DIRO
IFT 2425

EXAMEN INTRA

Max Mignotte

DIRO, Département d'Informatique et de Recherche Opérationnelle, local 2377
Http : //www.iro.umontreal.ca/∼mignotte/ift2425/
E-mail : mignotte@iro.umontreal.ca

Date : *21/02/2019*

I	Amplification d'Erreur (36 pts)
II	Erreur en Arithmétique Flottante (45 pts)
III	Recherche des racines d'une équation (40 pts)
Total	121 points.

Tous documents personnels, calculatrices et calculateurs autorisés

I. Amplification d'Erreur (36 pts)

La machine de George Atwood est Un appareil conçu (dès 1784) et souvent utilisé en classe de physique pour l'étude de la chute libre et l'étude de la mécanique newtonienne. Cet appareil se compose de deux masses M et m (avec $M > m$) attachées aux extrémités d'une chaîne (légère) qui passe sur une poulie sans frottement. Lorsque les masses sont libérées, il est facile de montrer que la masse M accélère avec une accélération :

$$a = g\,\frac{M-m}{M+m}$$

Avec $g = 9.81$, la constante de gravitation que l'on considère ici sans incertitude, $M = 100 \pm 1$ et $m = 50 \pm 1$, (les deux masses exprimées en grammes).

1. En utilisant la méthode de propagation d'erreur (basé sur l'approximation de Taylor de la fonction $a(M, m)$ au premier ordre), donner l'approximation de a et l'incertitude commise sur a (*i.e.*, exprimer $a = a^\star \pm \Delta a$. Servez-vous ensuite de ce calcul d'erreur pour souligner les chiffres significatifs "exacts" (cse) de la précédente approximation. Arrondir finalement cette approximation au nombre de cse adéquat.
   <**10 pts**>
2. Utiliser maintenant la méthode de la fourchette pour donner l'approximation de a et l'incertitude commise sur a. Comme dans l'exercice précédent, utiliser ce calcul pour exprimer a sous la forme $a^\star \pm \Delta a$ avec l'arrondissement adéquat et en soulignant le nombre adéquat de cse.
   <**10 pts**>
3. Le calcul de $a = a^\star \pm \Delta a$ peut être fait en calculant d'abord la valeur approximée et l'incertitude de $N = (M - m)$, puis de $D = (M + m)$, puis de $(a = g \cdot N/D)$ en utilisant les formules connues (et vues en cours) de l'incertitude d'une somme/différence puis l'incertitude d'une division.
   Utiliser cette stratégie de calcul pour exprimer de nouveau a sous la forme $a^\star \pm \Delta a$ avec l'arrondissement adéquat et en soulignant le nombre adéquat de cse.
   <**8 pts**>
4. Décrire (en pseudo code ou avec des phrases) comment on pourrait implémenter une procédure informatique **automatique** pour proposer une estimation numérique de l'incertitude Δa que l'on a sur la variable incertaine a (connaissant bien sur seulement les données de ce problème ; c'est-à-dire $g = 9.81$, $M = 100 \pm 1$, $m = 50 \pm 1$ et la formule $a = g\,(M - m)/(M + m)$). Si vous donnez non pas une mais les deux méthodes (différentes) que j'indiquerais dans le correctif de cet examen, alors vous gagnez un bonus de 4 points supplémentaires.
   <**8 pts**>

Réponse

$\boxed{1}$

On a donc $\Delta M = 1$ et $\Delta m = 1$.

