# Introduction à la géométrie numérique

**Mathématiques Appliquées**
**pour le Génie des Procédés et l'Énergétique** *

**Cours numéro 6, septembre 2019**

* Cours du CNAM "UTC101" ; leçon proposée par François Dubois

# Graphe d'une parabole avec le logiciel Python



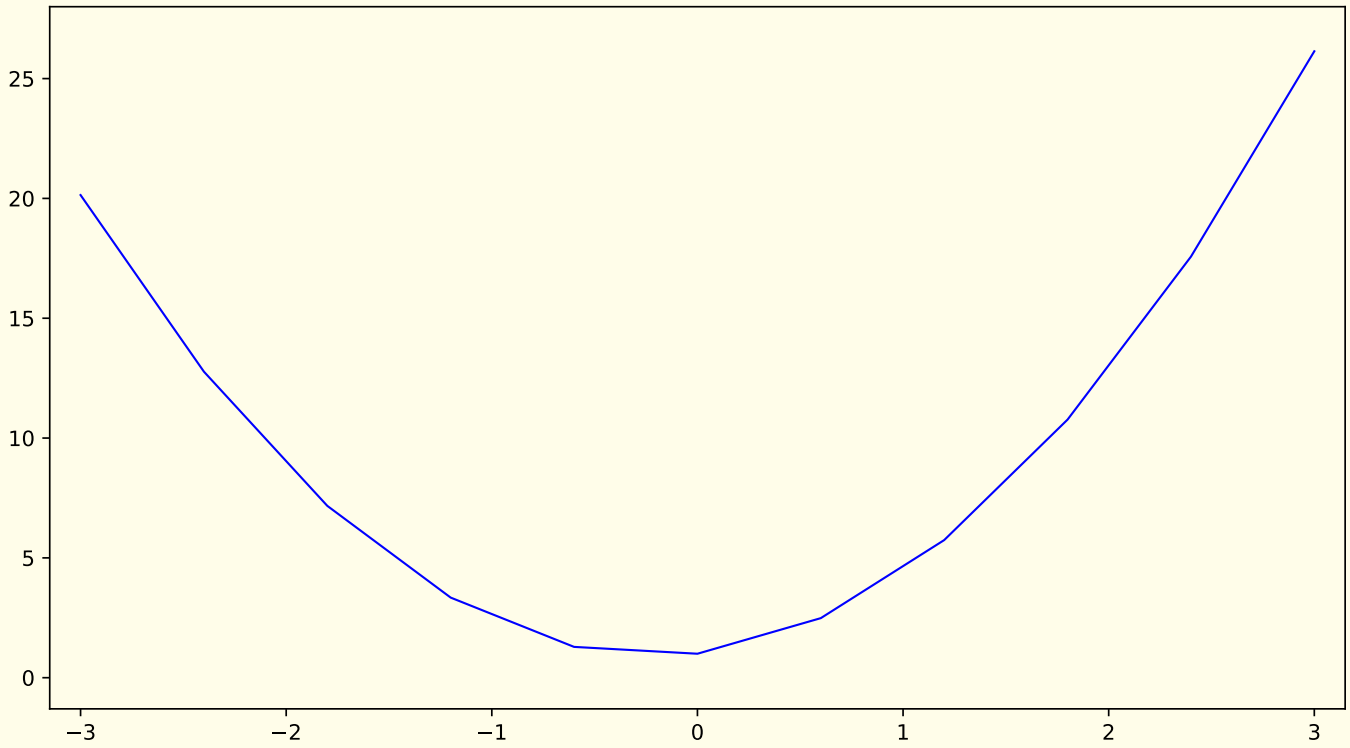
Un codage très élémentaire dans le langage Python peut s'écrire

```
import numpy as np                    ###    déclarations préliminaires
import numpy as np
import matplotlib.pyplot as plt
from pylab import plot, axis, savefig, show
from math import log

a = 2.46 ; b = 1. ; c = 1.                  ###    codage des données
nn = 10  ; xmin = -3. ; xmax = 3.
dx = (xmax-xmin)/nn
xx = np.zeros(nn+1) ; pp = np.zeros(nn+1)

for i in range(0,nn+1) :                    ###    début de boucle
    xx[i] = xmin + i*dx
                                      ### parabole  y = a x^2 + b x + c
    pp[i] = a*(xx[i]**2) + b*xx[i] + c
print ('xx =', xx) ; print ('pp =', pp)
```

# Graphe d'une parabole avec le logiciel Python *(ii)*



```
fig, ax = plt.subplots(figsize=(11,6))   ###   codage du graphique
ax.plot (xx, pp, 'b', linewidth=1)
plt.savefig('utc-parabole-01-fig.pdf', transparent=True)
plt.show()                               ###   on génère ainsi le graphe suivant
```




La discrétisation est clairement visible
à cause du faible nombre de points utilisés.

