SOLUTIONS WEEK 2

Solution 2.1. $X \sim \text{Exp}(x; 1)$ and $Y = g(X) = \alpha X^{1/\beta}$. We can compute g^{-1} using

$$g^{-1}(y) = \left(\frac{y}{\alpha}\right)^{\beta}.$$

Then using the transformation of random variables formula for 1D

$$\begin{aligned} p_Y(y) &= p_X(g^{-1}(y)) \left|\frac{\mathrm{d}g^{-1}(y)}{\mathrm{d}y}\right|, \\ &= \exp\left(-\left(\frac{y}{\alpha}\right)^{\beta}\right) \beta \left(\frac{y}{\alpha}\right)^{\beta-1} \frac{1}{\alpha}, \end{aligned}$$

which gives us the result. Algorithm for Weibull:

1. Generate $U \sim U(0, 1)$.
2. Set $X = -\log(1 - U)$ (so X exponential).
3. Set $Y = \alpha X^{\frac{1}{\beta}}$ (so Y is Weibull).

The code is provided below:

```python
import numpy as np
import matplotlib.pyplot as plt

n = 20000

# sample from weibull
def weibull_density(w, a, b):
    return b * a**(-b) * w**(b-1) * np.exp(- (w / a)**b)

a = 1
b = 2

samples = np.array([]) # list to store samples

for i in range(n):
    u = np.random.uniform(0, 1)
    x = - np.log(1 - u)
    w = a * x**(1/b)
    samples = np.append(samples, w)

xx = np.linspace(0, 4, 1000)
yy = weibull_density(xx, a, b)
plt.plot(xx, yy, 'k-')
plt.hist(samples, bins=100, density=True, rwidth=0.8, color='r', alpha
                                    =0.5)
plt.title("Weibull density and histogram")
plt.xlabel("x")
plt.xlim([0, 4])
plt.show()
```

Solution 2.2. Similar to the course examples, we can compute the density of X_1, X_2 as

$$p_{x_1,x_2}(x_1, x_2) = p_{r,\theta}(g^{-1}(x_1, x_2)) \left|\det J_{g^{-1}}\right|,$$

where g^{-1} is the inverse transformation. Let us construct the inverse transform:

$$r = \sqrt{x_1^2 + x_2^2}$$

by just observing that $\cos^2\theta + \sin^2\theta = 1$. We can also write that

$$\theta = \arctan(x_2/x_1).$$

Therefore, we can write the inverse transformation as

$$g^{-1}(x_1, x_2) = \left(\sqrt{x_1^2 + x_2^2}, \arctan(x_2/x_1)\right).$$

We next need to compute the Jacobian matrix as:

$$J_{g^{-1}} = \begin{bmatrix} \frac{\partial g_1^{-1}}{\partial x_1} & \frac{\partial g_1^{-1}}{\partial x_2} \\ \frac{\partial g_2^{-1}}{\partial x_1} & \frac{\partial g_2^{-1}}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} \\ \frac{1}{1+(x_2/x_1)^2}\frac{-x_2}{x_1^2} & \frac{1}{1+(x_2/x_1)^2}\frac{1}{x_1} \end{bmatrix}.$$

Therefore, the determinant is

$$\det J_{g^{-1}} = \frac{1}{\sqrt{x_1^2 + x_2^2}}.$$

Therefore, the density of X_1, X_2 is

$$\begin{aligned} p_{x_1,x_2}(x_1, x_2) &= \text{Unif}(\sqrt{x_1^2 + x_2^2}; 0, 1)\text{Unif}(\arctan(x_2/x_1); -\pi, \pi)\frac{1}{\sqrt{x_1^2 + x_2^2}} \\ &= \frac{1}{2\pi\sqrt{x_1^2 + x_2^2}} \qquad \text{for } x_1^2 + x_2^2 \leq 1. \end{aligned}$$

Solution 2.3. We would like to compute

$$M = \sup_x \frac{p(x)}{q(x)},$$

wbere $p(x)$ is the Beta density

$$p(x) = \text{Beta}(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1 - x)^{\beta-1}.$$

and note that $\Gamma(n) = (n - 1)!$ for $n \in \mathbb{N}$. We compute the derivative

$$\frac{\mathrm{d} \log p(x)/q(x)}{\mathrm{d}x} = \frac{\alpha - 1}{x} + \frac{1 - \beta}{1 - x}$$

The maximum is

$$x^\star = \frac{\alpha - 1}{\alpha + \beta - 2}.$$

Finding $x^\star$, we compute the supremum by plugging $x^\star$ into the ratio p/q which is given as

$$M = \frac{p(x^\star)}{q(x^\star)}.$$

This leads to

$$M = \frac{(\alpha - 1)^{\alpha-1}(\beta - 1)^{\beta-1}}{(\alpha + \beta - 2)^{\alpha+\beta-2}} \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}.$$

For the box example, we can find our optimal $M = 1.5$ by plugging $\alpha = 2$ and $\beta = 2$. The procedure is then given by

- Sample $X' \sim q(x) = \text{Unif}(0, 1)$
- Sample $U \sim \text{Unif}(0, 1)$
- If $U \leq p(X')/Mq(X')$,
  - Accept X'

Solution 2.4. In order to achieve this, we need to rely on transformation of random variables formula. However, since Y is a function of both X_1 and X_2, we need a transformation in two dimensions to be able to use transformation of random variables. In general, one chooses another auxiliary variable that makes computations easier (again, please practice transformation of random variables, that must have been the part of previous courses).

