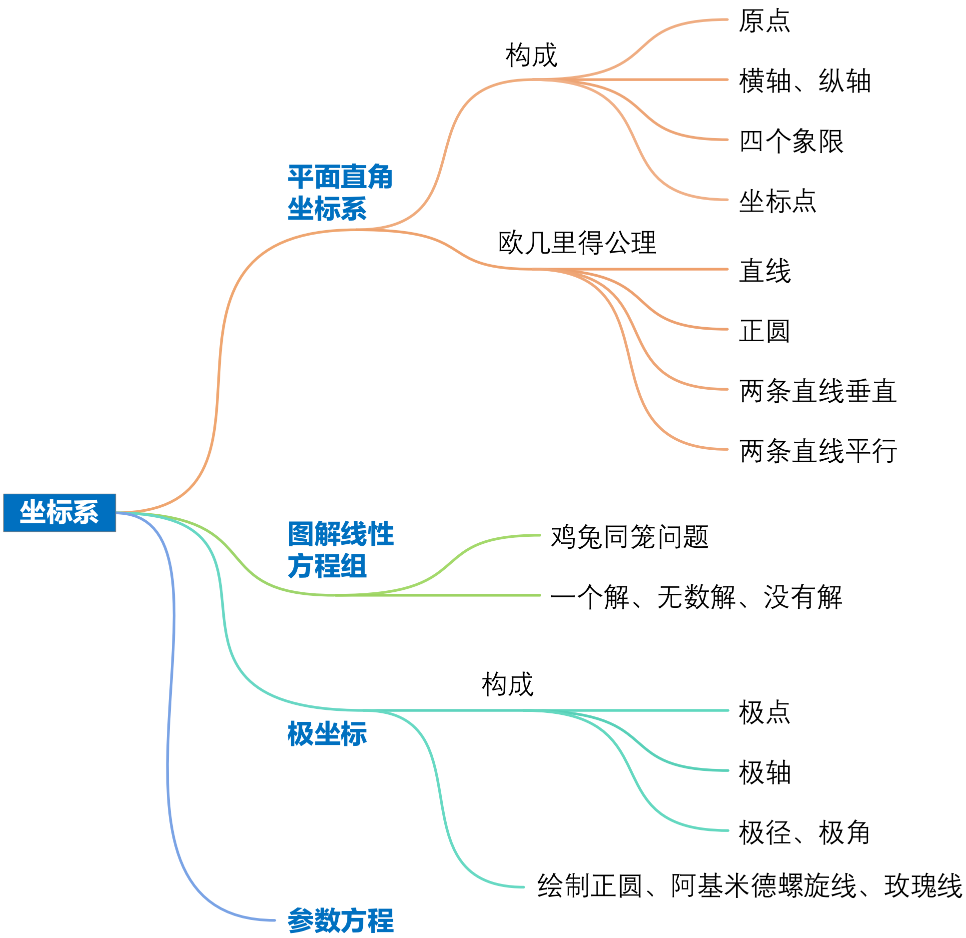
5 Cartesian Coordinate System 笛卡尔坐标系

几何代数一相逢，便胜却人间无数

> 我思，故我在。
>
> ***I think, therefore I am.***
>
> ***Cogito ergo sum.***
>
> —— 勒内·笛卡尔 (René Descartes) | 法国哲学家、数学家、物理学家 | 1596 ~ 1650

◄ `Axes3D.plot_surface()` 绘制三维曲面
◄ `matplotlib.pyplot.axhline()` 绘制水平线
◄ `matplotlib.pyplot.axvline()` 绘制竖直线
◄ `matplotlib.pyplot.plot()` 绘制线图
◄ `matplotlib.pyplot.scatter()` 绘制散点图
◄ `matplotlib.pyplot.text()` 在图片上打印文字
◄ `numpy.meshgrid()` 生成网格数据
◄ `plot_parametric()` 绘制二维参数方程
◄ `plot3d_parametric_line()` 绘制三维参数方程
◄ `seaborn.pairplot()` 成对散点图
◄ `seaborn.scatterplot()` 绘制散点图
◄ `sympy.is_decreasing()` 判断符号函数的单调性


坐标系
平面直角坐标系
构成
原点
横轴、纵轴
四个象限
坐标点
欧几里得公理
直线
正圆
两条直线垂直
两条直线平行
图解线性方程组
鸡兔同笼问题
一个解、无数解、没有解
极坐标
构成
极点
极轴
极径、极角
绘制正圆、阿基米德螺旋线、玫瑰线
参数方程

5.1 笛卡尔：我思故我在

笛卡尔 (René Descartes) 在《方法论》 (*Discourse on the Method*) 中写道："在我认为，任何事情都值得怀疑，但是这个正在思考的个体——我——一定存在。这样，我便得到第一条真理——我思故我在。"

