Compte rendu

TP n° 1

Tracés et résolution graphique

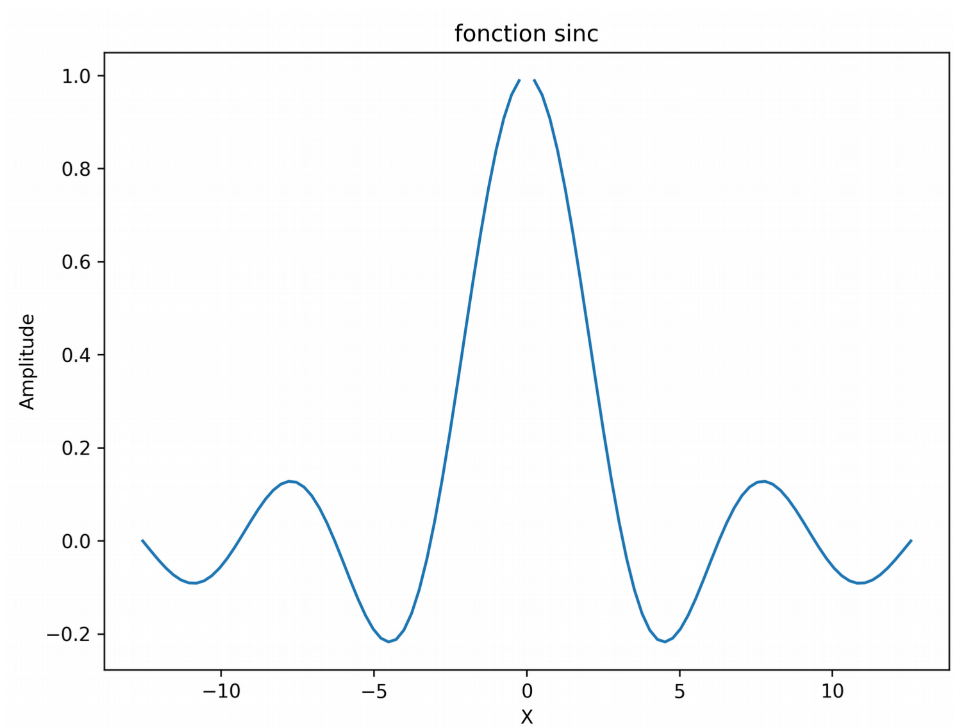
Baptiste Lucas L3 physique/Groupe B

1) Tracé de fonctions « à problème ».

1,1) Tracé de [x → sin(x)/x].

Tout d’abord on import ‘NumPy’ pour pouvoir utiliser ses nombreuses fonctionnalités. Puis un import de ‘matplotlib’ est nécessaire pour réaliser par la suite les graphiques

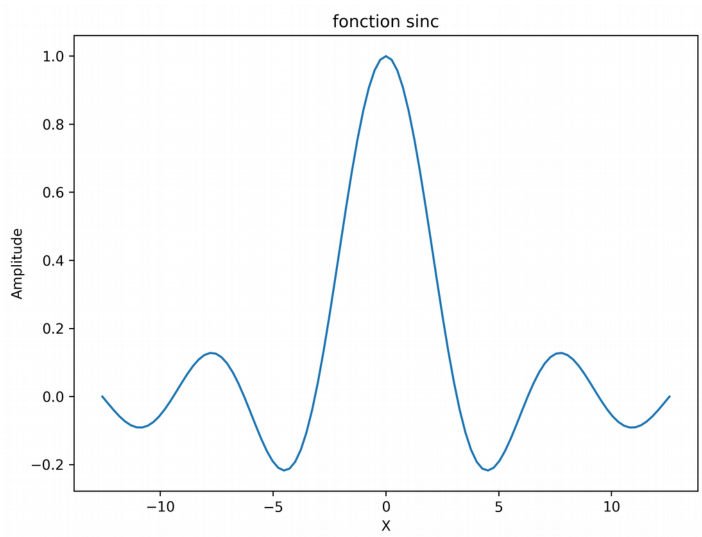
Il est donné une valeur de ‘x’ que l’on rentre dans le code, ensuite viens le tracé de la courbe: $sinc(x)=\frac{\sin(x)}{x}$,



