Lycée Thiers	**TD : Feuille d'Exercices 8** *Méthodes numériques*	Informatique 2BCPST1

1) - Recherche dichotomique (diviser pour régner)

La recherche dichotomique d'un élément dans un objet consiste à diviser l'objet en deux sous-objets dont les tailles sont les plus proches possibles et à déterminer dans quel sous-objet l'élément est sucseptible de se trouver. On applique alors ce qui précède à ce sous objet et on répète ce processus un nombre de fois suffisant pour répondre à la question posée.

Recherche dichotomique d'une solution de $f(x) = k$

Soit f est une fonction continue sur le segment $[a, b]$ avec ou bien $f(a) \leqslant k \leqslant f(b)$ ou bien $f(b) \leqslant k \leqslant f(a)$.

Description de la méthode de dichotomie dans le cas $f(a) \leqslant k \leqslant f(b)$ (cas particulier : f croissante)
On pose $c = \frac{a+b}{2}$. Si $f(c) \geqslant k$ alors on est sûr de trouver une solution de $f(x) = k$ dans $[a, c]$ donc on remplace $[a, b]$ par $[a, c]$. Sinon on remplace $[a, b]$ par $[c, b]$. On répète ce processus jusqu'à ce que la longueur de l'intervalle soit inférieure ou égale à la précision de l'approximation souhaitée.

Dans le cas $f(b) \leqslant k \leqslant f(a)$ (cas particulier : f décroissante), on fait le même test puis le choix opposé.

EXERCICE 1 Approximation d'une solution de l'équation $x \ln x = 2$ par la méthode de dichotomie

1. Écrire une fonction python `dicho(f,a,b,k,p)` qui renvoie une approximation d'une solution de $f(x) = k$ à la précision p sur $[a, b]$ par la méthode de dichotomie en supposant que $f(a) \leqslant k \leqslant f(b)$.
   (Facultatif) coder une fonction `dichoBis(f,a,b,k,p)` dans le cas $f(b) \leqslant k \leqslant f(a)$.
2. Tester cette fonction avec $f : x \mapsto x \ln x$ sur l'intervalle $[1, 3]$, $k = 2$ et $p = 10^{-10}$.

Recherche dichotomique d'un nombre dans une liste numérique triée

On compare le nombre cherché au terme central de la liste. S'il y a égalité on arrête la recherche sinon on choisit la demi-liste qui est susceptible de contenir le nombre et on recommence. Si à la fin de cet algorithme on n'a pas trouvé l'objet dans la liste alors on peut affirmer que l'objet ne s'y trouve pas.

EXERCICE 2 Recherche dichotomique dans une liste numérique quelconque

1. Écrire une fonction `recherche(x,lst)` qui renvoie le booléen `True` si le nombre `x` se trouve dans la liste numérique `lst` et le booléen `False` sinon. On commencera par trier `lst` soit par la méthode `sort` (commande : `lst.sort()`) soit en appliquant un tri rapide avec pivot aléatoire.
2. Tester cette fonction avec la liste `lst=[15,4,9,6,2,7,10,20,5]` et les nombres `20` et `3`.

2) - Méthode d'Euler

La méthode d'Euler est un algorithme itératif permettant d'approcher la solution d'une équation différentielle à l'aide de développements limités.
Soit (E) l'équation différentielle $y' = F(x, y)$.
Si y désigne la solution de (E) sur $[x_0, b]$ vérifiant $y(x_0) = y_0$ et (x_i) une suite de points de $[x_0, b]$ tels que $x_{i+1} = x_i + h$ alors le $\mathrm{DL}_1(x_i)$ de y s'écrit $y(x_{i+1}) = y(x_i) + (x_{i+1} - x_i)y'(x_i) + o(x_{i+1} - x_i)$, autrement dit $y(x_{i+1}) = y(x_i) + hF(x_i, y(x_i)) + o(h)$ donc si h est "petit" alors $y(x_i) + hF(x_i, y(x_i))$ est une approximation de $y(x_{i+1})$.

On retiendra que pour la suite définie par récurrence $y_{i+1} = y_i + hF(x_i, y_i)$, y_i est une approximation de $y(x_i)$.

EXERCICE 3 Méthode d'Euler appliquée à une EDL1

On considère l'équation différentielle linéaire $(E) : xy'(x) + 2y(x) = \dfrac{x^2}{1 + x^2}$
Soit y une solution de (E) sur $[x_0, b]$, prenant la valeur y_0 au point x_0.

On note $F(x, y) = \dfrac{x}{1 + x^2} - 2\dfrac{y}{x}$ et h un réel positif "assez petit".

