Random number generation

3F3 laboratory experiment: Full Technical Report

Theo A. Brown
Selwyn College, University of Cambridge

December 8, 2021

Contents

1 Introduction

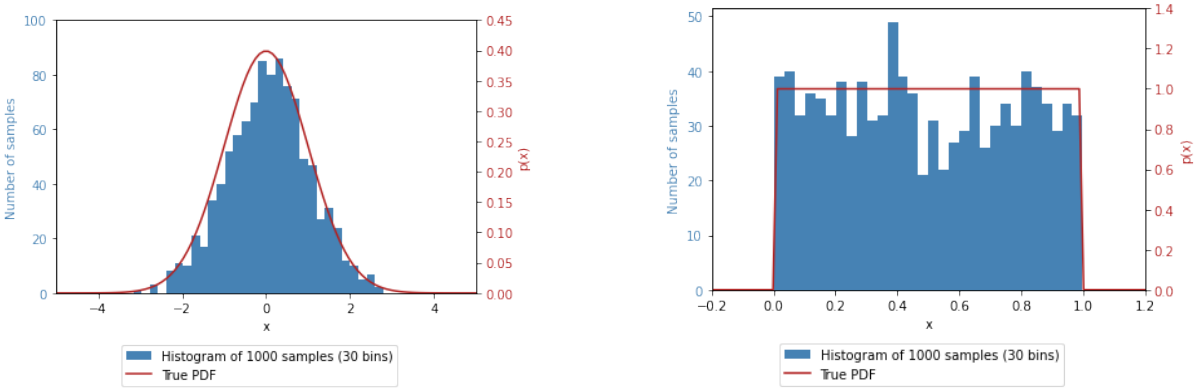
Random numbers are widely used in science and engineering, forming the basis of stochastic modelling and inference. Generators of Gaussian and Uniform random numbers are widely implemented, but in many applications samples from more complex distributions are required. This report investigates methods for generating random numbers from different distributions. Firstly, Uniform and Gaussian random number generators are used to investigate different methods for estimating probability density functions (PDFs) from samples. Secondly, samples from new distributions are generated by transforming samples from a Uniform or Gaussian distribution, using the Jacobian of the transformation. Thirdly, Uniformly distributed random samples are transformed to generate new distributions using the inverse cumulative density function of the desired distribution. Finally, samples from very complex distributions are generated using a scaled mixture of Gaussians, where the desired distribution can be written as a Gaussian distribution with a random variance, set by some PDF. In addition, Monte Carlo methods for estimating properties of a distribution from samples are investigated.

2 Generating random numbers from the Uniform and Normal distributions

A vector of 10000 samples from a unit Gaussian distribution was generated using `np.random.randn`, and a vector of 10000 samples from a unit Uniform distribution was generated using `np.random.rand`. These sample vectors are used throughout the experiment to investigate different methods of random number generation.

2.1 Comparison of histogram with true probability density function

The sample vectors were binned and plotted as histograms. The true analytical probability density function (PDF) was calculated for the unit Gaussian and unit Uniform distributions, and was overlaid on the histogram plots (Figure 1). The histograms closely follow the shape of the analytical PDF, showing that the generation of the random numbers for these distributions is accurate, and that histograms provide a good method for approximating the true PDF.



(a) Gaussian distribution

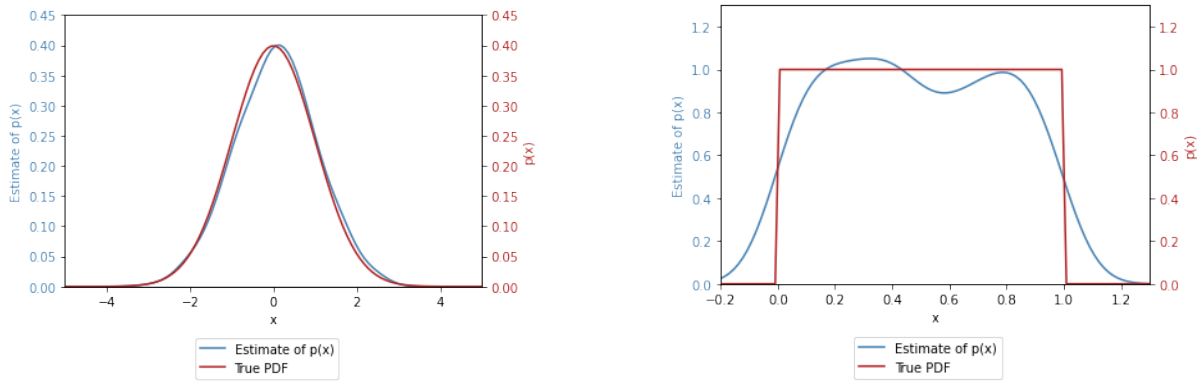
(b) Uniform distribution

Figure 1: Histogram of samples drawn from a distribution, overlaid with the true PDF of the distribution

2.2 Kernel density smoothing

A unit Gaussian kernel $\mathcal{N}(0, 1)$ is applied to the data to provide a continuous estimate for the PDF. The result is plotted for the Gaussian and Uniform distributions in Figure 2.

At each point, the kernel method takes the mean of N neighbouring values, weighted using a Gaussian distribution. This is advantageous as it can smooth random irregularities in the samples: for example, there may be histogram bins with a sample count that is greater than the expected sample count, which would give an incorrectly high estimate of the probability of a sample falling in this bin; the kernel



(a) Gaussian distribution

(b) Uniform distribution

Figure 2: Estimate of the PDF of a distribution, generated using Gaussian kernel smoothing of random samples from the distribution, overlaid with the true PDF of the distribution

smooths out this local peak by taking neighbouring values into account. However, this also means that the kernel smoothing method becomes inaccurate when there is a sudden change or discontinuity in the probability density function. In the Uniform distribution there is zero probability for values outside the range $0 < x < 1$, but as the kernel averages over a window of values it does not capture the step change at $x = 0$ and $x = 1$ and instead decays smoothly.

2.3 Multinomial distribution theory

Multinomial distribution theory can be used to estimate properties of a distribution from a histogram of samples.

2.3.1 Derivation of the multinomial distribution

For N samples and M bins, let N_i be the random variable representing the number of samples in bin i and p_i be the probability that a sample falls in bin i. Samples are drawn independently, so the probability of getting n_i samples in bin i is given by:

$$Pr(N_i = n_i) = \prod_{j=1}^{n_i} p_i = p_i^{n_i}$$

The joint probability of obtaining a given distribution of N samples across M bins is given by:

$$Pr(N_1 = n_1, N_2 = n_2, \ldots, N_M = n_M) = \prod_{i=1}^{M} \binom{N_{\text{available},i}}{n_i} p_i^{n_i} \tag{1}$$

where $N_{\text{available},i} = N - \sum_{j=1}^{i-1} n_j$.

The binomial coefficient in Equation 1 comes from considering all possible combinations of getting n_i samples in bin i:

- For the first bin, n_1 samples are chosen from a set of N
- For the second bin, n_2 samples are chosen from a set of $N - n_1$
- For the ith bin, n_i samples are chosen from a set of $N - n_1 \cdots - n_{i-1}$

Simplifying the coefficient in Equation 1 gives:

$$\begin{aligned}\binom{N}{n_1}\binom{N-n_1}{n_2}\cdots\binom{n_M}{n_M} &= \frac{N!}{n_1!(N-n_1)!}\frac{(N-n_1)!}{n_2!(N-n_1-n_2)!}\cdots\frac{n_M!}{n_M!} \\ &= \frac{N!}{n_1!n_2!\ldots n_M!}\end{aligned}$$

Consequently, Equation 1 simplifies to:

$$Pr(N_1 = n_1, N_2 = n_2, \ldots, N_M = n_M) = \frac{N!}{n_1! n_2! \ldots n_M!} p_1^{n_1} p_2^{n_2} \ldots p_M^{n_M}$$

From considering the probability $p_i = Pr(N_i = n_i)$ for N samples, the mean and standard deviation of N_i can be found:

$$\mu_i = \mathbb{E}[N_i] = Np_i \tag{2}$$

$$\sigma_i = \sqrt{Var[N_i]} = \sqrt{Np_i(1-p_i)} \tag{3}$$

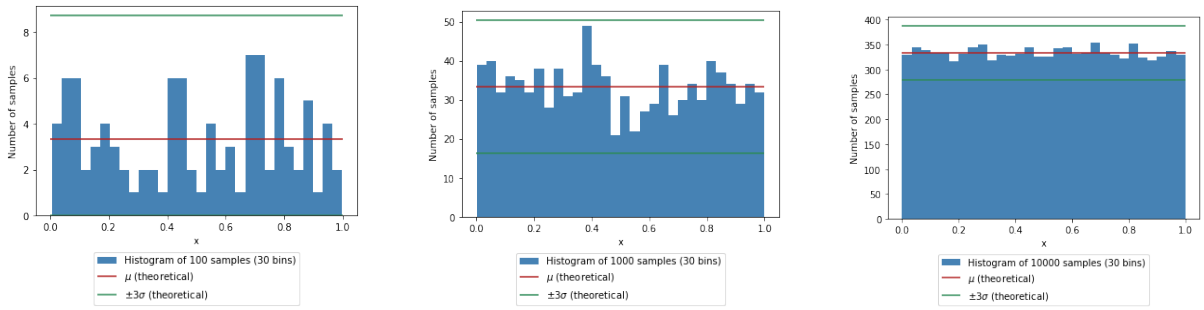
2.3.2 Application to the Uniform distribution

For the Uniform distribution between 0 and 1, the pdf $p(x)$ is defined as:

$$\begin{aligned} p(x) &= \frac{1}{x_{max} - x_{min}} \\ &= 1 \end{aligned}$$

Hence, using Equation 2 and Equation 3, the mean and standard deviation of the number of samples from the Uniform distribution in a histogram with bin width δ and bin centers c_i can be calculated:

$$\begin{aligned} p_i &= \int_{c_i - \frac{\delta}{2}}^{c_i + \frac{\delta}{2}} 1 \, dx \\ &= \delta \\ \mu_i &= N\delta \\ \sigma_i &= \sqrt{N\delta(1-\delta)} \end{aligned}$$



(a) $N = 100$ (b) $N = 1000$ (c) $N = 10000$

Figure 3: Histogram of N samples from a Uniform distribution, showing mean and ±3 standard deviation

Figure 3 shows that the sample count lies within the 3σ interval for almost all bins, which is consistent with the multinomial distribution theory.

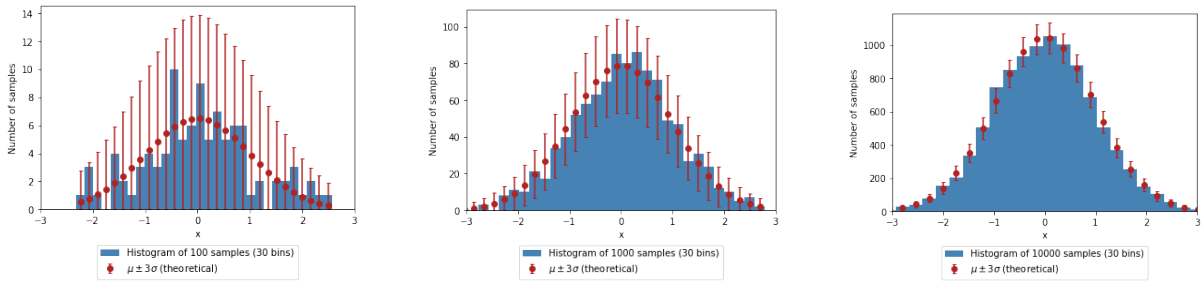
2.3.3 Application to the Gaussian distribution

For the Gaussian distribution:

$$\begin{aligned} p_i &= \int_{c_i - \frac{\delta}{2}}^{c_i + \frac{\delta}{2}} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2}x^2\right) dx \\ &= \Phi\left(c_i + \frac{\delta}{2}\right) - \Phi\left(c_i - \frac{\delta}{2}\right) \end{aligned}$$

The mean and variance for each bin are calculated according to Equation 2 and Equation 3.

Figure 4 shows that the sample count lies within the 3σ interval for almost all bins, which is consistent with the multinomial distribution theory.



(a) $N = 100$ (b) $N = 1000$ (c) $N = 10000$

Figure 4: Histogram of N samples from a Gaussian distribution, showing mean and ±3 standard deviation

2.3.4 Effect of increasing N

Multinomial distribution theory suggests that the method of using a histogram of samples to estimate properties of a distribution increases in accuracy as the number of samples N increases. For a given bin probability p_j, the standard deviation increases less rapidly than the mean with increasing N ($\sigma \propto \sqrt{N}$ whereas $\mu \propto N$). Consequently an increase in N will result in the standard deviation becoming smaller compared to the mean ($\frac{\sigma}{\mu} \propto \frac{1}{\sqrt{N}}$). This can be seen in Figure 3 and Figure 4, where the standard deviation reduces as N increases.

2.3.5 Effect of bin probabilities

Figure 4 shows that the variance in the number of samples N_i in bin i increases as p_i increases. The relationship between $Var[N_i]$ and p_i is clear from Equation 3. Figure 5 is a plot of the relationship. For the Gaussian distribution with 30 bins, all of the bin probabilities remain small, so the variance of N_i increases with increasing bin probability. For another distribution or a histogram with some bin probabilities greater than 0.5, it would be observed that the variance decreases with increasing bin probability.

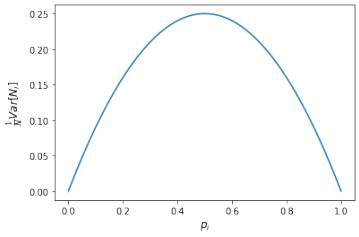


Figure 5: Plot of normalised variance against bin probability

3 Generating random numbers using functions of other random numbers

Samples from new distributions can be generated by transforming samples from existing distributions. The PDF of the new distribution, $p_Y(y)$, can be found using the initial distribution, $p_X(x)$, and the Jacobian of the transformation, as follows:

$$p_Y(y) = \sum_{k=1}^{K} \left. \frac{p_X(x)}{\left|\frac{dy}{dx}\right|} \right|_{x=x_k(y)}$$

where $\{x_1(y) \dots x_K(y)\}$ is the set of values for $f^{-1}(y)$.

3.1 $f(x) = ax + b$

For $x \sim \mathcal{N}(0, 1)$, let $y = f(x) = ax + b$. Then:

$$\begin{aligned} f^{-1}(y) &= \frac{y-b}{a} \\ \left|\frac{dy}{dx}\right| &= |a| \\ p_Y(y) &= \frac{1}{|a|} p_X\left(\frac{y-b}{a}\right) \\ &= \frac{1}{|a|} \times \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y-b}{a}\right)^2\right) \end{aligned}$$

Comparing with the equation for a general Gaussian distribution, it is clear that $p_Y(y)$ is a Gaussian distribution with mean b and variance a^2. The result is compared with a histogram of transformed samples in Figure 6. The close agreement suggests this hypothesis is correct.

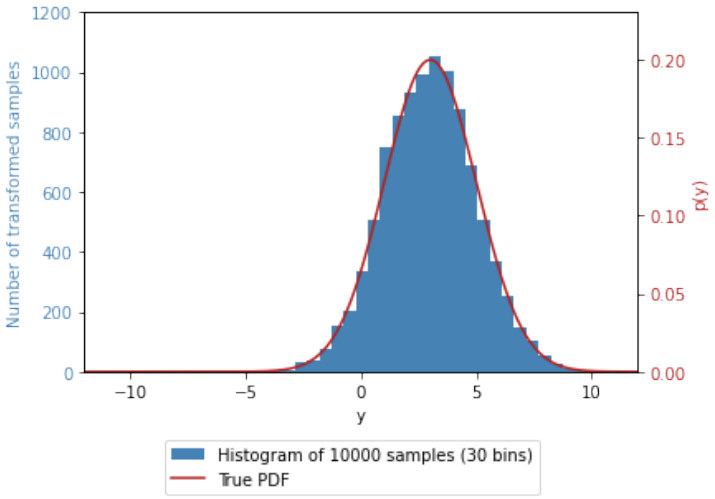


Figure 6: Histogram of linear function ($f(x^{(i)}) = 2x^{(i)} + 3$) of Gaussian samples overlaid with calculated PDF

3.2 $f(x) = x^2$

For $x \sim \mathcal{N}(0, 1)$, let $y = f(x) = x^2$. Then:

$$\begin{aligned} f^{-1}(y) &= \pm\sqrt{y} \\ \left|\frac{dy}{dx}\right| &= |2x| \\ p_Y(y) &= \sum_{i=1}^{2} \frac{1}{|2\sqrt{y}|} p_X\left(\sqrt{y}\right) \\ &= \frac{1}{\sqrt{2\pi y}} \exp\left(-\frac{1}{2}y\right) \end{aligned}$$

The result is compared with a histogram of transformed samples in Figure 7. The close agreement suggests this hypothesis is correct.

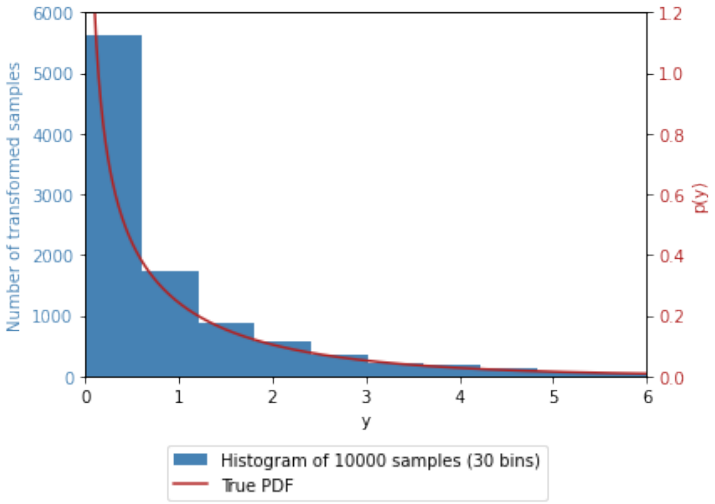


Figure 7: Histogram of quadratic function $\left(f(x^{(i)}) = \left(x^{(i)}\right)^2\right)$ of Gaussian samples overlaid with calculated PDF

3.3 $f(x) = \sin(x)$

For $x \sim \mathcal{U}(0, 2\pi)$, let $y = f(x) = \sin(x)$. Then:

$$\begin{aligned} p(x) &= \frac{1}{2\pi} \\ f^{-1}(y) &= \sin^{-1}(y) \\ \left|\frac{dy}{dx}\right| &= |\cos(x)| \\ p_Y(y) &= \sum_{i=1}^{2} \frac{\frac{1}{2\pi}}{\left|\cos\left(\sin^{-1}(y)\right)\right|} \end{aligned}$$

By drawing a right-angled triangle with hypotenuse of length 1 and a side of length y, and applying trigonometry and Pythagoras' theorem, it can be shown that:

$$\cos\left(\sin^{-1}(y)\right) = \sqrt{(1-y^2)}$$

Hence:

$$p_Y(y) = \frac{1}{\pi\sqrt{(1-y^2)}}$$

The result is compared with a histogram of transformed samples in Figure 8. The close agreement suggests this hypothesis is correct.

3.4 $f(x) = \mathbf{clip}(\sin(x), 0.7)$

For $x \sim \mathcal{U}(0, 2\pi)$, let[1]:

$$y = f(x) = \begin{cases} -0.7 & \sin(x) < -0.7 \\ \sin(x) & |\sin(x)| < 0.7 \\ 0.7 & \sin(x) > 0.7 \end{cases}$$

This produces a probability density with the following properties:

- $|y|$ will never be exceed 0.7. Hence $p_Y(y)$ will be zero for $|y| > 0.7$.

