Analyse et méthodes numériques

Outils Mathématiques pour la notion de complexité d'un algorithme

1 Motivations

Liste des algorithmes abordés cette année en mathématiques:
En mathématiques discrètes: Algorithme de Quine, Algorithme des carrés, RSA.
En algèbre linéaire: Algorithme du Pivot de Gauss, Calcul d'une puissance de matrice.
En théorie des graphes: Bellman, Dijkstra, Kruskal.

Question: Quelle est l'efficacité de ces algorithmes? Peut-on comparer les algorithmes à buts identiques entre eux? Comment varie l'efficacité suivant la taille de l'objet à traiter? (appeler parfois l'instance)
Il y a une nécessité d'inventer une mesure, indépendante des performances matérielles (rapidité du processeur, compilateurs, etc.), qui renseigne de manière "absolue" la "consommation" en instructions ou en ressource mémoire de l'algorithme : C'est ce qu'on appellera la complexité d'un algorithme, nombre dépendant de n où n désigne un paramètre d'entrée, le plus souvent la taille des données à traiter. (longueur d'une liste, nombre de sommets dans un graphe, etc.)

ATTENTION: Ne pas confondre "Théorie de la complexité", qui évalue la difficulté intrinsèque d'un problème posé indépendamment du type de solution choisie, avec la "complexité d'un algorithmique", qui mesure l'efficacité d'un algorithme particulier.
Nous n'aborderons ici que la notion de complexité d'un algorithme.

Diverses approches sont possibles pour évaluer la complexité d'un algorithme: Calcul dans le pire des cas, meilleur des cas ou cas moyen. (exemple pour la recherche d'un élément dans une liste ordonnée). L'approche la plus classique étant le calcul dans le pire des cas.

Nous ne calculerons la complexité d'un algorithme que dans des cas très simples (recherche d'un élément dans une liste par lecture simple, par dichotomie).

L'objet de ce cours est d'avoir des outils mathématiques pour pouvoir comparer des algorithmes au vue de leur complexité, il s'agit donc d'avoir une bonne notion des comportements asymptotiques et des croissances comparées de suites remarquables.

Extrait du cours M3103 d'Algorithmique Avancée (semestre 3):

			Taille des données		
Famille	**Notation**	**Exemples**	$n = 10^2$	$n = 10^3$	$n = 10^6$
Algo. constants	$\Theta(1)$	Échange 2 valeurs	10 ns	10 ns	10 ns
Algo. logarithmiques	$\Theta(\log n)$	Rech. dichotomique	25 ns	30 ns	60 ns
Algo. linéaires	$\Theta(n)$	Rech. séquentielle	1 μs	10 μs	10 ms
Algo. quasi-linéaires	$\Theta(n \log n)$	Tri par fusion	1 μs	10 μs	10^4 μs
Algo. quadratiques	$\Theta(n^2)$	Tri par sélection	100 μs	10 ms	2,8 h
Algo. cubiques	$\Theta(n^3)$	Produit 2 matrices	75 ms	10 s	16 ans
Algo. polynomiaux	$\Theta(n^p)$				
Algo. exponentiels	$\Theta(a^n)$	Suite Fibonacci			
Algo. factoriels	$\Theta(n!)$	Voyageur commerce	10^{80} ans	...	...

Temps d'exécution

Remarque:
Vous avez vu au collège la notion d'ordre de grandeur d'un nombre: c'est le terme 10^p dans sa notation scientifique $d \times 10^p$ où $|d| < 10$.
La complexité algorithmique est une sorte d'analogue de l'ordre de grandeur, mais qui renseigne sur la **croissance** du nombre d'opérations en fonction de la taille de l'instance. Ce nombre tendra toujours vers l'infini avec n grand, mais la question est de savoir à quelle vitesse, d'où l'étude des croissances comparées de suites.

2 Croissances comparées

On considère, dans toute la suite, n un nombre entier positif.

2.1 Rappels d'analyse

- Tout polynôme en n se comporte à l'infini comme son terme de plus haut degré.
  Exemple:
  $\lim\limits_{n\to\infty} n^5 - 6n^3 + n^2 - 5n + 3 = \lim\limits_{n\to\infty} n^5(1 - \frac{6}{n^2} + \frac{1}{n^3} - \frac{5}{n^4} + \frac{3}{n^5}) = \lim\limits_{n\to\infty} n^5 = +\infty$

- Rappels sur les **suites arithmétiques**:

  Définition par récurrence: $u_{n+1} = u_n + r$ avec $r \in \mathbb{R}$
  Terme général: $u_n = u_0 + nr$
  $\lim\limits_{n\to\infty} u_n \begin{cases} = & -\infty \quad si\, r < 0 \\ = & +\infty \quad si\, r > 0 \end{cases}$

- Rappels sur les **suites géométriques**:

  Définition par récurrence: $u_{n+1} = u_n \times q$ avec $q \in \mathbb{R}$
  Terme général: $u_n = q^n \times u_0$
  $\lim\limits_{n\to\infty} u_n \begin{cases} = & 0 \quad si|q| < 1 \\ = & +\infty \quad si\, q > 1 \\ n'existe\ pas & si\, q \leqslant -1 \end{cases}$

- **Notation exponentielle:**
  Pour tout $a > 0$ et $b \in \mathbb{R}$ on a $\boxed{a^b = e^{b\ln(a)}}$
  En particulier pour $n \in \mathbb{N}$ et $\alpha \in \mathbb{R}$, on a $\boxed{n^\alpha = e^{\alpha \ln(n)}}$

- Toutes les suites suivantes divergent vers l'infini:

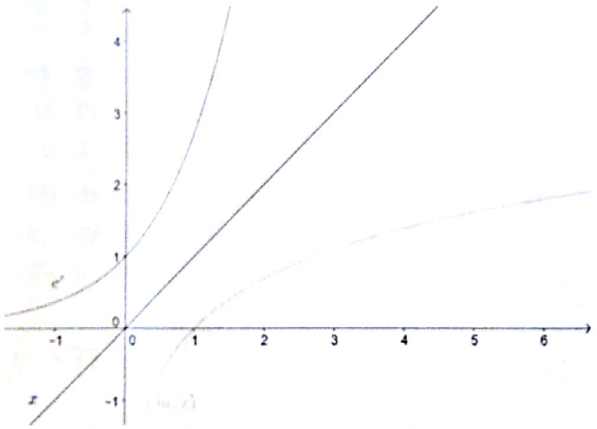
$$n, \quad n^\alpha\,(\alpha \in \mathbb{R}^+), \quad \ln(n), \quad n\ln(n), \quad e^n, \quad 2^n, \quad q^n\,(q > 1), \quad n!, \quad e^{n^2\ln(n)}, \ldots$$

La question est de savoir à quelle vitesse ces suites tendent vers l'infini, et de pouvoir comparer leurs vitesses de croissance les unes par rapport aux autres.

2.2 Croissances comparées

Vous avez vu au lycée:

$$\boxed{\lim_{x\to+\infty} \frac{e^x}{x} = +\infty \qquad \lim_{x\to+\infty} \frac{\ln(x)}{x} = 0 \qquad \lim_{x\to 0} x\ln(x) = 0}$$



On conjecture aisément qu'à l'infini les quotients $\frac{\ln(x)}{x}$ et $\frac{x}{e^x}$ tendent vers zéro.

De manière générale on a

$$\lim_{x\to+\infty} \frac{e^{ax}}{x^b} = +\infty \qquad \lim_{x\to+\infty} \frac{(\ln(x))^a}{x^b} = 0 \qquad \forall a, b > 0$$

Ainsi qu'en $-\infty$ on a:

$$\lim_{x\to-\infty} e^{ax}x^b = 0 \qquad \forall a, b > 0$$

et en zéro:

$$\lim_{x\to 0} |\ln(x)|^a x^b = 0 \qquad \forall a, b > 0$$

En conclusion, on retiendra pour les infiniment grands:

$$\lim_{n\to+\infty} \frac{e^{an}}{n^b} = +\infty \qquad \lim_{n\to+\infty} \frac{(\ln(n))^a}{n^b} = 0 \qquad \forall a, b > 0$$

On a aussi, pour les infiniment petits:

$$\lim_{x\to-\infty} e^{ax}x^b = 0 \qquad \lim_{x\to 0}(\ln x)^a x^b = 0 \qquad \forall a, b > 0$$

Visualisons le comportement des fonctions $x \mapsto x^{18}$, $x \mapsto 2^x$, $x \mapsto e^x$ et $x \mapsto x!$.

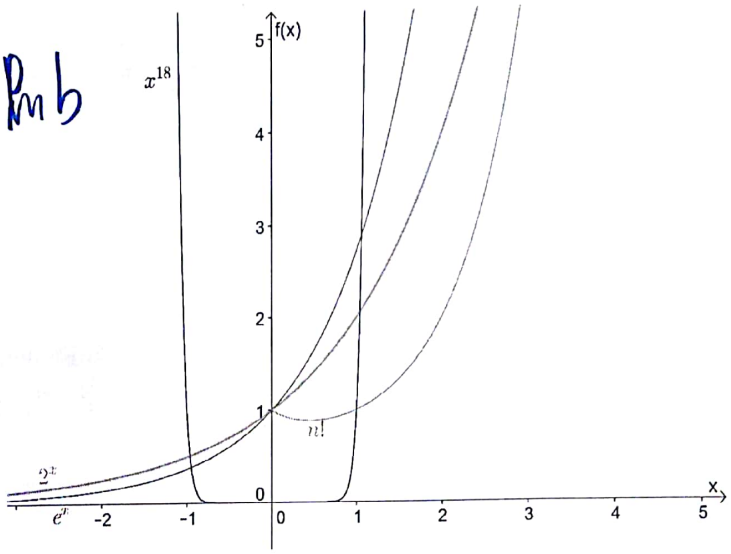
(Remarque: La fonction $f : x \mapsto x!$ pour $x \in \mathbb{R}$ est définie par $f(x) = \Gamma(x+1)$ où Γ est la fonction Γ d'Euler (vérifiant en particulier $\Gamma(x+1) = x\Gamma(x)$ pour $x \notin \mathbb{N}^-$), fonction que nous n'aborderons pas ici. Notons simplement que la restriction de f à $\mathbb{N}$ correspond à la définition de $n!$ connue.)

