Mathematica básica

E. Marques de Sá
DMUC, 2008-09

Instruções breves sobre o programa *Mathematica*.

Utilizo o *Mathematica 5.2*, versão já ultrapassada por outras mais recentes, *Mathematica 6* e *Mathematica 7*, mas que chega e sobra para se praticarem muitas tarefas, desde as mais elementares até um nível muito elevado de sofisticação.

Nos computadores de mesa, *Enter*[1] é a tecla geral de execução. Nos portáteis, essa função é desempenhada por *Shift*+↩.[2] A ajuda do *Mathematica* é de nível excepcional. Pode chamá-la de vários modos, em geral premindo a tecla F1; se colocar o cursor numa palavra de código, como `Table`, ou `Circle`, e premir F1, o *help-browser* abrirá no local apropriado, com explicações muito bem concebidas. Note que a ajuda funciona dentro da própria ajuda, como num dicionário (e com o mesmo grau de circularidade dos dicionários! :-).

Há certos comandos que têm opções que pode acrescentar à medida das suas necessidades; se quiser ver a lista completa de opções dum comando qualquer, execute `Options[`*comando*`]`. Claro que, em primeiro lugar, tem que aprender alguns dos comandos internos do *Mathematica*, sendo esse um dos objectivos deste texto.

Aquilo que escrevermos na janela aparecerá como *input*, e os resultados dos cálculos efectuados pela máquina aparecerão como *output*; as siglas são "`In`[m]:=" e "`Out`[n]=", onde m, n são números de ordem dos nossos *inputs* e dos *outputs* da máquina. Se o *input* for `2+5` o *output* será `7`. Se não quiser que a máquina escreva o *output*, use ponto-e-vírgula, *e.g.*, o *input* `2+5;` não terá *output* visível, mas a máquina efectua e memoriza o cálculo. A percentagem, `%`, representa o último *output* obtido pela máquina; por exemplo, o *input* `2+5; %^2` produz o *output* `49`. Se a seguir escrever `1+%` o *output* será `50`.

Máquina de calcular

Introduza uma expressão numérica simples, como `100!`, `2+2`, `(3-5)^2`, `4*(3+0.36)`, `2^(3!+1)`, `(2/3+1)^(1/2)`, e prima *Enter*. Escreva outra e

[1] Nas teclas numéricas na extrema direita do teclado autónomo.
[2] É assim, pelo menos, nos computadores em que trabalho.

prima *Enter*, etc. A multiplicação pode denotar-se por um ou vários espaços em branco: `2  3` é o mesmo que `2 3` e `2*3`, mas diferente de `23`.

O *Mathematica* é muitas vezes insensível a espaços em branco e mudanças de linhas. Por exemplo, `(((x-y)+(1-(z+w)^(2(a+b))))^(-1))^3` pode escrever-se de forma 'folgada' que facilite a escrita e a leitura

```
(
    (    (x - y) + 1 - (z + w)^(2(a + b))    )^(-1)
)^3
```

Mas os nomes de comandos, variáveis e funções não podem ser separados por espaços; por exemplo, `aaaa` é nome de variável, mas `aa aa` equivale a `aa*aa`.

Parêntesis

Em cálculo aritmético os parêntesis têm de ser curvos, como acima. Os rectos usam-se exclusiva e obrigatoriamente para abarcar os argumentos de funções, como em `f[x], Sin[x], Cos[Pi], Exp[1], Sqrt[2(x+y)]`. As chavetas usam-se para n-uplos ordenados, como em `{a,b,2(x+y),Sin[c]}`.

Criação de variáveis e funções

Para variáveis pode tomar os símbolos tradicionais, `x`, `y`, `z`, ou `x1`, `x2`, `x3`, etc., mas qualquer agrupamento de letras e algarismos que não comece por um algarismo serve como variável. Por exemplo: `hoje`, `amanha`, `eles`, `vari1b23`. Note que `3var2` significa o triplo da variável `var2`. Quando escreve e executa `hoje=a+x`, o *Mathematica* cria um lugar de memória a que chama `hoje`, calcula `a+x` usando os valores que nesse momento `a` e `x` têm e coloca o valor calculado no lugar de memória `hoje`.

