

Um Curso de Maple

UESC

Universidade Estadual de Santa Cruz
Departamento de Ciências Exatas e Tecnológicas

André Nagamine

andren@uesc.br

março de 2001

SUMÁRIO

I.	Introdução	3
1.	Sintaxe e Comandos Básicos	5
1.1	Sintaxe	5
1.2.	Operações Básicas	7
1.3.	Representação Decimal	8
1.4.	Aritmética	9
1.5.	Operações com número complexos	11
1.6.	Atribuindo letras e nomes	12
1.7.	Operações Simbólicas	13
1.8.	Somatórios	14
2.	Sequência, Listas, Conjuntos e “Array”	15
2.1.	Sequência	15
2.2.	Lista	16
2.3.	Conjunto	18
2.4.	Operações com Conjuntos e Listas	18
2.5.	Array	19
3.	Encontrando Soluções (sistemas e expressões), Funções e Polinômios	22
3.1.	Resolução de Equações e Sistemas	22
3.2.	Funções de uma variável	24
3.2.1	Funções reconhecidas (usuais)	26
3.2.2	Funções definidas por partes	28
3.3.	Operações com Polinômios	30
4.	Gráficos de Funções 2D e 3D	33
1.1.	Gráficos em duas dimensões	33
1.1.1	Gráficos de funções	33
1.1.2	Gráficos de curvas no plano	35
1.1.3	Gráficos em Coordenadas Polares	36
1.1.4	Múltiplos Gráficos em 2D	36
1.1.5	Animação de Gráficos em Duas Dimensões	38
1.2.	Gráficos em três dimensões	39
1.2.1	Gráficos de funções	39
1.2.2	Gráficos de superfícies no espaço	40
1.2.3	Múltiplos Gráficos em 3D	41
2.	Tópicos de Cálculo	42
2.1.	Limites	42
2.1.1	Limites de funções de uma variável	42

2.1.2	Limites laterais	43
2.1.3	Limites de funções de mais de uma variável	43
2.2.	Derivadas	44
2.2.1	Derivadas de funções de uma variável	44
2.2.2	Derivadas de ordem superior	45
2.2.3	Derivadas de funções de mais de uma variável	46
2.3.	Integrais	47
2.3.1	Integrais Múltiplas	48
3.	Tópicos de Programação	50
3.1.	Fundamentos	50
3.2.	Estruturas de Programação	51
3.2.1	O Comando “for”	52
3.2.2	O Comando “if”	53
3.2.3	O Comando “while”	54
3.3.	Controle de Parâmetros	55
3.4.	Outros Exemplos	55
3.4.1	Procedimentos Recursivos	55
3.4.2	Procedimento com Listas	57
3.4.3	O Método de Iteração de Newton	58
7.	Bibliografia	59

I. Introdução

O presente texto foi elaborado no sentido de nortear o andamento do curso intitulado “*Curso de Introdução ao Software Maple*”, ministrado no período de 18/04/2001 a 07/05/2001, o qual é parte integrante das atividades desenvolvidas pelo Grupo de Pesquisa em Ensino e Aprendizagem da Matemática em Ambiente Computacional – GPEMAC que vem atuando na área de matemática de DCET desde janeiro de 2000.

O Maple é um sistema de computação algébrica desenvolvido por Waterloo Maple Inc. (Ontário, Canadá). A expressão “sistema de computação algébrica”, deve-se ao fato de que o Maple permite aos seus usuários fazer cálculos não somente com números, mas também com símbolos, fórmulas, expressões, equações e assim por diante. Pode-se usar essa capacidade simbólica para obter soluções analíticas exatas para muitos problemas matemáticos, por exemplo, diferenciação, integração, sistemas de equações, expansão de funções em séries, problemas em Álgebra Linear, etc. Sistemas de computação algébrica, em particular o Maple, são poderosas ferramentas para matemáticos, físicos,

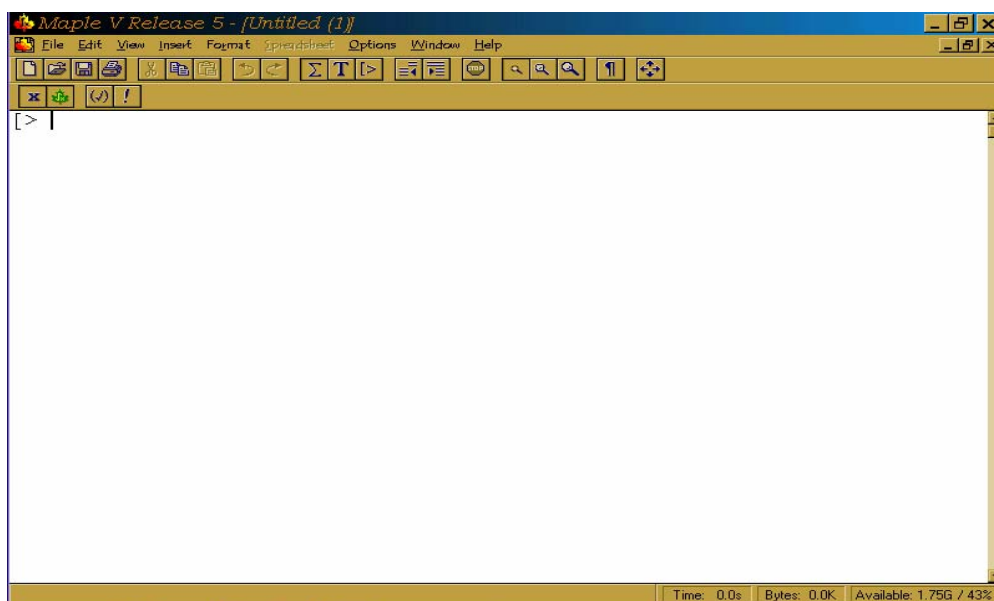
químicos, engenheiros, enfim para todos aqueles que necessitam de respostas rápidas e precisas para determinados problemas matemáticos.

O curso tem como objetivo fornecer ao participante um apanhado geral das potencialidades de Maple em relação a diversos tópicos da matemática. Não se tem, no desenvolvimento das atividades, a pretensão de detalhar e aprofundar os temas propostos, mesmo porque isso seria improvável dada a carga horária destinada ao curso e também dada a extensão do software. Pretende-se, sim, fornecer ao participante os caminhos possíveis para que ele próprio possa escolher no tratamento e resolução de seus problemas matemáticos. O curso é dividido em dois módulos, no primeiro serão abordados os seguintes tópicos: sintaxe e comandos básicos, sequência, listas, conjuntos, arrays, sistemas e expressões, funções e polinômios. No segundo módulo serão abordados os seguintes tópicos: gráficos 2d e 3d, tópicos de cálculo e tópicos de programação.

1. Sintaxe e Comandos Básicos

1.1 Sintaxe


Ao abrir o Maple nos é apresentada uma folha de trabalho(Worksheet ou prompt), no qual podemos acionar funções do aplicativo, produzir textos, hipertextos, cálculos, obter gráficos e animações. A interface gráfica do Maple não oferece dificuldades para os usuários. Na parte de cima da folha vemos três barras de *menu*, e na parte de baixo uma barra contendo informações sobre o estado atual do sistema.



Os procedimentos de abrir um novo documento do Maple, abrir um documento já existente do Maple, salvar um documento, imprimir, recortar, copiar e colar, são análogos aos do *Word* e podem ser acionados através dos ícones



ou ainda através dos menus *File* e *Edit*, selecionando os comandos *new*, *open*, *save*, *print*, *cut*, *copy* e *paste*, respectivamente. Cada um desses comandos também possuem uma combinação de teclas de atalho que também podem ser utilizadas.

Nesta primeira parte, através de exemplos, discutiremos algumas regras de sintaxe que são absolutamente indispensáveis na elaboração de um cálculo matemático com o Maple. Toda instrução de cunho matemático no Maple deve inicia-se com um *prompt* assinalado pelo símbolo $>$ logo após o `[`, em frente ao qual podemos escrever um comando a ser executado, e necessariamente deve termina com o sinal `(;)` ou `(:)`. O `:` deve ser usado apenas quando desejamos que um determinado comando seja apenas guardado na memória do Maple e não exibido em tela. Uma região tipo *input* pode ser modificada para uma região tipo texto, ou vice-versa, mediante o uso da tecla F5, ou usando o ícone  do *menu*.

Vejam os o que acontece quando esquecemos o sinal de ponto e vírgula (;) ao final de qualquer comando matemático.

```
>2*3
```

```
>
```

```
Warning, incomplete statement or missing semicolon
```

O sistema reclama em azul dizendo que a instrução **2*3** está incompleta ou que falta um ponto e vírgula. Neste caso devemos voltar e corrigir o problema.

```
>2*3;
```

6

De forma semelhante às calculadoras científicas existe uma ordem de precedência ao efetuar operações algébricas básicas. A multiplicação e divisão são efetuadas antes da adição e subtração e potências são efetuadas antes da multiplicação, obedecendo assim a ordem de precedência na avaliação de expressões. Cabe lembrar que como qualquer outro software matemático existem símbolos determinados para efetuar as operações algébricas básicas de acordo com o teclado do computador. O esquema abaixo ilustra o que foi dito.

ordem de precedência				
!	^ou**	/	*	+
-				
fatorial	potenciação	divisão	multiplicação	adição
subtração				

Por exemplo, se colocarmos:

```
> 2+2-8*3/2;
```

teremos como resposta **-8**, pois o Maple efetua primeiro a divisão **3/2** multiplica o resultado por **8** efetua a soma **2+2** e por último efetua a subtração **4-12**.

Para evitar confusões podemos utilizar parênteses () que é necessário para modificar a ordem de precedência. Porém colchetes [] e chaves { } não devem ser utilizados para este fim.

Exemplo:

```
> (2+2-8)*3/2;
```

-6

```
> (2+2-8*3)/2;
```

-10

```
> {2+2-8*3}/2;
```

```
Error, use intersect for sets
```

O Maple trabalha com um sistema multi-linha. Vários comandos podem ser colocados em uma mesma linha, ou um comando em várias linhas. Exemplos:

```
>1+2; 2+3; 3+4;
```

3

5

7

Obs.: Um comando pode utilizar várias linhas, desde que agrupadas em um único colchete.

Se terminarmos uma instrução com **:** (dois pontos), o Maple a executa e a guarda na memória, mas não mostra o resultado no tela. Por exemplo:

>**1+2:**

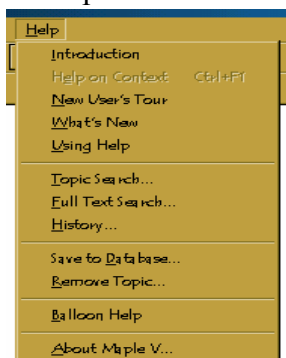
>**%+3; #somando 3 ao resultado anterior**

6

Para acionar o sistema de ajuda, *help on-line*, basta digitar no *prompt* o sinal **?** seguido da expressão da qual se deseja informação. Por exemplo, digitando

>**?limit**

e acionando a tecla **ENTER**, é apresentada na tela um folha com informações sobre o comando **limit**, sua sintaxe, exemplos e comandos correlatos. O sistema de ajuda pode ser acessado também através do *menu*. No item *help* do *menu* básico se encontra várias maneiras de buscar ajuda no Maple.



Ao encontrar uma falha em um comando o Maple responde com uma mensagem de erro. Na mensagem vem indicado o tipo de falha, e o cursor localiza a primeira falha. Os principais erros são: sintaxe inadequada, como equívoco na digitação, nome incorreto de comando, uso de palavras reservadas, ou ainda erro de domínio de funções matemáticas, cálculos que excedem a capacidade de memória ou de computação do sistema ou do aplicativo, etc. Exemplos:

>**7/0;**

Error, division by zero

>**6*-1;**

Syntax error, '-' unexpected

>**3x;**

Syntax error, missing operator or ';'`

>**tan(Pi/2);**

Error, (in tan) singularity encountered

>**1234567890^9876543210;** “você sabe quantos dígitos tem esse número?”

Error, integer too large in context

1.2. Operações Básicas

As operações aritméticas básicas são feitas com os seguintes símbolos:

+ (adição) **-** (subtração) ***** (multiplicação) **/** (divisão) **^** (potenciação).

>**5*3+8;**

23

```
>3^2*3;  
27
```

```
>(8*2+4)/(10*3-25);  
4
```

Obs.: Os resultados acima estão de acordo com a ordem de precedência discutida anteriormente.

1.3. Representação Decimal

Quando se realiza operações numéricas, a menos que os tipos numéricos sejam *float* (números em representação decimal) o Maple retorna sempre o resultado exato na forma simbólica, por exemplo:

```
> 1/2 + 5; 30/9; Pi; exp(2); ln(2);  
  
11/2  
10/3  
 $\pi$   
 $e^2$   
ln(2)
```

Em alguns casos para se obter o número na forma decimal basta acrescentar um ponto após o número. Isso faz com que o Maple leia o número em aritmética de ponto flutuante.

```
> 1/2 + 5.; 30./9; ln(2.);  
5.500000000  
3.333333333  
.6931471806
```

Obs.: o último número acima é o mesmo que 0,6931471806. O Maple usa ponto no lugar da vírgula e não mostra o número que está a esquerda ou a direita do ponto caso este seja apenas o zero. Nos exemplos acima 5. é o mesmo que 5,0, 30. é o mesmo que 30,0 e 2. é o mesmo que 2,0.

Uma forma mais geral é usar o comando **evalf** (avaliar com ponto flutuante) para se obter uma representação decimal de um número. Normalmente, o sistema utiliza dez algarismos significativos.

```
>evalf(176/47);  
3.744680851
```

Também podemos modificar o número de dígitos para quanto quisermos. Por exemplo:

```
>evalf(22/7, 50); #modificando localmente o número de  
dígitos para 50;  
3.1428571428571428571428571428571428571428571  
  
>evalf(Pi);
```


3.141592654

Obs.: O símbolo **#** utilizado acima significa comentário. Ele é utilizado quando se quer fazer algum comentário a respeito do comando. Tudo o que vem depois desse símbolo é ignorado pelo Maple.

O número de dígitos utilizado na representação *float* é, por *default*, 10. Este valor está alocado na variável global **Digits**. Podemos modificar o valor dessa variável. Por exemplo:

```
>Digits := 3
```

Digits:=3

```
>evalf(Pi);
```

 calculando o valor de π com 3 dígitos (incluindo o dígito inteiro)

3.14

Para restabelecer o valor original da variável global **Digits** implementamos o comando,

```
>Digits:=10;
```

Digits:=10

```
>1/3;
```

 nos dá o resultado em forma de fração

1/3

```
>evalf(%);
```

.3333333333

O símbolo **%** junto com o comando **evalf** representa o último valor calculado na forma de número decimal.

1.4. Aritmética

O comando **sqrt** (abreviação de *square root*) calcula a raiz quadrada.

```
> sqrt(25);
```

5

Se quisermos no Maple calcular o fatorial de um número, basta colocarmos depois do número o símbolo **!**. Por exemplo:

```
> 20!;
```

2432902008176640000

Agora um outro comando pode contar quantos dígitos tem o resultado acima, estamos falando do comando **length**. Como por exemplo:

```
>length(%);
```

19

O Maple tem muitos comandos que podem trabalhar com números *inteiros*, citaremos alguns importantes.

