

Mathematica 简介

Mathematica 是一个庞大且复杂的系统。它内置几千个函数，能完成科学、数学、工程等学科中的各种工作，包括数值和符号计算，编程，数据分析，知识表达，和信息可视化。在这个入门章节里，我们将通过一些各领域的编程及计算范例去感受其广度和深度。章末会介绍一些入门话题，包括如何输入并运行表达式，如何处理错误，以及如何通过文档系统获得帮助。本章对于已经熟悉 *Mathematica* 的用户来说，略读一下即可。

1.1 基本操作概述

数值和符号计算

在基础阶段，可以把 *Mathematica* 当作一个复杂的计算器来用。你可以输入数学表达式并计算它们的值。

$$\text{In[1]:= } \sqrt{2.0 \times 10 \pi} \left(\frac{10}{e} \right)^{10}$$

$$\text{Out[1]= } 3.5987 \times 10^6$$

你可以将结果存储在内存里以便在后续的计算中使用。例如，下面的三个输入算出了以半光速移动的物体的洛伦兹因数。

$$\text{In[2]:= } c = 299\,792\,458 \frac{\text{Meter}}{\text{Second}};$$

$$\text{In[3]:= } v = \frac{c}{2}$$

$$\text{Out[3]= } \frac{149\,896\,229 \text{ Meter}}{\text{Second}}$$

$$\text{In[4]:= } \gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}$$

$$\text{Out[4]= } 0.866025$$

Mathematica 与计算器和简单的计算机程序不同之处在于它能得出精确的结果并且能计算至任意的精度。

$$\text{In[5]:= } \frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \frac{1}{13}$$

$$\text{Out[5]= } \frac{40\,361}{30\,030}$$

$$\text{In[6]:= } 2^{1024}$$

$$\text{Out[6]= } 179\,769\,313\,486\,231\,590\,772\,930\,519\,078\,902\,473\,361\,797\,697\,894\,230\,657\,273\,430\,081\, \dots \\ 157\,732\,675\,805\,500\,963\,132\,708\,477\,322\,407\,536\,021\,120\,113\,879\,871\,393\,357\,658\,789\, \dots \\ 768\,814\,416\,622\,492\,847\,430\,639\,474\,124\,377\,767\,893\,424\,865\,485\,276\,302\,219\,601\,246\, \dots \\ 094\,119\,453\,082\,952\,085\,005\,768\,838\,150\,682\,342\,462\,881\,473\,913\,110\,540\,827\,237\,163\, \dots \\ 350\,510\,684\,586\,298\,239\,947\,245\,938\,479\,716\,304\,835\,356\,329\,624\,224\,137\,216$$

$$\text{In[7]:= } \sin[2017 \times 2^{1/5}], 40]$$

$$\text{Out[7]= } -0.9999999999999999785677712610609832590685$$

Mathematica 最引人注意的一个特征就是操作和计算符号表达式的能力。例如，你可以因式分解一个多项式或者化简三角表达式。

```
In[8]:= Factor[x7 - 1]
```

```
Out[8]= (-1 + x) (1 + x + x2 + x3 + x4 + x5 + x6)
```

```
In[9]:= TrigReduce[Sin[3 θ]5]
```

```
Out[9]=  $\frac{1}{16} (10 \sin[3 \theta] - 5 \sin[9 \theta] + \sin[15 \theta])$ 
```

你可以利用对表达式所含变量的假设进行化简。比如，假设 k 是整数，则 $\sin(2\pi k + x)$ 化简为 $\sin(x)$ 。

```
In[10]:= Assuming[k ∈ Integers, Simplify[Sin[2 π k + x]]]
```

```
Out[10]= Sin[x]
```

也存在用来解方程组的函数，例如下面就求得了一个 2 阶符号线性方程组的解。

```
In[11]:= LinearSolve[ $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ ,  $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ ]
```

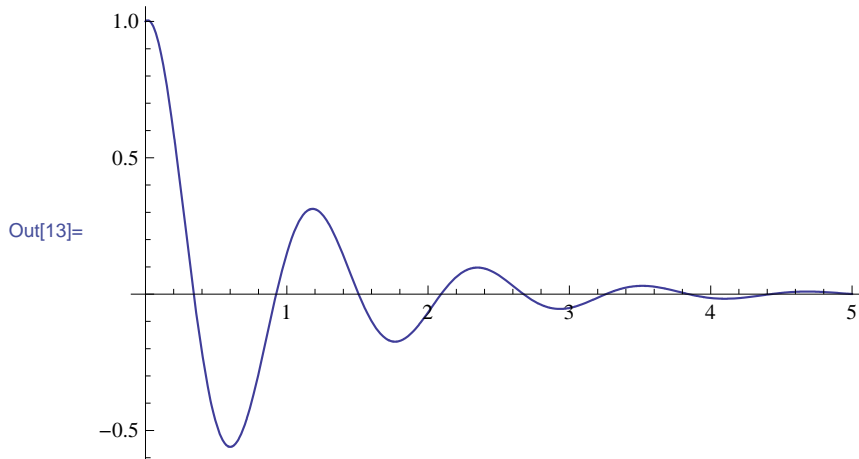
```
Out[11]=  $\left\{ \left\{ \frac{a_{22} x_1 - a_{12} x_2}{-a_{12} a_{21} + a_{11} a_{22}} \right\}, \left\{ \frac{a_{21} x_1 - a_{11} x_2}{a_{12} a_{21} - a_{11} a_{22}} \right\} \right\}$ 
```

你可以求解微分方程并绘图，比如一个表征线性阻尼摆的系统。

```
In[12]:= soln = DSolve[{y'[x] + 2 y'[x] + 30 y[x] == 0, y[0] == 1, y'[0] == 1/2}, y[x], x]
```

```
Out[12]=  $\left\{ \left\{ y[x] \rightarrow \frac{1}{58} e^{-x} (58 \cos[\sqrt{29} x] + 3 \sqrt{29} \sin[\sqrt{29} x]) \right\} \right\}$ 
```

```
In[13]:= Plot[y[x] /. soln, {x, 0, 5}, PlotRange → All]
```



你可以建立并处理分段函数。

```
In[14]:= sinc[x_] = Piecewise[{{1, x == 0}}, Sin[x] / x]
```

```
Out[14]=  $\begin{cases} 1 & x == 0 \\ \frac{\sin[x]}{x} & \text{True} \end{cases}$ 
```

```
In[15]:= Integrate[ $\frac{\text{sinc}[x^2]}{x}$ , x]
```

```
Out[15]=  $\frac{\text{CosIntegral}[x^2]}{2} - \frac{\sin[x^2]}{2 x^2}$ 
```

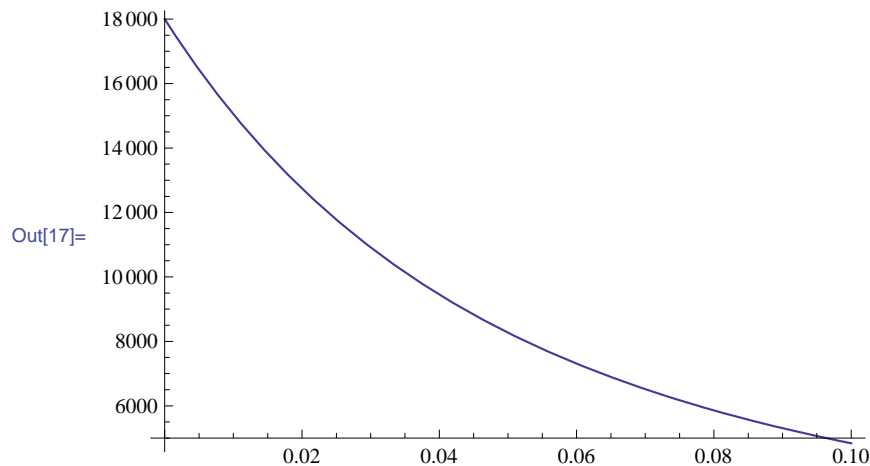
符号化的一个好处就是你可以迅速的看到当前的公式和算法。例如，这里算出了本金500美元期数36下的现值，其中实际利率用一个符号值表示。

```
In[16]:= presentValue = TimeValue[Annuity[500, 36], r, 0]
```


$$\text{Out[16]} = \frac{500 \left(-1 + (1 + r)^{36} \right)}{r (1 + r)^{36}}$$

图表清晰地显示了利率与年金的现值之间的关系。

```
In[17]:= Plot[presentValue, {r, 0.0, 0.10}]
```



事实上，符号表达式是非常通用的对象——可以像使用任意表达式一样使用它们。

```
In[18]:= Factor[ - 1]
```

$$\text{Out[18]} = \left(-1 + \begin{array}{c} \text{Leopard Print} \\ \text{Image} \end{array} \right) \left(1 + \begin{array}{c} \text{Leopard Print} \\ \text{Image} \end{array} + \begin{array}{c} \text{Leopard Print} \\ \text{Image} \end{array}^2 + \begin{array}{c} \text{Leopard Print} \\ \text{Image} \end{array}^3 + \begin{array}{c} \text{Leopard Print} \\ \text{Image} \end{array}^4 + \begin{array}{c} \text{Leopard Print} \\ \text{Image} \end{array}^5 + \begin{array}{c} \text{Leopard Print} \\ \text{Image} \end{array}^6 \right)$$

```
In[19]:= Rotate[Style["Mathematica", "Text"], 45 Degree]
```

```
Out[19]=
```

