第五章　*Maple* 图形绘制

图形无疑是数学中最令人着迷的部分, 一些枯燥的公式可以从图形看出其美. 历史上有许多学者利用函数图形解决了学科中的许多难题.

客观地说, Maple 不是一种可视化的语言—它不会产生出版品质的图形. 然后, 它的图形功能非常强大, 足以提供更多的关于函数的信息. 当然, 如果需要, 它的图形作适当改进即可满足出版要求.

限于篇幅, 本章所有图形未作打印, 读者只需在计算机上按照书中语句操作即可观其效果, 更多图形功能可通过 Maple 帮助获得.

1　二维图形制作

Maple 所提供的二维绘图指令 **plot** 可以绘制二维的函数图、参数图、极坐标图、等高线图、不等式图，等等. 这些绘图指令有些已经内嵌在其核心程序里, Maple 启动时即被装入，直接调用函数命令即可，有些则需要使用 **with(plots)**调用 plots 函数库才能完成.

1.1 基本二维绘图指令

plot (f(x), x=xmin .. xmax);

plot (f(x), x=xmin .. xmax, y=ymin .. ymax);

plot ([f1(x), f2(x), …], x=xmin .. xmax);

plot (f(x), x=xmin .. xmax, option);

其中，xmin..xmax 为 x 的变化范围，ymin..ymax 为 y(即 f(x))的变化范围. **option** 选项参数主要有：

axes：设定坐标轴的显示方式, 一般有 FRAME(坐标轴在图形的左边与下面)、BOXED(坐标轴围绕图形)、NORMAL(一般方式显示)或 NONE(无)

color：设定图形所要涂的颜色(可选用也可自设)

coords：指定绘图时所用的坐标系(笛卡尔坐标系(**cartesian**，默认)、极坐标系(**polar**)、

双极坐标系(bipolar)、**logarthmic**(对数坐标系)等

discont：设定函数在不是否用线段连接起来(**discont=true** 则不连接，默认是 **discont=false**)

labels：设定坐标轴的名称(**labels=[x, y]**, x 与 y 分别为 x 与 y 坐标轴的名称)

linestyle：设定所绘线条的线型(**linestyle=n**, n 为 1 是实线, 2 为点, 3 为虚线, 4 为虚线与点交错)

numpoints：设定产生一个函数图形所需的最少样点

scaling：设置 x 与 y 轴的比例(unconstrained 非约束，constrained 约束，比例为 1:1)

style：设定图形的显示样式(LINE(线形)、POINT(点)、PATCH(显示多边形与边线)、PATCHNOGRID(只显示色彩而无边界)

symbol：设定点的格式(主要有 BOX(方块)、CROSS(十字)、CIRCLE(圆形)、POINT(点)、DIAMOND(菱形)等几项)

thickness：设定线条的粗细(0、1、2、3 几种参数, 数值越大线条越粗)

tickmarks：设定坐标轴刻度的数目(设定 **tickmarks=[m, n]**, 则 x 轴刻度为 m, y 轴为 n)

title：定义图形的标题(要用" "把标题引起来)

view：设定屏幕上图形显示的最大坐标和最小坐标，缺省是整个曲线

下面通过一些实例学习：

```
>plot(sin(1/x),x=-0.1..0.1,title="y=sin(1/x)",axes=normal);

>plot(1/(2*sin(x)),x=-10..10,y=-30..30);
```

试比较下述三图的效果:

```
>plot(tan(x),x=-2*Pi..2*Pi);

>plot(tan(x),x=-2*Pi..2*Pi, y=-5..5);

>plot(tan(x),x=-2*Pi..2*Pi, y=-5..5,discont=true);
```

(此处命令 **discont=true** 的作用是去除垂直渐近线)

```
> plot(sin(cos(6*x))/x, x=0..15*Pi, y=-0.6..0.5, axes=NONE);
```

```
>plot(Zeta(x),x=-3..3,y=-3..3,discont=true);
```

除了绘制基本的函数图之外，**plot** 还可绘制自定义函数的图形，也可以同时绘制多个函数图.

```
>f:=x->sin(x)+cos(x)^2;
 plot(f(x),x=0..16);

>plot([sin(x),sin(x^2),sin(x^3/10)],x=-2*Pi..2*Pi);
```

