Master 1 Métiers de l'Enseignement, Mathématiques - ULCO, La Mi-Voix, 2011/2012

ANALYSE NUMÉRIQUE - TPs

Fiche de Mathématiques 3 - Le Δ^2 d'Aitken et l'ϵ-algorithme.

1 Le Δ^2 d'Aitken

Le Δ^2 d'Aitken est un procédé d'accélération de la convergence de suites en analyse numérique popularisé par le mathématicien Alexander Aitken en 1926. C'est l'un des algorithmes d'accélération de la convergence les plus populaires du fait de sa simplicité et de son efficacité. Une première forme de cet algorithme a été utilisé par Kowa Seki (fin du XVII siècle) pour calculer une approximation de π par la méthode des polygones d'Archimède.

1.1 Definition

Soit une suite convergente $x = (x_n)_{n\in\mathbb{N}}$, convergeant vers une limite que l'on souhaite déterminer, le procédé Δ^2 d'Aitken associe à cette suite une autre suite définie par :

$$Ax_n = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n} = x_n - \frac{(x_{n+1} - x_n)^2}{x_n - 2x_{n+1} + x_{n+2}}, \quad n \in \mathbb{N}. \tag{1}$$

C'est de la première écriture que le procédé tire son nom. Sous certaines conditions, la suite (Ax_n) converge plus vite que la suite initiale (x_n), ce qui permet d'estimer la limite de (x_n) avec une meilleure précision et/ou en effectuant moins de calculs. C'est un algorithme numériquement peu stable : il convient de calculer la suite (x_n) ainsi que (Ax_n) avec un nombre important de chiffres significatifs. Certaines écritures de l'algorithme propagent moins les erreurs d'arrondi, par exemple :

$$Ax_n = x_{n+1} + \frac{1}{\dfrac{1}{x_{n+2} - x_{n+1}} - \dfrac{1}{x_{n+1} - x_n}}.$$

(Ax_n) étant elle même une suite numérique, il est possible de lui appliquer le Δ^2, et ainsi de suite : c'est ce qu'on appelle une application itérée du Δ^2.

1.2 Propriétés

Le Δ^2 est un algorithme non linéaire d'accélération de la convergence. Mais, il vérifie la propriété :

$$A(ax_n + b) = aA(x_n) + b,\ a, b \in \mathbb{K}.$$

Le Δ^2 détermine une estimation de la limite de la suite (x_n) en partant de l'hypothèse que celle-ci vérifie l'équation aux différences suivante :

$$a_0(x_n - x_\infty) + a_1(x_{n+1} - x_\infty) = 0,\ a_0 + a_1 \neq 0.$$

En résolvant cette équation aux différences, les suites, dont le Δ^2 détermine de manière immédiate la limite, sont de la forme :

$$x_n = x_\infty + a\lambda^n,\ \lambda \neq 1.$$

Il est à noter que même si la suite (x_n) diverge, c'est-à-dire si $|\lambda| > 1$, le Δ^2 converge vers « l'anti-limite » de la suite.

Le théorème de convergence pour le procédé d'Aitken est le suivant :

Théorème 1.1 *Si la suite (x_n) converge vers x_∞ et si :*

$$\lim_{n\to\infty} \frac{x_{n+1} - x_\infty}{x_n - x_\infty} = \frac{x_{n+2} - x_{n+1}}{x_{n+1} - x_n} = \lambda \neq 1$$

alors (Ax_n) converge vers x_∞ plus vite que (x_n).

Le Δ^2 est donc particulièrement bien adapté aux suites à convergence linéaire (dont l'écart avec leur limite se comporte à l'infini comme une suite géométrique).
Le Δ^2 est un cas particulier de certaines transformations plus générales : la transformation de Shanks, le procédé d'Overholt.

1.3 Exemples d'application

• *Accélération de la convergence d'une série.*
Le Δ^2 peut être utilisé pour accélérer la convergence des séries en extrayant une suite numérique d'après leur somme partielle. Par exemple, la valeur de $\frac{\pi}{4}$ peut être calculée d'après la série de Leibniz $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$, réputée pour sa convergence très lente :

$$\frac{\pi}{4} = \sum_{n=1}^{\infty} \frac{(-1)^n}{2n+1} \simeq 0,785398.$$

n	terme	x_n = somme partielle	Ax_{n-2}
0	1	1	–
1	-0.33333333	0.66666667	–
2	0.2	0.86666667	0.79166667
3	-0.14285714	0.72380952	0.78333333
4	0.11111111	0.83492063	0.78492063
5	-9.0909091e-2	0.74401154	0.78567821
6	7.6923077e-2	0.82093462	0.78522034
7	-6.6666667e-2	0.75426795	0.78551795
8	5.8823529e-2	0.81309148	0.78531371

TABLE 1 – Accélération de la convergence de la série de Leibniz

La précision obtenue par le Δ^2 avec seulement 9 termes, serait obtenue en sommant plus de 4000 termes de la suite non accélérée !