Mathematica

图形与可视化

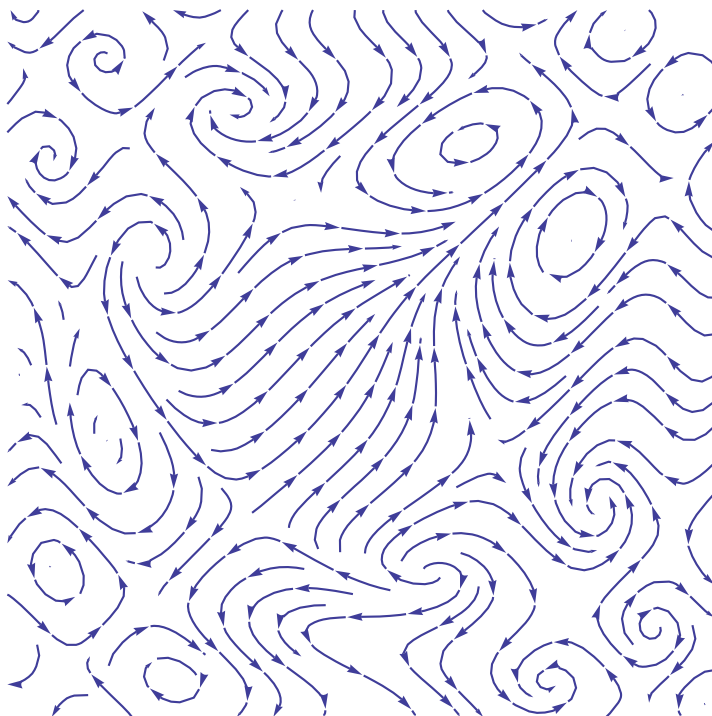
对函数或数据集的可视化通常能提供对其结构和性质更深入的洞悉。*Mathematica* 有着广泛的可视化能力，包括函数及数据集的2维和3维绘图，二元函数的等高图和密度图，数据的柱状图、直方图以

及其他各种图表和其他专业领域的函数，统计分析、金融分析、小波等等。此外，就如我们在第10章所要介绍的，通过 *Mathematica* 的编程语言，可以利用一些基本元素“自底而上”地构建图形。

这是矢量场 $\{\cos(1 - x + y^2), \sin(1 + x^2 - y)\}$ 的流线图。

```
In[20]:= strm = StreamPlot[{Cos[-1 - x + y^2], Sin[1 + x^2 - y]}, {x, -3, 3},
  {y, -3, 3}, Frame -> None]
```

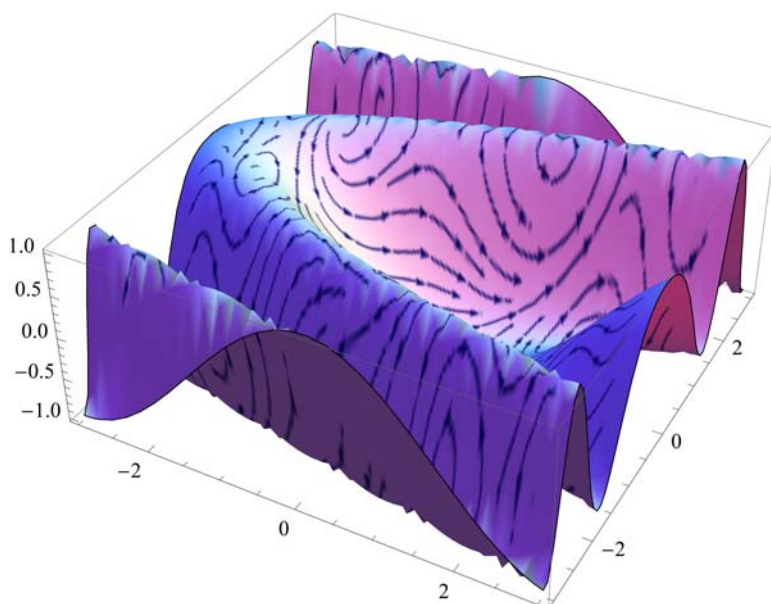
Out[20]=



这个图可以当成一个符号表达式在其他表达式中使用，比如一个表面上的纹理。

```
In[21]:= Plot3D[Sin[-1 - x + y^2], {x, -3, 3}, {y, -3, 3}, PlotStyle -> Texture[strm],
  Mesh -> None]
```

Out[21]=



当然，你通常遇到的是待分析和可视化的离散数据。这里我们导入一些同位素数据，并画出所有稳定同位素的原子量和结合能之间的关系。

```

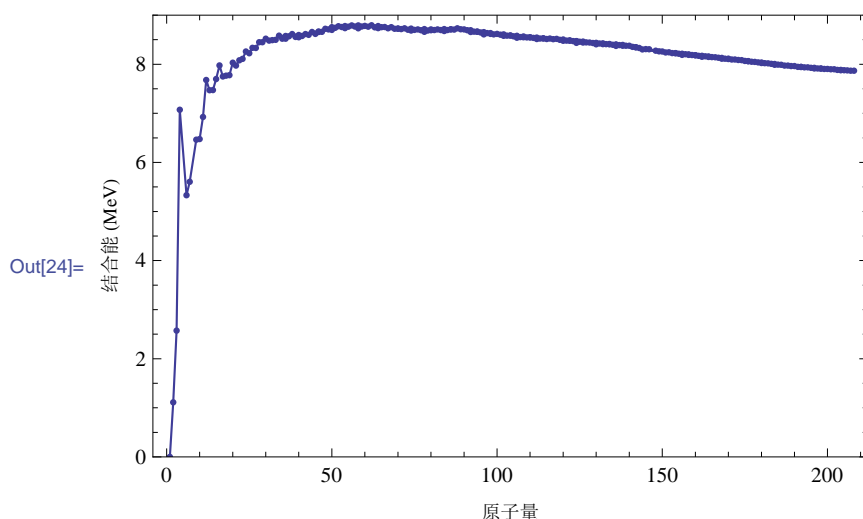
In[22]:= data = Outer[IsotopeData[#1, #2] &, IsotopeData["Stable"],
  {"MassNumber", "BindingEnergy", "Symbol"}];

In[23]:= Take[data, 8]

Out[23]= {{1, 0., 1H}, {2, 1.112283, 2H},
  {3, 2.572681, 3He}, {4, 7.073915, 4He}, {6, 5.332345, 6Li},
  {7, 5.606291, 7Li}, {9, 6.462758, 9Be}, {10, 6.475071, 10B}}

In[24]:= ListLinePlot[data[[All, {1, 2}]], Mesh → All, PlotRange → {0, 9},
  Frame → True, FrameLabel → {Style["原子量", 9], Style["结合能 (MeV)", 9]}]

```



从很多可用资源，采集员、数据库、或在线资源，获得的数据可以直接使用。例如，这里导入了一组由氨基酸残基分组的人类蛋白质上的原子位置信息。

```

In[25]:= positions = ProteinData["PAH", "AtomPositions", "Residue"];
Take[positions[[All, 2]], 8]

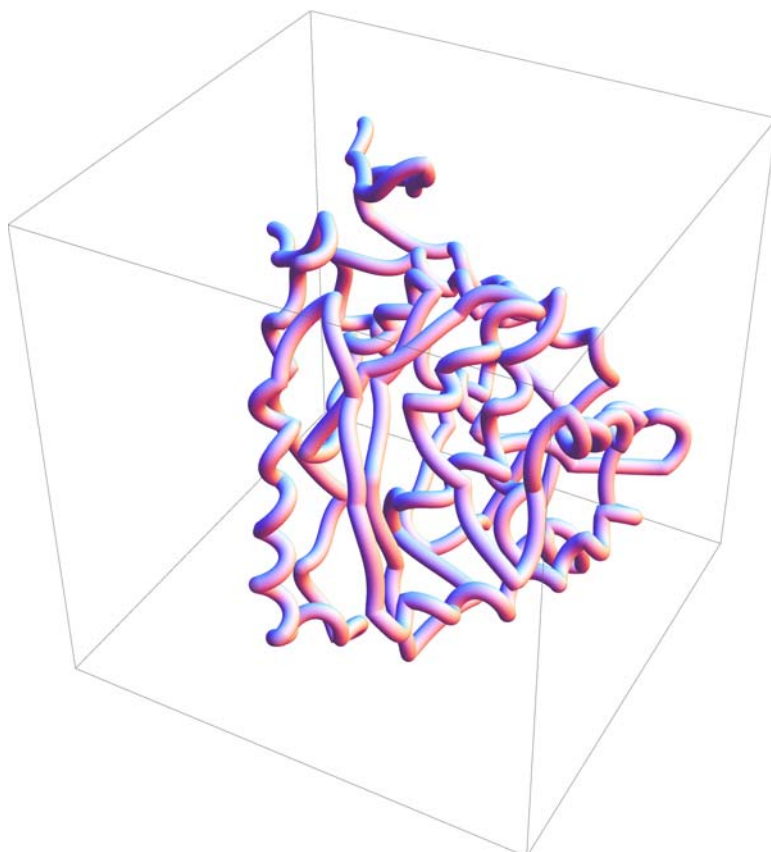
Out[26]= {{-2540.6, 3683.2, 1606.4}, {-2198.3, 3551.7, 1698.},
  {-2103.1, 3212.1, 1554.4}, {-2017.6, 2937.4, 1804.2},
  {-1649.6, 2912.8, 1901.5}, {-1451.6, 2689.9, 2141.6},
  {-1397.1, 2827.3, 2492.5}, {-1198.3, 2531.6, 2626.9}}

```

通过一条贯穿数据点的贝塞尔曲管，这些数据能用来可视化蛋白质骨架的构造。

```
In[27]:= Graphics3D[Tube[BezierCurve[positions[[All, 2]]], 80]]
```

```
Out[27]=
```



