21

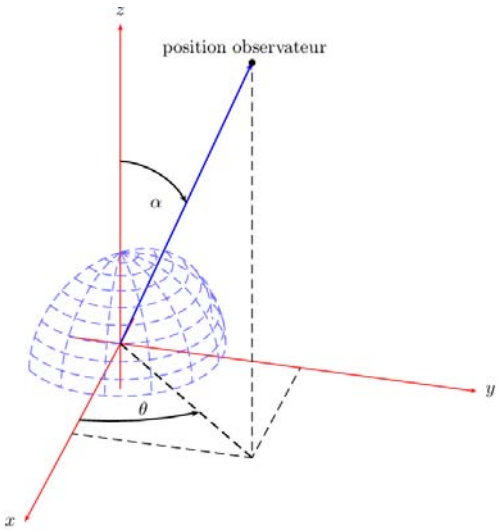
Tracés en trois dimensions

La réalisation de figures dans l'espace avec Scilab repose sur une description des points analogue à celle des figures planes mais utilisant trois coordonnées (notées `x,y,z` en général) au lieu de deux.

21.1. Direction d'observation

La principale différence avec le tracé de figure plane réside dans le fait que pour être affichée à l'écran une figure dans l'espace doit être projetée sur un plan (celui de l'écran de votre ordinateur). Le résultat dépendra donc fortement de l'angle sous lequel la figure est observée. Cette direction d'observation est décrite par deux valeurs (notées α et θ dans la suite) qui doivent être entendues comme des angles exprimés en degrés. Vous aurez souvent à vous aider du schéma de la Figure 21.1 pour choisir les valeurs des angles définissant le point de vue que vous souhaitez utiliser.

Figure 21.1 : *Point d'observation en fonction des angles α et θ*



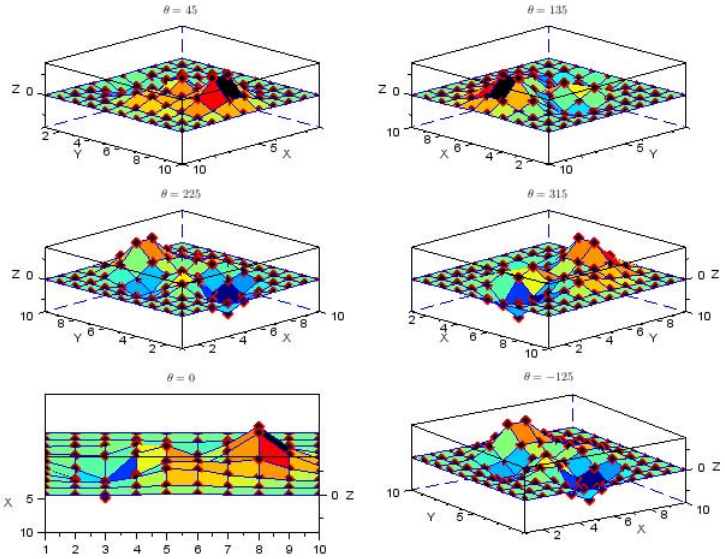
Ces données sont stockées dans la propriété `rotation_angles` du handle Axes associé au tracé. Par exemple, pour la figure affichée par la commande `surf()` (voir Exemple 19.1), ces angles sont respectivement `alpha=51` et `theta=-125`.

```
-->clf;

-->surf()

-->A=gca();

-->A.rotation_angles
 ans  =

    51.  - 125.

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

    51.

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

  - 125.
```

La plupart des fonctions de dessin en trois dimensions de Scilab acceptent des arguments optionnels de la forme `alpha=valeur` et `theta=valeur` pour définir le point de vue selon lequel la figure doit être affichée. Pour comprendre le lien entre ces valeur et la direction d'observation, modifiez les valeurs de la propriété `rotation_angles` et vérifiez que :

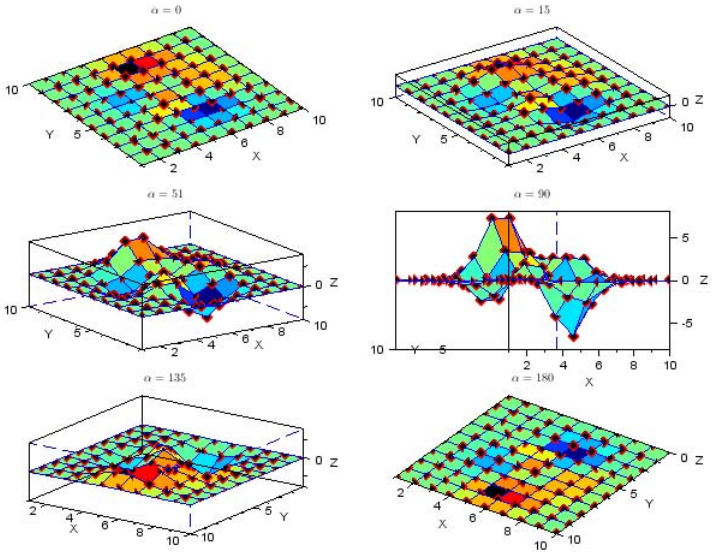
- `theta` peut varier entre `0` et `360` degrés avec :
  - `theta=0` vision depuis la direction `x=0` et `y>0` ;
  - `theta=90` vision depuis la direction `x>0` et `y=0` ;
  - `theta=180` vision depuis la direction `x=0` et `y<0` ;
  - `theta=270` vision depuis la direction `x<0` et `y=0`.