• *Accélération de la convergence des procédés itératifs.*
La convergence d'un procédé itératif de point fixe du type $x_{n+1} = f(x_n)$ peut être accélérée de plusieurs manières :
- classiquement en calculant le Δ^2 de la suite récurrente,
- en réinjectant périodiquement le résultat du Δ^2 dans la suite d'origine, afin de « l'aider » à converger.

Cette deuxième stratégie, appliquée systématiquement toutes les 3 itérations, est appelé procédé d'Aitken-Steffensen. Il permet dans la plupart des cas de transformer une convergence (ou divergence) linéaire en convergence quadratique, et une convergence quadratique en super-quadratique. Le procédé d'Aitken-Steffensen remplace l'itération d'origine $x_{n+1} = f(x_n)$ par

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

Exemple 1.1 L'équation

$$x - e\sin(x) = M$$

(appelée équation de Kepler, liée au calcul d'orbites en astronomie) où x est l'inconnue (M et e étant connus), peut être résolue par exemple par l'itération :

$$\begin{cases} x_0 = M \\ x_{n+1} = M + e\sin(x_n) \end{cases}.$$

D'après le théorème du point fixe, cette suite converge vers la solution de l'équation de départ, si $e < 1$. Mais cette suite sera d'autant plus lente à converger que e sera proche de 1 (cas fréquemment rencontré en pratique, car typique des orbites des comètes). Il sera intéressant dans ce cas d'accélérer la convergence avec le Δ^2 ou le procédé d'Aitken-Steffensen. Par exemple, pour $e = 0,9$ et $M = 0,01$ (solution $x = 0,0985644\ldots$), on obtient la TABLE 2 de la page suivante.
Les colonnes des Δ^2 ont été décalées vers le bas pour mettre sur une même ligne les itérés de base nécessaires à leur calcul, et ainsi mieux visualiser le gain apporté par l'accélération de la convergence vis à vis des itérations de base. On constate une nette accélération de la convergence de la suite initiale par le Δ^2. Les applications itérées

n	x_n = valeur itérée	Ax_{n-2}	A^2x_{n-4}	A^3x_{n-6}	Steffensen
0	0,0100000				0,0100000
1	0,0189999				0,0189999
2	0,0270988	0,0999107			0,0270988
3	0,0343860	0,0997946			**0,0999107**
4	0,0409413	0,0996601	0,1006471		0,0997701
5	0,0468369	0,0995222	0,1050054		0,0996442
6	0,0521378	0,0993898	0,0961648	0,1020862	**0,0985651**
7	0,0569027	0,0992677	0,0978300	0,0975661	0,0985650
8	0,0611848	0,0991582	0,0982150	0,0983308	0,0985649
9	0,0650320	0,0990622	0,0983714	0,0984784	**0,0985644**
10	0,0684875	0,0989792	0,0984494	0,0985271	0,0985644
11	0,0715906	0,0989082	0,0984927	0,0985467	0,0985644
12	0,0743765	0,0988482	0,0985183	0,0985555	**0,0985644**

TABLE 2 – Accélération de la convergence - Équation de Kepler

du Δ^2 accélère encore d'avantage, mais le résultat le plus spectaculaire est obtenu par la méthode de Steffensen (les nombres en gras visualisent l'application du Δ^2 toutes les 3 itérations). Pour obtenir la même précision que le Δ^2 après 12 itérations, il aurait fallu itérer 50 fois la formule de base, et l'itérer 300 fois pour être équivalent à la méthode de Steffensen.

