Lab 3 实验报告

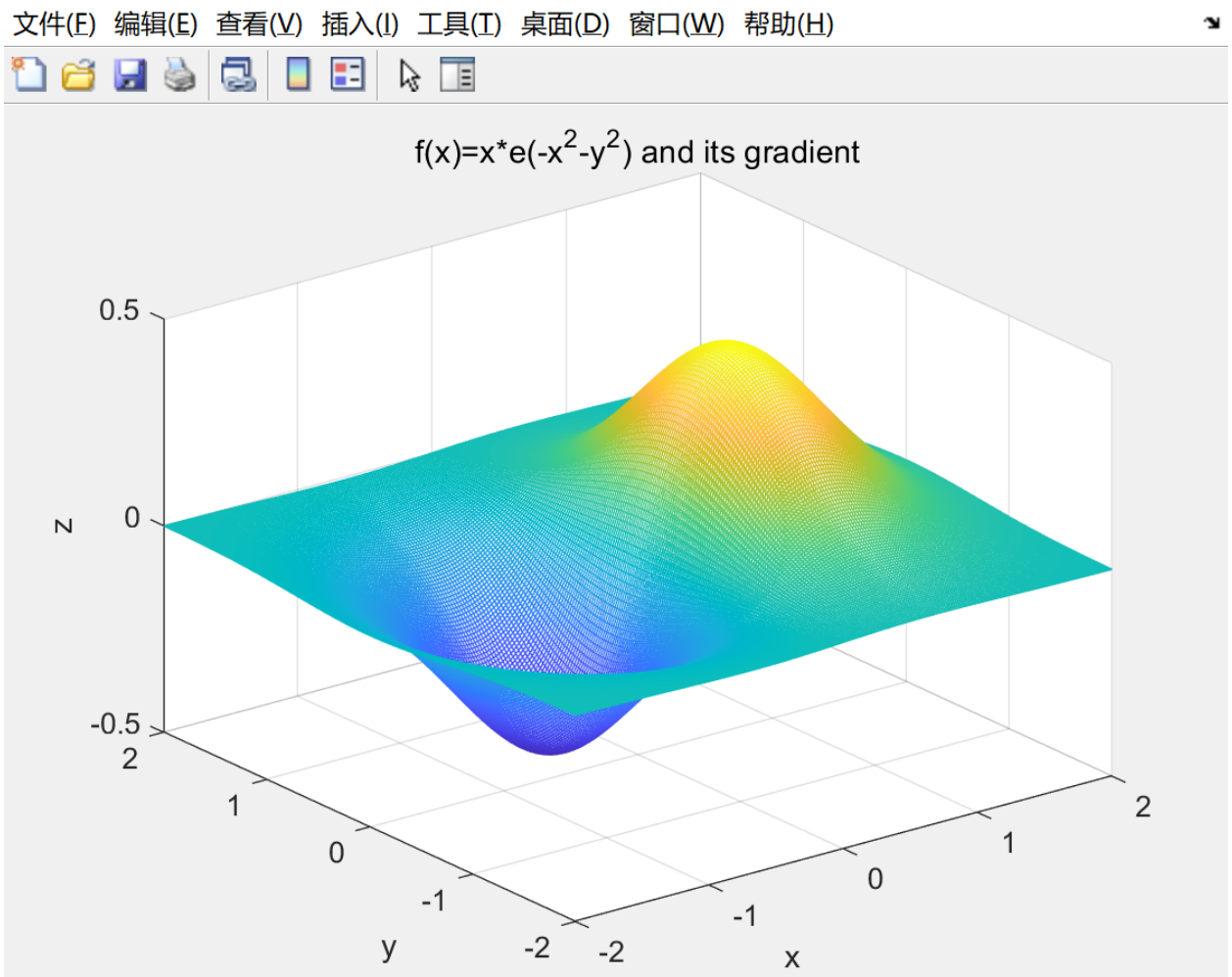
实验一

我自己设计的实验是针对函数$f(x) = xe^{-x^2-y^2}$用梯度下降找到局部最小值。代码如下

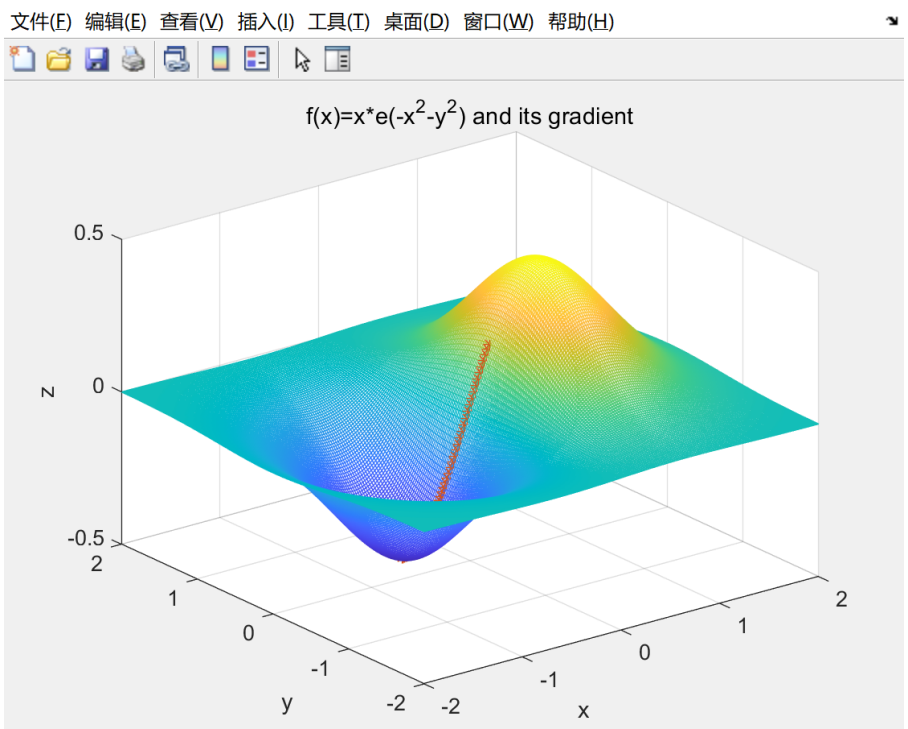
```
close,clc,clear;

x=-2:0.02:2;
y=x';
[X,Y]=meshgrid(x,y');
F = X.*exp(-X.^2-Y.^2);

surf(x,y,F);
hold on
xlabel('x');
ylabel('y');
zlabel('z');
title('f(x)=x*exp(-x^2-y^2) and its gradient');
mesh(x,y,F);

ax=gca;
ax.TickDir = 'out';
ax.TickLength = [0.02 0.02];

[fx,fy] = gradient(F,0.02);
x0 = 0.2;
y0 = 0;
z0=x0.*exp(-x0.^2-y0.^2);
alpha=0.02;
plot3(x0,y0,z0,'*');


X_grad(1)=x0;
Y_grad(1)=y0;
Z_grad(1)=z0;
n=1;

while(1)
t = (abs(x-x0)<0.01) & (abs(y-y0)<0.01);
indt = find(t);
f_grad = [fx(indt) fy(indt)];
if f_grad(1)^2+f_grad(2)^2<0.001
    break;
end
X0=[x0 y0]-alpha*f_grad;
x0=X0(1);
y0=X0(2);
z0=X0(1).*exp(-X0(1).^2-X0(2).^2);
n=n+1;
X_grad(n)=x0;
Y_grad(n)=y0;
Z_grad(n)=z0;
end

scatter3(X_grad,Y_grad,Z_grad,'*');
legend('f(x)=x*exp(-x^2-y^2)','','gradient');
text(X_grad(end),Y_grad(end),Z_grad(end),'\leftarrow it has zero gradients');

hold off
```

