第 5 章　数组计算和曲线绘图

用列表对象存储序列或表格等数据非常便捷。数组和列表非常相似，与列表相比，数组虽然牺牲了灵活性，但具有更高的效率。当用计算机进行数学计算时，常常遇到大量的数字以及相关联的数学运算。在这种情况下将数字用列表存储将会导致程序运行缓慢，而用数组存储则能使程序运行得更快，这对于工业和科学领域动辄运行数小时、数天乃至数周的大量高级数学应用非常重要。因此，任何能减少执行时间的巧妙想法都是至关重要的。

然而，人们可能会说，数学软件的程序员一直过分注重效率和巧妙的程序构想。这使得最终的软件常常难以维护和扩展。本书主张专注于清晰、精心设计、易于理解且功能正确的程序，速度优化则放到之后再考虑。幸运的是，数组对于代码的清晰、正确性和速度都有帮助。

本章主要介绍数组，包括如何创建数组以及数组的用途。数组计算通常得到许多数字，光看数字本身可能很难理解这些数字的意思。因为人类是一种视觉动物，所以一个理解数字的好方法就是可视化。本章介绍单变量函数的可视化曲线，即，$y=f(x)$形式的曲线。曲线的同义词是图形，屏幕上的曲线图像通常称为图(plot)。本章使用数组来存储曲线上点的信息。简言之，数组计算需要可视化，可视化需要数组。

本章中的所有示例程序均可在文件夹 src/plot[①] 中找到。

5.1　向量

本节简要介绍向量的概念。本节假设读者对平面向量和空间向量有所了解。当后面开始接触数组和曲线绘图时，这个背景知识将非常有帮助。

5.1.1　向量的概念

有些数学量与一组数字相关联。例如，数学上用两个坐标(实数)表示平面上的一个点，这两个坐标命名为 x 和 y，通常用(x,y)表示，即用括号将数字归为一组。除了符号，也可以直接使用数字：(0,0)和(1.5，−2.35)也是平面坐标的例子。

三维空间中的点有 3 个坐标，可以命名为 x_1、x_2 和 x_3，通常也用括号将它们归为一组：(x_1,x_2,x_3)；另一种方式是使用符号 x、y、z，并把点记作(x,y,z)。也可以直接用数字代替其中的符号。

你可能还记得高中的时候求解含有两个未知数的方程组。在大学，你会很快碰到用含有 n 个未知数的 n 个方程表示的问题。这类问题的解包含 n 个数字，用括号将这些数字从编号 1 到编号 n 归为一组：$(x_1,x_2,\cdots,x_n)$。

像(x,y)、(x,y,z)和$(x_1,x_2,\cdots,x_n)$这样的量在数学中被称为向量。向量的图形表示是从原点到另一个点的箭头。例如，向量(x,y)表示为平面上从原点(0,0)到点(x,y)的箭头。同样地，(x,y,z)表示为三维空间中从原点(0,0,0)到点(x,y,z)的箭头。

---

① http://tinyurl.com/pwyasaa/plot。

数学家发现，引入高于三维的空间也很方便。当把 n 个方程的解封装成一个向量$(x_1, x_2, \cdots, x_n)$时，可以把它想象成 n 维空间中的一个点或一个 n 维空间中从原点(0,0,…,0)到$(x_1, x_2, \cdots, x_n)$的箭头。图 5.1 以箭头的方式表示向量，可以从原点出发，也可以从任何其他点出发。两个大小和方向都相同的箭头(向量)在数学上是相等的。

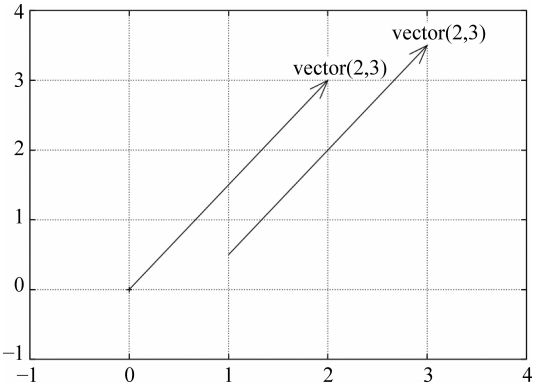

图 5.1　表示向量(2,3)的标准方式为从原点到点(2,3)的箭头，或按照数学等价的方式，从$\left(1, \frac{1}{2}\right)$(或任意点(a,b))到$\left(3, 3\frac{1}{2}\right)$(或(a+2,b+3))的箭头