Exemple 1.2 Lorsque l'itération de base a une convergence quadratique (par exemple avec la méthode de Newton), le Δ^2 ou la méthode de Steffensen, quoique accélérant la convergence, présente moins d'intérêt pratique. Le calcul d'une itération de base supplémentaire permet souvent d'obtenir le même résultat que la valeur accélérée.
La valeur de $\sqrt{2} \simeq 1.4142136$ peut être calculée par la méthode de Héron, en partant d'une valeur initiale x_0 et la suite récurrente (à convergence quadratique) :

$$x_{n+1} = \frac{x_n + \frac{2}{x_n}}{2}.$$

En partant de $x_0 = 1$, on obtient la TABLE 3.

n	x_n = valeur itérée	Ax_{n-2}
0	1	
1	1.5	
2	1.4166667	1.4285714
3	1.4142157	1.4141414
4	1.4142136	1.4142136

TABLE 3 – Accélération de la convergence - Méthode de Héron

On constate certes un gain en précision en accélérant la convergence, mais l'itéré de base suivant est du même ordre de précision, pour un moindre coût en calcul.

• *Autres applications.*
Ce procédé est notamment utilisé pour accélérer l'algorithme de décomposition de domaine de type Schwarz (additifs et multiplicatifs). En effet, on peut remarquer que sous certaines conditions, les coefficients de Fourier liés aux solutions itérées obtenus ont une convergence purement linéaire. Par ce principe, on peut réduire le nombre total d'itérations de l'algorithme à 3 ou 5 itérations pour des problèmes 1D ou 2D (respectivement).

2 L'ϵ-algorithme

En analyse numérique, l'ϵ-algorithme est un algorithme non linéaire d'accélération de la convergence de suites numériques. Cet algorithme a été proposé par Peter Wynn en 1956 pour calculer la transformation de Shanks. C'est l'un des algorithmes d'accélération de la convergence les mieux connus et des plus utilisés. C'est une généralisation de l'algorithme Δ^2 d'Aitken.

2.1 Présentation

Soit une suite numérique (S_n) dont on cherche à connaître la limite S. L'ϵ-algorithme consiste à calculer un tableau en initialisant la première colonne par la suite (S_n), et en calculant les autres valeurs à l'aide des relations suivantes :

$$\begin{aligned} &\epsilon_{-1}^{(n)} = 0,\ \epsilon_{2n}^{(-n-1)} = 0,\ \epsilon_0^{(n)} = S_n\ \forall n \\ &\epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + \frac{1}{\epsilon_k^{(n+1)} - \epsilon_k^{(n)}}\ \forall n, k. \end{aligned} \tag{2}$$

Les différentes colonnes d'indice k pair du tableau fournissent d'autres suites convergeant en général plus vite que la suite S_n d'origine. Les colonnes d'indice impair peuvent être considérées comme des intermédiaires de calcul. On présente traditionnellement les résultats de l'ϵ-algorithme sous forme d'un tableau aux lignes décalées : la formule de calcul de l'ϵ-algorithme relie ainsi les termes formant un losange dans le tableau.

$$\begin{array}{ccccccc}
0 & & 0 & & 0 & & \cdots \\
& \epsilon_0^{(0)} = S_0 & & \epsilon_2^{(-1)} & & \epsilon_4^{(-2)} & & \cdots \\
0 & & \epsilon_1^{(0)} & & \epsilon_3^{(-1)} & & \cdots \\
& \epsilon_0^{(1)} = S_1 & & \epsilon_2^{(0)} & & \epsilon_4^{(-1)} & & \cdots \\
0 & & \epsilon_1^{(1)} & & \epsilon_3^{(0)} & & \cdots \\
& \epsilon_0^{(2)} = S_2 & & \epsilon_2^{(1)} & & \epsilon_4^{(0)} & & \cdots \\
0 & & \epsilon_1^{(2)} & & \epsilon_3^{(1)} & & \cdots \\
& \epsilon_0^{(2)} = S_3 & & \epsilon_2^{(2)} & & \epsilon_4^{(1)} & & \cdots \\
\vdots & & \vdots & & \vdots & & \ddots \\
& \vdots & & \vdots & & \vdots &
\end{array}$$

Le tableau obtenu par l'ϵ-algorithme est une des méthodes utilisées pour calculer la transformation de Shanks d'une suite. En effet :

$$e_k(S_n) = \epsilon_{2k}^{(n)}\ \forall k, n \tag{3}$$

où $e_k(S_n)$ est la transformée de Shanks d'ordre k de la suite (S_n). L'ϵ-algorithme partage donc toutes les propriétés de la transformée de Shanks. Notamment, l'ϵ-algorithme est un moyen pour calculer la table de Padé d'une série entière. De même, la suite $\epsilon_2^{(n)}$ correspond au Δ^2 d'Aitken. Les approximants de Padé à plusieurs variables peuvent aussi très simplement être calculés à l'aide de l'ϵ-algorithme. Par exemple, pour une fonction à 2 variables, dont le développement de Taylor est