利用 **seq** 指令产生一个由函数所组成的序列，并将此函数的序列赋给变量，然后将函数序列绘于同一张图上.

```
>f:=x->sin(x)+cos(x);
 fs:=seq(f(x)^(n-1)+f(x)^n,n=1..4):
 plot([fs],x=0..20);

>f:=x->x*ln(x^2):g:=x->ln(x):
 plot({f,g},0..2,-1.5..1.5);
```

也可以直接把 **seq** 指令放在 **plot** 里来绘出一系列的函数图.

```
>plot([seq(f(x)^(2/n),n=1..3)],x=0..10);
```

1.2 二维参数绘图

更多情况下，我们无法把隐函数化成显函数的形式，因而 plot 指令无法在二维的平面里直接绘图. 但是，在某些情况下，我们可以把平面上的曲线 f(x, y)化成 x=x(t), y=y(t)的形式，其中 t 为参数(**parameter**). 据此即可绘图，其命令格式如下：

plot ([x(t), y(t), t=tmin .. tmax]);

plot ([x(t), y(t), t=tmin .. tmax], xmin .. xmax, y=ymin .. ymax);

plot ([x(t), y(t), t=tmin .. tmax], scaling=CONSTRAINED);

plot ([[x1(t), y1(t), t1=t1min .. t1max], [x2(t), y2(t), t2=t2min .. t2max],…]);

```
>plot([t*exp(t),t,t=-4..1],x=-0.5..1.5,y=-4..1);

>plot([sin(t),cos(t),t=0..2*Pi]);

>plot([sin(t),cos(t),t=0..2*Pi],scaling=CONSTRAINED);
```

上述两上语句都是绘制圆的命令，但由于后者指定的 x、y 坐标的比例为 1:1，所以才得到了一个真正的圆，而前者由于比例不同，则像个椭圆．下面则是内摆线的图形：

```
> x:=(a,b)->(a-b)*cos(t)+b*cos((a-b)*t/b);
```

$$x := (a, b) \to (a - b)\cos(t) + b\cos\left(\frac{(a-b)\,t}{b}\right)$$

```
> y:=(a,b)->(a-b)*sin(t)-b*sin((a-b)*t/b);
```

$$y := (a, b) \to (a - b)\sin(t) - b\sin\left(\frac{(a-b)\,t}{b}\right)$$

当 a=1, b=0.58 时，(x(a,b), y(a,b))图形绘制命令为：

```
> plot ([x(1,0.58), y(1,0.58), t=0..60*Pi], scaling=CONSTRAINED);
```

再作 a, b 取其它值时的情形：

```
> plot([x(2,1.2),y(2,1.2),t=0..6*Pi],scaling=CONSTRAINED);

> plot([x(2,8),y(2,8),t=0..16*Pi],scaling=CONSTRAINED);

> plot([x(2,12),y(2,12),t=0..16*Pi],scaling=CONSTRAINED);
```

下面再看同时绘制多个图形的情形．

```
> plot([[cos(3*t),sin(2*t),t=0..2*Pi],[sin(t),cos(3*t),t=0..2*Pi]]);
```

1.3 数据点绘图

如果所绘的图形是间断性的数据，而不是一个连续的函数，那么我们可以把数据点绘在 x-y 坐标系中，这就是所谓的数据点绘图．其命令格式如下：

plot([[x1, y1], [x2, y2], …], style=point);

plot([[x1, y1], [x2, y2], …]);

```
> data1:=seq([2*n,n^3+1],n=1..10):
  plot([data1],style=point);

> data2:=seq([n,1+(-1)^n/n],n=1..15):
  plot([data2],style=point,view=[0..20,0..2]);

> data3:=seq([t*cos(t/3),t*sin(t/3)],t=1..30):
  plot([data3],style=point);
```

1.4 其它坐标系作图

由于所研究的问题的特殊性，常常需要选用不同的坐标系，在 Maple 中除笛卡尔坐标系（**cartesian**，也称平面直角坐标系，默认)外，还提供了 **polar**(极坐标系)、elliptic(椭圆坐标系)、bipolar(双极坐标系)、maxwell(麦克斯韦坐标系)、logarithmic(双数坐标系)等 14 种二维坐标系，其中最常用的是极坐标系。设定坐标系的命令是 **coords.**

```
> plot(ln(x+1)^2,x=0..8*Pi, coords=polar, scaling=CONSTRAINED,thickness=2);

> plot(sin(6*x),x=0..68*Pi, coords=polar, scaling=CONSTRAINED, tickmarks=[3,3]);

> plot([sin(20*x),cos(sin(2*x))],x=0..2*Pi,coords=elliptic, scaling=CONSTRAINED,
  color=[red,blue]);

> plot(exp(sin(68*t)+cos(68*t)), t=0..2*Pi, coords=polar, scaling=CONSTRAINED);

> plot([seq(sin(t)+n*cos(t), n=-5..5)], t=0..Pi, coords=polar, scaling=CONSTRAINED);
```