$(x_1, x_2, \cdots, x_n)$被称为一个 n 维向量或有 n 个分量的向量，其中的每一个数字 $x_1, x_2, \cdots, x_n$ 就是一个分量或元素，分别称第 1 分量(元素)、第 2 分量(元素)……第 n 分量。

Python 程序可以用列表或元组表示向量：

```
v1 = [x, y]          #list of variables
v2 = (-1, 2)         #tuple of numbers
v3 = (x1, x2, x3)    #tuple of variables
from math import exp
v4 = [exp(-i * 0.1) for i in range(150)]
```

在这个程序中，v1 和 v2 是 2 维向量，v3 是三维向量，而 v4 是 150 维向量(包含指数函数的 150 个值)。由于 Python 列表和元组将 0 作为第一个下标，因此，在数学上也可以将向量(x_1, x_2)写成(x_0, x_1)。这在数学中并不常见，但使得问题的数学描述更贴近于它在 Python 中的形式。

图形无法表示出 150 维空间的样子。从二维空间到三维空间可以大概感觉到“添加一个维度”是什么意思。但是如果不讨论对空间的视觉感知，这件事在数学上就非常简单了：从 4 维向量到 5 维向量就像往符号或数字列表中加入一个元素那样简单。

5.1.2　向量的数学运算

由于向量可以被视为具有长度和方向的箭头，因此向量在几何和物理学中非常有用。汽车的速度、加速度具有大小和方向，汽车中一个点的位置也是一个向量。三角形的边可以被视为具有方向和长度的线(箭头)。

向量用于几何和物理学应用时，其数学运算很重要。下面将举例说明向量的一些最重要的运算。本节的目标不是讲解如何用向量进行计算，而是为了说明这样的计算是如何根据数学规则定义的。给定两个向量(u_1, u_2)和(v_1, v_2)，可以按以下规则将它们相加：

$$(u_1, u_2) + (v_1, v_2) = (u_1 + v_1, u_2 + v_2) \tag{5.1}$$

也可以使用类似的规则将两个向量相减：

$$(u_1,u_2)-(v_1,v_2)=(u_1-v_1,u_2-v_2) \tag{5.2}$$

向量可以乘以一个数字。下面的数字 a 通常指标量：

$$a\cdot(v_1,v_2)=(a\cdot v_1,a\cdot v_2) \tag{5.3}$$

两个向量的内积（也叫点乘或标量积）结果是一个数：

$$(u_1,u_2)\cdot(v_1,v_2)=(u_1v_1,u_2v_2) \tag{5.4}$$

在高中数学和物理学课程中，内积或点积也可以表示为两个向量长度的积再乘以它们之间夹角的余弦值，但本节不会用到那个公式。另外还有两个向量或 3 个向量的叉乘，但在这里没有列出叉乘公式。

向量的长度定义为

$$\|(v_1,v_2)\|=\sqrt{(v_1,v_2)\cdot(v_1,v_2)}=\sqrt{v_1^2+v_2^2} \tag{5.5}$$

同样的运算对于 n 维向量也适用。本书中的下标不采用数学中常见的从 1 开始编号的方式，而是采用 Python 从 0 开始编号的方式。两个 n 维向量的加减法公式如下：

$$(u_0,u_1,\cdots,u_{n-1})+(v_0,v_1,\cdots,v_{n-1})=(u_0+v_0,u_1+v_1,\cdots,u_{n-1}+v_{n-1}) \tag{5.6}$$

$$(u_0,u_1,\cdots,u_{n-1})-(v_0,v_1,\cdots,v_{n-1})=(u_0-v_0,u_1+v_1,\cdots,u_{n-1}-v_{n-1}) \tag{5.7}$$

