Fonctions et floats

Florian Bourse

Ce TP est séparé en trois parties indépendantes : la première partie consiste à évaluer la fonction $x \mapsto \frac{1}{\sqrt{x}}$, la seconde à estimer la valeur de $\int_{-\infty}^{+\infty} e^{-t^2} \mathrm{d}t$ et la troisième traite l'affichage graphique d'une fonction.

1 Fast Inverse Square Root – (OCaml)

L'opération $x \mapsto \frac{1}{\sqrt{x}}$ sert en particulier à normaliser un vecteur, opération très utilisée par les algorithmes de rendu graphiques en 3D par exemple. L'opération $x \mapsto \sqrt{x}$ étant coûteuse, on peut chercher à optimiser son calcul.

1.1 Dichotomie

Nous allons ici nous baser sur le fait que $\frac{1}{\sqrt{x}}$ est la solution positive de l'équation suivante, d'inconnue z :

$$z^2 - \frac{1}{x} = 0$$

Pour un x donné, il s'agit donc de trouver une racine de la fonction continue strictement croissante $z \mapsto z^2 - \frac{1}{x}$.
L'algorithme de dichotomie permet de trouver une approximation arbitrairement précise d'une racine à partir d'un intervalle contenant la solution. L'idée est la suivante :

Notons z le nombre recherché. À partir de a et b tels que $a < z < b$, avec $f(a) \times f(b) < 0$ (leurs signes sont différents), on réduit la taille de l'intervalle de recherche en considérant le milieu $c = \frac{a+b}{2}$ de l'intervalle $]a, b[$, et on continue l'algorithme en utilisant l'intervalle $]a, c[$ ou $]c, b[$ suivant le signe de $f(c)$. À chaque itération, l'intervalle de recherche est divisé par 2, ce qui nous permet d'estimer la finesse de l'approximation.

0 Implémenter cet algorithme en OCaml, et le tester. On pourra vérifier son résultat à l'aide de la fonction `sqrt : float -> float`.

1.2 Méthode de Newton-Raphson

Pour cette partie, nous nous baserons sur le fait que $\frac{1}{\sqrt{x}}$ est solution de l'équation suivante, d'inconnue z :

$$\frac{1}{z^2} - x = 0$$

La méthode de Newton-Raphson utilise aussi plusieurs itérations pour trouver une approximation du résultat. Plutôt que de partir d'un intervalle contenant la solution, elle part d'un point quelconque, utilisé comme première approximation, et raffine cette approximation au fur et à mesure, en utilisant la tangente à la courbe.

Notons y l'approximation de départ. La fonction $f : z \mapsto \frac{1}{z^2} - x$ est dérivable sur $\mathbb{R}^+$.

1 Calculer la dérivée $f'(z)$ de $f(z)$.

2 En déduire l'équation de la tangente Δ à la courbe $\mathcal{C}_f$ au point d'abscisse y.
On rappelle que la pente de la tangente est donnée par la dérivée de la fonction.

3 Déterminer l'abscisse y' du point d'intersection de Δ et de l'axe des abscisses.
Nous allons utiliser y' comme approximation du résultat, et réitérer cette étape.

4 Implémenter cette méthode en OCaml et la tester. Comparer les précisions obtenues avec cette méthode et les précisions obtenues par dichotomie, en fonction du nombre d'itérations.

2 Estimer une intégrale – (C)

En C, nous allons utiliser le type `double`, ainsi que la fonction `exp`, qui nécessite d'inclure le header `<math.h>`, et d'ajouter `-lm` à la commande de compilation.

On rappelle que si f est une fonction croissante sur l'intervalle $[a, b]$, alors on a l'encadrement suivant :

$$f(a) \leq \frac{1}{b-a} \int_a^b f(t)\mathrm{d}t \leq f(b)$$

5 Estimer la valeur de $\int_{-\infty}^{+\infty} e^{-t^2}\mathrm{d}t$, ainsi que la distance entre votre approximation et la valeur recherchée.

3 Représentation graphie, Méthode d'Euler – (OCaml)

On va utiliser le module `Graphics`, dont les fonction sont utilisables après la commande `#require "graphics";;` dans l'interpréteur, ou en compilant à l'aide de l'utilitaire `ocamlfind`, par exemple :

```
ocamlfind ocamlopt -package graphics graphics.cma fun.ml -o fun
```

L'instruction `open_graph ""` ouvre une fenêtre graphique, dont chaque pixel est représenté par un couple d'entier. Les abscisses vont de `0` pour les pixels les plus à gauche à `size_x () - 1` pour les pixels les plus à droite. Les ordonnées vont de `0` pour les pixels les plus bas à `size_y () - 1` pour les pixels les plus hauts. On peut effacer complètement le contenu de la fenêtre avec l'instruction `clear_graph ()`.

Nous allons définir des couples `(xmin,xmax)` et `(ymin,ymax)` qui définissent la fenêtre dans laquelle nous allons représenter notre fonction.

6 Écrire des fonctions qui permettent de convertir les coordonnées d'un point d'un système de coordonnée à l'autre. On peut convertir les entiers en flottants et vice-versa avec les fonctions `float_of_int` et `int_of_float`.

7 Tracer le graphe de la fonction exponentielle, on pourra utiliser la fonction

```
exp : float -> float
```

. On peut tracer un point avec la fonction `plot : int -> int -> unit`, ou un tableau de points avec la fonction `plots : (int * int) array -> unit`.

Nous allons essayer d'approximer cette représentation sans utiliser la fonction `exp`, en utilisant la méthode d'Euler : pour chaque abscisse de pixel, on va calculer la nouvelle position en suivant la tangente à la courbe.

8 Implémenter cette méthode et comparer la courbe obtenue avec celle tracée précédemment.