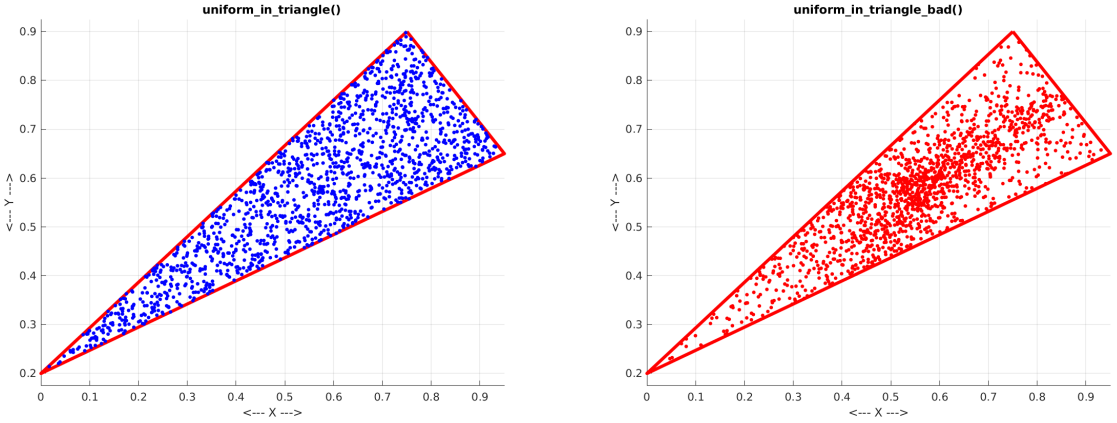
Project: Sampling a Triangle
Mathematical Programming with Python

MATH 2604: Advanced Scientific Computing 4
Spring 2025
Monday/Wednesday/Friday, 1:00-1:50pm
Room A202 Langley Hall

https://people.sc.fsu.edu/~jburkardt/classes/python_2025/triangle/triangle.pdf

---



1500 "random" points in a triangle, good and bad algorithms

Some algorithms require us to sample points from a geometric region, and to do so in a uniform way. We have already discussed in class several methods for trying to sample a circle. In this project, you will evaluate several proposed methods for uniformly sampling an abitrary triangle. For each method, you will generate 1000 sample points and plot them. From the picture, you should be able to judge which methods seem to be doing the required uniform sampling.

You might write a typical program as a function, perhaps of the form

```
method1 ( a, b, c, n )
```

which implements method #1 on a triangle whose (x,y) vertices are stored in a, b and c, and for which you are going to compute n random samples and then plot them.

The methods to consider are:

1. acceptance/rejection: draw a rectangle around the triangle. It is easy to compute random points in the rectangle. If a point is also in the triangle, add it to your list, and stop when you have accepted n points.
2. vectors: To generate a random point p in the triangle, compute two random values r_1 and r_2, and set $p = a + r_1 * (b - a) + r_2 * (c - a)$
3. trilinear: pick three random values r_1, r_2 and r_3. Set $p = (r_1 * a + r_2 * b + r_3 * c)/(r_1 + r_2 + r_3)$
4. reflect: pick random values r_1 and r_2, but if $1 < r_1 + r_2$, then replace r_1 and r_2 by $1 - r_1$ and $1 - r_2$. Then set $p = (1 - r_1 - r_2) * a + r_1 * b + r_2 * c$
5. weights: pick random values r_1 and r_2. Set $\alpha = \sqrt{r_1}$. Then $p = (1 - \alpha) * a + \alpha * (1 - r_2) * b + \alpha * r_2 * c$

I expect that you will find several (but not all!) methods produce good sampling of the triangle.

For the acceptance/rejection method, you will need to be able to determine whether each point p in your rectangle is also in your triangle. This is not hard to do, but you should be careful that you write and test your code for this task.