Seminario 1: ejemplo9. Cálculo concurrente del número π

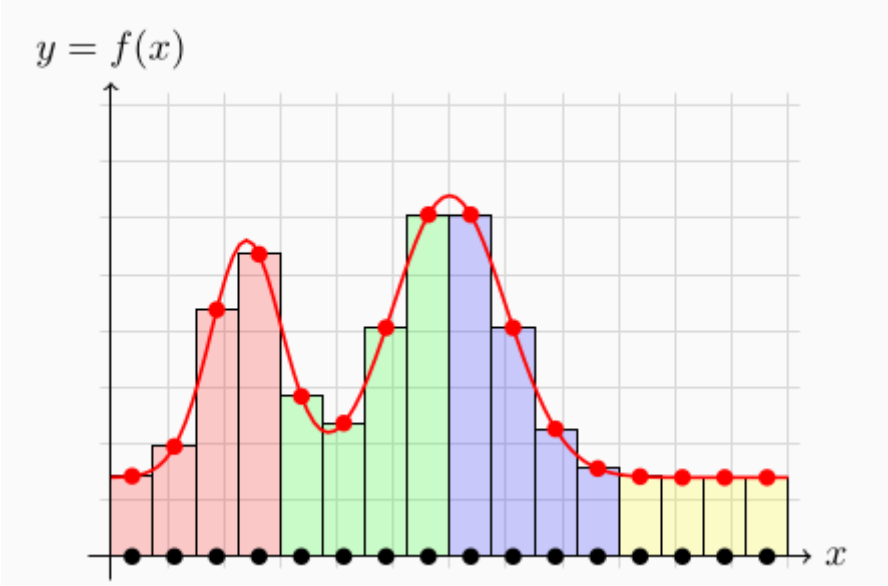
Dada la función secuencial para el cálculo del número π, una forma de implementar la versión concurrente es la siguiente:

```
double funcion_hebra(long i) {
    double suma = 0.0 ;

    for(long j = i * (m/n); j < (i+1) * (m/n); j++ ) {
        const double xj = double(j+0.5)/m ;
        suma += f(xj);
    }

    return suma/m;
}
```

El código para el cálculo es el mismo que en la versión secuencial pero en la función que realiza la hebra los índices están ajustados para que cada hebra calcule la función en una parte de las muestras que tenemos como en la siguiente gráfica:




```
double calcular_integral_concurrente( ) {
    double total = 0.0;

    future<double> hebras[n];
    for(int i = 0; i < n; i++) {
        hebras[i] = async(launch::async, funcion_hebra, i); //Crear y lanzar las
hebras
    }

    for(int i = 0; i < n; i++){
        total += hebras[i].get();
    }

    return total;
```

```
}
```

En la función que calcula la integral lo que hacemos es crear un vector de `futures` de tamaño n, que son las hebras que tendremos en el programa. Cuando cada hebra ha ejecutado su parte se suma el resultado al total y se devuelve el valor.

La salida del programa es la siguiente:

```
Número de muestras (m)   : 1073741824
Número de hebras (n)     : 4
Valor de PI              : 3.14159265358979312
Resultado secuencial     : 3.14159265358998185
Resultado concurrente    : 3.14159265358982731
Tiempo secuencial        : 3951.6 milisegundos.
Tiempo concurrente       : 1022.9 milisegundos.
Porcentaje t.conc/t.sec. : 25.89%
```