勒内·笛卡尔 (René Descartes)
法国哲学家、数学家和科学家 | 1596年 ~ 1650年
解析几何之父

这一天，房间昏暗，笛卡尔躺在床上、百无聊赖，可能在思考存在的问题。一只不速之客闯入他的视野，笛卡尔把目光投向房顶，发现一只苍蝇飞来飞去、嘤嘤作响。

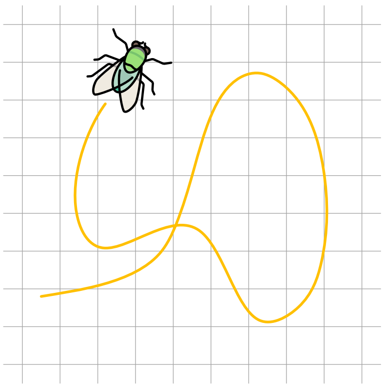


图 1. 笛卡尔眼中的苍蝇飞行

突然之间，一个念头在这个天才的大脑中闪过——要是在屋顶画上方格，我就可以追踪苍蝇的轨迹！

这个开创性的发明像一束耀眼的光束，瞬间洒满整个屋顶，照亮昏暗房间；它随即射入人类思想的夜空，改变了数学发展的路径。

笛卡尔坐标系让几何和代数这两条平行线交织在一起，再也没有分开。

几何形体就像是暗夜中大海上游弋的航船；坐标系就是灯塔，就是指引方位的北斗。

代数式每个符号原本瘦骨嶙峋、死气沉沉；坐标系让它们血肉丰满、生龙活虎。

毫不夸张地说，没有笛卡尔坐标系，就不会有函数，更不会有微积分。

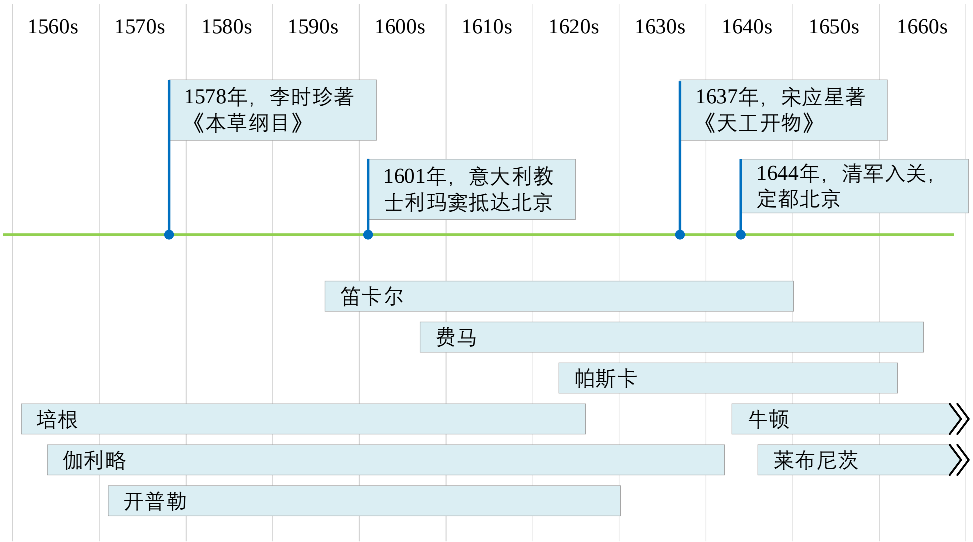


图 2. 笛卡尔时代时间轴

5.2 坐标系：代数可视化，几何参数化

平面直角坐标系

在平面上，**笛卡尔坐标系** (Cartesian coordinate system)，也叫平面直角坐标系，是两个相互垂直的实数轴，它们相交于**原点** (origin)。数学中，平面直角坐标系常记做 $\mathbb{R}^2$ 。

横轴 (horizontal number line) 常被称作 x 轴 (x-axis)，**纵轴** (vertical number line) 常被称作 y 轴 (y-axis)。注意，本书也常用 x_1 表示横轴，用 x_2 表示纵轴。

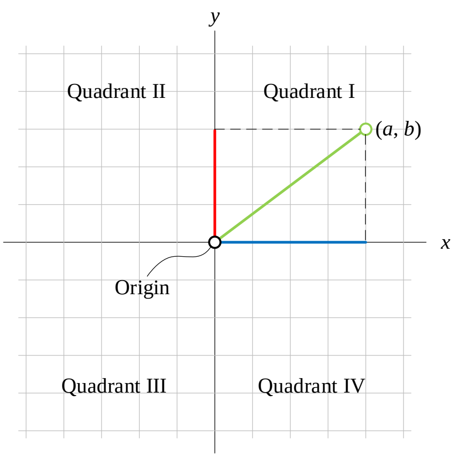


图 3. 笛卡尔坐标系

横纵轴将 xy 平面 (xy-plane) 分成四个**象限** (quadrants)。象限通常以**罗马数字** (Roman numeral) **逆时针方向** (counter-clockwise) 编号。注意，象限不包括坐标轴。

平面上的每个点都可以表示为坐标 (a, b)，a 和 b 两个值分别为**横坐标** (x-coordinate) 和**纵坐标** (y-coordinate)。图 4 所示为平面直角坐标系中 6 个点对应的坐标。

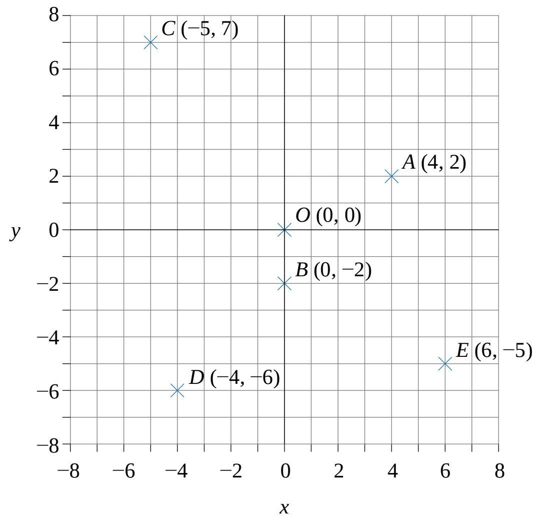


图 4. 平面直角坐标系中 6 个点的位置

以下代码绘制图 4 所示平面直角坐标系网格和其中 6 个点，并打印坐标值。

```python
# Bk_Ch5_01

import matplotlib.pyplot as plt
import numpy as np

x = [4,  0, -5, -4,  6, 0]
y = [2, -2,  7, -6, -5, 0]

labels = ['A', 'B', 'C', 'D', 'E', 'O']

fig, ax = plt.subplots()

plt.plot(x, y, 'x')

for label, i, j in zip(labels, x, y):
    plt.text(i, j+0.5, label + ' ({}, {})'.format(i, j))

plt.xlabel('x'); plt.ylabel('y')
plt.axhline(y=0, color='k', linestyle='-')
plt.axvline(x=0, color='k', linestyle='-')
plt.xticks(np.arange(-8, 8 + 1, step=1))
plt.yticks(np.arange(-8, 8 + 1, step=1))
plt.axis('scaled')

ax.set_xlim(-8,8); ax.set_ylim(-8,8)
ax.spines['top'].set_visible(False); ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False); ax.spines['right'].set_visible(False)
```

```
ax.grid(linestyle='--', linewidth=0.25, color=[0.5,0.5,0.5])
```