1. Écrire une fonction Python `F` de paramètres x et y qui renvoie la valeur de $F(x, y)$.
2. On construit par récurrence la suite (y_i) : $y_{i+1} = y_i + hF(x_i, y_i)$ (formule à retenir).
   Écrire une fonction `euler` qui renvoie, à partir des paramètres x_0, y_0, h, b, les deux listes $[x_i]$ et $[y_i]$.
3. Écrire une fonction python `courbe` d'arguments `x0,y0,h,b` qui trace la courbe obtenue par la méthode d'Euler. On choisit $h = 0.001$ et $b = 7$. Appeler `courbe` pour les conditions initiales $(0.005, 2)$, $(0.05, 2)$, $(0.2, 2)$, $(0.5, 2)$, $(1, 2)$, $(2, 2)$.
4. (Facultatif) Expliquer brièvement en commentaire dans le module de travail comment on adapte la fonction `euler` au cas où la condition initiale n'est pas relative à la borne inférieure de l'intervalle de représentation.

3) - Méthode de Newton

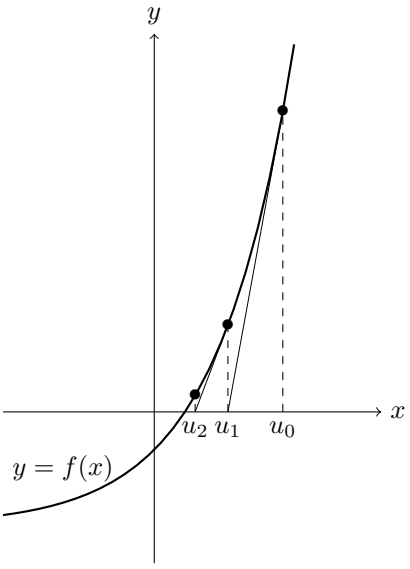
La méthode de Newton est un algorithme itératif permettant d'approcher une solution de l'équation $f(x) = 0$ où f est une fonction de classe C^1 vérifiant certaines conditions.

Soit f une fonction de classe C^1 sur un intervalle $[a, b]$ telle que $f(a)f(b) < 0$ et $f'(x) \neq 0$ sur $[a, b]$. Soit $u_0 \in [a, b]$.
On suppose définis les termes $u_0, \dots, u_n$ et on va définir u_{n+1} :
soit $\mathscr{T}_n$ la tangente à $\mathscr{C}_f$ au point u_n. Le nombre u_{n+1} est l'abscisse du point d'intersection de $\mathscr{T}_n$ avec l'axe des abscisses.

On définit ainsi par récurrence une suite $(u_n)_{n\in\mathbb{N}}$ lorsque f et u_0 vérifient certaines conditions que l'on n'exposera pas dans ce document.

On dit que la suite $(u_n)_{n\in\mathbb{N}}$ est obtenue à partir de la fonction f et du premier terme u_0 par la méthode de Newton.

Un calcul facile montre que $\boxed{u_{n+1} = u_n - \dfrac{f(u_n)}{f'(u_n)}}$ (formule à connaitre ou à établir à partir de l'illustration).
Si (u_n) converge vers ℓ alors par unicité de la limite et continuité, $\ell = \ell - \frac{f(\ell)}{f'(\ell)}$ donc $f(\ell) = 0$.



Illustration

EXERCICE 4 Approximation d'une solution de $x^2 - a = 0$ par la méthode de Newton

1. Écrire une fonction `babylone` de paramètres `a`, `p`, qui renvoie une valeur approchée de $\sqrt{\texttt{a}}$ à la précision `p` par la méthode de Newton ainsi que le nombre d'itérations. On choisira $u_0 = a$. On admettra que $|u_n - \sqrt{2}| < |u_{n-1} - u_n|$.
2. À l'aide de cette méthode, déterminer une valeur approchée de $\sqrt{2}$ à 10^{-10} près. Combien d'itérations ont-elles été nécessaires pour obtenir cette valeur approchée ? Comparer avec le nombre d'itérations nécessaires par la méthode de dichotomie à partir de l'intervalle $[1, 2]$ (on ne réécrira pas l'algorithme de dichotomie, on se basera uniquement sur le fait que l'intervalle initial $[a, b]$ est réduit de moitié à chaque itération).

EXERCICE 5 Méthode de Newton : cas général

1. Écrire une fonction `derivee(f,x)` qui renvoie $\frac{f(x+10^{-13}) - f(x)}{10^{-13}}$.
2. Écrire une fonction `newton` d'arguments `f`, `u0`, `p` qui renvoie u_n, n où n est le plus petit entier tel que $|u_{n-1} - u_n| \leqslant p$ en suivant la méthode de Newton, avec pour premier terme `u0`. Le nombre $f'(x)$ sera approché par `derivee(f,x)`.
3. Appeler cette fonction pour approcher une solution de l'équation $x \ln x - 2 = 0$ en choisissant $u0 = 3$ et $p = 10^{-10}$.

