TP15 : Représentation graphique de fonctions de deux variables

▶ Dans votre dossier `Info_2a`, créez le dossier `TP_15`.

I. Brève présentation des fonctions de deux variables

- On appelle **fonction de deux variables à valeurs réelles** toute fonction f définie sur une partie A de $\mathbb{R}^2$ et à valeurs dans $\mathbb{R}$.

$$f \ : \left| \begin{array}{ccc} A & \to & \mathbb{R} \\ (x,y) & \mapsto & f(x,y) \end{array} \right.$$

- L'ensemble des éléments $(x,y) \in \mathbb{R}^2$ en lesquels la fonction est f est définie est appelé **ensemble de définition** de f et est noté $\mathcal{D}_f$.
- Représenter f dans l'espace c'est représenter la **surface** S_f définie par les points (x,y,z) où $(x,y) \in \mathcal{D}_f$ et $z = f(x,y)$.

II. Représentation en Scilab

II.1. Avant-propos

- Considérons une fonction à deux variables f, et $(x,y) \in \mathscr{D}_f$.
- Formellement, représenter S_f nécessite de calculer $f(x,y)$ pour tout point $(x,y) \in \mathscr{D}_f$.
- $\mathscr{D}_f$ peut contenir un nombre infini de points : on se heurte alors à une limitation machine (la mémoire d'un ordinateur est finie). On se contente donc d'une représentation approchée de S_f : on sélectionne un nombre fini de points $(x,y) \in \mathscr{D}_f$ pour lesquels on va calculer $z = f(x,y)$. La représentation **Scilab** de S_f est obtenue en reliant ces points entre eux.

Pour ce faire :

× on se donne un vecteur `x` de valeurs de taille `n`, *(par exemple :* `x=linspace(-2,5)`*)*

× on se donne un vecteur `y` de valeurs de taille `p`, *(par exemple :* `y=-4:0.1:7`*)*

× on calcule la matrice `z` de taille `n` × `p` défini par :

$$\forall \texttt{i} \in [\![1,\texttt{n}]\!], \forall \texttt{j} \in [\![1,\texttt{p}]\!],\ \texttt{z(i,j)} = \texttt{f(x(i),y(j))}$$

II.2. Mise en œuvre

Dans la suite, on considère les fonctions f et g suivantes.

$$f \ : \left| \begin{array}{ccc} \mathscr{D}_f & \to & \mathbb{R} \\ (x,y) & \mapsto & \sqrt{x^2+y^2-1} \end{array} \right. \qquad g \ : \left| \begin{array}{ccc} \mathscr{D}_g & \to & \mathbb{R} \\ (x,y) & \mapsto & x\,\mathrm{e}^{-x^2-y^2} \end{array} \right.$$

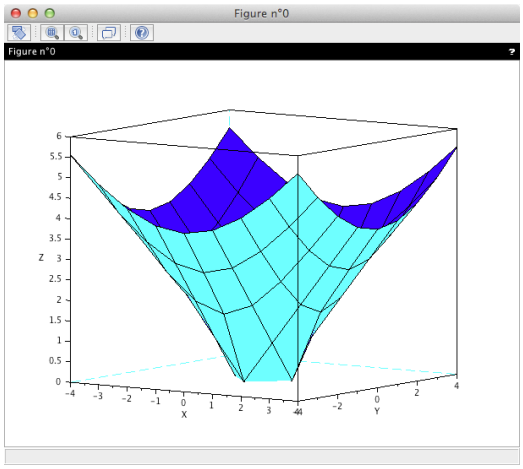
▶ Déterminer l'ensemble de définition de f et de g.

II.2.a) La fonction `fplot3d`

Afin d'illustrer le fonctionnement de la commande `fplot3d`, on donne les programmes suivants ainsi que le tracé obtenu à l'aide de ces programmes.

```
1  // La fonction f du TP21
2  function z = f(x,y)
3      z = sqrt(x^2 + y^2-1)
4  endfunction
```

```
1  // Intervalles de tracé
2  x=-4:1:4
3  y=-4:1:4
4  // Appel à la fonction de tracé
5  scf()
6  fplot3d(x,y,f)
```



- Dans votre dossier `TP_21`, créez les fichiers `f.sci` et `tracef.sce` correspondant aux programmes précédents.
- Exécutez ces deux programmes. Dans la fenêtre graphique ouverte, cliquez sur l'icône en haut à gauche. Faire alors pivoter le tracé afin d'obtenir le résultat présenté ci-dessus.
- Modifiez les intervalles de tracé en prenant `x=-4:0.1:4` et `y=-4:1:4`.
  Quel effet cela produit-il sur la figure ?

- Codez la fonction g dans un nouvel onglet **SciNotes**.
- Effectuez le tracé de la fonction g en choisissant les intervalles de tracé précédent. On sauvegardera le fichier de tracé sous le nom `traceg.sce`.
- D'après le graphique obtenu, la fonction g admet-elle des extrema ?

- Rappelez la définition de point critique d'une fonction et déterminez ceux de g.

On rappelle que si f est une fonction C^2, son développement limité au point (x, y) s'écrit :

$$f(x+h, y+k) = f(x,y) + {}^t\nabla(f)(x,y) \cdot \begin{pmatrix} h \\ k \end{pmatrix} + \frac{1}{2} \cdot (h\ k) \cdot \nabla^2(f)(x,y) \cdot \begin{pmatrix} h \\ k \end{pmatrix} + (h^2+k^2)\varepsilon(h,k)$$

où $\varepsilon(0,0) = 0$ et ε continue en $(0,0)$.

▶ Rappelez la définition de $\nabla^2(f)(x,y)$.

▶ Donnez une condition suffisante d'existence d'un extremum local.

II.2.b) La fonction `plot3d`

Avec la commande `fplot3d`, on fournit la valeur du vecteur `x`, du vecteur `y` et de la fonction à tracer. La valeur de la matrice `z` est alors automatiquement calculée. Ce n'est pas le cas avec la commande `plot3d` où l'on doit fournir la valeur de `z` de manière explicite.

▶ Écrire un programme (enregistré dans le fichier `traceg2.sce`) qui :

- × définit un vecteur `x`,
- × définit un vecteur `y`,
  (on pourra prendre les valeurs déjà utilisées)
- × stocke la **longueur** de `x` dans une variable `n`,
- × stocke la **longueur** de `y` dans une variable `p`,
- × définit une matrice `z` de taille `n` × `p` initialement remplie de 0,
- × pour tout `i` $\in [\![1, \texttt{n}]\!]$ et tout `j` $\in [\![1, \texttt{p}]\!]$ calcule la valeur de `z(i,j)` permettant le tracé de la fonction `g`.

On terminera le programme par l'appel : `plot3d(x,y,z)`.

II.2.c) Lignes de niveau

Si $a \in \mathbb{R}$, on appelle **ligne de niveau** a de f l'ensemble des points $(x, y) \in \mathbb{R}^2$ vérifiant $f(x, y) = a$. Ainsi, une ligne de niveau est l'intersection de S_f et du plan d'équation $z = a$.

On remplace la ligne `plot3d(x,y,z)` du programme précédent par le script suivant.

```
1 subplot(1,2,1)
2 plot3d(x,y,z)
3 contour(x,y,z,10,flag=[0,2,4])
4
5 subplot(1,2,2)
6 contour(x,y,z,10)
```

- Rappelez l'utilité de la commande `subplot`.

- À quoi sert la commande `contour` ? À quoi sert le paramètre optionnel `flag` ? On étudiera la rubrique d'aide pour répondre à ces questions.