MT 360 : Méthodes numériques et modèles déterministes

Méthodes itératives, convergence, vitesse, ordre et accélération de convergence

Séances de TD des 11-12 février 2025 et 11-12 mars 2025

Exercice 1 Calcul de π par la méthode d'Archimède On cherche à résoudre sur $[0, \infty[$ l'équation suivante :

$$x = -\ln(x) \tag{1}$$

1. Montrer qu'il existe une seule solution s et qu'elle est dans l'intervalle $]0, 1[$
2. Montrer que la suite de type point fixe $x_{n+1} = -\ln x_n$ diverge (on examinera en particulier la croissance de la suite des erreurs $e_n := x_n - s$)
3. A l'aide du changement de variable $x := e^{-y},\ y > 0$, proposer une autre itération de type point fixe qui converge vers s. Etablir le domaine de convergence (i.e. l'ensemble des conditions initiales pour lesquelles la suite récurrente converge vers s). Montrer que la convergence est d'ordre 1.
4. Proposer une itération de type Newton-Raphson pour trouver la solution s de $y - e^{-y} = 0$
5. Soit p l'ordre de convergence de cette itération Newton-Raphson. Montrer $p \geq 2$. Montrer $p = 2$.

Exercice 2 Cet exercice est une application du théorème de point fixe de Banach et des notions de vitesse, ordre et accélération de convergence. On cherche à résoudre dans $\mathbb{R}$ l'équations suivante :

$$e^{-x} = x \tag{2}$$

1. Montrer que l'application $f : [e^{-1}, 1] \to \mathbb{R} : x \mapsto e^{-x}$ satisfait les hypothèses du théorème de Banach

2. En appliquant le théorème de point fixe de Banach, prouver l'existence et l'unicité de la solution s du problème (2). Proposer l'itération de point fixe correspondante (avec un intervalle de conditions initiales x_0 admissibles) qui converge vers cette solution s du problème (2)

3. Avec cette itération simple, donner l'expression (symbolique) du nombre maximum d'itérations nécessaires à priori pour obtenir 10 chiffres significatifs (en base 10) de la solution s

4. Donner l'ordre de convergence de l'itération proposée. Donner également les valeurs asymptotiques de la vitesse de convergence et du nombre de chiffres significatifs gagnés par itération (expressions symboliques) pour le schéma proposé

5. Proposer un schéma d'ordre 2 pour le calcul de s. Donner approximativement le nombre d'itérations nécessaires pour obtenir avec ce schéma 10 chiffres significatifs (en base 10) de la solution s, supposant que la condition initiale est donnée avec un chiffre significatif

A titre indicatif, $e^{-1} \simeq 0,368$ *et* $e^{-e^{-1}} \simeq 0,692$

Exercice 3 On considère l'évolution d'une population de bactéries dont la dynamique de croissance est fixée par le modèle logistique :

$$\left\{ \begin{array}{l} x_{n+1} = \mu x_n(1-x_n) \\ x_0 \in [0,1] \end{array} \right. \tag{3}$$

où $x_n \in [0,1]$ désigne la concentration totale de bactéries à l'instant $n \geq 0$ et où le taux de croissance $\mu \in [0,4]$ est un paramètre constant.

1. Montrer que pour toute condition initiale $x_0 \in [0,1]$, on a $x_n \in [0,1]$, $\forall n \geq 0$

2. Montrer que les valeurs d'équilibre (points fixes) pour cette population ne peuvent être que 0 et $1 - \frac{1}{\mu}$. Sous quelle(s) condition(s) l'équilibre $1 - \frac{1}{\mu}$ existe-t-il?

3. Les expérimentateurs peuvent fixer le taux de croissance μ en contrôlant la température, les concentrations en oxygène et en sucre et la luminosité. Quelle valeur de μ doivent-ils fixer pour stabiliser la concentration de la population de bactéries à $x_\infty = \frac{1}{2}$. Cet équilibre est-il localement stable et si oui, quel est son bassin d'attraction (justifiez votre réponse en appliquant le théorème de point fixe de Banach)?

