I. Unidad 5 - Sesión 14 - Métodos para aumentar la eficacia - Ejercicio 14.1

II. Trabajo Grupal.

- Santiago Alaniz, CI 50826476, santiago.alaniz@fing.edu.uy
- Bruno De Simone, CI 49145550, bruno.de.simone@fing.edu.uy

III. Descripción del Problema

Partiendo de uno de los códigos elaborados para resolver el **Ejercicio 6.2**, utilizar el método de muestreo estratificado para calcular la integral de la función `x1*x2^2*x3^3*x4^4*x5^5` sobre el hipercubo Jm de dimensión m = 5 en base a 10^6 iteraciones. Calcular media, desviación estándar y un intervalo de confianza de nivel 95%.
Comparar con los resultados obtenidos con el código del **Ejercicio 6.2**.

Sugerencia:
Definir 5 estratos, en función del valor de `x5`, tomando los siguientes intervalos:

- [0, 0.72)
- [0.72, 0.83)
- [0.83, 0.90)
- [0.90, 0.95)
- [0.95, 1]

Hacer dos experimentos, uno tomando 10^6/5 iteraciones en cada estrato y otro tomando una cantidad de iteraciones proporcional a la probabilidad de cada estrato.

IV. Descripción de la Solución

Se tomó la solución en python debido a que ambos miembros tenían experiencia con el lenguaje. Los únicos cambios que se tuvieron que realizar para la adaptacion del codigo fueron:

- Primero se generaron los puntos para los distintos estratos, dependiendo de como se hacía la distribución de muestras por estratos es que se definen cuantas muestras para cada estrato.
  - En caso de distribución equitativa para cada estrato (5 estratos) se genera n/5 muestras.
  - En caso de distribucion proporcional se genera según la probabilidad de cada estrato la cantidad de muestras

```
- [0, 0.72)        ---> 72%
- [0.72, 0.83)     ---> 11%
- [0.83, 0.90)     --->  7%
- [0.90, 0.95)     --->  5%
- [0.95, 1]        --->  5%
```

```
def proportionaly_strat_sampling() -> list:
    aux = []
    prob_strats = [round((strat[1] - strat[0]), 6) for strat in x5_strats]

    for i, strat in enumerate(x5_strats):
        n_ = int(round(n * prob_strats[i],2))
        aux += strat_uniform(n=n_, a=strat[0], b=strat[1]).tolist()

    return aux
```

- En cada iteración generamos un vector de 4 dimensiones (x1, x2, x3, x4) y le agregamos el x5 generado previamente.
- Cambiar la función que en el 6.2 obtiene la altura del punto por la función de la que se quiere calcular la integral.

```
# Definición de constantes del ejercicio
def k(x) -> float:
    return x[0] * x[1]**2 * x[2]**3 * x[3]**4 * x[4]**5

def stratified_montecarlo_simulation(proportional=bool):
    # Init
    S = 0
    T = 0

    if proportional:
        x5_samples = proportionaly_strat_sampling()
    else:
        x5_samples = evenly_strat_sampling()

    for i in range(n):
        X_i = [uniform_samples.pop() for _ in range(4)]

        k_x_i = k(X_i + x5_samples.pop()) # k(X_i)

        T += (1 - (1/i)) * ((k_x_i - (S/(i-1))) ** 2) if i > 1 else 0
        S += k_x_i

    Int_ = (S/n)
    Var_F = T / (n - 1)
    Var_Int_ = Var_F / n
    ic_delta = norm.ppf(1 - delta/2) * math.sqrt(Var_Int_)
    IC = (Int_ - ic_delta, Int_ + ic_delta)

    return (Int_, Var_Int_, IC)
```

V. Resultados Computacionales

1. AMD Ryzen 7 1700x a 3.4 GHz, 16 GB DDR4 3200mhz, python 3.10
2. Semilla Inicial: 50826476;
3. $(anexo) python3 exercise_14_1.py

Ej	Int_	Var	IC (inferior)	IC (superior)	time
No Estratificado	0.001395	9.488065e-11	0.001376	0.001414	0:00:01.162361
Estratificado Evenly	0.003921	3.156507e-10	0.003886	0.003956	0:00:01.380145
Estratificado Proporcional	0.001386	9.379704e-11	0.001367	0.001405	0:00:01.358470

La solución analítica para la integral de la función es: 0.001389...

Como vemos la versión estratificada evenly generó una pésima estimación, esto demuestra la importancia de pensar cómo se distribuye la cantidad de repeticiones por estratos no solo en los intervalos de estratificación.

Por otro lado, a pesar de que no es el objetivo principal la versión estratificada proporcionalmente produce una mejor estimación que la versión no estratificada. Vemos que también mejora la varianza al reducirla (objetivo de estratificar el algoritmo).

Consideramos que fue inteligente estratificar x5 ya que es la V.A. que más influye en el valor de la integral de la función al ser elevada a la 5, por eso no interesa que esté bien distribuida y tenga buena representación.

VI. Anexo

Consultar la carpeta ~/anexo por los logs, capturas, código de la solución y demás archivos de interés.