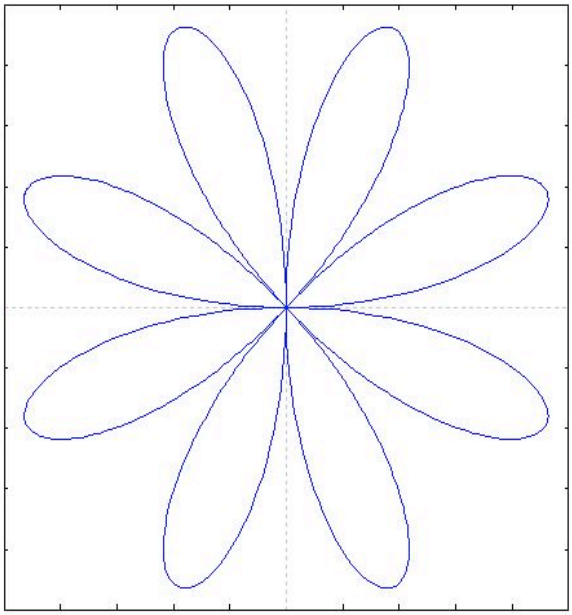
Introdução ao Software ***MAXIMA***

Bruna Santos



Orientação e Revisão de **Zélia da Rocha**

Centro de Matemática da Universidade do Porto
Dezembro 2009

Autora:

Bruna Santos é aluna de licenciatura em Matemática na Faculdade de Ciências da Universidade do Porto e bolseira de integração na investigação do Centro de Matemática da Universidade do Porto / Fundação para a Ciência e a Tecnologia.

Orientação e Revisão:

Zélia da Rocha é professora no Departamento de Matemática da Faculdade de Ciências da Universidade do Porto desde de 1994, sendo responsável por disciplinas das áreas de Cálculo Automático, Análise Numérica, Teoria de Aproximação e Polinómios Ortogonais e Aplicações. Durante estes anos, tem orientado vários alunos de Prática Pedagógica de Ensino Supervisionada, de Mestrado e de Doutoramento em Matemática Aplicada, sendo autora de vários artigos científicos, nas áreas de Matemática, Calculo Científico e Programação Simbólica.

Agradecimentos:

Este espaço é dedicado a todos aqueles que deram a sua contribuição para que este livro fosse realizado.

Em primeiro lugar, agradeço à Professora Doutora Zélia da Rocha pela forma como orientou o meu trabalho. Agradeço por ter acreditado, desde sempre, nas minhas capacidades e neste tutorial. As notas dominantes da sua orientação foram determinantes para o desenvolvimento deste manual assim como a cordialidade com que sempre me recebeu. Estou grata pela liberdade de acção que me permitiu, pois foi decisiva para que este trabalho contribuísse para o meu desenvolvimento pessoal e académico.

Gostaria de agradecer ao Professor Doutor Sílvio Gama pelos valiosos comentários sobre Estatística.

Um agradecimento especial ao Arlindo Trindade, por ter estado sempre na retaguarda, apoiando e incentivando mais este passo no meu percurso.

Deixo também uma palavra de agradecimento à Maria Freitas pelo o incentivo para escrever mais e melhor.

A todos os outros que me ajudaram a desenvolver este trabalho e cujo nome não foi aqui mencionado deixo aqui o meu agradecimento sincero.

Por fim, informo que a última parte deste trabalho foi realizada durante a vigência de uma Bolsa de Integração à Investigação, financiada pela Fundação para a Ciência e a Tecnologia, através do Centro de Matemática da Universidade do Porto, instituições às quais agradeço.

Porto, 21 de Dezembro de 2009

Bruna Santos

Prefácio:

O *MAXIMA* é uma linguagem computacional que permite realizar cálculos numéricos e simbólicos, representações gráficas e efectuar programação, possuindo uma grande variedade de comandos para os mais variados fins em Matemática e aplicações. É um software de livre acesso, disponível para os sistemas operativos usuais.

Desde o ano lectivo de 2008/2009, que tem vindo a ser adoptado em várias disciplinas no ensino superior em Portugal, em substituição de outros softwares comercializados do mesmo tipo, como o *Maple* ou o *Mathematica*.

Dado a escassa bibliografia actualmente existente sobre a utilização e aplicação deste manipulador algébrico, este livro será útil a quem deseje começar a usar o *MAXIMA* no âmbito da Matemática em geral e, em particular, das áreas de Análise Numérica e Probabilidade e Estatística, tanto nas vertentes interactiva como de programação.

Este manual, que começou por ser um pequeno bloco de notas da Bruna Santos, teve origem na realização, por esta aluna, das unidades computacionais propostas na disciplina de *Análise Numérica II*, por mim leccionada, da responsabilidade do *Departamento de Matemática* da *Faculdade de Ciências da Universidade do Porto*, ao qual pertenço. Foi, pois, com agrado e naturalidade, que apoiei a iniciativa da Bruna Santos para iniciar e desenvolver este tutorial como material de apoio para quem do *MAXIMA* necessite.

O trabalho apresentado será objecto de melhoramento e de complementação num futuro que desejamos próximo.

Porto, 21 de Dezembro de 2009.

Zélia da Rocha

Índice Geral:

Ao meu pai

"Preciso acreditar que algo extraordinário é possível."

Um pouco de história…

Um software livre é, segundo a definição criada pela ***Free Software Foundation,*** qualquer programa de computador que pode ser instalado, explorado e distribuído sem qualquer restrição. Um exemplo de um software livre é o *MAXIMA*.

O *MAXIMA* é derivado do sistema *Macsyma*, o lendário sistema de álgebra computacional desenvolvido entre os anos de 1968 e 1982 no ***Instituto de Tecnologia de Massachusetts*** (**MIT**) como parte do Projecto MAC. O **MIT** enviou uma cópia do código fonte do *Macsyma* para o Departamento de Energia em 1982, sendo que essa versão é agora conhecida como *Macsyma* DOE. Essa cópia foi mantida pelo Professor William F. Schelter da Universidade do Texas entre 1982 e 2001, ano do seu falecimento.

Em 1998, Schelter obteve permissão do Departamento de Energia para colocar disponível o código fonte do Macsyma DOE sob a Licença Pública GNU e em 2000 ele iniciou o projecto *MAXIMA* no SourceForge para manter e desenvolver o Macsyma DOE, agora chamado *MAXIMA*.

Um sistema de computação algébrica, como o *MAXIMA*, permite manipular e explorar expressões matemáticas de maneira simbólica e interactiva. O usuário digitaliza na janela do programa algumas fórmulas, comandos e o sistema avalia-os devolvendo uma resposta que pode ser manipulada posteriormente, caso seja necessário. É-nos permitido também obter soluções numéricas aproximadas e visualizar graficamente quer dados, quer funções matemáticas. O *MAXIMA*, como se trata de um software do tipo "freeware", com funcionalidades similares aos softwares comercializados, não estimula o uso de cópias não autorizadas.

Capitulo 1.

Mãos á obra

"A natureza está escrita em linguagem matemática." -Galileu

Os objectivos deste capítulo introdutório são:

- ✓ Adquirir uma ideia geral do que é o *MAXIMA* e como instala-lo;
- ✓ Introduzir comandos e obter respostas na janela;
- ✓ Formatar números e expressões;
- ✓ Familiarizar-se com algumas constantes, operadores e comandos básicos;
- ✓ Como guardar o trabalho no ambiente do WxMaxima.

1.1 Primeiro contacto com o MAXIMA

O Máxima é um potente software livre, que permite:

I. Efectuar cálculos numéricos e simbólicos;
II. Traçar gráficos bidimensionais e tridimensionais;
III. Elaborar implementações computacionais eficientes e precisas;

Para descarregar o *MAXIMA*, para o sistema operativo Windows, utilize o link:

http://sourceforge.net/projects/MAXIMA/files/MAXIMA/5.10.0-Windows/MAXIMA-5.10.0b.exe/download

Após descarregar o programa, prossiga a instalação atendendo às recomendações apresentadas. Pode optar-se pelas interfaces:

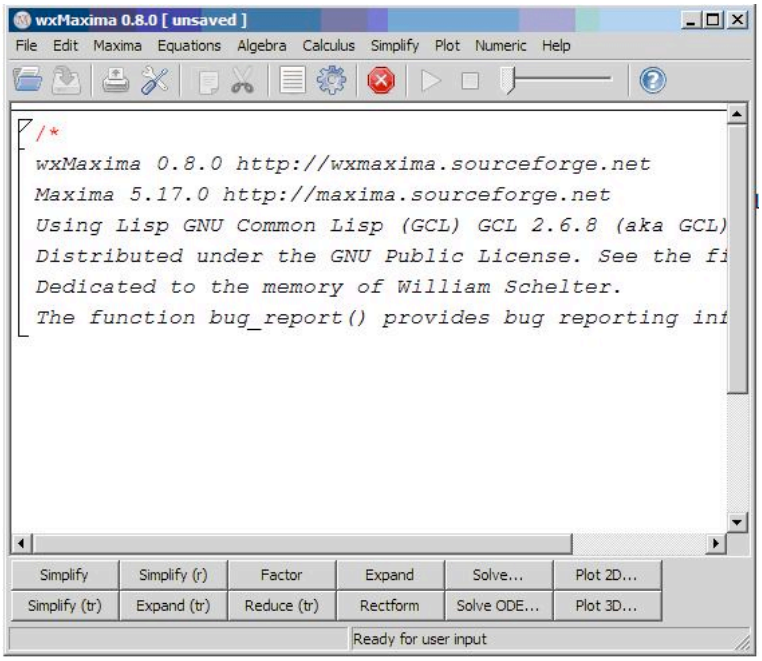
I. WxMaxima;
II. xMAXIMA;

Embora, ambos produzam os mesmos resultados, a interface WxMaxima revela-se mais interactiva com o utilizador do que a xMAXIMA, motivo pelo qual foi escolhida para desenvolver este manual. O *MAXIMA* está disponível para outros sistemas operativos, por exemplo, o Linux. Para mais informações sobre o Máxima para outros sistemas operativos consulte a página:

http://MAXIMA.sourceforge.net/

1.2 As janelas de interface com o utilizador

Após a instalação do *Máxima,* aparece uma janela com a seguinte apresentação (se optar pelo wxMáxima):



1 Interface:WxMaxima

1.3 Constantes Matemáticas:

Vamos ver como se pode realizar uma operação aritmética entre dois números:

(%i1) 5+2;

*Para visualizarmos o valor desta expressão é necessário empregar no final de cada instrução(**;**) e em seguida **Shift+Enter**. Podemos visualizar o resultado:*

(%o2) *7*

Exploremos alguns exemplos de constantes matemáticas conhecidas cujo os identificadores começam por **%**:

(%i2) % pi;
(%o2) ***%pi*** ← o output do programa será a mesma expressão que se escreveu

No entanto, se colocarmos o comando ***float*****,** visualizaremos um valor aproximado de π:

(%i3) float (%pi);
(%o3) ***3.141592653589793*** ← será impresso no programa um valor numérico da constante π

O número de Euler:

(%i4) float(%e);
(%o4) *2.718281828459045*

1.4 Manipulação de Variáveis:

Uma ferramenta importante no *MAXIMA* é a capacidade de atribuir e manipular variáveis. Uma variável, em programação, é um identificador ao qual se pode atribuír valores. No *MAXIMA* a instrução de atribuição concretiza-se empregando o símbolo **:** .

(%i1) a:7;
(%o1) 7

Se pretendemos visualizar o valor da variável ***a***, utilizamos o comando ***print.***

(%i2) print(a);
(%o2)7

Podemos manipular as variáveis efectuando, por exemplo, operações aritméticas:

(%i3) b:3;
(%o3) 3

(%i4) c:a+b;
(%o4) 10

(%i5) 2*c;
(%o5) 20

(%i6) b*c;
(%o6) 30

(%i6) b*c $** ← Empregando ***$* o resultado da expressão não é impresso no ecrã

(%i7) c^b;
(%o7) 1000

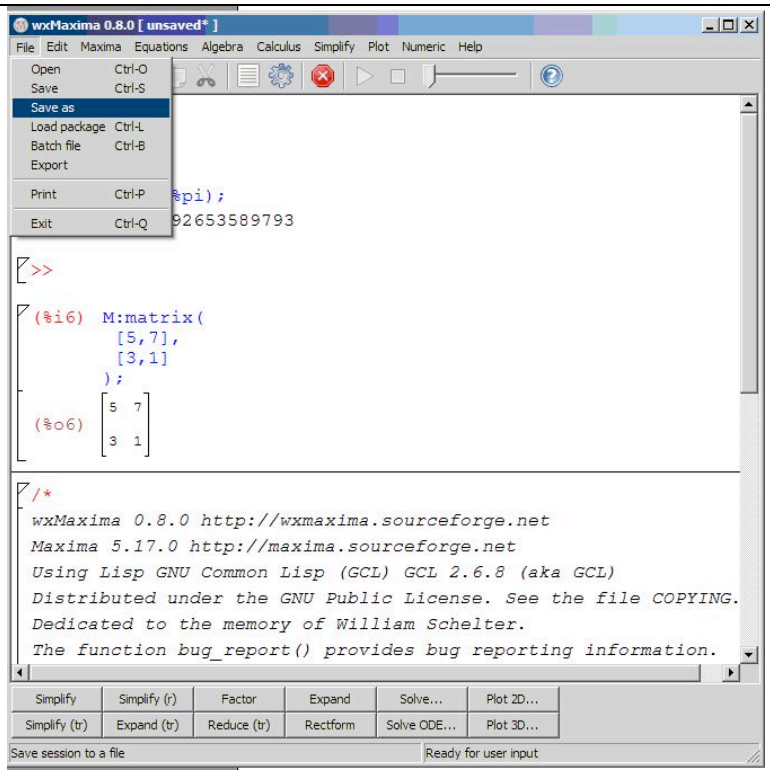
(%i8)(%);
(%o8) **1000** ← Quando se utiliza o símbolo ***%*** acede-se ao último valor calculado e armazenado na memória do *MAXIMA.*[1]

[1] Cada vez que um novo valor é calculado, o valor anterior é perdido.

1.5 Aprendendo a Guardar

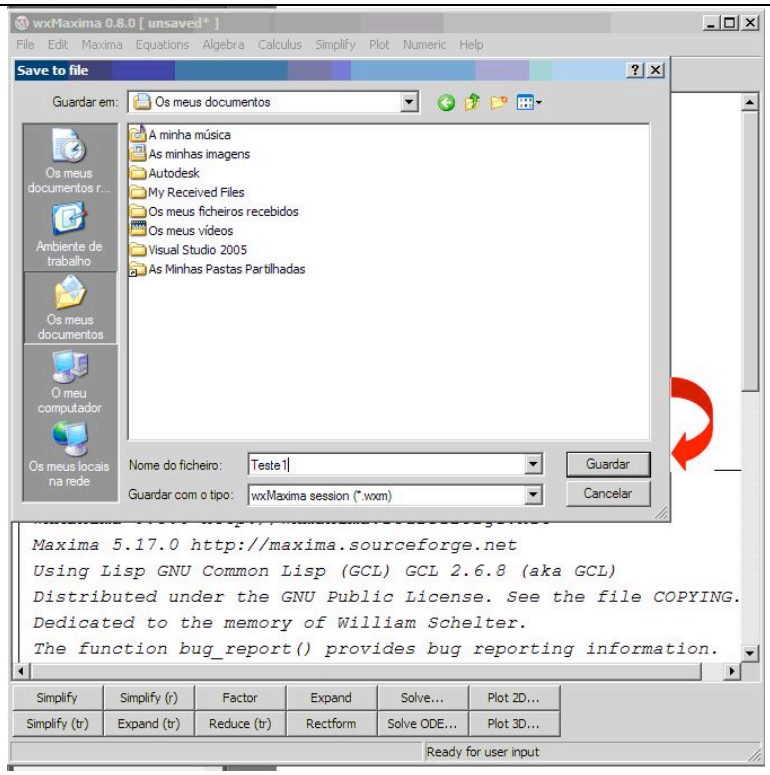
Se, após a realização de operações na interface do wxMáxima, pretendemos voltar a utilizar o trabalho, devemos seguir os seguintes passos:

1 - ***Selecciona-se a opção "Save As"***



2 Opção "Save As"

2- Escreve-se o nome do ficheiro e selecciona-se "Guardar":



3 Atribuir um nome ao ficheiro e guardar

Temos o "Teste 1" guardado no directório "Os meus documentos"

1-1 Guardar um documento em WxMaxima

1.6 Operadores e Funções Matemáticas

1.6.1 Operadores Aritméticos:

O quadro seguinte lista as operações aritméticas que se podem realizar entre expressões:

Operadores	Operação
+	Soma
-	Subtracção
/	Divisão
*	Multiplicação
^	Exponenciação

Tabela 1-2: Operações aritméticas entre expressões

1.6.2 Operadores Relacionais:

O objectivo dos operadores relacionais é fornecer respostas do tipo verdadeiro e falso a comparações. Assim, se a comparação for verdadeira vale ***true***, se for falsa, vale ***false.***
O *MAXIMA* possui operadores relacionais que podem ser usados para comparar números, variáveis, expressões, matrizes, etc.

Comando	**Descrição**
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual
=	Igualdade
#	Negação da igualdade

Tabela 1-3:Operadores Relacionais

1.6.3 Funções Matemáticas Elementares:

Comando	**Descrição do Comando**
abs (expressão)	Calcula o valor absoluto da expressão. Se a expressão for um número complexo, retorna o módulo: |x+iy|= $\sqrt{x^2 + y^2}$
factorial (x) ***x!***	Factorial de um número, x.
sqrt(x)	Raiz quadrada de x.
x^(a/b)	Raiz de índice b e expoente a
log(x)	Calcula o logaritmo neperiano de x.[2]
exp(x)	Calcula a exponencial de x.

Tabela 1-3 Funções Matemáticas Elementares

1.6.4 Funções Trigonométricas:

As funções trigonométricas, no *MAXIMA,* supõem que os ângulos estejam representados em **radianos.**

[2] O *MAXIMA* não possui uma função interna para o logaritmo de outra base, assim a definição $\log_a b := \frac{\log(b)}{\log(a)}$ será útil.

Função Trigonométrica	Descrição
sin(x) , sinh (x)	Seno, Seno hiperbólico
cos(x) , cosh(x)	Cosseno , Cosseno hiperbólico
tan(x) , tanh(x)	Tangente, Tangente hipérbolica
acos(x)	Arco-seno
asin(x)	Arco-cosseno
atan(x)	Arco-tangente

Tabela 1-4: Funções Trigonométricas

1.7 Números Decimais e Complexos

1.7.1 Números Decimais

A obtenção de uma representação decimal de um número real é feita por duas formas:

I. ***float(expressão)***

II. Após a expressão a calcular, acrescentar uma **vírgula** e a palavra ***numer.***

Exemplo - 1: Calcule o valor de $\pi - e$ e $\frac{\pi}{7}$ na forma decimal.

Resolução:

```
(%i1) %pi-%e;
(%o1) %pi-%e

(%i2) float(%);
(%o2)0.42331082513075

(%i3) %pi/7, numer;
(%o3) 0.44879895051283
```

Exemplo 1-1

Em cálculo numérico, por vezes, os resultados e os cálculos efectuados necessitam de ser expressos com um determinado número de casas decimais e/ou algarismos significativos. Podemos estabelecer essa precisão mediante a fixação de um valor à variável interna global ***fpprec*** (float point precision), que por defeito no Máxima é 16) ou através do menu ***Numeric*** do WxMaxima; na opção "***Set Precision***".

Exemplo - 2: Calcule o valor de e com 25 casas decimais:

Resolução:

```
(%i1) fprec:25;
(%o1) 25

(%i2) bfloat(%e);
(%o2)2.7182818284590452353602888b0
```

Exemplo 1-2

1.7.2 Números Complexos

No *MAXIMA* os números complexos são da forma a+b*%i, em que a parte imaginária é denotado por "%i" e a e b são números reais. Os comandos ***realpart*** e ***imagpart*** indicam a parte real e parte imaginária, respectivamente. Se desejarmos simplificar uma expressão envolvendo números complexos podemos utilizar, por exemplo, a função ***demoivre.***

Exemplo-3: Dado o número complexo ***2+5i*** aceda à parte real e imaginaria e simplifique o número.