试比较 y=sin(x)在不同坐标系中的图形显示：

```
> plot(sin(x),x=0..2*Pi,coords=polar,scaling=CONSTRAINED);

> plot(sin(x),x=0..2*Pi,coords=bipolar,scaling=CONSTRAINED);

> plot(sin(x),x=0..2*Pi,coords=elliptic,scaling=CONSTRAINED);

> plot(sin(x),x=0..2*Pi,coords=maxwell,scaling=CONSTRAINED);

> restart:
> with(plots,polarplot):
> r:=(n,theta)->cos(5*theta)+n*cos(theta);
```

$$r := (n, \theta) \rightarrow \cos(5\,\theta) + n\cos(\theta)$$

```
> plot({seq([r(n,t)*cos(t),r(n,t)*sin(t),t=0..Pi],n=-5..5)});

> polarplot((exp(cos(theta))-2*cos(4*theta)+sin(theta/12)^5),theta=0..24*Pi);
```

1.5 双轴作图

在最新 Maple12 中新增加了一个函数 dualaxisplot()用以实现双垂直轴作图，用以实现两垂直刻度范围相差较大的图形。

> dualaxisplot(exp1,exp2,range,opts) 或 dualaxisplot(p1,p2)

其中 exp1,exp2 为图形表达式，range 为横轴范围，opts 为参数定义同 plot/details

P1,p2 为已定义的图形数据; p1=plot();p2=plot();

2 三维绘图

2.1 基本三维绘图指令

三维空间的绘图比二维空间更有变化性和趣味性, 其命令函数为 plot3d, 可直接调用. 命令格式如下：

plot3d(f(x,y), x=xmin .. xmax, y=ymin .. ymax);

plot3d({f(x,y), g(x,y), …}, x=xmin .. xmax, y=ymin .. ymax);

plot3d(f(x,y), x=xmin .. xmax, y=ymin .. ymax, options);

其中，xmin..xmax 为 x 的变化范围，ymin..ymax 为 y(即 f(x))的变化范围. Option 选项参数与二维时的情形相似，这里只列示新增指令的意义：

cotours：设定等高线的数目或者等高线的值

grid：设定组成曲面的样点数或方形网格的数量

gridstyle：设定网格的形状(rectangular—矩形，triangular—三角形)

orientation：设定观看图形的视角(但设定视角的最佳方式是用鼠标拖动图形)

projection：设定投影的模式

shading：设定曲面着色的方式

与二维情形相同，在Maple中三维绘图坐标系的选定使用命令coords，缺省坐标系为笛卡尔坐标系(**cartesian**)，此外还有：bipolarcylindrical(双极坐标), bispherical(双球面坐标), cardioidal(心脏线坐标), cardioidcylindrical(心形柱坐标), casscylindrical( ), confocalellip(共焦椭球坐标), confocalparab(共焦抛物线坐标), conical(锥形坐标), cylindrical(柱坐标), ellcylindrical(椭柱坐标), ellipsoidal(椭球坐标), hypercylindrical (超圆柱坐标), invcasscylindrical, invellcylindrical(逆椭球坐标), invoblspheroidal( ), invprospheroidal( ), logcoshcylindrical(双数双曲余弦柱坐标), logcylindrical(对数柱坐标), maxwellcylindrical(麦克斯韦柱坐标), oblatespheroidal( ), paraboloidal(抛物面坐标), paracylindrical(参数柱坐标), prolatespheroidal(扁类球坐标), rosecylindrical(玫瑰形柱坐标), sixsphere(六球坐标), spherical(球坐标), tangentcylindrical(正切柱坐标), tangentsphere(正切球坐标)和toroidal(圆环面坐标).

```
> plot3d(x*y^2/(x^2+y^4),x=-1..1,y=-1..1,axes=boxed);

> plot3d(x*y/(x^2+y^2+2*x*y),x=-4..4,y=-4..4, axes=BOXED);
```