[1]Note that this function differs from the one in the handout by being symmetric, which is a better model of realistic signal clipping.

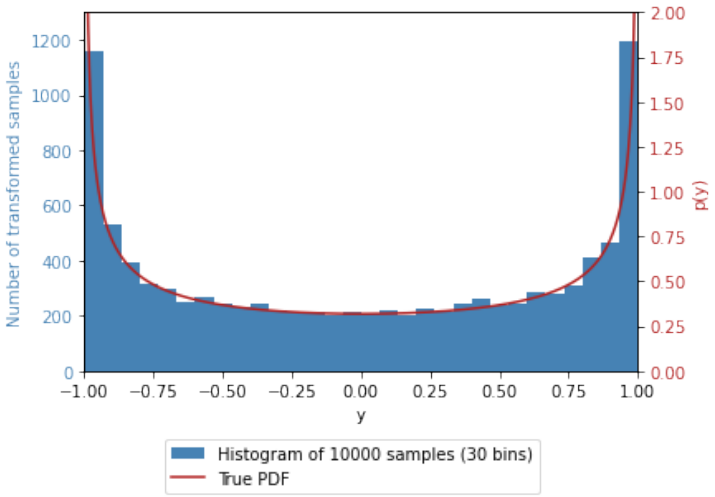


Figure 8: Histogram of sinusoidal function $\left(f(x^{(i)}) = \sin\left(x^{(i)}\right)\right)$ of Uniform samples overlaid with calculated PDF

- $f(x)$ collapses all values less than -0.7 to -0.7, and all values greater than 0.7 to 0.7. Hence $p_Y(y)$ will have delta functions at $y = 0.7$ and $y = -0.7$, containing all of the probability mass from $y > 0.7$ and $y < -0.7$ respectively.
- For $|y| < 0.7$, $p_Y(y) = \frac{1}{\pi\sqrt{(1-y^2)}}$ as before.

The area A of each delta function can be calculated using the fact that the PDF must integrate to 1:

$$\begin{aligned} A &= \frac{1}{2}\left(1 - \int_{-0.7}^{0.7} \frac{1}{\pi\sqrt{1-y^2}}\, dy\right) \\ &= \frac{1}{2} - \frac{1}{\pi}\sin^{-1}(0.7) \end{aligned}$$

Which gives the PDF as:

$$p_Y(y) = \begin{cases} A\delta(y+0.7) & y = -0.7 \\ \frac{1}{\pi\sqrt{(1-y^2)}} & |y| < 0.7 \\ A\delta(y-0.7) & y = 0.7 \\ 0 & \text{otherwise} \end{cases}$$

The presence of the delta function in the PDF can also be explained using the Jacobian formula. Consider the derivative of y:

$$\frac{dy}{dx} = \begin{cases} \cos(x) & 0 < x < \sin^{-1}(0.7) \\ 0 & \sin^{-1}(0.7) < x < \pi - \sin^{-1}(0.7) \\ \cos(x) & \pi - \sin^{-1}(0.7) < x < \pi + \sin^{-1}(0.7) \\ 0 & \pi + \sin^{-1}(0.7) < x < 2\pi - \sin^{-1}(0.7) \end{cases}$$

As the Jacobian formula involves dividing by $\frac{dy}{dx}$, $p_Y(y)$ is undefined for the range $|y| > 0.7$, which indicates the presence of a delta function in the PDF.

The calculated PDF is compared with a histogram of transformed samples in Figure 9. The close agreement suggests this hypothesis is correct.

4 Generating random numbers using the inverse CDF method

Samples from a distribution with PDF $p_Y(y)$ can sometimes be generated from the uniform distribution $p_X(x) = \mathcal{U}(x|0,1)$ by using the inverse cumulative density function of Y

$$y^{(i)} = F_Y^{-1}\left(x^{(i)}\right)$$

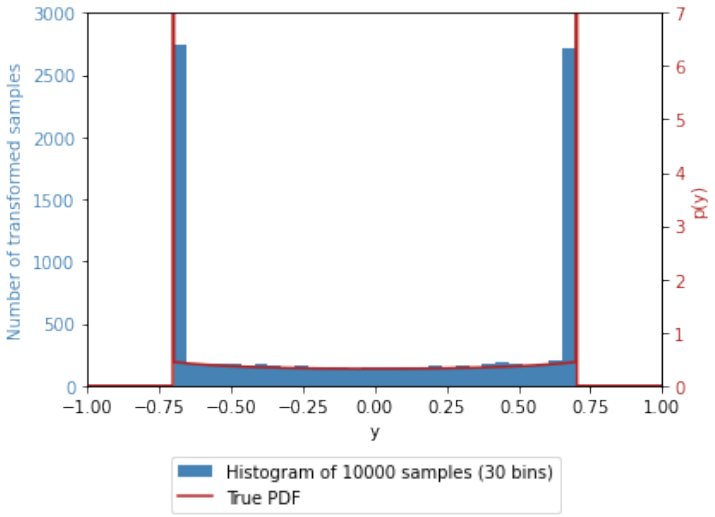


Figure 9: Histogram of limited sinusoidal function $\left(f(x^{(i)}) = \text{clip}\left(\sin\left(x^{(i)}\right), 0.7\right)\right)$ of Uniform samples overlaid with calculated PDF

4.1 Exponential distribution

For the exponential distribution with mean one:

$$\text{PDF: } p(y) = \exp(-y), \ y \geq 0$$
$$\text{CDF: } F(y) = \int_0^y p(y')dy' = 1 - \exp(-y)$$
$$\text{iCDF: } F^{-1}(x) = -\ln(1-x)$$

Figure 10 shows that samples from the exponential distribution generated using the iCDF follow the true PDF.

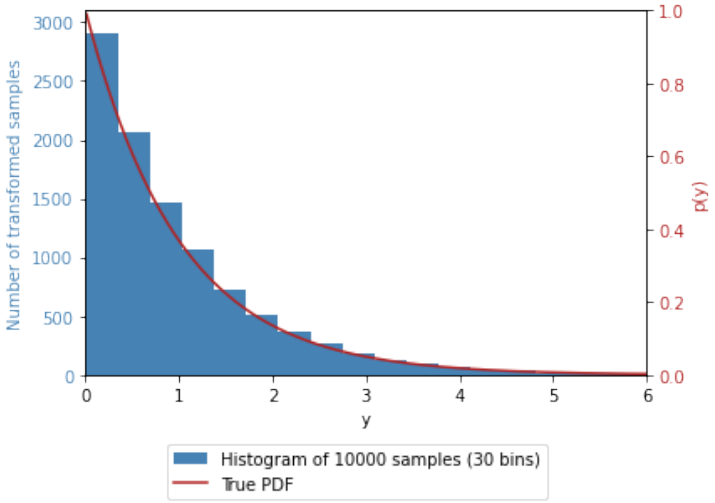


Figure 10: Histogram of samples drawn from the uniform distribution and transformed using the iCDF method to follow the exponential distribution, overlaid with the true exponential PDF

4.2 Estimating properties of the exponential distribution

Properties of the distribution can be estimated using Monte Carlo methods:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} y^{(i)}$$
$$\hat{\sigma^2} = \frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)}\right)^2 - \hat{\mu}^2$$

4.2.1 Properties of Monte Carlo mean estimator

The Monte Carlo mean estimate $\hat{\mu}$ can be shown to unbiased:

$$\begin{aligned}\mathbb{E}[\hat{\mu}] &= \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[y^{(i)}\right] = \frac{1}{N}\sum_{i=1}^{N}\mu \\ &= \frac{N\mu}{N} = \mu\end{aligned}$$

To derive the variance of the Monte Carlo mean estimator, consider the variance of the sum of random variables X and Y:

$$\begin{aligned}Var[X+Y] &= \mathbb{E}\left[(X+Y-\mu_X-\mu_Y)^2\right] \\ &= \mathbb{E}\left[((X-\mu_X)+(Y-\mu_Y))^2\right] \\ &= \mathbb{E}\left[(X-\mu_X)^2\right]] + \mathbb{E}\left[(Y-\mu_Y)^2\right] + 2\mathbb{E}\left[(X-\mu_X)(Y-\mu_Y)\right]\end{aligned}$$

Using the fact that X and Y are independent gives $\mathbb{E}\left[(X-\mu_X)(Y-\mu_Y)\right] = 0$, hence:

$$Var[X+Y] = Var[X] + Var[Y]$$

When multiplying a random variable by a scalar, the variance is multiplied by the square of the scalar:

$$\begin{aligned}Var[kX] &= \mathbb{E}\left[(kX)^2\right] - \mathbb{E}[kX]^2 \\ &= k^2\mathbb{E}\left[X^2\right] - k^2\mathbb{E}[X]^2 \\ &= k^2 Var[kX]\end{aligned}$$

Treating the samples $y^{(i)}$ as independent random variables and combining the above properties gives:

$$Var[\hat{\mu}] = Var\left[\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\right] = \frac{1}{N^2}\sum_{i=1}^{N} Var\left[y^{(i)}\right] = \frac{\sigma^2}{N}$$

The convergence of the Monte Carlo estimators for the mean and variance is shown in Figure 11. It is apparent that the mean estimator does indeed converge proportional to $\frac{1}{\sqrt{N}}$ (see Figure 11c).

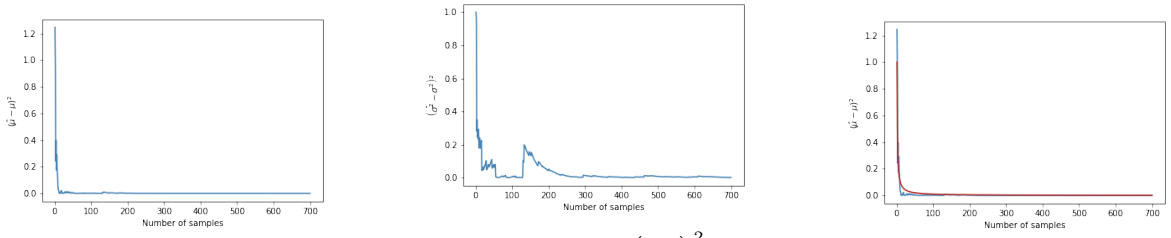
4.2.2 Application to the exponential distribution

The exponential distribution $p(y) = \exp(-y),\ y \geq 0$ has $\mu = 1$ and $\sigma^2 = 1$. Applying Monte Carlo estimation to 10000 samples gives $\hat{\mu} = 1.0035$ and $\hat{\sigma^2} = 1.0128$, which are close to the true values.

5 Generating random numbers using a scaled mixture of Gaussians

Samples from more complex distributions can sometimes be generated using samples from Gaussian distributions where the variance of the Gaussian is set by another distribution, $p(u)$:

$$p(x) = \int_0^\infty \mathcal{N}(x|0,u)p(u)du \tag{4}$$



(a) $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} y^{(i)}$ (b) $\hat{\sigma^2} = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}\right)^2 - \hat{\mu}^2$ (c) $\hat{\mu}$ with $\frac{1}{N}$ overlay

Figure 11: Behaviour of Monte Carlo estimators for increasing number of samples

5.1 Setting the variance with the Exponential distribution

Samples $u^{(i)}$ can be generated from the distribution with the following PDF:

$$p(u) = \frac{\alpha^2}{2}\exp\left(-\frac{\alpha^2}{2}u\right)$$

This distribution has:

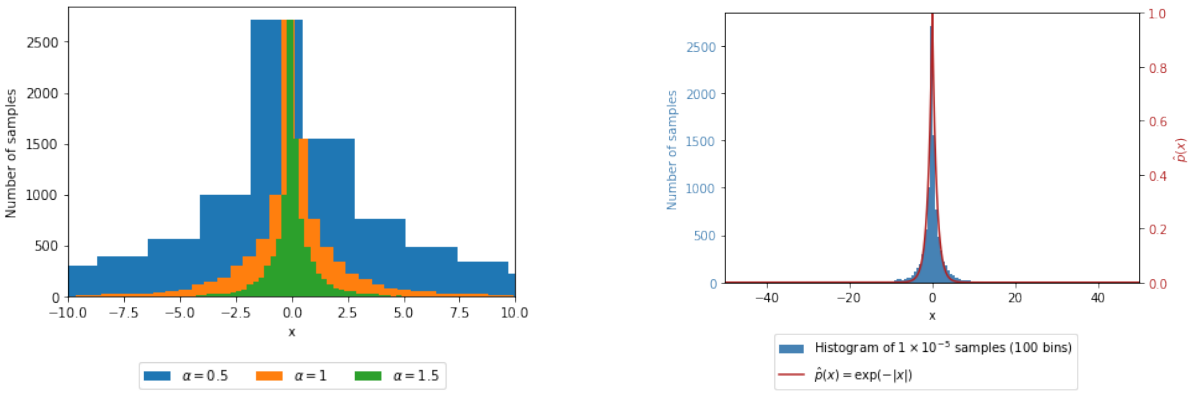
$$\text{CDF: } F(u) = \int_0^u p(u')du' = 1 - \exp\left(-\frac{\alpha^2}{2}u\right)$$

$$\text{iCDF: } F^{-1}(v) = -\frac{2}{\alpha^2}\ln(1-v)$$

Hence samples $u^{(i)}$ can be generated from Uniformly distributed random samples $v^{(i)}$ using the iCDF method. The samples $u^{(i)}$ are then used to set the variance of the random variable X:

$$p(x) = \int_0^\infty \mathcal{N}(x\,|0,u)\frac{\alpha^2}{2}\exp\left(-\frac{\alpha^2}{2}u\right)du \tag{5}$$

The limit of the integral is from zero to infinity because the standard deviation must be positive. A histogram of samples of $x^{(i)}$ is plotted in Figure 12 for different values of α. It appears that α sets the standard deviation of the distribution.



(a) Histogram for different values of α (b) Histogram for $\alpha = 1$ with $\hat{p}(x) = \exp(-|x|)$

Figure 12: 100-bin histogram of samples drawn from the distribution in Equation 5

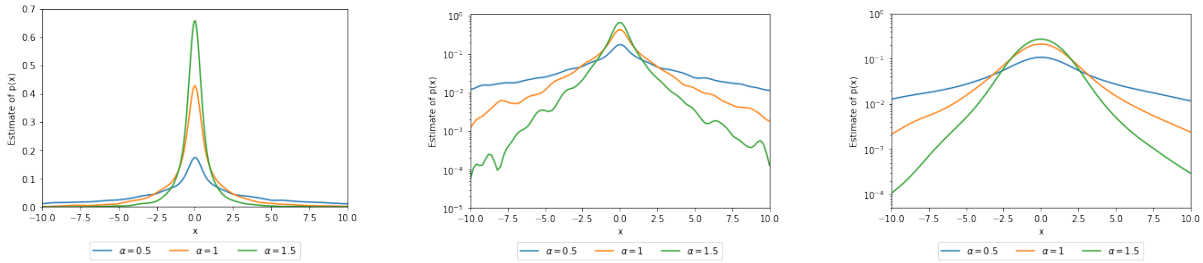
The kernel method was applied to the data, and the smoothed density was plotted on linear and log axes at different scales (Figure 13[2]). From the shape of the distribution, it appears to be a sharper version of a Gaussian distribution, with significantly higher probability density in the centre and tails that last for longer. From Figure 13c it appears that the distribution may follow some exponential

[2]Note that Figure 13b provides little useful information for the tail behaviour of the distribution, as there are very few samples for the extreme tails, so the plot is dominated by the behaviour of the kernel.

relationship, as the tails appear approximately linear on the logarithmic plot. It can also be observed that the distribution is symmetric about zero. Combining these properties suggests that the PDF might be:

$$p(x) \propto \exp(-|x|) \tag{6}$$

This PDF is overlaid on the histogram of samples for $\alpha = 1$ in Figure 12b. There is a close agreement between the shape of the histogram and the PDF, although it is apparent that additional scaling factors (presumably dependent on α) are required to find the exact PDF.



(a) Linear scale (b) Log scale, narrow kernel (c) Log scale, wide kernel

Figure 13: Kernel smoothed probability density estimates $\hat{p}(x)$

5.2 Setting the variance with the Gamma distribution

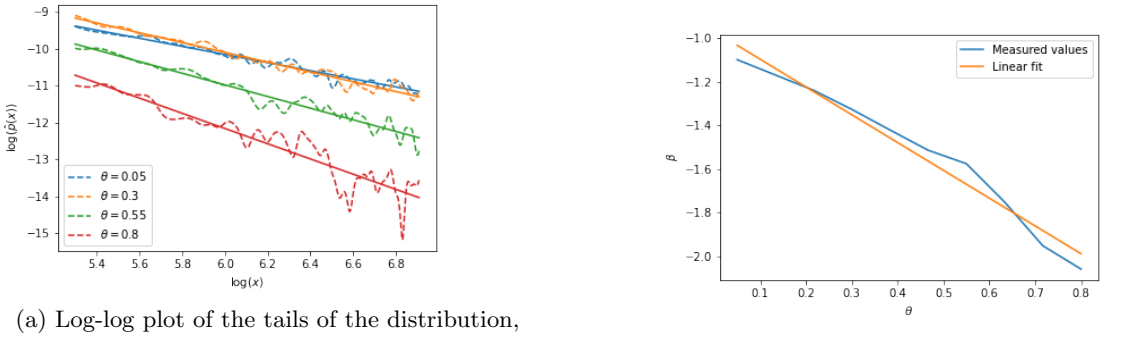
Another distribution can be generated using the Gamma distribution to set the variance of the Gaussian distribution:

$$p(x|u) = \mathcal{N}(x|0, v^{-1}),\ p(v) = \mathcal{G}(v|\theta, \theta^{-1}) \tag{7}$$

5.2.1 Properties of the resulting distribution

Histograms of samples from the distribution are plotted in Figure 15. For low values of θ, the distribution exhibits heavy-tailed behaviour, occasionally generating samples very far from the mean (Figure 15a, Figure 15c, Figure 15e). It is apparent that increasing θ flattens the distribution, reducing the peak at $x = 0$ and widening the central part of the distribution, while also reducing the heavy-tailed behaviour. As θ becomes large (Figure 15g, Figure 15i), the distribution converges to a more Gaussian distribution. For large θ, the distribution is almost unaffected by changes in θ. Compared to the previous distribution, this distribution has a much heavier tail.

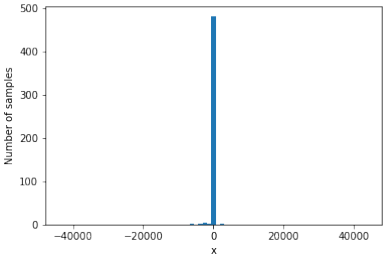
The tail probability is assumed to follow $p(x) = |x|^{-\beta}$. The value of β can be estimated by plotting a graph of $(\log(x), \log(\hat{p}(x)))$, which should give a linear relationship $\log(\hat{p}(x)) = -\beta \log(x)$. This is plotted in Figure 14a, with a least-squares linear fit. The linear regression fits well, which suggests the assumption of the form of $p(x)$ is correct. Plotting the tail gradient β for different values of θ (Figure 14b) allows the relationship $\beta = f(\theta)$ to be determined. From this data, $\beta = 1.28\theta + 0.968$.



