ALGORITHMIQUE EN STI2D-STL

Le programme
Exercice : autour de *e*
TP : Trapèzes contre Monte-Carlo

Le programme

- BO n°18 du 6 mai 2010 : « Dans tous les programmes de mathématiques des classes de [...] terminale (à partir de la rentrée 2011) sont ajoutées les deux dernières sections du programme de seconde intitulées respectivement :
  - Algorithmique (objectifs pour le lycée)
  - Notations et raisonnement mathématiques (objectifs pour le lycée) »

Le programme

- « les élèves sont entraînés :
  - à décrire certains algorithmes en langage naturel ou dans un langage symbolique ;
  - à en réaliser quelques uns à l'aide d'un tableur ou d'un petit programme réalisé sur une calculatrice ou avec un logiciel adapté ;
  - à interpréter des algorithmes plus complexes. »

Le programme

« L'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes posés doivent être en relation avec les autres parties du programme (fonctions, géométrie, statistiques et probabilité, logique) mais aussi avec les autres disciplines ou la vie courante. »

Le programme

« Instructions élémentaires (affectation, calcul, entrée, sortie)

Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables :

- d'écrire une formule permettant un calcul ;
- d'écrire un programme calculant et donnant la valeur d'une fonction ;
- ainsi que les instructions d'entrées et sorties nécessaires au traitement.

Le programme

Boucle et itérateur, instruction conditionnelle

Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables :

- de programmer un calcul itératif, le nombre d'itérations étant donné ;
- de programmer une instruction conditionnelle, un calcul itératif, avec une fin de boucle conditionnelle. »

Exercice : autour de *e*

- On considère l'algorithme ci-contre qui, à partir d'un entier naturel *k*, en calcule un autre noté *p*.
- Mettre en œuvre l'algorithme, calculer les valeurs de *p*, pour *k* de 1 à 20.
- Donner une formule permettant de relier l'entier *p* calculé à l'entier *k* choisi.

Entrées
Saisir *k*
Initialisation
p prend la valeur 1
Traitement
Pour *i* de 1 à *k*
p prend la valeur $p \times i$
FinPour
Sortie
Afficher *p*

Voir le fichier Scilab : Factorielle

Exercice : autour de *e*

- On considère la suite (u_n) définie pour n≥1 par

$$u_n = 1 + \frac{1}{1} + \frac{1}{1\times 2} + \ldots + \frac{1}{1\times 2\times \ldots \times n}$$

- Calculer u_1, u_2 et u_3.
- En modifiant l'algorithme précédent, calculer les valeurs de u_n pour n de 1 à 20.

Voir le fichier Scilab : ApproximationE

- On admet que la suite (u_n) admet pour limite *l*. Conjecturer la valeur de *l*.

Pour l'utilisation (facultative) d'une « fonction » (sous-programme), voir les fichiers Scilab : Factorielle_fonction et ApproximationE_fonction

Exercice : autour de *e*

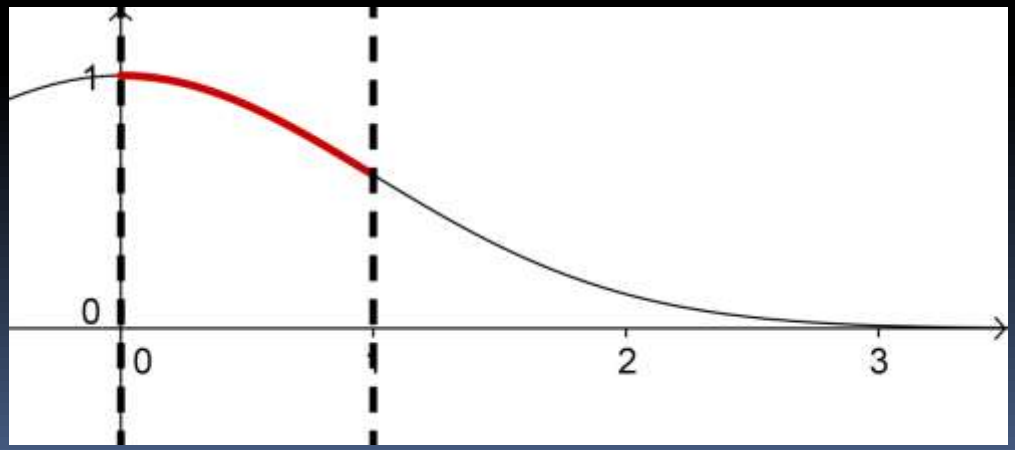
- Expliquer pourquoi la suite (u_n) est croissante et, en modifiant l'algorithme précédant, trouver la valeur de n à partir de laquelle

$$l - u_n \leq 10^{-8}.$$

Voir le fichier Scilab : Seuil

TP : Trapèzes contre Monte-Carlo

- On considère la fonction f définie sur
- l'intervalle [0,1] par $f(x) = e^{-\frac{x^2}{2}}$ et représentée sur la figure ci-dessous.



TP : Trapèzes contre Monte-Carlo

- Le logiciel Scilab donne ci-dessous une valeur approchée de l'intégrale (ce calcul est utile en probabilités).

```
-->integrate('exp(-x^2/2)','x',0,1)
 ans  =

    0.8556243918921
```

- On ne possède pas d'expression algébrique d'une primitive de f sur [0, 1]. Ce TP compare deux algorithmes de calcul approché de I.