标量 a 和向量 $(v_0,v_1,\cdots,v_{n-1})$ 的乘积定义为

$$(av_0,av_1,\cdots,av_{n-1}) \tag{5.8}$$

两个 n 维向量的内积或点乘定义为

$$(u_0,u_1,\cdots,u_{n-1})\cdot(v_0,v_1,\cdots,v_{n-1})=(u_0v_0,u_1v_1,\cdots,u_{n-1}v_{n-1})=\sum_{j=0}^{n-1}u_jv_j \tag{5.9}$$

n 维向量 $\boldsymbol{v}=(v_0,v_1,\cdots,v_{n-1})$ 的长度 $\|\boldsymbol{v}\|$ 定义为

$$\begin{aligned}\sqrt{(v_0,v_1,\cdots,v_{n-1})\cdot(v_0,v_1,\cdots,v_{n-1})}&=(v_0^2+v_1^2+\cdots+v_{n-1}^2)^{\frac{1}{2}}\\&=\left(\sum_{j=0}^{n-1}v_j^2\right)^{\frac{1}{2}}\end{aligned} \tag{5.10}$$

5.1.3 向量算术和向量函数

5.1.2 节的向量运算可能读者在高中数学课程中就学过。除此之外，还可以定义向量的其他运算，它们对于提高程序运行速度十分有用。接下来要讲到的这些运算几乎不被数学教材采纳，但它们在数学软件中具有重要作用，尤其是在 MATLAB、Octave、Python 和 R 等计算环境中。

一个单变量数学函数 $f(x)$ 作用于一个向量的结果是将 f 分别作用于向量的每个元素所得到的向量。若 $\boldsymbol{v}=(v_0,v_1,\cdots,v_{n-1})$ 是一个向量，那么：

$$f(\boldsymbol{v})=(f(v_0),f(v_1),\cdots,f(v_{n-1}))$$

例如，对 $\boldsymbol{v}$ 取正弦：

$$\sin\boldsymbol{v}=(\sin v_0,\sin v_1,\cdots,\sin v_{n-1})$$

向量的平方或 b 次方可以定义为将运算作用于每个元素：

$$\boldsymbol{v}^b=(v_0^b,v_1^b,\cdots,v_{n-1}^b)$$

在数学问题的计算机编程中经常出现的另一个运算是两个向量之间的“星号”乘法，它的定义是

$$\boldsymbol{u}*\boldsymbol{v}=(u_0v_0,u_1v_1,\cdots,u_{n-1}v_{n-1}) \tag{5.11}$$

标量与向量或数组相加可以定义为标量与每个元素相加。若 a 是标量，$\boldsymbol{v}$ 是向量，则有

$$a+\boldsymbol{v}=(a+v_0,a+v_1,\cdots,a+v_{n-1})$$

一个复合的向量表达式可能是以下形式：

$$v^2*\cos\boldsymbol{v}*\mathrm{e}^{\boldsymbol{v}}+2 \tag{5.12}$$

如何计算这个表达式？使用数学中的一般规则，按照从左到右的顺序逐个计算。注意，指数运算要先于乘、除运算，而乘、除运算又先于加、减运算。首先计算 $\boldsymbol{v}^2$，称计算结果为 $\boldsymbol{u}$。然后计算 $\cos v$ 且称计算结果为 $\boldsymbol{p}$。

接着计算 $\boldsymbol{u}*\boldsymbol{p}$，得到一个向量，称为 $\boldsymbol{w}$。下一步是计算 $e^{\boldsymbol{v}}$，称结果为 $\boldsymbol{q}$。随后计算 $\boldsymbol{w}*\boldsymbol{q}$，将结果存入 $\boldsymbol{r}$。最后计算 $\boldsymbol{r}+2$，得到最终的结果。将这些运算先后列出来可能更方便理解：

$$\begin{aligned}\boldsymbol{u} &= \boldsymbol{v}^2\\ \boldsymbol{p} &= \cos \boldsymbol{v}\\ \boldsymbol{w} &= \boldsymbol{u}*\boldsymbol{p}\\ \boldsymbol{q} &= e^{\boldsymbol{v}}\\ \boldsymbol{r} &= \boldsymbol{w}*\boldsymbol{q}\\ \boldsymbol{s} &= \boldsymbol{r}+2\end{aligned}$$

用向量 $\boldsymbol{v}=(v_0,v_1,\cdots,v_{n-1})$ 的一般形式写出向量 $\boldsymbol{u}$、$\boldsymbol{w}$、$\boldsymbol{p}$、$\boldsymbol{q}$、$\boldsymbol{r}$（务必要做！），从而表明式(5.12)的结果是下面这个向量：

$$(v_0^2\cos v_0 e^{v_0}+2, v_1^2\cos v_1 e^{v_1}+2,\cdots,v_{n-1}^2\cos v_{n-1}e^{v_{n-1}}+2)$$

