S1.02

Graphiques avec *sagemath*

9 décembre 2025 – B. Colombel

Sommaire

1 Comment ça marche **1**
1.1 Graphe d'une fonction 1
1.2 Autres tracés 3

2 Attracteur d'Hénon **5**

1 Comment ça marche

1.1 Graphe d'une fonction

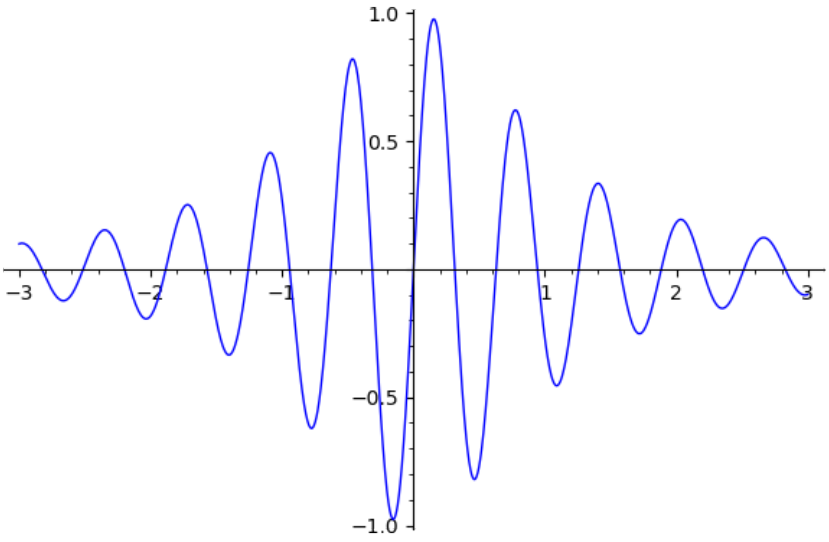
On veux représenter la fonction définie par :

$$f(x) = \frac{\sin 10x}{1+x^2}$$

sur l'intervalle $[-3;3]$.

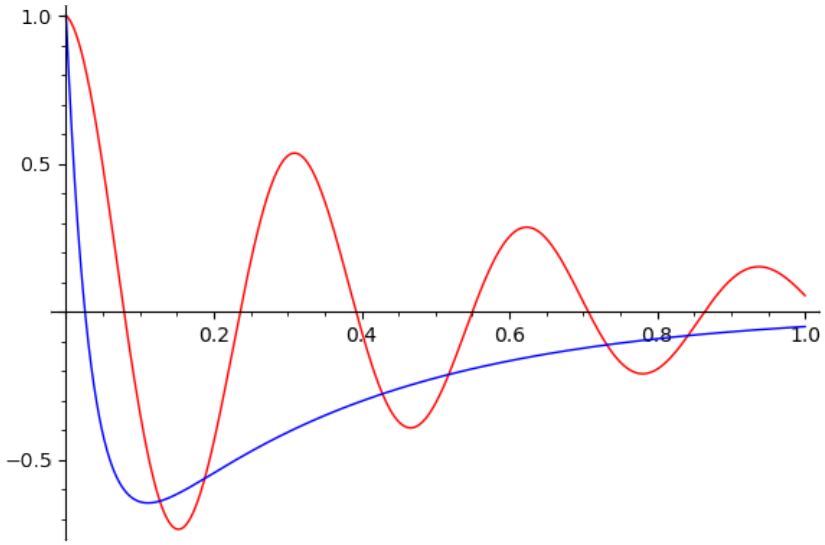
Pour tracer la courbe d'une fonction donnée, la commande est `plot()` . On donne le nom de la fonction et les valeurs minimales et maximales des abscisses à tracer mais tout d'abord, il faut indiquer à *sagemath* que la variable `x` est une variable *symbolique* ; elle ne contient aucune valeur.