> **plot3d(sin(x*y),x=-Pi..Pi,y=-Pi..Pi);**

> **plot3d({2*sin(x)*cos(y),-6*x/(x^2+y^2+1)},x=-4..4,y=-4..4);**

> **plot3d(sin(z/2), t=0..3*Pi/2, z=-4..4, coords=spherical);**

> **plot3d(1,t=0..2*Pi,p=0..Pi, coords=spherical, scaling=constrained);**

> **plot3d(sin(t)*sin(p^2), t=0..Pi, p=0..Pi, coords=spherical, grid=[35,35]);**

> **plot3d(theta,theta=0..8*Pi,phi=0..Pi, coords=spherical, style=wireframe);**

> **plot3d(theta,theta=0..8*Pi,phi=0..Pi, coords=toroidal(2), style=wireframe);**

> **plot3d(theta,theta=0..8*Pi,z=-1..1, coords=cylindrical, style=patch):**

2.2 三维参数绘图

当二元函数无法表示成 $z = f(x, y)$ 时，有时可以用一组参数方程表示，关于这类参数方程的 Maple 作图，指令如下：

plot3d([fx, fy, fz], t=tmin .. tmax, u=umin .. umax);

plot3d([fx, fy, fz], t=tmin .. tmax, u=umin .. umax, options);

```
> plot3d([sin((x+10)/2),cos(y^3/3),x],x=-4..4,y=1..4);
```

> **plot3d([cosh(u)*cos(v),cosh(u)*sin(v),u],u=-2..2,v=0..2*Pi);**

> **plot3d([cos(u)*cos(v),cos(u)*sin(v),u^2],u=-2..2,v=0..2*Pi,axes=FRAME);**

> **plot3d([cos(u)*cos(v),cos(u)*sin(v), sin(u)], u=-1..1, v=0..2*Pi, orientation=[146,21], scaling=CONSTRAINED);**

3 特殊作图

3.1 图形的显示与合并

```
> with(plots):
  g1:=plot(cos(x),x=-2*Pi..2*Pi):
  g2:=plot(sin(x),x=-2*Pi..2*Pi,thickness=5):
```

```
display(g1,g2,axes=BOXED);
```

```
> g3:=plot3d(2*exp(-sqrt(x^2+y^2)),x=-6..6,y=-6..6):
  g4:=plot3d(sin(sqrt(x^2+y^2)),x=-6..6,y=-6..6):
  display(g3,g4);
```

3.2 不等式作图

不等式作图基本上有 4 部分：

① 解区间(feasible region)：此区域完全满足所有的不等式;

② 非解区间(excluded region)：此区域不完全满足所有不等式;

③ 开线(open lines)：不等式的边界, 但不包含此边界;

④ 闭线(closed lines)：不等式的边界(包含此边界)

```
> with(plots):
  inequal(2*x-5*y<6,x=-3..3,y=-3..3);
```

```
> ineqns:={x-y+2>0,2*x+3*y+9>0,8*x+3*y-27<0};
  sol:=solve(ineqns,{x,y});
  ans:=map(convert,sol,equality);
  implicitplot(ans,x=-6..8,y=-10..10);
```

```
> inequal(ineqns,x=-6..8,y=-10..10,optionsexcluded=
  (color=wheat),optionsopen=(color=red));
```

```
> neweqs:=ineqns union{x>=0,y>=0}:
> inequal(neweqs,x=-6..8,y=-10..10,optionsexcluded=
  (color=wheat),optionsopen=(color=red));
```

3.3 空间曲线绘图

```
> with(plots):
spacecurve([cos(t/2),sin(t/2),t,t=0..68*Pi],numpoints=500);
```

```
> spacecurve({[3*cos(t), 3*sin(t), t, t=0..12*Pi], [2+t*cos(t), 2+t*sin(t), t, t=0..
  10*Pi]}, numpoints=200);
```

```
> spacecurve({[t*cos(2*Pi*t),t*sin(2*Pi*t),2+t],[2+t,t*
```

```
cos(2*Pi*t),t*sin(2*Pi*t)],[t*cos(2*Pi*t),2+t,t*sin
(2*Pi*t)]},t=0..10,shading=none,numpoints=500,style=
line,axes=boxed);
```

