InfinityQ take home research question

This is a research question, which we will use to test your research process, ability to decide on a solution while weighing pros and cons, and put together a solution for a complex problem.

To assess the quality of the solution, we will look at a combination of factors including the thoroughness of the research process, the ability to communicate your choices, the ability to understand the computational/engineering effects of your choices, the quality of code that your produce, and documentation that you create to justify your choices.

Again, there is no right answer! This is meant as an assessment of how you think, and this is an intentionally difficult question. We will accept partial solutions, and we know that this is not easy.

Question:

Imagine that you must sample from a probability distribution, but you must do it *many many times*, so you need to make sure that it is computationally efficient. The probability distribution is a modified gaussian distribution, which is skewed towards one end of the distribution. The goal is to produce a distribution where one end of the distribution is more likely than the other, to produce this "skewed" effect. It can be written as below:

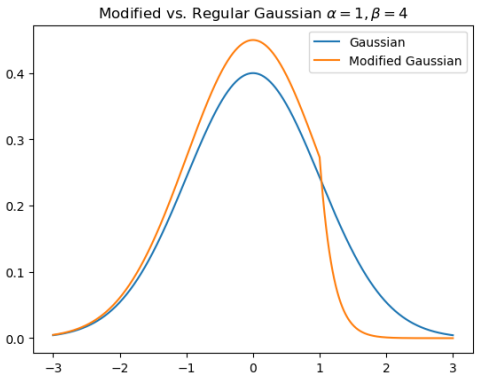
$$p(x) = \frac{1}{Z} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 - L_{\alpha,\beta}(x)}$$

$$L_{\alpha,\beta}(x) = \begin{cases} 0 \; if \; x < \alpha \\ \beta * (x - \alpha) \; if \; x \geq \alpha \end{cases}$$

The function $p(x)$ is parameterized by $\mu, \sigma, \alpha, \beta$. The function $L(x)$ is parameterized by α, β. The parameters $\mu, \sigma, \beta, \alpha \in \mathbb{R}$ and $\sigma > 0, \beta \geq 0$. The constant Z is a normalizing constant and can be written as $Z = \int_{-\infty}^{\infty} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 - L_{\alpha,\beta}(x)}$.

One thing to note, is that this function is very close to being a truncated gaussian distribution, and approaches such a distribution as $\beta \to \infty$.

As an example, below is the probability distribution function for a regular gaussian vs. a skewed gaussian:



This function will be used to sample in a loop to approximate a larger probability distribution, similar to a Gibbs Sampling method.

for i in range(num_samps):
generate sample, $x_i \sim p(x; \alpha, \beta, \mu, \sigma)$
generate sample $y_i \sim g(y; x_i)$
generate $(\alpha, \beta, \mu, \sigma) = f(x_i, y_i)$
store x_i, y_i
Estimate statistical quantities E[x] and E[y] using samples x_i and y_i

A few things to note about this sampling:

- It is *markovian*. The next set of parameters exclusively depend on the set of parameters generated from the last iteration.
- We only need 1 Independent Sample from $p(x)$ for each loop iteration
- At each loop iteration, the parameters $\alpha, \beta, \mu, \sigma$ which parameterize our distribution $p(x)$ change.
- Statistical Efficiency is important, you don't want to generate 1000 attempts for each one sample you get from p(x)
- Think about how the choice of parameters affects your solution quality, statistical efficiency, and computational difficulty.

Some things that may help you:

- You may approximate the function $L_{\alpha,\beta}(x)$ or the full function $p(x)$ to make it more tractable. If you do so, outline how tight your approximation is, and the effect of parameter choices $(\alpha, \beta, \mu, \sigma)$ may affect this.
- You may use any standard libraries, methods, or approaches to produce parts of your algorithm or method. For example, it is fine to say "assume that I can generate a normally distributed number", as this is a solved problem in most computing literature.
    - However, if you do this, you should outline and understand how computationally difficult whatever the standard library is doing.
- You may use any literature you choose, but please cite your sources if you choose some piece of literature!

- You may use ChatGPT (or some other LLM) if you choose. Same as above, cite your sources ☺
- If something is unclear, you can email us back and we will get back to you as soon as we can.

Here are a few methods to sample from this distribution, with references on how to understand them.

- Rejection Sampling: Sampling by generating a sample from an easy to generate proposal distribution, and then accepting or rejecting the sample probabilistically. Only accepted samples are from the distribution of interest.
    - For a proposal distribution of $f(x)$ with $Cf(x) \geq p(x)\ \forall x$ and some constant $C \geq 1$, then you can generate samples from the distribution $p(x)$
    - A good choice of proposal distribution here would be the original gaussian distribution
    - See, for example: https://towardsdatascience.com/what-is-rejection-sampling-1f6aff92330d
- Markov Chain Monte Carlo: Sampling by starting from an initial state, and then sequentially proposing and accepting/rejecting the next state
    - This can be done using Gibbs Sampling, or the Metropolis Hastings Algorithm.
    - See, for example: https://stephens999.github.io/fiveMinuteStats/MH_intro.html
- Inverse Transform Sampling: This directly uses the inverse CDF (cumulative distribution function) to sample from the given distribution.
    - The CDF of a distribution is the integral of the PDF (probability distribution function) listed above.
    - The CDF would be a piecewise function, defined by two gaussian distributions (I'll let you do out the math to figure out what that would look like).
    - See, for example: https://stephens999.github.io/fiveMinuteStats/inverse_transform_sampling.html

Question 1: Evaluate the methods listed above from sampling from the given function. Which of them is most algorithmically efficient? Which is most space efficient? Which is most computationally efficient?

Question 2: Write code using one of the methods that you outlined in Question 1 to sample from p(x). This can be done in whatever language you choose, using any libraries you choose.

Question 3: Analyze the complexity (in time, space, clock cycles, statistical efficiency) of your implementation. Compare this to a standard gaussian implementation and mention how you might improve your efficiency if given more time.

Question 4: If you needed to implement this sampling on a specialized piece of hardware (GPU, FPGA, TPU, ASIC, etc.), would you change your answer for which sampling algorithm would be most efficient? How would your implementation change?

Deliverables for this:

- A small report outlining your answers to question 1 & 3. Can be in word format, pdf, LaTeX, pen & paper, or crayon on the back of a pizza menu.
- Running code demonstrating your answer for question 2. The code should compile, run, and be clean & well documented.