+1/1/60+

EPFL

1

Enseignant : Roger Sauser
ICS - CMS
19 avril 2023
Durée : 105 minutes

RS

SCIPER : **22**

Attendez le début de l'épreuve avant de tourner la page. Ce document est imprimé recto-verso et il contient 16 pages. Ne pas dégrafer.

- Posez **votre carte d'étudiant** sur la table.
- **Aucun** document n'est autorisé.
- L'utilisation d'une **calculatrice** et de tout outil électronique est interdite pendant l'épreuve.
- Pour les questions à **choix unique** ("multiple choice"), on comptera :
  les points indiqués si la réponse est correcte,
  0 point s'il n'y a aucune ou plus d'une réponse inscrite,
  0 point si la réponse est incorrecte.
- Utilisez un **stylo** à encre **noire ou bleu foncé** et effacez proprement avec du **correcteur blanc** si nécessaire. Toute réponse doit être rédigée en utilisant la place réservée à cet effet à la suite de la question. N'écrivez **pas dans les marges !**
- Veuillez vous conformer aux indications suivantes pour les sujets qui demandent d'**écrire du code Python** (avec papier-stylo) :
  - respectez la syntaxe Python (parenthèses, crochets, accolades, deux points, mots-clés, etc.) ;
  - mettez en forme votre code pour qu'il soit formaté exactement comme si vous le tapiez en vue d'une exécution sans erreur ;
  - respectez les indentations (en sachant que la taille de l'indentation n'importe pas en soi, mais elle doit permettre d'identifier vos blocs de code de manière claire et immédiate) ;
  - votre réponse doit comporter uniquement du code exécutable, à l'exception de quelques commentaires (au format habituel) si ceux-ci sont véritablement nécessaires et aident à la compréhension.

Respectez les consignes suivantes | Observe this guidelines | Beachten Sie bitte die unten stehenden Richtlinien

choisir une réponse | select an answer Antwort auswählen	ne PAS choisir une réponse | NOT select an answer NICHT Antwort auswählen	Corriger une réponse | Correct an answer Antwort korrigieren
		

ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte

Première partie, trois questions de “type ouvert”

Répondez dans l’espace dédié. Laissez libres les cases à cocher : elles sont réservées au correcteur.

Question 1: *Cette question est notée sur 5 points.*

Pour cette question, vous devez écrire (à l’endroit prévu à cet effet) les résultats affichés suite à l’exécution (sans erreur) de la cellule Jupyter Notebook dont le contenu est reproduit ci-dessous.
Dans vos réponses vous devez respecter le contenu et la mise en forme de chaque affichage, mais vous ne devez ajouter **aucune explication** pour justifier vos réponses.

```
import numpy as np
x = np.array([[[0,1],[2,3],[4,5]]])
y = np.arange(1,7)
print('===1===')
print(x.ndim)
print(x.shape)
print(x[0,2,0]-x[0][2][1])
print('===2===')
print(y**2)
print('===3===')
x_1 = x.reshape(6,)
print(x_1+y)
print('===4===')
y_1 = y[1:4]
print(y_1)
```

Solution

```
===1===
3
(1, 3, 2)
-1
===2===
[ 1  4  9 16 25 36]
===3===
[ 1  3  5  7  9 11]
===4===
[2 3 4]
```

Question 2: *Cette question est notée sur 3 points.*

.5 .5 .5

0 1 2 3

Soit une fonction $f : \mathbb{R} \to \mathbb{R}$ qui est deux fois continûment différentiable ($f \in C^2$).
Supposons que l'on vient de prouver que la méthode de Newton appliquée pour trouver un zéro simple $\bar{x}$ de f est convergente dans un voisinage V du zéro recherché.
Terminez la preuve ci-dessous montrant que, dans ce cas, la convergence de la méthode de Newton est quadratique.