3.4 隐函数作图

```
> with(plots):
  eqn:=x^2+y^2=1;
  sol:=solve(eqn,x);
  plot([sol],y=-1..1,scaling=constrained);
```

```
> implicitplot(eqn,x=-1..1, y=-1..1, scaling=constrained);
```

```
> implicitplot((x^2+y)^2=x^2-y^2-1/60, x=-3..3, y=-3..3, grid=[100,100]);
```

```
> implicitplot3d(x^3+y^3+z^3+1=(x+y+z+ 1)^3,x=-2..2,y=-2..2,z=-2..2);
```

```
>implicitplot3d(r=(1.3)^x*sin(y),x=-1..2*Pi,y=0..Pi,r=0.1..5, coords=spherical);
```

```
> p:= proc(x,y,z) if x^2<y then x^2 + y^2 else x - y end if end proc:
  implicitplot3d(p,-2..2,-1..3,0..3);
```

3.5 等高线与密度图

```
> with(plots):
  expr:=6*x/(x^2+y^2+1);
  plot3d(expr,x=-6..6,y=-6..6,orientation=[-119,37]);
```

上面是 **expr** 的三维图，试看其密度图(**contourplot**)、等高线图(**densityplot**)：

```
> densityplot(expr,x=-6..6,y=-6..6,grid=[60,60],style=patchnogrid,axes=boxed);
```

```
> contourplot(expr,x=-6..6,y=-6..6,contours=[-2.7,-2,-1,1,2,2.7],grid=[60,60],thickness=2);
```

还可以用 **display** 将等高线图与密度图绘制在同一张图上：

```
> display(%,%%);
```

进一步，还可以为等高线图着色(用 **filled=true**)，并以 **coloring** 来指定着色的方向.

```
> contourplot(expr,x=-10..10,y=-6..6,filled=true,grid=[50,50],coloring=
  [white,red],axes=boxed);

> contourplot3d(expr, x=-6..6, y=-4..4, axes=boxed, orientation=[-124,67],
  filled=true,coloring=[navy,pink]);
```

3.6 对数作图

对数作图主要有三种情形：logplot(线性-对数)、loglogplot(对数-对数)、semilogplot(对数-线性).

```
> with(plots):
  logplot(x^2-x+4,x=1..12);

> loglogplot(x^2-x+4,x=1..12);

> semilogplot(x^2-x+4,x=1..12);

> loglogplot([cos(2*t)^2+3,sin(t^2)^2+1,t=0..3]);
```

3.7 高级作图指令

3.7.1 在图形上加上文字

textplot 和 **textplot3d** 指令可以分别在二维与三维图形上加上文字，其默认方式是文字居中对齐，如果想要改变对齐方式，可以利用 **align=direction** 来设定，**direction** 选项可以是 BELOW、ROGHT、ABOVE、LEFT 中的任一种，或是其中几种的组合.

```
> with(plots):
  g1:=textplot([3,0.2,"sin(2*x)/(x^2+1)"],align={right,above}):
  g2:=plot(sin(2*x)/(x^2+1),x=-6..6):
  display(g1,g2);

> textplot3d([[1,2,3,"My plot3d"],[1,-1.1,1,"z=sin(2*x+y)"]],color=blue,axes=frame):
  plot3d(sin(2*x+y),x=-1..2,y=-1..2):
  display(%,%%,orientation=[159,47]);
```

3.7.2 根轨迹作图

根轨迹图(**root locus plot**)是控制学上相当重要的一个部分，许多系统的特性(如稳定度(**stability**)均可从根轨迹图上显示出来. 所谓根轨迹图，也就是调整转换函数(**transfer function**)的特性方程式的某项系数，在复数平面上画出特性方程式的根变化情形(可能有实数根或共轭复数根).

```
> with(plots):
  rootlocus((s^5-s^3+2)/(s^2+1),s,-6..12,style=point);

> rootlocus((s^6+s^3+2)/(s^2+1),s,-6..12);

> rootlocus((s^2+2*s+2)/(s-1),s,-10..10);
```