数据处理

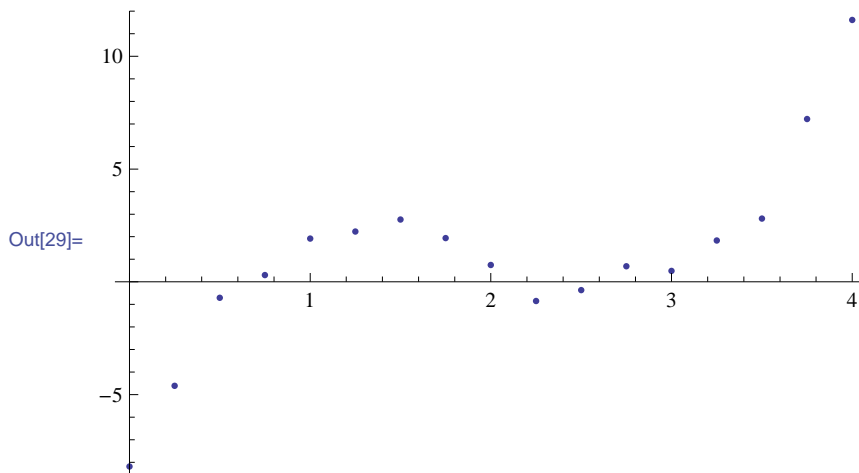
处理多类型数据的典型工作流程包括：导入、整理/过滤、分析、可视化、输出结果。数据本身能以各种不同的形式获得：表格化/数值化的数据、图像、声音文件、影片、HTML页面等等。一旦数据进入 *Mathematica*，各种统计和可视化工具就可以进行分析和可视化。例如，从一个电子表格导入示例数据。

```
In[28]:= data = Import["sampledata.xlsx", {"Data", 1}]
```

```
Out[28]= {{0., -8.18672}, {0.25, -4.6057}, {0.5, -0.709252},
           {0.75, 0.300171}, {1., 1.91848}, {1.25, 2.2322}, {1.5, 2.7596},
           {1.75, 1.94169}, {2., 0.748574}, {2.25, -0.852022},
           {2.5, -0.368416}, {2.75, 0.690119}, {3., 0.488073},
           {3.25, 1.83513}, {3.5, 2.80307}, {3.75, 7.2199}, {4., 11.6129}}
```

原数据的散点图能给我们关于变化趋势初步印象。

```
In[29]:= ListPlot[data]
```



用基函数 x , x^2 , 以及 x^3 对数据进行线性拟合。

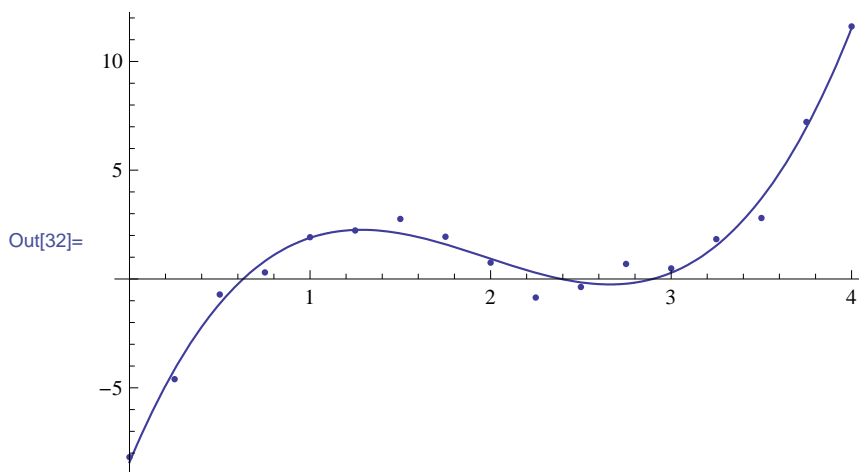
```
In[30]:= model = LinearModelFit[data, {x, x^2, x^3}, x];
```

```
In[31]:= model["BestFit"]
```

```
Out[31]= -8.42456 + 19.815 x - 11.4307 x^2 + 1.93117 x^3
```

将拟合模型与原数据一起显示。

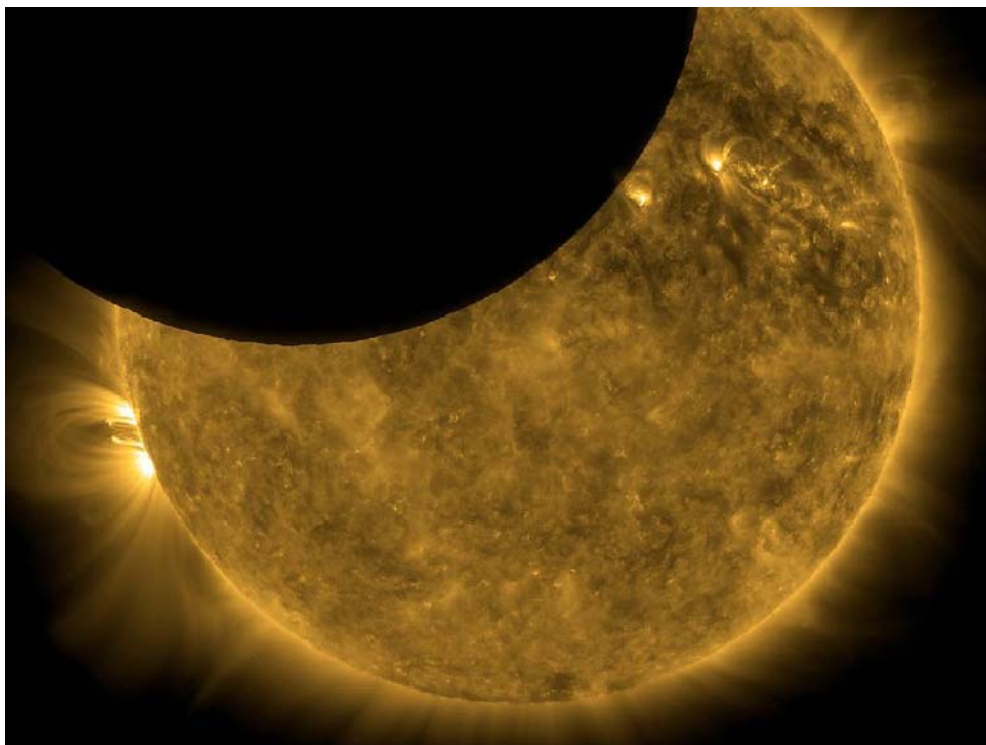
```
In[32]:= Show[Plot[model[x], {x, 0, 4}, PlotRange -> All], ListPlot[data]]
```



数据也可以直接从互联网导入，我们从NASA的网站导入一张图片然后用内置图像处理工具进行处理。

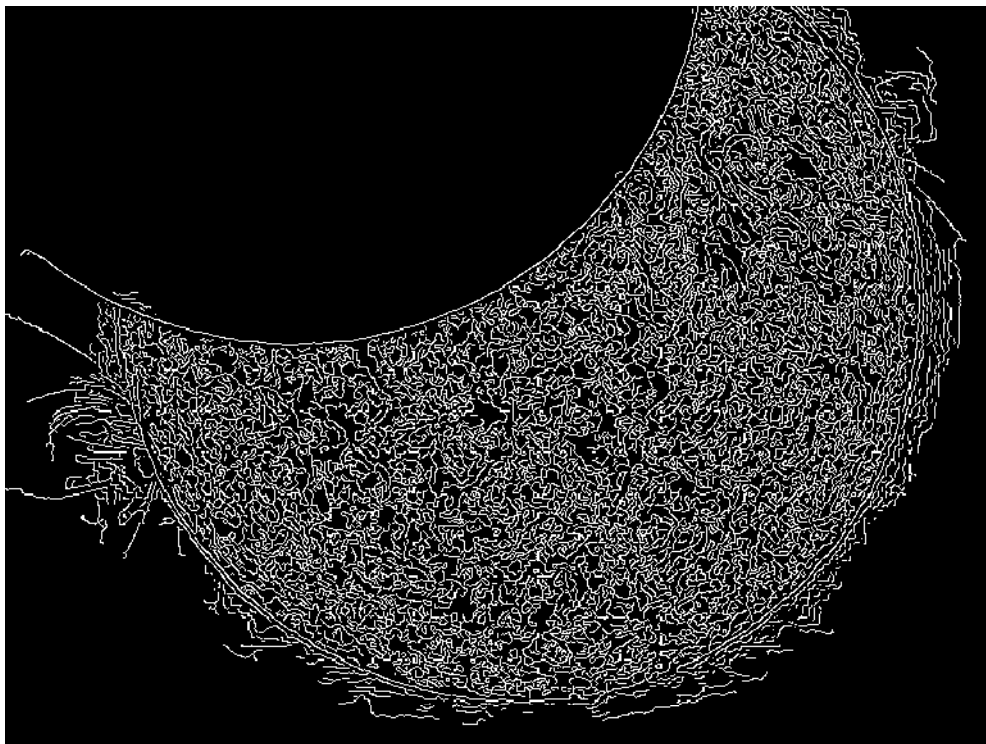

```
In[33]:= sun =
  Import[
    "http://img.ph.126.net/-qT2UDWe6QcLpGTKmOUXfw==/592786300969464095.
    jpg"] (*这里用了国内的一个链接，速度应该会快一点*)
```

Out[33]=



```
In[34]:= EdgeDetect[sun]
```

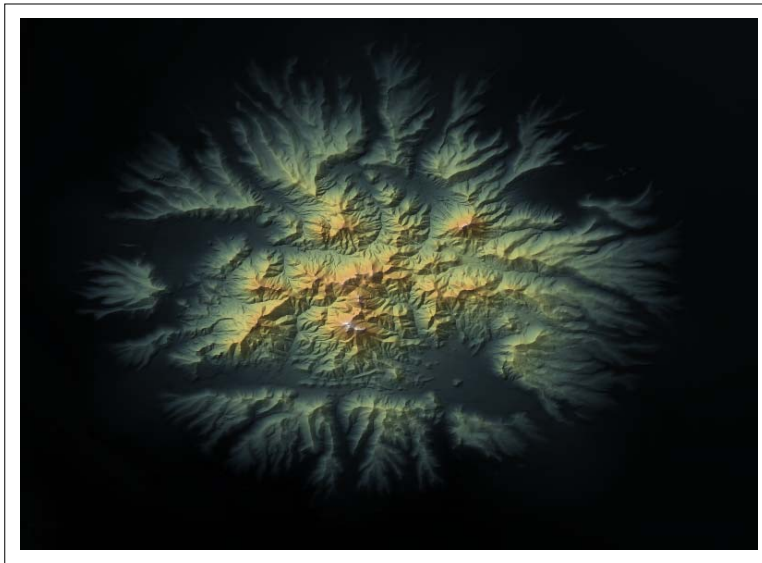
Out[34]=



下面，利用USGS国家高程数据集存档中的三维数字高程数据重建一个曲面。


```
In[35]:= Import["NED_40638016.zip"]
```

```
Out[35]=
```