Se quisermos obter a decomposição de um determinado número inteiro em fatores primos usamos o comando **ifactor**. Por exemplo:

```
> ifactor(60);  
 $(2)^2 (3) (5)$ 
```

De outro modo o Maple pode determinar o máximo divisor comum entre dois ou mais números, para isso usamos o comando **igcd**. Por exemplo:

```
> igcd(123, 45);  
3
```

O comando **iquo** calcula o quociente da divisão entre dois números inteiros, como por exemplo:

```
> iquo(25, 3);  
8
```

Também no Maple pode-se calcular o resto da divisão entre dois números inteiros, basta usarmos o comando **irem**.

```
> irem(25, 3);  
1
```

Na matemática, mas especificamente na álgebra vemos que para achar o resto da divisão entre dois números podemos aplicar o conceito de *mod*. No Maple tal conceito continua existindo, podendo assim também acharmos o resto da divisão entre dois números de outra maneira, usando o comando **mod**. Por exemplo:

```
> 25 mod 3;  
1
```

Obs.: O comando **mod** no exemplo acima também pode ser interpretado da seguinte forma: $\overline{25}$ em \mathbb{Z}_3 é igual a $\overline{1}$ (faça as contas e tente outros exemplos).

O comando **isprime** possibilita sabermos se um determinado número inteiro é primo ou não. Por exemplo:

```
> isprime(3345674);  
false  
  
> isprime(18002676583);  
true
```

Para acharmos no Maple o valor absoluto de um número, usamos o comando **abs**.

```
> abs(-5);
```

5

No Maple podemos provocar um retorno em um tipo numérico desejado, para isso basta usarmos a função **convert**.

```
>convert(sqrt(28),float);
```

5.291502622

```
>convert(8.66666666, fraction); convert(5/7,float);
```

26/3

.7142857143

```
>convert(247,binary);
```

 nos dá o número 247 em representação binária

11110111

Os comando **cos**, **sin**, **tan** nos dá o valor da função cosseno, seno e tangente respectivamente em um determinado ponto. Por exemplo;

```
>cos(Pi/3);
```

 nos dar o valor exato da função cosseno no ponto Pi/3

$\frac{1}{2}$

```
>sin(Pi/6);
```

 nos dar o valor exato da função seno no ponto Pi/6

$\frac{1}{2}$

```
>tan(Pi/4);
```

 nos dar o valor exato da função tangente no ponto Pi/4

1

No Maple para escrevermos a função exponencial e logarítmica . Devemos usar respectivamente o comando **exp** e **ln**.

```
>exp(1);
```

e

```
>ln(1);
```

0

```
>ln(exp(5));
```

5

1.5. Operações com número complexos

No Maple podemos trabalhar também com números complexos. O símbolo ***I*** representa matematicamente para o Maple o número $\sqrt{-1}$, podendo assim usarmos os conceitos de operações de números complexos no Maple. Por exemplo:

> **(2+5*I) + (1-I) ;**

3+4I

> **(1+I) / (3-2*I) ;**

(1/13) + (5/13)I

> **4*I + (5+3*I) ;**

5+7I

> **(4-3*I) - (8-9*I) ;**

-4 + 6 I

O Maple possibilita sabermos qual a parte imaginária e real de um determinado número complexo, para isso usamos respectivamente os comandos ***Im*** e ***Re***. Por exemplo:

> **2+7*I ;**

2+7I

> ***Im*** ; nos dá a parte imaginária do número complexo acima

7

> **3-6*I ;**

3-6I

> ***Re*** ; nos dá a parte real do número complexo acima

3

1.6. Atribuindo letras e nomes

Nos trabalhos mais complexos é importante podermos representar por letras ou nomes expressões complicadas. No Maple esta representação é feita através do símbolo (***:=***). A regra é simples: ***A := B*** significa que o lado direito (B) é a definição do lado esquerdo (A).

> **num := 4568/235 ;**

4568/235

> **var := x ;**

var:=x

> **term := x*y ;**

term:=xy

> **A1 := x*sqrt(4) ;**

A1:=2x

Uma vez que você atribuiu um nome ou uma letra a uma certa expressão você pode usá-lo como se fosse a própria expressão.

```
>A1^2;
4x^2
>2*var+var^2+var^3;
2x+x^2+x^3
>2*num+num;
13704/235
```

Atribuir letras ou nomes a expressões é algo muito útil quando você precisa utilizar uma determinada expressão diversas vezes.

1.7. Operações Simbólicas

Uma das vantagens do Maple na computação algébrica é a capacidade de se fazer cálculos simbólicos, ou seja, a manipulação de expressões algébricas. Veremos a seguir os comandos **expand** (expandir), **factor** (fatorar), **simplify** (simplificar) os quais são os mais utilizados para se trabalhar com expressões algébricas. Primeiramente podemos ver que o Maple trabalha muito bem com qualquer tipo de expressões algébricas.

```
> (1+x)^2;
(1+x)^2
> (1+x) + (1+x) - x - x^2 + 3*x^2;
2+x+2x^2
```

Note que, expressões envolvendo apenas somas ou subtrações, são automaticamente simplificadas.

```
>A2:= (x-y)*(x^2+x*y+y^2);
A2:= (x-y)(x^2+xy+y^2)
```

A2 acima é uma expressão qualquer definida no Maple através de um produto. Se desejarmos saber o resultado usamos,

```
>expand(A2);
-y^3+x^3
```

O comando **expand** tem primariamente a função de efetuar o produto em uma soma, como foi visto no caso acima.

```
>A3:=expand(A2*(y+1));
A3:=x^3y+x^3-y^4-y^3
```

Expand também pode ser usado para se expandir outras expressões,

```
>expand(cos(a+b));
cos(a)cos(b)-sin(a)sin(b)
>expand(exp(a+ln(b)));
e^ab
```

O comando **simplify** é usado para se aplicar regras de simplificação às expressões.

```
>A4:=A3/(y+1);
```

$$A4 := \frac{x^3 y + x^3 - y^4 - y^3}{y + 1}$$

> **simplify**(A4) ;

$$-y^3 + x^3$$

> **A5:=cos(x)^5+sin(x)^4+2*cos(x)^2-2*sin(x)^2-cos(2*x) ;**

$$A5 := \cos(x)^5 + \sin(x)^4 + 2\cos(x)^2 - 2\sin(x)^2 - \cos(2x)$$

> **simplify**(A5) ;

$$\cos(x)^5 + \cos(x)^4$$

O comando **factor** é usado para fatorar expressões algébricas.

> **A6:= x^2+2*x+1 ;**

$$A6 := x^2 + 2x + 1$$

> **factor**(A6) ;

$$(x + 1)^2$$

> **A7:= x^5-x^4-7*x^3+x^2+6*x ;**

$$A7 := x^5 - x^4 - 7x^3 + x^2 + 6x$$

> **factor**(A7) ;

$$x(x - 1)(x - 3)(x + 2)(x + 1)$$

Em um quociente o Maple o comando **factor** automaticamente simplifica os fatores comuns no numerador e denominador.

> **factor((x^2-1)/(x+1)) ;**

$$(x - 1)$$

1.8. Somatórios

No Maple podemos usar o conceito de somatório, para isso usamos o comando **Sum** ou **sum**. Os somatórios podem ser colocados de forma genérica, ou seja, a variação do índice não necessariamente precisa ser numérica. Vejamos alguns exemplos abaixo:

> **Sum(i, i=1..5) ;**

$$\sum_{i=1}^5 i$$

O comando **Sum** quando escrito com letra maiúscula, apenas representa a soma no símbolo de somatório, agora se este estiver escrito com a letra minúscula ele nos dará o valor da soma, por exemplo.

> **sum(i, i=1..5) ;**

$$15$$

Veremos outro exemplo:

> **Sum(1/i^2, i=1..5) ;**

$$\sum_{i=1}^5 \frac{1}{i^2}$$

Se usarmos o comando **value**, este também nos dar o valor da soma como acontece com o comando **sum**. Por exemplo:

>**value (%) ;**

137/60

Vejamos agora um exemplo onde a variação do índice não é numérica.

>**sum(i, i=1..n) ;**

$$\frac{1}{2}(n+1)^2 - \frac{1}{2}n - \frac{1}{2}$$

Note que esta última fórmula é bem conhecida, pois é a soma de n primeiros números inteiros.

Obs.: tudo o que foi dito acima para o somatório também é válido para o produtório, cujo comando no Maple é **product**.

2. Sequência, Listas, Conjuntos e “Array”

2.1. Sequência

No Maple uma *seqüência* é uma estrutura de dados básica. Esta é um simples grupo de expressões do Maple separadas por vírgulas.

>**s:= 1, 2, 3, 4 ;**

s:= 1, 2, 3, 4

>**x, y, z, w ;**

x, y, z, w

O ponto “.” é usado para concatenar sequências. Por exemplo:

>**b:= 1, 3, 5, 7 ;**

b:= 1, 3, 5, 7

>**a.b ;**

a1,a3,a5,a7

Pode-se também fazer múltiplas atribuições utilizando a notação de sequência. Por exemplo

>**f, g, h:= 3, 6, 1 ;**

f, g, h:= 3, 6, 1

>**f ;**

3

>**h ;**

1

No Maple ao se trabalhar com seqüências, caso os temos de uma seqüência sejam eles próprios seqüências, este automaticamente transforma tudo em uma única seqüência.

```
>s:= 1, 2, 3, 4;
s:= 1, 2, 3, 4

>b:= 1, 3, 5, 7;
b:= 1, 3, 5, 7

>c:= x, y, z, w;
c:= x, y, z, w

>s, b, c;
1, 2, 3, 4, 1, 3, 5, 7, x, y, z, w
```

O comando **seq** pode ser usado para construir seqüências da seguinte forma:

```
>s:= seq(i^4, i={x,y,z});
s:= x^4, y^4, z^4

>a:= seq(i^2, i={2, 3, 4});
a:= 4, 9, 16
```

2.2. Lista

Uma lista é uma seqüência delimitada por colchetes, ou seja, é algo do tipo:
[seqüência]

Exemplo:

```
>data_list := [1, 2, 3, 4, 5];
data-list:= [1, 2, 3, 4, 5]

>polynomials := [x^2+3, x^2+3*x-1, 2*x];
polynomials:= [x^2+3, x^2+3x-1, 2x]

>participantes := [Kathy, Frank, Rene, Niklaus, Liz];
participantes:= [Kathy, Frank, Rene, Niklaus, Liz]
```

Citaremos aqui alguns comandos usados para trabalhar com listas:

```
>L := [a, b, c, d, e];
L:= [a, b, c, d, e]

>L[2]; retorna o segundo elemento da lista L
b

>nops(L); retorna o número de elementos da lista L
5

>op(L); extrai os elementos da lista L
a, b, c, d, e

>map(x->x^3, L); aplica a função  $x \rightarrow x^3$  aos elementos da Lista L
[a^3, b^3, c^3, d^3, e^3]
```

Caso deseje substituir algum elemento de uma lista basta usar o comando *subs*.

```
>L := [a, b, c, d, e];
L:= [a, b, c, d, e]
```



```

>subs (b=2, L);
\[a, 2, c, d, e\]
>lista:= [sin(x), ln(x), x^2];
lista := \[sin\(x\), ln\(x\), x^2\]
>subs (x=Pi, lista);
\[sin\(π\), ln\(π\), π^2\]

```

Obs.: Cabe observar que em uma lista ou em uma sequência a ordem em que os elementos são dispostos é relevante, além disso os elementos podem se repetir.

2.3. Conjunto

É uma sequência delimitada por chaves. As propriedades são as mesmas das dos conjuntos em Matemática. Exemplos:

```
>data_set := {1, -1, 0, 10, 2};  
data_set:= { 1, -1, 0, 10, 2}  
  
>unknowns := {x, z, y}  
unknowns := {x, z, y}  
  
>set1:= {a, b, c}; set2:={c, d, e};  
set1:= {a, b, c}  
set2:={c, d, e}  
  
>set1 union set2; nos dá a união do conjunto set1 com o conjunto set2  
{a, b, e, d, c}  
  
>set1 intersect set2; nos dá a intersecção do conjunto set1 com o conjunto set2  
{c}  
  
>nops(set1); op(set1);  
3  
a, b, c  
  
>map(x->x^2, set1);  
{c^2, a^2, b^2}
```

O Maple não preserva a ordem dos elementos nem repetição dos mesmos. Isto é, conjuntos para o Maple tem as mesmas propriedades que estes possuem na Matemática.. Dessa forma os três conjuntos abaixo são idênticos.

```
>{a, b, c}, {c, b, a}, {a, a, b, c, a};  
{b,c,a}, {b,c,a}, {b,c,a}
```

2.4. Operações com Conjuntos e Listas

Veremos aqui algumas operações que são feitas pelo Maple a respeito de conjuntos e listas.

```
>numeros := {0, Pi/3, Pi/2, Pi};  
numeros:= {  $\pi$ , 0,  $\pi/3$ ,  $\pi/2$ }  
  
>map(sin, numeros); nos dá o valor da função seno em cada ponto do elemento do conjunto  
{0, 1,  $\sqrt{3}/2$ }  
  
>conj1 := [Kate, Tom, Steve];
```

conj1 := [Kate, Tom, Steve]
 >**member**(Tom, conj1); esse comando **member** diz se o elemento indicado é membro ou não da lista.

true

>**conj2 := {5, 6, 3, 7};**
conj2 := {5, 6, 3, 7}
 >**member**(2, conj2); nos diz se elemento 2 está no conjunto
false

Também o Maple possibilita localizar qual o elemento da lista, basta usar a notação $[n]$ onde n identifica a posição do elemento na lista. Por exemplo:

>**conj1 := [Kate, Tom, Steve];**
conj1 := [Kate, Tom, Steve]
 > **conj1[2];**
Tom

O Maple entende conjuntos e listas que não tem nenhum elementos. Por exemplo:

>**empty_set := {};**
empty_set := {}
 >**empty_list := [];**
empty_list := []

No Maple pode-se criar um novo conjunto, usando por exemplo, o comando **union**. Por exemplo:

>**velho := {2, 3, 4};**
velho := {2, 3, 4}
 >**novo := velho union {2, 5};**
novo := {2, 3, 4, 5}

>**conj3:= velho minus {2, 5};** esse comando minus nos dá todos os elementos que pertence ao conjunto velho e não pertence ao conjunto {2, 5}, .
conj3:= {3,4}

2.5. Array

Array é uma estrutura de dados que encontra muitas aplicações, por exemplo em Álgebra Linear. É uma maneira conveniente de definir vetores e matrizes usando listas. *Array* é uma extensão de uma lista. Uma lista como vimos acima é um grupo de itens onde se pode associar a cada item um número inteiro positivo. A estrutura de dados *array* é uma generalização dessa idéia. Cada elemento estará associado a um índice, mais não necessariamente restritos a uma dimensão. Além disso, esses índices poderão ser zero, ou números inteiros negativos. Vejamos alguns exemplos:

>**array1:=array(1..3);**
array1 := array(1 .. 3,[])

Ao realizar essa operação o Maple cria uma lista (como se fosse um vetor) de nome `array1`, de três coordenadas (devido a variação “1..3”) cujos valores ainda não estão definidos,

```
>array1[1];
```

array1₁

isso significa que o primeiro elemento do *array1* ainda não está definido. Podemos proceder da seguinte forma para definir os elementos de *array1*:

```
>array1[1]:= 1; array1[2]:= 2; array1[3]:= 3;
```

array1₁ := 1

array1₂ := 2

array1₃ := 3

Com isso, *array1* passa a ser uma lista com três entradas definidas. Para visualizá-la podemos fazer:

```
>print(array1);
```