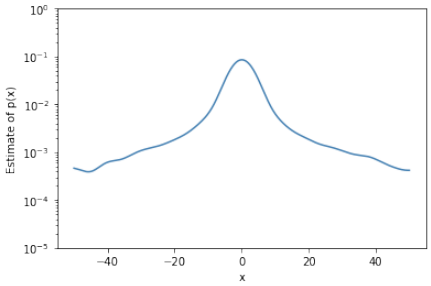
(a) Log-log plot of the tails of the distribution, with linear fit

(b) Plot of the tail gradient β against θ

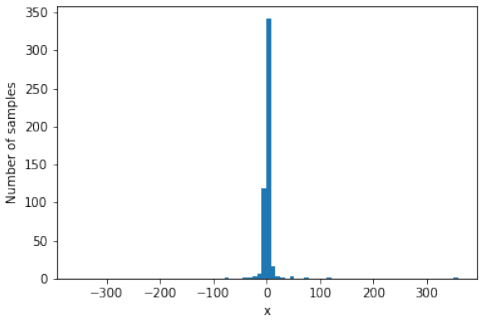
Figure 14: Estimating $p(x)$ in the tails of the distribution



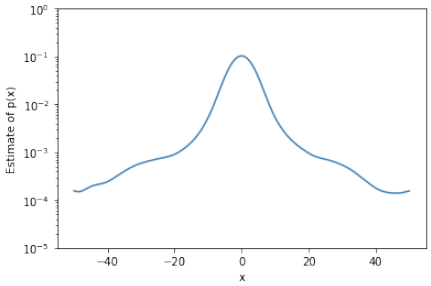
(a) Histogram, $\theta = 0.5$



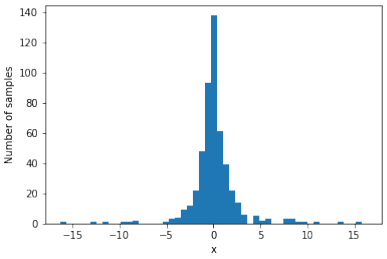
(b) PDF estimate, $\theta = 0.5$



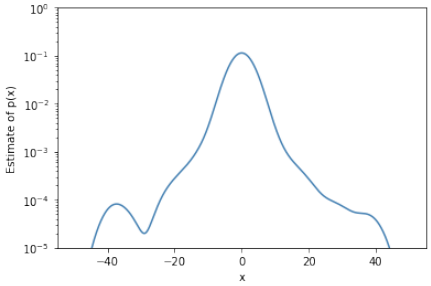
(c) Histogram, $\theta = 1$



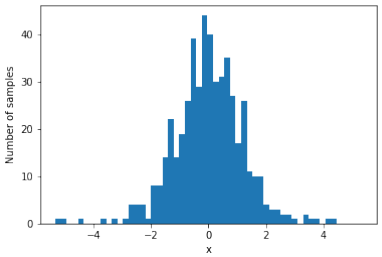
(d) PDF estimate, $\theta = 1$



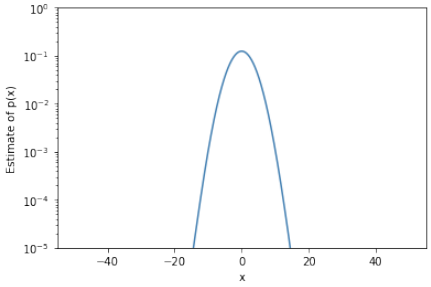
(e) Histogram, $\theta = 2$



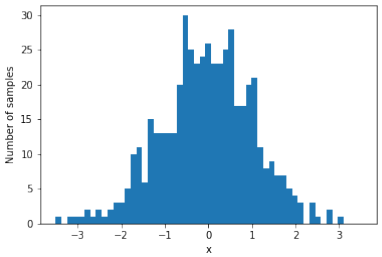
(f) PDF estimate, $\theta = 2$



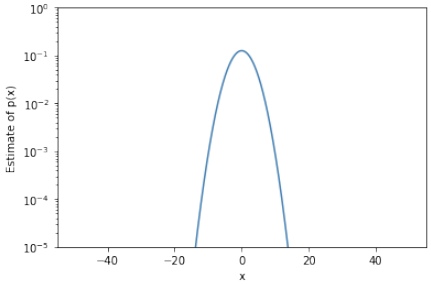
(g) Histogram, $\theta = 10$



(h) PDF estimate, $\theta = 10$



(i) Histogram, $\theta = 100$



(j) PDF estimate, $\theta = 100$

Figure 15: Histograms of samples from the distribution in Equation 7, and logarithmic plots of the kernel smoothed estimate of the PDF $\hat{p}$

5.2.2 PDF of $U = \frac{1}{V}$

The probability density function of the random variable U can be calculated by applying the Jacobian method:

$$\begin{aligned} p(v) &= \mathcal{G}(v|\theta, \theta^{-1}) \\ &= \frac{1}{\theta^{-\theta}\Gamma(\theta)} v^{\theta-1} e^{-v\theta} \\ \left|\frac{du}{dv}\right| &= \left|-v^{-2}\right| = v^{-2} \\ f^{-1}(u) &= u^{-1} \end{aligned}$$

Hence:

$$\begin{aligned} p(u) &= \left.\frac{p(v)}{v^{-2}}\right|_{v=u^{-1}} \\ &= \frac{\theta^\theta}{u^2\Gamma(\theta)} u^{1-\theta} e^{-\frac{\theta}{u}} \end{aligned}$$

$$p(u) = \frac{\theta^\theta}{\Gamma(\theta)} u^{-1-\theta} e^{-\frac{\theta}{u}} \tag{8}$$

5.2.3 PDF of $X \sim \mathcal{N}(0, u)$

The PDF of $X \sim \mathcal{N}(0, u)$, where $p(u)$ is taken from Equation 8, can be obtained using Equation 4:

$$\begin{aligned} p(x) &= \int_0^\infty \frac{1}{\sqrt{2\pi}} u^{-\frac{1}{2}} e^{-\frac{1}{2}x^2 u^{-1}} \frac{\theta^\theta}{\Gamma(\theta)} u^{-1-\theta} e^{-\theta u^{-1}} du \\ &= \frac{\theta^\theta}{\sqrt{2\pi}\Gamma(\theta)} \int_0^\infty \left(u^{-1}\right)^{\frac{3}{2}+\theta} e^{-\left(\theta+\frac{1}{2}x^2\right)u^{-1}} du \end{aligned}$$

Performing a change of variable on u by setting $u = v^{-1}$, and using the fact that $\frac{du}{dv} = -v^{-2}$, gives:

$$p(x) = -\frac{\theta^\theta}{\sqrt{2\pi}\Gamma(\theta)} \int_{-\infty}^0 v^{\frac{3}{2}+\theta} e^{-\left(\theta+\frac{1}{2}x^2\right)v} v^{-2} dv$$

$$p(x) = \frac{\theta^\theta}{\sqrt{2\pi}\Gamma(\theta)} \int_0^\infty v^{-\frac{1}{2}+\theta} e^{-\left(\theta+\frac{1}{2}x^2\right)v} dv \tag{9}$$

From the definition of the Gamma distribution, we know that:

$$\int_0^\infty v^{a-1} e^{-\frac{v}{b}} dv = b^a \Gamma(a)$$

Which is the integral from Equation 9 with

$$\begin{aligned} a &= \theta + \frac{1}{2} \\ b &= \left(\theta + \frac{1}{2}x^2\right)^{-1} \end{aligned}$$

Hence Equation 9 solves to become:

$$p(x) = \frac{\theta^\theta \Gamma\left(\theta + \frac{1}{2}\right)}{\sqrt{2\pi}\Gamma(\theta)} \left(\theta + \frac{1}{2}x^2\right)^{-\theta-\frac{1}{2}} \tag{10}$$

When $|x|$ is large, $\theta << \frac{1}{2}x^2$, so Equation 10 becomes

$$p(x) \approx \frac{2^{\theta+\frac{1}{2}}\theta^\theta \Gamma\left(\theta + \frac{1}{2}\right)}{\sqrt{2\pi}\Gamma(\theta)} x^{-2\theta-1} \tag{11}$$

Hence in the tails, $p(x) \propto x^{-\beta}$, with $\beta = 2\theta + 1$. This is similar to the estimated relationship in the previous section, although the exact coefficients of the linear relationship differ. This may be due to the interference of the Gaussian kernel in the data, or to the sparsity of values in the tails of the distribution, both of which render the tail gradient estimates inaccurate.

6 Conclusions

- Existing software methods were shown to be accurate at generating Uniform and Gaussian random samples.
- A Gaussian kernel was shown to be useful at providing a smooth estimate of a distribution's probability density function from samples drawn from the distribution, where the distribution does not exhibit discontinuities or sudden changes.
- Histogram bin probabilities were shown to follow a multinomial distribution, which can be used to show that the histogram increases in accuracy as the number of samples increases.
- Transformations of random samples were used to generate samples from new distributions, using the Jacobian of the transformation.
- The inverse cumulative density function was shown to be useful in generating random samples from new distributions given only samples from the uniform distribution.
- Monte Carlo methods for estimating the mean and the variance were shown to be accurate and consistent, despite exhibiting slow $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$ convergence.
- Scaled mixtures of Gaussians were used to generate more complex distributions, which can have specific properties such as heavier tails.

Jupyter Notebook

December 8, 2021

[161]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
```

[162]:
```
def gaussian_pdf(x, mu=0, sigma=1):
        u = (x - mu) / abs(sigma)
        y = (1 / (np.sqrt(2 * np.pi) * abs(sigma)))
        return y * np.exp(-0.5 * u**2)
```

[163]:
```
def uniform_pdf(x):
    return 1 * ((x >= 0) & (x <= 1))
```

[164]:
```
def kernel_smoothed_density(x_values, samples, width=0.3,␣
↪kernel_function=gaussian_pdf):
    # Generate an array of kernel values centred on the samples
    kernel_values = [kernel_function(x_value, samples, width) for x_value in␣
↪x_values]
    return np.average(kernel_values, axis=1)
```

[165]:
```
# General global variables
color1 = 'steelblue'
color2 = 'firebrick'
color3 = 'seagreen'
color4 = 'darkorange'
```

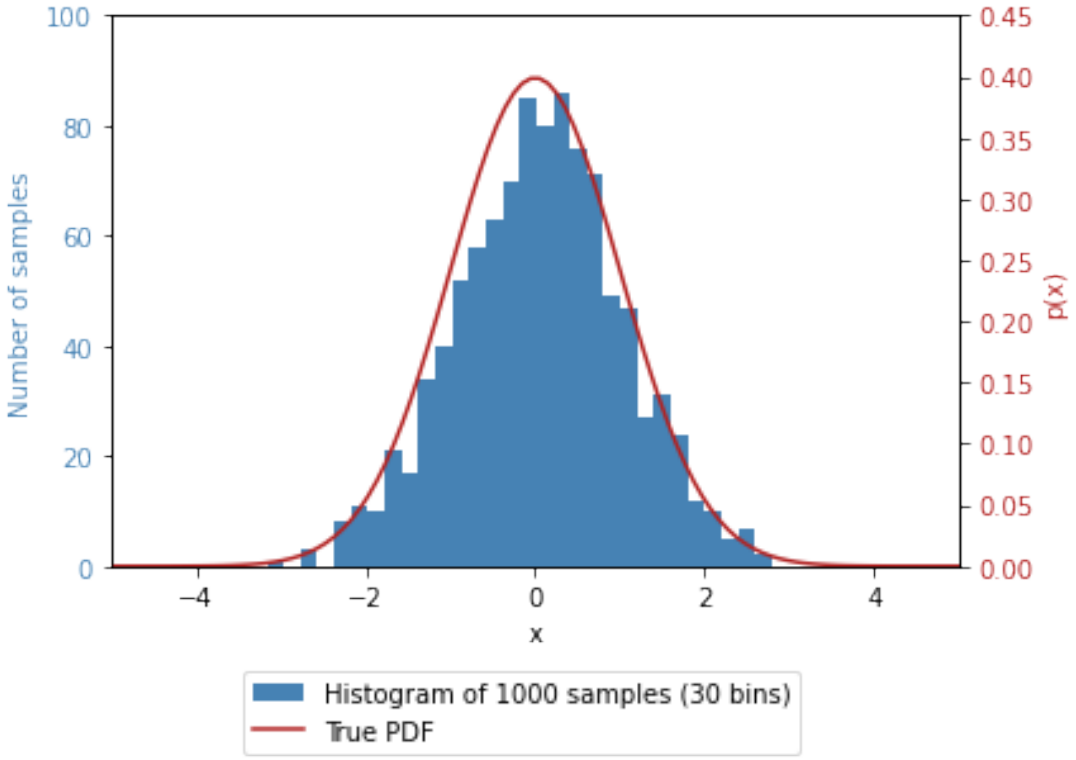
0.0.1 Section 1: Uniform and normal random variables

[166]:
```
X_gaussian = np.random.randn(10000)
X_uniform = np.random.rand(10000)
```

Histograms of random samples compared with true pdf

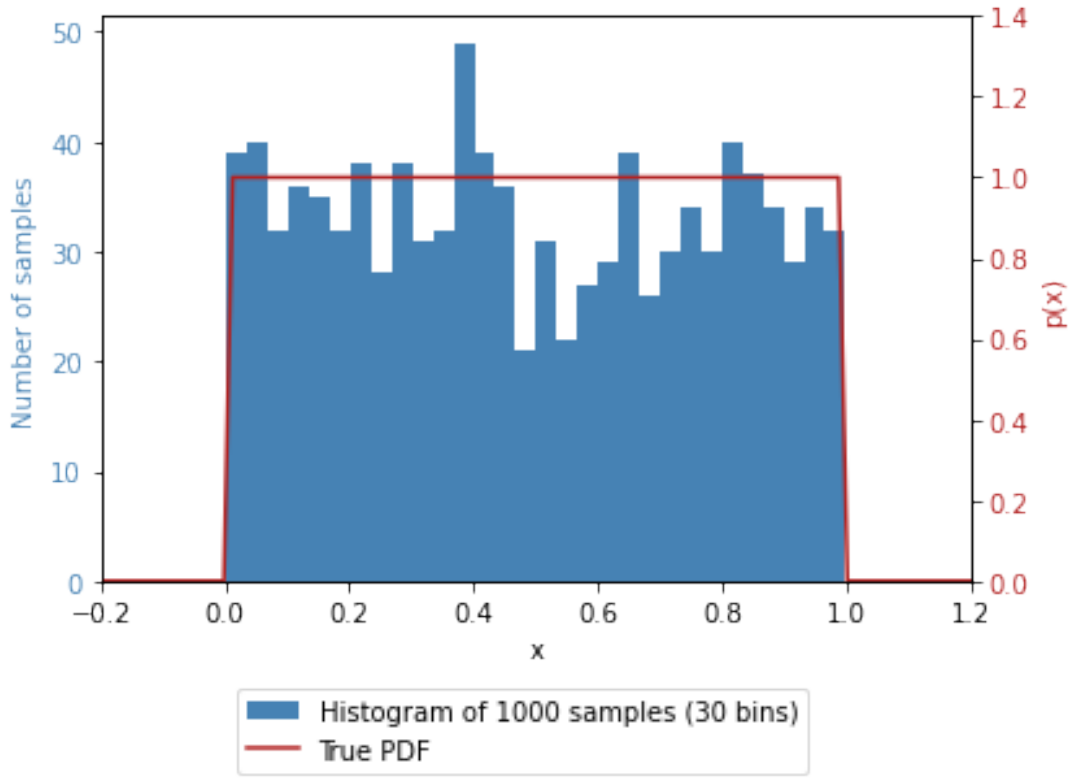
[167]:
```
fig, ax1 = plt.subplots()

ax1.hist(X_gaussian[:1000], bins=30, color=color1, label='Histogram of 1000␣
↪samples (30 bins)')
ax1.set_xlabel('x')
```

```
ax1.set_ylabel('Number of samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 100)


ax2 = ax1.twinx()
x = np.linspace(-5, 5, 100)
ax2.plot(x, gaussian_pdf(x), color=color2, label='True PDF')
ax2.set_ylabel('p(x)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 0.45)
ax2.set_xlim(-5, 5)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/gaussian_histogram_and_pdf.png', bbox_inches='tight')
```



[168]:
```
fig, ax1 = plt.subplots()

x = np.linspace(-0.2, 1.2, 100)
```

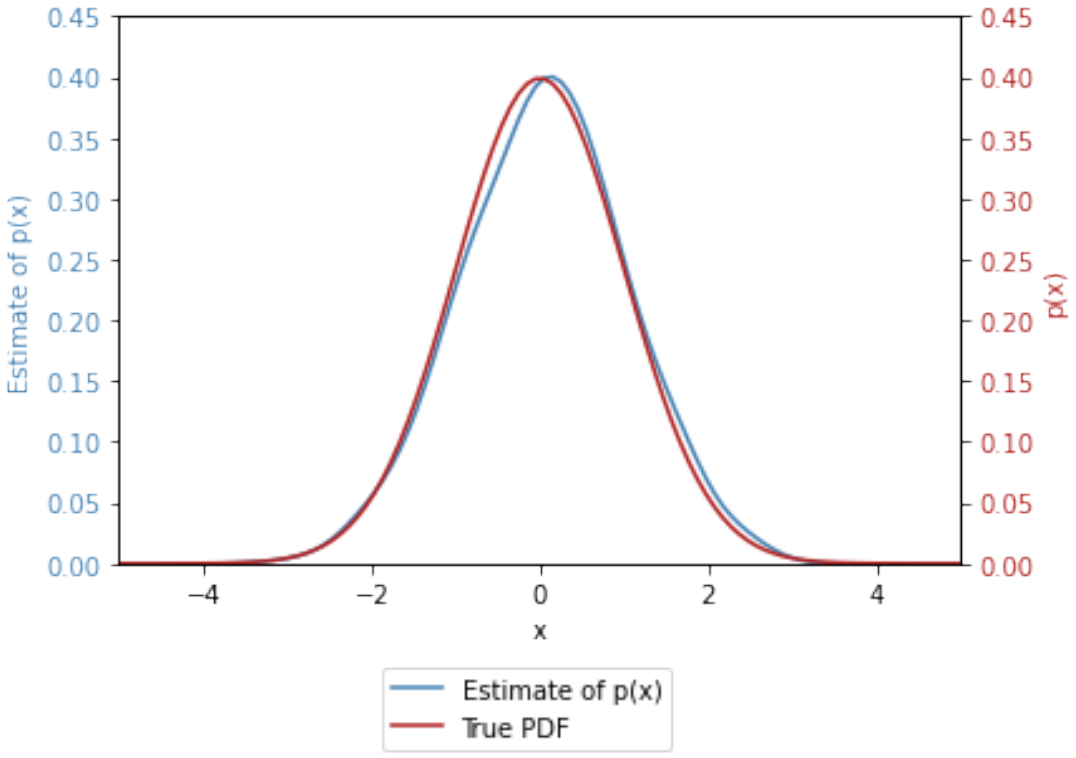
```
ax1.hist(X_uniform[:1000], bins=30, color=color1, label='Histogram of 1000␣
 ↪samples (30 bins)')
ax1.set_xlabel('x')
ax1.set_ylabel('Number of samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)

ax2 = ax1.twinx()
ax2.plot(x, uniform_pdf(x), color=color2, label='True PDF')
ax2.set_ylabel('p(x)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1.4)
ax2.set_xlim(-0.2, 1.2)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/uniform_histogram_and_pdf.png', bbox_inches='tight')
```



Kernel smoothing

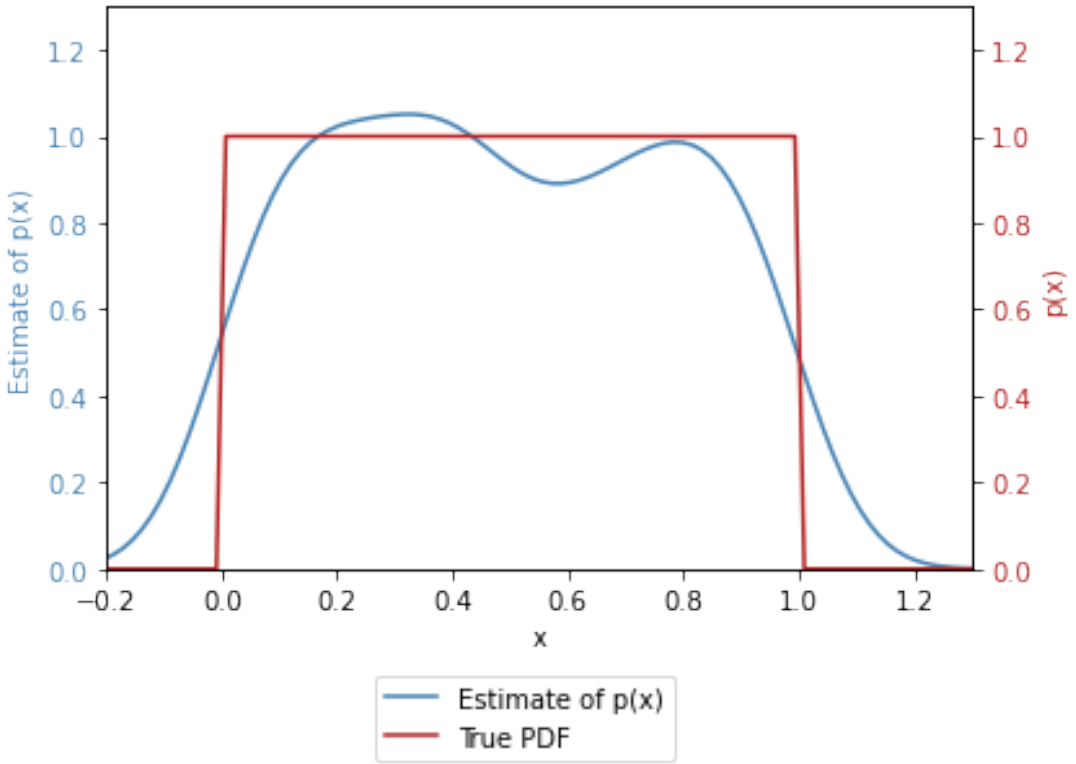
[169]:
```
fig, ax1 = plt.subplots()
```