Preuve que la convergence est quadratique :
On écrit (à l'étape n de la méthode de Newton) le développement de la fonction $f \in C^2$ autour du point x_n :

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{f''(\xi_n)}{2}(x - x_n)^2 \,,$$

où ξ_n appartient à l'intervalle (ouvert) d'extrémités x et x_n.
En imposant $x = \overline{x}$ dans cette dernière égalité et en tenant compte du fait que $f(\overline{x}) = 0$, on obtient

$$0 = f(x_n) + f'(x_n)(\overline{x} - x_n) + \frac{f''(\xi_n)}{2}(\overline{x} - x_n)^2 \,.$$

Etant donné qu'il existe un voisinage V du point fixe $\overline{x}$ tel que $f'(x_n) \neq 0$, on divise l'égalité ci-dessus par $f'(x_n)$:

$$0 = \frac{f(x_n)}{f'(x_n)} + \overline{x} - x_n + \frac{f''(\xi_n)}{2f'(x_n)}(\overline{x} - x_n)^2 \,,$$

Comme le terme suivant dans la suite de Newton est donné par $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, il s'ensuit que :

Solution
On reconnaît le terme suivant de la suite de Newton, x_{n+1} :

$$0 = \overline{x} - x_{n+1} + \frac{f''(\xi_n)}{2f'(x_n)}(\overline{x} - x_n)^2 \,.$$

Il s'ensuit que

$$|\overline{x} - x_{n+1}| = \frac{|f''(\xi_n|)}{2|f'(x_n)|}|\overline{x} - x_n|^2 \,.$$

Comme $|\overline{x} - x_{n+1}|$ représente l'erreur absolue à l'étape $n+1$ et $|\overline{x} - x_n|$ l'erreur absolue à l'étape précédente n, on note:

$$C = \frac{\max_{x \in V} |f''(x)|}{2 \min_{x \in V} |f'(x)|}$$

et on obtient

$$|\overline{x} - x_{n+1}| \leq C|\overline{x} - x_n|^2 \,.$$

Ainsi, la convergence de la suite récurrente correspondant à la méthode de Newton est bien quadratique.

Question 3: *Cette question est notée sur 15 points.*

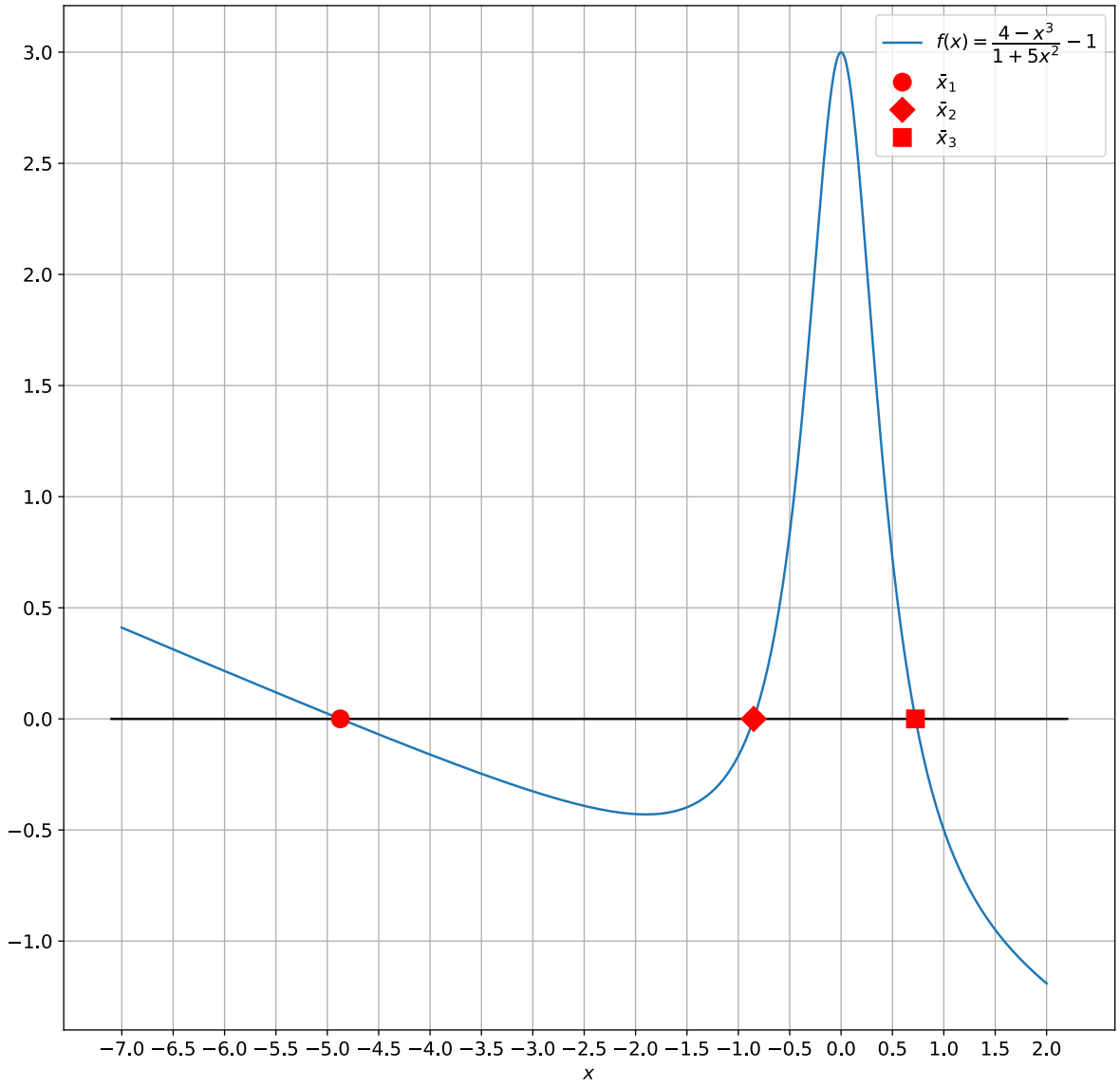
Soit la fonction réelle d'une variable réelle $f : \mathbb{R} \to \mathbb{R}$ définie par

