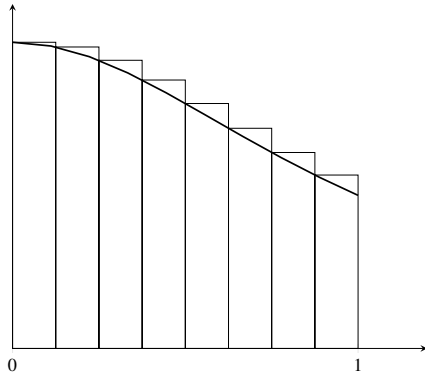


OpenMP et calcul de π

❶ Par calcul d'intégrale :



$$\frac{\pi}{4} = \arctan(1) = \int_0^1 \frac{4}{1+x^2} dx$$

Ce qui peut être approché par :

$$\pi \approx \sum_{i=0}^n \frac{4}{1+x_i^2} \Delta x \quad \text{Où :} \quad \begin{aligned} \triangleright \Delta x &= \frac{1}{n+1} \\ \triangleright x_i &= (i + \frac{1}{2}) \Delta x \end{aligned}$$

Cette formule calcule la somme des aires des différents rectangles compris sous $f(x) = \frac{4}{1+x^2}$, c-à-d une approximation de l'intégrale de $f(x)$.

Ce qui donne le programme C :

```
#include <stdio.h>

int main() {
    const int numPoints = 10000000;
    const double deltaX = 1.0/(double)numPoints;
    double pi = 0.0;
    for(int i = 0; i < numPoints; ++i)
    {
        double xi = (i + 0.5) * deltaX;
        pi += 4.0/(1 + xi * xi);
    }
    pi *= deltaX;
    printf("Value of pi: %.10g\n", pi);
}
```

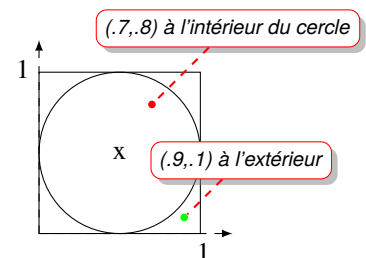
❷ La formule de Leibniz-Grégory : $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots \frac{1}{n}$ où il y a une alternance de somme et de différence des inverses des entiers impairs.

❸ La méthode de Monte-Carlo, par tirage aléatoire de points.

On choisit au **hasard** des points (x, y) à l'intérieur du carré, c-à-d compris entre $(0, 0)$ et $(1, 1)$:

Pour un point $p = (x, y)$:

- ▷ p est dans le cercle si la distance entre le centre $(0.5, 0.5)$ du cercle et p est inférieure ou égale à 0.5 , c-à-d $\sqrt{(x - .5)^2 + (y - .5)^2} \leq 0.5$



Pour chaque **tirage aléatoire** d'un point p :

- ▷ on ajoute 1 au nombre total T de points tirés au hasard ;
- ▷ si le point est à l'intérieur du cercle, alors on incrémente le nombre C de points à l'intérieur du cercle ;

La **probabilité** pour qu'un point choisi **au hasard** appartienne au cercle est $\frac{\text{aire cercle}}{\text{aire carré}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$

Si les points p sont choisis de **manière aléatoire** (distribution uniforme) alors le rapport entre C est T est une **approximation** de notre probabilité, qui tend vers $\frac{\pi}{4}$. Ce qui donne : $\pi = 4 \frac{C}{T}$

1 – Le programme calculant la méthode de Monte Carlo :

```
int main() {
    int i, count, n = NOMBREMAX;
    double x, y, z, pi;
    count = 0;
    for(i = 0; i < n; ++i) {
        x = (double)rand() / RAND_MAX;
        y = (double)rand() / RAND_MAX;
        z = x * x + y * y;
        if( z <= 1 ) count++;
    }
    pi = (double) count / n * 4;
    printf("Approximate PI = %g", pi);
}
```

- Donnez une version parallèle la plus efficace possible de la méthode ❶ par calcul d'intégral.
Vous expliquerez et justifierez vos choix.
- Pour la méthode ❷ par formule de Leibniz, pour des grandes valeurs de n est-ce qu'il y a un intérêt à précalculer les différentes valeurs impaires ?
- Donnez une version parallèle la plus efficace possible de la méthode ❷ par formule de Leibniz.
- Pour la méthode ❸ par Monte Carlo, est-ce que les « tasks » peuvent être intéressantes ?
- Donnez une version parallèle la plus efficace possible de la méthode ❸ par Monte Carlo.
- Quelle méthode vous paraît être la meilleure compte tenu des approximations dues au limitation de calcul d'un ordinateur ?
Vous expliquerez votre choix.

2 – On voudrait pouvoir vérifier un sudoku à l'aide d'un programme parallèle.

4	2	6	5	7	1	3	9	8
8	5	7	2	9	3	1	4	6
1	3	9	4	6	8	2	7	5
9	7	1	3	8	5	6	2	4
5	4	3	7	2	6	8	1	9
6	8	2	1	4	9	7	5	3
7	9	4	6	3	2	5	8	1
2	6	5	8	1	4	9	3	7
3	1	8	9	5	7	4	6	2

Proposez une **méthode de parallélisation** pour réaliser cette vérification :

- vous proposerez une méthode de parallélisation :
 - Vous expliquerez quel type de parallélisme peut être exploiter ;
 - vous choisirez la cible matériel de votre choix entre OpenMP et CUDA en expliquant votre choix ;
 - vous décrierez, expliquerez et justifierez si votre solution entraîne des problèmes d'accès concurrents en écriture et si vous pensez être arrivé à une solution optimale.
- Est-ce que votre solution est « scalable » ? Pourquoi ?
- Est-ce que le traitement pourrait être accéléré dans le cas de la vérification de très nombreuses grilles de sudoku ? Comment ?