```
x = np.linspace(-5, 5, 100)

ax1.plot(x, kernel_smoothed_density(x, X_gaussian[:1000]), color=color1,␣
 ↪label='Estimate of p(x)')
ax1.set_xlabel('x')
ax1.set_ylabel('Estimate of p(x)', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 0.45)

ax2 = ax1.twinx()
ax2.plot(x, gaussian_pdf(x), color=color2, label='True PDF')
ax2.set_ylabel('p(x)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 0.45)
ax2.set_xlim(-5, 5)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))
plt.savefig('figures/gaussian_kernel_smoothed.png', bbox_inches='tight')
```



[170]:
```
fig, ax1 = plt.subplots()

x = np.linspace(-0.3, 1.3, 100)
```

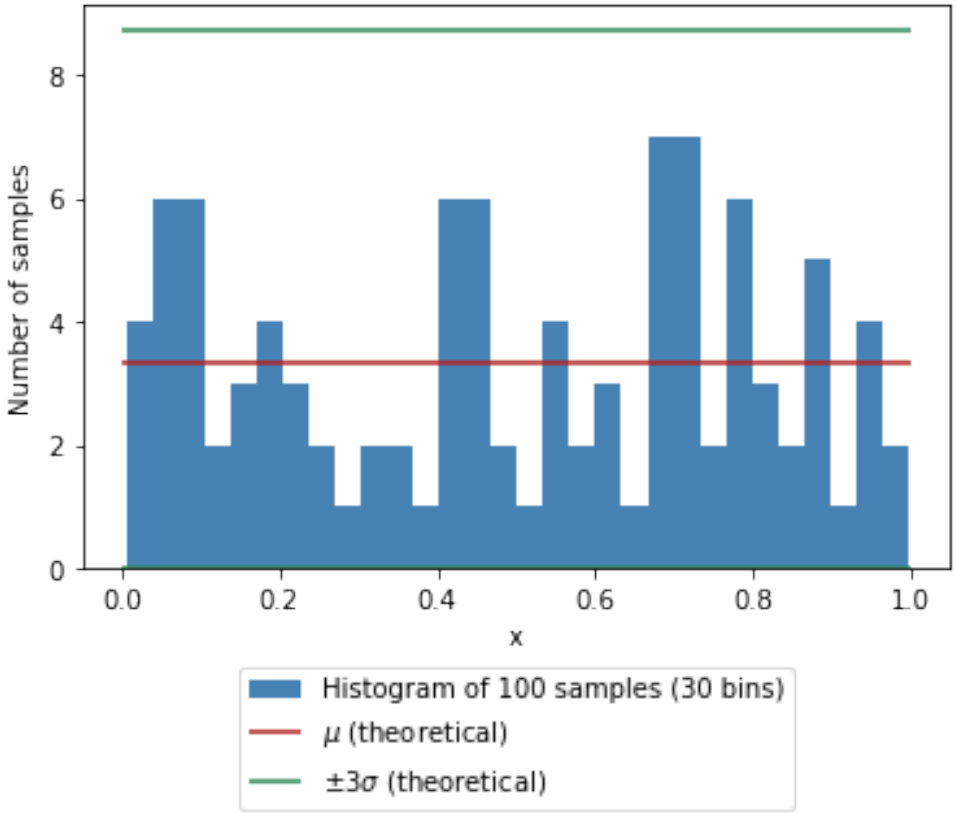
```
ax1.plot(x, kernel_smoothed_density(x, X_uniform[:1000], width=0.1),␣
 ↪color=color1, label='Estimate of p(x)')
ax1.set_xlabel('x')
ax1.set_ylabel('Estimate of p(x)', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 1.3)


ax2 = ax1.twinx()
ax2.plot(x, uniform_pdf(x), color=color2, label='True PDF')
ax2.set_ylabel('p(x)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1.3)
ax2.set_xlim(-0.2, 1.3)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))
plt.savefig('figures/uniform_kernel_smoothed.png', bbox_inches='tight')
```

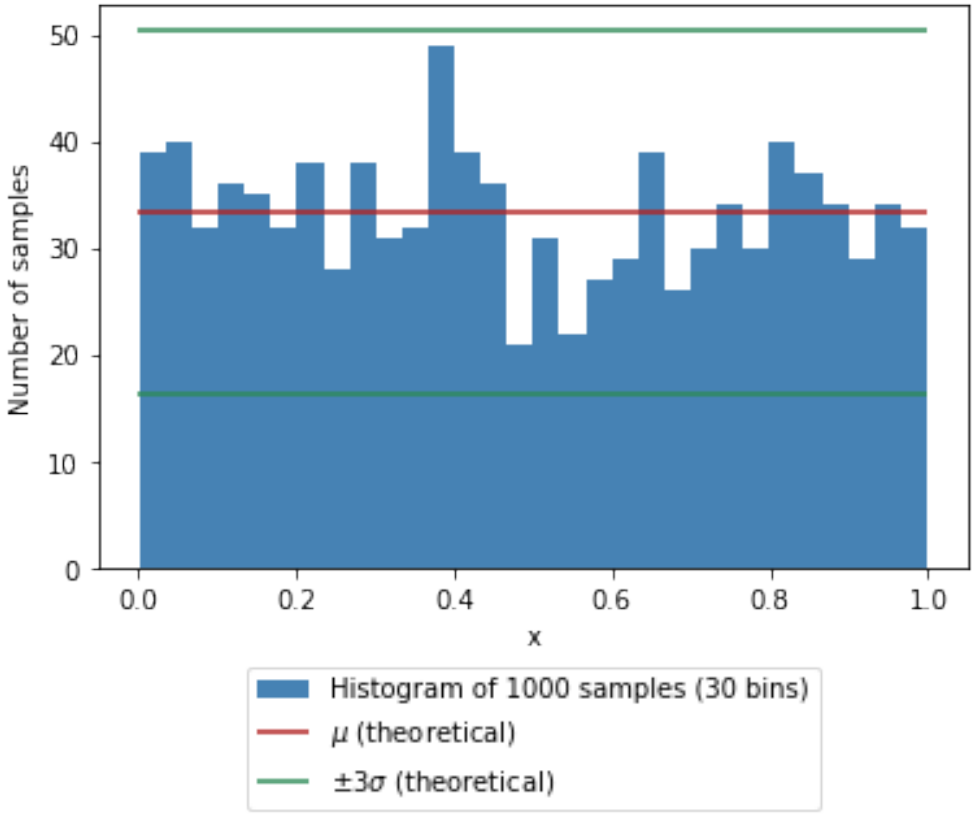


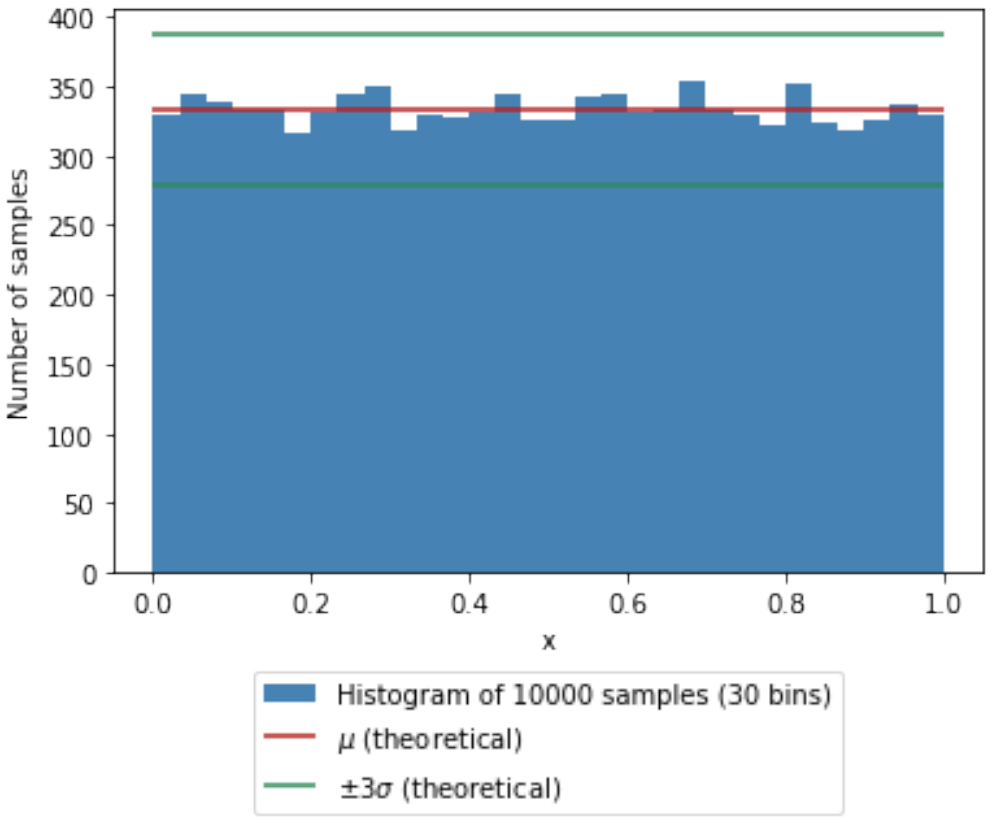
Multinomial theory: Uniform distribution

[171]:
```
def plot_uniform_histogram_mean_sd(N, nbins=30):
    plt.figure()

    plt.hist(X_uniform[:N], bins=nbins, color=color1, label=f'Histogram of {N}␣
↪samples ({nbins} bins)')

    bin_width = 1 / nbins
    mean = N * bin_width
    sd = np.sqrt(N * bin_width * (1 - bin_width))
    plt.hlines(y=mean, xmin=0, xmax=1, color=color2, label=r'$\mu$␣
↪(theoretical)')
    plt.hlines(y=[mean + 3 * sd, max(0, mean - 3 * sd)], xmin=0, xmax=1,␣
↪color=color3, label='$\pm 3 \sigma$ (theoretical)')

    plt.xlabel('x')
    plt.ylabel('Number of samples')
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15))
    plt.savefig(f'figures/uniform_histogram_{N}.png', bbox_inches='tight')
```

[172]:
```
plot_uniform_histogram_mean_sd(100)
plot_uniform_histogram_mean_sd(1000)
plot_uniform_histogram_mean_sd(10000)
```




Number of samples
x
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0
Histogram of 1000 samples (30 bins)
μ (theoretical)
$\pm 3\sigma$ (theoretical)

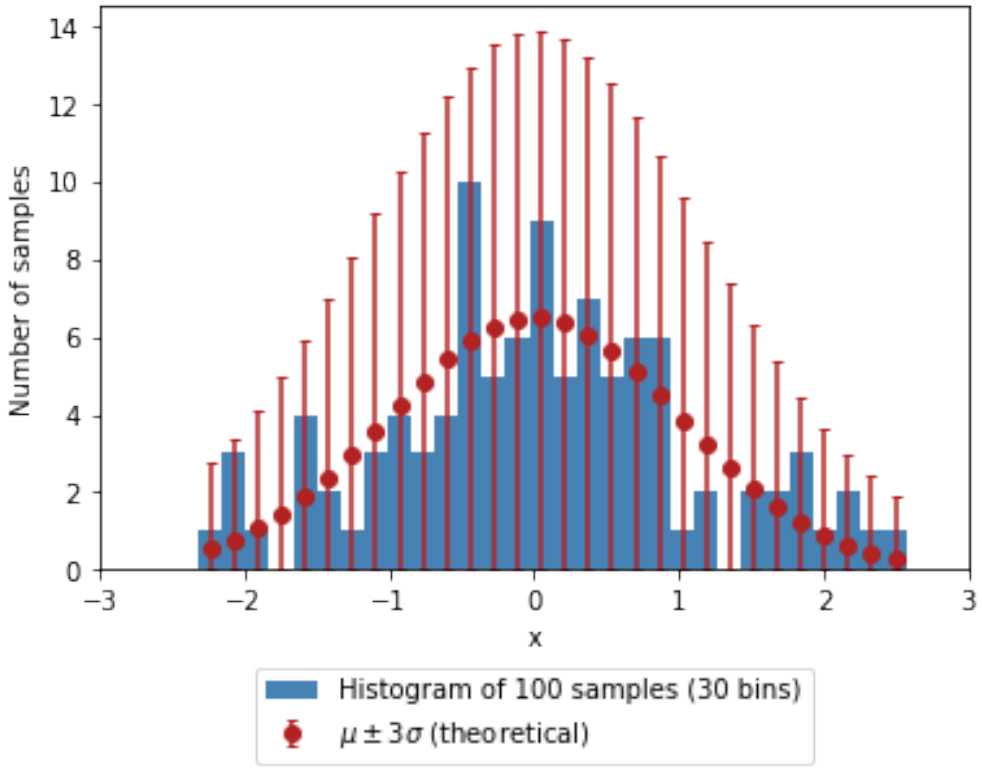


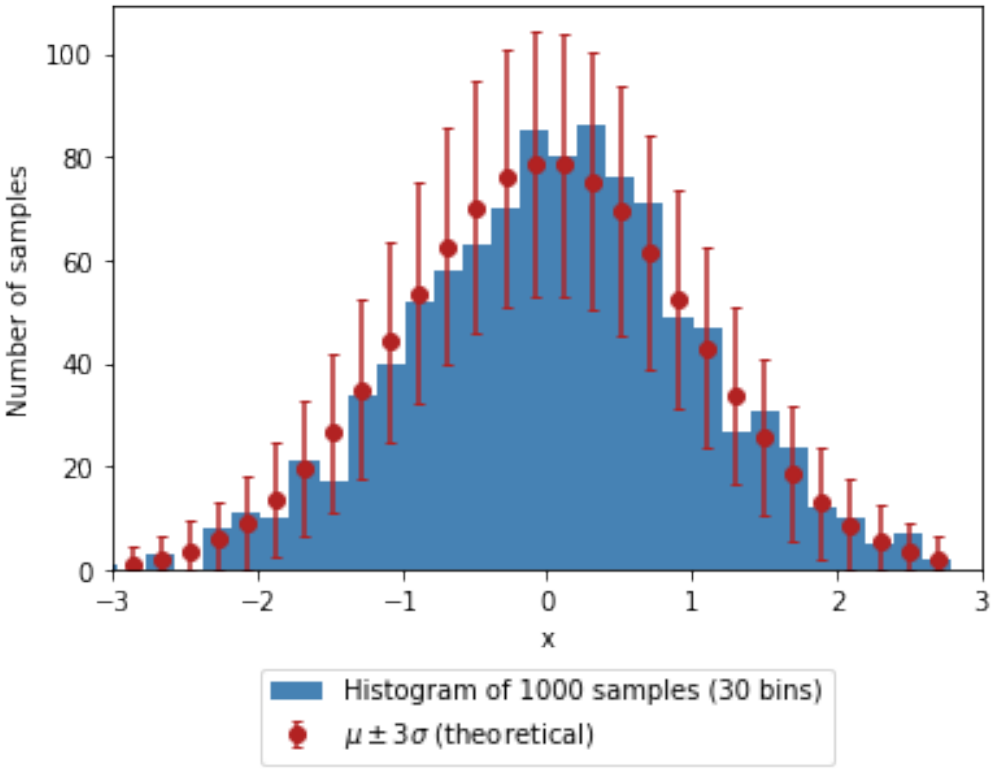
Multinomial theory: Gaussian distribution

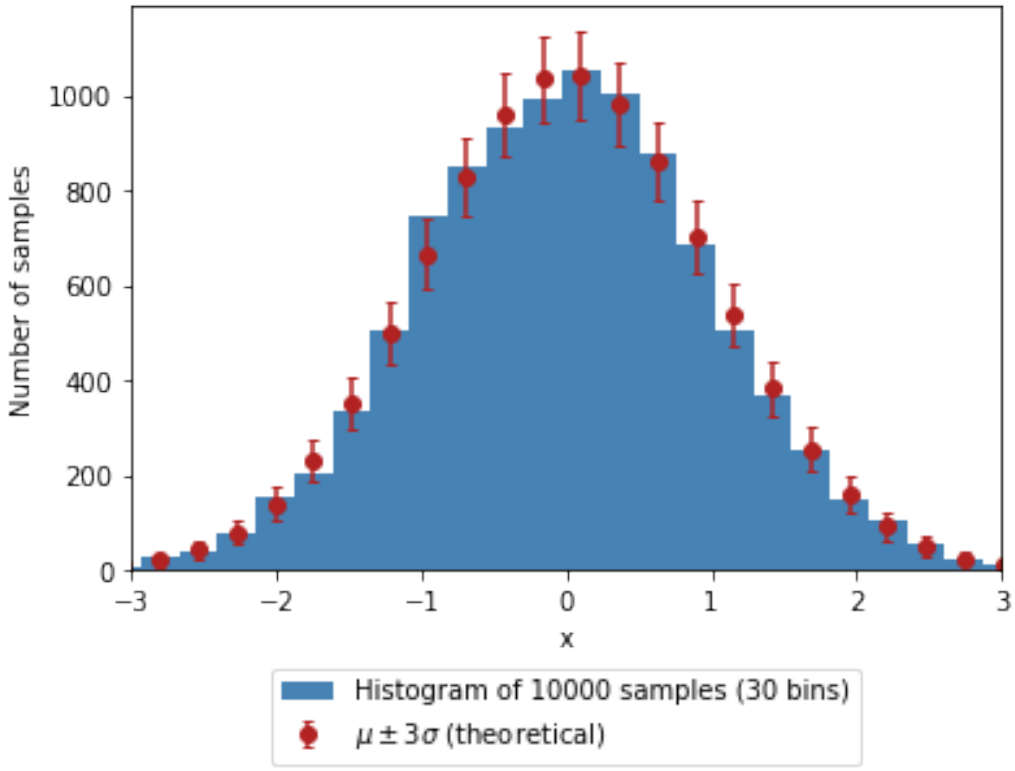
[173]:
```
def plot_gaussian_histogram_mean_sd(N, nbins=30):
    plt.figure()

    bin_counts, bin_edges, patches = plt.hist(X_gaussian[:N], bins=nbins,
                                              color=color1, label=f'Histogram␣
 ↪of {N} samples ({nbins} bins)')

    bin_centres = (bin_edges[:-1] + bin_edges[1:]) / 2
    bin_probabilities = norm.cdf(bin_edges[1:]) - norm.cdf(bin_edges[:-1])

    bin_means = N * bin_probabilities
    bin_sds = np.sqrt(N * bin_probabilities * (1 - bin_probabilities))
    upper_errorbar = 3 * bin_sds
    lower_errorbar = upper_errorbar
    lower_errorbar = np.minimum(3*bin_sds, bin_means)
    plt.errorbar(x=bin_centres, y=bin_means,
                 yerr=[lower_errorbar, upper_errorbar],
                 color=color2, fmt='o', capsize=2,
```