$$f(x) = \frac{4 - x^3}{1 + 5x^2} - 1$$

dont la représentation graphique est donnée ci-dessous sur l'intervalle $[-7, 2]$.
On cherche à résoudre numériquement l'équation

$$f(x) = 0,$$

c'est-à-dire que l'on cherche à trouver les trois valeurs $\bar{x}_1$, $\bar{x}_2$ et $\bar{x}_3$ vérifiant $f(\bar{x}_1) = 0 = f(\bar{x}_2) = f(\bar{x}_3)$ (ces trois zéros de f sont indiqués sur la représentation graphique).



On vous demande ci-dessous de déterminer une approximation...

(a) ...de $\bar{x}_1$ grâce à la **méthode de la bissection** ;
(b) ...de $\bar{x}_2$ grâce à la **méthode de Newton** ;
(c) ...de $\bar{x}_3$ grâce à la **méthode de Picard**.

(a) **Approximation de $\bar{x}_1$** (notée sur 2.5 points)
Déterminez la valeur approchée du zéro $\bar{x}_1$ obtenue après trois itérations en appliquant la **méthode de la bissection** "à la main" (sans écrire de code destiné à être exécuté par un ordinateur) sur l'intervalle $[-7, -1]$.

Précisez au mieux votre raisonnement et donnez les valeurs x_1, x_2 et x_3 obtenues pour chacune des trois premières itérations. Quelle est l'erreur maximale commise sur la valeur de $\bar{x}_1$ après trois itérations ?

(b) **Approximation de $\bar{x}_2$** (notée sur 8.5 points)
Completez le code Python ci-dessous de manière à ce qu'il permette d'approcher numériquement la solution $\bar{x}_2$ de l'équation $f(x) = 0$ à l'aide de la **méthode de Newton**.

Plus précisément, le code à compléter doit :

1. définir la fonction `f` ;
2. définir la fonction `f_prime`, première dérivée de $f(x)$ par rapport à x ;
3. définir une fonction Python nommée `Newton` permettant de trouver une solution approchée de l'équation $f(x) = 0$ par la méthode de Newton. Cette fonction doit avoir comme paramètres (arguments) :
   - la fonction `f` pour laquelle on veut trouver un zéro ;
   - le point de départ `x_0` de la méthode ;
   - la fonction dérivée `f_prime` de f ;
   - la tolérance `eps` souhaitée ;
   - le nombre maximum d'itérations autorisées `n_max` .

