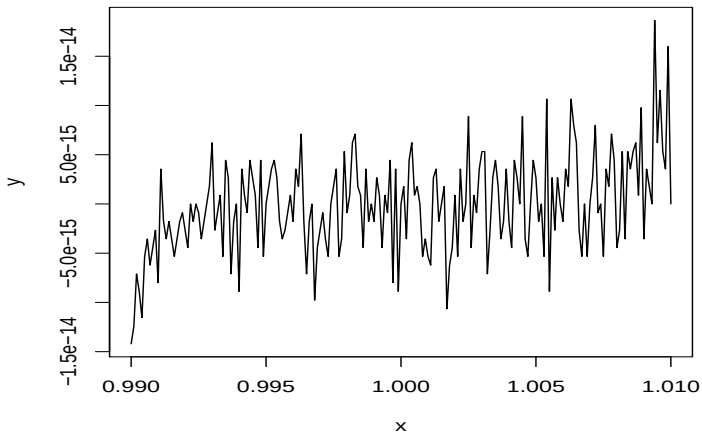
Université Paris I, Panthéon - Sorbonne

Licence M.I.A.S.H.S.

Cours de Méthodes Numériques

Jean-Marc Bardet (Université Paris 1, SAMM)



U.F.R. 27 et SAMM (Statistique, Analyse et Modélisation Multidisciplinaire)
Université Panthéon-Sorbonne, 90 rue de Tolbiac, 75013 Paris.

Plan du cours

Introduction

1. Les fondamentaux de l'utilisation d'un logiciel numérique de mathématiques
2. Quelques méthodes numériques en algèbre
3. Analyse numérique
4. Statistique et probabilités numériques

References

[1]

Documents accessibles librement sur internet

- Livre "Calcul Scientifique", A. Quarteroni, F. Saleri et P. Gervasio (Seconde édition, 2011), Springer.
- Polycopié de cours de Denis Pennequin (Université Paris 1)
- Polycopié de cours de T. Takahashi: `http://iecl.univ-lorraine.fr/ Takeo.Takahashi/polyAnaNum.pdf`

Quelques sites internet intéressants

- Le site de Toulouse III: `http://wwww.lsp.ups-tlse.fr`. Regarder les documents pédagogiques.
- Le site de Paris V: `http://wwww.math-info.univ-paris5.fr`. Regarder les documents pédagogiques.
- Le site de Paris VI: `http://wwww.proba.jussieu.fr`. Regarder les documents pédagogiques.
- Le site de la S.M.A.I.: `http://smai.emath.fr`. Regarder également la rubrique `Logiciels` dans laquelle de nombreux logiciels de mathématiques peuvent être téléchargés (en particulier, `Scilab` et `Mupad`).
- Le site français d'où l'on peut télécharger le logiciel R: `http://cran.cict.fr`.

Introduction

Dans ce cours, on verra par exemple:

1. Comment calculer l'inverse d'une matrice (1000×1000);
2. Comment faire pour approcher numériquement π ou pour donner une approximation d'un cosinus ou d'un logarithme d'un nombre quelconque;
3. Comment donner une valeur approchée à une intégrale non simplifiable théoriquement;
4. Comment créer du hasard à partir de procédés déterministes.
5. Comment exhiber la convergence d'un estimateur ou tracer une trajectoire d'une chaîne de Markov.

Plus fondamentalement:

- Les avancées des mathématiques ont toujours évolué en mêlant mathématiques pures et appliquées (exemple calcul de π, séries de Fourier);
- Avec la révolution informatique, tout un pan des mathématiques s'est développé en lien avec l'utilisation de logiciels de traitements numériques;
- Avec la révolution informatique, jamais il n'y a eu autant besoin de traitement numérique des données;
- Certains résultats théoriques ont pu être démontrés par l'utilisation d'un logiciel (Théorème des 4 couleurs par exemple), et même si elles ne démontrent pas, des expériences numériques permettent souvent de vérifier ou d'infirmer des hypothèses (à montrer ensuite);
- L'utilisation des méthodes numériques offre d'incroyables possibilités mais des limites structurelles existent, certains résultats ne se prêteront jamais à la numérisation, ce qui donne toute sa légitimité à continuer de faire des mathématiques d'autant que la tentation du presse-bouton sans aucune compréhension peut être très dangereuse.

1 Les fondamentaux de l'utilisation d'un logiciel numérique de mathématiques

1.1 Quelques éléments sur le fonctionnement d'un logiciel numérique

- Premiers essais: bouliers, machines à calculer (Pascal, Leibniz, Babbage), premiers ordinateurs (machine de Turing),... Désormais dans la partie Unité Arithmétique et Logique (UAL) des processeurs.
- Logique booléenne: décomposition en base $2 \Longrightarrow$ Addition de nombres.
- La base: le langage FORTRAN. Passage aux vecteurs et matrices.

1.2 Boucles, conditionnements et fonctions

En sus des commandes logiques (afficher un nombre, comparaison de 2 nombres), un logiciel de calcul numérique permet la mise en place de boucles et de conditionnements. Une boucle (commande `for`) permet de répéter une commande un nombre **fini** de fois, nombre défini à l'avance. Cela permet notamment de calculer tout ce qui est défini par récurrence.
On retiendra cependant qu'il faut éviter le plus possible les boucles pour les remplacer par des calculs directs sur des vecteurs ou des matrices.

Exemple.

Calcul de $\sum_{j=1}^{1000} 1/j^2$.

Les conditionnements (commandes `if` et `else`) permettent de traiter les éventualités, les cas et sous-cas. On peut également utiliser des boucles conditionnées (commande `while`) qui permettent de répéter des commandes jusqu'à l'obtention d'une certaine condition. Noter qu'une boucle conditionnée par une condition inatteignable peut conduire à programmes qui ne s'arrêtent jamais...
On peut enfin prédéfinir des fonctions dans un sens très général (pouvant notamment dépendre de variables qualitatives), que l'on pourra ensuite réutiliser dans des programmes. Cela permettra notamment de calculer cette fonction pour un certain nombre de valeurs, d'optimiser cette fonction, d'approcher son intégrale,...

A l'aide des boucles, conditionnements, fonctions,..., un logiciel de calcul numérique peut à peu près tout faire numériquement avec des possibilités totalement impensables pour un humain (par exemple inverser une matrice (1000×1000) en 1 seconde!). Il y a cependant 2 limites à sa puissance: le **temps de calcul** (inverser une matrice $(10^9 \times 10^9)$ est encore une gageure actuellement) et aussi établir des résultats dans lequel **l'infini** intervient comme par exemple obtenir la décomposition décimale infinie d'un nombre réel comme $\log 2$, ou bien calculer une limite, une dérivée,.... On remplacera tout ce qui a rapport à l'infini par des approximations quand n devient "très grand" ou x "très proche" de x_0, ce qui peut donner le plus souvent de bonnes indications du résultats, mais avec le risque également de commettre des erreurs! Il y aura donc encore du sens à **faire des mathématiques théoriques**, même en analyse ou probabilités et statistiques...

Exemple.

Déterminer numériquement la limite de $\sum_{n=2}^{\infty} \big(n \log(n)\big)\big)^{-1}$.

1.3 Format de nombres, erreurs et propagation des erreurs

Tout nombre utilisé par le logiciel est désormais représenté par un nombre fini de 0 et de 1. Cela entraîne que pour un nombre à développement décimal infini (comme certains rationnels, par exemple $10/7$ et tous les nombres réels non rationnels, comme $\sqrt{2}$), l'écriture du nombre est tronquée, c'est-à-dire que le développement décimal est arrêté après un certains nombre de chiffres (typiquement 16 en R). Il y a donc une erreur d'approximation, d'arrondi et cette erreur lorsqu'elle est multipliée ou répercutée dans une fonction ou une itération peut amener à des résultats faux.
Plus précisément, soit x un nombre réel. Si on prend un exemple comme $x = 1/3$, c'est-à-dire un nombre rationnel possédant une infinité de décimales et qu'on l'utilise avec un logiciel comme R, on obtient les résultats suivants:

```
> 1/3
[1] 0.3333333
> 1/3-0.3333333
[1] 3.333333e-08
> 1/3-0.3333333333333
[1] 3.330669e-14
> 1/3-0.3333333333333333
[1] 0
```

On s'aperçoit ainsi que l'affichage de x présente 7 chiffres après la virgule, mais qu'il en a mémorisé en fait 15. On observe également qu'il n'est pas capable d'aller plus loin. On retiendra ainsi d'une

manière générale qu'il y a le nombre réel x et qu'il y a sa représentation $r(x)$ par le logiciel, qui est une représentation tronquée de ce nombre.

Cette représentation est appelée **représentation à virgule flottante**. Plus précisément, tout x sera représenté sous une forme binaire (en base 2):

$$r(x) = \big(0.a_1a_2\cdots a_{53}\big)\cdot(-1)^s 2^t, \qquad \text{où}$$

- les a_i sont des 0 ou des 1, et $a_1 \neq 0$. On appelle **mantisse de** x le nombre $0.a_1a_2\cdots a_{53}$;
- $s = 0$ ou $s = 1$;
- t est un entier relatif appartenant à $\{-1021, \ldots, 1023\}$ (on remarque que $1024 = 2^{10}$).

Ainsi cela revient à écrire que $r(x) = (-1)^s 2^t \sum_{i=1}^{53} a_i 2^{-i}$. On remarquera ainsi que tout la représentation de tout nombre réel pour le logiciel, c'est une codification de $53 + 1 + 10$ 0 ou 1, ce sera donc au total 64 octets. En base 10, donc dans le système décimal usuel, cela revient à donner une précision en $2.2\,10^{-16}$ (qui équivaut à 2^{-52}), soit 15 chiffres significatifs, et un nombre compris appartenant forcément à $[-1.79769\,10^{308}, 1.79769\,10^{308}]$.

On peut se demander alors quelle erreur fait-on en remplaçant x par $r(x)$. Ainsi, on pourrait mesurer **l'erreur absolue** qui se définit par $\varepsilon(x) = |x - r(x)|$. Mais évidemment tout dépend de l'ordre de grandeur de x. Aussi préférera-t-on considérer **l'erreur relative** commise plutôt que l'erreur absolue, ce qui se définit par:

$$\varepsilon_r(x) = \Big|\frac{x - r(x)}{x}\Big|.$$

On peut facilement montrer alors que $\varepsilon_r(x) \leq 2^{-53} \simeq 1.1\,10^{-16}$.
On remarquera que du fait de la représentation à virgule flottante, il est possible d'avoir une très grande précision pour un nombre proche de 0. En effet, on peut aller jusqu'à définir $\pm 2^{-1074}$, soit environ $\pm 4.940656\,10^{-324}$. Ceci n'est plus du tout possible ailleurs qu'en 0, puisqu'alors la précision redevient $2.2\,10^{-16}$.