```
                label=r'$\mu \pm 3\sigma$ (theoretical)')

    plt.xlim(-3, 3)
    plt.xlabel('x')
    plt.ylabel('Number of samples')
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15))
    plt.savefig(f'figures/gaussian_histogram_{N}.png', bbox_inches='tight')
```

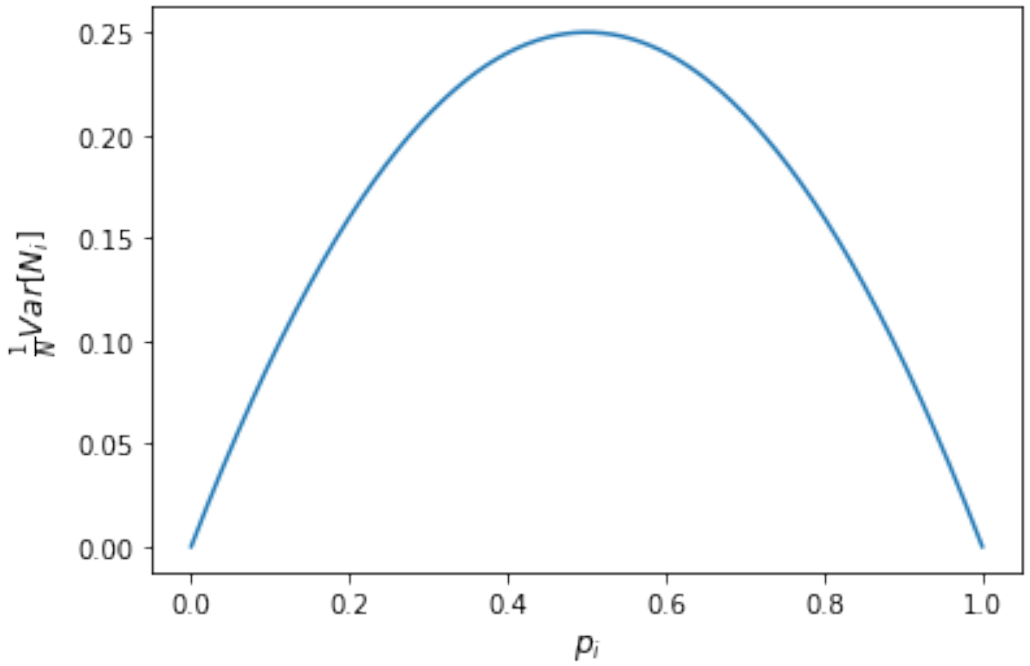
[174]:
```
plot_gaussian_histogram_mean_sd(100)
plot_gaussian_histogram_mean_sd(1000)
plot_gaussian_histogram_mean_sd(10000)
```




Number of samples
100
80
60
40
20
0
−3
−2
−1
0
1
2
3
x
Histogram of 1000 samples (30 bins)
$\mu \pm 3\sigma$ (theoretical)



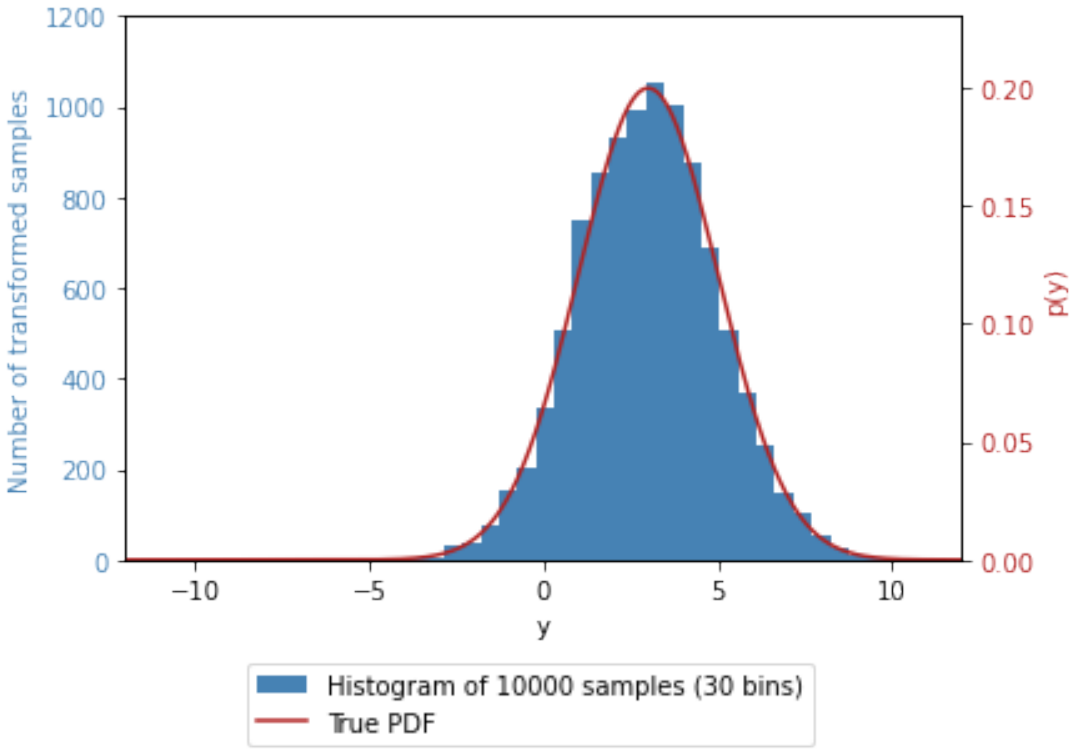
[175]:
```
plt.figure()
x = np.linspace(0, 1, 1000)
plt.plot(x, x*(1 - x))
plt.xlabel(r'$p_i$', fontsize=12)
plt.ylabel(r'$\frac{1}{N}Var[N_i]$', fontsize=12)
plt.savefig(f'figures/histogram_variance.png', bbox_inches='tight')
```



0.0.2 Section 2: Functions of Random Variables

f(x) = ax + b

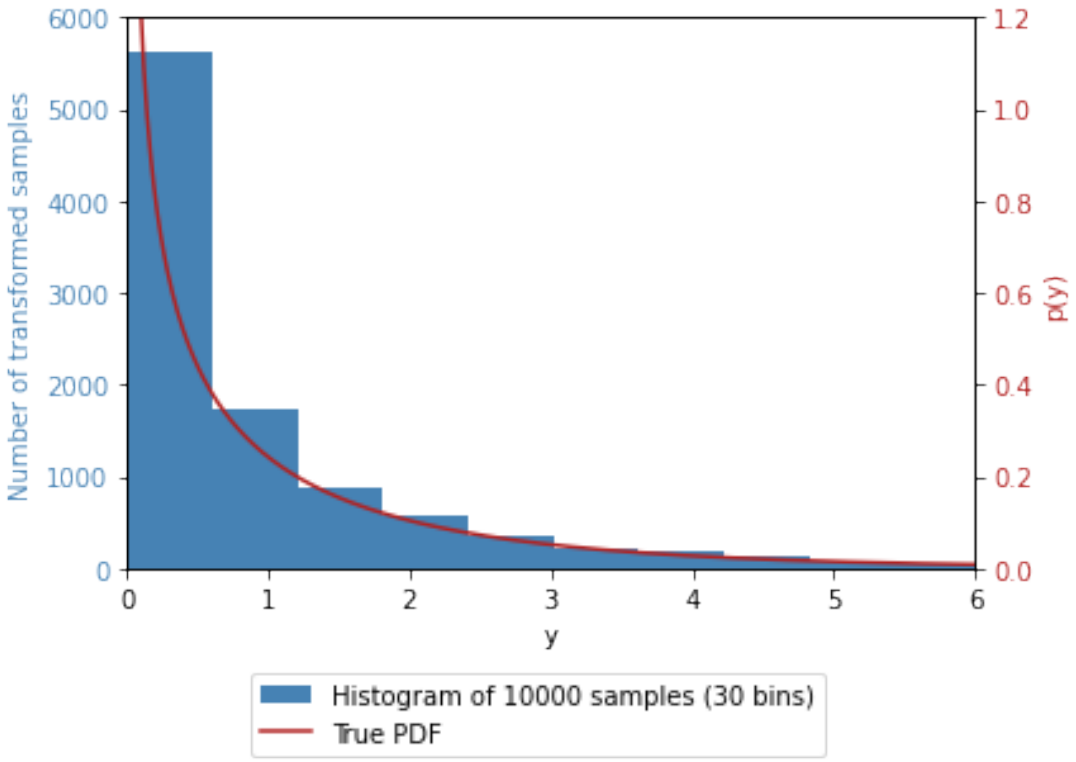
[176]:
```
fig, ax1 = plt.subplots()

Y = 2 * X_gaussian[:10000] + 3
ax1.hist(Y, bins=30, color=color1, label='Histogram of 10000 samples (30 bins)')
ax1.set_xlabel('y')
ax1.set_ylabel('Number of transformed samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 1200)


ax2 = ax1.twinx()
x = np.linspace(-12, 12, 1000)
ax2.plot(x, gaussian_pdf(x, mu=3, sigma=2), color=color2, label='True PDF')
ax2.set_ylabel('p(y)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 0.23)
ax2.set_xlim(-12, 12)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/linear_function_of_gaussian.png', bbox_inches='tight')
```



0.0.3 f(x) = x^2

[177]:
```
fig, ax1 = plt.subplots()

Y = X_gaussian[:10000] ** 2
ax1.hist(Y, bins=30, color=color1, label='Histogram of 10000 samples (30 bins)')
ax1.set_xlabel('y')
ax1.set_ylabel('Number of transformed samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 6000)


ax2 = ax1.twinx()
x = np.linspace(0.01, 12, 1000)
pdf = np.exp(-0.5*x) / np.sqrt(2*np.pi*x)
ax2.plot(x, pdf, color=color2, label='True PDF')
ax2.set_ylabel('p(y)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1.2)
ax2.set_xlim(0, 6)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))
```

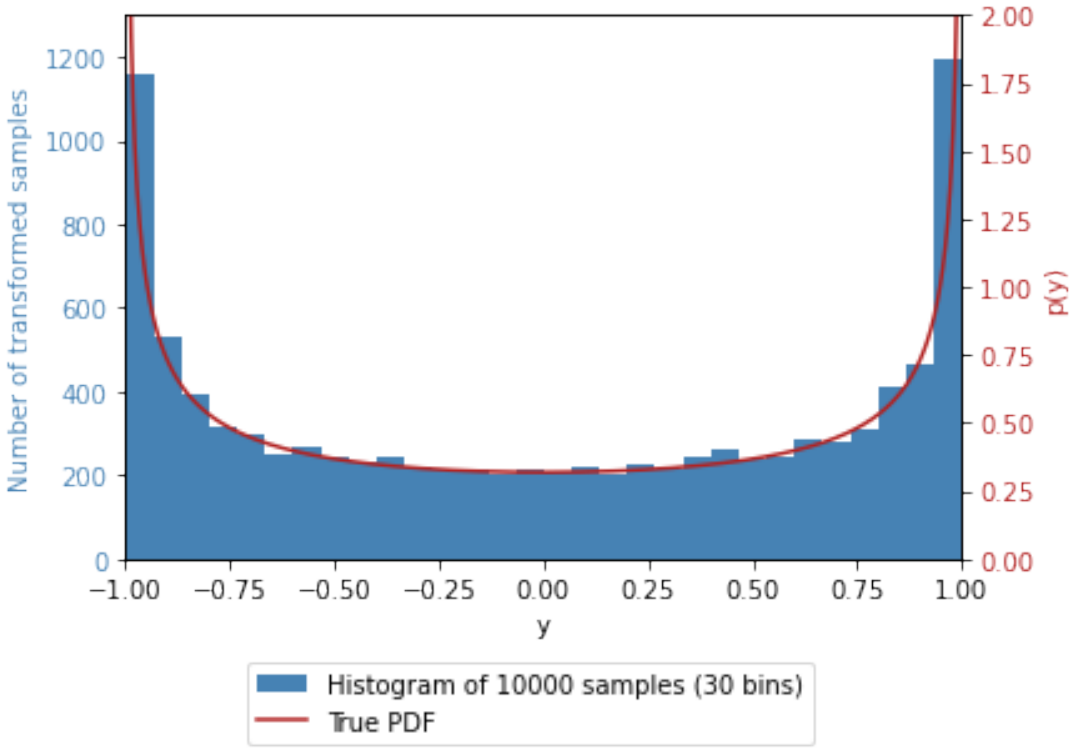
```
plt.savefig('figures/quadratic_function_of_gaussian.png', bbox_inches='tight')
```



0.0.4 f(x) = sin(x)

[178]:
```
fig, ax1 = plt.subplots()

Y = np.sin(X_uniform[:10000]*2*np.pi)
ax1.hist(Y, bins=30, color=color1, label='Histogram of 10000 samples (30 bins)')
ax1.set_xlabel('y')
ax1.set_ylabel('Number of transformed samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 1300)


ax2 = ax1.twinx()
y = np.linspace(-0.99, 0.99, 1000)
pdf = 1 / (np.pi * np.sqrt(1 - y**2))
ax2.plot(y, pdf, color=color2, label='True PDF')
ax2.set_ylabel('p(y)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
```

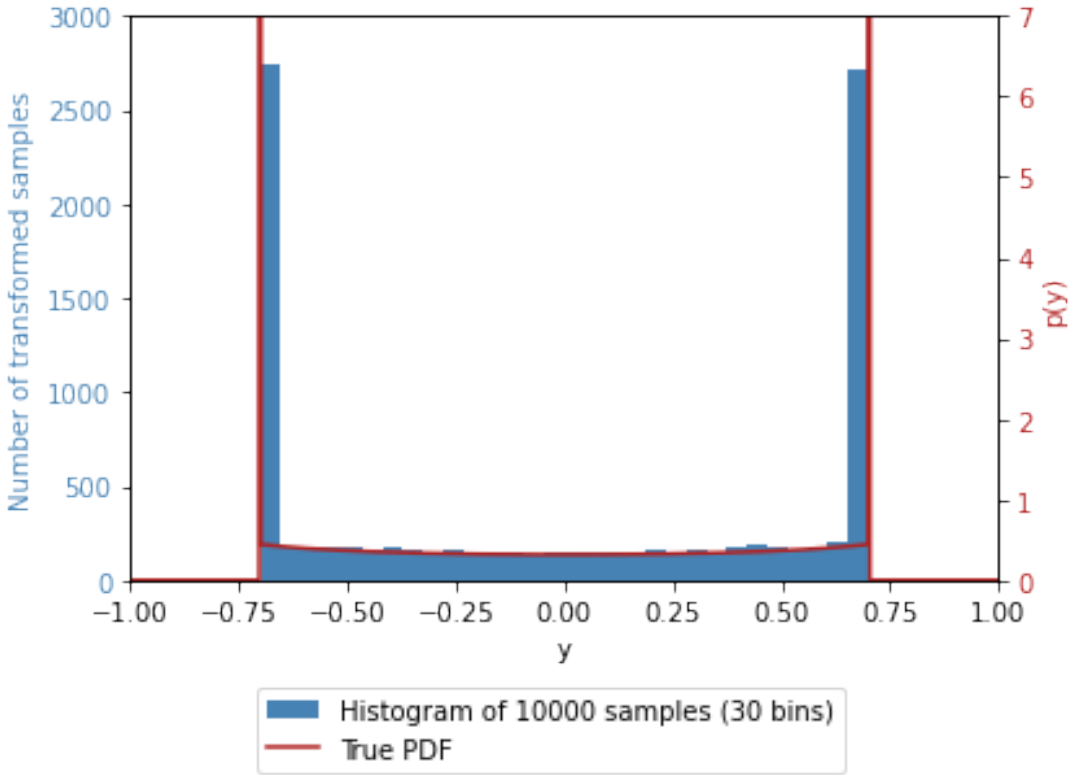
```
ax2.set_ylim(0, 2)
ax2.set_xlim(-1, 1)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/sinusoidal_function_of_uniform.png', bbox_inches='tight')
```



0.0.5 f(x) = limited(sin(x))

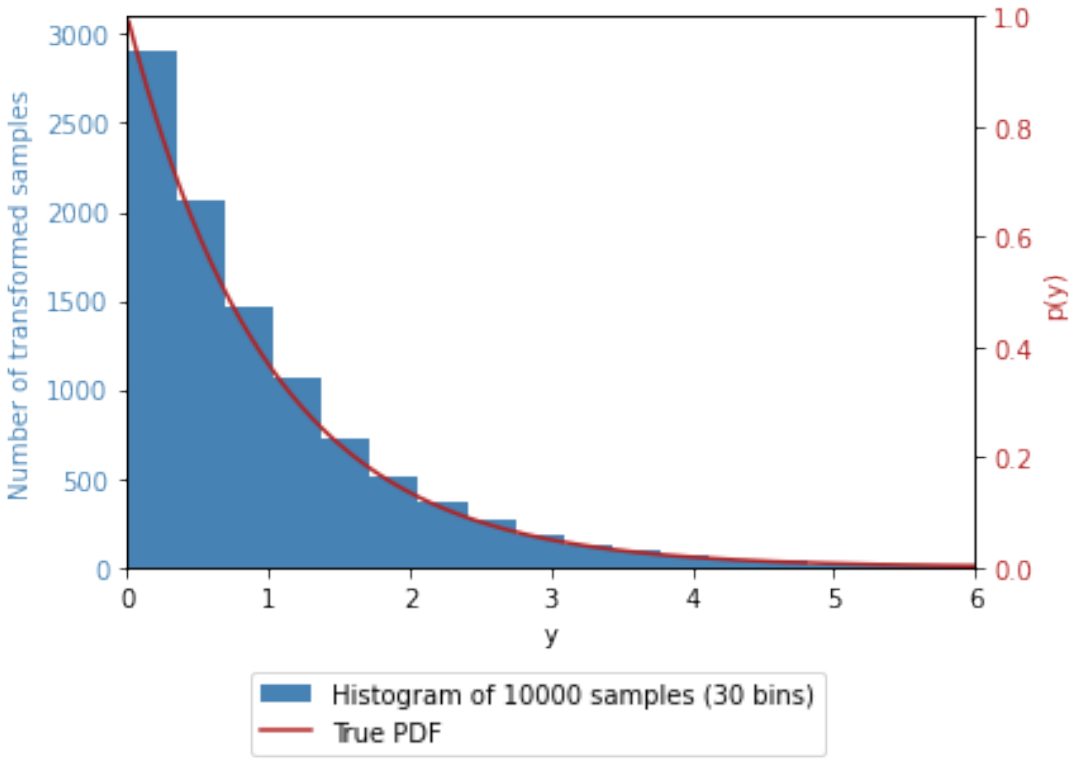
[179]:
```
fig, ax1 = plt.subplots()