Resolução:

```
(%i1) numcomplexo: 2+5*%i;
(%o1) 5*%i+2


(%i2) realpart(numcomplexo);
(%o2)2


(%i3) imagpart(numcomplexo);
(%o3) 5


(%i4) exp (numcomplexo);
(%o4) %e^(5%i+2)


(%i5) demoivre (%);
(%o5) %e^2 (%i sin(5)+cos(5))
```

Exemplo 1-3

Capitulo 2.

Listas, Matrizes e suas Operações

"A natureza está escrita em linguagem matemática." –Galileu

Neste capítulo abordaremos os seguintes itens:

- ✓ Criar listas e matrizes;
- ✓ Manipular listas e matrizes;
- ✓ Realizar operações com listas e matrizes;

2.1 Listas

As listas são utilizadas para representar vários tipos de dados aglomerados por uma determinada ordem. No Máxima uma lista define-se do seguinte modo:

(%i1) lista1 :[b,a,1,4,5];

(%o1) ***[b,a,1,4,5]***

O comando ***listp (expr)*** verifica se a expressão é ou não uma lista. Por exemplo:

(%i2) listp(lista1);

(%o2)*true* ← Retomando ***true*** quando é uma lista..

(%i3) listp(x:9);

(%o3) ***false*** ← Retomando ***false*** quando o argumento não é uma lista.

Os elementos de uma lista são indexados a partir de um. Para aceder a elementos da lista faz-se:

(%i4) first(lista1); ou lista[1];

(%o4) ***b*** ← Acede ao primeiro elemento da lista

(%i5) last(lista1);

(%o5) ***5*** ← Acede ao último elemento de uma lista

(%i6) rest (lista, 2);

(%o6) ***[1,4,5]*** ← Imprime uma lista com os dois primeiros elementos removidos.

(%i7) part (lista, 3); ou lista[3];

(%o7) ***1*** ← Acede ao terceiro elemento de uma lista

(%i8) length (lista);

(%o8) ***5***← Determina o número de elementos de uma lista

(%i9) sort(lista1);

(%o9) ***[1,4,5,a,b, X]*** ← Ordena os elementos de uma lista.

De seguida definimos uma nova lista que nos será útil para os comandos que se seguem:

(%i1) lista2:[3,5,6,a,d,t];

(%o1) ***[3,5,6,a,d,t]***

(%i2) join(lista1, lista2)

(%o2)*[b,3,a,5,1,6,4,a,5,d]* ← Cria uma nova lista contendo os elementos da **lista1** e da **lista2** intercalados. O comando ***join*** ignorou os elementos da lista mais longa.

(%i3) append(lista1, lista2)

(%o3) ***[b,a,1,4,5,3,5,6,a,d,t]*** ← Cria uma nova lista incluindo todos os elementos de **lista1 e lista 2** seguidos, incluindo os elementos da lista mais longa.

Existem comandos que nos permitem adicionar novos elementos a uma lista, tanto no início como no final de mesma. Vejamos como os comandos ***cons*** e ***endcons*** funcionam:

(%i1) cons(X,lista1);

(%o1) ***[X,b,a,1,4,5]*** ← Adiciona o elemento X no inicio da lista.

(%i2) endcons(X,lista);

(%o2)*[b,a,1,4,5,X]* ← Adiciona o elemento X no final da lista.

É possível realizar operações elementares, tais como a soma, subtracção, multiplicação, com os elementos de uma lista ou entre elementos de duas listas:

(%i1) lista3:[1,4,5,8,7];

(%o1) ***[1,4,5,8,7]***

(%i2) apply("+", lista3)

(%o2)*25* ← Somou todos os elementos da lista3

(%i3) apply("*", lista3);

(%o3) ***1120*** ← Multiplicou todos os elementos da lista3

Para aplicar uma função do tipo seno, factorial, etc. a cada elemento de uma lista recorre-se ao comando ***map***:

(%i1) map ("!", lista3);

(%o1) **[*1, 24, 120, 40320, 5040]***

(%i2) map (sin, lista3);

(%o2)*[sin(1),sin(4),sin(5),sin(8),sin(7)]*

Se aplicarmos o comando ***float*** à expressão anterior, obtemos o valor numérico do seno de cada elemento da lista:

(%i3) float(%);

(%o3) *[0.8414709848079,-0.75680249530793,-0.95892427466314,0.98935824662338,*

0.65698659871879]

Ao aplicar o comando ***map*** a um determinado tipo de funções, nomeadamente funções trigonométricas, exponencial e logarítmica; não é preciso empregar **""**.

Podemos efectuar as operações de subtracção (-), multiplicação (*) e divisão (/) entre listas. Note-se que a diferença entre os (.) e (*). O Com efeito, o símbolo (*) efectua a multiplicação do elemento [i] da lista 1 pelo elemento [i] (com o mesmo índice) da lista 2, enquanto o símbolo (.) efectua o produto escalar entre duas listas, isto é, a soma do produto dos elementos correspondentes das duas listas. Assim:

(%i4) lista2:[9,3,4,5,1];

(%o4) *[9,3,4,5,1]*

(%i5) lista2+lista3

(%o5) *[10,7,9,13,8]* ← Soma os elementos com mesmo índice da **listas 2** e **3**

(%i6) lista2*lista3;

(%o6) *[9,12,20,40,7]*

(%i7) lista2.lista3;

(%o7) *88*

Por vezes, é necessário construir listas com certos critérios e características. Para isso, o MAXIMA dispõe de dois comandos: o ***create_list*** e o ***makelist*** [3]

(%i1) create_list1(x^i,i,[1,3,7]);

(%o1) *[x, x^3 ,x^7]*

[3] Este dois comandos produzem os mesmo resultados

(%i2) makelist (x=y,y,[a,b,c]);

(%o2)*[x = a, x = b, x = c]*

(%i3) create_list1(1/i, i, 1, 4);

(%o3) *[1, 1/2, 1/3, 1/4]*

(%i4) x:makelist (x[k],k,0,4);

(%o4)*[x[0], x[1] ,x[2], x[4]]*

2.2 Quadro Resumo: Comandos sobre Listas

Comandos	Descrição
listp (expressão)	Determina se a *expressão* é ou não uma lista. No caso afirmativo retorna ***true*** senão retorna ***false.***
first (lista)	É primeiro elemento de uma lista.
last (lista)	É o último elemento de uma lista.
rest (lista,n)	É uma lista com os *n* primeiros elementos da lista removidos.
part (lista,n) ***ou*** ***lista[n]***	Acede ao elemento de índice *n* de uma lista.
length (lista)	Determina o comprimento de uma lista.
sort (lista)	Ordena os elementos de uma lista por ordem crescente e/ou alfabética.
join (lista1, lista2)	Cria uma nova lista com os elementos da lista 1 e da lista 2 intercalados. Este comando ignora os elementos da lista mais longa, se as listas não possuírem o mesmo comprimento.
append(lista1, lista 2)	Cria uma nova lista com os elementos da lista 1 e da lista 2 seguidos.
cons (elemento,lista)	Adiciona um elemento no início da lista, sem alterar a lista inicial. Para actualizar a lista inicial, é preciso efectuar a seguinte instrução de atribuição: **lista: cons (elemento, lista);**
endcons (elemento,lista)	Adiciona um elemento no fim da lista sendo que este comando possui as mesmas características que ***cons.***
creat_list (expressão, variável, lista)	Estes comandos permitem-nos criar listas com determinadas características.
makelist (expressão, variável, lista)	

Tabela 2-5

2.3 Matrizes

No Máxima uma matriz define-se do seguinte modo:

(%i1) A: matrix (lista1,lista2,..,listan);

Onde as lista1, lista2, ..., listan são as linhas da matriz e devem ter o mesmo comprimento.

Exemplo 1: Defina, no Máxima, a matriz A= $\begin{bmatrix} 5 & -3 & -4 \\ 1 & -7 & 12 \\ 10 & 4 & -6 \end{bmatrix}$ e determine o número de linhas e de colunas de A.

Resolução:

(%i1) A: matrix ([5, -3, -4],[1, -7, 12], [10,4,-6]);

(%o1) $\begin{bmatrix} 5 & -3 & -4 \\ 1 & -7 & 12 \\ 10 & 4 & -6 \end{bmatrix}$

(%i2) matrix_size (A);

(%o2) ***[3, 3]***

Exemplo 2-4

2.3.1 Operações Básicas com Matrizes

Com o Máxima é possível efectuar as operações básicas (soma, subtracção, multiplicação, inversa e exponenciação) com matrizes. Por exemplo, se A e B são duas matrizes com a mesma dimensão, A+B, é a soma as duas matrizes.

Exemplo 2: Sejam as matrizes A= $\begin{bmatrix} 5 & -3 & -4 \\ 1 & -7 & 12 \\ 10 & 4 & 6 \end{bmatrix}$ e B= $\begin{bmatrix} 1 & -3 & -6 \\ 2 & -7 & 12 \\ 10 & 1 & -6 \end{bmatrix}$ calcule através de comandos do Máxima:

a) A+B;
b) 3A;

c) B-2A;
d) $(A.B^{-1})^{4}$;
e) AB^{-1};

Resolução:

(%i1) A: matrix ([5, -3, -4], [1, -7, 12], [10, 4, 6])$

(%i2) B: matrix ([1, -3, -6], [2, -7, 12], [10, 1, -6])$

(%i3) A+B;

(%o3)
$$\begin{bmatrix} 6 & -6 & -10 \\ 3 & -14 & 24 \\ 20 & 5 & 0 \end{bmatrix}$$

(%i4) 3.A;

(%o4)
$$\begin{bmatrix} 15 & -9 & -12 \\ 3 & -21 & 36 \\ 30 & 12 & 18 \end{bmatrix}$$

(%i5) B – 3.A;

(%o5)
$$\begin{bmatrix} -14 & 6 & 6 \\ -1 & 14 & -24 \\ -20 & -11 & -24 \end{bmatrix}$$

(%i6) A.B;

(%o6)
$$\begin{bmatrix} -41 & 2 & -42 \\ 107 & 58 & -162 \\ 78 & -52 & -48 \end{bmatrix}$$

(%i7) $A.B^{-1}$;

[4] **Definição**: O inverso de uma matriz quadrada B é a matriz B^{-1}

(%o7) $\begin{bmatrix} \frac{1}{5} & 1 & \frac{3}{2} \\ 2 & 1 & 1 \\ 1 & \frac{1}{4} & -1 \end{bmatrix}$

Exemplo 2-5

2.3.2 Acrescentando e Eliminando Elementos:

Por vezes, há necessidade de acrescentar ou extrair colunas e/ou linhas a uma matriz.

Exemplo 3: Seja A=$\begin{bmatrix} 1 & -3 & -6 \\ 2 & -7 & 12 \\ 10 & 1 & -6 \end{bmatrix}$. Resolva as seguintes alíneas através do MAXIMA:

a) Aceda ao elemento da segunda linha e terceira coluna
b) Extraía a primeira linha;
c) Extraía a última coluna;
d) Adicione a linha [3,7,9] à matriz A;
e) Adicione a coluna $\begin{bmatrix} 2 \\ -1 \\ 5 \end{bmatrix}$

Resolução:

```
(%i1) A: matrix ([1, -3, -6], [2, -7, 12],  [10, 1, -6])$

(%i2) A[2,3];

(%o2)12

(%i3) row(A, 1);

(%o3) [1, -3, -6];

(%i4) col(A,3);
```

(%o4) $\begin{bmatrix} -6 \\ 12 \\ -6 \end{bmatrix}$

(%i5) addrow(A, [3, 7, 9]);

(%o5)
$$\begin{bmatrix} 1 & -3 & -6 \\ 2 & -7 & 12 \\ 10 & 1 & -6 \\ 3 & 7 & 9 \end{bmatrix}$$

(%i6) C:transpose ([2, -1, 5]);

(%o6)
$$\begin{bmatrix} 2 \\ -1 \\ 5 \end{bmatrix}$$

(%i7) addcol (A, C);

(%o7)
$$\begin{bmatrix} 1 & -3 & -6 & 2 \\ 2 & -7 & 12 & -1 \\ 10 & 1 & -6 & 5 \\ \end{bmatrix}$$

Exemplo 2-6

2.3.3 Elementos de Álgebra Linear

Definição: Uma matriz diagonal é toda a matriz quadrada em que os elementos que não pertencem à diagonal principal são iguais a zero. Sendo que, os elementos da diagonal principal podem ou não serem iguais a zero (não sendo necessariamente iguais).
A matriz diagonal tem as seguintes propriedades:

I. É simétrica.
II. Tem por valores próprios os elementos da diagonal principal e por vectores próprios os vectores da base canónica.
III. O determinante é igual ao produto dos elementos da diagonal principal.

Exemplo 4: Através de comandos do Máxima:

a) Defina uma matriz diagonal, MD, em que os elementos na diagonal são diferentes de zero;
b) Determine o determinante da matriz diagonal.
c) Calcule o polinómio característico, $P(\lambda) = \det(I - \lambda MD)$, de MD.
d) Encontre os valores próprios e vectores próprios de MD.

Resolução:

a)

(%i1) MD: matrix [[2,0],[0,7]];

(%o1) $\begin{bmatrix} 2 & 0 \\ 0 & 7 \end{bmatrix}$

b)

(%i2) determinant (MD);

(%o2) ***49***

c)

(%i3) charpoly (MD,x);

(%o3) $(2-x)(7-x)$

Para calcular os valores e vectores próprios, no Máxima, é necessário "chamar" o pacote " **eigen.mac**". através ***load("eigen")***.

(%i4) load ("eigen");

(%o4) *C:/Programas/MAXIMA-5.17.0/share/MAXIMA/5.17.0/share/matrix/eigen.mac*

(%i5) eigenvalues (MD);

(%o5) ***[[7],[2]]*** ← Retoma duas listas. A primeira lista retoma os valores próprios e a segunda lista retoma a multiplicidade dos mesmos. Neste caso 7 é valor próprio duplo.

d)

(%i6) eigenvectors (MD);

(%o6) ***[[[2,7],[1,1]],[1,0],[0,1]]*** ← Retorna uma lista de listas cuja primeira sub-lista é a saída de ***eigenvalues*** e as outras sublistas são os autovectores da matriz correspondente.

Exemplo 2-7

Definição: Uma matriz quadrada A_{nxn} diz-se invertível se existir uma matriz B, tal que AB=BA= *Identidade*

Exemplo 5: Seja A= $\begin{bmatrix} 2 & 5 \\ 1 & 3 \end{bmatrix}$. Calcule, através do Máxima, a inversa de A.

Resolução:

(%i1) A: matrix ([2, 5],[1, 3]);

(%o1) $\begin{bmatrix} 2 & 5 \\ 1 & 3 \end{bmatrix}$

(%i2) Id:ident(2);

(%o2) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

(%i3) B:invert (A);

(%o3) $\begin{bmatrix} 3 & -5 \\ -1 & 2 \end{bmatrix}$

Verifiquemos o nosso resultado:

(%i4) A.B;

(%o4) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

(%i5) B.A;

(%o5) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Exemplo 2-8

Definição: Uma matriz quadrada A diz-se ortogonal se e só se a sua inversa for igual à sua transposta, isto é:

$$A^{-1} = A^T \Leftrightarrow AA^T = A^T A = Identidade$$

Exemplo 6: Verifique, com comandos do Máxima, se a matriz C=$\begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}$ é ortogonal.

Resolução:

(%i1) C: matrix ([1/2, sqrt(3)/2] , [sqrt(3)/2 , -1/2]);

(%o1) $\begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}$

(%i2) InvC: invert(C);

(%o2) $\begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}$

(%i3) TC: transpose (C);

(%o3) $\begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}$

(%i4) C.TC;

(%o4) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Exemplo 2.3-9

2.4 Operações Básicas com Matrizes:

Operação	**Descrição**
A ± B	Soma ou diferença de matrizes (adiciona ou subtrai elementos correspondentes, ou seja, A[i,j] ± B[i,j] onde i,j ∈ IN)
k*M	Multiplicação de um escalar por todos

	elementos de uma matriz
A.B [5]	Produto entre duas matrizes.
$A.B^{-1}$	Produto de uma matriz pelo o inverso da outra.
M ^ k [1] ***k ^ M*** [2]	A exponenciação (^) é realizada, elemento a elemento.

Tabela 2-6: Operações básicas com Matrizes

2.5 Quadro Resumo: Comandos sobre Matrizes

Comandos	Descrição
matrix_size (M)	Determina o número de linhas e número de colunas de uma matriz
diagmatrix (n, variável)	Retorna uma matriz diagonal de dimensão *n* por *n* em que os elementos da diagonal principal são todos iguais à variável fornecida.
charpoly (M, variável)	Calcula o polinómio característico de uma matriz
eigenvalues (M)	Imprime uma lista de duas listas. A primeira sub-lista contem os valores próprios, a segunda sub-lista contém a multiplicidade dos mesmos na ordem correspondente.
eigenvects (M) ou eigenvectores (M)	Retorna uma lista contendo os vectores próprios da matriz *M*.
determinant (M)	Calcula o determinante da matriz *M*
invert (M)	Retorna a inversa da matriz *M*.
ident(n)	Retorna uma matriz identidade *n* por *n.*
transpose (M)	Calcula a matriz transposta de *M*.
addcol (M, coluna 1,…,coluna n)	Acrescenta a(s) coluna(s) dada(s) no final da matriz *M*.
addrow (M, lista 1,…, lista n)	Acrescenta a(s) linhas dada(s) no final da matriz *M*.

Tabela 2-7: Comandos sobre matrizes

[5] Corresponde ao somatório dos produtos da linha ***i*** da matriz A e das colunas ***j*** da matriz B. O produto entre matrizes requer que o número de colunas da matriz A deva ser igual ao número de linhas da matriz B.

Capitulo 3.

Equações, Sistemas de Equações e Equações Diferenciais

"Os números são as regras dos seres e a Matemática é o Regulamento do Mundo." - F. Gomes Teixeira

O objectivo deste capítulo é a manipulação de equações, sistemas de equações e equações diferenciais.

Após a leitura deste capítulo o leitor será capaz de:

- ✓ Definir e resolver equações no *MAXIMA;*
- ✓ Aceder aos membros de uma equação;
- ✓ Calcular raízes de equações;
- ✓ Definir e resolver sistemas de equações;
- ✓ Definir e resolver equações diferenciais;

3. Equações, Sistemas de Equações e Equações Diferenciais

3.1 Equações:

No *MAXIMA* as equações definem-se através do operador "=". Para aceder ao primeiro e ao segundo membro utilizam-se, respectivamente, os comandos "lhs" e "rhs" (que podemos traduzir como lado esquerdo e lado direito).

Exemplo 1: Defina a equação $x^3+4=\frac{6+x^2}{x}$ e aceda ao primeiro e ao segundo membro da mesma.

Resolução:

(%i1) eq:x^3 +4 =(6+x^2)/x;

(%o1) $x^3+4=\frac{6+x^2}{x}$

(%i2) lhs(eq);

(%o2) x^3+4

(%i3) rhs(eq);

(%o3) $\frac{6+x^2}{x}$

Exemplo 3.1-1

Podemos somar e multiplicar expressões a ambos os membros de uma equação. Por exemplo:

(%i4) (x/4)*eq;

(%o4) $\frac{x(x^3+4)}{4}=\frac{6+x^2}{4}$

(%i5) expand(%);

(%o5) $\frac{x^4}{4}+x=\frac{x^2}{4}+\frac{3}{2}$ ← a equação é simplificada.

Podemos definir uma função de uma ou mais variáveis utilizando o operador "**:=" :**

(%i1) f(x,y) := x+(x/y);

(%o1) $f(x,y) := x + \frac{x}{y}$

(%i2) f(9,3); ← calcula o valor da função no ponto (9,3).

