PCSI - mathématiques

Approximation

On étudie dans ce chapitre divers types de problèmes posés par les questions de calculs approchés.

1 Calcul approché des zéros d'une fonction

On considère ici des fonctions $f : I \to \mathbb{R}$, où I est un intervalle de $\mathbb{R}$.

1.1 Séparation des racines

$\xi \in I$ est solution *isolée* de l'équation (E) s'il existe $\alpha > 0$ tel que ξ soit la seule solution de (E) sur $]\xi - \alpha, \xi + \alpha[$, autrement dit si ξ est "assez loin" des autres racines de (E). Cette condition n'est pas toujours remplie (on pense à $x \mapsto x \sin \frac{1}{x}$ en 0). Mais si c'est le cas, et si l'on dispose d'un algorithme "ne sortant pas" de $]\xi - \alpha, \xi + \alpha[$, on est certain d'approximer la bonne racine.

Quelques remarques théoriques permettent de s'en assurer :

- Si $f(a) f(b) < 0$ et si f est continue, le th. des valeurs intermédiaires garantit l'existence d'une racine ξ entre a et b.
- Si f est strictement monotone sur $[a, b]$, elle est injective et ξ est l'unique racine sur $[a, b]$.
- Si f est de classe $\mathcal{C}^1$ au voisinage de ξ et si $f'(\xi) \neq 0$, il existe $\alpha > 0$ tel que f' garde le même signe strict que $f'(\xi)$ sur $]\xi - \alpha, \xi + \alpha[$, d'où la stricte monotonie sur cet intervalle, garantissant le caractère isolé de la racine ξ.

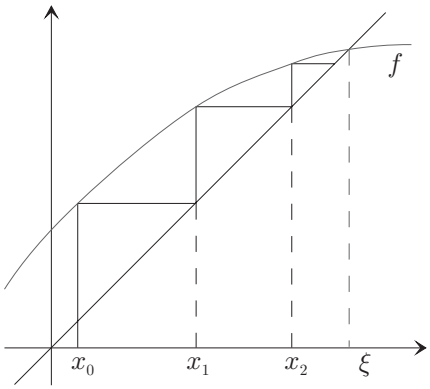
1.2 Résolution d'équations "$f(x) = x$"

Définition 1 *Une application $f : I \to \mathbb{R}$ est* contractante *si elle est k-lipschitzienne sur I avec une constante $k < 1$.*

Remarque 1 *Il ne faut pas confondre la définition : $|f(y) - f(x)| \leq k|y - x|$ avec la condition plus faible $|f(y) - f(x)| < |y - x|$, cf. ex. 5.*

Théorème 1 (du point fixe) *Soit $f : I \to \mathbb{R}$ une application k-contractante sur l'intervalle* fermé *I. Alors l'équation f admet un unique point fixe sur I. De plus, si $x_0 \in I$, la suite définie par $x_{n+1} = f(x_n)$ converge vers ξ et pour tout $n \in \mathbb{N}$:*

$$|x_n - \xi| \leq \frac{k^n}{1-k} |x_1 - x_0| .$$



La majoration de l'erreur s'exprime en disant que la convergence est géométrique. Cela ne donne pas un algorithme très performant. Typiquement, le nombre de décimales exactes augmente de la même valeur à chaque étape. Le cas où $f'(\xi) = 0$ donne lieu à une convergence notablement plus rapide (*cf.* ex. 7).

Programmation En pseudo-langage on peut écrire ainsi l'algorithme :

```
1: debut pointFixe
2: x ← x_0,
3: ecart ← |x−f(x)|/(1−k),
4: tant que ecart > precision faire
5:     x ← f(x),
6:     ecart ← k× ecart
7: fin faire
8: afficher x
9: fin pointFixe
```

1.3 Résolution d'équations "$f(x) = 0$"

1.3.1 Méthode par itération

On peut se ramener à une recherche de point fixe en remarquant que $f(x) = 0 \Leftrightarrow \lambda f(x) = 0$ $(\lambda \neq 0) \Leftrightarrow g(x) = x$ où $g(x) = \lambda f(x) + x$. Ceci ne convient que si l'on peut trouver λ tel que g soit contractante ; une étude de la fonction f est indispensable.