向量结果中的第 i 个元素等于将式(5.12)作用于 v_i 的结果，其中 $*$ 代表两个数的乘法。

引入以下函数：

$$f(\boldsymbol{x}) = \boldsymbol{x}^2\cos \boldsymbol{x}e^{\boldsymbol{x}}+2$$

那么，$f(\boldsymbol{v})$就是将函数 f 作用于 $\boldsymbol{v}$ 的每个元素的结果。结果与向量表达式(5.12)相同。

在 Python 编程中，为了使计算更快、更方便，常常需要将单变量函数，如 $f(x)$，作用于向量。它的数学意义正是上面所阐述的。习题 5.5 和习题 5.6 有助你理解向量计算。在积累更多的编程经验后，就会发现向量计算的实用性。本章接下来的内容并不要求对于向量计算有非常彻底的理解。

在程序中，常常用数组来表示向量，但数组的操作比向量要广泛得多。直到 5.8 节之前，都可以把程序中的数组和向量等同看待。

5.2　Python 程序中的数组

本节介绍 Python 中的数组编程。这里先创建一些列表并说明数组和列表的不同之处。

5.2.1　用列表来收集函数数据

假设 $f(x)$为函数，希望求出该函数在若干 x 点的值：$x_0,x_1,\cdots,x_{n-1}$。将 n 个数对$(x_i,f(x_i))$放在一个列表中；也可以把所有的 x_i 值$(i=0,1,\cdots,n-1)$放在一个列表中，而把与之对应的 $f(x_i)$放在另一个列表中。下面的交互式会话展示了如何创建这 3 种列表：

```
>>>def f(x):
...     return x**3     #sample function
...
>>>n =5                 #number of points along the x axis
>>>dx =1.0/(n-1)       #spacing between x points in [0,1]
>>>xlist =[i * dx for i in range(n)]
>>>ylist =[f(x) for x in xlist]
>>>pairs =[[x, y] for x, y in zip(xlist, ylist)]
```

这里使用列表解析来实现紧凑的代码。应理解这些列表解析的含义（如果不能理解，可以尝试用标准 for 循环写出同样的代码，每次循环添加一个新的列表元素）。

上述列表元素是具有相同类型的对象：pairs 中的任一元素都是由两个浮点对象组成的列表，而 xlist 和 ylist 中的任一元素都是浮点数。列表非常灵活，列表元素可以是任何类型的对象。例如：

```
mylist =[2, 6.0, 'tmp.pdf', [0, 1]]
```

mylist 包含一个整数、一个浮点数、一个字符串和一个列表。这种不同类型的对象组合而成的列表称为异构列表。也可以方便地将列表元素移除，或者在列表的任何地方添加新元素。对于程序员来说，列表的这种灵活性非常方便。不过，在元素类型相同、元素个数固定的情况下，可以用数组替代列表。数组的优点在于：计算更快，内存需求更小，并且数学运算更为丰富。由于其高效性和数学上的便捷性，本书将大量使用数组。在 MATLAB、Octave 和 R 等其他编程环境中，也大量使用了数组。当需要灵活地添加或删除元素，或列表元素存在不同类型时，再考虑选择列表来替代数组。

5.2.2 Numerical Python 的数组基础

数组对象可以视为列表的变体，只不过它具有如下的假设和性质：

- 所有元素必须类型相同，并且为了数学计算和存储的效率，最好是整数、实数或复数类型。
- 创建数组时，数组元素个数必须已知。
- 数组不属于标准 Python 数据类型，需要一个额外的 Numerical Python 包（常简写为 NumPy）。在 import 语句中，这个包的名字写为 numpy。
- 有了 NumPy，就可以直接对纯数组运用许多数学运算，而不需要对数组元素进行 for 循环。这通常被称为向量化。
- 含有一个索引的数组通常被称为向量。含有两个索引的数组是存储表格的数据结构，它的效率高于列表的列表。数组也可以含有 3 个或更多的索引。

有两点要注意。首先，在标准 Python 中，其实有一个 array 对象类型，但是这个数据类型在数学计算上效率不高，因此在本书中不使用它。其次，数组中的元素个数是可以改变的，但是计算代价非常大。

下面将列出 NumPy 中关于数组的一些重要功能。更完整的介绍可以参见以下优秀参考书：*NumPy Tutorial*、*NumPy User Guide*、*NumPy Reference*、*Guide to NumPy* 和 *NumPy for MATLAB Users*。它们都可以在 scipy.org[①] 找到。