Há centenas de variáveis e funções disponíveis internamente no *Mathematica*, e o utilizador pode criar quantas quiser. A notação funcional tem a seguinte sintaxe: f`[x]`, onde f denota um nome de função à sua escolha. *Todas* as funções e opções internas do *Mathematica* começam por letras maiúsculas, *e.g.*, `Sin`, `E`, `I`, `ParametricPlot`, `Random`, `AspectRatio`, `Line`, `Point`, `Rectangle`, `Do`, `Until`, etc.; por isso convém que variáveis e funções definidas por si sejam por si baptizadas com nomes começados por minúsculas.[3]

[3]Se, por distracção, utilizar as letras `A`, `B`, `C`, `D`, `E` como variáveis suas escrevendo, por exemplo, E=4, o *Mathematica* queixa-se dizendo-lhe que `E` é "símbolo protegido" (de facto, `E` representa o número de Neper).

Pode criar uma função sua utilizando funções internas do *Mathematica*, *e.g.*:

```
fi[x_]:=Sin[2x]^n
```

onde, à esquerda de `:=`, cada variável leva uma barra inferior, mas à direita não podem colocar-se tais barras. Premindo *Enter*, a máquina memoriza a expressão de `f[x]` sob a forma de um algoritmo, *sem output!* Depois, pode escrever coisas como `f[2]` ou `f[Pi]^2` e premir *Enter* para obter os valores pretendidos.[4]

Pode definir funções com várias variáveis, e pode defini-las usando funções suas anteriormente definidas, como em

```
modulo[alfa_,beta_]:=Sqrt[alfa^2+beta^2]
```

`etc[x_,y_]:=fi[x+y]+Tan[x^2]`, etc..

As variáveis e funções podem ter nomes extensos, mas não podem começar por números. Assim, `var21` pode ser o nome de uma variável, mas `21var` representa o mesmo que `21*var`.

Exemplos. Defina a área dum círculo como função do raio:
`area[raio_]:=Pi raio^2`; depois calcule, *e.g.*, `area[5]`. Defina o volume dum cilindro recto de revolução como `vol[raio_,alt_]:=Pi raio^2 alt` e calcule, *e.g.*, `vol[2,10]`.

A função `Clear`

Numa sessão contínua, desde que abre até que fecha o *Mathematica*, o programa memoriza *todas* as funções e variáveis criadas pelo utilizador. Se uma sessão for muito longa é natural que muitas funções e variáveis tenham sido criadas. Muitos dos erros e resultados inesperados de certos cálculos (numéricos, gráficos, simbólicos, etc.) devem-se a sobreposições de significados de certas expressões por nós definidas. Habitue-se a usar a função `Clear` para evitar atropelos. Por exemplo, se definiu `x=1; y=2;` e, uma hora mais tarde, quiser expandir um polinómio, digamos `Expand[(x + y)^3]`, não se admire que o resultado seja 27 e não `x^3+3x^2 y+3x y^2+y^3`. Para obter este desenvolvimento particular do binómio, mande em primeiro lugar que o *Mathematica* apague eventuais valores memorizados para as suas novas variáveis `x` e `y`:

```
Clear[x, y]
```

[4] O sinal `:=` indica uma *atribuição diferida*. Por exemplo, se escrever e executar `hoje:=a+x`, o *Mathematica* coloca no lugar de memória a expressão `a+x`, sem a calcular, adiando esse cálculo para cada uma das ocasiões em que você decida chamar a variável `hoje`.

```
Expand[(x + y)^3].
```