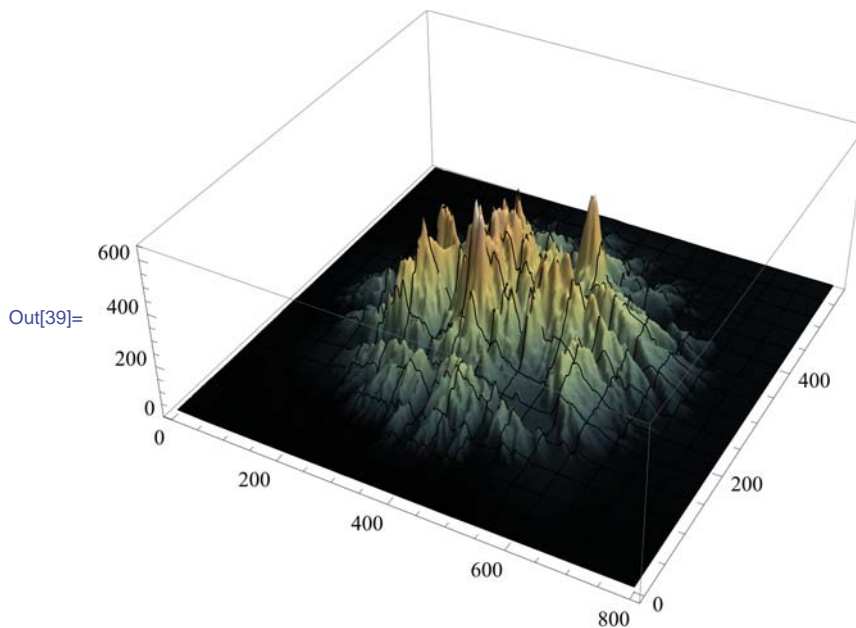
```
In[36]:= Import["NED_40638016.zip", "CoordinateSystemInformation"]
```

```
Out[36]= GEOGCS → {NAD83, DATUM → {North_American_Datum_1983,
      SPHEROID → {GRS 1980, 6 378 137, 298.257, AUTHORITY → {EPSG, 7019}},
      TOWGS84 → {0, 0, 0, 0, 0, 0, 0}, AUTHORITY → {EPSG, 6269}},
      PRIMEM → {Greenwich, 0, AUTHORITY → {EPSG, 8901}},
      UNIT → {degree, 0.0174533, AUTHORITY → {EPSG, 9108}},
      AXIS → {Lat, NORTH}, AXIS → {Long, EAST}, AUTHORITY → {EPSG, 4269}}
```

```
In[37]:= elevations = Import["NED_40638016.zip", {"ARCGrid", "Data"}];
Dimensions[elevations]
```

```
Out[38]= {575, 799}
```

```
In[39]:= ListPlot3D[elevations, MaxPlotPoints → 300, ColorFunction → "Topographic",
      PlotRange → All]
```

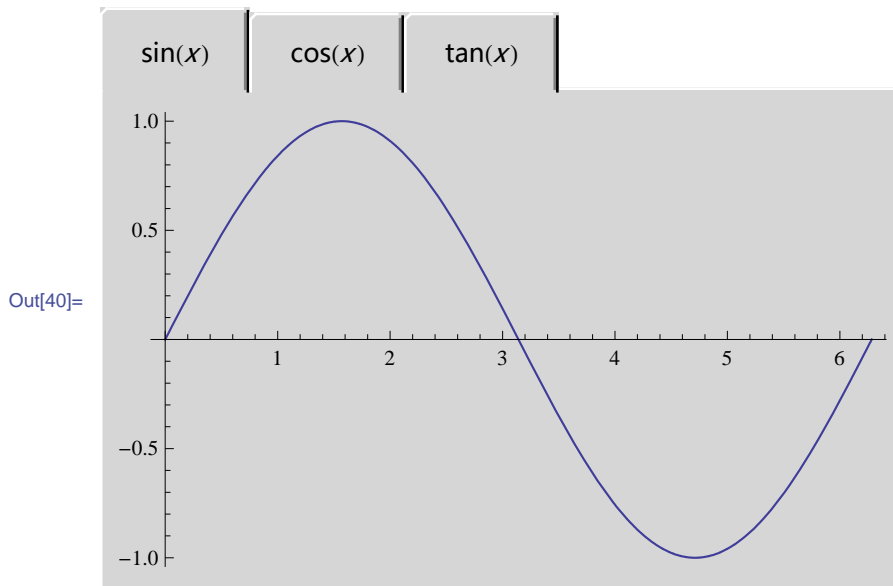


动态交互

除了上面提到的计算工具，*Mathematica* 也包含构建动态交互界面的工具，以关联当前处理的表达式。本节将通过几个小例子来展示种种可能性，等到11章我们再系统地介绍怎么编程。

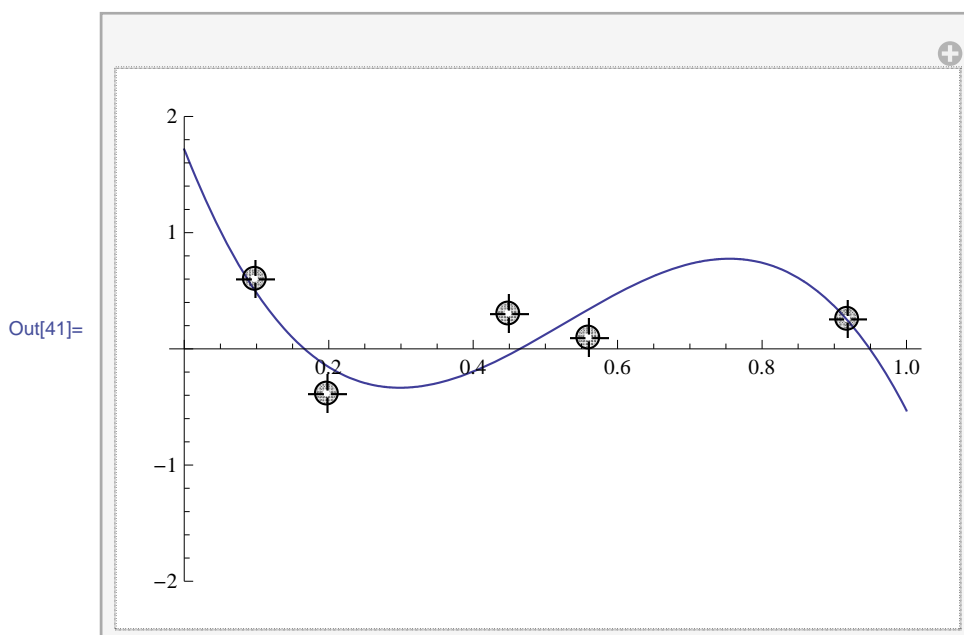
有若干个函数能用来构建界面，利用它你可以通过各种控件动态地操作参数，这些控件包括滑块、选项卡、选框、下拉菜单，以及其他鼠标驱动界面。

```
In[40]:= TabView[
  Table[TraditionalForm[f[x]] → Plot[f[x], {x, 0, 2 π}],
    {f, {Sin, Cos, Tan}}]]
```



你可以通过动态 `Locator` 对象直接与图表交互。在下面的例子中，用鼠标移动这些点将使拟合模型及其图像动态地更新。

```
In[41]:= Manipulate[
  model = LinearModelFit[pts, {x, x^2, x^3}, {x}];
  Plot[model[x], {x, 0, 1}, PlotRange → 2],
  {{pts, {{.1, .6}, {.2, -.4}, {.45, 0.3},
    {0.56, 0.1}, {0.92, .25}}}, Locator}]
```



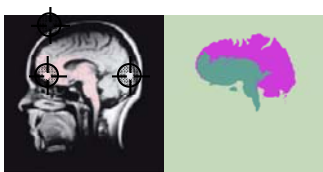
下面的小例子利用动态元素手动选择分割起始区域，完成了一个分割工作。

```
In[42]:= DynamicModule[{pts = {{81, 186}, {238, 188}, {89, 281}}},
  LocatorPane[Dynamic[pts],
    Row[
      {head,
        Dynamic[
          Image[Colorize[ImageForestingComponents[head, pts, 5]]]]],
    Initialization -> {head =
```



```
};}]
```

Out[42]=



编程

Mathematica 有3000多个内置函数，看起来你想要的任何东西都有一个现成的函数在那里。这种感觉是错的。现实中存在着比单个程序更多种类型的计算。不管你是想进行键渗流仿真还是找出环面上随机游走的均方距离，*Mathematica* 没有完成你想干的所有事的内置函数。它有的——且真正让它成为如此惊人的有用工具的是——让你定义自己的函数并像内置函数一样使用它们的能力。这被称为“编程”，也是这本书关注的全部。

有时，你所编的程序会很简短且着眼于非常特定的任务。*Mathematica* 拥有非常丰富的工具集以确保你可以快速且自然地将问题的陈述翻译为程序。例如，下面的程序定义了对完全数的测试，即真因子之和等于本身的自然数。

```
In[43]:= PerfectQ[n_] := DivisorSigma[1, n] == 2 n
```

定义第二个函数，它能从一定范围内的整数中找出所有通过 PerfectQ 测试的数。

```
In[44]:= PerfectSearch[n_] := Select[Range[n], PerfectQ]
```

找出所有小于1000 000的完全数。

```
In[45]:= PerfectSearch[106]
```

```
Out[45]= {6, 28, 496, 8128}
```

