Assessed Coursework 2

CID 01252821

Question 1

Part a

Under the Poisson likelihood

$$p(\boldsymbol{x}|\theta) = \prod_{i=1}^{n} p(x_i|\theta) = \frac{\theta^{\dot{x}} e^{-n\theta}}{x_1!x_2!...x_n!}$$

or

$$p(\boldsymbol{x}|\theta) \propto \theta^{\dot{x}} e^{-n\theta},$$

where $\dot{x} = \sum_{i=1}^{n} x_i$. If $\theta \sim Uniform(0, b)$, $b > 0$ then

$$p(\theta|b) = \frac{1}{b}$$

or

$$p(\theta|b) \propto 1.$$

Part b

By (4.2) from the lecture notes it follows

$$\pi(\theta|b) \propto p(x|\theta, b)p(\theta|b) \propto \theta^{\dot{x}} e^{-n\theta}$$

which is proportional to the density of $Gamma(\dot{x}+1, n)$. Hence, $\theta|\boldsymbol{x}, b \sim Gamma(\dot{x}+1, n)$.

Part c

The lower incomplete gamma function is defined as

$$\gamma(s, t) := \int_0^t u^{s-1} e^{-u} du.$$

The marginal likelihood is

$$p(\boldsymbol{x}|b) = \int_{\Theta} p(\boldsymbol{x}|\theta)p(\theta|b)d\theta = \int_0^b \frac{\theta^{\dot{x}} e^{-n\theta}}{x_1!x_2!...x_n!}\frac{1}{b}d\theta = \frac{1}{x_1!x_2!...x_n!b}\int_0^b \theta^{\dot{x}} e^{-n\theta} d\theta.$$

Let's substitute $y = n\theta$, then $\theta = \frac{y}{n}$, and

$$p(\boldsymbol{x}|b) = \frac{1}{x_1!x_2!...x_n!b}\frac{1}{n^{\dot{x}+1}}\int_0^{nb} y^{\dot{x}}e^{-y}dy = \frac{1}{x_1!x_2!...x_n!b}\frac{1}{n^{\dot{x}+1}}\gamma(\dot{x}+1, nb).$$

Part d

The log marginal likelihood is $\log p(\boldsymbol{x}|b)$, or if we want to be analytically explicit:

$$\log p(\boldsymbol{x}|b) = \log(\frac{1}{x_1!x_2!...x_n!b}\frac{1}{n^{\dot{x}+1}}\gamma(\dot{x}+1, nb)) = \log\gamma(\dot{x}+1, nb) - (\dot{x}+1)\log n - \log b - \log(x_1!x_2!...x_n!).$$

Defining a function in Python calculating the log marginal likelihood:

```python
from typing import List

import matplotlib.pyplot as plt
import numpy as np
import scipy.special as sp


def log_marginal_likelihood(x: List[int], b: float):
  """Calculates the log marginal likelihood log p(x | b)
  for a sequence of n observations x conditional on a value b > 0.
  """
  n = len(x)
  k = np.prod(sp.factorial(x))
  alpha = np.sum(x) + 1

  incomplete_gamma_function = sp.gammainc(alpha, n * b)
  # scipy uses a regularized version of gammainc
  incomplete_gamma_function *= sp.gamma(alpha)

  return np.log(incomplete_gamma_function) - alpha * np.log(n) - np.log(b) - np.log(k)
```