(%o2) ***12***

Outra maneira de definir uma função é através do comando ***define.*** Exemplificando:

(%i1) exp: log(y) + cos(x);

(%o1) *exp: log(y) + cos(x);*

(%i2) define (F1(x,y),exp);

(%o2) *F1(x,y):= log(y) + cos(x);*

(%i3) F1(2,3), numer;

(%o3) *0.68246545212097*

No *MAXIMA*, para calcular os zeros de uma equação podemos utilizar o comando ***solve (equação)*** ou ***solve (equação=0)***. Contudo, nem sempre é possível obter uma solução de forma analítica. Nesse caso pode utilizar-se comandos que têm como base a aplicação de métodos numéricos. Um exemplo deste tipo de comando é o ***find_roots(equação).***

Exemplo 2: Calcule as raízes da equação $e^{-x} = x$**.**

Resolução:

(%i1) f(x) :=exp(-x)-x;

(%o1) ***f(x) :=exp(-x)-x;***

(%i2) solve(f(x));

(%o2)$[x = \%e^{-x}]$ ← o comando solve não fornece uma resposta explícita.

```
(%i3) find_root (f(x),x,0,1);

(%o3) 0.56714329040978
```

Exemplo 3.1-2

3.2 Sistemas de Equações

Os sistemas de equações escrevem-se como listas de equações. No caso em que o sistema é linear, este pode resolver-se através do comando ***solve*** (ou ***linsolve***).

Exemplo 1: Resolva os seguintes sistemas:

i. $\begin{cases} x+2y=5 \\ 2x+y=4 \end{cases}$

ii. $\begin{cases} 2x+3y-z=0 \\ 4x+y+z=7 \\ -2x+y+z=4 \end{cases}$

Resolução:

```
(%i1) solve([x+2*y=5, 2*x+y=4], [x,y]);
(%o1) [[ x=1,y=2 ]]

(%i2) linsolve([2*x+3*y-z=0, 4*x+y+z=7, -2*x+y+z=4], [x,y,z]);
```

(%o2) $\left[\left[x=\frac{1}{2}, y=1, z=4\right]\right]$

Exemplo 3.2-1

3.3 Equações Diferenciais Ordinárias

***Definição*:** Uma equação diferencial ordinária (EDO) é uma equação da forma:

$$F(x, y(x), y'(x), y''(x), ..., y^{(n)}(x)) = 0$$

envolvendo uma única função incógnita ***y=y(x)*** e suas derivadas. A variável independente é ***x*** e ***y*** é a variável dependente. A derivada de ordem *n* denota-se pelo símbolo $y^{(n)}(x)$. A ordem da equação diferencial ordinária é dada pela mais alta derivada da função presente na equação.

O comando do *MAXIMA* que resolve equações diferencias ordinárias de primeira e segunda ordem é ***ode2*** (*ode=ordinary differential equation*). Ao usar o comando temos que definir a:

I. Equação diferencial;
II. Variável dependente;
III. Variável independente;

Exemplo 1: Defina e resolva a seguinte equação diferencial $(x-1)y^3+(y-1)x^3\frac{dy}{dx}=0$

Resolução:

(%i1) eqdif: (x-1)*y^3 + (y-1)*x^3* ‘ diff (y,x) =0;

(%o1) $(x-1)y^3+(y-1)x^3(\frac{dy}{dx})=0$

(%i2)resoleq:ode2(eqdif,y,x);

(%o2) $\frac{2y-1}{2y^2}=\%c-\frac{2x-1}{2x^2}$ ← a solução depende de uma constante arbitrária representada por *%c*.

Nota: É necessário o uso do apóstrofo (‘) antes de ***diff*** com o objectivo de evitar o cálculo da derivada

Exemplo 3.3-1

Exemplo 2: Defina e resolva a equação diferencial de segunda ordem $\frac{d^2y}{dx^2}+y(\frac{dy}{dx})^3=0$

Resolução:

(%i1) eqdif2: ‘diff(y,x,2)+y*’diff(y,x)^3=0;

(%o1) $\frac{d^2y}{dx^2}+y(\frac{dy}{dx})^3=0$

(%i2) resoleq2:ode2(eqdif2,y,x);

(%o2) $y=\frac{y^3+6\%k1y}{6}=x+\%k2$ ← A solução depende de duas constantes *%k1* e *%k2*.

Exemplo 3.3-2

Exemplo 3: Defina e resolva o seguinte problema de valor inicial (PVI) $x^2\frac{dy}{dx}+3xy=\frac{\sin(x)}{x}$, onde as condições iniciais são (π,0).

Resolução:

(%i1) eqdif3: x^2*'diff(y,x) + 3*x*y=sin(x)/x;

(%o1) $x^2(\frac{dy}{dx})+3xy=\frac{\sin(x)}{x}$

(%i2) resoleq3:ode2(eqdif3,y,x);

(%o2) $y=\frac{\%c-\cos(x)}{x^3}$

(%i3) ic1(resoleq3,x=%pi, y=0);

(%o2) $y=-\frac{1+\cos(x)}{x^3}$ ← a constante é determinada pela condição inicial, que corresponde a $y(\pi)=0$

Exemplo 3.3-3

Existem equações diferenciais cuja solução analítica não é possível de determinar. Assim, teremos que recorrer ao estudo qualitativo dessas equações, desenhando um campo de direcções nas duas dimensões *x* e *y*.

Exemplo 4: Obtenha o campo de direcções da equação diferencial $\frac{dy}{dx}=x-y^2$ e a trajectória da solução que passa por (-1,3) .

Resolução:

(%i1)load("plotdf")$

(%i2) plotdf(x-y^2, [trajectory_at,-1,3]);

Exemplo 3.3-4

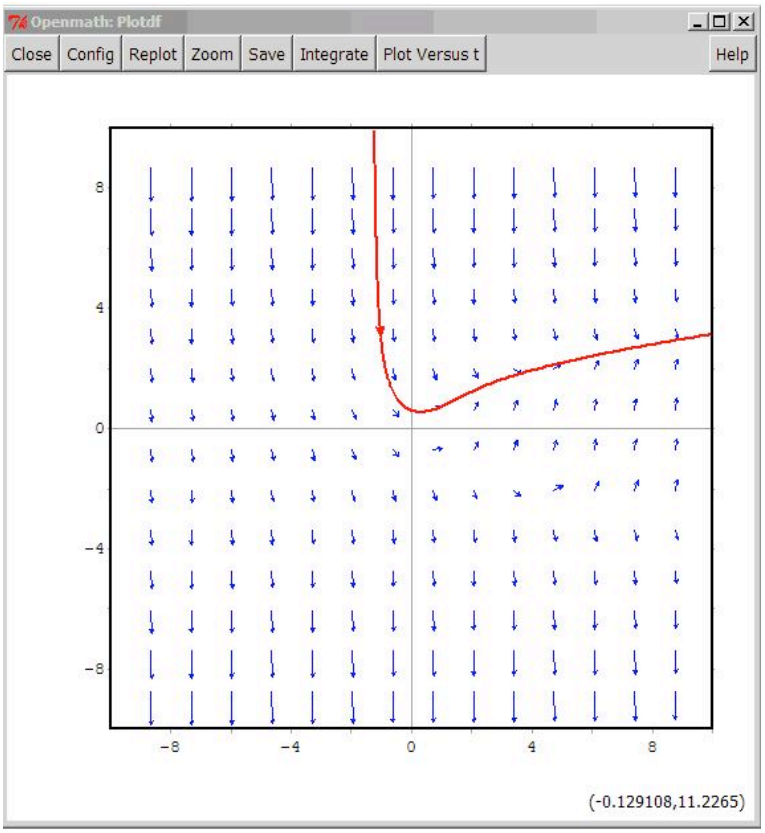


Ilustração 4 - Campo de direcções

3.4 Quadro – Resumo: Comandos do Capitulo 3

Comando	Descrição
lhs (expr)	Acede ao lado esquerdo (isto é, o primeiro membro) da expressão ***expr***, quando o operador de ***expr*** for relacional ou de atribuição.
rhs (expr)	Acede ao lado direito (isto é, o segundo argumento) da expressão ***expr,*** quando o operador de ***expr*** for relacional ou de atribuição.
solve (expr,x)	Resolve a equação algébrica, ***expr***, na variável x,
find_roots (f(x),x,a,b) ***find_roots (f,x,a,b)***	Determina a raiz da função f, ou da expressão f(x), na variável x (por exemplo), pertencente ao intervalo [a,b].
allroots (expr) ***allroots (eqn)***	Calcula, através de métodos numéricos, raízes, complexas ou reais, do polinómio (***expr***) ou de uma equação polinomial de uma variável (***eqn***).
linsolve ([expr1,expr2,...,exprn],[x1,x2,..,xn])	Resolve uma lista de equações lineares simultaneamente, ou seja, um sistema.

ode2(deqn, dvar, ivar)	Resolve equações diferenciais ordinárias (EDO) de primeira ou de segunda ordem. Recebe três argumentos: uma EDO ***deqn***, a variável dependente, ***dvar***, e a variável independente, ***ivar***
ic1(deqn1,xval, yval)	Resolve o problema de valor inicial constituído por uma equação diferencial de primeira ordem e as coordenadas ***xval*** e ***yval*** do ponto inicial.
plotdf	Produz o campo de direcções de uma equação diferencial em duas dimensões, x e y.

Tabela 3.1: Comandos introduzidos no capitulo 3

Capitulo 4.

Gráficos

"O livro da natureza foi escrito exclusivamente com figuras e símbolos matemáticos." - Galileu

O *Máxima* possui vários recursos para produzir diversos tipos de gráficos em duas e três dimensões. Os objectivos deste capítulo são:

- ✓ Traçar gráficos a duas dimensões de tipos: $\begin{cases} continuos \\ discretos \\ paramétricos \\ polares \end{cases}$.
- ✓ Inserir "texto" na área do gráfico;
- ✓ Traçar gráficos a três dimensões;

4.1 Introdução aos Gráficos

4.1.1 Gráficos de Funções de uma Variável – Coordenadas Cartesianas

Para visualizarmos graficamente funções de uma variável utiliza-se o comando ***plot2d()***. Acedemos a este comando de duas formas:

I. Utilizando a função ***plot2d()*** directamente.
II. No menu do *WxMaxima*, selecciona-se ***Plot*** e seguidamente a opção ***Plot2d*** donde surgirá uma janela de diálogo da forma:

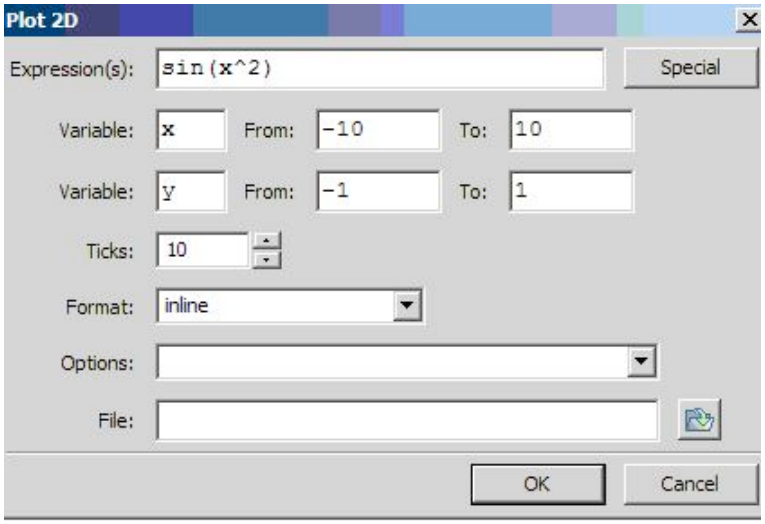


Ilustração 5- Janela de Diálogo de Gráficos 2D no *WxMaxima*

Através desta janela é possível definir vários parâmetros de uma forma interactiva. No nosso exemplo, introduzimos a função $sin(x^2)$ nos intervalos [-10,10] x [-1,1]. Após a conclusão da introdução dos vários parâmetros surgirá:

```
(%i1) wxplot2d ([sin(x^2)], [x,-10,10], [y,-1,1])
```

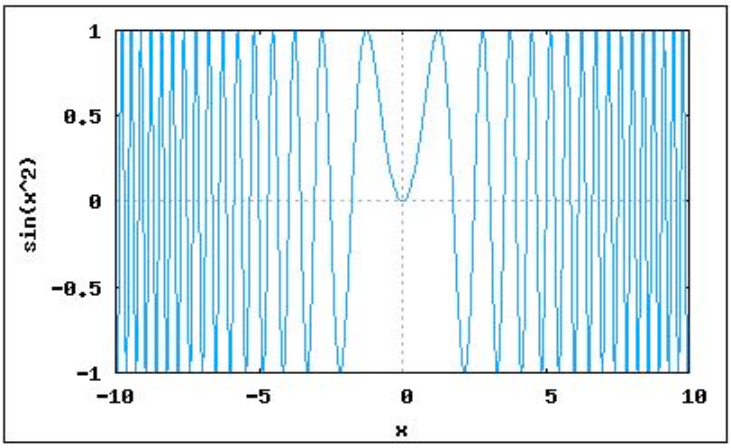


Ilustração 6 Gráfico da função sin(x^2)

Para visualizar múltiplos gráficos na mesma janela devemos colocar as funções na forma de lista como podemos ver no exemplo que se segue:

Exemplo 1: Represente na mesma janela, ou seja, no mesmo sistema de eixos, as duas funções $x^3, \sqrt{x}$ reduzindo o intervalo de valores de x e y entre [0,2] e [0,4], respectivamente.

Resolução:

```
(%i1) plot2d ([x^3, sqrt(x)], [x,0,1], [y,0,4]);
```

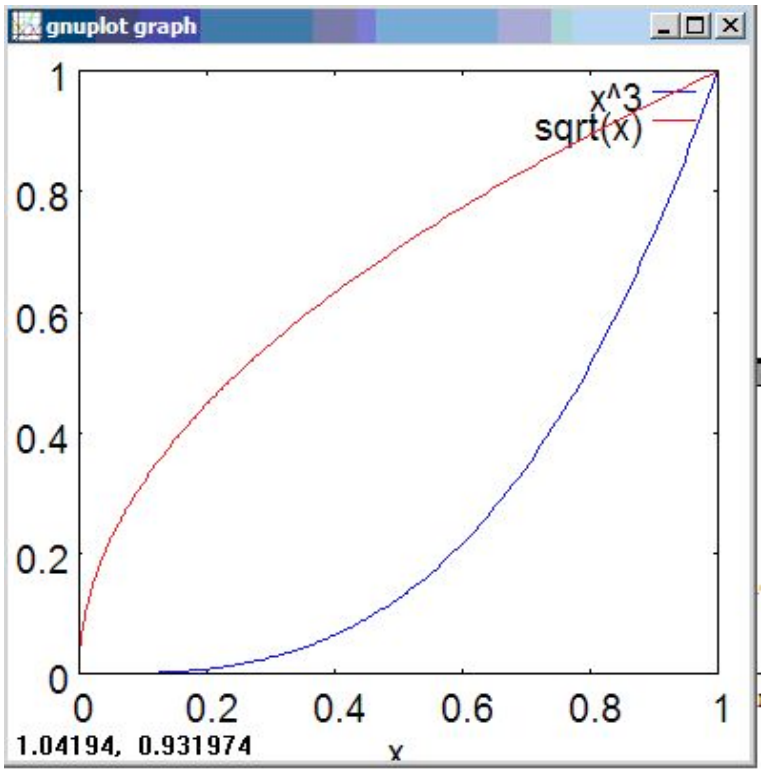


Ilustração 7 Gráfico do par de funções [x^2, sqrt(x)]

Exemplo 4.1-1

É possível assinalarmos as funções[6] assim como os eixos coordenados. Retomando o exemplo anterior:

(%i2) plot2d([x^3, sqrt(x)], [x,0,1], [y,0,4], [legend, "Grafico 1", "Grafico 2"], [xlabel, "Eixo - X"], [ylabel, "Eixo - Y"]);

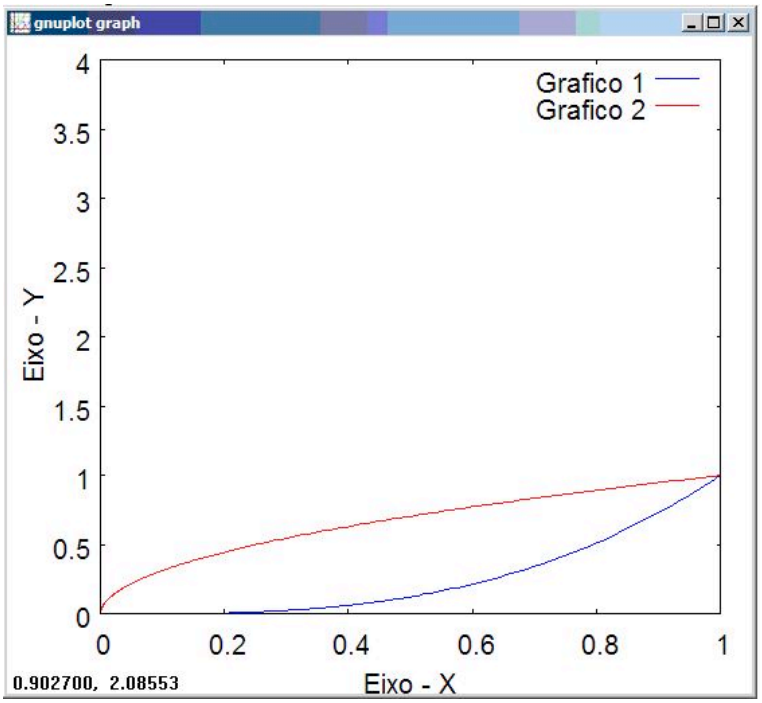


Ilustração 8 Gráfico legendado do par de funções [x^2, sqrt(x)]

4.1.2 Gráficos de Dados Discretos

Nas ciências experimentais, a recolha e análise de dados discretos, normalmente resultantes de experiências ou observações, tem um papel fulcral na obtenção de conclusões sobre determinados fenómenos e ou teorias. A opção ***discrete*** do ***plot2d*** permite representar lista de pontos.

Exemplo 2: Considere o registo de 4 em 4 horas da evolução da temperatura (em graus centigrados) durante 24 horas na cidade do Porto. [[0h, 10º],[4h,12º],[8h,15º], [12h, 20º], [15h, 25º], [20h, 18º]]. Represente os dados graficamente.

Resolução:

(%i1) temperaturas:[[0, 10],[4,12],[8,15], [12, 20], [15, 25], [20, 18]];

(%o1) *[[0,10],[4,12],[8,15],[12,20],[15,25],[20,18]]*

[6] *Ao colocar legendas nos gráficos o Gnuplot não aceita caracteres especiais, como, "ç"; nem acentos ortográficos.*

```
(%i2) plot2d([discrete, temperaturas], [style, points]);
```

Exemplo 4.1-2

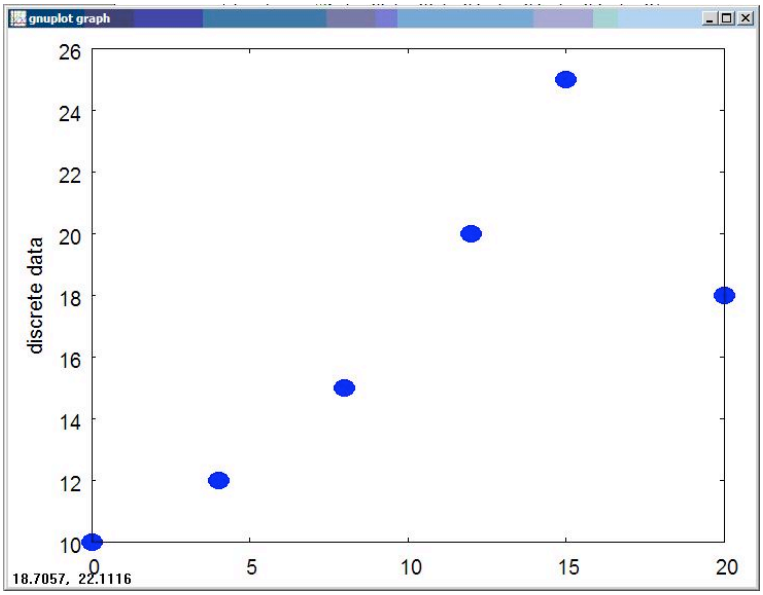


Ilustração 9 Representação gráfica dos valores da temperatura

Exemplo 3: Utilizando os dados do exemplo anterior verifique:

i. Se a função $f(x)=\frac{5}{3}x+2$, descreve aproximadamente a evolução da temperatura.

ii. Represente os dados unidos por segmentos de recta.