Gráficos de funções

Para o gráfico de `f[x]` no intervalo $[-1, 3]$, a sintaxe mais simples é

```
Plot[ f[x] , {x,-1,3} ]
```

Pode representar várias funções no mesmo diagrama:

```
Plot[  {f[t],g[t],Cos[t]}  ,  {t,0,1}  ]
```

Se quiser acrescentar opções, escreva-as separando-as por vírgulas:

`Plot[ f[x] , {x,-1,3} ,` *opção, opção, ...* `]`

Eis duas opções importantes:

```
AspectRatio->Automatic
```

```
PlotRange->{{x1,x2},{y1,y2}}
```

A primeira manda desenhar as figuras com escala do eixo dos x's igual à do eixo dos y's, a segunda especifica a janela de observação.

> Por defeito, `AspectRatio` é a razão áurea e `PlotRange` é determinado pela máquina: de facto, ela escolhe uma janela de observação de um modo 'inteligente' e determina a escala dos eixos de tal modo que o desenho final seja um rectângulo de ouro! Esta metodologia não serve caso queira respeitar as proporções das figuras a desenhar, ou ter uma percepção visual convincente da derivada... experimente traçar o gráfico de $\sin x$ em $[0, 4\pi]$....

A função `Options`

Trata-se de uma função muito útil para se saber o que pode o *Mathematica* fazer com certas funções internamente definidas. Por exemplo, o comando `Options[Sin]` dá-lhe uma lista vazia de opções para o seno.

Se quiser ver a lista completa das opções de que dispõe para a função `Plot`, mande executar `Options[Plot]`; o resultado é uma longa lista de opções com indicação dos respectivos valores por defeito. De início, você não saberá o significado muitas das opções que surgem na lista produzida pelo *Mathematica*, nem saberá como agir para as alterar; para o saber faça o seguinte: coloque o cursor sobre determinada opção e clique a tecla de ajuda F1.

Curvas em coordenadas paramétricas

Para desenhar este tipo de curvas a sintaxe mais simples é

```
ParametricPlot[  {f[t],g[t]}  ,  {t,0,1}  ]
```

Pode representar várias curvas na mesma figura:

```
ParametricPlot[{ {f[t],g[t]} , {h[t],i[t]} , {j[t],k[t]} },
                                                {t,0,1}  ]
```

Exemplos interessantes. Recorde a representação de curvas em coordenadas polares

$$\begin{cases} x = r(t)\ \cos t \\ y = r(t)\ \sin t. \end{cases}$$

Escolha, a seu gosto, uma função `r[t]` e introduza

```
ParametricPlot[ r[t]{Cos[t],Sin[t]}  ,  {t,0,2Pi}  ,
                                    AspectRatio->Automatic  ]
```

Experimente os casos $r(t) = t$, $r(t) = e^t$, $r(t) = 1.2 + \sin 3t$, etc..

Cálculo simbólico

O *Mathematica* trabalha razoavelmente bem com polinómios. Escreva

```
Clear[a,b,c,d,e,f,x]
plof=(a x^2+b x+c)(d x^3+e x+f).
```

> Note que `plof` não é uma função mas sim uma variável; o que a máquina faz é criar um lugar de memória chamado "`plof`" e colocar lá dentro a expressão simbólica `(a x^2+b x+c)(d x^3+e x+f)`[5]; mas a máquina não entenderá o significado de `plof[1]`, por exemplo. Se quisermos que `plof` seja uma função, teremos que introduzir um dos *inputs*
>
> `plof[x_]:=(a x^2+b x+c)(d x^3+e x+f)`, ou
> `plof[x_,a_,b_]:=(a x^2+b x+c)(d x^3+e x+f)`, ou...