   Cette fonction doit implémenter la méthode de Newton et itérer jusqu'à ce que l'une des trois conditions suivantes soit satisfaite :
   - soit la valeur absolue de f calculée pour la valeur approchée courante x de la solution est inférieure à la tolérance *eps* ; la fonction retourne alors la valeur approchée courante de la solution ainsi que le nombre d'itérations effectuées ;

- soit le nombre maximum d'itérations est atteint ; la fonction retourne alors la valeur approchée courante de la solution ainsi que le nombre d'itérations effectuées ;
- soit la fonction dérivée calculée pour la valeur approchée courante de la solution est inférieure à 10^{-10} ; la fonction lance alors une exception "ValueError" et affiche "Valeur trop petite de la dérivée !".

Il n'est pas nécessaire de compléter la définition de la fonction avec une docstring ;

4. faire appel à la fonction `Newton` pour trouver le zéro $\bar{x}_2$ de la fonction f définie ci-dessus (en choisissant la valeur de départ -1, en imposant une tolérance de 10^{-11} et un nombre maximum d'itérations de 30) ;
5. afficher la valeur approchée trouvée ainsi que le nombre d'itérations qui ont été nécessaires à la place des points d'interrogation dans le message suivant : "La valeur approchée trouvée après n = ? itérations est x = ? .".

```
# DEBUT DU CODE A COMPLETER
# (veuillez respecter les notations spécifiées dans l'énoncé ci-dessus)
#
# 1. définition de la fonction f




# 2. définition de la fonction f_prime, première dérivée de f
def f_prime(x):                                    # cette partie 2. du code est
    return -(40*x + 3*x**2 + 5*x**4)/(1 + 5*x**2)**2  # déjà complétée

# 3. définition de la fonction Python Newton


    nb_iterations = 0
    x = x_0
    valeur_f = f(x)

    while
        valeur_fprime = f_prime(x)
        if abs(valeur_fprime) < 1e-10:
            raise ValueError("Valeur trop petite de la dérivée !")









# 4. appel de la fonction Newton pour trouver l'approximation cherchée
x_0 =
eps =
n_max = 30



# 5. affichage de la valeur trouvée




# FIN DU CODE A COMPLETER
```

(c) **Approximation de $\bar{x}_3$** (notée sur 4 points)
Montrez que la fonction d'itération

$$g(x) = \frac{3x^3 - 12}{1 + 5x^2} + x + 3$$

est une fonction, a priori, acceptable pour une **méthode de point fixe (méthode de Picard)** destinée à trouver une valeur approchée d'un zéro de $f(x)$.

Dans cette Question, on s'intéresse à la fonction

$$f(x) = \frac{4-x^3}{1+5x^2} - 1 .$$

On observe donc que

$$g(x) = \frac{3x^3-12}{1+5x^2} + x + 3$$

$$= -3\left[\frac{4-x^3}{1+5x^2} - 1\right] + x = x - 3f(x).$$

Un zéro $\bar{x}$ de f est ainsi bien un point fixe de g :

$$g(\bar{x}) = \bar{x} - 3f(\bar{x}) = \bar{x} .$$

$f(\bar{x}) = 0$

On conclut que $g(x)$ est une fonction, a priori, acceptable pour une méthode de point fixe.

La figure en page suivante donne une représentation de $g(x)$. Indiquez sur cette figure (en utilisant règle et crayon) les trois points fixes de $g(x)$, ainsi que les points correspondant aux deux premières itérations de la méthode de Picard pour une valeur de départ $x_0 = 1$. Le point fixe $\bar{x}_3$ est-il attractif ou répulsif ?

