

Mathematica功能介绍之一

数值计算与赋值

```
(3 + 5 - 6) * 9 / 10
```

$$\frac{9}{5}$$

```
N[%] (*上述结果的小数表示或近似值*)
```

1.8

```
3^100  
% // N
```

515377520732011331036461129765621272702107522001

5.15378×10^{47}

```
Sqrt[5]  
N[%]
```

$$\sqrt{5}$$

2.23607

```
N[Pi, 100]  
N[E, 30]
```

3.1415926535897932384626433832795028841971693993751058209749445923078:
16406286208998628034825342117068

2.71828182845904523536028747135

```
(1 + 3 I) ^ 10
```

```
99712 - 7584 i
```

```
a = 100 (*赋值*)  
a
```

```
100
```

```
100
```

```
Clear[a] (*清除变量的值*)  
a
```

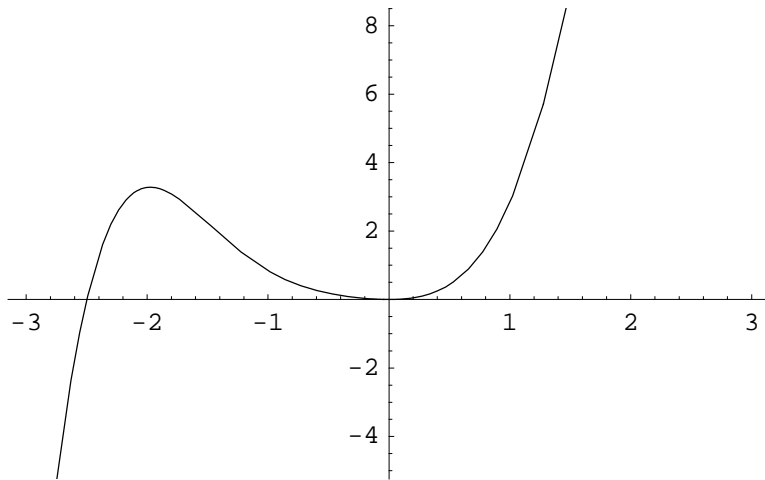
```
a
```

自定义函数

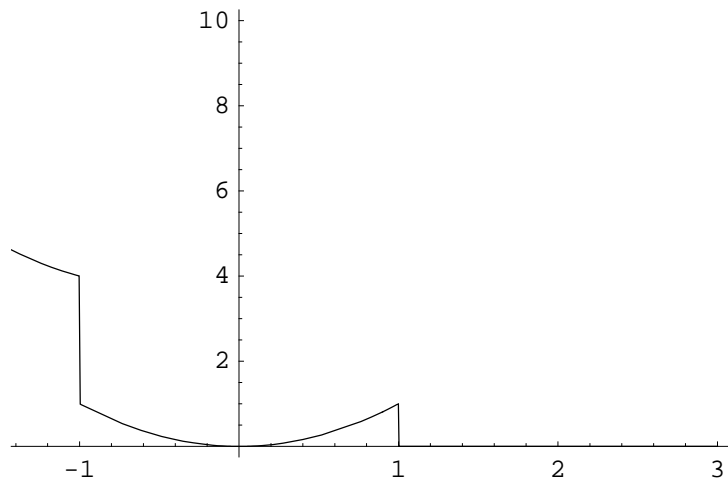
```
(*定义函数*)  
f[x_] := x^3 * (Sin[x] + 1) + x^2
```

```
(*两种方法定义分段函数*)  
g[x_] := Which[x ≤ -1, x^2 + x + 4, x ≤ 1, x^2, x > 1, 0]  
h[x_] := If[x ≤ -1, x^2 + x + 4, If[x ≤ 1, x^2, 0]]
```

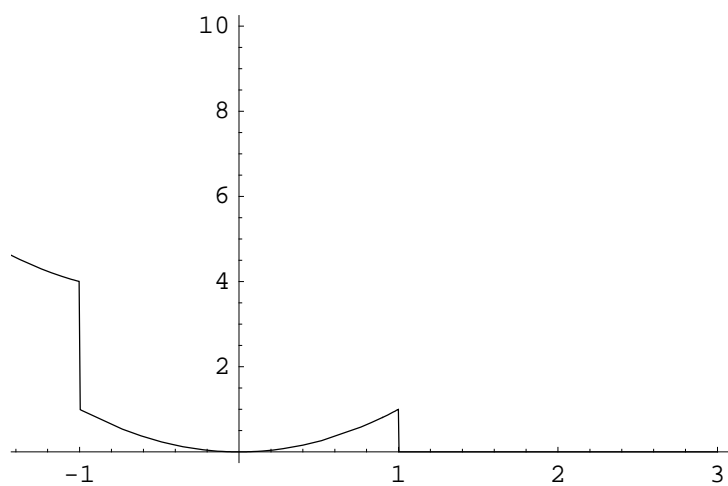
```
(*观察图形*)  
Plot[f[x], {x, -3, 3}]  
Plot[g[x], {x, -3, 3}]  
Plot[h[x], {x, -3, 3}]
```



- Graphics -



- Graphics -

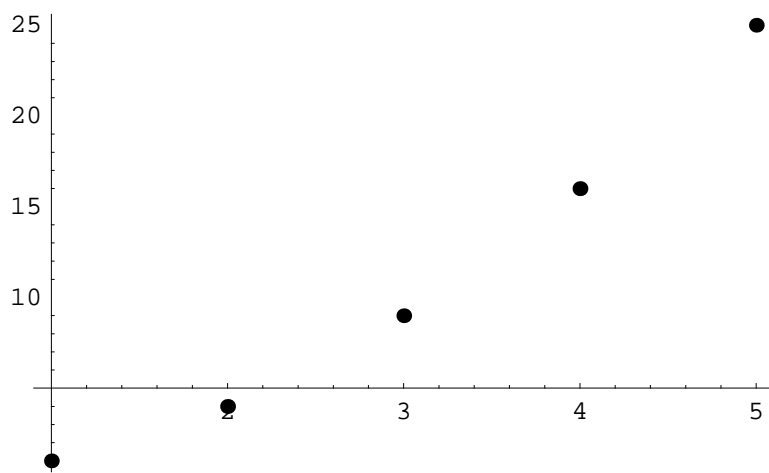


- Graphics -

数集与点图

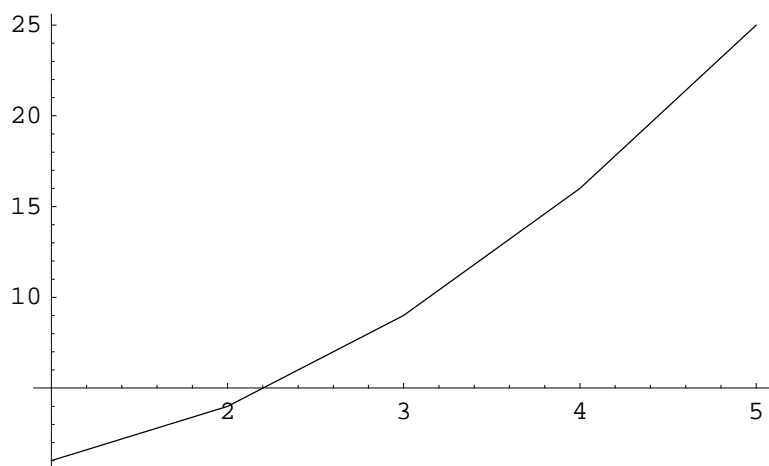
```
data1 = Table[n^2, {n, 1, 5, 1}];  
(*构成数集, 并储存在变量data1中*)
```

```
ListPlot[data1] (*作出点图*)
```



- Graphics -

```
ListPlot[data1, PlotJoined → True]  
(*作出点图并用直线连接相邻点*)
```

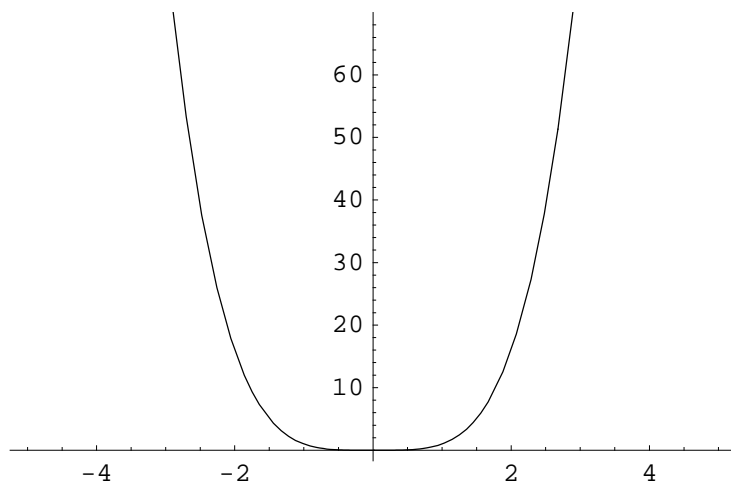


- Graphics -

平面图形

(*作出平面图形*)

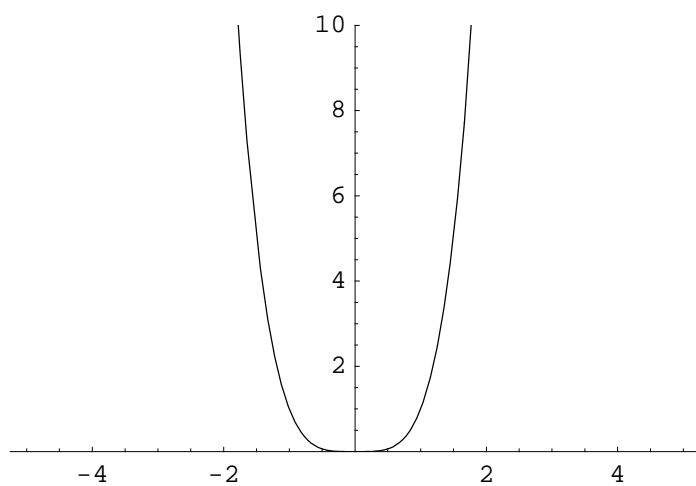
```
Plot[x^4, {x, -5, 5}]
```



- Graphics -

(*作出平面图形, 而且限制因变量显示范围*)

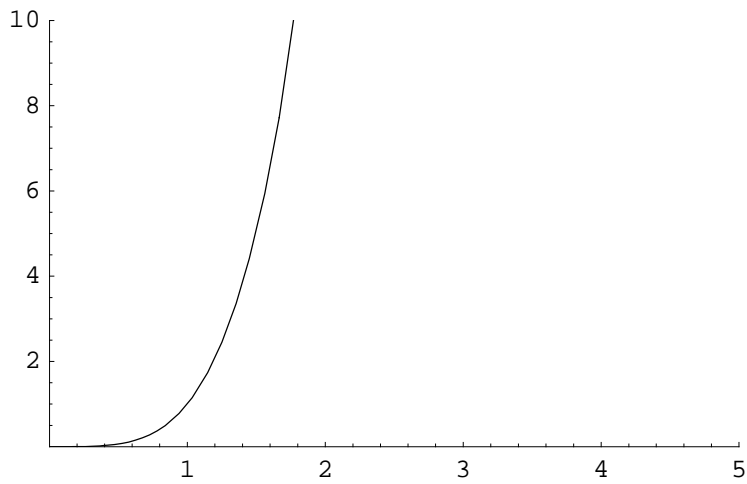
```
Plot[x^4, {x, -5, 5}, PlotRange -> {0, 10}]
```



- Graphics -

(*作出平面图形, 而且同时限制自变量和因变量显示范围*)

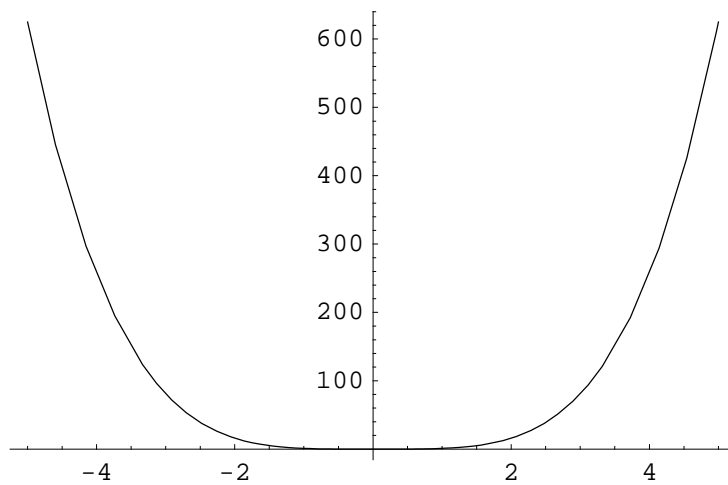
```
Plot[x^4, {x, -5, 5}, PlotRange -> {{0, 5}, {0, 10}}]
```



- Graphics -

(*作出平面图形, 而且显示全部图形*)

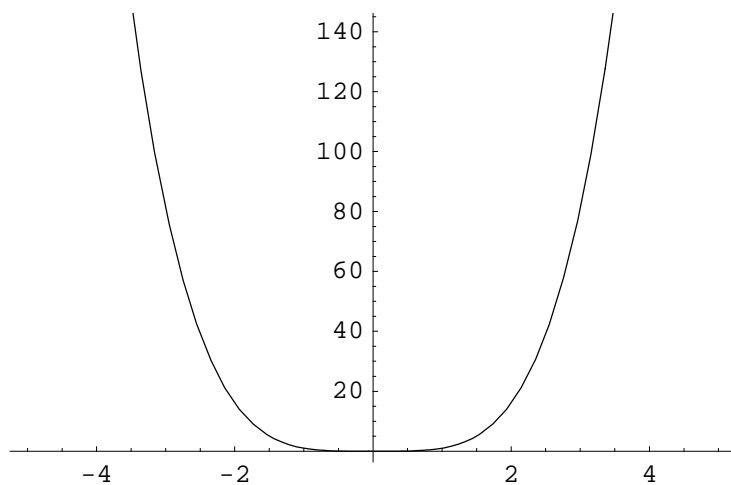
```
Plot[x^4, {x, -5, 5}, PlotRange -> All]
```



- Graphics -

(*作出平面图形, 而且规定采样点数*)

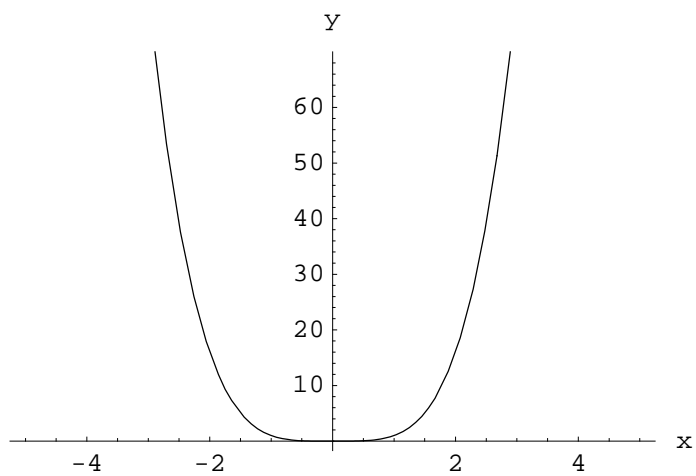
```
Plot[x^4, {x, -5, 5}, PlotPoints -> 50]
```



- Graphics -

(*作出图形, 而且指定坐标轴的名称*)

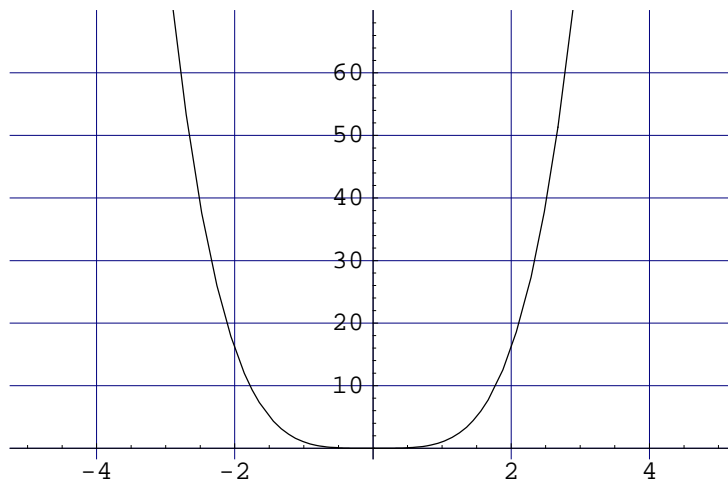
```
Plot[x^4, {x, -5, 5}, AxesLabel -> {"x", "y"}]
```



- Graphics -

(*作出图形, 而且自动给出标志线*)

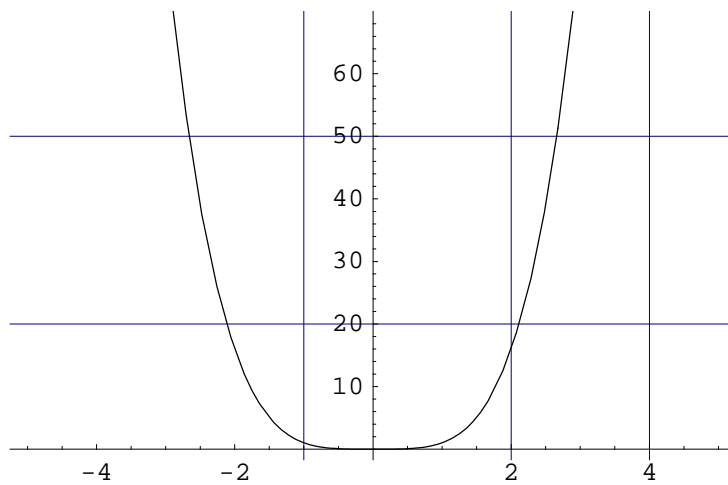
```
Plot[x^4, {x, -5, 5}, GridLines -> Automatic]
```



- Graphics -

(*作出图形, 而且按规定给出标志线*)

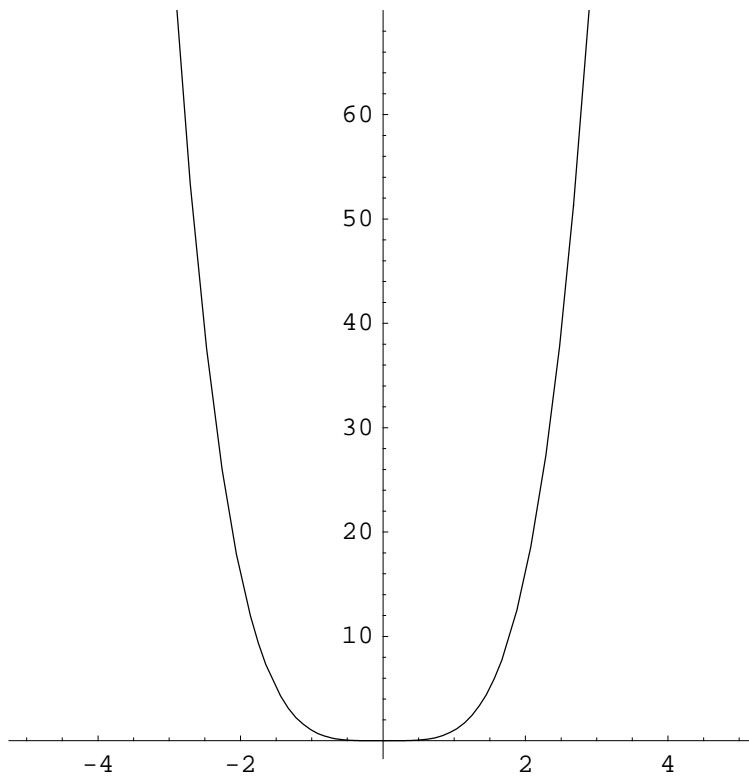
```
Plot[x^4, {x, -5, 5}, GridLines -> {{-1, 2, 4}, {20, 50, 100}}]
```



- Graphics -

(*作出图形, 而且指定图形的高与宽的比例*)

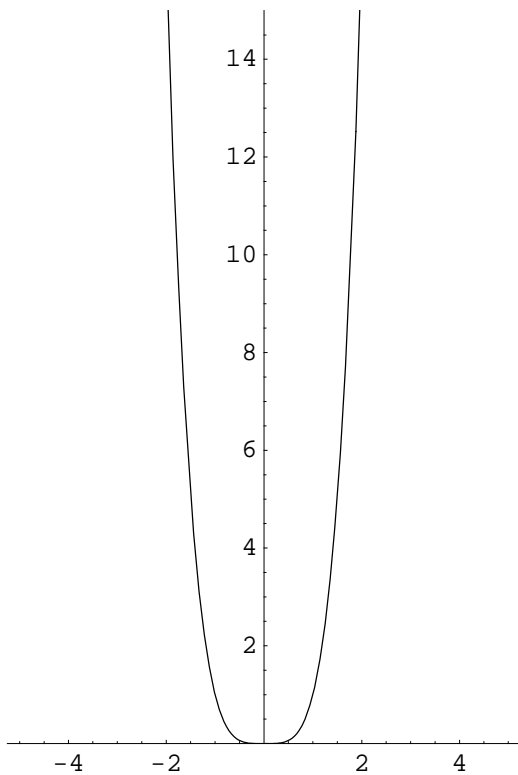
```
Plot[x^4, {x, -5, 5}, AspectRatio -> 1]
```



- Graphics -

(*作出图形, 按实际图形确定高与宽的比例*)

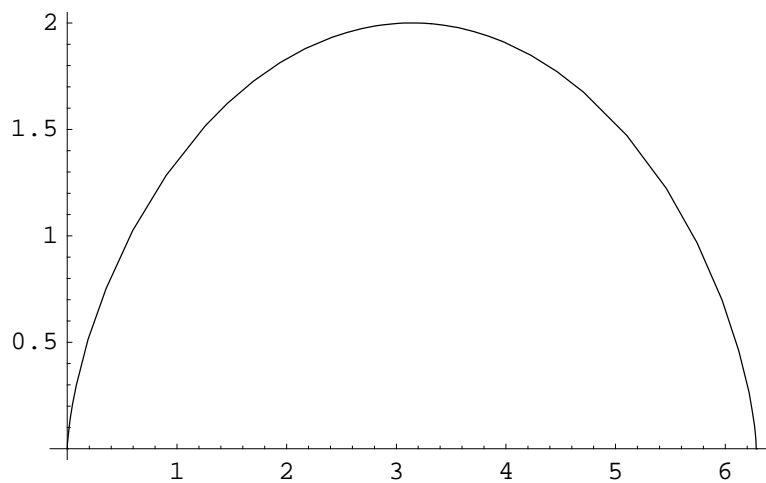
```
Plot[x^4, {x, -5, 5}, AspectRatio -> Automatic, PlotRange -> {0, 15}]
```



- Graphics -

(*用参数方程作出图形*)

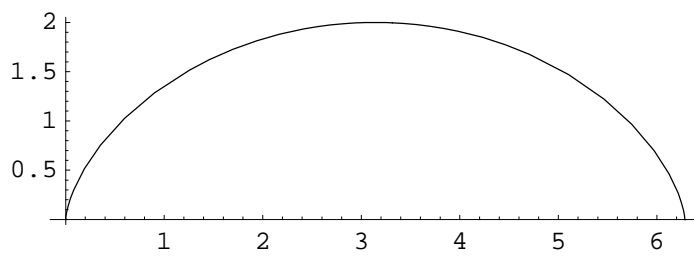
```
ParametricPlot[{t - Sin[t], 1 - Cos[t]}, {t, 0, 2 Pi}]
```



- Graphics -

(*用参数方程作出图形, 并且按实际图形确定高与宽的比例*)

```
ParametricPlot[{t - Sin[t], 1 - Cos[t]}, {t, 0, 2 Pi}, AspectRatio -> Automatic]
```