Le fait que l'erreur relative reste très petite, pourrait encourager à penser que la limitation causée par la représentation décimale est anecdotique. Pourtant, ce phénomène de **troncature** peut être la source d'erreurs conséquentes voire fatales. La preuve par l'exemple suivant:

Exemple.

Le but de l'exercice est de calculer une valeur approchée de : $\displaystyle\int_0^1 \frac{t^{20}}{10-t}dt$. *Pour ce faire, considérons la suite d'intégrales, pour* $n \geq 0$*:*

$$I_n = \int_0^1 \frac{t^{n+1}}{10-t}dt$$

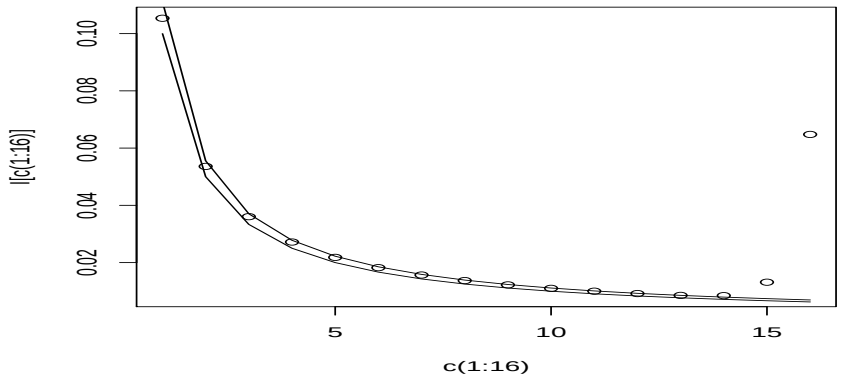
ce qui nous amènera à donner une valeur approchée de I_{21}*. Un calcul direct permet d'écrire que* $I_1 = \ln(10/9)$*, et la relation de récurrence suivante peut également être montrée pour tout* $n \in \mathbb{N}^*$*,*

$$10\,I_n - I_{n+1} - = \int_0^1 t^{n-1}dt = \frac{1}{n}, \quad \textit{soit} \quad I_{n+1} = 10\,I_n - \frac{1}{n}.$$

Par ailleurs, deux majorations simples de la fraction nous permettent d'écrire que:

$$\frac{1}{10\,n} < I_n < \frac{1}{9\,n} \quad \textit{pour tout } n \in \mathbb{N}^*.$$

Pourtant, si on utilise le logiciel R pour calculer I_{21} à partir de cette relation de récurrence, on obtient que $I_{21} \simeq 5780$ ce qui est très éloigné de l'encadrement ci-dessus. Plus en détail, la figure ci-dessous donne la valeur calculée par R, comparée à celles de l'intervalle $[\frac{1}{10\,n}\,,\,\frac{1}{9\,n}]$:



Comment expliquer ces importantes erreurs? Par le fait que d'une étape à l'autre on multiplie par 10 le résultat précédent, donc s'il y a une erreur dans la valeur initiale de la récurrence ($I_1 = \ln(10/9)$), alors le fait de multiplier par 10 le résultat précédent implique que tout le développement décimale de I_1 finit par être utilisé. Ainsi, si ce développement n'est pas exact jusqu'à l'ordre 20, alors des erreurs ont lieu et s'amplifient avec n. Or de rapides manipulations sur R montrent que le développement décimal de $\ln(10/9)$ n'est donné qu'à 10^{-11} près.
On peut également se rendre compte de ce fait, si on modifie un tout petit peu le valeur approchée de I_1: on a ainsi rajouté 10^{-15} à la valeur approchée de $\ln(10/9)$ calculée par R. On s'aperçoit alors, si on note (I'_n) la suite calculée à partir de I'_1 que $|I'_n - I_n|$ est de l'ordre de 10^{n-16}, donc pour le calcul de I_{21} on s'aperçoit que par l'algorithme utilisé, une erreur de 10^{-15} sur la valeur initiale de I_1 se répercute en une erreur de 10^5 sur I_{21}!
Une autre piste pour calculer I_n serait de poser $u = t - 10$ et d'appliquer la formule du binôme. On obtient ainsi, pour tout $n \geq 2$:

$$I_n = 10^{n-1}\left[\ln(10/9) + \sum_{k=1}^{n-1} \frac{(-1)^k}{k} \left(\begin{array}{c} n-1 \\ k \end{array} \right) \left(1 - (9/10)^k\right)\right].$$

Mais là encore, la multiplication par 10^{n-1} rend le résultat extrêmement dépendant de la décomposition décimale de $\log(10/9)$. Comment alors approcher la valeur de I_{21}? Deux pistes qui ne veulent plus un calcul exact (illusoire) mais se contente de calculs approchés: le développement en série entière de $(10-t)^{-1}$ et l'approximation directe de l'intégrale par la méthode de Simpson...

La propagation d'une erreur par une fonction peut parfois être contrôlée en utilisant un développement de Taylor-Lagrange. Ainsi si le vrai nombre réel considéré est x_0 mais que l'on dispose de $x_0 + \varepsilon$ au lieu de x_0, alors pour une fonction f de classe $\mathcal{C}^1$ dans un voisinage de x_0 on a:

$$\begin{array}{rcl} \big|f(x_0+\varepsilon) - f(x_0)\big| & = & \big|f'(\eta)\,\varepsilon\big| \\ & \leq & M_1\,|\varepsilon| \qquad \text{avec} \quad M_1 = \sup_{x\in\mathbb{R}} |f'(x)|. \end{array}$$

Si $M_1 \leq 1$, on voit que même en itérant les calculs avec f, alors l'erreur reste contenue. Mais lorsque M_1 devient grand (> 1), l'erreur commise peut très bien être amplifiée et lorsque l'on répète l'utilisation de f, le résultat peut-être très vite très différent de celui mathématiquement obtenu. On a pu s'apercevoir d'une telle incidence sur l'exemple précédent, puisque dans ce cas $I_{n+1} = f_n(I_n)$

avec $f_n(x) = 10\,x - 1/n$ et $f'(x) = 10 = M_1$ pour tout $x \in \mathbb{R}$: l'erreur est donc multipliée par 10 d'une étape à l'autre ce qui explique qu'elle prenne finalement le dessus sur la valeur de l'intégrale dès que $n \geq 15$.

Cette propagation des erreurs peut aussi se loger dans des cas qui semblent pourtant plus "contrôlés". Considérons l'exemple de la suite dite logistique:

Exemple.

Soit (u_n) *telle que*

$$u_0 = 0.6 \quad et \quad u_{n+1} = 4\,u_n(1 - u_n) \quad pour\ n \in \mathbb{N}.$$

Il est facile de voir que pour tout $n \in \mathbb{N}$, *dès que* $u_0 \in [0,1]$ *alors* $u_n \in [0,1]$ *pour tout* $n \in \mathbb{N}$. *Si on écrit* $u_{n+1} = f(u_n)$ *alors* $f(x) = 4\,x(1-x)$, *qui est clairement de classe* $\mathcal{C}^\infty$ *sur* $[0,1]$. *Mais* $f'(x) = 4 - 8x \in [-4,4]$ *et même* $|f'(x)| > 1$ *pour* $x \in [0, 3/8[\cup]5/8, 1]$. *Ainsi les erreurs ont tendance à s'amplifier pour la plupart des valeurs de* x *dans* $[0,1]$. *Ainsi, avec* $u_0 = 0.6$, *un calcul numérique amène à* $u_{50} = 0.020075845$, *mais si l'on choisit* $u_0 = 0.6 + 10^{-15}$, *on obtient* $u_{50} = 0.184015528$, *donc une différence conséquente entre les* 2 *termes d'ordre* 50 *alors que la différence initiale semblait anecdotique. Et ceci est reproductible pour tout premier terme choisi dans* $[0,1]$...

L'exemple précédent est assez difficile à traiter théoriquement même s'il paraît très simple.

La résolution de systèmes d'équations même linéaires et même peu nombreuses peut également être la source d'erreurs liées à la troncature des nombres réels par le logiciel. Considérons ainsi l'exemple suivant: supposons que A, matrice carrée d'ordre n inversible, soit telle que:

$$Y = A\,X$$

et que l'on veuille calculer Y alors que X est un vecteur de taille n connu. On ne disposera cependant que de $\widetilde{X} = X + \delta_X$, qui est par exemple $r(X)$ plutôt que X. Aussi obtiendra-t-on au final:

$$\widetilde{Y} = Y + \delta_Y = A\,\widetilde{X} = A\,(X + \delta_X) \quad \Longrightarrow \quad \delta_Y = A\,\delta_X.$$

L'erreur est ici vectorielle, aussi doit-on passer en norme pour la mesurer. On aimerait ainsi obtenir une majoration de $\|\delta_Y\|$ en fonction de $\|\delta_X\|$ et de A. Tout dépend de la norme choisie, mais il convient avant cela de définir une norme matricielle pour A:

Définition. *Pour* A *une matrice réelle d'ordre* n *et* $\|\cdot\|$ *une norme sur* $\mathbb{R}^n$, *on définit la norme* $\|A\|$ *par*

$$\|A\| = \sup_{x \in \mathbb{R}^n,\, x \neq 0_{\mathbb{R}^n}} \frac{\|A\,x\|}{\|x\|} = \sup_{u \in \mathbb{R}^n,\, \|u\|=1} \|A\,u\|.$$

L'égalité précédente est obtenue en posant $x = \frac{x}{\|x\|}\,\|x\|$. On peut alors considérer les 3 principales normes vectorielles dans $\mathbb{R}^n$. Notons ainsi pour $x = {}^t(x_1, \cdots, x_n)$:

$$\|x\|_1 = \sum_{i=1}^n |x_i|, \qquad \|x\|_2 = \Big(\sum_{i=1}^n |x_i|\Big)^{1/2} \quad \text{et} \quad \|x\|_\infty = \max_{1 \leq i \leq n} |x_i|.$$

Ainsi peut-on montrer les égalités suivantes:

Propriété. *Pour* $A = (a_{ij})_{1 \leq i,j \leq n}$ *une matrice réelle d'ordre* n *et les normes* $\|\cdot\|_i$ *définies précédemment sur* $\mathbb{R}^n$, *alors:*

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|, \qquad \|A\|_2 = \max_{\lambda \in Sp({}^tAA)} \lambda \quad et \quad \|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|.$$

Notons qu'il n'y a pas de relations d'ordre entre les $\|A\|_i$ (à voir avec des exemples...). Si on applique cela à notre problème d'incertitude de système linéaire, on obtient:

$$\|\delta_Y\| \leq \|A\| \, \|\delta_X\|,$$

avec possibilité d'avoir une égalité lorsque δ_X est choisi en particulier. On a donc une première majoration de l'erreur commise.

