1 Tp2_Calcul_de_pi

1.1 Buts:

- Utiliser les structures de contrôle (`loop`, `while`, `if`).
- Se familiariser avec les problèmes de précision numérique.
- Utilisation de `CONST`.
- Utilisation de conversions explicites.
- Utilisation des tableaux statiques.

1.2 Énoncé:

Écrire un programme qui calcule différentes approximations de la valeur de π et comparer leur précision.

1.3 Séries numériques

Dans un premier temps il faut calculer π avec de en vous servant des trois formules ci-dessous.

$$\sum_{n=1}^{\infty} \frac{1}{n^4} = \frac{\pi^4}{90}, \tag{1}$$

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n^2} = \frac{\pi^2}{12}, \tag{2}$$

$$\prod_{n=1}^{\infty} \left(\frac{2n}{2n-1}\right)\left(\frac{2n}{2n+1}\right) = \frac{\pi}{2}. \tag{3}$$

1.3.1 Travail à effectuer

Implémenter les formules et comparer les résultats obtenus avec une constante `PI` que vous aurez défini. Quelle méthode a besoin du moins d'itérations pour atteindre une précision donnée?

1.4 Méthode Monte-Carlo

Soit le carré $\Omega = [-1, 1] \times [-1, 1]$, dans le plan $\mathbb{R}^2$. Si on tire avec une probabilité uniforme un point $(x, y) \in \Omega$, quelle est la probabilité que ce point tombe dans le disque, D, de rayon 1, centré en $(0, 0)$? Celle-ci est en fait égale au rapport de l'aire du disque sur l'aire du carré (voir la fig. ci-dessous).

En effectuant N tirages et en comptabilisant le nombre, k, de fois où on tombe dans le disque par rapport au nombre total de tirages, on obtient une estimation de la probabilité de tomber dans le disque. Celle-ci est donnée par

$$p((x, y) \in D) = \frac{\pi r^2}{4} = \frac{\pi}{4}.$$

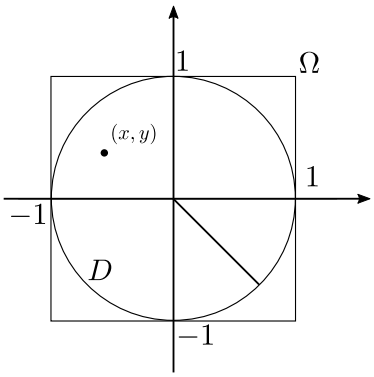

Figure 1: Le carré Ω de côté 2 et le disque inscrit, D, de rayon 1. Un point (x, y) tiré au hasard tombant dans le cercle.

On a donc que

$$\frac{k}{N} \cong \frac{\pi}{4}.$$

Plus N est grand et plus l'estimation sera bonne.

1.4.1 Travail à effectuer

Écrire un algorithme utilisant la méthode ci-dessus pour calculer π. Comparer les résultats obtenus avec une constante `PI` que vous aurez défini. Estimer l'erreur de la méthode en fonction de N.

1.5 L'aiguille de Buffon

Sur un parquet formé de planches de largeur $2a$, séparées par des rainures droites, parallèles et équidistantes, on jette une aiguille de longueur 2λ, avec $\lambda < a$. Soit l'événement

$$A = \{\text{l'aiguille coupe une des rainures}\},$$

que vaut la probabilité de réaliser A, notée $p(A)$?

Soit x la distance du milieu de l'aiguille à la rainure la plus proche: x prend une valeur aléatoire quelconque dans $[0, a]$. Soit θ, l'angle entre la rainure et l'aiguille: θ prend une valeur aléatoire quelconque dans $[0, \pi]$.

