21

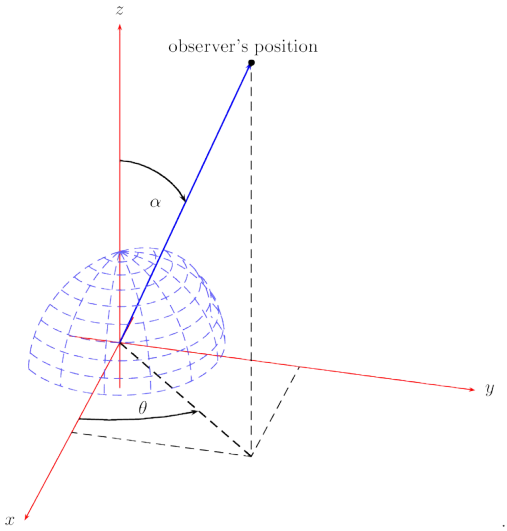
Three-dimensional Plots

Creating figures in 3D space with Scilab is based on the definition of points that are analogous to those used in a plane, but that have three coordinates (usually denoted `x,y,z`) instead of two.

21.1. View angle

The main difference compared to a planar plot is that in order to display a 3D plot on the screen, the figure needs to be projected onto a plane (the plane of your computer's screen). The result will strongly depend on the angle at which the figure is viewed. This view angle is defined by two values (from now on denoted α and θ) which are angles expressed in degrees. You will often need the help of the diagram in Figure 21.1 to pick the correct angle values that define the view origin you wish to use.

Figure 21.1 : *Viewing point as a function of angles α and θ*



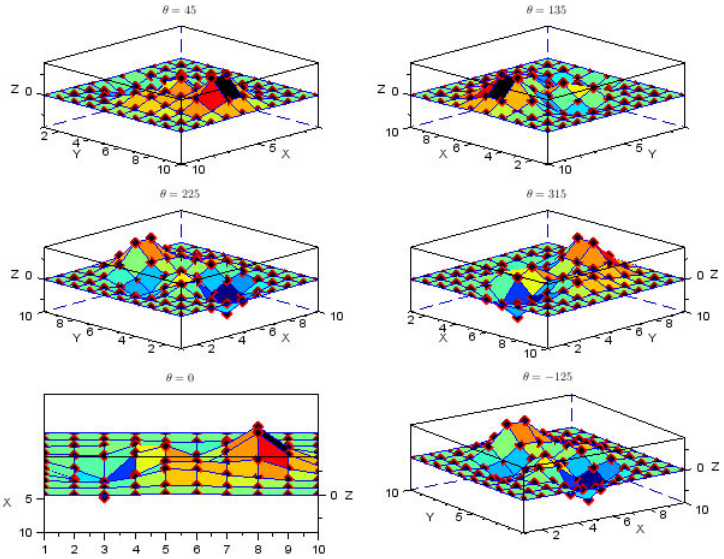
This data is stored inside the Axes handle's `rotation_angles` property for a given plot. For example, for a figure displayed by using the `surf()` command (see Example 19.1), these angles are `alpha=51` and `theta=-125`, respectively.

```
-->clf;

-->surf()

-->A=gca();

-->A.rotation_angles
 ans  =
    51.   - 125.

-->alpha=A.rotation_angles(1)
 alpha  =
    51.

-->theta=A.rotation_angles(2)
 theta  =
  - 125.
```

Most of Scilab's 3D plotting functions accept optional inputs of the format `alpha=value` and `theta=value` to define the view point for the way the figure gets displayed. To understand the link between these values and the view direction, modify the values of the `rotation_angles` property and check that:

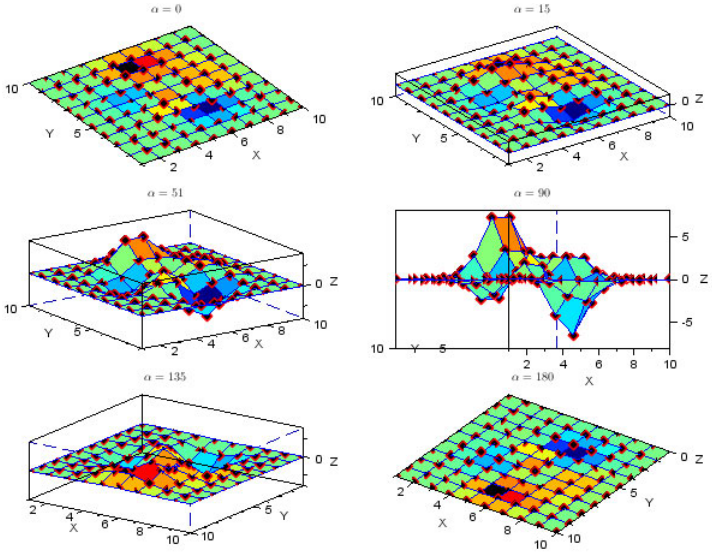
- `theta` can range from `0` to `360` degrees with:
  - `theta=0` view from the direction `x=0` and `y>0`
  - `theta=90` view from the direction `x>0` and `y=0`
  - `theta=180` view from the direction `x=0` and `y<0`
  - `theta=270` view from the direction `x<0` and `y=0`