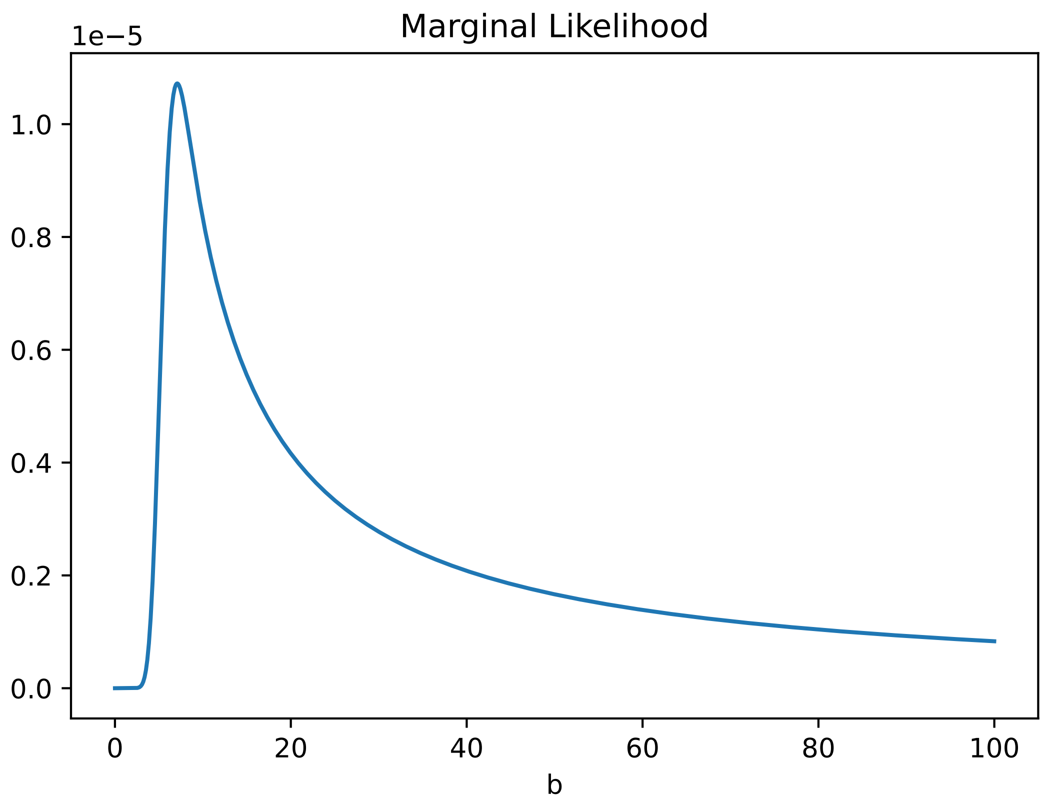
Part e

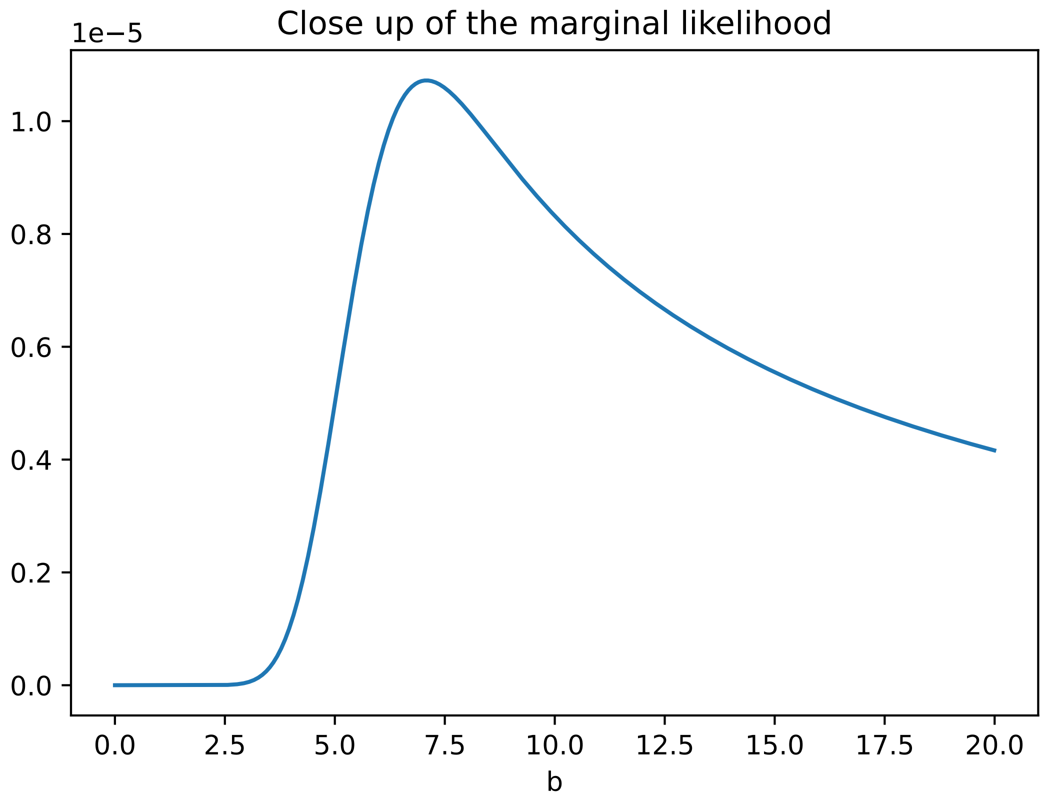
In order to plot the marginal likelihood we are going to use the previously defined function `log_marginal_likelihood()` and just take the exponential of the result from it:

```python
import scipy.stats as st

vecRk4 = np.vectorize(log_marginal_likelihood, excluded=['x'])
dtime = np.arange(0.001, 100, step=0.01)
S = np.exp(vecRk4(x = np.array([5, 3, 8, 7, 4]), b=dtime))

plt.plot(dtime,S)
plt.title("Marginal Likelihood")
plt.xlabel('b')
plt.show()
```


Marginal Likelihood
1e−5
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
b



We can identify a sharp peak around the value 7.5, suggesting the values of b around it are the most likely and are well supported by the given observed data. These values are more likely to be the true value of the parameter. The high marginal likelihood around 7.5 might also suggest that we are working with a complex model, or that we are overfitting.