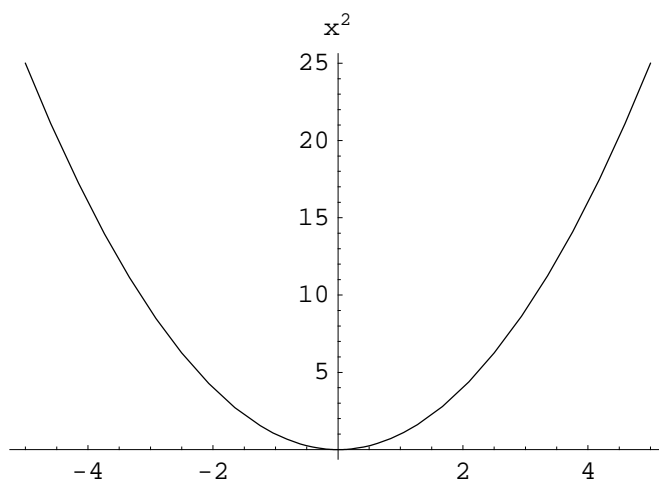
- Graphics -

Mathematica功能介绍之二

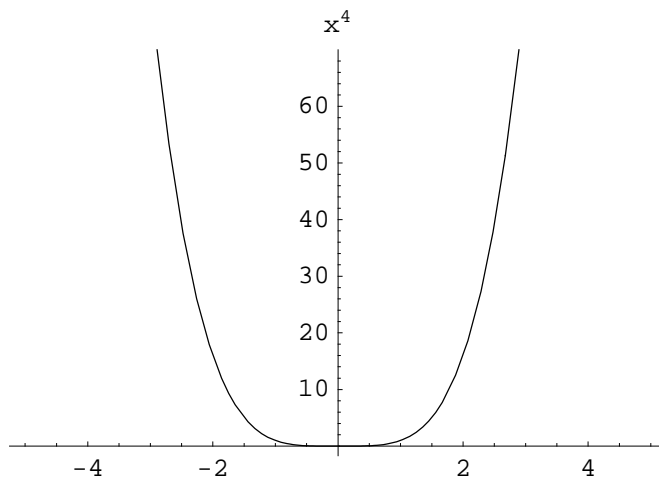
基本初等函数的图形

幂函数

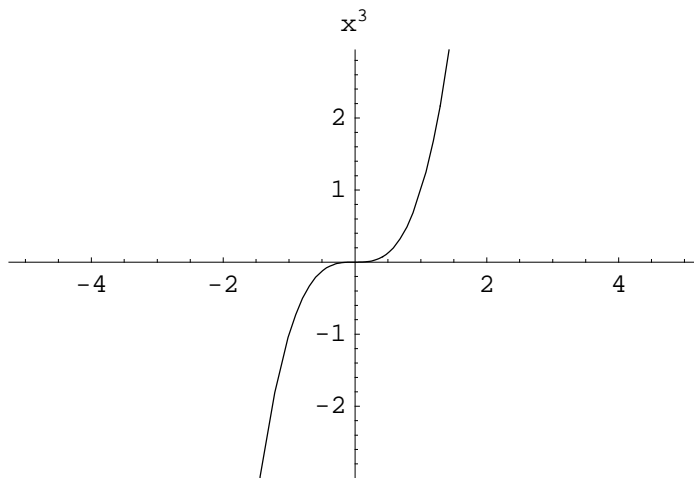
```
Plot[x^2, {x, -5, 5}, PlotLabel -> x^2]  
Plot[x^4, {x, -5, 5}, PlotLabel -> x^4]  
Plot[x^3, {x, -5, 5}, PlotLabel -> x^3]  
Plot[x^5, {x, -5, 5}, PlotLabel -> x^5]  
Plot[{x^2, x^4}, {x, -5, 5}]  
Plot[{x^3, x^5}, {x, -5, 5}]
```



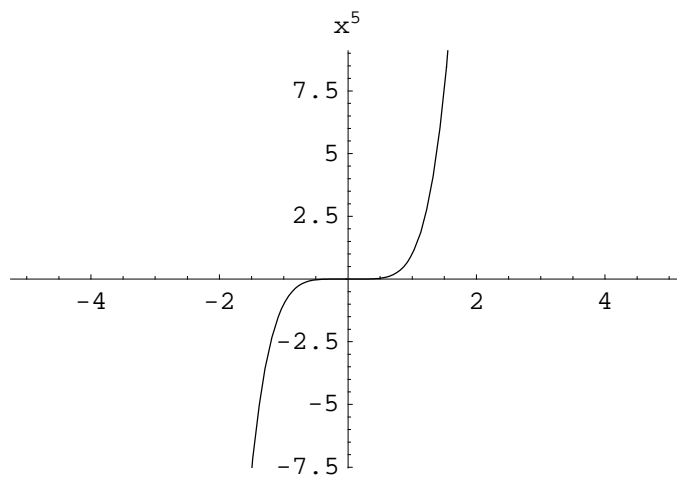
- Graphics -



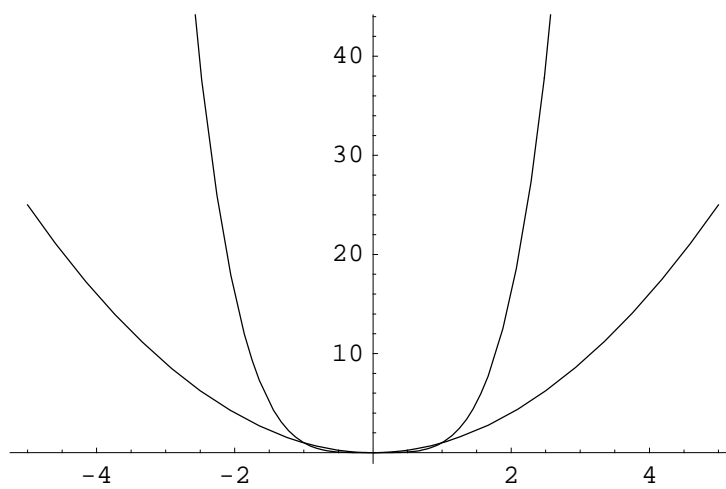
- Graphics -



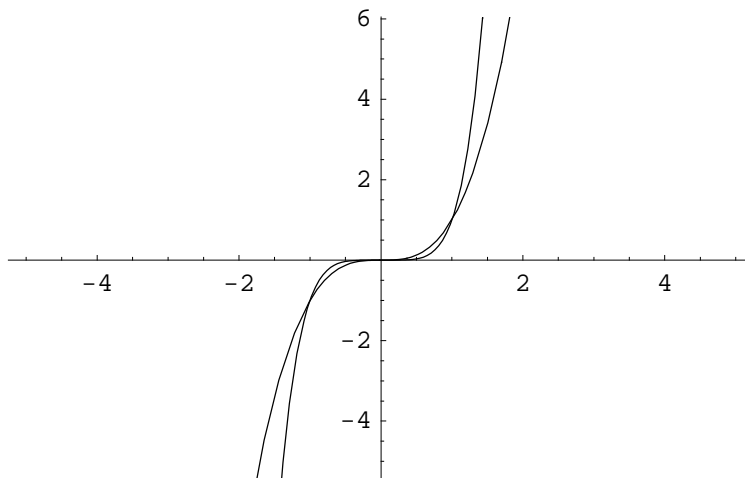
- Graphics -



- Graphics -



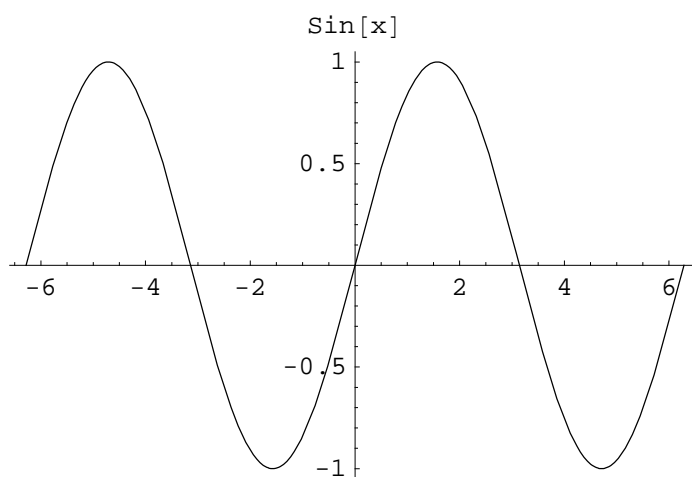
- Graphics -



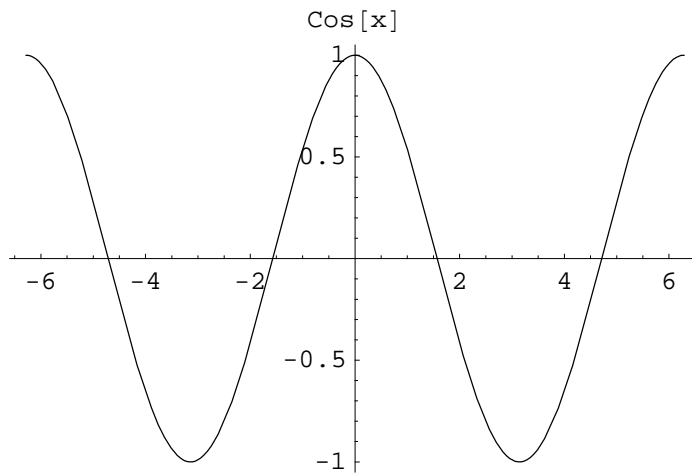
- Graphics -

三角函数

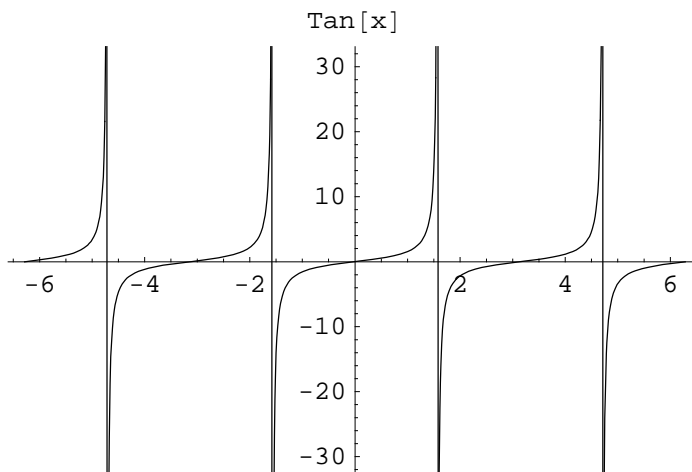
```
Plot[Sin[x], {x, -2 Pi, 2 Pi}, PlotLabel -> Sin[x]]  
Plot[Cos[x], {x, -2 Pi, 2 Pi}, PlotLabel -> Cos[x]]  
Plot[Tan[x], {x, -2 Pi, 2 Pi}, PlotLabel -> Tan[x]]  
Plot[Cot[x], {x, -2 Pi, 2 Pi}, PlotLabel -> Cot[x]]  
Plot[Sec[x], {x, -2 Pi, 2 Pi}, PlotLabel -> Sec[x]]  
Plot[Csc[x], {x, -2 Pi, 2 Pi}, PlotLabel -> Csc[x]]
```



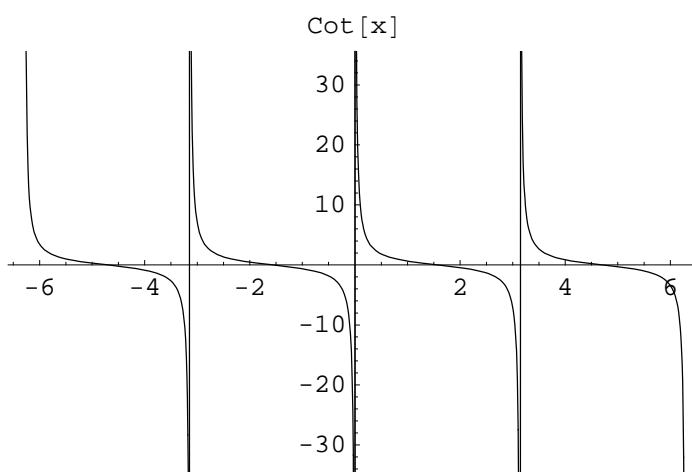
- Graphics -



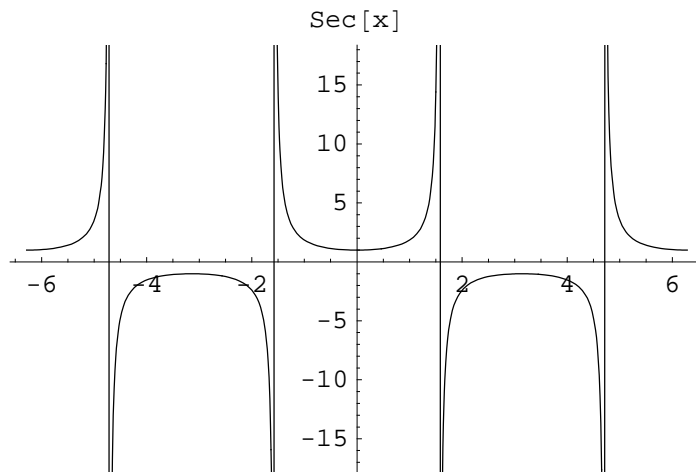
- Graphics -



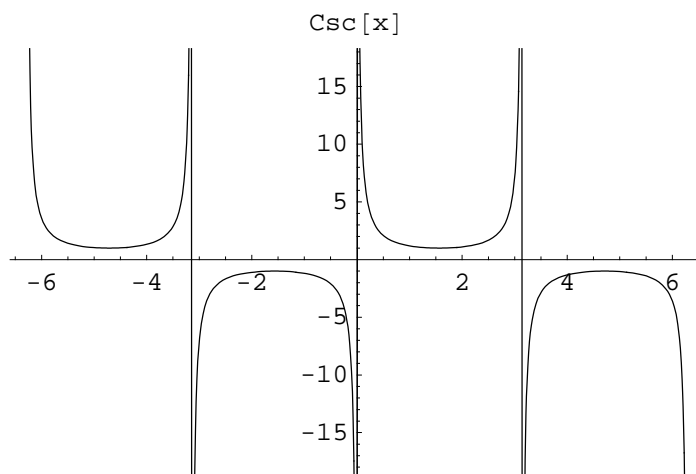
- Graphics -



- Graphics -



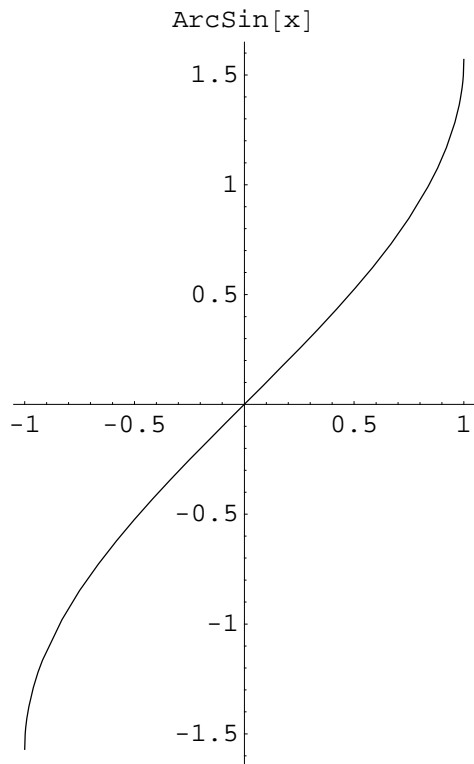
- Graphics -



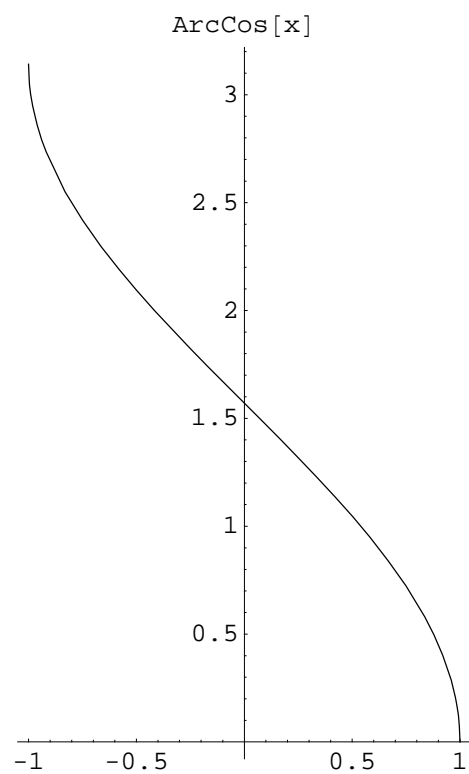
- Graphics -

反三角函数

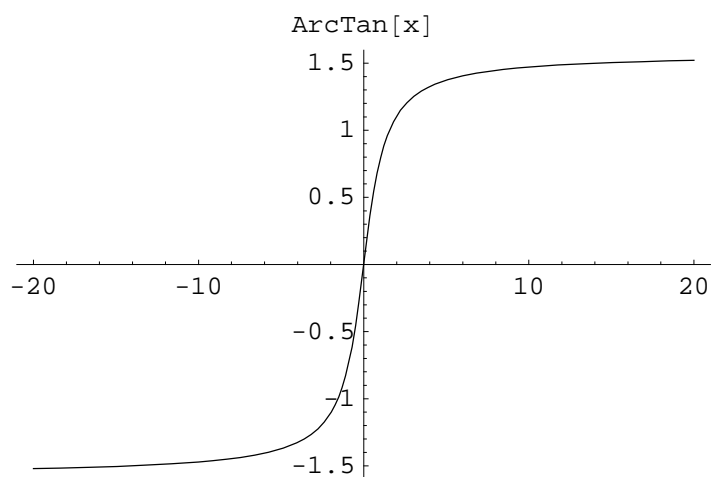
```
Plot[ArcSin[x], {x, -1, 1}, PlotLabel -> ArcSin[x], AspectRatio -> Automatic]  
Plot[ArcCos[x], {x, -1, 1}, PlotLabel -> ArcCos[x], AspectRatio -> Automatic]  
Plot[ArcTan[x], {x, -20, 20}, PlotLabel -> ArcTan[x]]  
Plot[ArcCot[x], {x, -20, 20}, PlotLabel -> ArcCot[x]]
```



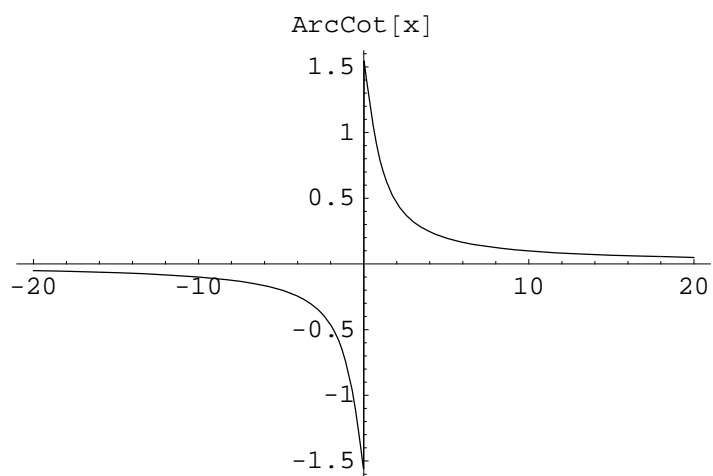
- Graphics -



- Graphics -



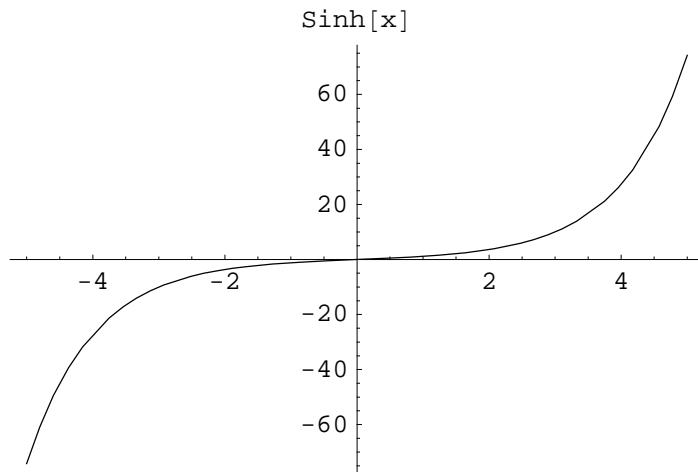
- Graphics -



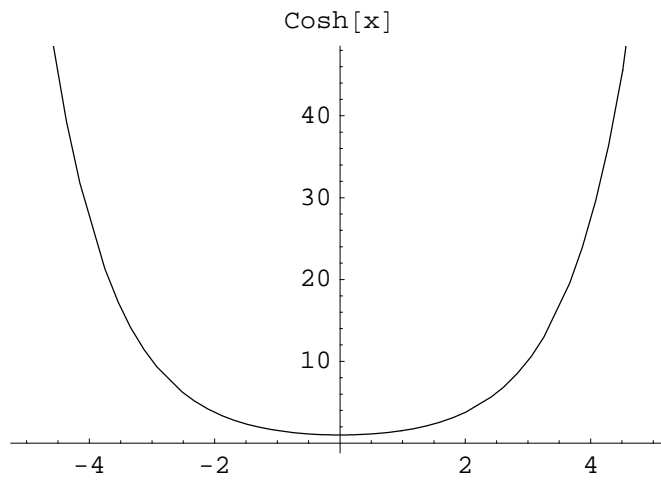
- Graphics -

双曲函数

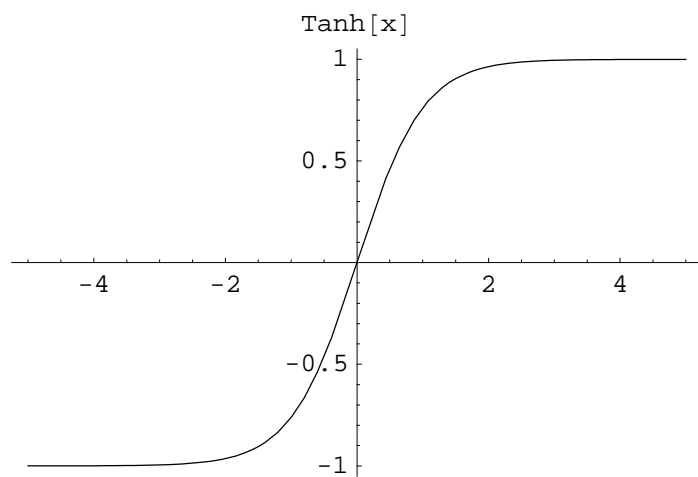
```
Plot[Sinh[x], {x, -5, 5}, PlotLabel -> Sinh[x]]  
Plot[Cosh[x], {x, -5, 5}, PlotLabel -> Cosh[x]]  
Plot[Tanh[x], {x, -5, 5}, PlotLabel -> Tanh[x]]
```



- Graphics -



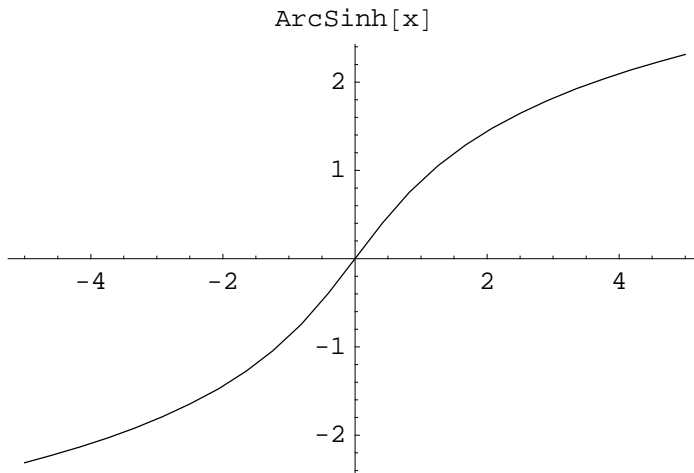
- Graphics -



- Graphics -

反双曲函数

```
Plot[ArcSinh[x], {x, -5, 5}, PlotLabel -> ArcSinh[x]]
Plot[ArcCosh[x], {x, -5, 5}, PlotLabel -> ArcCosh[x]]
Plot[ArcTanh[x], {x, -5, 5}, PlotLabel -> ArcTanh[x]]
```



- Graphics -

Plot::plnr :

ArcCosh[x] is not a machine-size real number at x = -5.. More...

Plot::plnr :

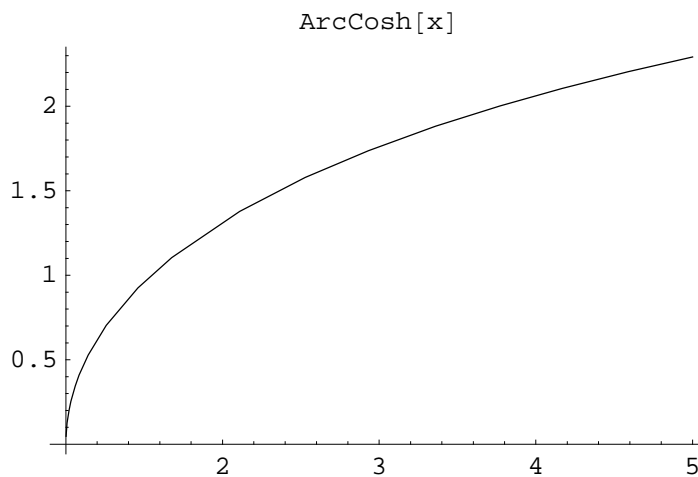
ArcCosh[x] is not a machine-size real number at x = -4.59433. More...

Plot::plnr :

ArcCosh[x] is not a machine-size real number at x = -4.15191. More...

General::stop : Further output of

Plot::plnr will be suppressed during this calculation. More...



- Graphics -

Plot::plnr :

ArcTanh[x] is not a machine-size real number at x = -5.. More...

Plot::plnr :

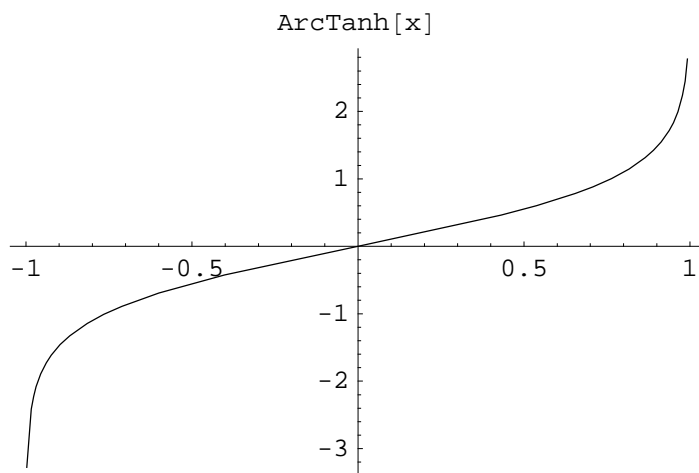
ArcTanh[x] is not a machine-size real number at x = -4.59433. More...

Plot::plnr :

ArcTanh[x] is not a machine-size real number at x = -4.15191. More...

General::stop : Further output of

Plot::plnr will be suppressed during this calculation. More...