[1, 2, 3]

Há outra forma mais direta de definir o *array1* acima. Vejamos:

```
>array1:=array(1..3, [1,2,3]);
```

array1 := [1, 2, 3]

Como foi dito acima a estrutura de dados *array* não necessariamente está restrita a uma dimensão apenas. Podemos usar *array* para construir por exemplo matrizes. Vamos utilizar o último procedimento acima.

```
>array2:=array(1..3,1..3, [[1,2,3],[4,5,6],[7,8,9]]);
```

array2 :=
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

note que os parâmetros “1..3, 1..3” significam que o *array2* terá dois índices cada qual variando de 1 até 3, daí o motivo do resultado ser uma matriz 3×3. Os colchetes representam as entradas da matriz, portanto deve-se tomar um certo cuidado na hora de defini-los. Pode-se usar esse procedimento para construir outros tipos de matrizes não necessariamente numéricas.

```
>array3:=array(1..2,1..3, [[feijão,arroz,carne],[1.4,1.0,6.3]]);
```

array3 :=
$$\begin{bmatrix} \text{feijão} & \text{arroz} & \text{carne} \\ 1.4 & 1.0 & 6.3 \end{bmatrix}$$

para seleccionar algum elemento da matriz, basta usar a notação “*matriz*[*i,j*]”.

```
>array3[1,2]; array3[2,2];
```

arroz

1.0

Suponha que quiséssemos colocar em sequência os elementos da primeira linha, então podemos proceder da seguinte forma:

```
>seq(array3[1,i],i=1..3);
```

feijão, arroz, carne

da mesma forma os elementos da segunda linha,

```
>seq(array3[2,i],i=1..3);
```

1.4, 1.0, 6.3

Pode-se construir arrays mais gerais que não sejam matrizes, por exemplo:

```
>array4:=array(1..2,1..2,1..2,[[[1,2],[3,4]],[[5,6],[7,8]]]);
```

```
array4 := array(1 .. 2, 1 .. 2, 1 .. 2,[
(1, 1, 1) = 1
(1, 1, 2) = 2
(1, 2, 1) = 3
(1, 2, 2) = 4
(2, 1, 1) = 5
(2, 1, 2) = 6
(2, 2, 1) = 7
(2, 2, 2) = 8
])
```

```
>array4[1,2,1]; array4[2,2,1];
```

3
7

Ao se construir matrizes com o comando array pode-se fazer operações usuais de matrizes normalmente.

```
>array5:=array(1..2,1..2,[[1,2],[3,4]]);
array6:=array(1..2,1..2,[[5,6],[7,8]]);
```

```
array5:=
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
array6:=
```

$$\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

```
>evalm(array5*array6); evalm(array5-array6);
```

$$\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$\begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}$$

Obs.: para efetuar a multiplicação de matrizes devemos utilizar o caracter “&” antes de *, além disso o comando **evalm** faz com que o cálculo seja efetuado. Para outros tipos de operações com matrizes devemos colocar o seguinte comando:

```
>with(linalg):
```

tal comando será discutido posteriormente.

>**det(array5) ;**

-2

>**inverse(array6) ;**

$$\begin{bmatrix} -4 & 3 \\ \frac{7}{2} & -\frac{5}{2} \end{bmatrix}$$

>**evalm(array6*%) ;**

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

>**charpoly(array5,x) ;**

$$x^2 - 5x - 2$$

o comando acima calcula o polinômio característico da matriz array5.

>**eigenvalues(array5) ;**

$$\frac{5}{2} + \frac{1}{2}\sqrt{33}, \frac{5}{2} - \frac{1}{2}\sqrt{33}$$

o comando acima calcula os autovalores da matriz array5.

3. Encontrando Soluções (sistemas e expressões), Funções e Polinômios

3.1. Resolução de Equações e Sistemas

O comando **solve** (resolver), serve para resolver equações diversas. No exemplo abaixo resolveremos uma equação na variável x .

>**equal := x^3+3*x^2-4=0 ;**

$$equal := x^3 + 3x^2 - 4 = 0 ;$$

>**solve(equal) ;**

$1, -2, -2$

Quando a equação possui mais de uma variável, é fundamental indicar ao sistema a incógnita do problema.

>**equa2 := 2*x+y=0 ;**

$$equa2 := 2x + y = 0 ;$$

>**solve(equa2, x) ;**

$$(-1/2)y$$

>**solve(equa2, y) ;**

$$-2x$$

O comando **solve** resolve também equações com raízes complexas. A letra (**I**) representa a unidade imaginária dos números complexos.

```
> solve(x^2+1,x);
```

$I, -I$

Se o comando **solve** não conseguir apresentar exatamente as raízes desejadas, podemos então executar o comando **fsolve** (resolver com ponto flutuante) para se obter raízes aproximadas.

```
> equa3 := x^6 - 2*x^2 + 2*x;
```

$x^6 - 2x^2 + 2x$

```
> solve(equa3,x);
```

$0, \text{RootOf}(_Z^5 - 2_Z + 2)$

A expressão **RootOf** na resposta acima é uma forma simbólica de o Maple dizer que as outras soluções da equação **equa3** são as raízes da equação $z^5 - 2z + 2$.

```
> fsolve({equa3},x);
```

$\{x = 0\}, \{x = -1.364430112\}$

Obs.: a utilização das chaves em torno de **equa3** acima faz com que as soluções sejam expressas em termos de conjuntos.

Por questões práticas, a função **fsolve** não mostra automaticamente as raízes complexas. É preciso adicionar a opção **complex**.

```
> fsolve({equa3},x,complex);
```

$\{x = 0\}, \{x = -1.364430112\}, \{x = -.1929606049 - 1.268917574I\}, \{x = -.1929606049 + 1.268917574I\}, \{x = -8751756609 - .3519221356I\}, \{x = -8751756609 + .3519221356I\}$

Os comandos **solve** e **fsolve** também funcionam com sistemas. A sintaxe é a seguinte:

solve({ equações }, { incógnitas }).

Observe a utilização das chaves **{ }** para representar conjunto de equações e de incógnitas.

```
> equa4 := x + 2*y + 3*z = 7;
```

$equa4 := x + 2y + 3z = 7$

```
> equa5 := 5*x - 2*y = 12;
```

$equa5 := 5x - 2y = 12$

```
> equa6 := 2*x - y + 3*z = -6;
```

$equa6 := 2x - y + 3z = -6$

```
> sol := solve( {equa4,equa5,equa6} , {x,y,z} );
```

$$sol:=\{x=\frac{63}{13}, z=-\frac{125}{39}, y=\frac{77}{13}\}$$

>evalf(sol,13);

$$\{x=4.769230769231, z=-3.205128205128, y=5.923076923077\}$$

Consideremos um sistema de cinco variáveis.

```
>eqn1:= x+2*y+3*z+4*t+5*u=41:
>eqn2 := 5*x+5*y+4*z+3*t+2*u=20:
>eqn3 := 3*y+4*z-8*t+2*u=125:
>eqn4 := x+y+z+t+u=9:
>eqn5 := 8*x+4*z+3*t+2*u=11:
```

Agora resolvendo o sistema com as cinco variáveis.

```
>s1 := solve({eqn1, eqn2, eqn3, eqn4, eqn5}, {x, y, z, t, u});
```

$$s1:= \{x=2, y=3, u=16, t=-11, z=-1\}$$

Podemos também resolver o sistema de equações acima só com três variáveis. Por exemplo:

```
>s2 := solve({eqn1,eqn2, eqn3}, {x, y, z})
```

$$s2:=\{z=-7t-(\frac{59}{13})u-(\frac{70}{13}), y=12t+(\frac{70}{13})u+\frac{635}{13}, x=-7t-(\frac{28}{13})u-\frac{527}{13}\}$$

Se quisermos podemos achar o valor de x , y , z no sistema acima quando atribuímos valores para u e t . Por exemplo:

```
>eval(s2, {u=1, t=1});
```

$$\{x=-\frac{643}{13}, y=\frac{861}{13}, z=-\frac{220}{13}\}$$

3.2. Funções de uma variável

Para definir-se uma função de uma variável deve-se escolher um nome para a mesma, por exemplo "f". Deve-se digitar esse nome seguido do comando "dois pontos – igual", que é o comando clássico de atribuição de nomes no Maple. A seguir deve-se digitar o nome da variável, depois o comando de transformação que é obtido através da sequência de caracteres "ifem – sinal de maior". Finalmente digita-se a lei de formação da função propriamente dita.

$$f := (\text{variáveis}) \rightarrow (\text{expressão contendo variáveis})$$

Exemplo: entrar com a função $f(x) = x^2 \sin(x)$.

```
>f := x -> (x^2)*sin(x);
```

$$f := x \rightarrow (x^2)\sin(x)$$

Entrar com a função $g(x) = 2 + t^2 + 3t^3 + 6t^4$. Pode-se digitar a entrada em sequência, isto é, sem os espaços digitados no exemplo acima.

>**g:=t->2+t^2+3*t^3+6*t^4;**

$$g:=t \rightarrow 2+t^2+3t^3+6t^4$$

Após a entrada das expressões das funções, elas ficam memorizadas e podem ser chamadas a qualquer instante, bastando digitar o nome da função e o ponto ou a variável que se quer calcular a função.

Chame novamente a expressão da função $f(x)$, e também o valor de $g(a)$ e $g(2.76)$. Convém observar que o comando abaixo só será executado a contento se os comandos acima tiverem sido executados previamente.

>**f(x);**

$$x \rightarrow (x^2)\sin(x)$$

>**g(a);**

$$2+a^2+3a^3+6a^4$$

>**g(2.76);**

$$420.8583066$$

Se o leitor entrar com uma nova função com um nome já utilizado, a primeira atribuição do nome será apagada da memória e só ficará disponível a nova definição da função. Por exemplo, se entrarmos agora com uma nova definição para o nome "**g**" a anterior será apagada. O leitor pode verificar:

>**g:=h->3*h^3;**

$$g:=h \rightarrow 3h^3$$

>**g(1);**

$$3$$

>**g(2.76);**

$$63.073728$$

>**g(x);**

$$3x^3$$

Pode-se também utilizar funções já definidas na definição de novas funções. Por exemplo, o leitor pode definir uma nova função que calcula a média aritmética dos valores das funções em determinados pontos .

> **f:=x->2*x;**

$$f:=x \rightarrow 2x$$

>**g:=x->6*x;**

$$g:=x \rightarrow 6x$$

>**m:=x->(f(x)+g(x))/2;**

$$m:=x \rightarrow (f(x)+g(x))/2;$$

>**m(x);**

$$4x$$

Deste modo o leitor pode definir qualquer operação aritmética envolvendo funções já definidas.

Para calcular a composta de **f** com **g**, isto é, **(fog)(x)** pode-se digitar exatamente a forma utilizada tradicionalmente na matemática:

> **f(g(x));**

$$12x$$

Exemplo:

> **g:=x->x^2;**

$$g:=x \rightarrow x^2$$

> **f:=x->2*x;**

$$f:=x \rightarrow 2x$$

> **g(f(x));**

$$4x^2$$

Agora se quisermos definir uma nova função como sendo a composta de **f** com **g**, então podemos utilizar o símbolo “@” que faz o papel da “bolinha” na composta de duas funções.

> **h:=x->(f@g)(x);**

$$h=f@g$$

> **h(x);**

$$12x$$

3.2.1 Funções reconhecidas (usuais)

Existe uma grande quantidade de funções reconhecidas pelo Maple. Para ver a totalidade das funções pré-definidas utilize o Help do menu principal e dar busca no tópico inifen.

As funções mais usadas estão listadas abaixo. Deve-se lembrar que as variáveis envolvidas podem ser nomeadas pelo leitor.

- > **exp(x);** função exponencial
- > **abs(x);** função modular
- > **sin(x);** função seno
- > **cos(x);** função cosseno
- > **tan(x);** função tangente
- > **sec(x);** função secante
- > **csc(x);** função cossecante
- > **cot(x);** função cotangente
- > **arcsin(x);** função arcoseno
- > **arccos(x);** função arcocosseno
- > **log10(x);** função logaritma
- > **ln(x);** função log natural

O Maple, nesta versão, trabalha preferencialmente com logaritmos naturais, tanto faz digitarmos **log** ou **ln**. Caso seja preciso o cálculo em outra base, por exemplo, pode-se usar o comando **log[b](x)**. Exemplo:

> **log[3](27);**

$$\ln(27)/\ln(3)$$

O leitor deve perceber que o Maple transforma outras bases no logaritmo natural, para ver o valor da expressão deve-se pedir o comando **evalf** na expressão anterior. O símbolo % é compreendido pelo Maple como a última expressão efetuada.

> **evalf(%);**

3.2.2 Funções definidas por partes

Em muitas situações é necessário definir-se uma função de uma variável com várias expressões algébricas diferentes, ou seja, uma função definidas por várias sentenças. Neste caso, para definir-se a função utiliza-se o comando `piecewise` como nos exemplos abaixo.

Defina a função $p(x) = \begin{cases} -x & \text{se } x < 0 \\ x & \text{se } x \geq 0 \end{cases}$.

A sintaxe do comando `piecewise` é a seguinte:

`piecewise(cond1, f1, cond2, f2, ..., condn, fn)` ou ainda
`piecewise(cond1, f1, cond2, f2, ..., condn-1, fn-1, fn)`

A primeira opção é utilizada quando se deseja obter cada uma das expressões exatamente em seus respectivos domínios,

```
>p:=x-> piecewise(x<0,-x,x>=0,x);
p:=x→piecewise(x<0,-x,x>=0,x)
>p(x);
```

$$\begin{cases} -x & x < 0 \\ x & 0 \leq x \end{cases}$$

A segunda é descrever apenas uma componente e atribuir outra expressão ao complementar do domínio dado:

```
>p:=x->piecewise(x<0,-x,x);
p:=x→piecewise(x<0,-x,x)
>p(x);
```

$$\begin{cases} -x & x < 0 \\ x & otherwise \end{cases}$$

Se, para construir uma nova função, for necessário três definições, pode-se utilizar do artifício de definir as expressões nas componentes infinitas e deixar a definição da componente limitada como restante. Por exemplo, para entrar com a definição de

$$h(x) = \begin{cases} 2x & x < 0 \\ \sin(x) & 0 \leq x \leq \frac{\pi}{2} \\ 1 & x > \frac{\pi}{2} \end{cases}, \quad \text{podemos proceder da seguinte forma}$$

```
> h:=x-> piecewise(x<0,2*x,x>Pi/2,1,sin(x));
p:=x→piecewise(x<0,2x,x>π/2,1,sin(x))
>h(x);
```

$$\begin{cases} 2x & x < 0 \\ 1 & \frac{1}{2}\pi < x \\ \sin(x) & otherwise \end{cases}$$

Uma outra forma de montar funções com três ou mais variações de domínio é fazer a descrição dos segmentos limitados do domínio utilizando duas desigualdades simples unidas com o comando **and**, se por exemplo for necessário entrar com a função,

$$g(x) = \begin{cases} 2x & x < 0 \\ \sin(x) & 0 \leq x < \frac{\pi}{2} \\ 1 & \frac{\pi}{2} \leq x < 3 \\ -x + 5 & x > 3 \end{cases}$$

deve-se digitar as variações de x em $[0, \frac{\pi}{2})$ e $[\frac{\pi}{2}, 3)$ como intersecção de duas desigualdades. O procedimento é descrito no exemplo:

>g:=x->piecewise(x<0,2*x,0<=x and x<Pi/2,sin(x),Pi/2<=x and x<3,1,3<x,-x+5);

$g:=x \rightarrow \text{piecewise}(x < 0, 2x, 0 \leq x \text{ and } x < \frac{1}{2}\pi, \sin(x), \frac{1}{2}\pi \leq x \text{ and } x < 3, 1, 3 < x, -x+5)$

>g(x);

$$\begin{cases} 2x & x < 0 \\ \sin(x) & -x \leq 0 \text{ and } x - \frac{1}{2}\pi < 0 \\ 1 & \frac{1}{2}\pi - x \leq 0 \text{ and } x - 3 < 0 \\ -x + 5 & 3 < x \end{cases}$$

Existe um comando que fica associado ao comando **piecewise** que é o **convert**. Ele pode ser utilizado quando se deseja transformar a expressão de uma função onde aparecem módulos, por exemplo, numa expressão **piecewise**.