$\ln(e) = 1$

$\ln(a \times b) = \ln a + \ln b$

$e^0 = 1$

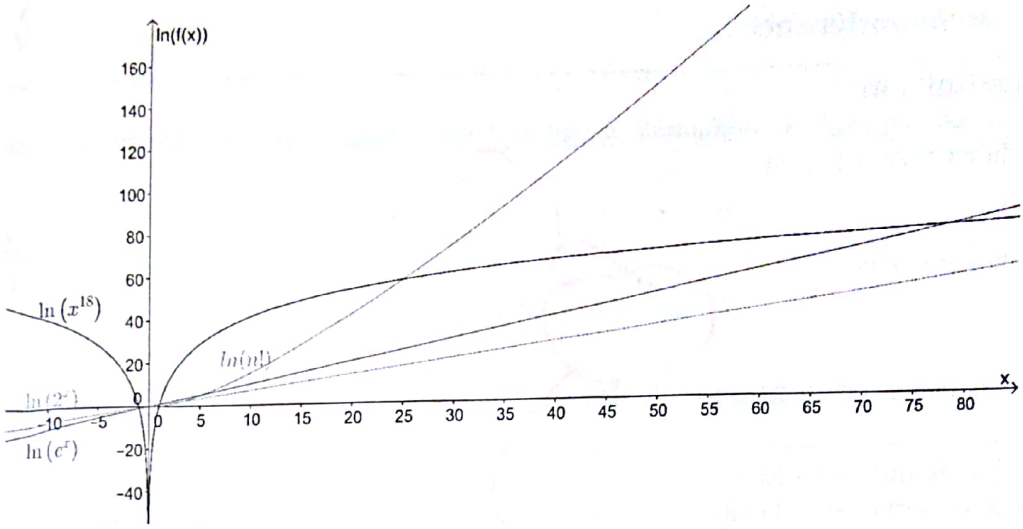
$e^{a+b} = e^a \times e^b$



On voit sur ce graphique qu'il est plus difficile de conjecturer les comportements asymptotiques relatifs de ces fonctions.

Un changement *linéaire* d'échelle (c'est-à-dire du type "une unité en abscisse pour 500 unités en ordonnées") ne suffit pas à "voir" ce qu'il se passe à l'infini.
Dans ce genre de cas assez courant en sciences (intensité sonore, définition des tons en musique, pH en chimie, taille mémoire en informatique, distance en astronomie...), on utilise une échelle logarithmique ou semi-logarithmique, qui permet de visualiser les ordres de grandeurs (puissance d'un nombre) plutôt que les grandeurs elles-mêmes.

Ainsi en reprenant le graphique précédent, en gardant en abscisse une progression linéaire, et en prenant en ordonnée une progression logarithmique (échelle dite semi-logarithmique), c'est à dire en traçant les fonctions $x \mapsto ln(f(x))$ plutôt que les fonctions $x \mapsto f(x)$, on obtient:



Dans ce type de graphique, les fonctions de type exponentielle ($x \mapsto e^{ax}$ mais aussi toutes les fonctions du type $x \mapsto a^x$) apparaissent donc comme des fonctions linéaires.

Remarque:
On constate alors qu'asymptotiquement, c'est la croissance de la fonction $x \mapsto 2^x$ qui l'emporte sur la croissance de $x \mapsto x^{18}$.

Notons qu'ici nous avons utilisé le logarithme Néperien ln, mais si on s'intéresse à l'ordre de grandeur en base 10 (par exemple pour des échelles de distance, de fréquence ou de prix), on pourra utiliser le logarithme décimal $\log_{10}$ ($\log_{10}(x) = \frac{\ln(x)}{\ln(10)}$), pour l'ordre de grandeur en base 2 (taille mémoire en informatique) on pourra utiliser le logarithme en base 2 ($\log_2(x) = \frac{\ln(x)}{\ln(2)}$), etc. Les comportements seront les mêmes qu'en prenant le logarithme Néperien classique, à un facteur multiplicatif près.

$(\ln n)^a \ll n^b \ll e^{cn} \ll n!$

3 Prépondérance, Domination, Equivalence, Notations de Landau

Dans toute la suite on considère deux suites (u_n) et (v_n) **ne s'annulant pas à partir d'un certain rang.**
Toutes les définitions suivantes sont également valables pour des fonction f et g avec les mêmes propriétés.

3.1 Prépondérance

Définition:

On dit que (u_n) est *négligeable* à l'infini devant (v_n), ou que (v_n) est prépondérante à l'infini devant (u_n) si

$$\lim_{n\longrightarrow+\infty} \frac{u_n}{v_n} = 0$$

On note alors

$$u_n \underset{+\infty}{=} o(v_n) \qquad ou \qquad u_n << v_n$$

(à la longue on omettra la précision $+\infty$)

On constate que, dans les notations de Landau, le signe "=" n'est plus transitif. Cependant, nous nous permettrons l'emploi du signe "=" entre deux "o".

Exemple: Vues les croissances comparées, on a

$n^a = o(e^{bn})$, $(\ln n)^a = o(n^b)$ et $e^n = o(n!)$ pour tout $a, b > 0$, c'est-à-dire

$$(\ln n)^a << n^b << e^{cn} << n! \qquad \forall a, b, c > 0$$

Règles de calcul avec o:

On peut démontrer facilement que

$o(u_n) + o(u_n) = o(u_n)$
$k \times o(u_n) = o(u_n)$ si $k \in \mathbb{R}$
$o(u_n) \times o(u_n) = o(u_n^2)$

3.2 Domination

Définition:

On dit que (u_n) est *dominée* à l'infini par (v_n), si la suite $\left(\frac{u_n}{v_n}\right)$ est bornée à partir d'un certain rang, c'est-à-dire si il existe $C \in \mathbb{R}$ et $N \in \mathbb{N}$ tel que pour tout $n > N$ on ait $|\frac{u_n}{v_n}| \leqslant C$. On note alors

$$u_n \underset{+\infty}{=} O(v_n) \qquad ou \qquad u_n \underset{+\infty}{\prec} v_n$$

(se lit "grand o")

Remarque: Il suffit que $\frac{u_n}{v_n}$ tende vers une limite finie pour que $u_n = O(v_n)$, mais ce n'est pas une condition nécessaire.

Exemple:

1. On a par exemple

$$n^2\ln(n) + n^2 - 2 = O(n^2\ln(n))$$

car $\left|\frac{n^2\ln(n)+n^2-2}{n^2\ln(n)}\right| = 1 + \frac{1}{\ln(n)} - \frac{2}{n^2\ln(n)} \longrightarrow 1$

2. Mais on a aussi:

$$(-1)^n \ln n + \frac{1}{n} = O(\ln n)$$

car $\left|\frac{(-1)^n \ln n + \frac{1}{n}}{\ln n}\right| \leqslant 1 + \left|\frac{1}{n \ln n}\right| \leqslant 1 + \frac{1}{2 \ln 2}$ à partir de $n = 2$.

Règles de calcul avec O:
On peut démontrer facilement que

$O(u_n) + O(u_n) = O(u_n)$
$k \times O(u_n) = O(u_n)$ si $k \in \mathbb{R}$
Si $u_n = o(v_n)$, alors $u_n = O(v_n)$

3.3 Equivalence

Définition:
On dit que (u_n) est *équivalente* à (v_n) en l'infini, si la suite $\left(\frac{u_n}{v_n}\right)$ tend vers 1.
On note alors

$$u_n \underset{+\infty}{\sim} v_n$$

Exemples et remarques:

1. On a par exemple

$$\ln(n) + n^2 - 2 \sim n^2$$

car $\frac{\ln(n)+n^2-2}{n^2} = \frac{\ln n}{n^2} + 1 - \frac{2}{n^2} \longrightarrow 1$

2. Attention "$\lim u_n = \lim v_n \Rightarrow u_n \sim v_n$" n'est valable que si la limite est **finie et non nulle.**

3. La relation d'"équivalence" est une relation d'équivalence (et ce n'est pas une blague) sur l'ensemble des suites non nulles à partir d'un certain rang.

4. La relation d'équivalence entre deux suites est compatible avec les opérations de multiplication par une autre fonction ou un réel, d'inversion, de division. En revanche, l'addition et la composition posent problème.

Règles de calcul avec $\sim$:
On peut démontrer facilement que

Si $u \underset{a}{\sim} v$ et $w \underset{a}{\sim} t$ alors $uw \underset{a}{\sim} vt$
Si $u \underset{a}{\sim} v$ avec u et v ne s'annulant pas, alors $\frac{1}{u} \underset{a}{\sim} \frac{1}{v}$
Si $u = v + w$ avec $w = o(v)$ alors $u \underset{a}{\sim} v$
Si $u_n \underset{a}{\sim} v_n$, alors $u_n \underset{a}{=} O(v_n)$

Attention!

- $u \sim v \not\Rightarrow e^u \sim e^v$ sauf si $\lim(u - v) = 0$
  En effet par exemple $n \sim n + 1$ mais $\frac{e^{n+1}}{e^n} = e \not\to 1$.

3.4 Notation Θ pour la complexité d'un algorithme

Définition:
$u_n = \Theta(v_n)$ signifie que $u_n = O(v_n)$ et $v_n = O(u_n)$, c'est à dire qu'à partir d'un certain rang on a $A \leqslant |\frac{u_n}{v_n}| \leqslant B$ avec $A > 0$ et $B > 0$.

$u_n = \Theta(v_n)$ signifie que u_n et v_n ont le même type de croissance à l'infini.
C'est ce type de relation que l'on cherche pour décrire la complexité d'un algorithme.

Remarques:
Dans la littérature informatique, on confond souvent $O(u_n)$ et $\Theta(u_n)$. Attention à bien comprendre que $O(u_n)$ ne fourni qu'une *majoration* du comportement à l'infini alors que $\Theta(u_n)$ fourni un *ordre de grandeur* de ce comportement à l'infini.

Règles de calcul avec Θ:
On peut démontrer facilement que

$\Theta(u_n) + \Theta(u_n) = \Theta(u_n)$
$k \times \Theta(u_n) = \Theta(u_n)$ si $k \in \mathbb{R}$
Si $u_n = \Theta(v_n)$ alors $v_n = \Theta(u_n)$
Si $u_n \sim v_n$, alors $u_n = \Theta(v_n)$

Insistons sur le fait que si (u_n) est équivalente à (v_n), alors $u_n = \Theta(v_n)$, mais cette condition n'est pas nécessaire.
Dans la pratique on cherchera donc d'abord à montrer l'équivalence des suites et, lorsque cette méthode est infructueuse, on reviendra à la définition.