欧几里得的五个公理

有了直角坐标系，欧几里得提出的五个公理就可以很容易被量化，如图 5 和图 6 所示。

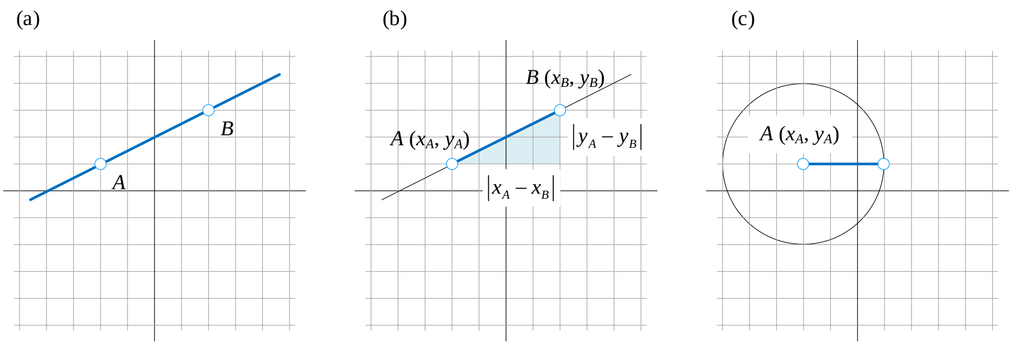


图 5. 在直角坐标系中展示直线、线段长度和圆

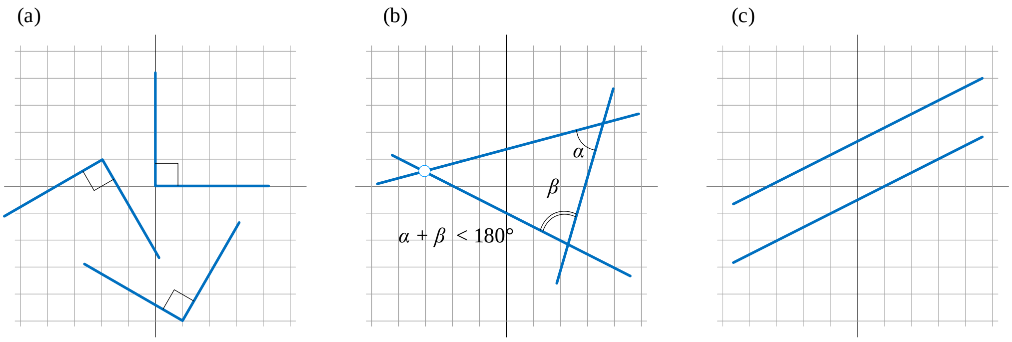


图 6. 在直角坐标系中展示直角、相交和平行

直线

任意两点可以画一条直线，这条直线一般对应代数中的二元一次方程。

$$ax + by + c = 0 \tag{1}$$

特别地，当 $a = 0$ 时，直线平行于横轴。

$$by + c = 0 \tag{2}$$

当 $b = 0$ 时，直线平行于纵轴。

$$ax + c = 0 \tag{3}$$

如果 a、b 均不为 0，(1) 可以写成。

$$y = -\frac{a}{b}x - \frac{c}{b} \tag{4}$$

当 x 为自变量、y 为因变量时，(4) 实际上就变成了一元一次函数。其中，$-a/b$ 为直线**斜率** (slope)，$-c/b$ 为**纵轴截距** (y-intercept)。

两点距离

如图 5 (b) 所示，A (x_A, y_A) 和 B (x_B, y_B) 两点之间直线的距离可以用勾股定理获得。

$$AB = \sqrt{\left(x_A - x_B\right)^2 + \left(y_A - y_B\right)^2} \tag{5}$$

正圆

如图 5 (c) 所示，以 A (x_A, y_A) 点为圆心，r 为半径画一个圆。圆上任意一点 (x, y) 到 A (x_A, y_A) 点的距离为 r，这样可以构造等式。

$$\sqrt{\left(x - x_A\right)^2 + \left(y - y_A\right)^2} = r \tag{6}$$

(6) 两边平方得到图 5 (c) 所示圆的解析式。

$$\left(x - x^{(A)}\right)^2 + \left(y - y^{(A)}\right)^2 = r^2 \tag{7}$$

特别地，当圆心为原点 (0, 0)，半径 $r = 1$ 时，圆为**单位圆** (unit circle)，对应的解析式为。

$$x^2 + y^2 = r^2 \tag{8}$$

有了平面直角坐标系，单位圆和各种三角函数之间联系就很容易可视化，具体如图 7 所示。

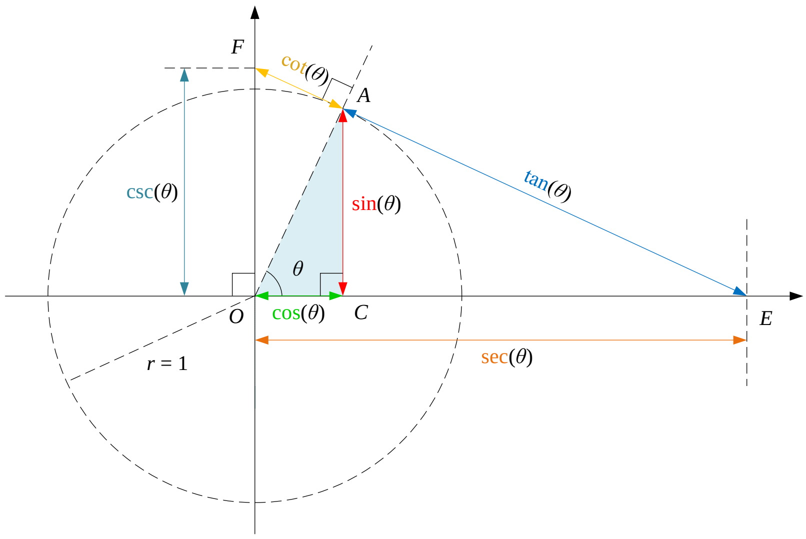


图 7. 三角函数和单位圆的关系

请大家特别注意 θ 为 $\pi/2$ (90°) 的倍数时，即 $\theta = \pi k/2$ (k 为整数)，有些三角函数值为无穷，即没有定义。比如 $\theta = 0$ (0°) 时，点 A 在横轴正半轴上，图 7 中 csc() 和 cot() 均为无穷。又如 $\theta = \pi/2$ (90°) 时，点 A 在纵轴正半轴上，图 7 中 sec(θ) 和 tan(θ) 均为无穷。