Considere o seguinte problema: seja $f(x) = 1 - |x|$ a função envolvendo módulo, reescreva-a como uma função definida por sentenças. Procedemos da seguinte forma:

>f:=x->abs(1-abs(x));

$$f:=x \rightarrow |1 - |x||$$

>convert(f(x),piecewise);

$$\begin{cases} -1 - x & x \leq -1 \\ 1 + x & x \leq 0 \\ 1 - x & x \leq 1 \\ x - 1 & 1 < x \end{cases}$$

A grande vantagem do comando **piecewise** é permitir a definição de funções utilizando expressões e desigualdades que não sejam imediatamente visíveis. Neste caso pode-se utilizar o comando **simplify** para simplificar a expressão da função, por exemplo

podemos definir a função $w(x) = \begin{cases} -x & x^2 - 4 < 0 \\ x^2 & \text{otherwise} \end{cases}$, e simplificar sua expressão.

> w:=x->piecewise(x^2-4<0, -x, x^2);

$$w:=x \rightarrow \text{piecewise}(x^2 - 4 < 0, -x, x^2)$$

>simplify(w(x));

$$\begin{cases} x^2 & x \leq 2 \\ -x & x < 2 \\ x^2 & 2 \leq x \end{cases}$$

O comando **simplify** também pode ser utilizado quando se deseja efetuar operações entre funções que são definidas por várias sentenças. Por exemplo, suponhamos que quiséssemos multiplicar a função $w(x)$ com a última função $g(x)$ definida acima.

>**g(x)*w(x) ;**

$$\left(\begin{cases} 2x & x < 0 \\ \sin(x) & -x \leq 0 \text{ and } x - \frac{1}{2}\pi < 0 \\ 1 & \frac{1}{2}\pi - x \leq 0 \text{ and } x-3 < 0 \\ -x+5 & 3 < x \end{cases} \right) \left(\begin{cases} -x & x^2 - 4 < 0 \\ x^2 & \text{otherwise} \end{cases} \right)$$

o Maple apenas deixa indicada de forma simbólica a multiplicação. Agora para ver o resultado da multiplicação usamos o **simplify**:

>**simplify(g(x)*w(x)) ;**

$$\begin{cases} 2x^3 & x \leq -2 \\ -2x^2 & x \leq 0 \\ -\sin(x)x & x \leq \frac{1}{2}\pi \\ -x & x < 2 \\ x^2 & x < 3 \\ 0 & x = 3 \\ -x^3 + 5x^2 & 3 < x \end{cases}$$

3.3. Operações com Polinômios

Em computação algébrica também podemos operar polinômios simbolicamente. Começamos com uma expressão polinomial. O Maple V automaticamente simplifica o polinômio juntando os monômios semelhantes.

>**x^2+x-8-3*x ;**

$$x^2 - 2x - 8$$

Para fatorar o polinômio digitamos

>**factor(%) ;**

$$(x+2)(x-4)$$

o comando **factor** fatora o polinômio.

Outro exemplo;

>**expand((x*y-x^2)*(y-x)*(y^2-x^2)) ;**

$$y^4x - 2x^2y^3 + 2x^4y - x^5$$

o comando **expand** efetua multiplicações entre polinômios.

Consideremos os dois polinômios abaixo:

>**p1:=7*x^4-3*x^3+7*x^2-3*x ;**

$$p1 := 7x^4 - 3x^3 + 7x^2 - 3x$$

>**p2:=5*x^5+3*x^3+x^2-2*x+1 ;**

$$p2 := 5x^5 + 3x^3 + x^2 - 2x + 1$$

Dividindo $p1$ por $p2$:

>p1/p2;

$$\frac{7x^4 - 3x^3 + 7x^2 - 3x}{5x^5 + 3x^3 + x^2 - 2x + 1}$$

Normalizando (simplificando) a expressão racional acima:

>normal(%);

$$\frac{(7x-3)x}{5x^3 - 2x + 1}$$

O comando **normal** acima pode ser utilizado quando se deseja simplificar expressões racionais envolvendo polinômios no numerador e denominador. O comando faz com que o Maple fatore o numerador e o denominador simplificando os fatores comuns. Portanto, determinadas expressões racionais permanecerão inalteradas mediante esse comando.

>normal((x^3-1)/(x^2+1));

$$\frac{x^3 - 1}{x^2 + 1}$$

Um outro tipo de simplificação, particularmente útil na resolução de integrais, é a transformação de uma função racional em frações parciais.

>(x^5+1)/(x^4-x^2);

$$\frac{x^5 + 1}{x^4 - x^2}$$

>convert(% , parfrac, x);

$$x + \frac{1}{x-1} - \frac{1}{x^2}$$

Vejamos agora outros comandos específicos no tratamento com polinômios. Vamos considerar polinômios da forma $a_n x^n + \dots + a_2 x^2 + a_1 x + a_0$. Ao entrar com um polinômio no Maple, este fornece exatamente o polinômio como você o digitou, ou seja, os termos não são ordenados pelo Maple em ordem ascendente ou decrescente em relação ao grau.

>x^2+x^5+2*x^4+1+x;

$$x^2 + x^5 + 2x^4 + 1 + x$$

Para arrumar os termos em ordem decrescente podemos usar o comando **sort**

>sort(%);

$$x^5 + 2x^4 + x^2 + x + 1$$

Dado um polinômio qualquer, o coeficiente dominante desse polinômio é o coeficiente do termo de mais alto grau. O Maple reconhece esse coeficiente, bem como todos os outros

>p3:=-17*x^6+11*x^4-20*x^3+13*x^2-3*x+20;

$$-17x^6 + 11x^4 - 20x^3 + 13x^2 - 3x + 20$$

>lcoeff(p3);

$$-17$$

>coeffs(p3,x,'potencias'), potencias;

$$\begin{matrix} 20, -3, 11, -20, 13, -17 \\ 1, x, x^4, x^3, x^2, x^6 \end{matrix}$$

Se quisermos saber o grau do polinômio $p3$,

```
>degree(p3) ;
```

6

Vejamos outros comandos no tratamento com polinômios,

```
>poly1:= x+x^2-x^3+1-x^4;
```

$poly1:=x+x^2-x^3+1-x^4$

```
>poly2:= x*y+z*x*y+y*x^2-z*y*x^2+x+z*x;
```

$poly2:=xy+zx y+yx^2-zyx^2+x+zx$

```
>collect(poly2, x) ;
```

$(y-zy)x^2+(y+zy+1+z)x$

```
>collect(poly2, z) ;
```

$(xy-yx^2+x)z+xy+yx^2+x$

Existem várias formas de se interpretar o polinômio **poly2** acima. Por exemplo, podemos interpretá-lo como um polinômio na variável x , onde todas as outras letras são consideradas constantes. Para proceder dessa forma utilizamos o comando **collect** como no primeiro exemplo. Podemos fazer o mesmo em relação a z , como no segundo exemplo.

O comando **divide** abaixo nos diz se a divisão dos polinômios considerados é exata. No caso afirmativo o Maple retorna a palavra *true* (verdadeiro) e caso contrário retorna *false* (falso).

```
>divide(x^3-y^3, x-y) ;
```

true

```
>divide(x^3-x, x-5) ;
```

false

Dado um polinômio qualquer às vezes nos interessa saber o valor desse polinômio para um determinado valor de x . Os exemplos abaixo ilustram como isso pode ser feito.

```
>poly3:= x^2+3*x-4;
```

$poly3:=x^2+3x-4;$

```
>eval(poly3, x=2) ;
```

6

```
>poly4:= 3*z-z^2+2*z-3*z+1;
```

$poly4:=3z-z^2+2z-3z+1$

```
>eval(poly4, z=sqrt(2)) ;
```

$2\sqrt{2}-1$

Um dos resultados mais importantes na teoria de polinômios dentro da Álgebra é o algoritmo da divisão (ou algoritmo de Euclides). Dados dois polinômios f e g , onde $g \neq 0$, o algoritmo da divisão nos diz que existem polinômios q e r tais que:

$$f = gq + r, \text{ onde } r = 0 \text{ ou } \partial r < \partial g$$

O polinômio q e r são chamados respectivamente o **quociente** e o **resto** na divisão de f por g .

Dados dois polinômios f e g o Maple pode calcular o quociente e o resto na divisão de f por g . Os Comandos são respectivamente **quo** e **rem**,

```
>f:= x^3+x+1;
```

$f:=x^3+x+1$


```

>g:= x^2+x+1;
g:= x^2+x+1
>q:=quo(f,g,x);
q:= x-1
>r:=rem(f,g,x);
r:= 2+x

```

Para confirmar, fazemos $gq + r$ e verificamos se o resultado é igual a f .

```

>expand(g*q+r);
x^3+x+1

```

ou ainda

```

>teste(f=g*q+r);
true

```

o Maple retorna *true* indicando que a igualdade é verdadeira.

Para finalizar, uma outra operação que pode ser realizada no Maple é o máximo divisor comum (*greatest common divisor – gcd*). Dados dois polinômios f e g , um polinômio d se diz máximo divisor comum de f e g se

- i) $d \mid f$ e $d \mid g$;
- ii) se d_1 é tal que $d_1 \mid f$ e $d_1 \mid g \Rightarrow d_1 \mid d$

O cálculo do mdc de dois polinômios pode ser efetuado de forma análoga ao processo utilizado com números inteiros, através do método das divisões sucessivas. O Maple realiza esses cálculos automaticamente através do comando **gcd**.

```

>f:= x^5+2*x^3-5*x^2+3;
f:= x^5+2x^3-5x^2+3
>g:= -x^3+5x^2-x+3;
g:= -x^3+5x^2-x+3
>gcd(f,g);
1

```

4. Gráficos de Funções 2D e 3D

4.1. Gráficos em duas dimensões

4.1.1 Gráficos de funções

Consideremos uma função qualquer de uma variável $y = f(x)$. O comando para traçar o gráfico de f é **plot** o qual possui algumas variações em sua sintaxe dependendo do que se deseja traçar. A sintaxe básica é a seguinte:

plot(f, h, v, ops)

onde,

f = expressão ou nome da função;

h = intervalo no eixo das abscissas no qual o gráfico será visualizado;

v = intervalo no eixo das ordenadas que controla a visualização vertical do gráfico;

ops = opções de visualização e formatação.

Os parâmetros **f** e **h** no comando **plot** são obrigatórios, enquanto que os parâmetros **v** e **ops** são opcionais. Vejamos alguns exemplos:

```
>plot(sin(x),x=-2*Pi..2*Pi);
>plot(x^2,x=-5..5);
>f:= x->7*sin(x)+sin(7*x);
>plot(f(x),x=0..8);
```

Quando o parâmetro **v** não é colocado o Maple automaticamente atribui ao “eixo **y**” (vertical) o maior e o menor dos valores funcionais correspondentes ao intervalo **h**. O parâmetro **v** é útil quando desejamos focalizar determinadas partes do gráfico.

```
>plot(f(x),x=0..3.2,y=0..8);
```

Uma outra situação em que é conveniente atribuir uma variação no parâmetro **v**, é quando se deseja modificar a escala do gráfico. Por exemplo,

```
>plot(x^3,x=-10..10)
```

note que o gráfico está completamente fora de escala. Podemos contornar este problema atribuindo uma variação conveniente no parâmetro **v**.

```
>plot(x^3,x=-10..10,y=-10..10);
```

Em relação ao parâmetro **ops** podemos destacar as seguintes opções:

axes = normal, boxed, frame ou none – muda a visualização dos eixos coordenados;

discont = true – força o Maple a exibir as descontinuidades de uma função, quando for o caso;

color = “nome da cor” – especifica a cor do gráfico;

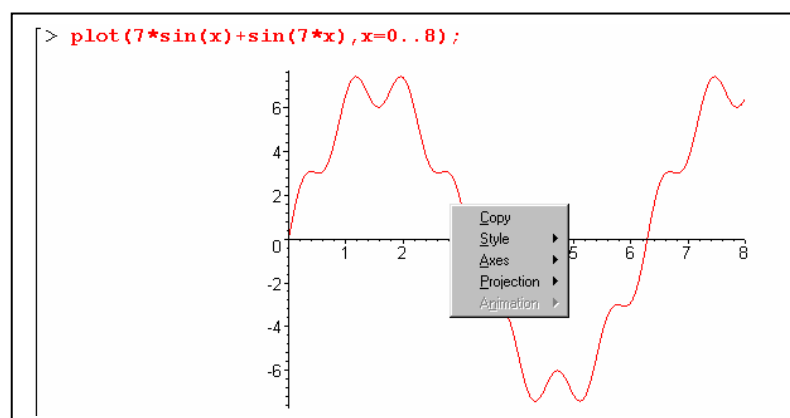
numpoints = *n* – especifica o número de pontos que o Maple vai utilizar para traçar um gráfico. Em geral, o Maple utiliza numpoints=50;

xtickmarks = *n* – especifica uma certa quantidade de pontos (não menor do que *n*) sobre o eixo **x**;

ytickmarks = *m* - especifica uma certa quantidade de pontos (não menor do que *m*) sobre o eixo **y**. Pode-se fazer a mudança nos dois eixos diretamente utilizando **tickmarks** = [*n*, *m*].

title = `título` - esta opção pode ser utilizada quando se deseja colocar um título para o gráfico. Note que o título deve estar delimitado entre dois acentos graves.

Há ainda outras opções, que podem ser testadas quando o gráfico é visualizado na tela. Dê um clique sobre o gráfico (com o botão direito do mouse), note o aparecimento de uma janela, a qual contém as opções:



Também, dando um clique sobre o gráfico (com o botão esquerdo) aparecerá uma moldura em torno do gráfico (a qual você poderá mudar o tamanho da forma que desejar), além disso na barra de ferramentas aparecerão os ícones



os quais também ocasionam mudanças no gráfico.

