Programmation en langage C

7 février 2015

1 factorisation d'Horner

(TD2) Soit le polynôme :
$P(x) = 0.01.x^9 - 0.03.x^8 + 0.1.x^7 - 0.3.x^6 + 0.2.x^5 + 0.4.x^4 - 5.x^3 + 2.x^2 + x - 3$

1.1 Forme canonique

Testez le programme « `Poly_canonique.c` ». Ce programme affiche 10 valeurs de P(x) pour : $x \in [0, 1]$. Ce programme utilisera pour calculer P(x), la fonction `poly(float x,float *A)` où `A` est un tableau contenant les coefficients du polynôme.

1.2 Forme d'Horner

Testez « `poly_Horner.c` » avec la fonction `Honer(float x,float *A)` qui calcul la valeur de P(x) suivant la factorisation d'Horner.

1.3 Test de rapidité

Commentez les instructions `printf` des programmes précédents. Avec M=10000 (rien ne vous interdit d'essayer des valeurs supérieures...) testez la rapidité de ces programmes avec la commande :

```
login@machine$ time ./Le_programme_a_tester
```

1.4 GSL

Testez le programme « `poly_gsl.c` » à l'aide de la bibliothèque gsl.

```
#include <gsl/gsl_poly.h>
double gsl_poly_eval (const double c[], const int len,
                      const double x);
// calcul P(x), c[] tableau contenant les coefficients de
// P(x) dans l'ordre croissant
```

1.5 Calcul interactif

Ajoutez au programme précédent une phase où l'on pourra saisir le degré du polynôme et ces coefficients. Vous devez utiliser un tableau dynamique pour les coefficients et donc utiliser la fonction « `malloc` ».

```
#include <stdlib.h>
float *coeff // Pointeur sur les coefficients
if((coeff=malloc(sizeof(float))==NULL)
   // Affichage d'une erreur
// si non utilisation de coeff. ex: coeff[]=1
```

1.6 Graphe

En vous inspirant du programme c2octave.c (TD2) écrivez un programme qui trace le graphe $y = P(x)$ pour $x \in [0, 1]$ et produit le fichier « `poly.png` ».

2 Matrice

2.1 Opérations élémentaires

Testez le programme « `sommeMat.c` » qui permet d'additionner ou soustraire deux matrices et d'afficher la matrice résultat.
Modifiez le programme pour obtenir le résultat suivant :

$x \quad x$
$x \quad x$
$x \quad x$

Les x doivent être remplacer par les éléments de la matrice résultat.

Lorsque vous obtiendrez le bon résultat redirigez la sortie de votre programme dans le fichier « `matrice.txt` ».

fonctions à utilisées pour ce programme :

```
#include <gsl/gsl_matrix.h>
gsl_matrix * gsl_matrix_alloc (size_t n1, size_t n2)
//fonction pour réservé de la mémoire
```

```
4 int gsl_matrix_add (gsl_matrix * a, const gsl_matrix * b)
5 / Addition de matrice
6 int gsl_matrix_sub (gsl_matrix * a, const gsl_matrix * b)
7 / Soustraction de matrice
8 void gsl_matrix_free (gsl_matrix * m)
9 / Libération mémoire
```

2.2 produit

Testez le programme « `prodMat.c` ». Modifiez le programme « `prodMat.c` » qui permet de saisir au clavier les deux matrices à multiplier en utilisant la fonction saisi à compléter :
exemple :

```
1 int saisir(gsl_matrix *m){
2    // Demande et saisie du nombre de ligne et de colonne.
3    // Allocation mémoire pour la matrice.
4    // Saisie des éléments en utilisant la fonction "gsl_matrix_fscanf" .
5    // Consultez la documentation de gsl.
6 }
7 // pour afficher le résultat utilisez la fonction "gsl_matrix_fprintf".
```