$$f(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{ij} x^i y^j,$$

sa table de Padé peut être obtenue à l'aide de l'ϵ-algorithme, en initialisant la première colonne par :

$$\epsilon_0^{(n)} = \sum_{m=0}^{n} \left(\sum_{i+j=m} a_{ij} x^i y^j \right).$$

2.2 Variantes

• *ϵ-algorithme vectoriel.*
Il existe plusieurs variantes de l'ϵ-algorithme pouvant êtres utilisées avec des suites vectorielles (ϵ-algorithme vectoriel, topologique ou scalaire appliqué à chacune des composantes du vecteur). L'ϵ-algorithme vectoriel est analogue dans sa forme à l'ϵ-algorithme classique :

$$\begin{aligned} &\epsilon_{-1}^{(n)} = 0,\ \epsilon_{2n}^{(-n-1)} = 0,\ \epsilon_0^{(n)} = S_n\ \forall n \\ &\epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + (\epsilon_k^{(n+1)} - \epsilon_k^{(n)})^{-1}\ \forall n, k \end{aligned} \tag{4}$$

où S_n et 0 sont des vecteurs.
Il reste à définir ce qu'est l'inverse d'un vecteur ! On pourra prendre par exemple comme inverse d'un vecteur U :

$$U^{-1} = \frac{U}{\|U\|^2}$$

où la norme du vecteur est la norme euclidienne. L'ϵ-algorithme vectoriel a de nombreuses applications pratiques mais est très sensible aux erreurs d'arrondi. Il permet notamment de généraliser la méthode d'Aitken-Steffensen à des systèmes d'équations non linéaires, fournissant ainsi un algorithme à convergence quadratique, ne nécessitant pas de calcul du Jacobien, à l'inverse de la méthode de Newton.

L'ϵ-algorithme matriciel existe pour les matrices carrées, dans ce cas l'inverse apparaissant dans l'algorithme est tout simplement celui d'une matrice carré classique.

• *ϵ-algorithme confluent.*
L'ϵ-algorithme classique accélère une suite numérique, c'est-à-dire une fonction d'une variable discrète 'n'. La version confluente de cet algorithme accélère la convergence d'une fonction d'une variable continue 't'. On l'obtient en faisant un changement de variable $x = t + n \times h$ à l'algorithme d'origine, et en faisant tendre h vers 0. On obtient :

$$\begin{aligned} &\epsilon_1(t) = 0,\ \epsilon_0(t) = f(t)\ , \\ &\epsilon_{k+1}(t) = \epsilon_{k-1}(t) + \frac{1}{\epsilon_k'(t)},\ k = 0, 1, \dots \end{aligned} \tag{5}$$

Cet algorithme est utilisé par exemple pour l'évaluation des intégrales impropres lorsque les dérivées de la fonction sont accessibles, ou à l'élaboration de séries asymptotiques pour certaines fonctions. Les dérivées des termes apparaissant dans la formule peuvent se ramener aux dérivées de la fonction de base $f(t)$ par calcul formel, où en utilisant les relations des déterminants de Hankel :

$$\epsilon_{2k}'(t) = \frac{H_k^{(1)} H_{k+1}^{(1)}}{(H_k^{(2)})^2} \quad \text{et} \quad \epsilon_{2k+1}'(t) = \frac{H_k^{(2)} H_{k+1}^{(2)}}{(H_{k+1}^{(1)})^2} \tag{6}$$

avec

$$\begin{cases} H_0^{(n)} = 1 \text{ et } H_1^{(n)} = f^{(n)}(t),\ n = 0, 1, \dots \\ H_{k+2}^{(n-1)} H_k^{(n+1)} + (H_{k+1}^{(n)})^2 = H_{k+1}^{(n-1)} H_{k+1}^{(n+1)},\ n, k = 0, 1, \dots \end{cases}$$

On trouve, en explicitant les premières expressions :

$$\frac{1}{\epsilon_1'(t)} = -\frac{[f'(t)]^2}{f''(t)} \text{ et } \frac{1}{\epsilon_2'(t)} = \frac{f'(t) f'''(t) - f''^2(t)}{f''(t)[f''(t) f^{(4)}(t) - f'''^2(t)]^2}.$$

