Aritmética, Álgebra
e Cálculo
com o
Mathematica

Antonio Cândido Faleiros
Departamento de Matemática
Instituto Tecnológico de Aeronáutica - ITA

1997

Antônio Cândido Faleiros

Endereço atual:

Centro de Matemática Computação e Cognição - CMCC
Universidade Federal do ABC - UFABC
Ministério da Educação

e-mail: antonio.faleiros@ufabc.edu.br

Aritmética, Álgebra e Cálculo com o Mathematica

Sugestões serão bem vindas.

Autor:
Faleiros, Antonio Cândido

Título:
Aritmética, Álgebra e Cálculo
com o Mathematica

São José dos Campos, SP
ITA, 1997

Editora Edgard Blücher Ltda

Impresso na Gráfica Palas Athena

ISBN 85-212-0146-X

1. Mathematica 2. Computação Simbólica 3. Tutorial

Dedicatória

Aos meus pais **Paulo** e **Nair**,

que partiram ao encontro da Sabedoria Eterna,
deixando saudades.

Aos meus irmãos **Auxiliadora**, **Immo**,
Fátima e **Paulo**,

amigos de uma infância feliz.

À minha esposa **Graça**,

companheira de uma agradável jornada.

Aos meus filhos

Alexandre,
Adriana,
André e
Júnior,

por me fazerem sentir orgulho de ser pai.

Prefácio

Este livro irá auxiliá-lo a resolver os problemas de aritmética, álgebra e cálculo, usando o software Mathematica que foi desenvolvido pela Wolfram Research, Inc. Com ele, você pode realizar tarefas computacionais num espectro bastante amplo, que se estende desde simples operações aritméticas até a realização do cálculo de derivadas, integrais, desenvolvimentos em séries, passando por manipulações algébricas, incluindo aquelas vetoriais e matriciais. Tais operações, quando realizadas manualmente, podem consumir muito tempo.

Este livro pode ser usado tanto por alunos recém ingressados na faculdade quanto por pesquisadores que queiram ser auxiliados por um programa de caráter geral como o Mathematica.

Programas desta natureza geraram nos alunos e profissionais da área de ciências exatas, a tendência de realizem no computador as tarefas que exigem manipulações algébricas e para as quais se criou um algoritmo. Este procedimento acelera a produtividade do usuário, possibilitando sua total fixação no desenvolvimento de processos que exigem criatividade, coisa que o computador (ainda) não realiza. Quem não seguir esta tendência, estará na posição de um ciclista em uma auto-estrada: mesmo sendo um atleta, chegará ao destino bem depois dos demais podendo, inclusive, ser atropelado.

Este software é, sem sombra de dúvida, uma ferramenta de valor inestimável para todos aqueles que se dedicam às ciências exatas. Há uma vasta bibliografia em Inglês sobre o assunto mas, em Português, dispomos de poucas referências.

Escrevi este livro com o intuito de proporcionar a você, meu caro leitor, um texto de leitura agradável. Espero que encontre em suas páginas a orientação necessária para resolver suas tarefas com o auxílio do Mathematica. Procurei ilustrar as principais funções e comandos com exemplos simples mas representativos. Recomendo que você procure reproduzir no Mathematica os exemplos apresentados, experimentando de próprio punho a potencialidade deste programa.

Os alunos do ITA, desde o primeiro ano do curso, são estimulados a usarem o Mathematica na solução de seus problemas físicos e matemáticos. O número crescente de pessoas que me procuravam em busca de informações sobre este

programa foi um fator motivante no desenvolvimento desta empreitada. As solicitações vinham tanto dos pós-graduandos quanto dos engenheirandos que estavam desenvolvendo seus trabalhos de graduação.

A edição desta obra foi feita em Latex, utilizando uma interface gráfica para Windows, denominada Scientific WorkPlace, desenvolvida pela TCI Software Research. Os gráficos foram realizados no Mathematica e importados para o texto.

Escrever um livro é como uma aventura na selva, durante a qual passamos por momentos de euforia, fortes emoções, jornadas intermináveis. É a realização de um projeto no qual acreditamos e em cuja materialização colocamos toda a nossa energia. Existem, sem dúvida, aqueles momentos em que nos sentimos com o espírito combalido. Prosseguimos apenas pelo estímulo daqueles que nos cercam. A todos, meus sinceros agradecimentos. Agradeço à minha esposa e filhos que, mesmo sentindo a ausência paterna nos momentos de trabalho mais intenso, nunca deixaram de me apoiar. Em particular, menciono a atuação do André e do Júnior que conferiram cada exemplo deste livro, digitando-os e executando-os com o Mathematica. Agradeço à professora Maria Cristina de Campos Vieira por ter lido o capítulo que trata da Análise Vetorial, apresentando sugestões. Meus agradecimentos aos colegas do Departamento de Matemática do ITA por proporcionarem um ambiente adequado para o desenvolvimento deste trabalho.

Antonio Cândido Faleiros

São José dos Campos, SP, 1997

Sumário

Capítulo 1

Introdução ao Mathematica

1.1 Apresentação

Os primeiros computadores foram idealizados para realizar cálculos aritméticos. Em meados da década de 60, apareceram os primeiros programas capazes de fazer manipulações simbólicas. No início, eram modestos e limitados, mas evoluíram bastante e hoje podem realizar praticamente todos os cálculos, sejam eles aritméticos ou algébricos.

O Mathematica é um programa elaborado para automatizar toda a tarefa puramente braçal dos desenvolvimentos matemáticos. Podemos entregar a ele efetuando toda a tarefa frequentemente demorada e tediosa dos desenvolvimentos puramente braçais, liberando o usuário para realizar as atividades criativas que são mais nobres e (ainda) não podem ser realizadas pelo computador.

O Mathematica é capaz de realizar operações aritméticas e algébricas, fazer substituições, desenvolver e simplificar expressões, resolver equações e operar com vetores e matrizes. Também possui funções capazes de derivar e integrar. Com ele, o usuário poderá obter gráficos de alta qualidade que podem ser impressos ou exportados para outros programas. Com ele podemos gerar fórmulas nos formatos exigidos pelo Fortran, pelo C ou pelo processador de texto TeX. Com simplicidade, pode-se transferir estas fórmulas para outros programas.

A título ilustrativo, me permita o leitor citar um renomado educador brasileiro, que escreveu sob o pseudônimo de Malba Tahan. Autor de uma vasta coletânea de livros, estimulou toda uma geração de jovens a estudar ciências exatas, principalmente a Matemática. Quando eu era adolescente, morando na casa de meus pais, lendo o livro Didática da Matemática de Malba Tahan, Editora Saraiva, 1962, me deparei na página 223 com a seção de número 14, cujo título é Curiosidade Aritmética. Nesta sessão, o autor, que com suas belas estórias inspirou muitas vocações, observava que os números 32^2 e 49^2

bem como 32^4 e 49^4 eram formados pelos mesmos algarismos, como nos mostra a tabela abaixo.

$$\begin{aligned} 32^2 &= 1024, \qquad 32^4 = 1048576 \\ 49^2 &= 2401, \qquad 49^4 = 5764801 \end{aligned}$$

Na seqüência, ele desafiava o leitor a ir adiante: "Outras potências desses dois números apresentarão propriedades análogas?" questionava ele. "É bem possível que 32^{16} e 49^{16} sejam expressas por números formados pelos mesmos algarismos. Estará o leitor disposto a verificar?". Em seguida completava: "O número 32 elevado à décima sexta potência terá nada menos de 25 algarismos." Na época não me habilitei a calcular tal valor mas persistiu a curiosidade. Quando me deparei com o Mathematica, percebi que poderia finalmente obter o tão almejado resultado sem grande esforço. Com dois comandos, executados instantaneamente, apareceram os valores de 32^{16} e 49^{16} que apresento abaixo.

$$\begin{aligned} 32^{16} &= 1\ 208\ 925\ 819\ 614\ 629\ 174\ 706\ 176 \\ 49^{16} &= 1\ 104\ 427\ 674\ 243\ 920\ 646\ 305\ 299\ 201 \end{aligned}$$

Como o próprio leitor pode verificar, não são formados pelos mesmos algarismos. É uma pena. Todavia, esta ação de poucos segundos (entre digitar e executar) satisfez plenamente minha curiosidade.

1.2 O núcleo e a interface com o usuário

O Mathematica é constituído de um núcleo e de uma interface com o usuário. O núcleo foi concebido para fornecer os mesmos resultados em todos os computadores. Ele contém as funções responsáveis pelos cálculos. Certamente computadores com mais memória poderão realizar operações mais complexas e os mais rápidos fornecerão a resposta em um tempo menor. A interface com o usuário é a parte visual do programa com a qual o usuário interage. Embora esta parte possa variar de uma arquitetura de computador para outra, ela se comporta de modo análogo nos sistemas operacionais que possuem uma interface gráfica, tal como o Windows para PC.

Vamos nos ater neste manual ao Mathematica em sua versão para WINDOWS. O usuário não encontrará dificuldades se seu computador dispuser de outro sistema operacional, desde que disponha de uma interface gráfica, que atualmente é o padrão.

1.3 Como entrar no ambiente do Mathematica

Para **entrar** no ambiente do Mathematica, abra o grupo no qual se encontra o programa e aperte duas vezes consecutivas o botão esquerdo do mouse com o cursor sobre o ícone do Mathematica. Aguarde enquanto o computador carrega o programa na memória. Quando esta tarefa se completa, aparece no alto da tela a mensagem

Mathematica for Windows – [Newnb–1]

e, logo na linha de baixo, o menu de comandos da interface, onde se lê

File Edit Cell Graph Action Style Options Window Help

que são comandos para auxiliar o usuário. Logo abaixo, segue uma barra de ferramentas e, seguindo-a aparece a **área de trabalho**, que estará em branco, aguardando um comando do usuário. Para executar um comando digite-o usando o teclado e, ao completá-lo, pressione

[Shift]+[Enter]

isto é, mantenha pressionada a tecla **Shift** enquanto pressiona e solta a tecla **Enter**. Outra opção consiste em apertar a tecla

[Insert]

Após a emissão de cada comando aparecerá as mensagens

In[n]:=

e

Out[n]=

onde "n" é o número da instrução executada.

Daremos o nome de **sessão** ao conjunto de todos os comandos emitidos e respostas fornecidas desde o instante que se entra até o momento em que se sai do Mathematica.

Exemplo 1.1 *Ao entrar no Mathematica e estando a tela vazia,*

Digite: ***3 + 4***

Pressione: ***[Shift]+[Enter]***

Após realizar esta operação, surgem na tela as mensagens

In[1]:= ***3 + 4***

Out[1]= 7

para indicar que o primeiro comando emitido foi ***3+4*** *e que o primeiro resultado obtido foi 7 (In vem da palavra inglesa input que significa entrada e Out vem da palavra inglesa output que significa saída).*

Em seguida, pode-se continuar emitindo novos comandos.

Digite: ***5 * 7***

Pressione: ***[Insert]***

Resposta do sistema

In[2]:= ***5 * 7***

Out[2]= 35

Observe que dentro dos colchetes de In e Out aparece agora o número 2. Ele indica que este é o segundo comando da sessão. Quando se sai do Mathematica e se inicia uma nova sessão este contador de comandos retorna com o valor 1.

1.4 Como sair do ambiente do Mathematica

Para **sair** do Mathematica, leve o cursor do mouse na barra do menu situada no alto da tela e clique o botão esquerdo do mouse na palavra **File.** Aparecerá na tela um retângulo com diversas opções. Coloque o cursor do mouse na palavra **Exit** e clique o botão esquerdo. Ao surgir a mensagem

Save current changes to Newnb–1?

clique no botão **Yes** para gravar a sessão que foi desenvolvida e siga o procedimento usual para salvar um arquivo. Uma sessão gravada pode ser recuperada posteriormente. Não desejando salvar a sessão, clique no botão **No.**

Não se deve escrever no próprio diretório em que se encontra o Mathematica. Salve a sessão em um diretório criado para este fim.

Uma segunda possibilidade de saída consiste em pressionar duas vezes consecutivas o botão esquerdo do mouse sobre o retângulo situado no canto superior esquerdo da tela do Mathematica.

Uma terceira opção consiste em pressionar

[Alt]+[F4]

isto é, mantenha pressionada a tecla **Alt**, enquanto pressiona e solta a tecla **F4**, soltando em seguida a tecla **Alt**.

1.5 Emitir comandos extensos

Quando um comando for grande e não couber em uma única linha, devemos seguir o procedimento que segue. Ao chegar no final da linha, pressione a tecla

[Enter]

para fazer o cursor mudar de linha e continue a digitar o comando, pressionando a tecla **Enter** sempre que chegar ao final de uma linha. Ao completar a digitação do comando, pressione

[Shift]+[Enter]

isto é, pressione a tecla **Shift** e, enquanto a mantém pressionada, pressione e solte a tecla **Enter**, soltando em seguida a tecla **Shift**. Com este procedimento o comando é executado.

Exemplo 1.2 *Digite:* ***1 + 2 + 3 + 4 + 5 + 6 +***

Pressione: ***[Enter]***

Digite: ***7 + 8 + 9 + 10***

Pressione: ***[Shift]+[Enter]***

Resposta: Out[3]= 55

Observe que o cálculo é efetuado apenas depois do ***[Shift]+ [Enter]****.*

Nos próximos capítulos descreveremos os comandos e funções existentes no Mathematica. Não seremos exaustivos. Para uma descrição completa do núcleo deste programa recomendamos o livro Mathematica, a System for Doing Mathematics by Computer, Second Edition, Addison-Wesley Publishing Company, Inc., 1991, escrito por Stephen Wolfram que foi um dos idealizadores e é o dono da companhia Wolfram Research, Inc., que desenvolve o Mathematica.

1.6 Obtendo socorro

O Mathematica possui um dispositivo de auxílio à disposição do usuário na barra do **Menu**, situado no alto da tela. Pressionando o botão do **mouse** com o cursor sobre a palavra **Help** do menu, surge na tela uma janela que irá auxiliá-lo na utilização do Mathematica.

Este **Help** funciona como todos os sistemas de Help do Windows.

Para solicitar o significado de um determinado comando ou função, digite

```
? nome
```

onde **nome** é o nome da função ou comando. Complete o comando pressionando a tecla **Insert**. Digitando-se

```
? nom*
```

e pressionando a tecla **Insert**, obtemos a relação de todos os comandos cujo nome começa com **nom**.

Exemplo 1.3 *Vamos obter a relação de todas as funções do Mathematica que se iniciam com* ***Pol***

Digite: ***? Pol****

Pressione: ***[Insert]***

Resposta:

PolyGamma	*PolynomialLCM*
Polygon	*PolynomialMod*
PolygonIntersections	*PolynomialQ*
PolyLog	*PolynomialQuotient*
PolynomialDivision	*PolynomialRemainder*
PolynomialGCD	

Para obter informações sobre o PolynomialGCD,

Digite: ***?PolynomialGCD***

Pressione: ***[Insert]***

Resposta:

PolynomialGCD[poly1, poly2, ...] gives the greatest common divisor of the polynomials poly1, poly2, ... PolynomialGCD[poly1, poly2, ..., Modulus- > p] gives the GCD modulo the prime p.

1.7 Interrompendo um cálculo interminável

Se por uma fatalidade o usuário cometer um erro ao emitir um comando e o cálculo se tornar interminável, ele poderá ser interrompido. Um dos processos de interrupção consiste em apontar o cursor sobre a palavra **Action** do menu e cliclar o botão do mouse. Quando se abrir o retângulo de opções, clique o botão do mouse com o cursor sobre a palavra **Interrupt**.

Outra possibilidade, consiste em pressionar

[**Alt**] + [**.**]

isto é, pressione a tecla **Alt** e enquanto a mantém pressionada, aperte e solte a tecla que contém o ponto final (**.**). Solte a tecla **Alt** em seguida. Aparece um retângulo com diversos botões. Clique o botão com a palavra **Abort**. Isto interromperá o cálculo.

Capítulo 2

Aritmética

2.1 Executando um comando

Como destacamos, um comando será executado apenas quando, ao ser totalmente digitado, for finalizado acionando-se

[**Shift**] + [**Enter**]

isto é, mantém-se pressionada a tecla **Shift** enquanto aperta e solta a tecla **Enter**, liberando a tecla **Shift** em seguida. Outro modo de finalizar, que particularmente considero mais simples, consiste em pressionar a tecla

[**Insert**]

2.2 Tipos de números

O Mathematica é capaz de trabalhar com números **inteiros**, **racionais**, **reais** e **complexos**. Inteiros são números da forma

sddddddddd...

onde **s** é o sinal, que pode ser + ou –, sendo o sinal positivo opcional e **d** é um dígito entre 0 e 9. São números inteiros

45 9848376716304958 123456789012345

Não há limite quanto ao tamanho do número inteiro.

A única limitação está relacionada à quantidade de memória disponível na máquina.

Racionais são os números obtidos pela razão entre dois números inteiros. São racionais

$$\frac{1}{3} \qquad \frac{34581}{27} \qquad \frac{8375503827326}{73665374848}$$

Reais são os números que possuem parte inteira e fracionária, sendo escritos com o formato

sddddd.dddd....

onde **s** é o sinal e **d** são os dígitos que compõem o número. A parte inteira é separada da parte fracionária por meio de um ponto. O que estamos chamando de número real é na realidade um número racional apresentado na forma decimal. São números reais

4.27891575

3.27 * 10^12 (é assim que se escreve $3,27 \times 10^{12}$)

Os números complexos possuem a forma

a + b I

onde **a** e **b** podem ser números inteiros, racionais ou reais e **I** é a unidade imaginária ($I = \sqrt{-1}$). O Mathematica diferencia **letras maiúsculas** de **minúsculas**. Portanto, atente para o fato de que a unidade imaginária **I** deve ser escrita com letra maiúscula. São números complexos

$$-1.87 + 2.36\ \mathrm{I} \qquad \frac{47}{11} - 964\ \mathrm{I} \qquad \frac{1}{3} + 7.8\ \mathrm{I}$$

2.3 Operações básicas

Se **A** e **B** forem dois números inteiros, racionais, reais ou complexos, então a soma, a diferença, o produto, a divisão e a potência, são denotados de acordo

com a tabela que segue

A + B	→	soma A com B
A – B	→	subtrai B de A
A * B ou **A B**	→	multiplica A por B
A / B	→	divide A por B
AˆB	→	eleva A à potência B

Nota 2.1 *Este programa aceita* ***multiplicação implícita****. Deste modo,* ***3 2*** *(com um espaço entre o* ***3*** *e o* ***2****) nos dará* ***6*** *por resultado. Por este motivo, não deixe espaços em branco entre dois dígitos de um mesmo número.*

2.4 Hierarquia das operações

Numa expressão envolvendo diversos operadores, pode-se usar o parênteses para agrupar operações, como usualmente se faz em Matemática. Numa expressão aritmética envolvendo diversas operações, elas são realizadas da esquerda para a direita, seguindo uma hierarquia. Primeiro se efetuam as potências, em seguida os produtos e divisões e finalmente as somas e subtrações. Sempre que houver parênteses encaixados, as operações dentro do abre e fecha parêntese mais interno são efetuadas primeiro.

Exemplo 2.1 *Considere uma expressão como*

*2 + 3 * 5 – (4 + 3)ˆ2*

Passo a passo ela é calculada do seguinte modo

2 + 15 – (4 + 3)ˆ2

17 – (4 + 3)ˆ2

17 – 7ˆ2

17 – 49

– 32

Nota 2.2 *Na potência,* ***3ˆ2 4*** *significa* ***(3ˆ2)*4*** *que é igual a* ***36*** *enquanto* ***3ˆ(2 4)*** *significa* ***3ˆ(2*4)*** *que é igual a* ***3ˆ8 = 6561****.*

2.5 Aritmética inteira

A aritmética com números inteiros e racionais é exata, não importando o tamanho do número. A aritmética com números reais é aproximada, sendo que o usuário tem controle sobre a precisão.

Exemplo 2.2 *Observe as diferenças entre os resultados que seguem. Inicie uma nova sessão e*

Digite: ***23/69***

Pressione: ***[Insert]***

Resposta: Out[1]= $\frac{1}{3}$

Digite: ***23./69***

Pressione: ***[Insert]***

Resposta: Out[2]= 0.333333

Neste último comando, ***23.*** *foi interpretado como número real e portanto o Mathematica ofereceu um resultado aproximado. No caso anterior,* ***23*** *e* ***69*** *são inteiros de modo que o programa usa a aritmética exata, nos oferecendo o número racional* 1/3*, que é exatamente a fração* 23/69 *simplificada.*

Vamos insistir um pouco mais na diferença entre a aritmética dos inteiros e racionais que é exata e a aritmética dos reais, que é aproximada. Um número inteiro é aquele que não possui o ponto decimal. Os que possuem o ponto decimal são considerados reais, mesmo que a parte fracionária seja nula, tais como

3. 567879.

No Mathematica, ao operarmos com números inteiros, os resultados obtidos serão **exatos**, mesmo que o número de dígitos da resposta seja muito grande.

Exemplo 2.3 *Observe os resultados*

Digite: ***4^99***

Pressione: ***[Insert]***

Resposta: Out[3]=

401734511064747568885490523085290650630550748445698208825344

A barra invertida ao final da primeira linha nos indica o resultado não está completo e continua na próxima linha.

Quando o Mathematica se depara com operações entre inteiros, ele nos devolve o resultado exato. Deste modo, quando lhe fornecemos uma fração, tal como **44/12** ele apenas a simplifica, nos dando como resposta a fração **11/3**. Esta divisão não foi escrita na forma decimal pois sua representação decimal é uma dízima periódica, com um número infinito de dígitos.

Em resumo: **A aritmética com números inteiros e racionais é exata.**

Quando operamos com números reais, o Mathematica nos fornece um resultado aproximado, com seis dígitos significativos.

Se solicitamos o cálculo de **11./3** teremos como resposta **3.66667.** A diferença entre este resultado e o obtido anteriormente com números inteiros, está no ponto decimal. O número **11.** (observe o ponto decimal) foi tratado como um número real. Basta um dos números ser real em uma expressão aritmética para que todas as operações sejam feitas aproximadamente.

Como exemplo, a operação **3. + 1/2** nos devolve o resultado **3.5**.

2.6 Controlando a precisão

Mesmo trabalhando com números inteiros é possível obter resultados aproximados, se terminarmos a expressão com

//N

onde o **N** deve ser escrito com letra maiúscula.

Exemplo 2.4 *Inicie uma nova sessão e*

Digite: ***N[4/3, 10]***

Pressione: ***[Insert]***

Resposta: Out[1]= 1.333333333

Digite: ***4/3 //N***

Pressione: ***[Insert]***

Resposta: Out[2]= 1.33333

No penúltimo comando, ***N[4/3, 10]*** *solicita o valor aproximado de* ***4/3*** *com 10 dígitos e, no último,* ***//N*** *solicita o valor aproximado de* ***4/3*** *com um número padrão de dígitos, estabelecido pelo próprio sistema.*

O formato geral para se obter o valor aproximado de uma expressão aritmética **expr** com **k** dígitos de precisão é

N[expr , k]

enquanto que

expr //N ou **N[expr]**

nos fornece o valor aproximado de **expr** com um número padrão de dígitos, estabelecido pelo próprio sistema.

Exemplo 2.5 *Continuando a sessão do exemplo anterior,*

Digite: ***1 / 2 + 1 / 3 + 1 / 4***

Pressione: ***[Insert]***

Resposta: Out[3]= $\frac{13}{12}$

Digite: ***1 / 2 + 1 / 3 + 1 / 4 //N***

Pressione: ***[Shift]+[Enter]***

Resposta: Out[4]= 1.08333

Valores aproximados podem ser particularmente úteis quando o resultado exato envolve um número muito grande de dígitos. Em muitos casos é preferível obter a ordem de grandeza de um número que o seu valor exato.

Solicite o cálculo de **2^1000** e **2^1000 //N**. A segunda informação, que nos dá o valor aproximado de **2^1000** é, na maioria dos casos, mais interessante.

Exemplo 2.6 *Vamos calcular $\sqrt{2}$ (Sqrt[2]) com vinte dígitos*

Digite: ***N[Sqrt[2] , 20]***

Pressione: ***[Shift]+[Enter]***

Resposta: Out[5]= 1.4142135623730950488

Digite: ***N[Sqrt[2]]***

Pressione: ***[Shift]+[Enter]***

Resposta: Out[6]= 1.41421

Digite: ***Sqrt[2] //N***

Pressione: ***[Shift]+[Enter]***

Resposta: Out[7]= 1.41421

2.7 O céu é o limite

Quando falamos em aritmética exata para os números inteiros e para o cálculo de funções com um número qualquer de casas decimais mediante o uso da função

N[exp , k]

não estabelecemos um limite para **k**. O limite será ditado pela máquina. Quanto maior a precisão, maior será a solicitação por mais memória. Se sua máquina for poderosa, o céu é o limite!

Exemplo 2.7 *Para obter o π com 200 dígitos,*

Digite: ***N[Pi , 200]***

Pressione: ***[Insert]***

Resposta: Out[8]=

3.14159265358979323846264338327950288419716939937510

58209749445923078164062862089986280348253421170679 8

21480865132823066470938446095505822317253594081284 8

11174502841027019385211055596446229489549303819 6

onde a barra invertida, ao final de cada linha, indica que o resultado não terminou e continua a ser apresentado na próxima linha.

2.8 Constantes pré-definidas

O Mathematica contém algumas constantes de uso geral tais como

Pi = 3.141592... (número pi)

E = 2.71828... (constante neperiana)

Degree = Pi / 180 (fator de conversão de graus para radianos)

I → (unidade imaginária = $\sqrt{-1}$)

Infinity → infinito

Observe que o nome de todas elas começam com uma **letra maiúscula**. Podemos usar o comando

N[const , k]

para obter estas constantes com a precisão que desejarmos. A constante **Degree** pode ser utilizada para calcular as funções trigonométricas em graus. Por exemplo,

Sin [20 Degree] //N

nos fornece o valor aproximado do seno do ângulo de 20 graus.

2.9 Uso de resultados anteriores

No cômputo de diversas expressões algébricas, com freqüência necessitamos efetuar diversos cálculos e unir os resultados no final. Prevendo esta necessidade, à medida que vamos emitindo comandos, os resultados vão sendo gravados, podendo ser utilizados num cálculo posterior, desde que não saiamos do Mathematica, isto é, enquanto permanecermos na mesma sessão. Quando encerramos uma sessão, os cálculos efetuados se perdem, a menos que os gravemos em um arquivo, como está explicado no capítulo que trata de arquivos.

Exemplo 2.8 *Inicie uma nova sessão com o Mathematica e emita os comandos*

Digite: ***4 * 5***

Pressione: ***[Insert]***

Resposta: *In[1]:=* ***4 * 5***
Out[1]= *20*

Digite: ***3 * 6***

Pressione: ***[Insert]***

Resposta: *In[2]:=* ***3 * 6***
Out[2]= *18*

É possível utilizar os resultados destas operações nos comandos subseqüentes. Para multiplicar por 5 o resultado de Out[2],

Digite: ***5 * %***

Pressione: ***[Insert]***

Resposta: *In[3]:=* ***5 * %***
Out[3]= *90*

para duplicar o resultado de Out[1],

Digite: ***2 * %%%***

Pressione: ***[Insert]***

Resposta: *In[4]:=* ***2 * %%%***
Out[4]= *40*

De modo geral, pode-se usar o n-ésimo resultado anterior repetindo n vezes consecutivas o símbolo %. Este processo não é recomendável quando o número de repetições para se recuperar o resultado da n-ésima saída anterior for muito grande. Neste caso, simplesmente se digita **In[n]** ou **Out[n]** para indicar a **n** ésima entrada ou **n** ésima saída, respectivamente.

Exemplo 2.9 *Para somar os resultados das operações de números 3 e 4,*

Digite: ***Out[3] + Out[4]***

Pressione: ***Insert***

Resposta: *In[5]:=* ***Out[3] + Out[4]***
Out[5]= *130*

Deste ponto em diante iremos omitir o trecho *In[n]:=* **comando** que vínhamos colocando na resposta. Quando o usuário digita um comando e pressiona a tecla **Insert** para executá-lo, o Mathematica coloca no lado esquerdo da tela e um pouco acima do comando a sua ordem de entrada, indicado no *In[n]:=* . Quando houver conveniência, incluiremos este trecho.

2.10 Corrigindo e recalculando um comando anterior

A interface gráfica do Mathematica possui uma barra vertical do lado direito, através da qual podemos navegar por toda a sessão do Mathematica, adotando o mesmo procedimento utilizado em qualquer editor de texto dentro do Windows. Desejando reaproveitar um comando anterior, traga-o para a parte visível da janela usando esta barra. Pressione o botão do mouse com o cursor sobre o comando. Agora o comando pode ser modificado com o mesmo procedimento usado em qualquer editor de textos no Windows. Com o cursor ainda na célula do comando, pressione a tecla **Insert** ou **Shift+Enter** para executar o comando. Observe que agora o número do comando é igual ao número do último comando emitido, acrescido de uma unidade.

2.11 Inserindo um comando entre dois outros

Os cálculos são agrupados em **células** que são trechos da sessão delimitados por traços verticais situados à direita da tela, na altura das entradas e saídas. Quando se coloca o cursor entre uma célula e outra, ele toma a forma de uma barra horizontal. Pressionando o botão do mouse nesta posição, surge um

traço horizontal ao longo da tela. Digitando-se algo, as teclas são ecoadas na tela, na posição marcada pelo traço horizontal.

Pode-se então digitar um comando e executá-lo pressionado a tecla **Insert**.

2.12 Números complexos

Os números complexos são digitados na forma

```
a + b * I
```

onde **a** e **b** são números inteiros, racionais ou reais. O **a** é a parte real e **b** é a parte imaginária do número complexo. Para efetuar soma subtração, multiplicação e divisão de números complexos, digite estas operações como se faz no caso real. Julgamos conveniente lembrar o leitor que a letra maiúscula **I** é uma palavra reservada pelo Mathematica para designar a unidade imaginária. Se **a** e **b** forem inteiros, será utilizada a aritmética inteira que, como comentamos, é exata.

Pode-se substituir o sinal de multiplicação * por um espaço. Quando **b** for um número, até o espaço é dispensável. Quando **b** for uma variável, o espaço ou o sinal * são obrigatórios.

Exemplo 2.10 *Apresentamos abaixo uma soma, uma subtração, uma multiplicação e uma divisão. Nos dois primeiros comandos, os parênteses são dispensáveis mas foram incluídos para destacar as operações de soma e subtração entre complexos. Inicie uma nova sessão e*

Digite: ***(1 + 7 * I) + (–5 + 2 * I)***

Pressione: ***[Insert]***

Resposta: Out[1]= – 4 + 9 I

Digite: ***(6 – 8.3 * I) – (– 9 + 3 * I)***

Pressione: ***[Insert]***

Resposta: Out[2]= 15 – 11.3 I

Digite: ***(3.9 + 2. * I) * (4.1 – I)***

Pressione: ***[Insert]***

Resposta: Out[3]= 17.99 + 4.3 I

Digite: ***(2 − 5 * I) / (−2 + 2 * I)***

Pressione: ***[Insert]***

Resposta: Out[4]= $-\frac{7}{4}+\frac{3\,I}{4}$

Digite: ***N[%]***

Pressione: ***[Insert]***

Resposta: Out[5]= − 1.75 + 0.75 I

Se alguma operação ou função resultar em um número complexo, o resultado será apresentado na forma **a + b I**.

Exemplo 2.11 *Continuando a sessão anterior, vamos verificar a resposta quando se solicita a* $\sqrt{-4}$ *que sabemos ser igual a* $2i$.

Digite: ***Sqrt[−4]***

Pressione: ***[Insert]***

Resposta: Out[6]= 2 I.

Para os números complexos, além das funções anteriormente enumeradas, temos

Re[**z**]	→	parte real de **z**
Im[**z**]	→	parte imaginária de **z**
Abs[**z**]	→	módulo de **z**
Arg[**z**]	→	argumento principal de **z**
Conjugate[**z**]	→	complexo conjugado de **z**

2.13 Números primos

Para determinar **números primos**, obter os **divisores primos** ou decompor um número em **fatores primos**, temos os comandos abaixo, onde **n** é um número inteiro.

Prime[**n**]	....	fornece o **n**-ésimo número primo
Divisors[**n**]	....	fornece a lista de divisores de **n**
FactorInteger[**n**]	....	fornece a lista de fatores primos de **n**.

PrimePi[**n**]	....	fornece o número de primos menores ou iguais a **n**
PrimeQ[**n**]	....	retorna o valor **True** se **n** for primo

Exemplo 2.12 *Vamos ilustrar os comandos acima. Nas saídas, faremos observações nossas entre parênteses. Inicie uma nova sessão e*

Digite: ***Prime[10]***

Pressione: ***[Shift] + [Enter]***

Resposta: Out[1]= 29 (o 29 é o décimo número primo)

Digite: ***PrimePi [10]***

Pressione: ***[Shift] + [Enter]***

Resposta: Out[2]= 4 (existem 4 números primos menores que 10)

Digite: ***PrimeQ[10]***

Pressione: ***[Shift] + [Enter]***

Resposta: Out[3]= False (o número 10 não é primo)

Digite: ***n = 5^2 * 11^3***

Pressione: ***[Shift] + [Enter]***

Resposta: Out[4]= 33275

Para obter o conjunto de divisores de ***33275****,*

Digite: ***Divisors[n]***

Pressione: ***[Shift] + [Enter]***

Resposta: Out[5]= { 1, 5, 11, 25, 55, 121, 275, 605,

1331, 3025, 6655, 33275 }

Digite: ***FactorInteger[n]***

Pressione: ***[Shift] + [Enter]***

Resposta: Out[6]= { { 5 , 2 } , { 11 , 3 } }

No conjunto de divisores acima, cada elemento é um subconjunto. O primeiro elemento do subconjunto é o divisor e o segundo é o expoente com que o divisor entra na fatoração do número. Assim, $33275 = 5^2 11^3$.

2.14 Números aleatórios

Há uma função encarregada de gerar números aleatórios. Para gerar um número aleatório real (na verdade, pseudo-aleatório) entre 0 e 1, use

Random[ ]

Para gerar um número aleatório real ou inteiro entre **xmin** e **xmax** comande

Random[**tipo, {xmin, xmax}**]

onde a palavra **tipo** deve ser substituída por **Real** ou **Integer**. Para gerar um número complexo pseudo-aleatório situado em um retângulo do plano complexo, comande

Random[Complex, {xmin, xmax}]

onde **xmin** e **xmax** são números complexos da forma **a + b I**, sendo **xmin** o vértice inferior esquerdo e **xmax** o vértice superior direito do retângulo.

Exemplo 2.13 *Este exemplo ilustra o uso da função* ***Random****. Inicie uma nova sessão do Mathematica e*

Digite: ***Random[ ]***

Pressione: ***[Insert]***

Resposta: Out[1] = 0.303756

A resposta a este comando muda a cada nova emissão. Se você emitir este comando, provavelmente obterá uma resposta diferente.

Digite: ***Random[Integer, { 0, 100 }]***

Pressione: ***[Insert]***

Resposta: Out[2] = 21

2.15 Funções elementares

O Mathematica possui uma grande quantidade de funções internas disponíveis para o usuário. Os nomes de todas elas começam com uma letra maiúscula e os argumentos das funções são delimitados por colchetes em lugar de parênteses como na notação usual. Nesta etapa citaremos as funções mais comuns

Sqrt[x]	→	raiz quadrada de x = $\sqrt{x}$
Exp[x]	→	exponencial de x = e^x

Log[x]	→	logaritmo neperiano de x = $\log x$
Log[b,x]	→	logaritmo de x na base b = $\log_b x$

Sin[**x**]	→	seno de x = sen x
Cos[**x**]	→	cosseno de x = cos x
Tan[**x**]	→	tangente de x = tan x

ArcSin[**x**]	→	arco seno de x = arcsen x
ArcCos[**x**]	→	arco cosseno de x = arccos x
ArcTan[**x**]	→	arco tangente de x = arctan x

Abs[**x**]	→	valor absoluto de x
Round[**x**]	→	inteiro mais próximo de x

Max[**x, y, ...**]	→	determina o maior valor dentre x, y, ...
Min[**x, y, ...**]	→	determina o menor valor dentre x, y, ...

n!	→	fatorial de n = $1 \times 2 \times 3 \times \cdots \times (n-1) \times n$
Mod[**n, m**]	→	resto da divisão de n por m

Estas funções tentam fornecer o resultado exato quando o argumento for inteiro. Se o resultado exato da operação for um número irracional, o Mathematica apenas repetirá o comando que foi digitado.

Exemplo 2.14 *Vamos usar algumas destas funções.*

Digite: ***Sqrt[3]***

Pressione: ***[Insert]***

Resposta: Out[11]= Sqrt[3]

Digite: ***Sqrt[3.]***

Pressione: ***[Insert]***

Resposta: Out[12]= 1.73205

A diferença entre os comandos ***Sqrt[3]*** *e* ***Sqrt[3.]*** *consiste na forma de se escrever o argumento. No primeiro caso o argumento é inteiro e portanto o sistema tenta responder com o valor exato da raiz. Como a raiz quadrada de 3 é um número irracional, ele não possui representação decimal exata. Deste modo, o sistema simplesmente repete a entrada. No segundo caso, o argumento é um número real e portanto o Mathematica fornece um resultado aproximado.*

2.16 Funções trigonométricas

Ao trabalharmos com expressões envolvendo funções elementares, nos deparamos com uma variedade muito grande de formas para apresentá-las. O Mathematica dispõe de pacotes capazes de realizar inúmeras transformações em funções elementares, para apresentar os resultados em uma forma conveniente ao usuário. O núcleo central do Mathematica, embora sendo menos poderoso que estes pacotes, possui recursos consideráveis para manipular estas funções. O comando

Expand[expr , Trig − > True]

desenvolve as funções trigonométricas, escrevendo sen $^2(x)$ em termos de $sin(2x)$, etc.

Factor[expr , Trig − > True]

fatora funções trigonométricas, substituindo $sin(2x)$ em termos de sen $^2(x)$, etc.

2.17 Base numérica

O Mathematica pode trabalhar com números em diferentes bases. Lembramos que os números normalmente são apresentados na base 10. Por exemplo, o número 1234 corresponde a

$$1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0.$$

Para ressaltar que 1234 está na base 10 pode-se escrever $(1234)_{10}$. Em geral, se a_0, a_1, ..., a_p são dígitos entre 0 e 9, então

$$(a_p a_{p-1} \cdots a_1 a_0)_{10} = a_p \times 10^p + a_{p-1} \times 10^{p-1} + \cdots + a_1 \times 10^1 + a_0 \,.$$

Eventualmente, é interessante usar outras bases numéricas. Os computadores trabalham internamente com números na base 2 e os converte para a base 10, que é a base na qual trabalhamos. Para representar um número na base dois, usamos apenas os dígitos 0 e 1. Deste modo, sendo a_0, a_1, ..., a_p são dígitos entre 0 e 1, então

$$(a_p a_{p-1} \cdots a_1 a_0)_2 = a_p \times 2^p + a_{p-1} \times 2^{p-1} + \cdots + a_1 \times 2^1 + a_0 \,.$$

De modo geral, para representar um número numa base n inteira, precisamos de n dígitos. Quando $n \leq 10$, usamos os dígitos 0, 1, ..., 9 para representar os números 0, 1, ..., 9. Quando $n > 10$, usamos as letras a, b, c, ..., para representar os números 11, 12, 13, ... O Mathematica segue esta convenção. Para os números na base n temos

$$(a_p a_{p-1} \cdots a_1 a_0)_n = a_p \times n^p + a_{p-1} \times n^{p-1} + \cdots + a_1 \times n^1 + a_0 \,.$$

Para passar um número $N = (a_p a_{p-1} \cdots a_1 a_0)_n$ de sua representação em uma base n para outra k, escrevemos $N = (b_q b_{q-1} \cdots b_1 b_0)_k$ e usamos a igualdade

$$(b_q b_{q-1} \cdots b_1 b_0)_k = (a_p a_{p-1} \cdots a_1 a_0)_n$$

que corresponde a

$$b_q \times k^q + b_{q-1} k^{q-1} + \cdots + b_1 \times k^1 + p_0 = a_p \times n^p + a_{p-1} \times n^{p-1} + \cdots + a_1 \times n^1 + a_0 \,.$$

Quando $N = (a_p a_{p-1} \cdots a_1 a_0)_n$ então o algoritmo para obter os bs em $N = (b_q b_{q-1} \cdots b_1 b_0)_k$ é

Forneça o número $N = (a_p a_{p-1} \cdots a_1 a_0)_n$
$q = 0$ e $N_0 = N$
Faça
    b_q = Resto da divisão de N por k.
    $N_{q+1} = (N_q - b_q)/k$
    $q = q + 1$
até que N_q seja igual a zero.

Para escrever o número $N = (a_p a_{p-1} \cdots a_1 a_0)_n$ dentro do Mathematica, usamos a forma $n\hat{}\hat{}a_p a_{p-1} \cdots a_1 a_0$. Quando a base n for 10, podemos escrever simplesmente $a_p a_{p-1} \cdots a_1 a_0$.

Para passar o número N da base n ($2 \leq n \leq 36$) para a base k, usamos o comando

BaseForm[n^^N , k]

Exemplo 2.15 *Vamos passar o número* $(10011001)_2$ *para a base* 10 *e o número* $(1234)_{10}$ *para a base* 2. *Inicie uma nova sessão e*

Digite: ***BaseForm[2^^10011001 , 10]***

Pressione: ***[Insert]***

Resposta: Out[1]= 153

Digite: ***BaseForm[1234 , 2]***

Pressione: ***[Insert]***

Resposta: Out[2]= 10011010010_2

2.18 Unidades de medida

Pode-se trabalhar com unidades de medida dentro do Mathematica e realizar mudança de unidades. Esta capacidade pode ser muito útil na resolução dos problemas de Física.

Exemplo 2.16 *Inicie uma nova sessão e*

Digite: ***10 metros***

Pressione: ***[Insert]***

Resposta: Out[1] = 10 metros

Digite: ***% + 5 metros***

Pressione: ***[Insert]***

Resposta: Out[2] = 15 metros

Digite: ***% / (5 segundo)***

Pressione: ***[Insert]***

Resposta: Out[3] $= \dfrac{3 \text{ metros}}{\text{segundo}}$

Digite: ***% /. { metros -> 0.001 kilo metros, segundo -> 1/3600 hora }***

Resposta: Out[4] $= \dfrac{10.8 \text{ kilo metros}}{\text{hora}}$

É conveniente escrever o prefixo ***kilo*** *separado da unidade metro. As palavras* ***kilo****,* ***metros*** *e* ***segundo*** *se comportam como variáveis às quais não atribuímos valor algum.*

Existe um pacote especial para manipular unidades de medida que reconhece as unidades de medida inglesas. Para carregá-lo comande

```
<< Miscellaneous`Units`
```

Exemplo 2.17 *Vamos ilustrar o uso do pacote que manipula unidades de medida.*

Digite: ***<< Miscellaneous`Units`***

Pressione: ***[Insert]***

Digite: ***velocidade = 100 Meter / (9.85 Second)***

Pressione: ***[Insert]***

Resposta: $\dfrac{10.1523 \text{ Meter}}{\text{Second}}$

Digite: ***Convert[velocidade , Kilo Meter / Hour]***

Pressione: ***[Insert]***

Resposta: $\dfrac{36.5482 \text{ Kilo Meter}}{\text{Hour}}$

Esta é a velocidade média aproximada do campeão mundial dos 100 metros rasos. Observe que o prefixo ***Kilo****, que indica múltiplo de mil, deve ser separado da unidade* ***Metro*** *por um espaço.*

Digite: ***Convert[1 Newton, Dyne]***

Pressione: ***[Insert]***

Resposta: 100000. Dyne

Digite: ***Convert[1 Calorie, Joule]***

Pressione: ***[Insert]***

Resposta: 4.1868 Joule

Digite: ***Convert[1 Ampere Hour, Coulomb]***

Pressione: ***[Insert]***

Resposta: 3600 Coulomb

Digite: ***Convert[1 Weber, Maxwell]***

Pressione: ***[Insert]***

Resposta: $1.\ 10^8$ *Maxwell*

O ***Weber*** *e o* ***Maxwell*** *são unidades de fluxo magnético.*

2.19 Pacotes adicionais

Além das funções do núcleo, o Mathematica dispõe de inúmeros pacotes que podem ser carregados na memória do computador quando solicitados. Para obter maiores informações sobre os pacotes disponíveis no Mathematica, consulte o *Guide to Standard Mathematica Packages*, Technical Report, Wolfram Research.