On remarque que en zéro la fonction n’est pas définie, spyder nous le fait savoir aussi :

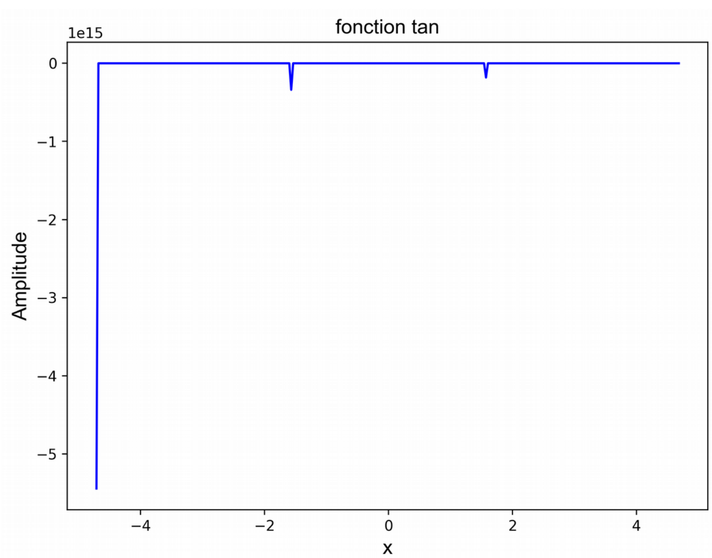
```
In [21]: runfile('C:/Users/lucas/Desktop/physique numérique/tp1 (tracés - Résol graphique)/exo1/Baptiste Lucas.py', wdir='C:/Users/lucas/Desktop/physique numérique/tp1 (tracés - Résol graphique)/exo1')
C:\Users\lucas\Desktop\physique numérique\tp1 (tracés - Résol graphique)\exo1\Baptiste Lucas.py:18: RuntimeWarning: invalid value encountered in true_divide
  y = (np.sin(x))/x
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
```

Pour combler ce trou en zéro, on addition à la variable ‘x’ une valeur en ‘x=0’(ligne 17). Ce qui nous donne le graphe suivant.



1,2) Tracé de [x → tan(x)].

On veut donc tracer la fonction tangente de x. Cette fois avec x dans l’intervalle [-3/2;3/2]π , avec un pas de 1/100*π



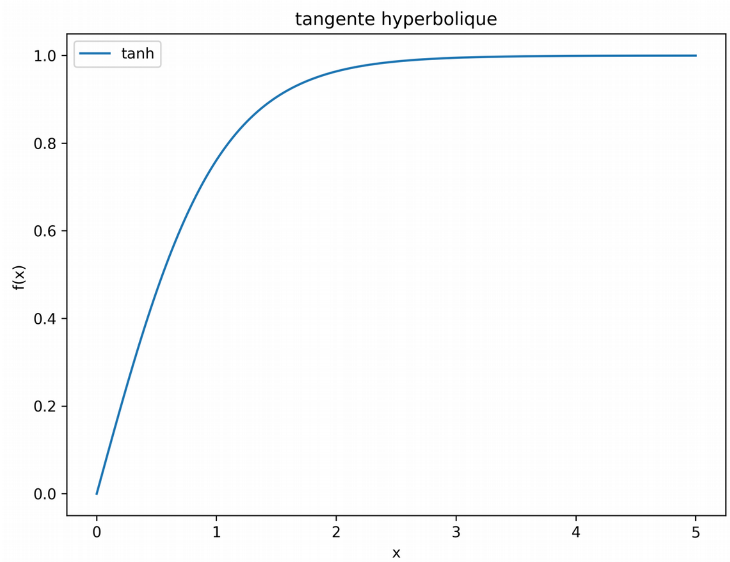
Le problème de cette représentation est clair. Les valeurs de tan(x) prisent pour le graphique sont bien trop important à certain endroit. Ce qui fausse le résultat obtenu, pour y remédier il faut éliminer des valeurs de tan(x) .