4. On souhaite atteindre la concentration d'équilibre $x_\infty = \frac{5}{7}$. A quelle valeur du taux de croissance μ correspond cette concentration d'équilibre? Avec ce taux de croissance, l'équilibre $x_\infty = \frac{5}{7}$ est-il attractif? Quelle conclusion pouvez-vous en tirer sur la "réalisabilité" de cet équilibre?

5. On propose de piloter le taux de croissance grâce à une rétro-action (feedback) de la manière suivante :

$$\mu(x_n, x_\infty) = \mu + \alpha(x_n - x_\infty)$$

où α est un paramètre constant. Comment choisir α pour avoir une convergence locale d'ordre 2 vers $x_\infty = \frac{5}{7}$? Avec cette valeur de α, pour quelle plage de valeurs de la concentration initiale x_0, l'équilibre x_∞ peut-il être atteint?

Exercice 4 On cherche à évaluer numériquement, à l'aide d'une méthode itérative, la valeur de $\sqrt{A}$ pour $A \in]1, 3[$.

1. Prouver que l'approximation x_n définie par

$$x_{n+1} = x_n + \frac{1}{2}\left(A - x_n^2\right) \tag{4}$$

converge vers la valeur souhaitée, pour $x_0 \in]0, 2[$. On décomposera le problème de la manière suivante :

   (a) Montrer qu'il existe $\epsilon > 0$ tel que l'intervalle $I := \left[\epsilon, \frac{A+1}{2}\right]$ est stable pour l'application $f : x \mapsto f(x) := x + \frac{1}{2}\left(A - x^2\right)$

   (b) Montrer que si l'itération (6) converge vers un point fixe s, alors $s \in \{-\sqrt{A}, +\sqrt{A}\}$

   (c) Montrer que le point fixe $+\sqrt{A}$ est localement attractif (stable) et que le point fixe $-\sqrt{A}$ est répulsif (instable)

   (d) Utiliser le théorème de point fixe de Banach pour montrer que $]0, 2[$ est inclu dans le bassin d'attraction du point fixe (équilibre) $s = +\sqrt{A}$ pour le système dynamique (6)

2. Montrer que l'approximation (6) converge de manière linéaire et donner une estimation de sa vitesse de convergence.

3. Montrer que l'approximation (6) est une approximation de type "cordes parallèles" pour la solution de l'équation $F(x) := x^2 - A = 0$. En déduire une amélioration de l'itération (avec une convergence toujours linéaire mais une vitesse de convergence plus élevée) dans le cas où $A = 2$.

4. Donner, sous la forme

$$y_{k+1} = g\left(y_k\right)\ ,\ k \geq 0$$

l'itération obtenue lorsqu'on applique l'accélération d'Aitken-Steffensen pour accélérer l'itération (6). Quelle condition initiale peut-on choisir pour y_0? Peut-on garantir la convergence de y_k? Si oui, vers quelle limite et avec quel ordre de convergence?

5. Donner l'itération Newton-Raphson pour calculer $\sqrt{A}$. Quel est l'ordre de convergence de cette itération? Quelle est la limite obtenue si A est remplacé par la valeur approchée $\hat{A} := A\,(1+\rho(A))$?

Exercice 5 On cherche à évaluer numériquement, à l'aide d'une méthode itérative, la valeur de la seule racine réelle du polynôme

$$P(x) := x^3 - 6x^2 + 12x - 6 \tag{5}$$

1. Montrer qu'il existe au moins un solution s de l'équation $P(s) = 0$ dans l'intervalle $]0,1[$.

2. **Méthode des cordes parallèles.** On considère l'itération de type point fixe

$$x_{n+1} = x_n - \frac{P(x_n)}{\mu} \tag{6}$$

   (a) Pour quelles valeurs de μ, l'intervalle $[0,1]$ est-il stable pour l'application

$$f : x \mapsto f(x) := x - \frac{P(x)}{\mu}\ ?$$

   (b) Pour quelles valeurs de μ, l'application f est-elle contractante dans $[0,1]$?