Y = np.clip(np.sin(X_uniform[:10000]*2*np.pi), -0.7, 0.7)
ax1.hist(Y, bins=30, color=color1, label='Histogram of 10000 samples (30 bins)')
ax1.set_xlabel('y')
ax1.set_ylabel('Number of transformed samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 3000)


ax2 = ax1.twinx()
y = np.linspace(-1, 1, 1000)
```

```
def pdf(y):
    return 1 / (np.pi * np.sqrt(1 - y**2))

ax2.plot(y,
         np.piecewise(y,
                      [np.abs(y) > 0.7,
                       np.abs(np.abs(y) - 0.7) < 0.002,
                       np.abs(y) < 0.7],
                      [0,
                       9e99,
                       pdf]),
         color=color2, label='True PDF')
ax2.set_ylabel('p(y)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 7)
ax2.set_xlim(-1, 1)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/limited_sinusoidal_function_of_uniform.png',␣
 ↪bbox_inches='tight')
```



0.0.6 Section 3: iCDF method

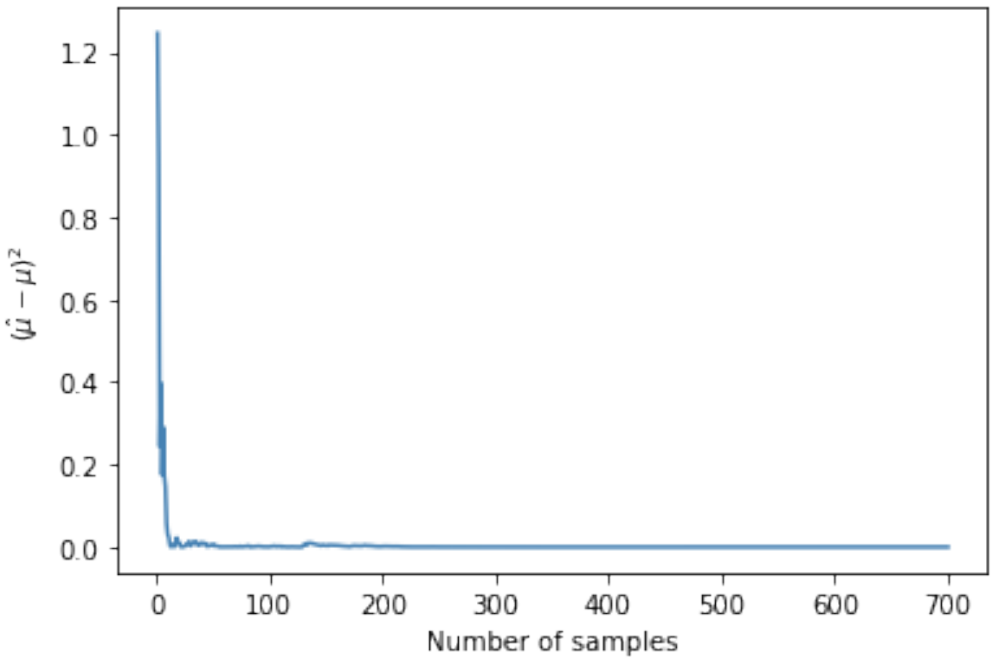
[180]:
```
fig, ax1 = plt.subplots()

Y = -np.log(1 - X_uniform)

ax1.hist(Y, bins=30, color=color1, label=f'Histogram of {len(Y)} samples (30␣
↪bins)')
ax1.set_xlabel('y')
ax1.set_ylabel('Number of transformed samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)
ax1.set_ylim(0, 3100)


ax2 = ax1.twinx()
x = np.linspace(0.01, 12, 1000)
pdf = np.exp(-x)
ax2.plot(x, pdf, color=color2, label='True PDF')
ax2.set_ylabel('p(y)', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1)
ax2.set_xlim(0, 6)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))

plt.savefig('figures/icdf_exponential.png', bbox_inches='tight')
```

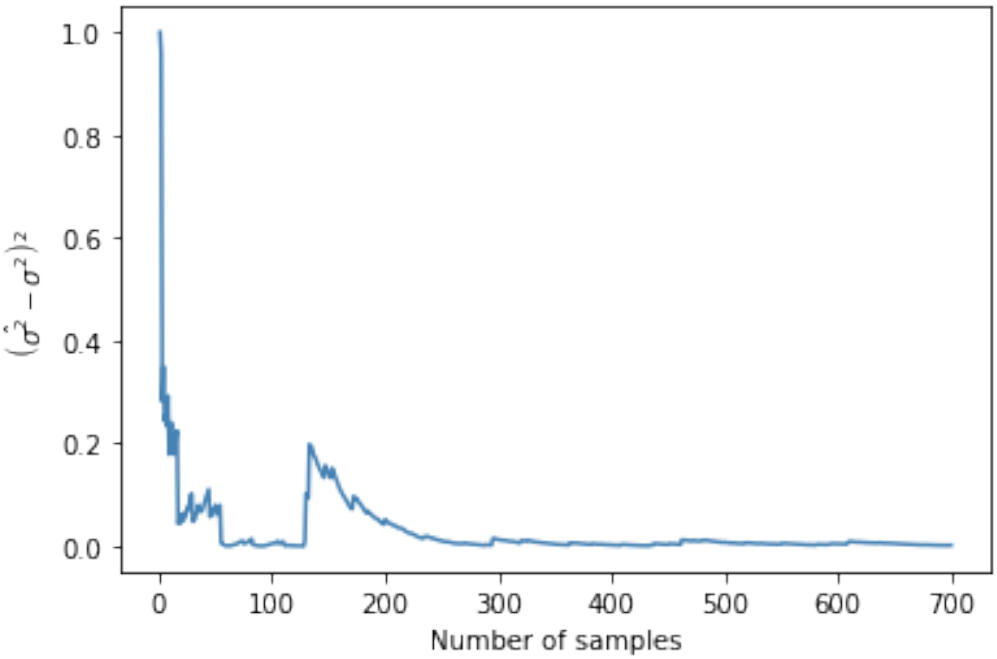


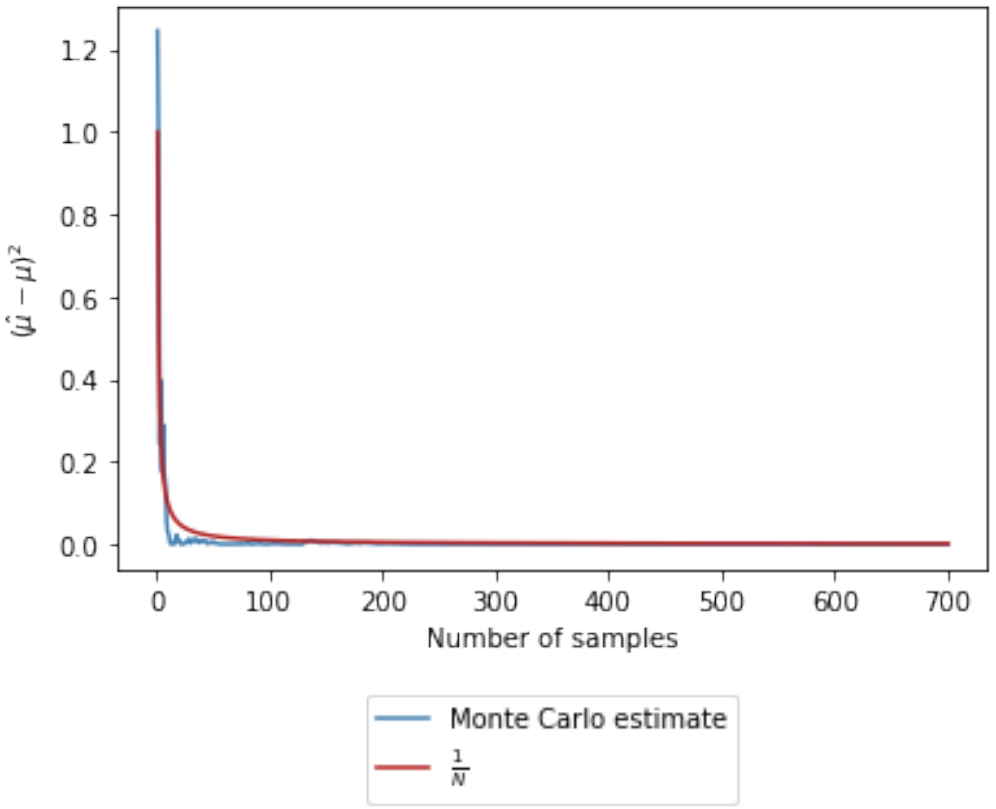
[181]:
```python
def estimate_mean(N):
    return np.mean(Y[:N])

def estimate_variance(N):
    return np.mean(Y[:N] * Y[:N]) - estimate_mean(N)**2

print(f"Mean: {estimate_mean(10000)}")
print(f"Variance: {estimate_variance(10000)}")

x = np.linspace(1, 700, 700)

plt.figure()
plt.plot(x, [(estimate_mean(int(n)) - 1)**2 for n in x], color=color1)
# plt.plot(x, 1/x)
plt.xlabel("Number of samples")
plt.ylabel(r'$(\hat{\mu} - \mu)^2$')
plt.savefig("figures/monte_carlo_mean.png")

plt.figure()
plt.plot(x, [(estimate_variance(int(n)) - 1)**2 for n in x], color=color1)
plt.xlabel("Number of samples")
plt.ylabel(r'$\left(\hat{\sigma^2} - \sigma^2\right)^2$')
```

```
plt.savefig("figures/monte_carlo_variance.png")

plt.figure()
plt.plot(x, [(estimate_mean(int(n)) - 1)**2 for n in x], color=color1,␣
 ↪label='Monte Carlo estimate')
plt.plot(x, 1/x, color=color2, label=r'$\frac{1}{N}$')
plt.xlabel("Number of samples")
plt.ylabel(r'$(\hat{\mu} - \mu)^2$')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.2))
plt.savefig("figures/monte_carlo_mean_best_fit.png")
```

```
Mean: 0.9944135469954328
Variance: 0.9748765893086265
```




1.0
0.8
0.6
0.4
0.2
0.0
$(\hat{\sigma}^2 - \sigma^2)^2$
0
100
200
300
400
500
600
700
Number of samples

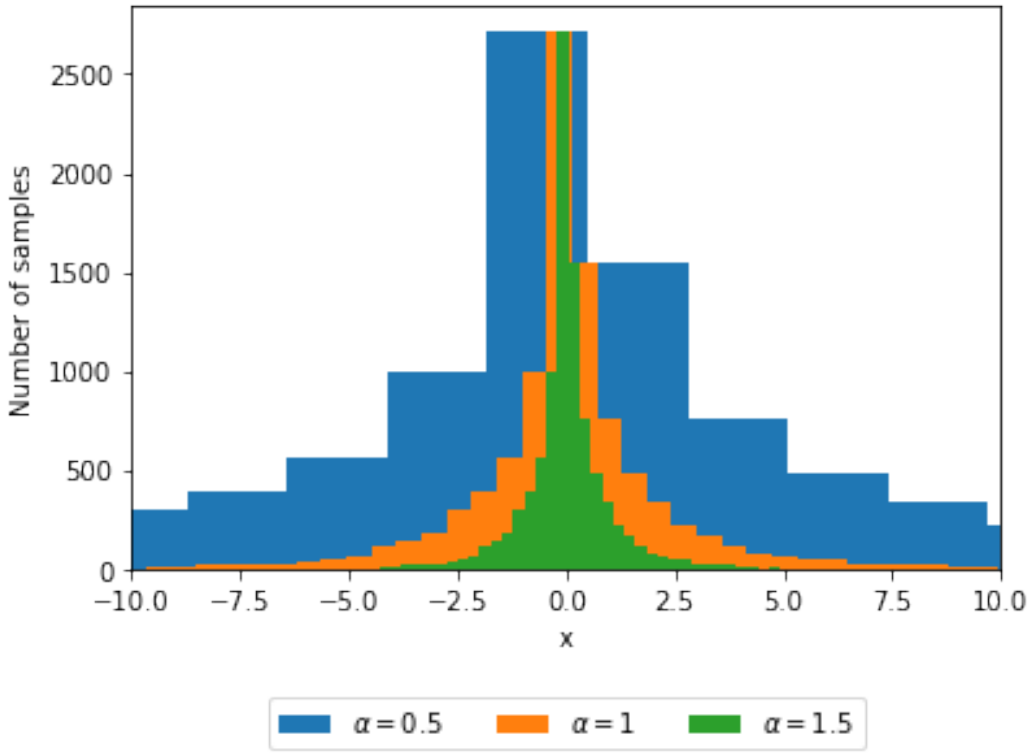

1.2
1.0
0.8
0.6
0.4
0.2
0.0
$(\hat{\mu} - \mu)^2$
0
100
200
300
400
500
600
700
Number of samples
Monte Carlo estimate
$\frac{1}{N}$

0.0.7 Section 4: Scaled mixture of Gaussians

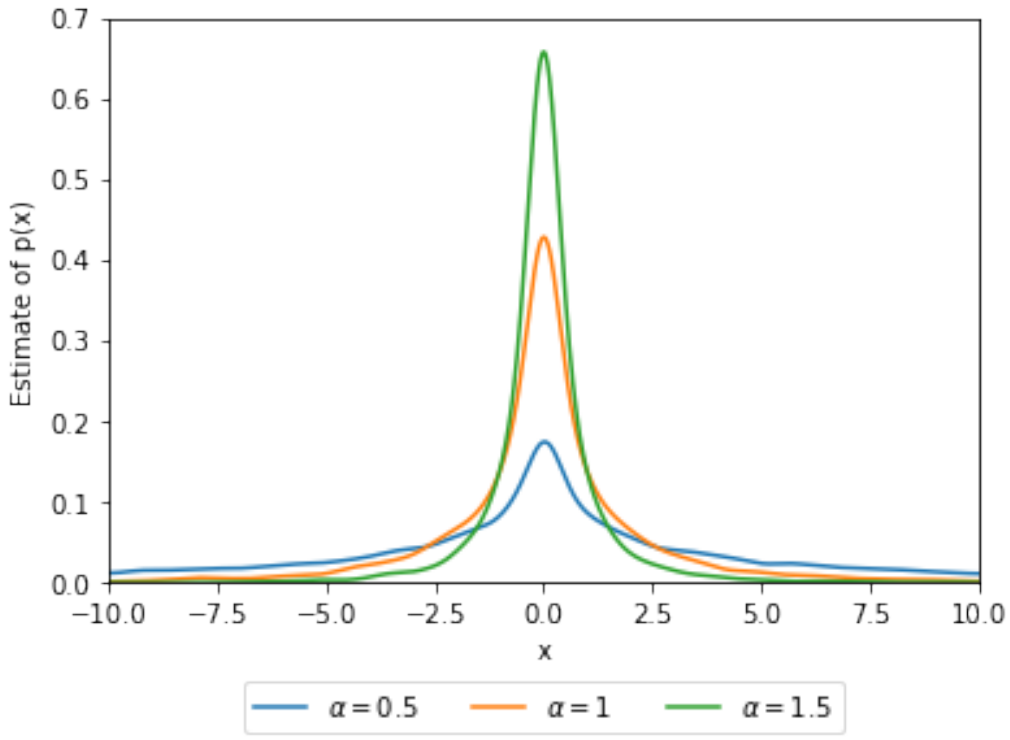
Exponential sampling

[182]:
```
def exponential_sampled_gaussian(N, alpha=1.5):
    exponential_samples = (-2 / (alpha ** 2)) * np.log(1 - X_uniform[:N])
    return X_gaussian[:N] * exponential_samples
```

[183]:
```
plt.figure()
N = 100000
for alpha in [0.5, 1, 1.5]:
    plt.hist(exponential_sampled_gaussian(N, alpha), bins=100, label=r'$\alpha␣
 ↪= {}$'.format(alpha))

plt.xlabel('x')
plt.ylabel('Number of samples')
plt.xlim(-10, 10)
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.2), ncol=3)
plt.savefig(f'figures/exponential_sampled_gaussian.png', bbox_inches='tight')
```

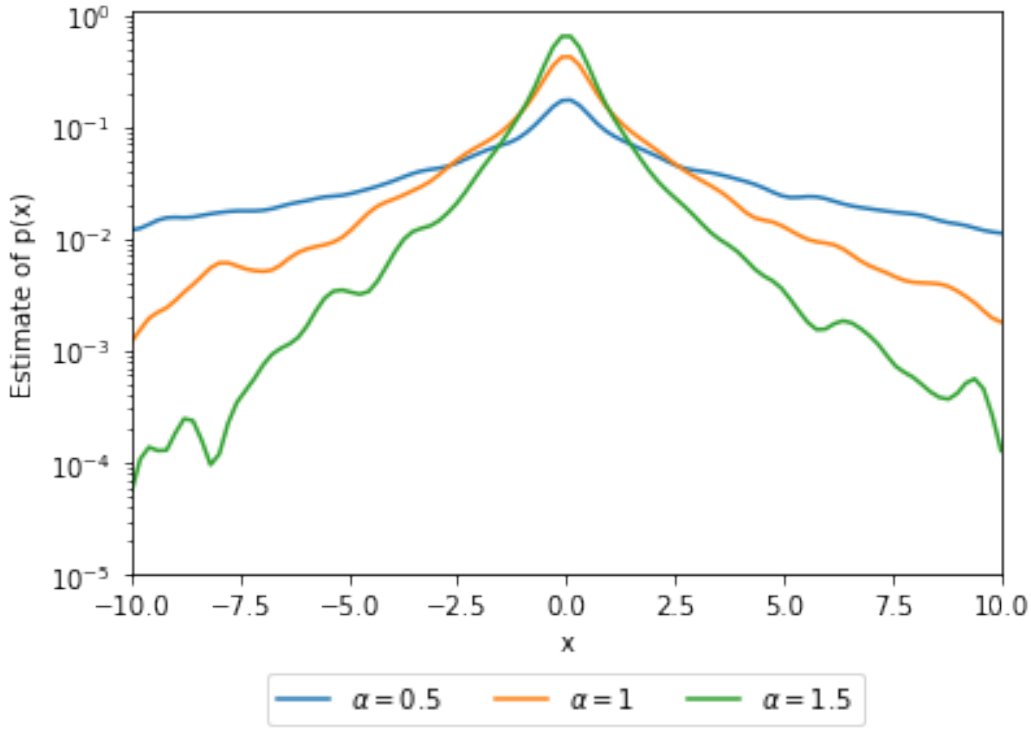


[184]:
```
plt.figure()