Exemples:

1. On a vu que $\ln(n) + n^2 - 2 \sim n^2$ donc $\ln(n) + n^2 - 2 = \Theta(n^2)$.
2. En revanche $\ln(n) + 3n^2 - 2$ n'est pas équivalent à n^2 car $\frac{\ln(n)+3n^2-2}{n^2} \longrightarrow 3$. Cependant si $\frac{\ln(n)+3n^2-2}{n^2} \longrightarrow 3$ alors pour à partir d'un certain rang $2 \leqslant |\frac{\ln(n)+3n^2-2}{n^2}| \leqslant 4$. et alors $\ln(n) + 3n^2 - 2 = \Theta(n^2)$.

4 Des outils pour étudier les limites de suites

4.1 Rappels et outils classiques

- Les formes indéterminées pour les études de limites sont du type $+\infty - \infty$ $0 \times \infty$, $\frac{0}{0}$ et $\frac{\infty}{\infty}$ (abus de notations!)
- Si f est une fonction continue sur $\mathbb{R}$, et si $u_n \underset{+\infty}{\longrightarrow} l$, alors $f(u_n) \underset{+\infty}{\longrightarrow} f(l)$
- Si $u_n \leqslant v_n$ à partir d'un certain rang et si $u_n \longrightarrow +\infty$ alors $v_n \longrightarrow +\infty$
- Si $|u_n| \leqslant v_n$ à partir d'un certain rang et si $v_n \longrightarrow 0$ alors $u_n \longrightarrow 0$

- Dans le cas d'une somme dont la limite *à l'infini* est une forme indéterminée, on pourra penser à factoriser par le terme prépondérant.
- Ne pas oublier la notation exponentielle dans les considération de puissances.

4.2 Exemples d'études de suites

1. Etudier la limite en $+\infty$ de $u_n = \frac{n^3-2n^2+3}{n \ln n}$.

$$\lim_{n\to+\infty} \frac{n^3-2n^2+3}{n\ln n} = \lim_{n\to+\infty} \frac{n^3}{n\ln n} = \lim_{n\to+\infty} \frac{n^2}{\ln n} = +\infty$$

2. Etudier la limite de la suite $u_n = n^3(\ln n)^2 - n^2(\ln n)^3$. Trouver un équivalent de u_n en $+\infty$.

$$\lim_{n\to+\infty} u_n = \lim_{n\to+\infty} n^2\left(n(\ln n)^2 - (\ln n)^3\right)$$
$$= \lim_{n\to+\infty} n^2\left(n - (\ln n)^3\right)$$
$$= \lim_{n\to+\infty} n^2(n)$$
$$= +\infty$$

5 Complexité de l'algorithme de Dichotomie

Rappel de l'algorithme:

Recherche d'une valeur α dans un intervalle $[a, b]$, similaire à la recherche d'une valeur dans une liste de longueur $b - a$.
Deux suites (u_k) et (v_k) sont définies comme suit:

- $u_0 = a, \quad v_0 = b$
- Si $\alpha \in [u_k, \frac{u_k+v_k}{2}]$, alors $u_{k+1} = u_k$ et $v_{k+1} = \frac{u_k+v_k}{2}$
- Si $\alpha \notin [u_k, \frac{u_k+v_k}{2}]$, alors $u_{k+1} = \frac{u_k+v_k}{2}$ et $v_{k+1} = v_k$

On itère le processus jusqu'à ce que la différence $|v_k - u_k|$ soit inférieure à la précision ε demandée. (dans le cas de la recherche dans une liste, on prendra par exemple $\epsilon = 1$) N'importe quelle valeur de l'intervalle $[u_k, v_k]$ fournit alors une approximation à ε près de la solution α recherchée.

Pour une longueur d'intervalle (ou longueur de liste) $b - a = n$, quelle est la complexité de l'algorithme de dichotomie dans le pire des cas? (Donner la réponse sous la forme $\Theta(g(n))$).

$\Theta(\log n)$

Le calcul à proprement parler de la complexité n'est pas toujours aisé. Ici une vidéo donnant quelques exemples et résultats abordables: `https://www.youtube.com/watch?v=c1Z4q5zPB1E` par Rachid Gerraoui de l'Ecole Polytechnique Fédérale de Lausanne.

6 Exercices

Exercice 1 On considère le logarithme en base 2 défini pour tout $x > 0$ par

$$\log_2(x) = \frac{\ln x}{\ln 2}$$

.

Cette fonction est très connue et utilisée en informatique et en théorie de l'information notamment.

1. Etude de la fonction $\log_2$:
   (a) Pour quelle valeur de x a-t-on $\log_2(x) = 0$?
   (b) Pour quelle valeur de x a-t-on $\log_2(x) = 1$?
   (c) Que vaut $\log_2(x)$ pour $x = 2^n$? (n étant un entier positif)
   (d) On pose $y = \log_2(x)$. Exprimer x en fonction de y.
   (e) Quelle est la limite de $\log_2$ à l'infini?
   (f) Tracer, dans un même repère, l'allure de la courbe de la fonction $x \mapsto \log_2(x)$ ainsi que celle de $x \mapsto \ln x$.

2. Comparaisons asymptotiques:
   Dans la suite de l'exercice, on considère n un entier positif.
   (a) A-t-on $\log_2(n) = o(\ln n)$?
   (b) A-t-on $\log_2(n) = O(\ln n)$?
   (c) A-t-on $\log_2(n) \sim (\ln n)$?
   (d) A-t-on $\log_2(n) = \Theta(\ln n)$?

3. Plus généralement, on défini le logarithme en base a ($a > 0$) par

$$\log_a(x) = \frac{\ln x}{\ln a}$$

   (a) Qui est la fonction $\log_e$?
   (b) De la même manière qu'en question 1., étudiez la fonction $\log_{10}$.
   (c) Comparer les comportements à l'infini de $\log_2$, ln et $\log_{10}$.

(1+1)+(1+2+2+2)+(1+3+2+3+33)

2 + 7 + 15

24

Exercice 2 Etudier la limite, quand elle existe, de chacune des suites suivantes:

1. $u_n = \frac{n^2-3n+5}{n^3 \ln n}$
2. $u_n = \frac{n^6}{2^n}$
3. $u_n = (-1)^n \ln(n)$
4. $u_n = \left(-\frac{1}{2}\right)^n$
5. $u_n = \left(\frac{6}{5}\right)^n$
6. $u_n = \frac{(\ln(n))^3}{n^2}$
7. $u_n = \frac{n^2 \ln(n)}{2^n}$
8. $u_n = 2^n - n!$
9. $u_n = \frac{n^2 - n\ln(n)+1}{(n+1)^2}$
10. $u_n = \frac{e^n}{3^n}$
11. $u_n = \frac{e^n}{2^n}$
12. $u_n = e^{\left(\frac{1}{n}\right)\ln\left(\frac{1}{n}\right)}$
13. $u_n = \frac{(n+1)^2}{n^n}$
14. $u_n = \frac{n^{127}}{n!}$
15. $u_n = \frac{3^n+5^n}{\ln(n)+6^n}$
16. $u_n = \frac{\left(\left(\frac{6}{5}\right)^n - \left(\frac{13}{10}\right)^n\right)}{\ln n}$
17. $u_n = \sqrt{\frac{n^{627}+1}{n^{626}+7}}$
18. $u_n = e^n - n!$
19. $u_n = \frac{e^n}{n^n}$
20. $u_n = \frac{n^2 e^n}{8^n}$
21. *(facultatif)* $u_n = \left(\frac{3n+1}{5n+5}\right)^n$
22. *(facultatif)* $u_n = (n+1)^{\frac{1}{n}} - (n)^{\frac{1}{n}}$
23. *(facultatif)* $u_n = \frac{1}{n^{\cos\left(\frac{1}{n}\right)}}$
24. *(facultatif)* $u_n = n \times \ln(1+\frac{1}{n})$
25. *(facultatif)* $u_n = \frac{\ln(1+\frac{1}{n})}{n}$

Exercice 3

1. On définit deux suites réelles (u_n) et (v_n) par les relations : $\begin{cases} u_{n+1} = 5u_n + 2v_n \\ v_{n+1} = -\frac{28}{3}u_n - \frac{11}{3}v_n \end{cases}$ où n désigne un entier non nul et où u_0 et v_0 sont deux réels. On note: $X_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$.

   On considère la matrice $A = \begin{pmatrix} 5 & 2 \\ -\frac{28}{3} & -\frac{11}{3} \end{pmatrix}$.

   On admet que A est diagonalisable et que $A = P \times D \times P^{-1}$ est une diagonalisation de A avec $D = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{3} \end{pmatrix}$; $P = \begin{pmatrix} 1 & -3 \\ -2 & 7 \end{pmatrix}$ et $P^{-1} = \begin{pmatrix} 7 & 3 \\ 2 & 1 \end{pmatrix}$.

   (a) Vérifier la relation $X_{n+1} = AX_n$, puis montrer que $X_n = A^n X_0$.

   (b) Exprimer A^n en fonction de D^n, de P et de P^{-1}.

   (c) Calculer A^n : les coefficients de cette matrice seront exprimés en fonction de n

   (d) En déduire l'expression de u_n et v_n en fonction de u_0 et v_0.

   (e) En déduire les limites des suites (u_n) et (v_n) quand n tend vers $+\infty$.

2. De manière général, on définit deux suites réelles (u_n) et (v_n) par les relations : $\begin{cases} u_{n+1} = au_n + bv_n \\ v_{n+1} = cu_n + dv_n \end{cases}$

où n désigne un entier non nul, a, b, c, d, u_0 et v_0 sont des réels. On note: $X_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$.

On admet que la matrice $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ est diagonalisable.

A quelle condition sur A le vecteur X_n admet-il une limite finie quand n tend vers l'infini?