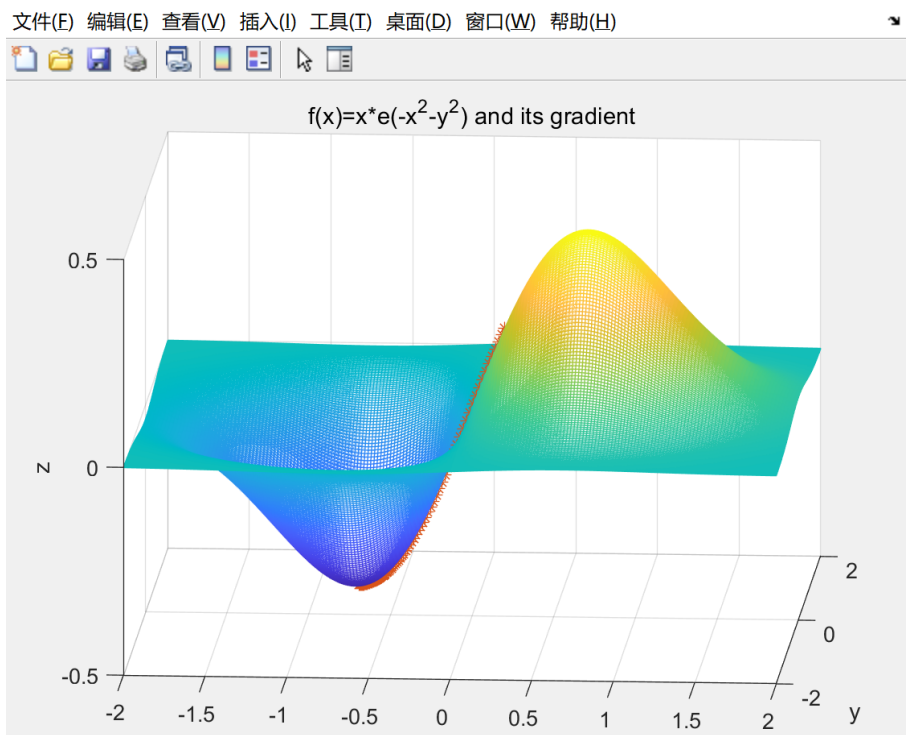
```
contour(x,y,F,'DisplayName','contour plot')
hold on

quiver(X,Y,fx,fy,'DisplayName','vector field')
xlabel('x');
ylabel('y');
title('contour plot and vector field');
```