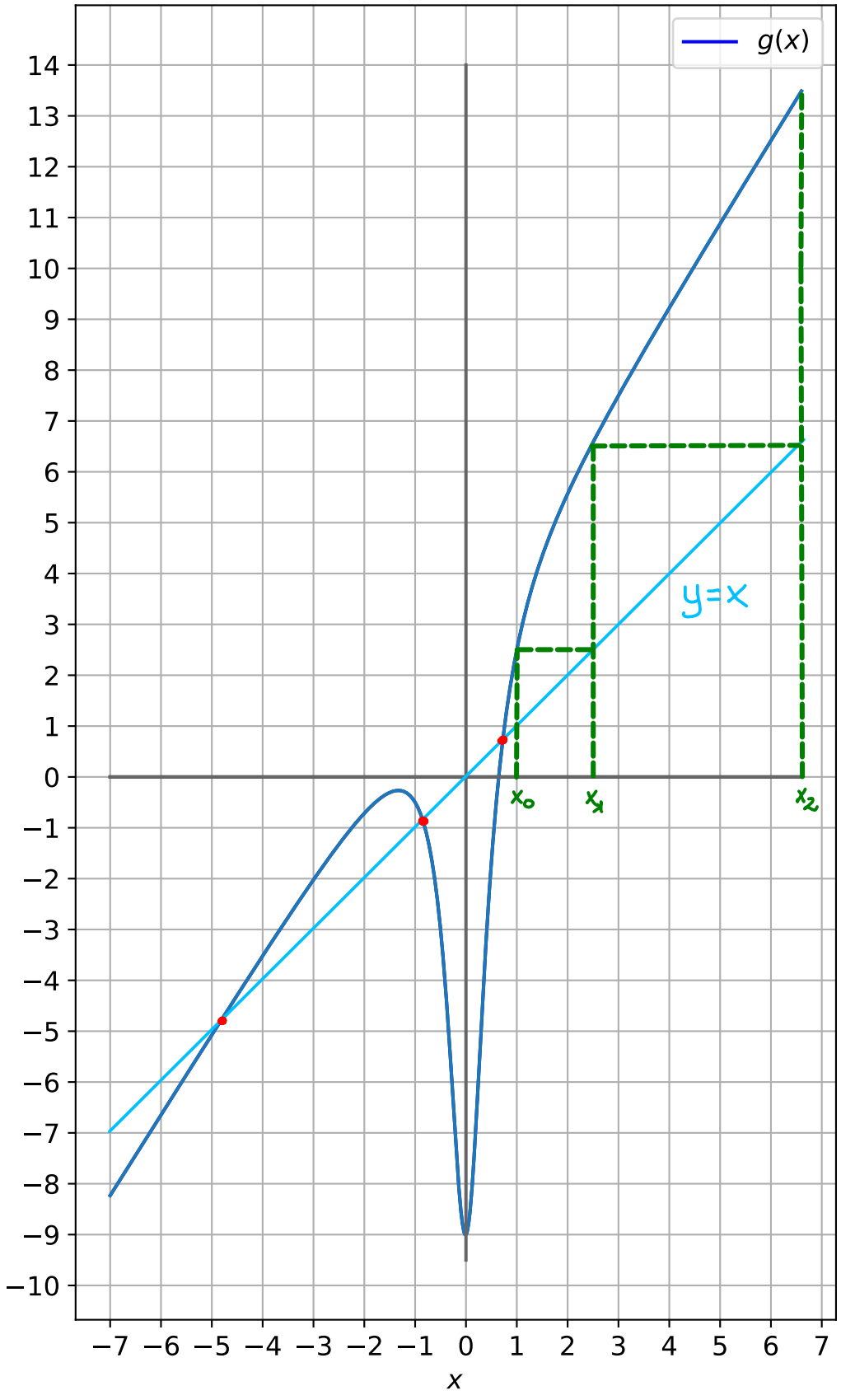
Points correspondants aux deux premières itérations :

$x_0 = 1$

$x_1 = 2.5$

$x_3 \cong 6.58$

La figure montre que le point fixe $\bar{x}_3$ est répulsif.



