Sampling 1D Distributions

Nonstandard $p(x)$

Sachin Shanbhag

Department of Scientific Computing
Florida State University,
Tallahassee, FL 32306.

Overall

We will learn two techniques to use the continuous uniform distribution $\mathbf{U}[0, 1]$, to sample from other distributions.

- Transformation Method
- Accept-Reject Method

The accept-reject method has somewhat wider applicability than the transformation method.

We saw a particular version of the accept-reject method, when we tried to find the area of a circle by throwing darts.

Transformation Method

The idea in this method is to use a sequence of random numbers from $U[0,1]$

$$U_1, U_2, ..., U_n \sim g(u) = U[0,1],$$

to produce a sequence of random numbers from the target PDF,

$$X_1, X_2, ..., X_n \sim f(x),$$

by seeking a mathematical mapping or transformation $x(u)$.

One idea to get a suitable mapping is to match the CDFs.

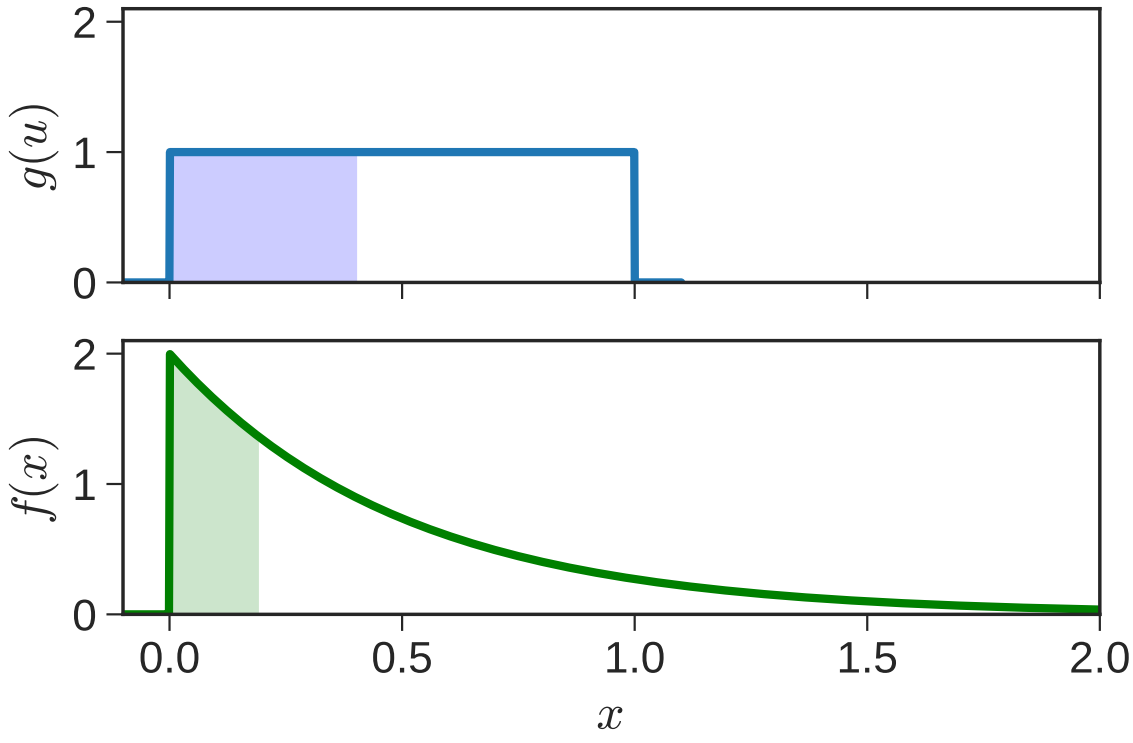
Note:

- The relationship between CDF and PDF is unique
- The mapping will depend on $f(x)$