# Ajout d'un point sur la courbe



### peu de nouvelles données

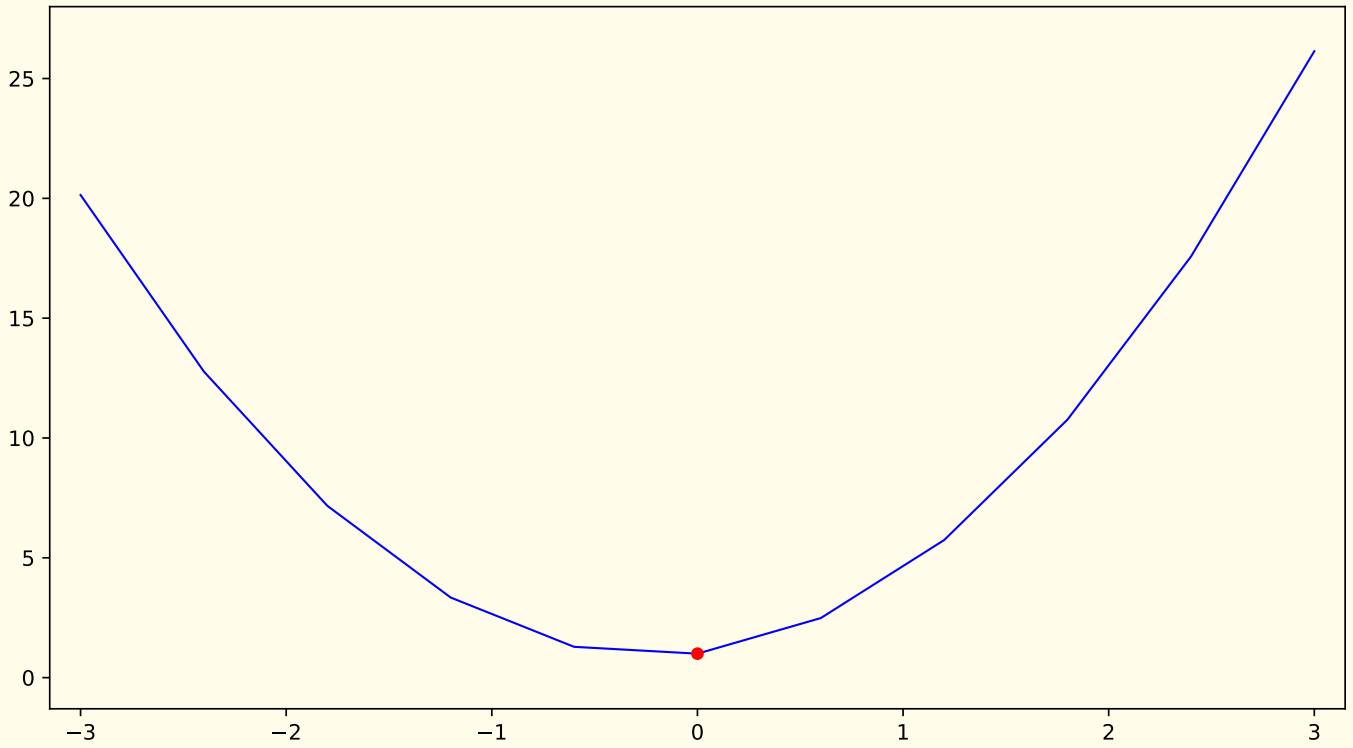
x0 = 0 ; y0 = c

### codage du graphique

```
fig, ax = plt.subplots(figsize=(11,6))
ax.plot (xx, pp, 'b', linewidth=1)
ax.plot (x0,y0 , 'r',marker='o',markersize=5)
plt.savefig('utc-parabole-02-fig.pdf', transparent=True)
plt.show()
```

# Ajout d'un point sur la courbe *(ii)*






la courbe est enrichie d'un point isolé.

# Tangente



Tracer la droite tangente
qui passe par le point que l'on s'est donné

On rappelle d'abord que
la droite tangente
à la courbe d'équation $y = f(x)$
au point de coordonnées $\big(x_0,\, y_0 = f(x_0)\big)$
est l'unique droite qui passe par le point $(x_0,\, y_0)$
et de pente $f'(x_0)$.

Elle a donc pour équation $y = y_0 + f'(x_0)\,(x - x_0)$.

Avec le langage Python,
on se contente de tracer un segment de droite.