La valeur de a approximée ($a^\star$) est la valeur de a en m et M approximée, *i.e.*, $a^\star = 9.81 \times (100 - 50)/(100 + 50) = 3.27$ et pour calculer l'incertitude de $a(M, m$, fonction de ses deux variables incertaines=, on doit maintenant calculer la différentielle de $a(M, m)$:

$$\Delta a(M,m) = \left| -\frac{2mg}{(M+m)^2} \right| \cdot \Delta M + \left| \frac{2Mg}{(M+m)^2} \right| \cdot \Delta m$$

Numériquement, on obtient :

$$\Delta a = \left| -\frac{2 \times 50 \times 9.81}{(150)^2} \right| \cdot 1 + \left| \frac{2 \times 100 \times 9.81}{(150)^2} \right| \cdot 1 = 0.0436 + 0.0872 = 0.1308 < 0.5 \times 10^0$$

Le rang du dernier cse est le rang 0 (celui des unités), donc une façon bien rigoureuse d'écrire cette variable incertaine a en arrondissant correctement au nombre de cse adéquat est d'écrire finalement :

$$a = \underline{3}.3 \pm 0.14$$

<10 pts>

$\boxed{\mathbf{2}}$

L'énoncé nous permet d'écrire que M et m varient dans les intervalles de confiance suivants :

$$M \in [M_{\min}=99 \quad M_{\max}=101] \quad \text{et} \quad m \in [m_{\min}=49 \quad m_{\max}=51]$$

Pour utiliser la méthode de la fourchette, on doit définir avec certitude l'intervalle de confiance dans lequel se balade la vrais valeur de a, *i.e.,* $[a_{\min} \quad a_{\max}]$. Dans notre cas, on remarque que chacun des deux termes du numérateur et du dénominateur est une fonction continue et surtout monotone sur un intervalle.

De plus, on peut facilement prévoir que la borne $a_{\min}$ est définie pour $m_{\max}$ et que la borne $a_{\max}$ est définie pour $m_{\min}$. Une fois cela dit, la fonction $a(M)$ a pour dérivée $a'(M) = 2m/(M+m)^2 > 0$ (toujours positive) donc croissante pour les valeurs croissante de M. Par conséquent, la borne supérieure est donnée par $(M_{\max}, m_{\min})$ et la borne inférieure est donnée par $(M_{\min}, m_{\max})$

Pour s'en convaincre, on peut calculer toutes les combinaisons de $M_{\min}/M_{\max}$ et de $m_{\min}/m_{\max}$ (quatre en tout), et de prendre pour $a_{\min}$ (borne inférieure) la plus petite de ces quatre combinaisons et pour $a_{\max}$ (borne supérieure) la plus grande de ces quatre combinaisons. On a :

$$\begin{aligned}
a(M_{\min}, m_{\min}) &= 0.3\overline{378}\, g \approx 3.3141 \\
a(M_{\min}, m_{\max}) &= 0.32\, g \approx 3.1392 \\
a(M_{\max}, m_{\min}) &= 0.34\overline{6}\, g \approx 3.4008 \\
a(M_{\max}, m_{\max}) &= 0.328947368\, g \approx 3.2270
\end{aligned}$$

Donc un intervalle de confiance pour a de $[a_{\min}{=}0.32 \cdot g \quad a_{\max}{=}0.34\overline{6} \cdot g]$. Puisque la variable approximée est le milieu de l'intervalle et l'incertitude, la demi largeur de l'intervalle, on trouve immédiatement : $a = 3.27 \pm 0.1308$ et arrondit comme suit :

$$a = \underline{3}.3 \pm 0.14$$

<10 pts>

$\boxed{\mathbf{3}}$

L'incertitude (absolue) d'une somme ou d'une différence est la somme des incertitudes.
Donc l'incertitude (absolue) de $N{=}(M-m)$ et celle de $D{=}(M+m)$ est : $\Delta N = \Delta D = \Delta M + \Delta m = 2$.

Pour la division ou quotient $Q = g \cdot (N/D)$, l'incertitude relative de ce quotient est la somme des incertitudes relatives de N et de D (g est une variable certaine (sans imprécision), donc son erreur relative est nulle.
Donc l'incertitude relative, $\Delta a/a \approx \Delta a/a^\star = \Delta N/N + \Delta D/D = 2/50 + 2/150 = 8/150$
Donc $\Delta a = a^\star \times 8/150 = 3.27 \times 8/150 = 0.1744$

On a donc $\Delta a = 0.1744$, en arrondissant correctement au nombre de cse adéquat :

$$a = \underline{3}.3 \pm 0.18$$

<8 pts>

Nota : On remarque ici, que le fait de décomposer la formule, apporte une erreur légèrement plus grande.