   (c) Pour quelles valeurs de μ l'itération (6) converge-t-elle vers la solution unique de $P(x) = 0$ dans l'intervalle $]0,1[$? Prouver votre résultat (existence, unicité et convergence) en vous servant du théorème de Point fixe de Banach.

3. **Vitesse et ordre de convergence.**

   (a) Quel est l'ordre de convergence de l'itération (6)?

   (b) Quelle est la vitesse de convergence (asymptotique) de l'itération (6)?

   (c) Quelle est la valeur optimale pour μ dans l'itération (6), en terme de vitesse (et d'ordre) de convergence?

4. **Méthode de Newton-Raphson.**

   (a) Montrer que l'itération Newton-Raphson pour résoudre le problème (5) s'écrit

$$x_{n+1} = \frac{2\left(x_n^3 - 3x_n^2 + 3\right)}{3\left(x_n - 2\right)^2} \tag{7}$$

   (b) L'itération (7) converge-t-elle localement? Si, oui, quel est l'ordre de cette convergence? Justifiez votre réponse

(c) Comment peut-on montrer numériquement, à partir des valeurs numériques x_n définies dans (7) et calculées en machine, que l'ordre de convergence réel (i.e. en machine) est bien (approximativement) celui prédit en théorie? Inspirez-vous du travail fait pendant les TP pour détailler votre réponse.

Exercice 6 En utilisant la formule de Taylor, imaginer un algorithme modifiant la méthode de Newton et assurant une convergence quadratique dans le cas d'une racine de multiplicité p en $x = s$ pour une application F, de classe $\mathcal{C}^2(\mathbb{R}, \mathbb{R})$.

Exercice 7 Etudier la convergence des méthodes de point fixe suivantes (on cherche à approcher $\sqrt{2}$). Appliquer une méthodes d'accélération de la convergence pour rendre ces algorithmes convergents quand ils ne convergent pas ou plus rapides quand ils convergent.

1. $x_{n+1} = 2/x_n$
2. $x_{n+1} = \frac{1}{2}(x_n + 2/x_n)$
3. $x_{n+1} = 2x_n - 2/x_n$
4. $x_{n+1} = \frac{1}{3}(2x_n + 2/x_n)$

Exercice 8 Pour les fonctions suivantes, trouver un algorithme de point fixe (autre que la méthode de Newton) qui converge vers la plus petite solution réelle positive de :

1. $x^3 - x - 1 = 0$
2. $x - \tan x = 0$
3. $e^{-x} - \cos x = 0$

Exercice 9 *Pour des explications supplémentaires concernant l'application de la méthode de Newton dans le cas vectoriel, voir annexe A ci-dessous*

1. Ecrire l'itération de point fixe correspondant à la méthode de Newton appliquée à la résolution du système :

$$\begin{cases} -5x + 2\sin x + 2\cos y = 0 \\ 2\cos x + 2\sin y - 5y = 0 \end{cases} \tag{8}$$

   et montrer que la suite ainsi construite est bien définie

2. Soit $s := (x^*, y^*)$ une solution de (8). Montrer qu'il existe une boule $B(s, \epsilon)$ (avec $\epsilon > 0$) telle que pour toute condition initiale $(x_0, y_0) \in B(s, \epsilon)$, la suite (x_n, y_n) construite au point précédent converge vers $s := (x^*, y^*)$. Que peut-on dire de l'ordre de cette convergence?