Exercice 4

1. Démontrer les affirmations suivantes:
   (a) $n^2(\ln n)^3 = o(n^3)$
   (b) $2^n \ln n + n^6 = O(e^n)$
   (c) Si $0 < a < b$, alors $a^n = o(b^n)$
2. Les affirmations suivantes sont-elles vraies ou fausses? Justifier.
   (a) En $+\infty$, $n^3 \ln n << n^4$
   (b) En $+\infty$, $3^n = o(e^n)$
   (c) En $+\infty$, $n^3(\ln n)^2 + n^3 - n = O(n^3(\ln n)^2)$
   (d) Un algorithme dont la complexité est $c(n) = n^3 + (\ln n)^3$ est à complexité polynômiale

MATHS TD1

Exercice 4

a) $n^2(\ln n)^3 = \mathcal{O}(n^3)$

$$\lim_{n\to+\infty} \frac{n^2(\ln n)^3}{n^3} = \lim_{n\to+\infty} \frac{n^3}{n^3}$$

$$= \lim_{n\to+\infty} \frac{1}{n}$$

$$= 0$$

donc on a bien $n^2(\ln n)^3 = \mathcal{O}(n^3)$

b) $2^n \ln n + n^6 = \mathcal{O}(e^n)$

$$\lim_{n\to+\infty} \frac{2^n \ln + n^6}{e^n} = \lim_{n\to+\infty} \frac{2^n}{e^n} \left(\left(\frac{2}{e}\right)^n\right)$$

$$= \lim_{n\to+\infty} \frac{e^{n\ln(2)}}{e^n} \quad \text{car } \frac{2}{e} < 1$$

$$= \lim_{n\to+\infty} \frac{e^n}{e^n}$$

$$= 0$$ On 0 est une limite finie

c)

c) Si $0 < a < b$ alors $a^n = o(b^n)$

$\lim_{n\to+\infty} \frac{a^n}{b^n} = \lim_{n\to+\infty} \left(\frac{a}{b}\right)^n$

$a < b$ donc $\frac{a}{b} < 1$

d'où $\lim_{n\to+\infty} \left(\frac{a}{b}\right)^n = 0$

2)

a) $\lim_{n\to+\infty} \frac{n^3 \ln n}{n^4} = \lim_{n\to+\infty} \frac{n^3}{n^4}$

$= \lim_{n\to+\infty} \frac{1}{n}$

$= 0$

donc on a : $n^3 \ln n = o(n^4)$

d'où $n^3 \ln n \ll n^4$

b)

$\lim_{n\to+\infty} \frac{n^3 (\ln n)^2 + n^3 - n}{n^3 (\ln)^2}$

$= \lim_{n\to+\infty} \frac{n\left(n^2 (\ln n)^2 + n^2 - 1\right)}{n^4 (\ln n^2)}$

$= \lim_{n\to+\infty} \frac{n^2 (\ln n)^2 + n^2 - 1}{n^2 (\ln)^2}$

$= \lim_{n\to+\infty} \frac{n^2 (\ln n)' + n^2}{n^2 (\ln)'}$

$= \lim_{n \to +\infty} \frac{(\ln n)^2}{(\ln n)^2}$

$= 1$ or 1 est une limite finie

donc on a bien :

$n^3 (\ln n)^2 + n^3 - n = O(n^3 (\ln)^2)$

b) $3^n = o(e^n)$

$\lim_{n \to +\infty} \frac{3^n}{e^n} = \lim_{n \to +\infty} \left(\frac{3}{e}\right)^n$ or $3 > e^1$ donc $\frac{3}{e} > 1$

donc o est faux d'où $\lim_{n \to} \left(\frac{3}{e}\right)^n = +\infty$

d) $\lim_{n \to +\infty} n^3 + (\ln n)^3 = \lim_{n \to +\infty} n^3$

or n^3 est un monome de degré 3. on a donc bien une complexité polynomiale.

Exercice 6

1) $T_1 : n^\alpha$ 2)

$T_2 : \ln n$

2) $\alpha_1 = \frac{3,25 - (-0,3)}{6,2 - 5,5} = \frac{3,}{1,1} = 3.$

$\alpha_2 = \frac{0,45 - (-2,5)}{} = \frac{2,95}{} \simeq$

b) on a :

$T_4(n) = \log(k \times n^3)$

$= \ln(k \times n^3)$

$\ln(T_4(n)) = a_1 \ln n + b$

$T_4(n) = e^{a_1 \ln n + b}$

$T_4(n) = n^{a_1} \times e^b$ idem $T_2(n)$

c)

$\lim_{n \to +\infty} \frac{k n^{a_1}}{n^{a_1}} = \lim_{n \to +\infty} \frac{n^{a_1}}{n^{a_1}} = 1$ on a bien une limite finie strictement positive,

donc a bien $T_4(n) = \Theta(n^{a_1})$ idem $T_2(n)$

d) $T_4(n)$ et $T_2(n)$ ont des complexité temporelle de $\Theta(n^4)$.

2) a) $n \ln(n) = o(n^\alpha)$

$\lim_{n \to +\infty} \frac{n \ln(n)}{n^\alpha} = \frac{n}{n^\alpha} = \frac{1}{n^{\alpha-1}} = 0$

donc on a bien $n \ln(n) = o(n^\alpha)$

c'est T_3 le plus efficace.

$= \lim_{n \to +\infty} \frac{(\ln n)^2}{(\ln n)^2}$

= 1 et 1 est une limite finie

donc on a bien :

$n^3 (\ln n)^2 + n^3 - n = O(n^3 (\ln)^2)$

b) $3^n = o(e^n)$

$\lim_{n \to +\infty} \frac{3^n}{e^n} = \lim_{n \to +\infty} \left(\frac{3}{e}\right)^n$ or $3 > e^1$ donc $\frac{3}{e} > 1$

donc o est faux d'où $\lim_{n \to} \left(\frac{3}{e}\right)^n = +\infty$

d) $\lim_{n \to +\infty} n^3 + (\ln n)^3 = \lim_{n \to +\infty} n^3$

or n^3 est un monome de degré 3. on a donc bien une complexité polynomiale.

Exercice 6

1) $T_1 : n^\alpha$ 2)

$T_2 : \ln n$

2) $\alpha_1 = \frac{3,25 - (-0,3)}{6,2 - 5,5} = \frac{3,}{1,1} = 3$

$\alpha_2 = \frac{0,45 - (-2,5)}{6,5 - 4,93} = \frac{2,95}{1,32} \simeq 2,23$

Exercice 5 Parmi les affirmations suivantes (qui sont toutes données au voisinage de $+\infty$), cochez celles qui sont vraies :

- ☐ $2n^2 = o(n^2)$
- ☒ $2n^2 = O(n^2)$
- ☒ $2n^2 = \Theta(n^2)$
- ☐ $2n^2 \sim n^2$

- ☒ $n^2 = o(n^3)$
- ☒ $n^2 = O(n^3)$
- ☐ $n^2 = \Theta(n^3)$
- ☐ $n^2 \sim n^3$

- ☐ $2n^2 - 3n + 1 = o(n^2)$
- ☒ $2n^2 - 3n + 1 = O(n^2)$
- ☒ $2n^2 - 3n + 1 = \Theta(n^2)$
- ☐ $2n^2 - 3n + 1 \sim n^2$

- ☒ $2n^2 - 3n + 1 = o(n^3)$
- ☒ $2n^2 - 3n + 1 = O(n^3)$
- ☐ $2n^2 - 3n + 1 = \Theta(n^3)$
- ☐ $2n^2 - 3n + 1 \sim n^3$

- ☐ $2n^2 - 3n + 1 = o(2n^2)$
- ☒ $2n^2 - 3n + 1 = O(2n^2)$
- ☒ $2n^2 - 3n + 1 = \Theta(2n^2)$
- ☒ $2n^2 - 3n + 1 \sim 2n^2$

- ☒ $2n^2 - 3n + 1 = o(2n^3)$
- ☒ $2n^2 - 3n + 1 = O(2n^3)$
- ☐ $2n^2 - 3n + 1 = \Theta(2n^3)$
- ☐ $2n^2 - 3n + 1 \sim 2n^3$

- ☒ $2n^3 - n\ln n + 2 = o(n^4)$
- ☒ $2n^3 - n\ln n + 2 = O(n^4)$
- ☐ $2n^3 - n\ln n + 2 = \Theta(n^4)$
- ☐ $2n^3 - n\ln n + 2 \sim n^4$

- ☐ $2n^3 - n\ln n + 2 = o(n^3)$
- ☒ $2n^3 - n\ln n + 2 = O(n^3)$
- ☒ $2n^3 - n\ln n + 2 = \Theta(n^3)$
- ☐ $2n^3 - n\ln n + 2 \sim n^3$

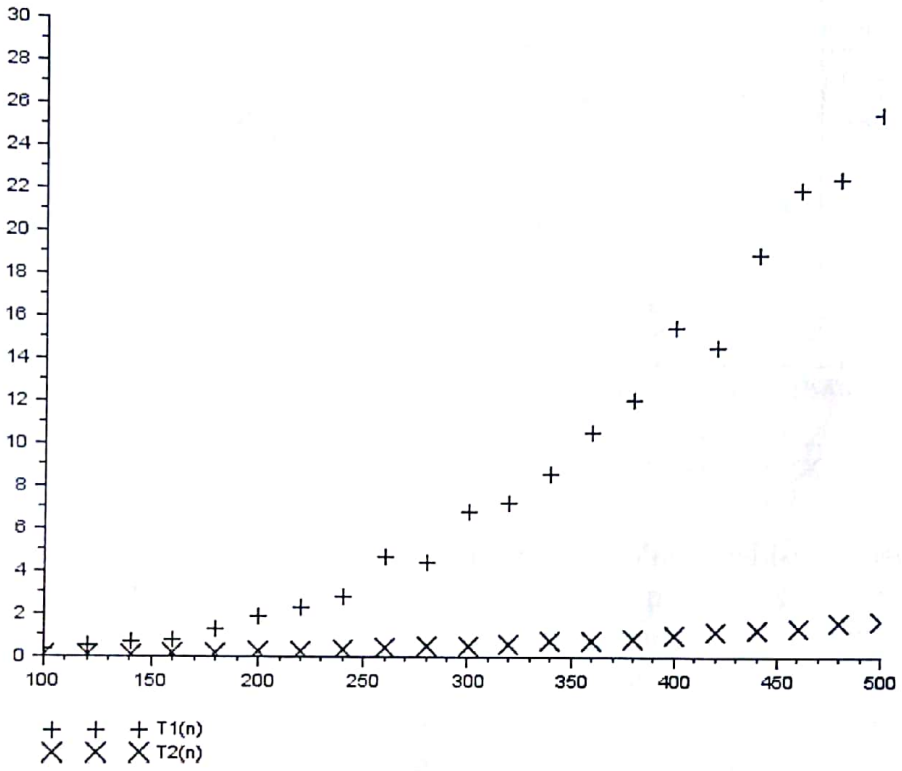
- ☒ $n^2 \ln n = o(n^3)$
- ☒ $n^2 \ln n = O(n^3)$
- ☐ $n^2 \ln n = \Theta(n^3)$
- ☐ $n^2 \ln n \sim n^3$

- ☐ $n^2 \ln n = o(n^2 \ln n)$
- ☒ $n^2 \ln n = O(n^2 \ln n)$
- ☒ $n^2 \ln n = \Theta(n^2 \ln n)$
- ☐ $n^2 \ln n \sim n^2 \ln n$