Resolução:

i)

```
(%i3) plot2d([[discrete, temperaturas], (5/3)*x+2],  [style, [points, 3, 4], [lines, 1, 2], [legend, "dados", "funcao teorica"]), [xlabel, "Tempo (Horas)"], [ylabel, "Temperatura (ºC)"]];
```

Exemplo 4.1-3

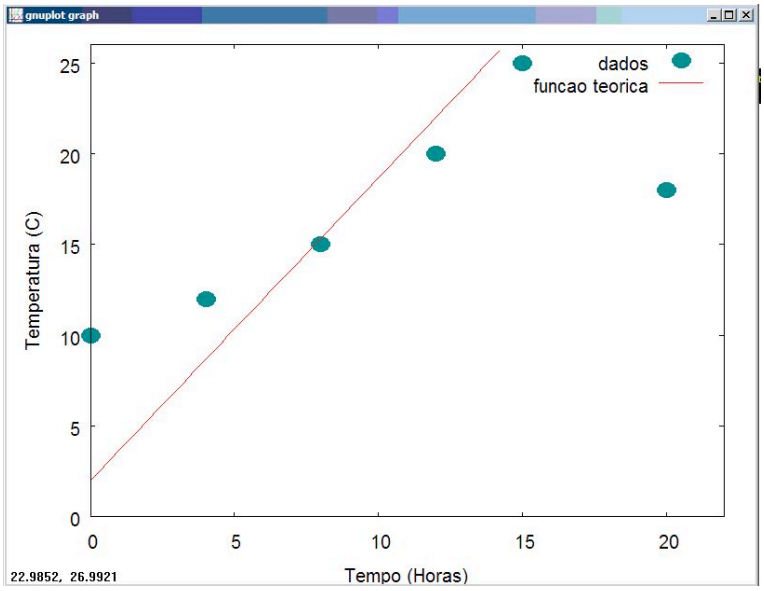


Ilustração 10 Representação das temperaturas e da função teórica

☑ *Em conclusão a recta não acompanha bem o conjunto de todos pontos representados.*

ii)

```
(%i4) plot2d([ discrete, temperaturas]);
```

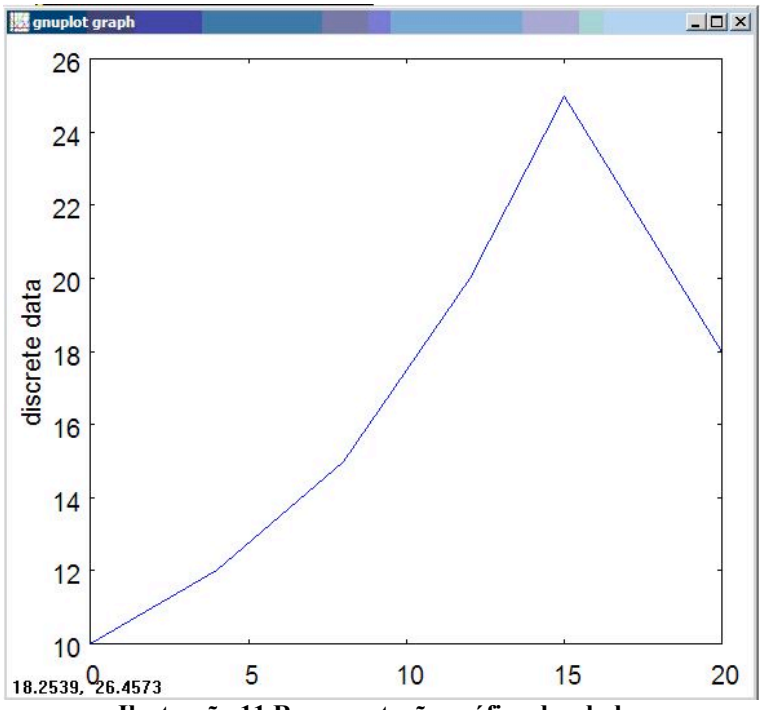


Ilustração 11 Representação gráfica dos dados

4.1.3 Gráficos de Funções Paramétricas

A representação de funções paramétricas pode se feita no *MAXIMA* através da opção ***parametric*** do ***plot2d.***

Exemplo 4: Represente a função paramétrica *(2cos (t), 2 sin(t))* no intervalo *[0, 2π].*

Resolução:

```
(%i3)  plot2d([parametric, 2*cos(t), 2*sin(t)], [t,0,2*%pi]);
```

Exemplo 4.1-4

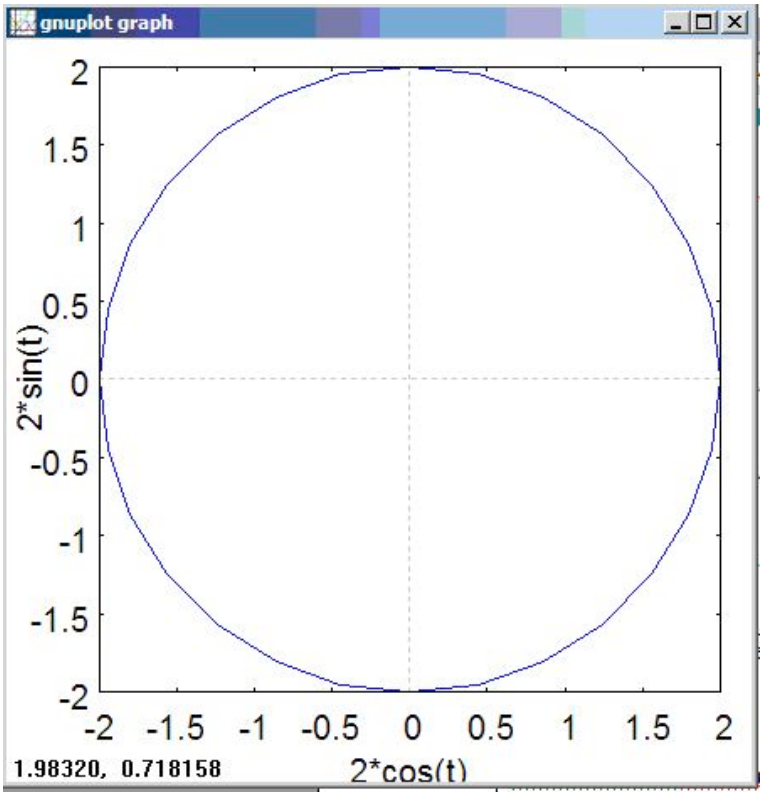


Ilustração 12 Gráfico da função paramétrica (2cos (t), 2 sin(t))

4.1.4 Gráficos em Coordenadas Polares

Um modo importante de representação de pontos num plano consiste no uso de coordenadas polares através da opção ***set polar*** do ***gnuplot_preamble.*** A opção ***set zeroaxis*** impede o traçado dos eixos coordenados.

Exemplo 5: Represente graficamente as seguintes funções.

i. $\alpha(\phi)= 0{,}5\sin(4\phi)$, $\phi \in [0, 4\pi]$.

ii. $\beta(\phi)= -3+2\cos(\phi)$, $\phi \in [0, 2\pi]$.

Resolução:

i)

```
(%i1) plot2d([0.5*sin(4*phi)],[phi,0,4*%pi],[gnuplot_preamble, "set polar; set zeroaxis;"]);
```

Exemplo 4.1-5

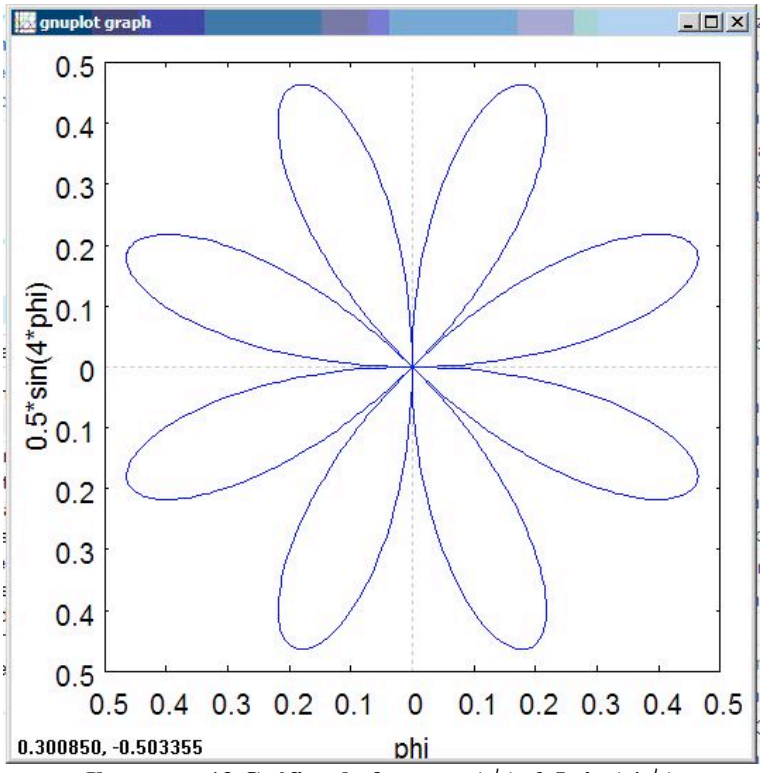


Ilustração 13 Gráfico da função $\alpha(\phi)=0{,}5 \sin(4\phi)$

ii)

```
(%i2) plot2d([-3+2*cos(phi)],[phi,0,2*%pi],[gnuplot_preamble, "set polar; set zeroaxis;"]);
```

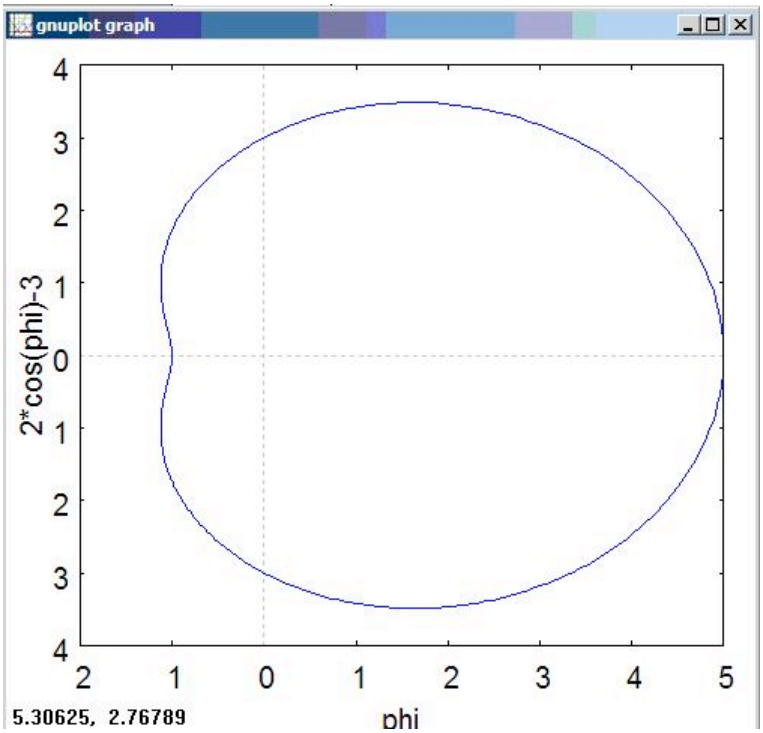


Ilustração 14 Gráfico da função $\beta(\phi)$=-3+2cos (ϕ) em coordenadas polares

4.1.5 Gráficos de Funções Implícitas

Existem várias maneiras de representar curvas no plano. Entre estas, destaca-se o ***implicit_plot.***

Exemplo 6: Represente graficamente a curva $x^2 - y^2 = 2$ no intervalo [-2π; 2π] x [4,4].

Resolução:

```
(%i1)  load(implicit_plot);

(%o1)C:/Programas/MAXIMA-
5.17.0/share/MAXIMA/5.17.0/share/contrib/implicit_plot.lisp


(%i2)  implicit_plot (x^2-y^2=2, [x, -4, 4], [y, -4, 4], [gnuplot_preamble, "set
zeroaxis"]);
```

Exemplo 4.1-6

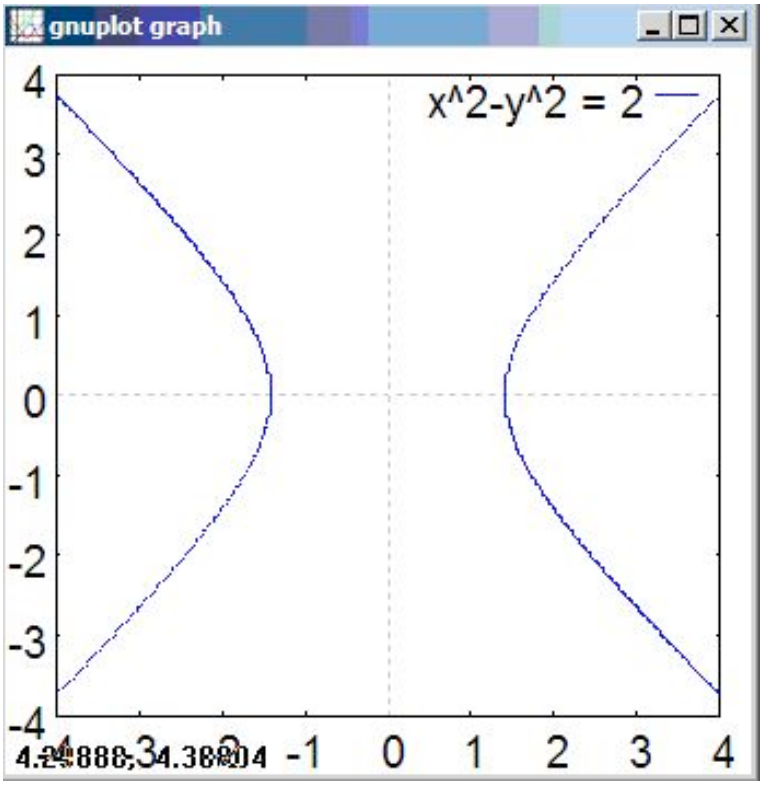


Ilustração 15 Curva: $x^2 - y^2 = 2$

4.2 Gráficos Tridimensionais

Existem distintos modos de criar gráficos tridimensionais no *MAXIMA*. O comando ***plot3d()*** permite fazê-lo**,** os seus argumentos são:

I. Função ou lista de funções.
II. Domínio.
III. Contradomínio.

Exemplo 1: Represente graficamente a função $f(x,y) = 3 - x^2 - y^2$ no domínio $[-4,4]\times[-5,5]$

Resolução:

```
(%i1) plot3d(3-x^2-y^2,[x,-4,4],[y,-5,5]);
```

Exemplo 4.2-1

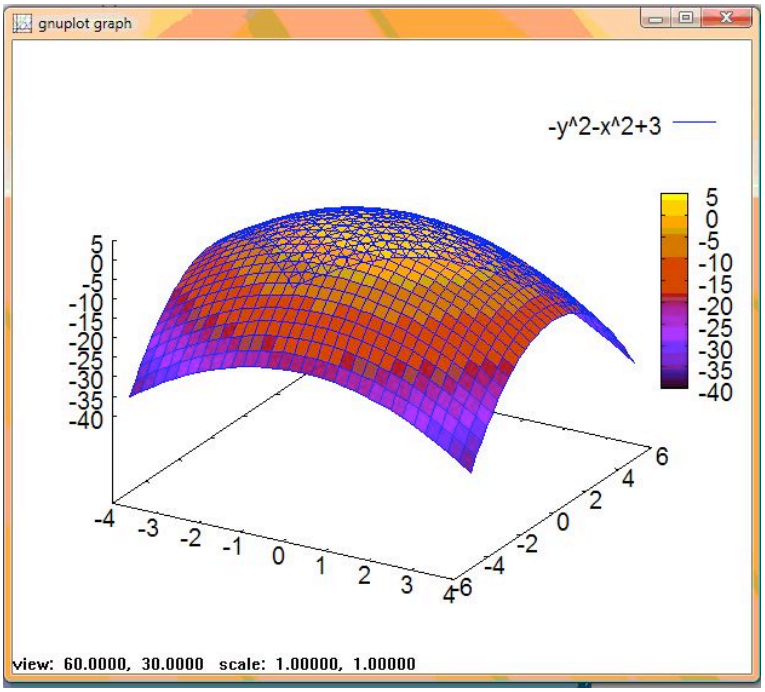


Ilustração 16 $f(x,y)=3-x^2-y^2$

Podemos visualizar um gráfico tridimensional, noutro formato, para além do ***Gnuplot***. Usando a função, $f(x,y)=3-x^2-y^2$, do exemplo anterior, e utilizando a opção ***openmath*** do ***plot_format*** (que se encontra instalado no Máxima), obtemos:

(%i2) plot3d (3-x^2-y^2, [x,-4,4], [y,-5,5], [plot_format, openmath]);

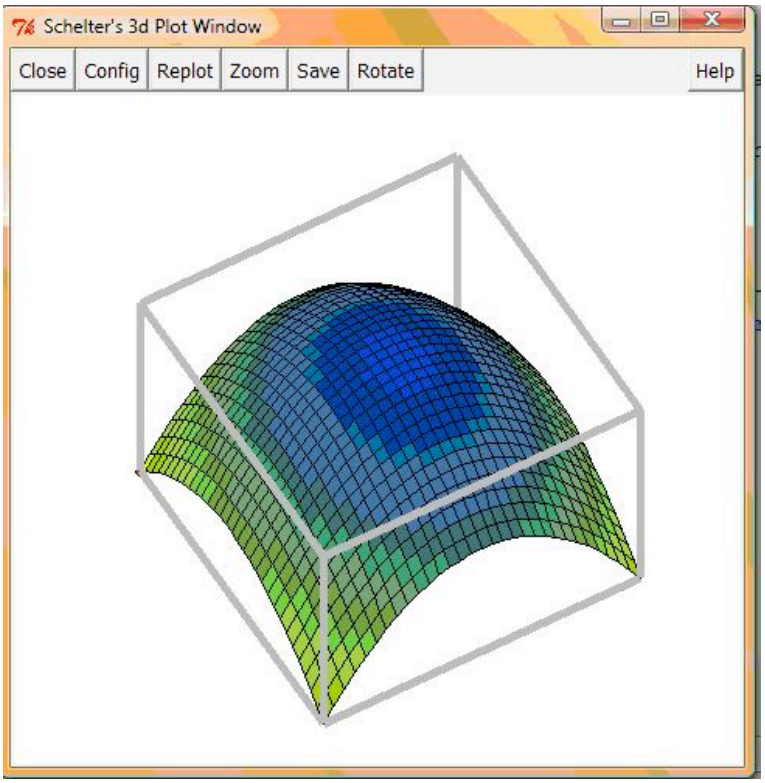


Ilustração 17 Gráfico da função $f(x,y)=3-x^2-y^2$

► Utilizando o menu **Config** (canto superior esquerdo), podemos modificar algumas opções.

4.2 Quadro Resumo: Opções gráficas:

Plot2d(expr, [x minimo, x maximo], [y mínimo, y máximo]), [opções])	Cria um gráfico a duas dimensões. ***Opções***
Style	***Impulses-*** Desenha linhas verticais entre a curva do gráfico e o eixo do x.
	Points – Para representar dados discretos.
	Parametric – Para representar funções paramétricas.
	Implicit plot- Para representar funções implícitas.
Opções	***set polar –*** Para representar funções em coordenadas polares.[7]
	set zero axis – Não desenha os eixos coordenados.
Plot3d(expr, [x minimo, x maximo], [y mínimo, y máximo]), [opções])	Cria um gráfico de uma ou três funções de duas variáveis.

Tabela 8: Opções Gráficas

[7] Por defeito assume –se que a variável independente é "phi".

Capitulo 5.

Cálculo Diferencial e Integral

"A imaginação é mais importante que o conhecimento." – Albert Einstein

Os objectivos deste capítulo são:

- ✓ Explorar comandos de manipulação algébrica de polinómios, tais como: simplificação, expansão e factorização;
- ✓ Calcular, através do *MAXIMA:*
  i. O valor de uma função num ponto;
  ii. O limite de uma função;
  iii. A derivada de uma função;
  iv. Os pontos críticos;
  v. O integral definido de uma função;

5.1 Operações com Polinómios:

No *MAXIMA*, é possível efectuar operações elementares sobre expressões, calcular limites, derivadas e integrais de funções de forma rápida e eficaz.