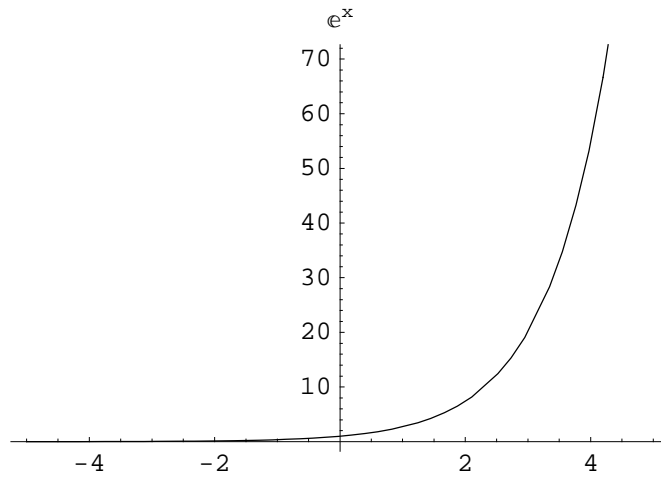
- Graphics -

指数函数

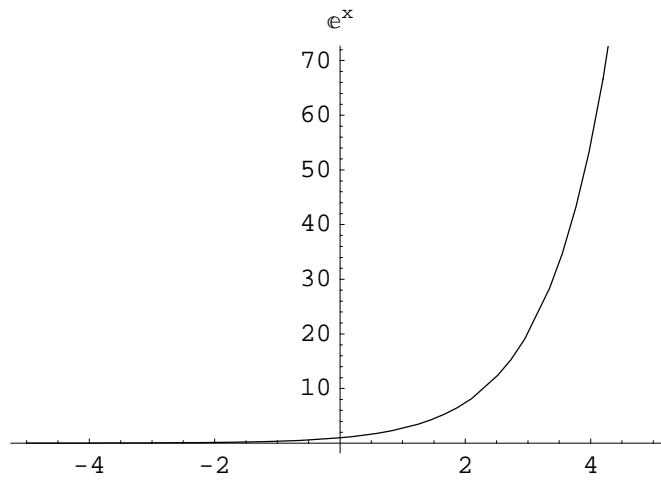
```
Plot[Exp[x], {x, -5, 5}, PlotLabel -> Exp[x]]
```

```
Plot[E^x, {x, -5, 5}, PlotLabel -> Exp[x]]
```

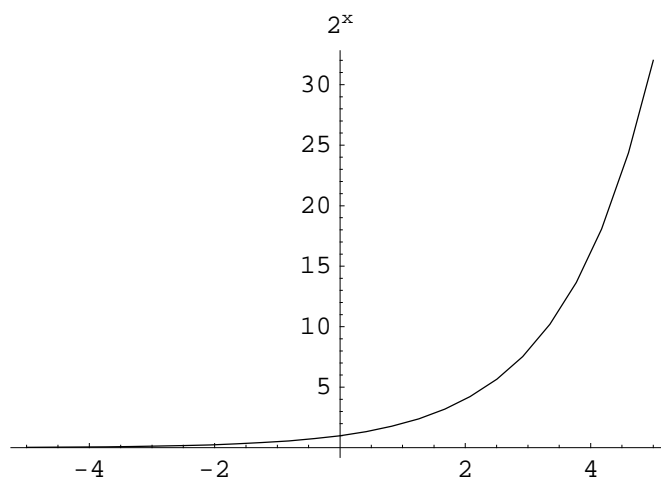
```
Plot[2^x, {x, -5, 5}, PlotLabel -> 2^x]
```



- Graphics -



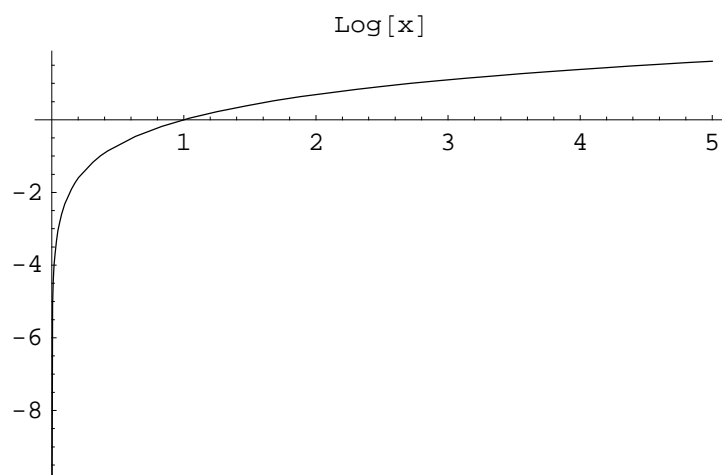
- Graphics -



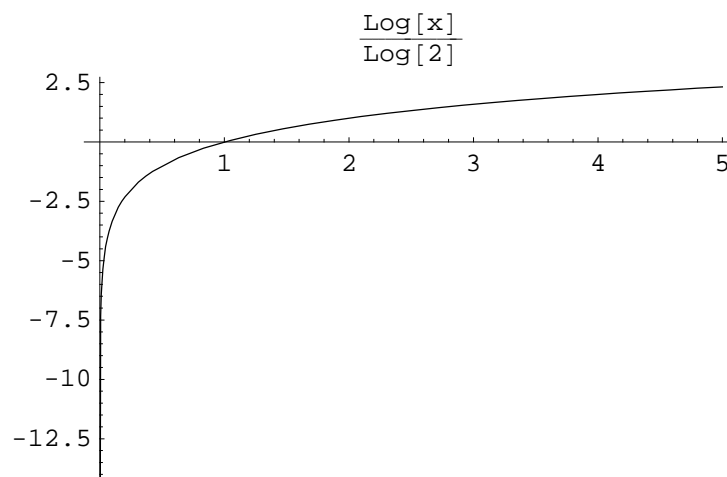
- Graphics -

对数函数

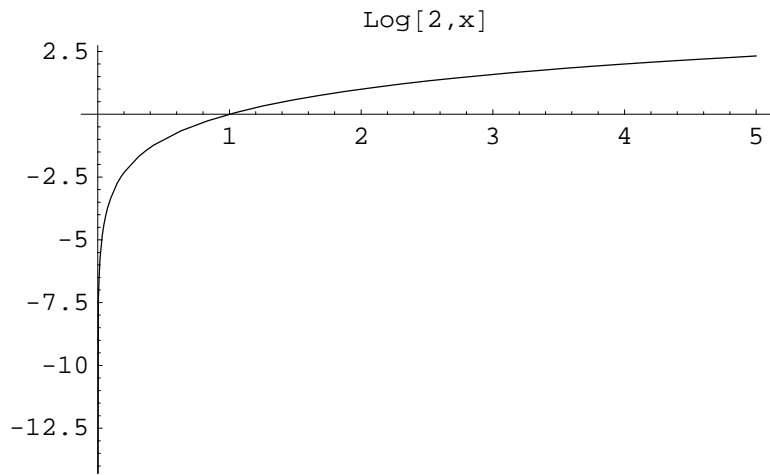
```
Plot[Log[x], {x, 0, 5}, PlotLabel -> "Log[x]"]  
Plot[Log[2, x], {x, 0, 5}, PlotLabel -> Log[2, x]]  
Plot[Log[2, x], {x, 0, 5}, PlotLabel -> "Log[2,x]"]  
Plot[{Log[x], Log[2, x]}, {x, 0, 5}]
```



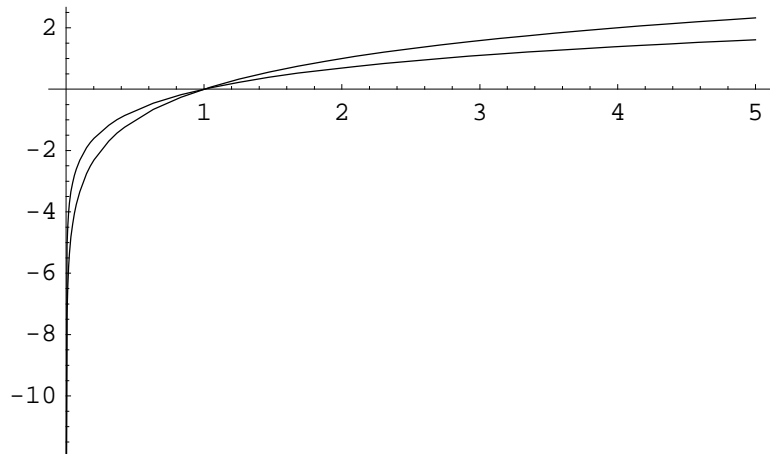
- Graphics -



- Graphics -



- Graphics -



- Graphics -

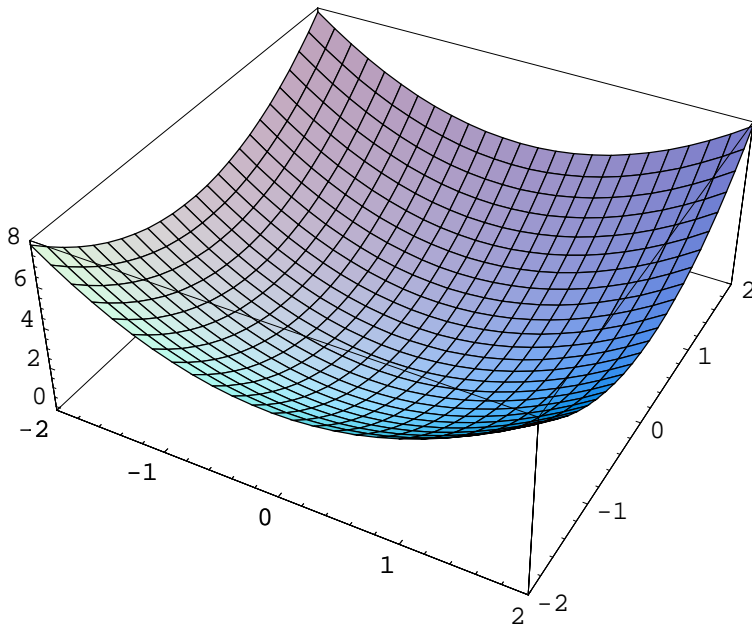
Mathematica功能介绍之三

绘制立体图形

立体图形

(*作出立体图形*)

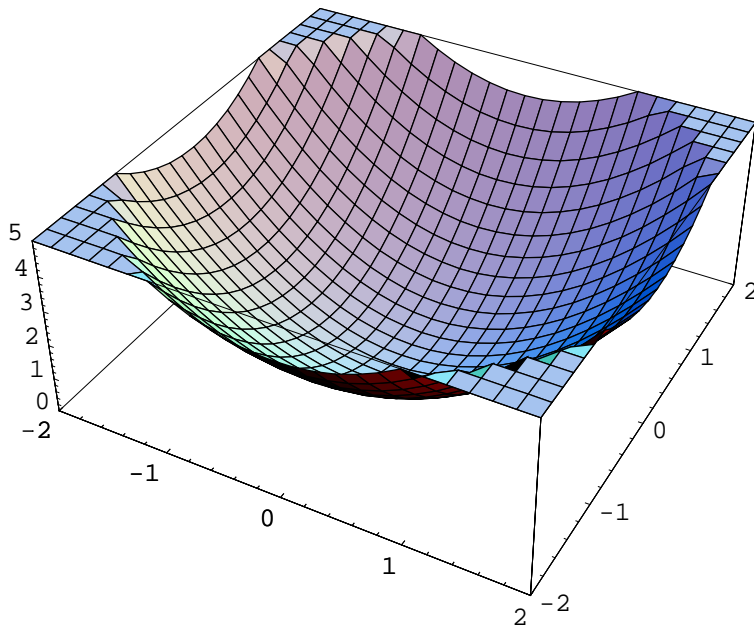
```
Plot3D[x^2 + y^2, {x, -2, 2}, {y, -2, 2}]
```



- SurfaceGraphics -

(*作出立体图形, 而且限制因变量显示范围*)

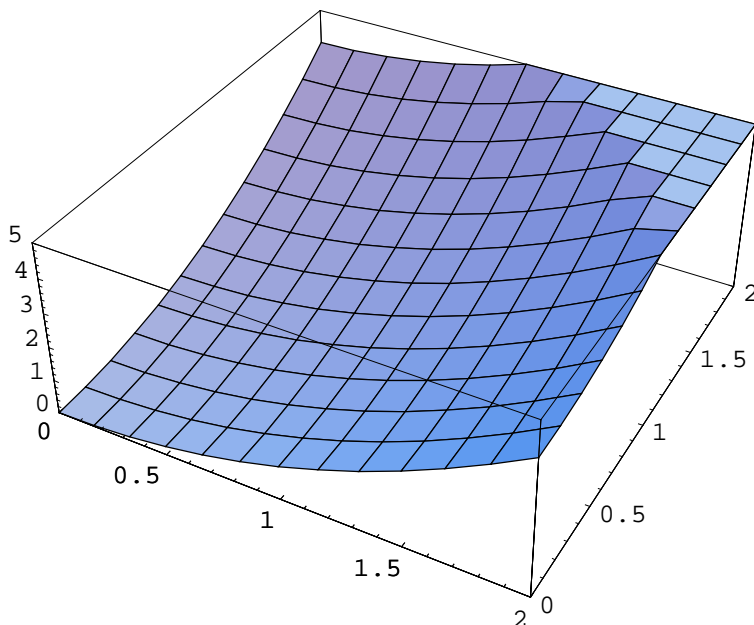
```
Plot3D[x^2 + y^2, {x, -2, 2}, {y, -2, 2}, PlotRange -> {0, 5}]
```



- SurfaceGraphics -

(*作出立体图形, 而且同时限制自变量和因变量显示范围*)

```
Plot3D[x^2 + y^2, {x, -2, 2}, {y, -2, 2}, PlotRange -> {{0, 2}, {0, 2}, {0, 5}}]
```

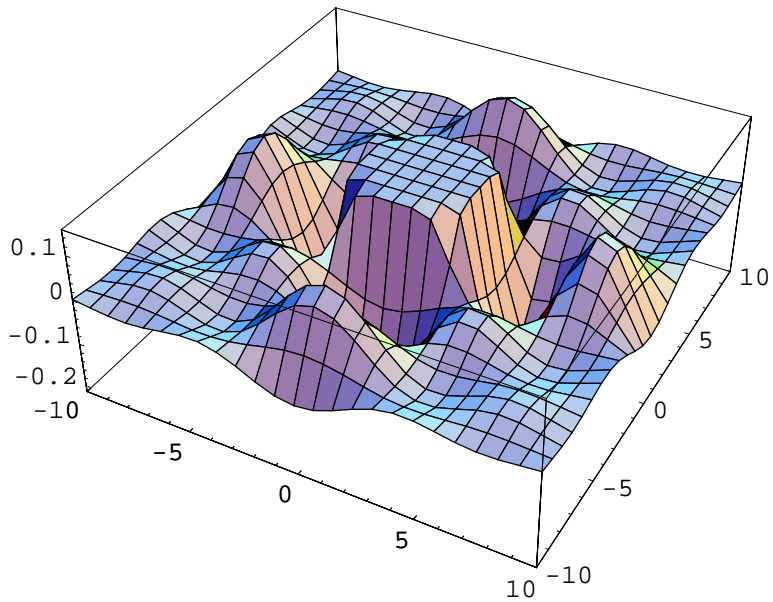


- SurfaceGraphics -

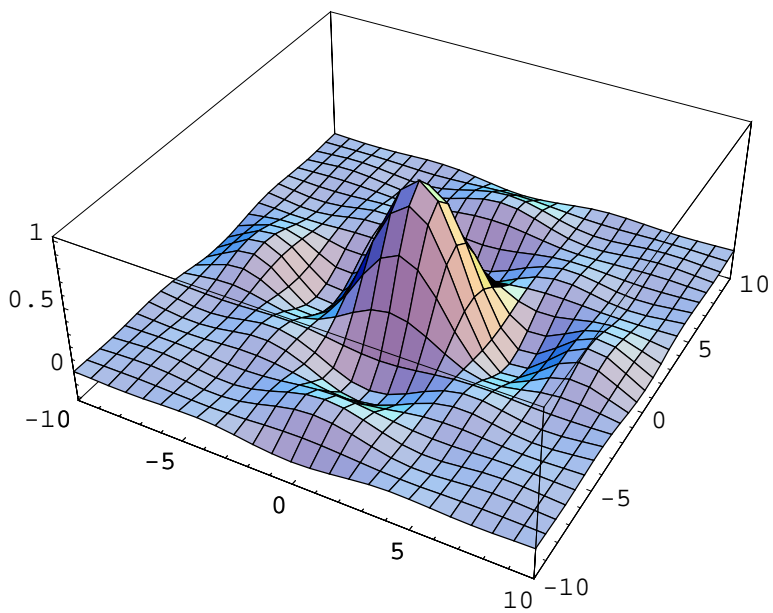
(*作出立体图形, 而且显示全部图形*)

```
Plot3D[Sin[x] / x * Sin[y] / y, {x, -10, 10}, {y, -10, 10}]
```

```
Plot3D[Sin[x] / x * Sin[y] / y, {x, -10, 10}, {y, -10, 10}, PlotRange -> All]
```



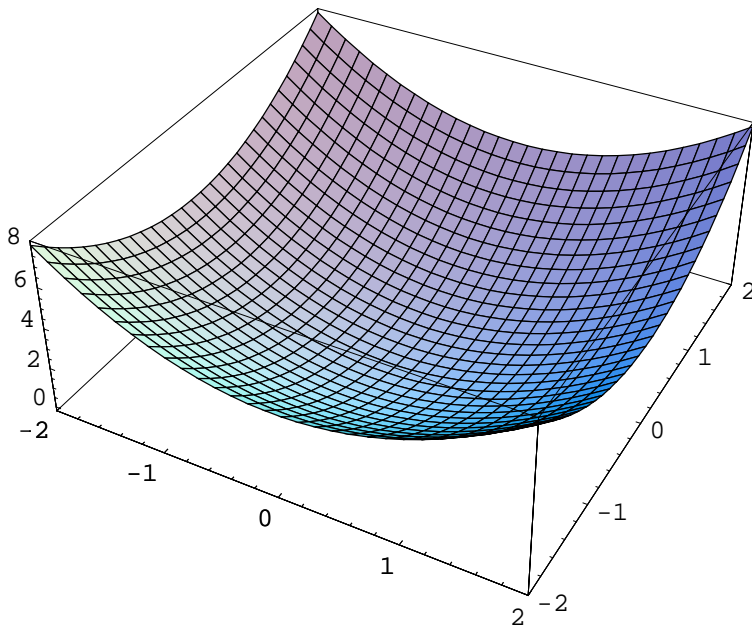
- SurfaceGraphics -



- SurfaceGraphics -

(*作出立体图形, 而且规定采样点数*)

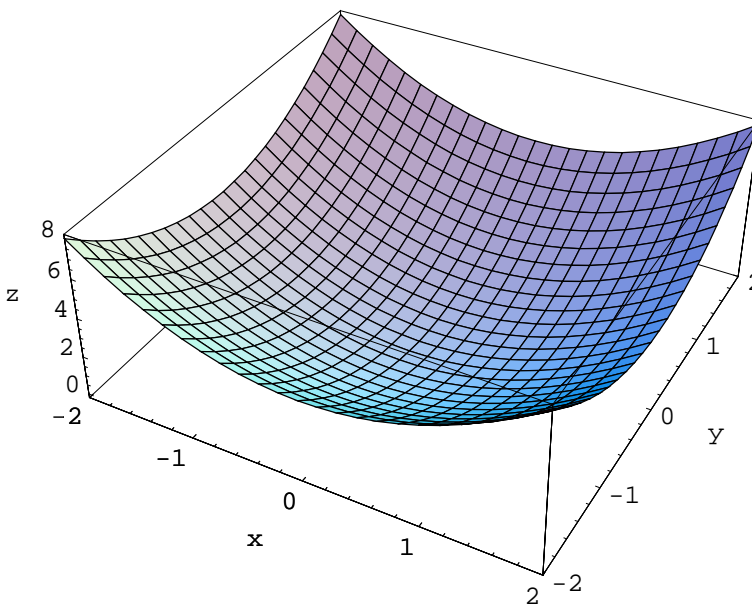
```
Plot3D[x^2+y^2, {x, -2, 2}, {y, -2, 2}, PlotPoints -> 30]
```



- SurfaceGraphics -

(*作出图形, 而且指定坐标轴的名称*)

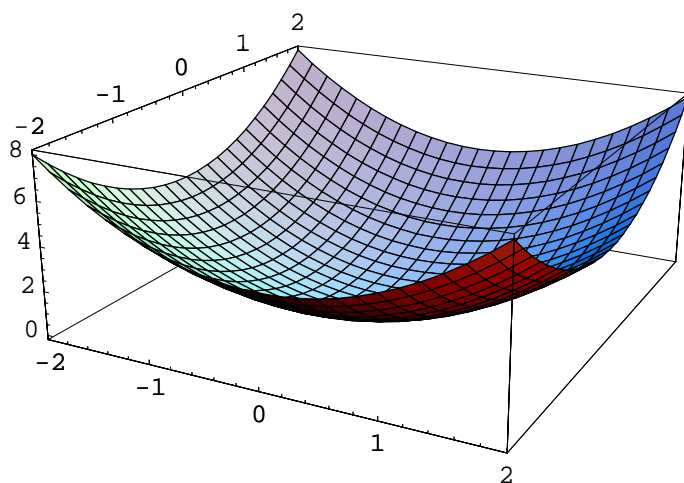
```
Plot3D[x^2+y^2, {x, -2, 2}, {y, -2, 2}, AxesLabel -> {"x", "y", "z"}]
```



- SurfaceGraphics -

(*作出图形, 而且规定观察点*)

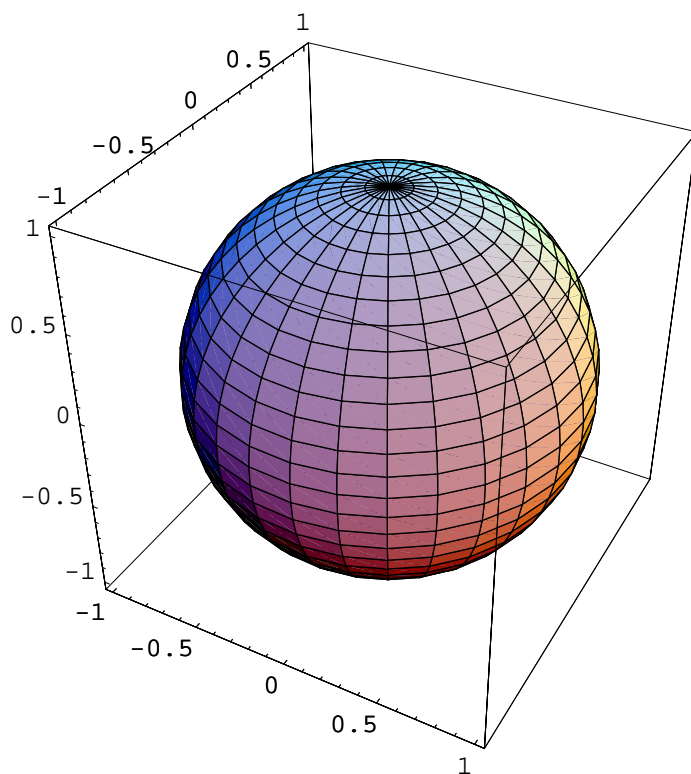
```
Plot3D[x^2+y^2, {x, -2, 2}, {y, -2, 2}, ViewPoint -> {1.3, -2.5, 1}]
```



- SurfaceGraphics -

(*用参数方程作出图形*)

```
ParametricPlot3D[{Sin[u] * Cos[v], Sin[u] * Sin[v], Cos[u]}, {u, 0, Pi}, {v, 0, 2 Pi}]
```



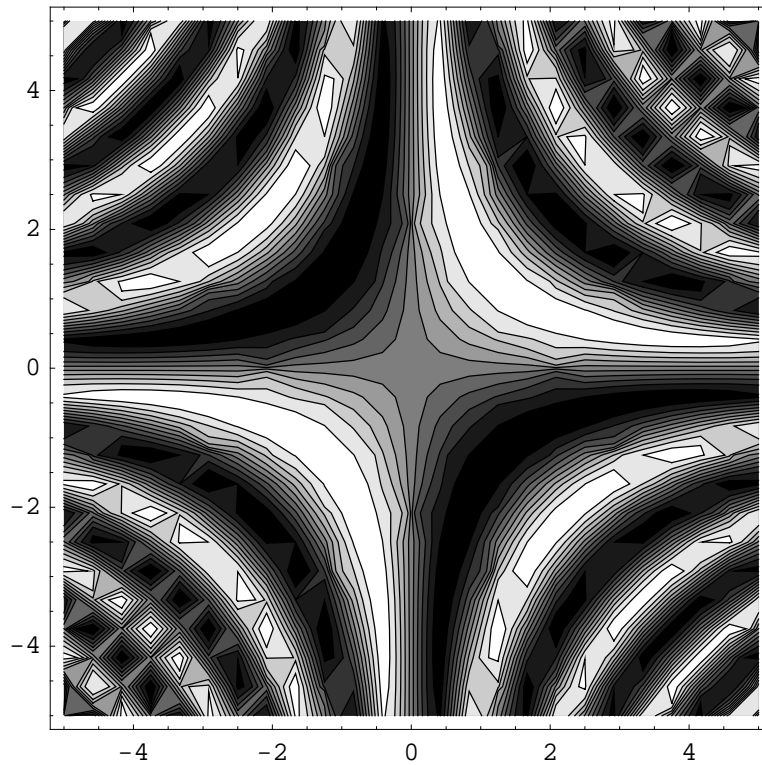
- Graphics3D -

Mathematica功能介绍之四

函数的等值线图形

(*作出函数的等值线图形*)