Numerical Python 包的标准 import 语句如下：

```
import numpy as np
```

用 NumPy 中的 array 函数将列表 r 转换为数组：

```
a =np.array(r)
```

要创建一个长度为 n、元素值均为 0 的新数组，可以使用 zeros 函数：

```
a =np.zeros(n)
```

数组 a 的元素类型对应于 Python 的 float 类型。可以用 np.zeros 的第二个参数来指定其他元素类型，例如 int。可以使用

```
a =np.zeros_like(c)
```

① http://scipy.org。

产生一个元素均为 0 的数组，其长度和元素类型都与数组 c 相同。有两个或更多索引的数组将在 5.8 节中讨论。

在数学问题中经常用到这样的数组：其 n 个元素的值在区间[p, q]上等间隔分布。NumPy 函数 linspace 可以创建这样的数组：

```
a =np.linspace(p, q, n)
```

和列表一样，可以通过方括号获取数组元素：a[i]。数组也可以与列表一样进行切片，例如，a[1，－1]挑选出除了第一个和最后一个元素外的其他所有元素。但和列表不同的是，a[1，－1]并不是 a 中数据的副本。因此，

```
b =a[1:-1]
b[2] =0.1
```

也会将 a[3]改成 0.1。切片 a[i:j:s]挑选出从下标 i 开始，步长为 s，直到(但不包括)j 的所有元素。省略 i 代表 i=0，省略 j 代表 j=n，其中 n 是数组中元素的个数。例如：a[0：－1：2]每两个元素取一个数，直到(但不包括)最后一个，而 a[：：4]则在整个数组中每 4 个元素取一个。

关于导入 NumPy 的说明

使用语句

```
import numpy as np
```

并将后面的所有 NumPy 函数和变量都加上前缀 np，已经逐渐成为成 Python 科学计算界的标准语法。然而，为了使 Python 程序看上去更接近 MATLAB 程序，同时降低 MATLAB 与 Python 语言之间转换的难度，可以用

```
from numpy import *
```

来去除前缀(这个做法已经逐渐成为交互式 Python Shell 中的标准)。本书建议采用 NumPy 数学函数不带前缀的写法，因为这样更接近于它们的数学表示。因此，$f(x)=\sinh(x-1)\sin wt$ 可编码如下：

```
from numpy import sinh, sin

def f(x):
    return sinh(x-1) * sin(w*t)
```

也可以采用最简单的办法：from numpy import *，就可以在程序中直接使用 NumPy 函数和变量。

5.2.3　计算坐标和函数值

在上述运算基础上，本节根据列表 xlist 和 ylist 创建数组：

```
>>>import numpy as np
>>>x2 =np.array(xlist)     #turn list xlist into array x2
>>>y2 =np.array(ylist)
```

```
>>>x2
array([0., 0.25, 0.5, 0.75, 1.])
>>>y2
array([0., 0.015625, 0.125, 0.421875, 1.])
```

可以不采用先创建列表，然后再将这个列表转换为数组的方法，而是直接对数组进行计算。x2 中的等间隔坐标可以直接用 np.linspace 函数计算，y2 数组可以通过 np.zeros 函数创建，从而确保 y2 的长度等于 len(x2)。然后，运行 for 循环来向 y2 填充 f 值：

```
>>>n =len(xlist)
>>>x2 =np.linspace(0, 1, n)
>>>y2 =np.zeros(n)

>>>for i in xrange(n):
...     y2[i] =f(x2[i])
...
>>>y2
array([0., 0.015625, 0.125, 0.421875, 1.])
```

注意，这里的 for 循环用 xrange 而不是 range。因为前者直接一个一个生成数值，避免了整数列表的生成和保存，因此对于长循环来说速度更快。于是，对于长数组的循环，程序员更喜欢使用 xrange 而不是 range。在 Python 3.x 中，range 和 xrange 是一样的。

创建一个指定长度的数组通常被称为分配数组。它只是为这个数组标记出一块计算机存储空间。在数学计算中进行长数组分配常常会占用大量存储空间。

通过列表解析创建 y2 数组可以简化上述代码。然而，列表解析产生的是列表而非数组，所以，需要进一步将列表对象转换为数组对象：

```
>>>x2 =np.linspace(0, 1, n)
>>>y2 =np.array([f(xi) for xi in x2])
```