Transformation Method: Sketch

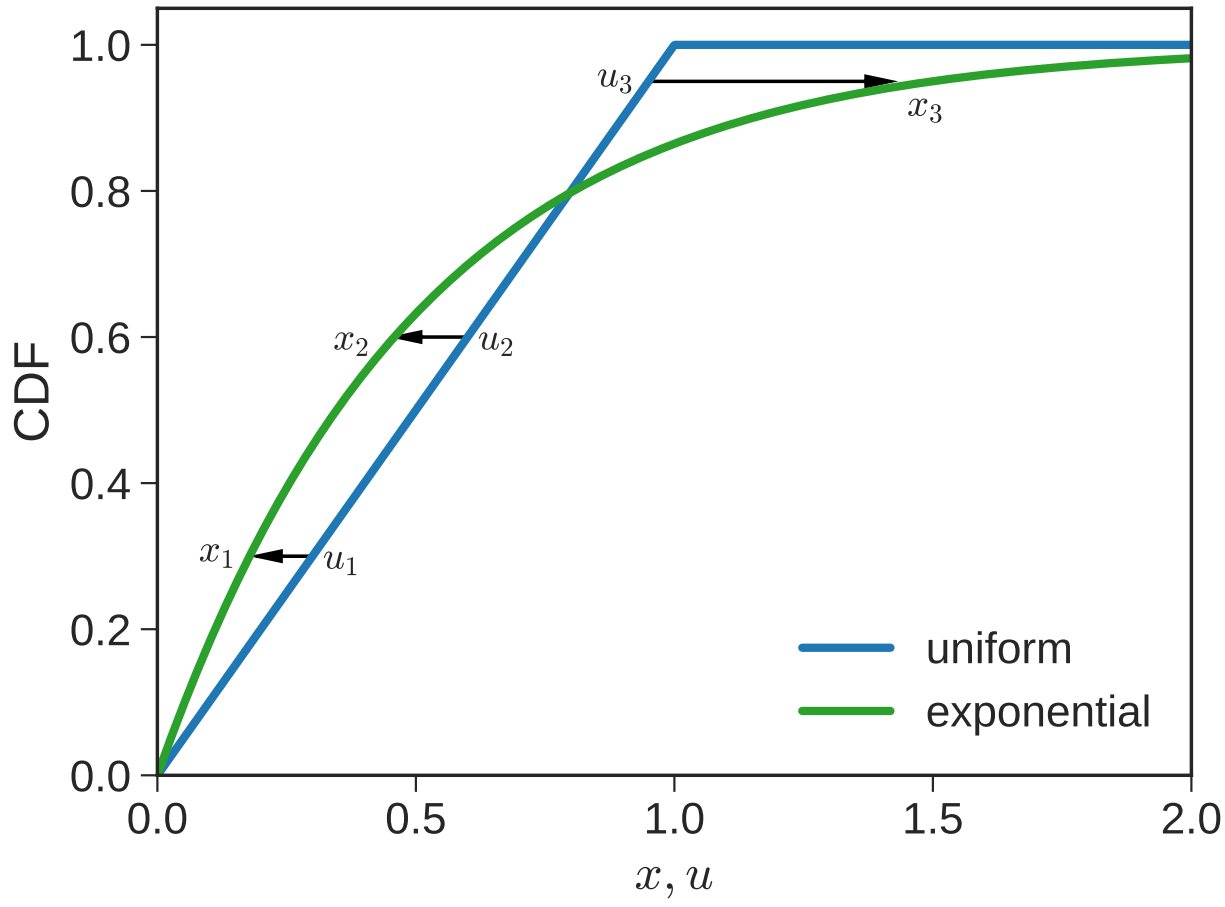
Let me try to illustrate the idea graphically.

Let us think in terms of PDFs first.



Given a u, we pick an appropriate x so that the blue and green areas in the plot above are the same.

Transformation Method: CDF sketch



We seek the mapping $u_i \to x_i$ by asserting $F(x) = u$.

Transformation Method

Let us now express this notion mathematically,

$$\begin{aligned} \mathsf{Pr}(u' \leq u) &= \mathsf{Pr}\left(x' \leq x(u)\right). \\ \int_{-\infty}^{u} g(u')du' &= \int_{-\infty}^{x(u)} f(x')dx' \\ u &= F(x(u)), \end{aligned}$$

where $F(x)$ is the CDF corresponding to $f(x)$.