Depois, execute `Expand[plof]`. De seguida, execute `Factor[%]`. Se quer as raízes, mande executar: `Solve[plof==0,x]`. Note que `plof=0` coloca no lugar `plof` da memória o número `0`; a expressão `plof==0` é outra coisa: trata-se de uma *condição* (ou *equação*) *que representa um valor lógico*, verdadeiro ou falso, para cada valor atribuído às variáveis envolvidas na expressão. É fundamental a explicitação da variável em ordem à qual pretende resolver a equação. Experimente executar `Solve[plof==0,a]` e veja a diferença.

[5] Expressão simbólica sem atribuição de valores às 7 variáveis por via da função `Clear`.

Execute `TrigExpand[Sin[a+b+c+d+e]]`. De seguida, execute `Simplify[%]`. Complique mais a expressão trigonométrica e adivinhe as potencialidades deste cálculo.

Derivadas

Memorizada uma expressão designatória, `expr`, a primeira derivada de `expr` em ordem a `x` é `D[expr,x]`. A derivada de ordem n é `D[expr,{x,`n`}]`.

Exemplos.

(a) Execute `Sin[x]^2` e, *repetidamente*, execute `D[%,x]`.

(b) Execute `exp=Tan[x]^10`. Depois, execute `D[exp,{x,10}]`. De seguida, execute `Simplify[%]`

Listas

Uma *lista* é uma sequência finita e ordenada de objectos, da natureza que se queira (números, letras, palavras, gráficos, ...) e com eventuais repetições; escrevem-se de acordo com a sintaxe dos seguintes exemplos:

```
{x1,x2}  {x,y,z}  {a,a,b,a,a}  {g,h,zz,zz,zz,zz}  ...
```

A maneira mais popular de construir listas é através da função `Table`, com a seguinte sintaxe:

```
Table[ f[x] , {x,5,9} ]
```

Este *input* produz a lista `{f[5],f[6],f[7],f[8],f[9]}`. Podemos também usar não inteiros e especificar o passo da variável x; por exemplo, o *input*

```
Table[ f[x] , {x,.1,2,.5} ]
```

produz a lista `{f[.1],f[.6],f[1.1],f[1.6]}`.

Uma lista pode ser constituída por pares ordenados. Pode, por exemplo, escolher duas funções interessantes, `f[x],g[x]`, e escrever

```
Table[ {f[x],g[x]} , {x,0,1,.01} ]
```

O *output* é uma lista ordenada de 101 pares ordenados. Neste exemplo

```
ptgr=Table[ {Cos[k*4Pi/5],Sin[k*4Pi/5]} , {k,0,5} ]
```

o *output* é a lista das coordenadas cartesianas dos vértices dum pentágono regular inscrito na circunferência unitária, com o vértice `{1,0}` repetido, ordenados de acordo com o traçado do pentagrama.

Dada uma lista `h`, `Length[h]` é o comprimento de `h`. `h[[i]]` é o `i`-ésimo elemento da lista. Se `h[[i]]` é uma lista, `h[[i,k]]` é o `k`-ésimo elemento de `h[[i]]`. Por exemplo, `ptgr[[3,1]]` é a abcissa do terceiro vértice do pentagrama; note que `ptgr[[1]]=ptgr[[6]]={1,0}`.

Gráficos *à la carte*

As funções `Plot` e `ParametricPlot` são macro-comandos que permitem fazer gráficos de funções à sua escolha, onde o *Mathematica* dispõe, por defeito, de larga autonomia na escolha inteligente do escalonamento dos eixos e do rectângulo de visualização. Mas os gráficos que provavelmente você gostaria de fazer não são desse tipo; poderão ser segmentos de recta, linhas quebradas, arcos de circunferência, pontos e rótulos, polígonos coloridos combinados de acordo com um plano seu. Chama-se a isso *objectos gráficos*, os quais vão ser construídos por combinação de objectos gráficos internamente definidos no *Mathematica*, estes chamados objectos gráficos *primitivos*,

Os objectos gráficos primitivos bidimensionais mais importantes são os seguintes (notação: `P,P1,P2,...` são pares ordenados de reais; `r` é um real positivo):

`Point[P]`, ponto

`Line[{P1,P2,...}]`, linha poligonal

`Circle[P,r]`, circunferência

`Disk[P,r]`, círculo preenchido

`Rectangle[P1,P2]`, rectângulo a cheio, com "diagonal" `[P1 P2]`

`Polygon[{P1,P2,...}]`, polígono preenchido

`Text[FontForm["texto", "Times-Italic", 12], P ]`, texto em `P`.