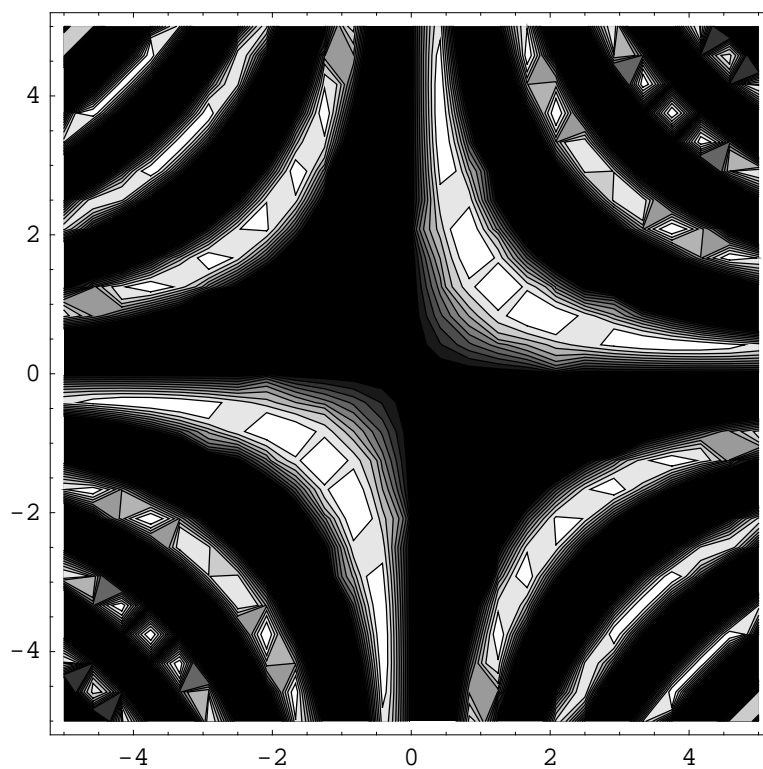
```
ContourPlot[Sin[x*y], {x, -5, 5}, {y, -5, 5}]
```



- ContourGraphics -

(*作出函数的等值线图形, 并规定图形的因变量显示范围*)

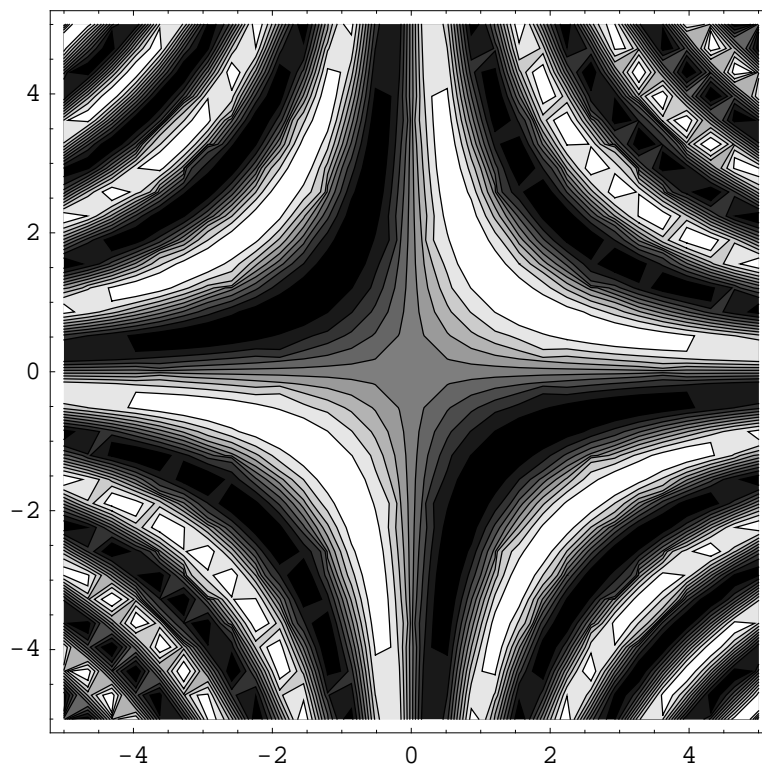
```
ContourPlot[Sin[x*y], {x, -5, 5}, {y, -5, 5}, PlotRange -> {0, 1}]
```



- ContourGraphics -

(*作出函数的等值线图形, 并规定采样点数*)

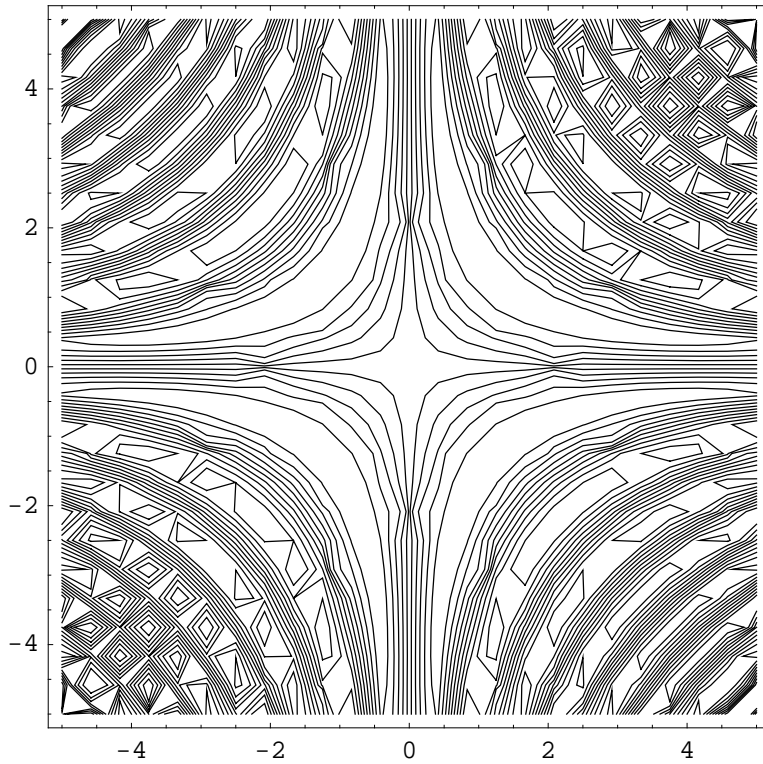
```
ContourPlot[Sin[x*y], {x, -5, 5}, {y, -5, 5}, PlotPoints -> 30]
```



- ContourGraphics -

(*作出函数的等值线图形, 并去掉阴影*)

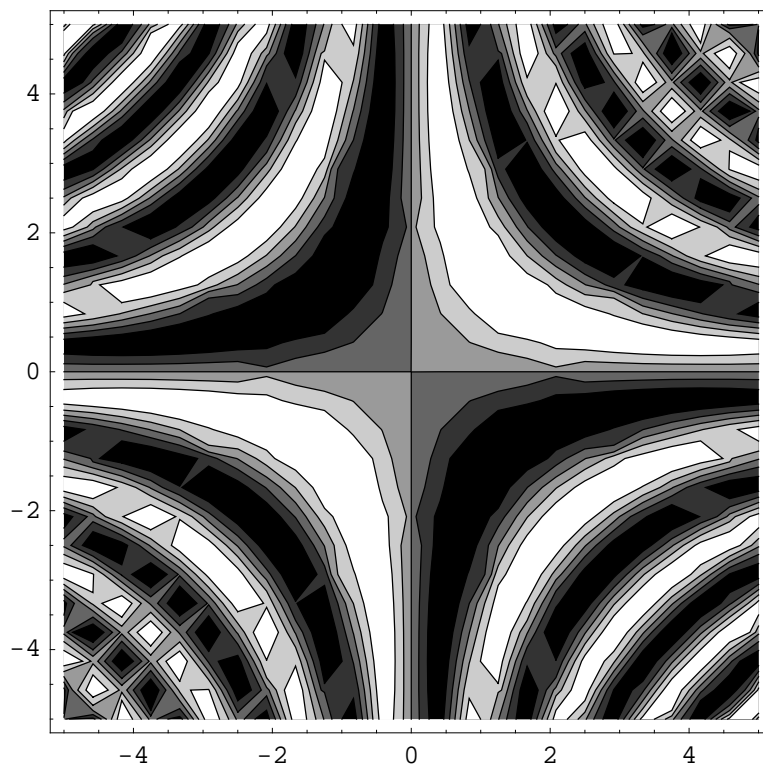
```
ContourPlot[Sin[x*y], {x, -5, 5}, {y, -5, 5}, ContourShading -> False]
```



- ContourGraphics -

(*作出函数的等值线图形, 并规定等值线数*)

```
ContourPlot[Sin[x*y], {x, -5, 5}, {y, -5, 5}, Contours -> 5]
```



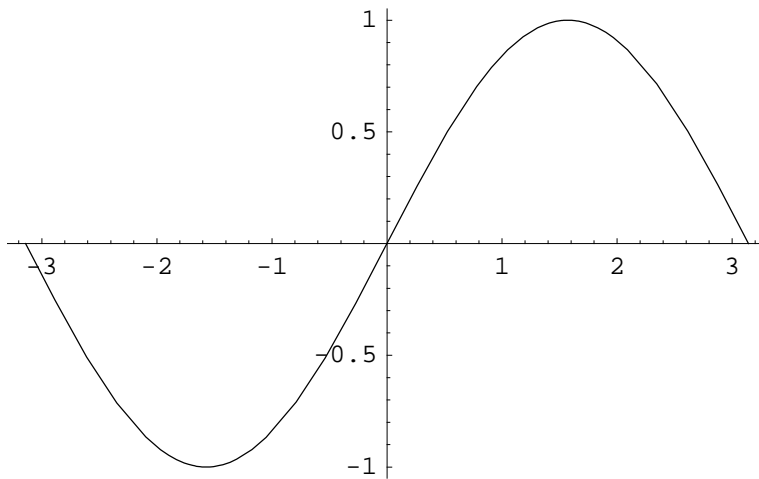
- ContourGraphics -

图形的存储、重画与重叠

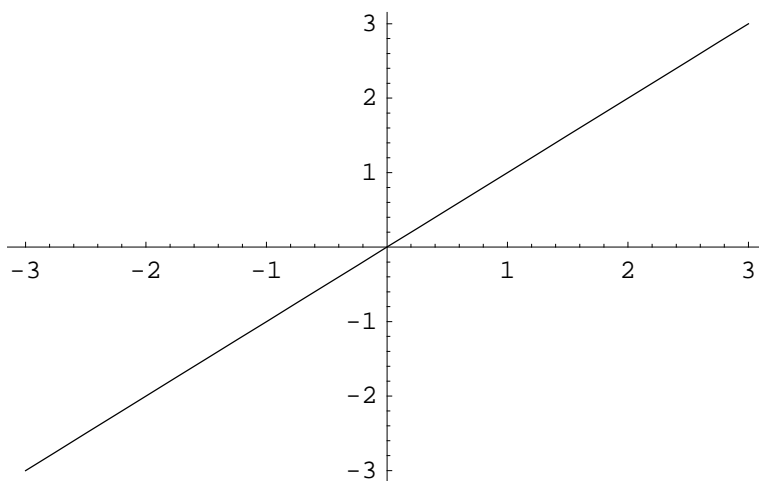
(*作出图形, 并分别存储在变量t1和t2中*)

```
t1 = Plot[Sin[x], {x, -Pi, Pi}]
```

```
t2 = Plot[x, {x, -3, 3}]
```



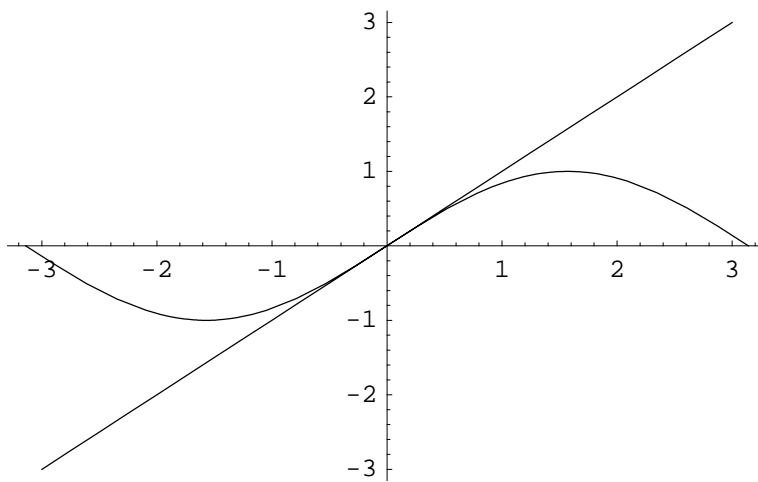
- Graphics -



- Graphics -

(*在同一坐标系中显示图形*)

```
Show[t1, t2]
```



- Graphics -

Mathematica功能介绍之五

极限的计算

函数极限

```
In[1]:= Limit[Sin[x] / x, x → 2]
```

```
Out[1]=  $\frac{\text{Sin}[2]}{2}$ 
```

```
In[2]:= (*重要极限*)  
Limit[Sin[x] / x, x → 0]  
Limit[(1 + 1/x)^x, x → Infinity]
```

```
Out[2]= 1
```

```
Out[3]= e
```

```
In[4]:= (*极限为无穷的例子*)  
Limit[2 / (x - 1), x → 1]
```

```
Out[4]=  $\infty$ 
```

```
In[5]:= (*函数 1/x 在 0 点的左极限*)  
Limit[1 / x, x → 0, Direction → 1]
```

```
Out[5]=  $-\infty$ 
```

```
In[6]:= (*函数 1/x 在 0 点的右极限*)  
Limit[1 / x, x → 0, Direction → -1]
```

```
Out[6]=  $\infty$ 
```

```
In[7]:= (*无极限的例子*)  
Limit[Sin[1/x], x -> 0]
```

```
Out[7]= Interval[{-1, 1}]
```

```
In[8]:= (*Mathematica也能处理符号极限*)  
Limit[(1+x)^a - 1/x, x -> 0]  
Limit[a^x - 1/x, x -> 0]
```

```
Out[8]= a
```

```
Out[9]= Log[a]
```

导数、偏导、(全)微分

```
In[10]:= (*Sin[x] 的导数*)  
D[Sin[x], x]
```

```
Out[10]= Cos[x]
```

```
In[11]:= (*二阶导数*)  
D[x^4 + 2 x^2 + x + 3, {x, 2}]
```

```
Out[11]= 4 + 12 x^2
```

```
In[12]:= (*高阶导数*)  
D[f[x], {x, 5}]
```

```
Out[12]= f^(5)[x]
```

```
In[13]:= (*全导数*)  
Dt[f[x, y], x]
```

```
Out[13]= Dt[y, x] f^(0,1)[x, y] + f^(1,0)[x, y]
```

```
In[14]:= (*偏导*)  
D[f[x, y], x]
```

```
Out[14]= f(1,0)[x, y]
```

```
In[15]:= (*全微分*)  
Dt[x * Cos[y] + y * Sin[x]]
```

```
Out[15]= y Cos[x] Dt[x] + Cos[y] Dt[x] + Dt[y] Sin[x] - x Dt[y] Sin[y]
```

积分

```
In[16]:= (*不定积分*)  
Integrate[Sin[x], x]
```

```
Out[16]= -Cos[x]
```

```
In[17]:= (*定积分*)  
Integrate[Sin[x], {x, 0, Pi}]
```

```
Out[17]= 2
```

```
In[18]:= (*重积分*)  
Integrate[x * Cos[y], {x, 0, Pi}, {y, 0, x}]
```

```
Out[18]=  $\pi$ 
```

```
In[19]:= (*数值积分*)  
NIntegrate[Sin[x] / x, {x, 0, Pi}]
```

```
Out[19]= 1.85194
```

方程求解

```
In[20]:= (*方程求解*)  
Solve[x^2 + 3 x + 2 == 0, x]
```

```
Out[20]= {{x -> -2}, {x -> -1}}
```

```
In[21]:= (*含参数方程求解*)  
Reduce[a * x + 2 == 0, x]
```

```
Out[21]= a ≠ 0 && x == - $\frac{2}{a}$ 
```

```
In[22]:= (*微分方程求解*)  
DSolve[y''[x] + 3 y'[x] + 2 y[x] == E^x, y[x], x]
```

```
Out[22]= {{y[x] ->  $\frac{e^x}{6}$  + e^{-2x} C[1] + e^{-x} C[2]}}
```

```
In[23]:= (*数值求解1*)  
NSolve[x^4 + x == 1, x]
```

```
Out[23]= {{x -> -1.22074}, {x -> 0.248126 - 1.03398 i},  
{x -> 0.248126 + 1.03398 i}, {x -> 0.724492}}
```

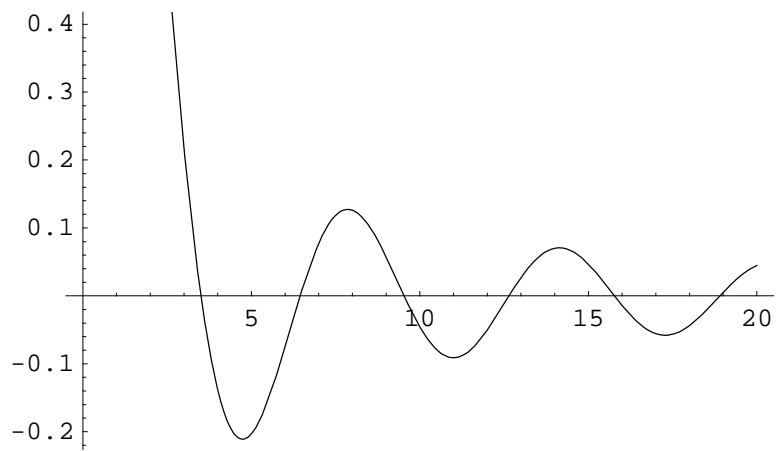
```
In[24]:= (*数值求解2*)  
FindRoot[Sin[x] + x == 1, {x, 0}]
```

```
Out[24]= {x -> 0.510973}
```

```
In[25]:= (*数值求解微分方程*)  
NDSolve[{y'[x] + x * y[x] == Sin[x], y[1] == 2}, y, {x, 1, 20}]
```

```
Out[25]= {{y -> InterpolatingFunction[{{1., 20.}}, <>]}}
```

```
In[26]:= f[x_] := y[x] /. %25  
Plot[f[x], {x, 1, 20}]
```



```
Out[27]= - Graphics -
```

和运算

```
In[28]:= Sum[1/n^2, {n, 1, Infinity, 1}]  
N[%]
```

```
Out[28]=  $\frac{\pi^2}{6}$ 
```

```
Out[29]= 1.64493
```

```
In[30]:= (*数值求和*)  
NSum[1/n^2, {n, 1, Infinity, 1}]
```

```
Out[30]= 1.64493
```

```
In[31]:= 幂级数展开和泰勒多项式  
Series[Sin[x], {x, 0, 5}]  
Normal[Series[Sin[x], {x, 0, 5}]]
```

```
Out[31]= 幂级数展开和泰勒多项式
```

```
Out[32]=  $x - \frac{x^3}{6} + \frac{x^5}{120} + O[x]^6$ 
```

```
Out[33]=  $x - \frac{x^3}{6} + \frac{x^5}{120}$ 
```

实验1 数列极限与生长模型

问题提出：

在生物学中，有一个刻画生物群体中的个体总量的增长情况的著名方程---Logistic方程(或称为Logistic模型)：

$$p[n]=k*p[n-1]*(1-p[n-1])$$

其中 $p[n]$ 为某一生物群体的第 n 代的个体总量与该群体所能达到的最大个体总量之比, k 为比例系数。

本实验将模拟该生物群体的变化情况。

```
(*定义递推函数*)  
f[x_] := k*x*(1-x)
```

```
(*设置初始值p[0]*)  
p[0] = 0.5;
```

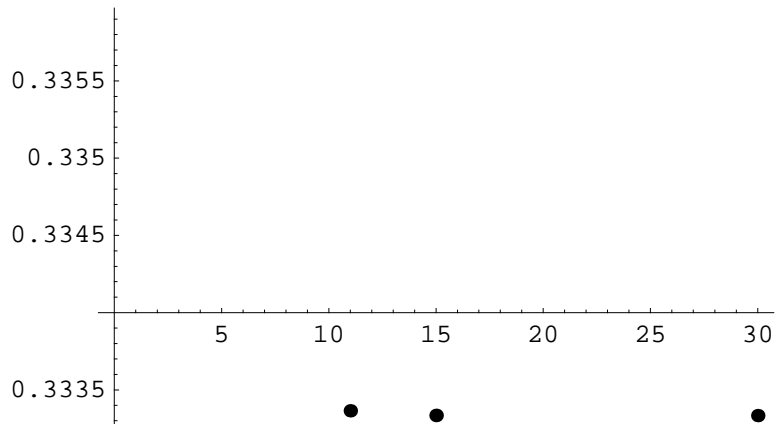
```
(*递推定义p[n]*)  
p[t_] := f[p[t-1]]
```

情形1：当 $k=1.5$ 时

```
k = 1.5;  
(*构造p[n] 的点集*)  
data = Table[p[i], {i, 30}]
```

```
{0.375, 0.351563, 0.341949, 0.33753, 0.335405, 0.334363,  
0.333847, 0.33359, 0.333461, 0.333397, 0.333365, 0.333349,  
0.333341, 0.333337, 0.333335, 0.333334, 0.333334, 0.333334,  
0.333333, 0.333333, 0.333333, 0.333333, 0.333333, 0.333333,  
0.333333, 0.333333, 0.333333, 0.333333, 0.333333, 0.333333}
```

```
(*作出p[n] 的点图*)
ListPlot[data]
```

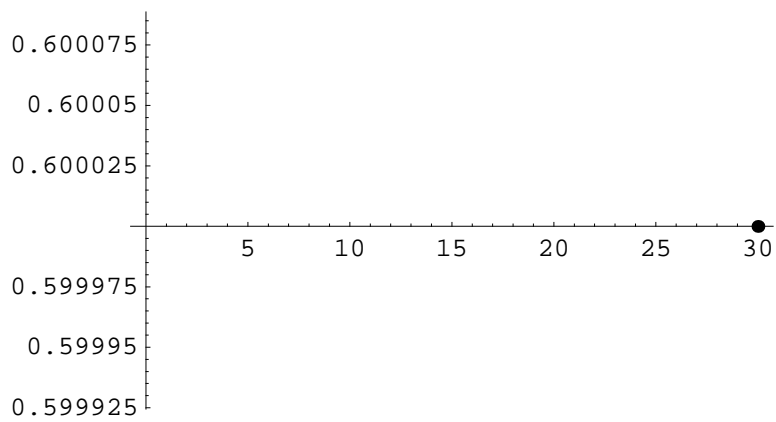


- Graphics -

情形2: 当k=2.5时

```
k = 2.5;
(*构造p[n] 的点集*)
data = Table[p[i], {i, 30}];
```

```
(*作出p[n] 的点图*)
ListPlot[data]
```



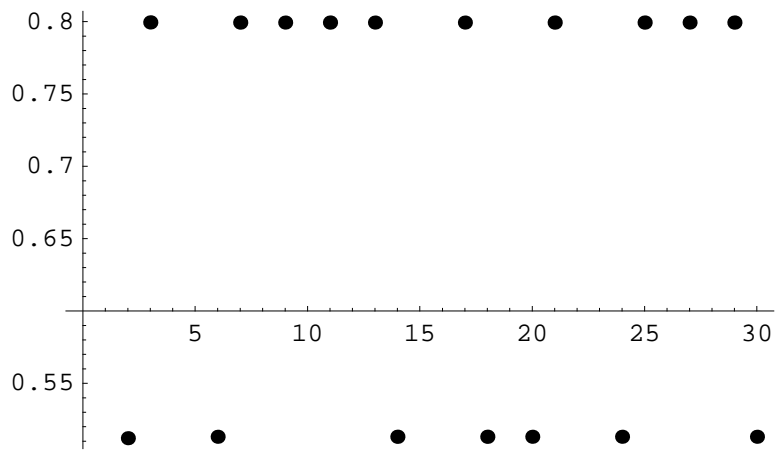
- Graphics -

这两个数列看起来都是收敛的, 这表明: 种群中的个体总量将趋于一个稳定值。

情形3: 当 $k=3.2$ 时

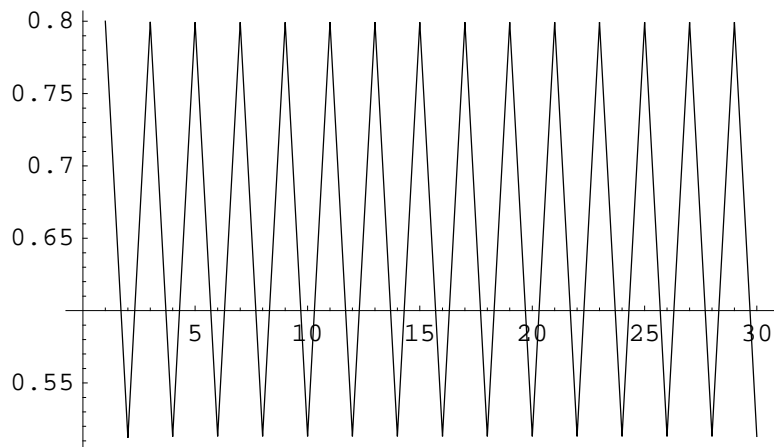
```
k = 3.2;  
(*构造p[n] 的点集*)  
data = Table[p[i], {i, 30}];
```

```
(*作出p[n] 的点图*)  
ListPlot[data]
```



- Graphics -

```
(*把相邻的p[n]用直线连接,再次作出p[n]的图形,即p[n]的折线图*)  
ListPlot[data, PlotJoined -> True]
```



- Graphics -

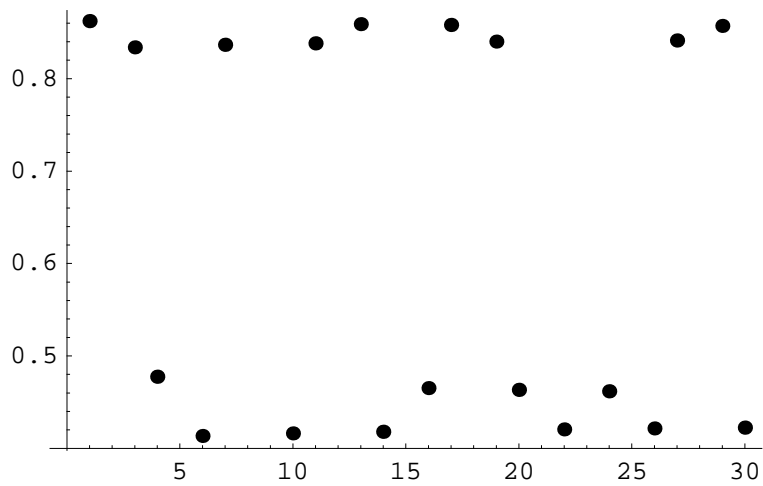
可见这时数列不收敛,而出现近似周期性的波动,数列的项几乎交替取两个不同的值.