图 8 所示为平面直角坐标系中，角度、弧度和常用三角函数的正负关系。本书后续会介绍各种三角函数的图像。

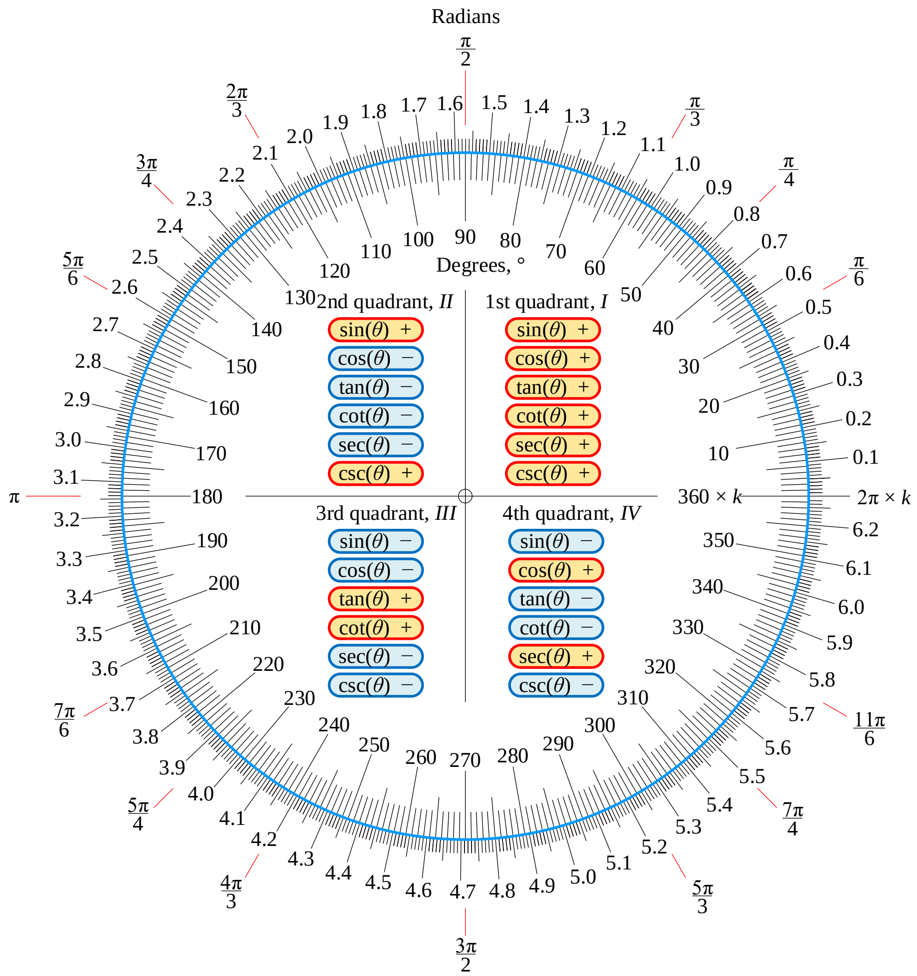


图 8. 平面直角坐标系中，角度、弧度和常用三角函数的正负关系

垂直

平面直角坐标系中，判断垂直变得更加简单。

给定 $ax + by + c = 0$ 和 $\alpha x + \beta y + \gamma = 0$ 两条直线，两者垂直时满足如下条件。

$$a\alpha + b\beta = 0 \tag{9}$$

如果系数 a、b、α、β 均不为 0 时，两条直线若垂直，则两条直线斜率相乘为−1，即。

$$\frac{a}{b}\frac{\alpha}{\beta}=-1 \tag{10}$$

图 9 (a) 所示为两条垂直线，它们分别代表 $y = 0.5x + 2$ 和 $y = -2x - 1$ 这两个一次函数；显然两个一次函数斜率相乘为−1 (= 0.5×(−2))。

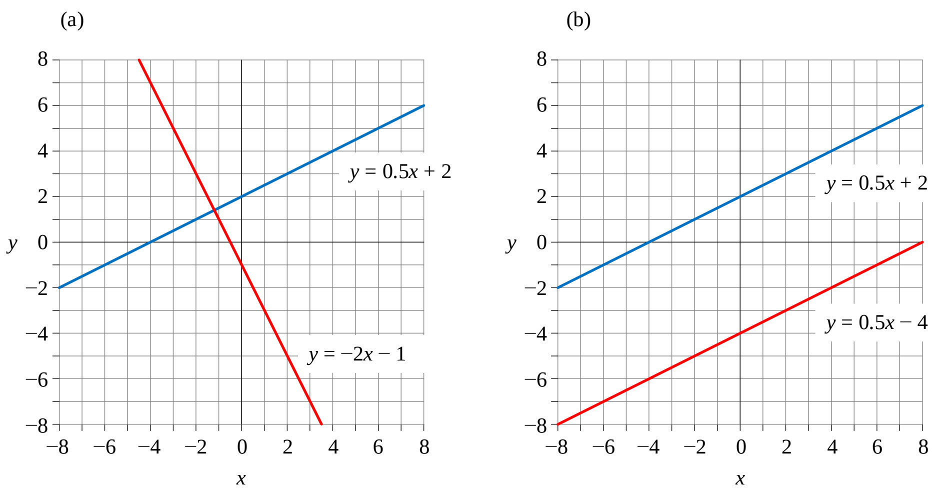


图 9. 两条垂直直线和两条平行线

平行

类似地，如果 $ax + by + c = 0$ 和 $\alpha x + \beta y + \gamma = 0$ 两条直线平行，系数满足。

$$a\beta - b\alpha = 0 \tag{11}$$

如果系数 a、b、α、β 均不为 0 时，两条直线若平行或重合，则两个斜率相同，即。

$$\frac{a}{b}=\frac{\alpha}{\beta} \tag{12}$$

图 9 (b) 所示为两条平行线。图 10 分别展示的是两条水平线和两条竖直线。

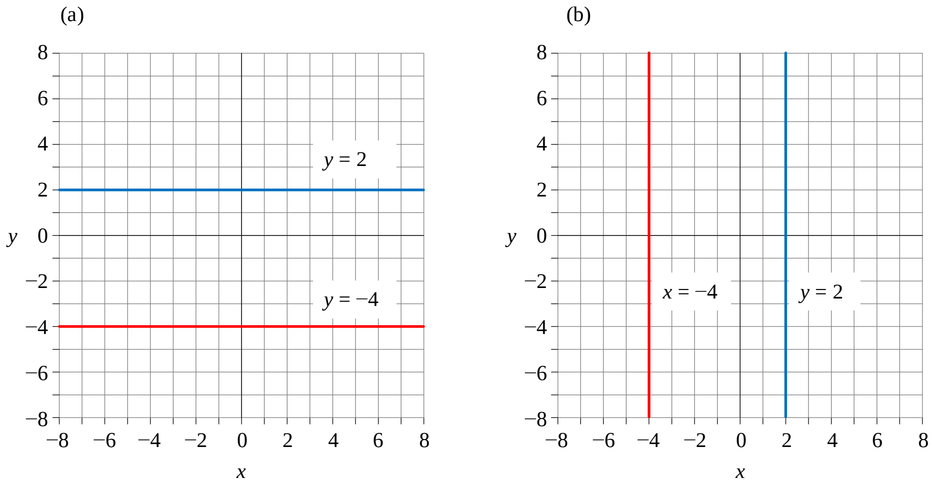


图 10. 两条水平线，和两条竖直线

表 1 总结有关坐标系的常用英文表达。

表 1. 有关坐标系的常用英文表达

数学或中文表达	英文表达
(a,b)	The point *a*, *b*
$P(a,b)$	The point capital *P* with coordinates *a* and *b*
$P(4,3)$	The *x*-coordinate of point *P* is 5; and the *y*-coordinate of point *P* is 8. The coordinates of point *P* are (5, 8). 5 is the *x*-coordinate and 8 is the *y*-coordinate *P* is 5 units to the right of and 8 units above the origin.
第一象限	First quadrant
y 轴正方向	Positive direction of the *y*-axis
y 轴负方向	Negative direction of the *y*-axis
x 轴正方向	Positive direction of the *x*-axis
x 轴负方向	Negative direction of the *x*-axis
关于 *x* 轴对称	To be symmetric about the *x*-axis
关于 *y* 轴对称	To be symmetric about the *y*-axis
关于原点对称	To be symmetric about the origin

以下代码可以用来绘制图 9 和图 10。

```
# Bk_Ch5_02

import numpy as np
import matplotlib.pyplot as plt

def fig_decor(ax):

    ax.set_xlabel('x')
    ax.set_ylabel('y')
```

```
    ax.hlines(y=0, xmin=-8, xmax=8, color='k')
    ax.vlines(x=0, ymin=-8, ymax=8, color='k')

    ax.set_xticks(np.arange(-8, 8 + 1, step=1))
    ax.set_yticks(np.arange(-8, 8 + 1, step=1))

    ax.axis('scaled')
    ax.grid(linestyle='--', linewidth=0.25, color=[0.5,0.5,0.5])

    ax.set_xbound(lower = -8, upper = 8)
    ax.set_ybound(lower = -8, upper = 8)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.spines['bottom'].set_visible(False)
    ax.spines['left'].set_visible(False)

x_array = np.linspace(-8,8)
y_array = np.linspace(-8,8)

# orthogonal
fig, ax = plt.subplots()

y1 = 0.5*x_array + 2
y2 = -2*x_array - 1

ax.plot(x_array, y1)
ax.plot(x_array, y2)
fig_decor(ax)

# parallel
fig, ax = plt.subplots()

y1 = 0.5*x_array + 2
y2 = 0.5*x_array - 4

ax.plot(x_array, y1)
ax.plot(x_array, y2)
fig_decor(ax)

# horizontal
fig, ax = plt.subplots()

y1 = 0*x_array + 2
y2 = 0*x_array - 4

ax.plot(x_array, y1)
# axhline
ax.plot(x_array, y2)
fig_decor(ax)

# vertical
fig, ax = plt.subplots()

x1 = 0*y_array + 2
x2 = 0*y_array - 4

ax.plot(x1, y_array)
# axvline
ax.plot(x2, y_array)
fig_decor(ax)
```