Exemple.

Pour $n \in \mathbb{N}^*$, on considère la matrice dite de Hilbert, définie par:

$$A = (a_{ij})_{1 \leq i,j \leq n} \qquad \text{avec} \qquad a_{ij} = \frac{1}{i+j-1} \qquad \text{pour } 1 \leq i, j \leq n.$$

Mais le plus souvent on aimerait plutôt connaître la précision de la solution d'un système linéaire. Ainsi, voudra-t-on plutôt déterminer le vecteur X tel que:

$$A\,X = B \qquad \text{où } B \text{ est un vecteur colonne quelconque.}$$

Si on suppose qu'il y a une imprécision sur B (et on sait qu'il y en a toujours une), et/ou sur A, alors il y aura une imprécision sur X. Ainsi peut-on écrire:

$$(A + \delta_A)(X + \delta_X) = B + \delta_B.$$

Nous traiterons ici uniquement le cas $\delta_A = 0$, ce dernier conduisant à des résultats plus techniques mais non significativement différents. Donc avec $\delta_A = 0$, on a:

$$\|\delta_X\| = \|A^{-1}\,\delta_B\| \quad \Longrightarrow \quad \frac{\|\delta_X\|}{\|X\|} = \frac{\|A^{-1}\,\delta_B\|}{\|X\|}$$

Or $\|B\| = \|A\,X\| \leq \|A\| \times \|X\|$ d'après l'inégalité usuelle des normes de matrices. D'où $\frac{\|A^{-1}\,\delta_B\|}{\|X\|} \leq \|A\| \, \frac{\|A^{-1}\,\delta_B\|}{\|B\|}$. Avec la même inégalité vérifiée par les normes de matrices, on a aussi $\|A^{-1}\,\delta_B\| \leq \|A^{-1}\| \times \|\delta_B\|$, d'où finalement:

$$\frac{\|\delta_X\|}{\|X\|} = \left(\|A\| \, \|A^{-1}\|\right) \frac{\|\delta_B\|}{\|B\|}$$

Aussi appelle-t-on **condionnement** d'une matrice inversible A le nombre

$$c_i(A) = \|A\|_i \, \|A^{-1}\|_i \quad \text{tel que} \quad \frac{\|\delta_X\|_i}{\|X\|_i} \leq c_i(A) \, \frac{\|\delta_B\|_i}{\|B\|_i},$$

et ceci pour $i = 1$, 2 ou ∞. Il vient d'après les propriétés des normes que $c_i(A) \geq 1$ pour tout A. Lorsque A est diagonalisable, il est également facile de voir que

$$c_2(A) = \frac{\lambda_{\max}}{\lambda_{\min}}.$$

Le conditionnement permet de mesurer la précision d'une résolution de système linéaire. Il est bien clair que lorsque ce nombre est "trop" grand (typiquement $\geq 10^{16}$) alors il y aura toujours une erreur relative conséquente sur la résolution du système (puisque l'erreur relative $\|\delta_B\|_i / \|B\|_i$ est toujours supérieure à $2\,10^{-16}$. C'est exactement ce qui se passe pour la matrice de Hilbert précédente, et ceci dès que $n \geq 12$ et le logiciel R ne permet plus d'obtenir directement l'inverse de la matrice.

D'une manière générale, il convient d'être très prudent lorsque des calculs sont itérés ou lorsque des systèmes linéaires sont résolus. Un calcul rapide du supremum de la dérivée de la fonction itérée ou du conditionnement de la matrice impliquée peut permettre d'éviter de prendre pour argent comptant des résultats faux. Un autre moyen en pratique pour vérifier la bonne marche du calcul effectué peut également être de perturber légèrement la condition initiale ou le second membre du système. Typiquement on peut lui rajouter une réalisation d'une variable aléatoire centrée ou d'une vecteur aléatoire centré de variance petite (par exemple 10^{-6}) et vérifier que la solution n'est que faiblement perturbée par ce bruit.

2 Quelques méthodes numériques en algèbre

Nous allons revenir sur de nombreux objets et résultats d'algèbre linéaire pour étudier leurs mises en place numériques. En particulier, nous examinerons tout ce que l'on peut faire à l'aide de matrices. Cela nous permettra de constater que ce domaine mathématique n'est pas si abstrait et a des applications nombreuses.

2.1 La méthode du pivot de Gauss

Le but initial de cette méthode, encore appelée méthode d'élimination de Gauss-Jordan, est la résolution de systèmes linéaires. Soit donc le système linéaire:

$$A\,X = B,$$

où $A = (a_{ij})_{1\leq i,j\leq n}$ est une matrice inversible de taille n, $B = (b_i)_{1\leq i\leq n}$ un vecteur colonne de taille n et X un vecteur colonne de taille n inconnu.
La méthode est alors bien connue:

Méthode du pivot de Gauss: on sélectionne une ligne i_0 de A dans laquelle $a_{i_0 1} \neq 0$ (au moins une de celles-ci existe puisque A est inversible). On permute alors les lignes i_0 et 1, donc les $a_{i_0 j}$ deviennent les $a^{(1)}_{1j}$, et les a_{1j} deviennent les $a^{(1)}_{i_0 j}$, les autres a_{ij} devenant des $a^{(1)}_{ij}$. Ensuite, on effectue les $(n-1)$ opérations: $L^{(1)}_j - \frac{a^{(1)}_{i1}}{a^{(1)}_{11}} L^{(1)}_1$, où $L^{(1)}_j$ désigne la j-ème ligne après la 1-ère permutation. Cela nécessite $2\times n$ multiplications, n soustractions pour chaque combinaison linéaire de lignes, donc $2n(n-1)$ opérations. On pourrait aussi évoquer l'examen d'au maximum n lignes pour trouver le pivot, donc on aura au maximum $n(2n-1)$ opérations.
On recommence le procédé sur les $n-1$ nouvelles lignes pour lesquelles tous les $a^{(2)}_{1j}$ sont nuls, ce qui amène $(n-1)(2n-3)$ opérations. On itère le procédé jusqu'à ce qu'il ne reste plus qu'une dernière ligne et un coefficient $a^{(n)}_{nn}$ qui reste le seul non nul.
On résout alors $a^{(n)}_{nn}\,x_n = b^{(n)}_n$ soit $x_n = \frac{b^{(n)}_n}{a^{(n)}_{nn}}$. Puis on remonte pour vers l'équation précédente $a^{(n-1)}_{(n-1),(n-1)} x_{n-1} + a^{(n-1)}_{(n-1),n} x_n = b^{(n-1)}_{n-1}$, ce qui permet d'obtenir x_{n-1}. On continue ainsi jusqu'à x_1: c'est **l'algorithme de remontée**. Celui-ci nécessite $(n+1-i)$ opérations pour obtenir x_i.

Au total, la méthode du pivot de Gauss nécessite:

$$\sum_{i=1}^{n} i(2i-1)+(n+1-i) = 2\,\frac{(2n+1)n(n+1)}{6}+n(n+1)-2\times\frac{1}{2}n(n+1) \sim \frac{2}{3}\,n^3 \quad \text{opérations au maximum.}$$

Cette méthode peut donc avantageusement être utilisée pour résoudre les systèmes linéaires précis. On verra cependant ci-dessous une méthode (la méthode LU) plus avantageuse pour mener de front

une résolution et l'obtention d'une matrice inverse.
Voyons cependant sur un exemple un écueil possible de cette méthode lorsque l'on ne fait pas attention aux erreurs d'arrondis.

Exemple.

Soit la matrice $A = \begin{pmatrix} 10^{-17} & 1 \\ 1 & 1 \end{pmatrix}$ *et le système* $AX = B \Longleftrightarrow \begin{cases} 10^{-17}x + y &= 1 \\ x + y &= 2 \end{cases}$. *Si on choisit comme premier pivot* $a_{11} = 10^{-17}$, *alors on obtient comme second système:*

$$\begin{cases} 10^{-17}x + y &= 1 \\ (10^{17} - 1)y &= (10^{17} - 2). \end{cases}$$

Or atteignant ici les limites d'arrondis du logiciel, le calcul effectué par celui-ci donnera $y = 1$ *et* $x = 0$. *Si on échange dès le départ la première et seconde ligne en prenant comme pivot* 1, *on obtient*

$$\begin{cases} x + y &= 2 \\ (1 - 10^{-17})y &= 1 - 2 \cdot 10^{-17}. \end{cases}$$

Atteignant les limites d'arrondis, on obtient encore $y = 1$, *puis* $x = 1$, *ce qui est beaucoup plus proche de la solution théorique* ($x \simeq 1 + 10^{-17}$ *et* $y \simeq 1 - 10^{-17}$).

De cet exemple, on s'aperçoit qu'il peut s'avérer crucial numériquement de bien choisir la suite de pivots par lesquels modifier les lignes. On adoptera ainsi le choix suivant:

Choix des pivots: A chaque étape, le pivot choisi sera celui dont la **valeur absolue sera la plus grande** ce qui évite de diviser par un pivot trop petit.