1.3.2 Méthode de dichotomie

On suppose la fonction f continue sur $[a, b]$ avec $f(a) f(b) < 0$. On définit deux suites (x_n) et (y_n) de manière récurrente par $x_0 = a$, $y_0 = b$ et

- si $f\left(\frac{x_n + y_n}{2}\right)$ a le même signe que $f(x_n)$: $x_{n+1} = \frac{x_n + y_n}{2}$ et $y_{n+1} = y_n$;
- si $f\left(\frac{x_n + y_n}{2}\right)$ a le même signe que $f(y_n)$: $x_{n+1} = x_n$ et $y_{n+1} = \frac{x_n + y_n}{2}$.

Proposition 1 *Les suites (x_n) et (y_n) sont adjacentes et convergent vers $\xi \in [a,b]$ tel que $f(\xi) = 0$. De plus, pour tout $n \in \mathbb{N}$,*

$$\left.\begin{array}{l} |x_n - \xi| \\ |y_n - \xi| \end{array}\right\} \leq \frac{|b-a|}{2^n}.$$

L'avantage de cette méthode est sa stabilité : tous les termes calculés restent dans $[a,b]$. Son inconvénient est sa lenteur, qui est du même ordre que la méthode du point fixe.

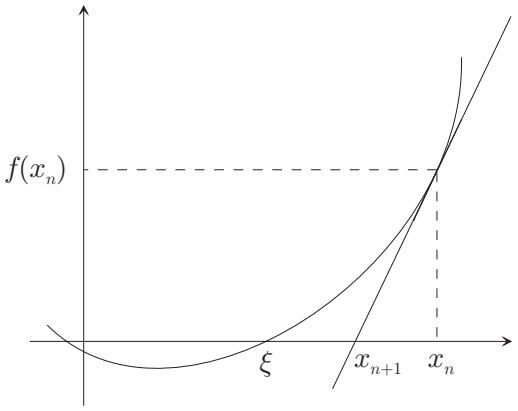
Programmation

```
1: debut dichotomie
2: si f(a) < 0 alors
3:    x ← a, y ← b
4: sinon
5:    x ← b, y ← a
6: fin si,
7: ecart ← b − a,
8: tant que ecart > precision faire
9:    m ← (x+y)/2,
10:   si f(m) > 0 alors
11:      y ← m
12:   sinon
13:      x ← m
14:   fin si,
15:   ecart ← ecart/2
16: fin faire
17: afficher (x+y)/2
18: fin dichotomie
```

1.3.3 Méthode de Newton

La méthode de Newton consiste à remplacer une estimation de la solution (x_n) par une nouvelle extrapolée à partir de la tangente à $\mathcal{C}_f$ en x_n. Cela correspond à l'itération de la relation suivante :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$



Proposition 2 *Si la fonction f est deux fois dérivable sur $[a,+\infty[$ avec $f(a) < 0$ et $f(x_0) > 0$, si $f' \geq m_1 > 0$, $0 \leq f'' \leq M_2$ alors*

1. *l'équation "$f(x) = 0$" admet une unique solution ξ ;*
2. *(x_n) est strictement décroissante et converge vers ξ ;*
3. *pour tout $n \in \mathbb{N}$:*

$$|x_n - \xi| \leq \frac{2m_1}{M_2}\left(\frac{M_2}{2m_1}(x_0 - \xi)\right)^{2^n}.$$

Ce type de convergence est appelé *quadratique*. Il se traduit généralement par un doublement du nombre de chiffres significatifs à chaque étape, *cf.* ex. 8.

