SOLUTIONS TO COURSEWORK 1

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

We will provide the solution in the same steps:

1. Note that the ratio is given by

$$
\begin{aligned}
R(x) = \frac{p_\nu(x)}{q_\lambda(x)} &= \frac{1}{2^{\nu/2}\Gamma\left(\frac{\nu}{2}\right)} \frac{x^{\frac{\nu}{2}-1}e^{-\frac{x}{2}}}{\lambda e^{-\lambda x}}, \\
&= \frac{1}{\lambda 2^{\nu/2}\Gamma\left(\frac{\nu}{2}\right)} x^{\frac{\nu}{2}-1} e^{-\left(\frac{1}{2}-\lambda\right)x}.
\end{aligned}
$$

Note that the requirement $0 < \lambda < 1/2$ ensures that this ratio is bounded. Next, we optimise w.r.t. x by taking the log and ignoring unrelated terms (to x):

$$
\log R(x) = \left(\frac{\nu}{2} - 1\right) \log x - \left(\frac{1}{2} - \lambda\right) x.
$$

Taking the derivative and setting it to zero, we obtain

$$
\frac{\mathrm{d} \log R(x)}{\mathrm{d}x} = \frac{\left(\frac{\nu}{2} - 1\right)}{x} - \left(\frac{1}{2} - \lambda\right) = 0,
$$

therefore we obtain

$$
x^\star = \frac{\nu - 2}{1 - 2\lambda}.
$$

Checking the second derivative

$$
\frac{\mathrm{d}^2 \log R(x)}{\mathrm{d}x^2} = \frac{1 - \frac{\nu}{2}}{x^2} < 0,
$$

for any x (hence $x = x^\star$) since $\nu > 2$, therefore this is a maximum.

Evaluating the ratio, we obtain the supremum

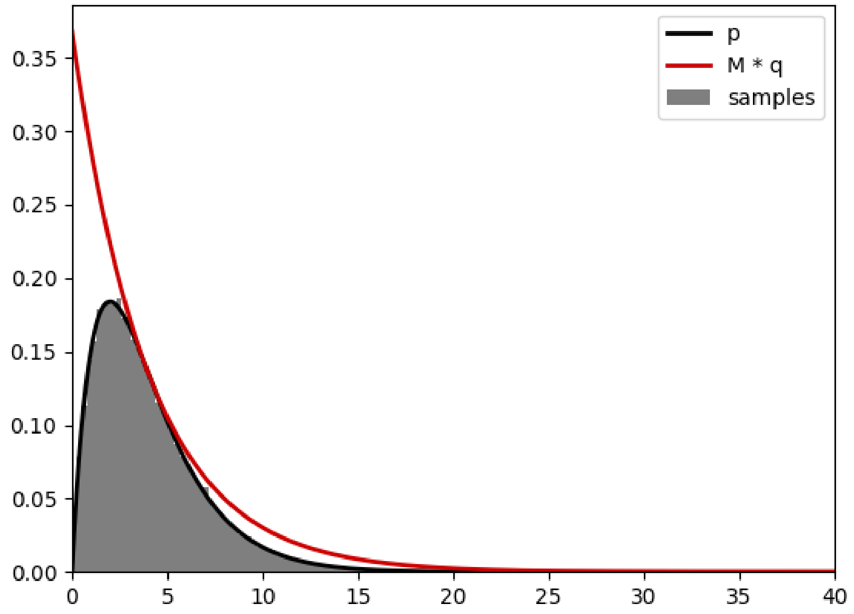
$$
\begin{aligned}
M_\lambda = R(x^\star) &= \frac{1}{2^{\nu/2}\Gamma\left(\frac{\nu}{2}\right)} \frac{\left(\frac{\nu-2}{1-2\lambda}\right)^{\frac{\nu}{2}-1} e^{-\left(\frac{1}{2}-\lambda\right)\left(\frac{\nu-2}{1-2\lambda}\right)}}{\lambda} \\
&= \frac{1}{2^{\nu/2}\Gamma\left(\frac{\nu}{2}\right)} \frac{\left(\frac{\nu-2}{1-2\lambda}\right)^{\frac{\nu}{2}-1} e^{-\left(\frac{\nu}{2}-1\right)}}{\lambda}
\end{aligned}
$$

2. Minimising M_λ w.r.t. λ, we first take log (and drop unrelated terms to λ)

$$
\log M_\lambda =^c - \left(\frac{\nu}{2} - 1\right) \log\left(1 - 2\lambda\right) - \log \lambda,
$$

taking the derivative and setting it to zero

$$
\frac{\mathrm{d} \log M_\lambda}{\mathrm{d}\lambda} = \frac{\nu - 2}{1 - 2\lambda} - \frac{1}{\lambda} = 0
$$



gives us $\lambda^\star = 1/\nu$. The second derivative

$$\frac{\mathrm{d}^2 \log M_\lambda}{\mathrm{d}\lambda^2} = \frac{2(\nu-2)}{(1-2\lambda)^2} + \frac{1}{\lambda^2},$$

and plugging $\lambda^\star = 1/\nu$ shows

$$\left.\frac{\mathrm{d}^2 \log M_\lambda}{\mathrm{d}\lambda^2}\right|_{\lambda=1/\nu} = \frac{2(\nu-2)}{(1-2/\nu)^2} + \nu^2 > 0,$$

which proves that this is indeed a minimum.