Exercice 6

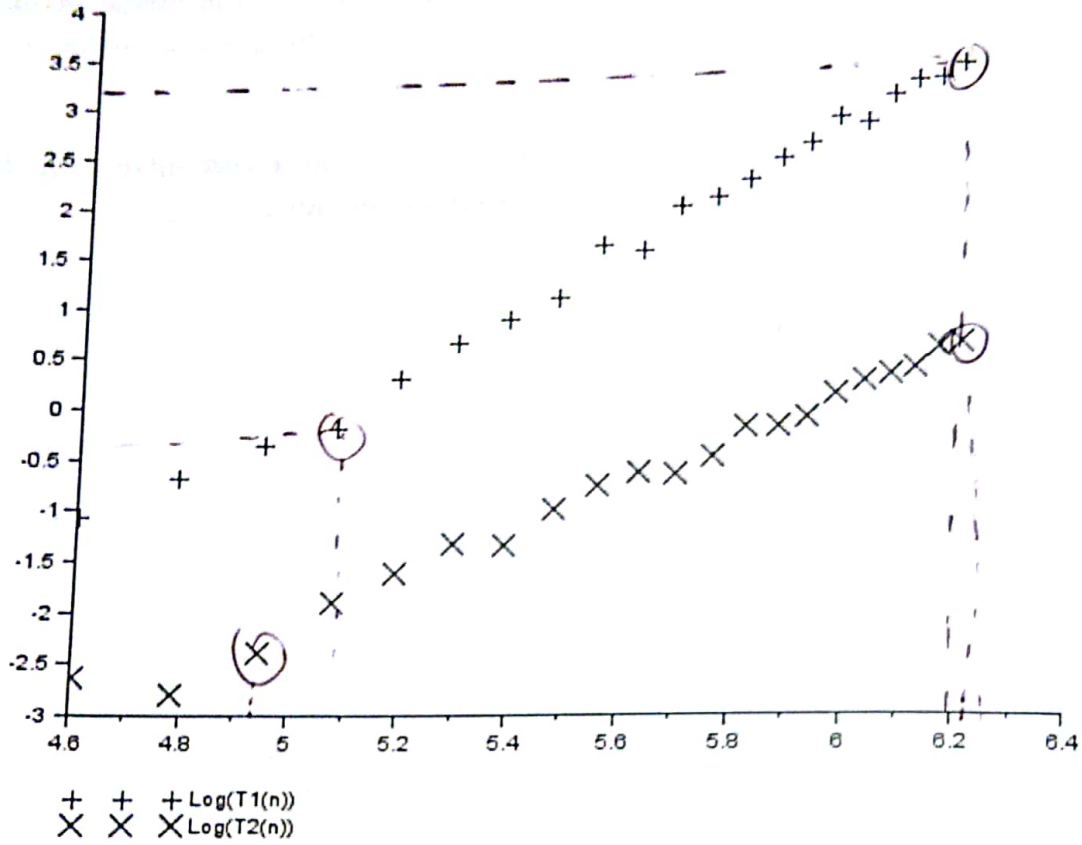
On considère trois algorithmes numérotés de 1 à 3 effectuant une même tâche sur une donnée de taille n. On s'intéresse au temps d'exécution $Ti(n)$ de chaque algorithme i en fonction de la taille n de l'argument d'entrée.

On effectue des simulations pour les algorithmes 1 et 2, avec des entrées aléatoires de taille n, pour n allant de 100 à 500. On obtient le graphique suivant:



1. Quel(s) type(s) de croissance asymptotique peut-on envisager pour les temps $T1(n)$ et $T2(n)$?

2. (a) Pour aller plus loin, on trace les courbes
$C_1 : \ln(n) \mapsto \ln(T1(n))$ et $C_2 : \ln(n) \mapsto \ln(T2(n))$ données ci-dessous.



On peut considérer qu'on obtient quasiment des droites.
En utilisant le graphique, estimez les valeurs approchées des pentes (ou coefficients directeurs) a_1 et a_2 de chacune des courbes C_1 et C_2. (on laissera en évidence les éventuels traits de construction sur le graphique et les éventuels calculs).
Indication: On devrait trouver $a_2 > 1$.

(b) En déduire que $T1(n) = K \times n^{a_1}$ et $T2(n) = L \times n^{a_2}$ où K et L sont des constantes positives. (On ne demande pas de déterminer K et L.)

(c) Montrer qu'alors $T1(n) = \Theta(n^{a_1})$ et $T2(n) = \Theta(n^{a_2})$.

(d) De quel type sont les complexités temporelles $T1(n)$ et $T2(n)$?

3. Une étude expérimentale du temps calcul du troisième algorithme a fourni $T3(n) = n\ln(n)$.

(a) Montrer que $n\ln(n) = o(n^{a_2})$.

(b) Quel est l'algorithme le plus efficace du point de vue de la complexité temporelle?

Exercice 7

On rappelle le principe de l'algorithme de dichotomie pour déterminer une valeur approchée de l'unique solution de $f(x) = 0$ dans l'intervalle $[a, b]$ avec $a < b$.
Deux suites (u_k) et (v_k) sont définies comme suit:

- $u_0 = a, \quad v_0 = b$
- Si $f(u_k) \times f(\frac{u_k+v_k}{2}) < 0$, alors $u_{k+1} = u_k$ et $v_{k+1} = \frac{u_k+v_k}{2}$
- Si $f(u_k) \times f(\frac{u_k+v_k}{2}) > 0$, alors $u_{k+1} = \frac{u_k+v_k}{2}$ et $v_{k+1} = v_k$

On itère le processus jusqu'à ce que la différence $|v_k - u_k|$ soit inférieure à la précision ε demandée. N'importe quelle valeur de l'intervalle $[u_k, v_k]$ fournit alors une valeur approchée à ε près de la solution α recherchée.

1. Montrer que pour tout $k \in \mathbb{N}$ $\quad |u_{k+1} - v_{k+1}| = \frac{1}{2}|u_k - v_k|$
2. En déduire que pour tout $k \in \mathbb{N}$ $\quad |u_k - v_k| = \frac{1}{2^k}(b - a)$
3. Soit $\varepsilon > 0$. Montrer que le nombre K d'itérations nécessaires pour que la longueur de l'intervalle $[u_k, v_k]$ soit inférieure à ε vérifie
$$K \geqslant \frac{\ln(b-a) - \ln\varepsilon}{\ln 2}$$
4. On s'intéresse à la complexité de l'algorithme de recherche d'un élément dans une liste de longueur n, basé sur la méthode par dichotomie:
Par analogie, on peut considérer que cela revient à effectuer une recherche du type décrit en préambule, dans un intervalle de longueur n, dans lequel chaque entier représente un élément de la liste donnée.
On considère que la précision ε est fixée (à 1 si on poursuit l'analogie). Si on considère que la valeur $(b - a)$ représente la taille n des données à traiter, que peut-on dire de la complexité (en terme de nombre d'instructions) de l'algorithme de dichotomie en fonction de n? On donnera la réponse sous la forme $\Theta(g(n))$ en précisant l'expression de $g(n)$.

Exercice 8

1. On considère que le temps d'exécution d'une instruction par un logiciel de calcul est proportionnel au nombre d'opérations élémentaires, autres que les additions, effectuées sur des nombres réels. On estime que les additions et les procédures d'affectation sont réalisées en un temps négligeable.
Combien d'opérations élémentaires, en négligeant les additions, sont-elles nécessaires pour effectuer le calcul du produit de deux matrices carrées de taille n?
2. En déduire, en fonction de n, la complexité $c(n)$ de l'algorithme de calcul direct (sans diagonalisation) de la puissance 25-ième d'une matrice carrée de taille n. Cette complexité est-elle de type polynômiale? Exponentielle?

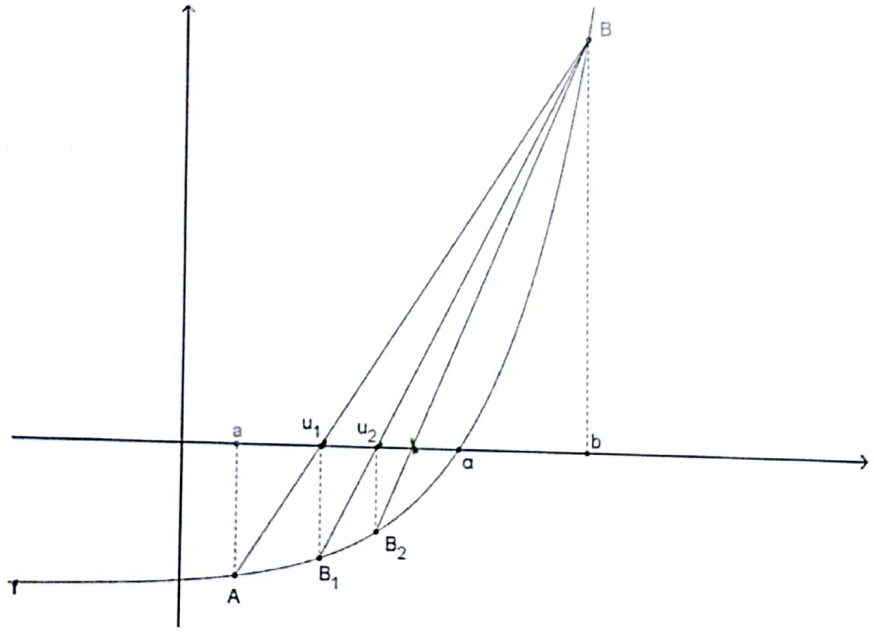
3. Comparer avec les conclusions tirées lors du TP-Scilab "Coût du calcul d'une puissance de matrice".

Exercice 9

On décrit ici la méthode de la sécante pour la résolution d'une équation du type $f(x) = 0$.

La solution α de l'équation $f(x) = 0$ ayant été localisée dans un intervalle $[a; b]$, l'idée de la méthode consiste à remplacer la fonction f par la fonction affine représentée graphiquement par la droite (AB) où A et B sont les points de coordonnées respectives $(a; f(a))$ et $(b, f(b))$.

Dans le cas de la figure ci-dessous, cette droite coupe l'axe des abscisses en un point d'abscisse u_1 avec $f(u_1) < 0$. On réitère en cherchant l'abscisse u_2 du point d'intersection de (BB_1) avec l'axe des abscisse, où B_1 est le point de coordonnées $(u_1; f(u_1))$, et ainsi de suite...