Exemplo 1:
- Factorize o polinómio $25x^2 - 10xy + y^2$;
- Expanda o polinómio $(z+k)^3(z^2-k)^5$;

Resolução:

(%i1) factor (25*x^2 -10*x*y +y^2);
(%o1) $(5x-y)^2$

(%i2) expand ((z + k) ^ 3 * (z - k)^2);
(%o2) $z^5 + kz^4 - 2k^2z^3 - 2k^3z^2 + k^4z + k^5$

Exemplo 5.1-1

5.2 Limites e Continuidade:

Por vezes, é necessário calcular o valor de uma função num ponto específico.

Exemplo 1: Calcule o valor da função $f(x) = x + \cos x$ no ponto $x = \pi$.

Resolução:

(%i1) f(x):= x + cos (x);

(%o1) *f(x):= x + cos (x);*

(%i2) f(%pi)

(%o2)*%pi-1*

Exemplo 5.2-1

***Definição*:** Seja f (x) uma função real de variável real e c um ponto de $\Re$. Seja K o limite de f(x), quando x converge para c. A notação habitual é:

$$\lim_{x \to c} f(x) = K$$

Exemplo 2: Calcule o limite da função $\frac{x^2+x+1}{x+1}$ no ponto $x=1$.

Resolução:

```
(%i1) limit ((x^2+x+1)/(x+1), x,1);
(%o1) 3/2
```

Exemplo 5.2-2

Exemplo 3: Calcule o limite, à direita e à esquerda, da função $\frac{1}{x-1}$ no ponto $x=1$.

Resolução:

```
(%i1) limit(1/(x-1),x,1, plus);
(%o1) inf
(%i2) limit(1/(x-1),x,1,minus);
(%o2) -inf
```

Exemplo 5.2-3

Exemplo 4: Calcule os limites:

a) $\lim_{x\to+\infty} \frac{1}{\sqrt{x}}$;

b) $\lim_{x\to-\infty} \frac{e^x-e^{-x}}{e^x+e^{-x}}$

c) $\lim_{x\to 1} \frac{1}{x-1}$

Resolução:

```
(%i1) limit (1/sqrt(x), x, inf);
(%o1) 0
(%i2) limit ((exp(x)-exp(-x))/(exp(x)+exp(-x)),x, minf);
(%o2) -1
```

[8] Se a expressão de saída for **ind ou und,** significa que o limite calculado é indefinido.

(%i3) limit (1/ (x-1), x, 1);

(%o3) infinity ← representa o infinito complexo.

Exemplo 5.2-4

5.3 Derivadas:

Definição: A derivada de uma função f no ponto x, $f'(x)$, é definida, a partir da noção de limite, por:

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

A interpretação geométrica da derivada de uma função num ponto é o declive da recta tangente nesse ponto. É importante salientar que se uma função f é derivável no ponto x=c, então a função é contínua nesse ponto.

Exemplo 1: Calcule a primeira e a segunda derivada da função $f(x)=x^3\sqrt{x^2-1}$ no ponto $x=2$.

Resolução:

(%i1) f(x):=x^3 * sqrt (x^2-1); ← definição da função $f(x)$

(%o1) $3x^2\sqrt{x^2-1}$

(%i2) define(df1(x),diff(f(x),x,1)); ← o comando ***define*** permite definir $f'(x)$ através do comando ***diff***.

(%o2) $\mathrm{df1(x)}:=3x^2\sqrt{x^2-1}-\frac{x^4}{\sqrt{x^2-1}}$

(%i3) df1(2); ← avalia $f'(2)$

(%o3) $12\sqrt{3}+\frac{16}{\sqrt{3}}$

(%i4) define(df2(x),(diff(f(x),x,2))); ← define a segunda derivada

(%o4) $\mathrm{df2(x)}:=6x\sqrt{x^2-1}+\frac{7x^3}{\sqrt{x^2-1}}-\frac{x^5}{\left(x^2-1\right)^{\frac{3}{2}}}$

(%i5) df2(2); ← avalia $f''(2)$

(%o5) $12\sqrt{3}+\frac{136}{3\sqrt{3}}$

Exemplo 5.3-1

Exemplo 2: Determine a equação da recta tangente ao gráfico da função $f(x)=2\sin(x)+\cos(2x)$ no ponto $(\pi,1)$, representando a função e a recta graficamente.

Resolução:

(%i1) f(x):=2* sin(x) + cos(2*x);

(%o1) $f(x):=2\sin(x)+\cos(2x)$

Para determinar a equação da recta tangente no ponto $(\pi,1)$, calcula - se $f'(\pi)$

(%i2) define(df1(x),(diff(f(x),x,1)));

(%o2) $2\cos(x)-2\sin(2x)$

(%i3) tangente(x):=df1 (%pi)*x+f(0)$

A equação da recta tangente tem a forma y(x)=mx+b. A equação é: y(x)=-2x+π+2.

(%i3) plot2d ([f(x), tangente(x)], [x,0,2*%pi], [y,-4,2]);

Exemplo 10

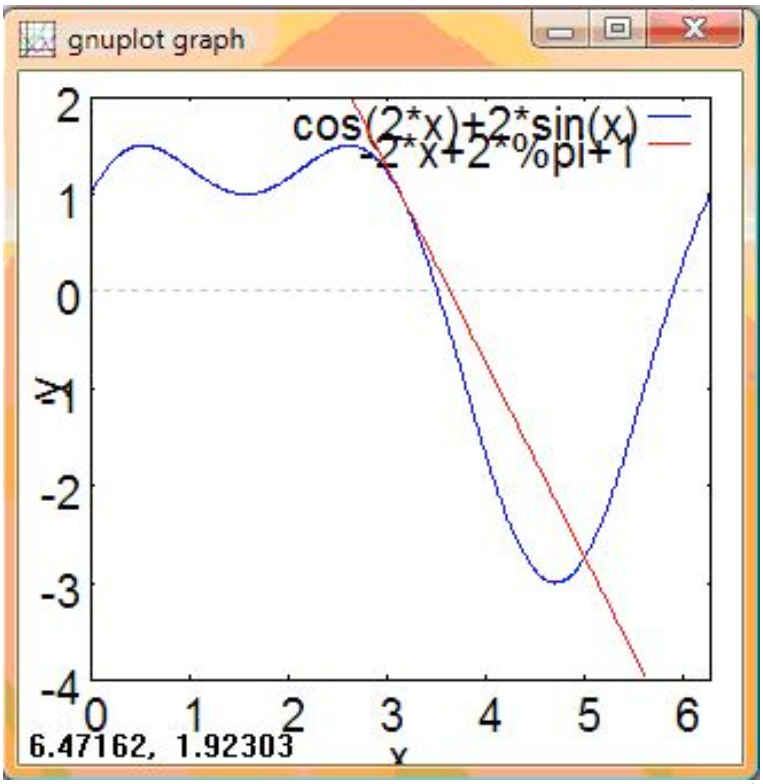


Gráfico de $f(x)=2\sin(x)+\cos(2x)$ **e da recta tangente ao ponto** $(\pi,1)$

Frequentemente, é necessário estudar o comportamento de uma função num determinado intervalo ***I***. Para isso, é necessário determinar os pontos críticos da função, assim como a monotonia. O *MAXIMA* é útil neste tipo de tarefas.

Exemplo 3: Calcule os pontos críticos e os pontos de descontinuidade da função $f(x)=\frac{x^2+3x}{x^3-5x^2}$. Represente graficamente a função.

Resolução:

(%i1) f(x):= (x^2+3x)/(x^3 -5*x^2);

(%o1) $f(x):=\frac{x^2+3x}{x^3-5x^2}$

(%i2) pcriticos1: solve(diff(f(x), x, 1)); ← os pontos críticos são os zeros de $f'(x)$

(%o2) $\left[x=-2\sqrt{6}-3, x=2\sqrt{6}-3\right]$

(%i3) pdesc1: solve(x^3-5*x^2); ← os pontos de descontinuidade são os zeros do denominador de $f(x)$

(%o3) $\left[x=0, x=5\right]$

Exemplo 5.3-3

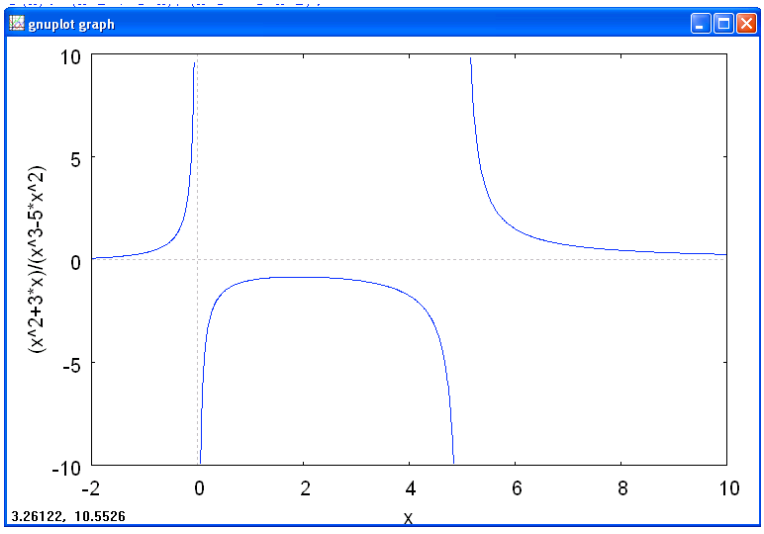


Ilustração 18 Gráfico da função $f(x)=\frac{x^2+3x}{x^3-5x^2}$

Exemplo 4: Dada a função racional $g(x) = -\frac{1}{(x-1)^2}$, determine:

a) Primeira derivada;
b) Segunda derivada;
c) Intersecção da função com eixo y;
d) A paridade da função;
e) O comportamento da função, quando $x \to 1^-, x \to 1^+$;

Resolução:

a)

(%i1)) g(x):= (-1)/((x-1)^2);

(%o1) $g(x) = -\frac{1}{(x-1)^2}$

(%i2) define(dg1(x),(diff(g(x),x,1)));

(%o2) $dg1(x) = \frac{2}{(x-1)^3}$

b)

(%i3) define(dg2(x),(diff(g(x),x,2)));

(%o3) $dg2(x) = -\frac{6}{(x-1)^4}$

c)

(%i4) soy: g(0);

(%o4) -1

d)

(%i5) ev (g(x) = g(-x),pred); ← *A função não é par, porque não verifica: g (x)=g (-x).*

(%o5) ***false***

e)

(%i6) is (g(x) = - g(-x));

(%o6) ***false*** ← *A função não é impar, porque não verifica: g (x)=- g (-x).*

f)

(%i7) limit (g(x), x, 1, plus); ← O limite da função para valores á direita de 1 é:

(%o7) $-\infty$

(%i8) limit (g(x), x, 1, minus); ← O limite da função para valores á esquerda de 1 é:

(%o8) $-\infty$

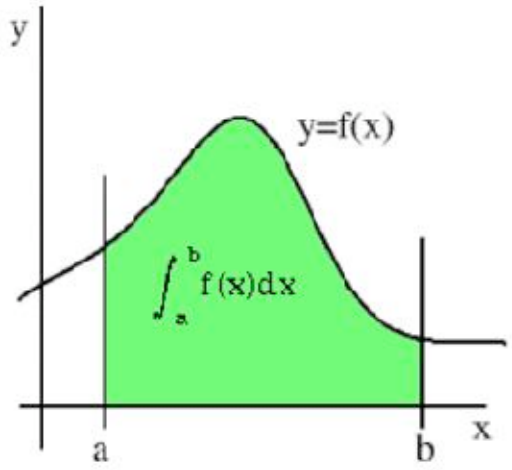
Exemplo 5.3-4

5.4 Primitiva e Integrais:

O cálculo de integrais revela-se de grande importância em áreas como a Física e Matemática.

Definição: Se uma função *f* é contínua e não negativa num intervalo fechado *[a,b]* , então a área da região delimitada pelo gráfico de *f*, pelo eixo dos *x* e pelas rectas verticais *x=a* e *x=b* é dada por:

$$Área = \int_a^b f(x)dx$$



Exemplo 1: Calcule a primitiva da função $f(x) = sen^5 x \cos x$.

Resolução:

(%i1) f(x):=sin(x)^5 * cos(x);

(%o1) $f(x) = sen^5 x \cos x$

(%i2) Intg:integrate (f(x), x);

(%o2) $\frac{\sin^6 x}{6}$ ←O comando ***integrate*** não apresenta a constante de integração. Desta forma, a primitiva da função é a que está indicada a menos da soma de uma constante arbitrária.

Exemplo 5.4-1

Exemplo 2: A probabilidade de um aluno se lembrar de uma percentagem entre a e b do material apreendido durante uma aula é dado por:

$$P_{a,b} = \int_a^b \frac{15}{7} x\sqrt{1-x}\,dx$$

, onde x representa a percentagem do conteúdo lembrado após a aula.
Escolhe-se um aluno aleatoriamente. Determine a probabilidade do aluno se lembrar de 50% a 70 % da matéria dada. Represente essa probabilidade graficamente e torne este gráfico apelativo através da opção ***impulses***.

Resolução:

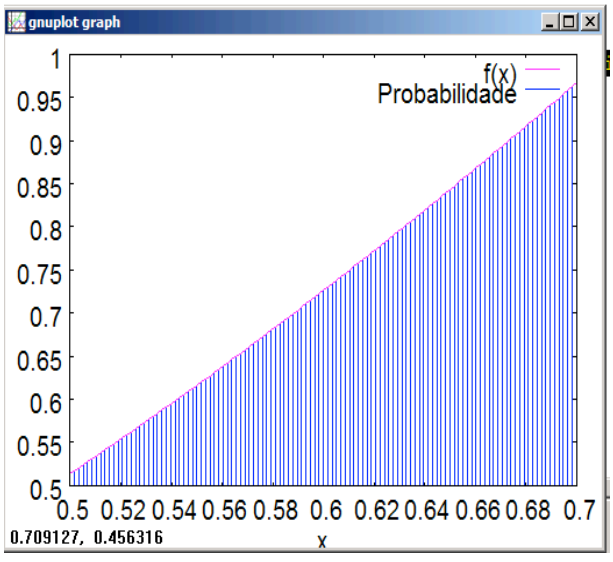
(%i1) p(x): =((15*x)/7)*sqrt(1-x); ← definição da função integranda.

(%o1) $\frac{15}{7} x\sqrt{1-x}$

(%i2) integrate (p(x), x, 0.50,0.70); ← calculo de $P_{0.50,0.70}$.

(%o2) ***0.1610680363702***

Exemplo 5.4-2



(%i3) plot2d([((15*x)/7)*sqrt(1-x), ((15*x)/7)*sqrt(1-x);], [x,0.5,0.7], [style,[lines,1,3], [impulses,1,2]], [legend,"f(x)","Probabilidade"]);

Exemplo 3: Calcule o integral $\int_0^1 \sin\left(\frac{1}{x+0.1}\right)dx$

Resolução:

(%i1) p:sin (1/(x+0.1));

(%o1) $\sin\left(\frac{1}{x+0.1}\right)$

```
(%i2)  integrate(p,x,0,1);
`rat' replaced 0.1 by 1/10 = 0.1
`rat' replaced 0.1 by 1/10 = 0.1
`rat' replaced 0.1 by 1/10 = 0.1
`rat' replaced 0.1 by 1/10 = 0.1
`rat' replaced 0.1 by 1/10 = 0.1
`rat' replaced 0.1 by 1/10 = 0.1
`rat' replaced 0.1 by 1/10 = 0.1
`rat' replaced 0.1 by 1/10 = 0.1
```

(%o2) $\int_0^1 \sin\left(\frac{1}{x+0.1}\right) dx$

Exemplo 5.4-3

Não é possível resolver analiticamente este integral através do comando ***integrate***. Será necessário aplicar métodos numéricos para calcular este tipo de integrais.

O *MAXIMA* disponibiliza um pacote designado ***Quadpack***. Neste estão presentes uma colecção de comandos que implementam algoritmos para o cálculo numérico de integrais definidos, entre os quais o ***Romberg*** correspondente ao método com o mesmo nome.

Vejamos um exemplo de como utilizar o pacote ***Quadpack*** e o comando ***Romberg***:

(%i2) quad_qags (p, x, 0, 1);

(%o2) ***[0.59450627843597,1.9863238307029738*10^{-10} 10,147,0]***

O comando ***quad_qags*** retorna uma lista com quatro elementos onde:

1. É o valor do integral;
2. É o erro absoluto estimado;
3. É o número de avaliações da função integranda;
4. É um código de erro;

O código de erro pode tomar 7 valores, que se caracterizam:

0. Se não foi encontrado nenhum problema
1. Se foram utilizados muitos sub-intervalos.
2. Se foi detectado um erro de arredondamento excessivo.
3. O integral comporta-se de um modo bastante oscilatório.
4. Não houve convergência.
5. O integral provavelmente é divergente ou converge muito lentamente.
6. O integral não é válido.

Vamos explorar o comando ***Romberg*** para calcular o valor do integral. Numa primeira estância temos que "chamar" o comando para que este fique disponível na secção para uso do utilizador. Assim, façamos:

(%i3) load (romberg);

(%o3) *C:/Programas/Máxima-5.17.0/share/MAXIMA/5.17.0/share/numeric/romberg.lisp*

(%i4) romberg (p,x,0,1);

(%o4) ***0.59450623879744***

5.5 Quadro Resumo: Comandos de Calculo Integral e Diferencial

Comando	Descrição
expand (*expressão*)	Expande uma expressão
factor (*expressão*)	Factoriza a expressão dada, independentemente do número de variáveis, em factores irredutíveis.
subst (*a,b,c*)	Substitui o valor ***a*** em ***b*** na expressão/função, ***c.***
[1]limit (*função, variável, limite pretendido*)	Calcula o limite de uma função.
diff(*função, variável, n*)	Calcula a n-ésima derivada da função na variável pretendida.
integrate (*função, variável*) ou integrate (*função, variável, a*, b)** a* e b* são os limites de integração superior e inferior, respectivamente.	Calcula o integral da função com ou sem limites de integração.
romberg (*função, variável, a*, b)** a* e b* são os limites de integração superior e inferior, respectivamente.	Calcula o valor de um integral através do método numérico de Romberg.

Tabela 9: Comandos de calculo diferencial e integral

5.6 Quadro Resumo: Comandos sobre Limites

No cálculo de limites o comando ***limit*** usa símbolos especiais, que encontramos descritos no seguinte quadro:

Símbolo	Descrição
inf	$+\infty$
minf	$-\infty$
plus	Calcula o limite de um valor pela direita.
minus	Calcula o limite de um valor pela esquerda.

Tabela 10: Comandos sobre limites

Capitulo 6.

Estatística Descritiva e Probabilidades

" Não há ramo da Matemática, por mais abstracto que seja, que não possa um dia vir a ser aplicado aos fenómenos do mundo real" – Lobachevsky

O *MAXIMA* é uma valiosa ferramenta em análise de dados e estatística.

Neste capítulo, introduziremos comandos concernentes aos seguintes tópicos:

- ✓ Cálculo de probabilidades de distribuições discretas;
- ✓ Cálculo de probabilidades de distribuições contínuas;
- ✓ Leitura de ficheiros no formato *txt*;
- ✓ Análise estatística de dados;
- ✓ Representação estatística de dados;

6.1 Distribuições Discretas

6.1.1 Distribuição Binomial

Para se efectuar cálculos estatísticos no *Máxima* é necessário "chamar" o pacote ***distrib***.

Em Estatística, a distribuição binomial é útil na caracterização probabilística do número de sucessos em n provas de Bernoulli realizadas de forma independente e com probabilidade de sucesso igual p**.**

Exemplo 1: Um fabricante de caixas de parafusos recebeu, numa dada altura, um elevado número de queixas quanto à qualidade dos parafusos que fabrica. Após uma análise de produção verificou-se que 15% dos parafusos saíam defeituosos.
Sabe-se que cada caixa fornecida aos compradores possui vinte pregos.

A. Calcule a probabilidade de numa caixa de parafusos:
   i) Haver apenas um parafuso defeituoso.
   ii) Haver mais do que três parafusos defeituosos.
B. Calcule o valor esperado de parafusos defeituosos numa caixa.
C. Simule uma amostra de quinze valores correspondentes ao número de parafusos defeituosos contidos em quinze caixas analisadas aleatoriamente e construa o respectivo gráfico de barras.

Resolução:

A.