Pour cela on utilise une fonction de Numpy qui est ‘valeurs=np,NaN’(ligne33), celle-ci indique que les valeurs égale à ‘np.Nan’ seront éliminé. Dans notre cas les valeurs non désirées sont : `tan( x )[ np.abs(tan(x)>=10 ]`

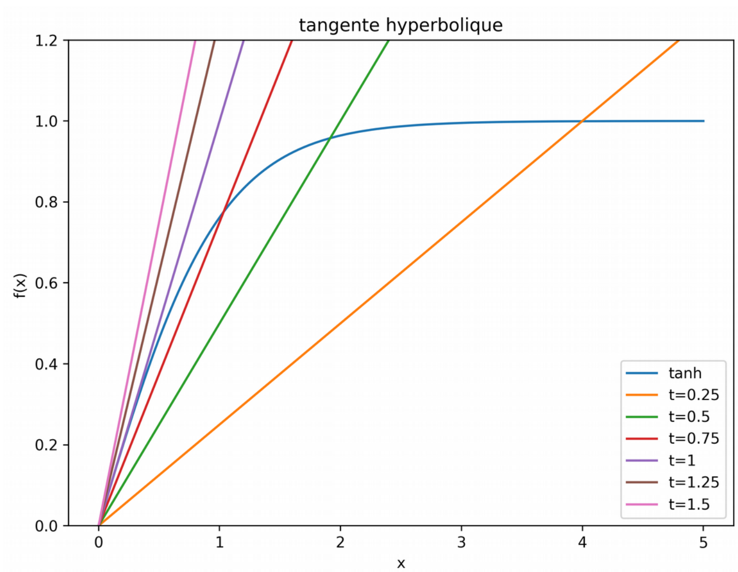
2) Résolution d’une équation non linéaire.

2,1) Résolution graphique

On trace initialement la fonction tangente hyperbolique grâce à *NumPy*, *np,tanh(x)*. Nous avons donc la courbe suivant.



a)Ensuite il faut tracé les droites $m = t \times x$,pour cela on introduit une boucle(for t in []) dans le code permettant de faire varier *t* entre *0,25* et *1,5* par pas de *0,25*. On défini aussi *x* grâce à *x=np,linspace(0,5,1000)* soit *x* qui varie entre 0 et 5 avec un pas de 1/200. Dans cette boucle on introduit une variable *y* qui prend la valeur *t*x*. Lors de chaque tour de la boucle la variable *y* aura donc une valeur différente. Avec ces deux variable et grâce aux outils de représentation graphique on peut obtenir le graphe de la ‘tanh(x)’ et de ‘y’ en fonction de *x* (ligne 27). il faut intégrer le *plot* à la boucle pour bien avoir chaque droite y_i.



.

b)Toutes les valeurs de *t* supérieur à *1* coupent la tangente hyperbolique en un seul point. Dans les valeurs expérimentales utilisé seul, *t=1,25* et *t=1,5* respecte cette condition.

c)Toutes les valeurs dans l'intervalle *[1,0[* coupent la tangent en deux points. Dans notre cas on en compte trois, *t=0,75, t=0,5 et t=0,25.*

d)De nos valeurs utilisé de *t*, celle pour qui la valeur de *m* est la plus proche de *1* est *t=0,25*. Plus la valeur de *t* tend vers 0^+, la limite de *m* tendra vers 1.

2,2) Résolution numérique

Pour la suite il est nécessaire d'importer de *scipy fsolve* ce qui permettra de résoudre des systèmes du type f(x)=0.

