)sin(x) - 1/4x - 1/3x^3$

> int((x^3) * sqrt(x^2 - a^2), x = a .. 2 * a);e [Enter] produz,

$14/5\ a^5\ 3^{1/2}$

> int(ln(x) , x = 0 .. 1); e [Enter] produz;

-1

Algumas vezes MAPLE não calcula uma solução em uma forma-fechada.

> int(1 / ((x - 1) ^ 2/3), x = 0 .. 3); [Enter] produz,

$\int_0^3 \frac{1}{(x-1)(2/3)} dx$

Nota: A "resposta" é apenas uma outra versão impressa do comando original.

Observação : A integral $\int_0^1 \sqrt{\frac{1+x}{1-x}} dx$ cujo valor é $\frac{\pi}{2} + 1$ quando calculada pelo Maple recebeu na versão estudada o valor incorreto $-\frac{\pi}{2} - 1$.

7.6.5 Somatórios

Sintaxe:

sum(somando, variável de indexação = inf .. sup) → somatório definido

sum(somando, variável de indexação) → somatório indefinido

Exemplos:

> sum(i^2, i = 1 .. j); e [Enter] produz,

$1/3((j+1)^3$ - $1/2(j+1)^2$ +1/6j + 1/6

> sum(3^i, i = a .. b);e [Enter] produz,

$1/23^{b+1} - 1/23^a$

Como na integração, o resultado requerido é o valor retornado como "resposta", quando nenhuma fórmula pode ser encontrada.

> sum(5/(5-i), i = 0 .. n); e [Enter] produz,

$$\sum_{i=0}^{n} 5\frac{1}{5-i}$$

Nota : Se digitarmos no rótulo de entrada lprint(conforme abaixo), a última expressão é retornada.

> lprint(''); e [Enter] produz

sum(5/(5-i), i = 0 .. n)

> sum(x ^ 4nl * (-1) ^ l , l = 1 .. (q-5)/4); e [Enter] produz;

$$\frac{-(-x^{4n})^{(q-1)/4}-x^{4n}}{x^{4n}+1}$$

7.6.6 Solução de Equações e Sistemas de Equações Algébricas

MAPLE usa solve para dar soluções de equações algébricas ou sistemas de equações algébricas, o valor de solve pode ser um número ou uma expressão, dependendo da equação, e se solve achar mais do que uma solução, as apresentará separadas por vírgulas. No caso de soluções múltiplas, solve retorna uma expressão sequência de soluções, discutidas mais detalhadamente no FIRST LEAVES - A tutorial introduction to MAPLE.

Sintaxe:

solve(equação)

solve(equação variável) ⟶ resolve uma equação para um termo desconhecido.

solve(conjunto de equações, conjunto de variáveis) ⟶ resolve um conjunto de equações para um conjunto de termos desconhecidos.

Exemplos:

> solve(7* x + 25 = 10 * x + 1); e [Enter] produz,

8

> solve(ln(x^2 - 1) = a, x); e [Enter] produz,

$-1/2(4\ exp(a)+4)^{1/2}$, $1/2(4\ exp(a)+4)^{1/2}$

Quando uma expressão é dada ao invés de uma equação, um " = 0 " é implicitamente assumido.

> solve(a^2 - b, b); e [Enter] produz,

a^2

> solve ({ m = n+2, n-m = p }, { m, n }); e [Enter] produz,

Warning: no solutions found

A solução de um sistema de equações lineares pode ser pedida também através de:

> eqnset: = { x + y = a, b * x - 4 / 5 * y = p }; e [Enter] produz,

eqnset: = { x + y = a, bx - 4/5 y = k }

> varset: = { x, y }; e [Enter] produz,

varset: = { x, y }

Nota : 'varset' significa o conjunto de variáveis.

> solutionset: = solve(eqnset, varset); e [Enter] produz,

solutionset: = { y = 5 $\frac{ab-p}{5b+4}$, x = $\frac{5p+4a}{5b+4}$ }

Nota : Existem vários outros comandos e funções internas construídas para a disposição das soluções de acordo com a entrada do investigador, e para quando não existem soluções ou apenas algumas soluções são encontradas, consultar o manual do software.

> solve(x = x + 3 ,x); e [Enter] produz,

Warning: no solutions found.

> solve(sin(x * Pi)=0, x); e [Enter] produz,

0

Maple também resolve sistemas de equações não lineares, embora não com total eficiência quanto a que resolve as lineares.

7.6.7 Vetores e Matrizes

A biblioteca do MAPLE contém vários pacotes, cada pacote é uma coleção de funções. Essa seção discute um pacote, o linalg, que é uma coleção de funções que trabalham com os tipos particulares de arrays usados pelo sistema para representar vetores e matrizes.

As funções linalg usam uma convenção onde um vetor é um array da forma array(1..n) e uma matriz é um array da forma array(1..m, 1..n).

Nota : with(linalg) é o comando que lê todas as funções do pacote linalg.

Apresentamos agora uma lista parcial das funções do linalg.

MRM e help(linalg) trazem mais detalhes sobre cada função da álgebra linear e como elas trabalham.

A tabela abaixo descreve os nomes das operações comuns de vetores e matrizes.

OPERAÇÃO	NOTAÇÃO MAPLE
u + v (vetor)	add(u, v) ou evalm(u + v)
A + B (matriz)	add(A, B) ou evalm(A + B)
AB	multiply(A, B) ou evalm(A & * B)
Av	multiply(A, v) ou evalm(A & * B)
u.v	dotprod(u, v)
u x v	crossprod(u, v)
A^{-1}	inverse(A) ou evalm(1/A)
det A	det (A)
tr A	trace(A)
adj A	adjoint(A)
Solve Ax = b	linsolve(A, b)
Acha os autovalores de A	eigenvals(A)
Número de componentes em um vetor v	vecdim(v)
Número de colunas em uma matriz A	coldim(A)
Número de linhas em uma matriz A	sowdim(A)

Define u como sendo um vetor de $\mathcal{R}^4$.
> u:= array([5 - t,t,9,t ^ 2]);

u:= array(1..4,

[5 - t,t,9,t ^ 2]
)
Carrega o pacote de álgebra linear(a função with seta definições de funções no pacote "linalg")
> with(linalg);

Warning: new definition for trace

E são também apresentadas várias capacidades do linalg.

Nota : Após entrarmos com o with, podemos usar, por exemplo, "det" ao invés de "linalg[det]". Os elementos de uma matriz são especificados linha por linha na função array e são impressos também linha por linha.

Define B como sendo uma matriz 3 x 2 e A como uma matriz 2 x 3.
> B:= array([[1, 2], [3, 5], [4, 8]]);
B:= array(1..3, 1..2,

[1, 2]

[3, 5]

[4, 8]
)

> A:= array([[1, 4, 1],[0, 7, 0]]);

```
A:= array(1..2, 1..3,
[1, 4, 1]
[0, 7, 0]
)
> rowdim(B);
3
> coldim(B);
2
> multiply(A, B);
[17, 30]
[21, 35]
> multiply(B, A);
[1, 18, 1]
[3, 47, 3]
[4, 72, 4]
> vectdim(u);
3
# Define C como sendo uma matriz 2 x 2
> C:= array([[1, 2], [3, t]]);
C:= array(1..2, 1..2,
[1, 2]
[3, t]
)
> det(C);
t - 6
# Acha os autovalores de C, formando o polinômio característico.
> band([-lambda], 2);
array(Sparse, 1..2, 1..2,
[-lambda, 0]
[0, -lambda]
)
> add(C, ");
```

array(1..2, 1..2,

[1 - lambda, 2]

[3, t - lambda]

)

> det(");

t - lambda - lambda t + $lambda^2$ - 6

> solve(", lambda);

$1/2 + 1/2$ t + $1/2\ (25 - 2t + t^2)^{1/2}$, $1/2 + 1/2$t - $1/2(25 - 2t + t^2)^{1/2}$

Nota : Podemos também usar as facilidades internas construídas para autovalores da matriz C.

> eigenvals(C);

$1/2 + 1/2$ t + $1/2\ (25 - 2t + t^2)^{1/2}$, $1/2 + 1/2$t - $1/2(25 - 2t + t^2)^{1/2}$

MAPLE pode criar vários tipos de matrizes especiais, tais como, matriz de Vandermonde, matriz de Hilbert, etc...

> vandermonde([u, v,w]);

array(1..3, 1..3,

[1, u, u^2]

[1, v, v^2]

[1, w, w^2]

)

> det(");

vw^2 - v^2w - uw^2 + u^2w + uv^2 - u^2v

> factor(");

(-w+u)(v-w)(-u+v)

> H:= hilbert(4);

H:= array(symmetric, 1..4, 1..4,

[1, 1/2, 1/3, 1/4]

[1/2, 1/3, 1/4, 1/5]

[1/3, 1/4, 1/5, 1/6]

[1/4, 1/5, 1/6, 1/7]
)

> HI:= inverse(H);
HI:= array(symmetric, 1..4, 1..4,

[16, -120, 240, -140]

[-120, 1200, -2700, 1680]

[240, -2700, 6480, -4200]

[-140, 1680, -4200, 2800]

MAPLE calcula o Jacobiano de uma função vetorial.
> D:= jacobian([sin(x+y)/(1-cos(x+y)), cos(x+y)/(1+sin(x+y))], [x, y]);
D:= array(1..2, 1..2,

$$\left[\frac{cos(x+y)}{1-cos(x+y)} - \frac{sin(x+y)^2}{(1-cos(x+y))^2}\right) \cdot \frac{cos(x+y)}{1-cos(x+y)} - \frac{sin(x+y)^2}{(1-cos(x-y))^2}\right]$$

$$\left[-\frac{sin(x+y)}{1+sin(x+y)} - \frac{cos(x+y)^2}{(1+sin(x+y))^2}\right) - \frac{sin(x+y)}{1+sin(x+y)} - \frac{cos(x+y)^2}{(1+sin(x+y))^2}\right]$$

)

Nota : Quando se usa o with para ter acesso a uma função de um pacote conveniente, ocasionalmente temos uma definição prévia de uma função interna ou definida pelo investigador.

Por exemplo: trace é definida por defaut para ser o programa para traçar funções; entretanto, após fazer with(linalg), trace é redefinido, significando o traço de uma matriz.

Além do linalg, existem pacotes para estudos para iniciantes em cálculo, estatística, manipulação de séries de potências, teoria dos números, formas diferenciais e polinômios ortogonais, entre outros.

A função with pode ser usada com qualquer um desses outros pacotes de maneira análoga ao seu uso com o linalg.

7.7 Equações diferenciais

Muitas equações diferenciais de primeira ordem, segunda ordem e até de ordens mais elevadas, podem ser resolvidas pelo MAPLE através do dsolve, tanto para soluções em forma fechada, quanto para soluções através de séries de potências; dsolve também tenta resolver sistemas de equações, explicitamente através do método de Laplace ou aproximadamente em termos de séries de potências. Existem várias formas de comandos:

dsolve(equação, depvar(indvar));

dsolve(conjunto de equações e condições iniciais, conjunto de variáveis);

dsolve(conjunto de equações e condições iniciais, conjunto de variáveis, opção)

Existem várias opções para resolver sistemas de equações, usando Laplace ou séries.

Opções para dsolve:

Opção	Funcionalidade
laplace	Uso do método das transformadas de Laplace para resolver uma equação ou um sistema de equações.
Séries	Acha uma solução em séries de potências. O número de termos é ditado pelo valor da variável global.Order, defaut 6
Explicit	Uso de solve para forçar que a solução a ser retornada explicitamente, seja em termos da variável dependente.

Exemplos:

> diff(f(t), t) = 1 + f(t)^2; e [Enter] produz;

$\frac{d}{dt}f(t) = 1 + f(t)^2$

> eq := " ; e [Enter] produz;

eq:= $\frac{d}{dt}f(t) = 1 + f(t)^2$

Essa equação pode ser resolvida diretamente usando o comando dsolve, senão, vejamos:

> dsolve({ eq, f(0) = 0 } , f(t)); e [Enter] produz;

f(t) = tan(t)

Aqui, { eq, f(0) = 0 } representa um conjunto de equações, uma delas é a condição inicial e estamos resolvendo para f(t).

Usa-se dsolve ao invés de solve pois queremos resolver isso como uma equação diferencial ao invés de algebricamente.