2.2 La décomposition LU

La méthode présentée ci-dessous La motivation de cette partie pourrait être la résolution de systèmes linéaires, mais nous verrons que bien d'autres résultats en découlent.
Õn commence par énoncer le résultat suivant:

Proposition. *Soit* $A \in \mathcal{M}_n(\mathbb{R})$, *ensemble des matrices carrés d'ordre* n *à coefficients réels. On suppose que* A *est inversible, ainsi que toutes les matrices* A_p *carrées extraites de* A, *composées des* p *premières lignes et colonnes de* A, *et ceci pour* $p = 1, \ldots, n-1$. *Alors il existe une infinité de manières d'écrire*

$$A = L\,U,$$

où L *et* U *sont des matrices de* $\mathcal{M}_n(\mathbb{R})$, *respectivement triangulaire inférieure (Low) et supérieure (Up). Si l'on spécifie que tous les éléments sur la diagonale de* L *sont des* 1, *alors cette décomposition devient unique.*

Proof. La preuve est assez "bestiale". On commence par écrire que $A = (a_{ij})_{1 \le i,j \le n}$, $L = (\ell_{ij})_{1 \le i,j \le n}$ avec $\ell_{ij} = 0$ pour $1 \le i < j \le n$ et $U = (u_{ij})_{1 \le i,j \le n}$ avec $u_{ij} = 0$ pour $1 \le j < i \le n$. Alors on a pour $1 \le i, j \le n$,

$$a_{ij} = \sum_{k=1}^{n} \ell_{ik} u_{kj} \quad \Longrightarrow \quad a_{ij} = \sum_{k=1}^{\min(i,j)} \ell_{ik} u_{kj}.$$

On a donc n^2 équations avec deux fois $n(n+1)/2$ inconnues (composantes de L et de U), soit $n^2 + n$ inconnues. Si on suppose que les $\ell_{ii} = 1$ pour tout $i = 1, \cdots, n$, il ne reste que n^2 inconnues.
Pour $i = 1$, on a $\ell_{1j} = 1$ si $j = 1$ et 0 si $j \ge 2$, et $u_{1j} = a_{1j}$ pour tout $j \in \{1, \cdots, n\}$.
Pour $i = 2$, on a $a_{21} = \ell_{21} u_{11}$, d'où $a_{21} = \ell_{21} a_{11}$, puis $a_{2j} = \ell_{21} u_{1j} + u_{2j}$ pour tout $j \in \{2, \cdots, n\}$. Comme $u_{1j} = a_{1j}$, on obtient que $u_{2j} = a_{2j} - \ell_{21} a_{1j}$. Comme $a_{11} \neq 0$, puisque l'on a supposé non nuls les déterminants des matrices extraites principales, on en déduit facilement ℓ_{21} et les u_{2j} pour tout $j \in \{2, \cdots, n\}$. Comme on a supposé que

$a_{11} \neq 0$ alors on obtient explicitement $\ell_{21} = a_{21}/a_{11}$ et $u_{2j} = a_{2j} - \frac{a_{21}}{a_{11}} a_{1j}$ pour $j \in \{2, \cdots, n\}$.
Pour $i \geq 3$, on a $a_{i1} = \ell_{i1}u_{11}$, $a_{i2} = \ell_{i1}u_{12} + \ell_{i2}u_{22}$, ..., $a_{i,(i-1)} = \ell_{i1}u_{1,(i-1)} + \ell_{i2}u_{2,(i-1)} + \cdots + \ell_{i,(i-1)}u_{(i-1),(i-1)}$ et $a_{i,j} = \ell_{i1}u_{1j} + \ell_{i2}u_{2j} + \cdots + \ell_{i,(i-1)}u_{(i-1),j} + u_{ij}$ pour tout $j \geq i$. D'où $\ell_{i1} = a_{i1}/u_{11}$, $\ell_{i2} = (a_{i2} - \ell_{i1}u_{12})/u_{22}$, ..., $\ell_{i,(i-1)} = (a_{i,(i-1)} - \ell_{i1}u_{1,(i-1)} - \ell_{i2}u_{2,(i-1)} - \cdots - \ell_{i,(i-2)}u_{(i-1),(i-2)})/u_{(i-1),(i-1)}$ et $u_{ij} = a_{i,j} - \ell_{i1}u_{1j} - \ell_{i2}u_{2j} - \cdots - \ell_{i,(i-1)}u_{(i-1),j}$ pour tout $j \geq i$. On peut ainsi par itération arriver à montrer l'obtention des composantes ℓ_{ij} et u_{ij}. □

Intéressons nous au nombre d'opérations élémentaires (additions ou multiplications) nécessaires à la décomposition LU. Pour $i = 1$, il n'y a pas d'opérations. Pour $i = 2$, il y a 1 opération pour les ℓ_{2j} et $2(n-1)$ opérations pour les u_{2j}. Pour $i \geq 3$, il y a $1 + 3 + \cdots + (2i-1) = 1 + i(i-1)$ opérations pour les ℓ_{ij} et $2(n-i+1) + 2(n-i) + \cdots + 2 = (n-i+1)(n-i+2)$ opérations pour les u_{ij}. Au total, on aura donc:

$$\begin{aligned} \sum_{i=1}^{n} 1 + i(i-1) + (n-i+1)(n-i+2) &= \frac{(2n+1)(n+1)n}{6} + \frac{(2n+3)(n+2)(n+1)}{6} \\ &\simeq \frac{2}{3}n^3 \quad \text{opérations élémentaires.} \end{aligned}$$

C'est donc un nombre d'opérations comparable à celui de l'algorithme du pivot de Gauss.

A quoi sert une telle décomposition? Une première utilisation réside dans la résolution de systèmes d'équations linéaires. En effet,

$$A\,X = B \Longleftrightarrow L\,U\,X = B \Longleftrightarrow L\,U\,X = B \Longleftrightarrow L\,Y = B \quad \text{et} \quad U\,X = Y.$$

On a donc deux systèmes d'équations linéaires plutôt qu'un! Mais ce sont des systèmes avec des matrices triangulaires, donc des systèmes très simples à résoudre. Ainsi le système $L\,Y = B$, permet d'écrire:

$$\left\{ \begin{array}{lcl} y_1 & = & b_1 \\ \ell_{21}y_1 + y_2 & = & b_2 \\ \vdots \quad \vdots & = & \vdots \\ \ell_{n1}y_1 + \ell_{n2}y_2 + \cdots + y_n & = & b_n \end{array} \right.$$

Aussi cette résolution demande $2 + 4 + \cdots + 2(n-1) = n(n-1)$ opérations pour obtenir Y. Il faut ensuite déterminer la solution de $U\,X = Y$, ce qui s'écrit:

$$\left\{ \begin{array}{rcl} u_{11}x_1 + u_{12}x_2 + \cdots + u_{1n}x_n & = & y_1 \\ u_{22}x_2 + \cdots + u_{2n}x_n & = & y_2 \\ \vdots \quad \vdots & = & \vdots \\ u_{nn}x_n & = & y_n. \end{array} \right.$$

Il faudra ainsi effectuer pour obtenir les x_i au maximum $1 + 3 + \cdots + (2n-1) = n^2$ opérations.
Ces deux étapes sont donc nettement moins coûteuse en temps que celle de la décomposition LU. On a donc une seconde méthode de résolution de système d'équations linéaires, après le pivot de Gauss, qui nécessite de l'ordre de $2n^3/3$ opérations. Cependant cette seconde méthode sera privilégiée par les logiciels, comme R, car elle offrira plus de possibilités.
Pourtant, l'hypothèse nécessaire à la décomposition LU (condition portant sur l'inversibilité des matrices principales) est trop restrictive. Pour lever cette condition et obtenir une décomposition beaucoup plus générale on peut écrire la proposition suivante:

Proposition. *Soit $A \in \mathcal{M}_n(\mathbb{R})$ telle que A soit inversible. Alors il existe une permutation des lignes de A transformant A en $P\,A$, où P est une matrice de permutation, c'est-à-dire une transformation*

de la matrice identité dont on a échangé les colonnes de la même manière qu'on échange les lignes de A, et telle que

$$P\,A = L\,U.$$

où L et U sont des matrices triangulaire respectivement inférieure et supérieure, les éléments sur la diagonale de L étant des 1.

Une fois cette permutation obtenue, qui est une application bijective, on voit qu'il revient désormais de résoudre:

$$P\,A = P\,B \quad \Longleftrightarrow \quad L\,U = P\,B.$$

Choix de la permutation: A chaque étape, on choisira la ligne dont le terme par lequel on divise (par exemple a_{11} pour commencer), c'est-à-dire le pivot, est de **valeur absolue maximale.**

Exemple.
Effectuer la décomposition LU de la matrice $A = \begin{pmatrix} 1 & 1-\varepsilon & 3 \\ 2 & 2 & 2 \\ 3 & 6 & 4 \end{pmatrix}$ *avec $\varepsilon = 1$, puis avec $\varepsilon = 0$ et enfin avec $\varepsilon = 10^{-17}$.*

2.3 Décomposition de Cholesky

On va se limiter ici à un cadre bien plus restreint: on va résoudre un système $A\,X = B$ dans lequel A est une matrice réelle symétrique définie positive, c'est-à-dire que ${}^tA = A$ (ce qui entraîne que A est diagonalisable) et toutes les valeurs propres de A sont strictement positives (ce qui équivaut au fait que pour tout vecteur colonne Z non nul, ${}^tZ\,A\,Z > 0$). On a alors la décomposition suivante:

Proposition. *Soit A une matrice réelle symétrique définie positive. Alors il existe une matrice triangulaire inférieure R telle que:*

$$A = R\,{}^tR: \quad \textit{c'est la décomposition dite de Cholesky.}$$

Proof. On reprend la même preuve que celle de la décomposition LU mais les conditions de symétrie simplifient et limitent les équations. □

En reprenant les étapes de cette preuve, on peut montrer que cette décomposition demande plutôt $\simeq n^3/3$, soit grossièrement 2 fois moins de temps de calcul que la décomposition LU.

Remarque: En sus des mêmes utilisations que la décomposition LU, la décomposition de Cholesky fournit également directement l'obtention d'une "racine" de la matrice A: c'est la matrice R. Cela peut être très intéressant pour simuler un vecteur gaussien non dégénéré de matrice de covariance Σ qui est une matrice définie positive. Si l'on a obtenu la décomposition de Cholesky $\Sigma = R\,{}^tR$, il suffira alors de considérer $R\,Z$, où Z est un vecteur gaussien standard, c'est-à-dire de matrice de covariance l'identité (dont toutes les composantes sont de variance 1 et indépendantes les unes les autres).