• *ϵ-algorithme dépendant d'une suite auxiliaire.*
Il arrive fréquemment que la suite à accélérer (S_n) dépende d'une suite auxiliaire (x_n), celle-ci tendant vers l'infini (par exemple des approximations par discrétisation avec un maillage de plus en plus fin). Il est possible d'utiliser l'ϵ-algorithme de base dans ces cas mais la façon dont évolue la suite (x_n) associée (par exemple l'évolution du nombre de mailles entre chaque S_n) est un renseignement précieux qui pourrait aider l'accélération de la convergence. Deux variantes de l'ϵ-algorithme utilisent cette information supplémentaire et donnent souvent de meilleurs résultats. La première est donnée par

$$\begin{aligned} &\epsilon_{-1}^{(n)} = 0,\ \epsilon_{2n}^{(-n-1)} = 0,\ \epsilon_0^{(n)} = S_n\ \forall n \\ &\epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + \frac{x_{n+1} - x_n}{\epsilon_k^{(n+1)} - \epsilon_k^{(n)}}\ \forall n, k. \end{aligned} \tag{7}$$

Et la deuxième variante :

$$\epsilon_{-1}^{(n)} = 0,\ \epsilon_{2n}^{(-n-1)} = 0,\ \epsilon_0^{(n)} = S_n\ \forall n$$
$$\epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + \frac{x_{n+k+1} - x_{n+k}}{\epsilon_k^{(n+1)} - \epsilon_k^{(n)}}\ \forall n, k. \qquad (8)$$

Ces algorithmes s'apparentent aux méthodes par extrapolation (extrapolation de Richardson ou ρ-algorithme). Ils peuvent accélérer des suites particulièrement récalcitrantes avec d'autres algorithmes.

• *ϵ-algorithme d'interpolation.*
L'ϵ-algorithme classique permet de calculer la table de Padé d'une fonction lorsque les termes de la suite (S_n) sont les sommes partielles du développement limité de cette fonction. Par analogie, il est possible aussi de construire la table des fractions rationnelles d'interpolation d'une fonction en partant de la suite des polynômes d'interpolation.

$$\begin{aligned} \epsilon_{-1}^{(n)} = 0,\ \epsilon_{2n}^{(-n-1)} = 0,\ \epsilon_0^{(n)} = P_n(x)\ \forall n, \\ \epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + \frac{1}{(x - x_{n+k+1})(\epsilon_k^{(n+1)} - \epsilon_k^{(n)})}\ \forall n, k, \\ R_{n,k}(x) = \epsilon_{2k}^{(n-k)},\ \forall n, k, \end{aligned} \qquad (9)$$

$P_n(x)$ étant le polynôme d'interpolation passant par les points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, que l'on pourra calculer par la méthode de Lagrange, de Newton ou de Neville, par exemple. $R_{n,k}(x)$ est quant à elle la fraction rationnelle de degré n au numérateur et k au dénominateur passant par les points $(x_0, y_0), (x_1, y_1), \ldots, (x_{n+k}, y_{n+k})$. Cet algorithme peut se présenter comme un complément aux méthodes de Thiele ou Bulirsch-Stoer, avec la particularité que l'ϵ-algorithme permet de calculer tout le tableau des fractions rationnelles d'interpolation au lieu d'une seule diagonale, ou un escalier. De plus, si le polynôme utilisé est un polynôme d'interpolation d'Hermite (imposant d'interpoler la fonction, mais aussi sa ou ses dérivées), l'ϵ-algorithme d'interpolation fournira la fraction rationnelle d'interpolation vérifiant les mêmes critères que le polynôme d'Hermite. La méthode d'interpolation de Newton avec confluence des abscisses (pour les points où les dérivées sont à renseigner) est toute indiquée pour calculer le polynôme d'Hermite, d'autant que la suite des abscisses utilisée (avec les confluences) sera nécessaire pour l'ϵ-algorithme d'interpolation.

Remarque 2.1 Il peut arriver lors du calcul du tableau de l'ϵ-algorithme, qu'à certains endroits se produisent des divisions par 0 où presque. Ceci peut générer des dégradations de précision pour le calcul des zones du tableau dépendant de la valeur suspecte, voire des plantage de programme. Plusieurs auteurs ont développé des algorithmes spéciaux pour contourner les valeurs problématiques en cas de détection de risque de division par 0. Ces algorithmes ne fonctionnent que si les valeurs singulières ne sont pas trop proches les unes des autres.