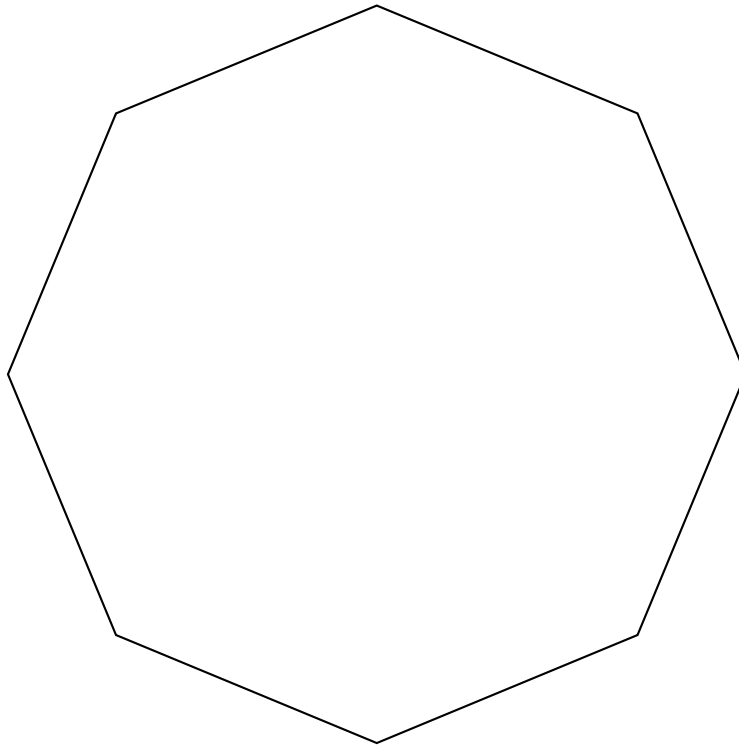
有时你需要创建一类新的对象然后像内置表达式一样操纵它们。比如，下面我们创造了一种和内置图形对象很类似的新图形对象。一个辅助函数定义了正n多边形的顶点，第二个函数，RegularPolygon，生成和内置的Circle、Line、Polygon对象一样显示的普通多边形对象。

```
In[46]:= vertices[n_] := Table[{Cos[2 π α / n], Sin[2 π α / n]}, {α, 0, n}]
```

```
In[47]:= RegularPolygon /: Graphics[RegularPolygon[n_]] :=
  Graphics[Line[vertices[n]], AspectRatio -> Automatic]
```

```
In[48]:= Graphics[RegularPolygon[8]]
```

```
Out[48]=
```



当然，你手中的任务迟早会要求更复杂的程序，代码量将从数行延展至数页。更复杂的程序，特别是那些目标用户是别人的，通常有可选参数、输入错误参数时的警告信息，用法信息等等特征。例如，这里有个包含上面所说特征的程序，能生成一个表示键渗流问题的网格。

```
In[49]:= Options[BondPercolation] = Options[Graph];
```

```
In[50]:= BondPercolation::baddim =
    "参数`1`和`2`表示网格的维度，为正整数。";
```

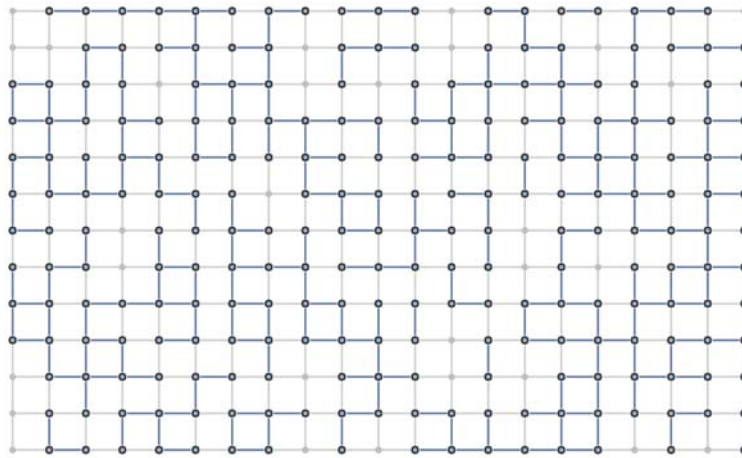
```
In[51]:= BondPercolation::usage =
    "BondPercolation[{m,n},prob] 模拟一个m x n正方形网格上的键渗流，";
```

```
In[52]:= BondPercolation[{m_, n_}, prob_, opts:OptionsPattern[]] :=
    Module[{gr}, If[!(IntegerQ[m] && IntegerQ[n]),
        Message[BondPercolation::baddims, m, n],
        gr = GridGraph[{m, n}]; Graph[Pick[EdgeList[gr],
            RandomVariate[BernoulliDistribution[prob]],
            EdgeCount[gr]], 1], opts]]]
```

下面模拟了一个13乘21的网格，假设任意两个顶点之间有键的概率为47%。

```
In[53]:= gr = BondPercolation[{13, 21}, 0.47];
HighlightGraph[GridGraph[{13, 21}],
  gr, GraphHighlightStyle -> "DehighlightGray"]
```

Out[54]=



让渗透程序返回一个Graph对象的好处让你能利用所有对于“图”的样式和计算的内置函数，例如计算连通分量；或者看看是否存在访问所有顶点仅一次的封闭路径；或者寻找从一条边到另一条边的路径。

```
In[55]:= Map[Length, ConnectedComponents[gr]]
```

```
Out[55]= {87, 32, 25, 24, 24, 8, 6, 5, 4, 4, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2}
```

```
In[56]:= HamiltonianGraphQ[gr]
```

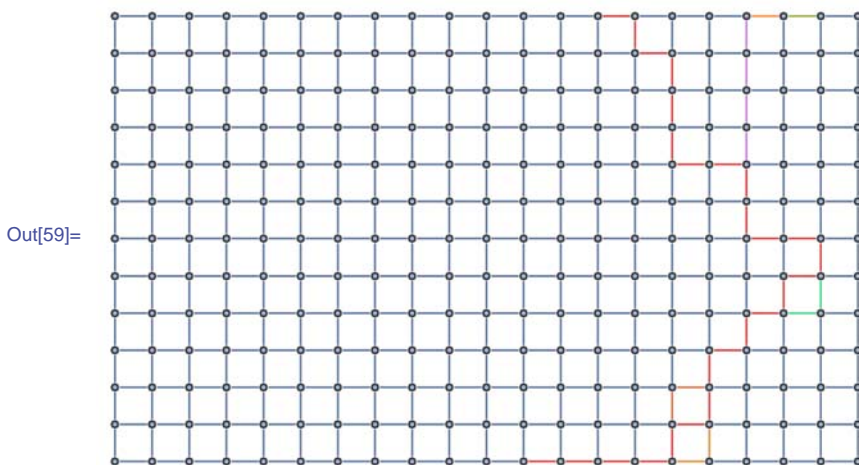
```
Out[56]= False
```

```
In[57]:= FindPercolationPath[gr_, dims : {dimx_, dimy_}] :=
Module[{vert, bot, top, spFun},
  vert = VertexList[GridGraph[dims]];
  bot = Select[vert, Mod[#, dimx] == 1 &];
  top = Select[vert, Mod[#, dimx] == 0 &];
  spFun = FindShortestPath[gr, All, All];
  Cases[Outer[spFun, bot, top],
    lis_List /; Length[lis] != 0, {2}]]
```

```

In[58]:= path = FindPercolationPath[gr, {13, 21}];
HighlightGraph[GridGraph[{13, 21}],
  Apply[UndirectedEdge, Map[Partition[#, 2, 1] &, path], {2}],
  GraphHighlightStyle -> "Thick"]

```



这些例子使用了多种编程风格和结构：函数式编程，基于规则的编程，纯函数等等。目前我们不指望你能理解本节的这几个编程例子——整本书讲的就是这个！你需要理解的是，从各方面来说，*Mathematica* 的设计旨在尽可能广泛的用途，同时，很多的计算在 *Mathematica* 里没有对应的内置函数，所以，为了完全利用它的能力，你有时需要编程。这本书的主要目的就是告诉你如何去编程。

另外一个目的是教你编程的基本理念。这些理念——赋值、定义规则、使用条件、递归、循环——对其他语言来说同样适用（当然在细节上很不同）。

1.2 新手上路

在真正使用 *Mathematica* 之前，你应当知道如何启动一次 *Mathematica* 会话，如何结束它，以及把事情搞砸之后该怎么办。本节介绍关于启动 *Mathematica*，使用笔记本界面，命令的基本语法，以及新手会感兴趣的一些话题。