On veut résoudre l'équation suivante :

$$m = \tanh\left(\frac{m}{t}\right) \quad \text{pour } m>0$$

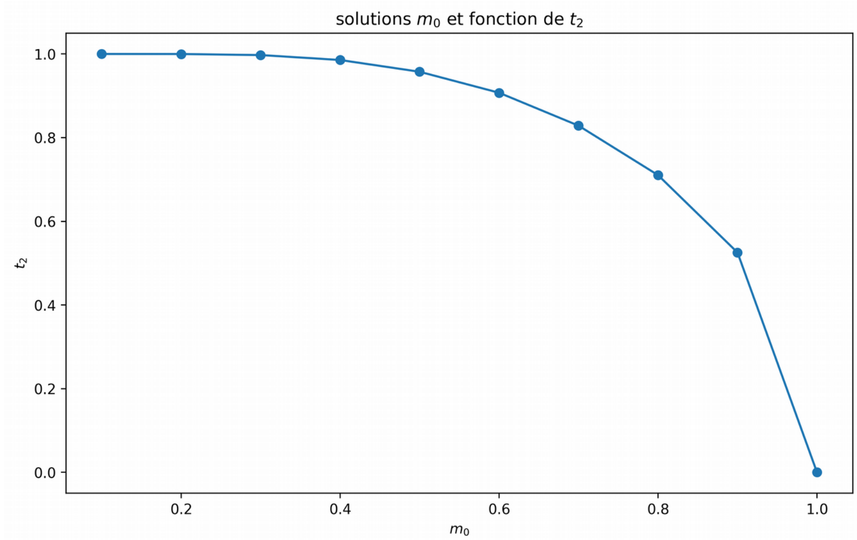
a)Tout d'abord, il faut définir notre fonction grâce à :

def nom_fonction(arguments) :

Ici on nomme les fonction *g* et on lui donne deux arguments *m* et *t*. Dans cette fonction on donne à une variable *y* la valeur *tanh(m/t)-m*. Comprenant les deux arguments de la fonction, puis on retourne cette variable.

b)Pour obtenir les solutions m_1 et m_2 pour des valeur respective de t qui sont *t=0,5* et *t=0,9*. On utilise donc *fsolve* avec des condition initiales égale à 1 et une valeur de t que l'on établie avant la fonction et que l'on rappelle dans *fsolve* grâce à *args=(t)*.

c)Pour cette question on cherche à déterminer les solutions m_0 et les tracer en fonction de t_2. Pour cela on définie une fonction g_0 similaire aux fonctions *g* précédentes. Puis on construit une boucle sur t_2 *entre 0,1 et 1,1 avec un pas de 0,1.* Ensuite on place *fsolve* dans cette boucle, et chaque valeur s de m_0 est introduite dans une liste pour pouvoir par la suite faire la représentation graphique suivante (*m* en fonction de t_2*)*



*d)*On réalise un subplot entre le graphe de la question antérieur et l'association des courbes t*x et tanh(x), pour cela j'ai copié le code des deux première courbes et collé sous le code de la question 2.2.c. Il est vrai que pour des questions d'ergonomie et de simplicité, il suffisait d'écrire plt.subplot() au début du programme .

e) Enfin il faut mettre des titres aux axes et aux graphiques, pour cela on utilise la bibliothèque matplotlib que l'on import ici en tant que plt . On utilise donc plt.title('') (pour mettre un titre au graphique), plt,xlabel('') et plt,ylabel('') (pour donner des noms aux axes).

Remarque : Il est aussi possible de d'une autre manière avec matplotlib pour crée des graphique. On peut écrire fig, (ax1, ax2) = plt.subplot() , ensuite il faut définir les titre et noms d'axes de façons différente en effet on doit écrire : (par exemple pour le graphique 1)

ax1.set_title('') , pour le titre
ax1.set_xlabel(''), pour l'axe des abscisses

3) Dénombrement d'états.

Je n'ai pas compris comment résoudre cette partie.