$\boxed{\mathbf{4}}$

Deux possibilités :

1. La première est de générer plein de couples de valeurs (m, M) (régulièrement discrétisées ou aléatoire) dans l'intervalle $[M_{\min}, M_{\max}]$ et $[m_{\min}, m_{\max}]$ et, pour ces différents couples de valeurs, calculer $a(M, m)$. Retenir ensuite la borne inférieure et supérieure de a, c'est-à-dire $a_{\min}$ et $a_{\max}$. L'incertitude absolue de a est ensuite la demie largeur de cette intervalle.

2. La seconde est d'estimer numériquement la différentielle de la série de Taylor (de la question 1) sur ordinateur, avec ϵ petit (*e.g.,* $\epsilon = 10^{-3}$), en m et M approximés *i.e.*, en utilisant deux dérivées numériques (une en fonction de la variable M et l'autre en fonction de la variable m) :

<8 pts>
$$\Delta a = \left|\frac{a(M^\star + \epsilon, m^\star) - a(M^\star, m^\star)}{\epsilon}\right| \cdot \Delta M \quad + \quad \left|\frac{a(M^\star, m^\star + \epsilon) - a(M^\star, m^\star)}{\epsilon}\right| \cdot \Delta m$$

II. Erreur/Amplification d'Erreur en Arithmétique Flottant (45 pts)

1. Donner un exemple, en arithmétique flottante, qui ne génère pas le message "Ma ERROR" (*i.e.*, Mathematical Error) ou ∞ ou $-\infty$ et pour lequel on a plus l'associativité de la multiplication :
$$a \times (b \times c) \neq (a \times b) \times c$$
<5 pts>
2. Dans l'exemple que je vous ai donné en classe ; $102 \pm 2 \;-\; 100 \pm 2 \;=\; 2 \pm 4$
Rappeler quel est le type de problème rencontré et dire combien de chiffres significatifs perd on dans cette opération ?
<5 pts>
3. Expliquez pourquoi sur ma calculatrice ou mon ordinateur, le calcul de :
$$\left[1.0 - 3.0\left(\frac{4.0}{3.0} - 1.0\right)\right] \neq 0$$
<3 pts>
4. Expliquer pourquoi le calcul numérique de ces trois expressions :
   (a) $\frac{x^{20}}{(x+5)^{10}}$
   (b) $0.2^x - 0.1^x$
   (c) $\frac{\ln x - \sin(\pi x)}{1-x}$

   peuvent conduire à des problèmes d'erreurs numériques (identifier aussi pour quelle(s) valeur(s) de x). Identifier et citer (tous) le(s) problème(s) numérique(s) associé(s) et proposer une expression mathématiquement équivalente (ou dans le pire des cas, lorsqu'on ne peut absolument pas, une expression approximativement équivalente) qui permettrait d'éviter ces problèmes numériques et/ou augmenter la précision des calculs de ces trois expressions.
<18 pts>
5. Expliquer pourquoi en notation flottante la condition IF (X/10) … … est différente de la condition flottante IF (0.1*X) … … et précisez laquelle est la moins dangereuse.
<4 pts>
6. Dans un programme d'optimisation comme le recuit simulé, on a besoin, à chaque itération, d'une variable *temp* qui représente symboliquement une température qui, au fil des itérations d'une boucle, décroît d'une température initiale (ici, 5.0) jusqu'à une valeur finalement (à la fin de la boucle) assez proche de zéro. Expliquer le type de problème que l'on risque d'avoir dans ce petit programme en C et quelle serait la bonne façon de programmer cela pour l'éviter.

```
float To=5.0;
float Temp=To;
float CoolRate=0.99999;
for(int k=0;k<1000000;k++)
{
Temp = Temp * CoolRate;
...
...
}
```

