

**Examen du 23 octobre 2019**  
**Durée : 1 heure 30**

- **Tous** les fichiers rendus devront contenir vos nom et prénoms.
- Il est très fortement suggéré d'envoyer des fichiers bien organisés, et qui ne créent pas d'erreurs à la compilation.
- Les deux parties sont indépendantes entre elles.

## 1 Probabilité de sortie d'un intervalle pour un mouvement Brownien

On cherche à calculer la probabilité  $p(x) = \mathbf{P}_x(W_t \in ]0, 1[, \forall 0 \leq t \leq 1)$  qu'un mouvement Brownien issu de  $x \in ]0, 1[$  ne quitte pas l'intervalle  $]0, 1[$  avant le temps  $t = 1$ .

Écrire dans un fichier `brownien.cpp` un programme qui calcule  $p(x)$  pour  $x \in \{0.1, 0.3, 0.5\}$ . Pour cela, on utilisera l'égalité en loi sous  $\mathbf{P}_x$

$$W_{k/N} \stackrel{\text{loi}}{=} x + \sum_{q=1}^k \frac{1}{\sqrt{N}} G_q,$$

où les  $G_k$  sont des Gaussiennes indépendantes, et on approchera l'évènement  $\{W_t \in ]0, 1[, \forall 0 \leq t \leq 1\}$  par l'évènement (plus grand)  $\{W_{k/N} \in ]0, 1[, \forall 0 \leq k \leq N\}$ .

On fera  $10^5$  simulations pour ce calcul, avec  $N = 100$  pas de discrétisation en temps.

## 2 Développement limités

On va programmer dans cette partie une méthode permettant de manipuler les développements limités d'une fonction en 0. Un développement limité à l'ordre  $n$  est une expression de la forme

$$f(x) = f_0 + f_1x + \dots + f_nx^n + o(x^n).$$

À partir de développements limités de  $f$  et de  $g$ , on peut obtenir un développement limité de  $f + g$ ,  $fg$ ,  $f/g$  (si  $g(0) \neq 0$ ), et de  $f \circ g$  (si  $g(0) = 0$ ).

Le but de cette partie est de représenter informatiquement de tels objets.

### 2.1 Ordre 1

On ne s'intéresse dans un premier temps qu'au cas de développements à l'ordre 1, de la forme  $f(x) = f_0 + f_1x + o(x)$ . Noter que l'on a  $f_0 = f(0)$  et  $f_1 = f'(0)$ .

1. Dans des fichiers `DL.cpp` et `DL.hpp`, écrire une classe `DL2`, contenant deux membres privés `f0` et `f1`, de type `double`, qui correspondront à la valeur de la fonction et celle de sa dérivée en 0.

- Écrire un constructeur pour cette classe, prenant en argument les deux valeurs à assigner à `f0` et `f1`. On fournira comme valeur par défaut 0 à la variable `f1`. Définir une variable globale constante notée `x` et correspondant au développement  $0 + x + o(x)$ .
- Surcharger l'opérateur `<<` pour permettre d'afficher un développement limité, sur le modèle de `1.5 + 2.1x`. On ne se souciera pas des affichages inhabituels en fonction des valeurs de `f0` et `f1`. Par exemple, on ne cherchera pas à éviter les affichages tels que `0 + -3.1x` ou `2.5 + 0x`.
- Écrire des surcharges des opérateurs `+`, `-`, `/` et `*`, correspondant aux opérations attendues sur les développements limités. Plus précisément, l'addition et la soustraction ont lieu terme à terme ; la multiplication et la division sont données par :

$$(a_0 + a_1x + o(x))(b_0 + b_1x + o(x)) = a_0b_0 + (a_0b_1 + a_1b_0)x + o(x)$$

et

$$\frac{a_0 + a_1x + o(x)}{b_0 + b_1x + o(x)} = \frac{a_0}{b_0} + \frac{a_1b_0 - a_0b_1}{b_0^2}x + o(x)$$

- Écrire un programme `fraction.cpp` qui donne la valeur numérique de la dérivée de  $\varphi : t \mapsto \frac{t+t^3}{1+t+t^5}$  au point  $t = 0.5$  (pour cela, faire un développement limité de  $\varphi(0.5 + x)$ ).

## 2.2 Ordre $n$

Dans cette partie, on s'intéresse maintenant à des développements d'ordre quelconque.

- Écrire une classe `DL`, contenant un membre privé `ordre` de type `int` et un autre nommé `coefs` de type `std::vector<double>` correspondant respectivement à l'ordre jusqu'auquel le développement limité est fait, et la liste des coefficients. Par exemple, le développement limité  $1 + x^2/2 - 3x^3 + o(x^4)$  serait stocké dans une variable pour laquelle le membre `ordre` vaudrait 4, et le membre `coefs` contiendrait les valeurs 1, 0, 0.5, -3 et 0.
- Écrire deux constructeurs pour cette classe : un premier prenant en argument un vecteur à copier dans `coef`, un deuxième prenant en argument un `double` et un `int`, qui correspond au développement limité  $\alpha + o(x^n)$  (on donnera par défaut la valeur 100 à  $n$ ).
- Surcharger l'opérateur `<<` pour permettre d'afficher un `DL`, sur le même modèle qu'en partie 2.1.
- Écrire des surcharges des opérateurs `+`, `-` et `*`, faisant les opérations usuelles sur les développements limités. La multiplication peut se faire en sommant tous les  $a_i b_j x^{i+j}$  jusqu'à l'ordre adéquat (ne pas oublier que  $x^n o(x^m) = o(x^{n+m})$  et que  $o(x^n) + o(x^m) = o(x^n)$  si  $n \leq m$ ).
- Surcharger l'opérateur `()` pour calculer la composée de deux développements limités. Pour le calcul du développement de  $f \circ g$ , on ajoutera successivement des puissances du développement de  $g$ .
- À l'aide de la multiplication et de la composition de développement limités, écrire le code d'une surcharge de `/` pour la division. On commencera par calculer l'inverse du dénominateur avec la formule  $(1 - x)^{-1} = 1 + x + x^2 + \dots + x^n + o(x^n)$ , puis on fera un produit.
- Écrire un programme `tan.cpp` qui calcul un développement limité à l'ordre 10 de  $\tan = \sin / \cos$  en 0.