COURSEWORK 1 (10%)

This assignment has two parts and graded over 20 pts. Some general remarks:

- The assignment is due on 9 Nov. 2022, to be submitted via Blackboard (see the instructions on the course website).
- You should submit a PDF report. You can do this via two ways (your choice):
  1. Prepare a PDF written in LaTeX. In this case, your report should have two parts: Main part (with derivations, text, and figures) and Appendix (just code). Please keep the report limited to 8 pages. Fewer is appreciated, the derivation is not too long and plots should not take too much space. The code should be posted in Appendix. Appendix should contain only code, code comments, and titles (Code for Q1, Code for Q2). You can use `pythonhighlight` package (or something similar) in LaTeX for readability (please do!).
  2. Prepare an IPython notebook and export it as a PDF. In this case, because the code, the text, and the equations will be next to each other, the limit is 8 total pages. Note, again, that much less is enough.

  Choose the best option for your report.
- Note that most of the things required in this assignment can be found in solved exercises and course material. You can reuse the code from the course material, but try to personalise your code in order to avoid having problems with plagiarism checks.

Q1: SAMPLING FROM CHI-SQUARED USING REJECTION SAMPLING (15 PTS)

This question is simple and similar to Gamma$(\alpha, 1)$ sampling example in the lecture notes (Example 2.13).

Consider the Chi-squared density

$$p_\nu(x) = \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)} x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}}, \qquad x > 0.$$

Here $\Gamma(n) = (n-1)!$ for integer n and we will only consider integer values[1]. In general, we assume that $\nu > 2$ and divisible by 2.

The goal is to sample from p_ν using rejection sampling. The proposal distribution is exponential

$$q_\lambda(x) = \lambda e^{-\lambda x}, \qquad x > 0,$$

where $0 < \lambda < 1/2$ for boundedness of the ratio.

We will follow the following steps.

1. Compute $M_\lambda = \sup_x p_\nu(x)/q_\lambda(x)$. Note that this optimisation is over x and you need to find $x^\star$ and evaluate the ratio (5 pts).
2. Next find the optimal $\lambda^\star$ in terms of ν (5 pts). For this you need to optimise M_λ over λ.

[1] The Python code that implements this is `np.math.factorial(n-1)`.

3. Finally, implement the rejection sampler for $\nu = 4$ using corresponding optimal $\lambda^\star$. Note that your $M_{\lambda^\star}$ will still depend on ν. Then,

   - Plot your histogram, $p_\nu(x)$, and $M_{\lambda^\star} q_{\lambda^\star}(x)$ (3pts) (Fig. 2.6 in Lecture Notes, right hand side (optimal part) is a good example)
   - Compute the acceptance rate (count the number of accepted samples / total number of samples) and compare it to the theoretical acceptance rate $\hat{a} = 1/M_{\lambda^\star}$, verify that they coincide or have very close values (2 pts).

Some notes:

- Do not use `np.random.exponential`[2] and use the inversion method to sample from exponential.
- You can use `np.random.uniform` to sample from uniform.
- To plot chi-square density, the following code will be useful (remove line breaks).

```
def p(x, nu):
    return x ** (nu / 2 - 1) * np.exp(-x / 2) / (2 ** (nu / 2) *
                                        np.math.factorial(int(nu /
                                         2) - 1))
```

Q2: SAMPLE FROM A MIXTURE OF CHI-SQUARED (5 PTS)

Convert your rejection sampler into a function that just gives you a single sample (e.g. runs until a single sample is accepted). Then, use this function to sample

$$p(x) = \sum_{i=1}^{3} w_i p_{\nu_i}(x), \qquad x > 0,$$

where each $p_{\nu_i}(x)$ is a Chi-squared density with

$$\begin{aligned} \nu_1 &= 4, \\ \nu_2 &= 16, \\ \nu_3 &= 40. \end{aligned}$$

The weights are given by $w_1 = 0.2$, $w_2 = 0.5$, $w_3 = 0.3$. Draw $n = 100000$ samples.

- Sample indices from the discrete distribution using inversion method (3 pts).
- Plot the histogram and your density (2 pts).

You can use the following code to plot the density:

```
def mixture_density(x, w, nu):
    return w[0]*p(x, nu[0]) + w[1]*p(x, nu[1]) + w[2]*p(x, nu[2])

xx = np.linspace(0, 60, 1000)
plt.plot(xx, mixture_density(xx, w, nu), color='k', linewidth=2)
```

[2] numpy's exponential has a different parameterisation!