Exemplos:

```
>plot(1/(1-x),x=-3..5,y=-5..5);
>plot(piecewise(x<-1,1,x<0,0,x<1,-1,x<2,1,3),x=-5..5);
(tente a opção discontinuous=true)
>plot(1/(1-x),x=-3..5,y=-5..5,axes=framed,discontinuous=true,
color=green,ytickmarks=7,title=`testando,testando`);
>plot(6*x^3+2*x^2-2*x+1,x=-2..2,y=-2..2);
```

Note que, neste último exemplo, ao visualizarmos o gráfico num fator de zoom maior as partes mais curvas do gráfico ficam distorcidas. Para corrigirmos esse problema utilizamos a opção **numpoints** para aumentar o número interno de pontos que o Maple utiliza para traçar o gráfico,

```
>plot(6*x^3+2*x^2-2*x+1,x=-2..2,y=-2..2,numpoints=500);
```

4.1.2 Gráficos de curvas no plano

Para entendermos os tópicos abordados neste parágrafo é importante recordarmos rapidamente a noção de curvas planas, as quais generalizam as curvas obtidas por meio de gráficos de funções $y = f(x)$.

Uma **curva plana** é um conjunto C de pares ordenados da forma
 $(f(t), g(t))$
 onde as funções f e g são contínuas em um intervalo I .

O gráfico da curva C da definição acima é o conjunto de todos os pontos do plano cartesiano que são da forma $P(t) = (f(t), g(t))$, onde t varia em I .

As equações abaixo,

$$x = f(t) \text{ e } y = g(t)$$

onde $t \in I$, são chamadas equações paramétricas de C ; t é o parâmetro.

Para traçarmos o gráfico de uma curva na forma paramétrica no Maple, utilizamos a seguinte sintaxe:

```
plot([f(t),g(t),t=a..b],h,v,ops);
```

onde,

f e g são as funções contínuas (que dependem do parâmetro t) no intervalo de variação de t ;

$t = a..b$ – intervalo de variação de t ;

h , v e ops – são as opções já vistas anteriormente, no caso de gráficos de funções.

Os itens h , v e ops nessa modalidade de gráfico são opcionais.

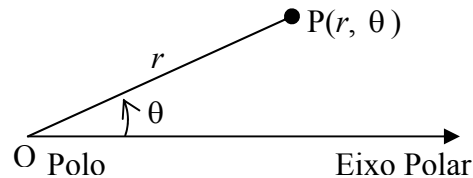
Exemplos:

```
>plot([t,t^2,t=-5..5]);
>plot([cos(t),sin(t),t=0..2*Pi]);
plot([cos(t),sin(t),t=0..2*Pi],scaling=constrained);
>plot([t-sin(t),t-cos(t),t=-3*Pi..3*Pi]);
>plot([cos(t)^3,sin(t)^3,t=-Pi..Pi]);
```

```
>plot([cos(t)^3,sin(t)^3,t=-Pi..Pi], color=blue, axes=
framed, title=`curva parametrizada`);
```

4.1.3 Gráficos em Coordenadas Polares

Coordenadas cartesianas é a forma usual utilizada pelo Maple para especificar um ponto no plano. Existem outras maneiras, dentre elas, o uso de coordenadas polares, (r, θ) .



Em coordenadas polares, r é a distância de O até P, enquanto que θ é o ângulo entre o segmento \overline{OP} e o eixo polar. Uma **equação polar** é uma equação em r e θ . Uma **solução** de uma equação polar é uma par ordenado (a, b) que conduz a igualdade quando se substitui r por a e θ por b . O gráfico de uma equação polar é o conjunto de todos os pontos que correspondem às soluções.

Exemplos:

$$r = 1 \quad ; \quad r = 4 \sin(\theta) \quad ; \quad r = \theta \quad ; \quad r = 2 \cos(3\theta) \quad ;$$

Para traçar gráficos em coordenadas polares, primeiramente precisamos carregar o pacote “plots” que possui várias opções gráficas, dentre as quais se destaca o comando que iremos utilizar. O pacote pode ser carregado através do comando:

```
>with(plots);
```

O comando que usaremos é o **polarplot** cuja entrada básica tem a seguinte sintaxe:

```
polarplot(expr(theta), theta=a..b, ops);
```

onde,

$\text{expr}(\theta)$ – é uma expressão qualquer envolvendo a variável θ (no Maple a variável θ pode ser obtida digitando-se “theta”);

$\theta = a..b$ – variação de θ (é usual neste caso indicar a variação em radianos);

ops – opções gráficas (ver item 1.1.1).

Exemplos:

```
>polarplot(1, theta=0..2*Pi);
>polarplot(theta, theta=0..20*Pi, scaling=constrained,
axes=framed);
>polarplot(4*(1-sin(theta)), theta=0..2*Pi);
>polarplot(8*sin(5*theta), theta=0..2*Pi);
>polarplot(-16*sin(2*theta), theta=0..2*Pi);
```

4.1.4 Múltiplos Gráficos em 2D

É possível no Maple traçar diversos gráficos de funções de uma só vez, ou seja, em uma única tela. Para isso, basta usar o comando **plot** colocando, no espaço reservado para a função, uma lista de funções, ou seja,

```
plot([f1, f2, ..., fn], h, v, ops)
```

Exemplos:

```
>plot([x, x^2, x^3, x^4, x^5], x=-10..10, y=-10..10);
```

Para podermos identificar qual função corresponde ao seu gráfico, podemos utilizar as opções de cores, ordenadas de acordo com a ordem que as funções estão na lista dentro do comando **plot**,

```
>plot([x, x^2, x^3, x^4, x^5], x=-10..10, y=-10..10, color=[black, red, blue, yellow, green]);
```

```
>plot([x, 2*x, 3*x, 4*x, 5*x], x=-5..5, y=-5..5);
```

```
>plot([sqrt(x), -sqrt(x), sqrt(-x), -sqrt(-x)], x=-5..5);
```

 (na visualização do gráfico parece estar faltando um pedaço, tente arrumar!!).

Também é possível utilizando a mesma sintaxe acima, “misturar” gráficos de funções com gráficos de curvas dadas pelas suas equações paramétricas.

Exemplos:

```
>plot([x^2, [t^2, t^3, t=-5..5]], x=-5..5, y=-5..5);
```

```
>plot([x, -x, [sin(x), cos(x), x=0..2*Pi]], x=-2..2, y=-2..2);
```

```
>plot([sin(x), 3*cos(x), x=0..2*Pi], [3*sin(x), cos(x), x=0..2*Pi], [sin(x)-cos(x), 3*cos(x), x=0..2*Pi], [3*sin(x), cos(x)+sin(x), x=0..2*Pi]], x=-4..4);
```

Há ainda uma outra forma de traçar diversos gráficos, incluindo-se aí os gráficos em coordenadas polares. Nesse caso temos que nomear cada gráfico separadamente e, em seguida, dar o comando

```
plots[display](["nomes dos gráficos"])
```

para o Maple traçar todos eles.

Exemplos:

```
>g1:=plot(x^2, x=-5..5, y=-5..5):
```

```
>g2:=plot([sin(x), 3*cos(x), x=0..2*Pi], x=-5..5, y=-5..5):
```

```
>g3:=polarplot(t, t=0..4*Pi, numpoints=200):
```

```
>plots[display]([g1, g2, g3]);
```

Obs.: 1) Ao atribuir os nomes dos gráficos como g1, g2 e g3 é importante finalizá-los com dois pontos ao invés de ponto e vírgula, pois neste último caso apareceriam na tela todos os pontos calculados pelo Maple para que este possa traçar o gráfico.

2) Embora tenhamos colocado, nos gráficos g1 e g2 acima, as mesmas variações pra x e pra y, cabe observar que o Maple utiliza sempre a maior das variações colocadas nos gráficos. Por exemplo, se tivéssemos colocado

```
>g1:=plot(x^2, x=-5..5, y=-2..2):
```

```
>g2:=plot([sin(x), 3*cos(x), x=0..2*Pi], x=-1..1, y=-5..5):
```

```
>g3:=polarplot(t, t=0..4*Pi, numpoints=200):
```

```
>plots[display]([g1, g2, g3]);
```

O maple exibiria o eixo x e o eixo y variando de -5 a 5.

4.1.5 Animação de Gráficos em Duas Dimensões

O uso de gráficos é útil em diversas situações dentro da matemática. No entanto, há outras situações em que deseja-se saber o que ocorre com um determinado gráfico quando alteramos certos parâmetros utilizados na sua construção. Por exemplo consideremos a função $f(x) = \sin(tx)$, onde t é uma constante real qualquer, e suponha que queiramos saber o que ocorre com o gráfico de f quando o parâmetro t sofre uma variação. Podemos fazer esta verificação através do comando **animate** o qual pode ser utilizado com a seguinte sintaxe,

plots[animate] (f(x,t), v, t)

onde,


$f(x,t)$ – expressão na variável x contendo também um parâmetro t ;


v – variação de x ;

t – variação de t .

Exemplo:

```
>plots[animate] (sin(t*x), x=-7..7, t=1..5);
```

Clique na figura (aparecerá uma moldura em torno do gráfico) e note o aparecimento dos botões  na barra de ferramentas. Para iniciar a

animação clique no botão . Os outros botões servem para visualizar a animação quadro a quadro, mudar a orientação da animação, aumentar ou diminuir a velocidade e colocar a animação no modo de parada automática ou no modo contínuo.

Quando você faz uma animação no Maple ele usualmente cria 16 molduras de gráficos (frames) e os coloca em sequência dando a impressão de filme. Podemos aumentar a quantidade de frames a fim de aumentar a precisão da animação. Basta colocar logo após a variação de t a opção **frames=n**, onde n indica a quantidade de frames que serão visualizados.

Exemplos:

```
>plots[animate] (sin(t*x), x=-7..7, t=1..5, frames=50, numpoints=200);
```

```
>plots[animate] (t*x, x=-2..2, t=1..15, view=[-2..2, -2..2]);
```

```
>plots[animate] ([t*sin(x), t*cos(x)], x=-4..4, t=1..4, frames=100, scaling=constrained);
```

```
>plots[animate] (t*theta, theta=0..8*Pi, t=1..4, coords=polar, numpoints=200);
```

```
>plots[animate] (8*sin(t*theta), theta=0..8*Pi, t=-1..4, coords=polar, numpoints=500, frames=50);
```

Nos dois últimos exemplos colocamos a opção **coords=polar** para que a animação seja feita em coordenadas polares.

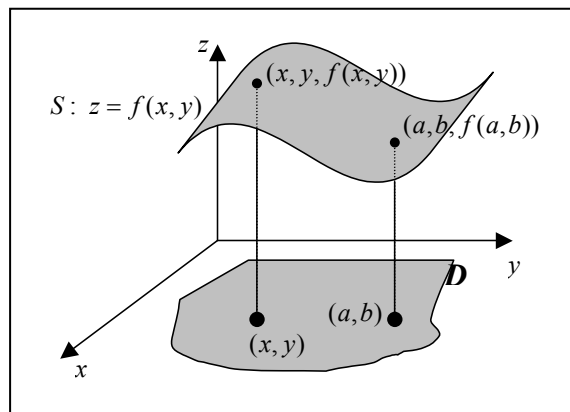
4.2. Gráficos em três dimensões

4.2.1 Gráficos de funções

Antes de ver as opções gráficas de funções em três dimensões, vamos recordar a noção de função de duas variáveis a valores reais.

Seja D um conjunto de pares ordenados de números reais, ou seja, $D \subset \mathbb{R}^2$. Uma função $f : D \rightarrow \mathbb{R}$, que a cada par (x, y) de D associa um único número real, denotado por $f(x, y)$, é denominada **função de duas variáveis**.

O **gráfico** de f é, por definição, o gráfico da equação $z = f(x, y)$ num sistema retangular tridimensional, sendo, assim, em geral uma superfície S de algum tipo. Representando D por uma região do plano xy , consideremos um ponto qualquer $(x, y) \in D$, os valores funcionais $f(x, y)$ são as distâncias orientadas do plano xy a S .



Para traçar o gráfico de uma função $f(x, y)$ em três dimensões utilizaremos o comando **plot3d**, o qual possui a mesma sintaxe do comando para gráficos em duas dimensões,

plot3d(f(x, y), h, v, ops)

onde,

$f(x, y)$ – expressão ou nome da função de duas variáveis;

h = intervalo no eixo das abscissas no qual o gráfico será visualizado;

v = intervalo no eixo das ordenadas no qual o gráfico será visualizado;

ops = opções de visualização e formatação.

Ao contrário do caso bidimensional, a variação v no comando plot é obrigatória, além disso, as variações h e v , caso sejam numéricos, formarão um retângulo no plano xy que será encarado pelo Maple como a região D da definição de gráfico de funções de duas variáveis.

Exemplo:

```
> plot3d(9-x^2-y^2, x=-10..10, y=-10..10);
```

Clicando sobre o gráfico e segurando o botão do mouse pressionado você poderá rotacioná-lo de várias formas possibilitando uma melhor análise e visualização do gráfico. Além disso, ao clicar sobre o gráfico note o aparecimento dos seguintes botões



os quais controlam determinadas propriedades gráficas.

Em relação às opções de visualização **ops** algumas são diferentes em relação ao caso bidimensional. Podemos destacar:

axes=" “ – muda a visualização dos eixos coordenados. As opções deverão ser colocadas como: *boxed*, *normal*, *framed* ou *none*. Usualmente o Maple usa *none*, ou seja, não são mostrados os eixos coordenados no gráfico;

grid=[m,n] – especifica as dimensões dos retângulos sob os quais os pontos serão calculados, **m** e **n** devem ser números naturais maiores do que 1. Se esta opção não for especificada o Maple usualmente utiliza uma quantidade satisfatória de pontos;

gridstyle=" “ – especifica se os “grids” serão retangulares ou triangulares. As opções deverão ser colocadas como *triangular* ou *rectangular*;

labels=[x,y,z] – especifica o nome dos eixos coordenados. No lugar de x, y e z, podem ser colocados quaisquer nomes. Para evitar erros, coloque os nomes entre aspas para que o Maple possa interpretá-los da forma com que são escritos;

linestyle=n – especifica o tipo de linha que será utilizado no gráfico. Se $n = 1$ ou $n = 0$ o tipo de linha será o usual, n deve ser maior ou igual a zero;

tickmarks = [l, m, n] - especifica uma certa quantidade de pontos (não menor do que l , m , e n) sobre os eixos coordenados;

title=" “ – esta opção define um título para o gráfico. Procure colocar o título entre aspas;

view=[a..b, c..d, e..f] – esta opção possibilita a visualização de determinadas partes do gráfico especificando-se o intervalo de variação de x (**a..b**), de y (**c..d**) e z (**e..f**);

Há ainda outras opções, mais estas não vem ao caso aqui.

Exemplos:

```
>plot3d(5-x^2-y^2,x=-4..4,y=-4..4);
>plot3d(5-x^2-y^2,x=-4..4,y=-4..4, axes=framed,
labels=[x,y,temperatura], title="Gráfico 1", view=[-4..4,-
4..4,0..6]);
>plot3d(4*(sin(x/2)+sin(y/4))+3*sin(x/2), x=0..20,y=0..70);
>plot3d(4*(sin(x/2)+sin(y/4))+3*sin(x/2),
x=0..20,y=0..70,axes=boxed, grid=[50,50],
gridstyle=triangular, linestyle=7);
>plot(x^2-y^2,x=-3..3,y=-3..3, title="A sela de cavalo");
```

4.2.2 Gráficos de superfícies no espaço

Procedendo de forma análoga ao que foi feito no parágrafo 1.1.2 podemos definir a noção de superfícies no espaço, as quais generalizam àquelas obtidas por meio da equação $z = f(x, y)$, ou seja, podemos obter a noção de superfície parametrizada.