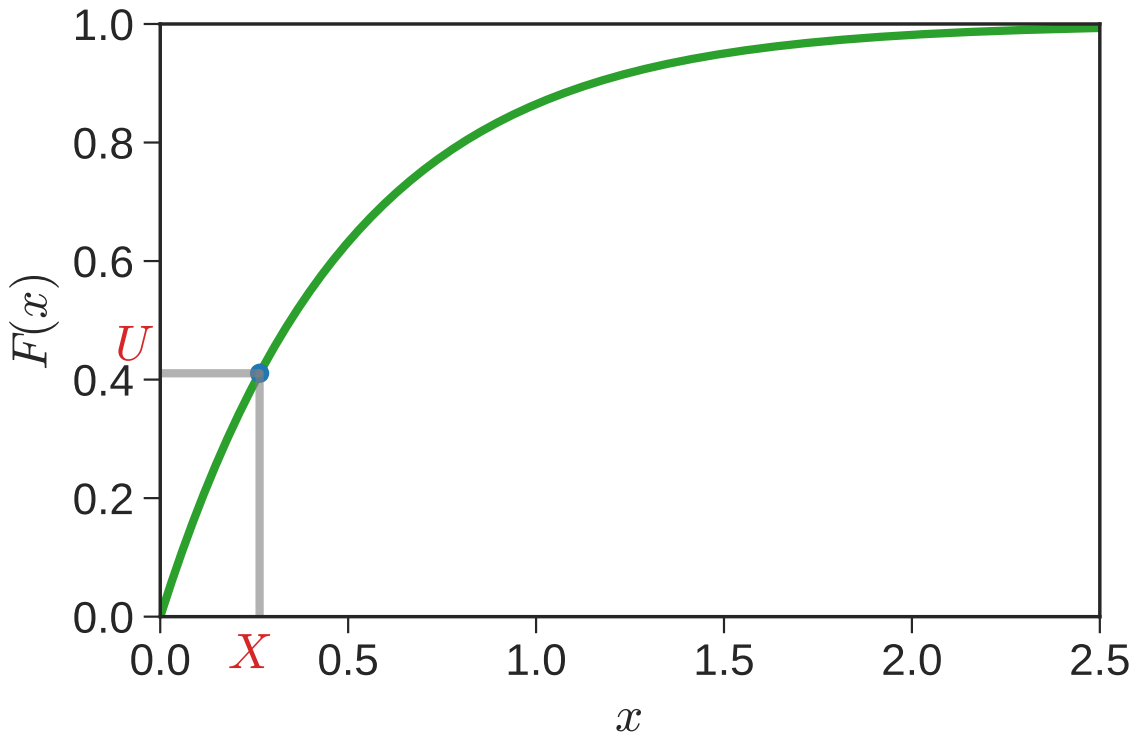
Given the target PDF $f(x)$, we find the corresponding CDF $F(x)$, and try to solve

$$u = F(x)$$

If it is possible to solve for $x = F^{-1}(u)$ in terms of u, then we are in business!

Intuition

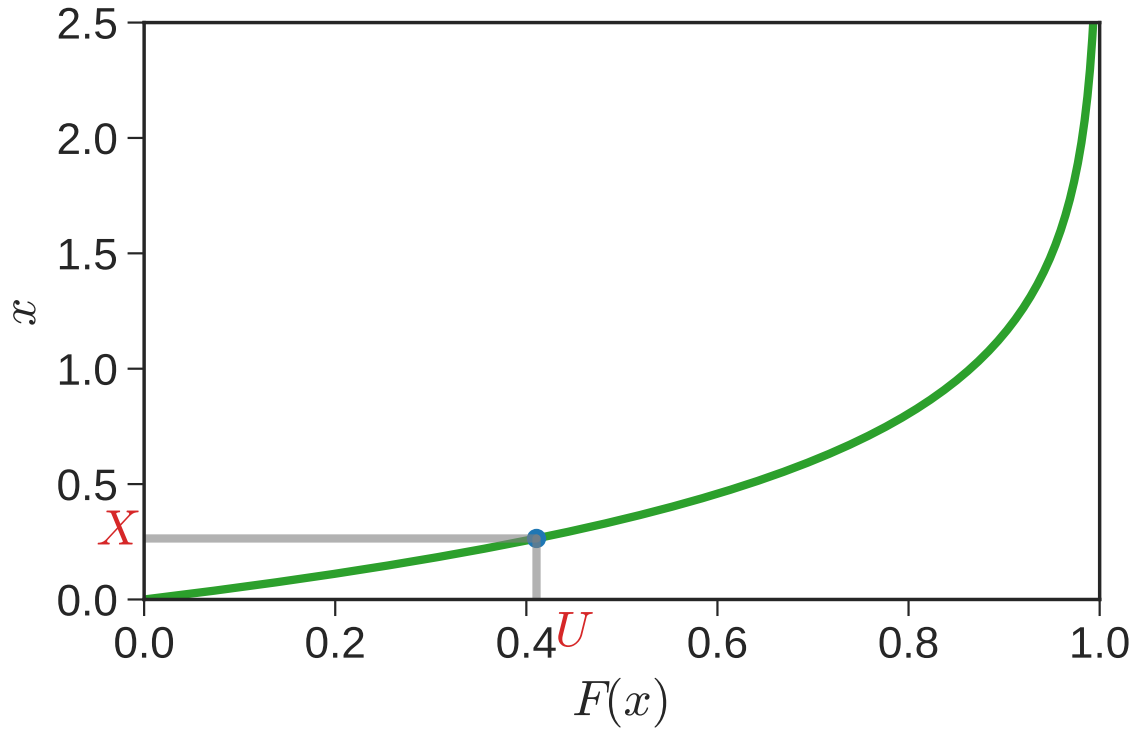
The domain of $U[0, 1]$ matches the range of the CDF, $[0 - 1]$.