Para disponibilizar um pacote para uso, o comando é

< < pacote‘nome‘

onde < < são dois sinais de "menor do que" consecutivos, **pacote** é o nome do pacote. Os pacote são divididos em diversos pacotes menores que recebem um **nome** que sucede o nome do pacote no comando.

Importante: O sinal ‘ é uma crase, que fica na mesma tecla que contém o til. Se seu computador estiver configurado para a lingua portuguesa, ao teclar a crase, ela não aparecerá na tela. Para fazê-la aparecer na tela, tecle a crase e, em seguida, pressione a barra de espaço.

Nos próximos capítulos, usaremos alguns pacotes.

Capítulo 3

Álgebra

O desenvolvimento algébrico das expressões matemáticas é uma tarefa que consome tempo. Este é um processo é puramente braçal e não envolve criatividade. Hoje em dia se pode contar com sistemas computacionais capazes de realizar este trabalho automaticamente. Atualmente, os programas projetados para realizar manipulações algébricas são ferramentas imprescindíveis para todo aquele que se dedica às ciências exatas.

O Mathematica está cotado como um dos melhores programas para manipulação algébrica dentre os existentes no mercado.

3.1 Expressões algébricas

Nos desenvolvimentos matemáticos é muito comum substituir números por letras, gerando expressões algébricas. O uso deste artifício ocasionou, sem dúvida alguma, um grande avanço na ciência. Para trabalhar com expressões algébricas, basta escrevê-las como usualmente se faz nos desenvolvimentos manuais, seguindo as regras que aprendemos nos cursos de Matemática.

Exemplos de expressões algébricas seguem abaixo

```
3 + 4x - 2x
```

```
(5 x^2 - 2 x + 7) / (6 x + 8)
```

Nas expressões algébricas, as letras representam variáveis. As variáveis são identificadas por um nome, que pode ser formado por uma única letra ou um conjunto de letras, dígitos e sublinha, sem espaço entre elas. O primeiro caractere que define uma variável deve ser uma letra. Não há limitações quanto ao comprimento do nome. Como veremos posteriormente, estas variáveis podem receber valores numéricos ou mesmo outras expressões algébricas.

Nota 3.1 *O Mathematica aceita multiplicação implícita como usualmente se convenciona nos livros ginasiais. Por esta razão,* **x y** *(com um espaço entre* **x** *e* **y***) indica o produto da variável* **x** *pela variável* **y** *e a expressão* **xy** *(sem espaço entre as letras) indica uma única variável cujo nome é* **xy**.

Nota 3.2 *Quando um número for posicionado à esquerda do nome de uma variável, sem operadores nem espaço em branco entre eles, o sistema interpreta que estão sendo multiplicados. Se o número estiver à direita do nome de uma variável, então o sistema interpretará este conjunto como uma nova variável.*

Nota 3.3 *Nas potências,* **x^2y** *significa* x^2y *ao passo que* **x^(2y)** *significa* x^{2y}.

Exemplo 3.1 *Inicie uma nova sessão e*

Digite: **x = 3**

Pressione: [**Insert**]

Resposta: 3

Digite: **2x**

Pressione: [**Insert**]

Resposta: 6

Digite: **x2**

Pressione: [**Insert**]

Resposta: x2

3.2 Letras maiúsculas e minúsculas

O Mathematica faz distinção entre **letras maiúsculas** e **minúsculas**. Os nomes de todas as funções internas do Mathematica começam com **letra maiúscula**. Para evitar a coincidência entre o nome de uma variável criada por você e uma palavra reservada pelo sistema, inicie o nome de suas variáveis com **letras minúsculas**.

3.3 Simplificando expressões

Sempre que se apresenta uma expressão para o Mathematica, ele tentará colocá-la em sua forma mais simples.

Exemplo 3.2 *Inicie uma nova sessão e*

Digite: **x^2 + 2x + 3x^2 − 4x**

Pressione: [**Insert**]

Resposta: Out[1]= $-2x + 4x^2$

Observe que a resposta foi apresentada num formato matemático padrão, com o expoente na linha superior.

Nas parcelas **2x** e **4x**, não há nenhum operador entre o número e a letra. Nestes casos o Mathematica interpretará que estão sendo multiplicados. Como destacamos anteriormente, a mesma interpretação ocorrerá se houver apenas espaços em branco entre dois números ou duas letras. A este produto sem operador denominamos de **produto implícito**.

Sempre que fornecemos uma expressão algébrica ao Mathematica, ele procura simplificá-la, de acordo com as regras usuais da álgebra. Não havendo a possibilidade de simplificar uma expressão, o Mathematica simplesmente a repete na saída.

Exemplo 3.3 *Continuando a sessão anterior,*

Digite: **Log[Sin[x] + Sqrt[x]]**

Pressione: [**Insert**]

Resposta: Out[2]= Log[Sqrt[x] + Sin[x]]

3.4 Atribuir valores a variáveis

Existem dois modos para se atribuir valores a uma variável. Um deles, que podemos denominar de **global**, tem efeito durante toda a sessão do Mathematica ou até que o valor seja removido ou modificado. O outro modo, que denominaremos de **local**, tem efeito apenas durante a execução do comando no qual se fez a atribuição.

A **atribuição global** é feita mediante um comando do tipo

nome_da_variável = expressão

Quando o Mathematica se depara com este comando, ele simplifica a **expressão** e grava o resultado na variável cujo nome aparece no lado esquerdo do sinal de igualdade.

Exemplo 3.4 *Continuando a sessão,*

Digite: **x = y^2 + 3**

Pressione: [**Insert**]

Resposta: Out[3]= $x = 3 + y^2$

Digite: **x + 4**

Pressione: [**Insert**]

Resposta: Out[4]= $7 + y^2$

Para **limpar** uma variável, eliminando o seu conteúdo, basta comandar

nome_da_variável = .

onde o ponto final faz parte do comando. Outra opção consiste em emitir o comando

Clear[nome_da_variável]

Havendo diversas variáveis para limpar, pode-se emitir o comando

Clear[var1, var2, ..., vark]

onde **var1, var2, ..., vark** são os nomes das variáveis cujos conteúdos se deseja apagar.

Recomenda-se limpar o conteúdo de uma variável quando não formos utilizá-lo mais. As variáveis ocupam espaço na memória e, se as mantivermos desnecessariamente, podemos ficar sem memória disponível para outras atividades.

Para **modificar** o valor de uma variável, basta repetir o comando

nome_da_variável = novo_valor

usando o novo valor que se deseja atribuir à variável. O outro modo de se atribuir valor a uma variável, que denominamos de **local**, consiste em emitir um comando do tipo

expressão /. nome − > valor

que fará com que as ocorrências da variável **nome** na **expressão** assuma o **valor** especificado apenas neste comando. Fazem parte do comando, a barra com o ponto (**/.**) e a seta (− >), que é formada pelo sinal de subtração, seguido pelo sinal de maior. Podemos atribuir valor a múltiplas variáveis, com o comando

expressão /. { var1 − > valor1, var2 − > valor2, ...}

Exemplo 3.5 *Continuando a sessão,*

Digite: **x + y − 3 /. x − > z + 2**

Pressione: [**Insert**]

Resposta: Out[5]= –1 + y + z

Digite: **% /. { y − > 4 , z − > 7 }**

Pressione: [**Insert**]

Resposta: Out[6]= 10

3.5 Manipulação de expressões algébricas

Quando o Mathematica recebe uma expressão algébrica, principalmente polinomial ou racional (expressão que resulta da divisão entre dois polinômios), ele procura simplificá-la ao máximo. Existe uma variedade muito grande para a escolha dos formatos em que uma expressão algébrica será apresentada. Nem sempre se deseja a mais simples ou compacta. Foi para atender às necessidades dos diversos usuários, que se implementou algumas funções capazes de realizar operações que alteram apenas o modo de apresentação de um resultado. Relacionamos abaixo as principais funções que desempenham esta tarefa, onde designamos genericamente por **expr** uma expressão algébrica ou numérica.

Expand[**expr**]	....	Desenvolve os fatores de **expr**
Factor[**expr**]	....	Fatora **expr** aonde for possível
Simplify[**expr**]	....	Coloca **expr** em sua forma mais simples

Quando se trabalha com funções racionais, poderão ser úteis as funções

ExpandAll[**expr**]	....	Desenvolve produtos e potências inteiras de todas as partes de **expr**
Together[**expr**]	....	Coloca toda **expr** sobre um denominador
Cancel[**expr**]	....	Cancela os fatores comuns do numerador e denominador de **expr**

Exemplo 3.6 *Inicie uma nova sessão e*

Digite: **(a + b) (a − b) / ((1 − x) (x^2 + y^2))**

Pressione: [**Insert**]

Resposta: Out[1]= $\dfrac{(a-b)(a+b)}{(1-x)(x^2+y^2)}$

Digite: **Expand**[%]

Pressione: [**Insert**]

Resposta: Out[2]= $\dfrac{a^2}{(1-x)\ (x^2+y^2)} - \dfrac{b^2}{(1-x)\ (x^2+y^2)}$

Digite: **ExpandAll**[%]

Pressione: [**Insert**]

Resposta: Out[3]= $\dfrac{a^2}{x^2-x^3+y^2-x\,y^2} - \dfrac{b^2}{x^2-x^3+y^2-x\,y^2}$

Digite: **Together**[%]

Pressione: [**Insert**]

Resposta: Out[4]= $\dfrac{-a^2+b^2}{-x^2+x^3-y^2+xy^2}$

Digite: **Factor**[%]

Pressione: [**Insert**]

Resposta: Out[5]= $\dfrac{(-a+b)\ (a+b)}{(-1+x)\ (x^2+y^2)}$

Digite: **Simplify**[%]

Pressione: [**Insert**]

Resposta: Out[6]= $\dfrac{(-a+b)(a+b)}{-x^2+x^3-y^2+x\,y^2}$

Na manipulação de séries ou de seus polinômios associados, existem ocasiões nas quais se deseja fatorar os termos segundo as potências crescentes de uma variável. Em outras ocasiões, se faz conveniente fatorar os termos que não dependem de uma determinada variável. Para realizar estas tarefas temos as funções do quadro que segue, onde **poli** é um polinômio e **expr** é uma expressão algébrica genérica.

Collect[**expr , x**]	...	Agrupa os termos com a mesma potência de **x** na expressão **expr**
FactorTerms[**poli**]		Fatora a constante comum a todos os termos de **poli**

Coefficient[**poli, expr**]	...	Devolve o fator de **expr** em **poli**
CoefficientList[**poli , x**]	...	Fornece a lista dos coeficientes de **poli,** em relação à variável **x**

PowerExpand[**poli**]	...	Transforma potências da forma $(xy)^n$ em potências do tipo $x^n y^n$.
ComplexExpand[**expr**]	...	Desenvolve **expr**, admitindo que as variáveis envolvidas são reais

Exemplo 3.7 *Inicie uma nova sessão e*

Digite: **expr = Expand[(2+a)^2 (a–2x)^2]**

Pressione: [**Insert**]

Resposta: Out[1]=

$$4\,a^2 + 4\,a^3 + a^4 - 16\,a\,x - 16\,a^2\,x - 4\,a^3\,x + 16\,x^2 + 16\,a\,x^2 + 4\,a^2\,x^2$$

Digite: **Collect[expr , x]**

Pressione: [**Insert**]

Resposta: Out[2]=

$$4\,a^2 + 4\,a^3 + a^4 + \left(-16\,a - 16\,a^2 - 4\,a^3\right)\,x + \left(16 + 16\,a + 4\,a^2\right)\,x^2$$

Digite: **poli = 4 − 39 x − 25 x^2 + 36 x^3**

Pressione: [**Insert**]

Resposta: Out[3]= $4 - 39\,x - 25\,x^2 + 36\,x^3$

Digite: **Coefficient[poli , x^2]**

Pressione: [**Insert**]

Resposta: Out[4]= –25

Desejamos destacar o comando

ComplexExpand[expr]

que tem a capacidade de fornecer as partes real e imaginária de funções complexas, tratando as variáveis que aparecem em **expr** como sendo reais.

Exemplo 3.8 *Vamos obter as partes real e imaginária da função seno complexa. Continuando a sessão anterior,*

Digite: **Expand[Sin[x + y * I]]**

Pressione: [**Insert**]

Resposta: Out[5]= Sin[x + y I]

Digite: **ComplexExpand[Sin[x + y * I]]**

Pressione: [**Insert**]

Resposta: Out[6]= Cosh[y] Sin[x] + I Cos[x] Sinh[y]

Neste exemplo o **ComplexExpand** foi capaz de desenvolver a função complexa **Sin[x + y * I]**, nos fornecendo suas partes real e imaginária. O **Expand** não realiza esta tarefa. Outro alerta: podemos usar **y * I** ou **y I**, com um espaço entre o **y** e o **I**. O Mathematica interpreta **yI** sem espaço entre o **y** e o **I**, como uma única variável que nada tem a ver com a unidade imaginária **I**.

3.6 Manipulação de polinômios

Destacamos agora outros comandos úteis na manipulação de polinômios, que denotaremos por **poli**

Exponent[**poli , x**]	=	grau de **poli** em **x**
Length[**poli**]	=	número de termos de **poli**
PolynomialGCD[**poli1 , poli2**]	=	máximo divisor comum entre **poli1** e **poli2**
PolynomialLCM[**poli1 , poli2**]	=	mínimo múltiplo comum entre **poli1** e **poli2**

O comando

PolynomialQuocient[**poli1 , poli2 , x**]

fornece o quociente da divisão de **poli1** por **poli2**, que serão tratados como polinômios na variável **x**. O comando

PolynomialRemainder[**poli1 , poli2 , x**]

fornece o resto da divisão de **poli1** por **poli2**, que serão tratados como polinômios na variável **x**.

Exemplo 3.9 *Continuando a sessão do exemplo anterior,*

Digite: **poli1 = Expand**[**(x – 1)^3 (2x + 1)^2**]

Pressione: [**Insert**]

Resposta: Out[7]= $-1 - x + 5\,x^2 + x^3 - 8\,x^4 + 4\,x^5$

Digite: **poli2 = Expand[(x – 1) (2x + 1) (x - 3)]**

Pressione: [**Insert**]

Resposta: Out[8]= $3 + 2\,x - 7\,x^2 + 2\,x^3$

Digite: **PolynomialGCD[poli1 , poli2]**

Pressione: [**Insert**]

Resposta: Out[9]= $-1 - x + 2\,x^2$

Digite: **PolynomialLCM[poli1 , poli2]**

Pressione: [**Insert**]

Resposta: Out[10]=

$$3 + 2\,x - 16\,x^2 + 2\,x^3 + 25\,x^4 - 20\,x^5 + 4\,x^6$$

Digite: **PolynomialQuotient[poli1 , poli2 , x]**

Pressione: [**Insert**]

Resposta: Out[11]= $9 + 3\,x + 2\,x^2$

Digite: **PolynomialRemainder[poli1 , poli2 , x]**

Pressione: [**Insert**]

Resposta: Out[12]= $-28 - 28\,x + 56\,x^2$

3.7 Manipulando expressões racionais

A divisão de dois polinômios é chamada de expressão racional. Como a ocorrência de expressões racionais é bastante comum e sua manipulação em geral consome bastante tempo, existem funções que nos permitem efetuar de modo automático estes desenvolvimentos. No quadro abaixo relacionamos estas funções, onde **racio** representa uma expressão racional.

Numerator[**racio**]	....	Separa o numerador da expressão **racio**
Denominator[**racio**]	....	Separa o denominador da expressão **racio**

Apart[**racio**]	....	Decompõe a função racional **racio** em frações parciais
Apart[**racio , var**]	....	Decompõe a função racional **racio** em em relação à variável **var**. As outras variáveis são tratadas como se fossem constantes.

Exemplo 3.10 *Desenvolva a sessão*

Digite: **racio = (2 x + 3 y^2) / (x^3 − 7 x^5)**

Pressione: [**Insert**]

Resposta: Out[1]= $\dfrac{2\,x + 3\,y^2}{x^3 - 7\,x^5}$

Digite: **Numerator**[**racio**]

Pressione: [**Insert**]

Resposta: Out[2]= $2\,x + 3\,y^2$

Digite: **Denominator**[**racio**]

Pressione: [**Insert**]

Resposta: Out[3]= $x^3 - 7\,x^5$

Exemplo 3.11 *Desenvolva a sessão*

Digite: **(4x^3+8x^2+7x+1) / (x^4+4x^3+7x^2+6x+2)**

Pressione: [**Insert**]

Resposta: Out[1]= $\dfrac{1 + 7\,x + 8\,x^2 + 4\,x^3}{2 + 6\,x + 7\,x^2 + 4\,x^3 + x^4}$

Digite: **Apart[%]**

Pressione: [**Insert**]

Resposta: Out[2]= $\dfrac{-2}{(1+x)^2} + \dfrac{3}{1+x} - \dfrac{1-x}{2+2\,x+x^2}$

Digite: **Clear[a , b , x , y]**

Pressione: [**Insert**]

Observe que não há resposta. Apenas aparece a mensagem In[3]:= antes do comando emitido. Com este comando, limpamos as variáveis **a**, **b**, **x, y** *por precaução.*

Digite: **numerador = (a+b)^2 (x+2y)**

Pressione: [**Insert**]

Resposta: Out[4]= $(a+b)^2(x+2y)$

Digite: **denominador = (a+b)(x+y)**

Pressione: [**Insert**]

Resposta: Out[5]= $(a+b)(x+y)$

Digite: **Expand[numerador]**

Pressione: [**Insert**]

Resposta: Out[6]= $a^2\,x + 2\,a\,b\,x + b^2\,x + 2\,a^2\,y + 4\,a\,b\,y + 2\,b^2\,y$

Digite: **Expand[denominador]**

Pressione: [**Insert**]

Resposta: Out[7]= $a\,x + b\,x + a\,y + b\,y$

Digite: **%% / %**

Pressione: [**Insert**]

Resposta: Out[6]=

$$\frac{a^2\,x + 2\,a\,b\,x + b^2\,x + 2\,a^2\,y + 4\,a\,b\,y + 2\,b^2\,y}{a\,x + b\,x + a\,y + b\,y}$$

Digite: **Cancel**[%]

Pressione: [**Insert**]

Resposta: Out[7]= $\dfrac{(a+b)\,(x+2\,y)}{x+y}$

Para expandir o numerador e o denominador de uma expressão racional **racio**, use, respectivamente,

ExpandNumerator[**racio**]

ExpandDenominator[**racio**]

3.8 Simplificando saídas intermediárias extensas

Eventualmente, os resultados intermediários podem ser extensos. Nem sempre precisamos analisá-los e, em alguns casos, até sua leitura é desnecessária. Tais resultados poderiam gerar páginas e páginas de dados inúteis. Neste caso, podemos evitar que estes resultados apareçam na tela ou produzir saídas parciais.

O controle desta saída simplificada pode ser feita de dois modos. Se colocarmos um **ponto e vírgula** no final do comando,

comando ;

os cálculos serão efetuados mas o resultado não será apresentado na tela. Outro modo de simplificar a saída, utiliza o comando

Short

que cancela parte da saída. Analisaremos seu efeito através de um exemplo.

Exemplo 3.12 *Inicie uma nova sessão e siga as instruções que seguem.*

Digite: **poli = Expand[(2 + xˆ2)ˆ20] ;**

Pressione: [**Insert**]

Observe que não houve nenhuma resposta a este comando pois ele foi terminado com um ponto e vírgula. Todavia, os cálculos, foram efetuados.

Digite: **poli //Short**

Pressione: [**Insert**]

Resposta: Out[2]= $1048576 + << 19 >> + x^{40}$

O polinômio **poli** *é muito grande e só foram apresentados o primeiro e o último termo. A expressão* $<< \mathbf{19} >>$ *indica que* 19 *parcelas foram omitidas e a saída se resumiu em uma única linha.*

Digite: **Short[poli , 3]**

Pressione: [**Insert**]

Resposta: Out[3]//Short=

$$1048576 + 10485760\, x^2 + 49807360\, x^4 + 149422080\, x^6 + 317521920\, x^8 + << 14 >> + 40\, x^{38} + x^{40}$$

Observe: A resposta foi apresentada em três linhas, contando com a linha ocupada por Out[3]=. A expressão $<< \mathbf{14} <<$ *indica que* 14 *parcelas foram omitidas na expressão completa do polinômio.*

Em resumo, para executar um comando sem que o Mathematica apresente o resultado na tela, termine-o com um ponto e vírgula, como no modelo que segue

comando ;

Para executar um comando e observar apenas um resultado parcial, use

comando //Short ou **Short[comando]**

Para executar um comando e observar parte do resultado, controlando o número de linhas da saída, use

Short[comando , num_linhas]

onde **num_linhas** é o número de linhas que serão utilizadas na apresentação do resultado dos cálculos. Para sermos informados sobre o número total de parcelas contidas num determinado resultado, usamos o

Length[comando]

Exemplo 3.13 *Siga as etapas abaixo, iniciando uma nova sessão*

Digite: **poli = Expand[(4 − x)^30] ;**

Pressione: [**Insert**]

Observe que não há resposta a este comando pois há um ponto e vírgula no seu final.

Digite: **Length[poli]**

Resposta: Out[2]= 31

Este é o número de parcelas de **(4–x)^30**

3.9 Apresentando na tela o valor das variáveis

Para obter o valor de uma variável, basta digitar o seu nome e pressionar o **Insert**. Desejando obter a lista de definições de uma variável, pode-se digitar o seu nome, precedido pelo sinal de interrogação e pressionar a tecla **Insert**. As sintaxes destes comandos são

nome_da_variável

ou

?nome_da_variável

Pode-se também emitir o comando

Print[var1 , var2 , ... , vark]

para obter na tela os valores das variáveis **var1**, **var2**, **...**, **vark**.

3.10 Linhas com múltiplos comandos

Quando desejarmos efetuar diversos cálculos intermediários cujos resultados não precisam ser examinados, podemos efetuá-los em uma única linha, separando-os com pontos e vírgulas

comando1 ; comando2 ; comando3 ; ...

Exemplo 3.14 *Desenvolva a sessão*

Digite: **a = 1 ; b = 2 ; c = 3 ;**

Pressione : [**Insert**]

Verifique que não houve resposta a este comando.

Digite: **(a + b) * c**

Pressione: [**Insert**]

Resposta: Out[2]= 9

Pode-se emitir múltiplos comandos entre parênteses e associar o valor do último comando executado a uma variável. A sintaxe de tal **comando múltiplo** é

var = (comando1 ; comando2 ; ... ; comandok)

que executará o **comando1**, o **comando2**, **...**, o **comandok** e armazenará o resultado do **comandok** na variável **var**.

Exemplo 3.15 *Desenvolva o exemplo, iniciando uma nova sessão*

Digite: **a = (b = 3 ; c = 4 ; d = bˆc)**

Pressione: [**Insert**]

Resposta: Out[1]= 81

Digite: **Print[a] ; d**

Resposta: 81

Resposta: Out[2]= 81

Surgem duas vezes o valor 81 que são, respectivamente, os valores de **a** *e* **d**. *Percebemos então que o último comando* **d = bˆc** *foi executado, atribuindo o valor* 81 *a* **d** *e para completar o comando composto, o último valor calculado foi atribuído à variável* **a**.

Capítulo 4

Listas, vetores e matrizes

Uma **lista** é um conjunto ordenado que, no ambiente do Mathematica, é denotado da seguinte forma

{ a, b, ..., z }

onde **a**, **b**, **...**, **z**, são chamados de **elementos da lista**. Uma lista é delimitada por um abrir e um fechar chaves sendo os elementos separados por vírgulas. Os elementos de uma lista podem ser números, expressões algébricas, regras de atribuições, etc. Os elementos podem ser inclusive outras listas. Quando um elemento de uma lista for outra lista, diremos que este elemento é uma **sub-lista**.

Exemplo 4.1 *Apresentamos abaixo algumas listas.*

{ *a, b, c* }

{ *1, 2, 3, 4, 5, 6, 7, 8,* { *9, 0*} },

{ {*1, 2, 3*}, {*4, 5, 6*} }

Nesta última lista, cada elemento é uma sub-lista, sendo **{*1, 2, 3*}** *e* **{*4, 5, 6*}** *elementos da lista.*

4.1 Convenção

O leitor adquiriu uma boa familiaridade com o Mathematica. Por este motivo, em nosso exemplos, iremos substituir as diretrizes

Digite: **comando**

Pressione: [**Insert**]

por

Comande: **comando**

Omitiremos algumas saídas quando as julgarmos desnecessárias. Em lugar de

Resposta: *Out[n]*= resultado de número n

escreveremos apenas

Resposta: resultado de número n

4.2 Criando listas

Pode-se **criar** listas digitando-as diretamente pelo teclado ou emitindo comandos que as criam automaticamente. O comando

Table[f[n] , { n , nmin , nmax , delta }]

cria a lista

{ f[n1] , f[n2] , ... , f[nk] }

onde

n1 = nmin ,
n2 = nmin + delta ,
n3 = nmin + 2 · delta ,
...,
nk = nmin + k · delta ,

sendo **k** o maior inteiro que satisfaz à desigualdade **nmin + k·delta** $\leq$ **nmax**
O comando

Table[f[n] , { n , nmin , nmax }]

é equivalente ao comando anterior com **delta = 1** e

Table[**f**[**n**] , { **n** , **nmax** }]

é equivalente ao comando anterior com **nmin = 1**. Finalmente,

Table[**x** , { **n** }]

cria a lista { **x** , **x** , ... , **x** } que contém **n** cópias de **x**.

Exemplo 4.2 *Execute os comandos e acompanhe os resultados abaixo.*

Comande: ***Table[n , { n , 1 , 2 , 0.3 }]***

Resposta: { 1, 1.3, 1.6, 1.9 }

Comande: ***Table[2 n , { n , 3 , 8 , 2 }]***

Resposta: { 6, 10, 14 }

Comande: ***Table[2 n , { n , 3 , 8 }]***

Resposta: {6, 8, 10, 12, 14, 16 }

Comande: ***Table[2 n , { n , 8 }]***

Resposta: { 2, 4, 6, 8, 10, 12, 14, 16 }

Comande: ***Table[n , { 3 }]***

Resposta: { *n, n, n* }

Comande: ***Table[Exp[x] , { x , 1 , 4}]***

Resposta: { E, E^2, E^3, E^4 }

Comande: ***N[%]***

Resposta: { *2.71828 , 7.38906 , 20.0855 , 54.5982* }

Dispomos de outros comandos para criar listas

Array[**f , n**]	....	cria a lista { f[1] , f[2] , ... , f[n] }
Range[**n**]	....	cria a lista { 1 , 2 , ... , n }
Range[**n1 , n2**]	....	cria a lista { n1 , n1+1 , ... , n2 }
Range[**n1 , n2 , dn**]	....	cria a lista { n1 , n1+dn , ... , n2 }

Pode-se obter o número de elementos de uma **lista** com o comando

Length[**lista**]

e apresentar os seus elementos em uma coluna com o

ColumnForm[**lista**]

4.3 Posição e nível

Cada elemento em uma lista ocupa uma **posição**. O primeiro elemento ocupa a posição 1, o segundo a posição 2 e assim por diante.

Uma lista tem um único **nível** quando nenhum dos seus elementos for uma lista. Quando os elementos de uma lista forem sub-listas, diz-se que a lista tem mais de um nível. Os elementos da lista ocupam o nível 1 e os elementos de uma sub-lista ocupam o nível 2 da lista. Se um elemento de uma sub-lista for outra sub-lista, diremos que seus elementos ocupam o nível 3 e assim por diante.

Exemplo 4.3 *A lista* ***lista1 = { x, y }*** *tem um único nível, com* ***x*** *ocupando a posição* ***1*** *e* ***y*** *ocupando a posição* ***2****. A lista*

lista2 = { {11, 12 }, {21, 22, 23, 24}, {31, 32, 33, 34} }

tem dois níveis. Os elementos

{11, 12}, {21, 22, 23, 24}, {31, 32, 33, 34}

são elementos do nível ***1*** *e ocupam, respectivamente, as posições* ***1, 2, 3.*** *Os elementos*

11, 12, 21, 22, 23, 24, 31, 32, 33, 34

são elementos do nível ***2*** *e ocupam, respectivamente, as posições*

{1, 1}, {1, 2}, {2, 1}, {2, 2}, {2, 3}, {2, 4}, {3, 1}, {3, 2}, {3, 3}, {3, 4}

onde observamos que o primeiro número de cada par fornece a posição da sub-lista na lista e o segundo a posição do elemento na sub-lista.

Para verificar se um determinado **elemento** está ou não em uma **lista**, dispomos das funções lógicas

MemberQ[lista , elemento]

que retorna o valor lógico **True** quando o **elemento** pertence à **lista** e

FreeQ[lista , elemento]

que retorna o valor lógico **True** quando o **elemento** não pertence à **lista**.

Temos uma função para fornecer a posição de um elemento na lista e outra para contar o número de ocorrências de um elemento na lista. A função

Position[lista , elemento]

fornece a posição ocupada pelo **elemento** na **lista** e a função

Count[lista , elemento]

fornece o número de vezes que o **elemento** aparece na **lista.**

Exemplo 4.4 *Inicie uma nova sessão do Mathematica e*

Comande: ***MemberQ[{ 1, 2, 3, 4 } , 4]***

Resposta: True

Comande: ***FreeQ[{1, 2, 3, 4 } , 6]***

Resposta: True

Comande: ***m = { { 1, 2, 3 } , { 3, 2, 1 } , { 4, 5, 6 } }***

Resposta: { { 1, 2, 3 } , { 3, 2, 1 }, { 4, 5, 6 } }

Comande: ***Position[m, 1]***

Resposta: { { 1, 1 }, { 2, 3 } }

O elemento 1 *está na posição* { *1, 1* } *e na posição* { *2, 3* } *da lista, isto é, está ocupando a primeira posição na primeira sub-lista e a terceira posição na segunda sub-lista.*

Comande: ***Count[{1, 2, 3, 1, 4, 5, 1, 6 } , 1]***

Resposta: 3

4.4 Operações com listas

Podemos somar, subtrair, multiplicar e dividir listas de acordo com as regras que seguem

$$\begin{aligned} \{a,b\} + \{c,d\} &= \{a+b, c+d\} \\ \{a,b\} - \{c,d\} &= \{a-b, c-d\} \\ \{a,b\} * \{c,d\} &= \{a*b, c*d\} \\ \{a,b\} \,/\, \{c,d\} &= \{a \,/\, b, c \,/\, d\} \end{aligned}$$

Em resumo, podemos efetuar com listas de mesmo tamanho, todas as operações permitidas para os números. Pode-se multiplicar lista por **expressões escalares** (expressão escalar é aquela que não é uma lista) de acordo com as regras

$$\begin{aligned} a+\{c,\,d\} &= \{c,\,d\}+a=\{a+c,\,a\ +\ d\} \\ a*\{c,\,d\} &= \{c,\,d\}*a=\{a*c,\,a\ *\ d\} \\ a-\{c,\,d\} &= \{a-c,\,a-d\} \\ \{c,\,d\}-a &= \{c-a,\,d-a\} \\ a\,/\,\{c,\,d\} &= \{a\,/\,c,\,a\,/\,d\} \\ \{c,\,d\}\,/\,a &= \{c\,/\,a,\,d\,/\,a\} \end{aligned}$$

Nota 4.1 *Estas operações se aplicam a listas maiores e a generalização é evidente.*

Exemplo 4.5 *Siga as instruções*

Digite: ***{ 1, 3 } * { 2, 4 } + 5***

Resposta: { 7, 17 }

Digite: ***a = { 4, 5, 6 } ; b = { 1, 2, 3 } ;***

Digite: ***Print[a + b , a – b , a * b , a / b]***

Resposta:

$$\{5,7,9\}\quad\{3,3,3\}\quad\{4,10,18\}\quad\{4,\frac{5}{2},2\}$$

Podemos usar a função **Apply** para realizar operações com lista. A função

Apply[f , { a1 , a2 , ... }]

aplica a função **f** na lista **{ a1, a2, ... }**, fornecendo **f[a1, a2, ...].**

Exemplo 4.6 *Vamos somar e multiplicar todos os elementos da lista* ***lis = { x , y , z , w }***

Comande: ***lis = { x , y , z , w }***

Resposta: { x , y , z , w }

Comande: ***Apply[Plus , lis]***

Resposta: $w + x + y + z$

Comande: ***Apply[Times , lis]***

Resposta: $w\ x\ y\ z$

que é o produto de x por y por z e por w. A ordem em que aparecem as variáveis segue um padrão interno.

4.5 Vetores

Listas com um único nível representam vetores. Para indicar o vetor (a, b, c) usamos a lista

{ a , b , c }.

Para multiplicar o vetor **vet** pelo escalar **k**, basta digitar

k vet ou **k * vet**

e pressionar a tecla [**Insert**]. Para fazer o produto escalar entre dois vetores **vet1** e **vet2** basta comandar

vet1 . vet2

Existe um pacote auxiliar que, se carregado com o comando

< < < Calculus‘VectorAnalysis‘

permite o cálculo do produto vetorial, do gradiente, do divergente, do rotacional e do laplaciano. Para calcular o produto vetorial de dois vetores **vet1** e **vet2**, comande

CrossProduct[vet1 , vet2]

Seja **f(x , y , z)** uma função real. O comando

Grad[f[x , y , z]]

calcula o gradiente de **f**. Sendo

$$\mathrm{v}(x,y,z) = (\ \mathrm{v1}(x,y,z),\ \mathrm{v2}(x,y,z),\ \mathrm{v3}(x,y,z)\ )$$

uma função vetorial então

Div[{ v1[x, y, z] , v2[x, y, z] , v3[x, y, z] }]

calcula o divergente de **v** e

Curl[{ v1[x, y, z] , v2[x, y, z] , v3[x, y, z] }]

calcula o seu rotacional.

Exemplo 4.7 *Inicie uma nova sessão com o Mathematica e*

Comande: $<<$ ***Calculus`VectorAnalysis`***

Comande: ***CrossProduct[{ 1, 2, 3 } , { 3, −1, 2 }]***

Resposta: { *7 , 7 , –7* }

Comande: ***Grad[x^2 + y]***

Resposta: { *2 x , 1 , 0* }

Comande: ***Div[{ x , y^2 , x + 2 z }]***

Resposta: 3 + 2 y

Comande: ***Curl[{ x , y^2 , x + 2 z }]***

Resposta: { *0 , –1 , 0* }

Logo que se carrega este pacote, ele assume automaticamente que se está usando o sistema de coordenadas cartesianas e que as variáveis **x**, **y** e **z** serão usadas para designar as coordenadas. Uma boa política consiste em emitir um comando **Clear[x, y, z]**, antes de carregar este pacote. Este pacote pode efetuar os cálculos em coordenadas curvilíneas. Num capítulo à parte, descreveremos o procedimento a ser seguido para trabalhar com as coordenadas curvilíneas.

4.6 Matrizes

Para representar uma matriz no Mathematica, usamos uma lista com dois níveis, onde cada elemento é uma sub-lista. Cada sub-lista representa uma linha da matriz. Para criar a matriz

$$m = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$$

digitamos

m = { { a11 , a12 , a13 } , { a21 , a22 , a23 } }

Este comando cria a matriz e a grava na variável **m**.

Se a_{ij} for uma expressão em i e j, o comando

Table[aij , { i , k } , { j , n }]

cria a matriz

$$(a_{ij})_{k\times n} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{pmatrix}$$

Se $a[i, j]$ for uma função de i e de j, o comando

Array[a , { k , n }]

cria a matriz de ordem $k \times n$, cujo elemento da linha i coluna j é $a[i, j]$.

O comando

IndentityMatrix[n]

cria uma matriz identidade **n**×**n** e

DiagonalMatrix[lista]

cria uma matriz diagonal, cujos elementos da diagonal principal são fornecidos pela **lista**. Para obter o número de elementos de uma **matriz**, use

Dimensions[matriz]

e, para visualizá-la na forma matricial, comande

MatrixForm[matriz]

4.7 Operações com matrizes

Se **mat**, **mat1** e **mat2** forem matrizes, **k** for um número inteiro, racional, real ou complexo e **n** for um número inteiro, então

k * mat	→	calcula o produto de **k** por **mat**
mat1 . mat2	→	calcula o produto de **mat1** por **mat2**

Transpose[**mat**]	→	calcula a transposta de **mat**
MatrixPower[**mat , n**]	→	calcula a **n**-ésima potência de **mat**

Inverse[**mat**]	→	calcula a inversa de **mat**
Det[**mat**]	→	calcula o determinante de **mat**

Eigenvalues[**mat**]	→	calcula os auto valores de **mat**
Eigenvectors[**mat**]	→	calcula os auto vetores de **mat**

Quando todos os elementos de uma matriz forem inteiros, os comandos **Eigenvalues**[**mat**] e **Eigenvectors**[**mat**] tentam obter o valor exato dos auto valores e auto vetores. Tais comandos nem sempre serão capazes de obter os valores exatamente. Em tais casos, pode-se usar

Eigenvalues[**N**[**mat**]]

Eigenvectors[**N**[**mat**]]

para obter os valores numéricos aproximados dos autovalores.

Exemplo 4.8 *Vamos construir a matriz*

$$\begin{pmatrix} 3 & 0 & 0 & 2 \\ 0 & 0 & -3 & 0 \\ 1 & 1 & -2 & 3 \\ 2 & 0 & 0 & 3 \end{pmatrix}$$

e calcular alguns itens para ilustrar os comandos acima.

Comande: ***m = { {3,0,0,2}, {0,0,–3,0}, {1,1,–2,3}, {2,0,0,3} }***

Resposta: { {3, 0, 0, 2}, {0, 0, −3, 0}, {1, 1, −2, 3}, {2, 0, 0, 3} }

Comande: ***MatrixForm[m]***

Resposta:

$$\begin{array}{cccc} 3 & 0 & 0 & 2 \\ 0 & 0 & -3 & 0 \\ 1 & 1 & -2 & 3 \\ 2 & 0 & 0 & 3 \end{array}$$

Comande: ***Det[m]***

Resposta: 15

Comande: ***Eigenvalues[m]***

Resposta:

$$\{\, 1,\ 5,\ \frac{-2 - 2\,\mathrm{I}\,\mathrm{Sqrt}[\,2\,]}{2},\ \frac{-2 + 2\,\mathrm{I}\,\mathrm{Sqrt}[\,2\,]}{2}\,\}$$

Comande: ***Eigenvalues[N[m]]***

Resposta: { *5.* , *–1. + 1.41421 I* , *–1. – 1.41421 I* , *1.* }

Comande: ***Eigenvectors[m]***

Resposta:

$$\begin{array}{r} \{\,\{\,-3,\ -3,\ 1,\ 3\},\ \{\,19,\ -6,\ 10,\ 19\,\}, \\ \{\,0,\ -\mathrm{I}\,(\mathrm{I} + \mathrm{Sqrt}[\,2\,])\,,\ 1,\ 0\,\}, \\ \{\,0,\ \mathrm{I}\,(-\mathrm{I} + \mathrm{Sqrt}[\,2\,])\,,\ 1,\ 0\,\}\,\} \end{array}$$

Exemplo 4.9 *Vamos construir a matriz*

$$m = \begin{pmatrix} -1 & 0 & 3 & 0 & 1 \\ 0 & -1 & -2 & 2 & -2 \\ 2 & -1 & -2 & 0 & 1 \\ 1 & -1 & -2 & -2 & -2 \\ 2 & -1 & -1 & -1 & 0 \end{pmatrix}$$

e calcular seus auto valores.

Comande: ***m = { { −1, 0, 3, 0, 1 } , { 0, −1, −2, 2, −2 } , { 2, −1, −2, 0, 1 } , { 1, −1, −2, −2, −2 } , { 2, −1, −1, −1, 0 } } ;***

Comande: ***Eigenvalues[m]***

Mensagem:

Eigenvalues : : eival :

Unable to find all roots of the characteristic polynomial.

Neste caso, os autovalores, por serem raizes de um polinômio do quinto grau, não puderam ser obtidos exatamente. Para obter os valores aproximados dos autovalores, proceda como segue.

Comande: ***Eigenvalues[N[m]]***

Resposta:

$$\begin{aligned} \{ & -3.52982 + 0.760526\ I, \\ & -3.52982 - 0.760526\ I, \\ & 2.67727, \\ & -0.808811 + 0.34543\ I, \\ & -0.808811 - 0.34543\ I \ \} \end{aligned}$$

4.8 Notação

Vimos que no Mathematica, os parênteses **( )** agrupam as operações, os colchetes [ ] delimitam os argumentos das funções, como em f[x], as chaves **{ }** delimitam as listas, como em { 1, 2, 3 } e colchetes duplos [[ ]] são usados para trabalhar com os índices das listas, como em a[[n]].

4.9 Extrair e manipular partes de uma lista

Uma lista é composta de vários elementos. Pode-se modificar os elementos de uma lista, extrair parte de seus elementos ou incluir novos elementos.

Podemos extrair partes de uma lista, usando o comando

Part[lis , pos] ou **lis[[pos]]**

onde **lis** é uma lista e **pos** é um número inteiro que fornece a posição do elemento que se deseja extrair da lista. O comando

Part[{a1 , a2 , a3 , ... , an , ... } , n]

nos fornece **an**.

Exemplo 4.10 *Vamos retirar partes da lista* ***a = { 1, {2, 3, 4}, 5, 6}.***

Comande: ***a = { 1, {2, 3, 4}, 5, 6}***

Resposta: { 1, { 2, 3, 4 }, 5, 6 }

Comande: ***Part[a , 2]*** *ou* ***a[[2]]***

Resposta: { 2, 3, 4 }

Comande: ***Part[a , 4]*** *ou* ***a[[4]]***

Resposta: 6

Podemos formar listas com partes de outra lista. O comando

Part[lista , lista_de_posições]

ou

lista[[lista_de_posições]]

forma uma outra lista usando os elementos da **lista** original. A nova lista é formada pelos elementos da **lista** cujas posições forem especificadas na **lista_de_posições.**

Exemplo 4.11 *Sendo* ***vetor = { a, b, c, d, e, f, g, h }****, vamos formar outras listas com suas partes. Para limpar eventuais valores atribuídos anteriormente para as variáveis* ***a****,* ***b****,* ***c****,* ***d****,* ***e, f****,* ***g*** *e* ***h****,*

Comande: ***Clear[a, b, c, d, e, f, g, h]***

Comande: ***vetor = { a, b, c, d, e, f, g, h }***

Comande: ***Part[vetor , { 1, 3, 1 }]***

Resposta: { a, c, a}

Os comandos

Part[vetor , { 1, 3, 1 }]

vetor[[{ 1, 3, 1 }]]

{ a, b, c, d, f, g, h } [[{ 1, 3, 1 }]]

são equivalentes.