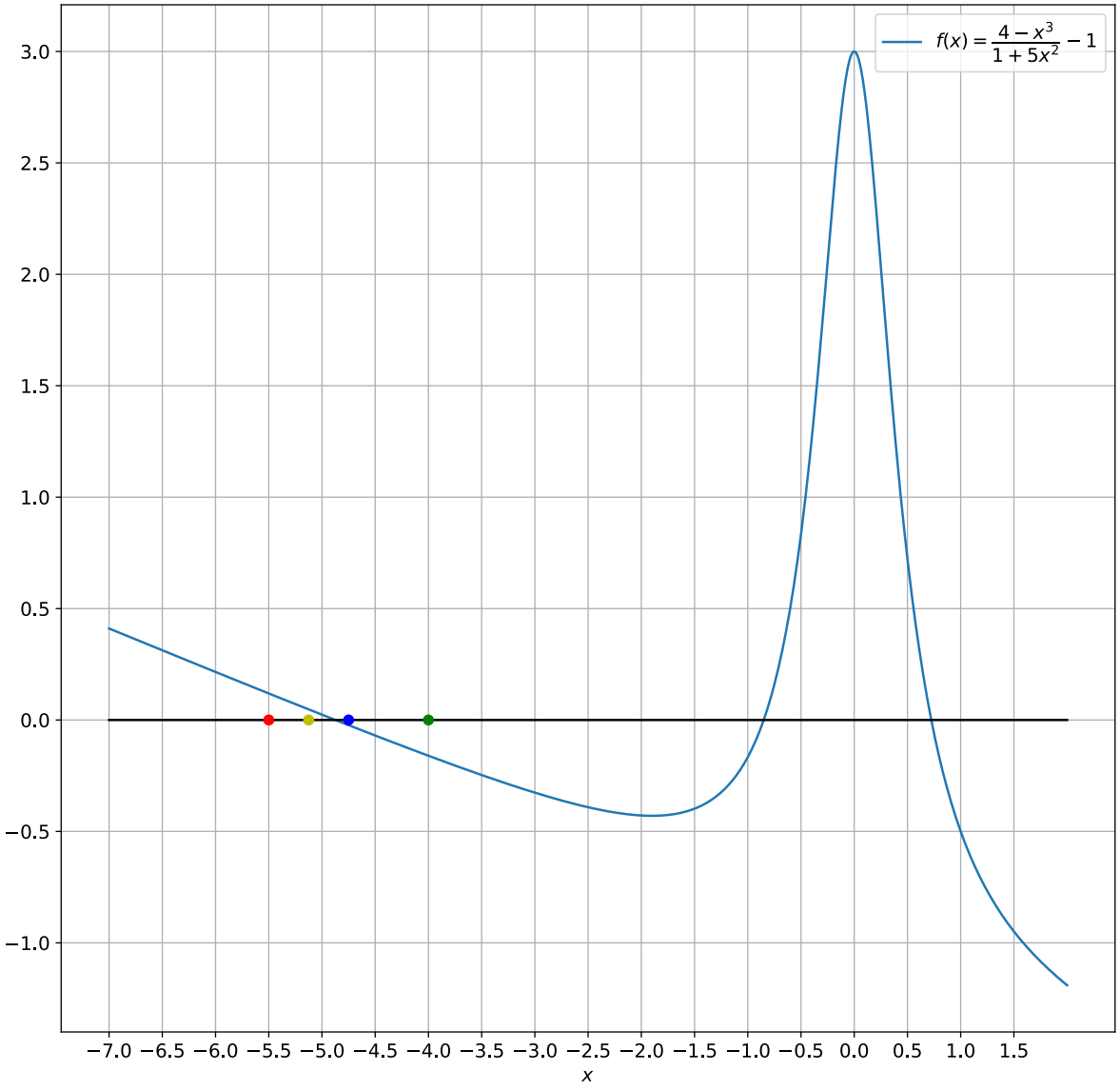
$\overline{x_1} \cong -4.87$

$\overline{x_2} \cong -0.85$

$\overline{x_3} \cong 0.72$

Solution

(a)




```
x_1 (vert) = -4.0
x_2 (rouge) = -5.5
x_3 (bleu) = -4.75
x_4 (jaune) = -5.125

erreur_max_1 = 3.0
erreur_max_2 = 1.5
erreur_max_3 = 0.75
erreur_max_4 = 0.375

x_1_bar = -4.873699902246699
```

(b)

```
# DEBUT DU CODE A COMPLETER
# (veuillez respecter les notations spécifiées dans l'énoncé ci-dessus)
#
# 1. définition de la fonction f
def f(x):
    return (4-x**3)/(1+5*x**2) -1

# 2. définition de la fonction f_prime, première dérivée de f
def f_prime(x):
    return -(40*x + 3*x**2 + 5*x**4)/(1 + 5*x**2)**2  # cette ligne est déjà complétée

# 3. définition de la fonction Python Newton
def Newton(f, x_0, f_prime, eps, n_max):
    nb_iterations = 0
    x = x_0
    valeur_f = f(x)
    while abs(valeur_f) > eps and nb_iterations < n_max:
        valeur_fprime = f_prime(x)
        if abs(valeur_fprime) < 1e-10:
            raise ValueError("Valeur trop petite de la dérivée !")
        x = x - valeur_f/valeur_fprime
        nb_iterations += 1
        valeur_f = f(x)
    return x, nb_iterations

# 4. appel de la fonction Newton pour trouver l'approximation cherchée
x_0 = -1
eps = 1e-11
n_max = 30

x, nb_it = Newton(f, x_0, f_prime, eps, n_max)

# 5. affichage de la valeur trouvée
print('La valeur approchée trouvée après',nb_it,'itérations est x =',x,'.')

# FIN DU CODE A COMPLETER
```

```
La valeur approchée trouvée après 5 itérations est x = -0.8502565872429864 .
```