- draw $U \sim U[0, 1]$,
- use the CDF (solve $F(X) = U$) to get X

Intuition

This is equivalent to flipping the graph above on its side.



The success of this method hinges on our ability to efficiently solve for $x = F^{-1}(u)$.

Transformation Method

If we can analytically solve $x = F^{-1}(u)$, this is likely our best option

Alternatively, we may be able to use some sort of piecewise curve (linear, cubic, spline) to fit $F(x)$, and then use inverse interpolation to generate the random deviates.

In general, we could use a nonlinear solver, but that may involve too much overhead.

Let us consider an example.

Example

Generate random numbers, x_i, from the exponential distribution:

$$f(x;\lambda) = \lambda \exp(-\lambda x), \quad x > 0.$$

Solution:

According to the transformation method:

$$\begin{aligned} u &= F(x) \\ &= \int_0^x \lambda \, \exp(-\lambda X) \, dX \\ &= 1 - e^{-\lambda x}. \end{aligned}$$

Thus,

$$x = \frac{-\ln(1-u)}{\lambda}.$$

Example

Let us try to write a program that implements this transformation:

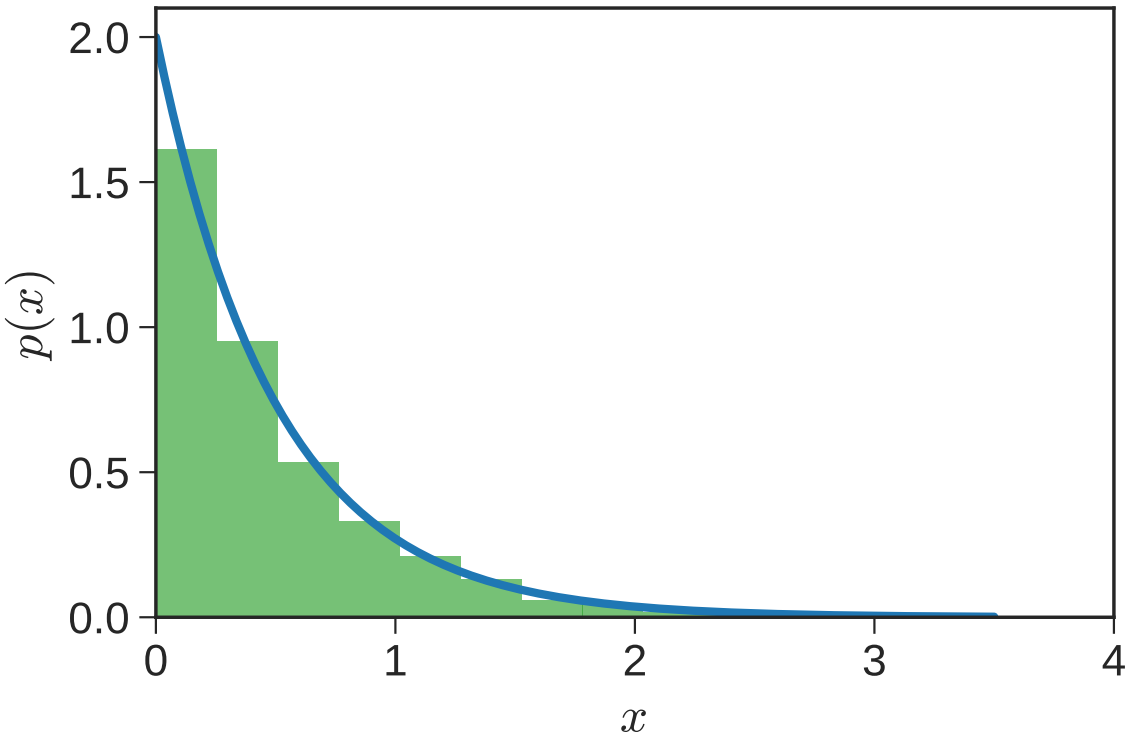
```
def myExponentialSampler(lam, num):
    """Generate -num- exponentially distributed samples"""
    u = np.random.random((num,1))
    x = -np.log(1-u)/lam
    return x
```

Suppose $\lambda = 2$. Generate 5000 numbers and histogram them.

```
x = myExponentialSampler(2.0, 5000)
n, bins, patches = plt.hist(x, 20, density=True,
                            facecolor='green', alpha=0.65)
```

Example

The blue line is the actual distribution $2\exp(-2x)$.