Para traçarmos o gráfico de uma superfície na forma paramétrica através do Maple, utilizamos a seguinte sintaxe:


```
plot3d([f(t,u),g(t,u),h(t,u)],t=a..b, u=c..d, ops)
```

onde,

f, g, h – são expressões que dependem dos parâmetros t e u ;

$t=a..b$ – intervalo de variação de t ;

$u=c..d$ – intervalo de variação de u ;

ops – opções gráficas citadas acima.

Exemplos:

```
> plot3d([x^3,y*x,5*y-x],x=-2..2,y=-2..2);
> plot3d([sin(u),cos(u)*sin(t),sin(t)], t=-Pi..Pi,u=-
Pi..Pi);
> plot3d([t*sin(t)*cos(u),t*cos(t)*cos(u),t*sin(u)],t=0..2*P
i, u=0..Pi);
> plot3d([t*sin(t)*cos(u),t*cos(t)*cos(u),t*sin(u)],t=0..2*P
i, u=0..Pi,grid=[50,40], view=[-5 ..5,-6..6,1..3]);
```

4.2.3 Múltiplos Gráficos em 3D

No caso tridimensional também é possível traçar vários gráficos de uma só vez. Só que neste caso usamos chaves ao invés de colchetes para a lista das expressões.

Exemplos:

```
> plot3d({x^2+(y^2),x+y},x=-5..5,y=-5..5);
> plot3d({x^2+1/(y^2),cos(x+y)},x=-5..5,y=-5..5,view=[-
5..5,-5..5,-5..20],grid=[50,50]); (note neste exemplo a necessidade da
opção "view=[-5..5,-5..5,-5..20]").
```

Também é possível traçar vários gráficos na forma parametrizada, utilizando-se a sintaxe acima combinada com a sintaxe para gráficos na forma paramétrica. Por exemplo,

```
> c1:= [cos(x)-2*cos(0.4*y),sin(x)-2*sin(0.4*y),y]:
> c2:= [cos(x)+2*cos(0.4*y),sin(x)+2*sin(0.4*y),y]:
> c3:= [cos(x)+2*sin(0.4*y),sin(x)-2*cos(0.4*y),y]:
> c4:= [cos(x)-2*sin(0.4*y),sin(x)+2*cos(0.4*y),y]:
```

primeiramente definimos as equações paramétricas acima, onde cada uma delas é um gráfico diferente. O que fizemos acima é apenas dar um nome para as equações para facilitar na hora de colocar dentro do comando plot3d. Com isso, colocamos

```
> plot3d({c1,c2,c3,c4},x=0..2*Pi,y=0..10,grid=[25,15]);
```

Uma curva espacial(não plana) pode ser traçada utilizando o comando **spacecurve**. O comando **spacecurve** não é carregado pelo Maple automaticamente, ele faz parte de um pacote específico de comandos (packages) que devem ser carregados manualmente quando desejamos utilizá-los e, para isso, digitamos "**with(plots);**" em qualquer linha, onde **plots** é o nome do package da qual o comando **spacecurve** faz parte.

```
> with(plots);
```

```
>c1:=[t*cos(2*Pi*t),t*sin(2*Pi*t),2+t]:
>c2:=[ 2+t, t*cos(2*Pi*t), t*sin(2*Pi*t)]:
>c3:=[ t*cos(2*Pi*t), 2+t, t*sin(2*Pi*t)]:
>spacecurve({c1,c2,c3},x=-8..12,y=-10..12,axes=boxed,
numpoints=400);
```

Obs.: atribuímos os nomes c1, c2 e c3 à cada uma das curvas presentes no gráfico apenas por conveniência, ou então, para possibilitar melhor visualização e evitar erros na digitação das expressões.

Uma outra forma de se obter vários gráficos de uma vez é utilizar o comando **display**. Da mesma forma este também é um comando que consta no package **plots**, no entanto, uma vez que já o carregamos para usar o comando “spacecurve” não precisamos fazê-lo novamente, pois o mesmo fica na memória do computador até que a seção do Maple em que se está trabalhando seja finalizada. Em seguida nomeamos cada um dos gráficos que queremos obter e executamos o comando **display**. Vejamos todo esse procedimento no exemplo,

```
>g1:= plot3d(x^2-y^2,x=-5..5,y=-5..5):
>g2:=plot3d([sin(x),cos(x),y],x=0..2*Pi,y=-20..20):
>display([g1,g2]);
```

Obs.: Ao atribuir os nomes dos gráficos como g1 e g2 é importante finalizá-los com dois pontos ao invés de ponto e vírgula, pois neste último caso apareceriam na tela todos os pontos calculados pelo Maple para que este possa traçar o gráfico.

5. Tópicos de Cálculo

5.1. Limites

5.1.1 Limites de funções de uma variável

Um dos tópicos mais importantes do Cálculo Diferencial é a noção de limites. Conhecer a definição, propriedades, teoremas e aplicações associadas a esse conceito é fundamental, principalmente para os estudantes do curso de Matemática. Embora existam vários métodos para o cálculo de limites, pode-se utilizar o Maple como ferramenta para se efetuar esses cálculos de limites.

Começamos com o caso de funções reais de uma variável. A sintaxe básica para o cálculo de limites é dada por:

limit(f(x),x=a)

onde,

f(x) – função qualquer de uma variável, ou ainda uma expressão qualquer;

x=a – faz o papel do $x \rightarrow a$ na simbologia usual de limites utilizada na literatura, ou seja, a deve ser o valor para o qual x se torna cada vez mais próximo.

Observação: no caso em que quisermos trabalhar com limites no infinito colocamos **x=infinity** ou **x=-infinity** (conforme o caso) para simbolizar $x \rightarrow \infty$ ou $x \rightarrow -\infty$ respectivamente.

Quando executamos o comando acima o Maple automaticamente retorna o resultado (se o cálculo é possível). Se o comando **limit** for colocado em letra maiúscula, ou seja, **Limit** então o Maple retornará a expressão simbólica de limites utilizada usualmente sem calculá-la.

Exemplos:

```
>Limit(x+1,x=1);
>limit(x+1,x=1);
>Limit((x^2-1)/(x-1),x=1);
>limit((x^2-1)/(x-1),x=1);
>limit((x^2+x)/(x+3),x=2);
>Limit((x^3-5*x^2+8*x+4)/(x^4-5*x-6),x=2);      limit((x^3-
5*x^2+8*x+4)/(x^4-5*x-6),x=2);
```

Note que neste último exemplo a resposta foi *undefined* significando que neste caso o limite não existe (pode-se visualizar isso graficamente).

```
>f:= x -> (x^(1/3)-2^(1/3))/(x-2);
>Limit(f(x),x=2);      limit(f(x),x=2);
>g:= x -> (x^(1/n)-a^(1/n))/(x-a);
>Limit(g(x),x=a);      limit(g(x),x=a);
```

Nesse último exemplo, pode-se perceber que em determinados casos o Maple também calcula o limite mesmo se a função contiver certos parâmetros indefinidos.

5.1.2 Limites laterais

Quando digitamos o comando **limit(f(x),x=a)** no Maple ele automaticamente entende este limite como sendo o limite bilateral, ou seja, quando x tende a a pela direita e esquerda. No entanto, pode-se calcular os limites laterais, para isso usamos uma opção a mais dentro do comando de limite já descrito, a saber **limit(f(x),x=a,left)** ou **limit(f(x),x=a,right)** onde left e right são respectivamente esquerda e direita.

Exemplos:

```
>f:= x-> piecewise(x<1,(x^2-1)/(x-1),3);
>limit(f(x),x=1); (note que a resposta a esse limite é undefined, no entanto,
podemos calcular os limites laterais)
>limit(f(x),x=1,left);
>limit(f(x),x=1,right); (note a simbologia usual quando colocamos Limit)
>limit(1/x,x=0,right);
>limit(1/x,x=0,left);
>g:=x->piecewise(x<>-1,(x^2+x)/(x+1),2);
>limit(g(x),x=-1,left);      limit(g(x),x=-1,right);
```

5.1.3 Limites de funções de mais de uma variável

Ao se tratar de funções de duas variáveis ou mais, podemos definir de forma análoga ao caso de uma variável a noção de limites. No caso de funções de duas variáveis, por exemplo, consideraremos o limite de $f(x,y)$ quando (x,y) tende a um determinado ponto (a,b) o que corresponde a dizer que $x \rightarrow a$ e $y \rightarrow b$. A sintaxe para se calcular limites deste tipo no caso geral é dada por,

limit(f(x1, x2, ..., xn), {x1=a1, x2=a2, ..., xn=an})

onde,

$f(x_1, x_2, \dots, x_n)$ – função de n variáveis;

$\{x_1=a_1, x_2=a_2, \dots, x_n=a_n\}$ – conjunto contendo os valores para o qual cada variável se aproxima.

As mesmas colocações feitas em 2.1.1 valem também neste caso.

Exemplos:

```
>limit((x^2-y^2)/(x^2+y^2), {x=0, y=1});
>limit((x^2-y^2)/(x^2+y^2), {x=0, y=0});
>limit((x^3-x^2*y+x*y^2-y^3)/(x^2+y^2), {x=0, y=0});
>limit((x*y-2*x-y+2)/(x^2+y^2-2*x-4*y+5), {x=1, y=2});
```

Neste último exemplo, o Maple não consegue calcular o limite proposto e com isso ele retorna a própria expressão digitada.

5.2. Derivadas

5.2.1 Derivadas de funções de uma variável

Quando estamos lidando com funções de uma variável podemos calcular suas derivadas através do Maple. A sintaxe básica para o cálculo da derivada de uma função f é a seguinte:

diff(f(x), x)

onde,

$f(x)$ – função ou expressão que se deseja obter a derivada;

x – variável em relação a qual se deseja obter a derivada.

De forma análoga ao que ocorre com o comando de limite se colocarmos o comando diff iniciando com letra maiúscula teremos a derivada na forma simbólica, ou seja, o cálculo não é efetuado.

Exemplos:

```
>Diff(x^5-2*x^3+3*x, x);    diff(x^5-2*x^3+3*x, x);
>Diff(1/x+1/x^2+1/x^3+1/x^4, x);
diff(1/x+1/x^2+1/x^3+1/x^4, x);
>Diff(x^(sin(3*x)), x);    diff((x^(sin(3*x))), x);
>Diff((1+1/x)^x, x);    diff((1+1/x)^x, x);
>Diff(tan(x)-cot(x), x);    diff(tan(x)-cot(x), x);
>Diff(exp(x^2)+ln(1/x)+sqrt(2*x-x^2), x);
diff(exp(x^2)+ln(1/x)+sqrt(2*x-x^2), x);
>Diff((a+cos(x+b)-ln(a*x))^2, x);    diff((a+cos(x+b)-
ln(a*x))^2, x);
```

Neste último exemplo, podemos perceber a necessidade de se indicar a variável de derivação, pois nesse caso a e b são considerados como constantes, e então o cálculo é efetuado perfeitamente pelo Maple.

5.2.2 Derivadas de ordem superior

Seja f uma função derivável em um certo conjunto $I \subset \mathbb{R}$, então podemos definir uma nova função $f': I \rightarrow \mathbb{R}$ dada por $x \mapsto f'(x)$, denominada função derivada, ou simplesmente derivada de f . Com isso, a derivada de f' é denominada derivada de 2ª ordem de f e indicada por f'' ou $f^{(2)}$. De modo análogo, define-se as derivadas de ordem superior a 2 de f .

No Maple se quisermos calcular as derivadas $f'', f''', \dots, f^{(n)}$, podemos utilizar a seguinte sintaxe:

diff(f(x), x, x, ..., x)
ou ainda
diff(f(x), x\$n)

onde **\$n** significa que a variável x deverá ser repetida n vezes.

Exemplos:

```
>Diff(3*x^3,x,x);    diff(3*x^3,x,x);
>diff(1/x,x$4);
>diff(sin(x),x,x,x);
>diff(cos(x),x$3);
>diff(sqrt(x),x$4);
```

É interessante observar que dependendo da função, a n -ésima derivada pode ser obtida de forma dedutiva. Essa dedução pode ser feita com o auxílio do próprio Maple, vejamos alguns exemplos:

- suponha que queiramos calcular $f^{(n)}$ onde $f(x) = \frac{1}{x}$. Para deduzirmos a fórmula podemos colocar algumas derivadas em sequência para que possamos deduzir a lei de formação, para isso podemos utilizar um comando já visto, a saber o **seq**:

```
> seq(diff(1/x,x$i), i=1..5);
```

$$-\frac{1}{x^2}, 2\frac{1}{x^3}, -6\frac{1}{x^4}, 24\frac{1}{x^5}, -120\frac{1}{x^6}$$

visualizando desta forma, ou seja, as derivadas todas em sequência não é difícil ver que:

$$f^{(n)} = (-1)^n n! \frac{1}{x^{n+1}}$$

- suponha que queiramos calcular $f^{(n)}$ onde $f(x) = \sqrt{x}$. Para deduzirmos a fórmula podemos colocar algumas derivadas em sequência juntamente com a ordem correspondente, de forma análoga ao caso anterior utilizando o comando **seq**:

```
>seq([diff(sqrt(x),x$i),n=i], i=1..10);
```

$$\left[\frac{1}{2} \frac{1}{\sqrt{x}}, n=1\right], \left[-\frac{1}{4} \frac{1}{x^{3/2}}, n=2\right], \left[\frac{3}{8} \frac{1}{x^{5/2}}, n=3\right], \left[-\frac{15}{16} \frac{1}{x^{7/2}}, n=4\right], \left[\frac{105}{32} \frac{1}{x^{9/2}}, n=5\right],$$

$$\left[-\frac{945}{64} \frac{1}{x^{11/2}}, n=6\right], \left[\frac{10395}{128} \frac{1}{x^{13/2}}, n=7\right]$$

visualizando dessa forma não é difícil deduzir a lei de formação geral da derivada n -ésima de $f(x) = \sqrt{x}$:

$$f^{(n)} = (-1)^{n-1} \frac{\prod_{i=1}^{n-1} (2i-1)}{2^n} \frac{1}{x^{\frac{2n-1}{2}}}$$

Outra situação onde se pode usar os recursos do Maple já vistos é em relação aos gráficos, ou seja, podemos traçar os gráficos de uma função e suas derivadas simultaneamente. Por exemplo,

```
>f:= x-> 1/20*x^5+x^3-2*x^2-3*x+4;
```

f e suas derivadas (até ordem 5) podem ser colocadas em sequência:

```
>f(x), seq(diff(f(x), x$i), i=1..5);
```

em seguida podemos traçar os gráficos:

```
>plot([f(x), seq(diff(f(x), x$i), i=1..5)], x=-10..10, y=-20..20, color=[red, blue, black, green, yellow, pink], numpoints=100);
```

as opções de cores no comando plot acima são colocadas para se identificar qual o gráfico que corresponde a cada função e a opção numpoints, para aumentar a resolução.

5.2.3 Derivadas de funções de mais de uma variável

Seja f uma função de duas ou mais variáveis, podemos então calcular as derivadas parciais de f em relação às suas variáveis. A sintaxe para o cálculo das derivadas parciais de f é semelhante ao usado para funções de uma variável:

```
diff(f(x1, x2, ..., xn), var)
```

onde,

$f(x_1, x_2, \dots, x_n)$ – função ou expressão em n variáveis;

var – variável(is) para (as) qual(is) se deseja calcular a(s) derivada(s) parcial(is).