情形4: 当 $k=3.45$ 时

```
k = 3.45;  
(*构造p[n]的点集*)  
data = Table[p[i], {i, 30}];
```

(*作出 $p[n]$ 的点图*)

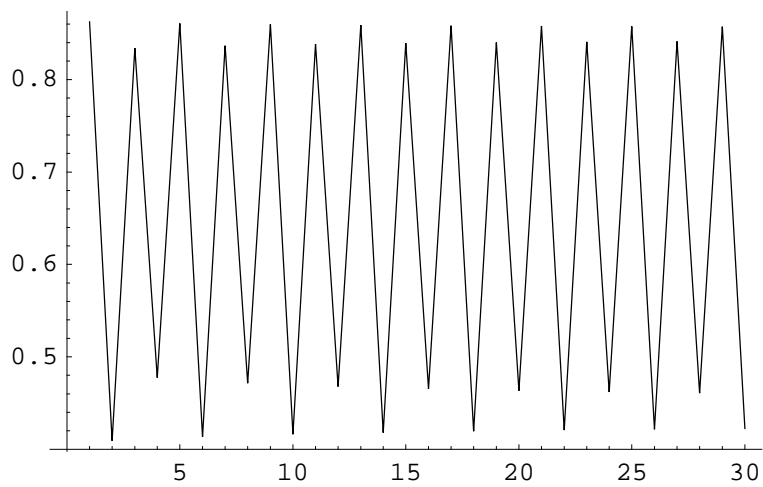
```
ListPlot[data]
```



- Graphics -

(*作出 $p[n]$ 的折线图*)

```
ListPlot[data, PlotJoined → True]
```



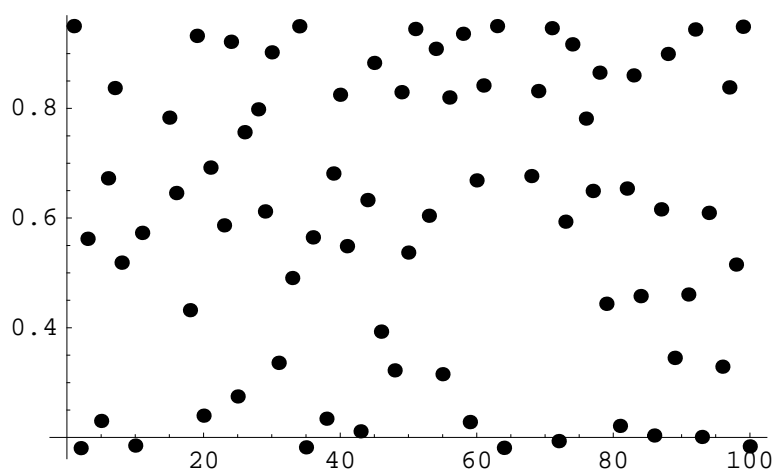
- Graphics -

可见, 这时数列仍出现周期性波动的情况, 但所取到的不同的值的个数明显增加。

情形5: 当 $k=3.8$ 时

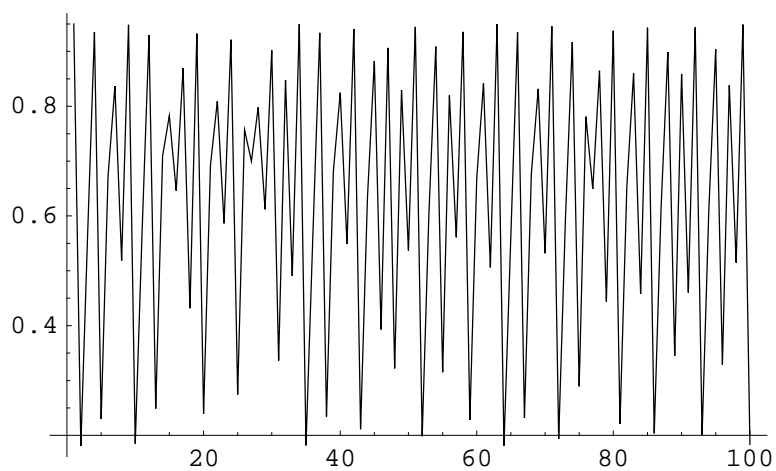
```
k = 3.8;  
(*构造p[n] 的点集*)  
data = Table[p[i], {i, 100}];
```

```
(*作出p[n] 的点图*)  
ListPlot[data]
```



- Graphics -

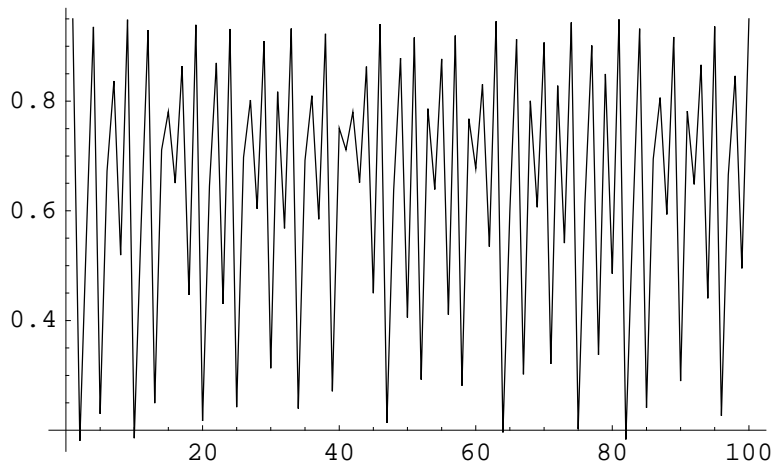
```
(*作出p[n] 的折线图*)  
ListPlot[data, PlotJoined → True]
```



- Graphics -

```
(*此时, 对初值p[0] 作微小变化*)  
k = 3.8; p[0] = 0.5 + 0.001;  
(*构造p[n] 的点集*)  
data = Table[p[i], {i, 100}];
```

```
(*作出p[n] 的折线图*)  
ListPlot[data, PlotJoined → True]
```



- Graphics -

可见这时数列的变化情况变得很紊乱。这里特别指出一种值得注意的现象:若给初值 $p[0]$ 一个微小的变动,比如上面增加0.001,那么数列后面的项将会出现很大的变动,这种现象在数学上被称为“混沌”。在某些特定条件下,昆虫种群的生长情况就会出现这种难以预测的混沌现象。

实验2 飞机安全降落曲线的确定

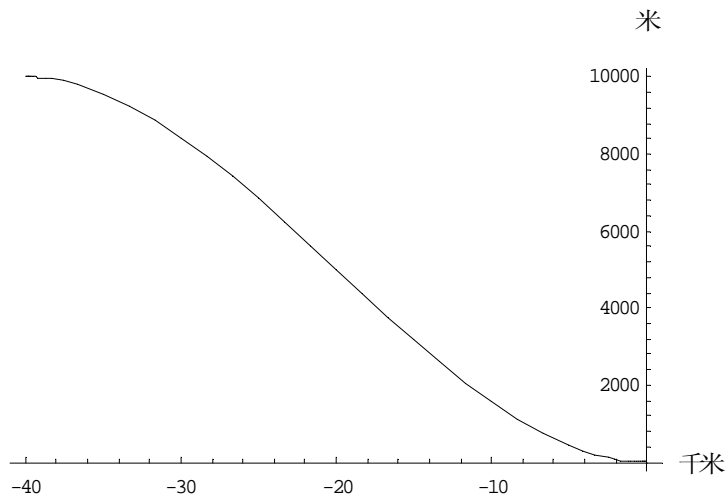


图1 飞机最佳降线示意图

上图是飞机降落迹线的示意图, 我们希望找出最佳降线的方程。

先定义下列变量, 并作出一些相应的初始假定

初始假定

飞机降落时, 距离落点的水平距离为 l (千米);

飞机开始降落时, 飞机高度为 h (米), 机场的地面高度取作0;

飞机在降落过程中保持水平速度 u (千米/秒)不变。

建模

现用三次多项式来模拟飞机迹线, 因此设

$$f(x)=ax^3+bx^2+cx+d,$$

基于上面的初始假定, 我们得到 $f(x)$ 满足的如下四个初始条件:

- (1) $f(0)=0$
- (2) $f'(0)=0$
- (3) $f(-l)=h$
- (4) $f'(-l)=0$

由这四个初始条件得到飞机下降曲线方程

```
f[x_] := a*x^3 + b*x^2 + c*x + d
```

(*由条件确定a、b、c、d*)

```
Solve[{f[0] == 0, f'[0] == 0, f[-1] == h, f'[-1] == 0}, {a, b, c, d}]
```

$$\left\{ \left\{ b \rightarrow \frac{3h}{1^2}, a \rightarrow \frac{2h}{1^3}, d \rightarrow 0, c \rightarrow 0 \right\} \right\}$$

(*根据上面求出的值给a、b、c、d赋值*)

```
a = 2 h / 1^3; b = 3 h / 1^2; d = 0; c = 0;
```

(*此时的函数表达式为*)

```
f[x]
```

$$\frac{3 h x^2}{1^2} + \frac{2 h x^3}{1^3}$$

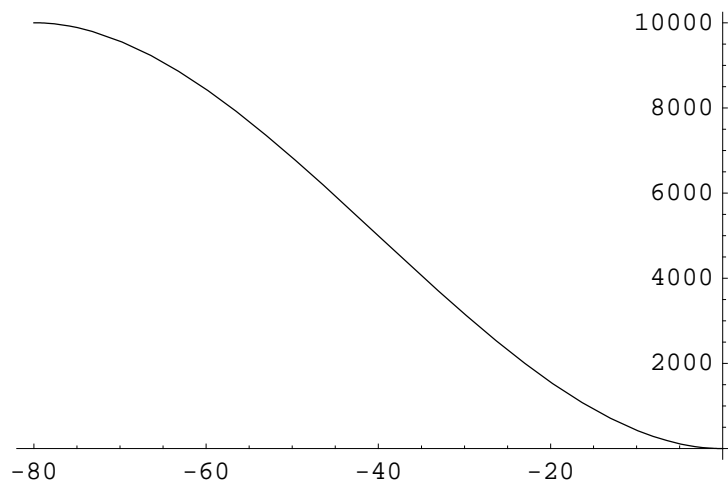
对具体数据进行验证

(*假定h=10000米; l=80千米时曲线的图形*)

```
h = 10000.0; l = 80.0;
```

```
Plot[f[x], {x, -l, 0}]
```

对具体数据进行验证



- Graphics -

(*清除变量h、l的值,为进一步的讨论作准备*)

```
Clear[h, l]
```

现在同时考虑水平速度

由条件得到 $dx/dt=u$, dy/dt 即为铅直速度, 再由复合函数可得铅直速度为

$$vy[x_] = u * f'[x]$$

$$u \left(\frac{6 h x}{l^2} + \frac{6 h x^2}{l^3} \right)$$

由此可得铅直加速度为

$$ay[x_] = u * vy'[x]$$

$$u^2 \left(\frac{6 h}{l^2} + \frac{12 h x}{l^3} \right)$$

从而在飞机降落过程中的铅直速度最大值为

$$aa = ay[0]$$

$$\frac{6 h u^2}{l^2}$$

现在自由落体加速度为9.8(米/秒.秒), 因此飞机的铅直加速度的最大值要远小于此值。

下面假设高度 $h=10000$ 米, 水平距离 $l=80$ 千米, 水平速度 $u=500$ 公里/小时

```
h = 10000.0; l = 80.0; u = 500.0;
aa / 3600^2 (*这里由于单位的关系, 因此除以3600^2*)
```

```
0.180845
```

```
(*清除变量h、l、u、aa、x的值, 为进一步的讨论作准备*)
Clear[h, l, u, aa, x]
```

进一步假设

现在假设飞机的水平运动 v_x 为匀加速, 开始降落时为 u_0 , 落地时为 u_1 , 降落所化的时间为 t_0 , 由此可知水平位移函数为

$$xx[t_] := -l + u_0 * t + (u_1 - u_0) / (2 t_0) * t^2$$

由于条件 $xx(t_0)=0$, 由此解出 t_0

$$\text{Solve}[xx[t_0] == 0, t_0]$$

$$\left\{ \left\{ t_0 \rightarrow \frac{2 l}{u_0 + u_1} \right\} \right\}$$

(*根据上面求出的值给 t_0 赋值*)

$$t_0 = 2 l / (u_0 + u_1);$$

(*构造铅直方向的位移函数*)

$$yy[t_] = f[xx[t]]$$

$$\frac{3 h \left(-1 + t u_0 + \frac{t^2 (-u_0 + u_1) (u_0 + u_1)}{4 l} \right)^2}{l^2} + \frac{2 h \left(-1 + t u_0 + \frac{t^2 (-u_0 + u_1) (u_0 + u_1)}{4 l} \right)^3}{l^3}$$

下面以具体的数据计算铅直加速度

(*先给以赋值*)

$$u_0 = 500; u_1 = 200; l = 80; h = 10000;$$

(*计算铅直方向的加速度函数, 其中除以 3600^2 是由于单位关系*)

$$yy''[t] / 3600^2;$$

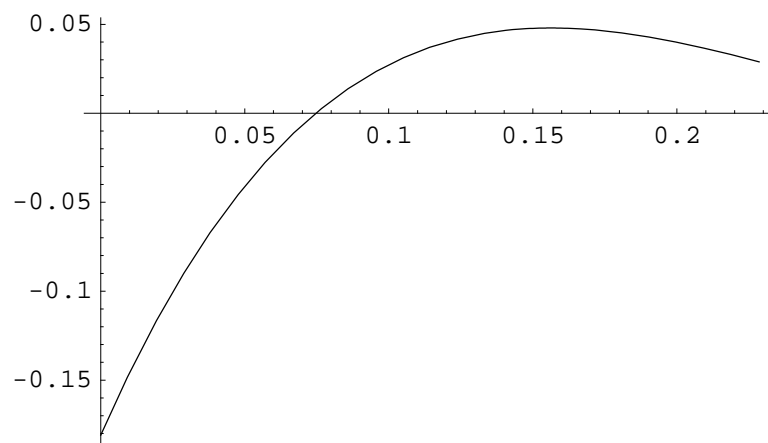
(*为了解此函数的情况, 对上述函数进行完全展开*)

$$\text{Expand}[yy''[t] / 3600^2]$$

$$-\frac{625}{3456} + \frac{101875 t}{27648} - \frac{966875 t^2}{49152} + \frac{3828125 t^3}{98304} - \frac{26796875 t^4}{1048576}$$

(*作出此函数的图形*)

```
Plot[yy''[t] / 3600^2, {t, 0, t0}, PlotRange -> All]
```



- Graphics -

(*计算最大加速度*)

```
yy''[0] / 3600^2 // N
```

-0.180845

实验3 泰勒公式与函数逼近

演示目的与要求:

演示函数及各次泰勒多项式的图形,展示泰勒多项式对函数的逼近情况.

演示内容

求出 $\sin(x)$ 的直到19阶的泰勒多项式,并作出 $\sin(x)$ 与它们的图形,进行观察与比较.

1.展开

(*在 $x=0$ 点展开函数 $\sin(x)$ 到一阶*)

```
p1 = Series[Sin[x], {x, 0, 1}]
```

$$x + O[x]^2$$

(*在 $x=0$ 点展开函数 $\sin(x)$ 到三阶*)

```
p3 = Series[Sin[x], {x, 0, 3}]
```

$$x - \frac{x^3}{6} + O[x]^4$$

(*在 $x=0$ 点展开函数 $\sin(x)$ 到五阶*)

```
p5 = Series[Sin[x], {x, 0, 5}]
```

$$x - \frac{x^3}{6} + \frac{x^5}{120} + O[x]^6$$

(*直接显示含余项多项式的图形是不行的*)

```
Plot[p5, {x, -Pi, Pi}]
```

```
SeriesData::ssdn : Attempt to evaluate a series
  at the number -3.14159; returning Indeterminate. More...
```

```
SeriesData::ssdn : Attempt to evaluate a series
  at the number -3.14159; returning Indeterminate. More...
```

```
Plot::plnr :
  p5 is not a machine-size real number at x = -3.14159. More...
```

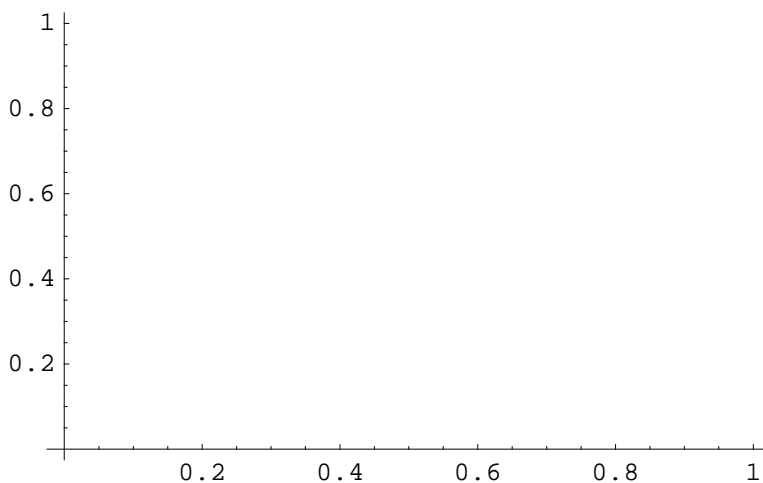
```
SeriesData::ssdn : Attempt to evaluate a series
  at the number -2.8867; returning Indeterminate. More...
```

```
General::stop : Further output of SeriesData::ssdn will
  be suppressed during this calculation. More...
```

```
Plot::plnr : p5 is not a machine-size real number at x = -2.8867. More...
```

```
Plot::plnr :
  p5 is not a machine-size real number at x = -2.60872. More...
```

```
General::stop : Further output of
  Plot::plnr will be suppressed during this calculation. More...
```



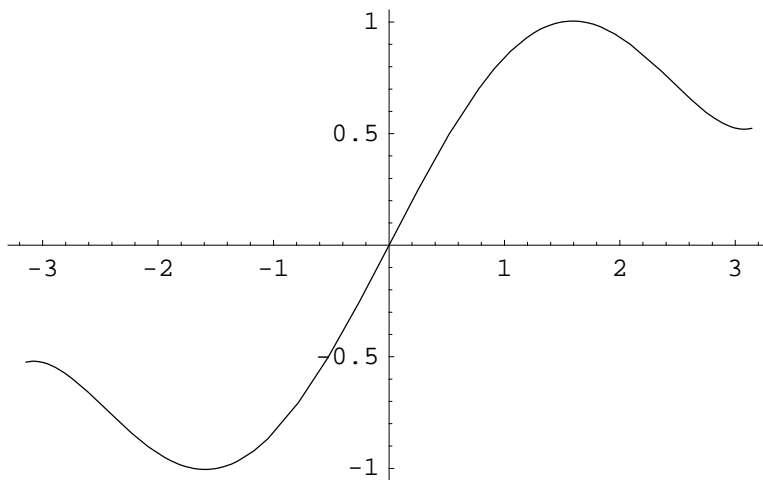
- Graphics -

(*必须使用Normal函数去掉余项, 下面显示五次泰勒公式、泰勒多项式, 并作出五次泰勒多项式的图形*)

```
p5  
Normal[p5]  
Plot[Evaluate[Normal[p5]], {x, -Pi, Pi}]
```

$$x - \frac{x^3}{6} + \frac{x^5}{120} + O[x]^6$$

$$x - \frac{x^3}{6} + \frac{x^5}{120}$$



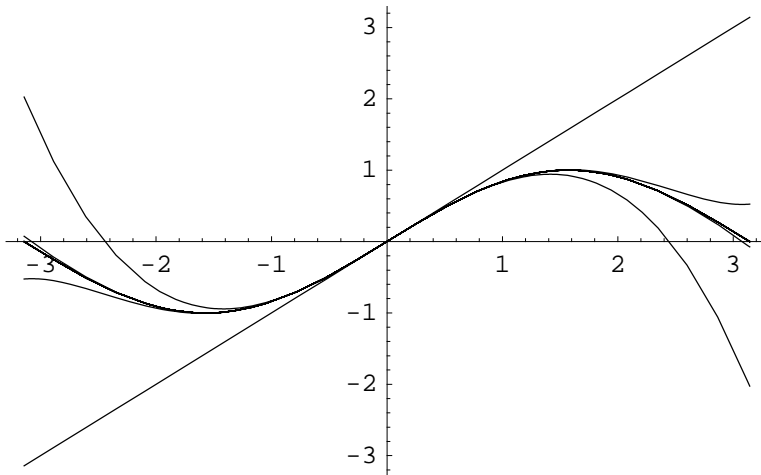
- Graphics -

2.重叠显示

(*对函数Sin(x) 在x=0 点的1阶、3阶、5阶直到19阶展开列表*)

```
t = Table[Normal[Series[Sin[x], {x, 0, i}]], {i, 1, 19, 2}];
```

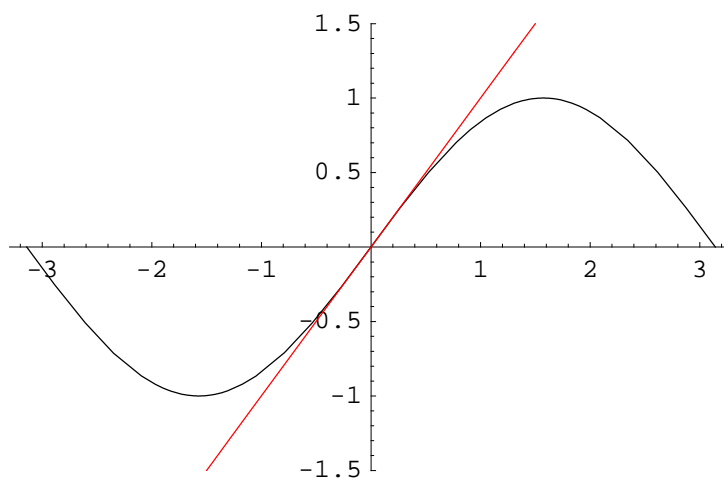
```
(*加上 Sin[x] 并绘图*)
PrependTo[t, Sin[x]];
Plot[Evaluate[t], {x, -Pi, Pi}]
```

