TP2

January 21, 2024

0.1 L1 informatique – calcul scientifique

0.2 TP 2 – Sympy

0.3 Exercice 1 – étude de fonctions avec Sympy

[5]:
```
import sympy as sy
```

1. Au moyen de la librairie `sympy`, définir de manière symbolique la fonction $f(x) = \frac{(x+4+(3x+5))}{(x+4)^2}$

Afficher f . On essayera plusieurs affichages (print, display, sp.pprint)

Afficher $f(2)$ puis calculer une valeur approchée de $f(2)$.

[6]:
```
# A compléter ...
# ...
```

```
avec print
(4*x + 9)/(x + 4)**2
17/36

avec display
```

$$\frac{4x+9}{(x+4)^2}$$

$$\frac{17}{36}$$

```
avec sp.pprint
4 x + 9

       2
(x + 4)
17

36
valeur approchée 0.472222222222222
```

2a. Calculez la dérivée de f(x).

[3]:
```python
# A compléter ...
# ...
```

$$\frac{4}{(x+4)^2} - \frac{2 \cdot (4x+9)}{(x+4)^3}$$

2b. Calculer les zéros de la dérivée.

[4]:
```python
# A compléter ...
# ...
```

[4]: [-1/2]

2c. Exprimer la dérivée sous forme factorisée (fonction factor) pour vérifier ses zéros.