[1]:
```
x = var('x')  # déclaration de la variable symbolique
f(x) = sin(10*x) / (1+x^2)
G1 = plot(f, (-3, 3))  # Construction du graphique
G1.show()  # affichage du graphique
```



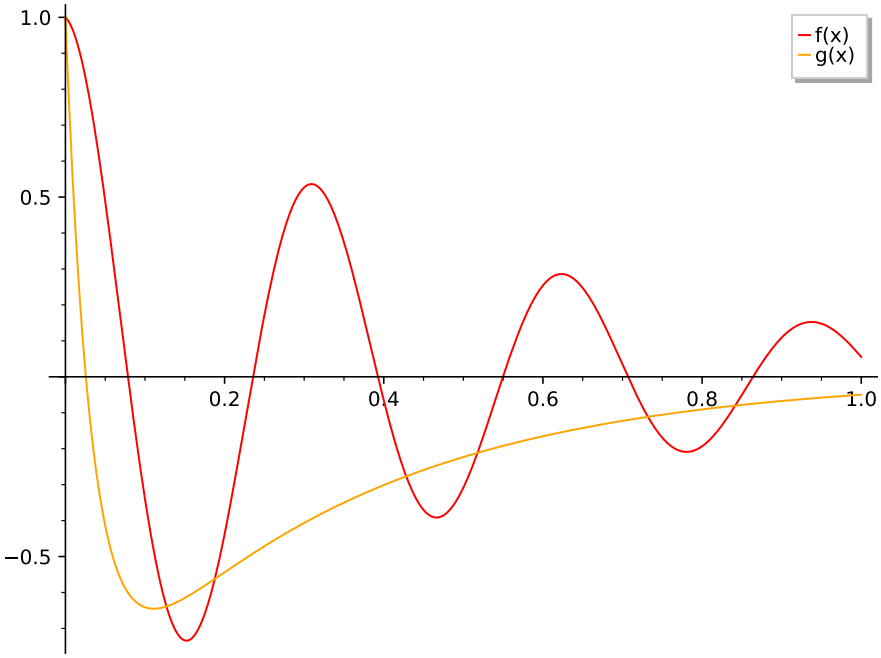
le logiciel *sagemath* offre un moyen très simple de fusionner deux graphiques l'un sur l'autre pour obtenir un graphique unique : l'**addition** !

[2]:
```
x = var('x')
g1 = plot(cos(20*x)*exp(-2*x), 0, 1, color='red')
g2 = plot(2*exp(-30*x) - exp(-3*x), 0, 1)
g = g1 + g2
g.show()
```



Hum . . . qui est qui ?
Pour n'importe quel graphique, une légende est **indispensable** !

[3]:
```
g1 = plot(cos(20*x)*exp(-2*x), 0, 1, legend_label='f(x)', color='red')
g2 = plot(2*exp(-30*x) - exp(-3*x), 0, 1, legend_label='g(x)', color='orange')
g = g1 + g2
g.show()
```



On peut aussi ajouter un titre avec l'argument `title = 'mon_titre'` dans un des appels de la fonction `plot()`.

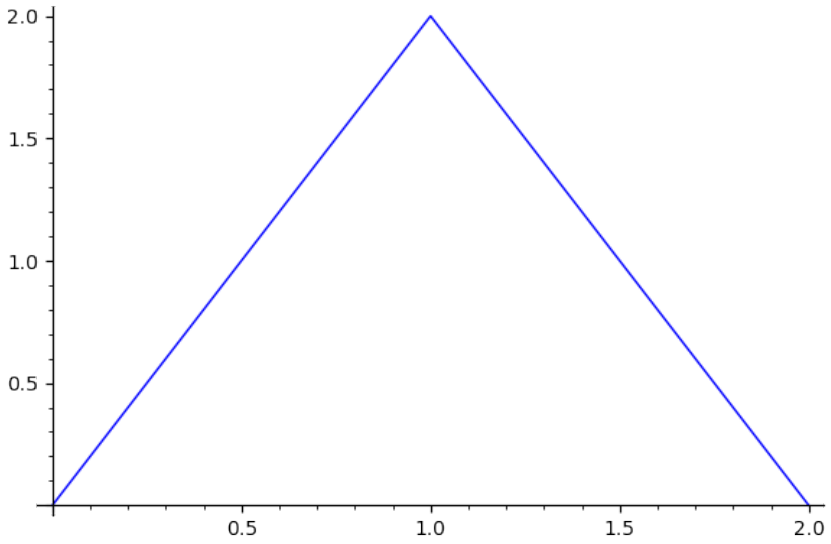
On peut sauver le graphique sous la forme d'une image avec la fonction `save()` :