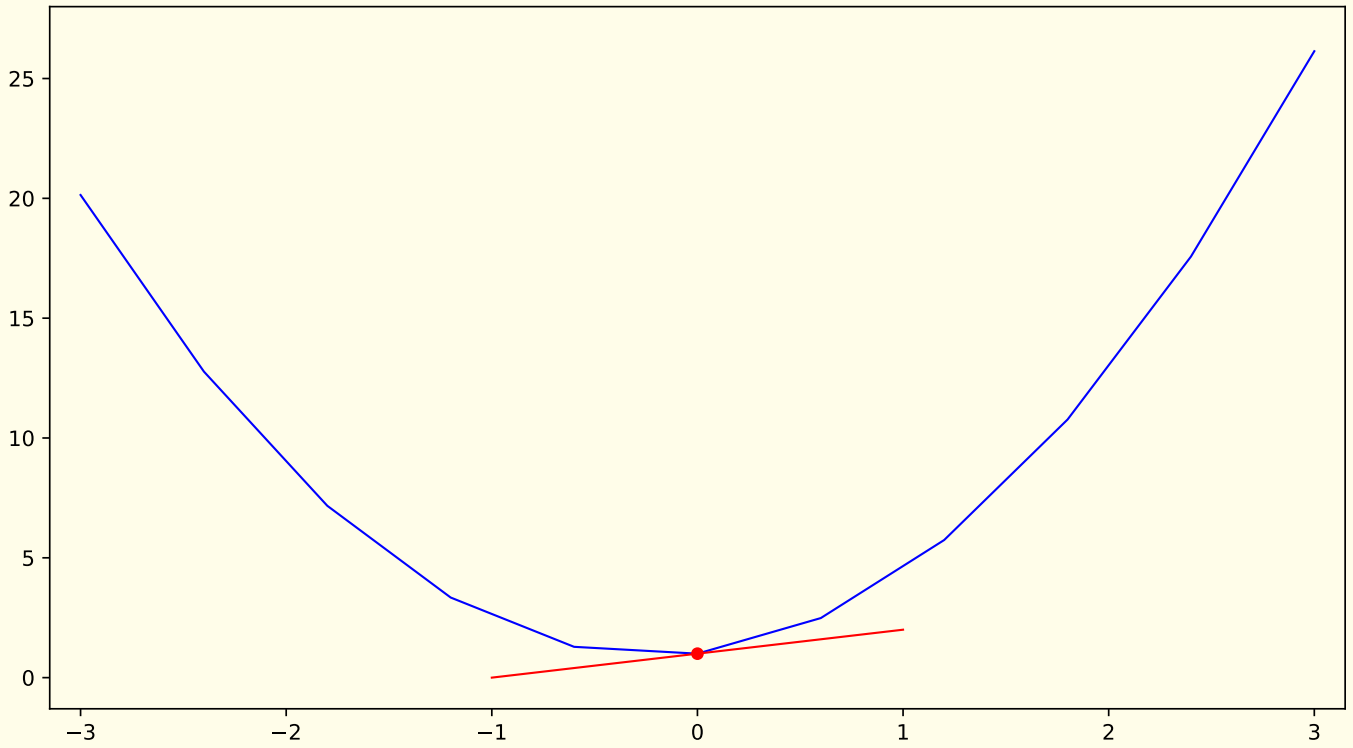
# Tangente *(ii)*



```
                                        ###      nouvelles données
x0 = 0 ; y0 = a*x0**2+b*x0+c ; yp0 = 2*a*x0 + b
                                    ###      points extrêmes de la tangente
xtmin = x0-1 ; xtmax = x0+1
                                        ###      valeurs des ordonnees
ytmin = y0 + yp0 * (xtmin-x0)
ytmax = y0 + yp0 * (xtmax-x0)
                                         ###      codage du graphique
fig, ax = plt.subplots(figsize=(11,6))
ax.plot (xx, pp, 'b', linewidth=1)
ax.plot (x0,y0 , 'r',marker='o',markersize=5)
    ###      un segment est décrit par ses deux points extrémaux
ax.plot ([xtmin,xtmax],[ytmin,ytmax] , 'r',linewidth=1)
plt.savefig('utc-parabole-03-fig.pdf', transparent=True)
plt.show()
```