5.3 图解“鸡兔同笼”问题

图解法

有了平面直角坐标系，我们就可以图解前文鸡兔同笼问题。

首先构造二元一次方程组，这次用 x_1 代表鸡，x_2 代表兔。

鸡、兔共有 35 个头，对应如下等式。

$$x_1 + x_2 = 35 \tag{13}$$

有 94 只足，对应等式。

$$2x_1 + 4x_2 = 94 \tag{14}$$

联立两个等式，得到方程组。

$$\begin{cases} x_1 + x_2 = 35 \\ 2x_1 + 4x_2 = 94 \end{cases} \tag{15}$$

用图解法，(13) 和 (14) 分别代表平面直角坐标系的两条直线，如图 11。两条直线的交点就是解 (23, 12)。也就是，笼子里有 23 只鸡，12 只兔。

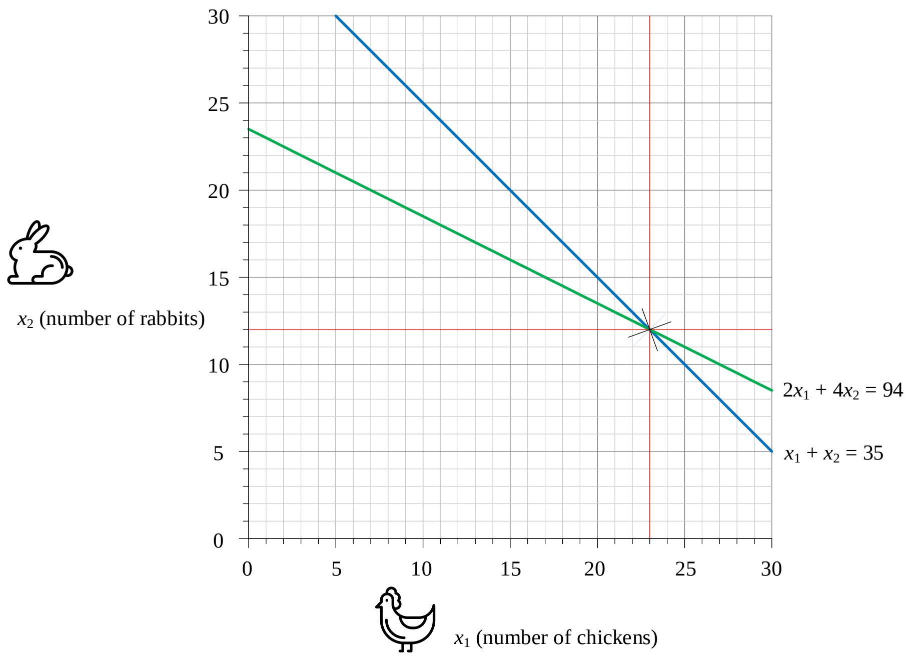


图 11. 鸡兔同笼问题方程组对应的图像

限制条件

实际上，图 11 两条直线并不能准确表达鸡兔同笼问题的全部条件。

鸡兔同笼问题还隐含着限制条件——x_1 和 x_2 均为非负整数；也就是说，鸡、兔的个数必须是 0 或正整数，不能是小数，更不能是负数。

有了这个条件作为限制，我们便可以获得如图 12 这幅图像。可以看到，方程对应的图像不再是连续的直线，而是一个个点；图 12 的网格交点对应整数坐标点，可以看到所有的 × 点都在网格交点处。

而且所有的点被限制在第一象限 (包含坐标轴)；这个区域对应不等式组。

$$\begin{cases} x_1 \geq 0 \\ x_2 \geq 0 \end{cases} \tag{16}$$

不等式区域是下一章要探讨的话题。

从另外一个角度来看，图 12 中 × 和 × 两组点对应的横纵轴坐标值分别构成等差数列 (arithmetic progression)。

等差数列是指从第二项起，每一项与它的前一项的差等于同一个常数的一种数列。注意，数列也可以看做是定义域不连续的特殊函数。

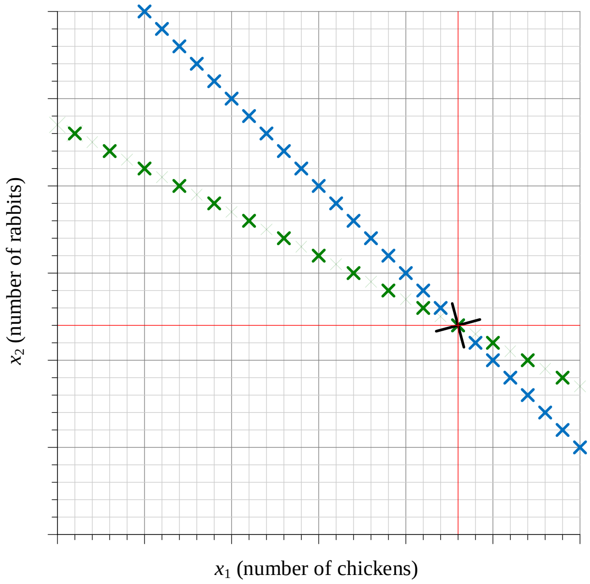


图 12. 鸡兔同笼问题方程组对应的非负整数图像

二元一次方程组解的个数

给出两个二元一次方程构成的方程组，可以求解得到一个解、无数解或者没有解。

有了图像，这一点就很好理解了。图 13 (a) 给出的两条直线相交于一点，也就是二元一次方程组有一个解；图 13 (b) 给出的两条直线相重合，也就是二元一次方程组无数解；图 13 (c) 给出的两条直线平行，也就是二元一次方程组没有解。