Nota : Em alguns casos, uma solução na forma fechada não é possível (ou pode não ser conhecida pelo MAPLE), daí tentarmos outra técnica como uma solução em séries, senão vejamos:

> dsolve({ eq, f(0) = 0 }, f(t) , séries); e [Enter] produz;

$f(t) = t + 1/3t^3 + 2/15t^5$

Vamos agora dar um exemplo de solução de uma equação diferencial de 2^a ordem:

> eqn := diff(y(x), x$2) + 2 * diff(y(x),x) + 2 *y(x) = sin(x);

> sol := dsolve((eqn,y(0)=0,y(6)=0),y(x));

sol :=
y(x)
= 1/5 sin(x) - 2/5 cos(x)
+ 1/5
$\frac{(-sin(6)exp(6)-2cos(6)+2cos(6)exp(6))sin(x)}{sin(6)exp(x)}$
+ 2/5 $\frac{cos(x)}{exp(x)}$

7.8 Programando no MAPLE

No texto vimos como usar o MAPLE como uma calculadora para computações simples, tanto algébricas quanto numéricas; nessa seção mostraremos de maneira suscinta como usar o sistema de maneira mais elaborada. MAPLE, ao contrário de linguagens de computação tais como FORTRAN ou PASCAL, foi desenhada para ser usada interativamente. Coisas simples tais como "dar o nome a para a fórmula $(x + 1)^2$" são dadas através de comandos em uma só linha. Declarações de tipos, BEGIN/END's e outras parafernálias sintáticas muitas vezes são colocadas em programas com um só passo; por incrível que pareça pessoas que já conhecem linguagens tais como FORTRAN, BASIC, ALGOL ou PASCAL às vezes têm dificuldades para escrever programas simples em MAPLE.

Uma diferença significante entre MAPLE e essas outras linguagens é, que MAPLE foi desenhada especificamente para computação algébrica: computação com polinômios, funções trigonométricas, equações etc., e nas linguagens de propósito geral não existem tantos símbolos com significados pré-definidos como na linguagem MAPLE. Por exemplo, MAPLE dá um significado próprio para int, diff, true, false, solve, taylor além de outros tais como if, := e done. Outro modo pelo qual MAPLE difere de outras linguagens de computação convencionais é que uma variável x no MAPLE pode significar apenas um símbolo com nenhum valor particular ou talvez até desconhecido. Combinações de símbolos - tais como adicioná-los - formam alguns dos objetos que podem ser impressos, ou atribuídos de um valor para uma variável de programação.

De maneira distinta a muitas das linguagens de propósito geral, que não foram concebidas para manipulação algébrica interativa, é também perfeitamente razoável no MAPLE se ter símbolos sem valores atribuídos a eles. MAPLE é muito rico em estruturas de dados construídas internamente, além disso, arrays (que são usados para representar vetores e matrizes entre outras coisas), tabelas, listas, conjuntos e sequências como também estruturas matemáticas específicas como somas, produtos e equações são encontradas sistematicamente. Por outro lado MAPLE não tem facilidades para modularidade, subrotinas, programação auto-disciplinada etc.

Embora o MAPLE continue a mudar e crescer de versão para versão, as facilidades para a construção das bibliotecas de matemática, às vezes, têm centenas de linhas de código escritas na própria linguagem de programação do sistema; os capítulos 2 e 3 do livro First Leaves: A Tutorial Introduction to MAPLE apresentam com riqueza de detalhes, os contrastes existentes entre a linguagem de programação do MAPLE e outras linguagens de programação. Apresentaremos a seguir alguns exemplos triviais do uso de procedimentos escritos na linguagem MAPLE.

O exemplo abaixo é um procedimento que usando variáveis locais, acha o máximo entre todos os números dados como argumentos, onde qualquer quantidade de números é aceita. Todos os parâmetros são referidos via args[i], para algum valor de i, portanto não há necessidade de descrever a sequência de parâmetros, apenas quando for um.

```
> maxN := proc ()
>     local result, i;
>     if nargs>0
>     then
>         result := args[1];
>         for i from 2 to nargs do
>             if args[i] > result then result := args[i] fi;
>         od;
>         result;
>     fi;
>end:
```

>maxN(36/11, 1424/550); e [Enter] produz

36/11

>maxN(36/11, 1424/550, 13/2); e [Enter] produz

13/2

O exemplo seguinte calcula os polinômios de Chebyshev de primeira espécie de graus 0 até n e

os armazena em uma tabela.

```
> Chebyshev := proc (n)
>     local p,k;
>     p[0] := 1; p[1] := x;
>     if n < = 1 then RETURN( op(p) )fi ;
>     for k from 2 to n do
>         p[k] := expand( 2*x*p[k-1] - p[k-2] )
>     od;
>     RETURN( op(p) )
> end:
```

> a:= Chebyshev (6);

> a[i] $ i = 0..6; e [Enter] produz

1, x, $2x^2$ - 1, $4x^3$ - 3x, $8x^4$ - $8x^2$ + 1, $16x^5$ - $20x^3$ + 5x, $32x^6 - 48x^4 + 18x^2 - 1$

Para calcularmos o valor dos sete primeiros polinômios de Chebyshev em x = $\frac{1}{2}$ faríamos:

> x:= 1/2:

> a[0], a[1], a[2], a[3], a[4], a[5], a[6] ; e [Enter] produz

> 1, 1/2, -1/2, -1, $-\frac{1}{2}$, $\frac{1}{2}$, 1

7.9 Formatos de Entrada e Saída

MAPLE tem outros estilos de saída diferentes dos usuais até aqui apresentados, usando vírgula ao invés de ponto e vírgula suprimindo a impressão de resultados de comandos interativos.

Podemos imprimir resultados usando a sintaxe de expressões lineares usando lprint, ou dentro de um arquivo via save. Nessa seção discutiremos formatos alternativos diferentes do print ou lprint.

A seguir apresentamos um resumo das saídas FORTRAN, LATEX e EQN.

7.9.1 Saída no Estilo FORTRAN

Sintaxe:

fortran(expr);

Imprime uma expressão usando a sintaxe da linguagem de programação FORTRAN 77; antes de usar esse comando pela primeira vez numa seção do MAPLE, digitar readlid(fortran): para carregar o FORTRAN da biblioteca, se a expressão ocupar mais que uma linha, usar as convenções de continuação do FORTRAN. Existem também sintaxes mais elaboradas, tais como:

fortran(expr, filename = name);

Escreve a saída no estilo FORTRAN no arquivo especificado e;

fortran(expr, optimized);

Toma uma expressão simples e produz uma sequência de enunciados num formato FORTRAN, que reduz o número de operações aritméticas necessárias para calcular uma expressão, através de subexpressões comuns e potências binárias.

Essas duas opções também podem ser combinadas;

fortran(expr, filename = 'f.f', optimized);

Escreverá uma versão otimizada da expressão do arquivo f.f.

Exemplo:

> expr1 := $5 * ln(x)\hat{}5 + 4 * ln(x)\hat{}4 - ln(x)\hat{}2$:

> readlib(fortran):

> fortran(b = expr1);

b = 5 * alog(x) ** 5 + 4 * alog(x) * * 4 - alog(x) * * 2

> fortran(expr1, optimized);

t1 = alog(x)

t2 = t1 * * 2

t3 = t2 ** 2

t8 = 5 * t1 * t3 + 4 * t3 - t2

7.9.2 Saída nos Estilos LATEX e EQN

De modo análogo ao usado no carregamento do FORTRAN, podemos usar as funções definidas no readlib, latex e eqn, que fornecem a saída de expressões num formato útil para entrada em programas de processamento de textos; no exemplo seguinte é gerada uma versão de uma matriz no LATEX e no EQN.

> readlib(latex): readlib(eqn):

```
> a := array([[sqrt((x ^ 2) + 1), binomial(4,n)],
>            [(( x ^ 3) + 1) / sqrt( sin(x) ^ 3 + 2), 49]]);
a := array(1 .. 2, 1 .. 2,
```

$$\begin{bmatrix} (x^2+1)^{1/2} & \text{binomial}(4,n) \\ \dfrac{x^3+1}{(\sin(x)^3+2)^{1/2}} & 49 \end{bmatrix}$$

```
)

> latex(a, latex.out);
> eqn(a);
```

```
left [matrix{ ccol{{{ sqrt{ {"x" sup 2} + 1 }}} above {{ {{ "x" sup 3} + 1}
over{ sqrt{ {sin ^ ( "x" )} sup 3} + 2}}}}} ccol{{{binomial ^ ({8, ~ "n"})}}
above{49}}}right]
```

que também produz ao escrevermos:

```
> latex(a);
```

```
\left[ \begin{array}{ c c } {\sqrt{{{x ^ {2}} + 1}}} & { \frac{{{x ^ {3}} + 1}}
{{ \sqrt{{{{ \sin(x)}^{3}} + 2}}}}} \\ {4 \choose n} & 49 \end{array} \right]
```

Processando esse arquivo com o LATEX, resulta a seguinte saída:

No arquivo latex.out;

$$\begin{bmatrix} (x^2+1)^{1/2} & \frac{x^3+1}{(sin(x)^3+2)^{1/2}} \\ \binom{4}{n} & 49 \end{bmatrix}$$

Nota: Nem o latex nem o eqn tentam resolver o problema de decidir onde partir uma expressão que requer mais de uma linha para ser impressa. O usuário deve manipular os resultados do latex e do eqn via um editor de textos para conseguir partir a expressão da forma desejada.

7.10 Tratamento Gráfico

Nessa versão estudada do MAPLE, plotagens de gráficos de funções matemáticas não são das melhores até mesmo quando comparados com softwares "menores", tais como o DERIVE por exemplo, e fica bastante aquém do MATHEMATICA, o bem mais elaborado na programação dessa capacidade extremamente importante em um sistema de computação algébrica.

O tempo para a geração da figura também foi o maior dentre os softwares acima citados.

A plotagem de funções de valores reais em qualquer intervalo é feita através do comando plot que produz um "gráfico de caracteres" da expressão expr, uma função de variável var no domínio especificado segundo a sintaxe abaixo:

plot(expr, var = domínio)

No sistema existe um suporte completo para plotagens paramétricas. A curva definida por

$$r(t) = [t^2, t]$$

é construída no MAPLE pelo comando,
> plot([t^2, t, t = -3 .. 3]); e [Enter] que produz a plotagem exibida na figura 1.
Neste exemplo, colchetes indicam uma lista de argumentos. Os primeiros dois elementos da lista indicam as coordenadas x e y como uma função do parâmetro t; e o domínio de t é indicado no terceiro argumento.

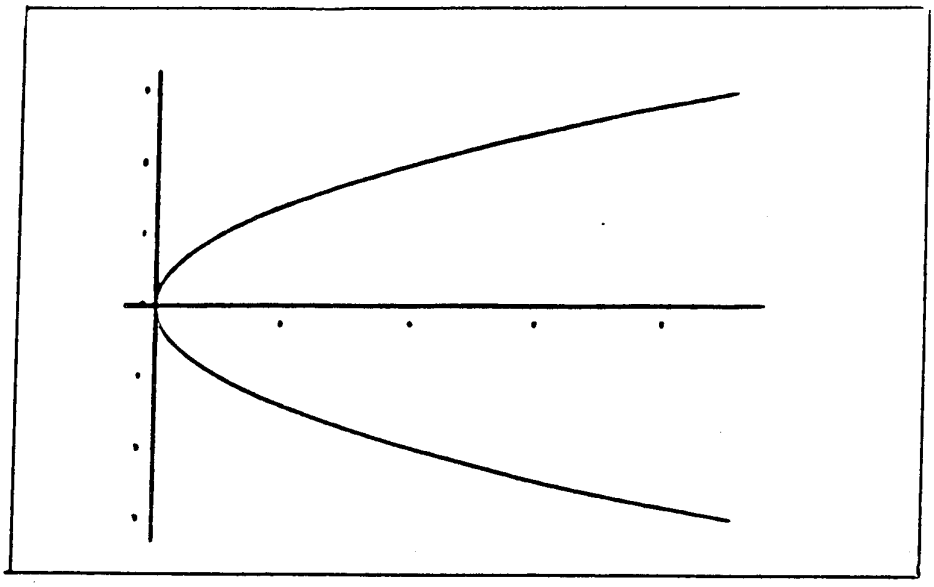


FIGURA 1

No exemplo seguinte, chaves ({ }) são usadas para indicar que um conjunto de expressões em x serão plotadas simultaneamente, como mostrado na figura 2.