2.4 Inverse de matrice et calcul de déterminant

Inverse de matrice

La décomposition LU fournit directement l'inverse de n'importe quelle matrice. En effet, si on suppose que $A\,X = Y$ soit $X = A^{-1}\,Y$, alors on a $L\,U\,X = P\,B$, d'où $X = U^{-1}\,L^{-1}\,P\,B$ et donc la matrice A^{-1} est telle que:

$$A^{-1} = U^{-1}\,L^{-1}\,P.$$

Il est clair que l'inverse d'une matrice calculée ainsi nécessite un nombre d'opérations élémentaires en $2n^3/3$. C'est évidemment beaucoup moins qu'une technique basée sur la transposée de la comatrice qui est en n..

Calcul de déterminant

Le calcul d'un déterminant par la formule avec les permutations, ou bien la formule avec le développement suivant une rangée amène un nombre d'opérations élémentaires de l'ordre de n, donc plus que prohibitif dès que $n \geq 10$ où $n \simeq 3\,10^6$. La décomposition LU amène très naturellement un calcul immédiat du déterminant de A:

$$\det(P\,A) = (-1)^p \det(A) = \det(L\,U) = \det(L)\,\det(U) = \prod_{i=1}^{n} u_{ii},$$

où p est le nombre de permutations effectuées pour permettre la décomposition LU. Le calcul du déterminant ne requiert donc une nouvelle fois qu'environ $2n^3/3$ opérations élémentaires, ce qui est sans commune mesure avec les autres méthodes de calcul évoquées...

2.5 Détermination de valeurs propres et vecteurs propres

Pour déterminer les valeurs et vecteurs propres d'une matrice, une idée simple pourrait être de prendre comme fonction le polynôme caractéristique qui est un déterminant, puis de chercher les zéros d'une telle fonction, ce que nous saurons faire dans le chapitre suivant. Mais il existe une méthode, celle employée par la fonction `eigen` de R, qui est bien plus intéressante numériquement et qui se base sur la décomposition suivante:

Proposition (Décomposition QR). *Soit A une matrice de $\mathcal{M}_n(\mathbb{R})$. Alors il existe une matrice Q de $\mathcal{M}_n(\mathbb{R})$ unitaire, donc telle que ${}^tQ\,Q$ soit une matrice nulle avec des 1 ou des 0 sur la diagonale, et une matrice triangulaire supérieure R de $\mathcal{M}_n(\mathbb{R})$ telle que:*

$$A = Q\,R.$$

Proof. On utilise une méthode similaire au procédé d'orthonormalisation de Gram-Schmidt, qui permet de construire une base orthonormale à partir d'une base quelconque. On écrit que $A = [a_1, a_2, \cdots, a_n]$, où les a_i sont des vecteurs colonnes. On définit alors $u_1 = a_1$, puis $u_i = a_i - \sum_{j=1}^{i-1} P_{<a_j>}(a_i)$ pour $j = 2, \cdots, n$, où $P_{<a_j>}(x)$ est la projection orthogonale (par rapport au produit scalaire euclidien classique sur $\mathbb{R}^n$) sur le sous-espace vectoriel engendré par a_j du vecteur x. Par convention, si $a_j = 0$, alors $P_{<a_j>}(x) = 0$. Sinon, on montre facilement que $P_{<a_j>}(x) = \frac{<x,a_j>}{<a_j,a_j>}$. Aussi a-t-on $u_i = P_{<a_1,\cdots,a_{i-1}>^\perp}(a_i)$, donc u_i orthogonal à $a_1, \cdots, a_{i-1}$, donc également aux $u_1, \cdots, u_{i-1}$. Pour $u_i \neq 0$, soit $e_i = \frac{u_i}{\|u_i\|}$, sinon $e_i = 0$, et notons

$$Q = [e_1, e_2, \cdots, e_n].$$

Les $(e_1, \cdots, e_n)$ forment une famille orthonormale de $\mathbb{R}^n$. Il est clair que chaque a_i s'écrit en fonction des e_j pour $1 \leq j \leq i$ et on a plus précisément $a_i = \sum_{j=1}^{i} < e_j, a_i > e_j$. Donc on chosit:

$$R = (r_{ij})_{1 \leq i,j \leq n} \qquad \text{et} \qquad r_{ij} = 0 \quad \text{si } i > j \quad \text{et} \quad r_{ij} =< e_j, a_i > \quad \text{pour } 1 \leq i \leq j \leq n.$$

□

Exemple.

Déterminer la décomposition QR de la matrice $A = \begin{pmatrix} -3 & 2 \\ 2 & -1 \end{pmatrix}$, puis de ${}^tA\,A$.

On va se servir de cette décomposition pour pouvoir approcher d'aussi près que l'on veut les valeurs propres d'une matrice, sous certaines conditions.

Proposition. *Soit A une matrice réelle possédant n valeurs propres réelles distinctes et non nulles (donc A est inversible). Soit la suite de décompositions QR construites à partir de A:*

$$A_1 = A = Q_1 R_1, \qquad A_{k+1} = R_k Q_k \quad \textit{pour } k \in \mathbb{N}^*.$$

Alors la suite de matrices (A_k) converge vers une matrice triangulaire supérieure dont la diagonale est constituée des valeurs propres de A. Si de plus A est symétrique, alors la limite est même une matrice diagonale.

Proof. On ne va démontrer cette propriété que dans le cas $n = 2$. □

3 Analyse numérique

3.1 Approximation de limites de suites

Les suites peuvent être utilisées pour approcher des nombres qui s'avèrent être leurs limites. Pour ce faire, il sera important au préalable de montrer la convergence des suites considérées. Il y a deux cadres alors à prendre en compte:

Proposition. *Soit $(u_n)_{n\in\mathbb{N}}$ une suite de nombres réels. alors:*

- *S'il existe une fonction f telle que $u_n = f(n)$ alors l'étude de la convergence de (u_n) est facile: elle se réduit à la convergence de f en $+\infty$. Ce sont rarement de telles suites qui vont nous servir...*

- *S'il existe une fonction f telle que $u_{n+1} = f(u_n)$, alors (u_n) est définie par récurrence. Par suite (u_n) sera convergente si on arrive à montrer que (u_n) est bornée (ce qui revient à montrer que f est bornée) et que (u_n) est monotone (notamment si f est croissante), ou bien si f est de classe $\mathcal{C}^1$, si $u_0 \in I$ et $f(I) \subset I$ avec I un compact et si $\sup_{x\in I} |f'(x)| \leq 1$. D'une manière générale, si f est convergente, alors sa limite est unique et vérifie $f(\ell) = \ell$ (point fixe).*

Ce sont donc surtout les suites récurrentes qui vont nous permettre d'approcher des nombres réels, non directement calculables à partir des opérations élémentaires:

Exemple.

Soit $u_0 = 1$ et $u_{n+1} = (u_n + 2)/(u_n + 1)$ pour $n \geq 0$. On peut choisir $I = [0, 2]$. Alors (u_n) converge vers ℓ avec $\ell(\ell + 1) = \ell + 2$ soit $\ell^2 = 2$. Comme sur $[0, 2]$, $\sup_{x\in I} |f'(x)| = 1$, donc (u_n) converge vers $\sqrt{2}$.
Comment alors mesurer la vitesse de convergence vers ℓ pour une suite récurrente convergente (u_n)? On peut souvent utiliser le théorème des accroissements finis de la manière suivante:

$$\begin{aligned} \left|u_{n+1} - \ell\right| &\leq \left|f(u_n) - f(\ell)\right| \\ &\leq \sup_{x\in I} |f'(x)| \left|u_n - \ell\right| \\ &\Longrightarrow \qquad \left|u_n - \ell\right| \leq M_1^n \left|u_0 - \ell\right| \qquad \text{avec} \quad M_1 = \sup_{x\in I} |f'(x)|. \end{aligned}$$

Si $M_1 < 1$, on a donc une vitesse exponentielle de convergence vers ℓ, puisque $M_1^n = e^{n \ln(M_1)}$. Le but sera alors de prendre I suffisamment petit, et à l'extrême limite $M_1 \simeq f'(\ell)$ lorsque l'on ressert I autour de ℓ.

3.2 Calcul de séries

Les séries peuvent être également intéressantes pour calculer des nombres (séries numériques) ou des fonctions (séries entières, séries de Fourier,...). Evidemment dans le cadre d'un logiciel numérique, tout revient à calculer des séries numériques. Revoyons déjà les réusltats principaux de convergence des séries:

Proposition. *Soit $\sum_n u_n$ une série numérique et $S_n = \sum_{k=0}^n u_k$ sa somme partielle. Alors:*

- $\sum |u_n| < \infty$ *(série absolument convergente)* $\Longrightarrow \sum u_n$ *converge.*
- *Si (u_n) et (v_n) sont deux suites telles que $|u_n| \sim |v_n|$ quand $n \to \infty$ alors $\sum |u_n| < \infty \Longleftrightarrow \sum |v_n| < \infty$. De même en remplaçant $<$ par $=$.*
- *Si (u_n) et (v_n) sont deux suites telles que $|u_n| \leq |v_n|$ quand $n \geq n_0$ alors $\sum |v_n| < \infty \Longrightarrow \sum |u_n| < \infty$ et $\sum |u_n| = \infty \Longrightarrow \sum |v_n| = \infty$.*
- *Si $|u_n| = f(n)$ et f décroissante vers 0, alors $\sum |u_n|$ a même nature que $\int_1^\infty f(x)dx$. Ainsi $\sum \frac{1}{n^\alpha \ln^\beta n}$ converge si et seulement si $\alpha > 1$ ou $\alpha = 1$ et $\beta > 1$.*
- *Si $u_n = (-1)^n a_n$ avec (a_n) suite positive décroissante tendant vers 0, alors $\sum (-1)^n a_n$ converge.*
- *Si $u_n = a_n\, b_n$ avec $(A_n) = (\sum_{k=0}^n a_k)_n$ suite bornée et (b_n) suite positive, décroissante et tendant vers 0, alors $\sum u_n = \sum a_n\, b_n$ converge.*

Ces résultats théoriques sont importants pour déterminer la convergence ou la divergence d'une série numérique. Cependant, une fois la convergence assurée, il sera intéressant de disposer de résultats permettant de mesurer la distance entre une somme partielle, seule quantité obtensible par le logiciel, et la limite. On dispose alors de deux principaux résultats:

Proposition. *Soit $\sum_n u_n$ une série numérique convergente, $S_n = \sum_{k=0}^n u_k$ sa somme partielle et $R_n = \sum_{k=n+1}^\infty u_k$:*

- *On suppose que $u_n = (-1)^n a_n$ avec (a_n) suite positive, décroissante et tendant vers 0. Alors $|R_n| \leq a_{n+1}$ pour tout $n \in \mathbb{N}$.*
- *On suppose (u_n) est une suite positive tel que $u_n = f(n)$ avec f décroissante. Alors $|R_n| \leq \int_n^\infty f(x)dx$.*

Grâce à ces résultats, on peut mesurer une majoration de l'erreur commise en arrêtant la série à un rang n plutôt qu'en allant à la limite ∞. On pourra également comparer deux séries numériques permettant d'approcher un nombre par le nombre de termes nécessaires à calculer pour obtenir une approximation d'un même ordre.