首先，画出完整的函数图像，并用mesh上色，可以观察出函数的变化。

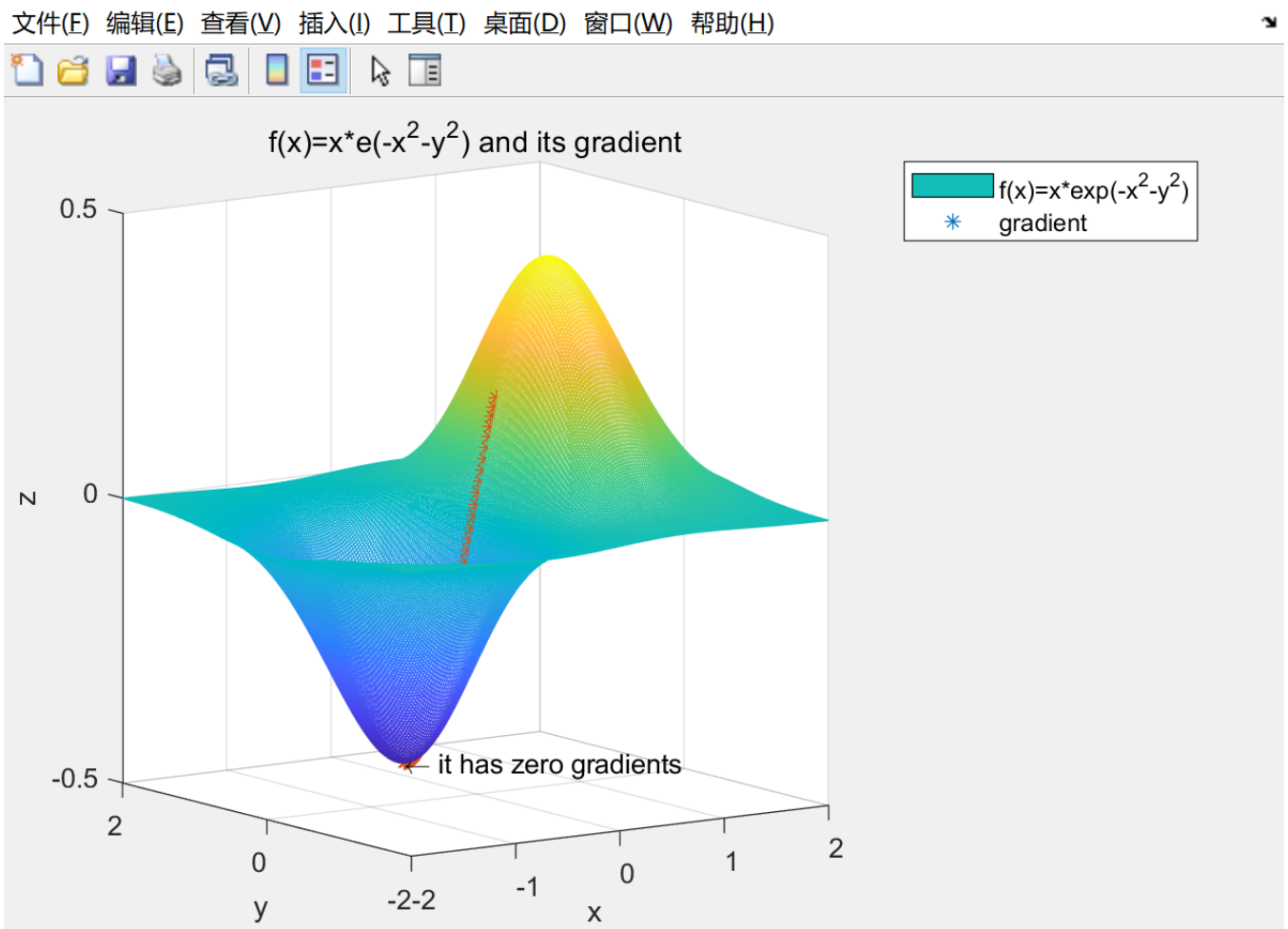


之后再用gradient函数计算F相对于x和y分别的梯度，并把学习率α设置为0.02，开始不断迭代，计算梯度下降，直至在某一点的梯度趋近于0.因为计算机浮点数精度的问题，可以设置为$\mathrm{grad}(x)^2 + \mathrm{grad}(y)^2 < 0.001$.迭代出的路径图如图所示。