Dans le cas de la figure ci-dessous (fonction convexe, tournée vers le haut), avec $f(a) < 0$ et $f(b) > 0$, on construit une suite (u_n) définie par

$$u_0 = a, \quad \forall n \in \mathbb{N} \quad u_{n+1} = u_n - f(u_n)\frac{b - u_n}{f(b) - f(u_n)}$$

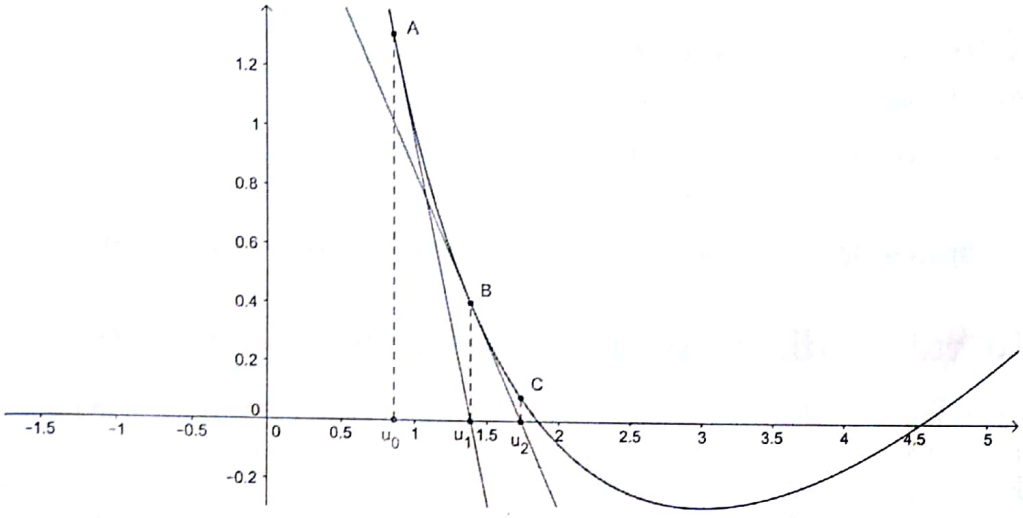
La suite (u_n) tend alors vers la solution de $f(x) = 0$ dans l'intervalle $[a, b]$.
On cherche à démontrer ces affirmations.

1. Pour $n \in \mathbb{N}$, déterminer une équation de la droite passant par les points de coordonnées $(b, f(b))$ et $(u_n, f(u_n))$.
2. En déduire l'expression de u_{n+1} en fonction de u_n.
3. En supposant que (u_n) tende vers $l \neq b$, montrer qu'alors $f(l) = 0$.

Exercice 10

On décrit ici la méthode de Newton pour la résolution d'une équation du type $f(x) = 0$.

La solution α de l'équation $f(x) = 0$ ayant été localisée dans un intervalle $[a; b]$ et une valeur initiale u_0 étant donnée, l'idée de la méthode consiste à remplacer la fonction f par sa tangente au point d'abscisse u_0. Cette tangente coupe l'axe des abscisses en u_1. On réitère le processus à partir de u_1, etc.
On construit ainsi une suite de nombre réels (u_n) qui, sous certaines conditions sur f, converge vers α.



On a alors

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)} \qquad (*)$$

et (u_n) converge vers α.

1. On souhaite prouver la relation $(*)$.
   (a) Donnez une équation de la tangente à la courbe de f au point d'abscisse a.
   (b) En déduire l'expression de u_{n+1} en fonction de u_n.
   (c) En supposant que (u_n) tende vers une limite finie l telle que $f'(l) \neq 0$, montrer qu'alors $f(l) = 0$.
   (d) En déduire une condition restrictive de la méthode de Newton.

$y = f(a) - f[a](x-a)$

2. On considère une fonction f de classe C^2 sur un intervalle I.
   On pose $g(x) = x - \frac{f(x)}{f'(x)}$. On a donc $u_{n+1} = g(u_n)$.
   (a) Donner l'expression de $g'(x)$ en fonction de f et de ses dérivées.
   (b) A quelle(s) condition(s) sur f a-t-on $g' > 0$? En déduire alors le sens de variation de g.
   (c) Sous ces conditions, montrer que $u_n - u_{n-1}$ et $u_{n+1} - u_n$ sont de même signe. $(n \geqslant 1)$
   (d) En déduire que (u_n) est monotone.

TP scilab : Coût du calcul d'une puissance de matrice

IUT Paris Descartes, DUT Informatique, 1ère année

Analyse, 2017-18

On rappelle que:

- `rand(3,3)` renvoie une matrice aléatoire de taille 3
- `plot2d(x,y)` affiche, lorsque x est le vecteur ligne $(x_1, x_2, ..., x_m)$ et y le vecteur ligne $(y_1, y_2, ..., y_m)$, une ligne brisée reliant les points de coordonnées (x_i, y_i)
- Si A est une matrice, la commande `diag(A)` renvoie le vecteur formé des termes diagonaux de A
- Si d est un vecteur, `diag(d)` renvoie une matrice dont la diagonale est le vecteur d

1 Coût du calcul direct d'une puissance de matrice

1. Dans cette question, on utilisera la fonction `timer` de Scilab (voir l'aide pour la syntaxe) pour mesurer le temps nécessaire à la réalisation d'un calcul.
   Ecrire un programme qui permet, pour $n = 100, 120, 140, 160, ..., 500$, de mesurer le temps T_n de calcul de la puissance 25-ième d'une matrice aléatoire de taille n (on veillera à faire afficher l'indice de la boucle à chaque itération pour pour visualiser l'avancement de l'algorithme).

2. Tracer dans une fenêtre graphique le temps T_n de calcul en fonction de la taille n de la matrice. Quel type de croissance semble-t-il suivre ?

3. Tracer la courbe log-log de T_n, c'est à dire la courbe de la fonction $\log(n) \mapsto \log(T_n)$
   Attention, `timer` peut parfois être très proche de zéro, la fonction log peut présenter des singularités, on peut alors rajouter une faible valeur à `timer` pour éviter ce problème.
   On obtient presque une droite.
   Calculer une valeur approchée de la pente.
   Conclure sur la "complexité" de l'algorithme de calcul direct de la puissance d'une matrice.

2 Comparaison avec l'utilisation de la diagonalisation

Vous avez vu en début d'année que, pour les matrices diagonalisables, c'est à dire qui peuvent s'écrire sous la forme $A = P \times D \times P^{-1}$ avec D diagonale et P inversible, on a $A^k = P \times D^k \times P^{-1}$. Cette méthode "coûte" donc la diagonalisation de A et le calcul de D^k.

Les matrices symétriques sont diagonalisables (c'est un résultat connu qu'on admettra). Remarquez que toute matrice du type $A = {}^t BB$ est symétrique.

1. Remarquez que la commande `R=rand(n,n);A=R'*R` renvoie une matrice carrée A symétrique de taille n.

2. Utiliser la commande `[P,D]=spec(A)` de Scilab pour diagonaliser A. Que représentent les arguments de sortie ?

3. **Calcul de D^{25}:**
Scilab ne "remarque pas" tous les zéros de la matrice diagonale D, et s'échine donc à faire des produits matriciels successifs au lieu d'utiliser le fait que

si $D = \begin{pmatrix} \lambda_1 & 0 & . & . & . & . \\ 0 & \lambda_2 & 0 & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & 0 \\ . & . & . & . & 0 & \lambda_p \end{pmatrix}$ alors $D^n = \begin{pmatrix} \lambda_1^n & 0 & . & . & . & . \\ 0 & \lambda_2^n & 0 & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & 0 \\ . & . & . & . & 0 & \lambda_p^n \end{pmatrix}$

Quelle ligne de commande, utilisant la commande `diag` de Scilab (voir préambule), permet alors de "forcer" Scilab à calculer D^{25} sans faire de produits de matrices ? (Vous avez déjà répondu à cette question lors du TP Scilab d'algèbre linéaire de la semaine du 18 décembre 2017).

4. Ecrire un programme qui permet, pour $n = 100, 120, 140, 160, ..., 500$, de mesurer:

   - le temps T_n de calcul de la puissance 25-ième d'une matrice aléatoire A de taille n par la méthode "directe" (ALGO 1)
   - le temps Q_n de calcul de la puissance 25-ième de la même matrice en utilisant une diagonalisation de A (ALGO 2)

   Faire afficher dans un même graphique les temps T_n et Q_n en fonction de n (on pourra utiliser les arguments optionnels `style=5` pour une courbe en rouge et `style=2` pour une courbe bleue).

5. Tracer les courbes correspondant à $\log n \mapsto \log(T_n)$ et $\log n \mapsto \log(T_n)$.
Que constate-t-on ?

6. En déduire une approximation de la complexité de chacun des algorithmes ALGO 1 et ALGO 2 sous la forme $\Theta(u_n)$, u_n étant une suite référence étudiée dans le chapitre "croissances comparées" de votre cours d'Analyse.

7. Quelle est, à l'heure actuelle, la complexité des meilleurs algorithmes de calcul de produit de matrices ? (Google est votre ami, une fois n'est pas coutume...)

3 Etude statistique

Le travail précédent permet de conjecturer le comportement des temps T_n et Q_n, mais il n'est basé que sur une seule simulation. Pour consolider notre conjecture, on souhaite faire une étude statistique sur une trentaine de simulations.

1. Ecrire un programme qui permet d'afficher dans un même graphique, en fonction de n, les moyennes des temps T_n et les moyennes des temps Q_n sur la trentaine de simulations demandées (réduire éventuellement la valeur maximale de n et/ou de k (puissance de A demandée) si l'exécution prend trop de temps).

2. Que constate-t-on ?

TP scilab : Approximation de valeurs et vecteurs propres

IUT Paris Descartes, DUT Informatique, 1ère année

Analyse, 2017-18

Il existe différentes méthodes de recherche de valeurs propres pour une matrice carrée, la première étant la recherche des racines du polynôme caractéristique $P(X) = \det(A - XI)$, ce qui revient dans la pratique à résoudre des équations du type $f(x) = 0$, problème dont nous avons vu dans un précédent Tp quelques méthodes numériques de résolution.
Nous voyons ici une autre méthode de recherche de valeur propre, basée sur des résultats d'algèbre linéaire, et qui peut s'avérer dans certains cas beaucoup plus rapide.
On rappelle que si X est un vecteur non nul,

$$\boxed{X \textbf{ vecteur propre de } A \text{ associé à } \lambda \Leftrightarrow AX = \lambda X}$$

1 Méthode de la puissance itérée

1.1 Introduction

Cette méthode s'applique au cas d'une matrice $n \times n$ admettant une *valeur propre strictement dominante* λ_1, ce qui signifie que λ_1 doit être strictement plus grande en valeur absolue que les autres valeurs propres.
Dans ce cas, la méthode produit une suite de réels convergeant vers λ_1, et une suite de vecteurs convergeant vers un vecteur propre associé.
Supposons pour simplifier que A est diagonalisable, et soit $(b_1, ...b_n)$ une base de $\mathbb{R}^n$ constituée de vecteurs propres de A, ordonnés de façon que les valeurs propres associées $\lambda_1, ..., \lambda_n$ décroissent en valeurs absolues.
On suppose donc que

$$| \lambda_1 | > | \lambda_2 | \geqslant ... \geqslant | \lambda_n | \qquad (1)$$

Si l'on écrit un vecteur X de $\mathbb{R}^n$ sous la forme $X = c_1 b_1 + ... + c_n b_n$, alors

$$A^k X = c_1 (\lambda_1)^k b_1 + c_2 (\lambda_2)^k b_2 + ... + c_n (\lambda_n)^k b_n \qquad (2)$$

$(k = 1, 2, ...)$.
En supposant $c_1 \neq 0$ et en divisant par $(\lambda_1)^k$, on obtient

$$\frac{1}{(\lambda_1)^k} A^k X = c_1 b_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k b_2 + ... c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k b_n \qquad (3)$$

D'après l'inégalité (1), les fractions $\frac{\lambda_2}{\lambda_1}, ... \frac{\lambda_n}{\lambda_1}$ sont strictement inférieures à 1 en valeur absolue, donc leurs puissances tendent vers 0. Ainsi,

$$(\lambda_1)^{-k} A^k X \longrightarrow c_1 b_1$$

si k tend vers l'infini.