Exemple.

On peut montrer que $\sum_{k=0}^\infty (-1)^k \frac{1}{k+1} = \ln 2$. Grâce au résultat précédent, on peut ainsi montrer que $\left|\ln 2 - \sum_{k=0}^n (-1)^k \frac{1}{k+1}\right| \leq \frac{1}{n+2}$. Par conséquent, il "suffit" de calculer 10^6 termes, pour obtenir une approximation à 10^{-6} près. Mais en fait, cela n'est pas une très bonne méthode pour approcher $\ln 2$ puisqu'il est impossible d'obtenir numériquement une approximation à 10^{-15} près (temps de calcul totalement prohibitif!).

3.3 Résolution de l'équation $f(x) = 0$

De nombreux problèmes de mathématiques reviennent à résoudre à trouver une ou des solutions à une équation de type $f(x) = 0$, où f est une fonction connue. On pourra citer par exemple:

- Trouver les racines d'un polynôme;
- Résoudre une équation de type $f(x) = g(x)$, par exemple trouver un point fixe tel que $f(x) = x$;
- Déterminer un tableau de variations d'une fonction en regardant les changements de signe de sa dérivée première, seconde ou plus;
- Optimiser une fonction en trouvant les points critiques de son gradient;
- Maximiser une vraisemblance.

Une méthode simple pour résoudre une telle équation $f(x) = 0$, lorsque la fonction f est continue, consiste à procéder par découpage itératif d'intervalles, ce qui s'appelle une **dichotomie**.

Proposition (Méthode par dichotomie). *Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction continue sur un voisinage de x_0, unique solution de l'équation $f(x_0) = 0$ dans ce voisinage. On suppose qu'il existe $[a_0, b_0]$ dans ce voisinage tel que $x_0 \in [a_0, b_0]$ et $f(a_0)f(b_0) < 0$. On définit alors les suites (a_n) et (b_n) telles que:*

$$a_{n+1} = \frac{1}{2}(a_n + b_n) \quad b_{n+1} = b_n \quad si \quad f(a_n)f\big(\frac{1}{2}(a_n + b_n)\big) \geq 0$$
$$b_{n+1} = \frac{1}{2}(a_n + b_n) \quad a_{n+1} = a_n \quad si \quad f(b_n)f\big(\frac{1}{2}(a_n + b_n)\big) \geq 0$$

En gros on se rapproche de plus en plus de x_0 en découpant en 2 chaque intervalle précédent. Ainsi à l'étape n, la taille de l'intervalle est $b_n - a_n = 2^{-n}(b_0 - a_0)$, ce qui mesure aussi la qualité de l'approximation de x_0. La méthode, bien que basique, est finalement assez rapide puisque si $b - a$ est de l'ordre de l'unité, en une cinquantaine d'itérations on a une approximation à 10^{-15} près.

Exemple.

On veut trouver une approximation de $\sqrt{2}$ grâce à $f(x) = x^2 - 2$ et $a_0 = 1$, $b_0 = 2$.

Mais d'autres méthodes permettent souvent de converger bien plus vite vers la racine d'une fonction, lorsque la fonction f est dérivable. Il s'agit d'abord, car elle est plus intuitive, la **Méthode de la sécante**. On se place dans le cadre précédent, c'est-à-dire que l'on suppose qu'il existe $u_0 < u_1$ tel que $x_0 \in]u_0, u_1[$ soit l'unique solution de $f(x_0) = 0$ dans $]u_0, u_1[$ (donc $f(u_0)f(u_1) < 0$). On définit u_2 de la manière suivante: u_2 est le point d'intersection entre la droite joignant $(u_0, f(u_0))$ et $(u_1, f(u_1))$ et l'axe des abscisses (faire un dessin!). D'où le nom de méthode de la sécante... Montrer qu'alors $u_2 = u_1 - f(u_1)\frac{u_1 - u_0}{f(u_1) - f(u_0)}$.
On peut généraliser cette démarche et ainsi définir la suite $(u_n)_n$ telle que

$$u_{n+1} = u_n - f(u_n)\frac{u_n - u_{n-1}}{f(u_n) - f(u_{n-1})} \quad \text{pour } n \geq 1.$$

Proposition (Méthode de la sécante). *Si f est de classe $\mathcal{C}^2$ sur $[u_0, u_1]$, la qualité de l'approximation est donnée par $|u_n - x_0| \leq \frac{C}{\gamma}\exp(-K\,\phi^n)$ lorsque $K = -\frac{1}{2+\phi}(\ln(C|u_0 - x_0|) + \phi\,\ln(C|u_1 - x_0|)) > 0$, avec $\phi = (\sqrt{5}+1)/2$ (le nombre d'or), $C = \frac{1}{2}M_2/m_1$ où $m_1 = \inf_{x\in[u_0,u_1]}|f'(x)|$ et $M_2 = \sup_{x\in[u_0,u_1]}|f''(x)|$.*

Proof. En utilisant le fait que $f(x_0) = 0$ (que l'on ne remplace pas partout), on peut facilement vérifier que:

$$|u_{n+1} - x_0| = |u_n - x_0|\,|u_{n-1} - x_0|\,\frac{\frac{f(u_n)-f(x_0)}{u_n-x_0} - \frac{f(u_{n-1})-f(x_0)}{u_{n-1}-x_0}}{f(u_n) - f(u_{n-1})}.$$

Ainsi, si l'on note $g(x) = \frac{f(x)-f(x_0)}{x-x_0}$, en utilisant le Théorème des Accroissements Finis, on obtient que:

$$\frac{\frac{f(u_n)-f(x_0)}{u_n-x_0} - \frac{f(u_{n-1})-f(x_0)}{u_{n-1}-x_0}}{u_n - u_{n-1}} = g'(\xi),$$

avec $\xi \in [u_{n-1}, u_n]$. Une dérivation simple montre que:

$$g'(x) = \frac{(x-x_0)f'(x) - (f(x) - f(x_0))}{(x-x_0)^2}.$$

Mais en utilisant la formule de Taylor-Lagrange d'ordre 2 de $f(x_0)$ en x, on obtient qu'il existe $\omega \in [x_0, x]$ tel que: $f(x_0) - f(x) = (x_0 - x)f'(x) + \frac{1}{2}(x - x_0)^2 f''(\omega)$. En remplaçant dans la formule de $g'(x)$, on obtient donc que:

$$g'(\xi) = \frac{1}{2}\,f''(\omega) \quad \text{avec } \omega \in [\xi, x_0].$$

Par conséquent, on obtient que

$$\frac{\frac{f(u_n)-f(x_0)}{u_n-x_0} - \frac{f(u_{n-1})-f(x_0)}{u_{n-1}-x_0}}{u_n - u_{n-1}} = \frac{1}{2}\,f''(\omega).$$

Comme, toujours d'après le Théorème des Accroissements Finis, il existe $\mu \in [u_{n-1}, u_n]$ tel que $f(u_n) - f(u_{n-1}) = f'(\mu)(u_n - u_{n-1})$, on en déduit donc que:

$$\frac{\frac{f(u_n)-f(x_0)}{u_n-x_0} - \frac{f(u_{n-1})-f(x_0)}{u_{n-1}-x_0}}{f(u_n) - f(u_{n-1})} = \frac{1}{2}\,\frac{f''(\omega)}{f'(\mu)}$$
$$\Longrightarrow \quad |u_{n+1} - x_0| \le C\,|u_n - x_0|\,|u_{n-1} - x_0|.$$

Si on multiplie l'inégalité précédente par C, en prenant le logarithme et en notant $\delta_n = \ln\big(C\,|u_n - x_0|\big)$, on obtient que:

$$\delta_{n+1} \le \delta_n + \delta_{n-1} \quad \text{pour } n \ge 1.$$

Pour espérer que (δ_n) tende vers $-\infty$, ce qui impliquerait que (u_n) tend vers x_1, on doit donc choisir u_0 et u_1 de telle manière que δ_0 ou δ_1 soit négatif. Plus précisément, si on note (δ'_n) telle que $\delta'_0 = \delta_0$, $\delta'_1 = \delta_1$ et $\delta'_{n+1} = \delta'_n + \delta'_{n-1}$ pour $n \ge 1$, alors $\delta_n \le \delta'_n$ pour tout $n \ge 0$. Mais la suite (δ'_n) est une suite de Fibonacci, ce qui implique qu'il existe λ_1 et λ_2 tels que $\delta'_n = \lambda_1\,\phi^n + \lambda_2\,(-\phi)^{-n}$ avec $\phi = (\sqrt{5}+1)/2$ le nombre d'or, qui vérifie $\phi^2 = \phi + 1$. Comme $\phi^n \underset{n\to+\infty}{\longrightarrow} \infty$ et $(-\phi)^{-n} \underset{n\to+\infty}{\longrightarrow} 0$, on veut nécessairement avoir $\lambda_1 < 0$. Or on a $\delta'_0 = \delta_0 = \lambda_1 + \lambda_2$ et $\delta'_1 = \delta_1 = \lambda_1\phi - \lambda_2/\phi$. D'où

$$\lambda_1 = \frac{1}{2+\phi}(\delta_0 + \phi\,\delta_1) \quad \Longrightarrow \quad (\lambda_1 < 0) \iff \frac{|u_0 - x_0|}{|u_1 - x_0|^\phi} < C^{\phi-1}.$$