Properties of the Transformation Method

- (-) requires a closed form expression for $F(x)$; or potentially costly numerics
- (-) may not always be the simplest or fastest for a particular target distribution (example Gaussian distribution)
- (-) can become unworkable for non-standard multidimensional PDFs
- (+) generally works quite fast
- (+) works for any monotonically increasing $F(x)$ (could be flat in some regions)
- (+) can be used for continuous, discrete, or mixed continuous-discrete

Exercises

Use the transformation method to sample from the following distributions:

1. Continuous $U[-1, 1]$:

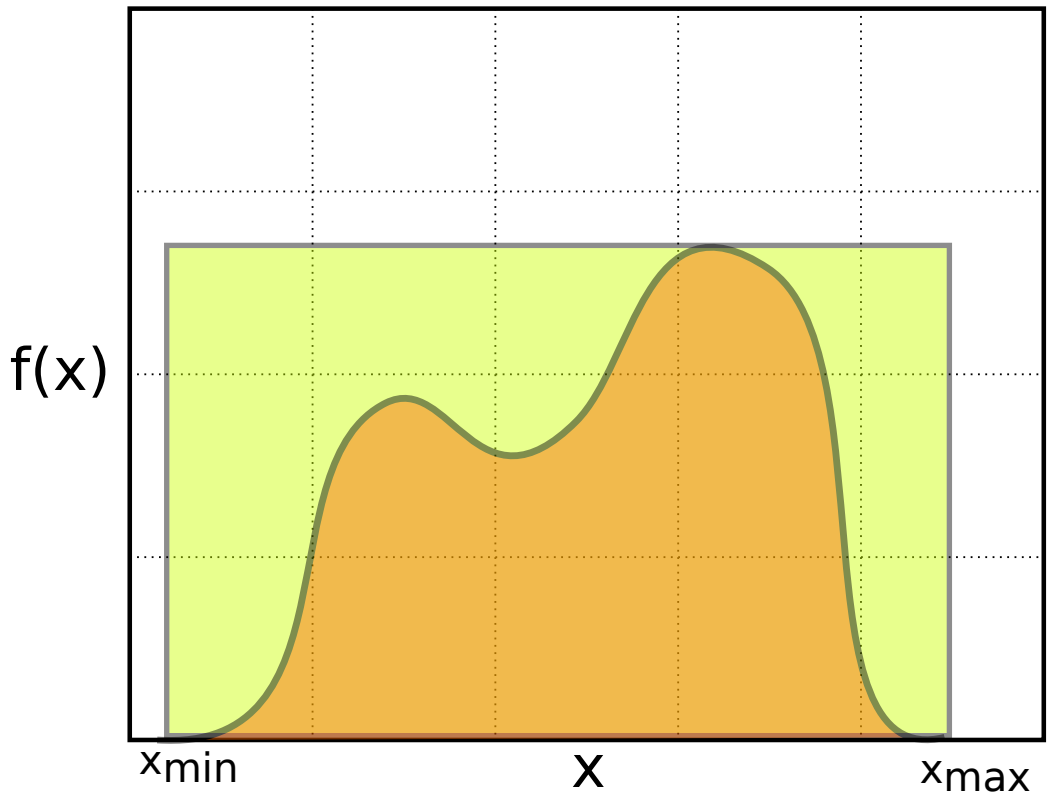
$$f(x) = \begin{cases} 0.5 & -1 \leq x \leq 1 \\ 0 & \text{elsewhere} \end{cases}$$

2. Discrete Coin Toss:

$$f(x) = \begin{cases} 0.5, & x = 0 \\ 0.5, & x = 1 \end{cases}$$

Accept-Reject Method

The idea is to enclose the distribution in an appropriate box, and throw darts at it.