Dans une représentation cartésienne du rectangle $\Omega = [0, \pi] \times [0, a]$, l'événement A est représenté par la partie hachurée délimitée par la courbe d'équation

$$x = \lambda \cdot \sin(\theta).$$

Pour obtenir $p(A)$ il suffit donc de faire le rapport entre la surface de Ω (en gris) et cette sous la courbe (hachurée) (voir la figure ce-dessous)

$$p(A) = \frac{\int_0^\pi \lambda \sin(\theta) \mathrm{d}\theta}{\pi a} = \frac{2\lambda}{\pi a}.$$

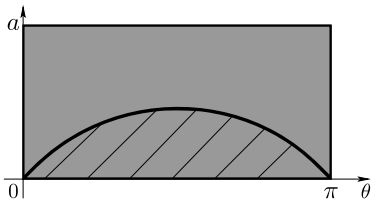

Figure 2: La surface du rectangle $[0, \pi] \times [0, a]$ (en gris) et la surface sous la courbe $x = \lambda \sin(\theta)$ (hachurée).

1.5.1 Travail à effectuer

Il s'agit d'écrire un programme qui utilise la méthode ci-dessus pour calculer la valeur de π. Comparer les résultats obtenus avec une constante `PI` que vous aurez défini. Estimer l'erreur de la méthode en fonction de N.

Afin de toucher la rainure il faut que

$$\sin(\theta) > \frac{2x}{\lambda}.$$

On peut donc en déduire la valeur de π en calculant

$$\pi = \frac{2\lambda}{ap(A)},$$

où $p(A)$ est la proportion de réussite mesurée.

Pour calculer π de cette manière on a besoin d'une table avec les valeur de $\sin(\theta)$ que vous pouvez trouver ci-dessous ($\theta \in [0, 90]$, avec $\delta\theta = 1°$)

```
let sin: [f32;91] =
  [0.0, 0.017452406, 0.034899496, 0.05233596, 0.06975647,
   0.087155744,0.104528464, 0.12186935, 0.1391731, 0.15643448,
   0.1736482, 0.19080901, 0.2079117, 0.22495106, 0.24192192,
   0.25881904, 0.27563736, 0.29237172, 0.309017, 0.32556814,
   0.34202015, 0.35836795, 0.3746066, 0.39073113, 0.40673664,
   0.42261827, 0.43837118,0.45399052, 0.4694716, 0.4848096,
   0.5, 0.51503813, 0.52991927, 0.54463905, 0.55919296,
   0.5735765, 0.58778524, 0.60181504, 0.6156615, 0.6293204,
   0.64278764, 0.656059, 0.6691306, 0.6819984, 0.6946584,
   0.70710677, 0.7193398, 0.73135376, 0.74314487,
   0.7547096, 0.76604444, 0.77714604, 0.7880108, 0.79863554,
   0.809017, 0.81915206, 0.8290376, 0.8386706, 0.8480481,
   0.8571673, 0.86602545, 0.8746197, 0.8829476, 0.8910066,
   0.89879405, 0.9063078, 0.9135455, 0.92050487, 0.92718387,
   0.9335804, 0.9396927, 0.94551855, 0.95105654, 0.9563048,
   0.9612617, 0.9659259, 0.9702957, 0.97437006, 0.9781476,
   0.98162717, 0.9848078, 0.98768836, 0.99026805, 0.99254614,
```

```
    0.9945219, 0.9961947, 0.9975641, 0.9986295, 0.99939084,
    0.9998477, 1.0];
```

1.6 Fonctions mathématiques à utiliser

Afin d'utiliser les fonctions mathématiques de Rust, il faut importer la librairie `rust_hepia_lib`. Pour ce faire il faut ajouter une dépendance dans le fichier `Cargo.toml` (on vous simplifie un peu la vie)

```
rust_hepia_lib = { path = "$CHEMIN/rust_hepia_lib" }
```

où `$CHEMIN` est l'endroit où vous avez cloné la libraire.

Puis, il faut importer la librairie externe dans le `main.rs` avec les lignes

```
extern crate rust_hepia_lib;
use rust_hepia_lib::*;
```

Les fonctions disponibles sont :

```
pub fn pow(x: i32, y: i32);
pub fn powf(x: f32, y: f32);
pub fn abs(num: i32);
pub fn absf(num: f32);
```

Ces fonctions permettent de faire des calculs de puissances et de valeurs absolues. Les fonctions `pow`, `abs`, respectivement `powf`, `absf` sont pour les entiers 32 bits, respectivement nombres à virgule flottante 32 bits.

N'oubliez pas d'utiliser également au moins une fonction vue la semaine passée...