[5]:
```python
# A compléter ...
# ...
```

$$-\frac{2 \cdot (2x+1)}{(x+4)^3}$$

3. Calculer les limites de f(x) quand $x \to -\infty$, $x \to +\infty$, $x \to 2$,$x \to -4$

[7]:
```python
# A compléter ...
# ...
```

0

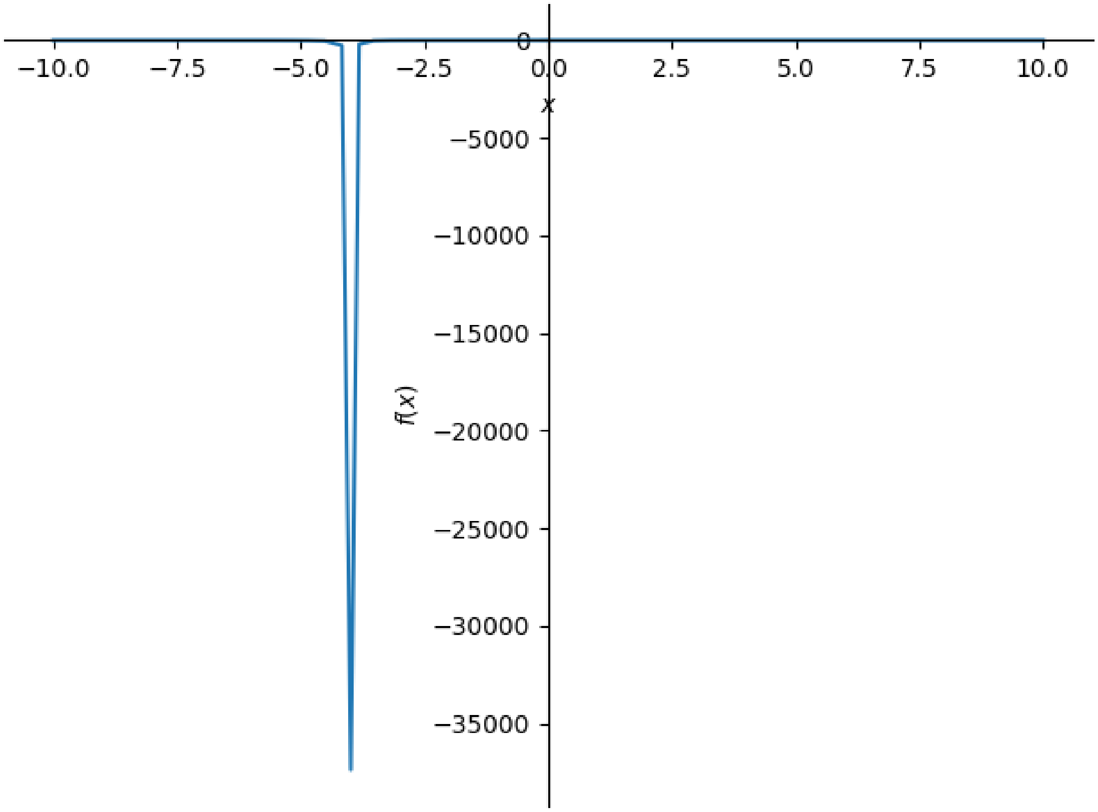
0

$\frac{17}{36}$

$-\infty$

4a. Tracer la courbe représentative de f

[7]:
```python
# A compléter ...
# ...
```

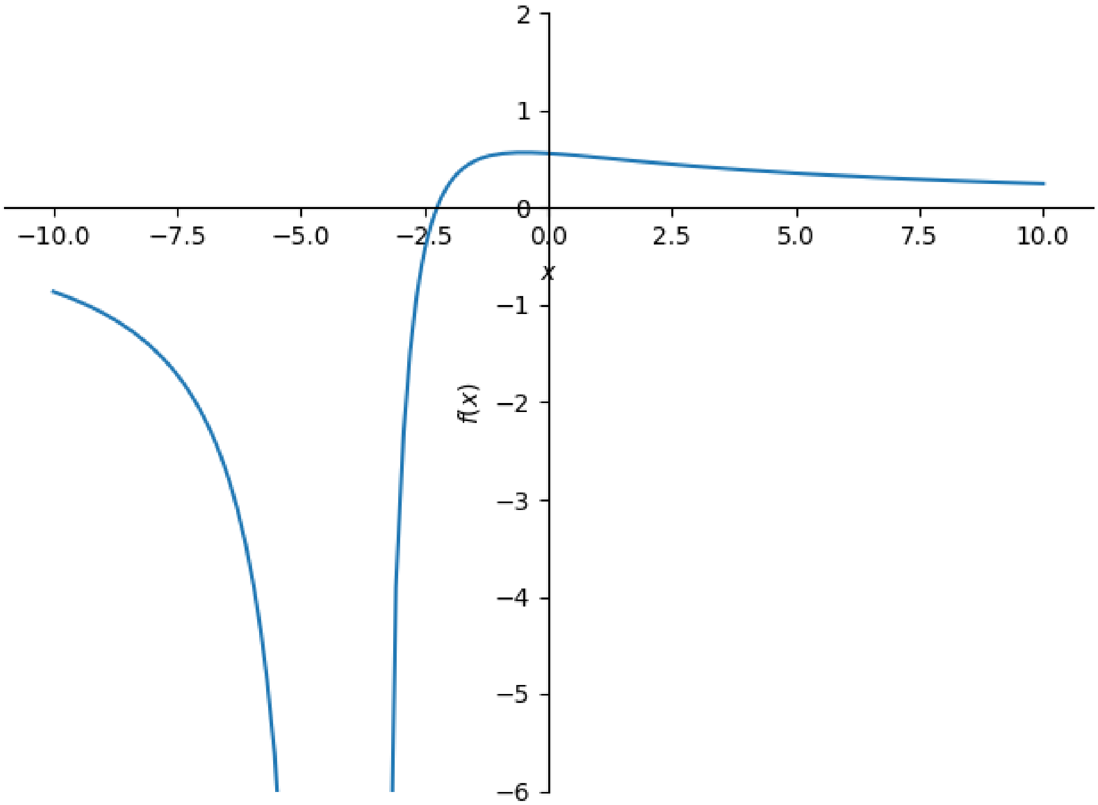


[7]: <sympy.plotting.plot.Plot at 0x7f6491f69fd0>

4b.Modifier la commande pour ne tracer la courbe que pour y compris entre -6 et +2 en utilisant l'option ylim

On écrira ylim=(a,b) à l'intérieur du plot.