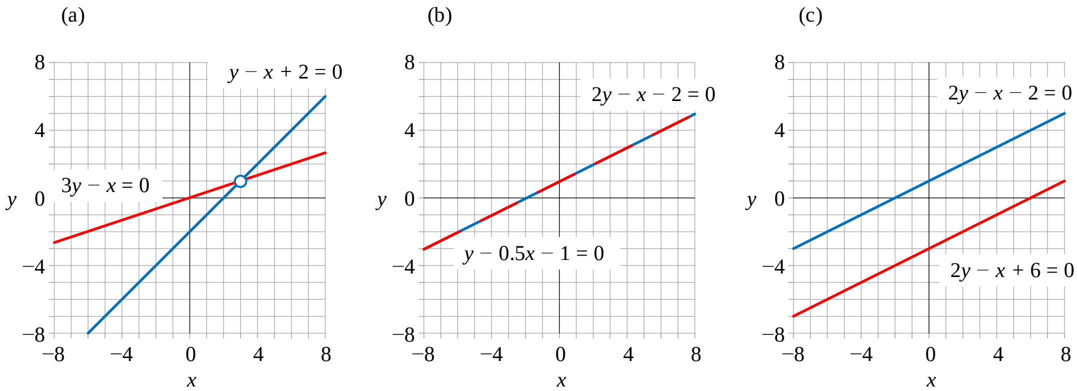


图 13. 两个二元一次方程组有一个解、无数解、没有解

以下代码绘制图 11。代码并没有直接计算出方程组的解，这个任务交给本书线性代数相关内容来解决。

```python
# Bk_Ch5_03

import numpy as np
import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True

x_array = np.arange(0, 30 + 1, step=1)
y_line_1 = 35 - x_array
y_line_2 = (94 - 2*x_array)/4

fig, ax = plt.subplots()

plt.plot(x_array, y_line_1, color = '#0070C0')
plt.plot(x_array, y_line_2, color = 'g')

# solution of linear equations
plt.plot(23,12,marker = 'x', markersize = 12)
plt.axvline(x=23, color='r', linestyle='--')
plt.axhline(y=12, color='r', linestyle='--')

plt.xlabel('$x_1$ (number of chickens)')
plt.ylabel('$x_2$ (number of rabbits)')
plt.axhline(y=0, color='k', linestyle='-')
plt.axvline(x=0, color='k', linestyle='-')
plt.xticks(np.arange(0, 30 + 1, step=5))
plt.yticks(np.arange(0, 30 + 1, step=5))
plt.axis('scaled')
plt.minorticks_on()
ax.grid(which='minor', linestyle=':',
        linewidth='0.5', color=[0.8, 0.8, 0.8])
ax.set_xlim(0,30); ax.set_ylim(0,30)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.grid(linestyle='--', linewidth=0.25, color=[0.5,0.5,0.5])
```

5.4 极坐标：距离和夹角

极坐标系 (polar coordinate system) 也是常用坐标系。如图 14 所示，平面直角坐标系中，位置由横轴、纵轴构成的网格确定；而极坐标中，位置由一段距离 r 和一个夹角 θ 来确定。

如图 14 右图所示，O 是极坐标的**极点** (pole)，从 O 向右引一条射线作为**极轴** (polar axis)，规定逆时针角度为正。

这样，平面上任意一点 P 的位置可以由线段 OP 的长度 r 和极轴到 OP 的角度 θ 来确定。(r, θ) 就是 P 点的极坐标。

一般，r 称为**极径** (radial coordinate 或 radial distance)，θ 称为**极角** (angular coordinate 或 polar angle 或 azimuth)。

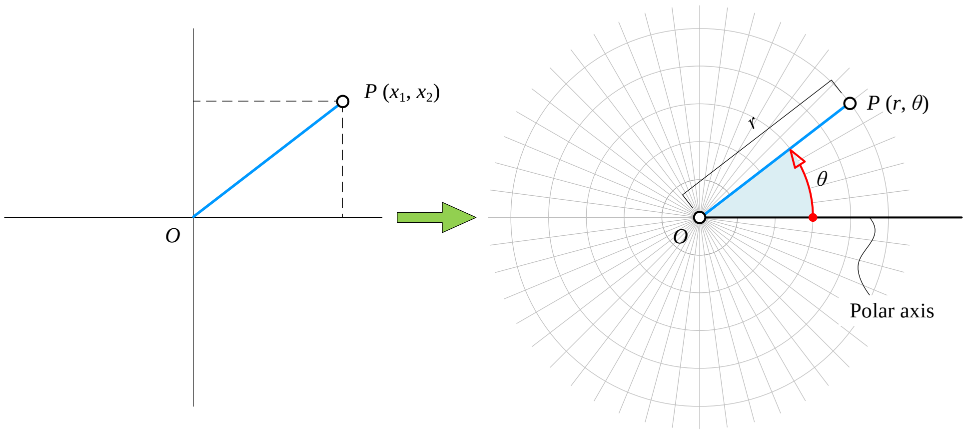


图 14. 从平面直角坐标系到极坐标系

平面极坐标让一些曲线可视化变得非常容易。

图 15 (a) 所示为极坐标中绘制的正圆，图 15 (b) 所示为阿基米德螺旋线 (Archimedean spiral)，图 15 (c) 为玫瑰线。

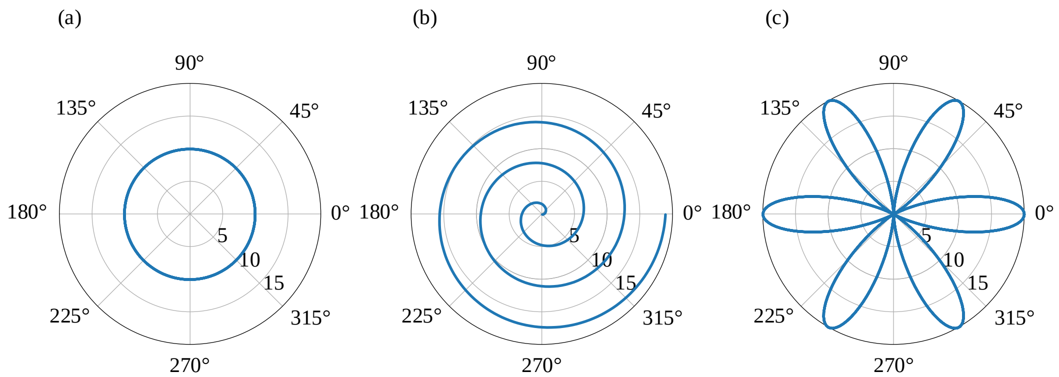


图 15. 平面极坐标中可视化三个曲线

图 8 所示为平面直角坐标系中，角度、弧度和三角函数的正负关系。

以下代码可以绘制图 15 三幅图像。

```
# Bk_Ch5_04

import numpy as np
import matplotlib.pyplot as plt

def plot_polar(theta, r):

    fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
    ax.plot(theta, r)
    # set radial axis limit
    ax.set_rmax(20)
    # set radial axis ticks
    ax.set_rticks([5, 10, 15, 10])
    # position radial labels
    ax.set_rlabel_position(-45)
    ax.grid(True); plt.show()

#%% circle
theta = np.linspace(0, 6*np.pi, 1000)

r = 10 + theta*0
plot_polar(theta, r)

#%% Archimedes' spiral
r = 1*theta
plot_polar(theta, r)

#%% Rose
r = 10*np.cos(6*theta) + 10
plot_polar(theta, r)
```