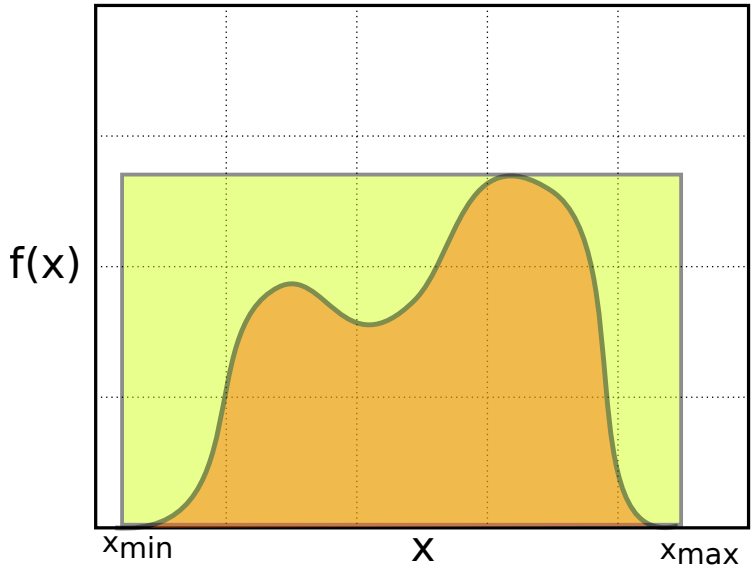


We used this method to compute integrals.

From MC perspective, integration $\subset$ sampling.

Accept-Reject Method

Suppose the distribution $f(x)$, for $x_{\min} \leq x \leq x_{\max}$.



Draw a box of height $f_{\max}$ which encloses the distribution.

The algorithm then works as follows:

- pick $X \sim U[x_{\min}, x_{\max}]$
- pick another random number $U \sim U[0, f_{\max}]$
- If $U \leq f(X)$, then accept X. If not, reject X and repeat.

Example

Consider the distribution

$$f(x) = \frac{3}{8}(1 + x^2), \qquad -1 \leq x \leq 1$$

If we plot the function, we realize that the maximum value occurs at $x = \pm 1$, thus $f_{\max} = f(\pm 1) = 3/4$

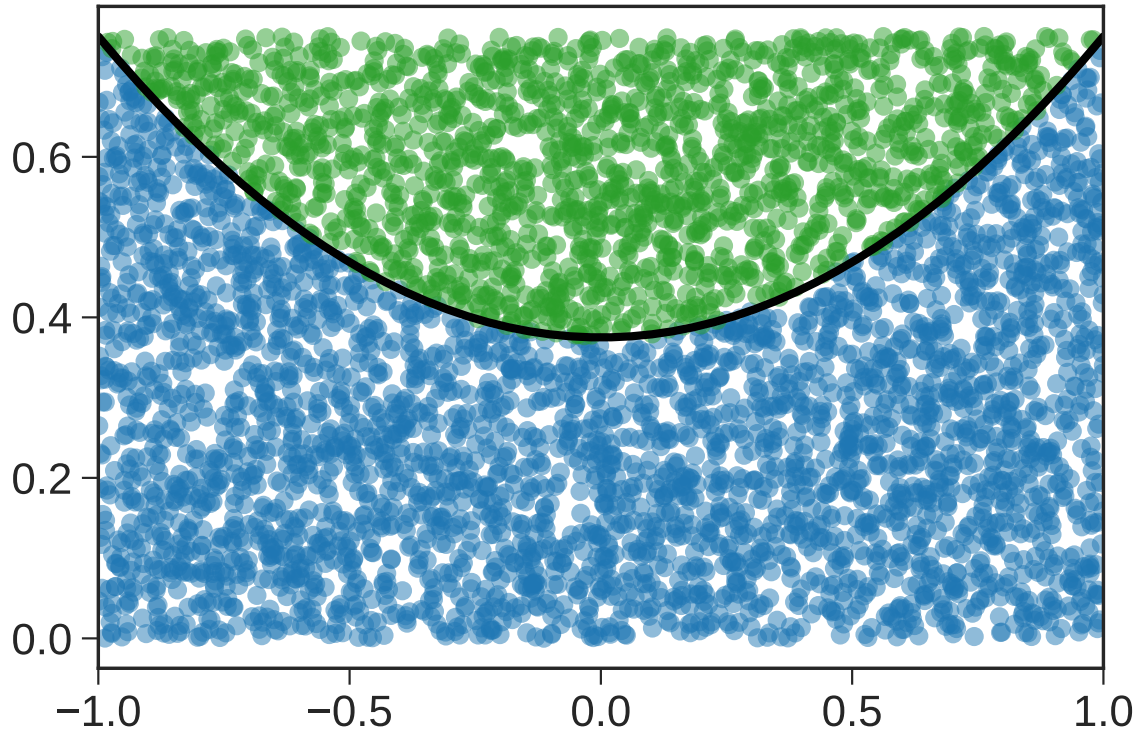
```
def acceptRejectExample(ndarts):
    xmin = -1.
    xmax = 1.
    fmax = 3./4

    x = np.random.uniform(xmin, xmax, ndarts)
    u = np.random.uniform(0., fmax, ndarts)

    y = x[u <= 3./8. * (1 + x**2)]
    return y
```