Soit la matrice $A = \begin{pmatrix} 1,8 & 0,8 \\ 0,2 & 1,2 \end{pmatrix}$, $b_1 = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$ et $X = \begin{pmatrix} -0,5 \\ 1 \end{pmatrix}$.

1. Vérifier que b_1 est vecteur propre de A associé à la valeur propre 2.
2. Vérifier que la commande `plot2d([0 4],[0 1],2);` permet de représenter le vecteur b_1.
3. Ecrire un algorithme permettant d'afficher dans une même fenêtre graphique, les vecteurs A^kX normalisés, pour k allant de 0 à 10.
4. Que constatez-vous ?
5. Modifier la valeur du vecteur X, que constate-t-on ?

La division par $(\lambda_1)^k$ que l'on a effectuée dans (3) est une normalisation destinée à obtenir une suite convergeant vers $c_1 b_1$. Ici on ne peut pas normaliser A^kX de cette façon car λ_1 n'est pas connue. Mais on peut le faire en s'arrangeant pour que la plus grande composante de A^kX soit égale à 1. On voit alors que la suite (X_k) ainsi obtenue converge vers un vecteur colinéaire à b_1 dont la plus grand composante vaut 1.
La suite (X_k) permet également d'estimer la valeur propre λ_1:
Si X_k est proche d'un vecteur propre de valeur propre λ_1, alors A^kX est proche de $\lambda_1 X_k$, chaque composante de AX_k valant approximativement λ_1 fois la composante correspondante de X_k. Comme la composante maximale de X_k vaut 1, celle de AX_k est proche de λ_1.

1.2 Algorithme

On peut ainsi établir la méthode de la puissance itérée pour estimer une valeur propre dominante par un algorithme simple:

```
INITIALISATION :
    | x_0 un vecteur initial de composante maximale 1
POUR k = 1, ..., +∞ :
    | x_{k+1} = (1/v_k) A x_k où v_k est une composante maximale de A x_k en valeur absolue.
FIN (POUR)
```

Pour presque tout choix de x_0, la suite (v_k) converge vers la valeur propre dominante et la suite (x_k) converge vers un vecteur propre associé.

1. Ecrire un script exécutant cet algorithme. On pourra par exemple écrire une fonction du type:
   ```
   function [x,v]=VapDom(A,x0)
   ...
   endfunction
   ```
2. Le tester sur A définie en introduction. Contrôler en vérifiant que le vecteur obtenu est bien vecteur propre associé à la valeur propre obtenue. (On pourra utiliser la commande `[P,D]=spec(A)` pour contrôler).
3. Tester et contrôler sur une matrice aléatoire de taille supérieure ou égale à 3.

2 Méthode de la puissance inverse pour estimer une valeur propre quelconque

2.1 Introduction

Cette méthode permet d'obtenir une approximation de *n'importe quelle* valeur propre d'une matrice A, à condition de disposer d'une bonne estimation initiale α de cette valeur propre λ. (Celle-ci peut par exemple être obtenue par une étude graphique du polynôme caractéristique de A.)
Dans ce cas, on pose $B = (A - \alpha I)^{-1}$ et l'on applique la méthode de la puissance itérée à B.
On peut montrer que si $\lambda_1, ..., \lambda_n$ sont les valeurs propres de A, alors celles de B sont

$$\frac{1}{\lambda_1 - \alpha}, \frac{1}{\lambda_2 - \alpha}, ..., \frac{1}{\lambda_n - \alpha}$$

et les vecteurs propres associés sont les mêmes que ceux de A.
Supposons par exemple que α soit plus proche de λ_2 que des autres valeurs propres de A. Alors $\frac{1}{\lambda_2 - \alpha}$ est une valeur strictement dominante de B.

En appliquant la méthode de la puissance itérée sur B, on obtient donc une approximation de $\frac{1}{\lambda_2 - \alpha}$, donc a fortiori une approximation de λ_2.

2.2 Algorithme

1. Ecrire un script exécutant cet algorithme. On pourra par exemple écrire une fonction du type:

```
function [x,v]=VapInv(A,x0,alpha)
...
endfunction
```

2. Tester et contrôler avec la matrice $A = \begin{pmatrix} 10 & -8 & -4 \\ -8 & 13 & 4 \\ -4 & 5 & 4 \end{pmatrix}$ et les estimations α des valeurs propres, $\alpha \in \{1.9\,; 3.3\,; 21\}$.

TP scilab : Algorithmes de recherche de solution d'équation du type $f(x) = 0$, méthode de la dichotomie et méthode de la sécante

IUT Paris Descartes, DUT Informatique, 1ère année

Analyse et Méthodes Numériques, 2017-18

1 Introduction

Pour une fonction f continue sur $\mathbb{R}$, il est en général impossible de résoudre algébriquement l'équation $f(x) = 0$. Nous ne savons en effet résoudre ce problème de manière systématique que lorsque f est un polynôme du 1er ou 2ème degré.
L'analyse permet plus généralement de prouver l'existence de solution d'une telle équation, et d'en donner des approximations à l'aide d'algorithmes. Nous abordons ici deux méthodes, dont on cherchera à évaluer les performances et les conditions d'application.
Toutes ces méthodes supposent, avant d'être mises en oeuvre, que l'on ait localisé la solution α dans un intervalle $]a, b[$ qui ne contient pas d'autres solutions de l'équation, c'est à dire dans lequel f ne s'annule qu'une seule fois.
Dans bien des cas, cette localisation est relativement facile (par des considérations graphiques, où lorsque c'est possible, par une étude des variations).

Nous travaillerons dans ce TP sur la fonction

$$f(x) = e^x - x^3 \ln(x)$$

Nous chercherons à approcher la solution de l'équation $f(x) = 0$ sur l'intervalle $[4; 6]$.

1. Ouvrir le fichier TP1DichotomieSecante dans l'emplacement $H : \backslash COMMUN \backslash DUT1ere\,annee \backslash Maths \backslash Analyse$ de votre ordinateur et le modifier de sorte qu'on obtienne une fenêtre graphique de la fonction $f(x) = e^x - x^3 \ln(x)$ dans l'intervalle $[0; 6]$.
   On pourra utiliser l'aide sur la commande `plot2d` de Scilab.
2. Constater que l'équation $f(x) = 0$ ne peut pas se résoudre par le calcul, puis constater graphiquement que cette équation admet une unique solution dans l'intervalle $[4; 6]$.

2 Dichotomie

La méthode consiste à définir une suite d'intervalles $[u_n, v_n]$ dans lesquels va se trouver la solution α, c'est à dire de sorte que $f(u_n)$ et $f(v_n)$ soient de signes contraires.
Les deux suites (u_n) et (v_n) sont alors définies comme suit:

- $u_0 = a, \quad v_0 = b$

- Si $f\left(\frac{u_n+v_n}{2}\right) \times f(u_n) > 0$ on pose $u_{n+1} = \frac{u_n+v_n}{2} \quad v_{n+1} = v_n$
- Si $f\left(\frac{u_n+v_n}{2}\right) \times f(u_n) < 0$ on pose $u_{n+1} = u_n \quad v_{n+1} = \frac{u_n+v_n}{2}$

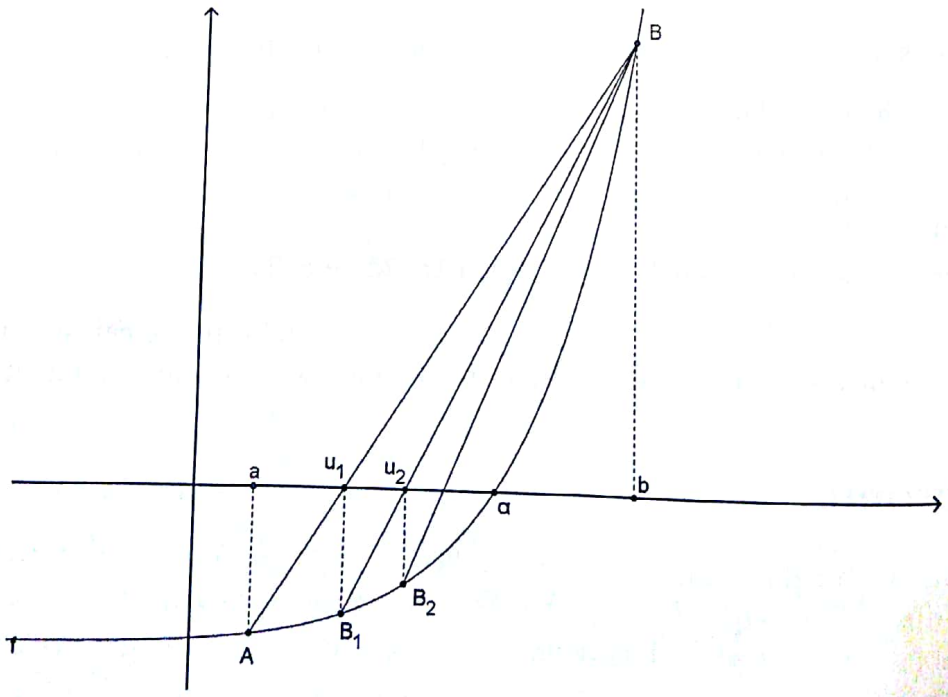
On itère le processus jusqu'à ce que la différence $|v_n - u_n|$ soit inférieure à la précision ϵ demandée. N'importe quelle valeur de l'intervalle $[u_n, v_n]$ fournit alors une approximation à ϵ près de la solution α recherchée.