Quando um elemento de uma lista é uma sub-lista, é possível extrair um de seus elementos fornecendo um terceiro argumento no comando **Part**. Este terceiro argumento especificará a posição do elemento na sub-lista.

Exemplo 4.12 *Observe a seqüência de comandos.*

Comande: ***lis = { 1 , 2 , { 31, 32, 33, 34, 35 } , 4 , 5 , 6 , 7 }***

Comande: ***Part[lis , 3 , 2]***

Resposta: 32

Comande: ***Part[lis , 3 , { 1 , 4 , 5 }]***

Resposta: { 31, 34, 35 }

O comando

Part[lista , n] = valor

ou

lista[[**n**]] = **valor**

faz com que o n-ésimo elemento da **lista** assuma o **valor** especificado.

Exemplo 4.13 *Consideremos a lista do exemplo anterior.*

Comande: ***Clear[x]***

Comande: ***Part[lis , 3] = x***

Este comando é equivalente a ***lis[[3]] = x****.*

Resposta: x

Comande: ***lis***

Resposta: { 1 , 2 , x , 4 , 5 , 6 , 7 }

Observe que a sub-lista ***{ 31, 32, 33, 34, 35 }*** *que ocupava a posição de número* ***3*** *na lista foi substituída por* ***x****.*

Em seguida, apresentamos outros comandos para manipular listas.

Part[**lista , –n**]	→	**n**-ésimo elemento da **lista** a partir do final
lista[[**–n**]]	→	equivalente ao comando anterior

Take[**lista , n**]	→	fornece os **n** primeiros elementos da **lista**
Take[**lista , –n**]	→	fornece os **n** últimos elementos da **lista**

First[**lista**]	→	fornece o primeiro elemento da **lista**
Last[**lista**]	→	fornece o último elemento da **lista**
Rest[**lista**]	→	fornece a **lista** sem seu primeiro elemento

4.10 Inserir e remover elementos

Para inserir, remover e modificar os elementos de uma lista, cujo nome é **lis**, dispomos das funções

Prepend[**lis, elem**]	→	inclui **elem** no início da **lis**
Append[**lis , elem**]	→	inclui **elem** no final da **lis**

Insert[**lis , el , p**]	→	insere **el** na posição **p** da lista
Insert[**lis , el , –p**]	→	insere na posição **p** contando do final
Insert[**lis, el, {p1, p2, ...}**]	→	insere **el** nas posições **p1**, **p2**, ...

Delete[**lis , p**]	→	**lis** sem o elemento da posição **p**
Delete[**lis, {n, m, ...}**]	→	**lis** sem os elementos das posições **n**, **m**, ...

Drop[**lis** , **n**]	→	**lis** sem seus primeiros **n** elementos
Drop[**lis** , –**n**]	→	**lis** sem seus últimos **n** elementos
Drop[**lis**,**{n,m}**]	→	**lis** sem os elementos entre as posições **n** e **m**

ReplacePart[**lis, el, p**]	→	substitui por **el** o elemento que está na posição **p** da **lis**
ReplacePart[**lis, el,** –**p**]	→	igual ao anterior, contando a partir do final da lista

O comando

ReplacePart[**lis, el , { p1 , p2 , ... }**]

substitui por **el** os elementos das posições **p1**, **p2**, ... da **lis** e, quando **lis** tem mais de um nível, o comando

ReplacePart[**lis , el , { { p1 , q1 } , { p2 , q2 } , ... }**]

substitui por **el**, todos os elementos que estiverem ocupando as posições **{p1, q1}**, **{p2, q2}**, ...

Exemplo 4.14 *Defina a matriz*
m = { { 11, 12, 13, 14 }, { 21, 22, 23, 24 } } *e*

Comande: ***ReplacePart[m , x , { {1 , 4} , {2 , 3} }]***

Resposta: { { 11, 12, 13, *x* }, { 21, 22, *x*, 24 } }

Podemos concatenar duas ou mais listas com o comando

Join[lista1 , lista2 , ...]

Exemplo 4.15 *Para concatenar as listas* ***lis1 = { a, b, c }***, ***lis2 = { 1, {21, 22}, 3 }*** *e* ***lis3 = { x, y }***,

Comande: ***Join[lis1 , lis2 , lis3]***

Resposta: { *a , b , c , 1 ,* { *21 , 22* } *, 3 , x , y* }

4.11 Reordenando listas

Para reordenar os elementos de uma **lista**, use

Sort[lista]	→	reordena a **lista** em uma ordem padrão
Reverse[lista]	→	inverte a ordem dos elementos da **lista**
RotateLeft[lista]	→	gira a lista uma posição à esquerda
RotateRight[lista]	→	gira uma posição à direita
RotateLeft[lista, n]	→	gira **n** posições à esquerda
RotateRight[lista, n]	→	gira **n** posições à direita

Para verificar se uma **lista** está disposta de acordo com uma ordem previamente estabelecida pelo sistema, use

OrderedQ[lista]

Exemplo 4.16 *Defina a lista* ***lis = { 1, 2, 3, 4, 5 }***

Comande: ***Reverse[lis]***

Resposta: { 5, 4, 3, 2, 1 }

Comande: ***RotateLeft[lis]***

Resposta: { 2, 3, 4, 5, 1}

Comande: ***RotateRight[lis]***

Resposta: {5, 1, 2, 3, 4 }

Comande: ***Sort[{ Mateus, Marcos, Lucas, João }]***

Resposta: { *João, Lucas, Marcos, Mateus* }

Comande: ***OrderedQ[%]***

Resposta: True

4.12 Listas aninhadas

No manuseio de listas, é comum a obtenção de listas aninhadas, isto é, listas cujos elementos são outras listas. As listas não aninhadas serão denominadas simples. Pode-se agrupar os elementos de uma lista simples em diversas listas aninhadas com a função **Partition**.

Partition[**lista, n**]	→	agrupa os elementos da **lista** formando sub-listas contendo **n** elementos
Partition[**lista, n, k**]	→	agrupa os elementos da **lista** formando sub-listas de **n** elementos. Cada nova sub-lista se inicia com um elemento **k** posições à direita do primeiro elemento que iniciou a sub-lista anterior.

Exemplo 4.17 *Defina a lista* ***lis = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 }***

Comande: ***Partition{ lis, 4 }***

Resposta: { { 1, 2, 3, 4 } , { 5, 6, 7, 8 } }

Note que os elementos 9 *e* 0 *foram eliminados, uma vez que não é possível incluí-los em uma lista com quatro elementos.*

Comande: ***Partition{ lis, 4, 2 }***

Resposta: { { 1, 2, 3, 4 } , { 3, 4, 5, 6 } , { 5, 6, 7, 8 } , { 7, 8, 9, 0 } }

Para obter o resultado contrário, temos

Flatten[lista]	→	nivela todos os níveis da **lista**
Flatten[lista, n]	→	nivela os **n** níveis superiores da lista
FlattenAt[lista, k]	→	nivela a sub-lista que ocupa a posição **k**
FlattenAt[lista, {k, j}]	→	no elemento **k** da lista, nivela a sub-lista que ocupa a posição **j**

Exemplo 4.18 *Defina a lista* ***lis = { {a, b}, { c, d}, { {1, 2}, {3, 4} } }***

Comande: ***Flatten[lis]***

Resposta: { *a, b, c, d, 1, 2, 3, 4* }

Comande: ***Flatten[lis , 1]***

Resposta: { *a, b, c, d,* {*1, 2*}, {*3, 4*} }

Comande: ***FlattenAt[lis , 3]***

Resposta: { {*a, b*}, {*c, d*}, {*1, 2*}, {*3, 4*} }

Comande: ***FlattenAt[lis , { 3, 2}]***

Resposta: { {*a, b*}, {*c, d*}, { {*1, 2*}, *3, 4* } }

4.13 Conjuntos

Podemos usar as listas para representar conjuntos. Se **conjunto1**, **conjunto2**, **...**, forem listas que representam conjuntos, o comando

Union[conjunto1 , conjunto2 , ...]

faz a **união** destes conjuntos, eliminando os elementos comuns, reordenando os restantes e gerando uma nova lista que representará a união dos conjuntos. Para fazer a **interseção** de dois ou mais conjuntos, use a função

Intersection[conjunto1 , conjunto2 , ...]

e, para obter o **complemento** de um **conjunto** em relação a um conjunto **universo**, use o

Complement[universo , conjunto]

Exemplo 4.19 *Defina as listas* $r = \{ b, a, c \}$ *e* $s = \{ a, d, c \}$.

Comande: ***Union[r, s]***

Resposta: { *a, b, c, d* }

Comande: ***Intersection[r, s]***

Resposta: { *a, c* }

Comande: ***Complement[{ a, b, c, d, e, f }, r]***

Resposta: { *d, e, f* }

4.14 Operações combinatórias

Para obter todas as **permutações** de uma **lista**, use

Permutations[lista]

e, para verificar se a **lista** é uma permutação **par** ou **ímpar** em relação a uma ordem interna pré-estabelecida, use

Signature[lista]

que retorna o valor 1 quando **lista** for uma permutação par e –1 quando for ímpar. Pode-se obter todas as **combinações** dois a dois dos elementos de **lista1** e **lista2**, com o comando

Outer[List , lista1 , lista2]

Neste comando, se fornecermos três listas,

Outer[List , lista1 , lista2 , lista3]

obteremos todas as combinações três a três, sendo um elemento de cada lista

Exemplo 4.20 *Construa as listas* ***lista1 = { a, b }*** *e* ***lista2 = { 1, 2, 3 }***

Comande: ***Permutations[lista2]***

Resposta: $\{ \{1,2,3\}, \{1,3,2\}, \{2,1,3\}, \{2,3,1\}, \{3,1,2\}, \{3,2,1\} \}$

Comande: ***Signature[{ 2, 1, 3 }]***

Resposta: -1

Comande: ***Outer[List, lista1, lista2]***

Resposta: { { {*a,1*} , {*a,2*} , {*a,3*} } , { {*b,1*} , {*b,2*} , {*b,3*} } }

Comande: ***Outer[List, lista1, lista1, lista1]***

Resposta: { { { {*a,a,a*}, {*a,a,b*} } , { {*a,b,a*}, {*a,b,b*} } },
{ { {*b,a,a*}, {*b,a,b*} } , { {*b,b,a*}, {*b,b,b*} } } }

Capítulo 5

Funções

5.1 Definindo funções

Além de inúmeras funções internas o Mathematica permite que o usuário defina suas próprias funções. Para **definir** uma função $f(x, y, \ldots)$, usamos a sintaxe

f[x_ , y_ , ...] = expressão de x, y, ... atribuída a f

ou

f[x_ , y_ , ...] := expressão de x, y, ... atribuída a f

A diferença consiste no sinal de atribuição. Numa se usa o sinal de **atribuição imediata** e na segunda o sinal de **atribuição com retardo**. Na primeira, a **expressão** do lado direito é calculada, simplificada e o resultado é atribuído a **f[x_, y_, ...]**. Na segunda, a expressão do lado direito é atribuída a **f[x_, y_, ...]** tal como foi digitada, sem nenhuma simplificação ou cálculo. Na próxima seção destacaremos melhor as diferenças.

Observe a sublinha à direita de cada argumento em **f[x_ , y_ , ...]**, que são obrigatórias na definição da **f**. No teclado, a **sublinha** ocupa a parte superior da tecla que contém o sinal de subtração. Quando formos usar a função **f**, as variáveis **x**, **y**, **...**, bem como seus valores, devem ser digitados sem a sublinha.

Exemplo 5.1 *Para definir a função* $f(x) = x^2 + 2x - 4$,

Comande: ***f[x_] = x^2 + 2x - 4***

Resposta: $-4 + 2x + x^2$

Para usar esta definição e calcular $f(3)$ *e* $f(t+1)$,

Comande: ***f[3]***

Resposta: 11

Comande: ***f[t + 1]***

Resposta: $-4 + 2(1+t) + (1+t)^2$

Observe agora a diferença. Vamos definir ***h*** *sem colocar a sublinha à direita da variável.*

Comande: ***h[x] = x^2 + 2x - 4***

Resposta: $-4 + 2x + x^2$

Comande: ***h[x]***

Resposta: $-4 + 2x + x^2$

Comande: ***h[1]***

Resposta: h[1]

Comande: ***h[y]***

Resposta: h[y]

A função ***h*** *está definida apenas para o argumento* ***x****, enquanto que* ***f*** *está definida para qualquer valor de seu argumento.*

5.2 Obter informações sobre uma função ou variável

Para obter informações a respeito da definição de uma função **f** ou variável **var**, use

? f	ou	**? var**

Exemplo 5.2 *Vamos definir* ***f*** *como função e variável.*

Comande: ***Clear[f]***

Comande: ***f[x] = (3/2) * x – 0.05 * Tan[x]***

Resposta: $\frac{3x}{2} - 0.05\,\mathrm{Tan}\,[x]$

Comande: ***f = Sin[3 x]***

Resposta: Sin[3 x]

Comande: ***?f***

Resposta: Global'f

*f = Sin[3 * x]*

*f[x] = (3 * x) / 2 – 0.05 * Tan[x]*

O ***f*** *recebeu duas definições distintas e ambas ficarão gravadas na tabela de definições do Mathematica até o final da sessão ou até serem removidas com o* ***Clear****.*

5.3 Limpar uma função ou variável

Sempre que o usuário definir uma função ou variável em uma sessão do Mathematica, ela perdurará durante toda a sessão ou até que um comando **Clear** limpe a definição. Ao finalizar a sessão, todas as variáveis e funções definidas pelo usuário se perdem. O usuário deve se precaver contra o uso indevido de variáveis ou funções que tenham sido definidas previamente durante a sessão. O uso de variáveis ou funções que foram definidas anteriormente pode causar um resultado indesejado. Uma boa norma ao utilizar variáveis e funções consiste em **limpar** o seu conteúdo sempre que ele não for mais utilizado. Para eliminar a definição de um conjunto de funções e variáveis da tabela de definições do Mathematica, use

Clear[nome1 , nome2 , ...]

onde **nome1**, **nome2**, **...** são os nomes das variáveis e funções cujas definições queremos eliminar.

Exemplo 5.3 *Continuando o exemplo anterior,*

Comande: ***Clear[f , y]***

Comande: ***f[x_] = Cos[x]***

Resposta: Cos[x]

Comande: ***y = Pi***

Resposta: Pi

Comande: ***f[y]***

Resposta: -1

Comande: ***? y***

Resposta: Global 'y

$y = Pi$

Comande: ***Clear[y]***

Comande: ***? y***

Resposta: Global 'y

5.4 Regras de atribuição global

Este é o momento oportuno para discutirmos as **regras de atribuição**. Existem dois processos para se atribuir um valor a uma variável ou definir uma função

func_ou_var = definição da função ou variável

ou

func_ou_var := definição da função ou variável

No primeiro caso temos a atribuição imediata, na qual a expressão do lado direito do sinal de atribuição (=) é calculado, simplificado e o resultado é atribuído à função ou variável do lado esquerdo. No segundo caso, temos a atribuição com retardo (:=). A expressão do lado direito do sinal := é atribuída à função ou à variável tal como foi digitada sem nenhum cálculo ou simplificação. Quando posteriormente se solicita o valor da variável ou da função, a expressão do lado direito é calculada, com os valores vigentes das variáveis naquele momento.

As atribuições acima perduram durante toda a sessão ou até que se limpe a definição com o comando **Clear**. Graças a esta propriedade de perdurar até o final da sessão do Mathematica, denominamos tais regras de atribuição como sendo **regras de atribuição global**.

Uma regra de ouro consiste em limpar as definições de funções ou variáveis assim que não forem mais utilizadas durante a sessão, para evitar que estas definições sejam utilizadas em outros contextos, produzindo resultados indesejados.

Exemplo 5.4 *Inicie uma nova sessão e emita os comandos, observando os resultados.*

Comande: ***f[n_] = Expand[(1 + x)^n] + n + n***

Resposta: $2n + (1 + x)^n$

A expressão do lado direito foi calculada e simplificada, sendo o resultado atribuído a ***f[n_]****. Como o valor de n não é conhecido, a função* ***Expand*** *foi incapaz de desenvolver o binômio. Vamos verificar a definição de* ***f****.*

Comande: ***? f***

Resposta: Global ‘ f

*f[n_] = 2 * n + (1 + x)^n*

Para calcular o valor de ***f[2]****,*

Comande: ***f[2]***

Resposta: $4 + (1 + x)^2$

Observe agora a diferença, quando usamos a definição com retardo.

Comande: ***g[n_] := Expand[(1 + x)^n] + n + n***

Não há resposta a este comando. Vamos pedir a definição de ***g****.*

Comande: ***? g***

Resposta: Global ‘ g

g[n_] := Expand[(1 + x)ˆn] + n + n

Observe que a expressão que define ***g*** *é idêntica ao texto digitado. Não houve nenhum cálculo nem simplificação.*

Comande: ***g[2]***

Resposta: $5 + 2x + x^2$

Na definição de ***g****, o lado direito é atribuído a* ***g[n_]*** *tal como foi digitado, sem ser calculado. Quando solicitamos* ***g[2]****, o* ***n*** *assume o valor* ***2*** *na expressão* ***(1+x)ˆn****, tornando-a igual a* ***(1+x)ˆ2*** *após esta substituição, a expressão* ***Expand[(1+x)ˆ2]*** *é calculada, resultando em* $1 + 2x + x^2$.

5.5 Regras de substituição local

Uma **regra de substituição local** é um modo de se atribuir valores a variáveis ou mesmo definir funções com efeito temporário. Os valores atribuídos às variáveis e as definições de funções efetuadas em uma **regra de substituição** valem apenas no comando em que a regra foi emitida. Por este motivo diremos que uma regra de substituição tem **efeito local**. A sintaxe para uma **regra_de_substituição** é

lado_esquerdo $->$ **lado_direito**

ou

lado_esquerdo $:>$ **lado_direito**

Para utilizar uma regra de substituição, segue-se um dos caminhos abaixo

expressão /. regra_de_substituição

ou

expressão //. regra_de_substituição

Numa regra de substituição, numa primeira etapa, a **expressão** é simplificada. Em seguida, ela é percorrida da esquerda para a direita e sempre que o **lado_ esquerdo** da **regra_ de_ substituição** for encontrado na **expressão**, ele é substituído pelo **lado_ direito** da **regra_ de_ substituição**.

Quando usamos o separador **/.** com uma única barra, a **expressão**, num primeiro passo, simplificada e, por vezes, reorganizada. Em seguida, é percorrida uma única vez, sendo a **regra_ de _substituição** utilizada apenas na primeira ocorrência do **lado_ esquerdo** na **expressão**.

Quando usamos o separador **//.** com duas barras, num primeiro passo, a **expressão** é calculada e simplificada para, em seguida, ser percorrida diversas vezes, usando a regra de substituição sempre que o **lado_esquerdo** for encontrado na **expressão**. O processo finaliza quando a expressão definida pelo **lado_esquerdo** não for mais encontrada na **expressão**.

Exemplo 5.5 *Observe o que acontece quando usamos* ***/.*** *e* ***//.***

Comande: ***Clear[f, x, y, z, a, b]***

Comande: ***f[x + y + z + t] /. f[a_ + b_] -> f[a] + f[b]***

Resposta: f[t] + f[x + y + z]

A regra de substituição foi usada uma única vez e por essa razão não desenvolveu f[x + y + z].

Comande: ***Cos[a + b] + a + b /. a + b -> t***

Resposta: t + Cos[a + b]

Embora pensássemos em substituir o argumento do ***Cos*** *por* ***t****, a expressão* ***Cos[a + b] + a + b*** *foi reorganizada, se transformando em* ***a + b + Cos[a + b]*** *e a primeira ocorrência de* ***a + b*** *foi substituída por* ***t****.*

Comande: ***f[x + y + z] //. f[a_ + b_] − > f[a] + f[b]***

Resposta: f[x] + f[y] + f[z]

A regra de substituição foi usada duas vezes consecutivas.

Na primeira, fornece f[x] + f[y + z] e, como ainda aparece uma soma de dois objetos no argumento de f, a regra foi usada novamente para fornecer a resposta final.

Nota 5.1 ***Um alerta:*** *a regra de substituição com repetição pode gerar resultados indesejados ou gerar um loop interminável.*

Exemplo 5.6 *Observe a resposta.*

Comande: ***a //. a − > a + 1***

Resposta: ReplaceRepeated : : rrlim :

Exiting after a scanned 65536 times.

65536 + a

O ***a*** *é substituído continuamente por* ***a + 1****, gerando um loop interminável, até que o programa detecta e aborta o cálculo.*

Vamos ver a diferença entre a regra de substituição, com o símbolo − > (o hífen seguido por um sinal de maior) e a regra com o símbolo : > (dois pontos seguidos pelo sinal de maior). Quando se usa a regra de substituição

lado_esquerdo − > lado_direito

inicialmente o **lado_direito** é computado usando as regras existentes. Este **lado_direito** já avaliado é usado para substituir o **lado_esquerdo** na expressão. Esta é a chamada **atribuição imediata**. Por outro lado, quando se usa a regra

lado_esquerdo : > lado_direito

o **lado_direito** substitui as ocorrências do **lado_esquerdo** na expressão tal como foi digitado, sem nenhuma avaliação prévia. É a **atribuição retardada**.

Exemplo 5.7 *Para avaliar a diferença, observe os resultados*

Comande: ***f[2] /. f[x_] − > Expand[(a+b)^x]***

Resposta: $(a+b)^2$

Comande: ***f[2] /. f[x_] :> Expand[(a+b)^x]***

Resposta: $a^2 + 2\,a\,b + b^2$

No primeiro comando, como os valores de ***a***, ***b*** *e* ***x*** *são indefinidos, o* ***Expand[(a+b)^x]*** *é executado mas não consegue desenvolver o binômio. Deste modo,* ***f[x]*** *recebe o valor* ***(a+b)^x****. Esta expressão, ao ser usada para calcular* ***f[2]*** *resulta na primeira resposta* $(a+b)^2$*. No segundo caso, o lado direito da regra de substituição não é avaliado e* ***f[x]*** *recebe a expressão* ***Expand[(a+b)^x]****. Quando esta expressão é usada para substituir* ***f[2]****, obtemos* ***Expand[(a+b)^2]****. Neste momento, com o expoente definido, o binômio é desenvolvido e obtemos* $a^2 + 2ab + b$*.*

Comande: ***{ x , x } /. x :> Random[ ]***

Resposta: { *0.77102 , 0.77102* }

Comande: ***{ x , x } /. x :> Random[ ]***

Resposta: { *0.0539033 , 0.665482* }

Se o leitor executar estes dois últimos comandos, certamente obterá números diferentes. O que importa é que, no primeiro caso, ***Random[ ]*** *gera um número pseudo aleatório e o atribui a* ***x****. Este valor é usado no primeiro e no segundo* ***x*** *da lista, ficando ambos iguais. No segundo caso, a variável* ***x*** *recebe a função* ***Random[ ]*** *que é passada para os dois* ***x*** *da lista, resultando em* ***{Random[ ], Random[ ]}****. Como* ***Random[ ]*** *é calculada duas vezes, são gerados dois números pseudo aleatórios distintos.*

Em lugar de uma única regra de substituição podemos ter uma **lista de regras de substituição**.

Exemplo 5.8 *Para atribuir o valor* 2 *para* ***x*** *e o valor* 3 *para* ***y*** *na expressão* $x^2 + 3y^2$ *e calcular o valor resultante,*

Comande: ***x^2 + 3 y^2 /. { x :> 2 , y :> 3 }***

Resposta: 31

5.6 Definição condicional

Para definir funções que obedecem fórmulas distintas em intervalos diferentes, use a função lógica **If**.

Exemplo 5.9 *Para definir a função*

$$f(x) = \begin{cases} 2x, & \textit{para} \quad x \leq 1 \\ \operatorname{sen}(x) & \textit{para} \quad 1 < x \leq 2 \\ \exp(x) & \textit{para} \quad x > 2 \end{cases}$$

Comande: ***g[x_] = If[x <= 1 , 2x , Sin[x]]***

Resposta: *If[x <= 1 , 2x , Sin[x]]*

Comande: ***f[x_] = If[x <= 2 , g[x] , Exp[x]]***

Resposta: *If[x ¡= 2 , g[x] , Exp[x]]*

Comande: ***Print[f[1] , " ", f[2] , " ", f[3]]***

Resposta: 2 *Sin[2]* E^3

Pode-se definir funções que obedecem a expressões diferentes em intervalos distintos usando uma **regra de substituição condicional**, usando o símbolo **/;** para separar a definição da função dos intervalos a serem usados.

Num comando do tipo

expressão /; condição

onde aparece a cláusula **/;** a **expressão** só é calculada se a **condição** for uma expressão lógica verdadeira.

Exemplo 5.10 *Vamos definir a função*

$$f(x) = \begin{cases} 0 & \text{se} \quad x \leq 0 \\ 10 & \text{se} \quad 0 < x \leq 2 \\ 20 & \text{se} \quad 2 < x \end{cases}$$

Comande: ***f[x_] := 0 /; x <= 0***
f[x_] := 10 /; x <= 2
f[x_] := 20

Pressione a tecla ***Enter*** *ao final das duas primeiras linhas e a tecla* ***Insert*** *ao final da última. Para verificar a definição de* ***f****,*

Comande: ***? f***

Resposta: *Global 'f*
f[x_] := 0 /; x <= 0
f[x_] := 10 /; x <= 2
f[x_] := 20

Importante: *À medida que definimos uma função ou variável usando argumentos condicionais, estas definições vão sendo gravadas em uma lista. Ao tentar calcular a* ***f*** *num ponto, o Mathematica percorre a lista de definições da* ***f*** *de cima para baixo, executando a primeira definição que se aplicar ao caso. Deste modo, deve-se definir a* ***f*** *começando com a definição mais restritiva e terminando com a mais geral.*

Para verificar os valores de ***f*** *nos pontos* ***0****,* ***1****,* ***2****, e* ***3****,*

Comande: ***Print[f[0] , " ", f[1] , " ", f[2] , " ", f[3]]***

Resposta: 0 10 10 20

5.7 Funções que exigem múltiplos comandos

Quando uma função necessitar de dois ou mais comandos para ser definida, estes comandos deverão ser agrupados dentro de um fechar e abrir parênteses enquanto os diversos comandos deverão ser separados por ponto e vírgulas. Os comandos serão executados um a um da esquerda para a direita. O valor da função será o resultado do último comando.

Exemplo 5.11 *Vamos construir uma função que desenvolve o binômio* $(1+x)^n$ *para, em seguida, extrair o coeficiente numérico do termo que contém a potência* x^k*.*

Comande: ***coef[n_ , k_] := (poli := Expand[(1+x)^n] ;***

Coefficient[poli , x^k])

Comande: ***coef[10 , 3]***

Resposta: 120

Comande: ***Table[coef[10 , j] , { j , 0 , 10 }]***

Resposta: { *1, 10, 45, 120, 210, 252, 210, 120, 45, 10, 1* }

Exemplo 5.12 *Vamos calcular o* ***wronskiano*** *das funções* $\mathrm{sen}(t)$, $\cos(t)$ *e* $\exp(t)$. *O wronskiano de três funções* $f(t)$, $g(t)$, $h(t)$ *é definido por*

$$W[f,g,h](t) = \begin{vmatrix} f(t) & g(t) & h(t) \\ f'(t) & g'(t) & h'(t) \\ f''(t) & g''(t) & h''(t) \end{vmatrix}$$

Comande: ***Clear[x , y , z]***

Comande: ***x = Cos[t] ; y = Sin[t] ; z = Exp[t] ;***

Comande: ***matriz_wronski[lista_ , var_] := (n := Length[lista] ;***

Table[D[lista , { var , k }] , { k , 0 , n − 1 }])

Comande: ***matriz_wronski[{ x , y , z } , t]***

Resposta: { { *Cos[t] , Sin[t] ,* E^t } ,

{ − *Sin[t] , Cos[t] ,* E^t },

{ − *Cos[t] , − Sin[t] ,* E^t } }

Comande: ***wronskiano = Simplify[Det[%]]***

Resposta: $2\,E^t$

5.8 Funções recursivas

Uma função recursiva é aquela definida sobre os números naturais e que, para ser calculada num ponto **n**, necessita dos seus valores em **0**, **1**, **...**, **n–1**. Dentre as funções recursivas, citamos a **seqüência de Fibonacci**, definida por

$$\begin{aligned} \mathrm{fib}(0) &= 1\,, \\ \mathrm{fib}(1) &= 1\,, \\ &\text{e} \\ \mathrm{fib}(n) &= \mathrm{fib}(n-1) + \mathrm{fib}(n-2) \end{aligned}$$

para $n = 2, 3, \ldots$ e o fatorial de um número natural

$$\begin{aligned} \mathrm{fat}(0) &= 1 \\ \mathrm{fat}(n) &= n \times \mathrm{fat}(n-1) \end{aligned}$$

para $n = 1, 2, \ldots$

Exemplo 5.13 *Vamos definir a seqüência de Fibonacci*

Comande: ***fib[0] = 1 ; fib[1] = 1 ;***

Comande: ***fib[n_] := fib[n − 1] + fib[n − 2] ;***

Para verificar a definição da seqüência,

Comande: ***? fib***

Resposta: *Global ‘fib*
fib[0] = 1
fib[1] = 1
fib[n_] := fib[n–1] + fib[n–2]

Comande: ***fib[6]***

Resposta: 13

5.9 Pilhas para funções recursivas

Na sessão anterior, para calcular **fib[6]** foi preciso calcular fib[5], fib[4], fib[3], fib[2]. Se solicitássemos o cálculo de **fib[7]**, seria necessário repetir o processo, calculando desde fib[2] até fib[6]. Pode-se acelerar o processo, definindo **fib** de modo que o Mathematica grave em uma pilha os resultados obtidos para utilizá-los quando necessário. Para tanto, deve-se definir a seqüência como no exemplo que segue.

Exemplo 5.14 *Vamos definir a seqüência de Fibonacci de modo que os valores obtidos previamente fiquem gravados para serem reutilizados.*

Comande: ***fib[0] = 1 ; fib[1] = 1 ;***

Comande: ***fib[n_] := fib[n] = fib[n − 1] + fib[n − 2] ;***

Comande: ***Timing[fib[50]]***

Resposta: { *0.06 Second, 20365011074* }

Comande: ***Timing[fib[51]]***

Resposta: { *0. Second, 32951280099* }

Foi mais rápido calcular ***fib[51]*** *do que* ***fib[50]*** *pois, após o cálculo de* ***fib[50]****, os valores de* ***fib[49]*** *e* ***fib[50]*** *estão guardados.*

A sintaxe para definir uma função de recorrência que grave os valores calculados é

func[x_ , y_ , ...] := func[x , y , ...] = definição da função

observando que, na expressão da esquerda, cada argumento recebe uma sublinha, ao passo que, na expressão do meio, não se coloca a sublinha.

5.10 Apply, Map, Fold, Nest, FixedPoint

Nesta seção descreveremos alguns comandos para manipular listas. Durante esta seção, **f** representará o nome de uma função, **x** uma expressão numérica, **n** um número inteiro e **{ a, b, ... }** uma lista genérica. Quando conveniente, indicaremos a lista genérica por uma lista com dois ou três elementos. Perde-se a generalidade mas ganha-se em clareza. Vamos descrever os seguintes comandos

Apply[f , { a , b , ... }] = f[a , b , ...]

Map[f , { a , b , ... }] = { f[a] , f[b] , ... }

Fold[f , x , { a , b , c }] = f[f[f[x , a] , b] , c]

FoldList[f , x , { a , b , ...}] = { x , f[x , a] , f[f[x , a]] , ...}

Nest[f , x , n] = f[f[... f[x] ...]] (n aplicações de f)

NestList[f , x , n] = { x , f[x] , f[f[x]] , ... , Nest[f , x , n] }

FixedPoint[**f , x , n**]	=	f[f[...f[x] ...]] (n aplicações de f)
FixedPoint[**f , x**]	=	x0 onde x0 = f[x0]

Nota 5.2 *As funções que agem sobre listas podem ser aplicadas a listas com tamanho arbitrário. Ao descrevê-las, tomamos listas particulares com dois ou três elementos apenas para tornar a sintaxe mais clara.*

O comando

Apply[**f , { a , b , ... }**]

fornece **f**[**a, b, ...**]. O comando

Map[**f , { a , b , ... }**]

fornece a lista **{ f**[**a**], **f**[**b**], **... }**. O comando

Fold[**f , x , { a , b , c }**]

fornece **f**[**f**[**f**[**x, a**] , **b**] , **c**]. Aqui lembramos que este comando se aplica a listas maiores. O comando

FoldList[**f , x , { a , b , ... }**]

fornece a lista **{ x, f**[**x, a**] , **f**[**f**[**x, a**], **b**] , **... }** sendo **Fold**[**f, x, {a, b, ...}**] o último elemento desta lista. O comando

Nest[f , x , n]

compõe **f** em **x** um número **n** de vezes. Assim, **Nest[f, x, 3] = f[f[f[x]]]**. O comando

NestList[f , x , n]

fornece a lista

{ Nest[f, x, 0] , Nest[f, x, 1] , ... , Nest[f, x, n] } =

= { x , f[x] , f[f[x]] , ... }

com **n+1** elementos, sendo **Nest[f, x, n]** o último elemento desta lista. O

FixedPoint[f , x , n]

calcula sucessivamente

f[x] , **f[f[x]]** , **f[f[f[x]]]** , ... ,

interrompendo o processo no **n**-ésimo passo e fornecendo o último valor calculado. O

FixedPoint[f , x]

é o limite da seqüência anterior. Ele fornece o **ponto fixo** de **f[x]**, isto é, calcula **xfix** tal que **xfix = f[xfix]**. Pode-se colocar uma opção de controle neste comando para determinar sua interrupção. Assim

FixedPoint[f , x , SameTest −> (Abs[#1 − #2] < erro) &]

solicita que o cômputo da seqüência de valores

f[x] , f[f[x] , f[f[f[x]]] , ... ,

seja interrompido quando o valor absoluto da diferença entre dois termos consecutivos desta seqüência for menor que o **erro**.

Exemplo 5.15 *Vamos ilustrar cada um dos comandos acima.*

Comande: ***{ Apply[Plus , { 3, 4 }] , Apply[Times , { 3, 4 }] }***

Resposta: { *7, 12* }

Comande: ***f[x_ , a_] = a / (1 + x)***

Resposta: $\dfrac{a}{1+x}$

Comande: ***Fold[f , x , { 2, 3 }]***

Resposta:

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

Comande: ***FoldList[f , x , { 5, 7 }]***

Resposta:

$$\{x, \frac{5}{1+x}, \frac{7}{1+\frac{5}{1+x}}\}$$

Comande: ***g[y_] : = D[y , x]***

Comande: ***Map[g , { Cos[x] , Sin[x] }]***

Resposta: { *− Sin[x] , Cos[x]* }

Vamos construir uma seqüência que converge para $\sqrt{2} \simeq 1.41421$.

Comande: ***h[x_] = (2/x + x) / 2***

Resposta: $\dfrac{2/x + x}{2}$

Comande: ***NestList[h , 1. , 4]***

Resposta: { *1. , 1.5 , 1.41667 , 1.41422 , 1.41421* }

Comande: ***Nest[h , 1. , 4]***

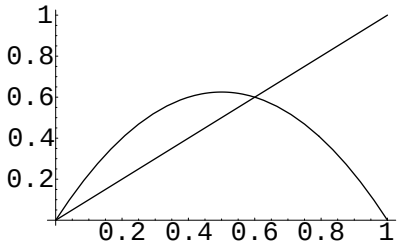
Resposta: 1.41421

O gráfico que segue mostra que $k(x) = 2.5x(1-x)$ *tem um ponto fixo que é a solução de* $2.5x(1-x) = x$. *Sem dificuldade, calculamos que* $x = 0,6$. *Vamos obter este ponto com o comando* ***FixedPoint****.*

Comande: ***k[x_] = 2.5 x (1 − x) ;***

Comande: ***Plot[{ x , k[x] } , { x , 0 , 1 }]***

Resposta:



Comande: ***FixedPoint[k , 0.7]***

Resposta: 0.6

5.11 Inner, Outer

Nesta seção, continuam válidas as observações da seção anterior. Vamos descrever as funções **Outer** e **Inner**.

Outer[f, { a, b }, { x, y }]	=	{ {f[a, x], f[a, y]}, {f[b, x], f[b, y]} }
Inner[f, { a, b }, { x, y }]	=	f[a , x] + f[b , y]

A função

```
Outer[ f , { a , b , c } , { x , y , z } ]
```

fornece a lista

{ { f[a,x], f[a,y], f[a,z] }, { f[b,x], f[b,y], f[b,z] }, { f[c,x], f[c,y], f[c,z] } }

e a

```
Inner[ f , { a , b , c } , { x , y , z } ]
```

calcula

f[a,x] + f[b,y] + f[c,z]

que é uma generalização do produto interno. De fato,

Inner[Times, { a, b, c } , { x, y, z }]

resulta no produto interno

a*x + b*y + c*z.

Exemplo 5.16 *Considere a transformação de coordenadas polares* (r, t) *para coordenadas cartesianas* (x, y)*, definida por*

$$x = r\cos(t) \quad e \quad y = r\text{sen}(t)$$

A matriz

$$\begin{bmatrix} \partial x/\partial r & \partial x/\partial t \\ \partial y/\partial r & \partial y/\partial t \end{bmatrix}$$

onde as derivadas parciais são calculadas no ponto (r, t) *é chamada de* ***matriz jacobiana*** *da transformação no ponto* (r, t) *e é denotada por* $J[x, y](r, t)$*. Pode-se omitir o ponto em que a matriz jacobiana é calculada e denotá-la apenas por* $J[x, y]$*. Para calcular a matriz jacobiana desta transformação,*

Comande: ***x = r * Cos[t] ; y = r * Sin[t] ;***

Lembrando que ***D[x , t]*** *calcula a derivada parcial da função* ***x*** *em relação a* ***t****, para calcular a matriz jacobiana,*

Comande: ***Outer[D , { x , y } , { r , t }]***

Resposta: { { *Cos[t] , –(r Sin[t])* } , { *Sin[t] , r Cos[t]* } }

O jacobiano da transformação é o determinante da matriz jacobiana. Para calcular o jacobiano, e simplificar a expressão,

Comande: ***Simplify[Det[%]]***

Resposta: r

Para calcular $\partial x/\partial r + \partial y/\partial t$,

Comande: ***Inner[D , { x , y } , { r , t }]***

Resposta: Cos[t] + r Cos[t]

5.12 Composição e função inversa

Podemos calcular a **composição** das funções f, g e h que se denota por $f \circ g \circ h$ usando

Composition[**f , g , h**] [**x**] = $f \circ g \circ h(x)$

Este comando aceita duas ou mais funções. Eventualmente o Mathematica nos fornece uma resposta simbólica em termos da **inversa** f^{-1} de uma função f. Esta inversa $x = f^{-1}(y)$ de $y = f(x)$ será representada por

InverseFunction[**f**] [**y**]

ou

$\mathbf{f}^{(-1)}[y]$

Quando a função tiver mais que um argumento, a resposta poderá vir nas formas

InverseFunction[f , n]

InverseFunction[f , n , tot]

onde **n** indica a função inversa em relação à **n**-ésima variável e **tot** o número total de variáveis. Para representar a função identidade, o Mathematica usa

Identity[x]

Nota 5.3 *A* ***InverseFunction*** *é obtida como resposta ao comando* ***Solve*** *que será visto posteriormente, quando a opção InverseFunctions estiver com o valor Automatic ou True.*

Nota 5.4 *A* ***InverseFunction*** *pode calcular o valor da função inversa em alguns casos. Por exemplo, InverseFunction[Sin] fornece a função ArcSin.*

Exemplo 5.17 *Vamos calcular a composta de* $f(x) = \text{sen}(x)$ *com* $g(x) = 3x$ *e resolver em* x *a equação* $h(x) = 2$*. Esta solução virá em termos da inversa de* h*. Na continuação, faremos* $h = \exp$*.*

Comande: ***Clear[f , g , h , x]***

Comande: ***f[x_] = Sin[x] ; g[x_] = 3x ;***

Comande: ***Composition[f , g] [x]***

Resposta: Sin[3 x]

Comande: ***Solve[h[x] == 2 , x]***

Resposta: { { x -> $h^{(-1)}[2]$ } }

O comando ***Solve[equação, x]*** *resolve a* ***equação*** *em relação à variável* ***x****. A* ***equação*** *deve ser escrita na forma*

LadoDireito == LadoEsquerdo

onde o sinal de = é substituído por um duplo sinal ==.

Comande: ***% /. h -> Exp***

Resposta: { { x -> *Log[2]* } }

Para calcular o valor aproximado de Log[2],

Comande: ***x /. %[[1]] //N***

Resposta: 0.693147

Para saber qual a função inversa do seno,

Comande: ***InverseFunction[Sin][y]***

Resposta: ArcSin[y]

Para calcular a função inversa do seno no ponto $y = 1$,

Comande: ***InverseFunction[Sin][1]***

Resposta: $\pi/2$.

5.13 Estrutura das expressões

Embora tenhamos estudado diversas estruturas, todas elas possuem a mesma sintaxe, que nem sempre fica visível ao usuário. Toda expressão no Mathematica é da forma

Nome[arg1 , arg2 , ...]

onde **Nome** é o nome de uma função e **arg1**, **arg2**, **...**, são os seus argumentos. Esta **estrutura** pode ser visualizada com o comando

FullForm[estrutura]

Exemplo 5.18 *Vamos observar a estrutura de uma lista, de uma soma e um produto*

Comande: ***Clear[a, b, c, f]***

Comande: ***FullForm[{ a, b, c }]***

Resposta: List[a , b , c]

Comande: ***FullForm[a + b + c]***

Resposta: Plus[a , b , c]

Comande: ***FullForm[a * b * c]***

Resposta: Times[a , b , c]

Comande: ***FullForm[f[a , b , c]]***

Resposta: f[a , b , c]

Retornando aos comandos **Apply** e **Map**, percebemos que o primeiro troca o nome da estrutura por **f** enquanto o segundo aplica **f** a cada argumento da estrutura. Podemos redefinir estes comandos escrevendo

Apply[f , Nome[a , b , c]] = f[a , b , c]

Map[f , Nome[a , b , c]] = Nome[f[a] , f[b] , f[c]]

Exemplo 5.19 *Observe os resultados. Limpe os valores de* ***a*** *e* ***b*** *e*

Comande: ***Apply[Plus , { a , b }]***

Resposta: a + b

Comande: ***Apply[Times , a + b]***

Resposta: a b

Exemplo 5.20 *Vamos calcular a média aritmética e geométrica da* ***lista = { 1, 2, 3, 4, 5, 6 }***

Comande: ***lista = { 1, 2, 3, 4, 5, 6 } ;***

Comande: ***arit = Apply[Plus, lista] / Length[lista]***

Resposta: 7 / 2

Comande: ***geo = Apply[Times, lista]^(1 / Length[lista]) //N***

Resposta: 2.9938

Como desafio, solicitamos ao leitor que generalize este exemplo, definindo uma função que calcule a média aritmética e outra que calcule a média geométrica de uma lista qualquer.