Exemplo:

```
>f:= (x,y) -> (x^2+y^2)/(x+y);
```

```
>diff(f(x,y), x); (derivada parcial de f com relação a x)
```

```
>diff(f(x,y), y); (derivada parcial de f com relação a y)
```

```
>diff(f(x,y), x,y); (derivada parcial de f primeiro com relação a x e depois com relação a y)
```

```
>diff(f(x,y), y,x); (derivada parcial de f primeiro com relação a y e depois com relação a x)
```

Como observado anteriormente se colocarmos o comando em letra maiúscula obteremos a notação simbólica da derivada:

```
>Diff((x^2-t^2)/(1+sin(3*y)), x,x,y);
```

```
diff((x^2-
```

```
t^2)/(1+sin(3*y)), x,x,y);
```

```
>Diff(ln(sqrt((t+v)/(t-v))),t,t,v,v);
diff(ln(sqrt((t+v)/(t-v))),t,t,v,v);
```

5.3. Integrais

Dada uma função f de uma de uma variável podemos utilizar o Maple para o cálculo da integral de f . Os comandos para o cálculo da integral indefinida e definida são dados respectivamente por:

```
int(f(x),x) e int(f(x),x=a..b)
```

onde,

$f(x)$ – função ou expressão a ser integrada;

x – variável de integração;

$x=a..b$ – limites de integração(no caso de integrais definidas).

Nos comando acima, se o Maple encontra uma solução para a integral ele automaticamente retorna a solução. Caso não seja possível encontrar a solução o Maple retorna a própria expressão digitada na forma simbólica. Aliás, para se obter a integral na forma simbólica usual da matemática basta colocar os comandos com a letra inicial em maiúscula(tanto no caso de integral definida quanto indefinida).

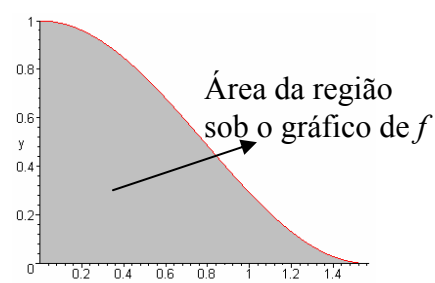
Exemplos:

```
>Int(x^n,x)= int(x^n,x);
>Int(exp(n*x),x)=int(exp(n*x),x);
>Int(1/x,x)=int(1/x,x);
>Int(sqrt(a^2+u^2),u)= int(sqrt(a^2+u^2),u);
>Int(x^2/((a+b*x)^2),x)= int(x^2/((a+b*x)^2),x);
>Int(u^6*cos(u),u)= int(u^6*cos(u),u);
```

Podemos perceber que a presença de constantes no integrando é perfeitamente possível, daí a necessidade de se indicar a variável de integração. Nos exemplos acima todas as integrais são indefinidas com isso, pode-se também notar que o Maple não exibe a constante de integração. Vejamos alguns exemplos de integrais definidas:

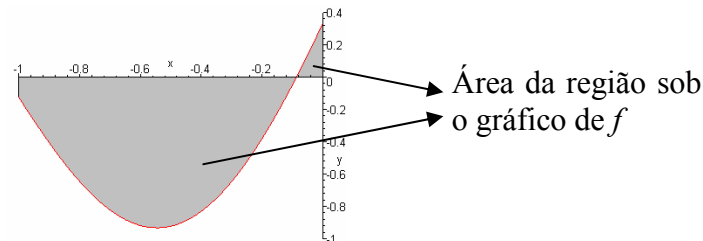
```
>Int((x^2+2*x)/(x^4+3*x^2+x),x=-0.2..3)=
int((x^2+2*x)/(x^4+3*x^2+x),x=-0.2..3);
>Int(sqrt(x),x=0..4)= int(sqrt(x),x=0..4);
>Int((1+t^2)/t^4)= int((1+t^2)/t^4);
>Int(sin(x)*cos(x)^2,x=0..Pi/3)=
int(sin(x)*cos(x)^2,x=0..Pi/3);
>Int(x^3/(16+x^4),x=-2..2)= int(x^3/(16+x^4),x=-2..2);
```

Sabe-se que a área sob o gráfico de uma função f contínua em um intervalo $[a, b]$ é definida como $\int_a^b |f(x)| dx$, podemos combinar o cálculo da integral com o respectivo esboço do gráfico da função integrada no intervalo $[a, b]$ (mais a título de visualização). Vejamos alguns exemplos:



```
>Int(1/2+1/2*cos(2*x),x=0..Pi/2)=
int(1/2+1/2*cos(2*x),x=0..Pi/2);
>plot(1/2+1/2*cos(2*x),x=0..Pi/2);

>Int(1/3*exp(3*x)+sin(3*x),x=-1..0)=
int(1/3*exp(3*x)+sin(3*x),x=-1..0);
>plot(1/3*exp(3*x)+sin(3*x),x=-1..0);
```



Obs.: o preenchimento na cor cinza nos gráficos acima, não foi gerado pelo Maple, no entanto a região é perfeitamente deduzida quando traçamos o gráfico no Maple.

Como foi dito acima quando o Maple não encontra a forma exata da solução de uma integral ele retorna a própria integral.

```
>Int(1/x,x=-1..1)= int(1/x,x=-1..1);
>Int(x^ln(x),x):=int(x^ln(x),x);
```

5.3.1 Integrais Múltiplas

O mesmo comando utilizado para o cálculo de integrais de funções de uma variável pode ser utilizado para o cálculo de integrais múltiplas. O procedimento consiste em utilizar o comando `int` repetidas vezes mudando-se a variável de integração. Por exemplo, suponhamos que queiramos calcular a integral dupla de $f(x,y) = \frac{1}{x^2} + \frac{1}{y^2}$

em relação a x e depois em relação a y , então podemos fazer:

```
>Int(Int(1/x^2+1/y^2,x),y)= int(int(1/x^2+1/y^2,x),y);
```

Vamos conferir o resultado:

```
>Diff(-y/x-x/y,x,y)= diff(-y/x-x/y,x,y);
```

```
>Int(Int(x*cos(y)-y*cos(x),y),x)= int(int(x*cos(y)-
y*cos(x),y),x);
```

```
>Int(Int(12*x*y^2-8*x^3,x=0..2),y=-1..1)= int(int(12*x*y^2-
8*x^3,x=0..2),y=-1..1);
```

```
>Int(Int(Int(ln(x*y*z),x),y),z)=
int(int(int(ln(x*y*z),x),y),z);
```

No caso da integral múltipla definida, as regiões de integração podem ser mais gerais ou seja, não necessariamente precisam ser intervalos numéricos. Por exemplo:

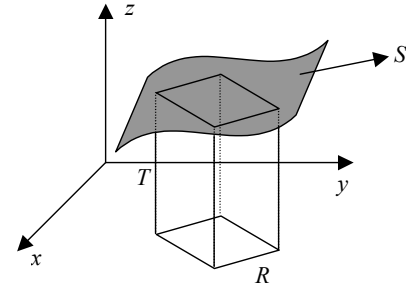
```
>Int(Int(x^3+4*y,y=x^2..2*x),x=0..2)=
int(int(x^3+4*y,y=x^2..2*x),x=0..2);
```



```
>Int(Int(2*y*cos(x),x=Pi/6..y^2),y=1..3)=
int(int(2*y*cos(x),x=Pi/6..y^2),y=1..3);
>Int(Int(x^2+4*y^2,y=x^2..-x^2+8),x=0..2)=
int(int(x^2+4*y^2,y=x^2..-x^2+8),x=0..2);
```

Há uma interpretação geométrica para integrais duplas, quando se tem uma função $f(x, y) \geq 0$ e contínua em toda uma região R do plano. Denotando por S o gráfico de f e por T o sólido abaixo de S e acima de R , teremos que o volume de T é dado por:

$$V = \iint_R f(x, y) \, dA$$



Podemos utilizar os recursos do Maple para podermos visualizar graficamente o cálculo da integral dupla. Como um exemplo, considere o seguinte problema:

“Calcular o volume do sólido que está sob o gráfico do parabolóide $z = x^2 + 4y^2$ e acima da região dada por $z = 0$, $x = 0$, $y = x^2$ e $y = -x^2 + 8$.”

Sol.: a região de integração, no plano $z = 0$, é dada por $x = 0$, $y = x^2$ e $y = -x^2 + 8$. No Maple essa região plana pode ser visualizada fazendo:

```
>plot([x^2, -x^2+8], x=0..3):
```

se quisermos calcular o ponto de intersecção das duas curvas fazemos,

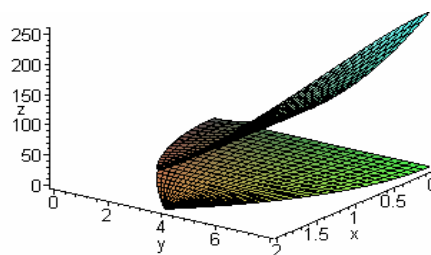
```
>solve(x^2=-x^2+8);
```

o que nos leva a concluir que $x = 2$. Portanto na integração sabemos que o x varia de 0 até 2 enquanto que y varia de x^2 a $-x^2 + 8$. Agora para podermos visualizar essa região de integração em três dimensões podemos parametrizar essa região. Para isso fazemos

$$\begin{cases} x = x \\ y = t * x^2 + (1-t) * (-x^2 + 8) \quad 0 \leq t \leq 1 \\ z = 0 \end{cases}$$

Dessa forma o sólido sob o gráfico de $z = x^2 + 4y^2$ e acima da região dada por $z = 0$, $x = 0$, $y = x^2$ e $y = -x^2 + 8$, pode ser visualizado fazendo-se:

```
>plot3d([ [x, t*x^2+(1-t)*(-x^2+8), 0], [ x, t*x^2+(1-t)*(-x^2+8), x^2 + 4*(t*x^2+(1-t)*(-x^2+8))^2 ] ], x=0..2, t=0..1);
```



6. Tópicos de Programação

6.1. Fundamentos

O Maple possui uma poderosa capacidade de programação matemática. Um procedimento (ou programa) no Maple é essencialmente um conjunto de comandos (do próprio Maple ou comandos usuais de programação) organizados em uma certa estrutura lógica que são executados pelo Maple, dependendo das necessidades do usuário. Os procedimentos implementados no Maple podem fazer uso de todos os comandos internos (incluindo os dos packages), ou seja, você pode construir seus procedimentos utilizando todas as estruturas e funções do Maple.

Todo procedimento no Maple tem basicamente a seguinte configuração:

```
nome:= proc(...)  
    local x1, x2,...,xn;  
    global x1, x2,...,xn;  
    .....  
    .....  
end;
```

ou seja, todo procedimento deve iniciar com o comando **proc(...)** e finalizar com o comando **end;** (ou **:**). As reticências nos parênteses representam as entradas de dados os quais denominaremos de **parâmetros** (não necessariamente precisam ser especificados, ou seja, os parênteses podem ficar em branco). Para que se possa utilizar um procedimento é conveniente dar-lhe um nome, e isso é feito utilizando a maneira usual de atribuição de nomes no Maple.

Vejamos um exemplo,

```
>metade:= proc(x) ;  
>evalf(x/2) ;  
>end;
```

Esse Procedimento pega um valor de x e calcula o valor aproximado desse número dividido por 2. Para executar esse procedimento basta utilizá-lo como um comando qualquer do Maple,

```
>metade(1/2) ;  
>metade (10) ;  
>metade(sqrt(3)) ;
```

Outro exemplo:

```
>val_e:=proc()  
>a:=exp(1) ;  
>evalf(a) ;  
>end;
```

O procedimento **val_e** dá uma aproximação, em decimais, do número e ,

```
>val_e() ;
```

Note que nesse procedimento nenhuma entrada de dados é necessária.

Obs.: Dentro de um procedimento, comandos isolados devem terminar com ponto e vírgula.

Quando definimos o procedimento **val_e** acima, note que o Maple exibe um aviso “Warning, `a` is implicitly declared local”. Esse aviso refere-se ao fato de termos usado uma variável dentro do procedimento (no caso a variável *a*). Há dois tipos de variáveis que podem ser definidas no Maple, **variável local** e **variável global**. Basicamente, variáveis locais são aquelas que você define dentro de um procedimento e cujo valor é usado pelo Maple apenas dentro deste procedimento, enquanto que variáveis globais são aquelas que são atribuídas dentro de um procedimento e que passam a ter esse valor atribuído mesmo fora do procedimento. Logo no início de um procedimento podemos definir quais variáveis são locais e/ou quais são globais, basta colocar na linha seguinte ao comando **proc** a opção **local** “nome das variáveis locais” e/ou **global** “nome das variáveis globais”, conforme o modelo geral de um procedimento visto acima. Quando não fazemos essa especificação o Maple decide internamente o tipo das variáveis e exibe um aviso (como ocorreu no procedimento **val_e** acima).

Vejamos um exemplo, que ilustra a diferença entre os tipos de variáveis:

```
>b:=2;
>val_e:=proc()
>local b;
>b:=exp(1);
>evalf(b);
>end;
```

Note que dentro do procedimento *b* foi definido como variável local e lhe foi atribuído o valor **exp(1)**, no entanto ao usarmos novamente o valor *b* fora do procedimento este permanece com o mesmo valor

```
>val_e();
>b;
```

Vejamos agora a diferença se fizermos,

```
>b:=2;
>val_e:=proc()
>global b;
>b:=exp(1);
>evalf(b);
>end;
```

Note que dentro do procedimento *b* foi definido como variável global e lhe foi atribuído o valor **exp(1)**, com isso, ao usarmos novamente o valor *b* fora do procedimento este já está com valor alterado

```
>val_e();
>b;
```

Observação: Todos os procedimentos constantes neste texto são apenas ilustrativos, ou seja, objetivam ilustrar apenas os recursos de programação. Não se tem nenhuma preocupação com relação ao rigor dos procedimentos e em relação à prevenção de erros.

6.2. Estruturas de Programação

O sistema de programação do Maple inclui estruturas usuais de programação, tais como o “loops” e afirmações condicionais que podem ser utilizadas dentro ou fora de um

procedimento. Tais estruturas são fundamentais em qualquer linguagem de programação e permitem uma infinidade de possibilidades em termos de programas que podem ser implementados. Discutiremos a seguir algumas dessas estruturas de programação.

6.2.1 O Comando “for”

A sintaxe geral no Maple para essa estrutura de programação, conhecida como “loop”, é a seguinte:

```
for i from a to b by c do ... od;
```

onde,

- i* – é a variável do loop;
- a* – valor inicial;
- b* – valor final;
- c* – é o salto que a variável sofre a cada passo do loop;
- ... – expressão(ões) a ser(ere)m executada(s) pelo loop.

Na estrutura deste comando algumas das cláusulas em vermelho são opcionais. Se, por exemplo, omitirmos o **from** ou o **by** eles serão automaticamente considerados como 1 pelo Maple. Necessariamente todo loop deve terminar com a cláusula **od**.

Vejam alguns exemplos:

Problema: calcular a soma dos 100 primeiros números naturais.