Muitas destas instruções primitivas têm parâmetros de controlo adicionais. Por exemplo, `Circle[P,{a,b},{`θ_1`,`θ_2`}]` representa um arco de elipse de semi-eixos `a,b` 'entre' os ângulos $\theta_1 < \theta_2$ (contados a partir do semi-eixo $\dot{O}X$). Pode também controlar a cor dos objectos preenchidos, a espessura de pontos e linhas. Estes pormenores e outros poderão ser elucidados à medida das necessidades, mediante a ajuda do *Mathematica*.

Para mandar desenhar um gráfico escreva algo com este formato:

```
Show[ Graphics[{obj1,obj2,...}], op1,op2,...]
```
(1)

onde `obj` e `op` denotam objectos gráficos e opções suas para a função `Show`. Por exemplo,

```
Show[   Graphics[{Line[ptgr],Circle[{0,0},1]}]   ]
```

produz um pentagrama e a circunferência unitária em que está inscrito; a representação sai deformada pois, por defeito, as figuras são encaixadas num rectângulo áureo. Para evitar isso, use a opção que já conhece e escreva:

```
Show[   Graphics[{Line[ptgr],Circle[{0,0},1]}]  ,
                                        AspectRatio->Automatic ]
```

Para desenhar o pentagrama a traço grosso e a circunferência unitária a tracejado, escreva e execute:

```
grosso={ Thickness[.02] , Line[ptgr] };
tracejado={ Dashing[{.02,.02}] , Circle[{0,0},1] };
Show[  Graphics[{grosso,tracejado}]  ]
```

Gráficos 3D

Para o traçado de gráficos tridimensionais, as instruções mais populares são do seguinte estilo

```
Plot3D[  x^2+y^2 , {x,-1,1}, {y,-1,1}]
ParametricPlot3D[ { r Cos[t],r Sin[t],r^2 }
                                        , {r,0,2}, {t,0,2Pi}]
```

Nestes exemplos, o *output* é o 'mesmo' parabolóide de revolução; no primeiro caso, o par (x, y) percorre um quadrado, no segundo percorre um círculo de raio 2 (note que `r` e `t` são as coordenadas polares no plano XOY).

A função `ParametricPlot3D` permite traçar curvas e superfícies que apenas dependem da sua imaginação: defina funções quaisquer, `x[s_]:=`$(\dots)$, `y[s_]:=`$(\dots)$, `z[s_]:=`$(\dots)$, e depois execute

```
ParametricPlot3D[ { x[s],y[s],z[s] }, {s,0,1}]
```

para obter uma curva em $\mathbb{R}^3$ (é curva por depender de apenas um parâmetro). Defina funções quaisquer de duas variáveis, `x[r_,t_]:=`$(\dots)$, `y[r_,t_]:=`$(\dots)$, `z[r_,t_]:=`$(\dots)$, e mande executar, por exemplo:

```
ParametricPlot3D[ { x[r,t],y[r,t],z[r,t] }
```

```
                                                    , {r,-1,2},{t,0,4}]
```

para obter uma superfície.