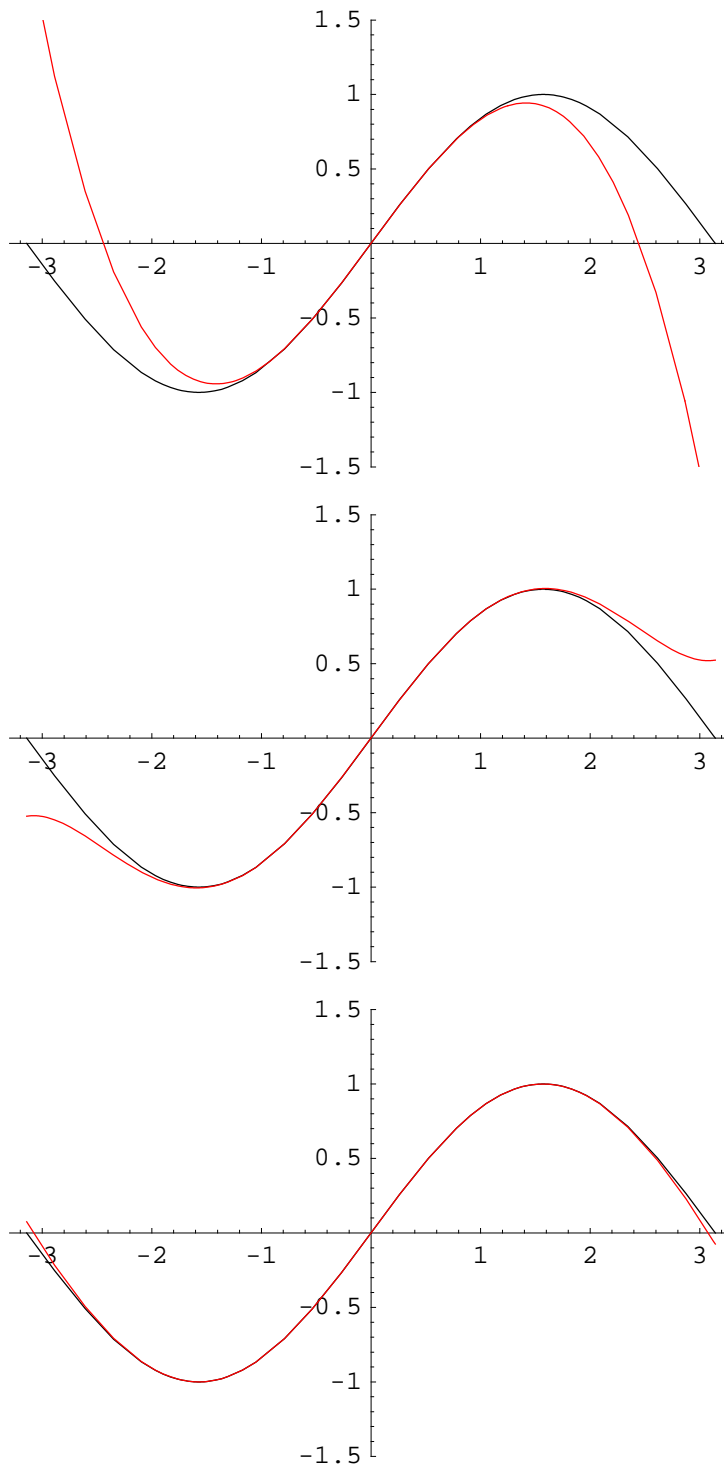


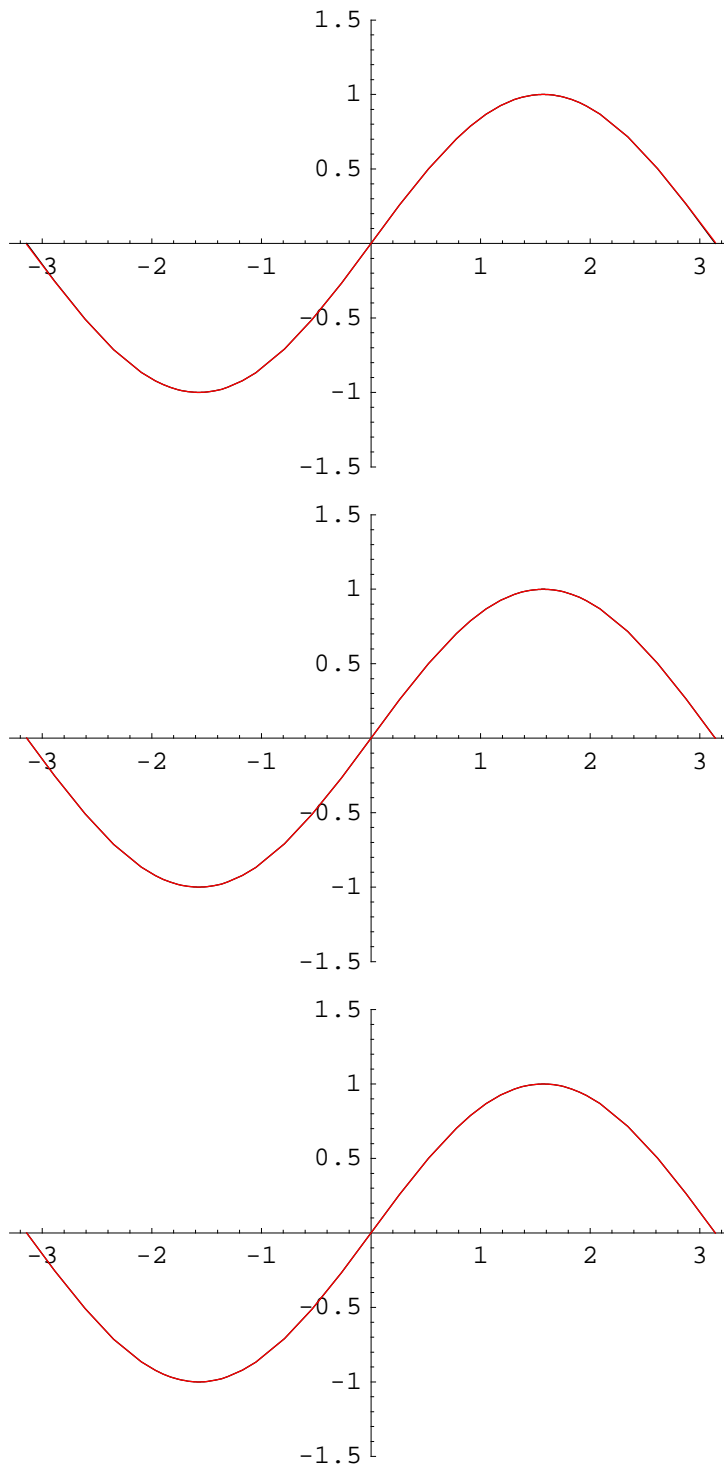
- Graphics -

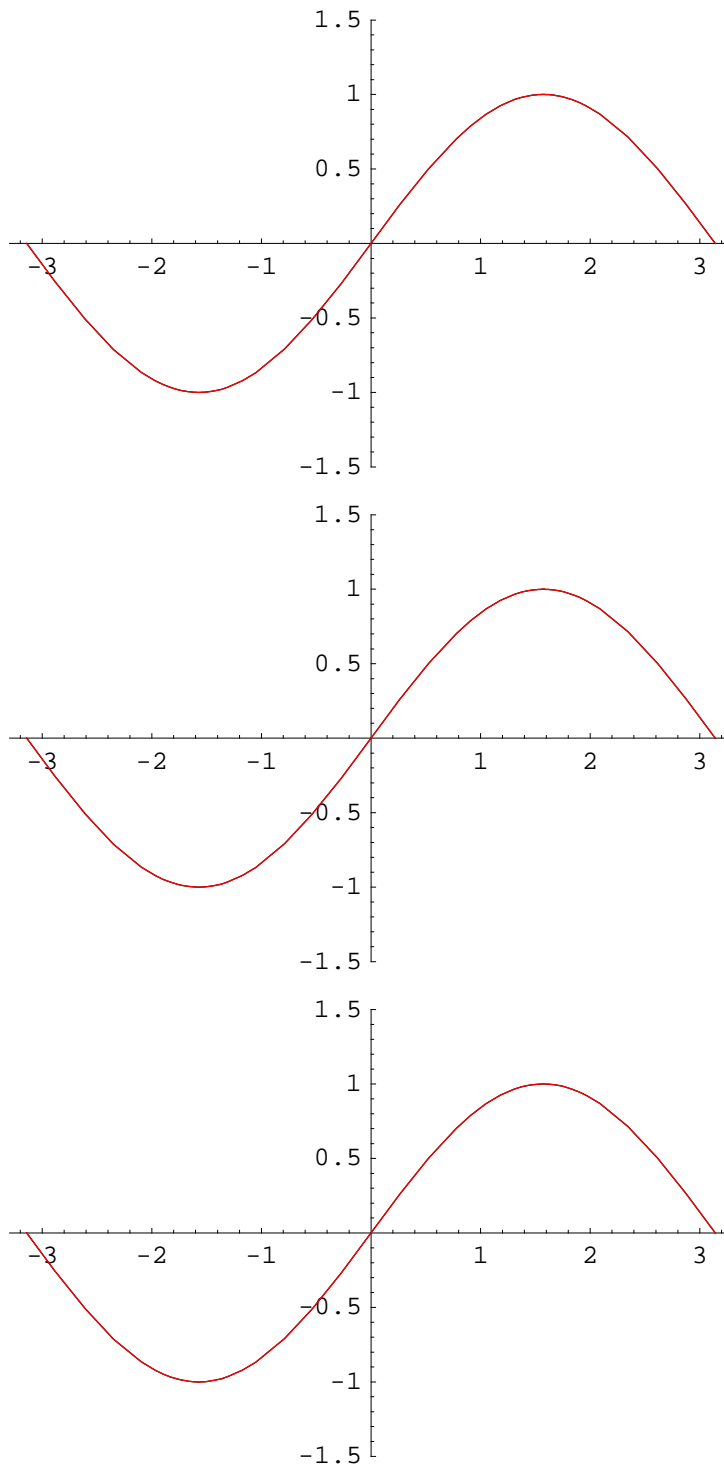
3.动画显示

```
(*将以上过程以动画展示, 其中红色的曲线表示sin(x)*)
Do[Plot[Evaluate[{Sin[x], Normal[Series[Sin[x], {x, 0, i}]}],
{x, -Pi, Pi}, PlotRange -> {-1.5, 1.5},
PlotStyle -> {RGBColor[0, 0, 0], RGBColor[1, 0, 0]}], {i, 1, 19, 2}]
```









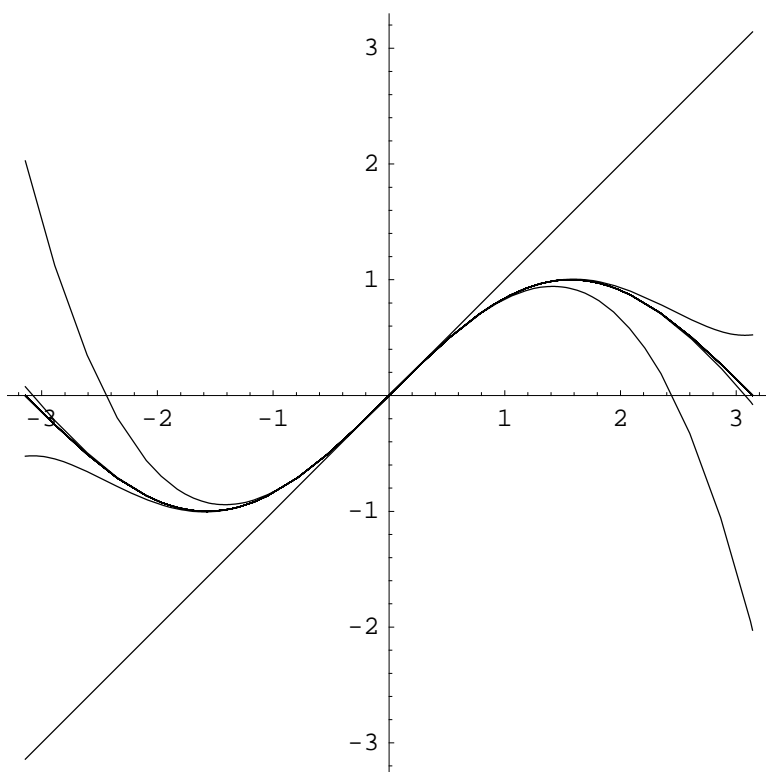
4.不同显示范围观察

(*对同样的图形,在 x 不同的取值区域内进行观察,可注意到在远离 $x=0$ 点时,低阶泰勒多项式逼近效果没有高阶逼近的好*)

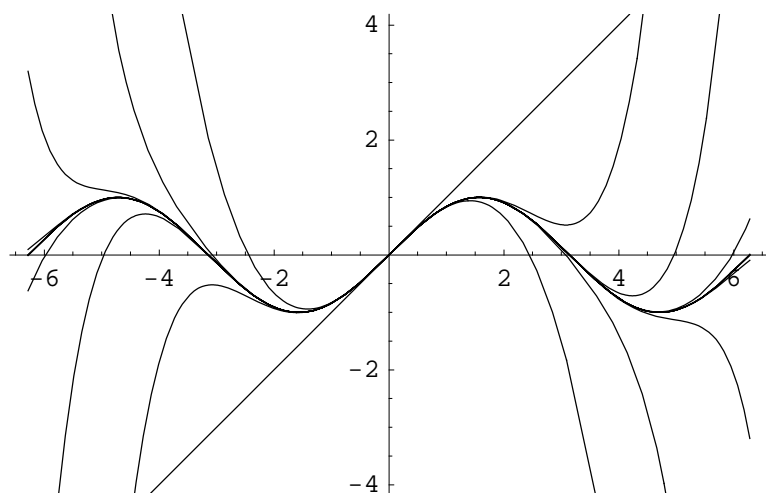
```
Plot[Evaluate[t], {x, -Pi, Pi}, AspectRatio -> Automatic]
```

```
Plot[Evaluate[t], {x, -2 Pi, 2 Pi}, AspectRatio -> Automatic]
```

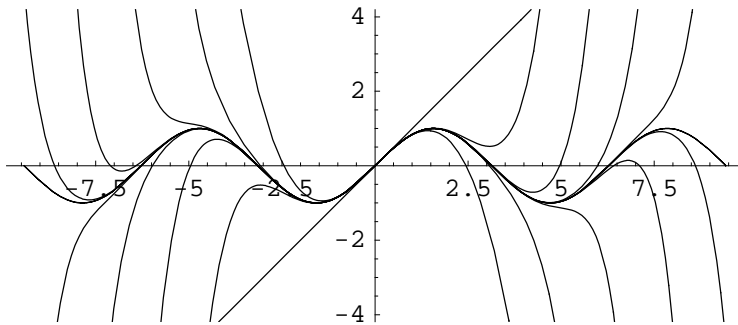
```
Plot[Evaluate[t], {x, -3 Pi, 3 Pi}, AspectRatio -> Automatic]
```



- Graphics -



- Graphics -



- Graphics -

观察要点及主要结果

注意观察泰勒多项式图形与函数图形的重合与分离情况.可以看到在 $[-\pi, \pi]$ 范围内 $\sin(x)$ 的9次泰勒多项式与函数几乎已无差别,而在 $[-2\pi, 2\pi]$ 范围内 $\sin(x)$ 的各次泰勒多项式陆续与 $\sin(x)$ 的图象分离,但其15次泰勒多项式仍紧靠着 $\sin(x)$,当在 $[-3\pi, 3\pi]$ 范围内其19次泰勒多项式的图形也与 $\sin(x)$ 的图象分离.可见,函数的泰勒多项式对于函数的近似程度,随着次数的提高而提高.但对于任一确定次数的多项式,它只在展开点的一个邻近邻域内才有较好的近似程度.

实验4 方程近似解的求法

1、二分法

(*定义函数*)

```
f[x_] := 2 x * Sqrt[(30. - x)^2 + 100.] - (30. - x) Sqrt[x^2 + 225.]
```

(*给出初始区间端点a0、b0, 误差delta, 最大循环次数k0*)

```
a0 = 0.; b0 = 30.; delta = 10^(-3); k0 = 100;
```

(*在循环中区间的端点为a、b, 中点为x*)

```
a = a0; b = b0;
```

```
Do[x = (a + b) / 2;
```

```
Print["k=", k, ", ", "x=", x];
```

```
If[f[x] == 0, Break[], If[N[f[x] * f[b]] < 0, a = x, b = x]];
```

```
If[N[Abs[b - a]] < delta, Break[], If[k == k0, Print[失败]], {k, k0}]
```

```
k=1,x=15.
```

```
k=2,x=7.5
```

```
k=3,x=11.25
```

```
k=4,x=9.375
```

```
k=5,x=8.4375
```

```
k=6,x=7.96875
```

```
k=7,x=7.73438
```

```
k=8,x=7.61719
```

```
k=9,x=7.67578
```

```
k=10,x=7.70508
```

```
k=11,x=7.69043
```

```
k=12,x=7.69775
```

```
k=13,x=7.69409
```

```
k=14,x=7.69226
```

```
k=15,x=7.69135
```

由计算结果可知近似值为7.691

2、切线迭代法

(*定义函数*)

```
f[x_] := x^3 + 1.1 x^2 + 0.9 x - 1.4
```

(*给出区间端点a0、b0, 误差delta, 最大循环次数为k0*)

```
a0 = 0; b0 = 1; delta = 10^(-4); k0 = 10;
```

(*计算导函数绝对值的最小值, 这将在误差判断时需要*)

```
m = Min[Abs[f'[a0]], Abs[f'[b0]]];
```

(*根据切线迭代法的要求, 首先选取循环初值a*)

```
If[f[a0] * f'[a0] > 0, a = a0, a = b0];
```

(*在循环中本次变量为a, 而下次循环变量为x*)

```
Do[x = a - f[a] / f'[a];  
  Print["k=", k, ", ", "x=", x];  
  If[N[Abs[f[x]]] < N[m * delta], Break[], If[k < k0, a = x, Print[失败]]], {k, k0}]
```

k=1,x=0.737705

k=2,x=0.674169

k=3,x=0.670668

实验5 定积分的近似计算

1、梯形法

```
(*定义函数*)
```

```
f[x_] := E^(-x^2)
```

```
(*定义积分区间端点a、b，误差delta，最大循环次数n0*)
```

```
a = 0; b = 1; m2 = 2; delta = 0.0001; n0 = 100;
```

```
(*定义n等分时，梯形法得到的近似值s[n]*)
```

```
s[n_] := (b - a) / n * ((f[a] + f[b]) / 2 +  
Sum[f[a + i * (b - a) / n], {i, 1, n - 1}])
```

```
(*循环计算s[n]，并显示结果，当满足误差要求时，停止循环，  
否则当到达最大循环次数还未满足误差要求时，显示字符“失败”*)
```

```
Do[Print["n=", n, ", ", "s[n]=", N[s[n]]];  
If[N[(b - a)^3 * m2 / 12 / n^2] < delta, Break[],  
If[n == n0, Print[失败]]], {n, n0}]
```

```
n=1,s[n]=0.68394
```

```
n=2,s[n]=0.73137
```

```
n=3,s[n]=0.739986
```

```
n=4,s[n]=0.742984
```

```
n=5,s[n]=0.744368
```

```
n=6,s[n]=0.745119
```

```
n=7,s[n]=0.745572
```

```
n=8,s[n]=0.745866
```

```
n=9,s[n]=0.746067
```

```
n=10,s[n]=0.746211
```

```
n=11,s[n]=0.746317
```

```
n=12,s[n]=0.746398
```

```
n=13,s[n]=0.746461
```

```
n=14,s[n]=0.746511
```

n=15,s[n]=0.746552
n=16,s[n]=0.746585
n=17,s[n]=0.746612
n=18,s[n]=0.746635
n=19,s[n]=0.746654
n=20,s[n]=0.746671
n=21,s[n]=0.746685
n=22,s[n]=0.746697
n=23,s[n]=0.746708
n=24,s[n]=0.746718
n=25,s[n]=0.746726
n=26,s[n]=0.746733
n=27,s[n]=0.74674
n=28,s[n]=0.746746
n=29,s[n]=0.746751
n=30,s[n]=0.746756
n=31,s[n]=0.74676
n=32,s[n]=0.746764
n=33,s[n]=0.746768
n=34,s[n]=0.746771
n=35,s[n]=0.746774
n=36,s[n]=0.746777
n=37,s[n]=0.746779
n=38,s[n]=0.746782
n=39,s[n]=0.746784
n=40,s[n]=0.746786
n=41,s[n]=0.746788

由计算结果可知道所求积分的近似值为0.7468

2、抛物线法

(*定义函数*)

```
f[x_] := E^(-x^2)
```

(*给出积分区间的端点a、b，四阶导函数绝对值的最小值m4，
误差delta，最大循环次数n0*)

```
a = 0; b = 1; m4 = 12; delta = 0.0001; n0 = 100;
```

(*定义2n等分时抛物线法的近似值p[n]*)

```
p[n_] := (b - a) / n / 6 * (f[a] + f[b] + 2 Sum[f[a + i * (b - a) / n / 2],  
      {i, 2, 2 n - 2, 2}] + 4 Sum[f[a + i * (b - a) / n / 2], {i, 1, 2 n - 1, 2}])
```

(*循环计算p[n]，并显示结果，当满足误差要求时，停止循环，
否则当到达最大循环次数还未满足误差要求时，显示字符“失败”*)

```
Do[Print["n=", n, ",", "p[n]=", N[p[n]]];  
  If[N[(b - a)^5 * m4 / 180 / (2 n)^4] < delta, Break[],  
  If[n == n0, Print[失败]], {n, n0}]
```

```
n=1,p[n]=0.74718
```

```
n=2,p[n]=0.746855
```

```
n=3,p[n]=0.74683
```

由计算结果可知道所求积分的近似值为0.7468

实验6 鲨鱼袭击目标的前进途径

等量线、隐函数及积分曲线的图形

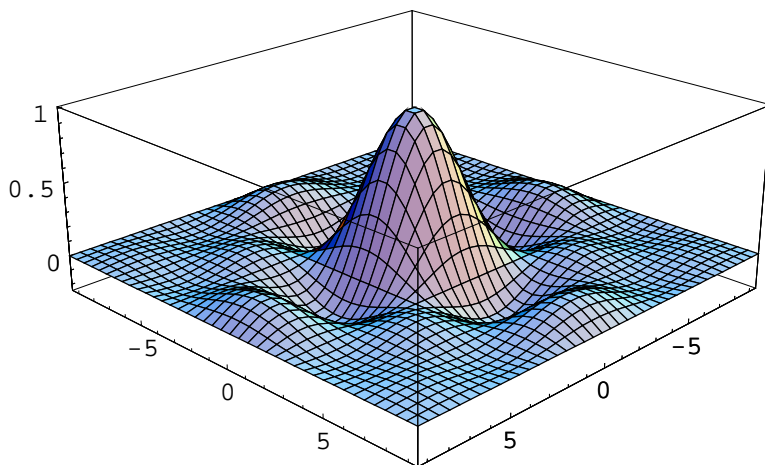
等量线的图形

(*定义函数*)

```
f[x_, y_] := If[x == 0, 1, Sin[x] / x] * If[y == 0, 1, Sin[y] / y]
```

(*先作出该函数所表示曲面的图形*)

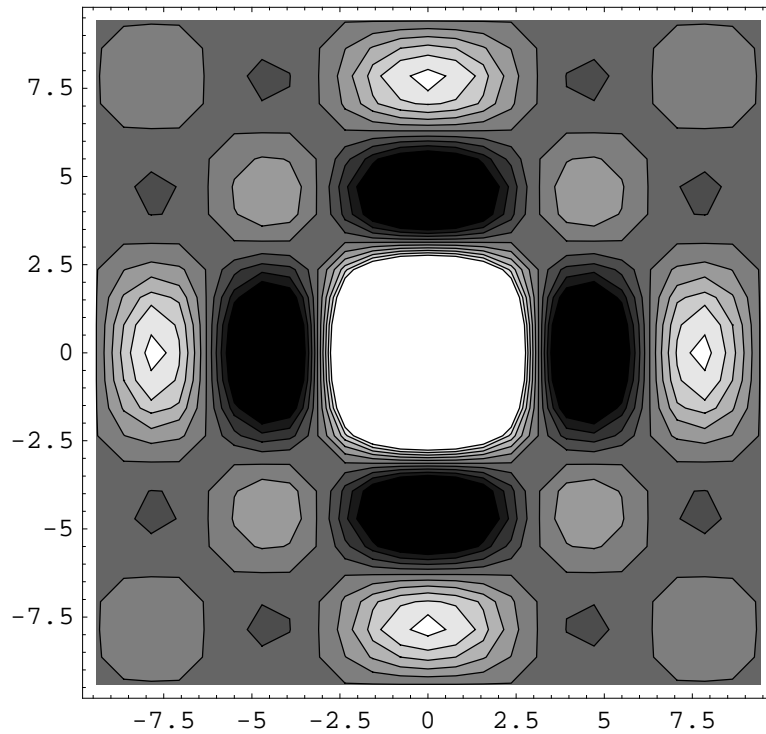
```
Plot3D[f[x, y], {x, -3 Pi, 3 Pi}, {y, -3 Pi, 3 Pi}, PlotPoints -> 40,  
ViewPoint -> {2.239, 2.204, 1.258}, PlotRange -> All]
```



- SurfaceGraphics -

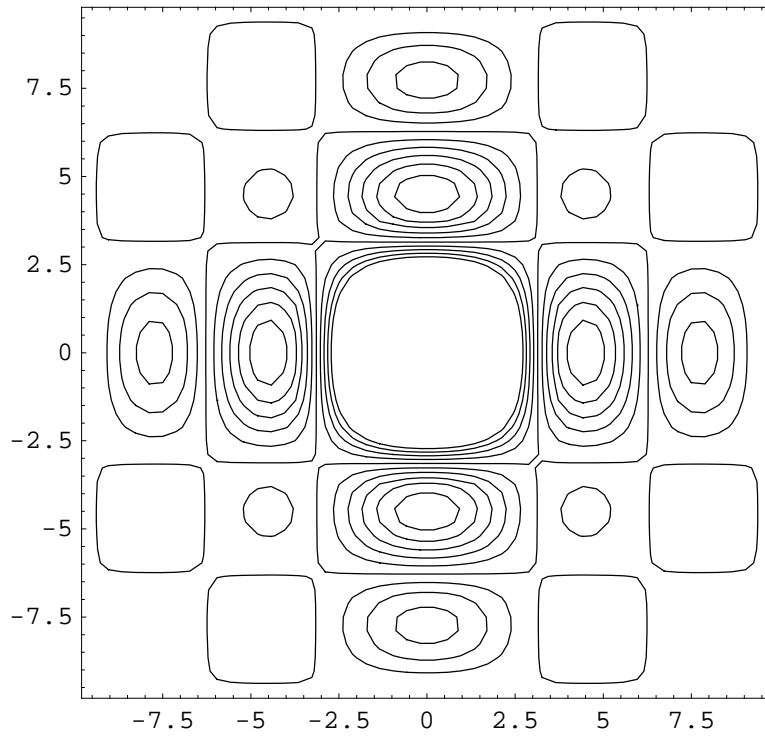
(*基本的等量线作图, 此时显示10条等高线, 并有阴影, 其中颜色深的地方表示值小, 颜色浅的地方值大*)

```
ContourPlot[f[x, y], {x, -3 Pi, 3 Pi}, {y, -3 Pi, 3 Pi}]
```