# Tangente *(iii)*






le point isolé est muni de sa tangente

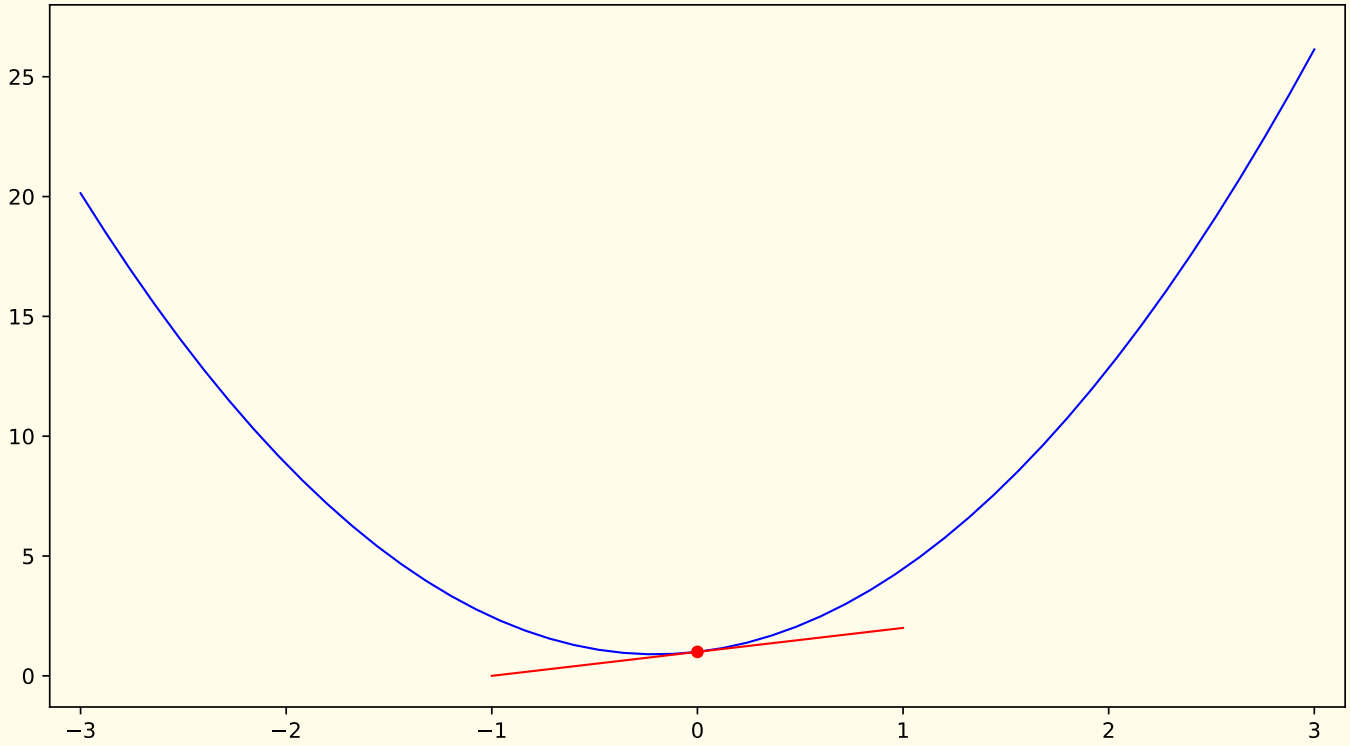
# Ajout de points



♯♯♯ afin d'avoir une parabole plus réaliste

On remplace l'instruction $nn = 10$ par l'instruction $nn = 50$
au début du programme lors du codage des données.

Le graphique est alors le suivant :




ce qui est beaucoup plus réaliste

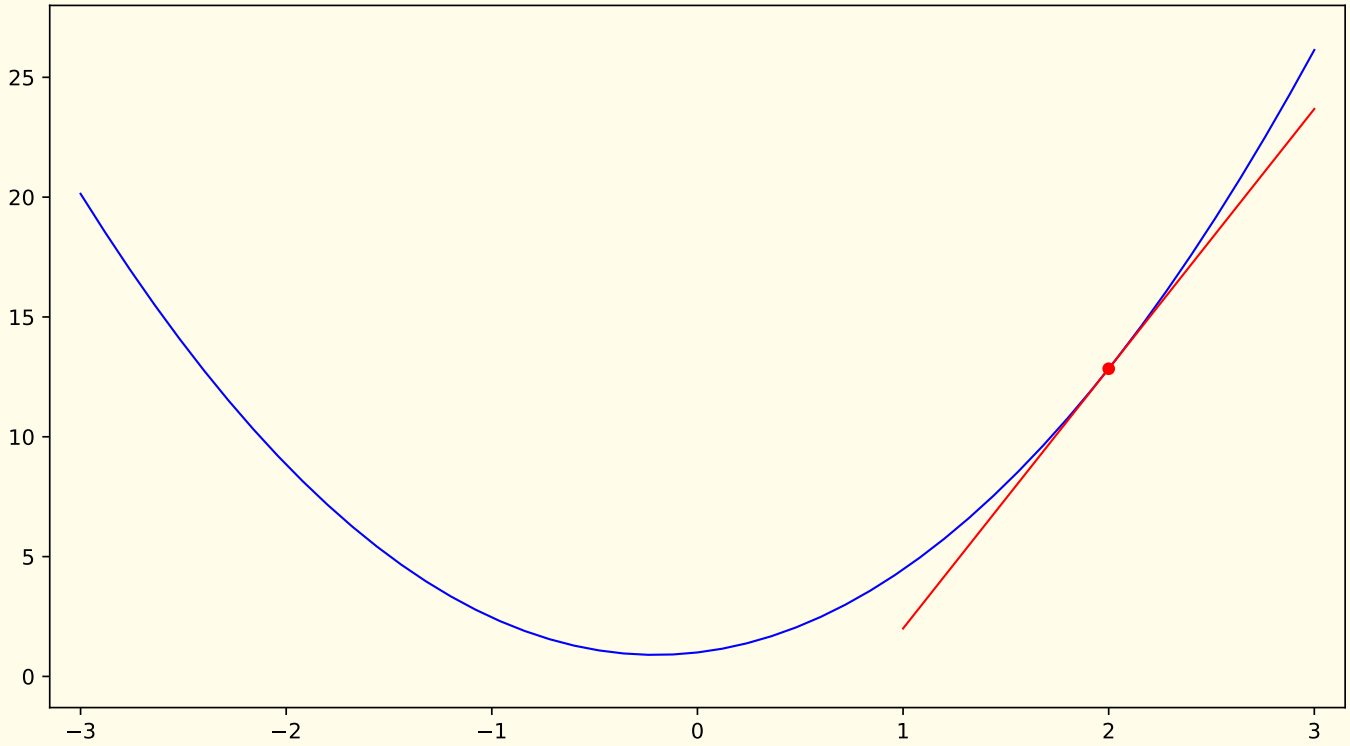
# Déplacer le point isolé



on ajoute simplement les lignes suivantes au programme précédent

### quelques données

```
x0 = 2 ; y0 = a*x0**2+b*x0+c ; yp0 = 2*a*x0 +b
xtmin = x0-1 ; xtmax = x0+1
ytmin = y0 + yp0*(xtmin-x0) ; ytmax = y0 + yp0*(xtmax-x0)
```




et le codage du graphe est inchangé.

# On dessine deux courbes



Dans cet exemple,

une courbe logarithmique représentée par la fonction

$$f_1(x) = a \log(x) + b$$

une hyperbole associée à la fonction

$$f_2(x) = \frac{1}{x}.$$

On rajoute aussi un point de même abscisse

sur chacune des courbes

et les tangentes associées.