5.5 参数方程：引入一个参数

在平面直角坐标系中，如果曲线上任意一点坐标 x、y 都是某个参数，比如 t，的函数：对于 t 的任何取值，方程组确定的点(x, y) 都在这条曲线上，那么这个方程就叫做曲线的**参数方程** (parametric equation)，t 简称为参数。

$$\begin{cases} x_1 = f(t) \\ x_2 = g(t) \end{cases} \tag{17}$$

图 16 所示为用参数方程法绘制的单位圆，对应的参数方程为。

$$\begin{cases} x_1 = \cos(t) \\ x_2 = \sin(t) \end{cases} \tag{18}$$

其中，t 为参数，取值范围为 $[0, 2\pi]$。容易发现这就是单位圆上点的极坐标。

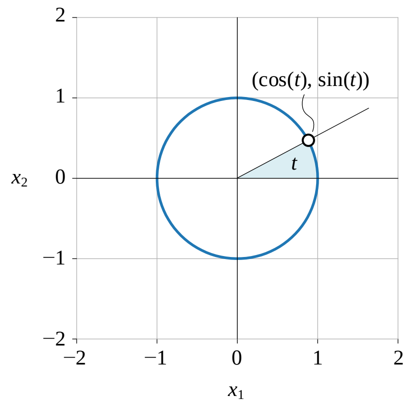


图 16. 参数方程绘制正圆

以下代码可以绘制图 16。

```
# Bk_Ch5_05

import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0,2*np.pi,100)

# parametric equation of unit circle
x1 = np.cos(t)
x2 = np.sin(t)

fig, ax = plt.subplots()
# plot the circle
plt.plot(x1,x2)

plt.show()
ax.set_xlabel('$x_1$'); ax.set_ylabel('$x_2$')
ax.set_xticks(np.arange(-2, 2 + 1, step=1))
ax.set_yticks(np.arange(-2, 2 + 1, step=1))
ax.axis('scaled')
ax.grid(linestyle='--', linewidth=0.25, color=[0.7,0.7,0.7])
ax.set_xbound(lower = -2, upper = 2); ax.set_ybound(lower = -2, upper = 2)
plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['left'].set_visible(False)
plt.gca().spines['bottom'].set_visible(False)
plt.axhline(y=0, color='k', linestyle='-')
plt.axvline(x=0, color='k', linestyle='-')
```

也可以采用 matplotlib.pyplot.plot() 可以用来绘制参数方程图像。比如，下列代码可以绘制图 16 所示单位圆。

```
# Bk_Ch5_06

import numpy as np
import matplotlib.pyplot as plt
```

```
t = np.linspace(0,2*np.pi,100)

# parametric equation of unit circle
x1 = np.cos(t)
x2 = np.sin(t)

fig, ax = plt.subplots()
# plot the circle
plt.plot(x1,x2)
plt.show()
```

我们也可以采用 sympy 工具包中的 `plot_parametric()` 函数绘制二维参数方程，比如下例便是通过 `t = symbols('t')` 定义符号变量 `t`；然后，利用 `plot_parametric()` 函数绘制单位圆。

```
# Bk_Ch5_07

from sympy import *
from sympy.plotting import plot_parametric
import math

t = symbols('t')

# parametric equation of unit circle
x1 = cos(t)
x2 = sin(t)

# plot the circle
plot_parametric(x1, x2, (t, 0, 2*pi))
```

5.6 坐标系必须是“横平竖直的方格”？

本章最后探究一下“坐标系”的内涵。广义来说，坐标系就是一个定位系统。

比如，地球上可以用经纬度来唯一确定地球上的唯一点，显然经纬度网格不是横平竖直，它更像本章讲到的极坐标。

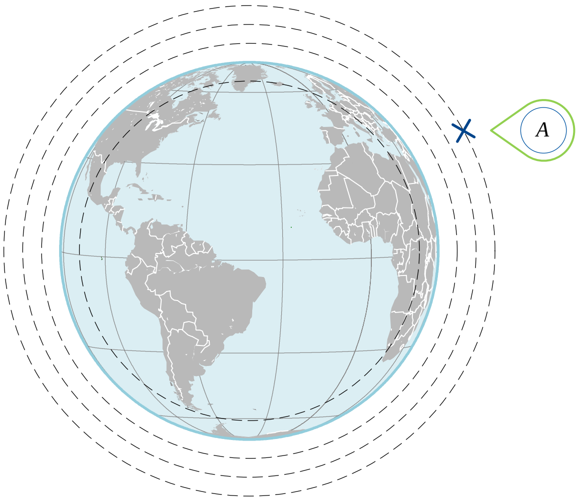


图 17. 经纬度加海拔定位

具体到某一个建筑内的位置时，我们加入楼层数这个定位参数；航空航天器定位时，会考虑海拔。

现在人类还是生存在地球“表面”；想象在不远的未来，人类可以大规模地在地下、海洋下方，甚至天空中生活，这时人们可能要自然而然地在经纬度基础上再加一个定位值，比如距离地心距离，或者海拔。很可能三座城市，经纬度几乎一致，却分别位于地表、地下和半空中。

坐标系的定义满足实际需求，也就是怎么方便，怎么来。

本章介绍的笛卡尔坐标系是数学中定位平面一点最常用的坐标系；笛卡尔坐标系是直角坐标系，白话说它用横平竖直的方格定位。

本章直角坐标系对应的都是横平竖直的“方格”，这是因为横纵坐标轴垂直，且尺度完全一致；很多情况，横纵坐标轴的数值尺度不同，这样我们获得“长方格”的直角坐标系。

如图 18 所示，横平竖直的方格，经过竖直方向、水平方向拉伸，得到两个不同长方格。当图像较复杂时，本系列丛书很多时候都不绘制网格，而只提供坐标轴上的刻度线和对应刻度值。

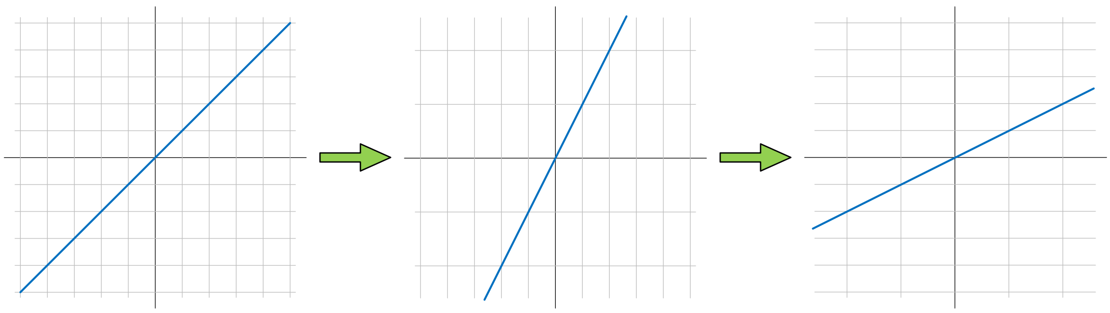


图 18. 直角坐标系，方格到长方格

图 18 中每幅图像中方格的大小还是保持一致。有些应用场合，一幅图像中方格大小还可能不一致；如图 19 右图所示图像的纵轴为对数坐标刻度 (logarithmic scale)。