Exemple 2.1 *Accélération de la convergence d'une suite.*
La méthode de Bernoulli permet, sous certaines conditions, d'évaluer la plus grande (ou la plus petite) racine d'un polynôme donné. Mais la suite générée par la méthode peut converger lentement vers la valeur de la racine : l'ϵ-algorithme permet d'accélérer cette convergence. Par exemple, avec le polynôme $x^2 - x - 1$ dont les racines sont le nombre d'or φ et $1/\varphi$, la suite générée par la méthode de Bernoulli donne la suite de Fibonacci F_n dont le ratio des termes successifs converge vers $\varphi = 1,6180339887499\ldots$

$\epsilon_0^{(n)} = F_{n+1}/F_n$	$\epsilon_2^{(n)}$	$\epsilon_4^{(-n)}$	$\epsilon_6^{(n)}$	$\epsilon_8^{(n)}$
0				
1				
0	−2			
2	1,625			
0	6	−162		
1,5	1,61904761	1,618805555		
0	−15	−1170	−45090,00	
1,66666667	1,61818181	1,61803278	1,6180339985	
0	40	−7880	780240,00	−103779601,9
1,6	1,61805555	1,61803405	1,6180339889	1,61803398875
0	−104	−54392	−14196624,02	−4926280210,4
1,625	1,61803713	1,61803398	1,6180339888	
0	273	−371826	253400249,03	
1,61538461	1,61803444	1,61803398		
0	−714	−2551122		
1,61904761	1,61803405			
0	1870			
1,61764705				

Dans cet exemple, seuls les termes d'indice haut positif ont été calculés. Nous constatons dans cet exemple qu'effectivement seules les colonnes paires convergent vers la suite d'origine, et ceci plus rapidement (12 chiffres exacts au lieu de 3 dans la suite initiale). Il est à noter que l'ϵ-algorithme possède une propriété particulière vis à vis de la méthode de Bernoulli qui permet de l'utiliser aussi dans un autre but que l'accélération de la convergence : le calcul simultané de toutes les racines du polynôme au lieu d'une seule à la fois.

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_p x^p$$

dont les racines λ_k sont réelles, telles que $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_p|$. La méthode de Bernoulli génère la suite (y_n) définie par :

$$y_{n+p} = -\frac{1}{a_p}(a_{n+p-1} y_{n-1} + \ldots + a_1 y_{n+1} + a_0 y_n),\ n = 0, 1, \ldots$$

dont le rapport des termes consécutifs y_{n+1}/y_n converge vers λ_1. La suite est initialisée avec $y_0, y_1, \ldots, y_{p-1}$ arbitraires, non tous nuls. En appliquant l'ϵ-algorithme, non pas sur le ratio y_{n+1}/y_n, mais directement sur la suite des y_n, on a :

$$\lim_{n\to\infty} \frac{\epsilon_{2k}^{(n+1)}}{\epsilon_{2k}^{(n)}} = \lambda_{k+1},\ k = 0, 1, \ldots, p-1 \text{ et } \lim_{n\to\infty} \frac{\epsilon_{2k+1}^{(n+1)}}{\epsilon_{2k+1}^{(n+1)}} = \lambda_{k+1},\ k = 0, 1, \ldots, p-1.$$

On constate dans ce cas que les colonnes impaires s'avèrent aussi intéressantes. La vitesse de convergence, pour k fixé est :

$$\frac{\epsilon_{2k}^{(n+1)}}{\epsilon_{2k}^{(n)}} = \lambda_{k+1} + \bigcirc\left[\left(\frac{\lambda_{k+2}}{\lambda_{k+1}}\right)^{n+k+1}\right].$$

On peut donc utiliser à nouveau l'ϵ-algorithme, mais cette fois pour accélérer la convergence de chacun de ces ratio, comme on l'a fait pour la suite y_{n+1}/y_n.

Exemple 2.2 *Résolution d'un système d'équations non linéaires.*
L'une des utilisations de l'ϵ-algorithme est de fournir une généralisation de la méthode d'Aitken-Steffensen pour des systèmes d'équations non linéaires. La méthode consiste à réécrire le système d'équations $F(X) = 0$ en un système de la forme $X = G(X)$ (X, F et G étant des vecteurs représentant les inconnues ou le système d'équations). En partant d'un vecteur X_0 arbitraire (de préférence proche de la solution du système) on va générer la suite de vecteurs X_k de la manière suivante :
- $U_0 = X_k$, $k = 0, 1, \ldots$,
- $U_{n+1} = G(U_n)$, $n = 0, 1, \ldots, 2p$,
- calculer l'ϵ-algorithme de cette suite de vecteurs U_n,
- $X_{k+1} = \epsilon_{2p}^{(0)}$,

p étant le nombre d'équations du système. La suite des X_k générés par cette méthode est à convergence quadratique. Plusieurs choix pour G étant possibles, il est préférable, bien que non nécessaire, de choisir G de telle sorte que la suite générée $U_{n+1} = G(U_n)$ converge vers la solution du système. L'ϵ-algorithme vectoriel paraît le plus adapté à cet algorithme, mais l'ϵ-algorithme scalaire appliqué à chacune des composantes du vecteur fonctionne aussi.

