Examen de calcul numérique en langage C

Lundi 9 janvier 2006

Questions sur le cours : *aucun document papier ou électronique n'est autorisé.*
Durée maximale : *1 heure.*
Rendre la copie : *avant de passer à la partie pratique.*
Exercice pratique : *document de cours et notes manuscrites autorisés.*
Durée prévue : *2 heures.*

1 Questions sur le cours

1. En langage C, on veut écrire une fonction spécifiée par `void compare (double x)` pour comparer la valeur de la variable `x` à l'unité et écrire à l'écran (en français) le résultat de la comparaison pour chaque cas possible (plus petit, égal, plus grand) : **écrire cette fonction**.
2. Un (très) mauvais programmeur à écrit le programme suivant pour calculer $S = \sum_1^{N_{MAX}} \frac{1}{n^2}$ et comparer à la valeur exacte $\frac{\pi^2}{6} = \sum_1^{\infty} \frac{1}{n^2}$

```
/*  Programme plein de fautes de syntaxe et de programmation /*
#INCLUDE [stdio.h]
#INCLUDE [stdlib.h]
#include <math.h>

#DEFINE NMAX=100
#DEFINE PRECISION=1.e-5

int main()
{
  int i ,n ,fn ,valeur_exacte;
  double pi;

  printf(" Calcul de la serie de Riemann" \n);
  pi = atan(1.);
  valeur_exacte = pi*pi / 6.;   printf("valeur exacte: \n  ,valeur_exacte")

  for (n=1;  n++; n<NMAX);{
    fn = 1 / (n^2)
    somme = somme + fn
    if (fn < PRECISION);{
      printf("pour %d termes, somme= %d \n ,n ,somme"  )
      return (0)
    }

 printf("\n %d termes ne suffisent pas pour une precision de %e \n"
, nmax ,precision);
}
```

Recopier le programme sur votre copie en corrigeant les fautes (qui ne sont pas seulement des fautes de syntaxe) **et sur la version originale que l'on joindra à la copie, les entourer**.

2 Exercice pratique

2.1 Introduction

On se propose de comparer les performances de plusieurs méthodes pour calculer numériquement la dérivée d'une fonction. Pour cela, on examine deux cas pour lesquels la dérivée est connue analytiquement :

$$f(x) = x^2 + 2x, \qquad \text{et} \qquad g(x) = \sin x.$$

2.1.1 Algorithmes de dérivation

Il est facile, à partir du développement de Taylor de la fonction $y(x)$, de montrer que sa dérivée y' peut être calculée en utilisant l'une des formules suivantes :

$$y'_0 = \frac{y_1 - y_0}{h} + o(h), \qquad y'_0 = \frac{y_1 - y_{-1}}{2h} + o(h^2), \qquad y'_0 = \frac{-y_2 + 8y_1 - 8y_{-1} + y_{-2}}{12h} + o(h^4),$$

où y_n est la valeur de la fonction en $x + nh$, $n \in \mathbb{Z}$.

2.2 Travail demandé : programmation et analyse des résultats

2.2.1 Instructions pratiques

1. Dans votre répertoire principal, créer un répertoire `examen` et travailler dans ce répertoire.
2. Utiliser un nom de fichier pour votre programme de la forme : *nom_prenom*`.c` (sans accent)
3. Indiquer en commentaire au début du programme, vos noms et prénoms, la nature de l'épreuve, la date et l'objet du programme.

2.2.2 Tabulation de la fonction

1. Créer les fonctions `double ma_fonction(double x)` et `double ma_derivee(double x)` donnant les valeurs exactes de la fonction f et de sa dérivée,
2. Prendre comme intervalle d'étude $x \in [2h, 10 - 2h]$ et tabuler la fonction dans un tableau `y[NMAX]`, avec `y[n]` $= nh$. On commencera avec `NMAX` $= 11$.
3. Écrire ligne par ligne les valeurs de x et de y.
4. Faire de même pour la fonction g en modifiant `ma_fonction` et `ma_derivee`.

2.2.3 Évaluation de la dérivée

Dans trois boucles successives, utiliser les trois formules du 2.1.1 pour évaluer la valeur de la dérivée et la comparer à la valeur exacte en calculant la valeur absolue de la différence.

1. Faire l'expérience numérique successivement pour f et g et enregistrer[1] les résultats dans les fichiers `deriv_f.dat` et `deriv_g.dat` .
2. Comment peut-on expliquer les résultats obtenus avec f ?
3. En faisant varier `NMAX` pour la fonction g, étudier pour chacun des trois algorithmes la dépendance de la précision obtenue en fonction de h. On pourra par exemple calculer le logarithme à base 10 de la valeur moyenne de la valeur absolue de l'erreur en fonction de h.

2.3 Incertitude sur le calcul de la dérivée

On suppose maintenant que la fonction f n'est connue que de façon approchée. Pour simuler cette situation, on rajoute à $f(x)$ un «bruit additif» sous la forme d'un terme supplémentaire donné par la fonction[2] système `rand`. On prendra une suite pseudo-aléatoire de nombres dans l'intervalle $[0, 1]$ et on recommencera l'étude de la précision obtenue en fonction de h. Conclusions ?

---

[1] On pourra par exemple diriger la sortie du programme `a.out` vers le fichier `deriv_f.dat`, en utilisant la commande `a.out > deriv_f.dat`

[2] On rappelle qu'elle est initialisée par `srand(seed)` et qu'on obtient une suite entre 0 et `b` avec `b * (double) rand() / RAND_MAX`