3.7.3 向量场与梯度向量场的作图

向量场(**vector field**)与梯度向量场(**gradient vector field**)的概念常用来描述电磁学中的电磁场，或者是流体力学中的流场.

```
> with(plots):
  fieldplot([sin(2*x*y),cos(2*x-y)],x=-2..2,y=-2..2,arrows=SLIM,axes=boxed,
  grid=[30,30]);

> fieldplot3d([sin(2*x*y),cos(2*x-y),sin(z)],x=-2..2,
  y=-2..2,z=0..2,arrows=SLIM,axes=frame,grid=[12,12,6]);

> fieldplot3d([(x,y,z)->2*x,(x,y,z)->2*y,(x,y,z)->1],-1..1,-1..1,-1..1,axes=boxed);

> gradplot(sin(x)*cos(y),x=-2..2,y=-2..2,arrows=SLIM,axes=boxed);

> gradplot3d(z*sin(x)+cos(y),x=-Pi..Pi,y=-Pi..Pi,z=0..2,
  arrows=SLIM,axes=boxed,grid=[6,6,6]);
```

4)复数作图

二维的复数作图 **complexplot** 是以 x 轴为实轴，以 y 轴为虚数轴来作图，而三维的复数作图 **complexplot3d** 则是以 x、y 轴所组成的平面为复数平面，z 轴为虚数轴来作图.

```
> with(plots):
  complexplot(x+x*I,x=0..8);
```

```
> complexplot(sinh(3+x*I),x=-Pi..Pi,scaling=constrained);

> complexplot3d(sech(z),z=-2-3*I..2+3*I,axes=frame);

> complexplot3d(GAMMA(z),z=-2.5-2*I..4+2*I,view=0..6,
  grid=[35,33], linestyle=2, orientation=[-132,76], axes=frame);

> complexplot([1+2*I, 3-4*I, 5+6*I, 7-8*I], x=0..12,style=point);
```

5)复数映射绘图

复数映射作图命令 conformal(f(z), range)是以 f(z)为映射函数, 按 range 所指定的范围映射到另一个复数平面.

```
> with(plots):
  conformal(sin(z),z=-Pi/2-1.5*I..Pi/2+1.5*I);

> conformal(tan(z),z=-Pi/4-I..Pi/4+I);

> conformal(1/z,z=-1-I..1+I,-6-6*I..6+6*I, color=magenta);

> conformal((z-I)/(z+I),z=-3-3*I..3+3*I,-4-4*I..4+4*I,grid=[30,30],style=LINE);

> conformal3d(sin(z),z=0..2*Pi+I*Pi);
```

6)圆管作图

```
> with(plots):
> tubeplot([2+t*cos(t),2+t*sin(t),t],t=0..5.6*Pi,radius=4,grid=[124,16]);

> tubeplot([3*sin(t),t,3*cos(t)],t=-3*Pi..4*Pi,radius=1.2+sin(t),numpoints=80);

> tubeplot({[cos(t),sin(t),0],[0,sin(t)-1,cos(t)]},t=0..2*Pi,radius=1/4);

> tubeplot({[cos(t),sin(t),0],[0,sin(t)-1,cos(t)]},t=0..2*Pi,radius=1/10*t);
```

在 Maple 的三维绘图中，我们甚至于可以使用一个程序或一个二元算子定义艳丽的色彩：

```
> F:=(x,y)->sin(x):
  tubeplot({[cos(t),sin(t),0],[0,sin(t)-1,cos(t)]},t=0..2*Pi,radius=1/4,color=F,style=patch);
```

7)曲面数据作图

```
> with(plots):
  pts:=[[[0,0,3],[0,1,3],[0,2,4]],[[1,0,4],[1,1,5],[1,2,5]],[[2,0,4],[2,1,5],[2,2,6]]]:
  surfdata (pts, labels=["x","y","z"], orientation=[-123,45], axes=boxed,
  tickmarks=[3,3,3]);
```

```
> pts:=seq([seq([x/2, y/2, -x*y/(x^2+y^2+1)], y=-8..8)], x=-8..8):
  surfdata([pts],axes=frame,orientation=[-60,-100]);
```

```
> cosdata :=[seq([ seq([i,j,evalf(cos((i+j)/2))], i=-5..5)], j=-5..5)]:
  sindata :=[seq([ seq([i,j,evalf(sin((i+j)/2))], i=-5..5)], j=-5..5)]:
  surfdata( {sindata,cosdata}, axes=frame, labels=[x,y,z],orientation=[-35,80] );
```