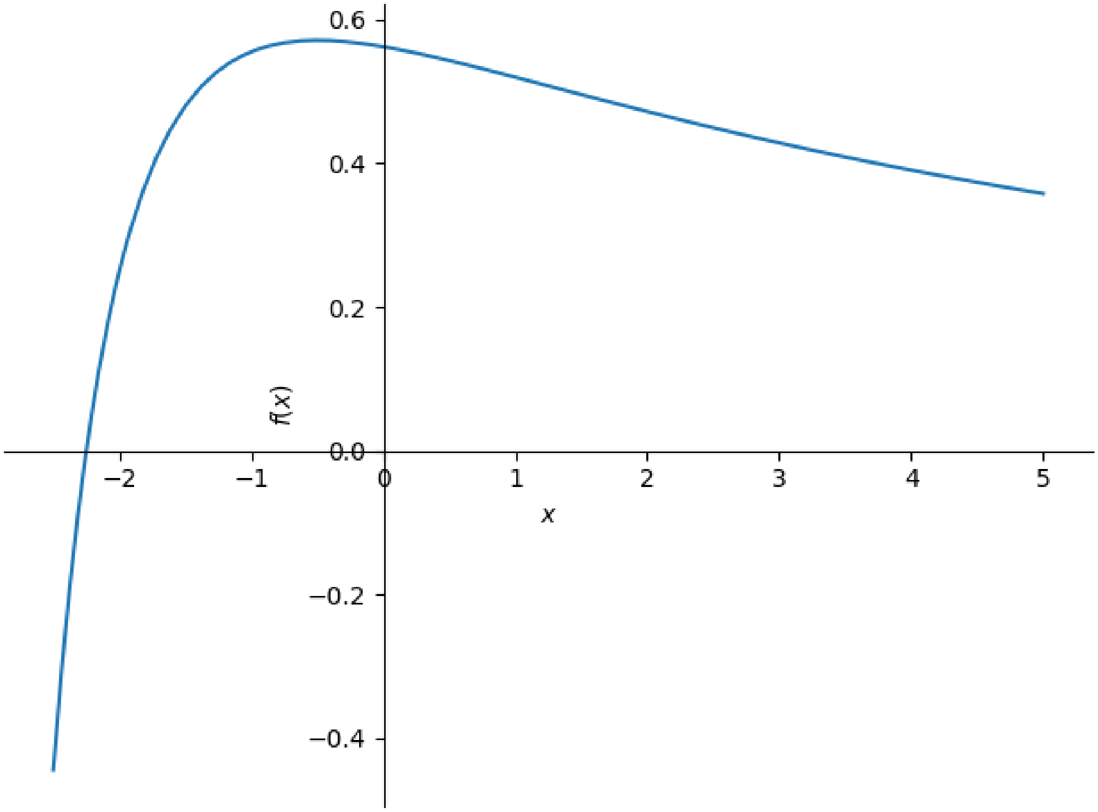
[8]:
```
# A compléter ...
# ...
```



[8]: `<sympy.plotting.plot.Plot at 0x7f64713a6730>`

4c.Limiter cette fois le dessin aux abscisses comprises entre -2.5 et 5

[9]:
```
# A compléter ...
# ...
```



[9]: <sympy.plotting.plot.Plot at 0x7f64713148b0>

0.4 Exercice 2

1) On considère les fonctions définies par:

$f_{a,b,c,d}(x) = -0.05*(ax^3 + bx^2 + cx + d)$ où a,b,c,d seront des paramètres qu'on pourra faire varier.

Définir $f_{a,b,c,d}$ à l'aide de sympy

[8]:
```
import warnings
warnings.filterwarnings('ignore') # pour ne pas afficher les warnings
from IPython.display import display, Math
from sympy import latex

sp.init_printing()

# Déclarer les symboles x, a, b, c et d
# ...

# Déclarer la fonction fab
# ...
```

```
# Afficher la fonction fab
# ...
```

$fab(x) = -0.05ax^3 - 0.05bx^2 - 0.05cx - 0.05d$

2) Calculer la limite limite de $f_{a,b,c,d}(x)$ quand x tend vers +infini avec `sympy`

[11]:
```
# A compléter ...
# ...
```

[11]:

$-\infty \operatorname{sign}(a)$

3) Calculer la dérivée de $f_{a,b,c,d}$.