5.14 Função anônima

Quando necessitamos usar uma função diversas vezes, nós a definimos atribuindo-lhe um nome. Com este procedimento, poderemos chamá-la sempre que for necessário. Em determinadas ocasiões, precisamos da função uma única vez. Eventualmente, este é o caso quando usamos as funções **Array**, **Apply**, **Fold**, **Map**, **Nest**, **FixedPoint**, **Outer**, **Inner**. Quando tal fato ocorrer, é vantajoso usar uma **função anônima**. Para aplicar uma função anônima nos valores **val1**, **val2**, **...**, use

Function[{ x1 , x2 , ... } , expressão] [val1 , val2 , ...]

onde **x1**, **x2**, **...**, são os argumentos da função anônima que aparecem na **expressão**. A **Function** substitui as ocorrências de **x1**, **x2**, **...**, por **val1**, **val2**, **...** e calcula a **expressão** resultante. Se a função possuir uma única variável, pode-se omitir as chaves que envolveriam esta variável. Pode-se também usar a sintaxe

(expressão) & [**val1 , val2 , ...**]

Os parênteses são dispensáveis devido a baixa prioridade do **&**. Todavia, iremos inclui-lo quando houver dúvida quanto à prioridade dos operadores envolvidos. Com esta última forma de apresentação de uma função anônima, os **argumentos** devem ser denotadas pelos símbolos abaixo

#	representa a primeira variável
#n	representa a n-ésima variável
##	representa todas as variáveis
##n	representa todas as variáveis, a partir da n-ésima

Exemplo 5.21 *Observem os resultados obtidos com os comandos deste exemplo.*

Comande: ***Clear[a , b]***

Comande: ***a[#] + b[#] & [x^2]***

Resposta: $a[\ x^2\ ] + b[\ x^2\ ]$

Comande: ***Function[a , 2 * a + Cos[a]] [Exp[x]]***

Resposta: $2\ E^x + Cos[\ E^x\ ]$

Comande: ***lis1 = { { a, b }, { x, y } }***

Resposta: {{a,b},{x,y}}

Comande: ***lis2 = Map[{ #[[2]], #[[1]] }&, lis1]***

Resposta: {{b,a},{y,x}}

Exemplo 5.22 *Vamos eliminar os termos que contêm* ***x*** *e selecionar os termos contendo* ***x^2*** *no polinômio*

$$x + y + x^2 + y^2 + xy + xy^2 + x^2y$$

Queremos ***alertar*** *o leitor de que existem comandos que efetuam estas tarefas de modo direto e que foram descritos no capítulo que trata da Álgebra.*

Comande: ***x + y + x^2 + y^2 + x*y + x*y^2 + x^2*y***

Resposta: $x + x^2 + y + x\,y + x^2y + y^2 + x\,y^2$

Comande: ***poli = Collect[% , x]***

Resposta: $y + y^2 + x^2\,(1 + y) + x\,(1 + y + y^2)$

Comande: ***Select[poli , FreeQ[# , x] &]***

Resposta: $y + y^2$

Comande: ***Select[poli , MatchQ[# , x^2_] &]***

Resposta: $x^2\,(1 + y)$

Observamos que ***x^2_*** *(observe a sublinha) é um* ***pattern*** *(padrão). O comando está pedindo que sejam selecionados todos os termos do polinômio que contêm o padrão* ***x^2_****.*

5.15 Comando de repetição Do

Vimos diversos comandos que executam tarefas repetidas, que se repetem à medida que vão modificando um parâmetro. Um comando repetitivo bastante importante é o

Do[com1 ; com2 ; .... , { i , imin , imax , di }]

Ele executa os comandos **com1**, **com2**, **...**, nesta seqüência, enquanto o parâmetro **i** recebe sucessivamente os valores

imin, imin + di, imin + 2 di, imin + 3 di, ..., imin + k di

sendo **k** o maior inteiro para o qual **imin + k di** $\leq$ **imax**.

A lista que fornece a variação do parâmetro **i** pode receber uma das formas simplificadas

{ i , imin , imax } ou **{ i , imax }** ou **{ imax }**

cujos significados explicamos anteriormente.

Podemos emitir o **Do** com dois ou mais parâmetros.

Do[com1; com2; .... , { i, imin, imax, di }, { j, jmin, jmax, dj }]

Exemplo 5.23 *Vamos calcular os polinômios de Legendre correspondentes a* ***n=1****,* ***n=2****,* ***n=3****, definidos pela fórmula de Rodrigues*

$$P_n(x) = \frac{1}{n!2^n}\frac{d^n}{dx^n}\left[(x^2-1)^n\right], \quad n = 0, 1, 2, \ldots$$

Comande: ***plegendre[n_ , x_] :=***

D[(x^2 - 1)^n , {x , n}] / (n! 2^n)

Comande: ***Do[Print[Simplify[plegendre[n, x]]] , { n, 1, 3 }]***

Resposta:

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

5.16 Interpolação

Consideremos a lista

dados = { { x1, y1 } , { x2, y2 } , ..., { xn, yn } }

A função

InterpolatingPolynomial[dados , x]

gera um polinômio do grau **n-1**, $y = P(x)$, que interpola os dados, isto é, **yk = P[xk]**, **k=1, 2, ..., n**. A função

Interpolation[dados]

gera uma **InterpolatingFunction** que pode interpolar os valores dos dados no intervalo [**xmin, xmax**], onde **xmin = min{ x1, x2, ..., xn }** e **xmax = max{ x1, x2, ..., xn }**

Exemplo 5.24 *Vamos trabalhar com estas funções.*

Comande: ***dados = { { 1 , 1}, { 2 , 2 } , { 4 , 4 } ,***

{ 6 , 6 } , { 7 , 1 } , { 8 , 8 } }

Resposta: { { 1 , 1 }, { 2 , 2 } , { 4 , 4 } , { 6 , 6 } , { 7 , 1 } , { 8 , 8 } }

Comande: ***poli[x_] = Expand[***

InterpolatingPolynomial[dados, x]]

Resposta: $-(\frac{128}{5}) + \frac{799}{15}x - 36\,x^2 + \frac{32}{3}x^3 - \frac{7}{5}x^4 + \frac{1}{15}x^5$

Comande: ***interpol = Interpolation[dados]***

Resposta: InterpolatingFunction[{ 1 , 8 } , < >]

Comande: ***poli[5] //N***

Resposta: 7.4

Comande: ***interpol[5] //N***

Resposta: 6.2

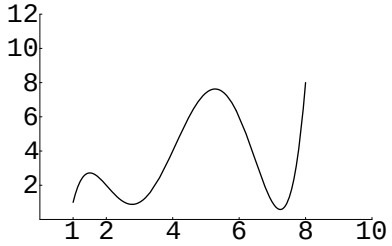
Observe que as interpolações geradas são diferentes. O ***InterpolatingFunction*** *gera uma interpolação local. Em cada intervalo ele gera um polinômio interpolador (esta função gera um "spline"). O* ***InterpolatingPolynomial*** *gera um único polinônio que interpola os dados em todo o intervalo que vai desde o menor até o maior valor de* ***x****. Para patentear esta observação,*

Comande: ***Plot[poli[x] , { x , 1, 8 } ,***

PlotRange -> { { 0 , 10 } , { 0 , 10 } } ,

Ticks -> { { 1 , 2 , 4 , 6 , 8 , 10 } , { 2 , 4 , 6 , 8 , 10 } }]

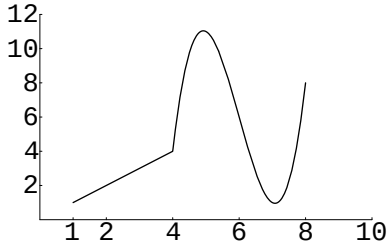
Resposta:



Comande: ***Plot[interpol[x] , { x , 1 , 8 } ,***

PlotRange -> { { 0 , 12 } , { 0 , 12 } } ,

Ticks -> { { 1 , 2 , 4 , 6 , 8 , 10 } , { 2 , 4 , 6 , 8 , 10 , 12 } }]



Capítulo 6

Equações

Este capítulo se destina ao estudo da solução de equações algébricas.

6.1 Teste de igualdade

No Mathematica existe uma diferença entre os sinais

= e ==

Quando escrevemos

x = y

estamos atribuindo à variável **x** o valor **y**.

Exemplo 6.1 *Siga as orientações*

Comande: ***var = 2 y + 3***

Resposta: var = 3 + 2 y

Observe que a resposta foi uma repetição do comando emitido. Apenas se atribuiu o valor ***2y+3*** *à variável* ***var****.*

Quando digitamos

```
x == y
```

estamos solicitando que o Mathematica verifique se **x** é ou não igual a **y**. Se forem iguais, o resultado da verificação será **True** (verdadeiro). Se não forem iguais, o resultado será **False** (falso).

Exemplo 6.2 *Desenvolva o exemplo*

Comande: ***2 + 3 == 5***

Resposta: True

Comande: ***2 + 3 == 6***

Resposta: False

Comande: ***x == 3***

Resposta: x == 3

Como não atribuímos valor a ***x****, a equação pode ser ou não verdadeira. Neste caso, o Mathematica devolve o comando digitado.*

Comande: ***x = 3***

Resposta: 3

Observe que a resposta repetiu o valor de ***x****.*

Comande: ***x == 3***

Resposta: True

Como agora ***x = 3****, a equação* ***x == 3*** *é verdadeira.*

Para evitar que o ***x*** *continue valendo* ***3*** *durante toda a sessão do Mathematica, limpe este valor.*

Comande: ***Clear[x]***

Comande: ***x***

Resposta: x

Depois de limpar o valor de uma variável, seu valor será o seu próprio nome

6.2 Equações algébricas

Uma **equação algébrica** deve ser colocada na forma

lado_esquerdo == lado_direito

onde o sinal de = usado na linguagem comum é substituída por um duplo sinal de igualdade (==). Usamos um único sinal de igualdade = para atribuir valor a uma variável ou definir uma função.

Exemplo 6.3 *No Mathematica, as equações* $x^2 + 1 = 0$ *e* $e^x = 1 + x$ *são escritas na forma*

```
x^2 + 1 == 0

Exp[ x ] == 1 + x
```

Quando se tem uma equação como as deste exemplo, o que se deseja é determinar que valores de **x** tornam a igualdade verdadeira. A busca destes valores se restringe a algum conjunto que, em geral, são os conjuntos dos números reais ou complexos. Tais valores são chamados de **soluções da equação**. A primeira equação tem duas soluções complexas $x = i = \sqrt{-1}$ e $x = -i = -\sqrt{-1}$. No conjunto dos números reais, a segunda tem uma única solução $x = 0$.

6.3 Inequação

Uma **inequação** é uma expressão do tipo

lado_esquerdo	**.op.**	**lado_direito**

onde **.op.** é um dos operadores relacionais

! = $\quad > \quad \geq \quad < \quad \leq$

cujos significados serão descritos na próxima seção.

Exemplo 6.4 *São inequações*

$x\hat{}2 + y\hat{}2 < 1$

$x + y > 2$

Resolver uma inequação consiste em determinar os valores das variáveis que tornam a relação que tornam a inequação verdadeira. A busca destes valores se restringe a algum conjunto. Os valores que tornam a inequação verdadeira são chamados de **soluções da inequação**.

6.4 Operadores relacionais e lógicos

Na resolução de equações e inequações, precisamos dos **operadores relacionais** que se aplicam entre duas expressões aritméticas ou algébricas fornecendo um resultado verdadeiro (**True**) ou falso (**False**). Sejam **x** e **y** duas expressões aritméticas ou algébricas. Os operadores relacionais disponíveis são

$\mathbf{x == y}$	....	verdadeiro quando **x** for igual a **y**
$\mathbf{x != y}$	....	verdadeiro quando **x** for diferente de **y**
$\mathbf{x > y}$	....	verdadeiro quando **x** for maior do que **y**
$\mathbf{x >= y}$	....	verdadeiro quando **x** for maior ou igual a **y**
$\mathbf{x < y}$	....	verdadeiro quando **x** for menor do que **y**
$\mathbf{x <= y}$	....	verdadeiro quando **x** for menor ou igual a **y**

Podemos também fazer testes múltiplos, tais como

$\mathbf{x == y == z}$	....	verdadeiro quando **x** for igual a **y** e a **z**
$\mathbf{x > y > z}$	....	verdadeiro quando **x** for maior que **y** e **y** for maior que **z**

O resultado da aplicação de uma operação relacional a duas expressões aritméticas ou algébricas é denominada de **expressão lógica**, que pode assumir os valores verdadeiro (**True**) ou falso (**False**).

No Mathematica, uma equação como

x^2 − 2 x + 1 == 0

não é verdadeira nem falsa, enquanto não se atribui algum valor a **x**. Como a única raiz do polinômio $x^2 - 2x + 1$ é $x = 1$, a equação será verdadeira quando **x = 1**.

Exemplo 6.5 *Limpe a variável* ***x*** *com o comando* ***Clear[x]*** *e acompanhe este exemplo.*

Comande: ***x^2 − 2 x + 1 == 0***

Resposta: $1 - 2x + x^2 == 0$

Comande: ***% /. x − > 1***

Resposta: True

Lembre-se, a parte ***/. x − > 1*** *do comando acima, atribui a* ***x*** *o valor* ***1*** *apenas nesta equação. Como* ***x = 1*** *é uma solução da equação* ***1− 2x+x^2 == 0****, ela se torna verdadeira.*

Os **operadores lógicos** são aqueles que, atuando sobre expressões lógicas, nos fornecem outra expressão lógica. Sejam **p** e **q** duas expressões lógicas. Então

!p	....	Não p
p && q && ...	....	**p** e **q** e **...**
p ‖ q ‖ ...	....	**p** ou **q** ou **...**
Xor[p , q , ...]	....	ou **p** ou **q** ou **...**

!p será verdadeiro, se e só se **p** for falso.

p && q && ... será verdadeiro se todas as expressões lógicas **p**, **q**, **...**, forem verdadeiras.

p || **q** || **...** será verdadeiro se pelo menos uma das expressões lógicas **p**, **q**, **...**, for verdadeira.

Xor [**p , q , ...**] será verdadeiro quando uma e apenas uma das expressões lógicas for verdadeira.

O abrir e fechar **parênteses** pode ser usado para designar agrupamento de operadores relacionais ou lógicos.

O comando

LogicalExpand [**explógica**]

desenvolve e simplifica a expressão lógica **explógica**.

Exemplo 6.6 *Execute este exemplo.*

Comande: ***LogicalExpand[(a && b || c) && !a]***

Resposta: *c && !a*

6.5 Gravar equação em variável

Pode-se gravar uma equação em uma variável com o comando

var = equação

onde **var** é o nome da variável conterá a **equação**.

Exemplo 6.7 *Acompanhe os comandos*

Comande: ***eq = x^2 - 2 x + 1 = = 0***

Resposta: $1 - 2x + x^2 = = 0$

Comande: ***eq /. x -> 1***

Resposta: *True*

6.6 O comando If

Há um operador para efetuar decisões lógicas muito importante em computação. Este é o comando

If[explógica , se_verdade , se_falsa]

onde **explógica** é uma expressão lógica. Sendo ela verdadeira, o comando **se_verdade** será executado. Sendo falsa, o comando **se_falsa** será executado. Podemos ler o comando acima do seguinte modo: se **expr_lógica** for verdadeira, então execute o comando **se_verdade**, senão, execute o comando **se_falso**.

Exemplo 6.8 *Acompanhe a sessão.*

Comande: ***Clear[a , y]***

Comande: ***If [2 < 4 , y = a + 4 , y = 2 a]***

Resposta: 4 + a

Comande: ***y***

Resposta: 4 + a

Observe que este é o valor de ***y****, uma vez que, sendo* $2 < 4$ *uma expressão lógica verdadeira, o comando executado foi* $y = a + 4$*.*

Comande: ***If [6 < 4 , y = a + 4 , y = 2 a]***

Resposta: 2 a

Comande: ***y***

Resposta: 2 a

Observe: Agora, $6 < 4$ *é falso, fazendo com que o comando executado seja* $y = 2a$*.*

6.7 Resolução de equações algébricas

Para resolver uma **equação algébrica linear ou polinomial** em relação a uma variável **var**, comande

Solve[equação , var]

ou

Reduce[equação , var]

onde **equação** é da forma

LadoEsquerdo == LadoDireito

(observe os dois sinais de igualdade adjacentes) e **var** é a variável que se deseja explicitar. Havendo apenas uma variável, pode-se usar estes comandos na forma abreviada

Solve[equação] ou **Reduce[equação]**

Exemplo 6.9 *Inicie uma nova sessão e siga as instruções abaixo.*

Comande: ***Solve[x^2 - 4x - 8 == 0 , x]***

Resposta:

$$\{\{x->\frac{4-4\ \text{Sqrt}[\,3\,]}{2}\},\{x->\frac{4+4\ \text{Sqrt}[\,3\,]}{2}\}\}$$

Observe que, sendo as raizes irracionais, a resposta foi dada em termos da raiz quadrada de 3. O resultado fornecido é exato. Para obter um valor numérico aproximado,

Comande: ***N[%]***

Resposta: { { $x->-1.4641$ } , { $x->5.4641$ } }

Comande: ***Solve [x^5 - 3 x^2 + 4 x - 1 == 0 , x]***

Resposta:

$$\{\ \text{ToRules}[\ \text{Roots}[\ 4\,x - 3\,x^2 + x^5 = 1\,,\ x\ ]\ ]\ \}$$

Entenda que, como a equação é do quinto grau, não há uma fórmula explícita que forneça as raizes exatas da equação. Este é o motivo da resposta obtida. Na continuação desta seção, explicaremos o ***Roots*** *e o* ***ToRules****.*

Para obter o valor numérico das raizes,

Comande: ***N[%]***

Resposta:

{ { *x -> -1.06396 - 1.32497 I* } ,

{ *x -> -1.06396 + 1.32497 I* } , { *x -> 0.331342* } ,

{ *x -> 0.898286 - 0.488128 I* } ,

{ *x -> 0.898286 + 0.488128 I* } }

Observe que algumas raizes são complexas e que ***I*** *representa a unidade imaginária.*

O resultado do comando **Solve** é uma lista de substituições do tipo

{ x1 -> valor1 , x2 -> valor2 , ... }

que pode ser usada logo em seguida mediante o comando

expressão em x1, x2 ... /. %

que substitui **x1** por **val1**, **x2** por **val2**, **...**, na **expressão em x1, x2 ...** A saída de um comando **Solve** também pode ser usada posteriormente, atribuindo seu valor a uma variável ou com o comando

expressão_de_x /. Out[n]

onde **n** é o número da saída gerada pelo **Solve**.

O **Reduce**, em lugar de uma lista de substituições, gera equações (==) e inequações (**!=**), combinadas com os operadores lógicos **&&** (conector lógico "e") e || (conector lógico "ou").

Exemplo 6.10 *Inicie uma nova sessão e*

Comande: ***Solve [x^2 + x - 6 == 0, x]***

Resposta: Out[1]= { { x -> -3 } , { x -> 2 } }

Comande: ***x^3 - 5 /. %***

Entenda que neste comando solicitamos o cálculo da expressão ***x^3 - 5*** *para os valores de* ***x*** *obtidos no comando* ***Solve*** *anterior. A resposta será uma lista de valores. O primeiro valor da lista corresponde à atribuição x -> -3 e o outro à atribuição x -> 2.*

Resposta: Out[2]= { -32, 3 }

Comande: ***x^5 + x /. Out[1]***

Entenda: ordenamos que o sistema calcule ***x^5 + x*** *fazendo* ***x*** *igual aos valores obtidos pelo comando Solve emitido em In[1]:= e cuja saída está em Out[1]=.*

Resposta: { -246, 34 }

O comando **Solve** foi projetado para fornecer soluções gerais das equações. Ele descarta todas as soluções cuja existência dependa de um valor particular de algum parâmetro. Por outro lado, o comando **Reduce** analisa todas as soluções possíveis de um sistema de equações, inclusive as que exigem um valor especial de algum parâmetro. No caso de haver restrições aplicáveis aos parâmetros que aparecem numa solução, o **Reduce** as coloca na resposta.

Exemplo 6.11 *Vamos ilustrar o uso dos comando* ***Solve*** *e* ***Reduce****, resolvendo a equação* $(ax)^2 - 1 = 0$*, em relação à variável* x*.*

Comande: ***Clear[a , x]***

Comande: ***Solve[(a * x)^2 - 1 == 0 , x]***

Resposta:

$$\{\{x->-\frac{1}{a}\},\quad \{x->\frac{1}{a}\}\}$$

Comande: ***Reduce[(a * x)^2 - 1 == 0 , x]***

Resposta:

$$a\ !=0\quad \&\&\quad x=-\frac{1}{a}\quad \|\quad a\ !=0\quad \&\&\quad x=\frac{1}{a}$$

Exemplo 6.12 *Se um objeto for atirado para o ar com uma velocidade de* 20 *m/s, sob um ângulo de* 60^0 *em relação ao plano horizontal e, admitindo a aceleração da gravidade igual a* 10 m/s^2, *a equação cartesiana deste movimento será* $y=\sqrt{3}x-5x^2/100$. *Faça o gráfico da trajetória.*

Comande: ***y[x_] = Sqrt[3] x - x^2 / 20***

Resposta: Sqrt[3] x - x^2 / 20

Para determinar o ponto que o objeto toca o solo,

Comande: ***var = Solve[y[x] == 0 , x]***

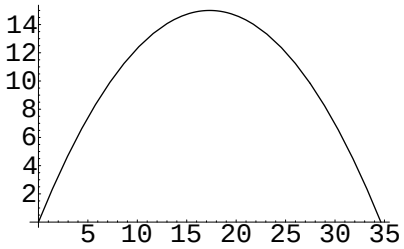
Resposta: { { $x->0$ } , { $x->20$ *Sqrt[3]* } }

Comande: ***var[[2 , 1]]***

Resposta: $x->20$ *Sqrt[3]*

Comande: ***Plot[y[x] , x , 0 , 20 Sqrt[3]]***

Resposta:



Pode-se também definir a trajetória como segue

Comande: ***y = Sqrt[3] x - x^2 / 20 ;***

e, para calcular o valor de ***y*** *para algum valor de* ***x****, use o comando de substituição. Para calcular* ***y*** *em* ***x = 1,***

Comande: ***y /. x -> 1***

Resposta: – 1 / 20 + Sqrt[3]

Vamos descrever o **Roots** e o **ToRules**, obtidos como resposta a um comando **Solve**. A função

Roots[equação , var]

busca os valores exatos das raizes, dando a resposta na forma de uma expressão lógica. Cada raiz é fornecida na forma de uma equação. Quando **Roots** não for capaz de obter as raizes exatas, use

NRoots[equação , var]

para obter os valores numéricos aproximados das raizes. Querendo transformar a resposta do **Roots** em regras de atribuição, use

{ ToRules[%] } ou **{ ToRules[Out[n]] }**

onde **n** é o número da resposta gerada pelo **Roots.**

Exemplo 6.13 *Vamos construir um polinômio cujas raizes são* 1, 2, 3, 4 *e no qual o coeficiente do termo de maior grau é unitário.*

Comande: ***p[k_] = Product[x – n , { n , 1 , k }]***

Comande: ***p[4]***

Resposta: $(-4+x)\,(-3+x)\,(-2+x)\,(-1+x)$

Comande: ***Expand[p[4]]***

Resposta: $24 - 50\,x + 35\,x^2 - 10\,x^3 + x^4$

Para obter as raizes deste polinômio

Comande: ***Roots[p[4] == 0 , x]***

Resposta: $x == 4 \parallel x == 1 \parallel x == 3 \parallel x == 2$

Comande: ***{ ToRules[%] }***

Resposta: { { $x -> 4$ } , { $x -> 1$ } , { $x -> 3$ } , { $x -> 2$ } }

6.8 Equações transcendentais

Os comandos **Solve** e **Reduce**, embora se apliquem preferencialmente a equações lineares e polinomiais, podem resolver algumas equações transcendentais. Quando não for possível obter a solução exata, pode-se ao menos obter um resultado numérico aproximado. Deste tópico cuidará a próxima seção.

Exemplo 6.14 *Analisemos os resultados obtidos com algumas equações transcendentais.*

Comande: ***Clear[x , y , f]***

Comande: ***Solve[Exp[x] == y , x]***

Mensagem: Solve : : ifun :

Warning : Inverse functions are being used by

Solve, so some solutions may not be found.

Resposta: { { $x -> Log[\ y\ ]$ } }

Depois de nos fornecer uma mensagem dizendo que o teorema da função inversa foi usado e que alguma solução pode não ter sido obtida, o sistema nos fornece a função Log que é a inversa da Exp.

Comande: ***Solve[Tan[x] == x , x]***

Mensagem: Solve : : ifun :

Warning: Inverse functions are being used by

Solve, so some solutions may not be found.

Solve : : tdep :

The equations appear to involve transcendental functions

of the variables in an essentially non-algebraic way.

Resposta: Solve [Tan[x] == x , x]

Depois de nos dizer que a solução envolve funções transcendentais, o sistema nos fornece como resposta apenas o comando emitido.

Comande: ***Solve[f[x]^2 + 3 f[x] == y , x]***

Resposta:

$$\{\{x->f^{(-1)}[\frac{-3-\text{Sqrt}[9+4\,y]}{2}]\},\ \{x->f^{(-1)}[\frac{-3+\text{Sqrt}[9+4\,y]}{2}]\}\}$$

Comande: ***Solve [f[2 x + 3] == y , x]***

Resposta:

$$\{\{x->\frac{-3+f^{(-1)}[y]}{2}\}\}$$

Estes exemplos mostram que o Mathematica pode resolver equações usando o conceito de função inversa .

Em breve mostramos como obter raizes aproximadas de uma equação transcendental.

O Mathematica usa a notação

$\mathbf{f^{(-1)}\ [\ y\ ]}$

para indicar a inversa da função $y = f(x)$. De modo geral, quando f for uma função de **n** variáveis,

InverseFunction(f , k , n)

indica a função inversa de **f** em relação à **k**-ésima variável. Deste modo, **InverseFunction(f, 1, 2)** indica a inversa de **f** em relação à primeira variável num total de duas variáveis. Para indicar a derivada inversa em um ponto **y**, o Mathematica usa

InverseFunction(f , n , k) [y]

6.9 Solução numérica

Para calcular numericamente a solução de **equações lineares e polinomiais**, pode-se usar o

NSolve

que tem a mesma sintaxe do **Solve.** Para obter numericamente uma solução de uma **equação transcendental** na variável **x**, use preferencialmente o

FindRoot[le == ld , { x, x0 }]

onde **le** e **ld** são, respectivamente, o lado esquerdo e o lado direito da equação. O **FindRoot** toma o valor **x0** como ponto de partida na a busca da raiz. Este valor deve ser tão próximo quanto possível da raiz que se deseja obter. O **FindRoot** busca a raiz por um processo iterativo e, para atingir este objetivo, devemos fornecer o valor inicial para que a busca de uma raiz se inicie. Este é o papel do **x0**. O comando

FindRoot[le == ld , { x, val1, val2 }]

busca uma solução numérica da equação, tomando **val1** e **val2** como pontos de partida. Esta forma deve ser usada quando não for possível calcular a derivada simbólica da equação. Para obter a solução numérica de uma equação no intervalo [**xmin, xmax**], tomando **x0** como ponto de partida, use

FindRoot[le == ld , { x , x0 , xmin , xmax }]

Para que o **FindRoot** localize a raiz desejada, é preciso que o usuário tenha uma idéia aproximada da localização das raizes da equação **le == ld** e fornecer o valor inicial **x0** próximo da raiz desejada. Caso isto não ocorra, o **FindRoot** pode fornecer uma raiz diferente daquela desejada.

Para se obter uma estimativa da raiz que se pretende obter, pode-se fazer os gráficos do **le** e do **ld** em uma mesma figura e verificar visualmente o valor aproximado de **x** para o qual **le == ld**. Neste ponto os gráficos devem se cruzar.

Exemplo 6.15 *Vamos determinar uma raiz da equação* $\tan x = x$, *nas vizinhanças de* $\pi/4$.

Comande: ***FindRoot[Tan[x] == x , { x , Pi/4 }]***

Reposta: { $x -> 0.00717571$ }

6.10 Sistema de equações

O Mathematica pode resolver sistemas de equações algébricas lineares e uma classe ampla de sistemas de equações polinomiais. Os comandos que realizam esta tarefa são o

Solve [{ eq1 , eq2 , ... } , { x , y , ... }]

e o

Reduce [{ eq1 , eq2 , ... } , { x , y , ... }]

onde **eqk** é uma equação da forma

LadoEsquerdo == LadoDireito

e

{ x , y , ... }

é a lista de variáveis em relação às quais se deseja resolver o sistema. Observe que o sistema de equações está delimitado por um abrir e um fechar chaves. Quando temos uma única equação ou uma única variável, as chaves são dispensáveis.

Exemplo 6.16 *Para ilustrar o* ***Solve*** *e o* ***Reduce*** *na resolução de sistemas,*

Comande: ***Clear[a , x , y]***

Comande: ***Solve[{ 2x + y == 5 , x - 2y == 2 } , { x , y }]***

Resposta:

$$\{\ \{\ x->\ \frac{12}{5}\ ,\ y->\ \frac{1}{5}\ \}\ \}$$

Comande: ***Solve[{ a * x + y == 4 , x - y == 3 } , { x , y }]***

Resposta:

$$\{\ \{x->\ 3-\frac{-4+3a}{1+a}\ ,\ y->\ -(\frac{-4+3a}{1+a})\ \}\ \}$$

O Solve pode resolver um sistema mesmo quando ele é literal. Fizemos uso da multiplicação implícita no comando ***Solve*** *e portanto o espaço entre o* ***a*** *e o* ***x*** *indica multiplicação.*

Quando a equação for **inconsistente**, a resposta será simplesmente { }

Exemplo 6.17 *Vamos analisar o comportamento do* ***Solve*** *e do* ***Reduce*** *para sistemas inconsistentes.*

Comande: ***Clear[x , y]***

Comande: ***Solve[{ x == 2 , y == x } , x]***

Resposta: { }

O sistema acima é inconsistente para todo ***y*** *diferente de* ***2****. Todavia, o* ***Reduce*** *nos fornece a solução, indicando as restrições existentes.*

Comande: ***Reduce[{ x == 2 , y == x } , x]***

Resposta: $x == 2 \quad \&\& \quad y == 2$

Os comandos **Solve** e **Reduce** são usados principalmente para **sistemas lineares e polinomiais**. Para **sistemas envolvendo equações transcendentais**, não lineares, use de preferência o

FindRoot [{ le1 == ld1 , le2 == ld2 , ... } ,
{ x, x0 } , { y, y0 } , ...]

que usa um processo iterativo para determinar a solução dos sistema, a partir do ponto inicial $(x_0, y_0, \ldots)$.

6.11 Sistemas lineares

Embora se possa usar o **Solve**, o **Reduce** e o **FindRoots** para resolver um **sistema linear**, é mais conveniente usar o

LinearSolve[m , b]

para determinar a solução **x** de um sistema linear de equações da forma

m . x = b

onde **m** é uma matriz quadrada ou retangular e **b** é um vetor. A resposta a este comando será uma lista, cujos elementos são os valores do vetor **x**.

Exemplo 6.18 *Vamos resolver o sistema linear*

$$3x + 4y = 1$$
$$4x + 5y = 2$$

usando o ***LinearSolve****.*

Comande: ***m = { { 3 , 4 } , { 4 , 5 } } ; b = { 1 , 2 } ;***

Comande: ***LinearSolve[m , b]***

Resposta: { *3 , – 2* }

Esta lista nos fornece $x = 3$ *e* $y = -2$.

6.12 Eliminação de variáveis em um sistema

Pode-se eliminar uma ou mais variáveis de um sistema de equações, usando o comando

Eliminate [{ eq1 , eq2 , ... } , { x , y , ... }]

onde **eqk**, **k** = **1**, **2**, **...**, são equações do tipo

lado_esquerdo == lado_direito

sendo **x, y, ...,** a relação de variáveis que se deseja eliminar do sistema. Quando se deseja eliminar uma única variável, o abrir e o fechar chaves que delimitam as variáveis se tornam opcionais.

O **Solve** e o **Reduce** podem realizar a eliminação de um conjunto de variáveis de um sistema ao mesmo tempo em que o resolvem. Para resolver um sistema de equações **eq1**, **eq2**, **...**, nas variáveis **v1**, **v2**, **...**, enquanto elimina as variáveis **eli1**, **eli2**, **...**, comande

Solve[{ eq1 , eq2 , ... } , { v1 , v2 , ... } , { eli1 , eli2 , ... }]

ou

Reduce[{ eq1 , eq2 , ... } , { v1 , v2 , ... } , { eli1 , eli2 , ... }]

Exemplo 6.19 *Vamos eliminar a do sistema de equações*

$$\begin{aligned} xy - y + a &= 0 \\ 3a + 3xy + x^2y - 3y &= 1 \end{aligned}$$

Comande: ***Clear[x, y]***

Comande: ***Eliminate[{ x*y - y + a == 0 ,***

3a + 3x*y + y*x^2 - 3y == 1 } , a]

Resposta: $x^2y == 1$

Comande: ***Solve[{ x*y - y + a == 0 ,***

3a + 3x*y + y*x^2 - 3y == 1 } , x , a]

Resposta: { $\{x- > -\left(\frac{1}{\text{Sqrt[y]}}\right)$ } , { $x- > \left(\frac{1}{\text{Sqrt[y]}}\right)$ } }

Capítulo 7

Cálculo diferencial e integral

7.1 Limite

Para calcular o **limite** de uma **expressão** que depende de **x**, quando **x** tende para um valor **L**, emite-se o comando

Limit[expressão , x − > L]

onde o símbolo − > é formado pelo sinal de menos (−) seguido pelo sinal de maior (>)

Exemplo 7.1 *Vamos calcular dois limites fundamentais*

$$\lim_{x\to 0}\frac{\operatorname{sen}(x)}{x} \qquad \text{e} \qquad \lim_{x\to\infty}\left(1+\frac{1}{x}\right)^{x}$$

Digite: ***Limit[Sin[x] / x , x − > 0]***

Resposta: 1

Digite: ***Limit[(1+1/x)^x , x − > Infinity]***

Resposta: E

O valor exato deste limite é um número irracional que, no Mathematica, é representado pela letra ***E****. O seu valor aproximado com cinco casas decimais é* $2,71828$*.*

Para calcular os **limites laterais** de uma **expressão** que depende de **x**, quando **x** tende para um valor **L**, emita o comando

Limit[expressão , x −> L , Direction −> 1]

para calcular o limite lateral à esquerda e

Limit[expressão , x −> L , Direction −> -1]

para calcular o limite lateral à direita.

Exemplo 7.2 *Para calcular os limites à esquerda e à direita da função* $f(x) = 1/x$ *no ponto* $x = 0$,
Comande: ***Limit[1/x, x −> 0, Direction −> 1]***
Resposta: $-\infty$
que é o limite à esquerda de $1/x$ *no ponto* $x = 0$.
Comande: ***Limit[1/x, x −> 0, Direction −> −1]***
Resposta: ∞
que é o limite à direita de $1/x$ *no ponto* $x = 0$.

Para calcular **limites iterados**, tais como

$$\lim_{y\to b}\lim_{x\to a} f(x,y)$$

emite-se o comando

Limit[Limit[f[x , y] , x −> a] , y −> b]

Exemplo 7.3 *Prova-se que, quando*

$$\lim_{(x,y)\to(a,b)} f(x,y)\,,$$

existe, então os limites iterados

$$\lim_{x\to a}\lim_{y\to b} f(x,y) \qquad \text{e} \qquad \lim_{y\to b}\lim_{x\to a} f(x,y)$$

existem e são iguais. Deste modo, quando os limites iterados forem diferentes, $f(x,y)$ não tem limite em (a,b). Para mostrar que nem sempre os limites iterados são iguais, vamos calcular

$$\lim_{x\to 0}\lim_{y\to 0}\left[\frac{x-y+3xy}{x+y}\right] \qquad \text{e} \qquad \lim_{y\to 0}\lim_{x\to 0}\left[\frac{x-y+3xy}{x+y}\right]$$

Comande: ***f[x_ , y_] = (x - y + 3 x*y) / (x + y)***

Comande: ***Limit[Limit[f[x , y] , y -> 0] , x -> 0]***

Resposta: 1

Comande: ***Limit[Limit[f[x , y] , x -> 0] , y -> 0]***

Resposta: -1

7.2 Derivada

Para calcular a derivada de uma função **f[x]** que depende de uma variável **x**, basta comandar

```
D[ f [x] , x ]
```

e, para calcular as derivadas de ordem superior, use

```
D[ f [x] , { x , n } ]
```

onde **n** é a ordem da derivada.

Exemplo 7.4 *Para calcular*

$$\frac{d}{dx}\left(x^3+\log x\right) \qquad \text{e} \qquad \frac{d^2}{dx^2}\left[\operatorname{sen}\left(x^2\right)\right]$$

Comande: **D[xˆ3 + Log[x] , x]**

Resposta: $3x^2 + \dfrac{1}{x}$

Comande: **D[Sin[xˆ2] , { x , 2 }]**

Resposta: $2\,\mathrm{Cos}[\,x^2\,] - 4\,x^2\,\mathrm{Sin}[\,x^2\,]$

Se **f** possuir duas ou mais variáveis, o comando anterior calcula sua **derivada parcial** em relação a **x**. Para calcular

$$\frac{\partial^2 f}{\partial x \partial y}(x, y)$$

comande

```
D[ f [ x, y ] , x , y ]
```

e, para calcular as derivadas parciais de uma função **f** em relação às variáveis **x**, **y**, **...**, sendo que **k** vezes em relação a **x**, **n** vezes em relação a **y** e assim por diante, basta comandar

```
D[ f [ x, y ] , { x , k } , { y , n} , ... ]
```

7.3 Notação da derivada na saída

Nos operadores de derivação, quando houver uma função **f(x)** com relação funcional desconhecida, sua derivada primeira fica indicada por **f'[x]**, sua derivada segunda por **f"[x]** e, a partir da derivada terceira por $\mathbf{f}^{(n)}[x]$ onde **n** é a ordem da derivada. Quando a função depender de duas variáveis então a derivada parcial

$$\frac{\partial^{n+p} f}{\partial x^n \partial y^p}(x, y)\,.$$

será denotada por $\mathbf{f}^{(n,p)}[\mathbf{x}, \mathbf{y}]$.

Exemplo 7.5 *Vamos calcular*

$$\frac{\partial^2}{\partial y \partial x}\left[x^2 \mathrm{sen}(y)\right] \qquad \text{e} \qquad \frac{\partial^5}{\partial y^3 \partial x^2}\left[\mathrm{sen}(x)\cos(y)\right]$$

Comande: ***D[x^2 Sin[y] , x , y]***

Resposta: 2 x Cos[y]

Comande: ***D[Sin[x] Cos[y] , { x , 2 } , { y , 3 }]***

Resposta: –(Sin[x] Sin[y])

Exemplo 7.6 *Vamos solicitar que seja calculada a derivada parcial*

$$\frac{\partial^5}{\partial x^2 \partial y^3} f(x,y)$$

sem termos definido f anteriormente. Para garantir que as variáveis envolvidas estão indefinidas,

Comande: ***Clear[f, x, y]***

Comande: ***D[f[x , y] , { x , 2 } , { y , 3 }]***

Resposta:

$$f^{(2,3)}[\,x\,,\,y\,]$$

Exemplo 7.7 *Seja $f(x,y)$ uma função real de duas variáveis reais. Nem sempre as derivadas parciais*

$$D_{1,2}f(a,b) = D_1 D_2 f(a,b) = \frac{\partial^2 f}{\partial x \partial y}(a,b)$$

e

$$D_{2,1}f(a,b) = D_2 D_1 f(a,b) = \frac{\partial^2 f}{\partial y \partial x}(a,b)$$

serão iguais. Quando $D_1 f$, $D_2 f$, $D_{1,2}f$, $D_{2,1}f$ existirem em um aberto que contém (a,b) e as derivadas parciais $D_{1,2}f$, $D_{2,1}f$ forem contínuas em (a,b), então $D_{1,2}f(a,b) = D_{2,1}f(a,b)$. Vamos apresentar um exemplo em que $D_{1,2}f(a,b) \neq D_{2,1}f(a,b)$. Seja $f(x,y)$ definida por $f(0,0) = 0$ e $f(x,y) = xy(x^2 - y^2)/(x^2 + y^2)$ quando $(x,y) \neq (0,0)$. Vamos mostrar que, para esta função,

$$D_{1,2}f(0,0) \neq D_{2,1}(0,0)\,.$$

Defina: ***f[x_ , y_] = x * y (x^2 − y^2) / (x^2 + y^2)***

Comande: ***dfx = D[f[x , y] , x] /. x − > 0***

Resposta: $-y$

Este é o valor de $D_1 f(0, y)$*, isto é, a derivada de* f *em relação a* x*, ao longo do eixo* y*.*

Comande: ***dfy = D[f[x , y] , y] /. y − > 0***

Resposta: x

Este é o valor de $D_2 f(x, 0)$*, isto é, a derivada de* f *em relação a* y*, ao longo do eixo* x*.*

Comande: ***{ D[dfx , y] /. y − > 0 , D[dfy , x] /. x − > 0 }***

Resposta: $\{ -1 , 1 \}$

Os elementos desta lista nos fornecem os valores de $D_{2,1} f(0, 0)$ *e* $D_{1,2} f(0, 0)$*. Assim,* $D_{2,1} f(0, 0) = -1$ *e* $D_{1,2} f(0, 0) = 1$

Para calcular a **derivada total** de uma função **f** em relação à variável **x**, comande

Dt[f [x, y] , x]

Quando se emite este comando, todas as variáveis que aparecem em **f** são consideradas dependentes de **x**. Desejando especificar que uma das variáveis independe de **x**, use a opção Constants.

Exemplo 7.8 *Limpe as variáveis* ***x, y*** *e* ***z*** *com o comando* ***Clear[x, y, z]***

Comande: ***Dt[x^3 + y^2 + z , x]***

Resposta: $3\,x^2 + 2\,y\,\mathrm{Dt}\,[y,\, x] + \mathrm{Dt}\,[z,\, x]$

O termo ***Dt[y , x]*** *indica a derivada de* ***y*** *em relação a* ***x****, que não pôde ser calculada explicitamente, uma vez que não se conhece a relação funcional entre* ***y*** *e* ***x****. A mesma observação se aplica ao termo* ***Dt[z, x]****. Observe a diferença quando emitimos o comando que fornece a derivada parcial*

Comande: ***D[x^3 + y^2 + z , x]***

Resposta: $3\,x^2$

Como não definimos que ***y*** *e* ***z*** *são funções de* ***x****, a derivada de* ***y^2*** *e de* ***z*** *em relação a* ***x*** *são nulas.*

Na derivada total, podemos especificar que uma ou mais variáveis independe daquela em relação à qual se deriva. Para calcular a derivada total de x^3+y^2+z*, quando apenas y for função de* x*, emita o comando que segue.*

Comande: ***Dt[x^3 + y^2 + z , x , Constants -> { z }]***

Resposta: $3x^2 + 2y\ Dt[y,\ x,\ Constants-> \{z\}]$

Para calcular a **diferencial** de uma função **f**, basta comandar

Dt[f [x, y, ...]]