启动 *Mathematica*

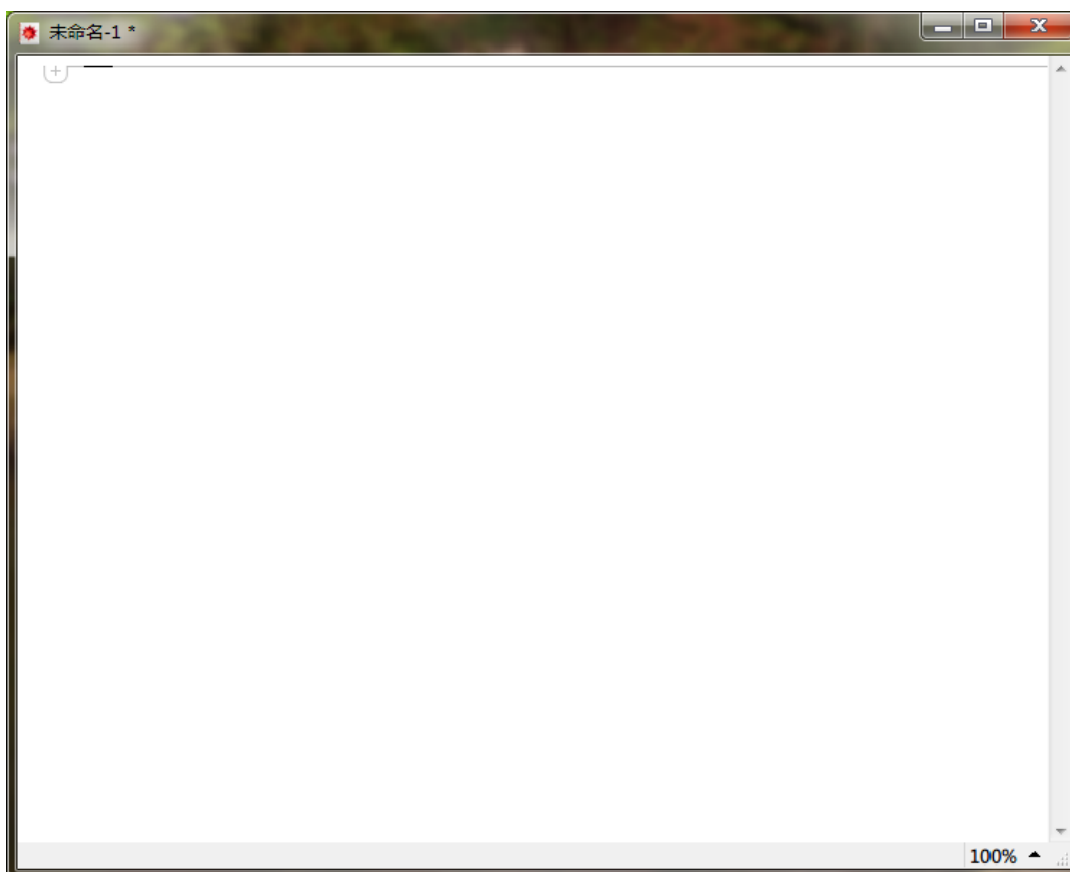
如何启动 *Mathematica* 在一定程度上取决于你的平台。

- Windows: 打开开始菜单选择**程序**，**Wolfram Mathematica**，**Mathematica X**（这里的X表示当前的版本，本书出版时是 *Mathematica* 8）。
- Macintosh OS X: 双击安装文件夹里的 *Mathematica* 图标，典型的位置是**Applications**文件夹。



- Linux/Unix: 在shell里输入**mathematica**，按Enter键。

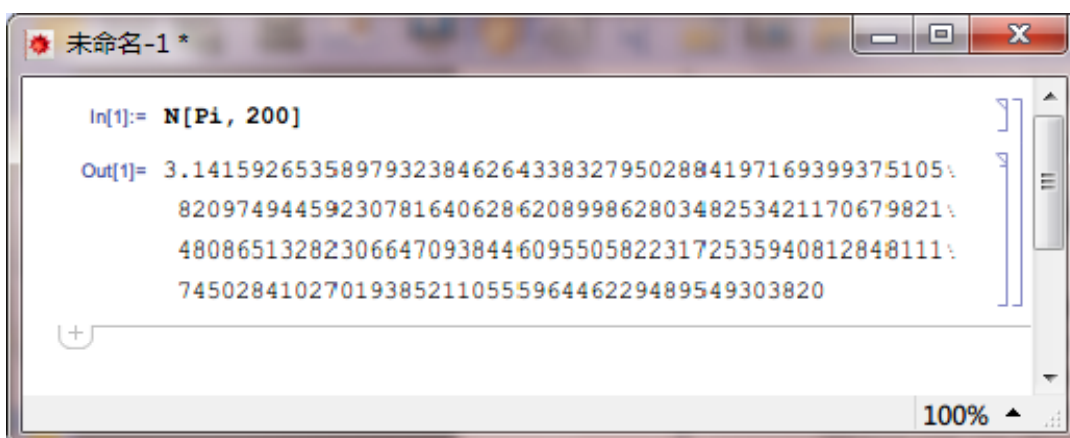
计算机将 *Mathematica* 部分载入内存，很快一个空白的窗口将会出现在屏幕上。这个窗口称为“笔记本”，它是 *Mathematica* 的图形界面。



笔记本界面

你在 *Mathematica* 里的所有工作主要是在被称为“笔记本”的东西里完成的。笔记本界面有很多和文字处理程序类似的工具——菜单、工具栏、面板——同时也包含一些 *Mathematica* 中特定的工具，用来撰写文档、输入和格式化数学公式、创建和编辑图形、匹配代码中的括号。此外，笔记本也提供了大纲和制作幻灯片的功能，这在演讲和演示时可能有用。当然，笔记本也是你进行计算，撰写和运行程序，创建图形，导入数据和文件的环境。

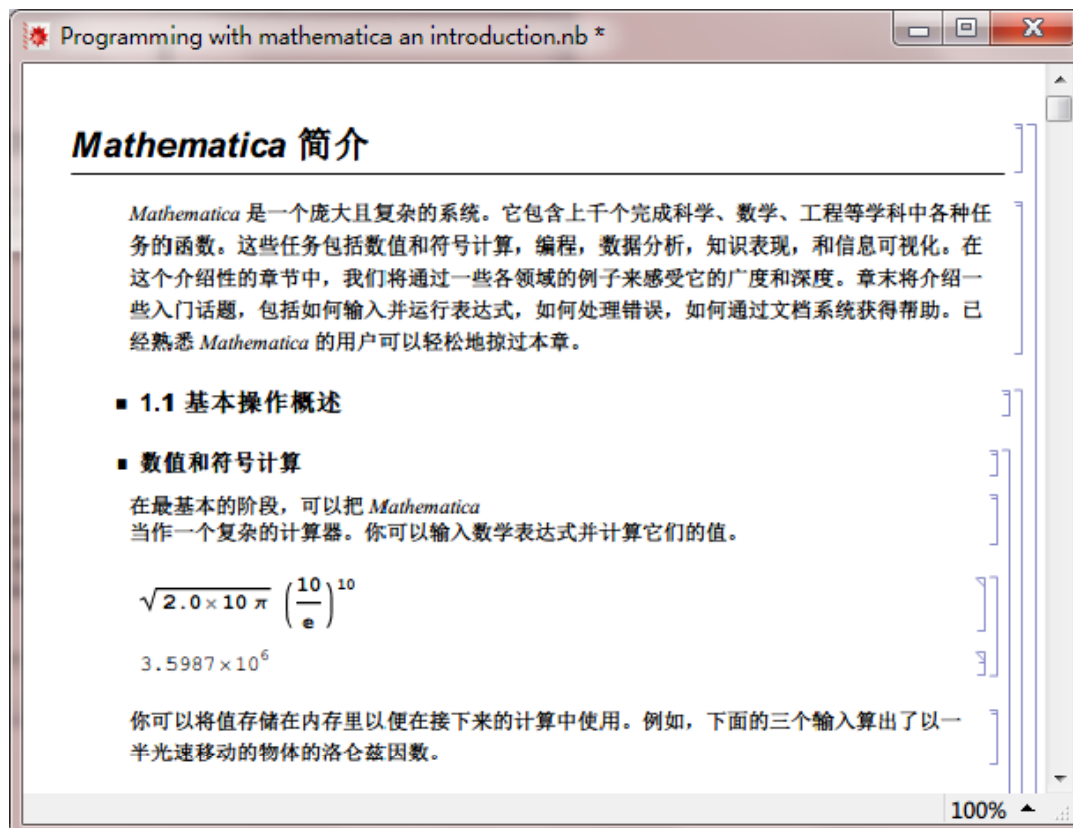
当一个空白笔记本出现在屏幕上，不管是来自刚启动好时的那个还是**文件**菜单里的**新建**命令，你就可以立即开始输入了。例如，输入 `N[Pi, 200]` 之后按 `[SHIFT][ENTER]`（按着 `[SHIFT]` 的同时按 `[ENTER]` 键）运行表达式。 *Mathematica* 将计算并在屏幕上打印 π 的200位小数近似值。



注意当你在笔记本中计算一个表达式的时候，*Mathematica* 会自动添加输入和输出提示符。也就是上面例子中的 `In[1]:=` 和 `Out[1]=`。这些提示符可以看作 *Mathematica* 会话过程中的参考标记（标

签)。

当你开始输入, *Mathematica* 会在窗口的最右边放置一个框出你当前工作“单元”的方括号。这些“单元方括号”主要作管理之用。双击单元括号可以打开任何收起的单元, 或者关闭一组单元。如下所示, 双击包含“1.1 基本操作概述”的单元括号会打开 (或关闭) 单元以显示 (或隐藏) 其内容:



通过单元括号你可以有序地管理你的工作并为你的材料列出大纲。关于单元括号和其它界面特性的完整描述请查阅内置教程的“单元”(WMDC)这部分内容, WMDC指 *Mathematica* 的参考资料中心。

其他关于打印、保存和编辑笔记本的信息, 请参考教程“使用笔记本界面”(WMDC)。

输入表达式

每当出现与笔记本等宽的水平直线时, 就可以键入输入了。如果当前位置不是你想输入的地方, 上下移动光标至其变为水平短线后单击鼠标。一条横穿窗口的水平线将会出现, 你可以立即开始输入, 一个新的输入单元将会生成。

你可以逐字地输入本书中的例子。为了使 *Mathematica* 计算你输入的表达式, 按 `[SHIFT][ENTER]` 组合键, 也就是按住 Shift 键后按 Enter 键 (在 Mac OS X 中, 按 `[SHIFT][RET]`)。

借助面板上的模版或键盘快捷键, 你可以以传统的二维格式输入数学表达式。例如, 下面的表达式就可以用“数学助手”面板 (面板菜单下) 来输入, 也可以借助一系列的快捷键。关于输入数学表达式的细节, 查阅教程“输入二维表达式”(WMDC)。

$$\text{In}[1]:= \int \frac{1}{1-x^3} dx$$

$$\text{Out}[1]= \frac{\text{ArcTan}\left[\frac{1+2x}{\sqrt{3}}\right]}{\sqrt{3}} - \frac{1}{3} \text{Log}[1-x] + \frac{1}{6} \text{Log}[1+x+x^2]$$