[12]:
```
# A compléter ...
# ...
```

$-0.15ax^2 - 0.1bx - 0.05c$

4) On s'intéresse à $h = f_{6,5,-1,-1}$ et $g = f_{-2,5,3,0}$

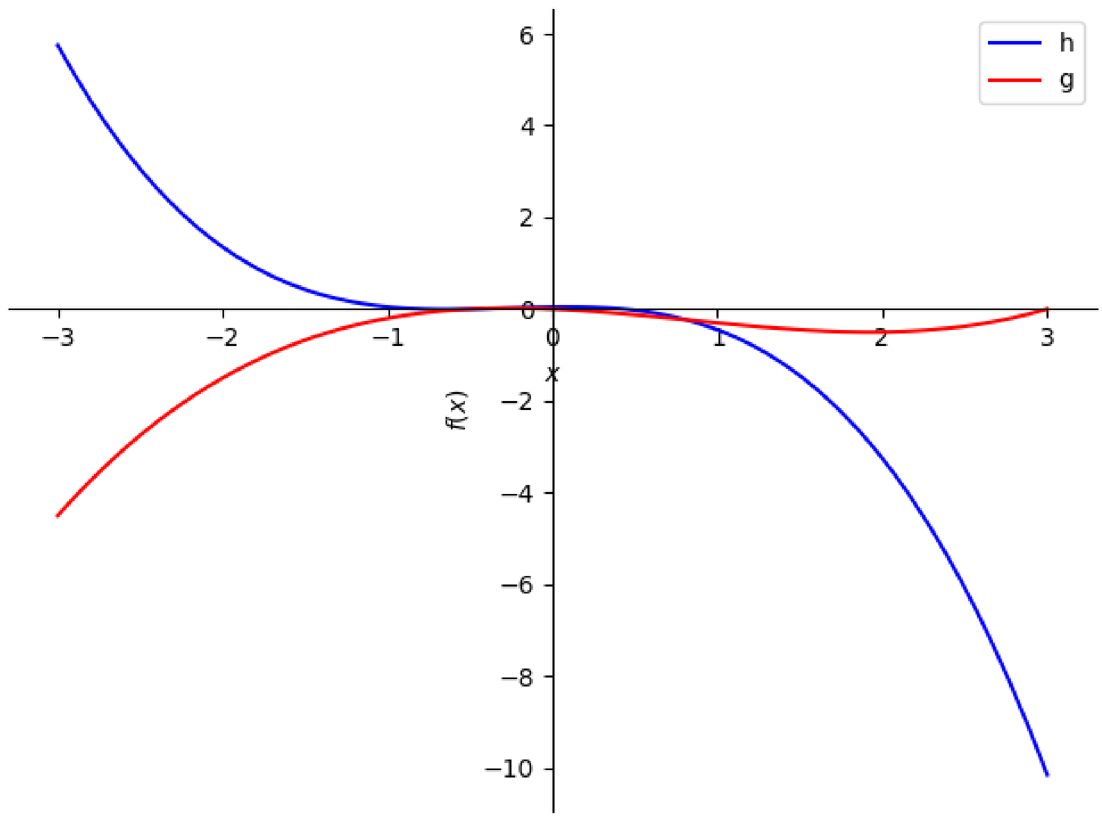
Ecrire les expressions correspondantes.

Tracer avec `sympy`, pour $x \in [-3, 3]$, le graphe des fonctions h et g , sur le même graphique et avec légende comme ci-dessous.