# On dessine deux courbes *(ii)*



```
                                        ###      données des deux courbes
a = 2 ; b = .5
nn = 50 ; xmin = .5 ; xmax = 2 ; dx = (xmax-xmin)/nn
xx = np.zeros(nn+1) ; f1 = np.zeros(nn+1) ; f2 = np.zeros(nn+1)
for i in range(0,nn+1) :
    xx[i] = xmin + i*dx
    f1[i] = a*log(xx[i]) + b -(1./xx[i])
    f2[i] = (1./xx[i])
                                        ###      points de même abscisse
x0 = .6 ; y1 = a*log(x0) + b ; y2 = 1/x0
yp1 = a/x0 ; yp2 = -1/(x0*x0)
xtmin = x0-.2 ; xtmax = x0+.5
y1tmin = y1 + yp1 * (xtmin-x0)
y1tmax = y1 + yp1 * (xtmax-x0)
y2tmin = y2 + yp2 * (xtmin-x0)
y2tmax = y2 + yp2 * (xtmax-x0)
```

# On dessine deux courbes *(iii)*

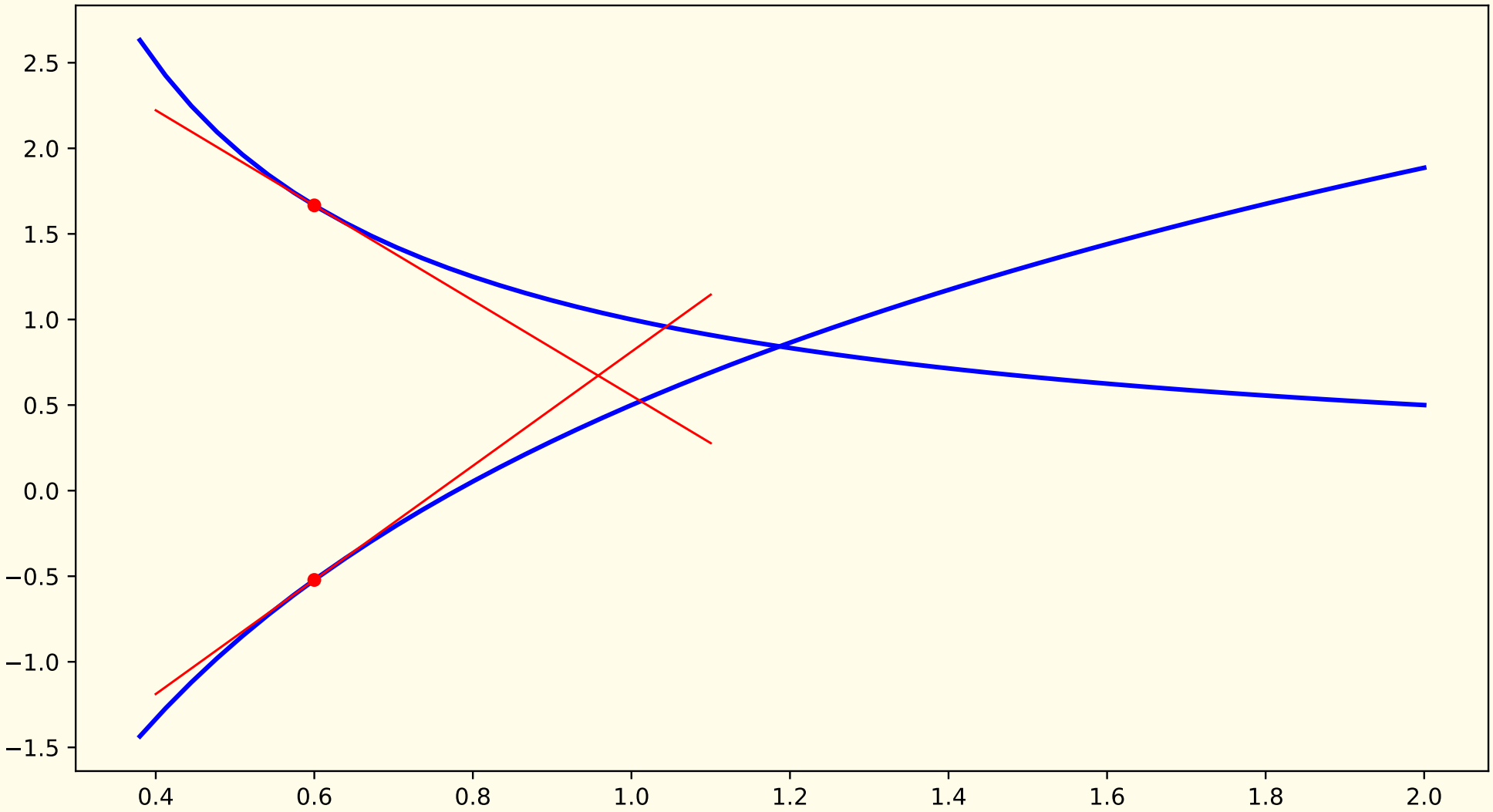


### graphique des deux courbes

```
fig, ax = plt.subplots(figsize=(11,6))
ax.plot (xx, hy, 'b',linewidth=2)
ax.plot (xx, f1, 'b',linewidth=2)
ax.plot (x0,y0 , 'r',marker='o',markersize=5)
ax.plot (x0,f2, 'r',marker='o',markersize=5)
ax.plot ([xtmin,xtmax],[y1tmin,y1tmax] , 'r',linewidth=1)
ax.plot ([xtmin,xtmax],[y2tmin,y2tmax] , 'r',linewidth=1)
plt.savefig('utc-intersection-courbes-fig.pdf', transparent=True)
plt.show()
```