<**4 pts**>

7. On veut estimer numériquement la somme suivante, faite sur un vecteur 1D de longueur 1 milliard de termes :

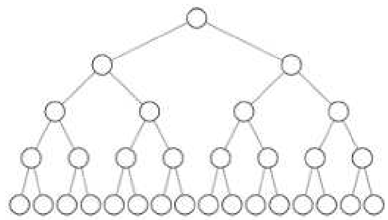
$$\sum_{n=0}^{n=10^9} \gamma(n) \qquad \text{avec} \quad 0 < \gamma(n) < 1 \tag{1}$$

et on programme et teste quatre stratégies :

(a) On programme cette sommation telle quelle.

(b) On réarrange les $\gamma(n)$ par ordre croissant et on réalise la somme sur ce nouveau vecteur contenant les $\gamma(n)$ croissants

(c) On additionne deux à deux les termes de cette somme comme le schéma de l'arbre binaire ci dessous



(d) On réarrange les $\gamma(n)$ par ordre croissant et on additionne deux à deux les termes de cette somme comme le schéma de l'arbre binaire.

Commenter et classez ces quatre stratégies et leur moindre sensibilité aux erreurs numériques et en fonction aussi de la répartition (*i.e.*, comment se distribuent) les $\gamma(n)$.

<**6 pts**>

Réponse

$\boxed{\mathbf{1}}$

Par exemple sur ma calculatrice *CASIO fx-7400G PLUS :*

$$\begin{aligned} 10^{-60} \times \left(10^{-60} \times 10^{50}\right) &= 10^{-70} \\ &\text{et} \\ \left(10^{-60} \times 10^{-60}\right) \times 10^{-50} &= 0 \end{aligned}$$

on peut donner aussi comme exemple (avec EPS=l'epsilon machine) :

$$\begin{aligned} 10 \times \left(0.25 \times EPS\right) &= 0 \\ &\text{et} \\ \left(10 \times 0.25\right) \times EPS &= 4EPS \end{aligned}$$

<**5 pts**>

$\boxed{\mathbf{2}}$

On a une opération du type : $V_1^\star \pm \Delta V_1 \; - V_2^\star \pm \Delta V_2 \; = \; R^\star \pm \Delta R$ qui pose le problème de la soustraction de deux nombres approximés quasi égaux avec $\Delta V_1 = \Delta V_2 = 2 < 0.5{\times}10^1$. Le rang du dernier chiffre significatif exact (cse) dans V_1 et V_2 est celui des dizaines donc deux cse dans V_1 et V_2 (celui des dizaines et centaines).

Pour le résultat final $\Delta R = 4 < 0.5{\times}10^1$, il n'existe aucun cse. On perd donc deux cse dans cette opération.

<**5 pts**>

$\boxed{\mathbf{3}}$

$(4/3) - 1 = 1/3$ n'a pas de représentation exacte ($(1/3)_2 = 0.\overline{01}010101...$) donc ce produit avec 3 ne nous donnera pas 1 et le résultat sera différent de 0.

<3 pts>

Nota : Avec ma calculatrice *CASIO fx-7400G PLUS :* j'obtiens 10^{-14} et sur mon ordinateur *Intel Core i7-4930K CPU @ 3.40GHz* (programmation C sous Linux en double) j'obtiens $x = 2.22044604925010^{-16}$.