N = 100000
x = np.linspace(-10, 10, 1000)

for alpha in [0.5, 1, 1.5]:
    plt.plot(x, kernel_smoothed_density(x, exponential_sampled_gaussian(N,␣
 ↪alpha)), label=r'$\alpha = {}$'.format(alpha))

plt.xlabel('x')
plt.ylabel('Estimate of p(x)')
plt.xlim(-10, 10)
plt.ylim(0, 0.7)
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=3)
plt.savefig(f'figures/exponential_sampled_gaussian_kernel_smoothed.png',␣
 ↪bbox_inches='tight')
```



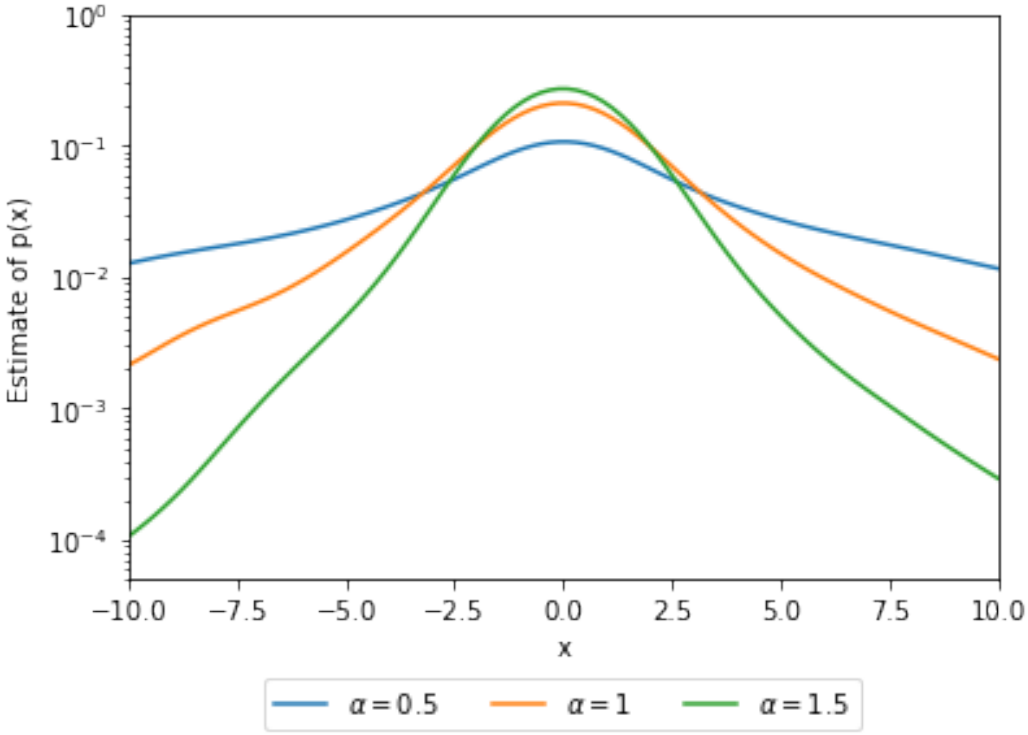
[185]:
```
plt.figure()

N = 100000
x = np.linspace(-10, 10, 100)
```

```
for alpha in [0.5, 1, 1.5]:
    plt.semilogy(x, kernel_smoothed_density(x, exponential_sampled_gaussian(N,␣
 ↪alpha)), label=r'$\alpha = {}$'.format(alpha))

plt.xlabel('x')
plt.ylabel('Estimate of p(x)')
plt.xlim(-10, 10)
plt.ylim(1e-5, 1.1)
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=3)
plt.savefig(f'figures/exponential_sampled_gaussian_kernel_smoothed_log_narrow.
 ↪png', bbox_inches='tight')
```



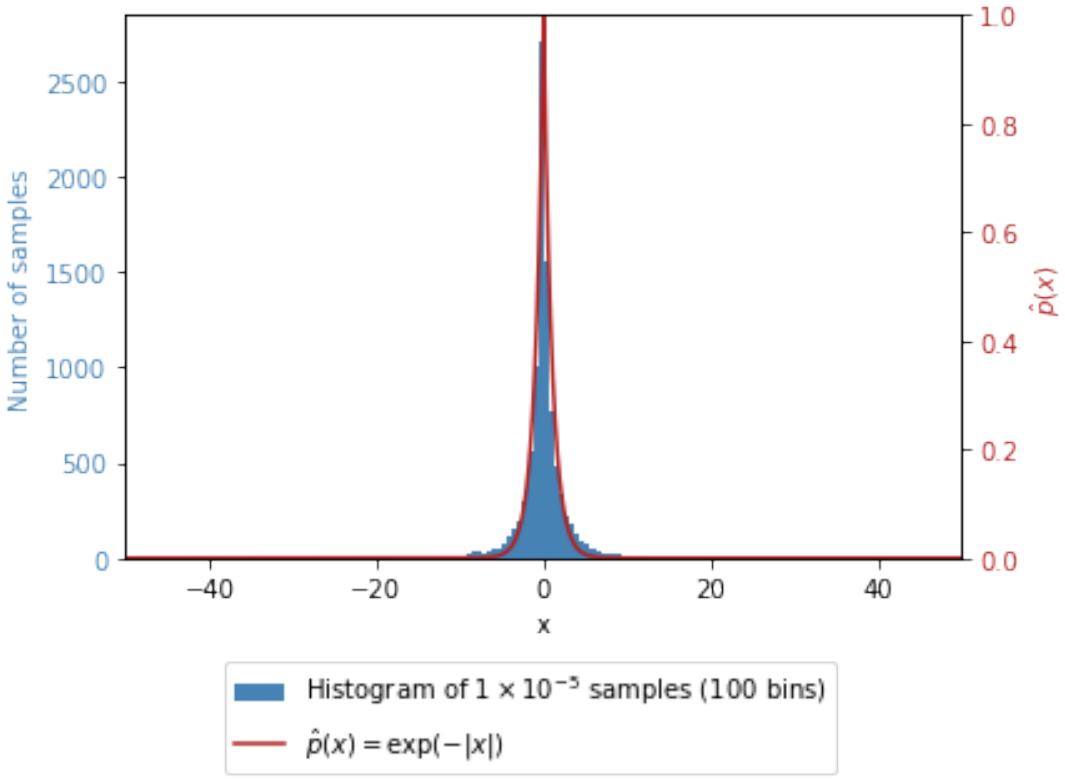
[186]:
```
plt.figure()

N = 100000
x = np.linspace(-10, 10, 100)

for alpha in [0.5, 1, 1.5]:
    plt.semilogy(x,
                 kernel_smoothed_density(x, exponential_sampled_gaussian(N,␣
 ↪alpha),
                                         width=1.2),
```

```
                label=r'$\alpha = {}$'.format(alpha))

plt.xlabel('x')
plt.ylabel('Estimate of p(x)')
plt.xlim(-10, 10)
plt.ylim(.5e-4, 1)
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=3)
plt.savefig(f'figures/exponential_sampled_gaussian_kernel_smoothed_log_wide.
 ↪png', bbox_inches='tight')
```



[187]:
```
fig, ax1 = plt.subplots()

N = 100000
ax1.hist(exponential_sampled_gaussian(N, 1),
         bins=100,
         label=r'Histogram of $1 \times 10^{-5}$ samples (100 bins)',
         color=color1)
ax1.set_xlabel('x')
ax1.set_ylabel('Number of samples', color=color1)
ax1.tick_params(axis='y', labelcolor=color1)

ax2 = ax1.twinx()
```

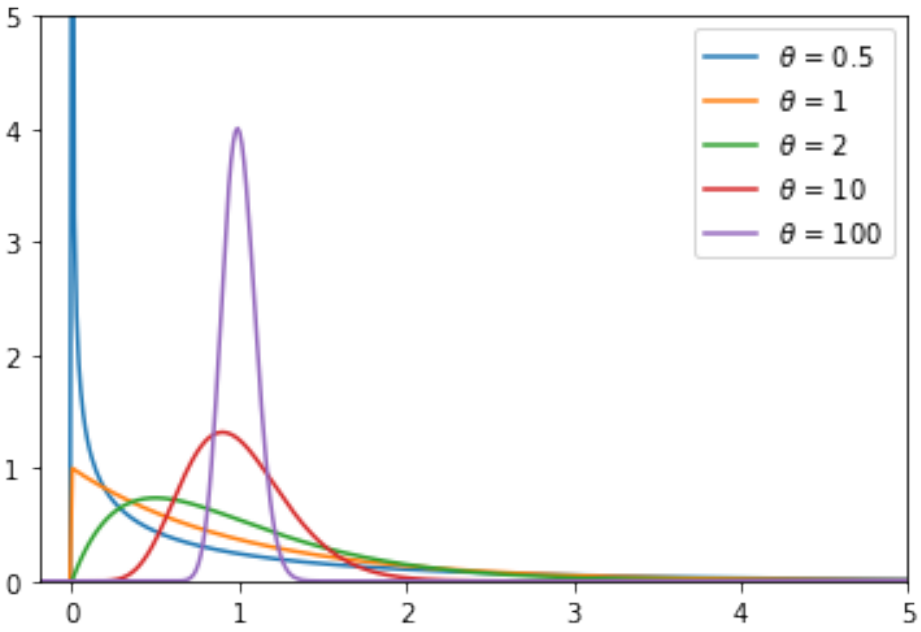
```
x = np.linspace(-50, 50, 10000)
ax2.plot(x, np.exp(-np.abs(x)), label=r'$\hat{p}(x) = \exp(-|x|)$',␣
↪color=color2)
ax2.set_ylabel(r'$\hat{p}(x)$', color=color2)
ax2.tick_params(axis='y', labelcolor=color2)
ax2.set_ylim(0, 1)
ax2.set_xlim(-50, 50)

fig.legend(loc='upper center', bbox_to_anchor=(0.5, 0))
plt.savefig(f'figures/exponential_sampled_gaussian_with_pdf.png',␣
↪bbox_inches='tight')
```



Gamma sampling

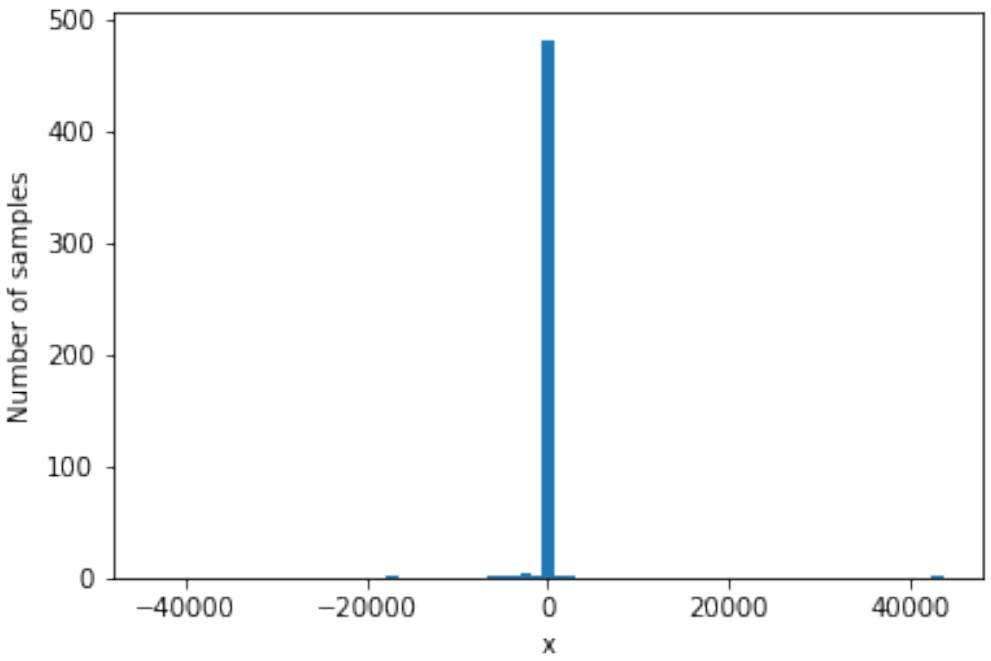
[188]:
```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

x = np.linspace(-1, 12, 1000)

plt.figure()
```

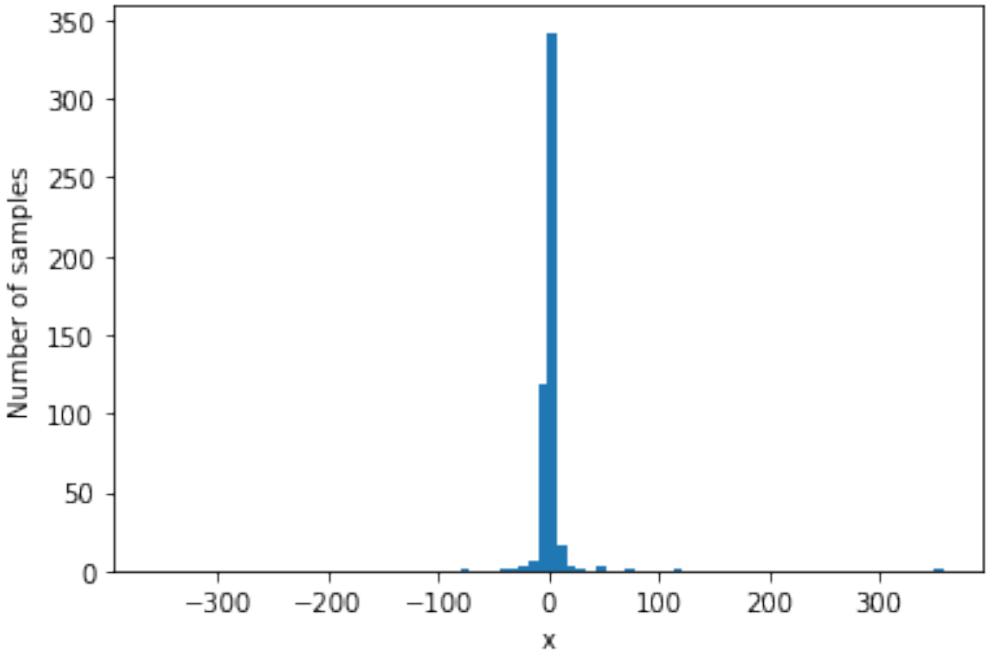
```
for t in [0.5,1,2,10,100]:
    plt.plot(x, stats.gamma.pdf(x, t, scale=1/t), label=r'$\theta$ = {}'.
 ↪format(t))
plt.xlim(-0.2,5)
plt.ylim(0,5)
plt.legend()
plt.savefig(f'figures/gamma_theta.png', bbox_inches='tight')
```

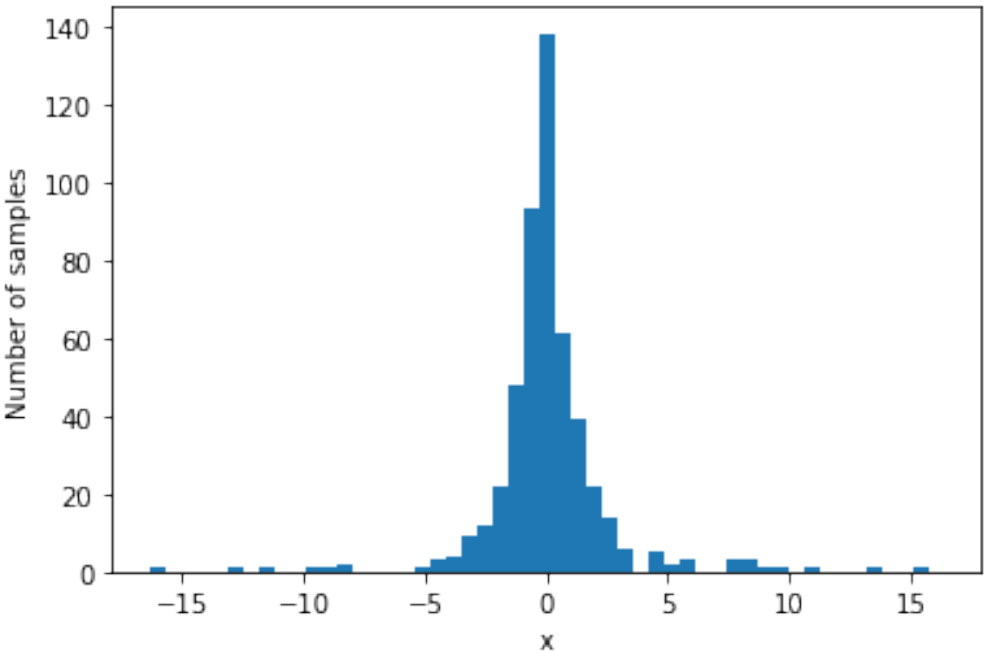


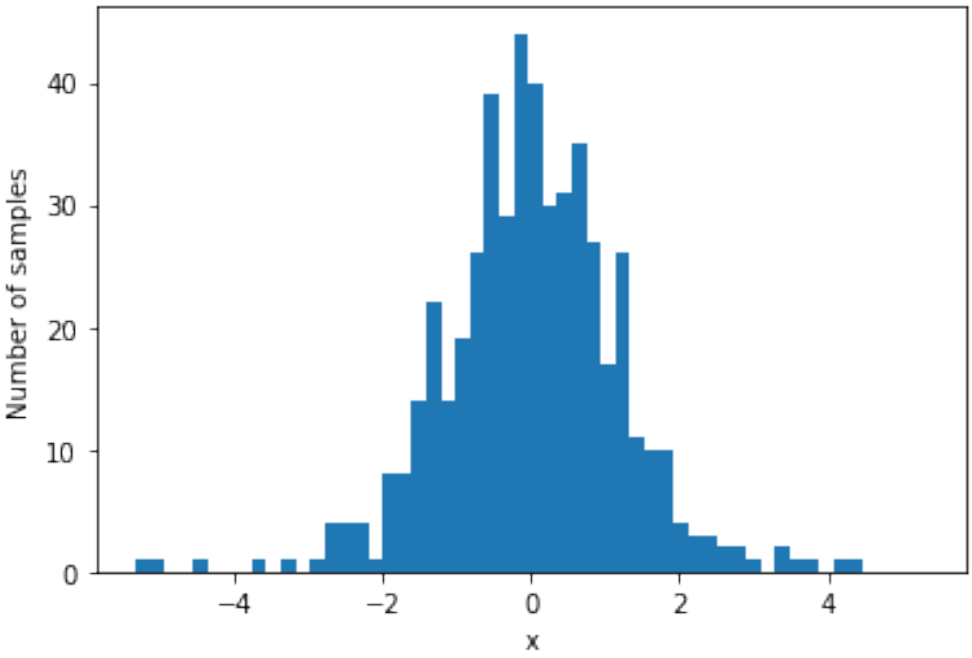
[189]:
```
def gamma_sampled_gaussian(N, theta):
    v = stats.gamma.rvs(a=theta, scale=1/theta, size=N)
    u = 1 / v
    return np.random.normal(loc=0, scale=u)