Seja a variável aleatória X:

X= "*Números de parafusos defeituosos numa caixa de vinte parafusos"*

Distribuição de X:

X~Bi(20; 0.15)

i)
P(X≤1)

```
(%i1) load(distrib);

(%o1)C:/Programas/Máxima-
5.17.0/share/MAXIMA/5.17.0/share/contrib/distrib/distrib.mac

(%i2) cdf_binomial(1,20,0.15);

(%o2) 0.17555787608868
```

ii)

P(X>3)= 1-P(X≤3)

(%i3) 1-cdf_binomial(3,20,0.15);

(%o3) *0.352274825843*

Exemplo 6.1-1

B.

(%i4) mean_binomial (20,0.15);

(%o4) *3.0*

C.

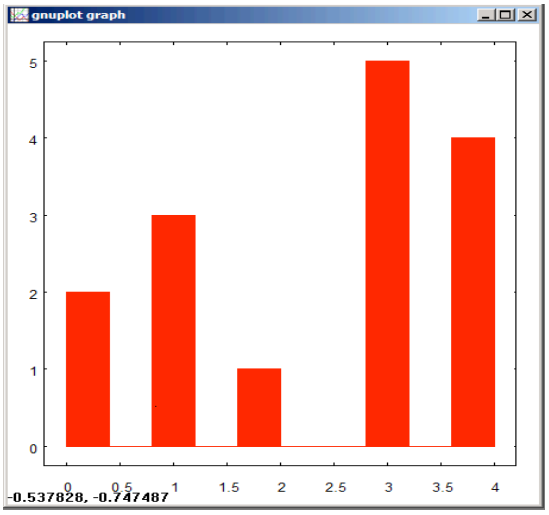
(%i5) load (descriptive);

(%o5)*C:/Programas/Máxima-5.17.0/share/MAXIMA/5.17.0/share/contrib/descriptive/descriptive.mac*

(%i6) lista: random_binomial (20,0.15,15);

(%o6) *[5, 2, 1, 6, 2, 3, 2, 1, 5, 1, 4, 4, 3, 3, 3]*

Exemplo 6.1-1



(%i7) histogram(lista);

6.1.2 Distribuição de Poisson

A distribuição de Poisson é frequentemente usada na contagem de ocorrências de certo tipo de evento em períodos fixos de tempo. Os eventos podem ser: chegadas, partidas, acidentes, falhas de equipamento, testemunhos verdadeiros em tribunal, etc.

A distribuição de Poisson tem a particularidade de possuir o valor esperado e a variância iguais ao parâmetro que define a distribuição, **λ.**

Exemplo 2: Numa aldeia existe uma padaria onde é vendido um determinado tipo de doce regional. O reabastecimento deste doce é feito, com periodicidade semanal, de sete unidades. Uma vez esgotado o stock semanal, o cliente tem que se dirigir a outra padaria. Estudos realizados revelam que a procura semanal do referido doce pelos habitantes da aldeia é uma variável aleatória de ***Poisson*** de média 5. Cada cliente compra no máximo uma unidade.

A. Calcule a probabilidade de numa determinada semana:
i) A padaria vender mais que quatro unidades.
ii) Haja ruptura de stock.

Resolução:

A.

Seja a variável aleatória X:

X= "*Procura semanal do referido doce na padaria***"**

Distribuição de X:

X~Po(5)

i) *P(X>4)*

```
(%i1) load(distrib);

(%o1)C:/Programas/Máxima-
5.17.0/share/MAXIMA/5.17.0/share/contrib/distrib/distrib.mac

(%i2) 1- cdf_poisson(4,5), numer;

(%o2) 0.55950671493479
```

ii) *P(X>7)*

```
(%i3) 1- cdf_poisson(7,5), numer;

(%o3) 0.13337167407001
```

Exemplo 6.1-2

6.2 Distribuições Contínuas

6.2.1 Distribuição Exponencial

A distribuição exponencial é uma distribuição contínua aplicada em vários problemas, nomeadamente, em filas de espera. Esta distribuição é definida pelo parâmetro **β**, denominado a média, que estabelece a média de chegadas por hora, por exemplo. Em geral a distribuição exponencial é aplicada a serviços por unidade de tempo.

Exemplo 1: O prazo de operação de uma máquina de embalagens de frascos sem interrupções para manutenção tem distribuição exponencial com média de duas horas. **A.** Qual é a probabilidade de essa máquina conseguir operar mais do que uma hora sem interrupção?

Resolução:

Seja a variável aleatória X:

X= "*Tempo entre interrupções (em horas)***"**

Distribuição de X:

X~Exp(β)

Parâmetro β:

$$\begin{cases} \beta : E(X) = 2 \\ \dfrac{1}{\beta} = 2 \\ \beta = 0.5 \end{cases}$$

P(X≥1)

```
(%i1) load(distrib);
```

(%o1)*C:/Programas/Máxima-5.17.0/share/MAXIMA/5.17.0/share/contrib/distrib/distrib.mac*

```
(%i2)  1-cdf_exp (1,0.5);
```

(%o2) *0.60653065971263*

Exemplo 6.2-1

6.2.2 Distribuição Normal

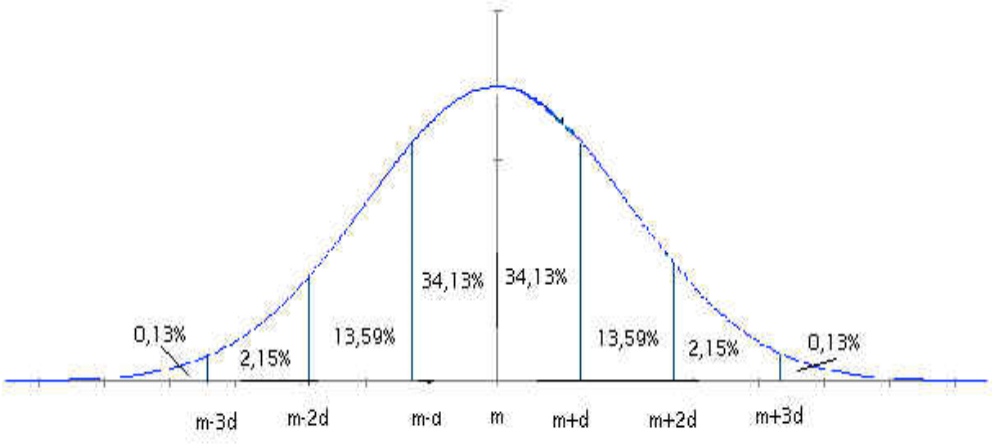
A variável aleatória X tem distribuição Normal com parâmetro μ e σ² se a sua função densidade de probabilidade é dada por:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}\ ,\ -\infty < x < +\infty$$

Os parâmetros designam-se por:

- μ é o valor esperado, isto é, média de x.
- σ² é a variância de X (σ² >0).

A notação para uma distribuição Normal é X~N ***(μ,σ²).***
O gráfico da função densidade de probabilidade de uma variável aleatória com distribuição N ***(μ,σ²)*** é a famosa curva em forma de sino, também dita **curva de Gauss** abaixo representada:



Exemplo 2: O tempo gasto, em minutos, de um exame de Estatística, tem distribuição Normal. O valor esperado é 120 minutos e a variância 15² minutos².
Sorteando um aluno ao acaso, qual é probabilidade de que ele demore no máximo 100 minutos a realizar o exame?

Resolução:

Seja a variável aleatória X:

X= "*Tempo de realização do exame de Estatística***"**

Distribuição de X

X~N (***120,15²).***

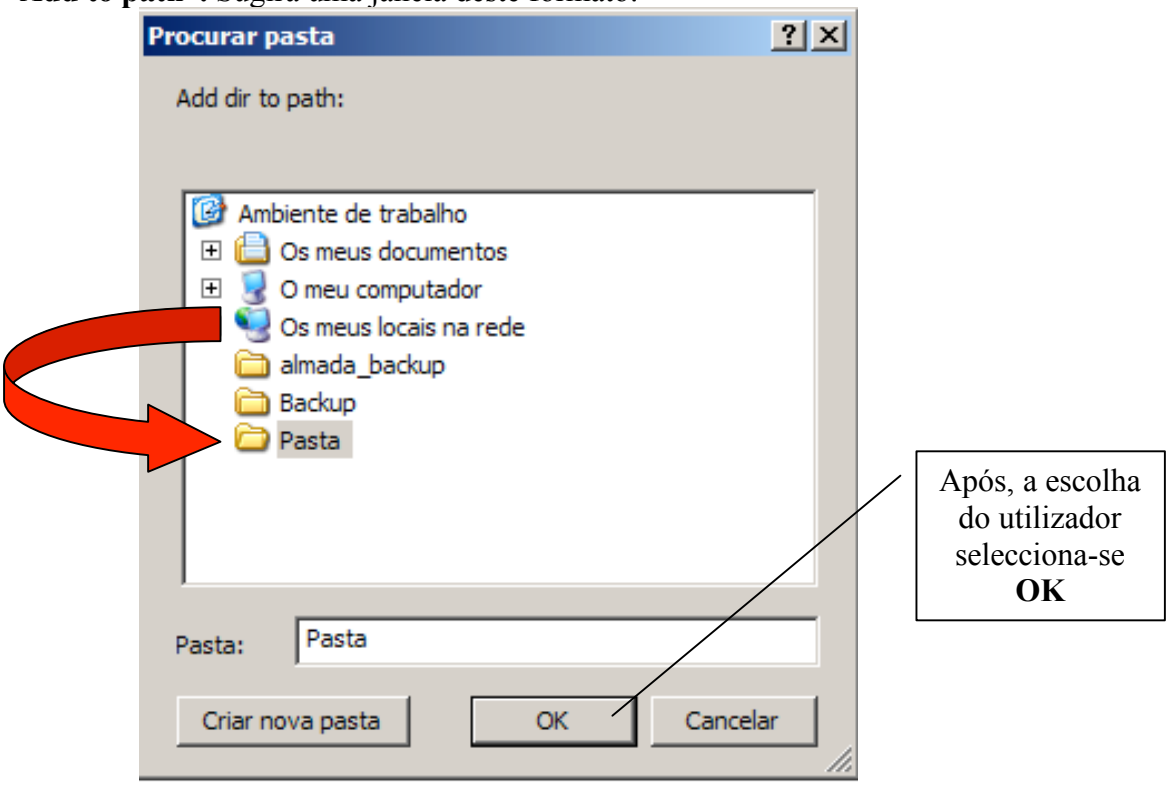
P(X<100)

```
(%i1) load(distrib);

(%o1)C:/Programas/Máxima-
5.17.0/share/MAXIMA/5.17.0/share/contrib/distrib/distrib.mac

(%i2)  cdf_normal (100, 120, 15);

(%o2) 0.091211219725868
```

Exemplo 6.2-2

Em diversas situações, dispomos de um conjunto de dados que pretendemos estudar a nível estatístico para consequentemente tirar conclusões. O *MAXIMA* possui ferramentas que nos ajudam nesta tarefa. Suponhamos que possuímos um conjunto de dados num ficheiro de texto, com extensão .txt, referentes à temperatura registada durante um determinado número de dias numa cidade. A questão é como migrar esses dados do ficheiro .txt para o *MAXIMA*?
A resposta é simples. Basta apenas seguir os seguintes passos:

1. Num directório, à escolha do leitor, coloque o ficheiro .**txt** numa pasta. (Neste exemplo, usou-se o ambiente de trabalho como **directório** e a pasta designada por "Pasta").
2. No menu do ***WxMaxima*** selecciona-se a opção **"MAXIMA"** e em seguida **"Add to path".** Sugira uma janela deste formato:



3. Surgirá na interface do ***WxMaxima*** uma linha da forma:

file_search_MAXIMA : cons(sconcat("C:/Documents and Settings/B/Ambiente de trabalho/Pasta/###.{lisp,mac,mc}"), file_search_MAXIMA);

4. Carregamos simultaneamente em **Shift+Enter;**

5.Digita-se o seguinte:

(%i1) dados: read_matrix(file_search("D.txt"));

6.Aparecerá os dados em forma de matriz.

(%o2)

$$\begin{bmatrix} 25 \\ 22 \\ 21 \\ 25 \\ 26 \\ 29 \\ 22 \\ 24 \\ 19 \\ 23 \\ 25 \\ 27 \\ 26 \\ 18 \\ 18 \\ 25 \\ 25 \\ 16 \\ 30 \\ 27 \end{bmatrix}$$

7. Para calcularmos a média, mediana, desvio-padrão, variância, etc, é necessário "chamar" o pacote ***descriptive.***

(%i3) load (descriptive);

(%o3)*C:/Programas/MAXIMA-5.17.0/share/MAXIMA/5.17.0/share/contrib/descriptive/descriptive.mac*

Calculemos:

Média

(%i4) mean(dados), numer;

(%o4) *23.65*

Variância
(%i5) var(dados), numer;

(%o5) *13.4275*

Desvio – padrão:
(%i6) std(dados), numer;

(%o6) *3.66435533211505*

Amplitude da amostra:
(%i7) range (dados);

(%o7) *14*

Mediana
(%i8) median (dados);

(%o8) *25*

Valor Mínimo
(%i8) mini (dados);

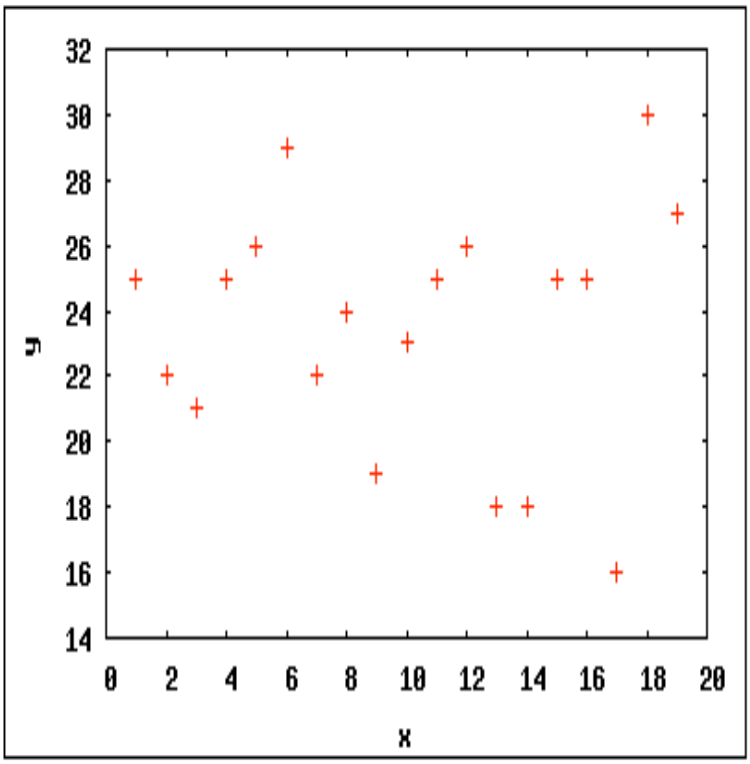
(%o8) *16*

Valor Máximo
(%i9) maxi (dados);

(%o9) *30*

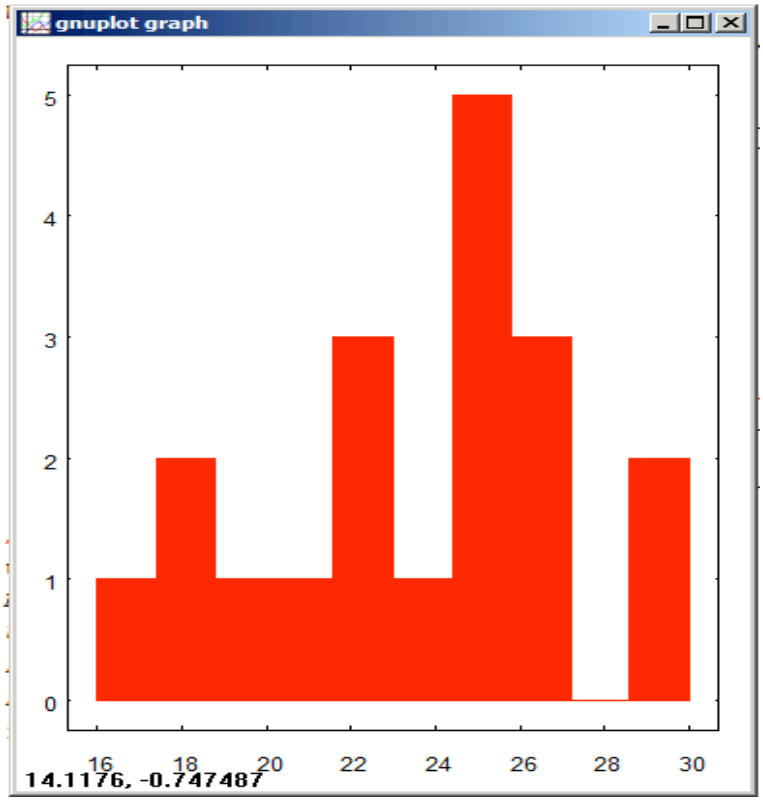
Podemos obter várias representações gráficas dos dados:

Representação gráfica dos pontos



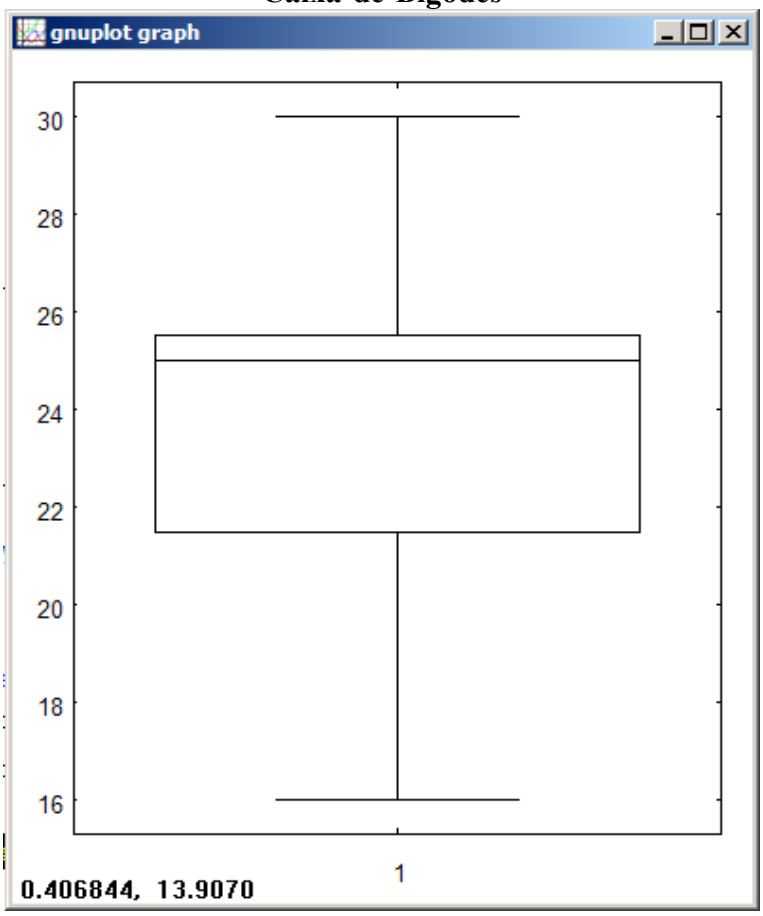
(%i10) dataplot (dados);

Diagrama de barras



(%i11) histogram (dados);

Caixa-de-Bigodes



(%i12) boxplot (dados);

6.3Método de Monte Carlo

O método de Monte Carlo é um método estatístico utilizado em simulações estocásticas com diversas aplicações em áreas como matemática, física, biologia, etc. Este método tem sido utilizado em aproximações numéricas, nomeadamente no cálculo do valor do integral de funções complexas. O método consiste na geração de uma amostra de números pseudo - aleatórios de alguma distribuição de probabilidade e o uso dessa amostra para calcular o valor do integral como um valor esperado. Consideremos o Integral: $\theta = \int_a^b h(x)dx$.

Quando é possível factorizar a função, h(x) do seguinte modo: h(x) = g(x)f(x), onde f(x) é a função densidade de probabilidade da variável aleatória X definida no intervalo [a,b], passa a ter: $\theta = \int_a^b g(x)f(x)dx = E[g(x)]$.