Exemplo 7.9 *Vamos calcular a diferencial de* $a^4 + \tan(x)$.

Comande: ***Dt[a^4 + Tan[x]]***

Resposta: $4\,a^3\,\mathrm{Dt}[a] + \mathrm{Dt}[x]\,\mathrm{Sec}[x]^{\,2}$

Na expressão acima, ***Dt[a]*** *e* ***Dt[x]*** *representam as diferenciais de* ***a*** *e* ***x****, respectivamente.*

Exemplo 7.10 *Para calcular a derivada de* $x^3 f(x)$,

Comande: ***Dt[x^3 f[x] , x]***

Resposta: $3x^2 f[x] + x^3 f'[x]$

7.4 Regra da cadeia

Sejam

u = u[x , y] , **w = w[x , y]**

e

F = F[u , w]

Podemos compor **u** e **w** em **F** para colocá-la em função de **x** e **y**. Podemos usar os operadores de derivação para aplicar a regra da cadeia e calcular as derivadas parciais de **F** em relação a **x** e **y**. Para calcular

$$\frac{\partial F}{\partial x}(x,y) = \frac{\partial F}{\partial u}(u,w)\frac{\partial u}{\partial x}(x,y) + \frac{\partial F}{\partial w}(u,w)\frac{\partial w}{\partial x}(x,y)$$

onde $u = u(x,y)$ e $w = w(x,y)$, comande

D[F , x]

Se $x = X(r,s)$, $y = Y(r,s)$ e $z = Z(x,y)$, podemos compor as funções para obter $z = F(r,s) = Z(X(r,s), Y(r,s))$. Pela regra da cadeia,

$$\frac{\partial z}{\partial r} = \frac{\partial F}{\partial r} = \frac{\partial Z}{\partial x}\frac{\partial X}{\partial r} + \frac{\partial Z}{\partial y}\frac{\partial Y}{\partial r}\,.$$

A aplicação da regra da cadeia com o Mathematica é imediata. Basta definir as variáveis **x**, **y**, **z** comandando em seguida **D[z, r]** para calcular $\partial z/\partial r$ e **D[z, s]** para calcular $\partial z/\partial s$.

Exemplo 7.11 *Sabendo que* $x = r^2 + s^2$, $y = \operatorname{sen}(5r + 2s)$ *e* $z = x + 3y$, *vamos calcular* $\partial z/\partial r$. *Limpe as variáveis* ***x, y, z, r*** *e* ***s*** *com o comando* ***Clear[x, y, z, r, s]*** *e*

Comande: ***x = r^2 + s^2 ;***

Comande: ***y = Sin[5r + 2s] ;***

Comande: ***z = x^2 + 3y***

Resposta: $(r^2 + s^2)^2 + 3\,\mathrm{Sin}[5r + 2s]$

Observe que o Mathematica efetuou a composição das funções envolvidas.

Comande: ***D[z , r]***

Resposta: $4r(r^2 + s^2) + 15\,\mathrm{Cos}[5r + 2s]$

7.5 Integral

Para calcular a **integral indefinida**

$$\int f(x)\,dx$$

comande

Integrate[f[x] , x]

e, para calcular a **integral definida**

$$\int_a^b f(x)\,dx$$

comande

Integrate [f[x] , { x , a , b }]

Exemplo 7.12 *Execute os comandos*

Comande: ***Integrate [Log[x] , x]***

Resposta: $-x + x\,Log[\,x\,]$

Comande: ***Integrate[1/x , { x , 1 , 2 }]***

Resposta: $Log[\,2\,]$

A integral de $1/x$ *desde* 1 *até* 2 *é igual ao logaritmo natural de* 2. *Para obter uma aproximação de* ***Log[2]****, com seis algarismos significativos,*

Comande: ***N[%]***

Resposta: 0.693147

7.6 Fórmula de Leibniz

Ao efetuar uma integral do tipo

$$\int_{a(t)}^{b(t)} f(t,x)\,dx$$

obtemos uma função de t. Conseqüentemente, podemos falar em derivada desta função em relação a t. A regra de Leibniz nos diz que

$$\frac{d}{dt}\int_{a(t)}^{b(t)} f(t,x)\,dx = \int_{a(t)}^{b(t)} \frac{\partial f}{\partial t}(t,x)\,dx + f\left(t,b(t)\right)b'(t) - f\left(t,a(t)\right)a'(t)$$

Exemplo 7.13 *Vamos obter a regra de Leibniz.*

Comande: ***D[Integrate[f[t, x] , { x, a[t], b[t] }] , t]***

Resposta: Integrate[$f^{(1,0)}[t,\,x\,],\{\,x,\,a[\,t\,],\,b[\,t\,]\,\}\,] - f[\,t,\,a[\,t\,]\,]\,a'[\,t\,]$
$+f[\,t,\,b[\,t\,]\,]\,b'[\,t\,]$

A expressão $f^{(1,0)}[\,t,\,x\,]$ *representa* $\partial f(t,x)/\partial t$.

7.7 Integrais duplas e triplas

Para calcular a **integral dupla**

$$\iint_R f(x,y)\,dx\,dy$$

onde R é o retângulo $[\mathbf{xmin},\mathbf{xmax}]\times[\mathbf{ymin},\mathbf{ymax}]$ comande

Integrate [f[x , y] , { x , xmin , xmax } , { y , ymin , ymax }]

Exemplo 7.14 *Execute o comando*

Comande: ***Integrate[x^2 + y^2 , { x , 0 , 1 } , { y , 0 , 1 }]***

Resposta: $\frac{2}{3}$

Para calcular a **integral tripla**

$$\iiint_V f(x,y,z)\,dx\,dy\,dz$$

onde V é o paralelepípedo [**xmin, xmax**]×[**ymin, ymax**]×[**zmin, zmax**] comande

Integrate [f[x , y , z] , { x , xmin , xmax } ,
{ y , ymin , ymax } , { z , zmin , zmax }]

7.8 Regiões não retangulares

Para calcular a integrar dupla

$$\iint_D f(x,y)\,dx\,dy$$

onde D é uma região do plano definida por

$$D = \left\{ (x,y) \in \mathbf{R}^2 \quad \text{tais que} \quad h(x) \le y \le k(x) \quad \text{com} \quad a \le x \le b \right\}$$

usamos o resultado que permite transformar uma integral dupla em duas integrais iteradas

$$\iint_D f(x,y)\,dx\,dy = \int_a^b \int_{h(x)}^{k(x)} f(x,y)\,dy\,dx$$

e comandar

Integrate[Integrate[f[x , y] , { y , h[x] , k[x] }] , { x , a , b }]

Exemplo 7.15 *Vamos calcular* $\iint_D xy\,dx\,dy$ *sendo* D *o conjunto dos pontos* (x,y) *do plano tais que* $0 \le x \le \pi/2$ *e* $0 \le y \le \operatorname{sen}(x)$. *Limpe as variáveis* ***x*** *e* ***y*** *com o*

Comande: ***Integrate[Integrate[x * y , { y, 0, Sin[x] }] ,***
{ x, 0, Pi/2 }]

Resposta: $\dfrac{1}{16} + \dfrac{1 + Pi^2/2}{16}$

7.9 Integração numérica

Para calcular integrais definidas é mais eficiente usar o **NIntegrate**, que calcula a integral numericamente. Sua sintaxe é idêntica à do **Integrate**. Para calcular numericamente uma integral dupla num retângulo [**xmin, xmax**] × [**ymin, ymax**], usamos

NIntegrate[f[x] , { x , xmin , xmax } , { y , ymin , ymax }]

Exemplo 7.16 *Vamos calcular algumas integrais.*

Comande: ***NIntegrate[1/x , { x , 1 , 2 }]***

Reposta: 0.693147

Comande: ***NIntegrate[x^2 + y^2 , { x , 0 , 1 } , {y , 0 , 1 }]***

Resposta: 0.666667

que é o valor aproximado de 2/3 *com seis casas decimais.*

7.10 Resíduo de uma função complexa

Para determinar o resíduo de uma função complexa $f(z)$ num ponto singular z_0, comande

Residue[f[z] , { z , z0 }]

Exemplo 7.17 *Vamos calcular o resíduo da função* $f(z) = \cos z/(z - 3i)$ *no ponto* $z = 3i$.

Comande: ***Residue[Cos[z] / (z – 3 I) , { z , 3 I }]***

Resposta: Cosh[3]

7.11 Minimização de funções

A função

FindMinimum[f[x] , { x , x0 }]

calcula um ponto de mínimo local de **f[x]** iniciando a busca no ponto **x0**. A função

FindMinimum[f[x] , { x , { x0 , x1 } }]

calcula um ponto de mínimo local de **f[x]** tomando **x0** e **x1** como pontos iniciais. A função

FindMinimum[f[x] , { x , x0 , a , b }]

calcula um ponto de mínimo local de **f[x]** partindo de **x0** e interrompendo os cálculos quando a busca sair do intervalo [**a, b**]. A função

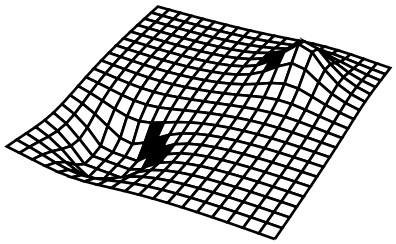
FindMinimum[f[x , y , ...] , { x , x0 } , { y , y0 } , ...]

calcula um ponto de mínimo de uma função de várias variáveis **f[x, y, ...]** iniciando a busca no ponto **(x0, y0, ...)**.

Exemplo 7.18 *Vamos determinar o ponto de mínimo da função*

$$\frac{10}{1+5(-1+x)^2+5(-1+y)^2}-\frac{20}{1+5(1+x)^2+5(1+y)^2}$$

cujo gráfico apresentamos abaixo e que possui um valor mínimo nas proximidades do ponto $(-1,-1)$.



Comande: ***ratio = 10 / (1 + 5 (x–1)^2 + 5 (y–1)^2) –***

20 / (1 + 5 (x+1)^2 + 5 (y+1)^2) ;

Vamos colocar $(0,0)$ *como ponto de partida para a busca do ponto de mínimo.*

Comande: ***FindMinimum[ratio , { x , 0 } , { y , 0 }]***

Resposta: { *–19.7562 ,* { $x->$ *–1.00059 ,* $y->$ *–1.00059* } }

Podemos calcular o valor da expressão ***ratio*** *no ponto* $(-1,1)$. *Para tanto,*

Comande: ***N[ratio /. { x -> –1 , y -> –1 }]***

Resposta: –19.7561

que observamos ser bem próximo do valor mínimo.

Nesta seção, as funções da forma

$$\mathbf{f[\ x\ ,\ y\ ,\ z\ ,\ ...] = k1\ x + k2\ y + k3\ z + \cdots}$$

com **k1**, **k2**, **k3**, **...**, constantes, serão chamadas **lineares**. Para minimizar ou maximizar funções lineares sujeitas a **vínculos lineares** definidos por inequações da forma

$$\mathbf{a1\ x + a2\ y + a3\ z + \cdots + b} \leq 0$$

onde **a1**, **a2**, **a3**, **...** e **b** são constantes use, respectivamente,

ConstrainedMin[f , { ineq1, ineq2, ... } , { x, y, z, ... }]

e

ConstrainedMax[f , { ineq1, ineq2, ... } , { x, y, z, ... }]

onde **ineq1**, **ineq2**, **...**, são as inequações que definem os vínculos lineares.

Exemplo 7.19 *Vamos calcular o valor mínimo da função $x-3y+4z$ na região delimitada pelas desigualdades $x \geq 0$, $y \geq 0$, $z \geq 0$, $x+y+z \leq 4$, $x+y+z \geq 2$.*

Comande: ***ConstrainedMin[x - 3y + 4 z ,***

{ x >= 0 , y >= 0 , z >= 0 ,

x + y + z <= 4 , x + y + z >= 2 } , { x , y , z }]

Resposta: { *-12, { x -> 0, y -> 4, z -> 0 } }*

O mínimo ocorre no ponto $(0, 4, 0)$ e o valor da função neste ponto é -12.

7.12 Programação Linear

Estes problemas de minimizar funções lineares submetidas a vínculos lineares nos leva a um ramo da Matemática chamado de Programação Linear. Nesta ciência, o **problema central** consiste em, dados os vetores **b**, **c** e a matriz **m**,

determinar o vetor **x**

que minimiza o produto escalar **c**·**x**,

satisfazendo os vínculos $\mathbf{x} \geq \mathbf{0}$ e $\mathbf{m}\cdot\mathbf{x} \geq \mathbf{b}$.

Para resolver este problema, use

LinearProgramming[c , m , b]

Exemplo 7.20 *O problema do exemplo anterior pode ser resolvido neste contexto, com*

$c = \{\ 1, -3, 4\ \}$, $m = \{\ \{\ -1, -1, -1\ \}\ ,\ \{\ 1, 1, 1\ \}\ \}$ *e* $b = \{\ -4, 2\ \}$.

Para resolvê-lo,

Comande: $c = \{\ 1\ ,\ -3\ ,\ 4\ \}\ ;\ b = \{\ -4\ ,\ 2\ \}\ ;$

Comande: $m = \{\ \{\ -1\ ,\ -1\ ,\ -1\ \}\ ,\ \{\ 1\ ,\ 1\ ,\ 1\ \}\ \}\ ;$

Comande: $x = LinearProgramming[\ c\ ,\ m\ ,\ b\ ]$

Resposta: { 0 , 4 , 0 }

Capítulo 8

Somas, produtos e séries

8.1 Somas

Para calcular

$$\sum_{n=\mathrm{ni}}^{\mathrm{nf}} a(n) = a(\mathrm{ni}) + a(\mathrm{ni}+1) + a(\mathrm{ni}+2) + \cdots + a(\mathrm{ni}+\mathrm{k})$$

onde **k** é o maior inteiro para o qual **ni + k** $\leq$ **nf**, comande

Sum[**a**[**n**] , **{ n , ni , nf }**]

No somatório, a variável **n** é acrescida de uma unidade a cada nova parcela. Se **ni = 1**, o termo entre chaves pode ser abreviado e reescrito na forma **{n, nf}.** Para calcular

$$a(\mathrm{ni}) + a(\mathrm{ni}+\mathrm{dn}) + a(\mathrm{ni}+2\mathrm{dn}) + \cdots + a(\mathrm{ni}+\mathrm{kdn})$$

comande

Sum[**a**[**n**] , **{ n , ni , nf, dn }**]

O último termo desta soma corresponde ao valor de **n=ni+k·dn** onde **k** é o maior inteiro para o qual **ni + k·dn ≤ nf**. Quando **ni = 1**, o termo entre chaves pode ser abreviado e escrito na forma **{n, nf, dn}.**

Para calcular

$$\sum_{n=\mathrm{ni}}^{\mathrm{nf}} \sum_{k=\mathrm{ki}}^{\mathrm{kf}} a(n,k)$$

comande

Sum[a[n , k] , { n , ni , nf } , { k , ki , kf }]

Para o índice **n** percorrer os valores entre **ni** e **nf**, recebendo acréscimos de **dn** unidades enquanto o **k** percorre os valores entre **ki** e **kf**, recebendo acréscimos de **dk** unidades, comande

Sum[a[n , k] , { n , ni , nf , dn } , { k , ki , kf , dk }]

Exemplo 8.1 *Siga o exemplo.*

Comande: **Sum[n , { n , 1 , 10 }]**

Resposta: 55

que é o valor de $\sum_{n=1}^{10} n$

Comande: **Sum [n , { n , 1 , 10 , 2 }]**

Resposta: 25

Observe: Nesta última soma, o **n** *recebe os valores* 1, 3, 5, 7, 9. *Observe que* **n** *não recebe o valor* 10 *pois se acrescentarmos* 2 *unidades a* 9, *obtemos* 11, *que é maior que* 10.

Comande: **Sum [2n − 1 , { n , 1 , 5 }]**

Resposta: 25

Como era de se esperar, reproduzimos o resultado anterior.

Comande: **Sum[k^n , { n , 0 , 5 }]**

Resposta: $1 + k + k^2 + k^3 + k^4 + k^5$

Observe a possibilidade de se realizar somas literais, onde **k** *é o nome de uma variável.*

Comande: **Sum [1 / n^2 + k , { n , 1 , 10} , { k , 0 , 10 }]**

Resposta: $\frac{720195619}{1270080}$

8.2 Séries

O Mathematica possui a capacidade de calcular a soma de séries. Para tanto, basta colocar **nf** ou **kf** igual a infinito (**Infinity**) como mostra o exemplo abaixo. O **ni** e o **ki** podem ser menos infinito (–**Infinity**).

Exemplo 8.2 *Comande:* **Sum [1 / n^2 , { n, 1, Infinity }]**

Resposta: Sum[n^{-2}, { n , 1 , Infinity }]

Observe *que a resposta foi uma mera repetição do comando porque o resultado exato é um número irracional. Para obter o valor aproximado desta soma,*

Comande: **N[%]**

Resposta: 1.64493

Pode-se pedir para calcular a soma de uma série divergente. Em resposta a tal comando, aparecem mensagens de erro e uma resposta desprovida de sentido. Sabemos que $\sum_{n=1}^{\infty} n^{-1}$ *diverge. Vamos solicitar o valor numérico desta soma.*

Comande: **N[Sum [1 / n , { n , 1 , Infinity }]]**

Em resposta a este comando, obtemos uma série de mensagens de alerta e, no final, uma resposta que nada tem a ver com o valor da série. As mensagens são

NIntegrate::slwcon: Numerical integration converging too slowly; suspect one of the following: singularity, oscillatory integrand, or insufficient WorkingPrecision.

NIntegrate::ncvb: NIntegrate failed to converge to prescribed accuracy after 7 recursive bisections in n near $n = 2.28833 \times 10^{56}$.

A resposta obtida é 23953.7

8.3 Notação das iterações

Os exemplos acima mostraram a notação usada pelo Mathematica para realizar iterações, escrevendo-se a variável a ser iterada, seus limites e seus acréscimos entre chaves. Esta notação é geral e será utilizada sempre que se desejar um cálculo repetido, onde uma ou mais variáveis poderão sofrer variações durante as iterações. Esta notação também é utilizada quando há iteração sem mudança no valor da variável.

A equipe que desenvolveu este programa estabeleceu que, nas iterações,

{ n , ni , nf }

faz com que as operações se repitam à medida que **n** percorre os valores

ni + 1 , **ni + 2** , **ni + 3** , **...** , **ni + k** ,

onde **k** é o maior inteiro para o qual **ni + k ≤ nf.** Observe que o **n** vai recebendo acréscimos unitários. Os termos entre colchetes

{ n , nf }

têm o mesmo efeito que o comando de repetição acima onde se admite implicitamente que **ni=1**. Com o formato

{ n , ni , nf , dn }

queremos indicar que o **n** percorre os valores de **ni** até **nf**, recebendo acréscimos de **dn** unidades, isto é, **n** recebe sucessivamente os valores

ni , **ni + dn** , **ni + 2dn** , **ni + 3dn** , **...** , **ni + k·dn** ,

onde **k** é o maior inteiro para o qual **ni + k·dn** $\leq$ **nf**.

Nota 8.1 *os limites* **ni**, **nf**, *e os acréscimos* **dn** *podem ser inteiros, racionais, reais ou complexos. No caso de* **ni** *ou* **nf** *serem complexos, deve-se cuidar para que* **dn** *seja um complexo de modo tal que os pontos* **ni + k·dn** (**k** *inteiro*) *estejam sobre a reta que liga os pontos* **ni** *e* **nf**.

Com a especificação de repetição

{ num }

a operação é repetida um número **num** de vezes, sem incrementar variável alguma.

Exemplo 8.3 *Execute os comandos.*

Comande: **Sum[3 , { 5 }]**

Resposta: 15

Observe *que o resultado corresponde a* $3+3+3+3+3$.

Comande: **Sum[n , { 5 }]**

Resposta: 5 n

8.4 Somas simbólicas

Existe um pacote computacional encarregado de efetuar somas simbólicas, isto é, somas cujos limites sejam variáveis. Para carregar este pacote, comande

<< Algebra‘SymbolicSum‘

onde ‘ é o acento grave, que fica na mesma tecla que o til (˜).

Para calcular uma soma

$$\sum_{n=\mathrm{j}}^{\mathrm{k}} a(n)$$

onde **j** e **k** são variáveis, comande

SymbolicSum[a[n] , { n , j , k }]

Sendo **j = 1**, podemos simplificar os termos entre chaves e escrever apenas **{n, k}**, seguindo a convenção previamente estabelecida.

Exemplo 8.4 *Vamos carregar o pacote de soma simbólica e calcular as somas*

$$\sum_{n=1}^{k} n\,, \qquad \sum_{n=1}^{k} n^2\,, \qquad \sum_{n=1}^{k} n^3\,.$$

Comande: **<< Algebra‘SymbolicSum‘**

Comande: **SymbolicSum[n , { n , 1 , k }]**

Resposta: $\dfrac{k(1+k)}{2}$

Exemplo 8.5 *Comande:* **SymbolicSum[n^2 , { n , 1 , k }]**

Resposta: $\dfrac{k(1+k)(1+2k)}{6}$

Comande: **SymbolicSum[n^3 , { n , 1 , k }]**

Resposta: $\dfrac{k^2(1+k)^2}{4}$

8.5 Produtos

Para calcular

$$\prod_{n=\mathrm{ni}}^{\mathrm{nf}} p(n) = p(\mathrm{ni})p(\mathrm{ni}+1)p(\mathrm{ni}+2)\cdots p(\mathrm{ni}+k)$$

onde **k** é o maior inteiro para o qual **ni + k ≤ nf**, comande

Product[p[n] , { n , ni , nf }]

e, para solicitar acréscimos iguais a **dn** a cada novo fator no índice **n**, comande

Product[p[n] , { n , ni , nf , dn }]

Para calcular o produtório duplo

$$\prod_{n=ni}^{nf} \prod_{k=ki}^{kf} p[n,k]$$

comande

Product[p[n , k] , { n , ni , nf } , { k , ki , kf }]

e, para que a cada iteração o **n** receba acréscimos iguais a **dn** e o **k** receba acréscimos iguais a **kn**, comande

Product[p[n , k] , { n , ni , nf , dn } , { k , ki , kf , dk }]

Exemplo 8.6 *Para calcular o produto* $2 \cdot 3 \cdot 4 \cdot \ldots \cdot 10 \cdot 11$,

Comande: **Product[a , { a , 2 , 11 }]**

Resposta: 39 916 800

Comande: **Product[n , { n , 2 , 11 , 2 }]**

Resposta: 3840

Este é o resultado do produto $2 \cdot 4 \cdot 6 \cdot 8 \cdot 10$.

Comande: **Product[a + b , { a , 1 , 10 } , { b , 1 , 10 }]**

Resposta:

28743918520213823413367094265334500185016781277678 69 \

34875132576597843557725765632000000000000000000000

Observe *o traço invertido no final da primeira linha da resposta. Ele indica que a segunda linha é uma continuação da anterior.*

Pode-se calcular **produtos infinitos**, colocando-se, como nas somas, **nf**, **kf** iguais a infinito (**Infinity**) ou **ni**, **ki** iguais a menos infinito (–**Infinity**). O pacote **Algebra'SymbolicSum'** pode manipular produtos infinitos e simbólicos.

Exemplo 8.7 *Siga o exemplo. Inicie uma nova sessão do Mathematica e*

Comande: **Product[1 + 1 / n^2 , { n, 1, Infinity}]**

Resposta: Product[1 + n^{-2} , { n , 1 , Infinity }]

Pode-se obter o valor numérico desta soma. Para tanto,

Comande: **N[%]**

Resposta: 3.67608

O valor exato deste produto é $\sinh(\pi)/\pi$. *Para obter este resultado,*

Comande: << **Algebra'SymbolicSum'**

Comande: **Product[1 + 1 / n^2 , { n, 1, Infinity}]**

Resposta: Sinh[Pi] / Pi

Para calcular

$$\prod_{k=1}^{n} k \qquad \text{e} \qquad \prod_{k=1}^{n} (2k+1)/k$$

Comande: **Product[k , { k , 1 , n }]**

Resposta: Gamma[1 + n]

Comande: **Product [(2k + 1) / k , { k , 1 , n }]**

Resposta: $\dfrac{2\ 2^{3n}\,\text{Gamma}[\,3/2+n\,]}{4^n\,\text{Sqrt}[\,\text{Pi}\,]\,\text{Gamma}[\,1+n\,]}$

8.6 Somas e produtos numéricos

Quando desejamos o valor numérico de uma soma ou produto, podemos usar o

NSum e o **Nproduct**

cuja sintaxe é a mesma do **Sum** e do **Product**. Assim, sendo **n** número inteiro positivo, querendo calcular

$$\sum_{k=1}^{n} a(k) \qquad \text{ou} \qquad \prod_{k=1}^{n} p(k)$$

use, respectivamente,

```
NSum[ a[ k ] , { k , 1 , n } ]
```

e

```
NProduct[ p[ k ] , { k , 1 , n } ]
```

8.7 Série de potências

Para obter o desenvolvimento de uma função **f(x)** em uma **série de potências** de **x**, até o termo de ordem **n**, em torno do ponto **x0**, comande

```
Series[ f[ x ] , { x , x0 , n } ]
```

Pode-se transformar a série num polinômio comum, com o comando **Normal**. Sendo

pot = Series[f[x] , { x , 0 , n }]

então

Normal[pot]

é o polinômio obtido quando se trunca a série logo após o termo de ordem **n**. O exemplo abaixo ilustra a diferença entre a série truncada e o polinômio correspondente.

Exemplo 8.8 *Vamos exemplificar os comandos anteriores trabalhando com a série da função exponencial.*

Comande: **exponencial = Series[Exp[x] , { x , 0 , 4}]**

Resposta: $1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \mathrm{O}\,[x]^5$

Entenda: A expressão $\mathrm{O}[x]^5$ *indica que os termos não apresentados são da ordem de* x^5*.*

Comande: **exponencial^2**

Resposta: $1 + 2\,x + 2\,x^2 + \frac{4\,x^3}{3} + \frac{2\,x^4}{3} + \mathrm{O}\,[x]^5$

Note que o quadrado da série foi truncado após o termo da ordem de x^4*. Isto ocorreu porque a série é conhecida apenas até o termo em* x^4*.*

Comande: **poli = Normal[%]**

Resposta: $1 + 2\,x + 2\,x^2 + \frac{4\,x^3}{3} + \frac{2\,x^4}{3}$

Observe que o termo indicador da ordem de truncamento desapareceu, transformando a série num polinômio comum.

Comande: **poli^2**

Resposta: $\left(1 + 2\,x + 2\,x^2 + \frac{4\,x^3}{3} + \frac{2\,x^4}{3}\right)^2$

Comande: **Expand[%]**

Resposta:

$$1 + 4x + 8x^2 + \frac{32}{3}x^3 + \frac{32}{3}x^4 + 8x^5 + \frac{40}{9}x^6 + \frac{16}{9}x^7 + \frac{4}{9}x^8$$

Enquanto trabalharmos com a série da **exponencial**, *ela sempre será truncada no termo de mesma ordem, que neste exemplo é* 4. *O polinômio, ao ser elevado ao quadrado, nos forneceu o polinômio correspondente do oitavo grau. Note que o polinômio obtido em resposta ao último comando não é a série de potências de* **Exp[x]^2** *truncada no termo de ordem* 8.

Comande: **Series[Exp[x]^2 , { x , 0 , 8 }]**

Resposta:

$$1 + 2\,x + 2\,x^2 + \frac{4\,x^3}{3} + \frac{2\,x^4}{3} + \frac{4\,x^5}{15} + \frac{4\,x^6}{45} + \frac{8\,x^7}{315} + \frac{2\,x^8}{315} + \mathrm{O}\,[x]^9$$

8.8 Mudança de variável

Seja **g(x)** uma função na variável **x**. Se quisermos substituir **x** por uma expressão em **t**, relacionada a **x** pela função **x = f(t)**, usamos a regra de substituição

g[x] /. x -> f[t]

Exemplo 8.9 *Vamos calcular a série de* **Exp[x]** *até o termo de ordem* **5** *e substituir* **x** *por* **t^2**.

Comande: **exponencial = Series[Exp[x] , { x , 0 , 5 }]**

Resposta: $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \mathrm{O}[x]^6$

Comande: **exponencial /. x -> t^2**

Resposta: $1 + t^2 + \frac{t^4}{2} + \frac{t^6}{6} + \frac{t^8}{24} + \frac{t^{10}}{120} + \mathrm{O}[t^2]^6$

Para derivar esta série em relação a **t**,

Comande: **D[%, t]**

Resposta: $2\,t + 2\,t\,t^2 + t\,t^4 + \frac{t\,t^6}{3} + \frac{t\,t^8}{12} + \mathrm{O}[t^2]^5$

8.9 Inversão de séries

Dada uma função $y = f(x)$, podemos obter seu desenvolvimento em séries de potências em torno do x_0 e gravar o resultado na variável **ser** com o comando

ser = Series[f[x] , { x , x0 , n }]

Obtida esta série, podemos calcular a série da inversa $x = f^{-1}(y)$ em torno do ponto $y_0 = f(x_0)$, até o termo de ordem **n** com o comando

InverseSeries[ser , y]

Exemplo 8.10 *Vamos obter a série de* $y = \exp(x)$ *em torno do ponto* $x = 0$, *até o termo de ordem* 5, *solicitando em seguida a série da inversa* $x = \ln(y)$, *em torno do ponto* $y = \exp(0) = 1$.

Comande: **ser = Series[Exp[x] , { x , 0 , 5 }]**

Resposta: $1 + x + \dfrac{x^2}{2} + \dfrac{x^3}{6} + \dfrac{x^4}{24} + \dfrac{x^5}{120} + O[x]^6$

Comande: **InverseSeries[ser , y]**

Resposta:

$$(-1+y) - \frac{(-1+y)^2}{2} + \frac{(-1+y)^3}{3} - \frac{(-1+y)^4}{4} + \frac{(-1+y)^5}{5} + O[-1+y]^6$$

Para calcular o valor aproximado de $\ln 1.2$,

Comande: **Normal[%] /. y -> 1.2**

Resposta: 0.182331

Comande: **Log[1.2]**

Resposta: 0.182322

Vemos que a aproximação oferecida pela série é razoável e pode ser melhorada aumentando-se a ordem da série.

Capítulo 9

Equações diferenciais ordinárias

Para obter a solução geral da **equação diferencial**,

$$F(\, x,\, y(x),\, y'(x),\, y''(x),\, \ldots\,) = 0\,,$$

onde **x** é a variável independente e **y(x)** é a função a ser determinada, use o comando

DSolve [**F**[**x** , **y**[**x**] , **y'**[**x**] , **y"**[**x**] , **...**] == **0** , **y**[**x**] , **x**]

ou

DSolve [**F**[**x** , **y**[**x**], **y'**[**x**] , **y"**[**x**] , **...**] == **0** , **y** , **x**]

Numa **equação diferencial**, a função **y(x)** deve ser escrita na forma **y**[**x**]. As suas derivadas devem ser denotadas por **y'**[**x**], **y"**[**x**], e assim por diante. Observamos que em **y"**[**x**], o sinal **"** não são aspas mas sim dois acentos agudos consecutivos. Em geral, como nosso computador está preparado para lidar com o Português que possui acentos, para fazer o acento agudo aparecer na tela, pressione e solte a tecla que contém o acento teclando, em seguida, a barra de espaço. Não se deve usar as formas abreviadas **y**, **y'**, **y"**, para representar a função e suas derivadas. Para designar derivadas de ordem quarta, quinta ou de ordem superior, fica inconveniente usar linhas para designar a derivada. Nestes casos, pode-se denotar a derivada de ordem **n** de **y** em relação a **x** por

D[y[x] , { x , n }]

Exemplo 9.1 *Vamos resolver a equação* $y''(x) + y(x) = 0$.

Comande: ***Clear[x , y]***

Comande: ***DSolve [y"[x] + y[x] == 0 , y[x] , x]***

Resposta: { { *y[x] – > C[2] Cos[x] – C[1] Sin[x]* } }

Comande: ***DSolve [y"[x] + y[x] == 0 , y , x]***

Resposta: { { *y – > Function[x, C[2] Cos[x] – C[1] Sin[x]]* } }

Observe as constantes arbitrárias C[1] e C[2] que surgem na solução geral da equação de segunda ordem. Na solução obtida no primeiro comando, o ***y*** *foi considerado como função de* ***x****, ao passo que na resposta ao segundo comando,* ***y*** *foi considerada como função anônima com um argumento. As diferenças entre os dois casos foram analisados no capítulo sobre funções.*

Anote: Pode-se aproveitar este resultado em uma expressão posterior, como ilustraremos na continuação deste exemplo.

Para calcular a derivada da solução anterior e somá-la ao dobro dela,

Comande: ***y'[x] + 2 y[x] /. %***

Resposta: { *–(C[1] Cos[x]) – C[2] Sin[x] + 2 (C[2] Cos[x] – C[1] Sin[x])* }

Quando uma equação não linear tiver solução na forma explícita, pode-se obter sua solução com o ***DSolve****. Vamos calcular as soluções de*

$$\frac{dy}{dx} = \frac{y^2 + 2xy}{x^2}$$

Comande: ***DSolve[y'[x] – (y[x]^2 + 2 * x * y[x]) / (x^2) == 0 ,***

y[x] , x]

Resposta:

$$\{ \{ y[x] -> \frac{x^2}{-x + C[1]} \} , \{ y[x] -> 0 \} \}$$

9.1 Problema de valor inicial

Para resolver o problema de valor inicial

$$F\,[x, y(x), y'(x)] = 0\,, \qquad y(x_0) = A\,,$$

comande

```
DSolve[ { F[x , y[x] , y'[x] ] == 0 , y[ x0 ] == A } , y[x] , x ]
```

e, para resolver um problema de valor inicial envolvendo uma equação de segunda ordem, tal como

$$F\,[x, y(x), y'(x), y''(x)] = 0\,, \qquad y(x_0) = A\,, \qquad y'(x_0) = B\,,$$

comande

```
DSolve[ { F[ x , y[x] , y'[x] , y"[x] ] == 0 ,
          y[x0] == A , y'[x0] == B } , y[x] , x ]
```

A generalização para equações de ordem superior é evidente. As condições iniciais entram no comando como equações.

Exemplo 9.2 *Vamos resolver o problema de valor inicial*

$$y'(x) + 2y(x) = 0\,, \qquad y(1) = 3\,.$$

Comande: ***DSolve[{ y'[x] + 2 y[x] == 0 , y[1] == 3 } , y[x] , x]***

Resposta:

$$\{\ \{\ y[x] -> 3\,E^{2-2x}\ \}\ \}$$

Vamos agora resolver

$$y''(x) + 4y(x) = 0\,, \qquad y(1) = 3\,, \qquad y'(1) = 0$$

Comande: ***DSolve[{ y"[x] + 4 y[x] == 0 ,***
y[1] == 3 , y'[1] == 0 } , y[x] , x]

Resposta:

{ { y[x] -> 3 Cos[2] Cos[2x] + 3 Sin[2] Sin[2x] } }

9.2 Sistemas de equações diferenciais

Para resolver o sistema de equações diferenciais

$$\begin{aligned} F\left[x, y(x), y'(x), z(x), z'(x)\right] &= 0 \\ G\left[x, y(x), y'(x), z(x), z'(x)\right] &= 0 \end{aligned}$$

comande

```
DSolve[ { F(x, y[x], y'[x], z[x], z'[x]) == 0 ,
         G(x, y[x], y'[x], z[x], z'[x]) == 0 } , { y[x] , z[x] } , x ]
```

observando que a quebra de linhas no comando acima seguiu nossa preferência estética. O usuário poderá escolher a quebra de linha no comando, no local que melhor lhe convier. Preste muita atenção nos lugares em que as chaves são colocadas.

Exemplo 9.3 *Para resolver o sistema de equações diferenciais ordinárias*

$$y' = -2y - 4z, \qquad z' = -y + z$$

Comande: ***DSolve [{ y'[x] == -2 y[x] - 4 z[x] ,***
z'[x] == - y[x] + z[x] } , { y[x] , z[x] } , x]

Resposta:

$$\begin{aligned} \{\{ y[x] -> & \left(\frac{4}{5\,E^{3x}} + \frac{E^{2x}}{5}\right) \mathrm{C}[1] + \\ & \left(\frac{4}{5\,E^{3x}} - \frac{4\,E^{2x}}{5}\right) \mathrm{C}[2]\,, \\ z[x] -> & \left(\frac{1}{5\,E^{3x}} - \frac{E^{2x}}{5}\right) \mathrm{C}[1] + \\ & \left(\frac{1}{5\,E^{3x}} + \frac{4\,E^{2x}}{5}\right) \mathrm{C}[2] \}\} \end{aligned}$$

A sintaxe para sistemas de ordem superior, com mais equações e condições iniciais é evidente.

9.3 Solução numérica

Quando o Mathematica não obtiver a solução explícita de um problema de valor inicial, pode-se obter sua **solução numérica** com o

NDSolve[equações , y , { x , xmin , xmax }]

onde **equações** é uma lista de equações, contendo as equações diferenciais e as condições iniciais. O Mathematica usa um método de integração auto adaptativo, que diminui o passo automaticamente nos intervalos em que a solução sofre variações mais bruscas. Para evitar que o intervalo continue a ser dividido indefinidamente, quando a precisão não é atingida após um certo número de subdivisões, a integração numérica é interrompida e aparece na tela uma mensagem de alerta.

Exemplo 9.4 *Vamos resolver o problema de valor inicial*

$$y' = x^2 + y^2 \,, \qquad y(0) = 0.1,$$

com x no intervalo $[1, 1.8]$.

Comande: ***sol = NDSolve[{ y'[x] == x^2 + y[x]^2 ,***

y[0] == 0.1 } , y , { x , 0 , 1.8 }]

Resposta: { { y -> *InterpolatingFunction[{0., 1.8 } , < >]* } }

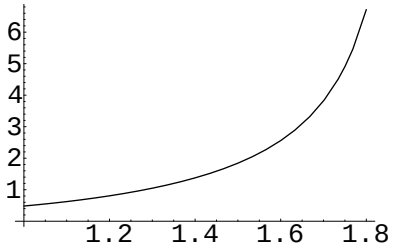
A resposta é fornecida em termos de uma função interpolada, cujo significado explicaremos em seguida. Podemos solicitar o valor da solução no ponto $x = 1.2$. Para tanto,

Comande: ***y[1.2] /. sol[[1]]***

Resposta: 0.80645

Comande: ***Plot[Evaluate[y[x] /. sol] , { x , 1 , 1.8 }]***

Resposta:



Comande: ***NDSolve[{ y'[x] == x^2 + y[x]^2 , y[0] == 0.1 } ,***

y , { x , 0 , 2 }]

Resposta: NDSolve :: ndsz :

At x = 1.94535, step size is effectively zero; singularity suspected.

{ { y −> InterpolatingFunction[{ 0., 1.94535 }, < >] } }

Obtivemos uma mensagem de alerta. Como a solução cresce muito e rapidamente nas proximidades de 1.94535, após um certo número de subdivisões do intervalo, o processo é abortado.

A solução numérica é dada na forma

{ { y −> InterpolatingFunction[{ 0., 1.94534 }, < >] } }

A função

InterpolatingFunction[{ a , b } , lista]

gera uma interpolação para uma **lista** de valores especificada no intervalo $[a, b]$. O **NDSolve**, ao integrar numericamente o problema de valor inicial, gera uma lista com os valores de (x_i, y_i), $i = 0, 1, ...$, correspondentes à solução e remete esta lista para a função de interpolação. Por ser grande, a **lista** é omitida na saída do **NDSolve.**

Para calcular o valor da função aproximada gerada pelo **InterpolatingFunction** num ponto **x**, basta comandar

InterpolatingFunction[{ a , b } , lista] [x]

Este foi o procedimento adotado no exemplo anterior, quando comandamos **y[1.2] /. sol**.

Exemplo 9.5 *Vamos calcular a solução numérica do sistema de equações diferenciais ordinárias*

$$\begin{aligned} x' + 3x - 4y &= 1/t \\ y' - 2x + 3y &= \operatorname{sen}(t) \\ x(1) &= 0 \\ y(1) &= 1 \end{aligned}$$

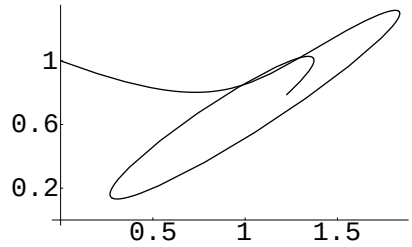
no intervalo $[1, 10]$ *e fazer o gráfico da curva parametrizada* $(x(t), y(t))$ *obtida pela integração deste problema de valor inicial.*

Comande: ***sol = NDSolve[{ x'[t] + 3 x[t] - 4 y[t] == 1 / t ,***
y'[t] - 2 x[t] + 3 y[t] == Sin[t] ,
x[1] == 0 , y[1] == 1 } , { x , y } , { t , 1 , 10 }
]

Resposta: { { *x -> InterpolatingFunction[{ 1., 10. }, <>] ,*
y -> InterpolatingFunction[{ 1., 10. }, <>] } }

Comande: ***ParametricPlot[***
Evaluate[{ x[t] , y[t] } /. sol] , { t , 1 , 10 } ,
Ticks -> { { 0 , .5 , 1 , 1.5 } , { .2 , .6 , 1.
} }]

Resposta:



Para obter informações sobre o **Plot** e o **ParametricPlot**, consulte o capítulo sobre gráficos.

Capítulo 10

Arquivos e transferência de dados

Quando estamos no meio de uma sessão do Mathematica e queremos interrompê-la, devemos salvar os resultados em disco. Neste capítulo, descreveremos os modos possíveis de salvar uma sessão ou parte dela em arquivos.

10.1 Salvando uma sessão

O modo mais simples para se salvar uma sessão consiste em clicar a palavra **File**, que aparece no menu, situado no alto da tela. Com isto surge uma janela com diversas opções. Aponte com o cursor do mouse a palavra **Save** e pressione o botão. A seguir, siga as instruções da tela para a gravar a sessão no local adequado. O procedimento é análogo ao de qualquer editor de texto para **Windows.**

Lembramos que um arquivo é identificado pelo seu **nome** e seu **tipo**. O tipo é opcional, podendo ou não existir. Existindo o tipo, ele é separado do nome por um ponto. Lembramos que o nome de um arquivo deve ter no mínimo um e no máximo 8 (oito) caracteres, sendo o primeiro uma letra. Os outros caracteres podem ser letras, números e o traço sublinhado. O tipo pode ter no máximo 3 caracteres, entre letras e números. Deste modo, o identificador de um arquivo é da forma **nome.tipo**.

Quando não se especifica o tipo do arquivo, o Mathematica lhe atribui o tipo **ma**. Este é um arquivo de texto que contém as entradas e saídas obtidas durante a sessão. Até os gráficos são transformados em textos, usando o **PostScript** que é uma linguagem desenvolvida pela **Adobe Systems Incorporated** que descreve as figuras através de textos. Graças a esta característica, estes arquivos podem ser manipulados por qualquer editor de texto. Juntamente com o arquivo de texto com tipo **ma**, o Mathematica gera automati-

camente um arquivo binário, com tipo **mb**, bem maior que o primeiro. Este arquivo não é indispensável para se recuperar a sessão mas a sua existência aumenta a velocidade de recuperação.