Exemplos. (*a*) Para desenhar uma hélice pode usar o programa:

```
x[t_]:=Cos[t]; y[t_]:=Sin[t]; z[t_]:=t/20
ParametricPlot3D[ { x[t],y[t],z[t] }, {t,0,60}]
```

Toda a curva do *Mathematica* é uma linha quebrada. Esta hélice mostra bem isso. Se quer uma linha quebrada de que não se notem as quebras, use a opção `PlotPoints` para aumentar o número de pontos calculados, *e.g.*,

```
ParametricPlot3D[ { x[t],y[t],z[t] }, {t,0,60}]
            , PlotPoints-> 1000, Boxed-> False, Axes->False]
```

A opção `Boxed-> False` serve para eliminar o caixilho que envolve, por defeito, todos os gráficos 3D, e `Axes->False` elimina os eixos coordenados.

(*b*) Se quer ver uma espiral cónica, altere as coordenadas paramétricas da alínea (*a*) para, *e.g.*, `x[t_]:=t Cos[t]; y[t_]:=t Sin[t]; z[t_]:=t`. Mande, então, executar a função

```
ParametricPlot3D[{x[t], y[t], z[t]}, {t, 0, 80Pi},
                Boxed->False, Axes->False, PlotPoints->2000]
```

Trata-se de uma espiral "de 40 voltas" da qual se calculam 2000 pontos. Claro que o números de pontos em `PlotPoints` deve ser proporcional ao intervalo de variação do parâmetro `t` e proporcional ao grau de "aparência de suavidade" que queira dar à representação da curva.

(*c*) Aqui vai uma superfície com coordenadas paramétricas escolhidas mais ou menos à sorte:

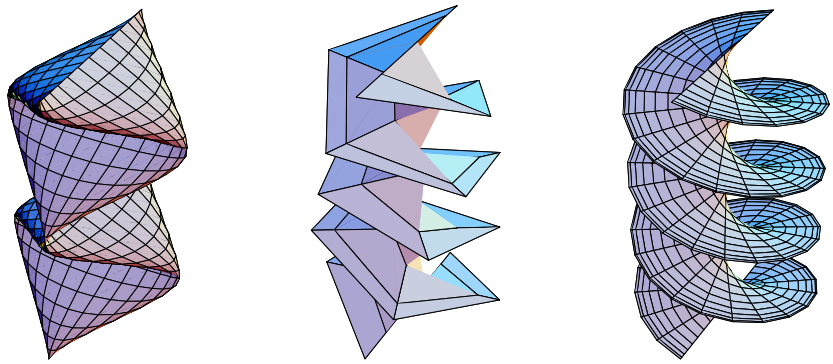
```
Clear[x,y,z]
x[r_,t_]:=Sin[r]Cos[t]
y[r_,t_]:=Sin[t]Cos[r]
z[r_,t_]:=r/2
ParametricPlot3D[ {x[r,t],y[r,t],z[r,t]},
            {r,0,2Pi}, {t,0,2Pi}, Boxed->False, Axes->False]
```

O resultado é a figura da esquerda na página a seguir. À direita surge o resultado do programa:

```
x[r_,t_]:=Sin[r]Cos[t]
y[r_,t_]:=Cos[t]Cos[r]
z[r_,t_]:=r/4
ParametricPlot3D[ {x[r,t], y[r,t], z[r,t]} ,
```

```
{r,0,4Pi}, {t,0,2Pi},
Boxed->False, PlotPoints->100, Axes->False]
```

A única alteração substancial relativamente ao caso anterior foi ter-se feito `y[r_,t_]:=Cos[t]Cos[r]` (a divisão por `4` em `z[r_,t_]:=r/4` apenas altera a escala do eixo OZ).



A figura central obteve-se com o mesmo programa da figura à direita, mas com a opção `PlotPoints->10`.

Elas ilustram bem como o *Mathematica* representa superfícies: por colagem de quadriláteros, cada um deles com vértices sobre a superfície. Trata-se de poliedros muito complicados que aproximam a superfície imaginada: o do centro tem cerca de 100 vértices e os das pontas têm cerca de 900 vértices cada um.

