Eléments de cours sur la résolution de problèmes stationnaires

Objectifs

— résoudre une équation algébrique linéaire ou non par la mise en œuvre de la méthode de dichotomie ;

— résoudre une équation algébrique linéaire ou non par la mise en œuvre de la méthode de Newton.

L'objectif de ce chapitre est de traiter la résolution des problèmes stationnaires, linéaires ou non, se réduisant à une équation algébrique du type $f(x)=0$. Deux méthodes vont être présentées :

— la méthode de dichotomie ;

— la méthode de Newton.

1 Méthodes de résolutions numériques

1.1 Présentation

La résolution mathématique des problèmes scientifiques nécessite l'utilisation de méthodes adaptées. En effet, la complexité des problèmes et leurs formes diverses (équations différentielles, problèmes matriciels, équations algébriques linéaires ou non...) ont conduit à l'élaboration de modèles mathématiques possédant une efficacité adaptée à chaque problème.

L'optimisation des algorithmes de résolution est une part active de la recherche puisque leur efficacité permet des gains évidents en terme de temps de calcul, de puissance de calcul mais également en terme de précision et de robustesse.

Avec l'augmentation de l'utilisation des outils numériques de simulation, la taille des problèmes à résoudre et leur complexité à largement augmenté, et pose aujourd'hui de nombreux problèmes à l'industrie qui se voit dans l'obligation d'adapter les problèmes et les méthodes de résolution associées à la puissance des calculateurs à disposition.

1.2 Performances des algorithmes de résolution

Les critères de performance des algorithmes de résolution sont essentiels pour permettre d'évaluer l'efficacité d'une méthode de résolution face à un problème numérique. On peut pour cela dégager trois critères principaux : la convergence, la rapidité et la robustesse.

1.2.1 Convergence

La convergence d'un algorithme itératif représente la capacité de la suite itérative à s'approcher à une distance donnée de la solution du système. On utilise généralement l'ordre de convergence qui utilise une analogie aux fonctions polynomiales pour décrire la convergence d'une suite.

Définition

On dit qu'une suite $(u_n)_{n\in\mathbb{N}}$ converge avec une précision ε vers $l \in \mathbb{R}$ en un nombre n_0 d'itérations si et seulement si :

$$\forall n \in \mathbb{N}, n \geqslant n_0 \Rightarrow |u_n - l| < \varepsilon$$

Définition

Soit x_n la valeur de la n-ième itération de l'algorithme. Soit r la solution exacte. Soit ε_n l'erreur commise à la n-ième itération. On appelle ordre de convergence le nombre $p \in \mathbb{N}$:

$$\exists a \in \mathbb{N} / \forall n \in \mathbb{N}, n > a \Rightarrow \varepsilon_{n+1} \approx \varepsilon_n^p$$

1.2.2 Rapidité

La rapidité d'un algorithme de résolution est le temps de calcul nécessaire à la convergence de cet algorithme vers la solution. Ce temps de calcul est déterminé à partir du nombre d'opérations mathématiques effectuées. Le temps effectif dépend également de la capacité de l'ordinateur à effectuer ces opérations.

Définition

La rapidité d'un algorithme de résolution est déterminée par le nombre d'opérations de calculs élémentaires à réaliser (complexité) pour déterminer la solution du problème posé.

1.2.3 Robustesse

La robustesse d'un algorithme de résolution est sa capacité à converger vers la solution en fonction des conditions initiales données.

Définition

On dit qu'un algorithme de résolution est robuste si sa convergence est indépendante des paramètres de calcul (pas de temps, conditions initiales).

2 Méthode de dichotomie

2.1 Mise en œuvre

La méthode de dichotomie repose sur deux propriétés simples :

— lorsque f est une fonction continue sur un intervalle $[a_0, b_0]$, à valeurs réelles, avec $f(a_0)$ et $f(b_0)$ de signe opposés, le théorème des valeurs intermédiaires nous assure que f s'annule entre a_0 et b_0 ;

— si une fonction f est strictement monotone et s'annule sur un intervalle $[a_0, b_0]$, alors la valeur d'annulation est unique.

