(Simulating a random vector)

Explain how to simulate a random two-dimensional vector ˜ x with a joint pdf f˜x(x) that is uniformly distributed in the shaded region. Assume that you have access to independent samples from a uniform distribution in [0, 1]. Explain your method, justifying why it works. Then implement it and submit the code along with a scatterplot of 1,000 samples.

Explanation of the Sampling Method

We wish to simulate a random vector $\tilde{x} = (X_1, X_2)$ that is uniformly distributed inside the triangle with vertices $A = (0, 0)$, $B = (1, 0)$, and $C = (0.5, 1)$. The idea is to transform two independent $\text{Uniform}(0, 1)$ random variables into a point that is uniform on this triangle.

Step 1: Generate (u, v) uniformly over the unit square.

Draw $u, v \sim \text{Uniform}(0, 1)$ independently. This gives a point that is uniformly distributed in $[0, 1]^2$.

Step 2: Use the reflection trick to obtain a point in the standard triangle.

The triangle $\{(u, v) : u \geq 0, v \geq 0, u + v \leq 1\}$ has area $1/2$. If $u + v \leq 1$ we keep (u, v). If $u + v > 1$, we reflect across the diagonal $u + v = 1$ and replace [(u,v) \mapsto (1-u,,1-v).] This reflection maps the region $\{u + v > 1\}$ bijectively and area-preservingly onto the triangle $\{u + v \leq 1\}$, so the resulting (u, v) is uniform on the unit triangle.

Step 3: Map to the target triangle using barycentric coordinates.

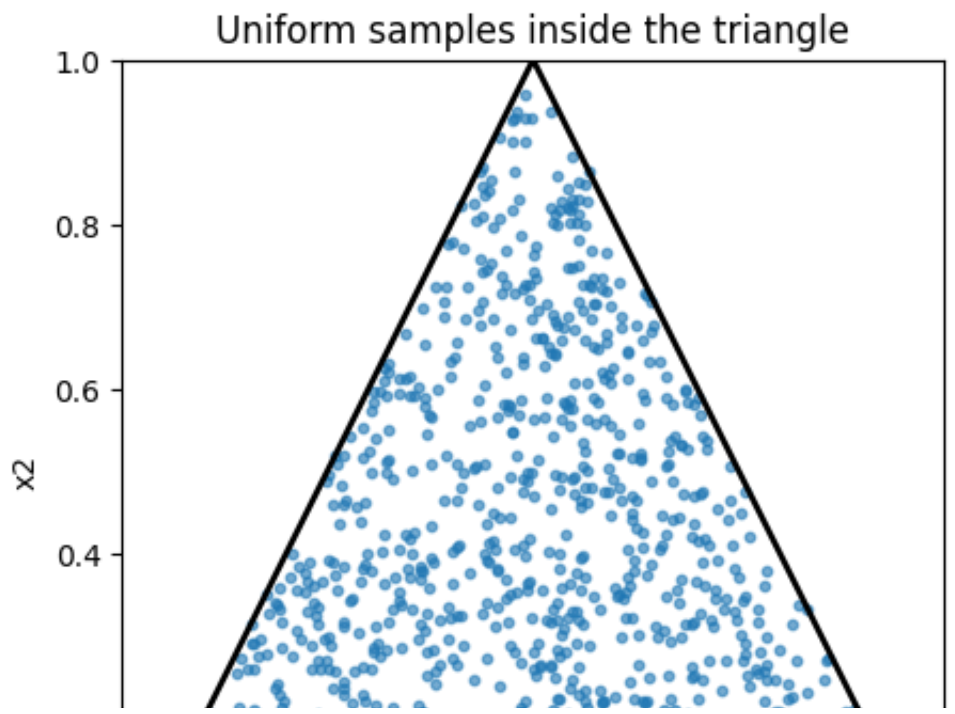
Once (u, v) lies in the unit triangle, we transform it to the triangle with vertices A, B, C using the affine map [X = A + u(B-A) + v(C-A).] Affine maps preserve uniformity (up to a constant Jacobian factor), so X is uniformly distributed over the desired triangular region.

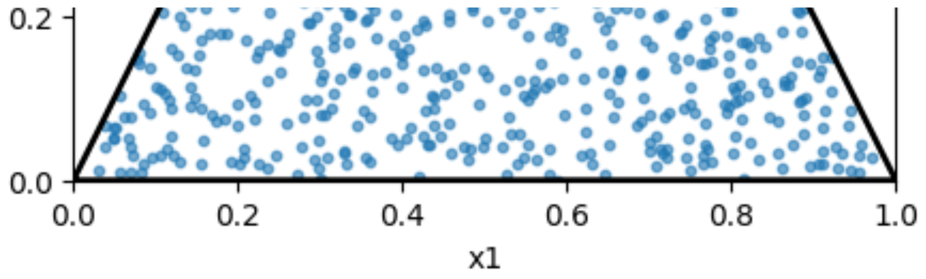
Why it works:

(1) The reflection step is area-preserving and converts uniform sampling on the square into uniform sampling on the standard triangle.
(2) An affine transformation of a uniformly distributed point remains uniform on the transformed region.
Therefore, the output point is uniform on the triangle with vertices A, B, C.

```
import numpy as np
import matplotlib.pyplot as plt

# Vertices of the triangle (from the figure)
A = np.array([0.0, 0.0])   # left base
B = np.array([1.0, 0.0])   # right base
C = np.array([0.5, 1.0])   # top

def sample_uniform_triangle(n_samples, A, B, C):
    """
    Sample n_samples points uniformly from the triangle with vertices A

    Method:
      1. Draw u, v ~ Uniform(0,1) independently.
      2. Reflect points with u + v > 1 via (u, v) -> (1 - u, 1 - v).
         This maps the unit square onto the triangle in an area-preserv
      3. Use barycentric coordinates: X = A + u*(B - A) + v*(C - A).
    """
    u = np.random.rand(n_samples)
    v = np.random.rand(n_samples)

    # Reflect points that fall in the upper-right half of the unit squa
    mask = (u + v > 1)
    u[mask] = 1 - u[mask]
    v[mask] = 1 - v[mask]
```

```
        v[mask] = 1 - v[mask]

        # Convert (u, v) into points inside the triangle via barycentric cc
        points = A[None, :] + u[:, None]*(B - A)[None, :] + v[:, None]*(C -
        return points

    # Generate 1000 samples
    n = 1000
    samples = sample_uniform_triangle(n, A, B, C)

    # Plot the samples
    plt.figure(figsize=(5,5))
    plt.scatter(samples[:, 0], samples[:, 1], s=10, alpha=0.6)

    # Draw the triangle outline for reference
    triangle_x = [A[0], B[0], C[0], A[0]]
    triangle_y = [A[1], B[1], C[1], A[1]]
    plt.plot(triangle_x, triangle_y, 'k-', linewidth=2)

    plt.xlim(0, 1)
    plt.ylim(0, 1)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.title('Uniform samples inside the triangle')
    plt.show()
```




0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
x1