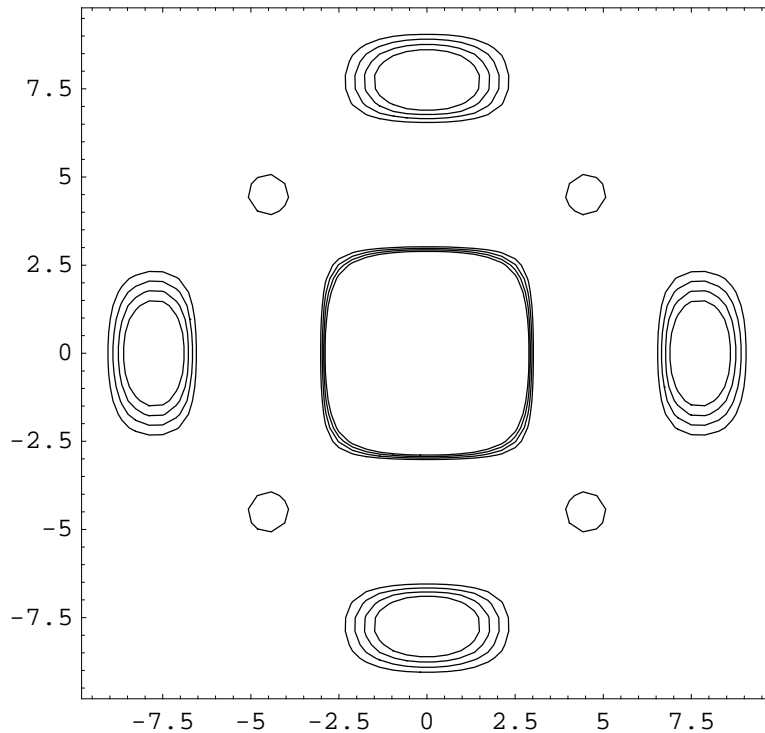
- ContourGraphics -

```
(*下面我们提高采样点数为50, 并且不显示阴影*)  
ContourPlot[f[x, y], {x, -3 Pi, 3 Pi}, {y, -3 Pi, 3 Pi},  
PlotPoints -> 50, ContourShading -> False]
```



- ContourGraphics -

```
(*下面再限制值的范围为[0.025,0.1]，等量线条数为4，
  采样点数为50，并且不显示阴影*)
ContourPlot[f[x, y], {x, -3 Pi, 3 Pi}, {y, -3 Pi, 3 Pi},
  PlotRange -> {0.025, 0.1}, Contours -> 4, PlotPoints -> 50,
  ContourShading -> False]
```



- ContourGraphics -

梯度线的绘制

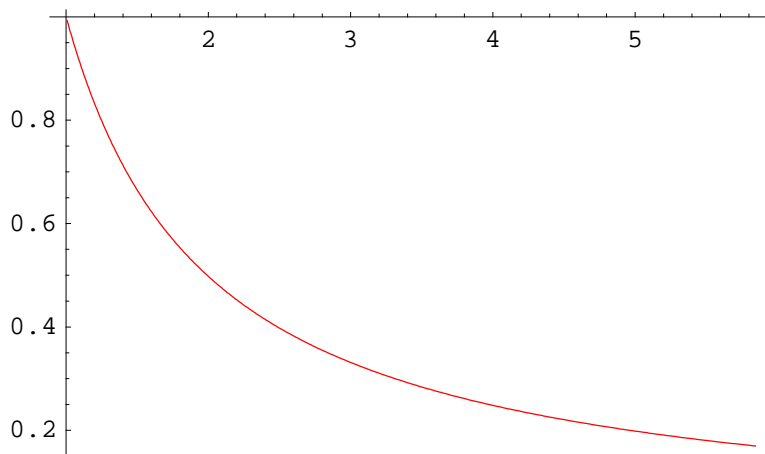
```
(*定义函数z=x^2-y^2*)
f[x_, y_] := x^2 - y^2
```

```
(*计算两个偏导数，这里采用了两种方法*)
fx[x_, y_] = D[f[x, y], x];
fy[x_, y_] = D[f[x, y], y];
```

```
(*给定初值c[0],d[0]，步长lamda*)
c[0] = 1.; d[0] = 1.; lamda = 0.01;
```

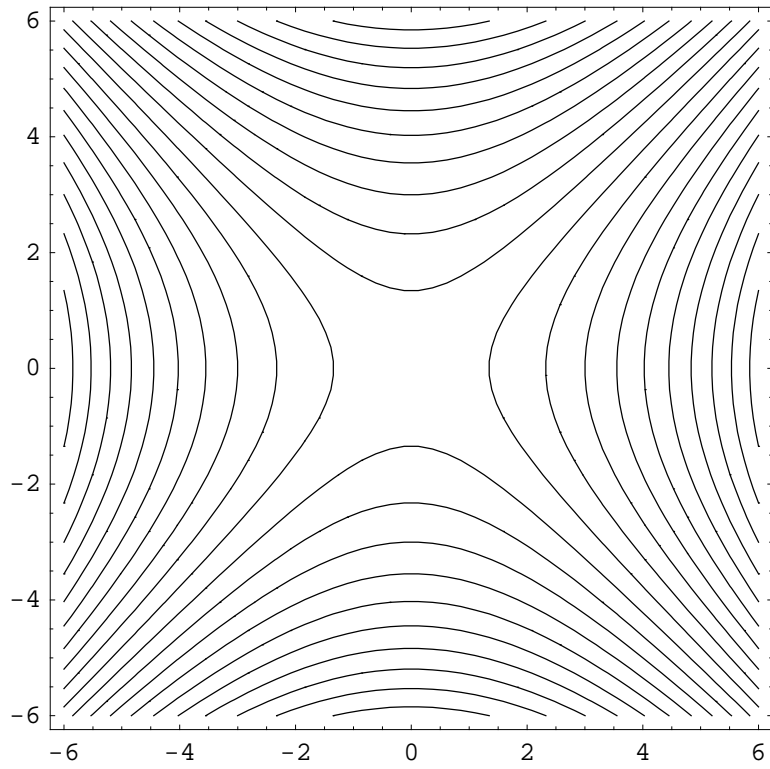
```
(*下面循环依次计算c[n],d[n] , 一直到n=500*)  
a = c[0]; b = d[0];  
Do[u = a + lamda * fx[a, b] / Sqrt[(fx[a, b])^2 + (fy[a, b])^2];  
  v = b + lamda * fy[a, b] / Sqrt[(fx[a, b])^2 + (fy[a, b])^2];  
  c[n] = u; d[n] = v; a = u; b = v, {n, 500}]
```

```
(*最后利用求出的点 (c[n],d[n]) 构造点集, 并作出梯度线*)  
data = Table[{c[n], d[n]}, {n, 500}];  
t1 = ListPlot[data, PlotJoined → True, PlotStyle → RGBColor[1, 0, 0]]
```

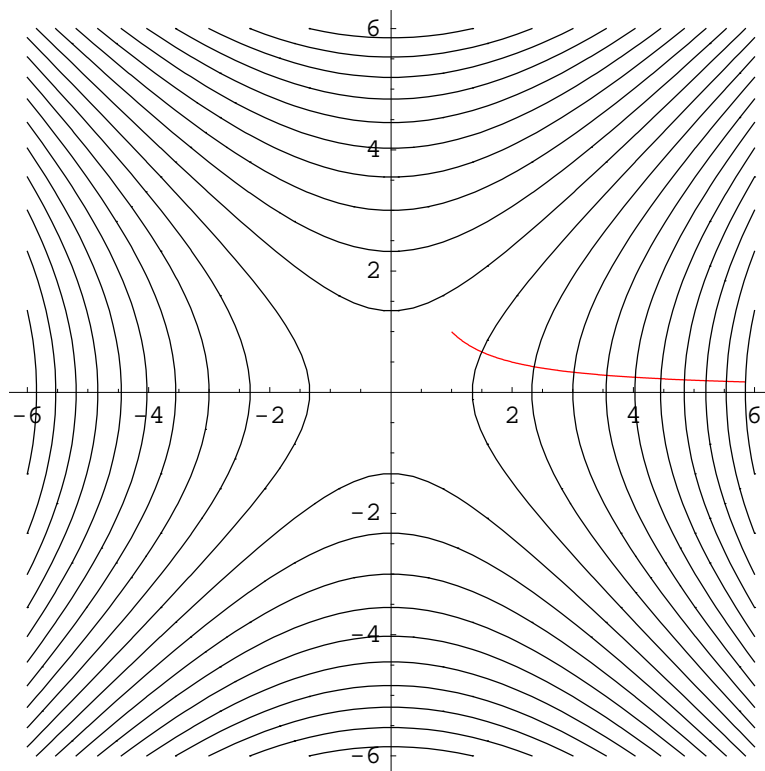


- Graphics -

```
(*再作出该函数的等量线, 并与梯度线在同一坐标系中显示*)  
t2 = ContourPlot[f[x, y], {x, -6, 6}, {y, -6, 6}, Contours → 20,  
  PlotPoints → 50, ContourShading → False]  
Show[t1, t2, AspectRatio → Automatic]
```



- ContourGraphics -



- Graphics -

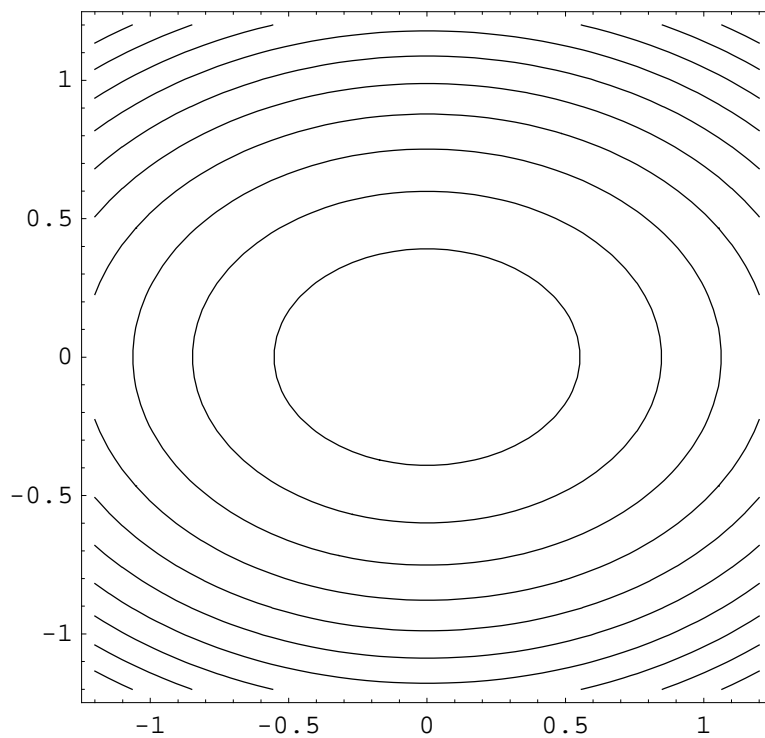
鲨鱼袭击目标的前进途径

(*定义函数*)

```
f[x_, y_] := E^(-(x^2 + 2 y^2) / 10^4)
```

(*作出该函数的等量线图形*)

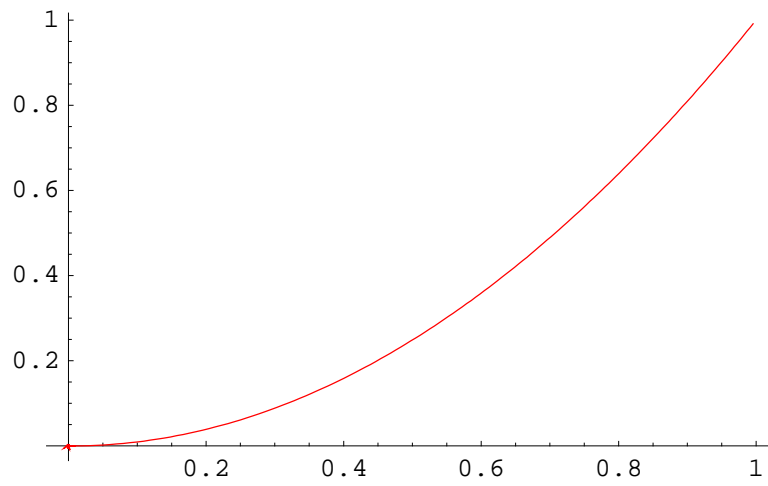
```
t1 = ContourPlot[f[x, y], {x, -1.2, 1.2}, {y, -1.2, 1.2},  
  PlotPoints → 50, ContourShading → False]
```



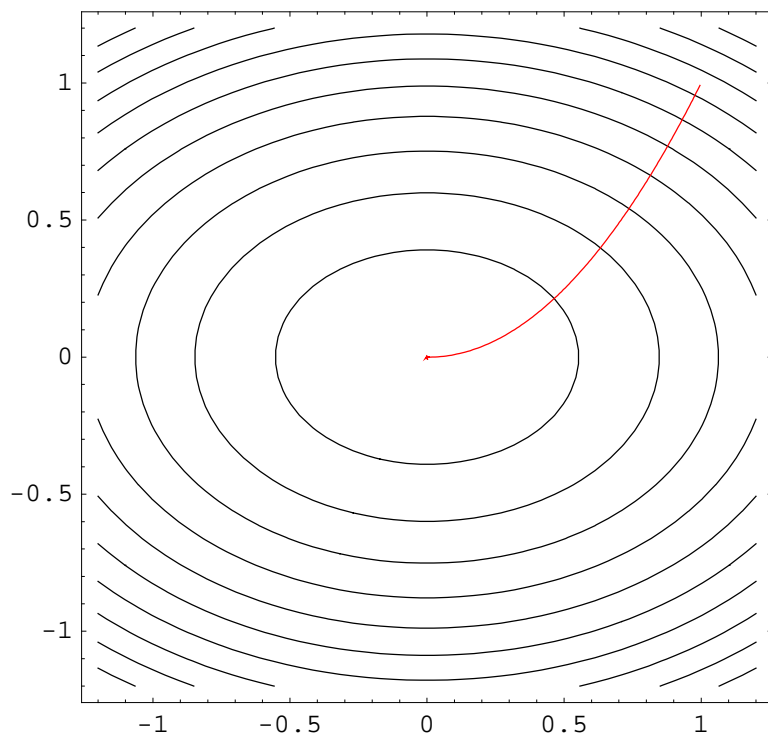
- ContourGraphics -

(*计算偏导数, 并作出梯度线*)

```
fx[x_, y_] = D[f[x, y], x];  
fy[x_, y_] = D[f[x, y], y];  
x0 = 1.; y0 = 1.; lamda = 0.01; a = x0; b = y0;  
Do[u = a + lamda * fx[a, b] / Sqrt[(fx[a, b])^2 + (fy[a, b])^2];  
  v = b + lamda * fy[a, b] / Sqrt[(fx[a, b])^2 + (fy[a, b])^2];  
  c[n] = u; d[n] = v; a = u; b = v, {n, 200}]  
data = Table[{c[n], d[n]}, {n, 200}];  
t2 = ListPlot[data, PlotJoined → True, PlotStyle → RGBColor[1, 0, 0]]  
Show[t1, t2, AspectRatio → Automatic]
```



- Graphics -

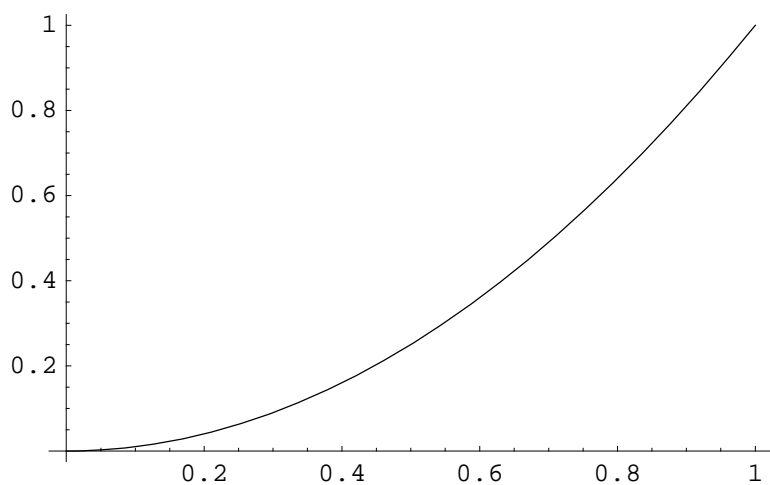


- Graphics -

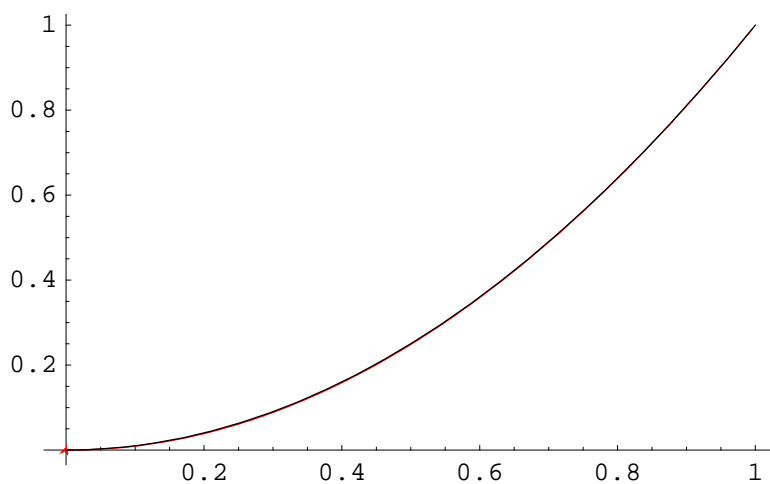
(*作出理论梯度线 $y=x^2$ ，并与前面的近似梯度线在同一坐标系中显示*)

```
t3 = Plot[x^2, {x, 0, 1}]
```

```
Show[t2, t3]
```



- Graphics -



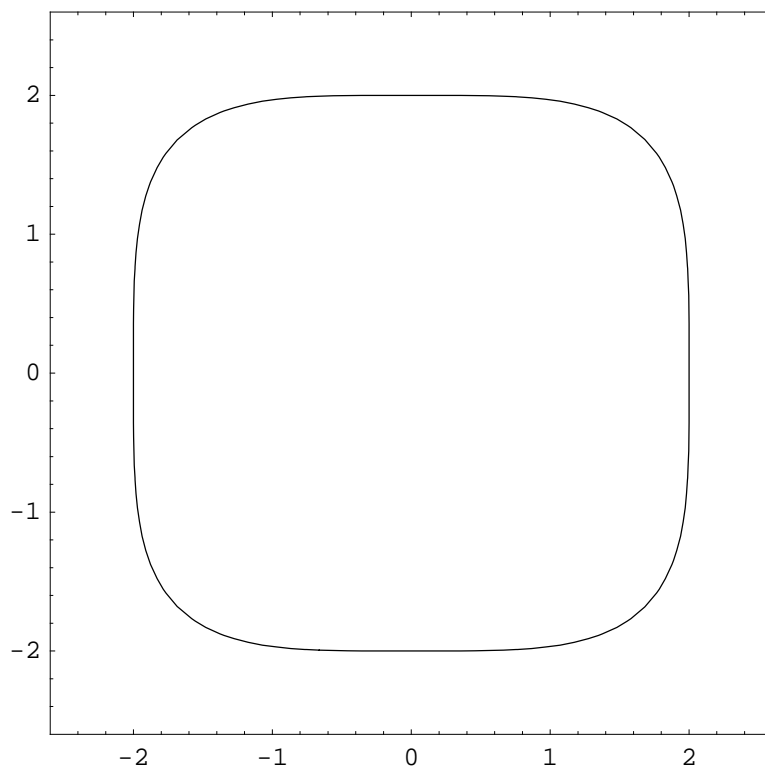
- Graphics -

一元隐函数及微分方程的积分曲线的图形

一元隐函数的图形

(*作出隐函数 $x^4+y^4=16$ 的图形, 此时必须限制值的范围为 $[16,16]$, 不显示阴影, 并且限制显示一条等高线*)

```
ContourPlot[x^4 + y^4, {x, -2.5, 2.5}, {y, -2.5, 2.5},  
PlotRange -> {16, 16}, ContourShading -> False,  
Contours -> 1, PlotPoints -> 50]
```



- ContourGraphics -

微分方程的积分曲线

(*求解微分方程*)

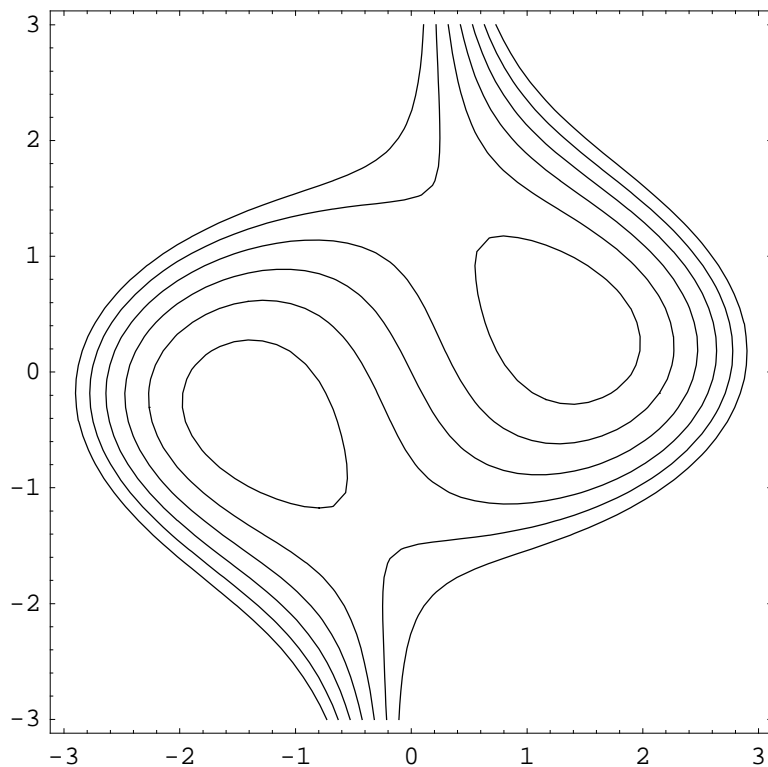
```
DSolve[(1 - 2 x * y[x]) y'[x] == x^2 + y[x]^2 - 2, y[x], x]
```

$$\left\{ \left\{ y[x] \rightarrow \frac{1}{2x} - \frac{\sqrt{\frac{1}{2} + 4x \left(x - \frac{x^3}{6}\right) + 2xC[1]}}{\sqrt{2}x} \right\}, \right. \\ \left. \left\{ y[x] \rightarrow \frac{1}{2x} + \frac{\sqrt{\frac{1}{2} + 4x \left(x - \frac{x^3}{6}\right) + 2xC[1]}}{\sqrt{2}x} \right\} \right\}$$

(*通过变形可得解为 $y - x^3/3 - x(-2 + y^2) = C$, 再用等高线的作图

命令作出积分曲线的图形, 即上述解的图形*)

```
ContourPlot[y - x^3/3 - x(-2 + y^2), {x, -3, 3}, {y, -3, 3},
PlotRange -> {-3, 3}, Contours -> 7, ContourShading -> False,
PlotPoints -> 50]
```



- ContourGraphics -

实验7 最小二乘法

刀具磨损问题

(*输入数据*)

```
yy1 = {{0, 27.0}, {1, 26.8}, {2, 26.5}, {3, 26.3},  
       {4, 26.1}, {5, 25.7}, {6, 25.3}, {7, 24.8}};
```

(*定义最小二乘的误差函数*)

```
m1[a_, b_] := Sum[(a * yy1[[k, 1]] + b - yy1[[k, 2]])^2, {k, 1, 8}]
```

(*求驻点得到最小二乘解*)

```
Solve[{D[m1[a, b], a] == 0, D[m1[a, b], b] == 0}, {a, b}]
```

```
{{a -> -0.303571, b -> 27.125}}
```

a t + b /. % (*代入最小二乘解得到拟合函数*)

```
{27.125 - 0.303571 t}
```

(*计算此时的误差*)

```
m1[-0.303571, 27.125]  
Sqrt[%]
```

```
0.108214
```

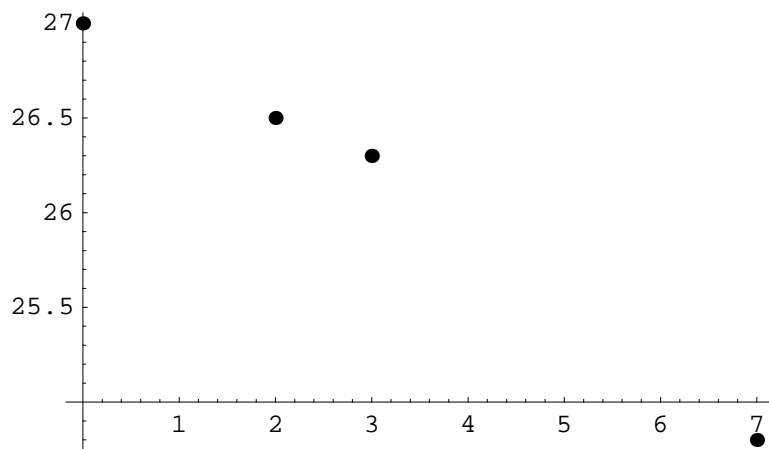
```
0.328959
```

(*直接用Mathematica求出拟合函数*)

```
Fit[yy1, {1, t}, t]
```

(*作出原数据的点图*)