8) 多边形和多面体绘制

```
> with(plots):
> ngon:=n->[seq([ cos(2*Pi*i/n), sin(2*Pi*i/n) ], i = 1..n)]:
  display([polygonplot(ngon(8)), textplot([0,0,`Octagon`]) ], color=pink);
```

```
> head:=[0,0],[-10,0],[-18,6],[-18,14],[-14,17],[-14,24],[-10,20],[0,20],[10,20],[14,24],
  [14,17], [18,14],[18,6],[10,0]:
  leye:=[-10,14],[-7,12],[-10,10],[-13,12]:
  reye:=[10,14],[7,12],[10,10],[13,12]:
  koko:=[-0.5,7.5],[0.5,7.5],[0,8.5]:
  polygonplot({[head],[leye],[reye],[koko]},axes=NONE);
```

```
> polyhedraplot([0,0,0],polyscale=0.6,polytype=hexahedron,scaling=CONSTRAINED,
  orientation=[-30,70]);
```

```
> polyhedraplot([0,0,0],polytype=octahedron);
```

```
>polyhedraplot([0,0,0],polytype=dodecahedron,style=PATCH, scaling=CONSTRAINED,
```

orientation=[-60,60],axes=boxed);

> **polyhedraplot([0,0,0],polyscale=0.6,polytype=icosahedron);**

> polyhedraplot([0,0,0],polytype=TriakisIcosahedron,style=PATCH,scaling=CONSTRAINED,orientation=[71,66]);

4 动 画

Maple 具有动画功能，存于 plots 库中的动画函数分别为 animate 和 animate3d. 要创建一个动画，必须在所需做动画的函数中加入附加参数(时间参数)并简单地告诉 animate 或 animate3d 函数需要多少次以及在那个时间内计算曲面，动画函数就可以足够快地播放图形的时间序列，以创建运动的现象. 其命令格式分别如下：

animate (F, x, t);

animate3d (F,x,y,t);

其中，F—要绘图的函数，x, y—横轴、纵轴的变化范围，t—结构参数的变化范围

> with(plots):

> animate(cos(3*t)*sin(3*x),x=0..2*Pi,t=0..2*Pi,frames=100,color=red,scaling=constrained);

> **animate([u*sin(t),u*cos(t),t=-Pi..Pi],u=1..8,view=[-8..8,-8..8]);**

> **s:=t->100/(100+(t-Pi/2)^8): r:=t->s(t)*(2-sin(7*t)-cos(30*t)/2):**
animate([u*r(t)/2,t,t=-Pi/2..3/2*Pi],u=1..2,numpoints=200,coords=polar,axes=none,
color=green);

> **animate3d(x*cos(t*u),x=1..3,t=1..4,u=2..4,coords=spherical);**

> **animate3d(sin(x)*cos(t*u),x=1..3,t=1..4,u=1/4..7/2,coords=cylindrical);**

> **animate3d([x*u,u*t,x*cos(t*u)],x=1..3,t=1..4,u=2..4,coords=cylindrical);**

> **animate3d(cos(3*t)*sin(3*x)*cos(3*y),x=0..Pi,y=0..Pi,t=0..2*Pi,frames=100,**
color=cos(x*y), scaling=constrained);

> **p:=seq(plot([[0,0],[sin(2*Pi*t/100),cos(2*Pi*t/100)]]),t=0..100):**
display([p],insequence=true,scaling=constrained,axes=none);

再看一个更复杂的动画例子—摆线的运动动画：

```
> restart; with(plots):
> revolutions:=omega*t/(2*Pi):
> translation:=[2*Pi*r*revolutions,0]:
> rotation:=[r*sin(omega*t),r*cos(omega*t)]:
> cycloid:=translation+rotation+[0,r]:
> omega:=1:r:=1:rollTime:=4*Pi:
> cycloidTrace:=animatecurve([cycloid[1],cycloid[2],t=0..rollTime],
      view=[0..r*omega*rollTime,0..4*r],scaling=constrained,color=blue,frames=100):
> disk:=animate([translation[1]+r*cos(s),r+r*sin(s),s=0..2*Pi],t=0..rollTime,
      scaling=constrained,frames=100,view=[0..r*omega*rollTime,0..4*r]):
> chord:=animate([translation[1]+(s/r)*rotation[1],(s/r)*rotation[2]+r,s=-r..r],t=0..
      rollTime,scaling=constrained,color=blue,frames=100,view=[0..r*omega*rollTime,0..4*r]):
> display([cycloidTrace,disk,chord],title=`Animation of a Cycloid`);
```