3. Montrer qu'il existe au moin une solution au problème (8)

Exercice 10 *Pour des explications supplémentaires concernant l'application de la méthode de Newton dans le cas vectoriel, voir annexe A ci-dessous*

1. Ecrire l'itération de point fixe correspondant à la méthode de Newton appliquée à la résolution du système :

$$\begin{cases} x^2 + 2xy = 0 \\ xy + 1 = 0 \end{cases} \tag{9}$$

2. calculer la solution exacte de (9)

3. Soit $s := (x^*, y^*)$ une solution de (9). Montrer qu'il existe une boule $B(s, \epsilon)$ (avec $\epsilon > 0$) telle que pour toute condition initiale $(x_0, y_0) \in B(s, \epsilon)$, la suite (x_n, y_n) construite au point précédent converge vers $s := (x^*, y^*)$. Que peut-on dire de l'ordre de cette convergence?

4. Montrer qu'il existe au moin une solution au problème (9)

Exercice 11 *Pour des explications supplémentaires concernant l'application de la méthode de Newton à l'évaluation de fonctions numériques avec une complexité (bit complexity)* easy*, voir annexe B ci-dessous*

1. La fonction ln (logarithme népérien) peut être évaluée avec une complexité *easy* à l'aide de la moyenne arithmético-géométrique de Gauss. Proposer une évaluation de la fonction exponentielle en vous appuyant sur la méthode de Newton.

2. On considère une bijection

$$\phi : \mathbb{R} \to \mathbb{R} : x \mapsto \phi(x) =: y$$

   continûment dérivable sur R et telle que $\phi(x)$ et $\phi^{-1}(y)$ peuvent être évaluées de manière *easy*, quels que soient $x, y \in \mathbb{R}$. Montrer que l'application

$$\phi^{-1} : \mathbb{R} \to \mathbb{R} : y \mapsto \phi^{-1}(y) =: x$$

   peut être évaluée de manière *easy* (pour toute valeur $y \in \mathbb{R}$).

3. Proposer un deuxième exemple d'application du résultat développé au point précédent

Annexe A : généralisation de la méthode de Newton au cas vectoriel

On considère une application

$$\begin{aligned} \boldsymbol{F} : U \subset \mathbb{R}^N &\to \mathbb{R}^N \\ \boldsymbol{x} &\mapsto \boldsymbol{F}(\boldsymbol{x}) \end{aligned}$$

définie et continûment différentiable dans un voisinage ouvert U d'une solution $\boldsymbol{s} := (s_1, s_2, \ldots, s_N)^T \in \mathbb{R}^N$ isolée du système de N équations à N inconnues $\boldsymbol{F}(\boldsymbol{s}) = \boldsymbol{0}$. On a donc :

$$\left\{ \begin{aligned} F_1 (s_1, s_2, \ldots, s_N) &= 0 \\ F_2 (s_1, s_2, \ldots, s_N) &= 0 \\ &\vdots \\ F_N (s_1, s_2, \ldots, s_N) &= 0 \end{aligned} \right. \tag{10}$$

Les N équations (10) sont supposées indépendantes (dans U) et la matrice jacobienne $\boldsymbol{J}(\boldsymbol{x})$ définie par

$$\boldsymbol{J}(\boldsymbol{x}) := \left. \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_N} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_N}{\partial x_1} & \frac{\partial F_N}{\partial x_2} & \cdots & \frac{\partial F_N}{\partial x_N} \end{pmatrix} \right|_{\boldsymbol{x} := (x_1, x_2, \ldots, x_N)^T} \tag{11}$$

est de plein rang, pour tout $\boldsymbol{x}$ dans U. Le développement de Taylor, à l'ordre 1, de la valeur $\boldsymbol{F}(\boldsymbol{s}) = \boldsymbol{0}$, au voisinage de l'estimation courante $\boldsymbol{x}^{[k]}$ (de rang k), dans U, donne

$$\boldsymbol{0} = \boldsymbol{F}(\boldsymbol{s}) = \boldsymbol{F}(\boldsymbol{x}^{[k]}) + \boldsymbol{J}(\boldsymbol{x}^{[k]}) \left(\boldsymbol{s} - \boldsymbol{x}^{[k]} \right) + o \left(\| \boldsymbol{s} - \boldsymbol{x}^{[k]} \| \right)$$

