Sup'Galilée Année 2020/2021
MACS1

Equations non linéaires – critères d'arrêt

Soit f et Φ deux fonctions continues sur un intervalle $[a, b]$ de $\mathbb{R}$. On chercher à trouver

- soit un zéro de f, c'est-à-dire $\alpha \in [a, b]$ tel que
$$f(\alpha) = 0,$$
- soit un point fixe de Φ, c'est-à-dire $\alpha \in [a, b]$ tel que
$$\Phi(\alpha) = \alpha.$$

En général, α ne peut pas être calculé explicitement et on doit utiliser les ordinateurs pour calculer une valeur approchée de α. Les méthodes pour approcher une racine α de f (ou un point fixe α de Φ) sont en général itératives : elles consistent à construire une suite $(x_k)_{k\geq 0}$ telle que

$$\lim_{k\to\infty} x_k = \alpha.$$

1 Convergence

Definition 1.1. *On définit l'erreur absolue à l'étape k par*

$$e_k = x_k - \alpha, \quad k \geq 0.$$

La convergence des itérations est caractérisée par :

Definition 1.2. *On dit qu'une suite $(x_k)_{k\geq 0}$ construite par une méthode numérique converge vers α avec un ordre $p \geq 1$ s'il existe $k_0 \in \mathbb{N}$ tel que*

$$\exists C > 0 : |e_{k+1}| \leq C|e_k|^p, \quad \forall k \geq k_0, \tag{1}$$

avec $C < 1$ si $p = 1$. Dans ce cas, on dit que la méthode est d'ordre p. Notons que si $p = 1$, il est nécessaire que $C < 1$ dans (1) pour que x_k converge vers α. On dit que la convergence est **linéaire** *si $p = 1$ (et $C < 1$),* **quadratique** *si $p = 2$, et* **cubique** *si $p = 3$. La constante C est appelée facteur de convergence de la méthode.*

En général, la convergence dépend du choix de la donnée initiale x_0 : le plus souvent on ne sait montrer la convergence que *localement*, c'est-à-dire seulement pour un x_0 "suffisament proche" de α. Les méthodes qui convergent vers α pour tout $x_0 \in [a, b]$ sont dites *globalement convergentes* vers α.

2 Critère d'arrêt

Soit $\{x_k\}$ une suite qui converge vers un zéro α de f (ou vers un point fixe α de Φ). Soit tol une tolérance fixée. En général on utilise soit un critère basé sur l'incrément, soit un critère basé sur le résidu.

2.1 Contrôle de l'incrément

Les itérations s'achèvent dès que :

$$|x_{k+1} - x_k| < tol \quad \text{ou} \quad |x_{k+1} - x_k| < tol|x_{k+1}|, \tag{2}$$

selon que l'on regarde la difference absolue ou relative de deux itérés successifs. Une question fondamentale est de se demander si les erreurs correspondantes $|x_{k+1} - \alpha|$ ou $|x_{k+1} - \alpha|/|\alpha|$ sont petites. Ce n'est pas le cas si la convergence est très lente. Par exemple si on considère la suite obtenue par l'algorithme de point fixe, $x_{k+1} = \Phi(x_k)$, alors, comme $\Phi(\alpha) = \alpha$, l'erreur $e_{k+1} = x_{k+1} - \alpha$ vérifie (en utilisant le théorème des accroissement finis) : il existe $\xi_k \in]\min(x_k, \alpha), \max(x_k, \alpha)[$ tel que

$$e_{k+1} = \Phi(x_k) - \Phi(\alpha) = \Phi^{'}(\xi_k)e_k.$$

On a donc

$$x_{k+1} - x_k = x_{k+1} - \alpha + \alpha - x_k = e_{k+1} - e_k = (1 - \Phi^{'}(\xi_k))e_k.$$

Si $\Phi^{'}(\xi_k)$ est très proche de $\Phi^{'}(\alpha)$, alors

$$e_k \simeq \frac{1}{1 - \Phi^{'}(\alpha)}(x_{k+1} - x_k).$$

Ainsi, le critère d'arrêt sur l'incrément

- est optimal pour les méthodes telles que $\Phi^{'}(\alpha) = 0$ comme la méthode de Newton,
- est satisfaisant si $-1 < \Phi^{'}(\alpha) < 0$,
- n'est pas satisfaisant si $\Phi^{'}(\alpha)$ proche de 1.

2.2 Contrôle du résidu

Regardons le cas de la recherche d'une racine α de f. Les itérations s'achèvent dès que :

$$|f(x_k)| < tol. \tag{3}$$

On peut montrer (voir [3]) que si α est une racine simple, alors

$$|e_k| \lesssim \frac{1}{|f'(\alpha)|}|f(x_k)|.$$

Ainsi,

— si $|f^{'}(\alpha)| \simeq 1$, alors $|e_k| \simeq tol$ et le critère d'arrêt sur le résidu est satisfaisant,
— si $|f^{'}(\alpha)| \ll 1$, le critère d'arrêt sur le résidu n'est pas adapté car $|e_k|$ peut être grand par rapport à la tolérance tol,
— si $|f^{'}(\alpha)| \gg 1$, le critère est trop restrictif car $|e_k| \ll tol$ dans ce cas.