3. For this part, the code is attached – see here the example figure. The theoretical acceptance rate is: 0.6795 and your code should output something close, matching the first two digits after the decimal point.

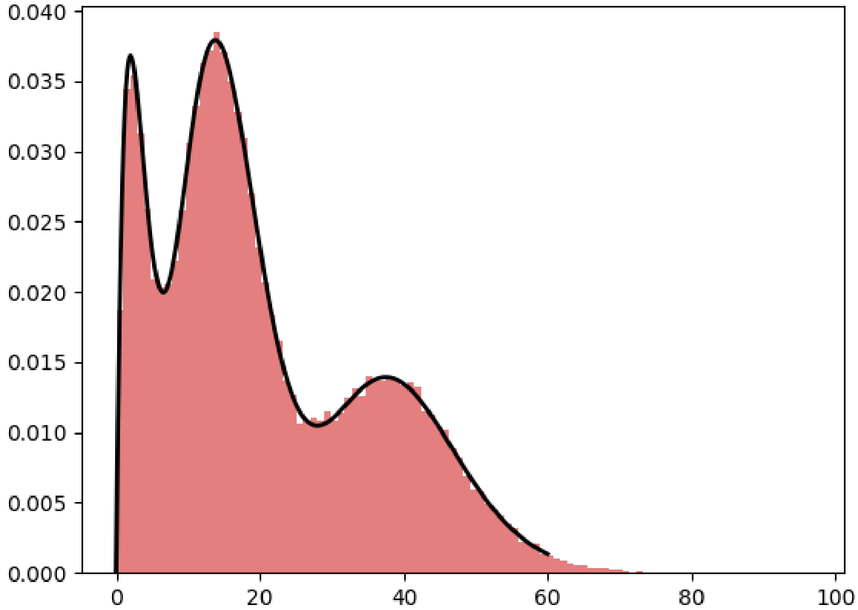
SOLUTION TO Q2: SAMPLE FROM A MIXTURE OF CHI-SQUARED (5 PTS)

Here the goal was to define first a rejection sampler that would return the first accepted sample from a component of the mixture $p_{\nu_i}(x)$. Then, the problem just becomes a regular sampling from a mixture, but each mixture distribution sampling procedure was a rejection sampling itself.

This can be seen in the code below. This is done in `chi_squared_rejection_sampling` function in the Appendix below. The resulting sample figure is:

APPENDIX

CODE FOR Q1




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

# define the chi-squared density
def p(x, nu):
    return x ** (nu / 2 - 1) * np.exp(-x / 2) / (2 ** (nu / 2) * np.math.factorial(int(nu / 2) - 1))

def q(x, lam):
    return lam * np.exp(-lam * x)

def M(nu, lam):
    x_star = (nu - 2) / (1 - 2 * lam)
    return p(x_star, nu) / q(x_star, lam)

nu = 4
lam = 1 / nu

n = 100000

samples = np.array([])
acc = 0

for i in range(n):

    u_exp = np.random.uniform(0, 1)
    x = -np.log(1 - u_exp) / lam
    u = np.random.uniform(0, 1)

    if u < p(x, nu)/(M(nu, lam) * q(x, lam)):
        samples = np.append(samples, x)
        acc += 1
```

```
print(acc/n)
print(1 / M(nu, lam))

xx = np.linspace(0, 40, 1000)
plt.plot(xx, p(xx, nu), color='k', linewidth=2, label='p')
plt.plot(xx, M(nu, lam) * q(xx, lam), color=[0.8, 0 , 0], linewidth=2, label='M * q')
plt.hist(samples, bins=100, density=True, color='k', alpha=0.5, label='samples')
plt.legend()
plt.xlim(0, 40)
plt.show()
```

CODE FOR Q2

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

# define the chi-squared density
def p(x, nu):
    return x ** (nu / 2 - 1) * np.exp(-x / 2) / (2 ** (nu / 2) * np.math.factorial(int(nu / 2) - 1))

def q(x, lam):
    return lam * np.exp(-lam * x)

def M(nu, lam):
    x_star = (nu - 2) / (1 - 2 * lam)
    return p(x_star, nu) / q(x_star, lam)


def chi_squared_rejection_sampling(nu, lam):
    sample = np.array([])

    while len(sample) == 0:
        u_exp = np.random.uniform(0, 1)
        x = -np.log(1 - u_exp) / lam
        u = np.random.uniform(0, 1)

        if u < p(x, nu)/(M(nu, lam) * q(x, lam)):
            sample = np.append(sample, x)

    return sample


def discrete_sampling(w):
    s = np.arange(len(w))
    c = np.cumsum(w)
    u = np.random.uniform(0, 1)

    for i in range(len(w)):
        if u <= c[i]:
            return s[i]
```

```

n = 100000
w = [0.2, 0.5, 0.3]
nu = [4, 16, 40]

samples = np.array([])

for i in range(n):
    j = discrete_sampling(w)
    sample = chi_squared_rejection_sampling(nu[j], 1/nu[j])
    samples = np.append(samples, sample)

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)
plt.hist(samples, bins=100, density=True, color=[0.8, 0, 0], alpha=0.5)
plt.show()
```