Figure 21.2 : *Appearance of figure* `surf()` *for different values of* `theta`



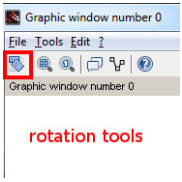
- `alpha` can range from `0` to `180` degrees with:
  - `alpha=90` view from the `z=0` plane
  - `alpha<90` view from a point of coordinate `z>0` (meaning "above" the `z=0` plane)
  - `alpha>90` view from a point of coordinate `z<0` (meaning "below" the `z=0` plane)

Figure 21.3 : Appearance of figure `surf()` for different values of `alpha`



Tip › *You can modify the viewing direction by using:*

- *the mouse (keep the right mouse button pressed, then move the mouse to rotate the figure)*
- *the Rotate tool in the graphics window (see the figure below):*
  - *from the TOOLS menu in the menu bar*
  - *the Rotate icon in the toolbar*



Zooming is performed the same way as for planar figures.

Caution › *The values of* `theta` *and* `alpha` *are angles (in degrees) and as such are defined with the modulo* `360`*.*

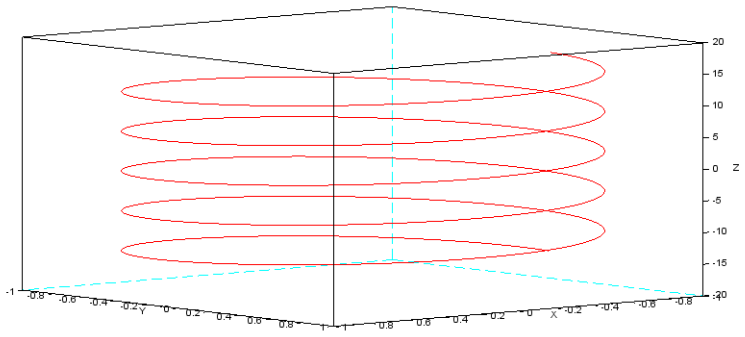
21.2. Curves in 3D space

To plot curves in 3D space, you will use the command `param3d`. For example, execute the following script to get the helix displayed in Figure 21.4.

```
function [x,y,z]=helix(t)
    x=cos(t)
    y=sin(t)
    z=t
endfunction
// compute coordinates of points
t=[-5*%pi:0.02:5*%pi];
[x,y,z]=helix(t);
// display the curve
clf;
param3d(x,y,z,alpha=15,theta=50)
E=gce();E.foreground=5 // modify the curve's color
```

The coordinate of points of the helix are calculated with the `helix` function, from a vector of values of the parameter `t` and are then displayed with `param3d`. To modify the curve's appearance, you need to retrieve the current handle (of type *polyline*) right after calling `param3d`. Here, we have assigned the number corresponding to the color red to the `foreground` property.

Figure 21.4 : *Plotting a curve in 3D space with* `param3d`

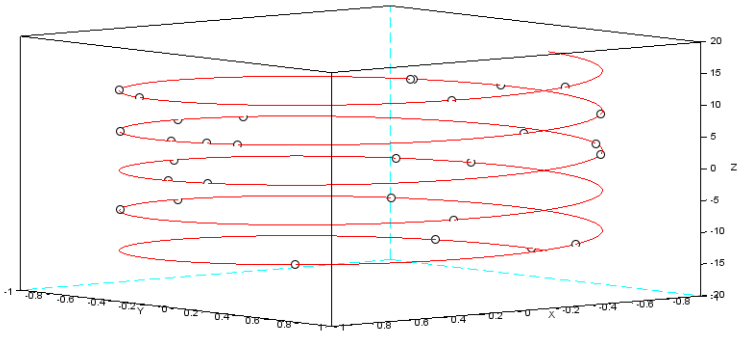


If you wish to specify the style of the curve to use, it is preferable to use the command `param3d1`. The syntax is a little more complicated: you need to replace the `z` coordinate with `list(z,style)` where style is the number which indicates the color or marker to use for the plot. For example, execute the following script to get Figure 21.5.

```
function [x,y,z]=helix(t)
    x=cos(t)
    y=sin(t)
    z=t
endfunction
// compute coordinates of points
t=[-5*%pi:0.02:5*%pi];
[x,y,z]=helix(t);
// display the curve
clf;
param3d(x,y,z,alpha=15,theta=50)
E=gce();E.foreground=5 // modify the curve's color
// display points marked by a "O"
t=10*%pi*grand(30,1,'def')-5*%pi;
[x,y,z]=helix(t);
param3d1(x,y,list(z,-9),alpha=15,theta=50)
```

Note that the `-9` in `list(z,-9)` corresponds to markers o according to the table in Figure 20.14.

Figure 21.5 : *Plotting a curve in 3D space with* `param3d1`



21.3. Facets and surfaces

Plotting surfaces in 3D with Scilab is based on drawing facets, which play the same role as segments for planar figures. This geometric structure is defined by a list of points and their associated cartesian coordinates. Scilab's graphics functions take as argument these lists of points stored inside vectors or matrices which they use to display the corresponding facets as in Figure 21.6.