$\boxed{4}$

[-a-] Mis à part la valeur interdite $x = 5$ (qui restera toujours !), pour la première expression, on aura possiblement un problème d'OVERFLOW lorsque $x \to \infty$ et un problème d'UNDERFLOW lorsque $x \in [0\ 1]$. Pour minimiser ces types de problème, il vaut mieux réécrire cette expression en la simplifiant de la façon suivante :

$$\frac{x^{20}}{(x+5)^{10}} = \frac{x^{10}}{(1+\frac{5}{x})^{10}}$$

<6 pts>

[-b-] Pour la deuxième expression, on sait que 0.1 et 0.2 n'ont pas de représentation exacte (0.2 résultant d'une multiplication de 0,1 par deux, c'est-à-dire du simple décalage de 0.1 d'un bit vers la gauche).
En plus cette expression pose le problème d'annulation de cse (chiffres significatifs exactes) due à la soustraction de deux nombres approximés quasi égaux à deux endroits différents quand $x \to 0$ et quand $x \to \infty$.

Pour $x \to 0$, on peut faire un développement limité au voisinage de 0, en essayant de réécrire 0.1 par 1/10 et 0.2 par 1/5 si possible), on obtient :

$$\begin{aligned} 0.2^x - 0.1^x &= \exp\big[(\ln 0.2)x\big] - \exp\big[(\ln 0.1)x\big] \approx 1 + (\ln 0.2)x + o(x^2) - \big[1 + (\ln 0.1)x + o(x^2)\big] \\ &= x(\ln(0.2) - \ln(0.1)) = x(\ln 10 - \ln 5) = x \ln(2) \quad \text{quand} \;\; x \to 0 \end{aligned}$$

qui est valide aussi pour $x \to \infty$.

Pour $x \to \infty$, on peut réécrire l'expression pour ne plus faire apparaître la soustraction de deux nombres approximés quasi égaux (proche de 0) :

$$I = 0.2^x - 0.1^x = \frac{2^x - 1}{10^x} \quad \text{quand} \;\; x \to \infty$$

$$\text{ou} \quad I = 0.2^x - 0.1^x = \Big(\frac{1}{5}\Big)^x - \Big(\frac{1}{10}\Big)^x = \frac{1}{5^x} - \frac{1}{10^x} = \frac{1}{5^x} - \frac{1}{2^x\,5^x} = 5^{-x}(1 - 2^{-x}) \quad \text{quand} \;\; x \to \infty$$

<6 pts>

Nota -1- : Lorsque $x \to 0$, on peut proposer aussi $0.2^x - 0.1^x = 0.1^x(2^x - 1) \approx 0.1^x(x \log 2)$.

Nota -2- : Lorsque $x \to 0$, l'expression I tend vers $x\ln(2) \approx 0.6931471806 \cdot x$.
Avec ma calculatrice *CASIO fx-7400G PLUS :* j'obtiens, pour $x = 10^{-12} \triangleright I = 7 \times 10^{-13}$ alors que le développement limité donne $I = 6.9314718 \times 10^{-13}$, expression exacte avec tout les cse.

[-c-]

Pour la troisième expression, en plus des valeurs pour lesquelles x est négative ou nulle et où cette expression n'est bien sur pas définie à cause du logarithme, il y a deux problèmes lorsque $x \to 1$. Le premier est un problème d'annulation de cse au numérateur et dénominateur. Le deuxième est une expression dangereuse indéterminée du type 0/0 pouvant potentiellement engendrer en notation flottante un INF ou NaN (ou Ma ERROR selon les calculatrice). Le plus simple, pour cette expression est de considérer, pour des valeurs proche de 1, son développement limité :

$$\text{Au voisinage de } x = 1, \text{ on a :} \quad f(x) \approx f(a) + f^{\prime)}(a)(x-a) + \tfrac{f''(a)}{2}(x-a)^2$$

$$\text{Donc :} \quad \ln(x) \approx (x-1) - \tfrac{1}{2}(x-1)^2 + o(x^3) \quad \text{et} \quad \sin(\pi x) \approx -\pi(x-1) + o(x^3)$$