> plot({ 4 * sin(x)/(x+1), 8 * sin(x)/(x+1) }, x = 0...4 * Pi);

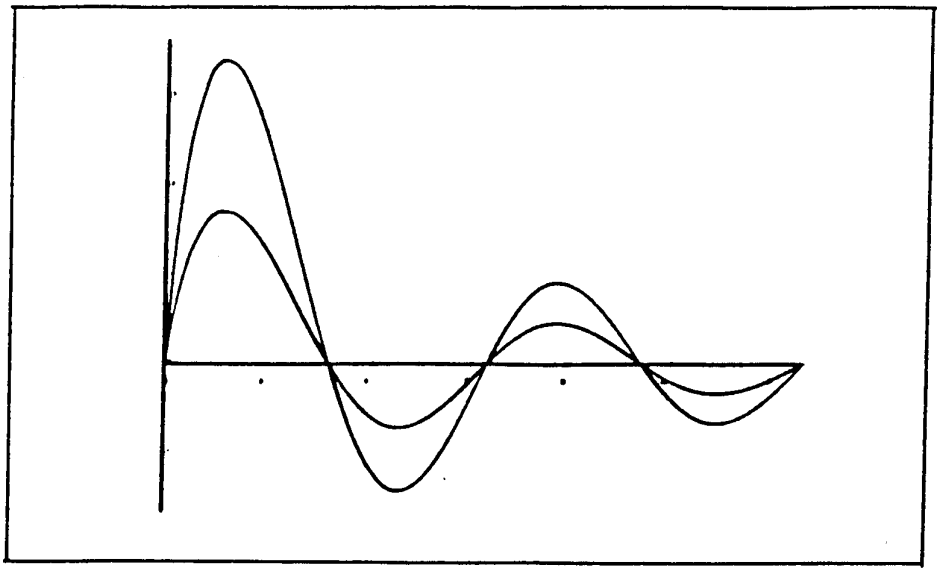


FIGURA 2

Na figura 2 plotamos duas funções em um mesmo gráfico. Em geral, podemos plotar várias funções no mesmo gráfico com a sintaxe abaixo;

plot(conjunto de expressões, var = domínio)

onde cada expressão do conjunto de expressões é uma função de valor real da variável var.
O plot aceita plotagens em várias outras espécies de saídas gráficas; para tal deve-se setar a variável global plotdevice para um dos seguintes valores se quisermos plotagens usando as capacidades especiais de um dos seguintes tipos de saídas:

Valores do plotdevice	
Valor	Finalidade
Char(defaut)	Plotagem de caracteres - para terminais sem suporte gráfico
vt 100	Como plotagens de caracteres, mas usará caracteres gráficos de linhas horizontais do vt 100. Especialmente efetiva no modo de 132 colunas.
tek	Para plotagens em terminais tek tronix 4014
regis	Para plotagens em terminais que aceitem protocolo gráfico REGIS.
postscript	Plotagens serão formatadas para serem impressas em impressoras PostScript compatíveis. A variável global plotoutput deverá ser atribuída um nome de um arquivo antes do comando plot ser dado. er MRM ou help(plot, device) para esclarecimentos.
ln 03	Plotagens serão formatadas para impressão em impressoras DEC LN03. Ver MRM ou help(plot, device) para esclarecimentos.

As facilidades de plotagem do MAPLE podem incluir plotagens de parâmetros, plotagem em coordenadas polares, plotagem de pontos e a saída de um gráfico sem um domínio limitado vertical de valores entre outras coisas. Por defaut, Maple calcula o valor de expr com um pequeno número de pontos igualmente espaçados entre si, no intervalo de plotagem. Esse defaut, como também o método defaut de interpolação podem ser alterados; ver MRM ou digitar o comando help(plot); no MAPLE para maiores detalhes.

7.11 Lista Parcial de Comandos do MAPLE

assign (<eqns>);	Faz múltiplas atribuições. <eqns> é uma equação ou conjunto de equações da forma a = b. As atribuições a:=b são feitas.
coeff(p, x, n);	Extrai o coeficiente de x^n
diff(y, x);	$\frac{dy}{dx}$
diff(y, x $ n);	$\frac{d^n y}{dx^n}$
diff(z, x, y);	$\frac{d^2 y}{dxdy}$
D(y);	(Inicialize batendo readlib(D):) Diferencia implicitamente a expressão ou equação y em termos da variável x.
denom(y);	Denominador da expressão racional y.
numer(y);	numerador da expressão racional (y).
evalf(y);	Calcula y para número de ponto flutuante.
expand(y);	Expande a expressão y.
factor(y);	Fatora a expressão y, (oposto de expand(y)).
int(f, x);	$\int$ f(x)dx
int(f, x = a..b);	$\int_a^b$ f(x)dx
limit(y, x = a);	Calcula $\lim_{x \to a} f(x)$ onde a é real , infinito ou -infinito.
limit(y, x = a , dir);	Calcula $\lim_{x \to a} f(x)$ onde dir é esquerdo, direito, real ou complexo.
normal(y);	Normaliza uma expressão racional y.
simplify(y);	Simplifica a expressão y
simplify(y, exp);	Simplifica expressões exponenciais em y

simplify(y, trig);	Simplifica expressões trigonométricas em y
simplify(y, power);	Simplifica expoentes, exponenciais e logaritmos na expressão y
solve(eqn, x);	Resolve a equação, eqn, para o termo desconhecido x.
[solve(eqn, x)];	Útil se existir mais do que uma solução (coloca as soluções como uma lista do MAPLE)
dsolve(deqn, y(x));	Resolve a equação diferencial deqn para y(x)
dsolve(deqn, y(x), option);	Resolve a equação diferencial deqn, para y(x), usando uma das três escolhas para opção. Explícita - Solução retornada explicitamente em termos da variável dependente. Laplace - Usa transformação de laplace para resolver a equação Séries - Usa métodos de séries para resolver a equação.
dsolve(deqn, y(A) = B, y(x));	Resolve a equação diferencial deqn, para y(x), dando a solução particular satisfazendo y(A) = B.

rsolve(deqn, y(n));	Resolve a equação de diferenças deqn para y(n). Esse comando tem opções similares ao dsolve.
fsolve(eqn, x);	Resolve a equação,eqn, numericamente para o termo desconhecido x.Usualmente, somente uma raiz é encontrada.
subs(x = a, y);	Substitui x = a em y.
taylor(y, x = a, n);	Calcula a série de Taylor truncada para y com respeito a x sobre a, até a ordem n.
convert(tylrseries, polynom);	Converte a série de Taylor, tylrséries, em um polinômio; (isto é, apaga o termo $O(x^n)$).

7.12 Lista Parcial de Comandos de Plotagem do MAPLE

plot(f,low..hl);	Plota f, como uma função da variável independente, no intervalo [low, hi]
plot(f,x = low..hl);	Plota f no intervalo [low, hi] onde a abscissa é x.
plot({ f,g,h },x = a..b,c..d);	Plota as funções f, g, h como funções de x, no intervalo [a, b], e domínio ordenado [c, d].

plot([r,theta,low..hl] ,a..b,c..d, coords = polar);	Plotagem em coordenadas polares com low..hi especificando o domínio do theta, e a..b e c..d especificando os domínios de plotagem horizontal e vertical.
plot([x(t),y(t),low..hl],a..b,c..d);	Plotagem paramétrica com low..hi destacando domínio do parâmetro t, e a..b e c..d especifica os domínios de plotagem horizontal e vertical.
plot(f,a..b,c..d,style=option);	Plota f, com domínio horizontal [a, b] e domínio vertical [c, d]. O domínio é opcional. Escolhas para "options" são: POINT - Plota somente pontos. SPLINE - O default. LINE - Liga pontos plotados com segmentos.
plot(f,a..b,c..d,numpoints=n);	Plota f, com domínios horizontal e vertical [a, b] e [c, d], (domínios são opcionais); numpoi especifica o número de pontos a serem gerados, (default 49 para plotagem infinita e 25 caso contrário).

7.13 Exercícios

1. Resolver o sistema $\begin{cases} 3a+4b-2c+d=-2 \\ a-b+2c+2d=7 \\ 4a-3b+4c-3d=2 \\ -a+b+6c-d=1 \end{cases}$ e checar a validade nas equações originais.

2. Calcular o coeficiente de x^{80} (somente) na expansão da série de Taylor de $\frac{1}{(1-x)(2-x)(3-x)}$ sobre 0.

3. Determinar o traço, o determinante, a inversa, número de colunas e os autovalores da matriz de Hilbert 6 x 6.

4. Resolver a equação diferencial $\frac{d^2}{dx^2}y(x) + 2(\frac{d}{dx}y(x)) + y(x) = x$

5. Resolver as seguintes integrais:
   a) $\int \frac{dx}{(x-1)(2x+1)}$ b) $\int_0^1 \sqrt{4-x^2}dx$

7.14 Publicações Pertinentes

A seguir apresentamos uma pequena lista de publicações sobre o software que certamente contribuirão para o aprofundamento do investigador sobre vários detalhes não enfocados, ou mostrados de modo pouco abrangente no nosso texto, e que vem sendo usado com frequência por usuários do sistema como ferramenta auxiliar.

1. GEDDES K.O., GONNET G.H., MONAGHAN M.B., CHAR B.W., WATT S.M., *MAPLE Reference Manual*, 5th edition, WATCOM Publications Ltd., Waterloo, Ontario, Canada, 1988.

2. GEDDES K.O., GONNET G.H., MONAGHAN M.B., CHAR B.W., FEE G.J., *A Tutorial Introduction to MAPLE*, Journal of Symbolic Computation 2(2), 1986, 179 - 200.

3. GEDDES K.O., GONNET G.H., MONAGHAN M.B., CHAR B.W., FEE G.J., WATT S.M., *On the Design and Performance of the MAPLE System*, in Proceedings of the 1984 MACSYMA User's Conference, University of Waterloo Computer Science Departament Research Report, 1984, 199 - 219.

4. GEDDES K.O., GONNET G.H., CHAR B.W., GENTLEMAN W.M., *The Design of MAPLE: A Compact, Portable, and Powerful Computer Algebra System* , Proceedings of Eurocal '88, Springer-Verlag Lecture Notes in Computer Science no. 162, 1983, 101 - 115.

5. Symbolic Computation Group, *MAPLE: A Sample Interative Session*, Departament of Computer Science, University of Waterloo,Ontario, Canada, 1989.

6. ELLIS W., *MAPLE for the Calculus Student: A Tutorial Brooks* - Cole, U.S.A. , 1989

7.15 Informações Adicionais

Para informações adicionais e complementares sobre particularidades do sistema como por exemplo, efetiva instalação no equipamento disponível, problemas e bugs, e também procedimentos para compra do software, fornecemos a seguir os endereços das distribuidoras do MAPLE.

Waterloo Maple Software

160 Columbia Street West

Waterloo, Ontario, Canada

N2L 3L3

Tel: (519) 747 - 2373 ou (519) 885 - 1211 Ext. 3055

Fax: (519) 747 - 5284

Watcom Products Inc.

415 Phillip Street

Waterloo, Ontario, Canada

N2L 3X2

Telephone : (519) 886 - 3700

Telex : 06 - 955458

Endereços Eletrônicos:

```
wmsi@watmum.uwaterloo.ca
wmsi@watmum.waterloo.edu
wmsi@watmum.watmath.uucp
```

Capítulo 8

CONSIDERAÇÕES GERAIS

8.1 Introdução

Esse capítulo fornece uma visão concisa de alguns dos problemas abordados nos capítulos anteriores, e apresenta comentários sobre novas versões dos sistemas de computação algébrica estudados previamente. São mostradas também tabelas auto-explicativas, abordagens de caráter geral sobre particularidades daqueles softwares, e uma "análise comparativa" com a finalidade de otimizar o trabalho do usuário.

Todos os sistemas estudados neste texto continuam em desenvolvimento, aprimorando versões antigas e fornecendo à comunidade usuária novos programas. Ao encerrarmos o trabalho pelo menos quatro dos cinco sistemas já estavam distribuindo softwares com novas capacidades, e com observação de algumas críticas a eles feitas por pesquisadores, tanto a nível de hardware quanto obviamente a nível de software.