Seconde partie, questions à choix unique

Pour chaque question, marquez la case correspondante à la réponse correcte sans faire de ratures.
Il n'y a qu'**une seule réponse correcte** par question.
Cette seconde partie comprend un **total de 7 points + 2 points de bonus**.

Question 4 (à 2 points)

Parmi les six expressions suivantes, laquelle donne la distance $|x_{n+1} - x_n|$ entre deux éléments successifs de la suite itérative définie par la méthode de la corde (aussi appelée méthode de Newton-corde) ?

- ☐ $\left|\frac{f(x_n)}{f'(x_n)}\right|$
- ☐ $|\lambda f(x_n)|$, où $\lambda \in \mathbb{R}$
- ■ $\left|\frac{f(x_n)}{f'(x_0)}\right|$
- ☐ $\left|\frac{f(x_n)}{\lambda}\right|$, où $\lambda \in \mathbb{R}, \lambda \neq 0$
- ☐ $\left|\frac{f'(x_n)}{f(x_n)}\right|$
- ☐ $\left|\frac{f'(x_0)}{f(x_n)}\right|$

Question 5 (à 1 point)

Parmi les cinq affirmations suivantes, laquelle est vraie pour toute fonction $g : \mathbb{R} \to \mathbb{R}$ qui est K-contractante (avec $K \in \mathbb{R}_+$, $K < 1$) ?

- ☐ La fonction g n'admet aucun point fixe $\bar{x}$ dans $\mathbb{R}$.
- ■ La fonction g possède un et un seul point fixe $\bar{x}$ dans $\mathbb{R}$.
- ☐ La fonction g possède au moins un point fixe réel positif $\bar{x}$.
- ☐ Le point $\bar{x} = 0$ est un point fixe de la fonction g.
- ☐ La fonction g admet au moins deux points fixes distincts $\bar{x}_1$ et $\bar{x}_2$ dans $\mathbb{R}$.

Question 6 (à 2 points)

Parmi les quatre fonctions de $\mathbb{R}$ dans $\mathbb{R}$ définies ci-dessous, laquelle est K-contractante (avec $K \in \mathbb{R}_+$, $K < 1$) sur tout intervalle fermé I contenu dans $\mathbb{R}$?

- ☐ $g(x) = 5x - 1 + \sin(x)$
- ☐ $g(x) = \frac{2}{3}\sin(5x^2)$
- ☐ $g(x) = x^6 + 3x - 5$
- ■ $g(x) = 5 + \frac{1}{2}\cos(x)$

Question 7 (à 2 points)

On mesure expérimentalement la position d'un objet en fonction du temps. Ces données sont regroupées dans un fichier texte, nommé `donnees.txt`, sous la forme de deux colonnes : la première contient les mesures de position, la seconde les mesures de temps correspondantes. Le fichier ne contient aucune autre information (aucun autre caractère).

Parmi les quatre cellules suivantes, qui s'exécutent sans erreur, laquelle ne représente pas correctement la position de l'objet en fonction du temps ?

■
```
import numpy as np
import matplotlib.pyplot as plt
t = np.loadtxt('donnees.txt', usecols = (1), unpack = True)
x = np.linspace(0,10000,len(t))
plt.xlabel("temps [s]")
plt.ylabel("position [m]")
plt.plot(x,t)
plt.show()
```

☐
```
import numpy as np
import matplotlib.pyplot as plt
donnees = np.loadtxt('donnees.txt', usecols = (0,1), unpack = True)
plt.xlabel("temps [s]")
plt.ylabel("position [m]")
plt.plot(donnees[1],donnees[0])
plt.show()
```