1. Ecrire une fonction Scilab `Dichotomie` qui prend comme argument trois nombres a, b, e réels et une fonction f, et qui renvoie une approximation de la solution de l'équation $f(x) = 0$ dans l'intervalle $]a, b[$ avec une erreur inférieure à e.
2. Tester la fonction `Dichotomie` pour rechercher les solutions de l'équation $e^x - x^3 \ln(x) = 0$ à 10^{-10} près dans l'intervalle $[4; 6]$.
3. Modifier la fonction `Dichotomie` de sorte quelle renvoie également le nombre n d'itérations nécessaires à l'approximation. Quel est le nombre d'itérations qui permet d'obtenir la précision demandée ?
4. A votre avis, existe-t-il des fonctions continues s'annulant une seule fois dans $[a; b]$ pour lesquelles la méthode par dichotomie ne peut pas s'appliquer ? ?

3 Méthode de la sécante

La solution α de l'équation $f(x) = 0$ ayant été localisée dans un intervalle $[a; b]$, l'idée de la méthode consiste à remplacer la fonction f par la fonction affine représentée graphiquement par la droite (AB) où A et B sont les points de coordonnées respectives $(a; f(a))$ et $(b, f(b))$.

Dans le cas de la figure ci-dessous, cette droite coupe l'axe des abscisses en un point d'abscisse u_1 avec $f(u_1) < 0$. On réitère en cherchant l'abscisse u_2 du point d'intersection de (BB_1) avec l'axe des abscisse, où B_1 est le point de coordonnées $(u_1; f(u_1))$, et ainsi de suite...



Dans le cas de la figure ci-dessous (fonction convexe, tournée vers le haut), avec $f(a) < 0$ et $f(b) > 0$, on construit une suite (u_n) définie par

$$u_0 = a, \quad \forall n \in \mathbb{N} \quad u_{n+1} = u_n - f(u_n)\frac{b - u_n}{f(b) - f(u_n)}$$

La suite (u_n) tend alors vers la solution de $f(x) = 0$ dans l'intervalle $[a, b]$.

1. Ecrire une fonction Scilab `Secante` qui prend comme arguments trois nombres réels a, b, err et une fonction f , et qui renvoie le premier terme u_n de la suite définie ci-dessus qui vérifie $|u_n - u_{n-1}| <$ err et $f(u_n) <$err.

2. Tester la fonction `Secante` pour rechercher les solutions de l'équation $e^x - x^3 \ln(x) = 0$ dans l'intervalle $[4, 6]$ avec err=10^{-10}.

3. Modifier la fonction `Secante` de sorte qu'elle renvoie le nombre d'itérations nécessaires à son exécution. Comparer avec le nombre d'itérations obtenu en question 2.3. Que constatez-vous ?

4. *(facultatif)* Expliquer comment a été définie la suite (u_n).

4 Etude comparée

1. Donner une valeur approchée à 10^{-5} près de la solution de l'équation

$$\frac{1}{3}x^3 - 2x^2 + 4x - 2 = 0$$

sur l'intervalle $[0; 3]$ par la méthode de dichotomie, puis par la méthode de la sécante.

2. Que constate-t-on ? Expliquer le phénomène.

TP scilab : Algorithme de recherche de solution d'équation du type $f(x) = 0$, méthode de Newton

IUT Paris Descartes, DUT Informatique, 1ère année

Analyse, 2017-18

1 Introduction

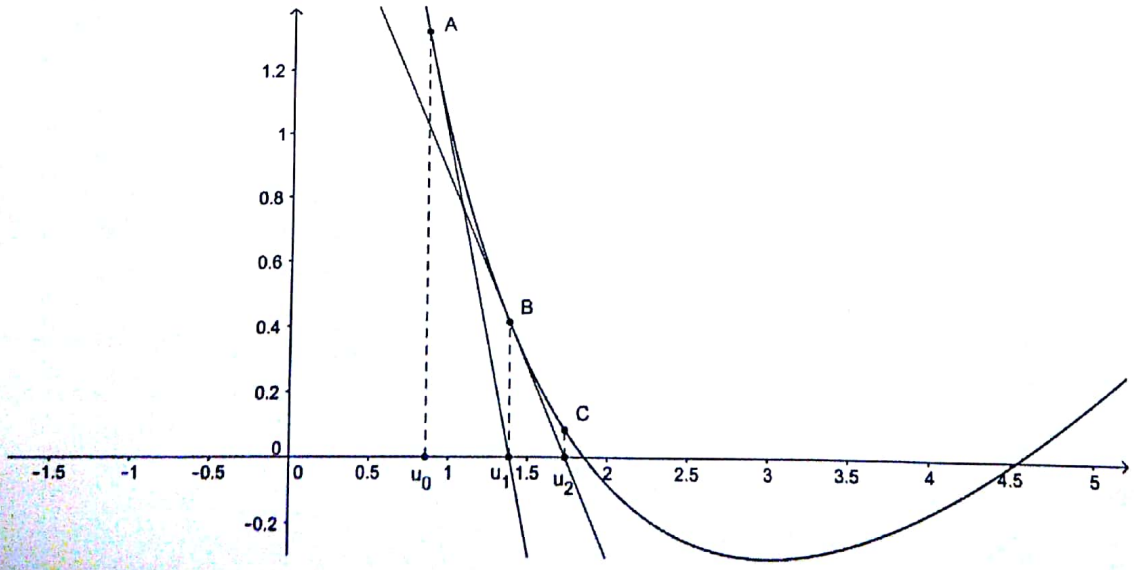
Vous avez vu dans un précédent TP Scilab deux méthodes de recherche de solution d'équation du type $f(x) = 0$: La dichotomie et la méthode de la sécante.
Nous abordons ici la méthode de Newton.

1. Ouvrez le fichier TPNewton du $H : \backslash COMMUN\backslash DUT1ereannee\backslash Maths\backslash Analyse$ de votre ordinateur.
2. Lire et comprendre le script ainsi que ses commentaires.

2 Principe de la méthode de Newton

La solution α de l'équation $f(x) = 0$ ayant été localisée dans un intervalle $[a; b]$ et une valeur initiale u_0 étant donnée, l'idée de la méthode consiste à remplacer la fonction f par sa tangente au point d'abscisse u_0. Cette tangente coupe l'axe des abscisses en u_1. On réitère le processus à partir de u_1, etc.
On construit ainsi une suite de nombre réels (u_n) qui, sous certaines conditions sur f, converge vers α.

Par exemple pour $f(x) = x - 3\ln(x)$ sur l'intervalle $]0, 3[$:



On admet que la tangente à la courbe de f au point d'abscisse a coupe l'axe des abscisse en $a - \frac{f(a)}{f'(a)}$. (démontré en TD)

On en déduit que

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)} \qquad (*)$$

Remarque: L'algorithme à implémenter doit donc évaluer à chaque pas le nombre dérivé de la fonction au point u_n. Nous donnerons ici explicitement l'expression de la dérivée selon la fonction étudiée.

1. Ecrire une fonction Scilab `Newton` qui prend comme arguments deux nombres a, e réels, qui renvoie une valeur approchée y de α, le nombre m d'itérations, et qui affiche progressivement dans la fenêtre graphique la suite des nombres u_n tels que $u_0 = a$.
   Indications:
   - On posera comme condition d'arrêt que la différence de deux termes consécutifs de la suite est inférieure à la valeur e.
   - On pourra utiliser la commande `xclick()` pour suspendre l'exécution du programme entre l'affichage de deux points consécutifs. (voir aide)
   - On précise que l'instruction `style=-4` comme argument optionnel dans `plot2d` permet d'afficher un point en "diamant". Voir aide pour plus de précisions.
2. (a) Tester la fonction `Newton` pour la fonction $f(x) = x - 3\ln(x)$ avec $u_0 \in \{0.5; 6; 2.5; 3.5\}$ et $e = 10^{-3}$.
   (b) Tester la fonction `Newton` pour la fonction $f(x) = x^3 - 3x^2 + 2x + 1$ sur l'intervalle $[-1, 2.5]$ en prenant $u_0 = 4$.
   (c) Tester la fonction `Newton` pour la fonction $f(x) = \frac{1}{3}x^3 - 2x^2 + 4x - 2$ sur l'intervalle $[0, 4]$ en prenant $u_0 = 2$ puis $u_0 = 2.5$. Que se passe-t-il ? Pourquoi ?
   (d) Selon vous, à quelle(s) condition(s) sur f la suite (u_n) est-elle croissante ? décroissante ? A quelle(s) condition(s) sur f la suite (u_n) change-t-elle de sens de variation ?
   (e) Essayez de mettre en défaut l'algorithme en choisissant judicieusement u_0.
3. Pour une fonction de votre choix, reprendre les fonctions `Dichotomie` et `Secante`, et comparer les nombres d'itérations avec celui de la fonction `Newton`. Quel(s) algorithme(s) vous semble(nt) le(s) plus efficace(s) ?
4. Quels sont les avantages et les inconvénients de chaque méthodes du point de vue de leurs conditions d'applications et de leurs vitesse de convergence ?

3 Let's prove it !

On considère une fonction f de classe C^2 sur un intervalle I.
On pose $g(x) = x - \frac{f(x)}{f'(x)}$. On a donc $u_{n+1} = g(u_n)$.

1. Donner l'expression de $g'(x)$ en fonction de f et de ses dérivées.
2. A quelle(s) condition(s) sur f a-t-on $g' > 0$? En déduire alors le sens de variation de g.
3. Sous ces conditions, montrer que $u_n - u_{n-1}$ et $u_{n+1} - u_n$ sont de même signe. ($n \geqslant 1$)
4. En déduire que (u_n) est monotone.

Méthodes de Résolution $f(x) = 0$

1) Dichotomie

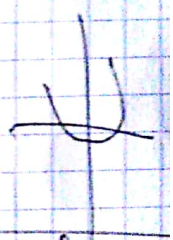
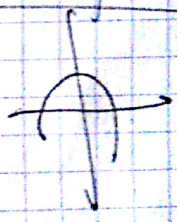
Voir cours

2) Méthode de la sécante

⇒ Le point cherché α est la limite de U_n quand $n \to +\infty$

Avec $U_n =$

Continue ; change de signe 1 fois

Convexe	
Concave	

	hypothèse	avantage	inconvénient
Dicho	toute fonction	marche # tps	lent
Sécante	convexité régulière	+ rapide	restrictive
Newton	dérivée qui ne s'annule pas	très rapide	si changement de convexité, valeur s'éman