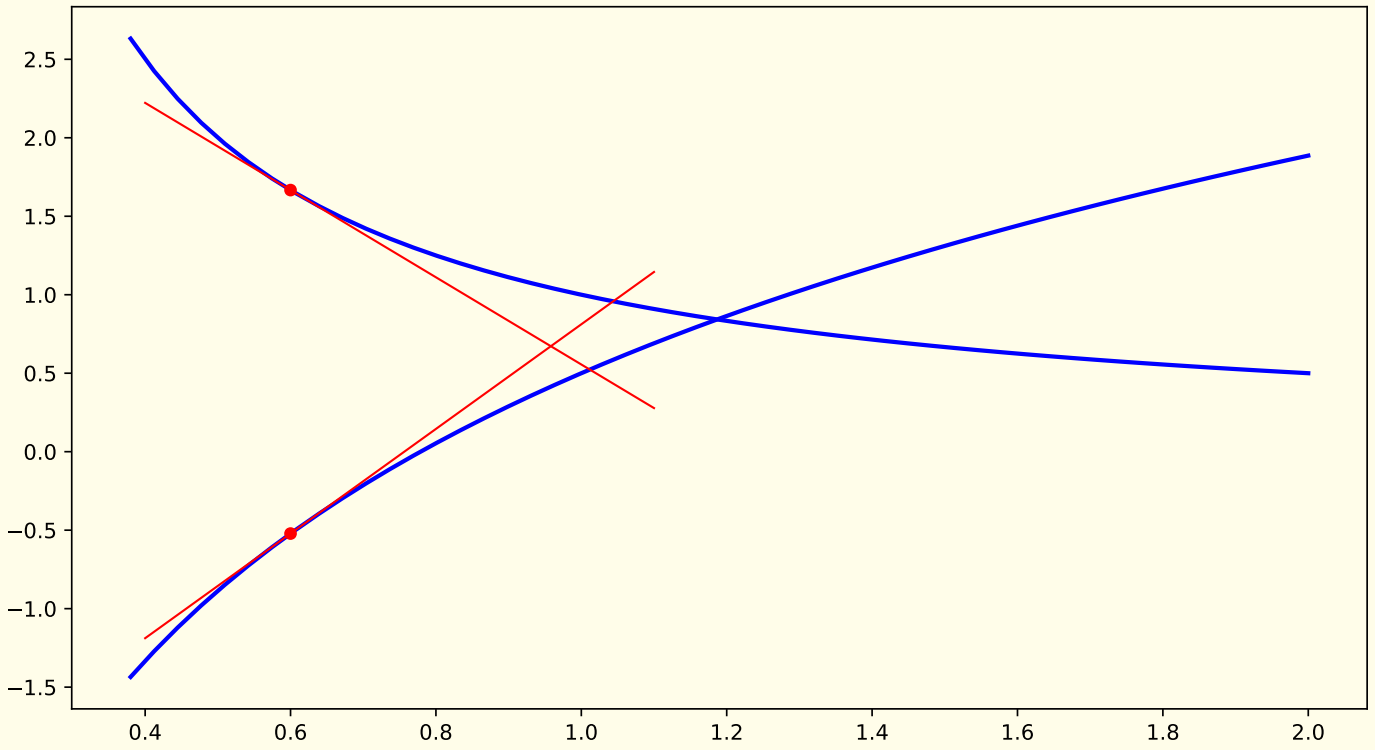
# On dessine deux courbes *(iv)*





# Algorithme de Newton

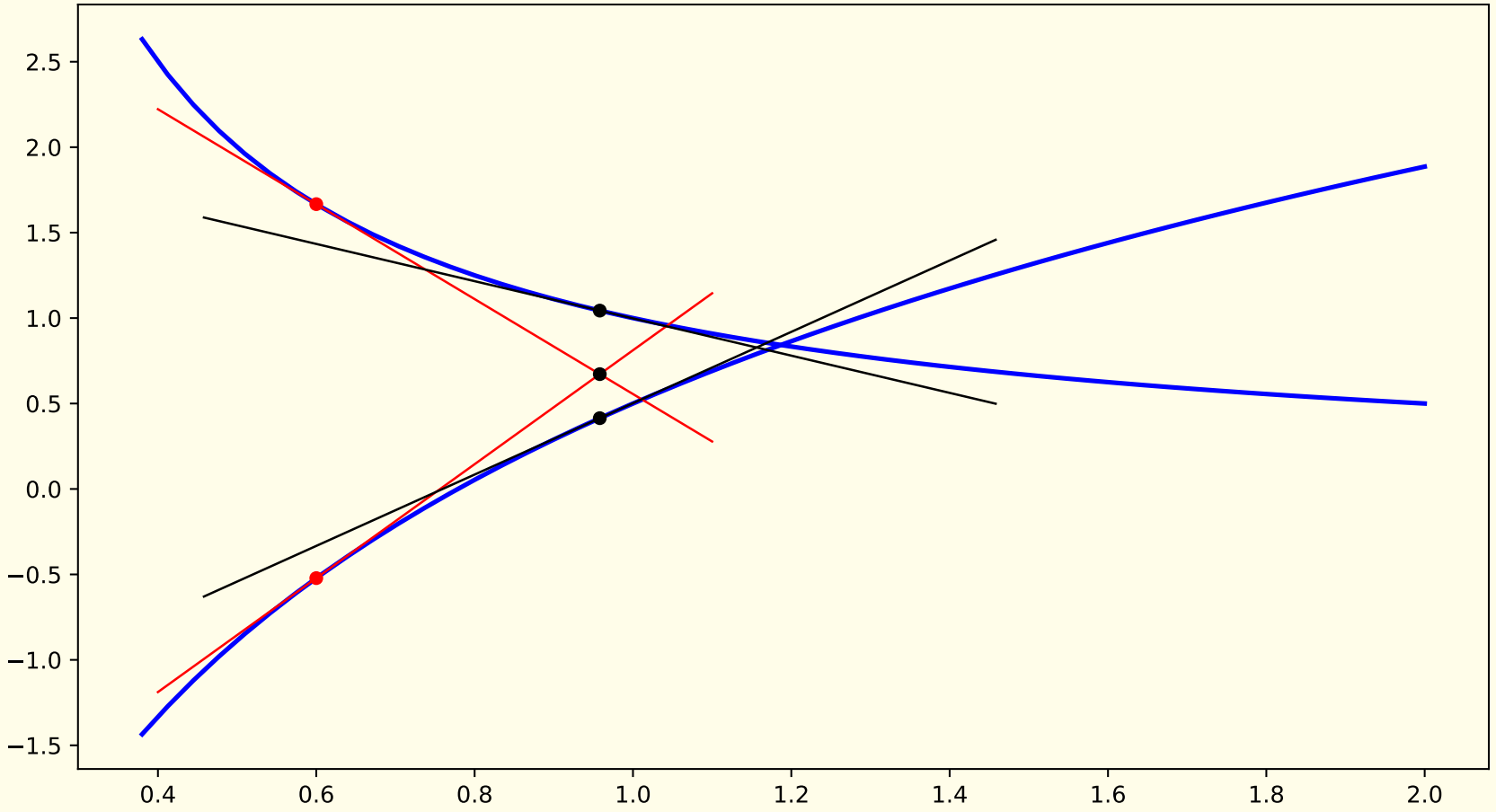





Si on veut calculer numériquement le point d'intersection
de ces deux courbes,
l'algorithme de Newton remplace la courbe par sa tangente.
Après calcul du point d'intersection des tangentes en rouge,