A partir do segundo semestre de 1991 os usuários de computação algébrica já podiam ter acesso ao REDUCE 3.4, ao MATHEMATICA 2.0, ao MAPLE 5.0 e 5.1, e às várias versões do DERIVE 2.0. Sobre novas versões do MACSYMA sugerimos consultar seus distribuidores através dos endereços dados nessa seção.

Essa "análise" que apresentamos a seguir foi feita nas versões fornecidas ao longo do trabalho, e refletem não só o nosso pensamento, mas também o de alguns pesquisadores brasileiros da área. O questionamento e a discussão sobre as "atribuições" e "conceitos" dados nas tabelas serão benvindos, e de forma alguma deixarão de ser considerados como contribuições, visto que, como estamos tratando de uma matéria emergente, colocações distintas só somam ao objetivo do nosso texto.

Outro fato interessante, e que deve ser considerado em qualquer crítica sobre "comparação" de

softwares de computação algébrica, é o perfil da área de quem está fazendo tal comparação, pois com certeza muitas diferenças inerentes a própria formação do autor deixam influências nos seus pontos de vista. Queremos dizer com isso que cada usuário desse tipo de pacote, que conheça pelo menos alguns sistemas, terá uma tendência natural a privilegiar aquele software que melhor atende as suas aspirações acadêmicas e profissionais.

8.2 Sistemas de Computação Algébrica mais Divulgados

Sistema	Ling. Implem.	Ano	C. Científico	Resp. Pesquisas
REDUCE	RLISP	63	U. Stanford	A. C. Hearn
CAMAL	BCPL	64	U. Cambridge	J. P. Fitch
LAM	LISP	68	U. London	I. Frick
ALAM	LISP	68	U. London	R. d'Inverno
CLAM	LISP	70	U. London	R. d'Inverno
SHEEP	LISP	71	U. Stockolm	I. Frick
MACSYMA	MACLISP	71	MIT	J. Moses
SCHOONSHIP	ASSEMBLY	72	CERN	M. Veltman
SCRACTHPAD	VMLISP	75	IBM Research	R. D. Yenks
muMATH	muLISP	80	U. Hawaii	D. Stoutemyer
SMP	C	80	CIT	S. Wolfran / Cole
MAPLE	C	80	U. Waterloo	Simb. Comp. Group
MATHEMATICA	C	83	Wolfran Research	S. Wolfran
DERIVE	muLISP	87	U. Hawaii	Soft Warehouse

8.3 Performance de Algumas Capacidades dos Sistemas

Capacidades	Macsyma	Reduce	Derive	Mathematica	Maple
Substituições	S	S	S	S	S
Simplificações	S	S	S	S	S
Fatorações	S	S	S	S	S
Limites	S		S	S	S
Diferenciação	S	S	S	S	S
Integ. defin.	S	S	S	S	S
Integ. indefin.	S	S		S	S
Séries	S		S	S	S
Séries de Taylor	S		S	S	S
Transf. Laplace	S			S	S
Eq. diferenciais	S			S	S
Fç. Transcend.	S	S	S	S	S
Manip. Tensor.	S				S
Cálculo exterior	S	S		S	S
Oper. com listas	S	S			S
Séries de Fourier	S			S	S
Manip. Matric.	S	S	S	S	S
Somatórios def.	S	S	S	S	S
Somatórios indef.	S				S
Séries de Poisson	S				
Sist. Eq. N. Lin.	S	S		S	S
Trat. Simbólico	S	S	S	S	S
Trat. Numérico	S	S	S	S	S
Trat. Gráfico	S		S	S	S
Help on line	S		S	S	S
Tamanho	4 Mb.	0.7 Mb.	0.2 Mb.	2.5 Mb.	1 Mb.
Interf. de saída	F / C / TEX	F / C	F / P / B	F / C / TEX	F / TEX
Ling. de Implem.	MACLISP	RLISP	muLISP	C	C

Significados: S -> SIM , F -> FORTRAN , P -> PASCAL , B -> BASIC.

8.4 Conceitos sobre Algumas Características dos Sistemas

Características	Macsyma	Reduce	Derive	Mathematica	Maple
Versão	417.100	3.3	1.59	1.2	4.2
Facil. de Utilização	***	***	*****	***	***
Abrangência	*****	****	***	****	****
Manual do Usuário	****	****	***	*****	****
Interf. com Usuário	****	***	*****	****	****
Facil. de Instalação	***	****	*****	****	****
Efic. dos Algoritmos	****	****	****	***	****
Aceit. em Máquinas	***	*****	***	****	***
Tratamento Gráfico	****		****	*****	***
Tratramento Numérico	***	***	**	***	****
Tratamento Simbólico	*****	****	***	***	*****
Interf. Externa	****	***	**	****	****
Desenvolv. Atual	****	****	****	****	****
Bibliog. Complement.	****	*****	***	****	***
Suporte	****	****	***	****	****
Help on Line	****		***	****	****
Tempo de Saída	***	***	***	***	***
Confiabil. na Saída	****	****	***	***	***
Divulgação	****	*****	****	****	****
Programação	***	***		***	***
(Problemas e Bugs)$^{-1}$	****	****	***	**	****
(Tamanho do Sist.)$^{-1}$	***	****	*****	***	****
(Preço do Sist.)$^{-1}$	**	****	*****	***	***
Total	81	76	75	79	81

8.5 Novas Versões dos Sistemas Estudados

8.5.1 REDUCE - Versão 3.4

A mais recente versão do REDUCE, (Versão 3.4) foi apresentada à comunidade em 1991 por Anthony Hearn, e já está sendo distribuída e comercializada. Muitas novas características foram acrescentadas, provendo melhorias significantes ás versões anteriores. Em acreéscimo às

capacidades do REDUCE 3.3 esse novo sistema tem vários aperfeiçoamentos. Entre as maiores capacidades dessa nova versão do sistema podemos destacar:

- Uma biblioteca numérica mais completa, incluindo a habilidade de mudar a precisão de cálculo de ponto flutuante de sem ter que usar a chave "bigfloat"; além disso números com ponto flutuante podem entrar com qualquer precisão.
- Melhorias no *solve*, funções complexas, integração e capacidades de Groebner básicas. Em particular, *solve* pode agora manusear várias outras classes de equações, incluindo equações polinomiais não lineares, resolvidas usando bases de Groebner.
- Expansão e ordenação de funções polinomiais e racionais.
- Substituições em uma larga variedade de formas.
- Simplificações de expressões tanto automaticamente quanto controladas pelo usuário e melhorias nas facilidades de produção de expressões otimizadas, tanto para computações numeéricas quanto simbólicas.
- Maior abrangência e novas facilidades para cálculos com matrizes simbólicas. Um pacote de impressão foi melhorado incluindo um formato bi-dimensional para matrizes.
- Aritmética inteira e real de precisão arbitrária, raízes reais e complexas calculadas com qualquer precisão.
- Facilidades para definições de novas funções e extensão da sintaxe para programação.
- Integração e diferenciação analítica.
- Facilidades para solução de maior variedade de equações algébricas.
- Facilidades para a saída de expressões nos formatos TEX e LATEX.
- Facilidades para geração de programas numéricos otimizados a partir de entrada simbólica.
- Cálculos matriciais de Dirac de interesse em cálculos de Física de Alta Energia. Pacotes adicionais para Teoria Quântica e para Física de Alta Energia.
- Vários pacotes para trabalhar com equações diferenciais ordinárias e parciais. Facilidades para resolução de equações diferenciais de primeira e segunda ordens.
- Melhorias e ampliações nas operações com listas e na geração de expressões em LISP.
- Acréscimo do SUPERCALC - para trabalhos em supersimetria -, aos pacotes especiais existentes (GENTRAN, EXCALC, etc.).

- Facilidades para o uso de WINDOWS para melhoria da performance do sistema.
- Transformadas de Laplace
- Truncamento de potências e cálculos em séries de Taylor.
- Dois pacotes para cálculo vetorial.

Nota: REDUCE 3.4 apresenta uma documentação atualizada, incluindo uma melhoria e expansão no manual do usuário para o formato LATEX, e uma bibliografia contendo cerca de 600 referências de publicações relacionadas com o REDUCE.

Em acréscimo às mudanças no próprio REDUCE, existem várias melhorias no LISP subjacente ao sistema.

- Uma versão do Portable Standard Lisp (PSL) do REDUCE é agora disponível para arquiteturas de mainframes IBM, trocando o interpretador SLISP em linguagem assembly, que não tinha a vantagem de operar em arquiteturas do tipo XA como as últimas PSL, MVS, VM e AIX.
- Suporte PSL para VAX / VMS foi aumentado para permitir o uso fácil de maiores partições de memória.
- A versão baseada no PSL para IBM-PC 386 e compatíveis usando coprocessador pode agora utilizar memória virtual portanto, programas extensos podem rodar efetivamente nesses sistemas de modo mais barato e em larga escala em várias máquinas.
- Um novo LISP, Codemist Standard Lisp (CSL) baseado no ANSI C, vem sendo desenvolvido para substituir o Cambridge Lisp baseado em BCPL. A companhia Codemist distribui também uma versão genérica ANSI C em acréscimo às implementações específicas de máquinas.

Uma tabela de implementações correntes do REDUCE 3.4, com nomes e endereços completos dos distribuidores são fornecidos nos folhetos que acompanham o software. As versões genéricas ANSI C e Common Lisp, requerem alguma experiência com a linguagem para a sua instalação; versões de máquinas específicas tem mais procedimentos exatos de instalação. Uma informação completa de pacotes, incluindo resumo das descrições dos vários LISP usados são obtidas por:

```
REDUCE Secretary
RAND
1700 Main Street
P.O. Box 2138
Santa Monica CA 90407-2138 U.S.A.
```

```
Telephone: +1-213-393-0411 Ext. 7681
Facsimile: +1-213-393-4818
Eletronic Mail: reduce@rand.org
```

Caso o usuário tenha acesso ao endereço eletrônico da INTERNET, informações podem ser obtidas enviando mensagens *send info-package* para *reduce-netlib@rand.org*, *reduce-netlib@can.nl* ou *redlib@elib.zib-berlin.de.*. Uma introdução ao REDUCE mostrando algumas das novas facilidades no REDUCE 3.4, podem também ser obtidas incluindo-se *send intro.tex* em uma linha separada da mensagem. Estas mensagens são respondidas por um servidor automático da biblioteca network do REDUCE. A biblioteca também contém muitos pacotes disponíveis desde a realização do REDUCE 3.4 e tenta consertar todos os bugs descobertos. Maiores informações sobre esta biblioteca, como também instruções de como se ligar ao forum eletrônico do REDUCE podem ser obtidas incluindo-se a palavra *help* em uma linha separada na mensagem acima. Resumindo, informações sobre a aquisição, instalação e a implementação da nova versão do software no equipamento disponível, são obtidas através dos seguintes endereços eletrônicos:

```
Submeter 'send info - package' para

         'reduce-netlib@rand.org'

         'redlib@elib.zib-berlin.de' ou

         'reduce-netlib@can.nl' ou

         'hearn%hilbert@rand.ORG'
```

Nota: Não apresentamos todas as novas capacidades e habilidades da nova versão do sistema nessa seção; para um detalhamento completo sugerimos a leitura do manual do REDUCE 3.4, ou do livro Algebraic Computation with REDUCE dos autores WRIGHT F.J. e MacCallum M.A.H. University of London, U.K., England, 1991.