[3]:
```
g.save('mongraphique.png')
```

1.2 Autres tracés

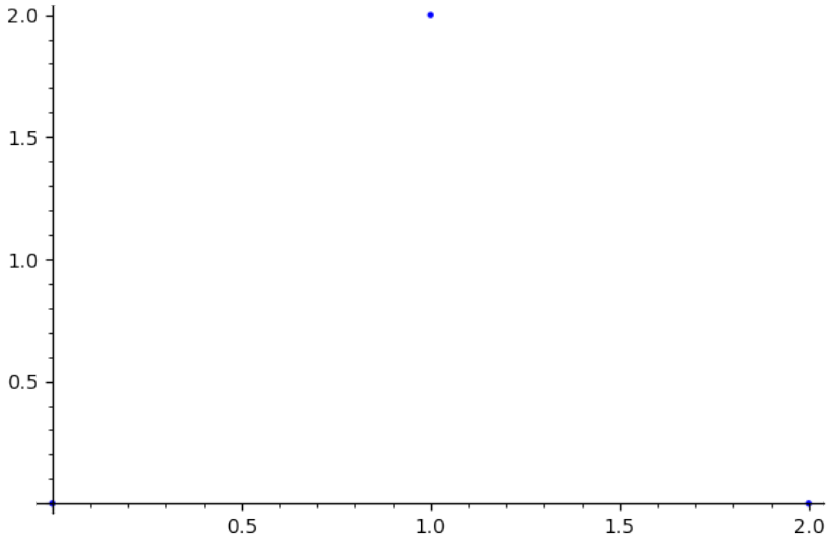
Une autre grande famille de tracés est celle où l'on ne trace plus les valeurs d'une fonction connue en tout points mais seulement en un certain nombre de points. Par exemple, si l'on souhaite tracer une courbe passant par les points de coordonnées (0,0), (1,2) et (2,0) on utilise la fonction `line()` .

[4]:
```
line([(0,0),(1,2),(2,0)])  # ou line([[0,0],[1,2],[2,0]])
```



Si l'on souhaite afficher uniquement les points sans les relier, on préférera la commande `point()` .

[5]:
```
point([(0,0),(1,2),(2,0)])  # ou point([[0,0],[1,2],[2,0]])
```



Cela peut-être utile pour représenter les termes d'une suite numérique par exemple.

Exemple 1. On souhaite représenter les premiers termes de la suite numérique :