Remarque 2 *ξ étant inconnu, on remplacera en pratique $x_0 - \xi$ par $x_0 - a$ pour estimer l'erreur commise.*

Remarque 3 *Le terme $\frac{M_2}{2m_1}(x_0 - \xi)$ n'est d'aucune utilité s'il est ≥ 1, mais la prop. 2 garantit la CV de (x_n). Pour n_0 assez grand, on aura donc $\frac{M_2}{2m_1}(x_{n_0} - \xi) < 1$ et la convergence quadratique entrera en jeu. Ce sont donc les premières étapes de la méthode de* Newton *qui risquent d'être coûteuses en temps, d'où l'intérêt de bien choisir x_0.*

Outre cette sensibilité à l'estimation initiale, il faut porter au passif de la méthode son exigence en hypothèses. Si celles-ci font défaut, l'algorithme risque de diverger grossièrement (cas d'une fonction "plate").

Programmation

```
1: debut Newton
2: x ← x_0, ecart ← (M_2/2m_1)(x_0 − a),
3: ε ← (M_2/2m_1) × precision,
4: tant que ecart > ε faire
5:    x ← x − f(x)/f'(x), {∗}
6:    ecart ← ecart^2
7: fin faire
8: affficher x
9: fin Newton
```

2 Valeur approchée de réels

2.1 Approximation de $\sqrt{\alpha}$

L'algorithme utilisé par Héron d'Alexandrie pour le calcul des valeurs approchées de $\sqrt{\alpha}$ remonte au I$^{\text{er}}$ siècle de notre ère. Il repose sur l'itération de la suite

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{\alpha}{x_n}\right)$$

à partir d'une valeur $x_0 \geq \sqrt{\alpha}$ (on peut remarquer que $\max(\alpha, 1)$ convient pour tout α). On voit facilement que

- x_{n+1} est la moyenne arithmétique de x_n ($> \sqrt{\alpha}$ — *cf. infra*) et $\frac{\alpha}{x_n}$ ($< \sqrt{\alpha}$), donc est une meilleure approximation de $\sqrt{\alpha}$ que x_n ;
- l'algorithme est en fait un cas particulier de la méthode de Newton lorsque $f(x) = x^2 - \alpha$. Avec les mêmes notations, $M_2 = 2$ et $m_1 = 2\sqrt{\alpha}$ qu'il faut aussi *minorer* pour connaître le "a" et contrôler la précision.

Nous ne reprenons pas l'algorithme de Newton en entier, il suffit de remplacer la ligne 5 repérée $\{*\}$ par la suivante :

```
5: x ← (1/2)(x + α/x)
```

La convergence est quadratique. Notons $\varepsilon_n = x_n - \sqrt{\alpha}$. Alors on obtient facilement $\varepsilon_{n+1} = \frac{\varepsilon_n^2}{2x_n} < \frac{\varepsilon_n^2}{2\sqrt{\alpha}}$ d'où

$$\varepsilon_n < 2\sqrt{\alpha}\left(\frac{\varepsilon_0}{2\sqrt{\alpha}}\right)^{2^n}.$$

Par exemple, pour $\alpha = 3$, avec $x_0 = 2$, comme $\frac{\varepsilon_0}{2\sqrt{\alpha}} < \frac{1}{10}$ on obtient $\varepsilon_5 < 4.10^{-32}$.

2.2 Approximation de e

e est par définition la limite de la suite (x_n) définie par

$$x_n = \sum_{k=0}^{n} \frac{1}{k!}.$$

Mais si $y_n = x_n + \frac{1}{n.n!}$ (avec $y_0 = 4$), (x_n) et (y_n) sont adjacentes et encadrent e à la (bonne) précision $\frac{1}{n.n!}$. L'algorithme tire parti d'un calcul de proche en proche des factorielles.

Programmation

1: **debut** E
2: $x \leftarrow 1$, $f \leftarrow 1$, $i \leftarrow 1$,
3: **tant que** $\frac{1}{f \times i} >$ precision **faire**
4: $\quad x \leftarrow x + \frac{1}{f}$,
5: $\quad i \leftarrow i + 1$,
6: $\quad f \leftarrow f \times i$
7: **fin faire**
8: **afficher** x
9: **fin** E