Exemple 2.3 *Variante des méthodes utilisant l'extrapolation de Richardson.*
Plusieurs méthodes numériques utilisent l'extrapolation de Richardson pour accélérer la convergence de méthodes d'ordre peu élevé. On trouve par exemple son utilisation dans :
- l'évaluation de la dérivée d'une fonction, par accélération de la convergence de la méthode des différences centrales,
- l'intégration numérique, en accélérant la méthode des trapèzes (méthode de Romberg) ou la méthode du point médian,
- l'intégration des équations différentielles, en accélérant la méthode du point milieu (méthode de Gragg-Bulirsch-Stoer).

Dans chacune de ces applications, il est possible de remplacer l'extrapolation de Richardson par l'ϵ-algorithme ou ses généralisations. La suite associée (x_n) devant tendre vers l'infini dans le cas des généralisations de l'ϵ-algorithme, on prendra l'inverse de la suite associée à l'extrapolation de Richardson. Cependant l'extrapolation de Richardson étant particulièrement bien adaptée à accélérer les suites générées par ces méthodes, l'ϵ-algorithme en général est moins performant. Dans certains cas, par exemple le calcul numérique d'une intégrale impropre, l'extrapolation de Richardson échoue à accélérer la convergence, et l'ϵ-algorithme devient alors la méthode de choix.

3 Exercices

Exercice 1 Programmer le Δ^2 d'Aitken sous SCILAB puis l'appliquer au calcul de la somme de la série de Leibniz.

Exercice 2 Retrouver les valeurs de la TABLE 2 en implémentant sous SCILAB le procédé d'Aitken-Steffensen.

Exercice 3 Implémenter l'ϵ-algorithme sous SCILAB puis l'appliquer au calcul de la somme de la série de Leibniz.

Exercice 4 On considère la suite (x_n) donnée par

$$\begin{cases} x_0 = 0, \\ x_{n+1} = x_n + 1 - \dfrac{x_n^2}{2} \end{cases}$$

1. Quelle est sa limite ?
2. Appliquer l'algorithme Δ^2 d'Aitken pour accélerer la convergence.
3. En utilisant $x_0, x_1, \ldots, x_8$, comparer l'erreur des suites obtenues avec ou sans Δ^2 d'Aitken.

Exercice 5 On considère une suite (S_n) qui satisfait

$$(S_{n+1} - S) = \rho_n (S_n - S) \text{ avec } \lim_{n\to+\infty} \rho_n = \rho \text{ et } \rho \neq 1.$$

1. Montrer que la suite (S'_n) donnée par le procédé d'Aitken converge plus vite vers S que la suite originale, c'est-à-dire
$$\lim_{n\to+\infty} \frac{S'_n - S}{S_n - S}.$$
2. Donner une suite divergente (S_n) pour laquelle la suite (S'_n) converge.
Indication. On sait que $\Delta S_n = (\rho_n - 1)(S_n - S)$, trouver une formule similaire pour $\Delta^2 S_n$.

Exercice 6 *La méthode de Kummer.*
La méthode de Ernst Kummer (1810-1893) date de 1837 et consiste à retrancher aux termes d'une série convergente $\sum_{k=1}^{+\infty} a_k$ les termes d'une autre série équivalente, dont la somme $C = \sum_{k=1}^{+\infty} b_k$ est connue. Précisément, on suppose que $\lim_{k\to+\infty} \left(\frac{a_k}{b_k}\right) = \rho \neq 0$. La transformation devient donc

$$\sum_{k=1}^{+\infty} a_k = \rho \sum_{k=1}^{+\infty} b_k + \sum_{k=1}^{+\infty} \left(1 - \rho \frac{b_k}{a_k}\right) a_k = \rho C + \sum_{k=1}^{+\infty} \left(1 - \rho \frac{b_k}{a_k}\right) a_k.$$

1. Pourquoi la seconde série du terme de droite converge-t-elle plus vite ?
2. On souhaite appliquer la méthode à la situation suivante :
$$a_k = \frac{1}{k^2} \text{ et } b_k = \frac{1}{k} - \frac{1}{k+1}.$$
   (a) Calculer C (de tête) et déterminer ρ.
   (b) Calculer la valeur exacte de S.
   (c) Comparer pour $n = 5, 10, 100$ et 1000, les valeurs approchées de s_n et $s'_n = \rho C + \sum_{k=1}^{n} \left(1 - \rho \frac{b_k}{a_k}\right) a_k$ avec S.
   (d) Combien de termes de la série (s'_n) faut-il calculer pour obtenir S avec autant de précision que s_{1000} ?