Pour $x \to 1$, on peut réécrire l'expression I :

$$\frac{\ln x - \sin(\pi x)}{1-x} = \frac{(x-1) - \frac{1}{2}(x-1)^2 + \pi(x-1)}{1-x} = -1 + (1/2)(x-1) - \pi = -\frac{3}{2} + \frac{x}{2} - \pi \text{ quand } x \to 1$$

<6 pts>

Nota : Lorsque $x \to 1$, l'expression I tend vers $(-\pi - 1) \approx -4.141592654$.
Avec mon ordinateur *Intel Core i7-4930K CPU @ 3.40GHz* (programmation C sous Linux en double) j'obtiens, pour $x = 1 + 10^{-12} \triangleright I = -4.34845655$ et pour $x = 1 - 10^{-12} \triangleright I = -3.94472888$ alors qu'avec l'expression du développement limité, j'obtiens dans les deux cas : $I = -4.14159265$, expression exacte avec tout les cse.

$\boxed{5}$

0.1 n'a pas de représentation exacte en notation flottante contrairement à 10 ($10 = (0.1010 \times 2^4)$. Les deux notations seront donc différentes et la moins dangereuse est bien sur la première.

<4 pts>

$\boxed{6}$

On risque une perte de précision pour la variable $Temp$.
Il faut mieux changer l'instruction :

TEMP=TEMP*COOLRATE ; par l'instruction TEMP=T0*POWF(COOLRATE,K) ;

qui calcule en fait temp par l'expression mathématique équivalente : $Temp = T0 \times CoolRate^k$.

<4 pts>

Nota : Avec mon ordinateur *Intel Core i7-4930K CPU @ 3.40GHz* (programmation C sous Linux en float) j'obtiens, pour la 500000 ième itération $temp = 0.0336906984$ dans le premier cas et $temp = 0.0334612466$ dans le deuxième.

$\boxed{7}$

La technique (a) sera sensible au fait, qu'en notation flottante, l'addition de deux nombres d'ordre de grandeur différent est source d'erreur numérique.

La technique (b) permettra d'améliorer un peu les choses et d'avoir un peu moins d'erreur numérique. Mais le gain sera d'autant moins grand que les $\gamma(i)$ sont mal equi-répartis uniformément autour de 0 et 1.

La troisième technique est la plus solide des quatre et évitera l'addition de deux nombres de grandeurs différents (quel que soit la répartition des $\gamma(i)$).

La quatrième technique ne sera pas meilleur que la troisième. Le réarangement par ordre croissant, dans ce cas, de somme en arbre binaire. est une perte de temps.

<6 pts>

III. Méthode de la Bissection/Newton/Point Fixe (40 pts)

On se propose de trouver numériquement dans R une valeur approchée de la racine de la fonction

$$f(x) = \cos(x) - x^3 \tag{2}$$

1. Méthode du Point Fixe
   (a) Montrer qu'il existe une racine unique r pour cette Eq. (2) dans l'intervalle $J = [0.0\ ,\ 1.0]$ et donner une forme $x = g_1(x)$, mathématiquement équivalente de $f(x) = 0$ (Eq. (2)), pour laquelle la suite itérative $r_{n+1} = g_1(r_n)$ converge lorsque, le premier élément de cette suite est $r_0 = 0.5$.
   <8 pts>
   (b) Calculer les 6 premières estimées $r_1, \ldots, r_6$, de cette suite itérative en partant de $r_0 = 0.5$.
   <6 pts>
   (c) Donner une forme $x = g_2(x)$, mathématiquement équivalente de $f(x) = 0$ (Eq. (2)), pour laquelle la suite itérative $r_{n+1} = g_2(r_n)$ ne converge EXPÉRIMENTALEMENT pas (on prendra toujours $r_0 = 0.5$, le premier élément de cette suite).
   <3 pts>
2. Méthode de la Bissection