Implementações do REDUCE correntemente disponíveis

Descrição do Sistema	Distribuidores
versão genérica ANSI C	Codemist(CSL)
versão genérica do Common LISP	Cologne (Common LISP genérico); IBUKI (IBUKI Common LISP)
Acorn Archimedes	Codemist(CSL)
Apollo séries 300-4000	ZIB(PSL) ; Forbs(Tuneup LISP)
Atari 1040ST e Mega	Codemist(CSL)
CDC Cyber 180 NOS/VE	Cologne(PSL)
CDC Cyber 910	ZIB(PSL)
CDC séries 4000	ZIB(PSL)
Convex C100 e C200	ZIB(PSL)
Cray X-MP, Y-MP e 2	ZIB(PSL)
Data General séries AViiON	ZIB(PSL)
DEC DECStation séries 2000, 3000 e 5000	ZIB(PSL)
DEC VAX rodando em VAX/VMS ou Ultrix	ZIB(PSL)
HP 9000/300 e séries 400	Forbs (Tuneup LISP); ZIB(PSL)
Hitachi H2050 e E7300/7500/7700	Forbs (Tuneup LISP)
Mainframes ICL rodando em VME	Codemist(CSL)
ICL DRS6000	Codemist(CSL)
MIPS e compatíveis	ZIB(PSL)

Descrição do Sistema	Distribuidores
PC's IBM - compatíveis baseados nos	CALCODE(UOLISP)
Intel 8088, 8086 e 80286	
PC's IBM - compatíveis baseados no	Codemist(CSL)
Intel 80286 com memória extendida	
PC's IBM - compatíveis baseados nos	CALCODE(PSL); ZIB(PSL)
Intel 80386 e 80486 rodando MS-DOS	
PC's IBM - compatíveis baseados nos	ZIB(PSL)
Intel 80386 e 80486 rodando UNIX	
IBM RISC System/6000	ZIB(PSL)
Arquiteturas de Sistemas IBM séries 370	Cologne(PSL)
e 3090 rodando AIX , MVS ou VM	
NEC EWS séries 4800	Forbs(Tuneup LISP)
NEC PC-9801 hline Silicon Graphics IRIS	B U G (Common LISP) ZIB(PSL)
Sony NEWS	Forbs(Tuneup LISP)
Stardent R2000	ZIB(PSL)
Sun 3	Forbs(Tuneup LISP); ZIB(PSL)
Sun 386i	ZIB(PSL)
Sun 4, SPARCStation 1 e 2 e compatíveis	Forbs(Tuneup LISP); ZIB(PSL)

Distribuidores do REDUCE 3.4

```
BUG :       BUG, Inc.
            31-33, Shimonopporo
            Atsubetsu-ku
            Sapporo 004, JAPAN
            Telefone: +81-11-807-6606
            Facsimile: +81-11-807-6645

CALCODE :   CALCODE Systems
            1057 Amoroso Place
            Venice C    0291, U.S.A.
            Telefone: +1-213-399-7612
            Facsimile: +1-213-399-7612 ent\~{a}o #3 depois do beep.
            E-Mail: calcode%calcode.uucp@rand.org

Codemist    Codemist Limited
            "Alt  Horsecombe Vale
```

Combe Down
Bath BA2 5QR, UNITED KINGDOM
Telefone: +44-225-837430
Facsimile: +44-225-826492

Cologne : Andreas Strotmann
Rechenzentrum
Universitaet zu Koeln
Robert-Koch-Strasse 10
D-500 Koeln 41, GERMANY
Telefone: +49-221-478-5524
E-Mail: reduce@rrz.uni-koeln.de

Forbs : Forbs System Co. Ltd
Kannai JS Building
207 Yamasitachou
Naka-ku
Yokohama 231, JAPAN
Telefone: +81-45-212-5020
Facsimile: +81-45-212-5023

IBUKI : IBUKI
399 Main Street
P.O. Box 1627
Los Altos CA 94022, USA
Telefone: +1-415-961-4996
Facsimile: +1-415-961-8016
E-Mail: reduce@ibuki.com

ZIB : Herbert Melenk
Konrad-Zuse-Zentrum fuer Informationstechnik Berlin (ZIB)
Heilbronner Str. 10
D 1000 Berlin 31, GERMANY
Telefone: +49-30-89604-195
Facsimile: +49-30-89604-125
E-Mail: melenk@sc.zib-berlin.de

8.5.2 MACSYMA

As mais recentes versões do MACSYMA diferem um pouco, em termos de análise apresentada, para os outros softwares de computação algébrica estudados no nosso texto. Não apresentaremos detalhes técnicos, embora os "conceitos", já tenham mostrado ao leitor nossa opinião sobre esse fantástico sistema.

Existem atualmente no mínimo 5 versões derivadas do sistema MACSYMA, portanto sugerimos ao leitor interessado em novos acréscimos e melhorias das novas versões do software, entrar em contato com os endereços abaixo.

1. MACSYMA da Simbolics (TM)
2. Versão do DOE-MACSYMA da Paradigm Associates
3. Versão do DOE-MACSYMA, "Maxima" de Bill Schelter
4. Versão na biblioteca DOE, "Vaxima" de Richard Fateman
5. Versão da Fort Pond Research, "ALJABR"

Nota: 1 , 2 e 3 são implementadas em Common Lisp; 4 e 5 em Franz Lisp. As características (maiores e menores) do sistema são idênticas em todas as versões.

Item 1 $\longrightarrow$ Estudado em detalhes no capítulo 3 do nosso trabalho.
Para maiores esclarecimentos:

```
Ver "Informacoes Adicionais" daquele capitulo.
```

Item 2 $\longrightarrow$ Atualmente a versão da Paradigm tem também uma versão comercial em Common Lisp que ao rodar por exemplo num Macintosh, dispõe de capacidades de plotagem de altíssimo nível.
Para maiores esclarecimentos contactar (em Cambridge MA):

```
LPH@PARADIGM.COM
```

Item 3 $\longrightarrow$ Caso o usuário tenha a licença do DOE MACSYMA, ele pode obter essa versão em rede (NETWORK).

128.83.138.20
rascal.ics.utexas.edu
uma sun-3/280 unix, é o endereço.
Para maiores esclarecimentos contactar:

wfs@nicolas.ma.utexas.edu (Bill Schelter)
wfs@rascal.ics.utexas.edu (Bill Schelter)

Item 4 $\longrightarrow$ Procedimentos para a obtenção de licença do Vaxima na biblioteca DOE podem ser obtidos em rede.
Para maiores esclarecimentos contactar:

fateman@renoir.berkeley.edu (Richard Fateman)
National Energy Software Center (312) 972-7250
Numero de acesso do Vaxima, NESC 9631

Item 5 $\longrightarrow$ Essa versão roda em Macintosh e é a de menor custo. ALJABR custa 200 dolares.
Para maiores esclarecimentos contactar:

odell@bu-it.bu.edu (Jim O'Dell)

8.5.3 DERIVE - Versão 2

A distribuidora do DERIVE apresentou através de seu principal pesquisador David Stoutemyer, no primeiro semestre de 1991, o primeiro membro da família de sistemas da sua nova versão, (versão 2.0) com várias alterações e melhorias. Mostraremos a seguir uma lista das maiores capacidades do DERIVE 2.06, isto significa que no mínimo 6 softwares foram produzidos no ano de 1991 pela Soft Warehouse.

Aritmética

- Aritimética exata para milhares de dígitos.

- Aritimética aproximada para precisão arbitrária.
- + , - , * , / , % (porcentagem) ,^ (exponenciação) , e operadores ! (fatorial).
- Notação racional, decimal e científica.
- Aritimética complexa e infinita.
- Fatoração inteira.
- Ramo de seleção flexível para expoentes fracionários.
- Unidades de conversão métricas.
- (Nova!) Constantes fundamentais da física.

Álgebra

- Simplificação inteligente de fórmulas contendo variáveis.
- Uso de nomes de variáveis em Grego e Latim (Inglês).
- Expansão ou fatoração de funções polinomiais e racionais com respeito a alguma ou a todas as variáveis.
- Colocação de expressões com denominador comum.
- Expansão parcial fracionária de funções racionais.
- Redução de fórmulas complexas à forma retangular.
- Definição de funções matemáticas.
- Atribuição de valores a variáveis.
- Declaração de domínios de variáveis.
- Substituição de valores para variáveis e subexpressões.
- Resolução de equações numericamente para precisão desejada.
- Resoluções algébricas de equações e inequações.
- Resoluções de sistemas de equações lineares e equações individuais quadráticas, cúbicas e quárticas exatamente.

Plotagens 2D

- Plotagem de uma ou mais funções para uma simples variável em coordenadas retangulares ou polares.
- Plotagem de uma ou mais funções paramétricas.
- Capacidade de se mover usando chaves de seta.
- Leitura de coordenadas do cursor.
- Centralização de posição de acordo com o cursor.
- Aumento e diminuição do Zoom.
- Aspecto apropriado para a tela do investigador.
- Seleção de cores de plotagem e de eixos.

Plotagens 3D

- Plotagem de funções de duas variáveis.
- Plotagem wire-frame com linhas escondidas removidas.
- Mudança de posição de observação.
- Zoom da escala de plotagem.
- Centralização opcional automática vertical e de escalas.
- Ajuste do número de linhas de malhas.
- Seleção de cores superiores, inferiores e dos eixos.

Cálculo

- Limites simbólicos finitos e infinitos.
- Derivadas parciais de primeira e n-ésima ordem.

- Aproximações de séries de Taylor e de Fourier.
- Integrais definidas e anti-derivadas na forma fechada.
- (Nova!) Transformadas de Laplace.
- (Nova!) Funções para resolução de equações diferenciais ordinárias de primeira e segunda ordem.

Vetores e Matrizes

- Vetores e Matrizes com elementos simbólicos.
- Produto interno, Produto Vetorial e outros produtos.
- Transpostas, Determinantes, Inversas e Traço.
- Redução de linhas em forma de escalonamento de matrizes.
- Polinômios característicos e autovalores.
- Vetores não comutativos e álgebra matricial.
- Cálculo vetorial, diferencial e integral.

Funções

- Exponenciais (exp , sqrt).
- Logarítmicas (ln , log).
- Trigonométricas (sin , cos , tan , cot , sec , csc).
- Trigonométricas inversas (arcsin , arctan , etc).
- Hiperbólicas (sinh , cosh , tanh , coth , sech , csch).
- Hiperbólicas inversas (arcsinh , arctanh , etc).
- Continuidade por pedaços (abs , sign , max , min , step , chi).
- Variáveis complexas (re , im , conj , phase).

- Probabilidades (fatorial , gamma , permutações , combinações).
- Estatísticas (média , variância , erros , desvio padrão e outros).
- Financeiras (valor presente , valor futuro , pagamentos , períodos de pagamentos , quantias).
- (Nova!) Bessel , Hipergeométricas , Chi-square , Zeta e outras funções especiais.

Interfaces e Menus

- Divisão e/ou sobreposição de fórmulas de álgebra e janelas de plotagem.
- Mobilidade entre janelas com apenas uma chave.
- Saída atrativa de fórmulas matemáticas em 2D.
- Iluminação e extração de sub-expressões.
- Rearranjo ou remoção de expressões.
- Edição de expressões existentes.
- Salvamento e carregamento de expressões em arquivos.
- Geração de arquivos FORTRAN , PASCAL e BASIC.
- Impressão de expressões para uma impressora ou um arquivo.
- Gráficos em CGA , EGA , MCGA , VGA e Hércules.
- Sistema mais claro de Help-on-line.
- Status de saída do sistema de controle de variáveis.
- Execução dos comandos do DOS de dentro do DERIVE.

Programação

- Definição de funções usando recursão ou iteração.
- Programa usando a construção if-then-else.
- Predicados podem incluir operadores lógicos e relacionais.
- Estudo da natureza cíclica e convergência de séries.

8.5.4 MATHEMATICA - Versão 2

A Wolfran Research apresentou através do seu principal responsável Stephen Wolfran, na Conferência do MATHEMATICA no primeiro semestre de 1991 em São Francisco, a Versão 2.0 do sistema. A nova versão traz 283 novas funções, chegando ao total de 843 e inclui melhorias para muitas das funções existentes e, ainda conforme seu criador afirmou na ocasião, em 3 meses já estaria totalmente pronta para viajar o mundo. Entre as características mais significantes podemos citar:

O novo resolvedor simbólico e numérico de equações diferenciais; a função NDSolve acha soluções numéricas para sistemas de equações diferenciais ordinárias, e o resolvedor simbólico DSolve manuseia uma grande variedade de equações diferenciais.

Um compilador para operações numéricas e uma manipulação mais completa de integração simbólica; Integrate inclui uma implementação completa do algoritmo de Risch para integração indefinida e algoritmos para integração definida baseados em funções hipergeométricas generalizadas. Isso significa que são manuseadas mais integrais do que as encontradas em qualquer livro de tabelas. NIntegrate também foi aperfeiçoada e tem mais opções.

Operações numéricas com matrizes também foram aceleradas e rotinas de programação linear foram acrescentadas, a Álgebra Linear numérica se tornou mais eficiente. Novas funções para Álgebra Linear incluem; MatrixPower e MatrixExp (simbólica e numérica) e QRDecomposition e SchurDecomposition (somente numérica).