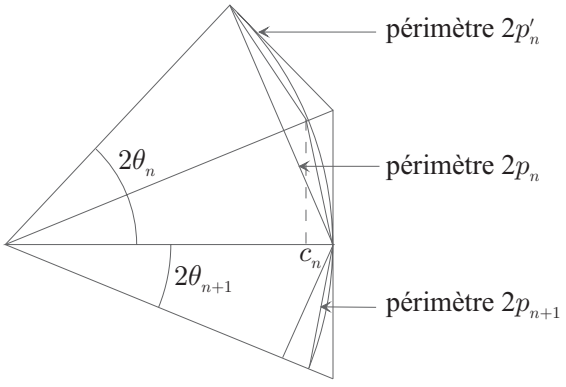
À titre indicatif, le nombre C_n de chiffres significatifs obtenu en fonction de n évolue ainsi :

n	10	20	50	70	100
C_n	7	19	66	101	159

2.3 Approximation de π

2.3.1 Méthode d'Archimède

Pour calculer une valeur approchée de π, Archimède, au III$^{\text{ème}}$ siècle av. J.C., utilisa des polygones réguliers inscrits dans et circonscrits au cercle, à 3, 6, ..., 3.2^n côtés[1]. Les relations entre les périmètres de ces polygones s'écrivent purement algébriquement. Notons p_n (*resp.* p'_n) le demi-périmètre du polygone à 3.2^n côtés inscrit (*resp.* circonscrit) et $\theta_n = \frac{\pi}{3.2^n}$ son demi-angle au sommet.



[1] Archimède s'arrêta à 96 côtés, mais la méthode qu'il décrit se prête à l'itération à un ordre quelconque.

Si $c_n = \cos\theta_n$, $s_n = \sin\theta_n$ on déduit facilement de $\theta_n = 2\theta_{n+1}$ les relations $c_n = 2c_{n+1}^2 - 1$ et $s_n = 2c_{n+1}s_{n+1}$ d'où compte tenu de $p_n = 3.2^n s_n = 3.2^{n+1}s_{n+1}c_{n+1}$:

$$c_{n+1} = \sqrt{\frac{1+c_n}{2}} \text{ et } p_{n+1} = \frac{p_n}{c_{n+1}},$$

$p'_n = 3.2^n \tan\theta_n = \frac{p_n}{c_n}$ étant là pour compléter par une majoration l'encadrement de $\pi = \lim p_n$. Les valeurs initiales sont $c_0 = \frac{1}{2}$ et $p_0 = \frac{3\sqrt{3}}{2}$.

Quelle est l'efficacité de la méthode ? On peut majorer $p'_n - p_n = p_n\left(\frac{1}{c_n} - 1\right)$ compte tenu de $\cos t \geq 1 - \frac{t^2}{2}$: $p'_n - p_n = p_n\frac{1-c_n}{c_n} < \pi\frac{\theta_n^2}{2-\theta_n^2} = \frac{\pi}{2/\theta_n^2 - 1} = \frac{\pi}{9.2^{2n+1}/\pi^2 - 1}$.

C'est peu pratique à utiliser comme test d'arrêt (mieux vaut employer l'expression initiale), mais cela suggère le caractère géométrique de la convergence qui est confirmé par une étude de $\varepsilon_n = \pi - p_n$: on montre que $\varepsilon_{n+1} \sim \frac{1}{4}\varepsilon_n$, *cf.* ex. 13.

Programmation

1: **debut** Archimede
2: $c \leftarrow \frac{1}{2}$, $p \leftarrow \frac{3\sqrt{3}}{2}$,
3: ecart $\leftarrow p$, $\{\frac{1}{c_0} - 1 = 1\}$
4: **tant que** ecart $>$ precision **faire**
5: $\quad c \leftarrow \sqrt{\frac{1+c}{2}}$,
6: $\quad p \leftarrow \frac{p}{c}$,
7: $\quad$ ecart $\leftarrow p\left(\frac{1}{c} - 1\right)$
8: **fin faire**
9: **afficher** $\frac{p}{2}\left(\frac{1}{c} + 1\right)$
10: **fin** Archimede