En négligeant dans ce développement les termes d'ordre supérieur à 1, on obtient à la place de $\boldsymbol{s}$ l'approximation $\boldsymbol{x}^{[k+1]}$ (de rang $k+1$) qui satisfait :

$$\boldsymbol{0} = \boldsymbol{F}(\boldsymbol{x}^{[k]}) + \boldsymbol{J}(\boldsymbol{x}^{[k]}) \left(\boldsymbol{x}^{[k+1]} - \boldsymbol{x}^{[k]} \right) \tag{12}$$

L'équation (12) peut-être résolue formellement pour trouver

$$\boldsymbol{x}^{[k+1]} = \boldsymbol{x}^{[k]} - \boldsymbol{J}^{-1}(\boldsymbol{x}^{[k]}) \boldsymbol{F}(\boldsymbol{x}^{[k]})$$

Cette dernière expression est l'équivalent formel de la formule de Newton-Raphson scalaire étendue au cas vectoriel. Dans le terme de correction, l'inverse de la matrice jacobienne au point courant (matrice de plein rang, par hypothèse) joue le même rôle que l'inverse de la dérivée au point courant dans le cas scalaire. En pratique, on écrit plutôt les itérations sous la forme :

$$\boldsymbol{x}^{[k+1]} = \boldsymbol{x}^{[k]} + \boldsymbol{\delta}^{[k]} \tag{13}$$

où le terme de correction $\delta^{[k]}$ est solution du système linéaire

$$\boldsymbol{J}(\boldsymbol{x}^{[k]})\boldsymbol{\delta}^{[k]} = \boldsymbol{F}(\boldsymbol{x}^{[k]}) \tag{14}$$

Le système linéaire (14) est en général lui aussi résolu en mettant en œuvre une méthode itérative[1] . Dans l'implémentation en machine des équations (13) et (14), la fonction $\boldsymbol{F}$ et la jacobienne $\boldsymbol{J}$ sont souvent définies explicitement au préalable (dans des scripts indépendants). Lorsque $\boldsymbol{J}$ n'est pas connue, elle peut être évaluée numériquement (avec les difficultés spécifiques liées à l'évaluation numérique des dérivées). Il arrive également que la jacobienne $\boldsymbol{J}(\boldsymbol{x}^{[k]})$ ne soit pas mise à jour à chaque itération k, voire ne soit pas mise à jour du tout (dans ce dernier cas, il s'agit de l'analogue vectoriel de la méthode des cordes parallèles).

Annexe B : application de la méthode de Newton au calcul easy

La complexité associée au calcul de la valeur numérique d'une fonction ou d'une constante peut être évalue en nombre d'opérations en virgule flottante (flops, pour floating point operations) ou en nombre de manipulations élémentaires de bits (*bit complexity*). On parle de complexité *easy* si le nombre de manipulations élémentaires de bits nécessaires pour trouver une évaluation numérique du résultat recherché avec n bits significatifs est en $\mathcal{O}\left(n^{1+\epsilon(n)}\right)$, avec

$$\lim_{n\to\infty} \epsilon(n) = 0$$

Il s'agit donc d'un niveau de complexité supérieur à la complexité linéaire, mais inférieure à toutes les complexités polynomiales (c'est en quelque sorte ce qu'on peut faire de mieux au dessus de la complexité linéaire). La multiplication de deux nombres de n bits, par exemple, est une opération dont la complexité est $M(n) = \mathcal{O}\left(n\log(n)\right)$ (en utilisant par exemple l'algorithme de transformée de Fourier rapide de Cooley et Tukey). Il s'agit donc d'une