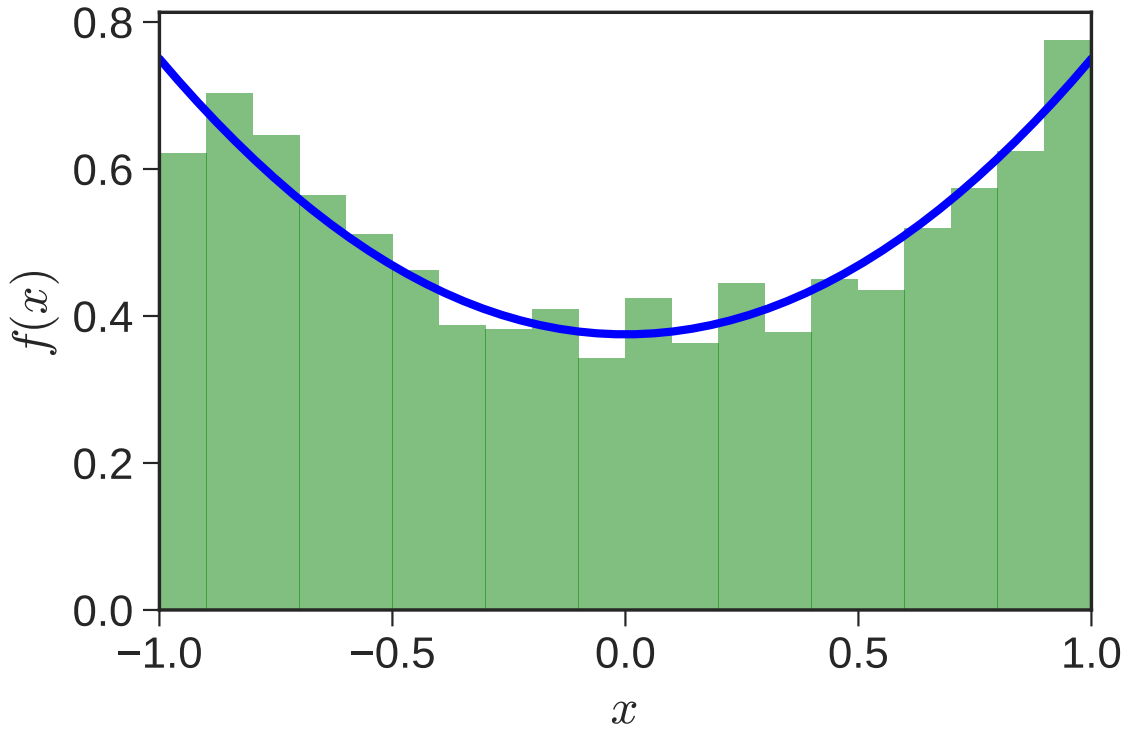
Example

```
x = acceptRejectExample(5000)
```



Let us consider a histogram of the sampled points X.

Example



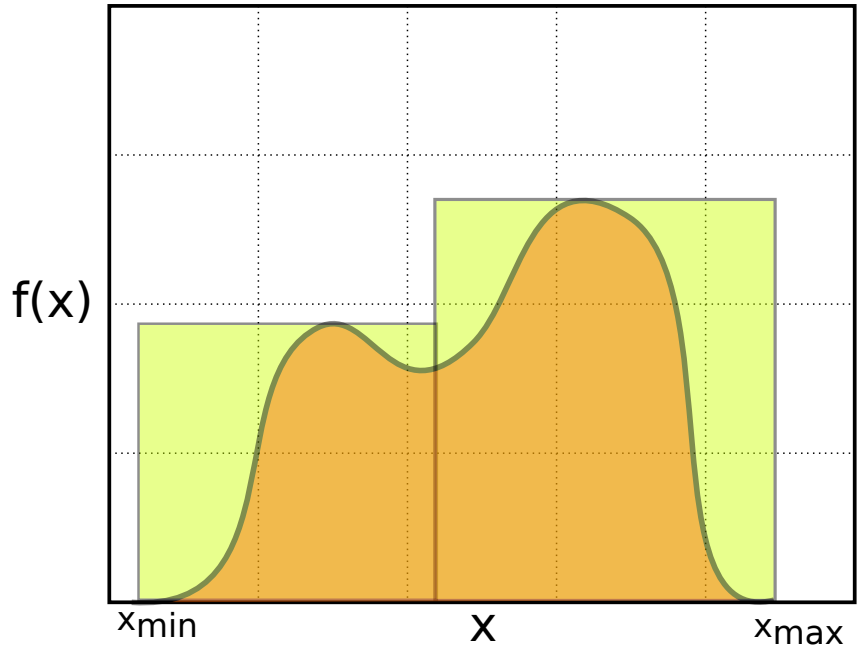
Blue line is the target distribution.