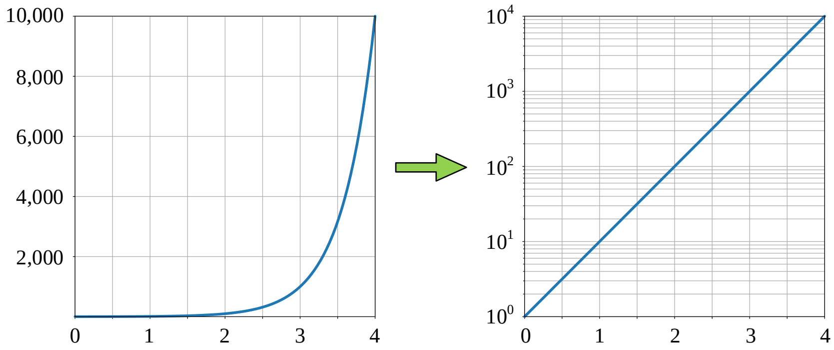


图 19. 直角坐标系到纵轴为对数刻度

不管怎么说，图 19 的刻度线还是“横平竖直”；有些时候，“横平竖直”这个限制也可以被打破。图 20 中 (a)、(b) 和 (c) 三幅图坐标网格还是横平竖直，剩下 6 幅图网格则千奇百怪，有旋转、伸缩等等几何操作。即便如此，图 20 中 9 幅图都可以准确定位点 A 和点 O 的位置关系。

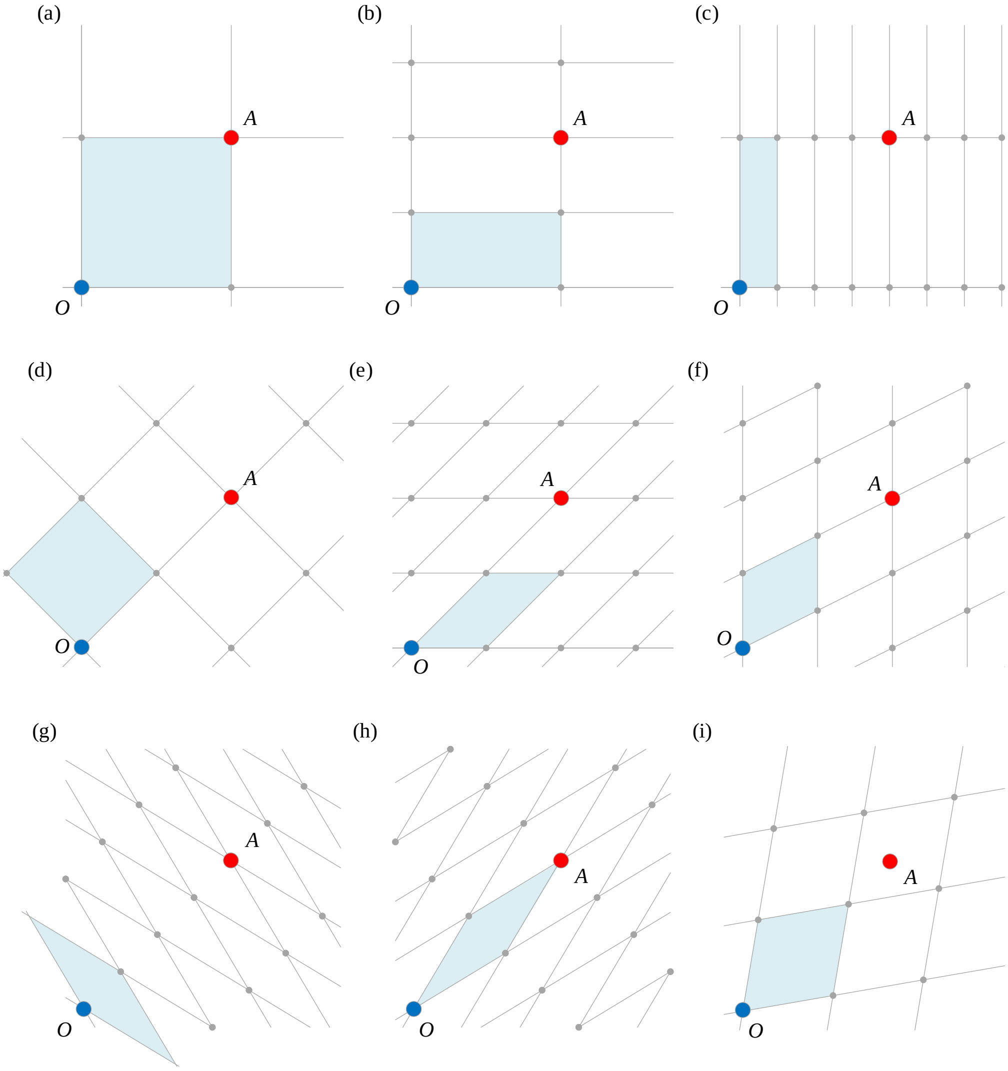


图 20. 不同坐标系表达点 A 和点 O 关系

即便如此，本节展示的各种坐标系还都束缚在同一个平面内；而这个平面最根本的坐标系就是笛卡尔直角坐标系。而各种坐标系似乎都和笛卡尔坐标系存在某种量化联系；目前我们介绍的数学工具还不足够解析这些量化联系，本系列丛书会讲解更多数学工具，慢慢给大家揭开谜底。

笛卡尔的坐标系像极了太极八卦。

太极生两仪，两仪生四象，四象生八卦。坐标系的原点就是太极的极，两极阴阳为数轴负和正；横轴 x 和纵轴 y 张成平面 $\mathbb{R}^2$，并将其分成为四个象限。

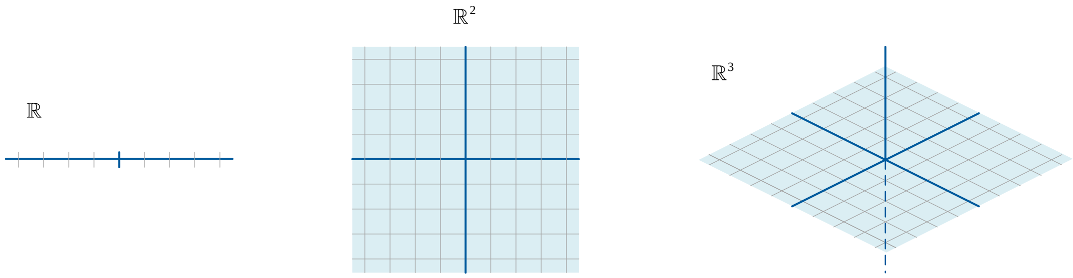


图 21. 数轴、平面直角坐标系、三维直角坐标系

垂直于 $\mathbb{R}^2$ 平面再升起一个 z 轴，便生成一个三维空间 $\mathbb{R}^3$；x、y 和 z 轴将三维空间割裂成八个区块。这是下一章要介绍的内容。

坐标系看似有界，但又无界；正所谓大方无隅，大器免成，大音希声，大象无形。

笛卡尔坐标系包罗万象，本章之后的所有数学知识和工具都包含在笛卡尔坐标系这个“大象”之中。