Várias melhorias nos gráficos são também são acrescentadas; a função GraphicsArray permite ao usuário superpor plotagens tridimensionais diferentes, a função RasterArray permite acrescentar cores ás imagens, o acesso direto ao PostScript permite que as espessuras das linhas e o tamanho dos pontos possam ser especificados absolutamente, uma variedade de novas opções foram aumentadas para os gráficos bi-dimensionais e tri-dimensionais e estes suportam agora perspectiva arbitrária, cores intrínsecas de superfície, plotagens paramétricas de superfícies mais controle sobre os eixos e espaçamentos de contornos e de cor.

O acréscimo de som na Versão 2.0 atraiu sobremaneira a atenção e a imaginação de muitos usuários na conferência; usando a mesma sintaxe das funções Plot, as novas funções Play e ListPlay transformam funções matemáticas e dados em formas modeladas de ondas de som.

A linguagem de programação do MATHEMATICA também foi melhorada; foi introduzida uma coleção de construções que permitem a criação de programas bem mais extensos, uma outra melhoria fundamental na linguagem de programação é dado pelo comando Trace, que prove uma "história" da execução do programa; na realidade uma capacidade poderosa de debugagem foi criada com o Trace. Mais mensagens de ajuda são emitidas quando erros de sintaxe são detectados.

Outra característica da Versão 2.0 são os Help Notebooks que grupam todas as funções do MATHEMATICA, com um texto descritivo completo que permite ao investigador utilizar e editar funções quando necessário.

Funções Trigonométricas, Fatoração, Funções Matemáticas Especiais (relacionadas a distribuições estatísticas, SinIntegral, JacobiZeta e outras), Interação Externa (C, Macintosh, Sistema Unix e outros), Debugagem e Detecção de Erros, Manipulação de Arquivos e Strings também são sinais distintivos de acréscimos e de melhorias no software. O MATHEMATICA certamente continuará em evolução, pois Wolfran é um dos raros homens de ciência e de marketing em ciência pois, pouco depois de ter afirmado na conferência de São Francisco - NDSolve permite aos usuários resolver sistemas de até 50 equações diferenciais numéricas com facilidade - disse também, embora a função para a resolução de equações diferenciais simbolicamente esteja em aperfeiçoamento ainda existem "coisas" a serem feitas.

Nota: Não listamos todas as novas habilidades e capacidades do MATHEMATICA Versão 2.0 nessa secão; sugerimos ao investigador interessado em um maior aprofundamento sobre esse tema, a leitura do THE MATHEMATICA JOURNAL Volume 1, Issue 3 Winter 1991.

8.5.5 MAPLE - Versão V

A distribuidora do MAPLE, Simbolic Computation Group - WATCOM, apresentou no primeiro semestre de 1991 as novas versões do software; MAPLE 5.0, 5.1, etc. Entre as características mais significativas das novas versões podemos destacar:
Essas novas versões do sistema, além dos equipamentos mencionados no capítulo 7, também rodam agora em PC's 386. Isso representa uma melhoria de pêso, pois o MAPLE era um dos poucos softwares do 2^o grupo que ainda não oferecia essa facilidade aos usuários.
As respostas são mais comprimidas do que nas versões anteriores, o que mostra que intensas pesquisas nas rotinas de simplificação foram realizadas.
Várias procedures são escritas na própria linguagem do software; por exemplo, a procedure de derivação tem 60 linhas de código escritas na linguagem MAPLE, que decide se usa a regra da cadeia ou não dependendo do tipo de problema a resolver, e onde há tratamentos distintos para se derivar constantes, somas, multiplicações, expenenciações etc.
Em várias procedures a programação recursiva é usada. Os procedimentos nas novas versões tem nome, são sequências de processos iterativos e na maioria das vezes são feitos com seriam feitos à mão, só que em segundos. A segunda vez que se pede um cálculo normalmente se usa um algoritmo diferente do primeiro que foi usado.
Por exemplo, MAPLE observa a densidade das matrizes para saber que tipo de algoritmo usará; esse é um meio de combinação de algoritmos que é utilizado com bastante frequência pelo software. Autovalores e Autovetores são feitos diretamente e muito mais fáceis de serem entendidos pelo usuário.
As novas versões oferecem muitas facilidades para se trabalhar com séries; Taylor, Laurents etc.
Mais talvez o maior e melhor acréscimo que as novas versões oferecem estão nas plotagens, tanto

em 2D quanto em 3D. Com o menor número de linhas de código já conseguido por um software de computação algébrica para essa capacidade, MAPLE agora oferece gráficos bi-dimensionais e tri-dimensionais de alta qualidade, e onde o investigador pode mudar a cor, alterar a posição de plotagem, usar interativamente, rodar a figura, se aproximar ou se afastar do objeto plotado, mudar o ângulo de visão além de muitas outras possibilidades.

Nota : Obviamente não listamos todas as novas capacidades e habilidades das novas versões do MAPLE nessa seção; um detalhamento completo é mostrado no manual do software e no guia do usuário, que também foram melhorados tanto em termos de impressão, quanto no aspecto didático da apresentação.

8.6 Tópicos em Educação

A cada exposição sobre novas versões dos sistemas de computação algébrica aumenta o número de educadores presentes. A revolução que esse tipo de software vem fazendo no ensino de matemática, e nas ciências de um modo geral, com suas variadas formas de uso em sala de aula é uma realidade da qual não se pode fugir.

A maioria das discussões sobre educação, salienta a responsabilidade dos professores na integração desses sistemas aos currículos do ensino universitário dos primeiros semestres, principalmente em cursos com base matemática; Matemática, Física, Engenharia etc.; alguns acham que primeiras aproximações com esses sistemas deveriam ser dadas em cursos de verão e outros acham que computação algébrica deveria ser uma disciplina obrigatória na graduação daqueles cursos.

Da mesma forma do que quando as calculadoras eletrônicas entraram nas salas de aula, existem resistências de alguns segmentos da comunidade acadêmica com a entrada dos softwares de computação algébrica no ensino universitário, embora mesmo esses, concordem que tais programas darão aos alunos a chance de pensar em conceitos muito mais elevados.

Alguns professores aqui no Brasil, tem usado como instrumento auxiliar de ensino, os sistemas de computação algébrica REDUCE, DERIVE e o MAPLE com bastante sucesso paralelamente aos cursos de Álgebra Linear e Cálculo do primeiros semestres. O sistema MATHEMATICA também vem sendo largamente utilizado e suas capacidades gráficas permitem aos alunos exercitarem seu lado direito do cérebro, responsável pela criatividade, tão bem quanto o lado esquerdo, mais lógico.

O custo desse tipo de software tende a diminuir e com certeza em pouco tempo várias universidades do nosso país contarão com mais essa ferramenta educativa. Falta no entanto mão de obra especializada para esse fim, até para dominar e gerenciar com eficácia as aplicações e

benefícios que os sistemas de computação algébrica podem oferecer em outras áreas do conhecimento.

Muitos autores têm usado esses softwares para corrigir "respostas" em livros de Cálculo, Álgebra Linear, Física etc. Levantamentos feitos em alguns países, constataram que em média 10% das soluções não se apresentavam corretas antes dos "recursos" computacionais terem sido utilizados. Nas publicações mais recentes, essas "respostas" são na grande maioria das vezes, construídas também com o auxílio dessas novas ferramentas tecnológicas.

Em vários artigos frequentemente encontramos conjecturas que não mencionam a computação algébrica envolvida; comentários tais como, "...uma análise extensiva de cálculo da presente solução, permite observar que não existem singularidades presentes..." e também como, ".. uma simples checagem com o computador mostra que A_0 tem 1269 termos, A_1 tem 954 termos e A_3 tem 318 tensores...", não enfatizam que anos de trabalho manual foram suprimidos com o uso desse tipo de software.

Para que comecemos a trabalhar em sistemas de computação algébrica, conhecer suas habilidades específicas, dominar suas sintaxes próprias, entender suas construções algorítimicas e outras capacidades, nos permitimos algumas sugestões; de que sejam criados grupos de trabalho com interesse em Álgebra Simbólica e Computacional, sejam realizados cursos paralelos sobre os sistemas mais divulgados, conferências sobre o tema, na medida do possível se criem laboratórios nas universidades, que seja incluída uma linguagem de computação naqueles cursos citados, como os primeiros passos para um crescimento cultural e tecnológico da área, que naturalmente engloba tantas outras como subconjuntos.

Indo de um pólo ao outro podemos citar por exemplo a Universidade da Australia, que tem mais de 100 MacIntosh só no Departamento de Matemática, onde os alunos saem das salas de aula e complementam seus trabalhos nas máquinas. Sabemos que é bem difícil se chegar a esse estágio de desenvolvimento estando no 3° mundo, mas mesmo assim, dentro das possibilidades, já está sendo feito um trabalho sério em algumas Universidades e Centros científicos do nosso país. No Rio de Janeiro já existem alguns centros de pesquisa que estudam sistemas de computação algébrica, tem grupos formados, organizam cursos, trazem com frequência pesquisadores estrangeiros, promovem workshops da área, realizam seminários, além de outras atividades que são:

C B P F ⟶ Centro Brasileiro de Pesquisas Físicas.

L N C C ⟶ Laboratório Nacional de Computação Científica.

P U C ⟶ Pontifícia Universidade Católica do Rio de Janeiro.

U F F ⟶ Universidade Federal Fluminense.

U F R J ⟶ Universidade Federal do Rio de Janeiro.

Capítulo 9

TÓPICOS SOBRE OUTROS SISTEMAS

Mais do que 60 sistemas de computação algébrica já foram implementados e colocados à disposição da ciência em centenas de máquinas e em vários países nos últimos anos. Desde que as idéias de Kahrimanian e Nolan foram levadas à sério por cientistas e pesquisadores no começo da década de 50, o interesse pela área de softwares de computação algébrica tem aumentado consideravelmente - e pela maneira com que as pesquisas se desenvolvem ainda continuará assim por algum tempo - tanto a nível de desenvolvimento de software quanto de hardware. Apresentamos a seguir outros sistemas, nem sempre escritos em LISP ou C, e não tratados explicitamente nesse trabalho, mas que de qualquer forma são conhecidos e são ou já foram utilizados.

9.1 CAMAL

É um acrônimo de Cambridge Algebra System, desenvolvido na Universidade de Cambridge na linguagem BCPL, a precursora da linguagem C, por John P. Fitch na década de 60, com o objetivo inicial de resolução de problemas de Mecânica Celeste e em Relatividade Geral, e atualmente já desenvolvido para ser um sistema algébrico mais geral. O software é pequeno, essencialmente composto de quatro módulos arranjados hierarquicamente. O módulo chave, residindo no nível mais baixo, é um manipulador de polinômios, no topo deles está o módulo das séries de Fourier, um módulo de funções elementares e finalmente um módulo de manipulação de tensores. O sistema é rápido para o que se propõe, mas modesto em capacidade de memória. Ainda hoje utilizado, embora perdendo espaço para sistemas mais sofisticados, como

por exemplo o desenvolvido pela Divisão de Ciência Espacial da Boeing, principalmente por estar disponível em apenas um tipo de computador, o TITAN.

Maiores esclarecimentos sobre o sistema são encontrados nas referências citadas a seguir:

FITCH J.P., An Algebraic Manipulator. PhD. Thesis, University of Cambridge, 1971.

FITCH J.P., CAMAL User's Manual, 3rd edition, Cambridge Computer Laboratory, 1983.

9.2 SAINT

O programa SAINT, acrônimo de "Symbolic Automatic INTegrator" foi escrito em LISP no MIT e implementado no computador digital, IBM 7090 para resolver integração elementar simbólica.

Maiores detalhes sobre esse sistema podem ser encontrados:

Dissertação de Doutorado de James R. Slagle, 1961, do MIT, e no artigo "A Heuristic Program that solves Simbolic Integration Problems in Freshman Calculus" de Slagle, no livro Computers and Thought da McGraw-Hill.

9.3 Engeli's SYMBAL

Um dos primeiros sistemas conhecidos, desenvolvido por Engeli em 67, 68 e 69. Esse sistema é muito parecido em carater e poder ao REDUCE e ao CAMAL, atualmente quase não é citado em revistas de aplicações científicas.