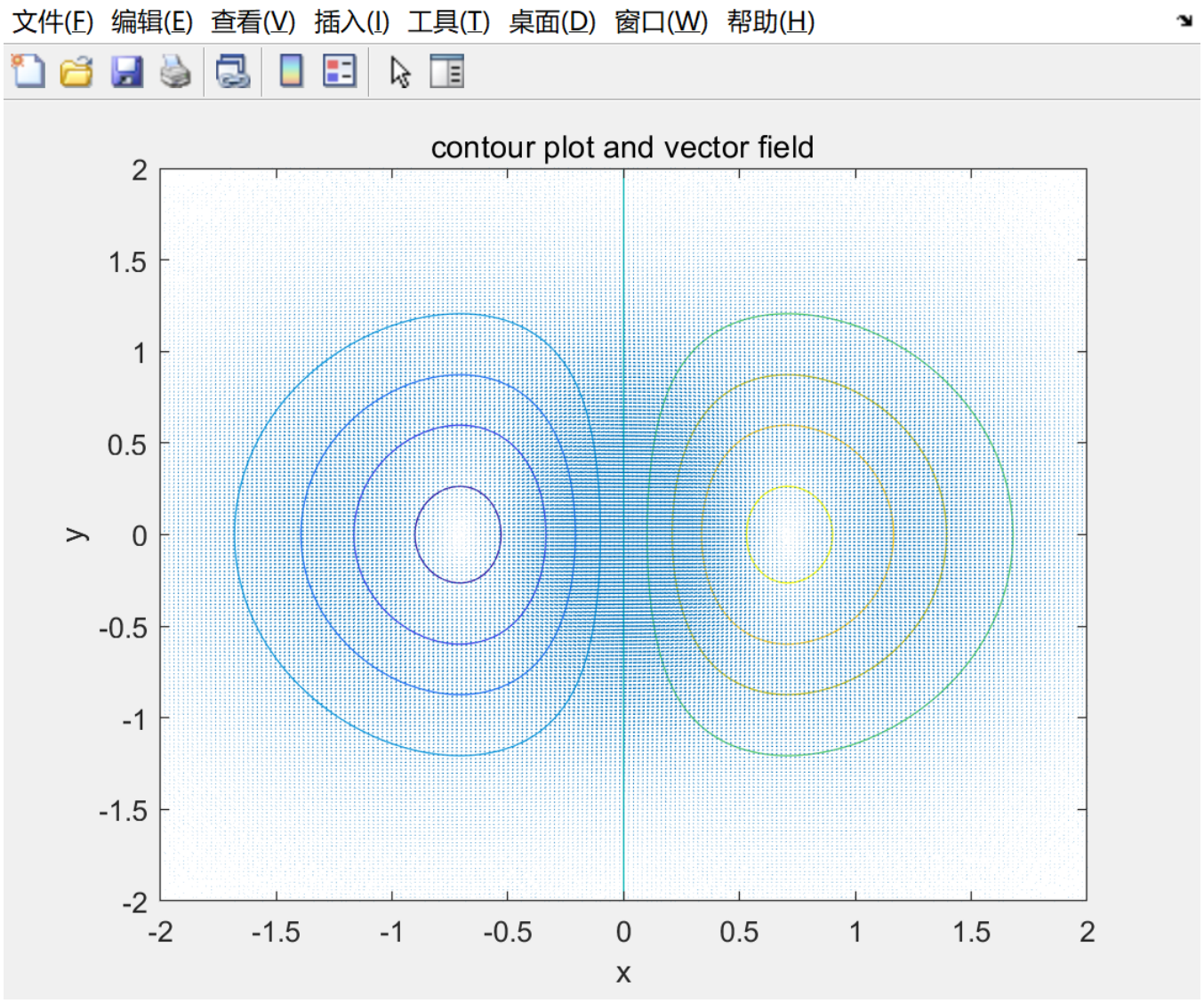




之后附上legend和标注出梯度为0的点。



最后，可以用单独的图像画出等高线和向量场。



可以发现等高线的中心区域和梯度很小的点基本重合，也符合梯度为0的点为局部的极值。

实验二

首先是写出大致的路径方程。螺旋线的半径是随着高度增加而增加。首先可以设 t 的时间的范围是$[0, 8\pi]$。

半径会随着时间增大，则设$x(t) = t \times \sin(t)$，$y(t) = t \times \cos(t)$，$z(t) = t$.

算出每个时间点的坐标，之后再画图。

第一种方法是用for循环来实现，每次都会打印1~k个点的曲线

```
close,clc,clear;

t=linspace(0,8*pi,100);
x_t=t.*sin(t);
y_t=t.*cos(t);
z_t=t;
for k=1:length(t)
    plot3(x_t(1:k),y_t(1:k),z_t(1:k),'b-');
     axis([-30,30,-30,30,0,30]);
     xlabel('x');
     ylabel('y');
     zlabel('z');
     title('Using for loop');
%     frame(k)=getframe(gcf);
    pause(0.001);
end

% printgif('task2_1_gif.gif',frame,0.001);
```