10.2 Localização dos arquivos

Um ponto fundamental na manipulação de arquivos consiste em sua localização. Quando se grava um arquivo, o usuário deverá fazê-lo em um diretório criado para esta finalidade. Não se deve, por exemplo, gravar arquivos gerados pelo usuário no mesmo diretório que contém o Mathematica ou qualquer outro programa comprado pelo usuário. No final de algum tempo, não se saberá mais o que foi comprado e o que foi gerado pelo usuário. Sem medo de exagerar, enfatizamos que **o usuário deverá criar novos diretórios com o objetivo de receber os arquivos criados por ele**.

Uma boa política consiste em criar no diretório raiz um diretório chamado **usuarios** (sem o acento agudo no **a**). Dentro dele, cada usuário cria outro diretório com seu nome. Este será o seu **diretório de trabalho**. Se o computador for usado por uma única pessoa, ele poderá criar um diretório com o seu nome diretamente no diretório raiz. Lembramos que o nome do diretório pode conter no máximo 8 (oito) caracteres, sendo o primeiro uma letra. Os outros caracteres podem ser letras, números e o traço sublinhado que fica na parte superior da tecla que contem o sinal de subtração (–). O Windows 95 permite nomes com mais de 8 caracteres. Consulte o manual para obter maiores detalhes.

O **diretório de trabalho** é aquele no qual os arquivos serão gravados e de onde serão recuperados. Para obter o diretório de trabalho corrente, use

```
Directory[ ]
```

e, para modificar o diretório de trabalho, comande

```
SetDirectory[ "novo_diretório"]
```

onde **novo_diretório** contém toda a trajetória (**path**) do novo diretório de trabalho e deve ser digitado entre aspas (”). Para obter a lista dos arquivos existentes no diretório de trabalho, use

```
FileNames[ ]
```

e, querendo apenas arquivos de um determinado nome ou tipo, use um dos comandos

```
FileNames[ ”nome.tip”]

FileNames[ ”*.tip”]

FileNames[ ”nome.* ”]
```

onde **nome** é o nome do arquivo e **tip** é o seu tipo. Observe as aspas delimitando o identificador dos arquivos. Estes comandos aceitam o * que desempenha o papel de curinga, substituindo uma seqüência qualquer de caracteres.

10.3 Recuperando uma sessão

Para recuperar uma sessão salva anteriormente, clique a palavra **File** e, ao aparecer a janela do menu, clique em **Open**. Agora, siga as instruções da tela para buscar o arquivo gravado. Também neste caso o procedimento é análogo ao de qualquer editor de texto para **Windows.**

Com este procedimento, apenas o texto foi recuperado. O valor de variáveis ou definições de funções se perdeu. Se o usuário precisar do valor de uma variável ou definição de uma função da sessão recuperada, deverá executar novamente o comando que a definiu, como explicamos em seguida.

Quando se recupera uma sessão por este processo, todos os comandos emitidos anteriormente retornam à tela. Ao conjunto destes comandos se dá o nome de **notebook** que, por ser um nome consagrado, vamos mantê-lo sem tradução. Pode-se deslocar pelo **notebook** pressionando o botão do mouse com o cursor sobre a faixa situada à direita da tela ou sobre as setas que

apontam para cima e para baixo, situadas respectivamente no canto superior direito e no canto inferior direito da parte visível do notebook. Retornando a um comando, pode-se modificá-lo e executá-lo novamente. Para tanto, basta pressionar o botão do mouse com o cursor sobre ele e efetuar as modificações como um editor de textos comum. Para executar o comando, basta pressionar **Insert** com o cursor posicionado em algum ponto do comando.

Desejando reutilizar uma função ou variável, retorne ao ponto em que foram definidas, clique o botão do mouse com o cursor sobre ela e pressione a tecla **Insert**. Com este procedimento, o comando é executado e o valor da variável ou função é recuperado.

10.4 Copiando e apagando arquivos

Para copiar o arquivo **arq_1** no arquivo **arq_2**, ambos residentes no diretório de trabalho, comande

```
CopyFile[ "arq_1 ", "arq_2 "]
```

e, para apagar o arquivo **arq** residente no diretório de trabalho, use

```
DeleteFile[ "arq "]
```

10.5 Células de inicialização

Os cálculos realizados pelo Mathematica se agrupam em **células**, definidas pelos traços verticais localizados no lado direito da tela, ao lado das **expressões**. Pode-se **selecionar uma célula**, colocando o cursor do mouse sobre a marca vertical e clicando o botão. Com este procedimento, aparece um retângulo preto na margem direita da tela, ao longo do traço vertical que delimita a célula. Isto indica que a célula está marcada.

Quando se recupera uma sessão salva anteriormente, **retorna para a tela o texto e apenas o texto da sessão anterior**. Para aproveitar variáveis e funções calculadas na sessão atual, os cálculos devem ser refeitos. Como vimos, este processo pode ser feito manualmente, expressão por expressão.

Este processo pode ser tedioso e, se o **notebook** for grande, poderemos nos esquecer de recalcular algum comando e obter resultados incoerentes. Para automatizar o processo podemos lançar mão das **células de inicialização**.

Quando se recupera uma sessão contendo células de inicialização, elas são recalculadas automaticamente.

Para **transformar** uma célula de entrada em célula de inicialização, marque-a, posicione o cursor do mouse sobre a palavra **Cell** do menu e pressione o botão. Surgindo a janela de opções, clique sobre a palavra **Initialization**. Faça isto com cada comando que deseja reaproveitar em uma sessão futura. Salve a sessão em um arquivo, como descrevemos acima. Quando a sessão for recuperada, os valores destas variáveis e funções serão recalculadas assim que o arquivo for carregado na memória do computador. Antes de recalcular, o Mathematica consulta o usuário para saber se ele realmente deseja a realização dos cálculos.

10.6 Buscando informações em arquivos

Para listar o conteúdo de um arquivo **arq**, situado no diretório de trabalho, use

```
FilePrint["arq"]
```

Querendo verificar se **arq** contém um determinado **texto**, use

```
FindList[ "arq", "texto"]
```

e, para procurar a existência do **texto** em todos os arquivos do diretório de trabalho, use

```
FindList[ FileNames[ ] , "texto"]
```

Se o **texto** não for encontrado, o Mathematica retorna a mensagem

```
Out[n] = { }
```

onde **n** é o número de ordem do comando emitido.

10.7 Salvando expressões

Pode-se gravar uma **expressão** no arquivo **nome.tip** situado no diretório de trabalho com o comando

```
expressão > > nome.tip
```

onde $> >$ são dois sinais de maior do que consecutivos. Antes de gravar, o Mathematica calcula e simplifica a **expressão** e grava o resultado no arquivo na forma de texto. Se o arquivo já existir, todo o seu conteúdo é apagado e a **expressão** é gravada. Se o arquivo ainda não existir, ele será criado e a **expressão** é gravada. Desejando acrescentar uma expressão em um arquivo pré-existente, use

```
expressão > > > nome.tip
```

com três sinais de maior do que. Os arquivos gerados por estes comandos, contêm apenas os textos das expressões e poderão ser editados pelo usuário. O comando

```
var > > nome.tip
```

guarda na variável **var** a última expressão salva no arquivo **nome.tip**.

10.8 Salvar e recuperar variáveis e funções

Querendo salvar as definições de funções ou variáveis no arquivo **arq**, situado no diretório de trabalho, comande

Save["arq", f , g , ...]

onde **f**, **g**, **...**, são os nomes das funções ou variáveis. Para restabelecer estas definições em uma sessão futura do Mathematica, comande

< < arq

Nota 10.1 *Não se esqueça de definir como diretório de trabalho, o diretório no qual se encontra* **arq**. *Para isto, use o* **SetDirectory**.

10.9 Salvando e recuperando dados

Se o arquivo **arq** é um arquivo do tipo texto, contendo textos, dados numéricos, definições de funções e variáveis, o comando

ReadList["arq"]

lê todo o conteúdo do arquivo, colocando-o em uma lista. O comando

ReadList["arq", Number]

lê apenas os números retornando-os ao Mathematica em uma lista enquanto que

ReadList["arq", String]

lê apenas os textos. O comando

ReadList["arq", Number , RecordList − > True]

carrega os números contidos em **arq** na sessão do Mathematica, colocando cada linha de **arq** em uma lista.

Exemplo 10.1 *Inicie uma nova sessão do Mathematica. Vamos supor que exista no diretório raiz um sub-diretório chamado* **usuarios** *(sem acento). Se ele não existir, crie-o.*

Comande: **Directory[ ]**

Resposta: C: \ WNMATH22

Comande: **SetDirectory["c: \ usuarios"]**

Resposta: C: \ USUARIOS

Comande: **Clear[f , m]**

Comande: **DeleteFile["teste.ma"]**

Comande: **f = x^2 + 3**

Resposta: $3 + x^2$

Comande: **m = { { 11 , 12 } , { 21 , 22 } }**

Resposta: { { 11 , 12 } , { 21 , 22 } }

Comande: **? f**

Resposta: Global 'f

f = 3 + x^2

Comande: **?m**

Resposta: Global ‘m

m = { { 11 , 12 } , { 21 , 22 } }

Comande: **Save["teste.ma", f]**

Comande: **m > > > teste.ma**

Comande: **ReadList["teste.ma"]**

Resposta: f = 3 + x^2

{ { 11 , 12 } , { 21 , 22 } }

Comande: **Clear[f, m]**

Comande: **ReadList["teste.ma"]**

Resposta: { $3 + x^2$, { { 11 , 12 } , { 21 , 22 } } }

Comande: **? f**

Resposta: Global ‘f

f = 3 + x^2

Comande: **? m**

Resposta: Global ‘m

Observe que o valor de **m** *não foi recuperado. O comando* **Save["teste.ma", f]** *salva a definição de* **f** *e a recupera. O comando* **m > > > teste.ma** *salva apenas o texto que define* **m** *mas não grava a definição de* **m**.

10.10 Gerar expressões em Fortran, C e TeX

Para obter uma **expressão** no formato aceito pela linguagem **Fortran**, comande

FortranForm[expressão]

e, para obtê-la no formato aceito pela linguagem **C**, use

CForm[expressão]

e, para obtê-la no formato aceito pelo **TeX**, processador de texto através do qual foi escrito este livro, comande

TeXForm[expressão]

Para obter as letras do alfabeto grego e as constantes i, e, π, ∞ no formato TeX, use a tabela

Forma no Mathematica	Saida em TeX
alpha, beta, gamma, ...	$\alpha,\ \beta,\ \gamma,\ ...$
ALPHA, BETA, GAMMA, ...	$A,\ B,\ \Gamma,\ ...$
I, E, Pi, Infinity	$i,\ e,\ \pi,\ \infty$

Exemplo 10.2 *Vamos desenvolver a expressão*

$$(1+x)^2\cos(x)$$

no Mathematica e traduzi-la para o Fortran, o C e o TeX.

Comande: **Expand[(1 + x)^2 Cos[x]]**

Resposta: $Cos[x] + 2xCos[x] + x^2Cos[x]$

Comande: **FortranForm[%]**

Resposta: Cos(x) + 2 * x * Cos(x) + x ** 2 * Cos(x)

Comande: **CForm[%%]**

Resposta: Cos(x) + 2 * x * Cos(x) + Power(x, 2) * Cos(x)

Comande: **TeXForm[%%%]**

Resposta: \ cos (x) + *2* \, *x* \ , \ cos (x) + { *x^2* } \ , \ cos (x)

10.11 Trocando informações entre programas

Pode-se marcar células clicando a barra vertical situada à sua direita e que a delimita ou então clicando e arrastando o mouse como se procede para marcar um trecho num editor de texto. Este texto pode ser copiado (**Copy**) ou cortado (**Cut**) usando o menu do **Edit**. O trecho copiado ou cortado fica gravado na área de descarte (**clipboard**) enquanto outro trecho não for copiado ou cortado. Podemos entrar em outro utilitário e colar o que está na área de transferência para este programa, usando o **Paste**, que está no menu do **Edit**.

Pode-se copiar trechos de um arquivo de texto para o Mathematica, fazendo o percurso inverso.

Capítulo 11

Gráficos

O Mathematica é dotado de uma gama variada de funções gráficas. Neste capítulo vamos descrever as principais funções e opções responsáveis pela execução de gráficos.

11.1 Gráficos bi-dimensionais

Para fazer o gráfico de uma função $y = f(x)$ no intervalo $[a, b]$, comande

```
Plot[ f[ x ] , { x , a , b } ]
```

e, para fazer o gráfico simultâneo de diversas funções $f_1(x)$, $f_2(x)$, ..., em uma mesma figura, use

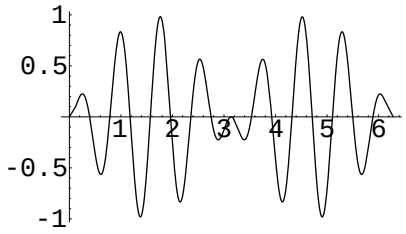
```
Plot[ { f1[ x ] , f2[ x ] , ... } , { x , a , b } ]
```

Nestes comandos, a escala vertical é escolhida automaticamente.

Exemplo 11.1 *Para obter o gráfico da função* $\text{sen}(x)\text{sen}(8x)$ *no intervalo* $[0, 2\pi]$,

Comande: ***Plot[Sin[x] Sin[8 x] , { x , 0 , 2 Pi }]***

Resposta:

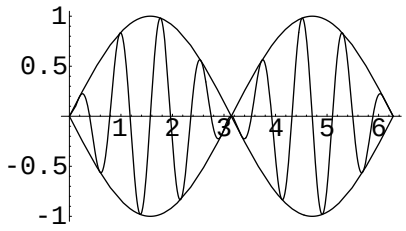


Para obter o gráfico simultâneo de $\mathrm{sen}(x)\mathrm{sen}(8x)$*, do* $\mathrm{sen}(x)$ *e de* $-\mathrm{sen}(x)$ *no intervalo* $[0, 2\pi]$,

Comande: ***Plot[{ Sin[x] Sin[8 x] , Sin[x] , – Sin[x] } ,***

{ x , 0 , 2 Pi }]

Resposta:



Exemplo 11.2 *Vamos integrar numericamente o problema de valor inicial*

$$y'' - 2y' + 5y = 0\,, \qquad y(0) = 0\,, \qquad y'(0) = -1$$

entre 0 *e* π *e fazer o gráfico da solução. Lembrando que* ***y"[x]*** *se escreve com dois acentos agudos consecutivos. Para aparecerem na tela, deve-se pressionar a barra de espaço após teclar cada acento.*

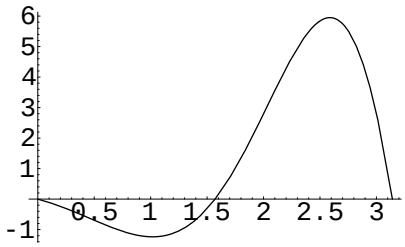
Comande: ***NDSolve[{ y"[x] – 2y'[x] + 5y[x] == 0 ,***

y[0] == 0 , y'[0] == –1 } , y , { x , 0 , Pi }]

Resposta: { { y –> *InterpolatingFunction[{ 0., 3.14159 }, < >]* } }

Comande: ***Plot[y[x] /. % , { x , 0 , Pi }]***

Resposta:



Para construir o gráfico de uma função, o Mathematica seleciona os pontos nos quais irá calcular as expressões e substitui estes valores numéricos nas expressões. Em seguida, ele calcula as expressões numéricas resultantes nestes pontos. O gráfico é obtido ligando os pontos calculados por segmentos de reta.

Na maioria dos casos este processo funciona bem. Todavia, em determinadas ocasiões, ele falha. Este processo não funciona quando precisamos da forma analítica da expressão para o seu cálculo. Isto ocorre, por exemplo, quando queremos fazer uma tabela ou derivar uma função.

Exemplo 11.3 *Vamos exemplificar o que foi dito no último parágrafo. Observe as saídas obtidas com os comandos abaixo.*

Comande: ***Plot[Table[x^n, { n, 1, 5 }] , {x, 0, 1 }]***

Comande: ***Plot[D[Cos[x^2], x] , { x, 0, 2 Sqrt[Pi] }]***

Em resposta a estes dois comandos obtivemos diversas mensagens de erro e nenhum gráfico. Isto aconteceu porque o comando ***Plot*** *substituiu os valores numéricos escolhidos para compor o gráfico nas expressões para somente depois tentar desenvolvê-las. Com isto, o Mathematica fica sem saber como desenvolver as expressões numéricas, pois, para aplicar os comandos* ***Table*** *e* ***D*** *ele precisa da forma algébrica das expressões.*

Em casos como os do exemplo anterior, precisamos acrescentar a função **Evaluate** no comando **Plot**, solicitando que as expressões sejam calculadas analiticamente, antes que os pontos sejam substituídos nas expressões. A sintaxe fica

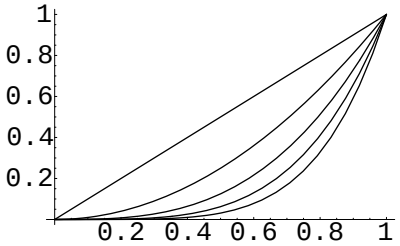
Plot[Evaluate[expr] , { x , a , b }]

Eventualmente podemos usar o **Evaluate** mesmo quando isto não for obrigatório, apenas para acelerar a construção do gráfico. Desenvolver a expressão para depois atribuir os valores é mais eficiente do que substituir os valores para desenvolver a expressão em seguida.

Exemplo 11.4 *Vamos repetir o exemplo anterior com o* ***Evaluate.***

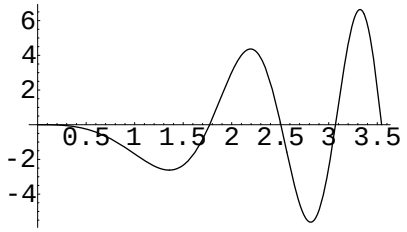
Comande: ***Plot[Evaluate[Table[x^n , {n , 1 , 5 }]] , { x , 0 , 1 }]***

Resposta:



Comande: ***Plot[Evaluate[D[Cos[x^2] , x]] , { x , 0 , 2 Sqrt[Pi] }]***

Resposta:



11.2 Opções do Plot

Ao gerar um gráfico, o Mathematica toma uma série de decisões automáticas a respeito da sua aparência final. De modo geral, as escolhas feitas nos conduzirão a resultados satisfatórios. Eventualmente, torna-se conveniente modificar o aspecto do gráfico, incluindo algumas opções.

O formato do comando **Plot** com opções é

Plot[expr , { x, xmin, xmax } , opção1, opção2, ...]

onde **opção1**, **opção2**, **...**, são da forma

opção $->$ **valor_da_opção**

As **principais opções** com seus valores automáticos são

$$\begin{aligned}
\textbf{AspectRatio} &-> \textbf{1/GoldenRatio} \\
\textbf{Axes} &-> \textbf{True} \\
\textbf{AxesLabel} &-> \textbf{None} \\
\textbf{AxesOrigin} &-> \textbf{Automatic} \\
\\
\textbf{Frame} &-> \textbf{False} \\
\textbf{FrameLabel} &-> \textbf{None} \\
\textbf{FrameTicks} &-> \textbf{Automatic} \\
\textbf{GridLines} &-> \textbf{None} \\
\\
\textbf{PlotLabel} &-> \textbf{None} \\
\textbf{PlotRange} &-> \textbf{Automatic} \\
\textbf{Ticks} &-> \textbf{Automatic}
\end{aligned}$$

Para fornecer a **razão** entre a altura e a largura do gráfico, use

AspectRatio $->$ **razão**

sendo o valor padrão da **razão** igual a **GoldenRatio**. Este número é dado pela expressão $(1+\sqrt{5})/2 \simeq 1,61803$. Com o valor da **razão** igual a **1**, a largura ficará igual à altura. Quando atribuímos o valor **Automatic** ao **AspectRatio**, o Mathematica evitará distorção de escala que produzem deformações no traçado de circunferências, nos dando a impressão de termos elipses. A opção

Axes $->$ **True**

solicita que os eixos coordenados sejam incluídos. Com o valor **False**, os eixos serão eliminados do desenho. A opção

AxesLabel − > { "título eixo horizontal", "título eixo vertical" }

define os textos que serão colocados nos eixos. Com a opção **None**, os eixos são apresentados sem textos. A opção

AxesOrigin − > { x0 , y0 }

define o ponto (x_0, y_0) no qual os eixos se cruzarão. Com o valor **Automatic**, o sistema escolherá automaticamente o ponto de interseção. Com

Frame − > True

o gráfico fica delimitado por um retângulo. A opção

FrameLabel − > { "hori_inf" , "vert_dir" ,"hori_sup" , "vert_esq" }

define os textos a serem incluídos nas quatro arestas externas do gráfico, iniciando na aresta horizontal inferior e prosseguindo em sentido anti-horário. Com o valor **None**, o quadro fica sem textos. O

FrameTicks − > Automatic

estabelece automaticamente a posição dos traços e posicionamento dos valores ao longo do retângulo que delimita o gráfico, estabelecendo as coordenadas neste retângulo. Não desejando traços nem valores ao longo das bordas, atribua o valor **None** a esta opção.
Desejando que os traços e valores sejam colocados nas posições **x1**, **x2**, **...**, **xn** na borda horizontal e nas posições **y1**, **y2**, **...**, **yk** na borda vertical, use a opção

FrameTicks − > { { x1 , x2 , ... , xn } , { y1 , y2 , ... , yk } }

Desejando que traços horizontais e verticais cortem o gráfico, use

GridLines − > Automatic

e, para que os traços sejam posicionados em determinadas coordenadas, use

GridLines − > { { x1 , x2 , ... , xn } , { y1 , y2 , ... , yk } }

como no **FrameTicks**. Com o valor **None**, que é atribuído automaticamente, as linhas da grade são omitidas. Desejando colocar um título na parte superior do gráfico, use

PlotLabel − > "Título do Gráfico"

esta opção recebe automaticamente o valor **None.**

Em geral, estabelecido o intervalo de **x** no qual se deseja o gráfico, o Mathematica escolhe adequadamente a escala vertical. Eventualmente, isto não ocorre. Nestes casos, para obter o gráfico de todos os valores calculados no domínio especificado, usa-se o

PlotRange − > All

Para especificar os limites da escala vertical, use

PlotRange − > { ymin , ymax }

e, para estabelecer tanto a escala vertical quanto a horizontal, use

PlotRange − > { { xmin , xmax } , { ymin , ymax } }

Quando esta opção não for incluída, ela recebe o valor **Automatic**. A opção

Ticks

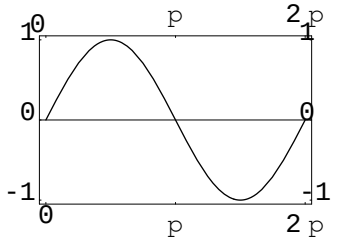
tem para os eixos a mesma função que o **FrameTicks** tem para as bordas. Os valores possíveis do **Ticks** são os mesmos do **FrameTicks**. Esta opção estabelece os tracinhos e os valores que deverão ser colocados nos eixos coordenados. Quando esta opção não for especificada, ela recebe o valor **Automatic**.

Exemplo 11.5 *Para fazer o gráfico do* $\mathrm{sen}(x)$ *no intervalo entre* 0 *e* 2π, *cujas bordas possuam traços nas posições horizontais* 0, π *e* 2π *e nas posições verticais* -1, 0 *e* 1,

Comande: ***Plot[Sin[x] , {x , 0 , 2 Pi } , Frame − > True ,***

FrameTicks − > { { 0 , Pi , 2 Pi } , { −1 , 0 , 1 } }]

Resposta:

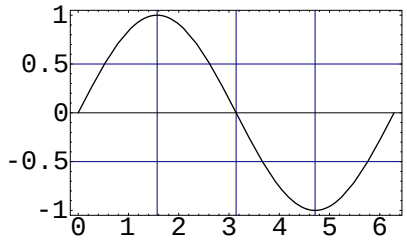


Desejando que traços verticais e horizontais sejam desenhados ao longo do gráfico, nos pontos com abcissas $\pi/2$, π, $3\pi/2$ e nos pontos de ordenadas $-1/2$ e $1/2$,

Comande: ***Plot[Sin[x] , {x , 0 , 2 Pi } , Frame -> True ,***

GridLines -> { { Pi/2 , Pi , 3Pi/2 } , { -1/2 , 1/2 } }]

Resposta:

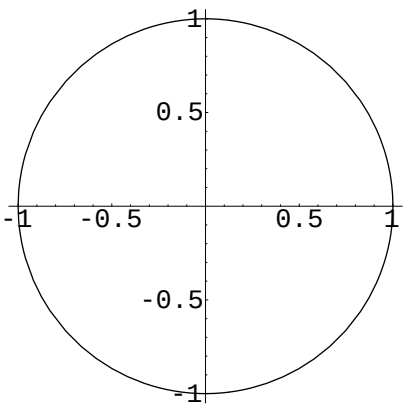


Para obter o gráfico de uma circunferência com centro na origem e raio unitário, sem distorção de escala,

Comande: ***Plot[{ Sqrt[1 - x^2] , - Sqrt[1 - x^2] } , { x , -1 , 1 } ,***

AspectRatio -> 1]

Resposta:



Dentre as várias opções, são comuns os valores

Automatic → use o algoritmo interno

None → não inclua esta opção

All → inclua todos

True → faça isto

False → não faça isto

11.3 Ampliar, diminuir e movimentar

O usuário pode **ampliar** ou **diminuir** uma figura, colocando o cursor do mouse sobre ela e clicando uma vez. Surge um quadrado limitando a figura com oito pontos, sendo um em cada vértice e um no meio de cada lado. Coloque o cursor sobre um destes pontos. O cursor toma a forma de um segmento de reta com setas nas extremidades. Mantenha pressionado o botão do mouse e o arraste. Com este movimento, a figura se ampliará ou diminuirá.

Para modificar a **posição** da figura na tela, pressione o botão esquerdo do mouse sobre a figura e, mantendo o botão pressionado, arraste o mouse, posicionando a figura no lugar desejado.

11.4 Melhorar a qualidade de um gráfico

Quando se solicita o gráfico de uma função, o Mathematica calcula o valor da função em um certo número de pontos e, através de uma análise da curvatura da função em cada intervalo, subdivide este intervalo ao meio, calculando a função neste novo ponto. A idéia consiste em obter o valor da função em um número de pontos suficiente para se capturar as características da função. Certamente este processo se encerra após um número determinado de subdivisões. Quando a função oscila rapidamente em um certo intervalo, pode-se perder algumas de suas características. Neste caso, o usuário poderá modificar os valores das opções **PlotPoints** e **PlotDivisions**. O primeiro define o número inicial de pontos no qual se calcula a função enquanto o segundo define o número máximo de vezes que um intervalo inicial pode ser dividido. Seus valores assumidos automaticamente são

PlotPoints $->$ **25**

e

PlotDivisions $->$ **20**

Quando se percebe que o resultado não foi satisfatório, pode-se aumentar estes valores, incluindo-os como uma opção dentro do comando gráfico.

Exemplo 11.6 *Para ilustrar este fato, sugerimos que o leitor emita os comandos*

Plot[Sin[30 x] , { x , 0 , 2 Pi }]

e, ampliando a figura, observe que há uma falha no gráfico desta função periódica. Emita em seguida o comando

Plot[Sin[30 x] , { x , 0 , 2 Pi } , PlotPoints $->$ ***100]***

e, ao ampliar a figura, observe que a falha foi corrigida.

11.5 Informando as opções

Tanto a função **Plot** quanto as demais funções do Mathematica possuem opções que podem ser informadas com o comando

Options[função]

Desejando o valor atual de uma dada **opção**, comande

Options[função , opção]

Para obter informações mais completas sobre uma opção, use

FullOptions[função , opção]

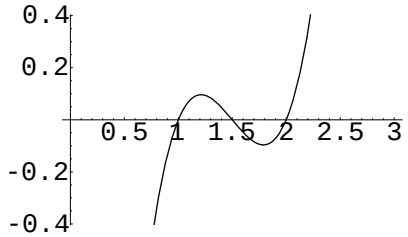
Este comando fornecerá informações sobre a opção mesmo quando seu valor for **Automatic** ou **All**. Para modificar os valores das opções de uma função, use

SetOptions[função , opção1 − > valor1 , opção2 − > valor2 , ...]

Exemplo 11.7 *Vamos exemplificar esta consulta a opções com a função* ***Plot****. Carregue o Mathematica, iniciando uma nova sessão.*

Comande: ***gr = Plot[(x − 1) (x − 2) (2x − 3) , { x , 0 , 3 }]***

Resposta:



Comande: ***Options[Plot , PlotRange]***

Resposta: { *PlotRange* − > *Automatic* }

Comande: ***FullOptions[gr , PlotRange]***

{ { *−0.075, 3.075* } , { *−0.403488, 0.403386* } }

11.6 Agrupando gráficos em uma única figura

Podemos compor diversos gráficos em uma única figura usando o **Show**, que aceita as mesmas opções do comando **Plot**. Esta função permite que gráficos feitos anteriormente sejam reaproveitados e compostos em uma figura. Assim,

Show[gráfico]

desenha o **gráfico** novamente e

Show[gráfico , opção − > valor]

reapresenta o gráfico com a nova opção. Para agrupar diversos gráficos em uma única figura, use

Show[gráfico1 , gráfico2 , ...]

Para dispor diversos gráficos em uma única figura, uma ao lado da outra, use

Show[GraphicsArray[{ graf1 , graf2 , ...}]]

para obter a combinação dispondo-os um embaixo do outro, use

Show[GraphicsArray[{ graf1 } , { graf2} , ...]]

e, para distribuí-los em linhas e colunas, numa disposição matricial, use

Show[GraphicsArray[{{graf11, graf12, ...}, {graf21, graf22, ...}}]]

Existe uma opção para controlar o espaçamento entre os gráficos. Sua sintaxe é

GraphicsSpacing -> { dist_hor , dist_vert }

onde **dist_hor** e **dist_vert** estabelecem as distâncias horizontal e vertical entre os gráficos, sendo frações da largura e altura do espaço ocupado por um dos gráficos que compõem a figura.

Exemplo 11.8 *Vamos iniciar uma nova sessão e emitir os seguintes comandos*

Comande: ***gr1 = Plot[Sin[x] , { x , 0 , 2 Pi }]***

Comande: ***gr2 = Plot[Sin[2x] , { x , 0 , 2 Pi }]***

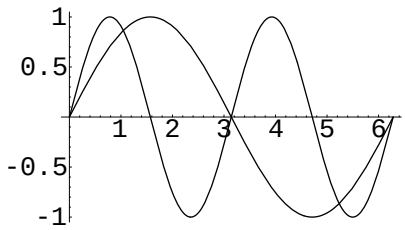
Comande: ***gr3 = Plot[Sin[3x] , { x , 0 , 2 Pi }]***

Comande: ***gr4 = Plot[Sin[4x] , { x , 0 , 2 Pi }]***

Cada gráfico recebeu, automaticamente, um número de saída. Vamos usar o ***Show*** *para agrupar estes gráficos.*

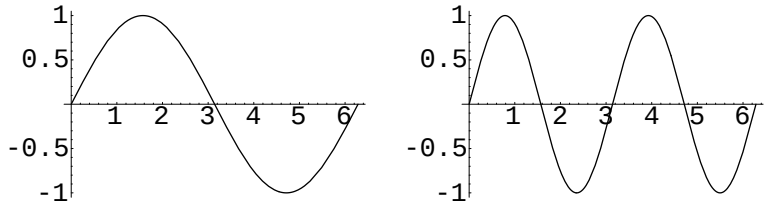
Comande: ***Show[gr1 , gr2]***

Resposta:



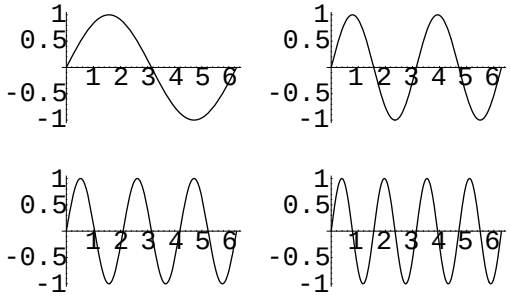
Comande: ***Show[GraphicsArray[{ gr1 , gr2 }]]***

Resposta:



Comande: ***Show[GraphicsArray[{ { gr1 , gr2 } , { gr3 , gr4 } }]]***

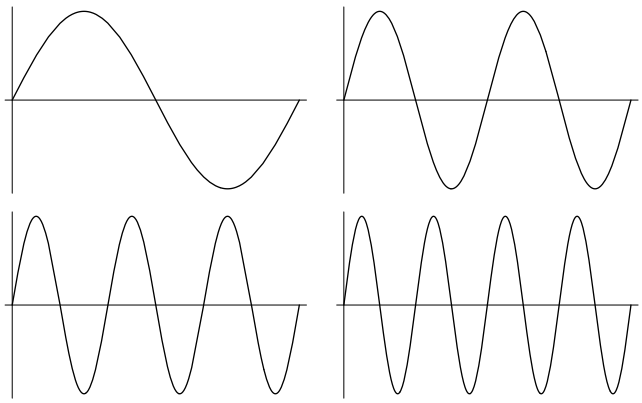
Resposta:



Vamos mostrar como se muda uma opção em todos os gráficos da figura, retirando os valores das ordenadas que aparecem nos eixos. Para tanto, utilizamos o comando de substituição local ***/.***

Comande: ***Show[% /. (Ticks − > Automatic) − > (Ticks − > None)]***

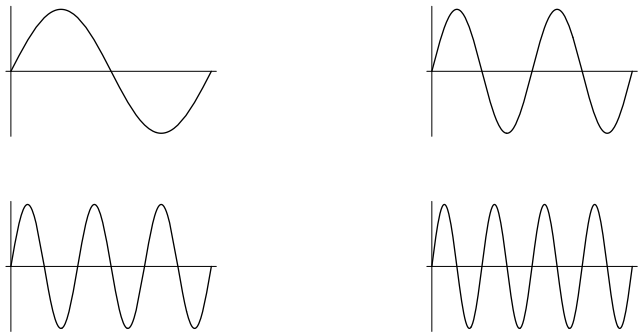
Resposta:



Em seguida, modificamos o espaçamento entre os gráficos

Comande: ***Show[% , GraphicsSpacing*** $->$ ***{ 1 , 0.5 }]***

Resposta:



11.7 Curvas de nível e relevo

Para desenhar curvas de nível de uma função $z = f(x, y)$ temos o

ContourPlot[f[x , y] , {x, xmin, xmax} , {y, ymin, ymax}]

e, para desenhar o relevo, indicando com cores diversas as diferentes alturas, temos o

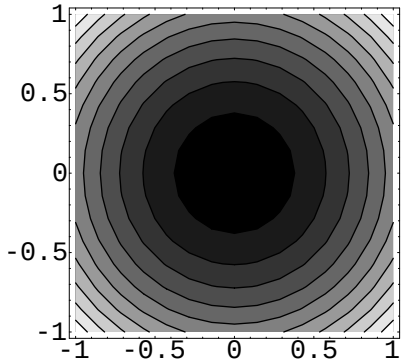
DensityPlot[f[x , y] , {x, xmin, xmax} , {y, ymin, ymax}]

Tanto uma função quanto a outra fornecem gráficos sombreados, a menos que se mude suas opções automáticas. As regiões correspondentes a maiores valores de **f** são mais claras.

Exemplo 11.9 *Vamos desenhar as curvas de nível e o relevo da função* $f(x, y) = x^2 + y^2$.

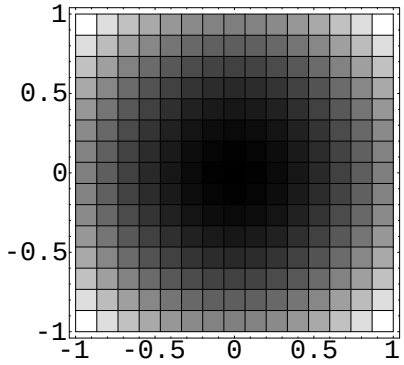
Comande: ***ContourPlot[x^2 + y^2 , { x , −1 , 1 } , { y , −1 , 1 }]***

Resposta:



Comande: ***DensityPlot[x^2 + y^2 , { x , −1 , 1 } , { y , −1 , 1 }]***

Resposta:



11.8 Opções do ContourPlot

Para a função **ContourPlot** temos as opções abaixo onde destacamos os valores automaticamente atribuídos pelo sistema.

Para definir o número de pontos no qual se calcula a função use

PlotPoints − > número_de_pontos

cujo valor padrão é **15**.

As opções abaixo também podem ser usadas com o **Show**.

Para definir as cores do sombreado temos o **ColorFunction**. Com o valor padrão

ColorFunction − > Automatic

obtemos um sombreado cinza. Com o valor **Hue**, são usados matizes diferentes para as diversas alturas.
A opção **Contours**, cujo valor padrão é

Contours − > 10

define o número de contornos que serão incluídos no gráfico.

O **PlotRange** define os valores das alturas a serem plotadas. Seu valor padrão é

PlotRange − > Automatic

que segue um padrão definido internamente. Outros valores possíveis são **All** para incluir todos os valores de $z = f(x, y)$ no retângulo (x, y) especificado. Outro valor possível é **{zmin, zmax}** para estabelecer os valores de alturas a serem incluídos na figura. Os valores fora do intervalo [**zmin, zmax**] são excluídos do gráfico.

Para estabelecer o sombreado, temos a opção

ContourShading − > True

que pode ser removido trocando seu valor para **False**.

Desejando suavizar os contornos usando uma interpolação cúbica, use

ContourSmoothing − > True

Não especificando esta opção, ela assume o valor **False** e, neste caso, se usa uma interpolação linear.

11.9 Opções do DensityPlot

Para o **DensityPlot** temos as opções abaixo. Estas opções, excetuando a primeira, podem ser usadas com o **Show**.

PlotPoints

ColorFunction

Mesh

As duas primeiras funcionam como as opções do **ContourPlot**. A opção **Mesh**, cujo valor padrão é **True**, traça uma malha de retas verticais e horizontais na figura. Com o valor

Mesh $->$ **False**

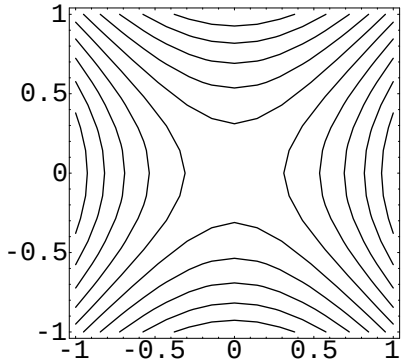
esta malha é retirada da figura.

Exemplo 11.10 *Consideremos a função* $z = x^2 - y^2$. *Vamos utilizar algumas opções com valores diferentes dos padrões.*

Comande: ***ContourPlot[x^2 - y^2 , { x , -1 , 1 } , { y , -1 , 1 } ,***

ContourShading -> False]

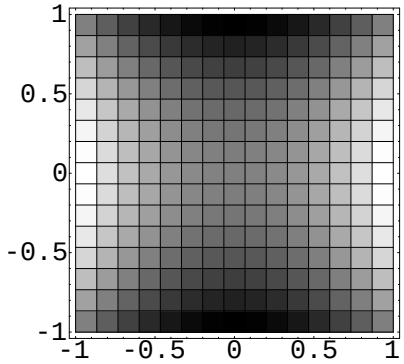
Resposta:



Comande: ***DensityPlot[x^2 - y^2 , { x , -1 , 1 } , { y , -1 , 1 } ,***

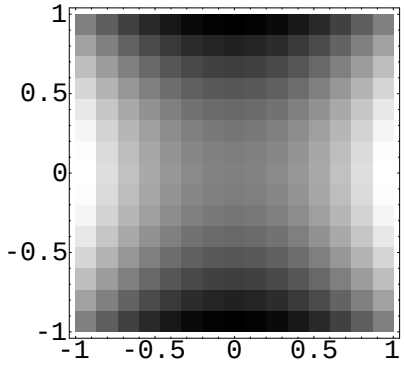
PlotPoints -> 15]

Resposta:



Comande: ***Show[%, Mesh − > False]***

Resposta:



11.10 Gráfico de superfícies

Para fazer o gráfico da função $z = f(x, y)$, no retângulo $[a, b] \times [c, d]$, use

```
Plot3D[ f[ x , y ] , { x , a , b } , { y , c , d } ]
```

11.11 Opções do Plot3D

O **Plot3D** aceita opções com a mesma sintaxe estabelecida na função **Plot**. As principais **opções** com seus valores automaticamente assumidos, são

Axes	**– >**	**True**
AxesLabel	**– >**	**None**
Boxed	**– >**	**True**
ColorFunction	**– >**	**Automatic**
FaceGrids	**– >**	**None**
HiddenSurface	**– >**	**True**
Lighting	**– >**	**True**
Mesh	**– >**	**True**
PlotRange	**– >**	**Automatic**
Shading	**– >**	**True**
ViewPoint	**– >**	**{ 1.3, –2.4, 2 }**
PlotPoints	**– >**	**15**

Todas estas opções, exceto a última, podem ser usadas na função **Show**.

Com

Axes – > False

os eixos são eliminados do gráfico. Para colocar títulos nos eixos, use

AxesLabel – > { "título_x ", "título_y ", "título_z "}

Para eliminar o paralelepípedo que delimita o gráfico, use

Boxed – > False

Para obter um quadriculado nas face, coloque a opção

FaceGrids – > All

e, para obter matizes diferentes em cada quadriculado, use

ColorFunction – > Hue

Desejando que as linhas invisíveis da superfície sejam desenhadas, use

HiddenSurface – > False

Com

Lighting – > True

a superfície é desenhada simulando uma iluminação. Desejando eliminar a malha xy que é desenhada sobre a superfície, use

Mesh − > False

Querendo estabelecer a região a ser desenhada, use

PlotRange − > { zmin , zmax }

ou

PlotRange − > { {xmin , xmax} , {ymin , ymax} , {zmin , zmax} }

Com

Shading − > False

a superfície será desenhada na cor branca e com

Shading − > True

ela aparecerá sombreada, simulando o efeito de profundidade. Querendo mudar a posição do olho do observador para outra posição **{x0, y0, z0}** , use a opção

ViewPoint − > { x0 , y0 , z0 }

O Mathematica calcula o valor da função em quinze pontos em cada direção, num total de 225 pontos. Se este número de pontos não for suficiente para fornecer uma figura de boa qualidade, mude este valor com a opção

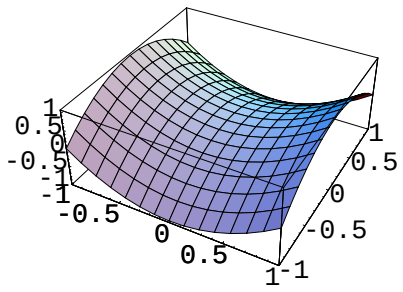
PlotPoints − > n

onde **n** é o número de pontos em que a função será calculada em cada direção.

Exemplo 11.11 *Vamos fazer o gráfico da superfície* $z = x^2 - y^2$.

