L1 Maths-Info

TP 1 : Résolution d'équations non linéaires par méthodes numériques

Objectif : Comparer différentes méthodes numériques pour résoudre des équations non linéaires de la forme

$$f(x) = 0 \qquad (E)$$

où f est une fonction numérique et x **est appelé racine de f.**

1. Affichage des nombres à virgule flottante en C++

Pour pouvoir afficher en C++ un nombre en virgule flottante avec un certain nombre de chiffres après la virgule, inclure dans votre programme les instructions suivantes :

```
#include <iostream>
#include <iomanip>

using namespace std;

int main() {
int p = 5;  // Nombre de chiffres après la virgule souhaité
cout<<fixed<<setprecision(p)<<endl;  // Affichage en notation
décimale : affiche p chiffres après la virgule

double n = 1.12345678;
cout << "n = " << n << endl;
return 0;
}
```

2. Détermination de la précision

- Ecrire une fonction `int precision( double e)` qui reçoit en paramètre un réel e s'exprimant sous la forme 10^{-p} et qui renvoie l'entier *p* (avec p>0).

Exemple. `precision(0.0001)` doit renvoyer 4

Testez votre fonction en remplaçant la variable `p` dans le `main` par un appel à `precision(eps)`, où `eps` est une puissance de 10 inférieure à 1.

3. Méthode des approximations successives (méthode du point fixe)

La méthode des approximations successives est une technique itérative utilisée pour trouver une solution approchée à l'équation (E). Pour résoudre cette équation (E), on la reformule sous la forme :

$$F(x) = x$$

La valeur *x* vérifiant cette équation est appelé point fixe de F. Remarquez que F n'est pas unique. Par exemple, pour l'équation $f(x) = x^3 + x - 1 = 0$, on peut choisir :

- $F(x) = 1 - x^3$
- $F(x) = \frac{1}{1+x^2}$

Algorithme du point fixe.

1. Choisir une approximation initiale x_0 de la racine de l'équation (E).
2. Calculer itérativement x_1, x_2, x_3, … en utilisant la suite $x_{n+1} = F(x_n)$.
3. Arrêter l'algorithme si :
   - Le nombre d'itérations dépasse un maximum prédéfini MAXITER, ou
   - La différence $|x_{n+1} - x_n|$ devient inférieure à une tolérance `eps` (précision souhaitée).

Exemple d'implémentation

Pour résoudre, l'équation $f(x) = 1 - \sin(x)\cos(x) - x = 0,$ on définit une fonction F par :

```
double F(double x)
{
return ( 1 - sin(x)*cos(x) );
}
```

- Ecrire une fonction `int pointFixe(double &s, double eps)` qui demande une valeur initiale s qui sera modifié pour contenir en sortie la solution de l'équation, eps est la précision souhaitée. La fonction renvoie le nombre d'itérations effectué.

- Tester cette fonction en affichant la solution s, le nombre d'itérations, et la valeur f(s).

4. Méthode d'Aitken pour accélérer la convergence

La méthode d'Aitken, souvent appelée procédé Δ^2 d'Aitken, permet d'accélérer la convergence d'une suite obtenue par la méthode du point fixe.

Algorithme d'Aitken

Données :

- Une valeur initiale x_0.
- Une tolérance `eps` pour le critère d'arrêt.
- Un nombre maximal d'itérations (`MAXITER`).

1. Calculer $x_1 = F(x_0)$.
2. Calculer $x_2 = F(x_1)$.

Pour n = 0,1, 2, …

- Calculer le dénominateur $d = |x_{n+2} - 2x_{n+1} + x_n|$
- **Si** d<eps, **alors** la transformation d'Aitken ne peut pas être appliquée : arrêtez les itérations.
- **Sinon**
  - $y_n = x_n - \frac{(x_{n+1}-x_n)^2}{x_{n+2}-2x_{n+1}+x_n}$
  - $x_n = y_n, \quad x_{n+1} = F(x_n), \; x_{n+2} = F(x_{n+1})$
- Arrêter si le nombre d'itérations dépasse MAXITER ou si $|x_{n+1} - x_n| < e$ps

- Ecrire une fonction `int pointFixeAitken (int &s, double eps)` qui prend une valeur initiale s (qui sera modifié pour contenir la solution), la précision souhaitée eps. La fonction utilise l'algorithme d'Aitken, elle renverra -1 si le dénominateur est égal à 0, sinon elle renverra le nombre d'itérations effectué.
- Tester cette fonction en affichant la solution s, le nombre d'itérations, et la valeur f(s).

5. Méthode de Newton

La méthode de Newton permet de déterminer la racine d'une équation non linéaire $f(x) = 0$. Elle utilise la suite :

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

On doit donc déclarer la fonction f, par exemple :

```
double f(double x)
{
return ( 1 - sin(x)*cos(x) - x );
}

double fp(double x)
{
return (…..); //la dérivée de f
}
```

- Reprenez le programme des approximations successives pour l'adapter à la méthode de Newton. Il ne faudrait pas oublier de prendre en compte le cas où la dérivée s'annule en cours d'itérations.

6. Méthode de la sécante

Cette méthode permet également de déterminer la racine d'une équation non linéaire sans utiliser la dérivée de la fonction. Elle s'appuie sur la suite :

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

C'est une suite d'ordre 2. Elle nécessite, contrairement aux 2 suites précédentes, la saisie de 2 valeurs initiales.

- Reprenez le programme de Newton pour l'adapter à la méthode de la sécante. Là encore, il ne faudrait pas oublier de tenir compte du cas où $f(x_n) = f(x_{n\_1})$.

- Une fois que tous les programmes marchent, intégrez les trois méthodes dans un même programme et comparez les pour les mêmes valeurs initiales.

N.B. N’oubliez pas de prêter attention au temps de calcul. Entre autres, évitez de faire appel aux fonctions plusieurs fois pour la même valeur...