☐
```
import numpy as np
import matplotlib.pyplot as plt
donnees = np.loadtxt('donnees.txt', usecols = (0,1), unpack = True)
temps = donnees[1]
position = donnees[0]
plt.xlabel("temps [s]")
plt.ylabel("position [m]")
plt.plot(temps,position)
plt.show()
```

☐
```
import numpy as np
import matplotlib.pyplot as plt
x,t = np.loadtxt('donnees.txt', usecols = (0,1), unpack = True)
plt.xlabel("temps [s]")
plt.ylabel("position [m]")
plt.plot(t,x)
plt.show()
```

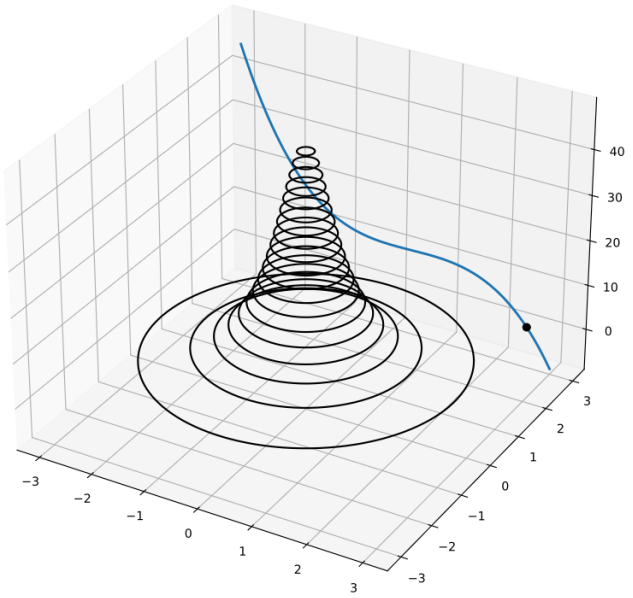
Question 8 (à 2 points, question bonus)

On aimerait que le code Python ci-dessous produise la figure ci-contre. Par quelles lignes de code doit-on remplacer le "BLOC DE CODE MANQUANT" pour que ce soit effectivement le cas ?

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize
def f(x):
    return -x**3+x**2+10
plt.figure(figsize=(10,10))
ax=plt.subplot(projection='3d')
t=np.linspace(-3,3,1000)
courbe_x=t
courbe_y=3*np.ones_like(t)
courbe_z=f(t)

# BLOC DE CODE MANQUANT

plt.show()
```



Choisissez parmi les cinq propositions suivantes, le bloc de code permettant d'obtenir la figure souhaitée :

☐
```
x=y=np.linspace(-3,3,1000)
z=20/(x**2+y**2+0.4)
ax.plot(courbe_x,courbe_y,courbe_z,linewidth=2)
ax.plot([optimize.newton(f,2)],[3],[0],'o',color='black')
ax.contour(x,y,z,22,colors='black')
```

☐
```
x=y=np.linspace(-3,3,1000)
xv,yv=np.meshgrid(x,y,indexing='ij')
zv=20/(xv**2+yv**2+0.4)
ax.plot(xv,yv,zv,linewidth=2)
ax.contour(courbe_x,courbe_y,courbe_z,22,colors='black')
```

☐
```
x=y=np.linspace(-3,3,1000)
xv,yv=np.meshgrid(x,y,indexing='ij')
zv=20/(xv**2+yv**2+0.4)
ax.plot(courbe_x,courbe_y,courbe_z,linewidth=2)
ax.contour(xv,yv,zv,22,colors='black')
```

■
```
x=y=np.linspace(-3,3,1000)
xv,yv=np.meshgrid(x,y,indexing='ij')
zv=20/(xv**2+yv**2+0.4)
ax.plot(courbe_x,courbe_y,courbe_z,linewidth=2)
ax.plot([optimize.newton(f,2)],[3],[0],'o',color='black')
ax.contour(xv,yv,zv,22,colors='black')
```

☐
```
x=y=np.linspace(-3,3,1000)
xv,yv=np.meshgrid(x,y,indexing='ij')
zv=20/(xv**2+yv**2+0.4)
ax.plot(courbe_x,courbe_y,courbe_z,linewidth=2)
ax.plot([optimize.newton(f,2)],[3],[0],'o',color='black')
ax.plot_surface(xv,yv,zv)
```