Le méthode de dichotomie nécessite donc d'avoir une fonction f continue et strictement monotone sur un intervalle $[a_0, b_0]$, et s'annulant sur cet intervalle.

Méthode

— choisir un intervalle contenant la solution ($[a_0, b_0]$ sur la figure 1) ;

— calculer la valeur de la fonction au milieu de cet intervalle (le milieu est noté c_0 sur la figure 1) ;

— isoler alors le demi-intervalle qui contient la solution ($[a_1, b_1]$ sur la figure 1) ;

— réitérer la méthode jusqu'à obtenir un intervalle contenant la solution d'une largeur (précision) donnée ($(b_2 - a_2) <$ précision sur la figure 1).

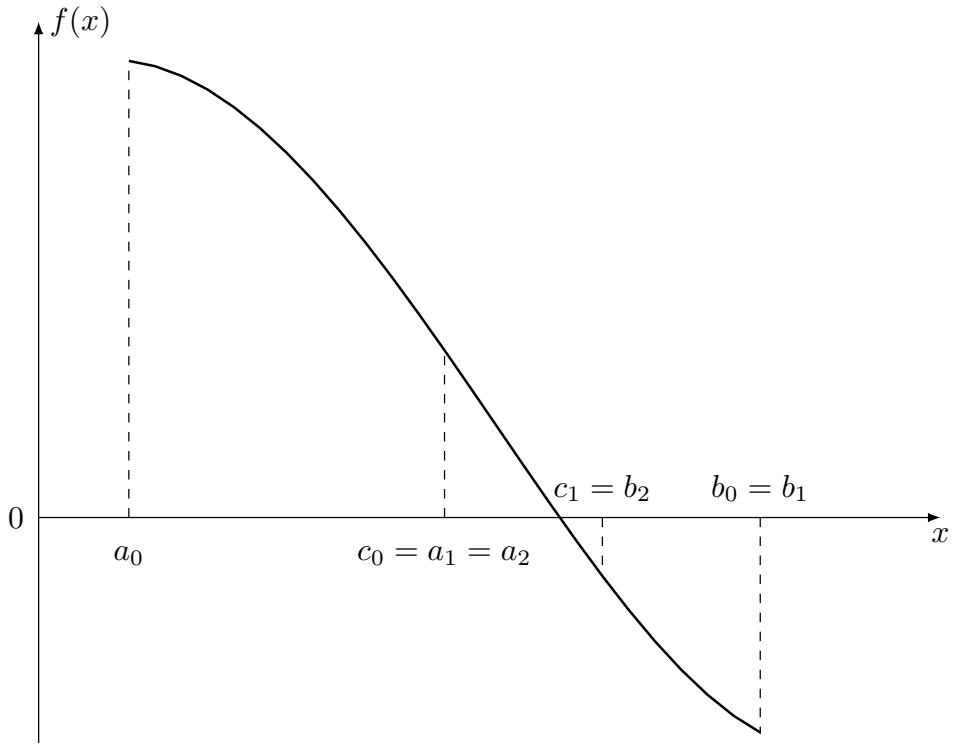

FIGURE 1 – Illustration de la méthode de la dichotomie.

Remarque

Pour choisir le bon demi-intervalle, on fera le produit des valeurs de la fonctions aux bornes : $f(a_i)\cdot f(c_i)$:

— si ce produit est positif, la fonction a le même signe sur l'intervalle, elle ne passe donc pas par 0 sur cet intervalle ;

— si ce produit est négatif, la fonction change de signe sur l'intervalle, elle passe donc par 0 sur cet intervalle.