Gráficos 3D *à la carte*

Reveja o caso 2D e tenha em conta que, em 3D, os pontos têm 3 coordenadas. Tudo é semelhante ao caso bidimensional: os objectos gráficos primitivos são os mesmos, os comandos `Thickness` e `PointSize` funcionam da mesma forma, etc., e o processo de produção é quase o mesmo que (1) na página 8:

```
Show[ Graphics3D[{obj1,obj2,...}], op1,op2,...]
```

Problemas

1. *Sucessões pseudo-aleatórias.* Certas sucessões dizem-se *aleatórias*[6] pelo carácter errático, imprevisível, caótico que apresentam.[7] Por exemplo, para um observador externo às bolsas de valores, as oscilações das cotações bolsistas, expressas diariamente, constituem sucessões aleatórias. Há programas de computador que geram sucessões, deterministicamente!, com um aspecto de coisa aleatória. Essas sucessões chamam-se *pseudo-aleatórias*; frequentemente, chamamos *aleatórias* a essas sucessões pseudo-aleatórias e os programas que as produzem dizem-se *geradores de números aleatórios.* Os mais populares são baseados na iteração de uma função f, muito bem escolhida, com domínio e conjunto de chegada iguais a $[0,1]$. Escolhe-se um número inicial, s, chamado semente ("*seed*") e itera-se f:

   $$x_0, f(s), ff(s), fff(s), ffff(s), \ldots$$

   Descubra uma função f tal que a sucessão acima tenha um "aspecto aleatório". Para visualizar uns milhares de termos da sua sucessão, faça uma tabela e um gráfico usando `ListPlot` (experimente a opção `PlotJoined`). Vá melhorando a sua descoberta, até que o gráfico se pareça com uma evolução bolsista.

   O *Mathematica* possui uma função f =`Random[ ]`, internamente definida, que produz sucessões (pseudo)aleatórias. Cada vez que se executa a instrução `Random[ ]` o *output* é $f(\alpha)$, onde α é o número aleatório calculado pelo *Mathematica* imediatamente antes da dita execução. Se quiser fixar uma semente, execute `SeedRandom[`n`]`, com n um inteiro à sua escolha. Experimente o programa:

   ```
   SeedRandom[4]; Table[Random[ ],{i,1,20}]
   ```

   Execute-o outra vez, veja o resultado e explique-o...

2. `{Random[ ],Random[ ]}` produz um par (pseudo-)aleatório no quadrado $[0,1]\times[0,1]$. O *input*

   ```
   nuvem = Table[Point[{Random[ ],Random[ ]}], {i,1,10000}];
   ```

   gera uma nuvem de 10000 pontos aleatórios.

   (a) Represente graficamente essa nuvem.

   (b) Represente graficamente nuvens de pontos aleatórios de coordenadas `{Random[ ]^n,Random[ ]^m}`, para diversos valores de `m` e `n` à sua escolha. Interprete as diferenças, bem visíveis, entre essas nuvens.

[6] "*Random*", em inglês.

[7] Não se trata, aqui, de uma definição precisa, pois a nenhum dos termos —*errático*, *imprevisível* e *caótico*— é dado um significado preciso.

3. Desenhe um polígono regular de n lados. Desenhe estrelas de n pontas. (*Sugestão.* Reveja o caso do pentágono e do pentagrama. Escreva um programa envolvendo a variável n e coloque, no início, n =*valor numérico à sua escolha.*)

4. Represente graficamente funções do seguinte tipo, ditas *ondas sinusoidais de amplitude modulada* ("AM", na gíria radiofónica!):

$$f(t) = A(t)\sin(\omega t + \varphi).$$

   Experimente valores diversos de $A(t)$, por exemplo, funções $A(t)$ periódicas, e valores elevados da constante ω.

5. Represente graficamente funções do seguinte tipo, ditas *ondas sinusoidais de frequência modulada* ("FM", na gíria radiofónica!):

$$f(t) = A\sin(\omega(t)\, t).$$

   Experimente casos diversos de $\omega(t)$. Considere, por exemplo, funções $\omega(t)$ periódicas.