Uma comparação de SYMBAL com outros sistemas é apresentada no artigo, F. Comput. Phys. de Campbell.

9.4 ALTRAN

Desenvolvido no Bell Telephone Laboratories por McIlroy e Brown a partir do sistema ALPAK. É escrito em FORTRAN IV e roda numa grande variedade de máquinas. Não é um sistema interativo e conta para funcionar, com a apresentação de um programa completo que então obedece; os resultados são consequentemente obtidos na maneira convencional por bath jobs.

Referências sobre ALTRAN e FORMAC são encontradas nas publicações:

Hyde J.P. 1964 Bell Syst. Tech. F. 43 1547-62, em Collins e Griesmer 1966 SYGSAM Bul-

letin No. 4 (New York: Ass. Comput. Mach.), Bonde E. R. e Cundall P. A. 1968, Symbol Manipulations Languages and Tecniques (Amsterdan: North Holland) pp 116-32, Brown W.S. 1969a Bell Telephone Labs. Inc. Report AL-69-1 p. 10 e Xenakis J. 1971 Proc. 2nd Symp. on Symbolic and Algebraic Manipulation ed S.R. Petrick (New York: Ass. Comput. Mach.) pp 105-14.

9.5 A família LAM

LAM é um acrônimo de Lisp Algebraic Manipulator e foi construída por R.A. d'Inverno no King's College, Londres em 1968. Tendo investigado os méritos relativos de várias linguagens, d'Inverno optou por LISP, pois era claro desde o começo que mesmo em pequenos cálculos quando prosseguindo com o computador ou tomando um tempo extremamente grande para completar um cálculo, a memória do computador iria a exaustão com facilidade, e LISP possuia um coletor de lixo que minorava este problema.

LAM foi então implementada no maior e mais rápido computador disponível, o Atlas 1, entretanto ele se tornou extremamente lento pois a versão do LISP do Atlas, não possuia a então chamada facilidade de compilar. Para passar por cima dessa dificuldade, uma nova versão do sistema, chamada ALAM (Atlas LAM), foi escrita diretamente em linguagem assembly do Atlas e esse foi o sistema empregado nas primeiras aplicações.

ALAM continuou a ser desenvolvido e vinha sendo modificado toda vez que problemas mais complexos surgiam. Quando o Atlas 1 foi substituído por um CDC 6600, o sistema foi transferido para a nova máquina por R.A. Russel-Clark e o período de transferência também foi usado para repensar alguns problemas estruturais do software. ALAM se tornou obsoleta quando os computadores Atlas desapareceram, além disso, praticamente todos os usuários tinham que aprender ou LISP ou a linguagem assembly do Atlas quando trabalhavam com aplicações mais complicadas.

Por essa razão, o sucessor do ALAM, chamado CLAM (CDC LAM), foi escrito em uma linguagen de comandos muito simples. O novo sistema resultante não era extremamente eficiente pois não era muito rápido, mas mesmo assim provou ser tanto simples como flexível em aplicações métricas. Uma versão do LAM chamada ILAM foi implementada por I. Frick num IBM 360/70 e era bem mais veloz do que CLAM.

Atualmente todo o desenvolvimento do CLAM está parado, existem apenas duas versões, CLAM 2.0 e CLAM 3.0 embora esse último também não esteja operando depois que o sistema operacional do CDC 6600 foi alterado.

A família LAM, é considerada por alguns pesquisadores como linguagem morta mas tem ainda um parente muito conhecido e largamente utilizado que apresentamos a seguir.

9.6 SHEEP

O último membro da família LAM é um pouco diferente, a grande vantagem de SHEEP sobre seus predecessores é que SHEEP é interativo. Foi desenhado por I. Frick no Instituto de Física Teórica de Estocolmo para o DEC PDP 10 e escrito em LISP. Além do mais, muitas facilidades básicas e algoritmos foram redesenhados e significantemente aprimorados. Em particular o formato de entrada para expressões algébricas é feito numa notação infix similar a que é encontrada no REDUCE e no FORMAC. Ele tem um pacote geral de manipulação de tensores que pode ser usado em qualquer dimensão, e permite a fácil definição de novos tensores, um pacote para cálculos tetraédricos, facilidades para a expansão em termos de pequenos parâmetros, usando algoritmos de séries truncadas, abilidade de executar diferenciação funcional para uso em problemas variacionais, rotinas para mudança de variável, aritmética complexa, debugging extensivos, dispositivos de monitoração e instruções de comunicação com REDUCE. Uma das grandes facilidades, é que se necessitarmos de qualquer integração ou fatoração polinomial, então as expressões podem ser manipuladas para REDUCE, que os retorna quando o cálculo é completado, e, além disso, como REDUCE se comunica com FORTRAN, é também permitido se tentar computação numérica extensiva. SHEEP tem também três níveis de substituição que estão em comunicação completa entre si. L. Hornfeld desenvolveu dois pacotes experimentais baseados em SHEEP. O primeiro é uma rotina para geração automática de algoritmos tensoriais, que são úteis na definição de novos tensores. O segundo pacote é para cálculo geral (indicial) de tensores. SHEEP é extremamente sofisticado quando comparado com CLAM; um exemplo bastante conhecido foi quando R.A. d'Inverno, queria checar por cálculo direto, se a solução de Hauser, para um problema físico na referência [33], era o vácuo, pois haviam surgido dúvidas sobre o resultado. O resultado desse problema demorou semanas para ser resovido completamente usando CLAM, quando d'Inverno esteve em Estocolmo deu esse problema para Frick, que usando SHEEP, confirmou o resultado em não mais que uma hora. O sistema é organizado em uma série de componentes chamados módulos, arrumados de maneira hierarquica onde nem todos precisam ser utilizados para trabalhar em um determinado cálculo. O módulo básico é o de Álgebra, que inclui LISP 1.6 como um subconjunto, os outros módulos são para manipulação de tensores, aplicações métricas em componentes de coordenadas e aplicações métricas em componentes frame. O sistema total é tanto pequeno quanto rápido,(mesmo uma grande quantidade de trabalho pode ser feito em 30K de palavras). O manual é suficientemente completo para garantir a distribuição do sistema, não é difícil de ler, e começa com um número de exemplos de seções interativas que servem para mostrar ao usuário exatamente como começar. SHEEP também pode ser utilizado como calculadora algébrica de mesa. Uma desvantagem de SHEEP, é de certa forma sua dependência de máquinas, mas já existem planos para acabar com essa restrição.

Para esclarecimentos recomendamos:

d'INVERNO R.A., FRICK I., Interating with SHEEP.

FRICK I., The Computer Algebra SHEEP, what it can do and cannot do in General Relativity,

Stockholm University, Sweden, 1977.

9.7 SCHOONSHIP

Desenvolvido no CERN, em Genebra, em 1972 por M. Veltman em linguagem de máquina Assembly, com o objetivo de resolver problemas de cálculos em Aplicações Quânticas. O sistema é rápido e suas exigências quanto a capacidade de memória dos computadores é pequena. O programa é de fácil uso em computadores grandes, da linha CDC, como por exemplo CDC 6600; mas difícil de ser utilizado em máquinas de menor tamanho. Foi utilizado com sucesso em pesquisas em gravidade quântica, por Derek Capper no Queen Mary College em 1982 e 1983.

9.8 SMP

Desenvolvido no California Institute of Tecnology no início dos anos 80 em linguagem C por Wolfran e Cole. Embora seja disponível para os dois sistemas operacionais mais populares não está disponível para vários outros. SMP tem um número razoavelmente grande de funções internas construídas, é um sistema recente e a cada versão apresenta novas melhorias. SMP pode ser considerado um sistema conhecido pela comunidade usuária, embora alguns de seus próprios criadores, estejam investindo alto em tempo e pesquisas, no seu herdeiro mais famoso, o MATHEMATICA.

9.9 SCRATCHPAD

É um produto das pesquisas da IBM que vem sendo desenvolvido desde 1975; é escrito no LISP do sistema VM-CMS. É um sistema interativo que opera num terminal de comunicação IBM 2741 e roda no computador IBM 360/91 no Centro de Pesquisas Thomas J. Watson em Yorktown Heights. O sistema não está ainda totalmente disponível, continua em constante desenvolvimento, é de um design avançado em sua natureza matemática, até na maneira pela qual o usuário entra com os dados, com as linhas aparecendo em letras minúsculas; os símbolos matemáticos entram como são usados normalmente em matemática, por exemplo, na maioria dos sistemas para se pedir uma integração se usa INT, Integrate ou outro input qualquer dependendo da sintaxe de programação do sistema, no SCRATCHPAD o usuário entra com $\int$.

O principal objetivo do SCRATCHPAD é o de formalizar uma linguagem próxima a notação matemática convencional, de modo que o investigador possa se comunicar com o sistema algébrico

na forma conveniente o que certamente ajudará a resolver certas limitações de outros sistemas. Até o presente momento, só está planejada uma quantidade bem restrita de implementações em alguns centros científicos.

Para maiores esclarecimentos sobre o sistema recomendamos a bibliografia abaixo:

SCRATCHPAD User's Manual (Griesmer J.H., Jenks R.D. e Yun D.Y.Y.) IBM Research Publication RA70, June 1975.

JENKS R. D., SCRATCHPAD/360: Reflections on a Language Design, ACM SIGSAM Bulletin 13, no. 1, Fev. 1979, 16 - 26.

JENKS R. D. e TRAGER B. M., A Primer Keys to New SCRATCHPAD, Proc. Eurosam 84, Lecture Notes in Computer Science 174, Springer-Verlag, 1984.

Computer Algebra Group, The SCRATCHPAD II Computer Algebra System Interactive Environment Users Guide, Mathematical Sciences Dept., IBM Research Division, Yorktown, N. Y., 1988.

9.10 muMATH

Sistema desenvolvido por D. Stoutemyer e A. Rich a partir de 1978 na Universidade do Hawaii. muMATH é um programa totalmente interativo, escrito em muSIMP, um dos dialetos LISP, com a principal finalidade de maior utilização de softwares desse tipo em microcomputadores - apenas para pequenos problemas de pesquisa - , e também com a intenção de utilização em ensino universitário. A disponibilidade de sistemas de computação algébrica para microcomputadores revolucionou o ramo de programas nessa área, o que fez o muMATH um dos mais conhecidos softwares do terceiro grupo na classificação dos sistemas. muMATH é ainda utilizado em centenas de centros científicos, industrias e universidades. muMATH é disponível para microcomputadores que operem com os sistemas CP/M e DOS. Os criadores do muMATH começaram a trabalhar no desenvolvimento de outro software de computação algébrica, com várias melhorias e usando o muMATH como fonte inspiradora, o que detalhamos no cap´ítulo 4 do nosso texto.

Para maiores esclarecimentos recomendamos: "muMATH: A Microcomputer Algebra System" (C. Wooff e D. Hodgkinson), October 1987, Academic Press.

9.11 MATHPERT

É um sistema "esperto" em Matemática, desenvolvido por Michael Benson em 1989 no Departamento de Matemática e Ciência da Computação da Universidade do Estado de San José.

O sistema é explicitamente desenhado para ensino de Álgebra, Trigonometria e Cálculo dos primeiros semestres de graduação. É um sistema capaz de resolver (quase) todos os problemas de simplificação, expoentes, radicais, fatoração, resolução de equações, incluindo as transcendentes, identidades trigonométricas, limites, diferenciação e integração. Também inclui gráficos e facilidades numéricas. Nesse aspecto é similar ao MACSYMA, MAPLE ou MATHEMATICA entretanto, MATHPERT difere destes em vários aspectos, que não são apenas importantes mas vitais para educação. A maior dessas diferenças é que MATHPERT é uma caixa de vidro transparente, que segundo seu criador, não possui apenas respostas mas sim soluções, passo a passo, onde cada um desses passos é claro e justificado na tela, e não feitos internamente por algoritmos super poderosos e secretos como nos grandes sistemas. MATHPERT também incorpora um modelo de usuário suficientemente sofisticado, e o utiliza para produzir soluções passo a passo num nível de detalhamento apresentado especialmente e de acordo com o conhecimento do usuário. Esse modelo de usuário (que também funciona bem para não estudantes), contém entre outras coisas, as informações concernentes às várias partes do aprofundamento do cálculo em relação ao investigador em, bem conhecidas, conhecidas, aprendidas ou desconhecidas, de forma que MATHPERTH possa usar o modelo para sua saída, de acordo com essas informações. MATHPERT usa a palavra *solution* para significar uma sequência de passo intelegíveis, onde a última linha é *answer*. Exemplificando, ele pode dar uma solução em 6 linhas para o problema de denominador comum (1/x + 1/y) explicando passo a passo, (ou em uma linha para um estudante mais "avançado"); pode também fazer problemas de cálculos razoavelmente complexos, como calcular ($\frac{d}{dx}\sqrt{x}$) diretamente da definição de derivada mostrando e justificando cada passo em linhas separadas, e manuseando corretamente a lógica tão bem quanto a computação simbólica.