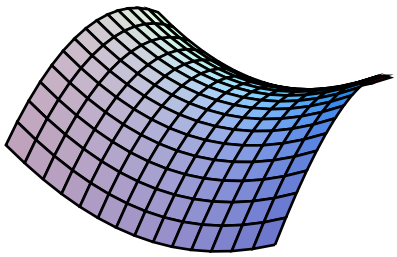
Comande: ***Plot3D[x^2 −y^2 , { x , −1 , 1} , { y , −1 , 1 }]***

Resposta:



Comande: ***Show[% , Axes -> False , Boxed -> False]***

Resposta:



Esta superfície é o gráfico da parte real da função complexa $(x + iy)^2$*. É interessante observar que o gráfico de sua parte imaginária tem esta mesma forma, rotacionada de* 90 *graus. Observe que ela apresenta um ponto de sela origem.*

A parte real ou imaginária da função $(x + iy)^3$ *apresenta um ponto de sela de macaco na origem. Vamos obter seu gráfico.*

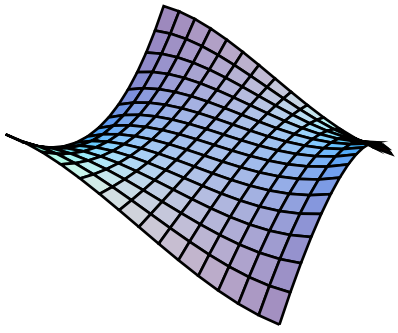
Comande: ***ComplexExpand[Re[(x + I * y)^3]]***

Resposta: $x^3 - 3\,x\,y^2$

Comande: ***Plot3D[% , {x , -1 , 1 } , { y , -1 , 1 } ,***

Boxed -> False , Axes -> False]

Resposta:



Sugerimos que o leitor faça o gráfico das partes reais e imaginárias das funções $(x + iy)^n$*, com* $n = 2, 3, \ldots$ *. Lembre que a parte imaginária de um número complexo* $(x + iy)^n$ *é fornecida pelo comando*

ComplexExpand[Im[(x + I * y)^n]]

11.12 Passando de um tipo de gráfico a outro

Para passar de um tipo a outro de gráfico, use as funções descritas a seguir. Seja **fig** o gráfico de uma superfície no espaço, ou de suas curvas de nível ou ainda o gráfico do relevo desta superfície. Utilize

Show[ContourGraphics[fig]]

para obter as curvas de nível de **fig**. Utilize

Show[DensityGraphics[fig]]

para obter o gráfico do relevo de **fig**. Utilize

Show[SurfaceGraphics[fig]]

para obter o gráfico da superfície **fig**. Utilize

Show[Graphics[fig]]

para transformar **fig** em uma imagem bidimensional. Pode-se usar o **GraphicsArray** para combinar dois ou mais gráficos em uma única figura.

Exemplo 11.12 *Vamos usar como exemplo a superfície gerada pelo gráfico da parte imaginária de* $(x+iy)^4$.

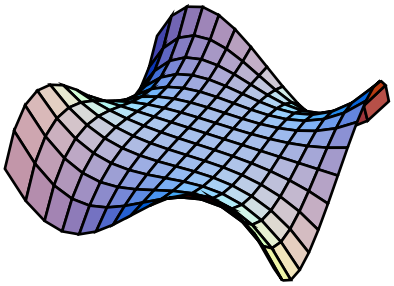
Comande: ***expr = ComplexExpand[Im[(x + I * y)^4]]***

Resposta: $4x^3y - 4xy^3$

Comande: ***fig1 = Plot3D[expr , { x , −1 , 1 } , { y , −1 , 1 } ,***

Boxed −> False , Axes −> None]

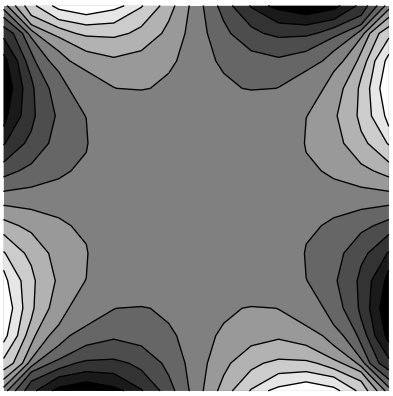
Resposta:



Comande: ***fig2 = Show[ContourGraphics[fig1] ,***

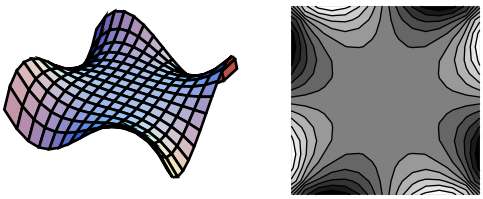
Ticks −> None , Frame −> False]

Resposta:



Comande: ***Show[GraphicsArray[{ fig1 , fig2 }]]***

Resposta:



11.13 Curvas planas parametrizadas

Uma curva no plano pode ser especificada pelas suas equações paramétricas. Em Física, as equações horárias das trajetórias de partículas são exemplos de curvas parametrizadas. No plano, as equações paramétricas de uma curva são da forma

$$x = x(t), \quad y = y(t), \quad \text{com} \quad t \in [a, b].$$

Para obter o gráfico desta curva, use

ParametricPlot[{ x[t] , y[t] } , { t , a , b }]

e, para obter o gráfico de várias curvas em uma única figura, use

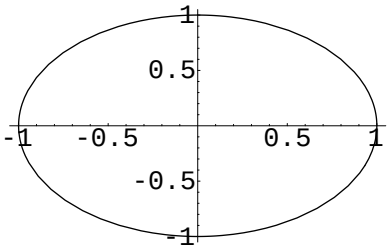
ParametricPlot[{x1[t] , y1[t]} , {x2[t] , y2[t] } , ... , { t , a , b }]

Com a opção **AspectRatio** $->$ **Automatic**, esta função preserva a forma das curvas.

Exemplo 11.13 *Vamos fazer o gráfico de uma circunferência de raio unitário, fornecendo suas equações paramétricas.*

Comande: ***ParametricPlot[{ Sin[t] , Cos[t] } , { t , 0 , 2 Pi }]***

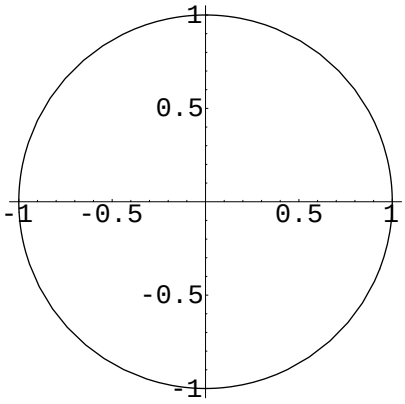
Resposta:



Comande: ***ParametricPlot[{ Sin[t] , Cos[t] } , { t , 0 , 2 Pi } ,***

AspectRatio -> Automatic]

Resposta:



11.14 Curvas e superfícies parametrizadas

No espaço, uma curva pode ser definida pelas suas equações paramétricas

$$x = x(t), \quad y = y(t), \quad z = z(t), \quad \text{com} \quad t \in [a, b]$$

que dependem de um único parâmetro t. As equações paramétricas de superfícies dependem de dois parâmetros u e v

$$x = x(u, v), \quad y = y(u, v), \quad z = z(u, v),$$

com $u \in [a, b]$ e $v \in [c, d]$. Para fazer o gráfico destes objetos geométricos, use

ParametricPlot3D[{ x[t] , y[t] , z[t] } , { t , a , b }]

ParametricPlot3D[{ x[u,v] , y[u,v] , z[u,v] }, {u, a, b}, {v, c, d}]

Desejando agrupar diversos objetos em uma figura, use

ParametricPlot3D[{ { x1[t], y1[t], z1[t] } ,
{ x2[t], y2[t], z2[t] } , ... } , { t, a, b }]

ou, para o caso de superfícies,

ParametricPlot3D[{ { x1[u,v], y1[u,v], z1[u,v] } ,

{ x2[u,v], y2[u,v], z2[u,v] } , ... } , { u, a, b } , { v, c, d }]

Particularmente, recomendo o **Show** para compor gráficos parametrizados.

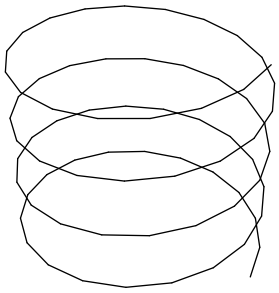
Exemplo 11.14 *Acompanhe o exemplo.*

Comande: ***fig1 = ParametricPlot3D[***

{ Cos[4 Pi * t] , Sin[4 Pi * t] , t } , { t , −1 , 1 } ,

Boxed − > False , Axes − > None]

Resposta:



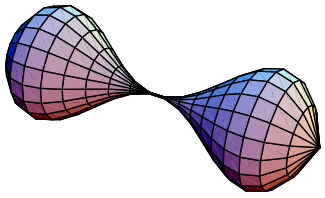
Comande: ***g[r_] = r^2 Cos[r]***

Comande: ***fig2 = ParametricPlot3D[{ r, g[r] Cos[t], g[r] Sin[t] }***
,

{ r , −Pi/2 , Pi/2 } , { t , 0 , 2 Pi } ,

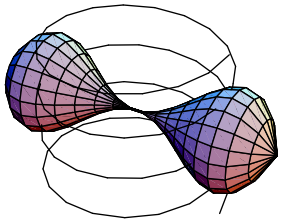
Boxed − > False , Axes − > None]

Resposta:



Comande: ***Show[fig1 , fig2]***

Resposta:



11.15 Gráfico de uma lista de dados

Para fazer o gráfico de uma lista de dados, **(x1, y1), (x2, y2), ...** use

ListPlot[{ { x1 , y1 } , { x2 , y2 } , ... }]

Se os valores de **x** forem 1, 2, ..., basta digitar

ListPlot[{ y1 , y2 , ... }]

Desejando ligar os pontos por segmentos de reta, inclua a opção

PlotJoined $->$ **True**.

Para obter o gráfico de uma lista matricial cujos valores sejam dados por

z11, z12, ...

z21, z22, ...

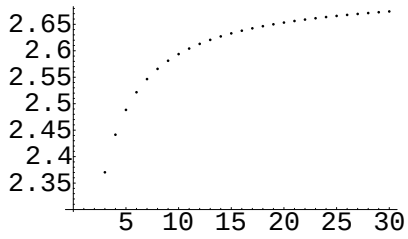
...

use o comando

ListPlot3D[{ { z11 , z12 , ... } , { z21 , z22 , ... } , ... }]

Exemplo 11.15 *Comande:* ***ListPlot[Table[(1+1/n)^n , {n, 1, 30}]]***

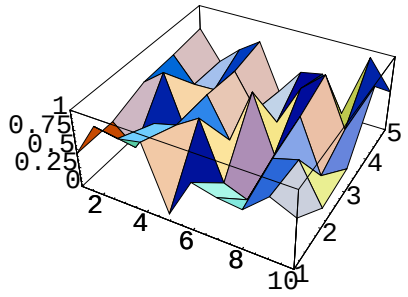
Resposta:



Comande: ***ListPlot3D[Evaluate[***

Table[Random[ ] , { i , 5 } , { j , 10 }]]]

Resposta:



Os comandos

ListContourPlot[matriz]

ListDensityPlot[matriz]

desenham, respectivamente, o gráfico das curvas de nível e o gráfico do relevo da **matriz**, que deve ser da forma

matriz = { { z11 , z12 , ... } , { z21 , z22 , ... } }

11.16 Pacotes gráficos adicionais

Existem diversos pacotes no Mathematica com a função de desenhar gráficos especiais e que devem ser carregados pelo usuário para serem utilizados. Vamos oferecer um breve receituário de alguns destes pacotes. Para maiores detalhes, consulte o livro Guide to Standard Mathematica Packages, Version 2.2 da Wolfram Research.

Um dos pacotes é o **Graphics‘Graphics‘**. Para carregá-lo, comande

```
<< Graphics‘Graphics‘
```

Em seguida, descrevemos as funções existentes neste pacote. Uma delas é

```
LogPlot[ f[ x ] , { x , xmin , xmax } ]
```

que gera um gráfico mono-log de $y = f(x)$. Para gerar o gráfico log-log desta mesma função, temos o

```
LogLogPlot[ f[x] , { x , xmin , xmax } ]
```

Dada uma curva em coordenadas polares $r = r(t)$, com $t \in [a, b]$, onde r é a distância do ponto à origem e t é o ângulo que o vetor posição forma com o eixo horizontal, use

```
PolarPlot[ r[ t ] , { t , a , b } ]
```

para obter o traço desta curva.

Para gerar o gráfico mono-log de uma **lista**, use

```
LogListPlot[ lista ]
```

e, para gerar o gráfico log-log, use

```
LogLogListPlot[ lista ]
```

Para gerar o gráfico de uma lista de dados com barras de erro, use

```
ErrorListPlot[ { { x1 , y1 , dy1 } , { x2 , y2 , dy2 } , ... } ]
```

onde **(xi, yi)** são os dados e **dyi** as margens de erro. Desejando gerar o gráfico de uma lista de dados colocando textos em lugar de pontos, use

```
TextListPlot[ { {x1 , y1 , "texto1"} , {x2 , y2 , "texto2"} , ...} ]
```

ou

```
LabeledListPlot[ {{x1 , y1 , "texto1"} , {x2 , y2 , "texto2"} , ...} ]
```

No gráfico produzido por este comando, aparece o ponto com o texto ao lado. No gráfico produzido pelo comando anterior, aparece apenas o texto na posição do ponto.

Para obter um gráfico de barras relativo a uma lista de valores do tipo

```
lista = { val1 , val2 , ... }
```

use

```
BarChart[ lista ]
```

e, para obter um gráfico em formato de pizza, use

```
PieChart[ lista ]
```

Um outro pacote, que se carrega com o comando

```
<< Graphics`ParametricPlot3D`
```

possui algumas funções para desenhar superfícies parametrizadas. Para obter o gráfico de uma superfície $z = z(r, \theta)$, onde (r, θ, z) são as coordenadas cilíndricas, com r percorrendo o intervalo $[r1, r2]$ e θ o intervalo $[\theta 1, \theta 2]$, use

```
CylindricalPlot3D[ z[ r , teta ] , { r , r1 , r2 } , { teta , t1 , t2 } ]
```

e, para obter o gráfico da superfície definida em coordenadas esféricas por $r = r(\theta, \varphi)$, com (θ, φ) no retângulo **[t1, t2]**×**[f1, f2]** use a função

```
SphericalPlot3D[ r[ teta , fi ] , { teta , t1 , t2 } , { fi , f1 , f2 } ]
```

Outro pacote, que pode ser carregado com o comando

```
<< Graphics`PlotField`
```

tem as funções gráficas abaixo. Dado um campo bidimensional de vetores **(vx, vy)**, a função

```
PlotVectorField[ { vx , vy } , { x , x1 , x2 } , { y , y1 , y2 } ]
```

nos fornece o gráfico deste campo no retângulo **[x1, x2]**×**[y1, y2]**.

Sendo $f = f(x, y)$ uma função escalar, podemos obter o campo do gradiente de f num retângulo **[x1, x2]**×**[y1, y2]** com o comando

```
PlotGradientField[ f[ x , y ] , { x , x1 , x2 } , { y , y1 , y2 } ]
```

Dada uma **lista**, podemos gerar o campo de vetores correspondente a ela com a função

```
ListPlotVectorField[ lista ]
```

e, para obter o gráfico de um campo de vetores **(vxn, vyn)** com **n** = **1**, **2**, **...**, com origens nos pontos **(xn, yn)** com **n** = **1**, **2**, **...**, respectivamente, use

```
ListPlotVectorField[ { { { x1 , y1 } , { vx1 , vy1 } ,
                         { x2 , y2 } , { vx2 , vy2 } , ... } ]
```

Exemplo 11.16 *Vamos mostrar que* $(n-1)!$ *pode ser aproximado por*

$$\sqrt{2\pi/n}(\,n/e)^n,$$

para grandes valores n. Para tanto, construiremos o gráfico mono-log do erro relativo

$$\frac{(n-1)! - \sqrt{\frac{2\pi}{n}}(\frac{n}{e})^n}{(n-1)!} = 1 - \frac{\sqrt{\frac{2\pi}{n}}\,\left(\frac{n}{e}\right)^n}{(n-1)!}$$

verificando que ele se aproxima de zero à medida que o n cresce. Inicie uma nova sessão do Mathematica e

Comande: $<<$ ***Graphics'Graphics'***

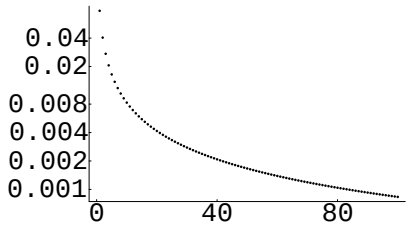
Comande: ***Table[1 - N[Sqrt[2Pi / n] (n / E)^n] / (n - 1) !***
,

{ n , 1 , 100 }] ;

Comande: ***LogListPlot[% ,***

Ticks -> { { 0, 40, 80 } , { 0.001, 0.002, 0.005, 0.008} }]

Resposta:

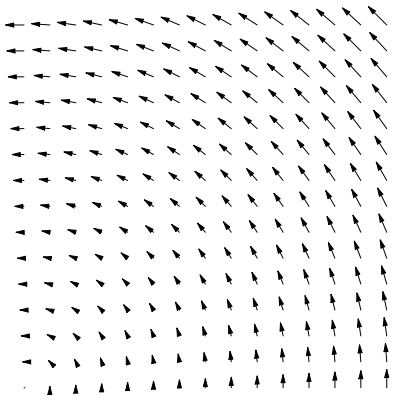


Agora vamos gerar o gráfico do campo de vetores $(y, -x)$*, no retângulo* $[-1, 1] \times [-1, 1]$.

Comande: $<<$ ***Graphics'PlotField'***

Comande: ***PlotVectorField[{ -y , x } , { x , 0 , 1 } , { y , 0 , 1 }]***

Resposta:



Capítulo 12

Análise vetorial

Estamos habituados a trabalhar com as coordenadas cartesianas dos pontos do espaço. Em certas ocasiões, para aproveitar a simetria de uma equação ou fenômeno físico, é conveniente trabalhar com coordenadas curvilíneas, assunto para o qual dedicamos este capítulo.

12.1 Coordenadas curvilíneas

Sendo (x, y, z) as **coordenadas cartesianas** de um ponto do espaço e $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$, os **versores** nas direções dos eixos coordenados x, y, z, então

$$\mathbf{r}(x, y, z) = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$$

é o **vetor posição** deste ponto em relação à origem do sistema de coordenadas. Sendo (x_1, x_2, x_3) as coordenadas deste mesmo ponto num sistema de **coordenadas curvilíneas**, podemos relacioná-las com (x, y, z) mediante as fórmulas de **mudança de coordenadas**

$$\begin{aligned} x &= x(x_1, x_2, x_3)\,, \\ y &= y(x_1, x_2, x_3)\,, \\ z &= z(x_1, x_2, x_3)\,. \end{aligned}$$

Substituindo na expressão anterior, obtemos vetor posição nestas novas coordenadas

$$\mathbf{r}(x_1, x_2, x_3) = x(x_1, x_2, x_3)\,\mathbf{i} + y(x_1, x_2, x_3)\,\mathbf{j} + z(x_1, x_2, x_3)\,\mathbf{k}\,.$$

Exemplo 12.1 *Um sistema de coordenadas curvilíneas bem clássico é o cilíndrico, onde $x_1 = \rho$, $x_2 = \theta$, $x_3 = z$. As fórmulas de mudança de coordenadas*

do sistema cilíndrico para o cartesiano são

$$\begin{aligned} x &= \rho\cos\theta \\ y &= \rho\text{sen}\,\theta \\ z &= z \end{aligned}$$

onde ρ, θ *e* z *percorrem os intervalos* $\rho \geq 0$, $-\pi \leq \theta < \pi$, $-\infty < z < \infty$. *Neste sistema,*

$$\mathbf{r}(\rho,\theta,z) = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = \rho\cos\theta\,\mathbf{i} + \rho\text{sen}\,\theta\,\mathbf{j} + z\,\mathbf{k}\,.$$

O Mathematica pode efetuar operações vetoriais em diversos sistemas de coordenadas, utilizando um **pacote especial** que deve ser carregado na memória interna do computador durante a sessão. Preferencialmente, carregue o pacote logo no início da sessão. Para carregá-lo, comande

```
<< Calculus`VectorAnalysis`
```

onde << é formado por dois sinais consecutivos de "menor do que" (<) e ` é o acento grave. Estando o Windows configurado para trabalhar com a língua portuguesa, pressione a barra de espaço depois do acento grave para fazê-lo aparecer na tela.

A menos que se especifique outro, o sistema de coordenadas usado durante uma sessão do Mathematica é o cartesiano. As letras **x**, **y** e **z** são os nomes das variáveis usadas para representar as coordenadas cartesianas. Para especificar outro sistema de coordenadas, deve-se emitir o comando

```
SetCoordinates[ NomeDoSistema ]
```

onde **NomeDoSistema** é o nome do sistema de coordenadas curvilíneas com o qual se deseja trabalhar e poderá ser um dos que aparecem na próxima tabela.

Bipolar	**EllipticCylindrical**
Bispherical	**OblateSpheroidal**
Cartesian	**ParabolicCylindrical**
ConfocalEllipsoidal	**Paraboloidal**
ConfocalParaboloidal	**ProlateSpheroidal**
Conical	**Spherical**
Cylindrical	**Toroidal**

Quando se emite o comando **SetCoordinates**, o Mathematica automaticamente atribui nomes às variáveis que representarão as coordenadas. Alguns sistemas, além das coordenadas, possuem parâmetros que poderão ser fornecidos pelo usuário. As variáveis destinadas pelo Mathematica para representar as coordenadas e os parâmetros nos diversos sistemas de coordenadas são

Bipolar[u , v , z , a]

Bispherical[u , v , phi , a]

Cartesian[x , y , z]

ConfocalEllipsoidal[lambda , mu , nu , a , b , c]

ConfocalParaboloidal[lambda , mu , nu , a , b]

Conical[lambda , mu , nu , a , b]

Cylindrical[r , theta , z]

EllipticCylindrical[u , v , z , a]

OblateSpheroidal[xi , eta , phi , a]

ParabolicCylindrical[u , v , z]

Paraboloidal[u , v , phi]

ProlateSpheroidal[xi , eta , phi , a]

Spherical[r , theta , phi]

Toroidal[u , v , phi , a]

Os três primeiros argumentos representam as coordenadas sendo os restantes os parâmetros do sistema.

Querendo atribuir outros nomes às variáveis que, de acordo com nossa preferência, são mais convenientes, podemos usar

SetCoordinates[NomeDoSistema[var1 , var2 , var3]]

Agora, **var1**, **var2** e **var3** serão os nomes das variáveis. Havendo interesse e o sistema em questão comportar, podemos neste comando fornecer os valores dos parâmetros, incluindo outros argumentos depois das coordenadas.

Nota 12.1 *Quando o pacote* **Calculus‘VectorAnalysis‘** *for usado, deve-se tomar muito* **cuidado** *com a escolha dos nomes destinados para designar. Como as variáveis atribuidas às coordenadas pelo sistema bem como aquelas escolhidas pelo usuário começam com letra minúscula, elas poderão ser confundidas com outra variável com o mesmo nome definida anteriormente durante a sessão. Quando o usuário perceber esta possibilidade, antes de carregar o pacote ou o sistema desejado, limpe as variáveis que poderão causar problema com o comando* **Clear**. *Quando for usar um pacote especial dentro do Mathematica, de preferência, carregue-o no início da sessão.*

O comando

Coordinates[ ]

informa os nomes das variáveis que estão representando as coordenadas do sistema em vigor.
O comando

CoordinateSystem

informa que sistema de coordenadas que está em uso. Para obter o domínio de variação das variáveis, use

CoordinateRanges[ ]

12.2 Mudança de coordenadas

Há uma certa discrepância entre a definição dos diversos sistemas na literatura. Pode-se obter a definição utilizada pelo Mathematica, relacionando o sistema em pauta com o cartesiano. Por exemplo, sendo **{ r, t, z }** as coordenadas cilíndricas de um ponto, suas coordenadas cartesianas serão fornecidas pelo comando

CoordinatesToCartesian[**{ r , t , z } , Cylindrical**]

Se **{ r, t, z }** forem as coordenadas cilíndricas de um ponto genérico, a resposta a este comando será uma lista

{ x(r, t, z) , y(r, t, z) , z(r, t, z) }

de modo que

$$\begin{aligned} x &= x(r,t,z) \\ y &= y(r,t,z) \\ z &= z(r,t,z) \end{aligned}$$

são as fórmulas de passagem das coordenadas cilíndricas para cartesianas.

Sendo **{ x, y, z }** as coordenadas cartesianas de um ponto genérico do espaço, obtém-se as fórmulas de passagem

$$\begin{aligned} r &= r(x,y,z) \\ t &= t(x,y,z) \\ z &= z(x,y,z) \end{aligned}$$

do sistema cartesiano para o cilíndrico mediante o comando

CoordinatesFromCartesian[{ x , y , z } , Cylindrical]

Se **{x, y, z}** forem as coordenadas de um ponto específico, este comando nos fornece as coordenadas cilíndricas deste ponto.

Nota 12.2 *Estes comandos se aplicam a todos os sistemas de coordenadas. O nome Cylindrical pode ser substituído pelo nome de qualquer outro sistema reconhecido pelo Mathematica. Quando o nome do sistema é omitido no comando, subentende-se que se deseja as transformações em relação ao sistema em uso naquele momento.*

Exemplo 12.2 *Vamos trabalhar com o sistema bipolar.*

Comande: **SetCoordinates[Bipolar]**

Resposta: Bipolar[u , v , z , 1]

Estamos sendo informados que as coordenadas serão designadas por **u**, **v** *e* **z**, *e que o parâmetro livre* **a** *recebeu o valor* 1.

Comande: **CoordinatesToCartesian[{ u , v , z } , Bipolar]**

Resposta:

$$\{ \frac{\text{Sinh}\,[v]}{-\text{Cos}\,[u]+\text{Cosh}\,[v]} \ , \ \frac{\text{Sin}\,[u]}{-\text{Cos}\,[u]+\text{Cosh}\,[v]} \ , \ z \}$$

Para obter uma lista dos parâmetros do sistema de coordenadas em uso, comande

Parameters[ ]

e, para obter que valores podemos atribuir a eles, comande

ParameterRanges[ ]

Exemplo 12.3 *Continuando o exemplo anterior, estando no sistema bipolar,*

Comande: **{ Parameters**[ ] **, ParameterRanges**[ ] **}**

Resposta: { { 1 } , 0 < #1 < Infinity }

nos informa que o sistema bipolar tem apenas um parâmetro e que o primeiro parâmetro pode assumir qualquer valor entre 0 e infinito.

Para acionar o sistema bipolar atribuindo-se um valor genérico **a** *para o parâmetro,*

Comande: **SetCoordinates**[**Bipolar**[**u , v , z , a**]]

Resposta: Bipolar[u , v , z , a]

12.3 Curvas e superfícies coordenadas

Denotemos por (x_1, x_2, x_3) as coordenadas curvilíneas de um ponto do espaço. O conjunto de pontos obtidos quando se varia uma das coordenadas mantendo fixas as outras duas é uma curva denominada **curva coordenada**. Fixando uma coordenada e variando as outras duas, os pontos assim percorridos descrevem uma superfície, denominada **superfície coordenada**.

Exemplo 12.4 *Designemos por x, y, z as coordenadas no sistema cartesiano. No sistema cilíndrico (ρ, θ, z), as curvas coordenadas são semi-retas iniciando no eixo z e paralelas ao plano (x, y) (quando variamos ρ), circunferências*

com centro no eixo z e paralelas ao plano (x, y) (quando variamos θ) e retas paralelas ao eixo z (quando variamos z). As superfícies coordenadas do sistema cilíndrico são cilindros cujos eixos coincidem com o eixo z (quando fixamos ρ), semiplanos com origem no eixo z (quando fixamos θ) e planos paralelos ao plano (x, y) (quando fixamos z).

12.4 Fatores de escala e vetores tangentes

Sendo $\mathbf{r} = \mathbf{r}(x_1, x_2, x_3)$ o vetor posição num sistema curvilíneo, onde x_1, x_2 e x_3 são as variáveis que representam as coordenadas, definimos os **fatores de escala** deste sistema por

$$h_1 = \left|\frac{\partial \mathbf{r}}{\partial x_1}\right|, \qquad h_2 = \left|\frac{\partial \mathbf{r}}{\partial x_2}\right|, \qquad h_3 = \left|\frac{\partial \mathbf{r}}{\partial x_3}\right|,$$

de modo que

$$\mathbf{e}_1 = \frac{1}{h_1}\frac{\partial \mathbf{r}}{\partial x_1}, \qquad \mathbf{e}_2 = \frac{1}{h_2}\frac{\partial \mathbf{r}}{\partial x_2}, \qquad \mathbf{e}_3 = \frac{1}{h_3}\frac{\partial \mathbf{r}}{\partial x_3},$$

são vetores de módulos unitários, tangentes às curvas coordenadas. Estes versores apontam no sentido de crescimento da coordenada correspondente. Um sistema de coordenadas é **ortogonal** quando os vetores $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ forem mutuamente ortogonais. No sistema cilíndrico,

$$h_1 = 1\,, \qquad h_2 = \rho\,, \qquad h_3 = 1$$

e

$$\begin{aligned}
\mathbf{e}_1 &= \frac{1}{h_1}\frac{\partial \mathbf{r}}{\partial x_1} = \frac{\partial \mathbf{r}}{\partial \rho} = \cos\theta\,\mathbf{i} + \operatorname{sen}\theta\,\mathbf{j} \\
\mathbf{e}_2 &= \frac{1}{h_2}\frac{\partial \mathbf{r}}{\partial x_2} = \frac{1}{\rho}\frac{\partial \mathbf{r}}{\partial \theta} = -\operatorname{sen}\theta\,\mathbf{i} + \cos\theta\,\mathbf{j} \\
\mathbf{e}_3 &= \frac{1}{h_3}\frac{\partial \mathbf{r}}{\partial x_3} = \frac{\partial \mathbf{r}}{\partial z} = \mathbf{k}
\end{aligned}$$

Este sistema é ortogonal porque os produtos escalares $\mathbf{e}_1 \cdot \mathbf{e}_2$, $\mathbf{e}_2 \cdot \mathbf{e}_3$ e $\mathbf{e}_3 \cdot \mathbf{e}_1$, são todos nulos, indicando sua ortogonalidade.

Pode-se calcular os fatores de escala do sistema em uso com o

ScaleFactors[Nome [r , s , t]]

A resposta a este comando será uma lista

{ h1, h2, h3 }

contendo os fatores de escala h_1, h_2, h_3 no sistema **Nome**, usando **r**, **s** e **t** como coordenadas. Se os nomes **r**, **s** e **t** não forem fornecidos, o Mathematica usa as variáveis escolhidas pelo próprio sistema, de acordo com a tabela. Se o **Nome** do sistema for omitido, receberemos os fatores de escala do sistema em uso no momento.

Exemplo 12.5 *Siga*

12.5 Operadores diferenciais vetoriais

Vamos verificar como se usa o Mathematica para calcular o **gradiente**, o **divergente**, o **rotacional**, o **laplaciano** e o **biharmônico** em coordenadas curvilíneas.

Seja **f** uma função que a cada ponto **p** do espaço associa um número real. A título de exemplo, citamos que a temperatura e a pressão são funções deste tipo. Podemos representar tal função usando as coordenadas curvilíneas do ponto e escrever

$$f = f(x_1, x_2, x_3)\,.$$

Num sistema de coordenadas curvilíneas ortogonais, o **gradiente** de f e o seu **laplaciano** são, respectivamente, dados por

$$\begin{aligned}
\nabla f &= \frac{1}{h_1}\frac{\partial f}{\partial x_1}\mathbf{e}_1 + \frac{1}{h_2}\frac{\partial f}{\partial x_2}\mathbf{e}_2 + \frac{1}{h_3}\frac{\partial f}{\partial x_3}\mathbf{e}_3 \\
\nabla^2 f &= \frac{1}{h_1 h_2 h_3}\left[\frac{\partial}{\partial x_1}\left(\frac{h_2 h_3}{h_1}\frac{\partial f}{\partial x_1}\right) + \frac{\partial}{\partial x_2}\left(\frac{h_1 h_3}{h_2}\frac{\partial f}{\partial x_2}\right) + \frac{\partial}{\partial x_3}\left(\frac{h_1 h_2}{h_3}\frac{\partial f}{\partial x_3}\right)\right]
\end{aligned}$$

O **biharmônico** de uma função é o laplaciano do laplaciano desta função.

Sejam **r, s** e **t** as variáveis usadas para designar as coordenadas num sistema curvilíneo. Para calcular o gradiente, o laplaciano e o biharmônico de uma função real **f[u, v, w]**, comande

```
Grad[ f [ r , s , t ] ]

Laplacian[ f [ r , s , t ] ]

Biharmonic[ f [ r , s , t ] ]
```

respectivamente.

Seja **v** uma função que a cada ponto **p** do espaço associa um vetor. Citamos como exemplos, o vetor posição de **p**, a velocidade, a aceleração e a força. Podemos representar uma tal função no sistema curvilíneo, escrevendo

$$\mathbf{v}(x_1, x_2, x_3) = v_1(x_1, x_2, x_3)\mathbf{e}_1 + v_2(x_1, x_2, x_3)\mathbf{e}_2 + v_3(x_1, x_2, x_3)\mathbf{e}_3 \, .$$

Estando no sistema desejado, podemos definir a função vetorial $\mathbf{v}(x_1, x_2, x_3)$ no Mathematica usando a lista

{ v1[x1 , x2 , x3] , v2[x1 , x2 , x3] , v3[x1 , x2 , x3] }

O **divergente** deste campo é dado por

$$\nabla \cdot \mathbf{v} = \frac{1}{h_1 h_2 h_3}\left[\frac{\partial}{\partial x_1}(h_2 h_3 v_1) + \frac{\partial}{\partial x_2}(h_1 h_3 v_2) + \frac{\partial}{\partial x_3}(h_1 h_2 v_3)\right]$$

e o seu **rotacional** é

$$\begin{aligned}
\nabla \times \mathbf{v} \;\; = \;\; & \frac{1}{h_2 h_3}\left[\frac{\partial}{\partial x_2}(h_3 v_3) - \frac{\partial}{\partial x_3}(h_2 v_2)\right]\mathbf{e}_1 \\
+ \quad & \frac{1}{h_3 h_1}\left[\frac{\partial}{\partial x_3}(h_1 v_1) - \frac{\partial}{\partial x_1}(h_3 v_3)\right]\mathbf{e}_2 \\
+ \quad & \frac{1}{h_1 h_2}\left[\frac{\partial}{\partial x_1}(h_2 v_2) - \frac{\partial}{\partial x_2}(h_1 v_1)\right]\mathbf{e}_3
\end{aligned}$$

Para calcular o divergente e o rotacional de uma lista

v = { v1[x1, x2, x3] , v2[x1, x2, x3] , v3[x1, x2, x3] }

comande, respectivamente,

Div [v]

Curl [v]

Exemplo 12.6 *Vamos trabalhar com o sistema cilíndrico e calcular o gradiente, o laplaciano e o biharmônico de*

$$f(\rho, \theta, z) = \rho\hat{}2\cos\theta$$

o divergente e o rotacional de

$$\mathbf{v}(\rho, \theta, z) = \rho(1 - z)\mathbf{e}_1 + z\mathbf{e}_3$$

onde (ρ, θ, z) *representam as coordenadas no sistema cilíndrico. Vamos utilizar as variáveis* **r**, **t** *e* **z** *para representar* ρ, θ *e* z, *respectivamente.*

Inicie uma nova sessão e, para carregar o pacote que faz os cálculos em coordenadas curvilíneas,

Comande: $<<$ **Calculus‘VectorAnalysis‘**

Para solicitar que o sistema utilize coordenadas cilíndricas e use as variáveis **r**, **t**, **z**, *para representar as coordenadas,*

Comande: **SetCoordinates[Cylindrical[r , t , z]]**

Resposta: Cylindrical[r , t , z]

Pode-se perguntar o sistema de coordenadas que está sendo usado e as variáveis utilizadas. Para tanto,

Comande: **{ CoordinateSystem, Coordinates[ ] }**

Resposta: {*Cylindrical,* { *r, t, z* } }

Para calcular os fatores de escala h_1, h_2, h_3, *do sistema cilíndrico,*

Comande: **ScaleFactors[ ]**

Resposta: { 1, *r,* 1 }

Para definir f ,

Comande: **f = r^2 Cos[t]**

Resposta: r^2 Cos[t]

Vamos calcular o gradiente de **f**.

Comande: **Grad[f]**

Resposta: { 2 r Cos[t], –(r Sin[t]), 0 }

Logo, $\nabla \mathbf{f} = 2\,\rho\cos(\theta)\,\mathbf{e}_1 - (r\,\mathrm{Sin}[t])\,\mathbf{e}_2$

Vamos calcular o laplaciano de **f**.

Comande: **Laplacian**[**f**]

Resposta: 3 Cos[t]

Vamos calcular o laplaciano do laplaciano de **f**.

Comande: **Biharmonic**[**f**]

Resposta: $\dfrac{-3\,\mathrm{Cos}[\,t\,]}{r^2}$

Para definir **v**,

Comande: **v = { r (1 – z) , 0 , z }**

Resposta: { r (1 – z) , 0 , z }

Vamos calcular o divergente de **v**.

Comande: **Div**[**v**]

Resposta: $\dfrac{r + 2r(1-z)}{r}$

Comande: **Simplify**[%]

Resposta: $3 - 2\,z$

Vamos calcular o rotacional de **v.**

Comande: **Curl**[**v**]

Resposta: { 0 , $-r$, 0 }

Logo, $\nabla \times \mathbf{v} = -\,r\,\mathbf{e}_2$

12.6 Produto escalar, vetorial e misto

Estando num sistema de coordenadas curvilíneas, e dados três pontos **pt1**, **pt2** e **pt3** no espaço, com coordenadas curvilíneas

$$\mathbf{c1 = \{ x1 , y1 , z1 \}} \ , \quad \mathbf{c2 = \{ x2 , y2 , z2 \}} \quad \text{e} \quad \mathbf{c3 = \{ x3 , y3 , z3 \}}$$

podemos calcular o **produto escalar**, o **produto vetorial** e o **produto misto** dos vetores posições **vet1**, **vet2** e **vet3** dos pontos **pt1**, **pt2** e **pt3** em relação à origem. Para calcular o produto escalar **vet1** $\cdot$ **vet2**, use

```
DotProduct[ c1 , c2 ]
```

Observe com atenção que o que se coloca como argumento do **DotProduct** são as coordenadas curvilíneas do ponto e não seu vetor posição. Desejando calcular o produto escalar em um sistema de coordenadas denominado **NomeDoSistema**, que seja diferente do sistema em vigor, devemos informar este fato, usando o comando

```
DotProduct[ c1 , c2 , NomeDoSistema ]
```

e, desejando usar variáveis diferentes daquelas automaticamente atribuídas pelo sistema, deve-se incluir no comando as variáveis

```
DotProduct[ c1, c2, NomeDoSistema[ var1 , var2 , var3 ] ]
```

Para calcular o produto vetorial **vet1** $\times$ **vet2** comande

```
CrossProduct[ c1 , c2 ]
```

especificando o sistema de coordenadas e as variáveis, quando estas forem diferentes do sistema em vigor, como fizemos no **DotProduct**. O resultado obtido é uma lista da forma

{ cord1, cord2, cord3 }

que fornece as coordenadas curvilíneas **cord1**, **cord2** e **cord3** do ponto cujo vetor posição é **vet1**×**vet2**.

Para calcular o produto misto **vet1** · **vet2** × **vet3**, use

ScalarTripleProduct[**c1 , c2 , c3**]

Nestes dois últimos comandos, como no **DotProduct** pode-se especificar o sistema de coordenadas e as variáveis a serem utilizadas quando estas forem distintas das variáveis automáticas ou daquelas que estão sendo usadas no sistema atuante.

Exemplo 12.7 *Vamos calcular o produto vetorial dos vetores posições dos pontos* **pt1** *e* **pt2***, cujas coordenadas cilíndricas são* $(3, \pi/2, 0)$ *e* $(1, 0, 1)$. *Inicie uma nova sessão, e*

Comande: **<< Calculus`VectorAnalysis`**

Comande: **SetCoordinates[Cylindrical[r , t , z]]**

Resposta: Cylindrical[r , t , z]

Comande: **c1 = { 3 , Pi / 2 , 0 } ; c2 = { 1 , 0 , 1 } ;**

Comande: **CrossProduct[c1 , c2]**

Resposta: { 3, 0 , –3 }

12.7 Integrais de linha

Sejam x_1, x_2, x_3 as coordenadas em um sistema curvilíneo ortogonal e

$$\begin{aligned} x_1 &= x_1(t) \\ x_2 &= x_2(t) \qquad t \text{ em } [a, b] \\ x_3 &= x_3(t) \end{aligned}$$

as equações paramétricas de uma curva λ neste sistema. Seja

$$\mathbf{r}(t) = x(t)\mathbf{i} + y(t)\mathbf{j} + z(t)\mathbf{k}$$

o vetor posição dos pontos desta curva, isto é,

$$\begin{aligned} x(t) &= x\left[x_1(t), x_2(t), x_3(t)\right] \\ y(t) &= y\left[x_1(t), x_2(t), x_3(t)\right] \\ z(t) &= z\left[x_1(t), x_2(t), x_3(t)\right] \end{aligned}$$

Lembramos que (x, y, z) representam as coordenadas cartesianas do ponto. Seja $f(x_1, x_2, x_3)$ uma função real definida em termos das coordenadas curvilíneas dos pontos do espaço. A integral de linha de f ao longo da curva λ, em relação ao comprimento de arco s, é definida por

$$\int_\lambda f(x_1, x_2, x_3)ds = \int_a^b f\left[x_1(t), x_2(t), x_3(t)\right] |\mathbf{r}'(t)|\, dt\,.$$

Para calcular

$$|\mathbf{r}'(t)| = \sqrt{\mathbf{r}'(t)\cdot\mathbf{r}'(t)} = |x'(t)\mathbf{i} + y'(t)\mathbf{j} + z'(t)\mathbf{k}|\,,$$

chamado de **fator de escala** do comprimento de arco, comande

ArcLengthFactor[{ x1(t) , x2(t) , x3(t) } , t]

Para calcular a integral de linha, basta usar o comando

Integrate[f[x1[t] , x2[t] , x3[t]] * fe, { t, a, b }]

onde **fe** é o fator de escala, calculado com o comando

fe = ArcLengthFactor[{ x1(t) , x2(t) , x3(t) } , t]

Caso queiramos integrar numericamente, podemos usar o

NIntegrate

que tem a mesma sintaxe do **Integrate**.

Para calcular o fator de escala do comprimento de arco em um sistema diferente do sistema em vigor, especifique o **NomeDoSistema**

ArcLengthFactor[{ x1(t) , x2(t) , x3(t) } , t , NomeDoSistema]

Exemplo 12.8 *Vamos calcular*

$$\int_\lambda r^2 \cos^2\theta \, ds$$

onde λ *é uma hélice, parametrizada em coordenadas cilíndrica* (r, θ, z) *por*

$$r = 4\,, \qquad \theta = 2\pi t\,, \qquad z = t\,,$$

com t *percorrendo o intervalo* $[0, 1]$.

Inicie uma nova sessão e carregue o pacote que realiza operações vetoriais.

Comande: $<<$ **Calculus‘VectorAnalysis‘**.