on recommence le processus décrit plus haut.

# Algorithme de Newton *(ii)*



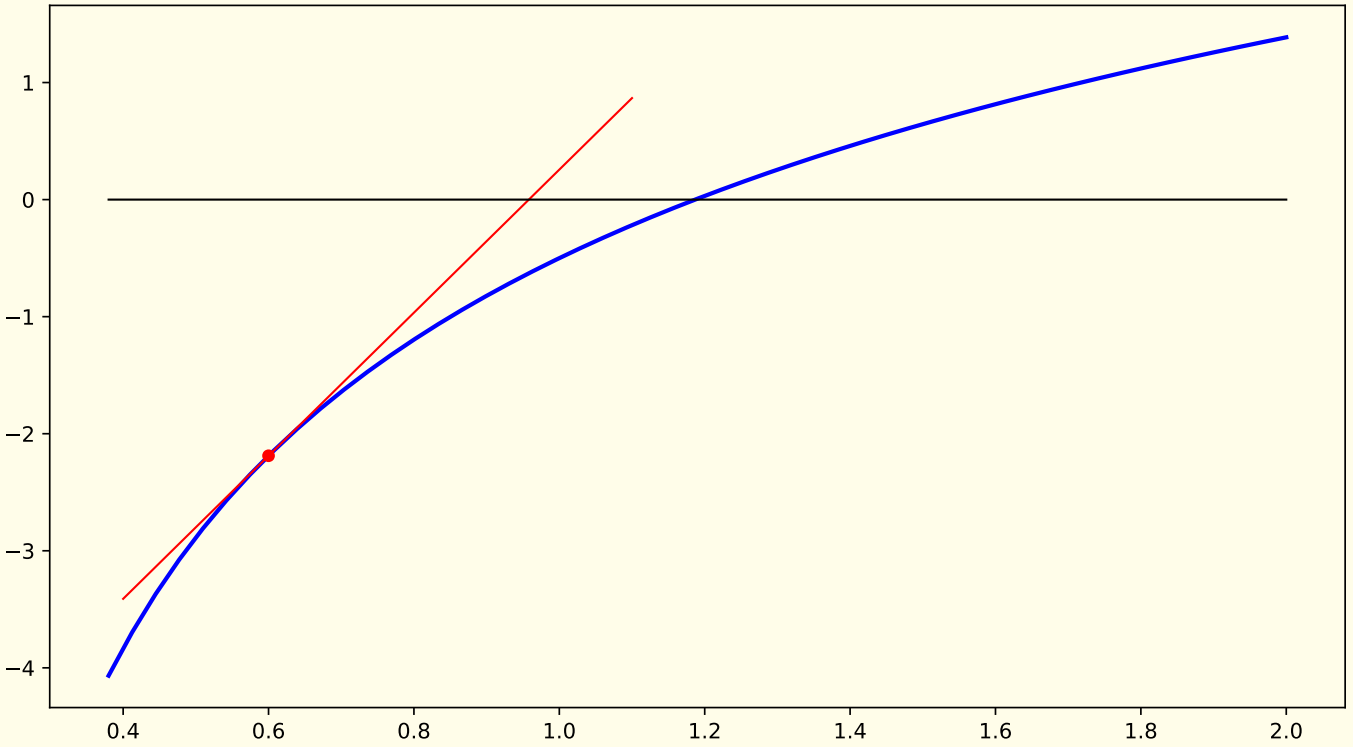



à la seconde itération de l'algorithme de Newton, le point obtenu,
intersection des deux tangentes en **noir**,
est très proche visuellement du point d'intersection
des deux courbes bleues.