Deste modo o problema da avaliação do integral definido simples transforma-se num problema de estimação bastante familiar: a estimação do valor esperado da variável

aleatória, g(X). Caso se considere que X~Uniforme(a,b), uma estimativa razoável e centrada para θ é: $\theta \approx (b-a)\frac{\sum_{i=1}^{n} h(x_i)}{n}$.

Exercício de Aplicação 5: Calcule o valor do integral I= $\int_0^1 4\sqrt{1-x^2}\,dx$ utilizando o método de Monte Carlo - montecarlosimples(a,b,f,n) – para n=10000.

Resolução:

```
(%i1)  load ("distrib")$

(%i2)  load ("descriptive ")$

(%i3)  F(x):=4*sqrt(1-x^2)$

(%i3)  Integral :mean(F(random_continuous_uniform (0,1,10000)));

(%o3) 3.145570252322345
```

Exercício de Aplicação 5: Método de Monte Carlo

6.4 Quadro Resumo: Comandos de Probabilidade e Estatística Descritiva

6.4.1 Distribuições Continuas e Discretas

Existe uma convenção de nome no pacote ***distrib***. Todos os nomes de funções têm duas partes, a primeira faz referência à função ou ao parâmetro que queremos calcular:

Função	Comando
Função densidade de probabilidade	pdf_*
Função distribuição de probabilidade	cdf_*
Quartil	quantile_*
Média	mean_*
Variância	var_*

Desvio padrão	std_*
Coeficiente de assimetria	skewness_*
Coeficiente de curtose	kurtosis_*
Variável estatística pseudo-aleatória	random_*

A segunda parte é uma referência explícita á distribuição que queremos aceder:

Distribuições:	**Comando**
Discretas	
Binomial	*binomial
Poisson	*poisson
Bernoulli	*bernoulli
Geométrica	*geometric
Discreta uniforme	*discrete_uniform
Contínuas	
Exponencial	*exp
Normal	*normal
Student	*student_t
Contínua uniforme	*continuous_uniform
Pareto	*pareto

6.4.2 Análise de Dados:

O pacote ***descriptive*** contém um conjunto de comandos para efectuar cálculos de estatística descritiva e desenhar gráficos estatísticos.

Comando	**Descrição**
mean (lista/matriz)	Calcula a média aritmética de um conjunto de dados de uma única variável (lista) ou várias variáveis (matriz)
var (lista/matriz)	Calcula a variância de um conjunto de dados, de uma única variável (*lista*) ou várias variáveis (*matriz*)
std (lista/matriz)	Calcula o desvio-padrão de um conjunto de dados, de uma única variável (*lista*) ou várias variáveis (*matriz*)
range (lista/matriz)	Calcula a diferença entre o maior valor e o menor valor de um conjunto de dados
median(lista/matriz)	Calcula a média aritmética de um conjunto de dados que se apresenta sob a forma de matriz ou lista.
mini(lista/matriz)	Acede ao valor mínimo de uma amostra

	de dados.
maxi (lista/matriz)	Acede ao valor máximo de uma amostra de dados.

6.3.3 Representação Gráfica de Dados:

Comando	**Descrição**
histogram (lista/matriz com uma coluna, apenas, [opções])	Desenha um gráfico de um histograma. Podemos escolher opções desde a cor do histograma até ao número de classes em que queremos dividir os dados.
dataplot (lista/matriz)	Visualização directa de dados de amostra de uma única variável (*lista*) ou de várias variáveis (*matriz*).
boxplot(lista/matriz)	Visualização directa de dados de amostra de uma única variável (*lista*) ou de várias variáveis (*matriz*).

Capitulo 7.
O *MAXIMA* como linguagem de programação

"Todas as máximas já foram escritas. Resta apenas pô-las em prática" - Blaise Pascal

Este capítulo é dedicado à programação numa perspectiva mais abrangente.
O *MAXIMA* pode ser usado de uma forma interactiva, ou seja, instruções de entrada versus respostas de saída, ou como uma linguagem de programação com a formulação de blocos constituídos por várias instruções e argumentos que posteriormente serão executados fornecendo os resultados previstos.

Abordaremos:

- ✓ Instruções de controlo do fluxo de programa:

- ✓ Instruções condicionais:

  I. ***If…then…end***;
  II. ***If…then…else…end***;

- ✓ Instruções de ciclos:

  I. ***For…step…thru do***;

  II. ***For…step…while…do***;

- ✓ Formulação de algoritmos com o comando ***block***;
- ✓ Formulação de comentários dentro de uma célula *input* através dos operadores:
- ✓ /* */
- ✓

Os comandos e as capacidades de programação serão utilizados conjuntamente com conteúdo de capítulos anteriores.

7.1 Programação no MAXIMA

Além das potencialidades do *MAXIMA* para realizar cálculos numéricos e simbólicos assim como representações gráficas, também é possível implementar algoritmos com o objectivo de optimizar o tempo e o esforço computacional. Assim, podemos voltar a reutilizar os programas formulados quantas vezes forem necessários.

7.1.1 Instrução if:

Para tomar uma decisão numa situação condicional, o *MAXIMA* tem as instruções:

If condição ***then*** instrução;

End

Instrução 1: If…then..end

If condição ***then*** instrução1;

else

instrução 2;

end;

Instrução 2:If …then…eslse…end

Exemplo 1:

```
(%i1) a:5;
(%o1) 5


(%i2) if a>0 then print("Numero Positivo");
(%o2) Numero Positivo


(%i3) b:-2;
(%o3) -2


(%i4) if b>0 then print("Numero Positivo") else (“Numero Negativo”);
(%o4) Numero Negativo
```

Exemplo 7.1-1

7.1.2 Instrução: for...end:

for variável:valor1 ***step*** valor 2 ***thru*** valor 3 ***do*** instrução;

Instrução 3:for…step…thru…do

Outra possibilidade de controlar as iterações é com a versão:

for variável:valor1 ***step*** valor2 ***while*** condição ***do*** instrução;

Instrução 4: for…step…while…do

Exemplo 2:

/* Calcula o quadrado dos números impares no intervalo de 1 até 10 */

(%i1) for i:1 step 2 thru 10 do print (i^2);

(%o1)
1
9
25
49
81
done

Exemplo 7.1-2

7.1.3 Instrução block:

No entanto, se desejarmos utilizar comandos para definir procedimentos muito mais elaborados que englobem sequências de instruções, devemos utilizar a instrução ***block***, cuja estrutura é:

```
Nome_do_comando(argumento1,argumento2,...,argumentok):=block (
[variável_local1, variável_local2,…, variável_localk ],

instrução1,

instrução2,

. . .

instrução n );
```

Instrução 5: Instrução *block*

Exemplos:

1) ***Programa que calcula a área e o perímetro de um círculo de raio r:***

```
(%i1) circulo(r):=block([área, perimetro],

area:%pi*r^2,

perimetro:2*%pi*r,

return([area,perimetro]));

(%o1) circulo(r):=block([area], area:%pi*r^2, perimetro:2*%pi*r,
return([area,perimetro]));


(%i2)circulo(3);

(%o2) [9%pi, 6%pi]
```

2) Programa que dada uma lista de notas calcula a média aritmética, a nota máxima e a nota mínima:

```
(%i3) notas(lista):= block([media,maximo,minimo,n],

n:length(lista),

media:sum(lista[i],i,1,n)/n,

maximo:lmax(lista),

minimo:lmin(lista),

print("Media das notas da turma=",media, "Nota maxima =:",maximo," Nota
minima =:",
minimo) )$


(%i4)lista:[10, 9, 15, 13, 11, 7, 19, 10, 15, 17, 12, 13, 13, 14, 12, 11, 8, 7, 19, 13];

(%o4) [10, 9, 15, 13, 11, 7, 19, 10, 15, 17, 12, 13, 13, 14, 12, 11, 8, 7, 19, 13]


(%i5) notas(lista);

(%o5) Media das notas da turma=12.4 Nota maxima=19 Nota minima=7
```

Exemplo 7.1-3

7.2 Quadro Resumo: Instruções de Programação

Instrução	Descrição
For	Executa ciclos sujeitos a contagem.
Do/While	Executa ciclos sujeitos a condições.
Block	Avaliar e executa instruções que se encontram em sequência. Retorna o valor da ultima expressão executada.[9]
Print	Imprime valores e/ou resultados.
Return	Usado para sair explicitamente de um bloco, trazendo o valor do argumento
/* */	Permite colocar comentários dentro de uma célula ***input.***

[9] . A instrução ***block*** pode aparecer dentro de outra instrução ***block.***

Capitulo 8.

Análise Numérica

"Cada problema que resolvi, tornou-se numa regra, que serviu depois para resolver outros problemas"- René Descartes

Esta secção apresenta diversas aplicações de cálculo numérico em *MÁXIMA*, nomeadamente:

- ✓ Série de Taylor;
- ✓ Interpolação Polinomial de Lagrange;
- ✓ Método dos Mínimos Quadrados por Regressão Linear
- ✓ Resolução Numérica de Equações Não Lineares – Método de Newton;
- ✓ Polinómios Ortogonais de Legendre;
- ✓ Série de Fourrier.
- ✓ Resolução Numérica de Equações Diferenciais Ordinárias – Método de Runge–Kutta.

8.1 Série de Taylor

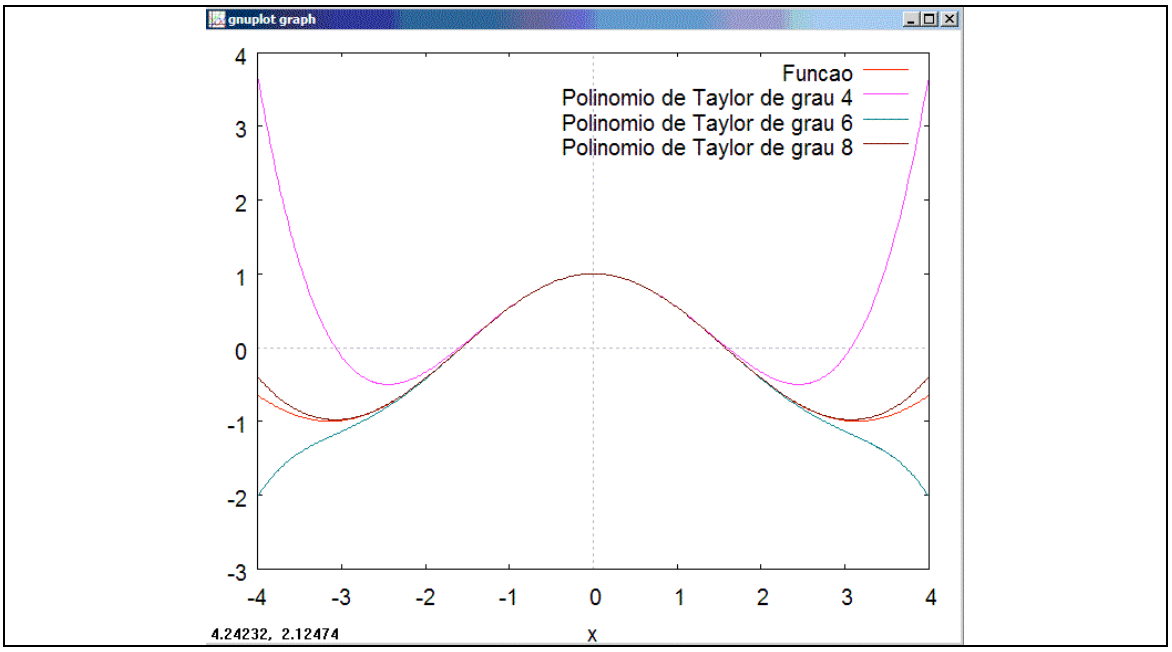
Seja f uma função com derivadas de ordem k, para k = 1(1)*n,* na vizinhança de x_0. A soma parcial de ordem n do seu desenvolvimento em série de Taylor com pólo em x_0 é dada por:

$$S_n(f,x_0)=\sum_{k=0}^{n}\frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$$

Exercício de Aplicação 1: Dada a função $f(x)=\cos(x)$, calcule o polinómio de Taylor $S_n(f,x_0)$, para n=4,6 e 8, em $x_0=0$, e represente graficamente a função e os polinómios numa vizinhança de x_0.

Resolução:

```
(%i1) PolT(f,x0,n,x):=taylor(f(x),x,x0,n)$

(%i2) g(x):=cos(x)$

(%i3) S4(x):=PolT(g,0,4,x)$

(%i4) S6(x):=PolT(g,0,6,x)$

(%i5) S8(x):=PolT(g,0,8,x)$

(%i6)plot2d([cos(x),S4(x),S6(x),S8(x)],[x,-4,4],[style,[lines,1,2],[lines,1,3],
[lines,1,4],[lines,1,5]], [legend, "Funcao", "Polinomio de Taylor de grau 4"," Polinomio de
Taylor de grau 6"," Polinomio de Taylor de grau 8"]);
```



Exercício de Aplicação 1: Série de Taylor

☑ Como podemos constatar, os polinómios de Taylor constituem uma boa aproximação da função na vizinhança do pólo, neste caso, a origem. E a aproximação é tanto melhor e ao longo de um maior intervalo, quanto maior for o valor de n.

8.2 Interpolação Polinomial

A interpolação consiste em determinar uma função que assume valores conhecidos em certos pontos, a que chamaremos nós de interpolação. A classe de funções escolhida para a interpolação é, á priori, arbitrária e deve ser adequada às características do problema. Neste caso, iremos considerar uma função polinomial ou polinómio.

Assim, dados n+1 pontos de abcissas distintas, $\{(x_k, y_k)\}_{k=0(1)n}$ o polinómio interpolador de Lagrange, $L_n(x)$, define-se:

$$L_n(x) := \sum_{j=0}^{n} y_j l_j^n(x),$$

$$l_j^n(x) = \prod_{i \neq j=0}^{n} \frac{(x - x_j)}{(x_i - x_j)}, j = 0(1)n$$

e verifica as condições de interpolação:

$$L_n(x_k) = y_k, k = 0(1)n.$$

Exercício de Aplicação 2: Considerando a lista de pontos {(-2, 4), (3, 2), (5, -1)}, calcule o polinómio interpolador de Lagrange que por eles passa. Represente graficamente os pontos e o polinómio.

Resolução:

(%i1) Pontos:[[-2,4], [3, 2],[5, 1]];

(%o1) *[[-2,4],[3,2],[5,1]]*

(%i2) load("interpol")$

O pacote *interpol* possui o comando *lagrange*, que calcula o polinómio interpolador.

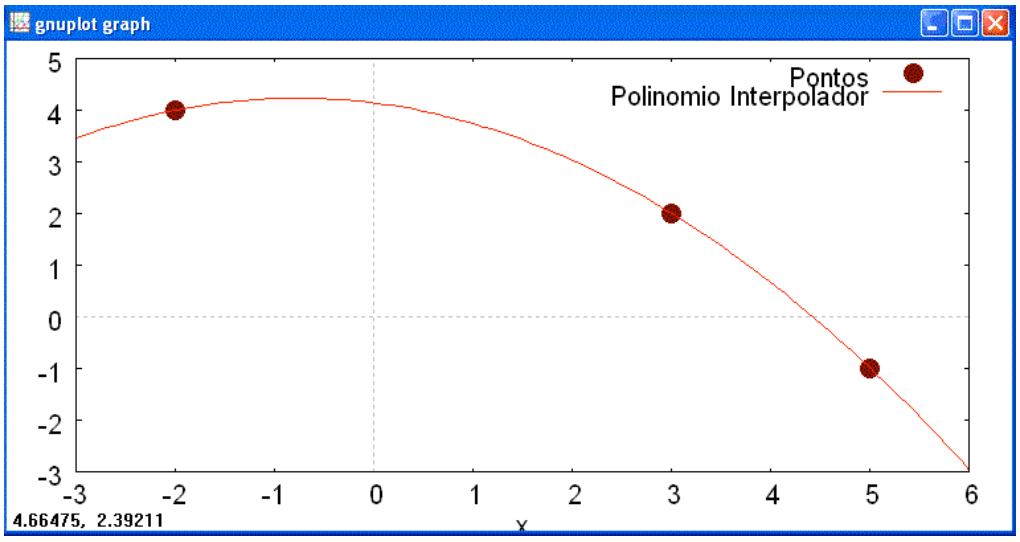
(%i3) lagrange(Pontos);

(%o3) $$-\frac{(x-3)(x+2)}{14}-\frac{(x-5)(x+2)}{5}-\frac{4(x-5)(x-3)}{35}$$

(%i4) p:ratsimp(%); *Simplificação do polinómio anterior.*

(%o4) $$-\frac{11x^2+17x-290}{70}$$

(%i5) plot2d([[discrete, Pontos],p],[x,-3,6],[style,[points,6,5],[lines,1,2]], [legend, "Pontos", "Polinomio Interpolador"]);



Exercício de Aplicação 2: Interpolação polinomial

8.3 Regressão linear

Frequentemente, em ciências experimentais, é preciso estabelecer uma relação entre:

- Uma variável dependente ou de resposta, ***y;***
- Uma, ou mais, variáveis independentes, ***x*** (ou $x_1,...,x_n$).

Essa relação pode ser traduzida por um modelo de regressão linear pelo método dos mínimos quadrados, que determina a recta:

$$y = ax + b,$$

- a= coeficiente linear
- b= coeficiente angular

Exercício de Aplicação 3: Pretende-se estudar a relação da produção de uma variedade de trigo, em Kg, e a quantidade, em Kg de adubo fertilizante utilizado:

Fertilizante (x_i)	16	24	32	40	48	56	64	72	80
Quantidade de trigo (y_i)	199	214	230	248	255	305	298	323	359

Resolução:

```
(%i1) dados:[[16,199], [24,214 ], [32,230], [40, 248], [48,255], [48,262], [48,279],
[48,267], [56, 305], [64,298], [72, 323], [80,359]]$

(%i2)  load("stats")$

(%i3) simple_linear_regression (dados);


        SIMPLE LINEAR REGRESSION
```

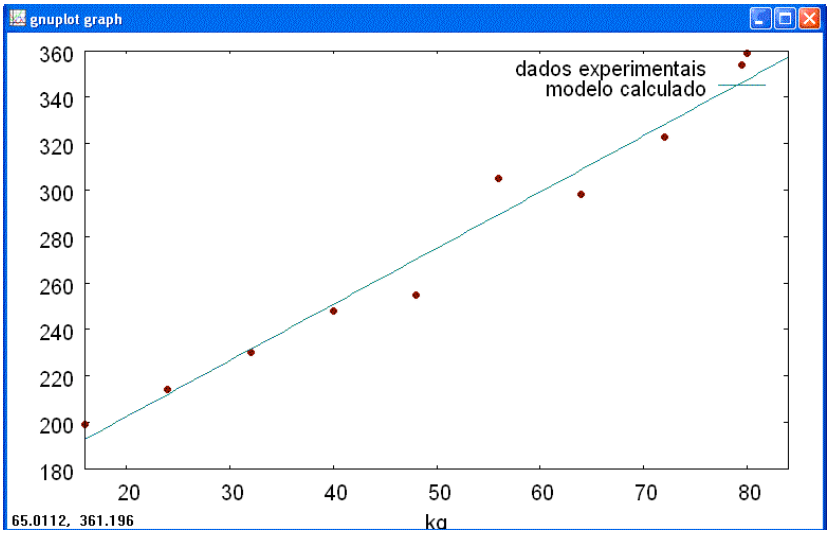
(%o3)

$$\begin{vmatrix} model = 2.41666666668x + 154.111111111 \\ correlation = 0.98266512147707 \\ v\_estimation = 114.0317660317459 \\ b\_conf\_int = [2.009183744425169, 2.824149588908167] \\ hypothese = H_0 : b = 0, H_1 : b \# 0 \\ distribution = [student\_t, 7] \\ statistic = 14.0239216561971 \\ pvalue = 6.484956191640386.10^{-7} \end{vmatrix}$$

Represenção gráfica do modelo calculado e dos dados:

(%i4) f(x):=2.41666666666666*x+154.111111111$

(%i5) plot2d([[discrete, dados],f(x)],[x,16,84],[style, [points,1,5],[lines,1,4]], [legend,"dados experimentais","modelo calculado"],[xlabel, "kg"]);



Exercício de Aplicação 3: Regressão Linear

8.4 Resolução Numérica de Equações Não Lineares – Método de Newton

O método de Newton permite resolver numericamente uma equação não linear:

$$f(x)=0,\ x \in [a,b].$$

Partindo de um valor inicial, ou de arranque, $x_0 \in [a,b]$, calcula uma sucessão de valores aproximados:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n \in N_0,$$

que converge, sob certas condições, para a solução exacta da equação.