如上所述, *Mathematica* 会为你添加 In 和 Out 提示符。你不必输入这些提示符。执行输入之后便会

看到它们。

上一个计算的结果用 % 引用。

```
In[2]:= 2100
Out[2]= 1 267 650 600 228 229 401 496 703 205 376
```

```
In[3]:= % + 1
Out[3]= 1 267 650 600 228 229 401 496 703 205 377
```

引用任意一个前面的计算结果可以用标签 Out[i]，或者等价形式%i。

```
In[4]:= Out[1]
Out[4]= 
$$\frac{\text{ArcTan}\left[\frac{1+2x}{\sqrt{3}}\right]}{\sqrt{3}} - \frac{1}{3} \text{Log}[1-x] + \frac{1}{6} \text{Log}[1+x+x^2]$$

```

```
In[5]:= %2
Out[5]= 1 267 650 600 228 229 401 496 703 205 376
```

数学表达式

你可以用几乎所有计算机语言通用的线性算数符语法输入数学表达式。

```
In[6]:= 39 / 13
Out[6]= 3
```

输入此表达式的传统形式只需键入 39， $\boxed{\text{CTRL}}\boxed{/}$ ，13。

```
In[7]:=  $\frac{39}{13}$ 
Out[7]= 3
```

向上的箭头 (^) 用于表示指数。

```
In[8]:= 25
Out[8]= 32
```

以更传统的方式输入，键入 2， $\boxed{\text{CTRL}}\boxed{H}\boxed{^}$ ，5。

```
In[9]:= 25
Out[9]= 32
```

就像数学中那样，乘法可以用两个乘数之间的空格表示。*Mathematica* 将会自动的在两个数之间显示传统的乘法符号，×。大部分计算机语言的传统表示法星号 (*) 也是可行的。

```
In[10]:= 2 × 5
Out[10]= 10
In[11]:= 2 * 5
Out[11]= 10
```

操作符的优先级和数学中一样。特别地，乘法和除法的优先级就比加和减高：3 + 4 × 5 等于23而不是35。

```
In[12]:= 3 + 4 × 5
Out[12]= 23
```

有多种方式输入排版表达式：上面说的快捷键方式，完全的函数式形式，或者通过面板菜单下的面

板。表 1.1 展示了一些最常用的排版表达式的键盘键入方式。为了轻松的输入本书中的表达式，一定要熟悉这些东西的输入。

表 1.1. 输入排版表达式

显示形式	完全（函数式）形式	快捷键
x^2	<code>Subscript[x,i]</code>	<code>x, [CTRL]+6, 2</code>
x_i	<code>Subscript[x,i]</code>	<code>x, [CTRL]+_, i</code>
$\frac{x}{y}$		<code>□</code>
\sqrt{x}	<code>SqrtBox[x]</code>	<code>[CTRL]+2, x</code>
$x \geq y$	<code>GreaterEqual[x,2]</code>	<code>x, [ESC], >=, [ESC], y</code>

函数的语法

内置的函数和数学书上写法一样，不过函数名以大写字母开头且其参数在一对方括号里。

```
In[13]:= Factor[x5 - 1]
```

```
Out[13]:= (-1 + x) (1 + x + x2 + x3 + x4)
```

就如上面的例子，几乎所有的内置函数都是以完整拼写出来的。特例是一些广为人知的缩写，比如 `D` 用作微分，`Sqrt` 表示平方根，`Log` 表示对数，`Det` 表示矩阵的行列式等等。完整拼出函数名的传统在你不知道是否存在某个函数完成特定任务时十分有用。例如，为了求一个复数的共轭，学富五车的你可能会这么猜：

```
In[14]:= Conjugate[3 + 4 i]
```

```
Out[14]:= 3 - 4 i
```

就像传统数学记法一样，多参数函数以逗号分隔其参数。例如，`RandomReal` 在一个参数的情况下可以生成一个 0 到 10 之间的随机数，然而两个参数的形式可以用来生成一个随机向量或矩阵。

```
In[15]:= RandomReal[10]
```

```
Out[15]:= 4.22089
```

```
In[16]:= RandomReal[10, 12]
```

```
Out[16]:= {2.15451, 4.39215, 5.46212, 2.32682, 7.39991, 5.8042,
1.27914, 4.4759, 1.37478, 8.32759, 1.09099, 3.85064}
```

列表

列表是 *Mathematica* 中基本的数据类型，用来表示向量或矩阵（或者任意阶张量），同时也作为 `Plot` 和 `Integrate` 等函数的附加参数。虽然 `[` 和 `]` 被用来表示函数的参数范围，花括号 `{` 和 `}` 用来表示“列表”或值的范围。

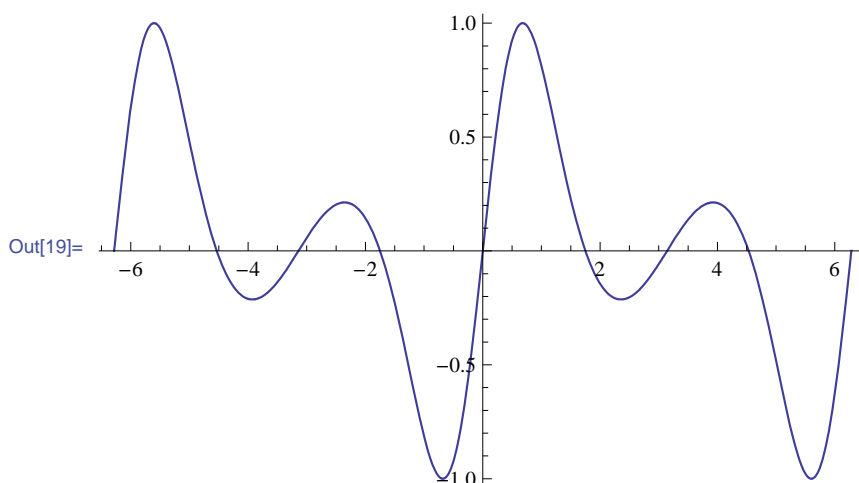
列表可用来表示矩阵，下面利用传统记法计算两个向量的点乘。

```
In[17]:= Clear[c];
{a, b, c} . {x, y, z}
```

```
Out[18]:= a x + b y + c z
```

列表也可作为很多内置函数的参数。

```
In[19]:= Plot[Sin[x + Sqrt[2] Sin[x]], {x, -2 Pi, 2 Pi}]
```



```
In[20]:= Integrate[Cos[x], {x, a, b}]
```

```
Out[20]= -Sin[a] + Sin[b]
```

```
In[21]:= RandomReal[{-100, 100}, {5, 5}]
```

```
Out[21]= {{5.81443, 75.0459, 96.8296, -63.503, 82.5413},
          {-21.844, -84.5292, -59.2765, 19.0311, -56.527},
          {-35.9254, 70.9676, -30.0071, 8.46947, 15.0549},
          {-49.8758, -68.4884, -20.7154, 92.1269, 69.8578},
          {5.57931, -23.1106, 24.6621, 44.9722, 7.54523}}
```

Plot的例子中，列表 $\{x, -2\pi, 2\pi\}$ 表示需要画出函数 $\sin(x + \sqrt{2} \sin(x))$ 在 x 从 -2π 到 2π 取值的图形。上面的 Integrate 表达式等价于积分 $\int_a^b \cos(x) dx$ 。最后一个 RandomReal 的例子中，第一个列表表示所选数字的范围，第二个列表指定维度，在这里是一个5阶方阵。

第3章我们会探索 *Mathematica* 的列表处理能力。

分号

如果你以一个分号 (;) 结束某个表达式，*Mathematica* 会计算其值而不显示它。这在表达式的结果十分长并且不需要查看时十分有用。下面的例子中，我们首先生成一个 1 到 10000 的整数列表，用分号抑制其输出；然后计算其总和以及平均数。

```
In[22]:= nums = Range[10 000];
```

```
In[23]:= Total[nums]
```

```
Out[23]= 50 005 000
```

```
In[24]:= 
$$\frac{\%}{\text{Length[nums]}}$$

```

```
Out[24]= 
$$\frac{10\,001}{2}$$

```

在笔记本界面里，你可以在一个输入单元里输入任意多个你想要的计算；在你按下 **SHIFT+ENTER** 后，*Mathematica* 会依次计算它们。

输入语法变体

书写表达式的方法有很多中。通常，你尽可以用传统的记法，比方说 `fun[x]`。但是你应当意识到还

有多种时常使用的变体（查看表 1.2）。

这是一个标准函数记法的例子，即一个参数情况下的 N 。

```
In[25]:= N[ $\pi$ ]
```

```
Out[25]= 3.14159
```

这是使用前缀操作符的形式。

```
In[26]:= N@ $\pi$ 
```

```
Out[26]= 3.14159
```

这是后缀操作符的形式。

```
In[27]:=  $\pi$  // N
```

```
Out[27]= 3.14159
```

对于两个参数的函数，你可以使用中缀操作符。下面的表达式等价于 $N[\pi, 30]$ 。

```
In[28]:=  $\pi \sim N \sim 30$ 
```

```
Out[28]= 3.14159265358979323846264338328
```

表 1.2 函数记法变体

```
In[29]:= Grid[{{{"记法", "输入"}, {"传统", "N@ $\pi$ "}, {"前缀", "N@ $\pi$ "}, {"后缀", " $\pi$  // N"}, {"中缀", " $\pi \sim N \sim 100$ "}}], Frame  $\rightarrow$  All, Spacings  $\rightarrow$  1.5]
```

Out[29]=

记法	输入
传统	N@ π
前缀	N@ π
后缀	π // N
中缀	$\pi \sim N \sim 100$

最后，输入和处理数学表达式时，许多人更喜欢传统的语法。比如，以标准 *Mathematica* 语法计算一个积分。

```
In[30]:= Integrate[1 / Sin[ $x$ ],  $x$ ]
```

```
Out[30]= -Log[Cos[ $\frac{x}{2}$ ]] + Log[Sin[ $\frac{x}{2}$ ]]
```