TP : Trapèzes contre Monte-Carlo

Trapèzes

- on partage l'intervalle en n intervalles de même longueur ;
- sur chacun de ces n intervalles, on remplace la courbe représentative de f par un segment de droite coïncidant avec les valeurs de f aux extrémités de l'intervalle ;
- la somme des aires des n trapèzes obtenus est une valeur approchée de I.

TP : Trapèzes contre Monte-Carlo Trapèzes

- Montrer que pour n=4, la somme *S* des aires des quatre trapèzes vaut :

$$s = \frac{1}{4}\left(\frac{f(0)+f(1)}{2} + f\left(\frac{1}{4}\right) + f\left(\frac{2}{4}\right) + f\left(\frac{3}{4}\right)\right)$$

- On rappelle la formule de l'aire d'un trapèze et on donne une première étape avant simplification.

TP : Trapèzes contre Monte-Carlo Trapèzes

- On traduit l'algorithme en langage Scilab :

```
1 s=0
2 n=input("entrer le nombre de trapèzes n = ")
3 for i=1:n-1
4     s=s+exp(-(i/n)^2/2)
5 end
6 s=(1+exp(-0.5))/2+s
7 disp('Valeur approchée, pour n='+string(n)+', : '+string(s/n))
```

- Donner une expression de la valeur de la variable *s* en sortie de boucle en fonction de *n* et en utilisant le symbole Σ.
- En déduire une expression de à la fin de l'algorithme.

TP : Trapèzes contre Monte-Carlo Monte-Carlo

- Programmer cet algorithme sur un ordinateur (nommer le programme « trapezes » et l'enregistrer). Combien de décimales exactes de I obtient-on pour $n = 10$ et pour $n = 100$?
- Rappel : $I \approx 0{,}855624391821$

Voir le fichier Scilab : Trapezes

TP : Trapèzes contre Monte-Carlo Monte-Carlo

Cette méthode est la suivante :

- on prend au hasard *n* points de cordonnées (*x*, *y*) comprises entre 0 et 1 ;
- un compteur *s* détermine le nombre de points situés sous la courbe représentative de *f* ;
- la fréquence est une valeur approchée de *I* (car *I* est la probabilité de tirer un point sous la courbe de *f*).

TP : Trapèzes contre Monte-Carlo Monte-Carlo

- On a traduit ci-après l'algorithme de Monte-Carlo en langage Scilab.

```
1 s=0
2 n=input("Entrer le nombre de points n = ")
3 for i=1:n
4     x=rand()
5     y=rand()
6     if y<=exp(-x^2/2) then s=s+1
7     end
8 end
9 disp('Valeur approchée, pour '+string(n)+' points : '+string(s/n))
```

- À quoi correspond le test de la ligne 6 ?

TP : Trapèzes contre Monte-Carlo Monte-Carlo

- Programmer cet algorithme sur un ordinateur (nommer le programme « Monte-Carlo » et l'enregistrer). Combien de décimales exactes de I obtenez-vous pour $n = 100$ et pour $n = 1\,000$?
- Rappel : $I \approx 0{,}855624391821$

Voir le fichier Scilab : MonteCarlo

Comparaison des performances Trapèzes

- On montre que la précision obtenue avec n trapèzes dans le premier algorithme est de l'ordre de $\frac{1}{n^2}$. Quelle précision obtient-on pour $n = 32$?

Comparaison des performances Trapèzes

- Pour mesurer le temps de calcul du premier programme, insérer l'instruction tic() avant la boucle for et l'instruction temps=toc() avant l'affichage. Faire afficher la variable temps en fin de programme.
- Quel est le temps de calcul affiché pour $n = 32$? pour $n = 1\,000$?

Voir le fichier Scilab : Trapezes_tictoc

Comparaison des performances Monte-Carlo

- On montre que la précision obtenue avec n points pris au hasard dans le second algorithme est de l'ordre de $\frac{1}{\sqrt{n}}$ avec une confiance de 95% .
- Combien de points prendre au hasard pour obtenir une précision de 10^{-3} avec une confiance de 95% ?

Comparaison des performances Monte-Carlo

- Pour mesurer le temps de calcul du second programme, insérer l'instruction tic() avant la boucle for et l'instruction temps=toc() avant l'affichage. Faire afficher la variable temps en fin de programme.
- Quel est le temps de calcul affiché pour $n = 10^6$?

Voir le fichier Scilab : MonteCarlo_tictoc

Un peu d'histoire

- La méthode d'intégration approchée des trapèzes, a été introduite par Isaac Newton (1642-1727) et Roger Cotes (1682-1716).

Un peu d'histoire

- Les méthodes de Monte-Carlo consistent à calculer des quantités en utilisant des procédures aléatoires. Très utilisées lorsque des moyens de calcul plus directs ne sont pas envisageables, les méthodes de Monte-Carlo ont été notamment développées par John Von Neumann (1903-1957) durant la Seconde Guerre mondiale dans le cadre des recherches sur la bombe atomique.