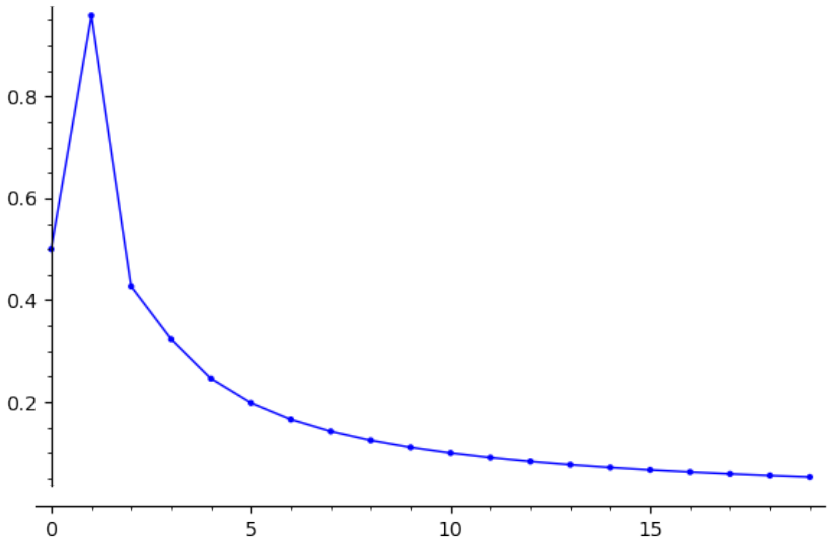
$$\begin{cases} u_0 & = 0{,}5 \\ u_{n+1} & = \frac{\sin(u_n)}{n \times u_n} \end{cases}$$

Le code ci-dessous permet de représenter les 20 premiers termes (de u_0 jusqu'à u_{19}).

[6] :
```
# Formule de récurrence
def suite(n) :
    u0 = 0.5
    U = [u0]
    for i in range(1,n) :
        u = (sin(U[i-1]) / (i*U[i-1])).n()
        U.append(u)
    return U

# Liste des coordonnées des points
n = 20
U = suite(n)
Coord = [(i, U[i]) for i in range(len(U))]

## affichage du graphique
G = line(Coord) + point(Coord)
G.show()
```



Exercice 1. (à vous de jouer !)
On considère la fonction

$$f(x) = \frac{3x}{1+2x}$$

1. Représenter la courbe de la fonction f sur l'intervalle $[0; \frac{3}{2}]$.
2. Sur le même graphique représenter la droite d'équation $y = x$.
3. Représenter, sur un nouveau graphique, les 10 premiers termes de la suite numérique définie par $u_0 = \frac{1}{2}$ et $u_{n+1} = f(u_n)$.

Exercice 2. Utilisation d'un fichier de données
Le fichier `enigme.txt` contient sur chacune de ses lignes deux nombres x et y correspondant aux coordonnées d'un point. Le code ci-dessous permet de le lire.

[5]:
```
def readFile(filename):
    v = []
    with open(filename) as f:
        for line in f:
            v.append([float(x) for x in line.split()])

    return v
```

1. Lire le fichier `enigme.txt`.
2. Que retourne la fonction `readFile` ?
3. À l'aide des fonctions `line()` et/ou `point()`, représenter les données du fichier `enigme.txt`.

   Énigme : De quoi s'agit-il ?

2 Attracteur d'Hénon

L'attracteur de Hénon est l'ensemble de points généré par l'application répétée de la fonction :

$$H(x;y) = (y + 1 - \alpha x^2, \beta x)$$

où α et β sont des nombres réels.

On le construit de la manière suivante :
En partant de $x_0 \in \mathbb{R}$ et $y_0 \in \mathbb{R}$, on trace l'ensemble des points de coordonnées $(x_n; y_n)$ où x_n et y_n sont définis par la relation de récurrence :

$$\begin{cases} x_{n+1} &= y_n + 1 - \alpha x_n^2 \\ y_{n+1} &= \beta x_n \end{cases}$$

Dans la suite, on choisira :

$$x_0 = y_0 = 0 \quad ; \quad \alpha = 1{,}4 \quad \text{et} \quad \beta = 0{,}3$$

Exercice 3. Écrire une fonction `H(x, y, alpha = 1.4, beta = 0.3)` qui retourne l'image de $(x; y)$ par la fonction H.

Exemples d'utilisation.