En conséquence on en déduit que pour n grand, $\delta_n \le \frac{1}{2+\phi}(\delta_0 + \phi\,\delta_1)\,\phi^n$, soit:

$$\big|u_n - x_1\big| \le C'\,\exp\big(\lambda_1\,\phi^n\big).$$

avec $C' = \exp(\phi\delta_0 - \delta_1)/(2\phi - 1))/C$. □

Si maintenant on suppose directement que $f : \mathbb{R} \to \mathbb{R}$ est une fonction de classe $\mathcal{C}^1$ sur un voisinage I de x_0, unique solution de l'équation $f(x_0) = 0$ dans ce voisinage, on voit avec la méthode de la sécante que pour n suffisamment grand, u_n se rapproche de x_0 et donc $\frac{u_n - u_{n-1}}{f(u_n)-f(u_{n-1})} \sim 1/f'(u_n)$. Aussi peut-on définir la méthode de **Newton-Raphson** qui conduit à la construction de la suite (v_n) telle que:

$$v_{n+1} = v_n - \frac{f(v_n)}{f'(v_n)} \quad \text{pour } n \in \mathbb{N} \text{ et } v_0 \in I.$$

On obtient ainsi

Proposition (Méthode de Newton-Raphson). *Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction de classe $\mathcal{C}^2$ sur un voisinage I de x_0, unique solution de l'équation $f(x_0) = 0$ dans ce voisinage. On suppose qu'il existe u_0 dans ce voisinage tel que f' soit de signe constant sur $[u_0, x_0]$ (ou $[x_0, u_0]$) avec $f'(x_0) \neq 0$. Si f est de classe $\mathcal{C}^2$ sur I, et si $m_1 = \inf_{x\in I} |f'(x)|$ et $M_2 = \sup_{x\in I} |f''(x)|$ et $|u_0 - x_0| \leq 2M_1/M_2$,*

$$|u_n - x_0| \leq \Big(\frac{M_2}{2m_1}\Big)^{2^n-1} |u_0 - x_0|^{2^n} \underset{n\to+\infty}{\longrightarrow} 0.$$

Proof. On vérifie d'abord que x_0 est une limite possible de la suite (u_n). On montre à l'aide d'un développement de Taylor-Lagrange d'ordre 2 de $f(x_0)$ en u_n, que $f(x_0) = f(u_n) + (u_n - x_0)f'(u_n) + \frac{1}{2}(u_n - x_0)^2 f'(\theta)$ avec θ situé entre x_0 et u_n, soit $f(u_n) = (u_n - x_0)f'(u_n) - \frac{1}{2}(u_n - x_0)^2 f''(\theta)$. D'où en remplaçant , alors

$$\big|u_{n+1} - x_0\big| = \Big|\frac{1}{2}(u_n - x_0)^2 \frac{f''(\theta)}{f'(u_n)}\Big| \leq \frac{M_2}{2m_1}|u_n - x_0|^2 \leq \Big(\frac{M_2}{2m_1}\Big)^{2^{n+1}-1} |u_0 - x_0|^{2^{n+1}}.$$

Donc la convergence est dite quadratique et si u_0 est suffisamment proche de x_0 (si $\frac{M_2}{2m_1}|u_0 - x_0| < 1$). □

Exemple.

Pour trouver une approximation de $\sqrt{2}$ grâce à $f(x) = x^2 - 2$, on peut utiliser $u_{n+1} = u_n - \frac{u_n^2-2}{2u_n}$. Si on prend $I = [1, 2]$, alors $\frac{M_2}{2m_1} = \frac{1}{2}$, d'où la convergence quadratique quand $u_0 = 1$ ou $u_0 = 2$.
Pour ces deux dernières méthodes, en général on préfère utiliser un critère d'arrêt pour décider du rang auquel on s'arrête afin d'obtenir une approximation d'un certain ordre. Ce critère est le suivant:

Si on désire une approximation de x_0 à l'ordre ε, on arrête de calculer la suite pour n tel que

$$\big|u_{n+1} - u_n\big| \leq \varepsilon.$$

Aussi utilisera-t-on fréquemment une commande `while` pour réaliser l'algorithme.

3.4 Calcul approché d'intégrales

Une autre question intéressante est de pouvoir calculer numériquement des intégrales. En effet, les cours de mathématiques "classiques" ont l'habitude de présenter des intégrales que l'on peut calculer explicitement par une technique ou une autre (recherche d'une primitive, intégration par parties, changement de variable, méthode des résidus,...). Cependant, même en ne considérant que des intégrales définies (non généralisées), trouver explicitement la valeur d'une intégrale ne peut être effectué que dans des cas très particuliers. Considérons l'exemple simple suivant:

Exemple.

Soit X une variable suivant une loi normale centrée réduite. Quelle est la probabilité que $|X|$ soit inférieure à 2? Cela revient à calculer $\frac{1}{\sqrt{2\,\pi}} \int_{-2}^{2} e^{-t^2/2} dt$. Mais on ne peut calculer explicitement cette intégrale à partir de primitives usuelles. On peut cependant l'approcher (voir ci-dessous) et on obtient à peu près 0.95... Une première idée naturelle pour approcher une intégrale définie $I = \int_a^b f(x)dx$ est d'utiliser des sommes de Riemann. Ainsi on pourra considérer:

$$S_0(n) = \frac{(b-a)}{n} \sum_{k=0}^{n-1} f\big(a + \frac{k(b-a)}{n}\big).$$

Il est clair que si f est continue. On peut même dire un peu plus dès que l'on suppose que f est de classe $\mathcal{C}^1$ sur $[a, b]$.

Proposition (Méthode d'approximation dite des rectangles). *Si f est de classe $\mathcal{C}^1$ sur $[a,b]$, avec $M_1 = \sup_{a\leq x\leq b} |f'(x)|$, alors*

$$\big|I - S_0(n)\big| \leq \frac{1}{2n}\,(b-a)^2\,M_1.$$

Proof. On a $|I - S_0(n)| = \Big|\sum_{k=0}^{n-1} \int_{a+\frac{k(b-a)}{n}}^{a+\frac{(k+1)(b-a)}{n}} \big(f(t) - f\big(a+\frac{k(b-a)}{n}\big)\big)\,dt\Big| \leq \sum_{k=0}^{n-1} \int_{a+\frac{k(b-a)}{n}}^{a+\frac{(k+1)(b-a)}{n}} \Big|f(t) - f\big(a+\frac{k(b-a)}{n}\big)\Big|\,dt$. En notant $M_1 = \sup_{a\leq x\leq b}|f'(x)|$ et en utilisant l'Inégalités des Accroissements Finis, on obtient:

$$\begin{aligned}
\big|I - S_0(n)\big| &\leq \sum_{k=0}^{n-1} \int_{a+\frac{k(b-a)}{n}}^{a+\frac{(k+1)(b-a)}{n}} M_1 \Big|t - \big(a+\frac{k(b-a)}{n}\big)\Big|\,dt \\
&\leq M_1 \sum_{k=0}^{n-1} \int_0^{\frac{(b-a)}{n}} t\,dt \\
&\leq \frac{1}{2n}\,(b-a)^2\,M_1.
\end{aligned}$$

□

On a donc une qualité d'approximation en $1/n$. Il est clair que la qualité d'approximation aurait été la même si l'on avait décalé les rectangles de $(b-a)/n$ en abscisse, c'est-à-dire si l'on avait utilisé une somme allant de $k=1$ à n plutôt que de $k=0$ à $n-1$.
On peut facilement gagner en précision par rapport à cette méthode. Il suffit juste de moyenner les 2 approximations précédentes. C'est ce que l'on appelle la méthode des trapèzes, car cela revient également à remplacer les rectangles précédents par des trapèzes, les sommets de chaque trapèze étant les $(f\big(a+\frac{k(b-a)}{n}\big))_k$.

$$S_1(n) = \frac{(b-a)}{n} \sum_{k=0}^{n-1} \frac{1}{2}\Big\{f\big(a+\frac{k(b-a)}{n}\big) + f\big(a+\frac{(k+1)(b-a)}{n}\big)\Big\}.$$

On peut aussi voir cette méthode comme une approximation du premier ordre de la fonction f: la méthode des rectangles consiste à remplacer f par une fonction constante par morceaux, la méthode des trapèzes consiste à remplacer la fonction f par une succession de segments, donc une fonction linéaire par morceaux. On obtient alors le résultat suivant:

Proposition (Méthode d'approximation dite des trapèzes). *On suppose que f est de classe $\mathcal{C}^2$ sur $[a,b]$ avec $M_2 = \sup_{a\leq x\leq b}|f''(x)|$. Alors*

$$\big|I - S_1(n)\big| \leq \frac{1}{12\,n^2}\,(b-a)^3\,M_2.$$

Proof. En reprenant les calculs précédents, on obtient:

$$\big|I - S_1(n)\big| \leq \Big|\sum_{k=0}^{n-1} \int_{a+\frac{k(b-a)}{n}}^{a+\frac{(k+1)(b-a)}{n}} f(t) - \frac{1}{2}\Big\{f\big(a+\frac{k(b-a)}{n}\big) + f\big(a+\frac{(k+1)(b-a)}{n}\big)\Big\}\,dt\Big|.$$

Pour simplifier les notations, on va majorer $\int_{x_1}^{x_2} f(t) - \frac{1}{2}\big(f(x_1)+f(x_2)\big)\,dt$. En utilisant une double intégration par partie, il est facile de montrer que:

$$\int_{x_1}^{x_2} f(t) - \frac{1}{2}\big(f(x_1)+f(x_2)\big)\,dt = \frac{1}{2}\int_{x_1}^{x_2} (x-x_1)(x_2-x)f''(x)dx.$$

On en déduit ainsi que:

$$\begin{aligned}
\Big|\int_{x_1}^{x_2} f(t) - \frac{1}{2}\big(f(x_1)+f(x_2)\big)\,dt &\leq \Big| \leq \frac{M_2}{2}\int_{x_1}^{x_2}(x-x_1)(x_2-x)dx \\
&\leq \Big| \leq \frac{M_2}{12}\,(x_2-x_1)^3.
\end{aligned}$$

On peut ainsi appliquer ce résultat à $x_1 = f\big(a+\frac{k(b-a)}{n}\big)$ et $x_2 = f\big(a+\frac{(k+1)(b-a)}{n}\big)$ d'où le résultat final. □

On voit ainsi le gain important (on est désormais à une vitesse d'approximation en $1/n^2$) offert par cette méthode. Notons que la méthode des rectangles ne permet pas d'atteindre une telle vitesse même lorsque l'on suppose la fonction de classe $\mathcal{C}^2$, hormis dans de rares cas particuliers.