2.3.2 Formules en arctan

Les formules en arctangente inventées par Euler au tournant du XVIII$^{\text{ème}}$ ont donné un nouvel élan aux approximations de π. Elles sont basées sur le *développement en série entière*[2] de cette fonction : $\arctan x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}$. À titre d'exemple, à partir de $\arctan\frac{1}{\sqrt{3}} = \frac{\pi}{6}$ on obtient

$$\pi = 6\sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)\left(\sqrt{3}\right)^{2n+1}} = 2\sqrt{3}\sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)3^n}$$

L'erreur commise en approchant $\arctan x$ par $\sum_{n=0}^{N} \frac{(-1)^n x^n}{(2n+1)}$ est bien contrôlée grâce à la majoration du *reste de la série alternée*[2]

$$\left|\arctan x - \sum_{n=0}^{N} \frac{(-1)^n x^n}{(2n+1)}\right| \leq \frac{x^{2N+1}}{2N+1}$$

d'où en substituant $x = \frac{1}{\sqrt{3}}$ et en multipliant par 6 :

$$\left|\pi - 2\sqrt{3}\sum_{n=0}^{N} \frac{(-1)^n}{(2n+1)3^n}\right| \leq \frac{2\sqrt{3}}{(2N+1)3^N}$$

[2] Voir cours de spé.

Programmation

1: **debut** Euler
2: $n \leftarrow 0$, $i \leftarrow 1$, $p \leftarrow 1$, $s \leftarrow 0$,
3: ecart $\leftarrow 1$, $\delta \leftarrow \frac{\text{precision}}{2\sqrt{3}}$,
4: **tant que** ecart $> \delta$ **faire**
5: $\quad s \leftarrow s + (-1)^n \times$ ecart,
6: $\quad n \leftarrow n+1$, $i \leftarrow i+2$, $\{i = \text{impairs}\}$
7: $\quad p \leftarrow 3p$,
8: $\quad$ ecart $\leftarrow \frac{1}{i \times p}$
9: **fin faire**
10: **afficher** $2s\sqrt{3}$
11: **fin** Euler

En ce qui concerne la vitesse de convergence, le facteur $\frac{1}{3^N}$, même accompagné du $\frac{1}{2N+1}$, n'a rien d'impressionnant par rapport au $\frac{1}{4^N}$ (environ) de la méthode précédente. Cependant, on améliore facilement l'efficacité en trouvant de meilleures valeurs que $\frac{1}{\sqrt{3}}$ à substituer dans la fonction arctan.

La célèbre formule de MACHIN (1706) :

$$\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{139}$$

a été la première à véritablement faire progresser le calcul, fournissant 100 décimales[3]. On passe de $\frac{1}{3^N}$ à $\frac{1}{25^N}$.

3 Calcul approché d'une intégrale

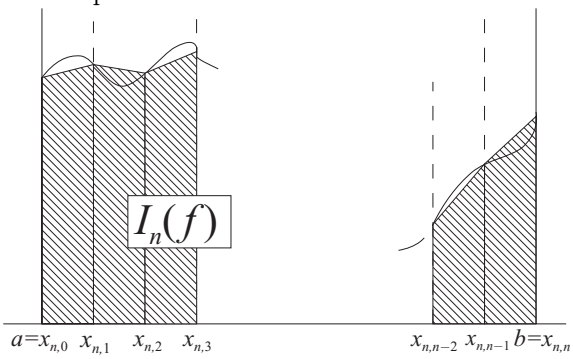
Soit f une fonction intégrable de $[a,b]$ dans $\mathbb{R}$. On utilise la subdivision de pas constant $d_n = (x_{n,i})_{0 \le i \le n}$ dont les points sont définis par $x_{n,i} = a + i\frac{b-a}{n}$.