Figure 21.2 : *Aspect de la figure* `surf()` *suivant la valeur de* `theta`



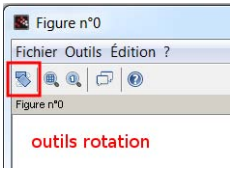
- `alpha` peut varier entre `0` et `180` degrés avec :
  - `alpha=90` vision depuis le plan `z=0` ;
  - `alpha<90` vision depuis un point de coordonnée `z>0` (donc "au-dessus" du plan `z=0`) ;
  - `alpha>90` vision depuis un point de coordonnée `z<0` (donc "au-dessous" du plan `z=0`).

Figure 21.3 : Aspect de la figure `surf()` suivant la valeur de `alpha`



Astuce *› Vous pouvez modifier la direction d'observation en utilisant :*

- *la souris (maintenir le bouton droit enfoncé puis déplacer la souris pour faire tourner la figure) ;*
- *l'outil Rotation de la fenêtre graphique (voir figure ci-dessous) :*
  - *depuis le menu OUTILS de la barre de menus ;*
  - *l'icône Rotation de la barre d'outils ;*



Le zoom s'effectue de la même manière que pour les figures planes.

Attention *› Les valeurs de* `theta` *et* `alpha` *sont des angles (en degrés) et sont donc définies modulo* `360`*.*

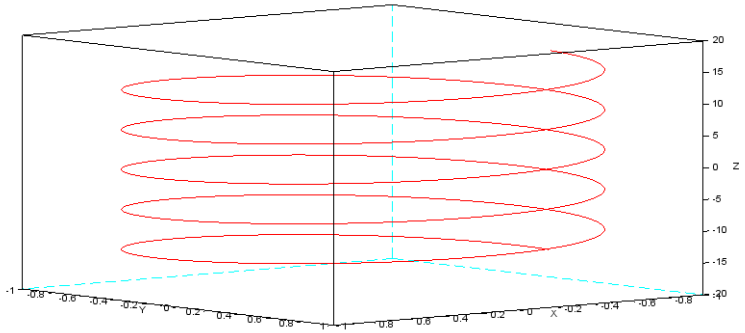
21.2. Les courbes dans l'espace

Pour tracer des courbes dans l'espace, vous utiliserez la commande `param3d`. Par exemple, exécutez le script suivant pour obtenir l'hélice dessinée à la Figure 21.4.

```
function [x,y,z]=helice(t)
    x=cos(t)
    y=sin(t)
    z=t
endfunction
// calcul des coordonnées des points
t=[-5*%pi:0.02:5*%pi];
[x,y,z]=helice(t);
// affichage de la courbe
clf;
param3d(x,y,z,alpha=15,theta=50)
E=gce();E.foreground=5 // modifier la couleur de la courbe
```

Les coordonnées des points de l'hélice sont calculées par la fonction `helice`, à partir d'un vecteur de valeurs du paramètre `t`, et sont ensuite affichées avec `param3d`. Pour modifier l'apparence du tracé, il faut récupérer le handle courant (de type *polyline*) juste après `param3d`. Ici, on a affecté à la propriété `foreground` le numéro de couleur correspondant au rouge.

Figure 21.4 : *Tracé d'une courbe dans l'espace avec* `param3d`



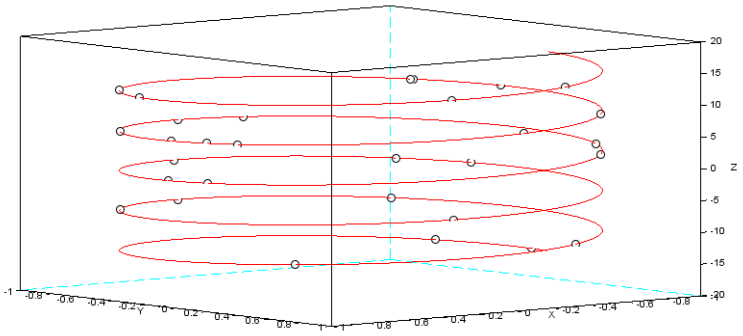
Si vous souhaitez spécifier le style de tracé à utiliser pour la courbe, utilisez plutôt la commande `param3d1` . La syntaxe est un peu plus complexe : vous devez remplacer la coordonnée `z` par `list(z,style)` où style est le numéro indiquant la couleur ou la marque à utiliser pour le tracé. Par exemple, exécutez le script suivant pour obtenir la Figure 21.5.

```
function [x,y,z]=helice(t)
    x=cos(t)
```

```
    y=sin(t)
    z=t
endfunction
// calcul des coordonnées des points
t=[-5*%pi:0.02:5*%pi];
[x,y,z]=helice(t);
// affichage de la courbe
clf;
param3d(x,y,z,alpha=15,theta=50)
E=gce();E.foreground=5 // modifier la couleur de la courbe
// placer des points marqués par un "O"
t=10*%pi*grand(30,1,'def')-5*%pi;
[x,y,z]=helice(t);
param3d1(x,y,list(z,-9),alpha=15,theta=50)
```

Remarquez que le `-9` dans `list(z,-9)` correspond aux marques de `o` d'après la table de la Figure 20.14.

Figure 21.5 : *Tracé d'une courbe dans l'espace avec* `param3d1`



21.3. Facettes et surfaces

Le tracé de surfaces dans l'espace avec Scilab repose sur le tracé de facettes, qui jouent le même rôle que les segments pour les figures planes. Cette structure géométrique est décrite par des listes de points, identifiés à leurs coordonnées cartésiennes. Les fonctions graphiques de Scilab prendront en argument ces listes de points, stockées dans des vecteurs ou des matrices, pour ensuite afficher les facettes correspondantes comme à la Figure 21.6.