[11]:
```
# A compléter ...
# ...
```

$h(6, 5, -1, -1) = \frac{4x + 9}{(x + 4)^2}$

$g(-2, 5, 3, 0) = 0.1x^3 - 0.25x^2 - 0.15x$



5) à L'aide de la dérivée, déterminer les abscisses des points extremums de h

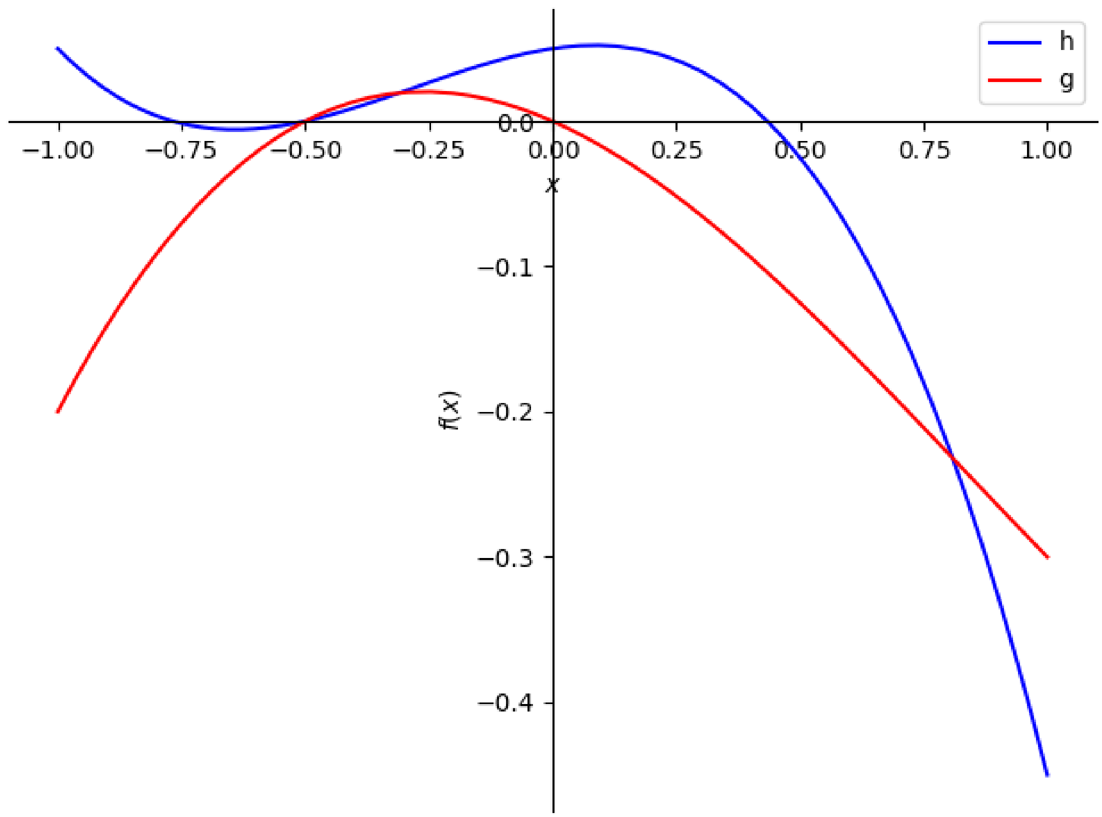
[17]:
```
# A compléter ...
# ...
```

$hprime = -0.9x^2 - 0.5x + 0.05$

```
extremum pour h [-0.642079918016778, 0.0865243624612223]
```

6) Vu le dessin, on voudrait mieux observer ce qui se passe entre -1 et 1. Refaire le shéma.

[15]:
```
# A compléter ...
# ...
```



7) Déterminer avec sympy les abscisses des points d'intersection de h et de g.

[16]:
```
# A compléter ...
# ...
```

```
Toutes les solutions [-0.500000000000000, -0.309016994374947, 0.809016994374947]
```

8) On appellera x1 et x2 les deux abscisses des points d'intersection de h et g les plus grandes (les deux dernières).

Déterminer l'aire comprise entre les deux courbes pour x variant entre x1 et x2.

[17]:
```
# A compléter ...
# ...
```

```
Aire entre les deux courbes: 0.0698771242968684
```

0.5 Exercice 3 : logique

Les valeurs Booléennes qui représentent le vrai et le faux sont True et False. Les opérations logiques utilisées sont :
- AND : a & b - OR : a | b - NOT : ~a - OR : a ^ b - => : a » b

1) Représentez la fonction : f = (x y) (y z) (z x) où x,y et z sont des symboles

[30]:
```
# A compléter ...
# ...
```

[30]:

$(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$

2) Que vaut f quand x et y sont vrai, et z faux?

Et quand x est vrai, y et z sont faux?

[19]:
```
# A compléter ...
# f quand  x est vrai, y et z sont faux?
# ...
```

[19]: True

[20]:
```
# A compléter ...
# f quand x et y sont vrai, et z faux?
# ...
```

[20]: False

3) On peut aussi mélanger valuation et variables: Que sera le résultat de f si on substitue FAUX à la variable y ?

[21]:
```
# A compléter ...
# ...
```

[21]:

$x \wedge z$

5) Simplifier l'expression logique suivante :

$$(A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee B)$$

[25]:
```
# A compléter ...
# ...
```

$A \wedge B$

6) Vérifier sur toutes les possibilités que le résultat est correct (on écrira une boucle!)

[28]:
```
# A compléter ...
# ...
```

```
 pour   (True, True)
Expression initiale : True
Expression symplifiée : True
 pour   (True, False)
Expression initiale : False
Expression symplifiée : False
 pour   (False, True)
Expression initiale : False
Expression symplifiée : False
```

```
 pour   (False, False)
Expression initiale : False
Expression symplifiée : False
```

Mise sous forme normale: toute expression peut être mise sous forme normale - d'une conjonction de disjonctions (FN conjonctive) - d'une disjonction de conjonctions (FN disjonctive)

Pour trouver la FN conjonctive (resp. disjonctive) de , on fait appel à la fonction to_cnf (resp. to_dnf) de sympy

7) Définir la fonction f : $ ¬(z →x) (x y)$

[34]:
```python
# A compléter ...
# ...

print("Avec print")
print(f)
print("Avec display")
display(f)
```

```
Avec print
(x | y) & ~(Implies(z, x))
Avec display
```

$z \not\Rightarrow x \wedge (x \vee y)$

[25]:
```python
# A compléter ...
# ...
```

```
Forme conjonctive : z & ~x & (x | y)
```

[26]:
```python
# A compléter ...
# ...
```

```
Forme disjonctive : (x & z & ~x) | (y & z & ~x)
```

8) Simplifier f

[27]:
```python
# A compléter ...
# ...
```

```
Forme simplifiée : y & z & ~x
```