In our case, let $Y = \frac{X_1}{X_1+X_2}$ and define an auxiliary $Z = X_1 + X_2$. We aim at finding the density of $p_{y,z}(y, z)$ and this is given by

$$p_{y,z}(y, z) = p_{x_1,x_2}(g^{-1}(y, z))|\det J_{g^{-1}}|. \tag{1}$$

The inverse g^{-1} can be constructed in both arguments from the fact that

$$X_1 = YZ, \quad \text{This is why we chose } Z = X_1 + X_2,$$

and

$$X_2 = Z - X_1 = Z - YZ = Z(1 - Y).$$

Therefore, we obtain $g^{-1}(y, z) = (yz, z(1 - y))$, therefore $g_1^{-1} = yz$ and $g_2^{-1} = z(1 - y)$. Now we compute the Jacobian:

$$\begin{aligned} J_{g^{-1}} &= \begin{bmatrix} \frac{\partial g_1^{-1}}{\partial y} & \frac{\partial g_1^{-1}}{\partial z} \\ \frac{\partial g_2^{-1}}{\partial y} & \frac{\partial g_2^{-1}}{\partial z} \end{bmatrix} \\ &= \begin{bmatrix} z & y \\ -z & (1 - y). \end{bmatrix} \end{aligned}$$

Therefore $\det J_{g^{-1}} = z - zy - (-zy) = z$. Therefore, the formula (1) becomes

$$\begin{aligned} p_{y,z}(y, z) &= \text{Gamma}(yz; \alpha, 1)\text{Gamma}(z(1 - y); \beta, 1)z, \\ &= \frac{1}{\Gamma(\alpha)}(yz)^{\alpha-1}e^{-yz}\frac{1}{\Gamma(\beta)}(z(1 - y))^{\beta-1}e^{-z(1-y)}z \end{aligned}$$

We are interested ultimately in $p_y(y)$, therefore, we integrate this

$$\begin{aligned} p_y(y) &= \int p_{y,z}(y, z)\mathrm{d}z \\ &= \frac{1}{\Gamma(\alpha)\Gamma(\beta)}y^{\alpha-1}(1 - y)^{\beta-1}\int e^{-z}z^{\alpha+\beta-1}\mathrm{d}z, \\ &= \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}y^{\alpha-1}(1 - y)^{\beta-1}. \end{aligned}$$

which is the Beta distribution as intended. The last line follows from the definition of the Gamma function:

$$\Gamma(a) = \int z^{a-1}e^{-z}\mathrm{d}z.$$

Solution 2.5. (a) We first write the ratio

$$R(x) = \frac{p(x)}{q_\alpha(x)} = \frac{1/\sqrt{2\pi}\exp(-x^2/2)}{(\alpha/2)\exp(-\alpha|x|)}.$$

This is not differentiable at 0, but we can do a piecewise computation to verify the maximum. First note that

$$R(0) = \alpha^{-1}\sqrt{\frac{2}{\pi}}$$

Since below computation excludes the case $x = 0$, we keep this in mind to determine maximum later.

For $x > 0$, we have the ratio

$$R(x) = \frac{p(x)}{q_\alpha(x)} = \frac{1/\sqrt{2\pi}\exp(-x^2/2)}{(\alpha/2)\exp(-\alpha x)},$$

Taking derivative of $\log$ and setting it to 0, we obtain

$$\frac{\mathrm{d}\log R(x)}{\mathrm{d}x} = -x + \alpha = 0,$$

hence $x^\star = \alpha$ (since the second derivative is negative). Similarly for $x < 0$, the computation shows

$$x^\star = -\alpha.$$

Note however that $R(\alpha) = R(-\alpha)$ (due to the use of square and absolute value), we obtain that

$$R(\alpha) = \alpha^{-1}\sqrt{\frac{2}{\pi}}\exp(\alpha^2/2).$$

To verify that this is the value at maximum, we also verify $R(\alpha) > R(0)$ as $\exp(\alpha^2/2) > 1$. Hence, we can conclude that

$$R(\alpha) = M_\alpha = \sup_x \frac{p(x)}{q_\alpha(x)}.$$

Next, we would like to optimise

$$M_\alpha = \alpha^{-1}\sqrt{\frac{2}{\pi}}\exp(\alpha^2/2).$$

Computing the derivative of M_α and setting it to zero,

$$\frac{\mathrm{d}\log M_\alpha}{\mathrm{d}\alpha} = -\frac{1}{\alpha} + \alpha = 0,$$

which implies $\alpha^2 = 1$. Since we assumed $\alpha > 0$, we conclude $\alpha^\star = 1$.

(b) In the lectures (Lecture 4), we have seen that

$$\hat{a} = \frac{1}{M}.$$

Our optimal M here is

$$M := M_{\alpha^\star} = \sqrt{\frac{2}{\pi}} e^{1/2}.$$

Therefore,

$$\hat{a} = \frac{1}{M} = \sqrt{\pi/2e}.$$

(c) The code is given below:

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

rng = np.random.default_rng(6)

N = 100000

# Gaussian density
def p(x):
    return 1/np.sqrt(2*np.pi)*np.exp(-x**2/2)

# Proposal density (Laplace)
def q(x):
    return (alpha/2) * np.exp(-alpha*np.abs(x))

alpha = 1 # as computed in part (a)
M_a = np.sqrt(2/np.pi) * np.exp(1/2) # as computed in part (b)


# implement rejection sampling

acc = 0 # count for accepted samples
x_samples = np.array([])

for i in range(N):
    # sample from proposal
    x_prime = rng.laplace(scale=1/alpha)
    u = rng.uniform(0,1)
    # accept/reject
    if u < p(x_prime)/(M_a*q(x_prime)):
        acc += 1
        x_samples = np.append(x_samples, x_prime)

print("Acceptance rate: ", acc/N)
```