Mais cela donne également l'idée d'améliorer encore la qualité d'approximation, en choisissant une approximation locale d'ordre 2 de la fonction f, donc une approximation de f par des fonctions paraboliques par morceaux.

On commence par définir plus généralement ce qu'est un polynôme d'interpolation:

Définition (Polynôme d'interpolation de Lagrange). *Soit $x_1 < x_2 < \cdots < x_n$, n réels distincts, avec $n \geq 1$ et f une fonction définie en les x_i. Alors il existe un unique polynôme P_L de degré $(n-1)$ tel que $P(x_i) = f(x_i)$ pour tout $i = 1, \cdots, n$, appelé polynôme d'interpolation de Lagrange. On a:*

$$P_L(x) = \sum_{i=1}^{n} f(x_i) \frac{\prod_{j\neq i}(x - x_i)}{\prod_{j\neq i}(x_j - x_i)} \quad \textit{pour tout } x \in \mathbb{R}.$$

Pour "mieux" approcher la fonction f entre x_1 et x_2 qu'avec une fonction constante (typiquement $f(x_1)$ ou $f(x_2)$) ou le segment reliant $(x_1, f(x_1))$ à $(x_2, f(x_2))$, on peut utiliser une approximation parabolique en rajoutant le milieu de $[x_1, x_2]$, donc le point $\frac{1}{2}(x_1+x_2)$, et la valeur de f en ce point, soit $f\big(\frac{1}{2}(x_1+x_2)\big)$. Ce polynôme s'écrit donc entre x_1 et x_2 sous la forme:

$$\widetilde{f}_2(x) = -\frac{2}{(x_2-x_1)^2}\Big(f(x_1)\,\big(x-\frac{x_1+x_2}{2}\big)(x-x_2) - 2\,f\big(\frac{x_1+x_2}{2}\big)\,(x-x_2)(x-x_1) + f(x_2)\,\big(x-\frac{x_1+x_2}{2}\big)(x-x_1)\Big).$$

En intégrant un tel polynôme entre x_1 et x_2, on obtient que la surface approchée de f entre x_1 et x_2 vaut:

$$\begin{aligned}\int_{x_1}^{x_2} \widetilde{f}_2(x)dx &= \frac{2}{(x_2-x_1)^2}\Big(f(x_1)\int_{x_1}^{x_2}\big(x-\frac{x_1+x_2}{2}\big)(x-x_2)dx - 2\,f\big(\frac{x_1+x_2}{2}\big)\int_{x_1}^{x_2}(x-x_2)(x-x_1)dx \\ &\qquad + f(x_2)\int_{x_1}^{x_2}\big(x-\frac{x_1+x_2}{2}\big)(x-x_1)dx\Big) \\ &= \frac{(x_2-x_1)}{6}\Big(f(x_1) + 4\,f\big(\frac{x_1+x_2}{2}\big) + f(x_2)\Big).\end{aligned}$$

Aussi peut-on maintenant définir l'approximation de Simpson de l'intégrale $I = \int_a^b f(t)dt$ par la somme:

$$S_2(n) = \frac{(b-a)}{6n}\sum_{k=0}^{n-1}\big\{f\big(a+\frac{k(b-a)}{n}\big) + 4\,f\big(a+\frac{(2k+1)(b-a)}{2n}\big) + f\big(a+\frac{(k+1)(b-a)}{n}\big)\big\}.$$

On peut alors montrer le résultat suivant:

Proposition (Méthode d'approximation de Simpson). *On suppose que f est de classe $\mathcal{C}^4$ sur $[a,b]$ avec $M_4 = \sup_{a\leq x\leq b}|f^{(4)}(x)|$. Alors*

$$\big|I - S_2(n)\big| \leq \frac{1}{2880\,n^4}\,(b-a)^5\,M_4.$$

Proof. Posons $g(t) = \int_{m-t}^{m+t} f(x)dx - \frac{1}{6}\big(f(m-t) + 4f(m) + f(m+t)\big)$. On va majorer $|g(t)|$, et on remplacera ensuite t par $(b-a)/2n$ et m par $a + (2k+1)(b-a)/2n$. La fonction g est de classe $\mathcal{C}^4$ et on obtient:

$$\begin{aligned} g'(t) &= \frac{1}{3}\big(2\,f(m+t) + 2\,f(m-t) - 4\,f(m) + t\,f'(m-t) - t\,f'(m+t)\big) \quad \text{et } g'(0) = 0 \\ g''(t) &= \frac{1}{3}\big(f'(m+t) - f'(m-t) - t\,f''(m-t) - t\,f''(m+t)\big) \quad \text{et } g''(0) = 0 \\ g^{(3)}(t) &= \frac{1}{3}\,t\,\big(f^{(3)}(m-t) - f^{(3)}(m+t)\big). \end{aligned}$$

Il est clair, avec une Inégalité des Accroissements Finis que:

$$|g^{(3)}(t)| \leq \frac{M_4}{3}\, t^2.$$

On écrit ensuite que $g''(t) = \int_0^t g^{(3)}(x)dx$ car $g''(0) = 0$. D'où,

$$|g''(t)| \leq \int_0^t |g^{(3)}(x)|dx \leq \frac{M_4}{9}\, t^3.$$

Même chose en écrivant $g'(t) = \int_0^t g''(x)dx$, grâce à $g'(0) = 0$. On obtient alors:

$$|g'(t)| \leq \int_0^t |g''(x)|dx \leq \frac{M_4}{36}\, t^4.$$

Enfin, dernière étape, on obtient:

$$|g(t)| \leq \int_0^t |g'(x)|dx \leq \frac{M_4}{180}\, t^5.$$

Comme $t = (b-a)/2n$, on voit apparaître un 2^5 supplémentaire, dont le produit avec 180 fait bien 2880. Apparaît aussi un $1/n^5$, que l'on somme n fois, ce qui amène à un terme en $1/n^4$. □

4 Statistique et probabilités numériques

On rappelle que si $(\Omega, \mathcal{A}, \mathbb{P})$ est un espace de probabilités, alors une variable aléatoire X est une fonction $X : \omega \in \Omega \mapsto X(\omega) \in \mathbb{R}$ telle que $F_X(x) = \mathbb{P}(X \leq x)$ existe pour tout $x \in \mathbb{R}$. Lorsque l'on observe le résultat d'une variable aléatoire au cours d'une expérience, cela revient à fixer $\omega \in \Omega$ et à observer $X(\omega)$, un réel, appelé le plus souvent **une réalisation de la variable aléatoire**. C'est ce dont on dispose en statistique, d'une donnée et non d'une variable aléatoire. Si l'expérience est recommencée, on disposera alors de $X(\omega')$, où $\omega' \in \Omega$ est généralement différent de ω.

4.1 Simulation de réalisations de variables aléatoires

L'ordinateur fonctionne de manière purement déterministe. Pourtant il est possible de produire avec un ordinateur des vecteurs de nombres qui se comportent quasiment comme des réalisations de variables indépendantes de loi uniforme sur $[0, 1]$ (le fait que ce ne soit qu'une approximation n'est pratiquement pas décelable). Obtenir un "vrai hasard" nécessiterait une infinité d'opération ce qui n'est pas accessible. Une fois ces approximations obtenues, il est possible de simuler des quasi-réalisations de n'importe quel échantillon de v.a.i.i.d., lorsque la loi de ces variables est connue.

La première étape est donc la simulation de ces quasi-réalisations de variables indépendantes de loi uniforme sur $[0, 1]$. Plusieurs algorithmes sont possibles avec les logiciels (voir notamment la page qui décrit tous les algorithmes programmés avec R), mais ils procèdent tous à peu près du même principe que nous décrivons maintenant.

Soit a et b deux nombres entiers positifs et premier entre eux (sans diviseur commun). On considère la suite d'entiers $(x_n)_{n\in\mathbb{N}}$,

$$x_{n+1} = a \times x_n \qquad \mod(b), \qquad x_0 \in \{0$$

4.2 Théorèmes limites

Les deux résultats majeurs qui permettent de relier la statistique (c'est-à-dire un travail sur des données) à des calculs théoriques probabilistes (typiquement des caractéristiques de lois de probabilités) sont la Loi Forte des Grands Nombres et le Théorème de la Limite Centrale. Avant d'en

dire plus, on rappelle deux définitions importantes relatives à la convergence de suites de variables aléatoires:

Définition. *Soit $(X_n)_{n\in\mathbb{N}}$ une suite de variables aléatoires définies sur le même espace de probabilités $(\Omega, \mathcal{A}, \mathbb{P})$. On dit que (X_n) converge* **en probabilité** *vers une variable aléatoire Y, ce qui est noté $X_n \xrightarrow[n\to+\infty]{\mathcal{P}} Y$ lorsque pour tout $\varepsilon > 0$, $\mathbb{P}\big(|X_n - Y| \geq \varepsilon\big) \underset{n\to+\infty}{\longrightarrow} 0$.*

Concrètement, cette convergence signifie que les réalisations de X_n se rapprochent de plus en plus de celles de Y avec une grande probabilité et si n augmente encore, elles se rapprochent encore plus. Cela peut s'illustrer lorsque Y est une constante que X_n s'en rapproche globalement de plus en plus, comme une convergence habituelle d'une suite à l'infini vers sa limite, à ceci près que si l'on trace l'ensemble des points $(n, X_n(\omega))$ la trajectoire produite a une asymptote mais s'en rapproche de façon très irrégulière. Une autre convergence va être essentielle dans la suite:

Définition. *Soit $(X_n)_{n\in\mathbb{N}}$ une suite de variables aléatoires définies sur le même espace de probabilités $(\Omega, \mathcal{A}, \mathbb{P})$. On dit que (X_n) converge* **en loi** *vers une variable aléatoire Y, ce qui est noté $X_n \xrightarrow[n\to\infty]{\mathcal{L}} Y$ lorsque $F_{X_n}(x) \underset{n\to+\infty}{\longrightarrow} F_Y(x)$ pour tout $x \in \mathbb{R}$ tel que F_Y soit continue en x.*

Théorème.

Quelques applications:

1. Intervalles de confiance d'un estimateur
2. Test sur un paramètre
3. Méthode de Monte-Carlo pour estimer la valeur d'une intégrale

4.3 Régression par moindres carrés