4) - Méthodes numériques de calcul d'intégrales

On divise le segment $[a, b]$ en n sous-intervalles définis par une suite de points $(x_k)_{k\in[\![0,n]\!]}$ avec $x_k = a + k\frac{b-a}{n}$.
Au-dessus du segment $[x_k, x_{k+1}]$ on définit le rectangle gauche de hauteur $f(x_k)$ et le rectangle droit de hauteur $f(x_{k+1})$.

La méthode des rectangles à gauche consiste à approcher $\int_a^b f(x)\,\mathrm{d}x$ par la somme des aires des rectangles gauches pour k variant de 0 à $n-1$ c'est-à-dire $\sum_{k=0}^{n-1}(x_{k+1} - x_k)f(x_k) = \boxed{\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a + k\frac{b-a}{n}\right)}$. La méthode des rectangles à droite procède de la même façon avec les rectangles droits et donne l'approximation $\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a + (k+1)\frac{b-a}{n}\right)$.

Méthode des trapèzes : $\int_a^b f(x)\,\mathrm{d}x$ est approchée par la somme des moyennes des aires des rectangles gauches et droits.

EXERCICE 6 Approximations d'une intégrale par les méthodes des rectangles et des trapèzes

1. Écrire une fonction `rectangles(f,a,b,n)` qui renvoie une approximation de $\int_a^b f(x)\,\mathrm{d}x$ par la méthode des rectangles (à droite ou à gauche) avec une subdivision régulière en n intervalles.
2. Coder une fonction `trapèzes` de mêmes paramètres utilisant la méthode des trapèzes.
3. Appeler ces deux fonctions pour approcher $\int_0^1 \sqrt{1 - x^2}\,\mathrm{d}x$ avec $n = 1000$ puis en tirer une conclusion sur la meilleure méthode d'approximation d'intégrale sachant que $\int_0^1 \sqrt{1 - x^2}\,\mathrm{d}x = \frac{\pi}{4}$.

5) - Méthode de Heun

Soit (E) l'équation différentielle $y' = F(t, y)$.
La méthode de Heun est une amélioration de l'algorithme d'Euler qui consiste à remplacer la pente en t_i par la moyenne de la pente en t_i et de la pente en t_{i+1} si on avait appliqué la méthode d'Euler où $t_{i+1} = t_i + h$.

La méthode de Heun aboutit à la relation : $y_{i+1} = y_i + h\frac{F\big(t_i,y_i\big)+F\big(t_i+h,y_i+hF(t_i,y_i)\big)}{2}$.

EXERCICE 7 Méthode de Heun appliquée à l'équation différentielle non linéaire du modèle logistique

On considère l'équation différentielle non linéaire $(E) : y'(t) = ry \times (1 - y)$ où t est la variable correspondant au temps, r le taux de croissance maximum et y le rapport entre la taille de la population et la taille maximale supportée par le milieu.
On remarque que la variable t n'apparait pas dans l'équation. De telles équations différentielles sont appelées autonomes.

Ce modèle inventé par le mathématicien belge Pierre François Verhulst corrige le modèle de Malthus régi par l'équation $y' = ry$ par un facteur $1 - y$ proportionnel à l'écart entre la taille maximale supportée par le milieu et la taille de la population. Le modèle logistique est donc régulé par les capacités de l'environnement à subvenir aux besoins de la population.

Soit y la solution de (E) sur $[0, b]$, prenant la valeur y_0 au temps 0. La courbe de y est appelée courbe logistique.

On note $G(y, r) = ry(1 - y)$ et h un réel positif "assez petit" correspondant au pas de l'algorithme.

1. Écrire une fonction Python `G` de paramètres y et r qui renvoie la valeur de $G(y, r)$.
2. On construit par récurrence la suite (y_i) en suivant la méthode de Heun : $y_{i+1} = y_i + h\frac{G\big(y_i,r\big)+G\big(y_i+hG(y_i,r),r\big)}{2}$.
   Écrire une fonction `heun` qui renvoie, à partir des paramètres y_0, h, b, r les deux listes $[t_i]$ et $[y_i]$.
3. Écrire une fonction python `courbeH` d'arguments `y0, h, b, r` qui trace la courbe logistique obtenue par la méthode de Heun. On choisit $h = 0.01, b = 7$. Appeler `courbeH` pour les couples (r, y_0) suivants :
   $(1, 2), (5, 2), (15, 2), (1, 0.2), (5, 0.2), (15, 0.2)$.
   Procéder de la même façon en prenant $h = 0.1$. Que remarque-t-on ?
4. (Facultatif) Comment se comportent les courbes logistiques lorsque r augmente ?