还有更快的计算 y2 的方法，将在 5.2.4 节中解释。

5.2.4 向量化

长数组的循环可能运行缓慢。数组的一个很大的好处是可以摆脱循环而直接对整个数组应用数学运算函数：

```
>>>y2 =f(x2)
>>>y2
array([0., 0.015625, 0.125, 0.421875, 1.])
```

f(x2)之所以能起作用的神奇之处是建立在 5.1.3 节的向量运算概念上的。除了调用 f(x2)外，还可以等价地直接写函数公式 x2**3。

NumPy 为任意维数的数组实现了向量运算。而且，NumPy 本身也提供了许多常用的数学函数，例如 cos、sin、exp、log 等。它们能够处理数组参数，将数学函数作用于数组的每个元素。下面的代码分别计算每一个数组元素：

```
from math import sin, cos, exp
import numpy as np
x =np.linspace(0, 2, 201)
r =np.zeros(len(x))
for i in xrange(len(x)):
    r[i] =sin(np.pi * x[i]) * cos(x[i]) * exp(-x[i]**2) +2 +x[i]**2
```

下面是直接对整个数组进行计算的相应代码：

```
r =np.sin(np.pi * x) * np.cos(x) * np.exp(-x**2) +2 +x**2
```

很多人更喜欢下面没有 np 前缀的公式：

```
from numpy import sin, cos, exp, pi
r =sin(pi * x) * cos(x) * exp(-x**2) +2 +x**2
```

一个需要理解的重点是，math 模块的 sin 函数和 NumPy 提供的 sin 函数是不同的。前者不允许数组参数；而后者既接受实数作为参数，也接受数组作为参数。

像上面那样，用 sin(pi * x) * cos(x) * exp(－x**2)＋2＋x**2 这样的向量/数组表达式替代计算 r[i]的循环，称为向量化。循环版本则常被称为标量代码。例如：

```
import numpy as np
import math
x =np.zeros(N); y=np.zeros(N)
dx =2.0 / (N-1)            #spacing of xcoordinates
for i in range(N):
    x[i] =-1 +dx * i
    y[i] =math.exp(-x[i]) * x[i]
```

就是标量代码，而它相应的向量化版本是

```
x =np.linspace(-1, 1, N)
y =np.exp(-x) * x
```

注意，以下列表解析是标量代码：

```
x =array([-1 +dx * i for i in range(N)])
y =array([np.exp(-xi) * xi for xi in x])
```

因为它依旧使用显式的、缓慢的 Python for 循环操作标量。向量化代码的要求是：没有显式的 Python for 循环。在 NumPy 包中，计算每个数组元素的循环使用运算速度快的 C 或 FORTRAN 代码实现。

大多数使用标量参数 x 的 Python 函数，例如：

```
def f(x):
    return x**4 * exp(-x)
```

能够应用到数组参数 x 上：

```
x =np.linspace(-3, 3, 101)
y =f(x)
```

前提是 f 定义中使用的 exp 函数接受数组参数。这意味着 exp 必须是以 from numpy import * 或者明确地以 from numpy import exp 的方式导入的。当然也可以用 exp 的前缀形式：np. exp，不过这样会降低公式中数学语法的简洁性。

当一个 Python 函数 f(x)适用于数组参数 x 时，就说这个函数是向量化的。如果 f 中的数学表达式涉及 math 模块中的算术运算和基本的数学函数，那么只要导入时用 numpy 代替 math，就能使 f 自动向量化。然而，如果 f 中的表达式涉及 if 检测，那么代码必须被重写才能适用于数组。5.4.1 节展示了一些在函数向量化时必须进行特殊处理的例子。

向量化对于加速 Python 程序中重量级的数组运算非常重要。而且，向量化使得代码更简短，从而更易于阅读。在第 8 章中将会看到，向量化对于统计学模拟尤为重要。

5.3 绘制函数曲线

函数 $f(x)$的可视化是通过在 xy 坐标系中画出 $y=f(x)$的曲线来实现的。其在计算机上的实现方式称为绘制曲线。从技术上讲，可以通过在曲线上 n 个点之间画直线的方式来绘制曲线。点越多，曲线就显得越平滑。