6. Desenhe rosáceas utilizando a função `ParametricPlot`. Tudo vai depender da sua imaginação!

7. Faça uma lista das 30 primeiras derivadas de uma função à sua escolha.

8. Desenhe uma folha de papel quadriculado. Coloque um símbolo à sua escolha no centro de cada quadradinho. (*Sugestão.* Defina uma função `sim[x_,y_]` que seja o seu símbolo preferido colocado no ponto genérico `{x,y}`. Use `Table` para fazer a grelha quadriculada e para fazer uma lista dos símbolos a colocar.)

9. Desenhe uma malha triangular regular, dentro dum triângulo equilátero.

10. Desenhe uma malha hexagonal regular, dentro dum hexágono regular.

11. Desenhe um tabuleiro de *Hex*.

12. Simule o lançamento de um dado, gerando aleatoriamente naturais de 1 a 6. Se tiver gosto nisso, construa as seis faces do dado e faça-as aparecer no output em vez dos respectivos números. (*Sugestão.* Escreva `Random` e chame a ajuda. Para mostrar os bonecos das faces use `Which`.)

13. *O triângulo de Sierpinsky.* Tome um triângulo equilátero dado pelos seus vértices: `tri={a,b,c}`. Defina a função de `med[p_]` como sendo o ponto médio do segmento de une `p` e a um vértice de `tri` aleatoriamente escolhido. Comece com um ponto qualquer p0 e itere a função `med` muitas vezes. Desenhe a nuvem de pontos assim construída. (*Sugestão.* Construa uma tabela de pares em que cada par é a imagem por `med` do par anterior. Depois aplique a função `Point` a cada elemento da tabela (chame a ajuda e veja `Map`). Depois faça `Show`, como de costume.)

14. Construa a tabela do seguinte jogo de estratégia. Dois jogadores tiram alternadamente pedras de um monte. Na sua vez de tirar, cada um pode tirar um quadrado perfeito positivo. Perde quem tirar a última.

15. Faça a tabela do jogo de tirar da *Olimpíada Ibero Americana de Caracas 2000*.

16. *Jogo dos 24.* Trata-se de um jogo muito popular na China, tendo sido, ao que parece, criado em Xangai há cerca de meio século. São dados 4 números, com eventuais repetições; com eles, e só eles, cada jogador procura "fazer 24" usando as 4 operações aritméticas elementares e os parêntesis que queira. Todos os 4 números têm que ser usados, com as devidas repetições. Por exemplo, dados $2, 2, 4, 8$, pode fazer 24 assim: $(4 - 2/2) * 8$; mas, se os números forem $3, 10, 10, 11$, não vale usar a expressão $3 + 10 + 11$, pois "10" deve ser usado duas vezes. Ganha o jogador que primeiro consiga dizer uma expressão válida, com esses números, que dê 24.

    O jogo costuma ser jogado com um baralho tradicional de 52 cartas, cada uma com o seu valor facial de 1 a 13 (valetes, damas e reis valendo 11, 12 e 13, respectivamente).

    Faça um programa que jogue este jogo. Por exemplo, dados quatro números, construa um programa que tenha *output* "S", caso exista uma expressão que, com eles, dê 24, e output "N", no caso contrário; ou construa um programa que tenha como *output* a lista de *todas* as expressões que, com esses números, dêem 24.

    Produza uma lista de todos os quartetos de inteiros, de 1 a 13, com os quais não é possível fazer 24.

17. *Passeio aleatório.* Um indivíduo embriagado deambula aleatoriamente num plano. Os passos que dá têm todos o mesmo comprimento mas, após cada passo, a direcção do próximo é escolhida ao acaso. Desenhe uma trajectória de muitos passos nestas condições. Veja se descobre experimentalmente a relação (estatística!) entre o número n de passos dados e a maior distância à origem atingida no decorrer desse passeio de n passos.