Maiores detalhes sobre o sistema em:

BEESON M., *The User Model in MATHPERT; An Expert System for Learning Mathematics* , Proceedings of the Conference of Artificial Intelligenge and Education, Amsterdam, 1989.

BEESON M., *Learning Mathematics with MATHPERT*, (1990).

9.12 GAUSS

É um sistema do terceiro grupo e até 1991 oferecia duas versões, GAUSS 2.0 e GAUSS 386, este, usando o Phar Lap DOS para acessar conjuntos maiores de dados e programas mais complexos. A versão 2.0 requer 640K RAM, 380K de espaço livre em disco, coprocessador matemático e DOS versão 3.0 ou posterior. A versão 386 requer um PC 386, com 2MB de RAM, 700K de espaço livre em disco, coprocessador matemático e DOS versão 3.0 ou posterior. A interface do software com o usuário não é das melhores, pois o código deve ser escrito em outro editor, ou no editor do sistema para depois ser calculado. Não existem menus ou qualquer help on line na linguagem, existem poucas telas principais - na verdade duas -, embora comandos possam

ser armazenados para uso posterior. GAUSS tem um grande número de funções matemáticas e procedimentos, especialmente sobre cálculo matricial. Existe possibilidade de escolha entre gráficos "rápidos" ou em "qualidade publicação" em 2D e 3D, que podem ser comparados em qualidade com os do MATHEMATICA e do Mathgraf.

Mas na realidade o ponto alto do sistema é sua velocidade, ele se apresentou de 5 a 50 vezes mais rápido, em baterias de problemas testados, do que a maioria dos sistemas estudados no nosso texto. Naturalmente o sistema não foi sempre mais veloz, o MATHEMATICA tem cinco comandos numéricos especialmente desenhados, como por exemplo NSUM, que produz tempos comparáveis com os do GAUSS e oferece mais precisão; o MAPLE até um número bem alto, tem internamente uma tabela de fatorial etc.

Maiores informações sobre o sistema são fornecidas por:

Aptech Systems, 26250 196th Place South East, Kent,

WA 98042; (206) 631-6679

9.13 RIEMANN II

Foi apresentado em Julho de 1991 no International Symposium on Symbolic and Algebraic Computation (ISSAC - 91) em Bonn, Alemanha. RIEMANN II utiliza LISP como linguagem de implementação e é um dos mais recentes softwares da área, com capacidades simbólicas, numéricas e gráficas (2D / 3D). RIEMANN II é um sistema matemático interativo que tem também programação, roda nos Atari ST e TT embora estejam previstas a curto prazo implementações para PC's e workstations.
O sistema é compatível com muSIMP e com o muMATH-83 e sua tela de trabalho lembra bastante o lay-out e a interface com o usuário do DERIVE. RIEMANN II pode ser classificado como um novo membro dos sistemas do terceiro grupo e opera com aritmética, álgebra, análise, vetores e matrizes, funções, diferenciação, integração, plotagens, programação interna, manipulação tensorial e várias outras capacidades.

Maiores detalhes sobre o sistema são obtidos na sua distribuidora.

Begemann & Niemeyer

Softwareentwicklung GbR

Schwarzenbrinker Str. 91

4930 Detmold 1

05231 / 68302

9.14 MathCAD

É também um outro sistema do terceiro grupo, requer 640K de RAM, DOS versão 2.0 ou posterior, e é um programa numérico de fácil uso e excelente interação com o usuário. MathCAD versão 2.5 exige apenas que o investigador lembre um pequeno número de chaves arbitrárias como " " para plotagens ou "&" para resolver integrais. Acertando a chave "&", o sistema fornece o sinal de integral com blocos onde são colocados seus argumentos, o usuário move o cursor através desses blocos, os preenche, liga uma chave e a integral é calculada. O software é realmente bastante abrangente e fornece uma larga escala de ferramentas numéricas, incluindo funções especiais, funções estatísticas, ajuste de curvas, vetores e matrizes, números complexos e soluções de equações. A partir da versão 2.0, foram acrescentadas plotagens em 3D às em 2D já existentes, com mudanças de pontos de vista e outras melhorias.

MathCAD não tem características de manipulação simbólica e não tem ainda programação interna construída, mas se um usuário trabalhar ná área numérica e precisar de um sistema rápido e eficiente para esse fim, o sistema é uma excelente escolha.

Maiores esclarecimentos sobre o MathCAD 2.5 são fornecidas por:

MathSoft Inc., 201 Broadway, Cambridge,

MA 02139; (800) 628-4223, (617) 577-1017.

POSTSCRIPTUM

Sustentando as especificações e qualificações sobre alguns sistemas de computação algébrica fornecidas ao longo do trabalho, poderíamos emitir uma opinião pessoal sobre os méritos relativos e os usos desses e de outros sistemas, obviamente estamos pondo de lado considerações sobre a determinação do equipamento, pois muitos sistemas só rodam num restrito tipo de máquina.

Para estudantes ou iniciantes no uso de sistemas de computação algébrica, MATHPERT, MathCAD, TKSolver!, muMATH e o DERIVE são os mais indicados por várias razões, sendo algumas delas (principalmente para o DERIVE); o preço , roda em qualquer microcomputador, manual pequeno, facilidade de uso imediato, etc.

Se formos trabalhar em problemas que utilizem cálculos de um modo geral, com certeza REDUCE, MAPLE, MACSYMA e o MATHEMATICA, fornecerão praticamente todo o ferramental necessário para esse tipo de tarefa.

Se entretanto necessitarmos de pacotes que operem cálculos mais específicos, como resolução de alguns tipos de equações diferenciais, não hesitaríamos em propor MAPLE, MACSYMA e o MATHEMATICA.

Para gráficos em duas e três dimensões nossa aposta seria que com o MATHEMATICA, com o MACSYMA, com o GAUSS e com as novas versões do MAPLE (5.0, 5.1, etc.), estaríamos em excelente companhia.

Se tivermos interesse em problemas de aplicações métricas certamente SHEEP deverá estar entre os nossos maiores desejos.

Para pesquisas em relatividade geral, mecânica celeste e física quântica indicaríamos sem dúvida SCHOONSHIP, CAMAL e SHEEP.

Em aplicações não padronizadas requeridas CAMAL, REDUCE, MACSYMA, DERIVE, MAPLE, MATHEMATICA e o SHEEP tem prestado excelentes serviços.

Se formos trabalhar em pesquisas mais direcionadas para a área de Álgebra Abstrata - Teoria de Grupos, Grupos Solúveis, Caracteres de Grupos, Anéis e Módulos - então os sistemas CAS, SOGOS, CAMAC e CAYLEY oferecem ótimo material; para Geometria Algébrica o MACAULAY é uma boa opção.

Queremos dizer também, que pessoas com interesse em computação algébrica, principalmente na Europa, avisam que o esperado sistema SCRATCHPAD, enquanto sendo altamente desejável pela comunidade de usuários para muitas dessas aplicações, não está ainda totalmente disponível. Entretanto, alguns pesquisadores afirmam que a companhia NAG começará a distribuir o SCRATCHPAD II, agora chamado AXIOM, para várias Universidades e Centros Científicos a partir de 1992.

Finalmente queremos desejar aos usuários boa sorte e bom trabalho.

CONCLUSÃO

Esperamos com esse trabalho, ter de alguma maneira contribuido com conhecimentos sobre o uso, a utilidade e a versatilidade dos sistemas de computação algébrica da nossa época, ter apresentado mais uma fonte de divulgação dessa disciplina que emerge no mundo presente, e também ter mostrado ao usuário um leque de softwares, acompanhado de algumas recomendações específicas, que certamente virão a otimizar seu tempo e facilitar a sua escolha. Este texto se dirige a estudantes e pesquisadores do meu país que estejam se iniciando nessa área, de modo que possam a partir desse embrião, seguir seus próprios caminhos e interesses e enveredar por novas pesquisas. Gostariamos finalmente, de agradecer sobremaneira aos pesquisadores que me incentivaram, ensinaram e iniciaram na educação pertinente a esse campo novo e fascinante da ciência, e que são; Dr. Vitoriano Ruas de Barros Santos (PUC/RJ) , Dr. Carlos José Pereira de Lucena (PUC/RJ) , Dr. Marcelo Rebouças (CBPF) e Dr. Renato Portugal (CBPF).

Obrigado

O autor.

Referências Bibliográficas

[1] ABRAHAMS P.W., McCARTHY J., EDWARDS D.J., HART T.P. & LEVIN M.I., *LISP 1.5 Programmer's Manual*, MIT, 1962.

[2] AKRITAS A., *Elements of Computer Algebra* , John Wiley, USA, 1989.

[3] BUCHBERGER B., COLLINS G. E. e LOOS R., *Computer Algebra: Symbolic and Algebraic Computation*, Springer-Verlag, 1983.

[4] DAVENPORT J.H., SIRET Y. e TOURNIER E., *Computer Algebra Systems and Algorithms for Algebraic Computation*, Academic Press, 1988.

[5] ELLIS W., *DERIVE for the Calculus Student: A Tutorial* , Brooks/Cole Publishing Co., U.S.A., 1991.

[6] ELLIS W., *MAPLE for the Calculus Student: A Tutorial* , Brooks/Cole Publishing Co., U.S.A., 1989.

[7] FATEMAN R. J., *A Review of MATHEMATICA* , Artigo, Computer Science Division, Universty of California, Berkeley, U.S.A., 1991.

[8] FOSTER K. R., HAIM H. B., *Symbolic Manipulations Programs for the Personal Computer*, (Software Review), Science 243, Fev. 3, 1989, 679 - 684.

[9] GEDDES K., GONNET G., *MAPLE Reference Manual* , 5th Edition, Symbolic Computation Group - Departament of Computer Science, University of Waterloo, Canadá, 1988.

[10] GEDDES K., GONNET G., *First Leaves: A Tutorial Introduction to MAPLE* , Revised for MAPLE version 4.2, WATCOM, University of Waterloo, Canada, 1988.

[11] GILLIGAM L., MARQUARDT J., *Calculus and the DERIVE Program* , Gilmar Publishing Company, USA, 1990.

[12] GLYNN J., *Exploring Math from Algebra to Calculus with DERIVE, A Mathematical Assistant* , Mathware, USA, 1989.

[13] JANSEN R., *Trends in Computer Algebra* , Notes Computer Science 296, Springer-Verlag, USA, 1987.

[14] HEARN A. C., *REDUCE User's Manual Version 3.3* , The Rand Corporation, Santa Monica, 1987.

[15] KERNIGHAM B. W., RITCHIE D.M., *C A Linguagem de Programação Padrão ANSI* , Editora Campus, 1990.

[16] MAEDER R., *Programming in MATHEMATICA* , Addison-Wesley, U.S.A., 1989.

[17] MOSES J., *MACSYMA: The Fifth Year* , Proc. Eurosam 74, Stockholm, Sweden, ACM SIGSAM Bulletin 8 no. 3, Agosto, 1974, 105 - 110.

[18] PAVELLE R., *MACSYMA: Capabilities and Aplications to Problems in Engineering and the Sciences* , Symbolics Inc, Cambridge, MIT, 1980.

[19] RAYNA G., *REDUCE Software for Algebraic Computation* , Springer-Verlag, 1987.

[20] STOUTEMYER D., RICH A., *DERIVE User Manual* , Softwarehouse, Honolulu, Havaii, 1990.

[21] WOLFRAN S., *MATHEMATICA, A System for Doing Mathematics by Computer* , Wolfran Research Inc., Addison-Wesley Publishing Company, University of Illinois, 1988.

[22] WRIGHT F. J., MacCALLUM M. A. H., *Algebraic Computation with REDUCE* , Queen Mary College, University of London, U.K., England, 1991.