(a) Si on avait utilisé la méthode de la bissection, avec combien d'itérations serait-on arrivé à une valeur approchée de la racine à 10 cse après la virgule ? (Nota : on vous demande d'obtenir une estimation de ce nombre en utilisant les propriété de la méthode de la bissection mais sans calculer les différentes estimation $r_0, r_1, \ldots,$ donné par cette méthode)
<**5 pts**>

3. Méthode de Newton

(a) Donner la relation itérative $r_{n+1} = g_{\text{newt.}}(r_n)$ intervenant dans la méthode itérative de Newton pour la résolution itérative de la racine r de l'équation $f(x) = 0$.
<**5 pts**>

(b) Montrer qu'avec $r_0 = 0.5$, la relation itérative de Newton converge.
<**5 pts**>

(c) Calculer les 4 premières estimées $r_1, \ldots, r_4$, en partant de $r_0 = 0.5$.
<**5 pts**>

(d) Pourrait-on savoir a l'avance que cette méthode allait être la plus rapide. Expliquer pourquoi.
<**3 pts**>

Réponse

$\boxed{\textbf{1.(a)}}$

L'étude des variations de la fonction f sur $J = [0.0, 1.0]$ montre que la fonction est continue et décroissante sur J, donc monotone car $f'(x) = -\sin(x) - 3x^2 < 0$ sur J.
De plus, on a $f(0) = 1$ et $f(1.0) \approx -0.4597$ donc $f(0)f(1) < 0$ et, puisque la fonction f est continue et décroissante sur J, il existe donc une racine r unique dans cet intervalle.
<**3 pts**>

De plus $f(x) = 0$ est équivalent à l'équation $x = (\cos(x))^{1/3} = g_1(x)$ avec :

$$|g'(x)| = \left|(-1/3)\frac{\sin(x)}{(\cos(x))^{2/3}}\right|$$

Sur $J = [0.0\,,\,1.0]$ la fonction $\sin(x)$ est croissante (jusqu'à $\pi/2$) et la fonction $\cos(x)$ est décroissante (jusqu'en π) et à valeur positive jusqu'en $\pi/2$). Donc ; la fonction $\sin(x)/(\cos(x))^{2/3}$ est croissante sur $J = [0.0, 1.0]$. Sur $J = [0.0\,,\,1.0]$, $|g'(x)|$ sera minimale en $|g'(x=0)|$ et maximale sur $|g'(x=1)|$. Donc :

$$\text{Sur} \quad J = [0.0\,,\,1.0], \quad |g'(x)| < |g'(x=1)| \approx 0.423$$

La fonction $g(x)$ est donc contractante sur J et la convergence de notre point fixe est donc assurée.
<**5 pts**>

$\boxed{\textbf{1.(b)}}$

En partant de $r_0 = 0.5$, on a, $r_n = g_1(r_{n-1})$ ou $r_n = (\cos(r_{n-1}))^{1/3}$ et,

$$\begin{aligned} r_1 &= 0.9574056696 \\ r_2 &= 0.8318617459 \\ r_3 &= 0.8765553835 \\ r_4 &= 0.8616851126 \\ r_5 &= 0.8667538751 \\ r_6 &= 0.8650399272 \end{aligned}$$

qui va converger doucement vers la valeur $r = 0.8654740331$.
<**6 pts**>

Nota : Pour ceux qui m'ont indiqués dans leur devoir qu'ils n'ont pas le temps de pitonner (!) cette suite itérative ! (certains en mettant etc...!) Il est impératif de savoir qu'une suite itérative se programme

facilement sur n'importe quel calculatrice de base !
Avec une calculatrice du type *CASIO fx-7400G PLUS :*, par exemple, cette suite itérative se programme facilement en affichant 0.5 puis $\boxed{\text{EXE}}$ puis en affichant $(\cos(ANS))^{1.0/3.0}$. Chaque appel à la touche $\boxed{\text{EXE}}$ nous permet ensuite d'avoir une itération de la suite !