Vejam como resolver esse problema usando o loop.

```
>a:=0
>for i from 1 to 100 do
>a:=a+i;
>od;
```

Obs.: 1) se tivéssemos feito:

```
>a:=0
>for i from 2 to 100 by 2 do
>a:=a+i;
>od;
```

teríamos como resultado a soma dos 50 primeiros números naturais pares;

2) como dito anteriormente, podemos omitir certas cláusulas do comando,

```
>a:=0
>to 100 do
>a:=a+2;
>od;
```

Na maioria dos casos o comando **for** é muito útil no desenvolvimento de procedimentos. Por exemplo podemos construir um procedimento para resolver o problema acima.

```
>soma:=proc(n)
>local a;
>a:=0
>for i from 1 to n do
>a:=a+i;
>od;
>end;
```

```
>soma(100);
>soma(20);
```

6.2.2 O Comando “if”

A sintaxe geral para essa estrutura de programação no Maple, conhecida como condicional, é a seguinte:

```
if p then q else m fi;
```

onde,

p – expressão condicional;
q e *m* – sequência de comandos.

O comando acima deve necessariamente terminar com a cláusula **fi**. A expressão condicional *p* é qualquer expressão que admita o valor verdadeiro ou falso conhecida como expressão booleana, que contenha operadores de relação (tais como <, <=, >=, =, <>), operações lógicas (and, not, or) e valores (true, false).

Vejamos um problema que pode ser resolvido através de um procedimento utilizando o comando acima.

Problema: implementar um procedimento que calcule o valor absoluto de um número real.

```
>modulo:=proc(x)
> if x>=0 then
>   x;
> else
>   -x;
> fi;
>end;
```

```
>modulo(-3);
>modulo(-3/2);
```

O comando **if** pode ser utilizado em sequência, ou seja, um dentro do outro. Por exemplo, definir a função

$$f(x) = \begin{cases} 0 & \text{se } x \leq 0 \\ x & \text{se } 0 < x \leq 1 \\ 2-x & \text{se } 1 < x \leq 2 \\ 0 & \text{se } x > 2 \end{cases}$$

através de um procedimento.

```
>f:=proc(x)
> if x<=0 then 0 else
>   if x<=1 then x else
>     if x<=2 then 2-x else
>       if x>2 then 0
>       fi;
>     fi;
>   fi;
> fi;
>end;
```

Podemos encurtar o procedimento f acima utilizando uma abreviação dos comandos **else** e **if**, quando esses aparecem seguidamente, a saber o comando **elif**. Isso é útil pois elimina a necessidade de se finalizar cada um dos comandos **if** com o **fi**. Vejamos:

```
>f2:=proc(x)
> if x<=0 then 0
>   elif x<=1 then x
>   elif x<=2 then 2-x
>   else 0
> fi;
>end;
```

Obs.: claramente os procedimentos vistos anteriormente não estão completos, servem apenas para exemplificar o uso dos comandos. Por exemplo, se colocarmos:

```
>modulo(x);
Error, (in modulo) cannot evaluate boolean
```

notamos que o Maple retornou um erro pois se analisarmos o algoritmo vemos que não é possível para o Maple saber se x é maior ou menor que zero. Veremos mais adiante como controlar as entradas de parâmetros dos procedimentos para evitar certos erros.

6.2.3 O Comando “while”

O comando **while** é outra importante estrutura de programação, na verdade é um tipo de loop. A sintaxe para o comando **while** é a seguinte:

while p **do** q **od**;

onde,

- p – expressão condicional;
- q – seqüência de comandos.

O Maple lê a condição e enquanto ela for verdadeira ele executa os comandos. Vejamos um exemplo.

Problema: fazer um procedimento que divida sucessivamente um número inteiro n por 2 (deixando resto zero) e exiba quantas divisões foram possíveis.

Para fazer esse procedimento usaremos os comandos **iquo**(a,b) e **irem**(a,b) que calculam o quociente e o resto, respectivamente, na divisão de a por b .

```
>ndiv2:=proc(n)
>local q, i;
>q:=n; i:=0
>if irem(q,2)<>0 then i else
> while irem(q,2)=0 do
>   q:=iquo(q,2);
>   i:=i+1;
> od;
>fi;
>end;
```

6.3. Controle de Parâmetros

Como já foi mencionado os procedimentos exemplificados até agora são todos numéricos, com isso se ou usuário colocar uma letra ou símbolo na hora de executar o procedimento obterá um erro ou um resultado incorreto. Por exemplo:

```
>ndiv2:=proc(x);
```

0

o que não é a resposta correta pois o valor de x não é conhecido.

Uma forma de contornar esse tipo de problema e avisar ao usuário o tipo correto de entrada que o procedimento aceita, é definir logo no início com que parâmetro o procedimento trabalhará. Para isso, dentro do comando **proc** colocamos a opção “**::**...” logo após o parâmetro, onde no lugar das reticências deve-se colocar a opção desejada. Para uma lista dos tipos disponíveis digite “**?type;**” no prompt do Maple. Por exemplo, poderíamos reescrever o procedimento **ndiv2** da seguinte forma:

```
>ndiv2:=proc(n::integer)
>local q, i;
>q:=n; i:=0
>if irem(q,2)<>0 then i else
>  while irem(q,2)=0 do
>    q:=iquo(q,2);
>    i:=i+1;
>  od;
>fi;
>end;
```

Com isso, note a diferença,

```
>ndiv2:=proc(x);
```

```
Error, ndiv2 expects its 1st argument, n, to be of type integer, but received
x
```

6.4. Outros Exemplos

6.4.1 Procedimentos Recursivos

Em diversas situações, surge a necessidade de se fazer cálculos recursivos, ou seja, procedimentos onde os valores calculados vão sendo reutilizados para o cálculo de novos valores. Como um exemplo, consideremos os números de Fibonacci os quais são definidos por:

$$f_n = f_{n-1} + f_{n-2} \quad n \geq 2, \text{ onde}$$
$$f_0 = 0 \quad \text{e} \quad f_1 = 1$$

Por exemplo, para $n = 4$ temos:

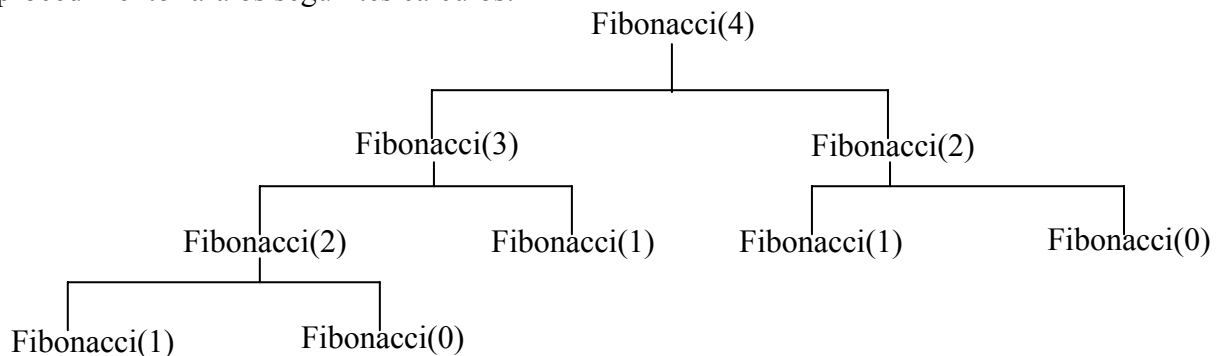
$$f_4 = f_3 + f_2$$
$$f_3 = f_2 + f_1$$
$$f_2 = f_1 + f_0 = 1 + 0 \Rightarrow f_3 = 1 + 1 \Rightarrow$$
$$f_4 = 2 + 1 = 3$$

Note que para o cálculo de f_4 precisamos dos valores de f_1 e f_2 , os quais são obtidos também de modo recursivo conhecendo-se os valores de f_1 e f_0 . Objetivaremos construir um procedimento que calcule f_n para todo natural $n \geq 2$.

```
> Fibonacci := proc (n :: nonnegint)
> if n < 2 then n
> else
>   Fibonacci (n-1) + Fibonacci (n-2)
> fi;
> end;
> Fibonacci (15), Fibonacci (16);
```

Obs.: A forma com que o procedimento Fibonacci foi construído, ilustra o fato de que podemos utilizar dentro de um procedimento o próprio procedimento que estamos construindo.

Note que, quando o valor de n é grande o tempo de cálculo aumenta muito. Vejamos o porquê isso ocorre. Por exemplo, tomamos $n = 4$. Internamente o procedimento fará os seguintes cálculos:



Todos esses cálculos são efetuados pelo Maple de maneira separada, ou seja, **Fibonacci(2)** é calculado 2 vezes, **Fibonacci(1)** é calculado 3 vezes e **Fibonacci(0)** é calculado 2 vezes. Com isso, quando n é grande a quantidade de cálculos se multiplica.

Uma forma de contornar esse problema é fazer com que o Maple memorize todos os cálculos feitos, para isso, temos que usar a opção **remember** nas linhas iniciais do procedimento. Isso evitará repetição de cálculos. Vejamos:

```
> Fibonacci2 := proc (n :: nonnegint)
> option remember;
> if n < 2 then n
> else
>   Fibonacci2 (n-1) + Fibonacci2 (n-2)
> fi;
> end;
> Fibonacci2 (30);
> Fibonacci2 (400);
```

Obs.: Há outras formas de construir o procedimento **Fibonacci2** sem utilizar a opção **remember**, só que com a mesma rapidez.

6.4.2 Procedimento com Listas

Os procedimentos vistos até agora trabalham apenas com números ou fórmulas, no entanto, às vezes aparecem situações na qual temos que trabalhar com uma estrutura de dados, como listas por exemplo. Nesse caso, para o desenvolvimento de procedimentos que têm como entrada certas estruturas de dados, são necessários os principais comandos que se referem à seqüências, listas e conjuntos. Como exemplo, consideremos o seguinte problema: “escrever um procedimento no qual dados os números x_1, x_2, \dots, x_n , ($n > 0$) calcule sua média aritmética simples, dada por:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

Antes de escrevermos o procedimento vejamos os principais elementos que serão necessários. Primeiramente os números x_1, x_2, \dots, x_n podem ser colocados em uma lista, por exemplo, se quisermos calcular a média aritmética de 1.3, 1, 2, 2.5, 5.4, podemos colocar esses dados em uma lista, $X := [1.3, 1, 2, 2.5, 5.4]$. A partir daí para efetuar o cálculo precisaremos:

- do número de elementos da lista, o qual pode se obtido através do comando **nops (X)**
- a soma dos elementos, que no caso de listas pode ser efetuada pelo comando **add (i, i=X)**.

Para a construção do procedimentos temos que sistematizar esses elementos.

```
>media:=proc(x::list)
>local n, i, soma;
>n:= nops(x);
>soma:=add(i, i=x);
>soma/n;
>end;

>media([1.3,1,2,2.5,5.4]);
```

Neste procedimento um erro pode ocorrer caso venhamos a colocar uma lista vazia, **media([])**. Para tornar um procedimento mais expressivo no tratamento de erros podemos utilizar o comando “**ERROR**” para especificar exatamente qual o tipo de erro que ocorreu. Vejamos:

```
>media2:=proc(x::list)
>local i, n, soma;
>n:=nops(x)
>if n=0 then
>ERROR("a lista não possui nenhum elemento");
>soma:=add(i, i=x);
>soma/n;
>end;
```

Note que essa mesma técnica de exibir erros também pode ser usada para substituímos a mensagem de erro que é exibida pelo Maple caso x não seja uma lista, por uma mensagem nossa. Dessa forma não precisaríamos colocar a opção **::list** no comando **proc**.

```
>media2:=proc(x)
```

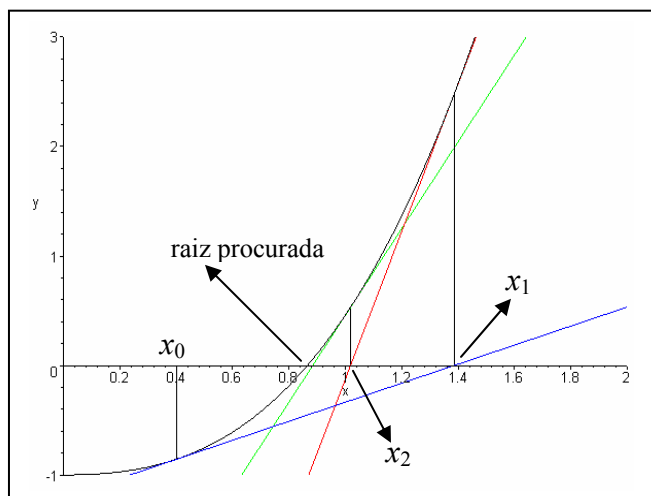
```

>if not type(x,list) then
>ERROR("a entrada deve ser uma lista");
>local i,n,soma;
>n:=nops(x)
>if n=0 then
>ERROR("a lista não possui nenhum elemento");
>soma:=add(i,i=x);
>soma/n;
>end;

```

6.4.3 O Método de Iteração de Newton

O método de Newton é um dos tópicos usuais no estudo do Cálculo Numérico, e tem como objetivo fazer a aproximação da raiz de uma função de uma variável. Sem a preocupação com o rigor matemático podemos descrevê-lo da seguinte forma: primeiramente marcamos um ponto x_0 no eixo x (“chute inicial”) que acreditamos estar próximo da raiz. Em seguida consideramos a reta tangente ao gráfico da função no ponto x_0 e observamos onde a reta intercepta o eixo x (ponto x_1). A partir do ponto x_1 traçamos outra reta tangente ao gráfico e tomamos a interseção com o eixo x (ponto x_2). Continuando dessa forma obteremos uma aproximação para a raiz da função. A equação que permite, dado o ponto x_0 , encontrar os demais pontos é dada por:



$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Podemos então implementar um procedimento no Maple, onde dada uma função f e o ponto x_0 , calcule um certo número de aproximações para a equação $f(x) = 0$.

```

>Newton:=proc(f::{procedure,algebraic},y,n)
>local x0,interac;
>x0:=y;
>interac:=unapply(x-f/diff(f,x),x);
>to n do
> print(x0);
> x0:=evalf(interac(x0)) od;
>end;

>Newton(x-cos(x),0.4,5);

```

Comumente costuma-se definir uma função da maneira usual no Maple e em seguida nos referimos a essa função apenas pelo seu nome. No procedimento acima, no entanto, se fizemos:

```

>f:=x->x-cos(x);

f:= x -> x - cos(x)

>Newton(f,0.4,5);
Error, (in Newton) division by zero

```

Note que um erro ocorre, pois dentro do procedimento o f é lido pelo Maple como uma constante e não uma função. Podemos arrumar o procedimento para evitar esse tipo de erro.

```
>Newton2:=proc(f::{procedure,algebraic},y,n)
>local x0,interac;
>x0:=y;
>if type(f,procedure) then
>  interac:=(x->x)-eval(f)/D(eval(f));
>  else
>  interac:=unapply(x-f/diff(f,x),x);
>fi;
>to n do
>  print(x0);
>  x0:=evalf(interac(x0)) od;
>end;
```

Finalizamos então essa seção colocando que os aspectos acima são iniciais e básicos, aspectos mais gerais e técnicos podem ser consultados nas bibliografias que tratam do assunto. Por outro lado, a exposição acima é suficiente para muitos propósitos e a aplicação ou não destes conceitos dependerá das necessidades de cada um.

7. Bibliografia

HEAL, K. M.,...[et al]. *MAPLE V: Learnig guide*. Springer-Verlag, 1998.

HECK, André. *Introduction to Maple*. New York, Springer-Verlag, 1993.

MONAGAN, M. B., [et al]. *MAPLE V: Programming Guide*. Springer-Verlag, 1998.

SWOKOWSKI, Earl William. *O cálculo com Geometria Analítica*. São Paulo: McGraw-Hill.
Vol 1 e 2.