假设要绘制函数 $f(x)$的曲线，x 的取值范围为 $a\leqslant x\leqslant b$。首先在区间$[a, b]$上选出 n 个 x 坐标，称它们为 $x_0, x_1, \cdots, x_{n-1}$；然后，对每个 $i=0,1,\cdots,n-1$ 计算 $y_i(=f(x_i))$。这样就得到 $y=f(x)$曲线上的 n 个点 (x_i, y_i)，$i=0,1,\cdots,n-1$。通常，等间距地选择 x_i 坐标，即

$$x_i = a + ih, \quad h = \frac{b-a}{n-1}$$

如果将 x_i 和 y_i 的值保存到两个数组 x 和 y 中，那么，就可以用一个类似于 plot(x, y)的指令来绘制这条曲线。

有时候，独立变量和函数的名字不是 x 和 f，但绘制的过程是相同的。

5.3.1 用 Matplotlib 实现 MATLAB 风格的绘图

Python 中标准的曲线绘制包是 Matplotlib。鉴于许多读者了解一些关于 MATLAB 的知识或者将来会用到 MATLAB，下面首先举例说明 Matplotlib 与 MATLAB 非常相似的一种绘图方法。

基础绘图

绘制 $y=t^2\exp(-t^2)$，t 在 0 到 3 之间取值。首先生成等间距的 t 坐标，例如取 51 个值(50 个区间)；然后，计算这 51 个点对应的 y 值；最后，调用 plot(t, y)指令绘制曲线。下面是完整的程序：

```
from numpy import *
from matplotlib.pylab import *

def f(t):
    return t**2 * exp(-t**2)

t =linspace(0, 3, 51)       #51 points between 0 and 3
y =zeros(len(t))            #allocate y with float elements
for i in xrange(len(t)):
    y[i] =f(t[i])
```

```
plot(t, y)
show()
```

该程序先为数组 y 分配存储空间，然后通过 for 循环为元素逐一赋值。也可以一次性对数组 t 进行操作，这样生成的代码更快、更短：

```
y = f(t)
```

为了将图形嵌入电子文档，需要 PDF、PNG 或者其他图形格式的文件。savefig 函数能将图形保存为多种格式的文件：

```
savefig('tmp1.pdf')      #produce PDF
savefig('tmp1.png')      #produce PNG
```

文件名后缀决定文件格式：.pdf 是 PDF 文件，.png 是 PNG 文件。图 5.2 展示的就是绘图结果。

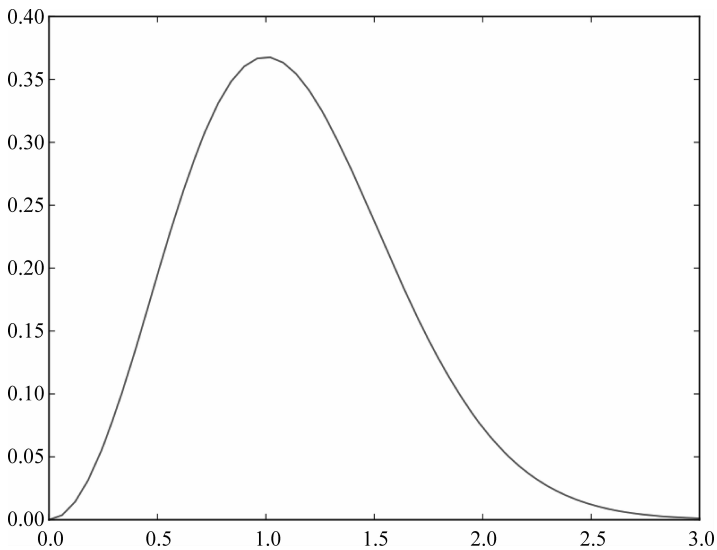

图 5.2　简单绘制的 PDF 格式图形(Matplotlib)

装饰图形

曲线图形中的 x 和 y 轴应该有标签，在这里分别是 t 和 y。同样，曲线也应该用标签标注，或者更通用的说法：图例。图形上方带标题也很常见。另外，还可以控制轴的范围(尽管大部分绘图程序会根据数据范围自动调节轴的范围)。以上所有内容都能轻易地在 plot 指令之后进行添加：

```
plot(t, y)
xlabel('t')
ylabel('y')
legend(['t^2 * exp(-t^2)'])
axis([0, 3, -0.05, 0.6])      #[tmin, tmax, ymin, ymax]
title('My First Matplotlib Demo')
savefig('tmp2.pdf')           #produce PDF
show()
```