2.2 Algorithme

La fonction `dichotomie` est alors :

```python
def dichotomie(f,a,b, epsilon ):
    # les données (ou entrées) sont les suivantes :
    # 1. la fonction f (continue et monotone sur l'intervalle d'étude)
    # 2. la borne inférieure de l'intervalle a
    # 3. la borne supérieure de l'intervalle b
    # 4. la précision voulue du résultat epsilon
    while abs(a-b)>epsilon:
        c=(a+b)/2
        if f(a)*f(c)<=0:
            b=c
        else :
            a=c
    return (a+b)/2
```

2.3 Convergence

La convergence de la méthode est simple à démontrer, puisqu'à chaque itération, on sait que la racine r solution du problème $f(x)=0$ se trouve dans le segment $[a_i, b_i]$. Soit $[a_i, b_i]$ l'intervalle à l'itération i. En définissant ε_i tel que $\varepsilon_i = b_i - a_i$. L'intervalle sur lequel se trouve la racine étant divisé par 2 à chaque itération, alors on obtient rapidement la relation :

$$\varepsilon_{i+i} = \frac{\varepsilon_i}{2} = \frac{\varepsilon_0}{2^{i+1}} = \frac{b-a}{2^{i+1}}$$

La convergence se fait donc de manière linéaire.

Pour le test d'arrêt de l'algorithme, on peut estimer le nombre d'itérations n nécessaires à obtenir la précision souhaitée :

$$\varepsilon \leq \frac{b-a}{2^n} \quad \Rightarrow \quad 2^n \leq \frac{b-a}{\varepsilon} \quad \Rightarrow \quad n \geq \frac{\ln\left(\frac{b-a}{\varepsilon}\right)}{\ln 2}$$

La convergence est lente mais robuste : il suffit que la fonction f soit définie et continue sur l'intervalle $[a, b]$ et que cette fonction n'admette qu'une seule racine sur cet intervalle $[a, b]$ pour assurer la convergence.

3 Méthode de Newton

3.1 Mise en œuvre

La méthode de Newton est basée sur l'utilisation de la dérivée première de la fonction f. Celle-ci permet, à l'aide d'un développement de Taylor à l'ordre 1, d'évaluer une valeur de la fonction en un point approché de la racine de l'équation $f(x)=0$. La méthode repose alors sur les propriétés suivantes :

— le développement de Taylor d'une fonction f de classe C^1 à l'ordre 1 sur un intervalle $[a, b]$ donne :

$$f(b) = f(a) + f'(a)\cdot(b-a) + \circ(b-a)$$

— à partir d'un point initial x_0, le point approché x_1 donnant $f(x_1) = 0$ d'après le développement de Taylor précédent permet d'avoir la relation :

$$0 = f(x_1) = f(x_0) + f'(x_0)\cdot(x_1 - x_0) \quad \Rightarrow \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

On obtient alors x_1, une approximation de la solution de l'équation $f(x)=0$.

Méthode

— on choisit un point de départ x_0 (figure 2) ;

— on trace la tangente en x_0 de la courbe représentant f, dont l'équation est $y = f'(x_0)\cdot(x-x_0)+f(x_0)$;

— on détermine le point x_1, intersection de la tangente et de l'axe des abscisses d'après la formule de Taylor à l'ordre 1 : $0 = f(x_1) = f(x_0) + f'(x_0)\cdot(x_1 - x_0) \quad \Rightarrow \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

— on réitère la méthode à partir du point x_1 jusqu'à convergence : $|f(x_i)|$ <précision