Exercise: Solve this problem using the transformation method.

Accept-Reject Method

This is a general method that can be applied to more than 1D.

- The target distribution need not be invertible
- even if the domain is infinite, the computational domain (e.g. floats) is not infinite
- If the distribution is sharply peaked we may draw multiple boxes of different heights to reduce rejection

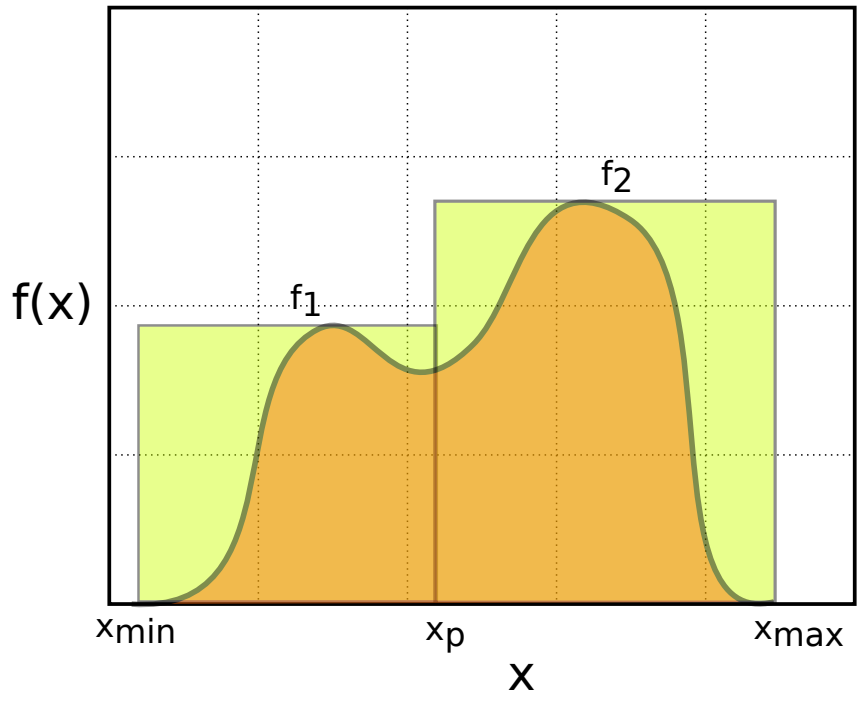


Two Boxes

Be careful not to oversample from the smaller box

A solution: Set the probability of picking x from a particular box proportional to the area of the box.

Suppose the heights of the two boxes are f_1 and f_2, and x_p partitions the domain into two parts.



Two Boxes

- let $A_1 = (x_p - x_{\min})f_1$ and $A_2 = (x_{\max} - x_p)f_2$

$$\text{pick } X \sim \pi(x) = \begin{cases} \dfrac{A_1}{A_1 + A_2} & x < x_p \\ \dfrac{A_2}{A_1 + A_2} & x \geq x_p \end{cases}$$

- depending on X pick U according to, pick

$$U \sim \begin{cases} U[0, f_1] & x < x_p \\ U[0, f_2] & x \geq x_p \end{cases}.$$

- If $u \leq f(x)$, then accept x. If not, reject x and repeat.

Adaptive Rejection Sampling

This modification to the rejection sampling method tries to improve the basic algorithm.

We will not consider it in detail in this class, but it essentially:

- defines the envelope distribution in log space (e.g. log-probability or log-density)
- uses a piecewise linear density function as the envelope
- The envelope is adapted as the sampling is carried on

ARSpy is a pure python package that can perform adaptive rejection sampling in 1D.

PostFace

For 1D sampling, one of these two methods should generally suffice.

As the number of dimensions increases:

- the **transformation method** gets increasingly hard to apply, even if we use numerical rather than analytical inversion
- the number of rejections in the **accept-reject method** increases rapidly

Even sophisticated variants of these simple algorithms can no longer cope with this complexity

Next, we will look at 2D distributions, as a window into these higher dimensional spaces.

As the dimensionality and complexity increase, MCMC methods - which are not particularly competitive for "simple textbook problems" - start to shine!