Defina o sistema cilíndrico como sendo o sistema em uso. Use as variáveis **r**, **theta** *e* **z** *para designar as coordenadas cilíndricas* ρ, θ *e* z.

Comande: **SetCoordinates[Cylindrical]**

Resposta: Cylindrical[r , theta , z]

Para definir as equações paramétricas da curva, usamos as variáveis auxiliares **rt**, **thetat** *e* **zt**. *Para defini-las, comande*

Comande: **rt = 4 ; thetat = 2 Pi * t ; zt = t ;**

Para calcular o fator de escala do comprimento de arco,

Comande: **fator = ArcLengthFactor[{ rt, thetat , zt } , t]**

Resposta: Sqrt[$1 + 64\,\mathrm{Pi}^2$]

Para obter a integral,

Comande: **Integrate[rt^2 Cos[thetat]^2 * fator , { t , 0 , 1 }]**

Resposta: 8 Sqrt[1 + 64 Pi^2]

Para obter o resultado numérico, pode-se usar o **NIntegrate** *ou então,*

Comande: **N**[%]

Resposta: 201.221

12.8 Integrais de superfície

Se (x_1, x_2, x_3) representarem as coordenadas curvilíneas de um ponto do espaço, então as equações paramétricas de uma superfície S neste sistema é da forma

$$\begin{aligned} x_1 &= x_1(u,v)\,, \\ x_2 &= x_2(u,v)\,, \qquad u \text{ em } [a,b]\,, \quad v \text{ em } [c,d] \\ x_3 &= x_3(u,v)\,, \end{aligned}$$

A integral de $f(x_1, x_2, x_3)$ nesta superfície é definida por

$$\iint_S f(x_1,x_2,x_3)dA = \int_a^b \int_c^d f\left[x_1(u,v), x_2(u,v), x_3(u,v)\right] \left|\frac{\partial \mathbf{r}}{\partial u} \times \frac{\partial \mathbf{r}}{\partial v}\right| dv\, du\,.$$

onde $\mathbf{r}(u,v)$ é o vetor posição dos pontos da superfície em relação à origem. Podemos definir as funções $x_1(u,v)$, $x_2(u,v)$, $x_3(u,v)$ com um comando do tipo

x1 = x1[u , v] ; **x2 = x2[u , v] ;** **x3 = x3[u , v] ;**

Estando no sistema cartesiano, para obter **r**, comandamos

```
posi = CoordinatesToCartesian[ { x1, x2, x3 },
                               Sistema[ x1, x2, x3 ] ]
```

onde **Sistema** é o nome do sistema de coordenadas curvilíneas. Em seguida, calcula-se o produto vetorial $\partial\mathbf{r}/\partial u \times \partial\mathbf{r}/\partial v$ com

```
provet = CrossProduct[ D[ posi , u ] , D[ posi , v ] ]
```

para, finalmente, calcular o seu módulo $|\partial\mathbf{r}/\partial u \times \partial\mathbf{r}/\partial v|$, mediante o comando

```
fator = Simplify[ Sqrt[ provet . provet ] ]
```

onde o ponto em **provet . provet** é indispensável por se tratar de um produto escalar. A integral é calculada com

Integrate[f[x1 , x2 , x3] * fator , { u , a , b } , { v , c , d }]

Exemplo 12.9 *Supondo que (r, θ, z) são as variáveis que representam as coordenadas cilíndricas de um ponto, vamos calcular a integral*

$$\int_S 2r \cos\theta \, dA$$

onde S é o cilindro $r = 5$, $\theta = u$, $z = v$, onde u percorre o intervalo $[-\pi/2, \pi/2]$ e v percorre o intervalo $[0, 2]$.

Inicie uma nova sessão e

Comande: $<<$ **Calculus‘VectorAnalysis‘**

Sem sair do sistema cartesiano,

Comande: **posi = CoordinatesToCartesian[{ r , theta , z } ,**
Cylindrical[r , theta , z]]

Resposta: { r Cos[theta] , r Sin[theta] , z }

Comande: **r = 5 ; theta = u ; z = v ;**

Comande: **provet = CrossProduct[D[posi , u] , D[posi , v]]**

Resposta: { 5 Cos[u] , 5 Sin[u] , 0 }

Comande: **fator = Simplify[Sqrt[provet . provet]]**

Resposta: 5

Comande: **Integrate[2 r * Cos[theta] * fator ,**
{ u , – Pi/2 , Pi/2 } , { v , 0 , 2 }]

Resposta: 200

Se o **Integrate** não for capaz de obter o valor exato da integral, pode-se usar o

NIntegrate

ou então, após o **Integrate**, emitir o comando **N[%]**.

12.9 Mudança de variáveis em integrais triplas

Seja (x, y, z) as coordenadas cartesianas e $f(x, y, z)$ uma função real. Quando houver alguma simetria que possa ser explorada no cálculo da integral tripla

$$\iiint_R f(x, y, z)\, dx\, dy\, dz$$

onde R é uma determinada região do espaço, faz-se conveniente efetuar uma mudança de coordenadas

$$\begin{aligned} x &= x(t, u, v) \\ y &= y(t, u, v) \\ z &= z(t, u, v) \end{aligned}$$

para um outro sistema de coordenadas que possa se beneficiar com esta simetria. O teorema da mudança de variáveis para integrais triplas nos diz que

$$\iiint_R f(x, y, z)\, dx\, dy\, dz = \iiint_D g(u, v, z) \left| \frac{\partial(x, y, z)}{\partial(t, u, v)} \right| dt\, du\, dv$$

onde $g(u, v, z) = f(x(u, v, z), y(u, v, z), z(u, v, z))$, D é a imagem de R mediante a mudança de coordenadas e

$$\frac{\partial(x, y, z)}{\partial(t, u, v)} = \det \begin{pmatrix} x_t & x_u & x_v \\ y_t & y_u & y_v \\ z_t & z_u & z_v \end{pmatrix}$$

é o jacobiano (determinante da matriz jacobiana) da transformação. Os traços verticais indicam o valor absoluto do jacobiano. Para se calcular a matriz jacobiana de uma transformação use

JacobianMatrix[ ]

JacobianMatrix[**NomeDoSistema**]

JacobianMatrix[**ponto , NomeDoSistema**]

JacobianMatrix[**NomeDoSistema**[**variáveis**]]

Com o primeiro comando, obtemos a matriz jacobiana da mudança de coordenadas cartesianas para o sistema de coordenadas em vigor, usando as variáveis reservadas pelo sistema. Com o segundo comando, obtemos a matriz jacobiana da mudança de coordenadas do sistema cartesiano para o sistema especificado, usando as variáveis do sistema. Com o terceiro comando, obtemos a matriz jacobiana no ponto e no sistema especificados. Com o quarto comando, obtemos a matriz jacobiana no sistema especificado com as variáveis fornecidas.

Para calcular o jacobiano, use

JacobianDeterminant[ ]

JacobianDeterminant[**NomeDoSistema**]

JacobianDeterminant[**ponto , NomeDoSistema**]

JacobianDeterminant[**NomeDoSistema**[**variáveis**]]

onde as interpretações são as mesmas dadas para o **JacobianMatrix**. Lembramos que, para calcular o valor absoluto de um número **dj**, usa-se o

Abs[**dj**]

Exemplo 12.10 *Vamos calcular*

$$\iiint_R dV \qquad \text{e} \qquad \iiint_R r^2 \text{sen}/;^2\theta \, dV \, ,$$

onde R é a esfera com centro na origem e raio unitário. Lembramos que esta esfera, em coordenadas esféricas é definida como sendo o conjunto de pontos (r, θ, ϕ) com r percorrendo o intervalo $[0, 1]$, θ o intervalo $[0, 2\pi]$ e ϕ o intervalo $[0, \pi]$. Neste exemplo, D é o paralelepípedo $[0, 1] \times [0, 2\pi] \times [0, \pi]$. Inicie uma nova sessão do Mathematica e

Comande: $<<$ **Calculus‘VectorAnalysis‘**

Comande: **SetCoordinates**[**Spherical**]

Resposta: Spherical[r, theta, phi]

Para calcular o determinante jacobiano,

Comande: **determinante = JacobianDeterminant[ ]**

Resposta: r^2 Sin[theta]

Para calcular a primeira integral,

Comande: **NIntegrate[Abs[determinante] ,**

{ r , 0 , 1 } , { theta , 0 , 2 Pi } , { phi , 0 , Pi }]

Resposta: 4.18879

Este é o volume aproximado da esfera de raio unitário. De fato,

Comande: **N[4 Pi / 3]**

Resposta: 4.18879

Para calcular a segunda integral,

Comande: **NIntegrate[r^2 Sin[theta]^2 Abs[determinante] ,**

{ r , 0 , 1 } , { theta , 0 , 2 Pi } , { phi , 0 , Pi }]

Resposta: 1.67552

O valor exato desta integral é $8\pi/15$. *Para verificar esta afirmação,*

Comande: **N[8 Pi / 15]**

Resposta: 1.67552

12.10 Integrais triplas em sistemas genéricos

Se porventura as novas variáveis não se enquadrarem num dos sistemas dentre os reconhecidos pelo Mathematica, podemos efetuar os cálculos sem necessitar das funções específicas para coordenadas curvilíneas. Se **{ x1, x2, x3 }** designar as variáveis originais e **{ y1, y2, y3 }** as novas variáveis, podemos definir a relação entre elas com os comandos

x1 = F1[y1 , y2 , y3]

x2 = F2[y1 , y2 , y3]

x3 = F3[y1 , y2 , y3]

onde **Fk[y1, y2, y3]**, **k=1, 2, 3,** são as relações funcionais entre as variáveis novas e as originais. Para calcular a matriz jacobiana, basta comandar

mj = Outer[D , { x1 , x2 , x3 } , { y1 , y2 , y3 }]

e, para calcular o jacobiano, use

jacobiano = Det[mj]

Finalmente, para calcular o valor absoluto,

Abs[jacobiano]

Para calcular a integral tripla, utilizaremos novamente a fórmula

$$\iiint_R f(x,y,z)\,dx\,dy\,dz = \iiint_D g(u,v,z)\left|\frac{\partial(x,y,z)}{\partial(u,v,z)}\right| du\,dv\,dz$$

onde $g(u,v,z) = f(x(u,v,z), y(u,v,z), z(u,v,z))$.

Exemplo 12.11 *Vamos calcular a integral*

$$\iiint_R z e^{(y-x)/(y+x)}\,dV$$

onde R é a região definida pelas desigualdades $1 \le y - x \le 2$, $1 \le y + x \le 2$, $0 \le z \le 1$, fazendo a mudança de variável $u = y - x$, $v = y + x$ e $z = z$. Nestas novas variáveis, R se transforma no paralelepípedo D que, nas variáveis (u, v, z), é definido pelas desigualdades $1 \le u \le 2$, $1 \le v \le 2$, $0 \le z \le 1$.

Para garantir que as variáveis **x**, **y**, **z**, **u** *e* **v** *estão limpas,*

Comande: **Clear[x , y , z , u , v]**

Para obter as variáveis originais em termos das novas,

Comande: **sol = Solve[{ u == y − x , v == y + x } , { x , y }]**

Resposta: { { x –> – u + $\frac{u+v}{2}$, y –> $\frac{u+v}{2}$ } }

Para definir x *e* y *devemos capturar as expressões $-u+(u+v)/2$ e $(u+v)/2$ da lista obtida na resposta anterior. Para tanto,*

Comande: **x = sol[[1 , 1 , 2]] ; y = sol[[1 , 2 , 2]]**

Para obter a matriz jacobiana da transformação,

Comande: **mj = Outer[D , { x , y , z } , { u , v , z }]**

Resposta: $\{\,\{-(\frac{1}{2}), \frac{1}{2}, 0\,\}, \{\,\frac{1}{2}, \frac{1}{2}, 0\,\}, \{\,0\,, 0\,, 1\,\}\,\}$

Para calcular a integral,

Comande: **NIntegrate[z Exp[u / v] Abs[Det[mj]] ,**

{ u , 1 , 2 } , { v , 1 , 2 } , { z , 0 , 1 }]

Resposta: 0.739861

Capítulo 13

Transformadas integrais e séries de Fourier

13.1 Transformada de Fourier

Seja $f(t)$ uma função definida para todo t real. A **transformada de Fourier** de $f(t)$ é definida por

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt.$$

Conhecida a transformada de Fourier $F(\omega)$ é possível recuperar a função $f(t)$ através da **transformada inversa de Fourier**, fornecida por

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega) e^{-i\omega t} d\omega \, .$$

Na versão 2 do Mathematica, para calcular a transformada de Fourier e sua inversa é preciso carregar o pacote

```
<< Calculus`FourierTransform`
```

onde lembramos que << é formado por dois sinais de menor do que e o sinal ` é o acento grave, que fica na tecla que contém o til. Nas versões posteriores, a carga deste pacote é dispensável. Para calcular a transformada de Fourier $F(\omega)$ da função $f(t)$ comande

FourierTransform[f[t] , t , w]

e, para calcular a transformada inversa de Fourier de $F(\omega)$, para recuperar $f(t)$, comande

InverseFourierTransform[F[w] , w , t]

Na literatura há certa discrepância quanto às constantes que multiplicam as integrais na transformada de Fourier e sua inversa. Todas elas se enquadram nos casos abaixo: A função **FourierTransform[f[t], t, w]** calcula

$$F(\omega) = \sqrt{\frac{|b|}{(2\pi)^{1-a}}} \int_{-\infty}^{\infty} f(t) e^{ib\omega t} dt$$

ao passo que **InverseFourierTransform** calcula a integral

$$f(t) = \sqrt{\frac{1}{|b|\,(2\pi)^{1+a}}} \int_{-\infty}^{\infty} F(\omega) e^{-ib\omega t} dt,$$

sendo que os parâmetros a e b são estabelecidos pela opção

FourierParameters -> {a, b}

O valor default para o FourierParameters é $\{0, 1\}$, usado na Física Moderna, enquanto que, na Física Clássica, o valor usado é $\{-1, 1\}$. O valor $\{1, -1\}$ é usado na Matemática e na Engenharia. No Processamento de Sinais é usual encontrar $\{a, b\} = \{0, -2\pi\}$. Observe que, para qualquer valor de a e b,

$$\sqrt{\frac{|b|}{(2\pi)^{1-a}}} \sqrt{\frac{1}{|b|\,(2\pi)^{1+a}}} = \frac{1}{2\pi}.$$

Duas funções, o **delta de Dirac** e a **degrau unitário** desempenham um papel importante na teoria das transformadas integrais. O delta de Dirac não é propriamente uma função. Para defini-la com todo o rigor necessário, precisaríamos da Teoria das Distribuições. Para nossos propósitos, basta dizer que, para todo $t \neq 0$,

$$\delta(t) = 0$$

e, para qualquer função $f(t)$ contínua em zero,

$$\int_{-\infty}^{\infty} f(t)\delta(t)\,dt = f(0)\,.$$

Não se define o delta de Dirac em $t = 0$. Podemos, todavia, imaginá-la infinita neste ponto. Integrando por partes, podemos provar que

$$\int_{-\infty}^{\infty} f(t)\delta^{(n)}(t)\,dt = (-1)^n f^{(n)}(0)\,,$$

onde $\delta^{(n)}(t)$ e $f^{(n)}(0)$ denotam as derivadas de ordem n nos pontos considerados. A função degrau unitário $u(t)$ é definida por

$$u(t) = \begin{cases} 0, & \text{se} \quad t < 0\,, \\ 1, & \text{se} \quad t \geq 0\,. \end{cases}$$

No Mathematica, estas funções são denotadas por

```
DiracDelta[ t ]
```

e

```
UnitStep[ t ]
```

Exemplo 13.1 *Vamos calcular as transformadas de Fourier da função delta de Dirac e de*

$$f(t) = \begin{cases} 0, & \text{se} \quad |t| > b\,, \\ 1, & \text{se} \quad |t| \leq b\,. \end{cases}$$

Comande:(na versão 2) $<<$ ***Calculus`FourierTransform`***

Comande: ***Clear[f , g]***

Comande: ***f[t_] = UnitStep[t + 1] − UnitStep[t − 1]***

Resposta: *− UnitStep[− 1 + t] + UnitStep[1 + t]*

Comande: ***g[w_] = FourierTransform[f[t] , t , w]***

Resposta: $\sqrt{\frac{2}{\pi}}\frac{\text{Sin}[w]}{w}$

Nota: *Na versão 4, foi preciso usar as funções ExpToTrig e Simplify para se chegar a este resultado.*

Comande: ***InverseFourierTransform[g[w] , w , t]***

Resposta: (versão 2) – UnitStep[–b – t] + UnitStep[b – t]

Resposta: (versão 4) (1/2)(Sign[1 - t] + Sign[1 + t])

Nota: *A função Sign[x] é igual a* -1 *quando* x *é negativo, é igual a* 1 *quando* x *é positivo e igual a* 0 *quando* x *é a zero. Embora as duas versões forneçam respostas diferentes na aparência, ambas são idênticas.*

Comande: ***FourierTransform[DiracDelta[t] , t , w, FourierParameters -> { 1, -1}]***

Resposta: 1

Comande: ***InverseFourierTransform[% , w , t]***

Resposta: DiracDelta[t]

13.2 Transformada seno e cosseno de Fourier

O pacote que efetua a transformada de Fourier calcula as **transformadas seno e cosseno de Fourier**. Se $f(t)$ for definida para $t \geq 0$, a sua transformada seno de Fourier é igual a

$$F_s(\omega) = \sqrt{\frac{2}{\pi}} \int_0^\infty f(t)\text{sen}(\omega t)\, dt\,,$$

e a sua transformada cosseno de Fourier é dada por

$$F_c(\omega) = \sqrt{\frac{2}{\pi}} \int_0^\infty f(t)\cos(\omega t)\, dt\,.$$

Para calcular estas transformadas, basta comandar

FourierSinTransform[**f**[**t**] , **t** , **w**]

e

FourierCosTransform[f[t] , t , w]

Se uma função $f(t)$ for definida para todo t real e for par, então sua transformada de Fourier se reduz à transformada cosseno e, se for ímpar, a transformada de Fourier se reduz à transformada seno. As **transformadas inversas** são

$$f(t) = \sqrt{\frac{2}{\pi}} \int_0^\infty F_s(\omega)\text{sen}(\omega t)\, d\omega\,,$$
$$f(t) = \sqrt{\frac{2}{\pi}} \int_0^\infty F_c(\omega)\cos(\omega t)\, d\omega.$$

Para calcular estas inversas, basta usar os mesmos comandos que calculam as transformadas, trocando as posições de ω e t.

Da mesma forma que a transformada de Fourier, as transformadas seno e cosseno possuem outras definições que diferem uma da outra por um coeficiente multiplicativo e uma constante a multiplicar o argumento do seno ou do cosseno e que podem ser sumarizadas mediante o uso de dois parâmetros a e b. Esses parâmetros podem ser informados na opção **FourierParameters**. Para maiores detalhes, consulte o manual.

Exemplo 13.2 *Vamos calcular a transformada seno de Fourier de* $\exp(-t)$ *e sua inversa. Continuando a sessão anterior (para a versão 2, carregue o pacote*

$<<$ ***Calculus‘FourierTransform‘****).*

Comande: ***FourierSinTransform[Exp[–t] , t , w]***

Resposta: $\left(\sqrt{2/\pi}\right) w/(1+w^2)$

Comande: ***FourierSinTransform[% , w , t]***

Resposta: E^{-t}

13.3 Séries de Fourier

Se $f(x)$ for periódica, com período $T = 2L > 0$, define-se a sua **série de Fourier trigonométrica** por

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\frac{n\pi x}{L} + b_n \text{sen}\frac{n\pi x}{L}$$

onde

$$a_n = \frac{1}{L}\int_{-L}^{L} f(x)\cos\frac{n\pi x}{L}\,dx$$
$$b_n = \frac{1}{L}\int_{-L}^{L} f(x)\mathrm{sen}\frac{n\pi x}{L}\,dx$$

e define-se a sua **série de Fourier exponencial** por

$$\sum_{n=-\infty}^{\infty} c_n e^{in\pi x/L}$$

onde

$$c_n = \frac{1}{2L}\int_{-L}^{L} f(x)e^{-in\pi x/L}dx\,.$$

Se a função $f(x)$ for conhecida em um período definido pelo intervalo $[a, b]$, pode-se substituir os limites $-L$ e L nas integrais acima, por a e b.

Para utilizar as funções que calculam as séries de Fourier, é preciso usar o pacote

<< Calculus`FourierTransform`

Suponha que $f(x)$ seja conhecida num período definido pelo intervalo $[a, b]$. Para calcular a sua série de Fourier trigonométrica e exponencial até o termo de ordem k, comande

```
FourierTrigSeries[ f[ x ] , { x , a , b } , k ]
```

e

```
FourierExpSeries[ f[ x ] , { x , a , b } , k ]
```

Para obter o coeficiente a_n , comande

```
FourierCosSeriesCoefficient[ f[ x ] , { x , a , b } , n ]
```

para obter o coeficiente b_n, comande

FourierSinSeriesCoefficient[f[x] , { x , a , b } , n]

e, para obter o coeficiente c_n, comande

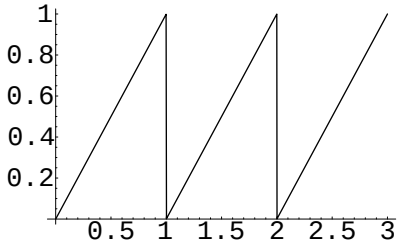
FourierExpSeriesCoefficient[f[x] , { x , a , b } , n]

Exemplo 13.3 *Vamos calcular a série trigonométrica de Fourier da função dente de serra, até o termo de terceira ordem. Tal função é definida por* $f(x) = x$, *para* $0 \leq x < 1$ *e, fora deste intervalo, pela relação de periodicidade* $f(x+1) = f(x)$, *válida para todo* x *real.*

Estando com o pacote $<<$ ***Calculus'FourierTransform'*** *carregado na memória, vamos visualizar o gráfico da função* $f(x)$.

Comande: ***Plot[x – Floor[x] , { x , 0 , 3 }]***

Resposta:



Comande: ***FourierTrigSeries[x , { x , 0 , 1 }, 3]***

Resposta: $\dfrac{1}{2} - \dfrac{Sin[2\ \mathrm{Pi}\ x]}{\mathrm{Pi}} - \dfrac{Sin[4\ \mathrm{Pi}\ x]}{(2\ \mathrm{Pi})} - \dfrac{Sin[6\ \mathrm{Pi}\ x]}{(3\ \mathrm{Pi})}$

Comande: ***FourierSinSeriesCoefficient[x , { x , 0 , 1 }, 3]***

Resposta: $\dfrac{-1}{3\ \mathrm{Pi}}$

Comande: ***FourierCosSeriesCoefficient[x , { x , 0 , 1 } , 3]***

Resposta: 0

13.4 Séries e transformadas numéricas

Os comandos das seções anteriores calculam os valores exatos das transformadas e dos coeficientes. Podemos obter os coeficientes numéricos aproximados, usando os mesmos comandos, precedidos por um **N**.

Exemplo 13.4 *Seja* $f(x) = x^2$, *para* $0 \leq x < 1$ *e* $f(x+1) = f(x)$, *para todo* x *real. Vamos calcular sua série trigonométrica de Fourier até o termo de ordem dois. Vamos calcular a série com os coeficientes exatos para, em seguida, calcular os valores numéricos aproximados.*

Comande: ***FourierTrigSeries[x^2 , { x , 0 , 1 } , 2] ;***

Se o usuário emitir este comando sem o ponto e vírgula no final, notará que o resultado não estará em sua forma mais simples. Para simplificar o resultado,

Comande: ***Simplify[%]***

Resposta: (4 Pi^2 + 12Cos[2 Pi x] + 3 Cos[4 Pi x]−

12 Pi Sin[2 Pi x] − 6 Pi Sin[4 Pi x]) / (12 Pi^2)

Comande: ***NFourierTrigSeries[x^2 , { x , 0 , 1 } , 2]***

Resposta: 0.333333 + 0.101321 Cos[2 Pi x] +

0.0253303 Cos[4 Pi x] – 0.31831 Sin[2 Pi x] –

0.159155 Sin[4 Pi x]

Comande: ***NFourierSinSeriesCoefficient[x^2 , { x , 0 , 1 } , 1]***

Resposta: –0.31831

13.5 Transformada de Laplace

Se $f(t)$ for definida para $t \geq 0$, define-se a sua **transformada de Laplace** por

$$F(s) = \int_0^\infty f(t)e^{-st}dt\,.$$

A transformada inversa existe e é obtida pela fórmula integral de Bromwich (também conhecida por fórmula de Mellin). Esta fórmula envolve integrais de linha no plano complexo. Quando a transformada for simples, podemos

calcular a transformada inversa usando tabelas, sem recorrer à fórmula integral. Existe um pacote encarregado de calcular as transformadas de Laplace e sua inversa. Para carregá-lo, comande

```
<< Calculus`LaplaceTransform`
```

- Este pacote reconhece as funções delta de Dirac e degrau unitário, descritas na seção sobre a transformada de Fourier.

Conhecida a função **f[t]**, para calcular a sua transformada de Laplace **F[s]**, comande

```
LaplaceTransform[ f[ t ] , t , s ]
```

Conhecida a transformada de Laplace **F[s]** de uma função **f[t]**, para obter sua **transformada inversa de Laplace**, basta comandar

```
InverseLaplaceTransform[ F[ s ] , s , t ]
```

Exemplo 13.5 *Para calcular a transformada de Laplace de* $\exp(-t)\cos(t)$ *e a transformada inversa correspondente, carregue o pacote que calcula a transformada de Laplace. Para tanto,*

Comande: ***<< Calculus`LaplaceTransform`***

Carregado o pacote,

Comande: ***LaplaceTransform[Exp[-t] Cos[t] , t , s]***

Resposta: $\dfrac{1+s}{1+(1+s)^2}$

Comande: ***InverseLaplaceTransform[% , s , t]***

Resposta: $\dfrac{\mathrm{Cos}\,[\,t\,]}{\mathrm{E}^t}$

13.6 Delta de Dirac e degrau unitário

Precisando apenas das funções delta de Dirac ou degrau unitário descritas na seção sobre transformadas de Fourier, pode-se carregar o pacote

```
<< Calculus`DiracDelta`
```

no lugar do que calcula as transformadas, desde que não se necessite delas.

Capítulo 14

Constantes e funções embutidas

O Mathematica contém uma gama enorme de funções embutidas. O objetivo deste capítulo consiste em listar aquelas que julgamos serem as principais.

14.1 Constantes

E

representa o valor do limite $E = \lim_{x\to\infty}(1 + \frac{1}{x})^x \simeq 2.71828$

EulerGamma

calcula a constante de Euler $\gamma \simeq 0.577216$

I

representa a unidade imaginária $\sqrt{-1}$

Infinity

representa o infinito positivo ∞

Pi

representa o número π

Autor: Antonio Cândido Faleiros

14.2 Aritmética

Ceiling[**x**]

fornece o menor inteiro maior ou igual a x

Chop[**x**]

substitui **x** por zero, se **x** for menor que 10^{-10}

EulerPhi[**n**]

fornece o número de inteiros positivos menores ou iguais a **n** que são relativamente primos com **n**

Factorial[**n**]

igual a **n!** (fatorial de **n**). Se **n** não for inteiro, **n! = Gamma[1+n]**

Factorial2[**n**]

calcula o fatorial duplo **n!!** de **n**, sendo $n!! = n(n-2)(n-4) \times \ldots$

FactorInteger[**n**]

fornece a lista de fatores primos de **n** juntamente com seus expoentes.

Floor[**x**]

fornece o maior inteiro menor ou igual a **x**

FromDate[{ ano, mês, dia, hora, min, seg }]

retorna o número de segundos transcorridos desde as zero horas do dia 01 de janeiro de 1900 até o instante especificado pela lista **{ ano, mês, dia, hora, min, seg }**

GCD[n1, n2, ...]

fornece o máximo divisor comum de **n1**, **n2**, **...**

Im[z]

fornece a parte imaginária do número complexo **z**

JacobiSymbol[n, m]

calcula o símbolo de Jacobi

$$\left(\frac{n}{m} \right)$$

LCM[n1, n2, ...]

fornece o mínimo múltiplo comum de **n1**, **n2**, **...**

Max[x1, x2, ...]

calcula o maior valor dentre **x1**, **x2**, **...**

Min[x1, x2, ...]

calcula o menor valor dentre **x1**, **x2**, **...**

Mod[m, n]

fornece o resto da divisão de **m** por **n**

NBernoulli[n]

Autor: Antonio Cândido Faleiros

fornece o número de Bernoulli B_n

NProduct[f[n], { n, n1, n2 }]

calcula o valor numérico do produto $\Pi_{n=n1}^{n2} f(n)$

NSum[f[n], { n, n1, n2 }]

calcula o valor numérico da soma $\sum_{n=n1}^{n2} f(n)$

Round[x]

fornece o inteiro mais próximo de **x**

Sqrt[z]

calcula a raiz quadrada de **z**

TimeUsed[ ]

fornece o tempo total de CPU utilizado durante a sessão do Mathematica.

Timing[expr]

calcula a expressão **expr** e o tempo gasto nos cálculos

ToDate[n]

fornece o instante **{ ano, mês, dia, hora, min, seg }** correspondente a **n** segundos depois das zero horas do dia 1 de janeiro de 1900

Autor: Antonio Cândido Faleiros

14.3 Números combinatórios

Binomial[**n** , **m**]

calcula o coeficiente binomial $\binom{n}{m}$ que fornece o número de combinações de **n** elementos, tomados **m** a **m**

Multinomial[**n1**, **n2**, **...**]

fornece o coeficiente multinomial $\frac{(n_1+n_2+\cdots)!}{n_1!n_2!\ldots}$

14.4 Funções trigonométricas

ArcCos[**z**]

calcula o arco cujo cosseno vale **z.**

ArcCot[**z**]

calcula o arco cuja cotangente vale **z**

ArcCsc[**z**]

calcula o arco cuja cossecante vale **z**

ArcSec[**z**]

calcula o arco cuja secante vale **z**

ArcSin[**z**]

calcula o arco cujo seno vale **z**

ArcTan[**z**]

calcula o arco cuja tangente vale **z**

Cos[**z**]

calcula o cosseno de **z**

Cot[**z**]

calcula a cotangente de **z**

Csc[**z**]

calcula a cossecante de **z**

Sec[**z**]

calcula a secante de **z**

Sin[**z**]

calcula o seno de **z**

Tan[**z**]

calcula a tangente de **z**

14.5 Funções hiperbólicas e exponencial

ArcCosh[**z**]

calcula o cosseno hiperbólico inverso de **z**

ArcCoth[**z**]

calcula a cotangente hiperbólica inversa de **z**

ArcCsch[**z**]

calcula a cossecante hiperbólica inversa de **z**

ArcSech[**z**]

calcula a secante hiperbólica inversa de **z**

ArcSinh[**z**]

calcula o seno hiperbólico inverso de **z**

ArcTanh[**z**]

calcula a tangente hiperbólica inversa de **z**

Cosh[**z**]

calcula o cosseno hiperbólico de **z**

Coth[**z**]

calcula a cotangente hiperbólica de **z**

Csch[**z**]

calcula a cossecante hiperbólica de **z**

Exp[**z**]

calcula a exponencial de **z**

Log[**z**]

calcula o logaritmo natural de **z**, isto é, na base e

Log[**b, z**]

calcula o logaritmo de **z** na base **b**

Sech[**z**]

calcula a secante hiperbólica de **z**

Sinh[**z**]

calcula o seno hiperbólico de **z**

Tanh[**z**]

calcula a tangente hiperbólica de **z**

14.6 Funções matriciais

Eigensystem[**m**]

fornece uma lista **{ autovalores, autovetores }** com os auto valores e auto vetores da matriz **m**

Eigenvalues[**m**]

fornece uma lista com os auto valores da matriz **m**

Eigenvectors[**m**]

fornece uma lista contendo os auto vetores da matriz **m**

MatrixExp[**m**]

calcula a exponencial da matriz **m**

MatrixPower[**m, n**]

calcula $\mathbf{m^n}$, a **n**-ésima potência da matriz **m**

SchurDecomposition[**m**]

calcula a decomposição de Schur da matriz **m**

SingularValues[**m**]

calcula a decomposição singular da matriz **m**

Transpose[**m**]

calcula a transposta da matriz **m**

14.7 Funções especiais

AiryAi[**z**]

calcula a função de Airy $Ai(z)$.

AiryAiPrime[**z**]

calcula a derivada da função de Airy $Ai'(z)$.

AiryBi[**z**]

calcula a função de Airy $Bi(z)$.

AiryBiPrime[**z**]

calcula a derivada da função de Airy $Bi'(z)$.

BesselI [**n, z**]

calcula a função de Bessel modificada de primeira espécie $I_n(z)$

BesselJ [**n, z**]

calcula a função de Bessel de primeira espécie $J_n(z)$

BesselK [**n, z**]

calcula a função de Bessel modificada de segunda espécie $K_n(z)$

BesselY [n, z]

calcula a função de Bessel de segunda espécie $Y_n(z)$

Beta[a, b]

calcula a função beta de Euler $B(a, b)$

Beta[z, a, b]

calcula a função beta incompleta $B_z(a, b)$

ChebyshevT[n, x]

fornece o polinômio de Chebyshev de primeira espécie $T_n(x)$

ChebyshevU[n, x]

fornece o polinômio de Chebyshev de segunda espécie $U_n(x)$

CosIntegral[z]

calcula a função integral cosseno Ci $(z) = -\int_z^\infty \frac{\cos t}{t}\, dt$

EllipticE[phi, m]

calcula a integral elíptica de segunda espécie
$E(\phi \,|\, m) = \int_0^\phi \sqrt{1 - m\text{sen}^{\,2}\theta}\, d\theta$
enquanto que **EllipticE[m]** calcula a integral elíptica completa de segunda espécie $E(m) = E(\pi/2 \,|\, m)$

EllipticF[phi, m]

calcula a integral elíptica de primeira espécie $F(\phi|m) = \int_0^\phi 1/\sqrt{1 - m\text{sen}^{\,2}\theta}\, d\theta$

EllipticK[m]

---

calcula a integral elíptica completa de primeira espécie $K(m) = F(\pi/2|m)$

EllipticPi[n, phi, m]

calcula a integral elíptica incompleta
$\Pi(n;\phi|m) = \int_0^\phi \left[(1 - n\text{sen}^{\,2}\theta)\sqrt{1 - m\text{sen}^{\,2}\theta}\right]^{-1} d\theta$
e **EllipticPi[n, m]** calcula a integral elíptica completa de terceira espécie $\Pi(n;m) = \Pi(n;\pi/2|m)$

EllipticTheta[a, u, q]

calcula a função elíptica teta $\theta_a(u|q)$ $(a = 1, \ldots, 4)$

Erf[z]

calcula a função erro erf $(z) = (2/\sqrt{\pi}) \int_0^z \exp(-t^2)\, dt$ e **Erf[z0,z1]**, calcula erf (z_1) − erf (z_2).

Erfc[z]

calcula a função erro complementar erfc $(z) = 1 -$ erf (z)

EulerE[n, x]

fornece o polinômio de Euler $E_n(x)$ que satisfaz à equação geratriz
$2\exp(xt)/(\exp(t) + 1) = \sum_{n=0}^{\infty} E_n(x)\ t^n/n!$.
A função **Euler[n]** calcula $E_n = 2^n E_n(1/2)$

ExpIntegralE[n, z]

calcula a função integral exponencial $E_n(z) = \int_1^\infty \exp(-zt)/t^n dt$

ExpIntegralEi[n, z]

calcula a função integral exponencial Ei $(z) = -\int_{-z}^{\infty} \exp(-t)/t\, dt$

Gamma[z]

calcula a função gama $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$

Gamma[a, z]

calcula a função gama incompleta $\Gamma(a,z) = \int_z^\infty t^{a-1}e^{-t}dt$ e **Gamma[a, z0, z1]** calcula a função gama incompleta generalizada $\Gamma(a,z_0) - \Gamma(a,z_1)$

GegenbauerC[n, m, x]

calcula os polinômios de Gegenbauer $C_n^m(x)$

HermiteH[n, x]

fornece o polinômio de Hermite $H_n(x)$ que é uma solução da equação diferencial $y'' - 2xy' + 2ny = 0$.

Hypergeometric0F1[a, z]

calcula a função hipergeométrica
${}_0F_1(;a;z) = \sum_{k=0}^\infty z^k / \left[(a)_k k!\right]$, onde $(a)_k = \Gamma(a+k)/\Gamma(a)$.

Hypergeometric1F1[a, b, z]

calcula a função hipergeométrica confluente de Kummer
${}_1F_1(a;b;z) = \sum_{k=0}^\infty (a)_k z^k / \left[(b)_k k!\right]$, onde $(a)_k = \Gamma(a+k)/\Gamma(a)$.

Hypergeometric2F1[a, b, c, z]

calcula a função hipergeométrica
${}_2F_1(a;b;c;z) = \sum_{k=0}^{\infty} \left[(a)_k (b)_k/(c)_k\right] z^k/k!$, onde $(a)_k = \Gamma(a+k)/\Gamma(a)$.

HypergeometricU[a, b, z]

calcula a função $U(a,b,z) = (1/\Gamma(a)) \int_0^{\infty} e^{-zt} t^{a-1} (1+t)^{b-a-1} dt$

InverseJacobiPQ[v, m]

onde **P** e **Q** pode ser qualquer par distinto de letras **S**, **C**, **D** e **N**, num total de doze funções, calcula as inversas das funções elípticas de Jacobi $pq^{-1}(v|m)$

JacobiAmplitude[u, m]

fornece a amplitude $\mathrm{am}\,(u|m)$ das funções elípticas de primeira espécie. Se $u = F(\phi|m)$ for a integral elíptica de primeira espécie, então $\phi = \mathrm{am}\,(u|m)$

JacobiP[n, a, b, x]

fornece os polinômios de Jacobi $P_n^{(a,b)}(x)$, que são funções ortogonais com peso $(1-x)^a(1+x)^b$

JacobiPQ[v, m]

onde **P** e **Q** pode ser qualquer par distinto de letras **S**, **C**, **D** e **N**, num total de doze funções, calcula as funções elípticas de Jacobi $pq^{-1}(v|m)$

JacobiZeta[phi, m]

calcula a função zeta de Jacobi $Z(\phi|m) = E(\phi|m) - E(m)F(\phi|m)/K(m)$

LaguerreL[n, a, x]

fornece o polinômio de Laguerre $L_n^a(x)$, uma solução de $xy''+(a+1-x)y'+ny = 0$.

LegendreP[n, x]

fornece o polinômio de Legendre $P_n(x)$

LegendreP[n, m, x]

fornece o polinômio associado de Legendre $P_n^m(x)$

LegendreQ[n, z]

fornece a função de Legendre de segunda espécie $Q_n(z)$

LegendreQ[n, m, z]

fornece a função associada de Legendre de segunda espécie $Q_n^m(z)$

LerchPhi[z, s, z]

calcula a função transcendental de Lerch $\phi(z, s, a) = \sum_{k=0}^{\infty} z^k/(a+k)^s$

LogGamma[z]

fornece $\log\Gamma(z)$, o logaritmo da função gama.

LogIntegral[z]

calcula o logaritmo da função integral logarítmica $\mathrm{li}\,(z) = \int_0^z (1/\log t)dt$

Pochhammer[**a, n**]

calcula o símbolo de Pochhammer $(a)_n = \Gamma(a+n)/\Gamma(a)$

PolyGamma[**z**]

calcula a função digama $\psi(z) = \Gamma'(z)/\Gamma(z)$

PolyGamma[**n, z**]

calcula a **n**-ésima derivada $\psi^{(n)}$ da função digama $\psi(z) = \Gamma'(z)/\Gamma(z)$, sendo que

PolyGamma[**z**] calcula $\psi(z)$.

PolyLog[**z**]

calcula a função poli-logaritmo $\mathrm{Li}_n(z) = \sum_{k=1}^{\infty} z^k/k^n$

RiemannSiegelTheta[**t**]

calcula a função teta de Riemann-Siegel $\vartheta(t) = \mathrm{Im}\ \log\Gamma(1/4+it/2) - (t/2)\log\pi$ para t real

---

RiemannSiegelZ[**t**]

calcula a função **Z** de Riemann-Siegel $Z(t) = \exp(i\vartheta(t))\zeta(1/2+it)$ onde ϑ é a função teta de Riemann-Siegel e ζ é a função zeta (veja **Zeta**) de Riemann

---

SinIntegral[**z**]

calcula a função integral seno $\mathrm{Si}\,(z) = \int_0^z \frac{\mathrm{sen}\,(t)}{t} dt$

WeierstrassP[u, g2, g3]

calcula a função elíptica de Weierstrass $\wp(u; g_2, g_3)$, que fornece o valor de x para o qual $u = \int_{\infty}^{x} (4t^3 - g_2 t - g_3)^{-1/2} dt$

WeierstrassPPrime[u, g2, g3]

calcula a derivada em relação a u da função elíptica de Weierstrass $\wp(u; g_2, g_3)$,

---

Zeta[s]

calcula a função zeta de Riemann $\zeta(s) = \sum_{k=1}^{\infty} k^{-s}$

Zeta[s, a]

calcula a função zeta de Riemann generalizada $\zeta(s, a) = \sum_{k=1}^{\infty} (k + a)^{-s}$

14.8 Transformada discreta de Fourier

Fourier[{ a1 , a2 , ... , an }]

calcula a transformada discreta de Fourier da **lista={ a1, a2, ..., an }**. Esta transformada é uma outra lista **{ b1, b2, ..., bn }**, definida por
$b_s = (1/\sqrt{n}) \sum_{r=1}^{n} a_r \exp[2\pi i (r-1)(s-1)/n]$

InverseFourier[lista]

calcula a lista **{ a1, a2, ..., an }**, que é a transformada de Fourier inversa discreta da **lista={ b1, b2, ..., bn }**. Deste modo,
$a_r = (1/\sqrt{n}) \sum_{s=1}^{n} b_s \exp[-2\pi i (r-1)(s-1)/n]$

14.9 Física quântica

ClebschGordan[{ j1, m1 } , { j2, m2 } , { j, m }]

fornece os coeficientes de Clebsch-Gordan para a decomposição de $|j, m>$ em termos de $|j_1, m_1> |j_2, m_2>$

SixJSymbol[{ j1, j2, j3 } , { j4, j5, j6 }]

calcula os símbolos **6-j** de Racah

14.10 Comandos repetitivos

Do[expr , { n, ni, nf, dn }]

calcula o valor da expressão **expr** diversas vezes. Inicialmente, faz **n=ni** e calcula **expr**. O processo se repete fazendo sucessivamente **n = ni + dn**, **n = ni + 2dn, ...** e interrompendo o processo quando **n** se tornar maior que **nf**.

For[início, teste, acréscimo, corpo]

executa o **início**, em seguida, se o **teste** for verdadeiro, executa o **acréscimo** que contém uma variável que modifica o valor do **teste**. Finalmente o **corpo** é calculado. O **teste** é novamente executado e, enquanto ele for verdadeiro, o processo se repete.

Which[teste1, expr1, teste2, expr2, ...]

verifica sucessivamente **teste1**, **teste2**, ... Quando o primeiro teste verdadeiro for encontrado, o **Which** calcula a expressão que se encontra à sua frente e devolve o seu valor.

While[teste, corpo]

calcula **teste** e **corpo** repetidamente, até que o **teste** seja falso.

Índice Remissivo