同样的积分可以用面板或快捷键以更传统的方式表示。

```
In[31]:=  $\int \frac{1}{\sin[x]} dx$ 
```

```
Out[31]= -Log[Cos[ $\frac{x}{2}$ ]] + Log[Sin[ $\frac{x}{2}$ ]]
```

很多的数学函数都有与其操作对应的传统符号，如果有的话，这可以用来代替完整拼写出的函数名。比如，两个集合的并集可以用 `Intersection` 函数输入。

```
In[32]:= Intersection[{a, b, c, d, e}, {b, f, a, z}]
```

```
Out[32]= {a, b}
```

或者你也可以用更传统的记法来进行同样的计算。

```
In[33]:= {a, b, c, d, e}  $\cap$  {b, f, a, z}
```

```
Out[33]= {a, b}
```

为了了解如何快速输入这些或者其他记号，不管是用面板还是直接用快捷键，参考教程“输入二维表达式”(WMDC)。

注释

利用一对(*和*)，注释可以包括在输入里，它是不会被计算的文本。注释是无效的，它会被 *Mathematica* 计算内核忽略。

```
In[34]:= D[Sin[x], {x, 1}] (*sin(x) 的一阶导数*)
Out[34]= Cos[x]
```

错误

人无完人。在 *Mathematica* 的使用和编程过程中，你将会遇到各种各样的错误，有些很明显，有些很微妙，有些挺容易改正。最常见的错误可能就是拼错函数名了。*Mathematica* 使用语法着色来帮你识别拼错的符号名。例如，下面的输入中，sin 被故意拼错。*Mathematica* 会将它不知道的符号变为蓝色。如果你执行这个输入，它会原样返回，因为 *Mathematica* 没有内置名为 Sine 的函数的规则。

```
In[35]:= Sine[30 Degree]
Out[35]= Sine[30 °]
```

对于函数 Sin 当然有对应的规则了。

```
In[36]:= Sin[30 Degree]
Out[36]=  $\frac{1}{2}$ 
```

你的原始表达式被原样返回是你将经常碰到的问题。除了拼错了函数名或者用了不存在的函数这两种情况，还有一种是参数的个数不对，特别是自定义的函数。比如，早先定义的 PerfectSearch 函数的只有一个；如果错误地给它两个参数的话，由于 *Mathematica* 没有函数 PerfectSearch 两个参数时的规则，所以会返回未计算的形式。

```
In[37]:= PerfectSearch[106, 4]
Out[37]= PerfectSearch[1 000 000, 4]
```

一些输入会触发真正的错误信息。上面的语法错误就是一个例子。内置函数被设计为在遇到类似的错误输入时给予警告。下面的第一个例子中，我们给函数 Det 输入了一个非方阵。第二个例子中，由于 FactorInteger 只能作用于整数，所以实数参数会触发错误条件。

```
In[38]:= Det[{{1, 2, 4}, {2, 4, 8}}]
```

Det::matsq : 位置 1 处的参数 {{1, 2, 4}, {2, 4, 8}} 不是一个非空方阵. >>

```
Out[38]= Det[{{1, 2, 4}, {2, 4, 8}}]
```

```
In[39]:= FactorInteger[34.2]
```

FactorInteger::exact : FactorInteger[34.2] 中的参数 34.2 不是一个精确数. >>

```
Out[39]= FactorInteger[34.2]
```

5.7节会介绍为自编程序建立和分配消息的基本框架。

逃离困境

虽然让 *Mathematica* 来告诉你哪里做错了是很方便的，但是你总是会遇到这样的情况，执行了一个导致 *Mathematica* 行为异常的输入，导致它静默无声长时间不返回结果，或者整页整页的打印毫无用处的信息。这时，应当尝试去“中断”计算。中断的方法取决于你的操作系统：

- Macintosh OS X: ⌘[.]

- Windows: `[ALT][.]`
- Linux/Unix: `[CTRL][.]`

终止计算的尝试有时可能会失败。如果在等了足够长的时间后，*Mathematica* 仍然没动静的话，你就得“杀掉内核”了。不过在杀掉内核之前要确保计算真的在死循环里，而不是一次耗时间的高强度计算。选择**计算**菜单里的**退出内核**选项可以杀掉内核。在不关闭前端的情况下可以通过**计算**菜单下的**启动内核**•**Local**命令重启内核。你也可以简单地执行一个命令，然后新的内核会自动启动。

前端和内核

你用的 *Mathematica* 其实由两个独立程序组成。这就是“前端”和“内核”。前端是用户界面。它由笔记本和菜单系统、面板（其实本质上就是笔记本）和其它任何能接受鼠标及键盘输入的部分组成。内核是执行计算的程序。所以一个典型的用户（你）与 *Mathematica* 之间的交互包括下面几步，每一步所涉及的程序在括号中指出：

- 在笔记本里输入（前端）；
- 键入 `[SHIFT][ENTER]` 将输入送至内核计算（前端）；
- 计算出结果并将之返回给前端（内核）；
- 在笔记本里格式化并显示结果（前端）；

还有一点我们没提到。因为内核和前端是两个独立的程序，一个在两个程序之间“通信”的手段就是必要的了。这个 *Mathematica* 自带的通信协议被成为 *MathLink*。它一直在背后工作，对用户完全透明。

MathLink 是非常通用的通信协议，它的应用不仅仅局限于前端和内核之间的通信，而且也能用于建立前端与你电脑里其他程序的通信，像是编译后的 C 或者 Fortran 代码。它也能用于建立内核与字处理或电子表格等等程序之间的连接。

事实上，*Mathematica* 拥有多个通信协议。比如你可以通过 *DatabaseLink* 与 SQL 数据库通信，通过 *J/Link* 与 Java 通信，通过 *.NET/Link* 与 .NET 通信。这些协议能让你扩展 *Mathematica* 至其他领域并在 *Mathematica* 的界面内使用它们。这些都超出了本书的范围，但是如果你感兴趣的话，参考资料中心有关于它们大量的文档，同时关于这些协议也有若干本书和文章（参见书末的参考书目）。

1.3 获得帮助

函数信息

Mathematica 包含大量的文档，有多种方式访问它们。它同时也被设计为可以让你自己为自己的程序和函数撰写文档，且获得它们的途径和内置函数完全一样。

如果我们知道某个函数的名字但是不知道它的语法或用法，最简单的方法是执行 `?function`。例如，这是 `Map` 的帮助信息。

`In[40]:= ? Map`

`Map[f, expr]` 或 `f /@ expr` 将 *f* 应用到 *expr* 中第一层的每个元素。
`Map[f, expr, levelspec]` 将 *f* 应用到 *levspec* 指定的 *expr* 的部分中. [>>](#)

同时如果不确定某个函数的名称的话，可以使用通配符找出所有包含特定字符的函数。例如，找出所有以“Random”开头的函数。

In[41]:= ? Random*

▼ System`

Random	Random-Complex	Random-Graph	Random-Integer	Random-Prime	Random-Sample	Random-Variate
Random-Choice	Random-Function	Random-Image	Random-Permutation	Random-Real	Random-Seed	Random-WalkProcess

点击其中一个链接会产生一个此函数的简短用法陈述。例如，如果单击链接 `RandomGraph` 笔记本里就会显示下面的内容。

`RandomGraph[{n, m}]` 给出具有 n 个顶点和 m 条边的伪随机图。

`RandomGraph[{n, m}, k]` 给出 k 个伪随机图组成的列表。

`RandomGraph[gdist, ...]` 来自随机图分布 $gdist$ 的样本。 >>

点击超链接 >> 将会把你直接带到参考资料中心，那里可以找到更为详尽的解释。

另一种获得帮助的方法是高亮任何一个 *Mathematica* 函数并按 F1 键（Macintosh OS X 上面是 `⌘+SHIFT+F`）

参考资料中心

Mathematica 有一个被称为参考资料中心的庞大参考内容集合。参考资料中心让你很容易地搜索函数，它提供了大量的文档，例子，以及相关链接。

打开参考文档只需选择帮助菜单下的**参考资料中心**。你将马上看到类似于下面的东西：



注意到这八个类别：核心语言、数学与算法、可视化与图形、数据操纵、动态交互、笔记本与文

档、系统界面与部署。点击任何一个类别将出现一个此领域的延展话题列表。

假如你在找关于三维参数化绘图的信息。首先点击“可视化与图形”类别，然后是“函数可视化”。此时的参考资料中心看起来应该是这样的：



点击链接 ParametricPlot3D 会把你带到此函数的参考页面。

ParametricPlot3D - Wolfram Mathematica

搜索 ParametricPlot3D

在所有页中寻找 ParametricPlot3D.

Mathematica > 可视化与图形 > 函数可视化 > ParametricPlot3D >
 Mathematica > 可视化与图形 > 数据可视化 > 函数可视化 > ParametricPlot3D >
 Mathematica > 数据处理 > 统计数据分析 > 统计可视化 > 函数可视化 > ParametricPlot3D >

ParametricPlot3D 8 的更新 显示变化

`ParametricPlot3D[{fx, fy, fz}, {u, umin, umax}]`
 产生参数 u 从 u_{min} 到 u_{max} 的三维空间曲线的参数化图形。

`ParametricPlot3D[{fx, fy, fz}, {u, umin, umax}, {v, vmin, vmax}]`
 产生参数 u 和 v 的三维空间曲线的参数化图形。

`ParametricPlot3D[{fx, fy, fz}, {gx, gy, gz} ...]`
 绘制几个对象。

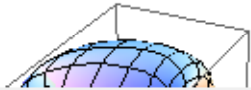
更多信息

范例

例 (5)

绘制一个参数曲面:

```
In[1]:= ParametricPlot3D[
  {Cos[u], Sin[u] + Cos[v], Sin[v]}, {u, 0, 2 π}, {v, -π, π}]
```



100%

另一种方法是输入 `?ParametricPlot3D` 然后点击用法信息最后的 `>>` 链接。

参考资料中心还有一些其它的特性, 包括关于每个函数用法的大量示例, 应用以及相关函数。