2.3 Nombre d'itérations maximal

Il est nécessaire, en plus d'un test de type (2) ou (3), d'imposer un nombre d'itérations maximal (au cas où le test portant sur l'incrément ou le résidu ne serait pas vérifié), c'est-à-dire :

$$k < k_{max} \tag{4}$$

où k_{max} est le nombre d'itérations maximal que l'on se fixe.

3 Calcul de l'erreur en pratique

Le calcul numérique de l'erreur peut servir d'une part à vérifier les estimations théoriques, et d'autre part à comparer – en vitesse de convergence– différentes méthodes pour approcher une racine α de f (ou un point fixe α de Φ). Ainsi, en plus du calcul d'une approximation de α, on souhaite calculer $|e_k| = |x_k - \alpha|,\ k \geq 0$. Le problème est que l'on ne connait pas α en général, et même si α est connu, sa valeur risque d'être approchée sur un ordinateur (par exemple si $\alpha = \pi$). Aussi on calculera en pratique un vecteur $\boldsymbol{e}$ de $(k+1)$-ième composante $e_{k+1} = |x_k - \alpha_a|,\ 0 \leq k \leq k_{it}$, où $k_{it} \leq k_{max}$ est le nombre d'itérations, et

— α_a vaut α si α est connu,
— sinon α_a sera une valeur approchée de α aussi précise que possible.

Notons que si $\alpha_a \neq 0$, on regardera plutôt l'erreur relative $e_{k+1} = |x_k - \alpha_a|/|\alpha_a|,\ 0 \leq k \leq k_{it}$.

Représentation de l'erreur en pratique : pour visualiser l'erreur on utilisera deux types de graphes :

• d'une part le tracé du vecteur $\boldsymbol{e}$ (en échelle logarithmique) en fonction du nombre d'itérations, c'est-à-dire le tracé de $log_{10}(\boldsymbol{e})$ en fonction de k, $1 \leq k \leq k_{it}$. Pour cela on peut utiliser la fonction *semilogy* de Matlab :

```
>> semilogy(e)
```

En particulier, on utilisera cette représentation de l'erreur pour observer et comparer les vitesses de convergence des différentes méthodes. Par exemple si $\boldsymbol{e_D}$ est le vecteur erreur correspondant à l'algorithme de dichotomie, et $\boldsymbol{e_N}$ celui correspondant à l'algorithme de Newton, on utilisera la représentation suivante :

```
>> semilogy(e_D)
>> hold on
>> semilogy(e_N)
>> hold off
```

• d'autre part, le tracé de e_{k+1} en fonction de e_k ,$1 \leq k \leq k_{it-1}$, de façon à trouver l'ordre de la méthode. Rappelons qu'une méthode est d'ordre $p \geq 1$ si

$$\exists C > 0 : |e_{k+1}| \leq C|e_k|^p, \quad \forall k \geq k_0$$

où $n_0 \geq 0$ est un entier. En pratique on aura alors

$$|e_{k+1}| \approx C|e_k|^p, \quad \forall k \geq k_0$$

soit encore

$$ln(|e_{k+1}|) \approx ln(C) + p * ln(|e_k|), \quad \forall k \geq k_0$$

Ainsi, les points $(ln(|e_k|), ln(|e_{k+1}|))$ sont sur une droite de pente p, pour $k \geq k_0$. Donc pour trouver l'ordre de la méthode, il faut tracer cette droite et regarder sa pente. Pour cela, on introduit les vecteurs $\boldsymbol{y}$ et $\boldsymbol{x}$ définis par :

$$y_k = ln(e_{k+1}) \text{ et } x_k = ln(e_k), \quad 1 \leq k \leq k_{it-1},$$

et on trace $\boldsymbol{y}$ en fonction de $\boldsymbol{x}$. Sauf éventuellement sur les premières itérations ($k \leq k_0$) la courbe que l'on obtient est une droite de pente p. On peut tracer sur la même figure (en rouge) la droite $y = px$:

```
>> y=log(e(2:length(e)));
>> x=log(e(1:length(e)-1));
>> plot(x,y,'b',x,p*x,'r')
```

Références

[1] J. P. Demailly. *Analyse Numérique et Equations Différentielles*, PUG, 1994.

[3] A. Quarteroni, R. Sacco et F. Saleri. *Méthodes numériques pour le calcul scientifique. Programmes en MATLAB*, Springer, 2000.

[4] J. Rappaz and M. Picasso. *Introduction à l'analyse numérique*, PPUR, 1997.