[10]:
```
H(0, 0)
```

[10]:
```
[1.00000000000000, 0.000000000000000]
```

[11]:
```
x, y = 0, 0
for i in range(5) :
    [x, y] = H(x, y)
    print([x, y])
```

[11]:
```
[1.00000000000000, 0.000000000000000]
[-0.400000000000000, 0.300000000000000]
[1.07600000000000, -0.120000000000000]
[-0.740886400000000, 0.322800000000000]
[0.554322279213056, -0.222265920000000]
```

Exercice 4. Écrire une fonction `henon(x0, y0, n, alpha = 1.4, beta = 0.3)` dont les paramètres sont le point initial (x_0,y_0), le nombre d'itérations n, ainsi que α, β et qui retourne la liste des points itérés :

$$(x_0; y_0); \quad (x_1; y_1); \quad (x_2; y_2) \quad \cdots \quad (x_n; y_n)$$

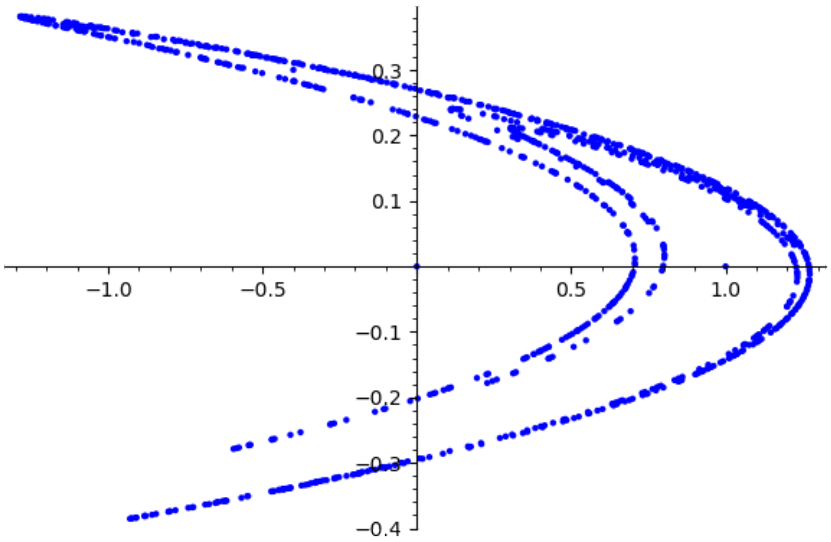
avec $(x_{k+1}; y_{k+1}) = H(x_k; y_k)$.

On prendra $x_0 = y_0 = 0$, $\alpha = 1{,}4$, $\beta = 0{,}3$ et $n = 1000$.

Exemple d'utilisation.

[13]:
```
x0 = 0
y0 = 0
alpha = 1.4
beta = 0.3
n = 1000

P = henon(x0, y0, n, alpha, beta)
g = point(P)
g.show()
```



Exercice 5.

1. Écrire une fonction `zoom_henon()` qui ajoute des paramètres $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$ et qui affiche seulement les itérés dont les abscisses sont comprises entre $x_{\min}$, $x_{\max}$ et les ordonnées sont comprises entre $y_{\min}$, $y_{\max}$.
2. Soit $x_0 = y_0 = 0$, $n = 10\,000$, $\alpha = 1{,}4$, $\beta = 0{,}3$ et le zoom $x_{\min} = 0{,}30$, $x_{\max} = 0{,}32$, $y_{\min} = 0{,}20$, $y_{\max} = 0{,}22$. Combien y a-t-il de points dans cette fenêtre ?
   Pour de belles images afficher ensuite plus de points.

Exercice 6. Écrire une fonction `sensibilite_henon()` qui a pour arguments deux points initiaux $(x_0; y_0)$ et $(x_0'; y_0')$ et qui retourne le tracé des points dont les abscisses sont les abscisses des itérés à partir du premier point alors que les ordonnées sont les abscisses des itérés à partir du second point.
On prendra $x_0 = y_0 = 0$, $x_0' = 0{,}000\,01$, $y_0' = 0{,}000\,001$, $\alpha = 1{,}4$, $\beta = 0{,}3$ avec $n = 10$ puis $n = 100$.
Qu'en pensez vous ?