Remarque 3.1 On peut gagner en rapidité en répétant le processus de Kummer plusieurs fois.

Exercice 7 *Le processus d'extrapolation de Richardson.*
Supposons que les sommes partielles $(s_n)_{n\in\mathbb{N}}$ admettent un développement sous la forme

$$s_n = S + \frac{\alpha_0}{n^p} + \frac{\alpha_1}{n^{p+1}} + \ldots + \frac{\alpha_k}{n^{p+k}} + \circ\left(\frac{1}{n^{p+k}}\right).$$

Le processus d'extrapolation de Lewis Fry Richardson (1881-1953) consiste à calculer la suite $(\sigma_n)_{n\in\mathbb{N}}$ définie par

$$\forall n \in \mathbb{N}^\star,\ \sigma_n = \frac{2^p s_{2n} - s_n}{2^p - 1}.$$

1. Montrer que le procédé de Richardson permet de gagner un ordre dans la précision de la série, c'est-à-dire montrer que
$$\sigma_n = S + \frac{\alpha_1}{2n^{p+1}} + \circ\left(\frac{1}{n^{p+1}}\right).$$
2. Écrire une procédure **Richardson** fonction de (T, p) qui calcule la liste des premiers termes de la suite $(\sigma_n)_{n\in\mathbb{N}}$ à partir de la liste T des premiers termes de $(s_n)_{n\in\mathbb{N}}$ et de l'ordre p.
3. Quel ordre de précision supplémentaire obtient-on en réitérant le processus de Richardson k fois supplémentaires ?
4. Soit α_n l'aire du polygone régulier à n côtés inscrit dans le disque unité. En calculant α_n pour $n = 6.2^k$ avec $1 \leq k \leq 4$, Archimède a pu donner les premières approximations de la constante π.
   (a) Calculer α_n pour ces valeurs.
   (b) Combien de décimales supplémentaires Archimède aurait-il pu obtenir avec les mêmes valeurs de α_n en appliquant le processus de Richardson 4 fois (on prendra $p = 2$ dans ce cas) ?

Exercice 8 *Processus Δ^2 d'Aitken.*

On peut démontrer que sous l'hypothèse $\lim\limits_{k\to+\infty} \dfrac{a_k}{a_{k+1} - a_k} = 0$, le reste à l'ordre n de la série S est équivalent à $\sum\limits_{k=n+1}^{+\infty} a_k \underset{n\to+\infty}{\sim} \dfrac{a_{k+1}^2}{a_{k+2} - a_{k+1}}$. Le processus d'Alexander Aitken (1895-1967) consiste à utiliser cet équivalent en introduisant une suite (σ_n) définie par

$$\sigma_n = s_{n+1} - \frac{a_{n+1}}{a_{k+2} - a_{k+1}}.$$

1. Exprimer σ_n en fonction de s_n, s_{n+1} et s_{n+2}.
2. Écrire une procédure **Aitken** fonction de $(T, list)$ qui renvoie la liste des premiers termes de la suite σ à partir de la liste T des premiers termes de la suite s.
3. La série $\lim\limits_{n\to+\infty} \sum\limits_{k=0}^{n} \dfrac{(-1)^k}{k+1}$ converge vers $\ln(2)$. Combien de décimales exactes obtient-on en calculant s_5, s_{10}, s_{100} et s_{1000} ? Comparer avec σ_5, σ_{10}, σ_{100} et σ_{1000}.

Références

[1] Encyclopédie en ligne Wikipédia.
http ://fr.wikipedia.org/wiki/Delta-2,
http ://fr.wikipedia.org/wiki/Epsilon_algorithme

[2] Ernst HAIRER. *Polycopié d'analyse numérique.*
http ://www.unige.ch/∼hairer/poly/chap1.pdf

[3] Bertrand MEYER. *Travaux Pratiques - Processus d'accélération de convergence de séries.*
http ://www.math.u-bordeaux1.fr/∼meyer/DocMath/MHT304/SujetsTP/TP22.pdf