N=500
nbins=50

samples = {theta: gamma_sampled_gaussian(N, theta) for theta in [0.5, 1, 2, 10,␣
 ↪100]}

for theta, s in samples.items():
    plt.figure()
    values, bins, patches = plt.hist(s, bins=nbins)
    xlim = 1.1*max(abs(min(bins)), abs(max(bins)))
    plt.xlim(-xlim, xlim)
    plt.xlabel('x')
```

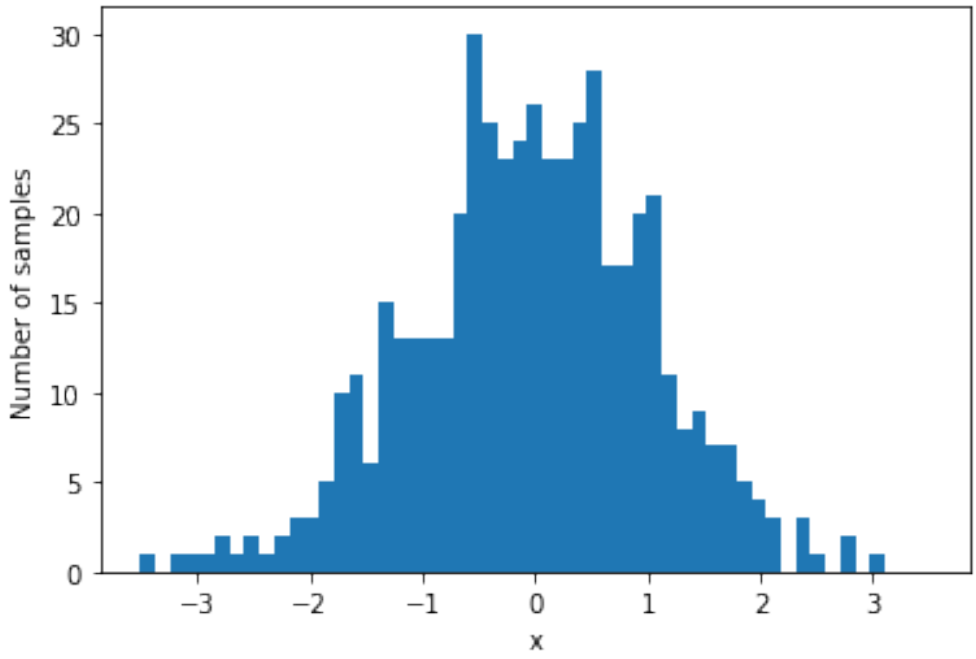
```
    plt.ylabel('Number of samples')
    plt.savefig(f"figures/gamma_sampled_gaussian_histogram_{theta}.png")
```



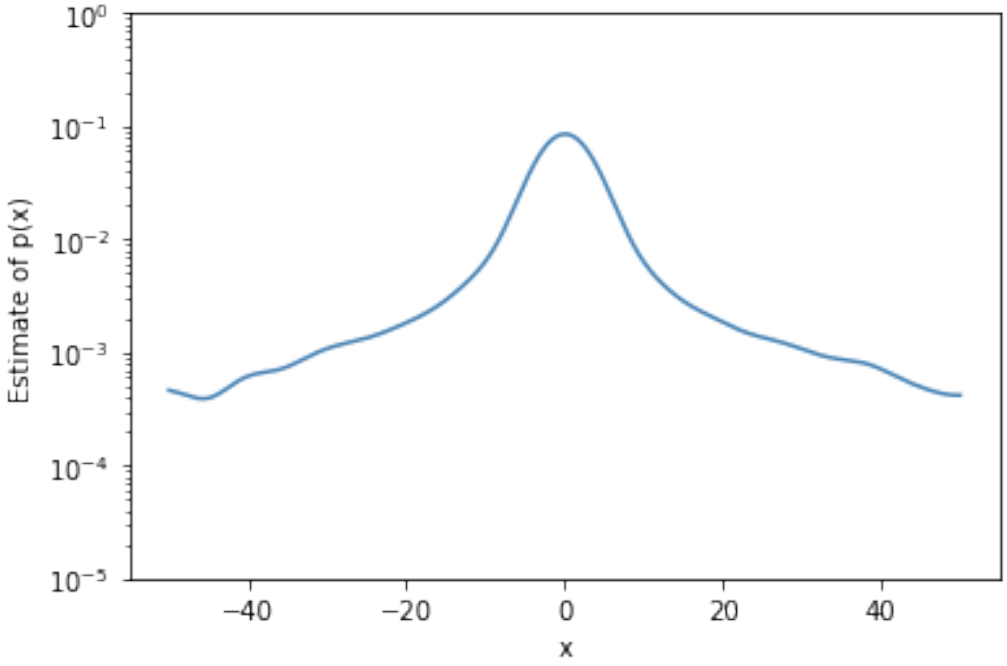


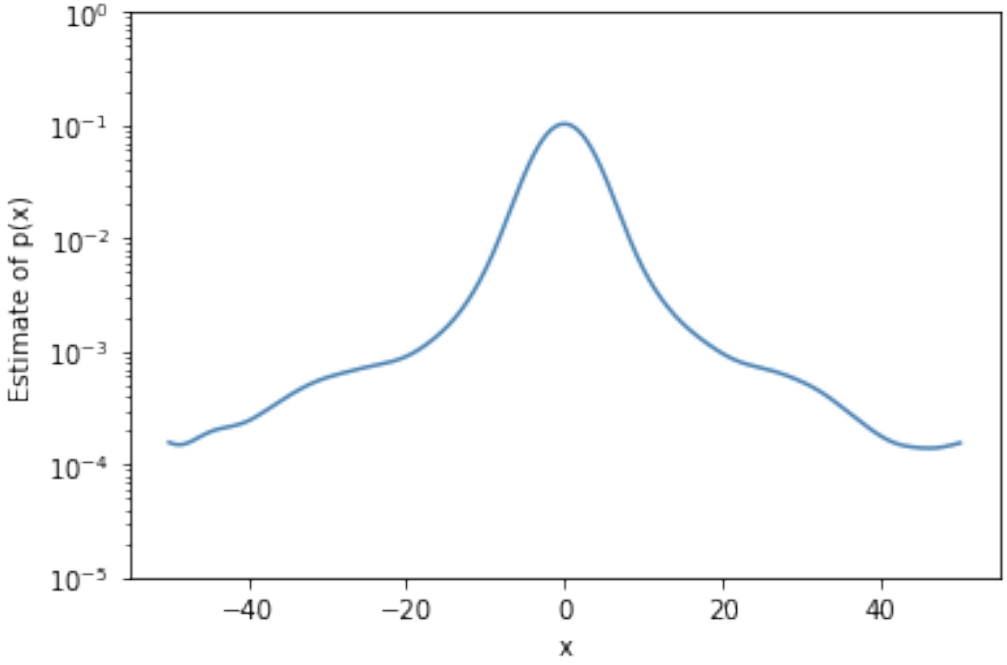

Number of samples
140
120
100
80
60
40
20
0
−15
−10
−5
0
5
10
15
x

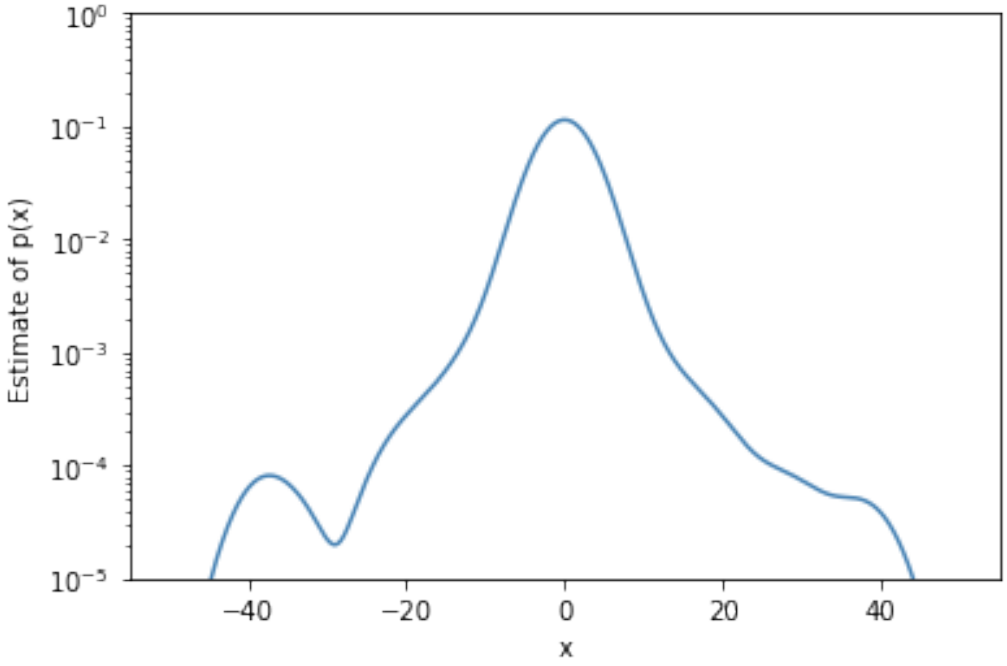

Number of samples
40
30
20
10
0
−4
−2
0
2
4
x

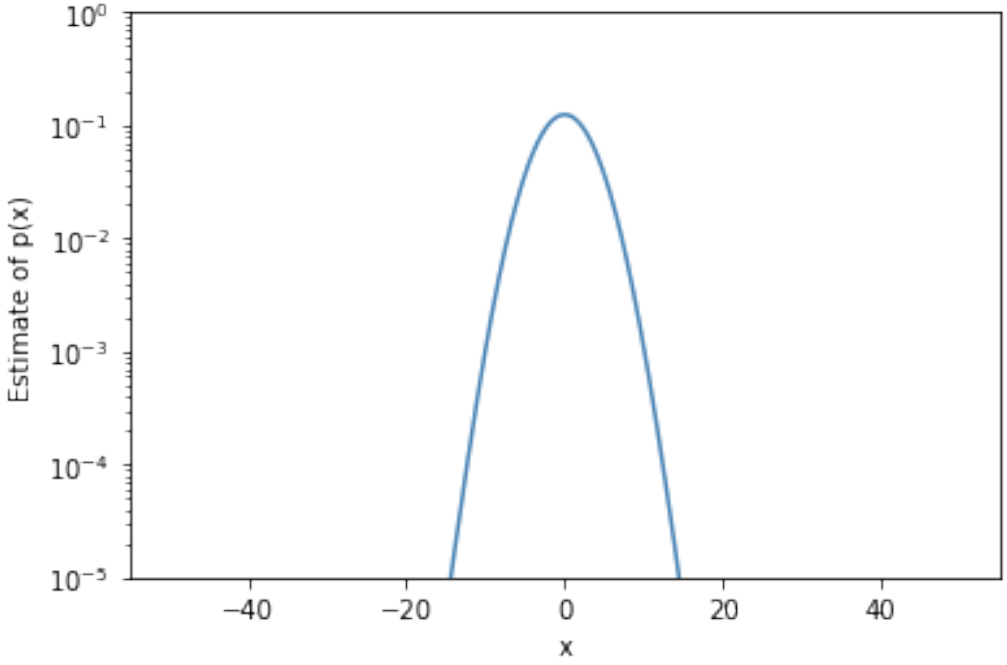


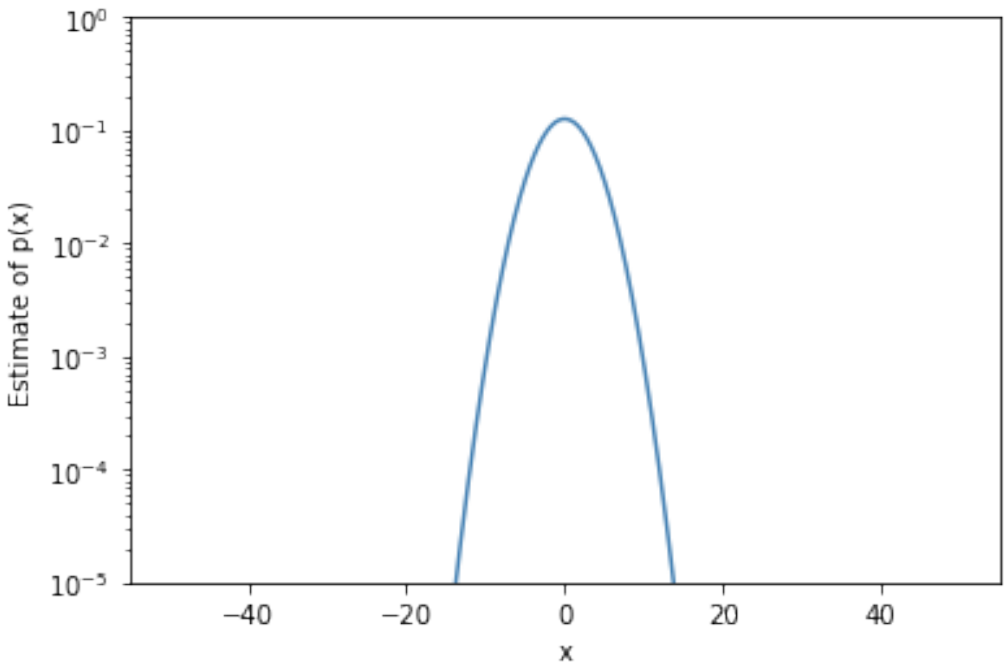
[190]:
```
N = 10000
x = np.linspace(-50, 50, 1000)

for theta in [0.5, 1, 2, 10, 100]:
    fig, ax1 = plt.subplots()

    ax1.semilogy(x,
                 kernel_smoothed_density(x,
                                         gamma_sampled_gaussian(N, theta),
                                         width=3),
                 label=r'$\theta = {}$'.format(theta),
                 color=color1)

    ax1.set_ylim(1e-5, 1)
    ax1.set_xlabel('x')
    ax1.set_ylabel('Estimate of p(x)')

    plt.savefig(f'figures/gamma_sampled_gaussian_smoothed_{theta}.png',␣
 ↪bbox_inches='tight')
```


Estimate of p(x)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
−40
−20
0
20
40
x


Estimate of p(x)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
−40
−20
0
20
40
x


Estimate of p(x)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
−40
−20
0
20
40
x


Estimate of p(x)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
−40
−20
0
20
40
x



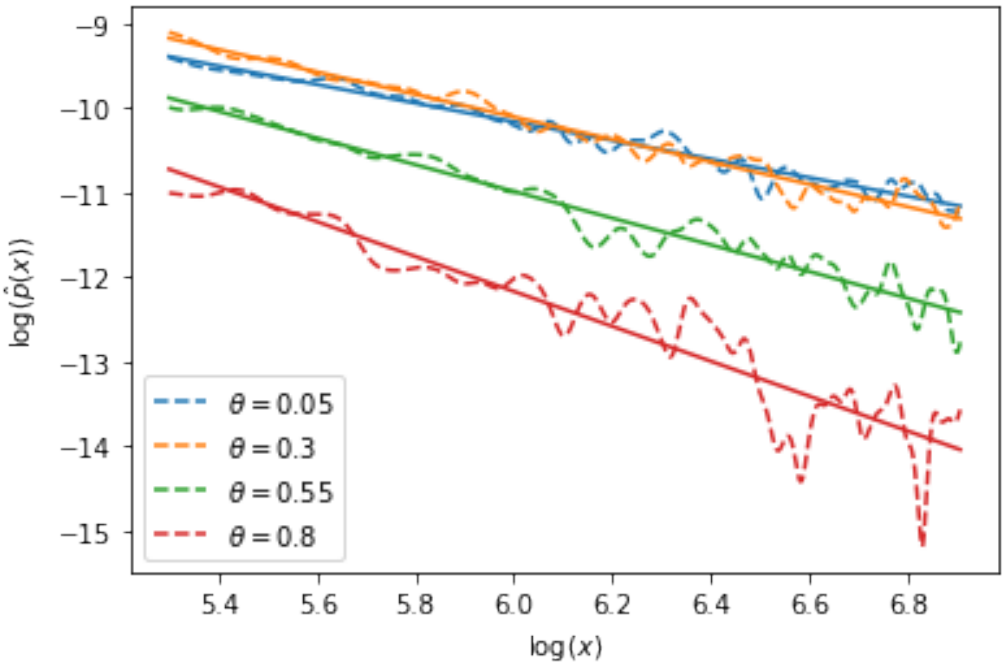
[191]:
```
N = int(1e5)
samples = {theta: gamma_sampled_gaussian(N, theta) for theta in np.linspace(0.
↪05, 0.8, 10)}
```

[192]:
```
width = 10
x = np.linspace(0, int(1e3), int(1e3))
smoothed_samples = {theta: kernel_smoothed_density(x, samples_theta, width)
                    for theta, samples_theta in samples.items()}
```

[193]:
```
from scipy.optimize import curve_fit

fit = {}
for theta, smoothed in smoothed_samples.items():
    xdata = np.log(x[200:])
    ydata = np.log(smoothed[200:])
    np.nan_to_num(ydata, neginf=-9e99)
    fit[theta] = curve_fit(lambda x, a, b: a*x+b, xdata=xdata, ydata=ydata)[0]
```

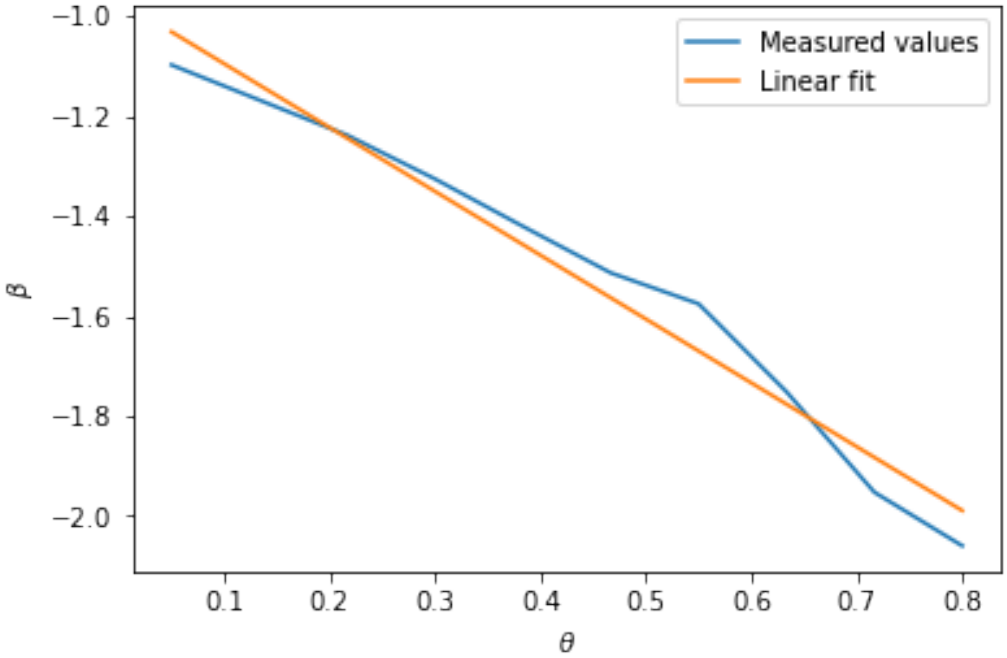
[194]:
```
plt.figure()
for i, theta in enumerate([0.05, 0.3, 0.55, 0.8]):
    m, c = fit[theta]
    logx = np.log(x[200:])
    plt.plot(logx, np.log(smoothed_samples[theta][200:]),
```

```
            label=r'$\theta={}$'.format(theta), linestyle='dashed',␣
 ↪color=f'C{i}')
    plt.plot(logx, m*logx+c)

plt.xlabel(r'$\log(x)$')
plt.ylabel(r'$\log(\hat{p}(x))$')
plt.legend()
plt.savefig('figures/gamma_sampled_gaussian_tail_fit.png')
```



[197]:
```
plt.figure()
thetas = [theta for theta in smoothed_samples.keys()]
gradients = [fit[theta][0] for theta in thetas]
plt.plot(thetas, gradients, label='Measured values')
m, c = curve_fit(lambda x, a, b: a*x+b, xdata=thetas, ydata=gradients)[0]
plt.plot(np.linspace(0.05, 0.8), m*np.linspace(0.05, 0.8) + c, label='Linear␣
 ↪fit')
# plt.scatter(0.8, fit[0.8][0], label=r'$\theta=0.8$ (discarded)')
plt.xlabel(r'$\theta$')
plt.ylabel(r'$\beta$')
plt.legend()
plt.savefig('figures/beta_vs_theta.png')
print(f"B = {m}T + {c}")
```

```
B = -1.2760084202219186T + -0.9683391662146092
```


Measured values
Linear fit
−1.0
−1.2
−1.4
−1.6
−1.8
−2.0
β
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
θ