# Une autre façon de poser l'algorithme de Newton



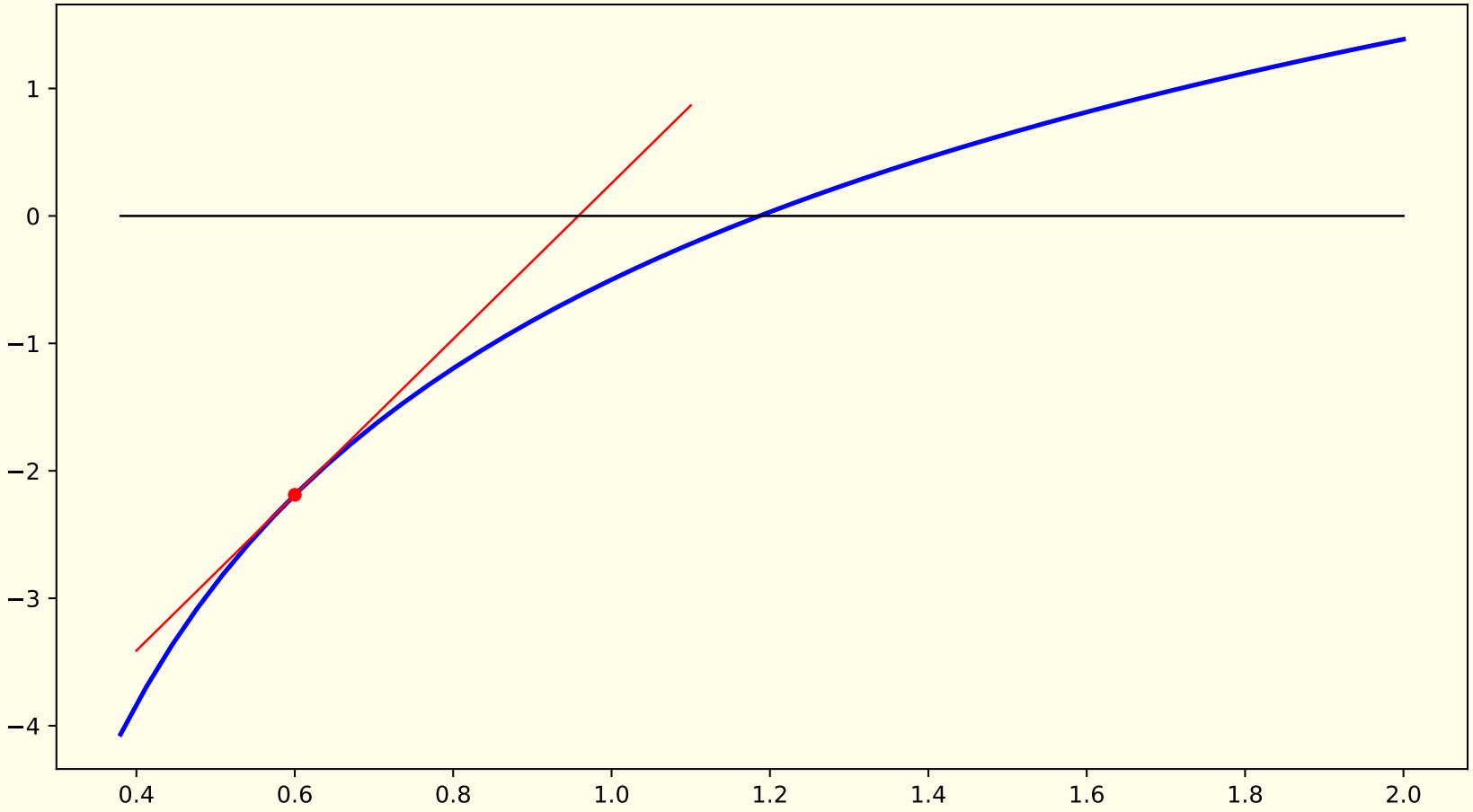
Calculer le point d'intersection des courbes représentatives des deux fonctions $f_1$ et $f_2$ revient à résoudre l'équation $a\log(x) + b - \frac{1}{x} = 0$, c'est à dire à chercher la valeur du réel $x$ qui annule la fonction $f$ définie par $f(x) = a\log(x) + b - \frac{1}{x}$.




le graphe de cette fonction est donné ci-dessus.

# Une autre façon de poser l'algorithme de Newton *(ii)*






La première itération de l'algorithme de Newton revient à calculer le point d'intersection de la tangente en rouge avec l'axe des abscisses **en noir**.

Voir les détails en travaux pratiques !