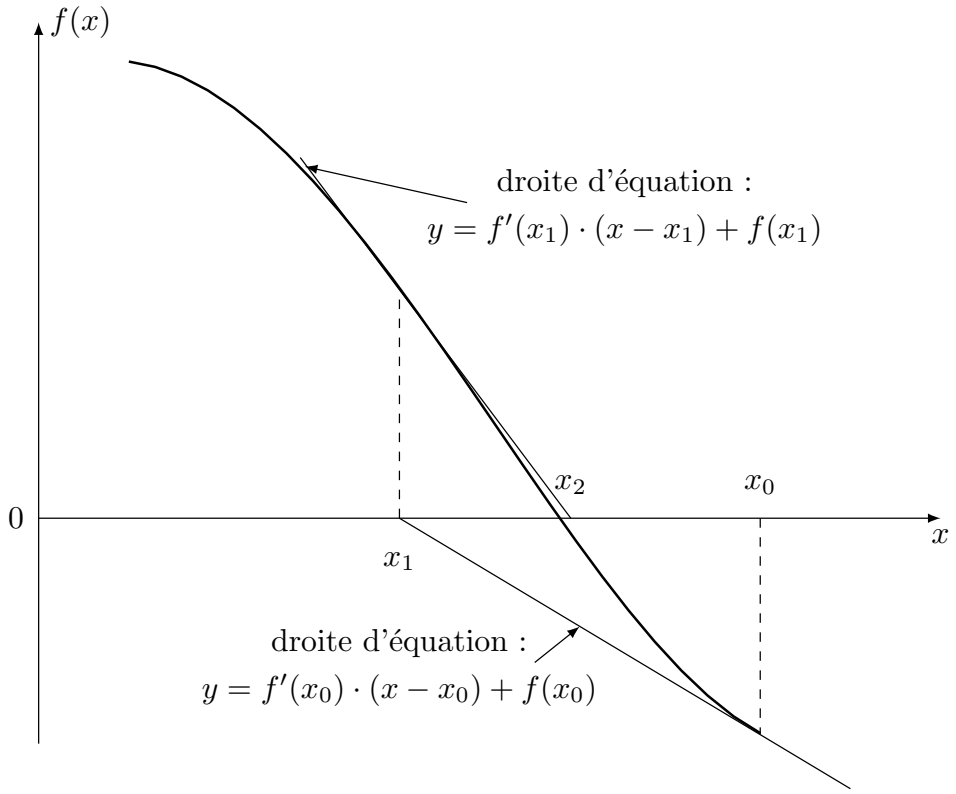

FIGURE 2 – Illustration de la méthode de Newton.

3.2 Algorithme

La fonction `newton` est alors :

```python
def newton(f,fp,x0,delta):
    # les données (ou entrées) sont les suivantes :
    # 1. la fonction f (de classe 1)
    # 2. sa dérivée fp (non nulle)
    # 3. la valeur initiale x0
    # 4. la valeur f(résultat) pour laquelle on arrête le calcul : delta
    x = x0
    while abs(f(x))> delta:
        fp1 = fp(x)
        x = x - f(x) / fp1
    return x
```

3.3 Convergence

La convergence de cette méthode se démontre de la manière suivante. Soit r la racine de la fonction f sur l'intervalle d'étude et x_n, la énième valeur calculée par itération de Newton. Notons alors $\varepsilon_n = x_n - r$. Le développement de Taylor à l'ordre 2 au voisinage de r donne :

$$f(x_n) = f(r) + f'(r)\cdot\varepsilon_n + f''(r)\cdot\frac{\varepsilon_n^2}{2} + \circ(\varepsilon_n^2)$$
$$f'(x_n) = f'(r) + f''(r)\cdot\varepsilon_n + \circ(\varepsilon_n)$$

Le calcul de l'erreur ε_{n+1} donne alors :

$$\varepsilon_{n+1} = x_{n+1} - r = x_n - r - \frac{f(x_n)}{f'(x_n)} = \varepsilon_n - \frac{f'(r)\cdot\varepsilon_n + f''(r)\cdot\frac{\varepsilon_n^2}{2} + \circ(\varepsilon_n^2)}{f'(r) + f''(r)\cdot\varepsilon_n + \circ(\varepsilon_n)}$$

$$\varepsilon_{n+1} = \frac{\varepsilon_n\cdot\left(f'(r) + f''(r)\cdot\varepsilon_n + \circ(\varepsilon_n)\right) - f'(r)\cdot\varepsilon_n - f''(r)\cdot\frac{\varepsilon_n^2}{2} - \circ(\varepsilon_n^2)}{f'(r) + f''(r)\cdot\varepsilon_n + \circ(\varepsilon_n)} = \frac{\varepsilon_n^2}{2}\cdot\frac{f''(r)}{f'(r)} + \circ(\varepsilon_n^2)$$

La convergence est donc quadratique.

Pour le test d'arrêt de l'algorithme, le problème est beaucoup plus compliqué puisque la dérivée de f peut entraîner une division par 0. En pratique, une conditions suffisante pour s'assurer de la convergence est d'avoir f de classe C^1, convexe sur un intervalle I sur lequel f s'annule et de choisir x_0 dans I.