FONCTIONS — TABLEAUX — ENTRÉES-SORTIES

Exercice 1

Soit la fonction polynomiale

$$f(x) = ax^2 + bx + c \tag{1}$$

où a, b et c sont des réels non nuls. Dans un programme C, on souhaite évaluer la fonction f aux points $x_i = x_0 + i\delta x$ où $i = 1, 2, \cdots, N$ et δx est le pas de discrétisation spatiale en général petit. N est initialisé à l'aide d'un `#define` placé en début de programme (par exemple $N = 10$ pour débuter). On évalue f en $x + \delta x$ à l'aide de la relation suivante :

$$f(x + \delta x) \approx f(x) + f'(x)\delta x \tag{2}$$

1. Fixer à l'aide de `#define` les valeurs de a, b, c, x_0 et δx.
2. Calculer f en x_0 à l'aide de la formule (1). On utilisera ce résultat pour initialiser la variable notée `fapprox` de type `double` qui stockera lesapproximations successives de f.
3. Écrire la fonction `fprime` qui retourne la dérivée de f en un point x entré comme argument. Utiliser le prototype suivant : `double fprime(double x) ;`
4. Écrire la fonction `remplace` qui remplace la valeur approchée de f, `fapprox`, en x par celle en $x+\delta x$, en utilisant la formule (2). Utiliser le prototype suivant : `void remplace(double *fapprox, double x) ;`.
5. À l'aide d'une boucle itérative, évaluer pas à pas les approximations successives de f dans l'intervalle $[x_0, x_0 + N\delta x]$.
6. Écrire dans un fichier les valeurs de x, $f(x)$, et de son approximation `fapprox` sous la forme de trois colonnes.
7. Visualiser la fonction $f(x)$ et son approximation à l'aide du logiciel `gnuplot`.

Exercice 2

Dans un programme C,

1. initialiser à l'aide de tableaux de type `double` deux vecteurs $\vec{a}$ et $\vec{b}$ définis de la manière suivante,

$$\vec{a} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ N \end{pmatrix} \qquad \text{et} \qquad \vec{b} = \begin{pmatrix} N \\ N-1 \\ N-2 \\ \vdots \\ 1 \end{pmatrix}$$

   N est fixé à 10 par un `#define`.
2. Effectuer les opérations vectorielles : $\vec{c} = \vec{a} + \vec{b}$; $\vec{c} = \vec{a} - \vec{b}$.
3. Afficher le résultat à l'aide de la fonction `AfficheVecteur`. Respecter le prototype suivant : `void AfficheVecteur(char *nom, double u[], int n) ;`.
   Pour afficher par exemple un vecteur $\vec{t} = (2.1; 1.1; -3.9)$, on appelle la fonction de la manière suivante : `AfficheVecteur("vecteur t",t,3)`. Ce qui a pour résultat d'afficher dans le terminal de commandes les lignes suivantes :

```
vecteur t
2.1
1.1
-3.9
```

4. À l'aide de boucles imbriquées, initialiser une matrice tridiagonale $N \times N$ de la façon suivante :
$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$
5. Sur le même modèle que la fonction `AfficheVecteur`, écrire une fonction,
`void AfficheMatrice(char *nom, double M[][N], int n, int m) ;`
qui affiche dans le terminal une matrice $m \times n$ sous la forme d'un tableau de m lignes et n colonnes.
6. Effectuer le produit matrice vecteur suivant : $\vec{c} = \mathbf{M}\vec{b}$ et afficher le résultat à l'aide de la fonction `AfficheVecteur`.

Exercice 3

Dans un fichier `g.data`, on a stocké, à raison d'une valeur par ligne du ficher, les valeurs d'une fonction g évaluée aux points $x_i = i\delta x$, avec $i = 0, 1, 2 \cdots, n$ où n est donné au début du fichier. Le pas d'espace vaut $\delta x = 0.01$ et est défini en début de programme à l'aide d'un `#define`.

1. Écrire un programme en C permettant de lire dans le fichier `g.data` les valeurs de $g(x_i)$ et de les stocker dans un tableau prédimensionné à M (constante définie par un `#define`).
2. On s'assurera grâce à un test logique que $M > n$. Si ce n'est pas le cas le programme s'arrête avec un message d'erreur adéquat.
3. Écrire la fonction `deriveeP` qui calcule la dérivée première d'une fonction à l'aide de la relation :
$$g'(x_i) \approx \frac{g(x_{i+1}) - g(x_i)}{\delta x}$$
Le prototype de la fonction `deriveeP` est le suivant :
`double deriveeP(int i, double delta, double g[]) ;`
la valeur retournée est la valeur de la dérivée en x. Noter bien que la dérivée première ne peut être évaluée en x_n.
4. De la même manière, écrire la fonction `deriveeS` qui calcule la dérivée seconde d'une fonction à l'aide de la relation :
$$g''(x_i) \approx \frac{g(x_{i+1}) - 2g(x_i) + g(x_{i-1})}{\delta x^2}$$
Le prototype de la fonction `deriveeS` est le suivant :
`double deriveeS(int i, double delta, double g[]) ;`
la valeur retournée est la valeur de la dérivée seconde en x. Noter bien que la dérivée seconde ne peut être évaluée en x_0 et x_n.
5. Écrire dans un fichier `derivees.data`, les valeurs de x_i, $g(x_i)$, $g'(x_i)$ et $g''(x_i)$ sous le format quatre colonnes pour $i = 1, 2, \cdots, n-1$.
6. Représenter avec le logiciel `gnuplot` les courbes correspondantes.

Exercice 4 : pour les plus avancés

Écrire une fonction de type `double` qui calcule le déterminant d'une matrice 3×3. Respecter le prototype :
`double determinant(double mat[][3]) ;`
La tester dans un programme de votre choix.

Exercice 5 : pour les encore plus avancés

Écrire un programme en C permettant de résoudre l'équation différentielle du première ordre :

$$ay' + y = 0$$

où a est un réel positif de votre choix.

1. Se servir de la fonction :
   `void pdx(double *y, double delta)`
   dont le rôle est de calculer $y(x + \delta x)$ à partir de la valeur de $y(x)$ rentré en argument.
2. La solution analytique de cette équation est notée $y_a(x)$.
3. Ne pas se servir de tableau : à chaque pas δx, remplacer la valeur de y en x par celle en $x + \delta x$.
4. La résolution pas à pas de l'équation différentielle nécessite la connaissance de $y(0)$: prendre $y(0) = 1$.
5. Dans un fichier, écrire pour chaque valeur de x les valeurs de x, $y(x)$ et $y_a(x)$ sur trois colonnes.
6. Visualier $y(x)$ et $y_a(x)$ à l'aide du logiciel `gnuplot`.