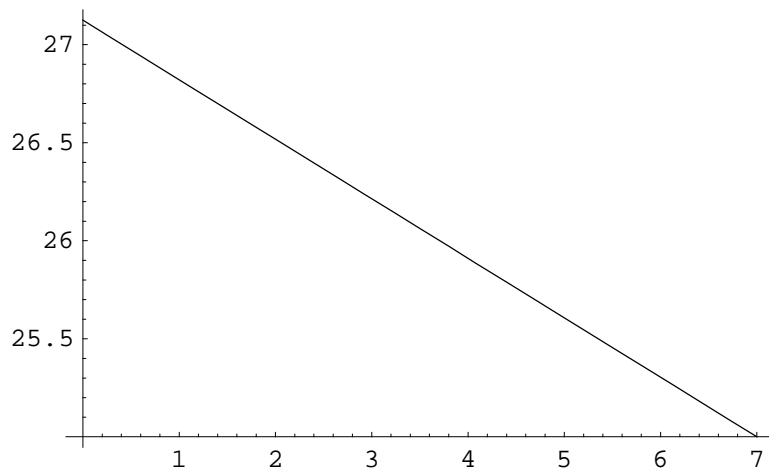
```
t1 = ListPlot[yy1]
```



- Graphics -

(*作出拟合函数的图形*)

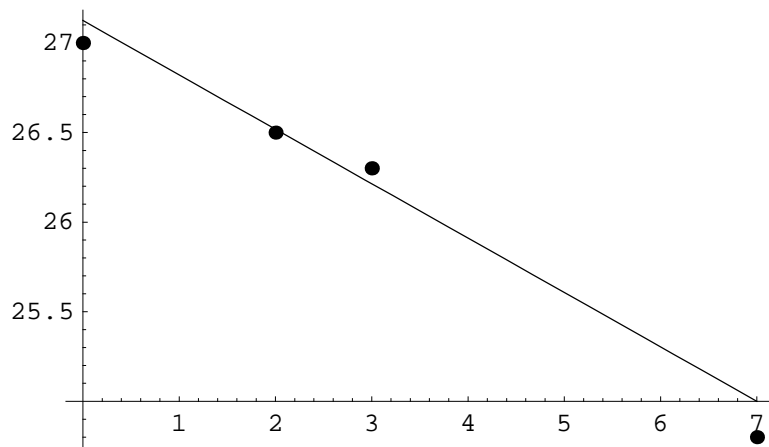
```
t2 = Plot[27.125 - 0.303571 t, {t, 0, 7}]
```



- Graphics -

(*在同一坐标系中显示点图和拟合曲线的图形*)

```
Show[t1, t2]
```



- Graphics -

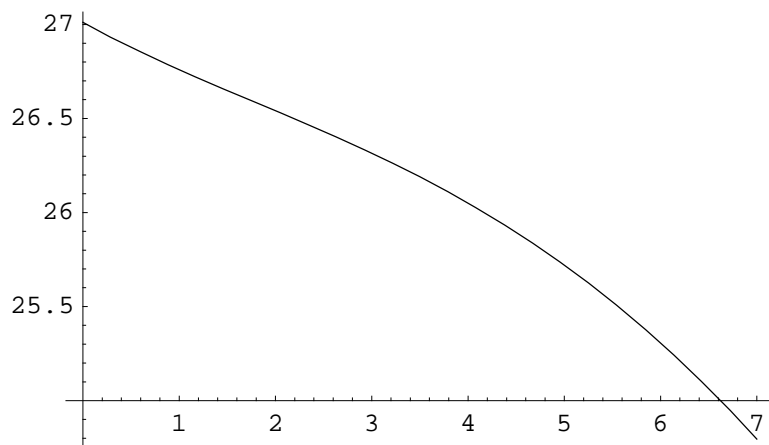
(*下面是用Mathematica求出四次拟合函数*)

```
ff[t_] = Fit[yy1, {1, t, t^2, t^3, t^4}, t]
```

$$27.011 - 0.282422 t + 0.0391098 t^2 - 0.0082702 t^3 + 0.000284091 t^4$$

(*作出四次拟合函数的图形*)

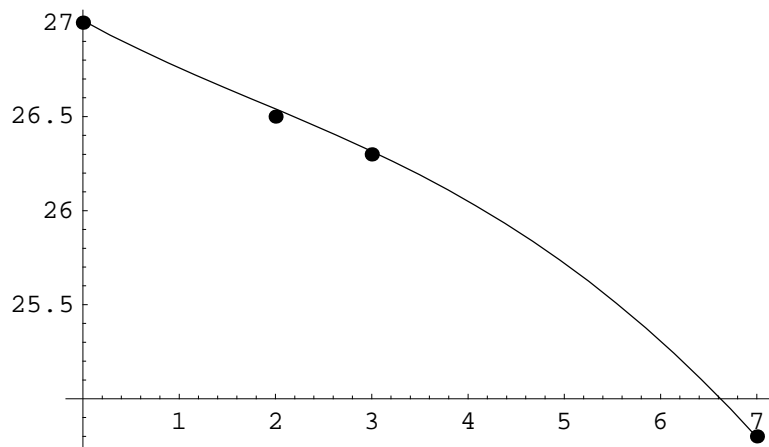
```
t3 = Plot[ff[t], {t, 0, 7}]
```



- Graphics -

(*同时显示点图和四次拟合曲线的图形*)

```
Show[t1, t3]
```



- Graphics -

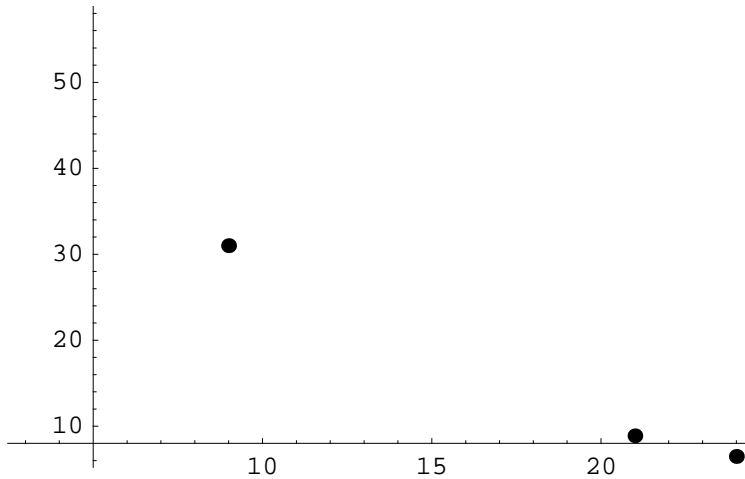
化学反应问题

(*输入数据*)

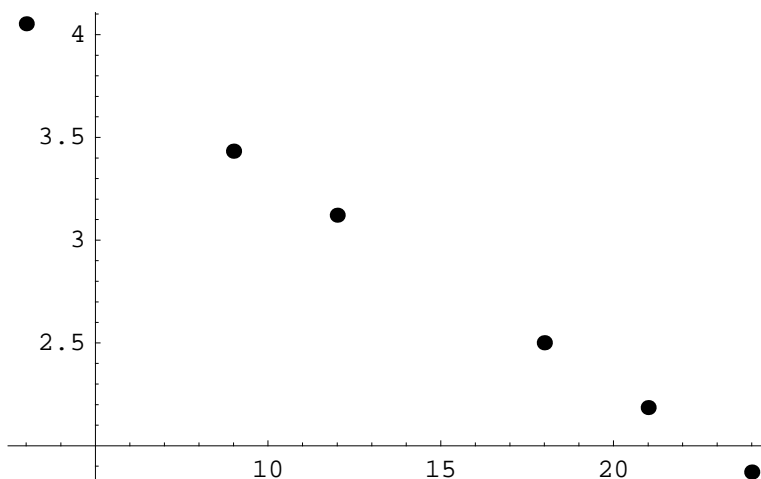
```
xy = {{3, 57.6}, {6, 41.9}, {9, 31.0}, {12, 22.7},  
      {15, 16.6}, {18, 12.2}, {21, 8.9}, {24, 6.5}};
```

(*作出点图, 判断曲线图形的基本形状*)

```
s1 = ListPlot[xy]
xy1 = Table[{xy[[k, 1]], Log[xy[[k, 2]]]}, {k, 1, 8}];
ListPlot[xy1]
```



- Graphics -



- Graphics -

(*构造最小二乘的误差函数*)

```
m[a_, b_] := Sum[(a * xy1[[k, 1]] + b - xy1[[k, 2]])^2, {k, 1, 8}]
```

(*求出驻点, 即得到最小二乘解*)

```
Solve[{D[m[a, b], a] == 0, D[m[a, b], b] == 0}, {a, b}]
```

```
{{a -> -0.103686, b -> 4.36399}}
```

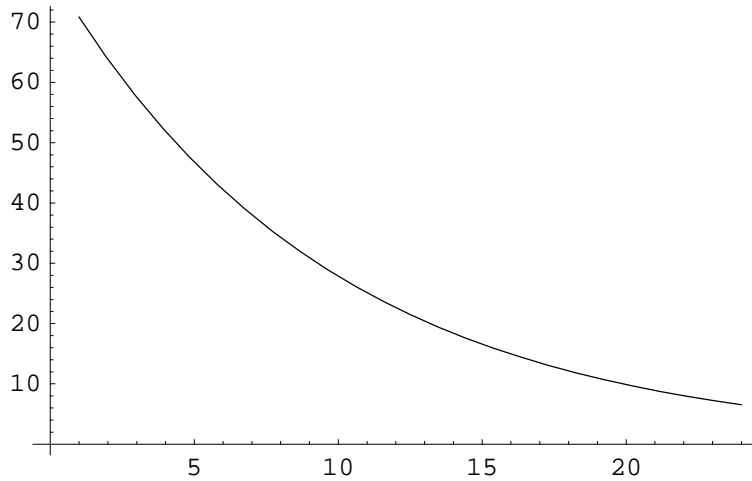
```
{E^(a*t+b), E^b} /. % (*代入得到拟合函数及相应数据*)
```

```
{{e4.36399-0.103686 t, 78.57}}
```

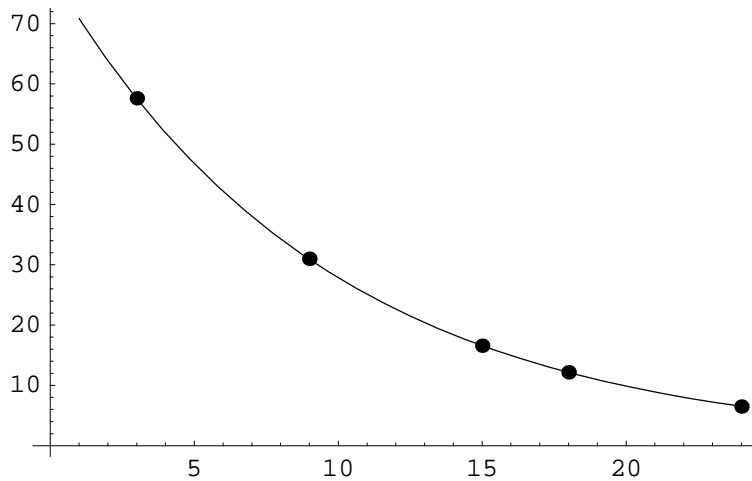
(*作出点图和最小二乘曲线的图形, 并在同一坐标系中显示*)

```
s2 = Plot[E^(4.36399 - 0.103686 t), {t, 1, 24}]
```

```
Show[s1, s2]
```



- Graphics -



- Graphics -

Mathematica的读入数据功能

(*依次读入文件1.txt中的数据*)

```
ReadList["1.txt", Number]
```

```
{1, 1, 7, 2, 2, 14, 3, 3, 21, 4, 4, 28, 5, 5,  
 37, 6, 6, 42, 7, 7, 49, 8, 8, 56, 9, 9, 63, 10, 10, 70}
```

(*按行依次读入文件1.txt中的数据*)

```
ReadList["1.txt", Number, RecordLists → True]
```

(*这里的文件1.txt需放在Mathematica的同一目录内, 该文件的格式请参见与本文件同目录的1.txt*)

```
{{1, 1, 7}, {2, 2, 14}, {3, 3, 21}, {4, 4, 28}, {5, 5, 37},  
 {6, 6, 42}, {7, 7, 49}, {8, 8, 56}, {9, 9, 63}, {10, 10, 70}}
```

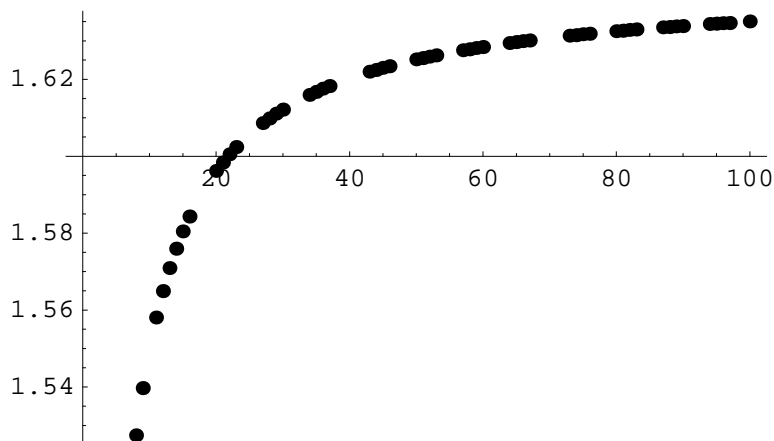
实验8 无穷级数和函数逼近

级数之和

```
s[n_] := Sum[1/k^2, {k, n}] (*定义级数的前n项和*)
```

```
data = Table[s[n], {n, 100}]; (*构成直到前100项和的一个数集*)
```

```
ListPlot[data] (*作出前n项和的点图*)
```



- Graphics -

```
N[Sum[1/k^2, {k, Infinity}]] (*求出该数列和的近似值*)
```

```
(*数值求和*)  
NSum[1/k^2, {k, Infinity}]
```

1.64493

```
(*我们知道该级数的和为 $\pi^2/6$ *)
```

```
N[Pi^2/6, 40]
```

1.644934066848226436472415166646025189219

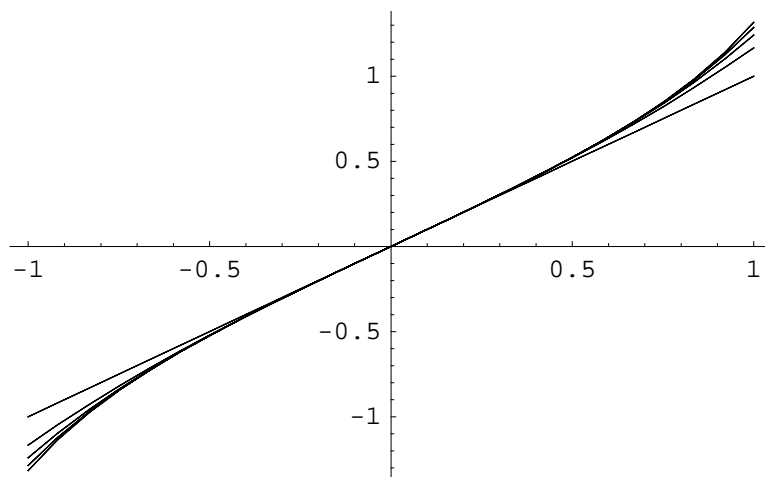
函数幂级数展开

```
Clear[f, g, h, s] (*清除变量的值*)  
(*定义函数*)  
f[x_] := ArcSin[x]
```

```
(*函数在x=0 处的n阶导数*)  
g[n_] := D[f[x], {x, n}] /. x -> 0
```

```
(*函数在x=0 处的n阶泰勒多项式*)  
s[n_, x_] := Sum[g[k] * (x^k) / k!, {k, n}]
```

```
(*构造函数直到10次的泰勒多项式，并作出它们的图形*)  
t = Table[s[n, x], {n, 1, 10}];  
Plot[Evaluate[t], {x, -1, 1}]
```



- Graphics -

幂级数应用-近似计算

下面利用幂级数计算 $(240)^{(1/5)}$ 的近似值

```
s[n_] :=
  3 (1 - 1/5 / 3^4 - Sum[Product[5 i - 1, {i, 1, k - 1}] / 5^k / k! / 3^(4 k), {k, 2, n - 1}])
r[n_] := Product[5 i - 1, {i, 1, n - 1}] / 5^n / n! / 3^(4 n - 5) / 80
delta = 10^(-10); n0 = 100;
Do[Print["n=", n, ", ", "s[n]=", N[s[n], 10]];
  If[N[r[n]] < delta, Break[]];
  If[n == n0, Print["failed"], {n, n0}]
```

n=1,s[n]=2.992592593

n=2,s[n]=2.992592593

n=3,s[n]=2.992556013

n=4,s[n]=2.992555742

n=5,s[n]=2.992555739

傅里叶级数

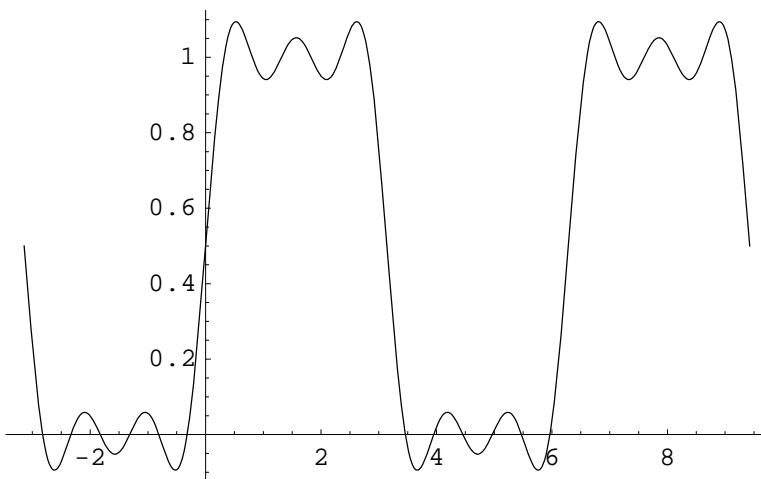
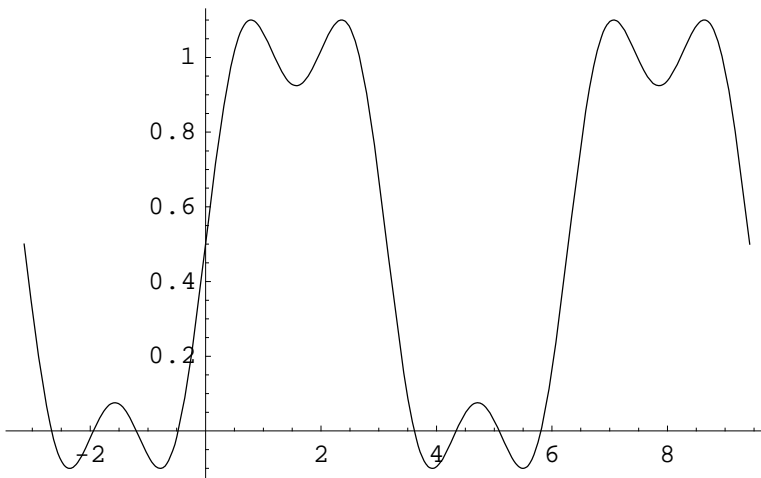
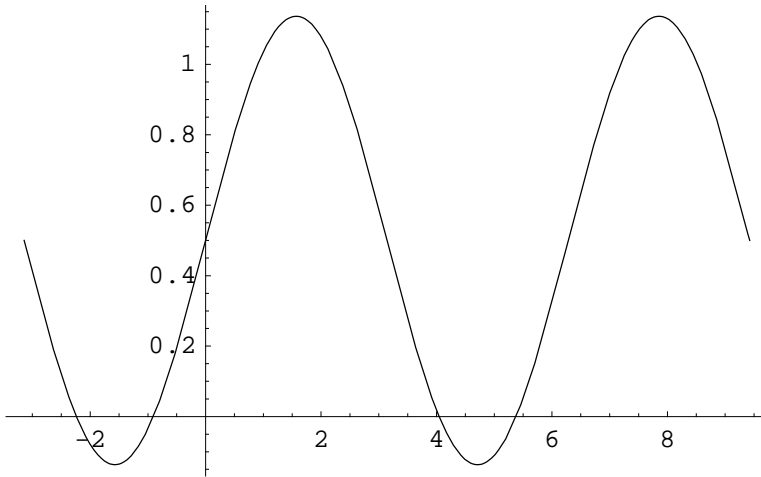
下面考察函数的傅里叶级数逼近的情况

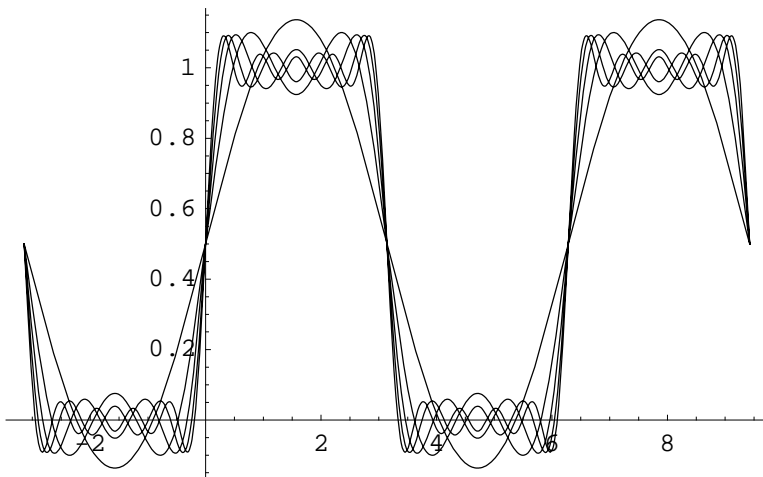
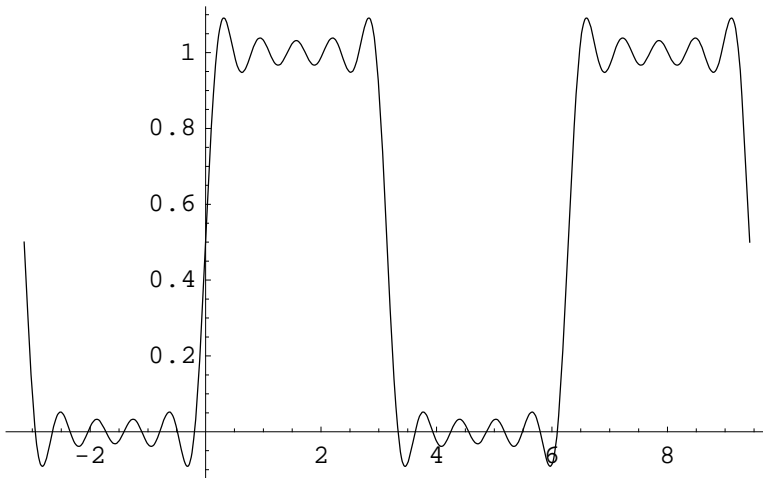
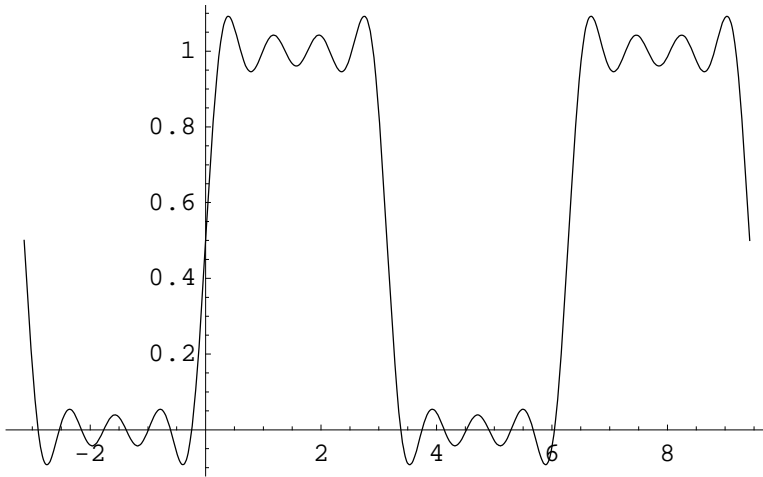
(*定义在一个周期内的函数表达式*)

```
f[x_] := If[x ≥ 0, 1, 0]
```

(*由于分段函数只能得到近似值,因此下面根据公式直接计算其傅里叶系数*)

```
Clear[a, b, n, s, t]
a[n_] := Integrate[Cos[n*t], {t, 0, Pi}] / Pi
b[n_] := Integrate[Sin[n*t], {t, 0, Pi}] / Pi
s[x_] := a[0] / 2 + Sum[a[k] * Cos[k*x] + b[k] * Sin[k*x], {k, 1, n}]
Do[Plot[Evaluate[s[x]], {x, -Pi, 3 Pi}], {n, 1, 10, 2}]
t = Table[s[x], {n, 1, 10, 2}];
Plot[Evaluate[t], {x, -Pi, 3 Pi}]
```





- Graphics -