Exercício de Aplicação 4: Calcule o valor aproximado dos zeros da seguinte função:

$$f(x) = \cos(x) - (x-2)^2 + 1,$$

com erro inferior a 10^{-5} e represente-a graficamente.

Resolução:

(%i1) f(x):=cos(x)-(x-2)^2 +1$

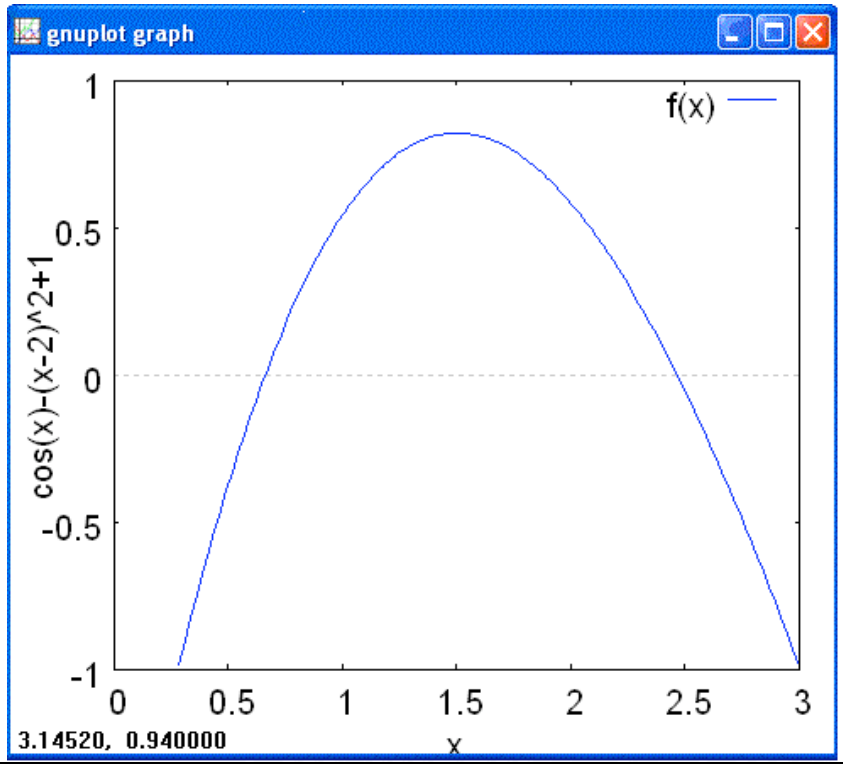
(%i2) load("newton1")$

(%i3) newton(f(x),x,0,10^-5); ← $x_0 = 0$
(%o4) ***0.66271812270536***

(%i5) newton(f(x),x,4,10^-5); ← $x_0 = 4$

(%o5) ***2.467609314656555***

(%i6) plot2d([f(x)],[x,0,3],[y,-1,1],[legend, "f(x)"]);



Exercício de Aplicação 4: Resolução de Equações não Lineares - Método de Newton.

8.5 Polinómios Ortogonais – Legrendre

Um sistema polinomial, denotado por $\{p_n(x)\}_{n\geq 0}$ é ortogonal no intervalo $[a,b]$, relativamente a uma função peso *w(x)* se e só se:

$$(p_n,p_m)=\int_a^b p_n(x)p_m(x)w(x)dx=0, n\neq m; n,m=0,1,...;$$
$$(p_n,p_n)\neq 0, n=0,1,....$$

Os polinómios de Legendre, implementados no *MÁXIMA,* são ortogonais no intervalo $[-1,1]$, relativamente ao peso unitário e satisfazem a normalização: $p_n(1)=1$ Além disso, satisfazem as seguintes condições iniciais e relação de recorrência*:*

$$p_0(x)=1, p_1(x)=x,$$
$$p_{n+1}(x)=\frac{2n+1}{n+1}xp_n(x)-\frac{n}{n+1}xp_{n-1}(x), n=1,2,...$$

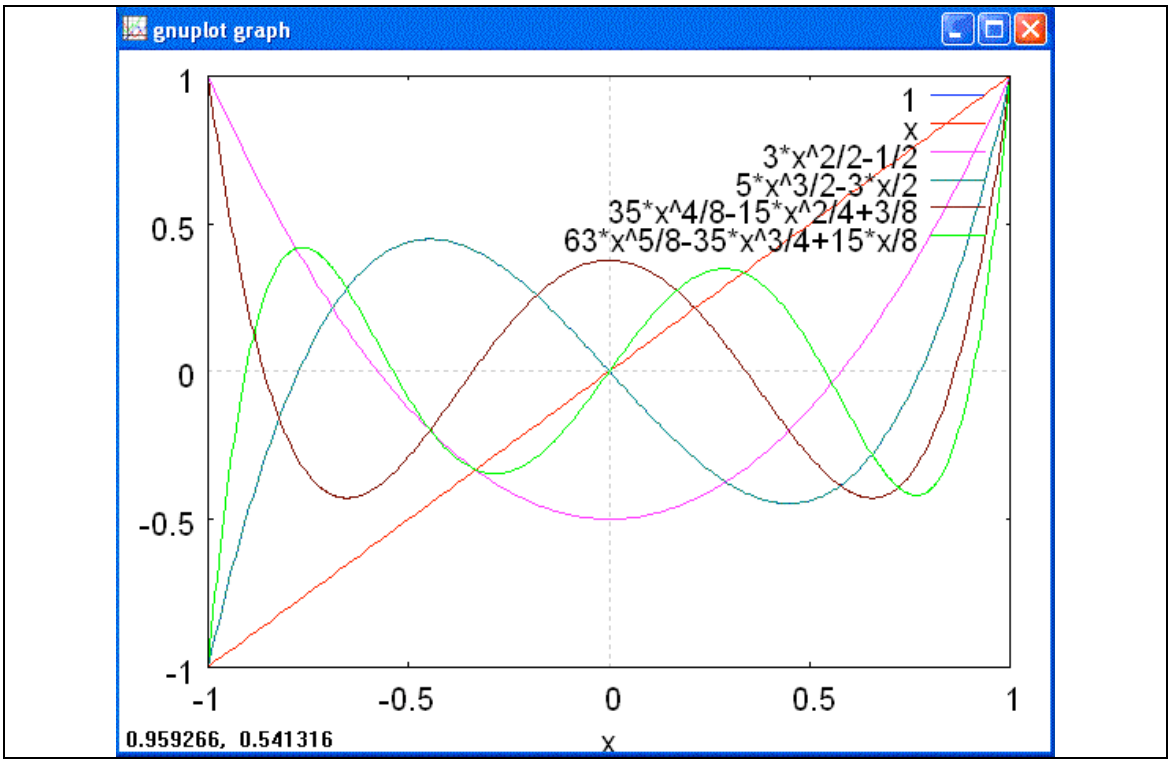
Exercício de Aplicação 4: Calcule os polinómios ortogonais de Legendre de graus *de 0 a 5*, simplifique-os e represente-os graficamente.

Resolução:

```
(%i1) load("orthopoly")$

(%i2) L05:makelist(expand(legendre_p(n,x)),n,0,5);
```

(%o2) $\left[1, x, \frac{3x^2-1}{2}, \frac{5x^3-3x}{2}, \frac{35x^4+3}{8}-\frac{15x^2}{4}, \frac{63x^5+15x}{8}-\frac{35x^3}{4}\right]$

```
(%i5)plot2d(L05,[x,-1,1]);
```



Exercício de Aplicação 5: Polinómios Ortogonais de Legendre

☑ Pela observação gráfica podemos constatar, entre outras características, a propriedade de simetria destes polinómios:

$$p_{2n}(-x) = p_{2n}(x), p_{2n+1}(-x) = -p_{2n+1}(x), n \geq 0.$$

8.6 Série de Fourier

Seja *f* uma função periódica, com período *2L,* e contínua no intervalo *[-L, L].* Então a ***Série de Fourier*** de *f* é a série de funções:

$$a_0 + \sum_{n=1}^{\infty}\left[a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right]$$

Os coeficientes a_n e b_n designam-se por ***coeficientes de Fourier*** e definem-se do seguinte modo:

$$\begin{cases} a_0 = \dfrac{1}{2L}\displaystyle\int_{-L}^{L} f(x)dx \\ a_n = \dfrac{1}{L}\displaystyle\int_{-L}^{L} f(x)\cos\left(\frac{n\pi x}{L}\right)dx, n = 1,2,.... \\ b_n = \dfrac{1}{L}\displaystyle\int_{-L}^{L} f(x)\sin\left(\frac{n\pi x}{L}\right)dx, n = 1,2,... \end{cases}$$

Exercício de Aplicação 7: Calcule a série de Fourier da função:
$f(x) = |x| - 2, x \in [-1,1]$.

Resolução:

(%i1) load ("fourie")$ ← Disponibiliza comandos para computação simbólica de séries de Fourier.

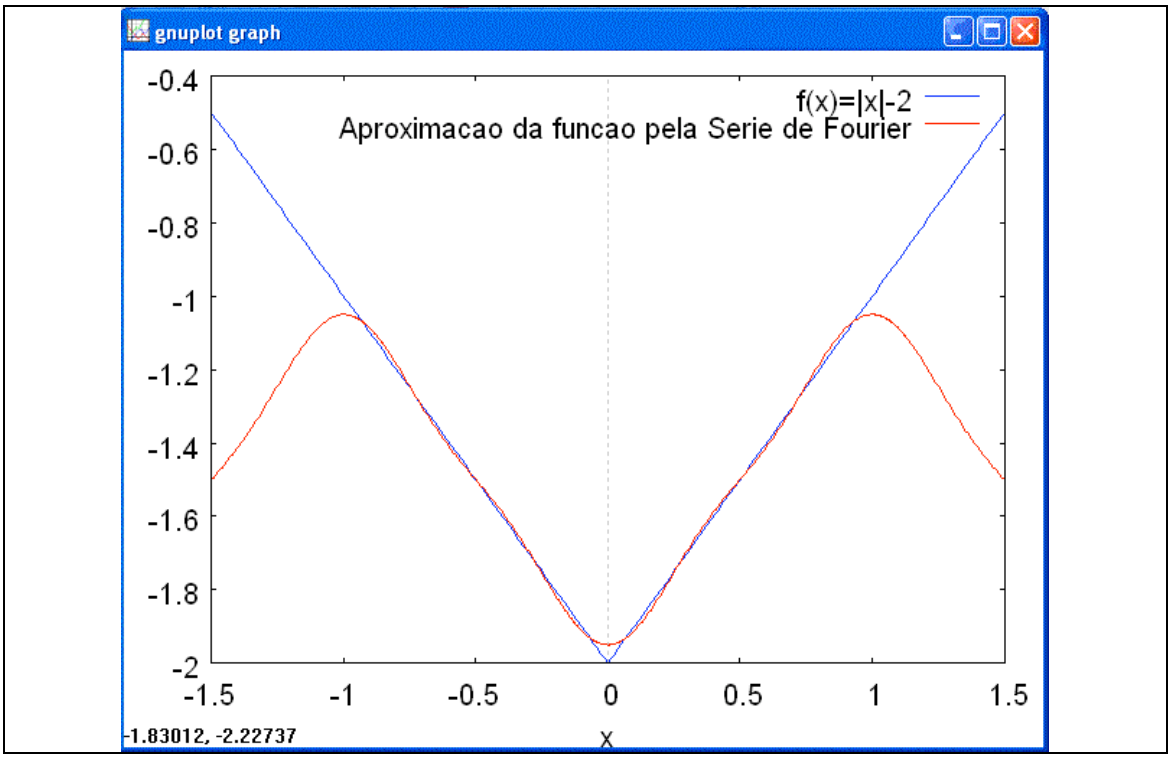
(%i2) f(x):=abs(x)-2$

(%i3) g:fourexpand(fourier (f(x), x, 1),x,1,3)$

Is sin(%pi*n) positive, negative, or zero? positive;

$$\left[a_0 = -\frac{3}{2},\ a_n = 2\left(-\frac{\sin(\%pin)}{\%pin} + \frac{\cos(\%pin)}{\%pi^2 n^2} - \frac{1}{\%pi^2 n^2} \right),\ b_n = 0 \right]$$

(%i4) plot2d([f(x),g],[x,-1.5,1.5]);



Exercício de Aplicação 6: Série de Fourier

☑ Como podemos constatar, a série de Fourier constitui uma boa aproximação da função no intervalo dado [-1, 1].

8.7 Equações Diferenciais Ordinárias: Método de Runge-Kutta

Considere o seguinte ***problema de valor inicial*** **(PVI)** constituído por uma equação diferencial ordinária e respectiva condição inicial:

$$\begin{cases} y'(x) = f(x, y(x)) \\ y(a) = y_0 \end{cases}, x \in [a,b].$$

O método de Runge–Kutta calcula uma aproximação numérica da solução do PVI, $\{(x_k, y_k)\}_{k=0(1)n}$, onde y_k é um valor aproximado de $y(x_k)$, $y_k \approx y(x_k)$, e $x_k = a + kh,\ k = 0(1)n$, são abcissas igualmente espaçadas no intervalo $[a,b]$.
A integração numérica de PVIs pelo o método de Runge-Kutta define-se:

$$\begin{cases} K_1 = f(x_k, y_k) \\ K_2 = f(x_k + \frac{h}{2}, y_k + \frac{hK_1}{2}) \\ K_3 = f(x_k + \frac{h}{2}, y_k + \frac{hK_2}{2}) \\ K_4 = f(x_k + h, y_k + K_3) \\ \\ y_{k+1} = y_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), k = 1(1)n-1, \end{cases}$$

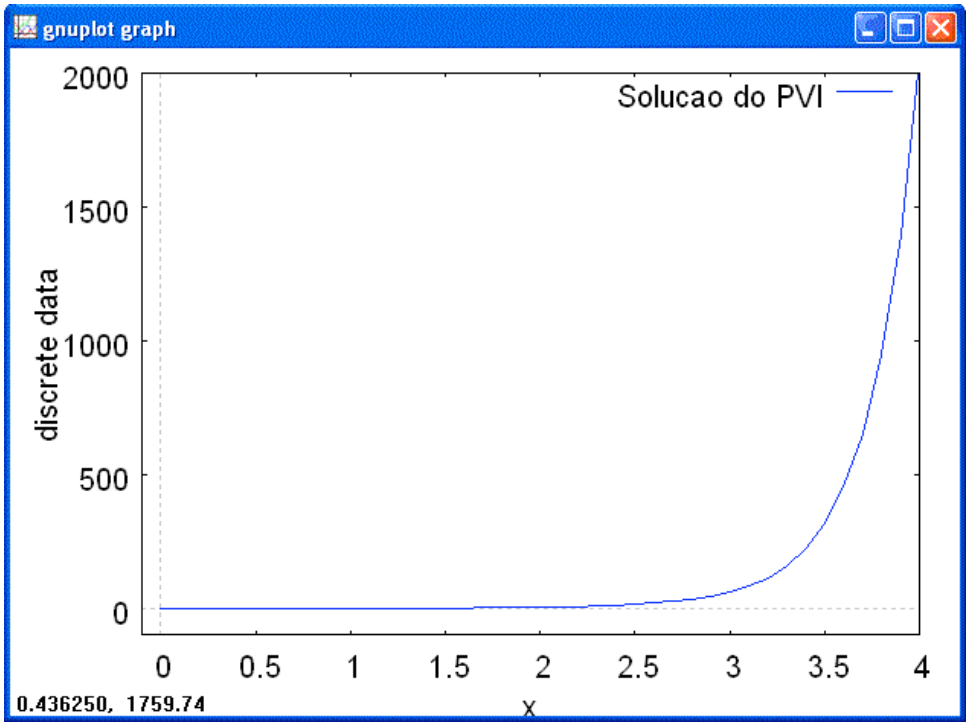
sendo (x_0, y_0) o ponto de arranque.

Exercício de Aplicação 7: Aplique o método de Runge-Kutta ao seguinte PVI:

$$\begin{cases} \frac{dy}{dx} = 0.2e^x + xy \\ y(0) = 0 \end{cases}, x \in [0,2].$$

Resolução:

```
(%i1) load ("dynamics")$

(%i2) solpvi: rk([0.2*exp(x)+x*y],[y],[0],[x,0,2, 0.1])$   ← h =0.1, n=20;

(%i3) plot2d([discrete, solpvi], [style, lines],[x,-.1,4],[y,-100,2000], [legend,
"Solucao do PVI"]);
```



Exercício de Aplicação 7: Método de Runge-Kutta

8.8 Quadro Resumo: Comandos de Análise Numérica

Instrução	Descrição
taylor (f(x),x,x0,n)	Expande a função f(x) numa série de Taylor truncada, contendo os termos até á ordem n, na variável x, em torno do pólo x0.
fourier (*f, x, p*)	Retorna uma lista de coeficientes de Fourier de `f(x)` definidos no intervalo `[-p, p]`.
fourexpand (*l*, *x*, *p*, *limit*)	Retorna a série de Fourier partindo da lista *l* dos coeficientes de Fourier até ao termo *limit* (*limit* pode ser `inf`). *x* e *p* possuem o mesmo significado que em ***`fourier`***.
lagrange (lista de pontos)	Calcula o polinómio interpolador de Lagrange nos pontos da lista dadd.
newton(g(x),x,x0,erro)	Retorna uma solução aproximada da equação g(x)=0, usando o método de Newton.
legendre_p(n,x)	Calcula o polinómio de Legendre de grau n, na variável x, definido no intervalo [-1,1] e que em 1 vale 1.
rk([eq1,eq2,…,eqn],[variavel1, variável 2,…, variaveln],[dominio])	Implementa o método de Runge-Kutta de 4ªordem para a resolução de equações diferenciais.
simple_linear_regression (x);	Calcula a regressão linear simples de uma matriz de duas colunas ou de uma lista de pares. A saída desta instrução inclui diversos parâmetros, além da equação ajustada do modelo de regressão linear simples, nomeadamente: • Correlação; • Intervalo de Confiança (por padrão é 0,95); • P_value; • Correlação de diferentes tipos de dados; • Estimador de variância residual.

8.9 Pacotes ("Packages") utilizados em Análise Numérica

Pacote	Descrição
stats	Disponibiliza um conjunto de procedimentos de inferência clássica estatística e procedimentos de teste.

fourie	Disponibiliza um conjunto de funções para computação simbólica de séries de Fourier.
interpol	Disponibiliza os métodos lagrangianos, lineares e os de splines cúbicos para interpolação polinomial.
newton1	Permite aplicar o método de Newton para a resolução de equações.
orthopoly	Avalia e calcula vários tipos de polinómios ortogonais.
dynamics	Inclui comandos para explorar sistemas dinâmicos e a implementação do método de Runge-Kutta de 4ª ordem.

Bibliografia:

[1] Marinez, Wendy L., *Computational Statististics Handbook with MATLAB*, Chapman & Hall/CRC, 2002.

[2] Figueiredo, Fernanda; Figueiredo, Adelaide; Ramos, Alexandra; Teles, Paulo; *Estatística Descritiva e Probabilidades – Problemas resolvidos e propostos com aplicações em **R***; Escolar Editora; 2007.

[3] Villate, Jaime E.; *Introdução aos Sistemas dinâmicos – Uma Abordagem prática com Maxima*; 2006; http://fisica.fe.up.pt/pub/MAXIMA/sistdinam.pdf

[4] Site do sistema operativo ***MAXIMA***: http://MAXIMA.sourceforge.net/

[5] da Rocha, Zélia; *Unidades computacionais da disciplina de Análise Numérica I*; Departamento de Matemática da Faculdade de Ciências da Universidade do Porto; 2008/2009.

[6] Chaves, Gabriela; *Cálculo Infinitesimal*; Departamento de Matemática Pura da Faculdade de Ciências da Universidade do Porto; 2000.

Índice de ilustrações:

Contacto:

Se tem dúvidas sobre o *MAXIMA* ou questões relacionadas com este livro introdutório, por favor, visite o blog: sobreviveraomaxima.blogspot.com, ou envie um e-mail para blogmaxima@gmail.com.