1.(c)

$f(x) = 0$ est équivalent à l'équation $x = (\arccos(x^3)) = g_2(x)$. En partant de $r_0 = 0.5$, on a $r_{n+1} = \arccos(r_n^3)$ qui diverge tout de suite.

<3 pts>

Nota : $f(x) = 0$ est équivalent aussi à l'équation $x = \cos(x) - x^3 + x = g_2(x)$ ou $x = \cos(x)/x^2 = g_2(x)$. En partant de $r_0 = 0.5$ qui diverge aussi.

2.

La méthode de la bissection permet de construire à partir de l'intervalle $[0.0\,,\,1.0]$ contenant r, un nouvel intervalle de longueur moitié contenant r. En appliquant n fois consécutives la méthode, on obtient un intervalle de longueur $1/2^n$ contenant r. A l'itération n, on a un majorant de l'erreur absolu donné par $\Delta r = \frac{|1|}{2^n}$. Or on veut $\Delta r < 0.5 \times 10^{-10}$, donc $2^n > 2 \times 10^{10}$ ce qui, dans notre cas est vrai des que $n = 35$.
<5 pts>

3.(a)

La fonction $f(x)$ est dérivable sur $\mathbb{R}$ et la méthode itérative de Newton permet d'écrire :

$$r_{n+1} = r_n - \frac{f(r_n)}{f'(r_n)} = r_n + \frac{\cos(r_n) - r_n^3}{\sin(r_n) + 3\,r_n^2}$$

<5 pts>

3.(b)

Dans ce cas, le plus simple est de montrer que la fonction $f(x)$, sur l'intervalle $J = [0, 1]$ (intervalle dans lequel on prendra $r_0 = 0.5$ et dans lequel une racine unique s'y trouve) ne présente pas d'*extrema*, i.e., de valeurs pour laquelle, $f'(x)$ s'annule.

Dans notre cas, $f(x) = \cos(x) - x^3 = 0$ et $f'(x) = -\sin(x) - 3x^2 < 0$, toujours négative sur J donc aucun problème.
<5 pts>

3.(c)

En partant de $r_0 = 0.5$, on a, $r_n = g_{\text{newt.}}(r_{n-1})$ et :

$$\begin{aligned} r_1 &= 1.112141637 \\ r_2 &= 0.9096726937 \\ r_3 &= 0.8672638182 \\ r_4 &= 0.8654771353 \end{aligned}$$

<5 pts>

Nota -1- : Sur ma calculatrice, l'estimation se stabilise très rapidement, à partir de $r_5 = 0.8654740331$!! (à comparer avec la méthode du point fixe précédente qui se stabilisait à partir de $r_{>22}$). Cela nous indique aussi que la fonction dont on cherche la racine est très linéaire au voisinage de la racine.
On peut voir l'effet d'une racine simple sur la convergence de Newton qui, dans ce cas, est quadratique.

Nota -2- : Avec une calculatrice du type *CASIO fx-7400G PLUS :*, par exemple, cette suite itérative se programme facilement en affichant 0.5 puis $\boxed{\text{EXE}}$ puis en affichant $ANS + (\cos(ANS)) - ANS^3)/(\sin(ANS) +$

$3 \times ANS^2$). Chaque appel à la touche $\boxed{\text{EXE}}$ nous permet ensuite d'avoir une itération de la suite. Cela évite de remplir une page de calcul dans son devoir et de sauver plusieurs minutes !

$\boxed{\textbf{3.(d)}}$

Racine simple recherchée (montré à la question 1.(a) avec un intervalle qui encadre la racine) donc convergence quadratique pour la méthode de Newton.

Graphiquement la fonction à un comportement linéaire au voisinage de la racine caractéristique d'une convergence quadratique pour la méthode de Newton.

<3 pts>

Nota : Cette méthode est plus rapide (et la seule quadratique) que la méthode du point fixe de la question 1. car $g'(r \approx 0.8659) \approx 0.338 \neq 0$