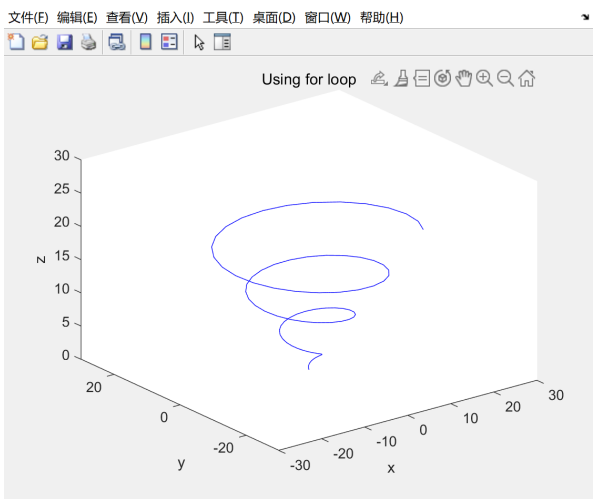
第二种方法使用comet函数来实现的

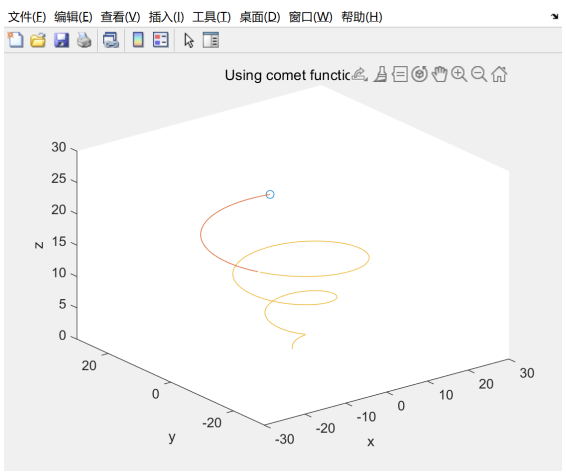
```
close,clc,clear;

t=linspace(0,8*pi,1000);
x_t=t.*sin(t);
y_t=t.*cos(t);
z_t=t;

plot3(0,0,0);
axis([-30,30,-30,30,0,30]);
hold on
xlabel('x');
ylabel('y');
zlabel('z');
title('Using comet function');
comet3(x_t,y_t,z_t);
```

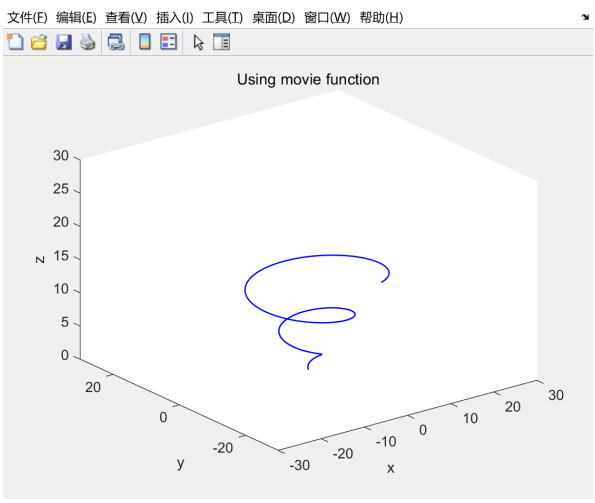
第三种方式用跟movie相关的函数实现的，获取每一帧储存之后，用movie函数进行播放，并也可以通过printgif储存起来。

```
t=linspace(0,8*pi,1000);
x_t=t.*sin(t);
y_t=t.*cos(t);
z_t=t;

M=moviein(1000);

for k=1:length(t)
    plot3(x_t(1:k),y_t(1:k),z_t(1:k),'b-','linewidth',1);
    hold on
    axis([-30,30,-30,30,0,30]);
    xlabel('x');
    ylabel('y');
    zlabel('z');
    title('Using movie function');
    pause(1/1000),
    M(k)=getframe(gcf);
end
movie(M);
printgif('task2_3.gif',M,0.001);
```

以下是截取的一些动画过程中的图像







最后呈现的动图是task2_3.gif

实验三

实验三用view可以从不同角度观察图形，并把每一个角度记录在每一帧。

```
close,clc,clear;

t=linspace(0,8*pi,100);
x_t=t.*sin(t);
y_t=t.*cos(t);
z_t=t;

plot3(x_t,y_t,z_t,'b-');
axis([-30,30,-30,30,0,30]);
xlabel('x');
ylabel('y');
zlabel('z');
title('different view of curve');

M=moviein(360);
for i=0:359
    view(i,25);
    pause(0.05)
    M(i+1)=getframe(gcf);
end

printgif('task3.gif',M,0.05);
```

这里记录了从0~359度，这样整个动图显得比较连续，并可以旋转一周。

这里的动图是task3.gif