La *méthode des trapèzes* consiste à approcher l'intégrale $I = \int_{[a,b]} f$ de la fonction f par

$$\begin{aligned} I_n(f) &= \frac{b-a}{2n} \sum_{i=0}^{n-1} \left(f(x_{n,i}) + f(x_{n,i+1})\right) \\ &= \frac{b-a}{2n} \left(f(a) + f(b) + 2 \sum_{i=1}^{n-1} f(x_{n,i}) \right) \end{aligned}$$

Cela signifie qu'à la fonction f, on substitue sur chaque segment $[x_i, x_{i+1}]$ la fonction *affine* qui coïncide avec f en $x_{n,i}$ et $x_{n,i+1}$. En effet, le terme $\frac{b-a}{2n}\left(f(x_{n,i}) + f(x_{n,i+1})\right)$ représente l'aire du *trapèze* ayant ses sommets en $(x_{n,i}, 0)$, $(x_{n,i}, f(x_{n,i}))$, $(x_{n,i+1}, f(x_{n,i+1}))$ et $(x_{n,i+1}, 0)$, d'où le nom de la méthode.

Cela correspond au schéma suivant :



[3] W. Shanks calcula même 706 décimales en 1873 (mais seules les 527 premières étaient correctes).

On montre (*cf.* ex. 9), à l'aide du théorème de ROLLE le

Lemme 1 *Si f est de classe $\mathcal{C}^2$ sur $[\alpha,\beta]$ $(\alpha < \beta)$ et si φ est la fonction affine qui coïncide avec f en α et β, pour tout $x \in [\alpha,\beta]$ il existe $c \in\ ]\alpha,\beta[$ tel que*

$$f(x) - \varphi(x) = \frac{1}{2}(x-\alpha)(x-\beta)\ f''(c)$$

d'où la majoration

$$|f(x) - \varphi(x)| \le \frac{(x-\alpha)(\beta-x)}{2} \|f''\|_\infty .$$

On l'applique ensuite sur chaque segment $[x_{n,i}, x_{n,i+1}]$ pour $0 \le i \le n-1$ pour en déduire la majoration de l'erreur commise en remplaçant $\int_{[a,b]} f$ par son approximation $I_n(f)$.

Théorème 2 *Si f est de classe $\mathcal{C}^2$ de $[a,b]$ dans $\mathbb{R}$,*

$$|I - I_n(f)| \le \frac{|b-a|^3}{12n^2} \|f''\|_\infty .$$

En ce qui concerne la programmation et l'efficacité de l'algorithme, un majorant $O\left(\frac{1}{n^\alpha}\right)$ n'est pas fameux[4]. Mais une remarque permet d'améliorer notablement la situation.

Supposons que la précision obtenue avec I_n soit insuffisante. Si l'on recalcule naïvement I_{n+1} il faudra déterminer (presque) tous les $x_{\boldsymbol{n+1},i}$ (et leurs images par f) alors qu'ils n'ont rien de commun avec les $x_{\boldsymbol{n},i}$ (hormis a et b). Il n'est pas plus coûteux de passer directement au calcul des $x_{\boldsymbol{2n},i}$ dont plus de la moitié sont déjà calculés puisque $x_{2n,2k} = x_{n,k}$.

On calculera ainsi I_1, I_2, ..., I_{2^p} d'où une majoration bien plus favorable

$$|I - I_{2^p}(f)| \le \frac{|b-a|^3}{12.2^{2p}} \|f''\|_\infty$$

qui redonne une convergence géométrique du même ordre que les méthodes vues précédemment (hors NEWTON), *cf.* ex. 14 et 15.

Le procédé d'*accélération de convergence* de ROMBERG permet d'améliorer encore les performances.

Programmation

1: **debut** trapeze
2: $h \leftarrow b - a$,
3: $s \leftarrow \frac{f(a)+f(b)}{2}$,
4: ecart $\leftarrow \frac{h^3 M_2}{12}$,
5: **tant que** ecart $>$ precision **faire**
6: $\quad x \leftarrow a + \frac{h}{2}$,
7: $\quad$ **tant que** $x < b$ **faire**
8: $\quad\quad s \leftarrow s + f(x)$,
9: $\quad\quad x \leftarrow x + h$
10: $\quad$ **fin faire**
11: $\quad h \leftarrow \frac{h}{2}$, ecart $\leftarrow \frac{\text{ecart}}{4}$
12: **fin faire**
13: **afficher** $h \times s$
14: **fin** trapeze

[4] On parle de *convergence logarithmique* : typiquement, chaque décimale ou groupe de décimales demande deux fois plus de calculs que l'étape précédente.