[1]il y a donc deux itérations emboîtées : à chaque pas de l'itération Newton (13), une itération emboîtée est mise en oeuvre pour résoudre le système linéaire (14) . La mise en oeuvre d'une méthode itérative pour résoudre (14) est facilitée par le fait qu'une estimation initiale $\delta_0^{[k]} := 0$ est en général proche de la solution $\delta^{[k]}$ recherchée

opération *easy* car

$$\begin{aligned}\log\left(M(n)\right) &= \log(n) + \log\left(\log(n)\right) \\ &= \log(n)\left(1 + \frac{\log\left(\log(n)\right)}{\log(n)}\right) \\ &= \log\left(n^{1+\frac{\log(\log(n))}{\log(n)}}\right)\end{aligned}$$

Dès lors

$$M(n) = n^{1+\frac{\log(\log(n))}{\log(n)}} = n^{1+\epsilon(n)}$$

avec

$$\epsilon(n) = \frac{\log\left(\log(n)\right)}{\log(n)} \to 0$$

pour $n \to \infty$. La méthode de Newton permet l'évaluation numérique de certaines opérations ou fonctions avec une complexité *easy*. En prenant l'exemple de la division

$$z := a/b$$

avec $a, b > 0$, on peut écrire

$$z = a \times (1/b)$$

La multiplication est une opération *easy*. Le calcul de l'inverse de b l'est aussi en utilisant l'algorithme de Newton. En effet $1/b$ est la solution positive de l'équation

$$F(z) := 1/z - b = 0$$

L'itération de Newton pour résoudre cette équation est

$$z_{k+1} = z_k - \frac{F\left(z_k\right)}{F'\left(z_k\right)} = z_k\left(2 - bz_k\right)$$

Chaque itération comporte donc deux multiplications et une addition (soustraction). Etant donnée la convergence quadratique de la méthode de Newton, pour obtenir $n = 2^k$ bits significatifs, k itérations sont nécessaires, avec $k = \log_2(n)$. En évaluant les opérations à la précision minimale nécessaire à chacune des itérations[2], on obtient donc pour la complexité $D(n)$ de cet algorithme de division :

$$D(n) = D\left(\frac{n}{2}\right) + 2M\left(n\right) + A\left(n\right)$$

[2]il n'est pas nécessaire d'évaluer chacune des itérations à la précision maximale (finale) nécessaire

où $A(n)$ désigne la complexité (*bit complexity*) de l'addition de deux nombres de n bits (complexité linéaire, négligeable devant celle de la multiplication). En appliquant la formule précédente de manière récurrente, on obtient :

$$\begin{aligned} D(n) &\sim D\left(\frac{n}{2}\right) + 2M\left(n\right) \\ &\sim D(1) + 2\left(M(2^k) + M(2^{k-1}) + \ldots + M(2^1)\right) \\ &\sim 2\left(M(2^k) + M(2^{k-1}) + \ldots + M(2^1)\right) \end{aligned}$$

D'autre part

$$M(2^k) \sim C2^k \log\left(2^k\right) > 2C2^{k-1} \log\left(2^{k-1}\right) = 2 \cdot M(2^{k-1})$$

On obtient donc

$$\begin{aligned} D(n) &< 2M(n)\left(1 + \frac{1}{2} + \ldots + \frac{1}{2^{k-1}}\right) \\ &= 4M(n)\frac{2^k - 1}{2^k} \\ &= 4M(n)\frac{n-1}{n} \\ &\sim 4M(n) \end{aligned}$$

En utilisant l'algorithme de Newton, la complexité d'une division est donc essentiellement 4 fois la complexité d'une multiplication. Il s'agit donc d'une opération *easy* :

$$D(n) = \mathcal{O}\left(n \log(n)\right)$$

La même stratégie peut être appliquée à de nombreuses fonction. Par exemple, l'évaluation de la racine carrée

$$\sqrt{a}$$

avec $a > 0$, peut être réalisée en appliquant la méthode de Newton à la résolution de l'équation

$$F(z) := z^2 - a$$

On obtient alors l'itération

$$z_{k+1} = \frac{1}{2}\left(z_k + \frac{a}{z_k}\right)$$