Question 2

Part a

Part i

By (4.2) from the lecture notes the posterior is

$$\pi(\theta) \propto p(\boldsymbol{x}|\theta)p(\theta).$$

For the joint likelihood, for $x_i \in [0,1]$ we have

$$p(\boldsymbol{x}|\theta) = \prod_{i=1}^{n} p(x_i|\theta) = \prod_{i=1}^{n} (\theta+1)(\theta+2)x_i^{\theta}(1-x_i) =$$

$$= (\theta+1)^n(\theta+2)^n x_1^{\theta}...x_n^{\theta}(1-x_1)...(1-x_n) =$$

$$= (\theta+1)^n(\theta+2)^n e^{\theta\sum_{i=1}^{n}\log x_i}(1-x_1)...(1-x_n),$$

or

$$p(\boldsymbol{x}|\theta) \propto (\theta+1)^n(\theta+2)^n e^{\theta \sum_{i=1}^n \log x_i}.$$

Now, having $p(\theta) \propto (\theta+1)^{a-1}e^{-b\theta}$ for $\theta > 0$, for the posterior it follows

$$\pi(\theta) \propto p(\boldsymbol{x}|\theta)p(\theta) \propto (\theta+1)^n(\theta+2)^n e^{\theta \sum_{i=1}^n \log x_i}(\theta+1)^{a-1}e^{-b\theta} =$$

$$= (\theta+1)^{n+a-1}(\theta+2)^n e^{-(b-\sum_{i=1}^n \log x_i)\theta}.$$

Part ii

The posterior mode is defined as

$$m_n = \arg\max_{\theta} \pi(\theta).$$

To be able to find the values for which $\pi(\theta)$, $\theta > 0$ is maximised, we should solve

$$\frac{d}{d\theta}\pi(\theta) = 0$$

or for simplicity (the maximum of $\pi(\theta)$, $\theta > 0$ is the same as the maximum of $\log \pi(\theta)$, $\theta > 0$):

$$\frac{d}{d\theta}\log \pi(\theta) = 0$$

$$\frac{d}{d\theta}(n+a-1)\log(\theta+1) + n\log(\theta+2) - (b - \sum_{i=1}^n \log x_i)\theta = 0$$

$$\frac{n+a-1}{\theta+1} + \frac{n}{\theta+2} - b + \sum_{i=1}^n \log x_i = 0$$

$$(b - \sum_{i=1}^n \log x_i)\theta^2 - (2n+a-1-3b+3\sum_{i=1}^n \log x_i)\theta - (3n+2a-2-2b+2\sum_{i=1}^n \log x_i) = 0.$$

The quadratic equation has two analytical solutions, and let's substitute $S = \sum_{i=1}^n \log x_i$:

$$\theta_{1,2} = \frac{2n+a-1-3b+3S \pm \sqrt{(2n+a-1-3b+3S)^2 + 4(b-S)(3n+2a-2-2b+2S)}}{2(b-S)}.$$

We are interested in real solutions for $\theta > 0$, and for m_n we should choose this value for which the solution is greater than 0.

Part iii

The function $g(\theta) = e^{-\frac{(\theta-1)^2}{2}}$ is positive for every θ. Let's define

$$h(\theta) = \log p(\boldsymbol{x}|\theta) + \log p(\theta) = (n + a - 1)\log(\theta+1) + n\log(\theta+2) + \theta\sum_{i=1}^{n}\log x_i - b\theta + \sum_{i=1}^{n}\log(1-x_i),$$

$$\tilde{h}(\theta) = \log g(\theta) + \log p(\boldsymbol{x}|\theta) + \log p(\theta) = -\frac{(\theta-1)^2}{2} + h(\theta).$$

The Laplace approximation equation for $E_\pi\{g(\theta)\}$ is

$$E_\pi\{g(\theta)\} = \frac{\int_\Theta \tilde{h}(\theta)d\theta}{\int_\Theta h(\theta)d\theta} \approx \frac{|H_n|^{\frac{1}{2}} g(\tilde{m}_n) p_{\boldsymbol{x}|\theta}(\boldsymbol{x}|\tilde{m}_n) p_\theta(\tilde{m}_n)}{|\tilde{H}_n|^{\frac{1}{2}} p_{\boldsymbol{x}|\theta}(\boldsymbol{x}|m_n) p_\theta(m_n)},$$

where

$$m_n = \arg\max_\theta \pi(\theta),$$
$$\tilde{m}_n = \arg\max_\theta \pi(\theta)g(\theta).$$

We have

$$g(\tilde{m}_n) = e^{-\frac{(\tilde{m}_n-1)^2}{2}},$$
$$p_{\boldsymbol{x}|\theta}(\boldsymbol{x}|\tilde{m}_n) = (\tilde{m}_n+1)^n(\tilde{m}_n+2)^n e^{\tilde{m}_n\sum_{i=1}^n \log x_i}(1-x_1)...(1-x_n),$$
$$p_{\boldsymbol{x}|\theta}(\boldsymbol{x}|m_n) = (m_n+1)^n(m_n+2)^n e^{m_n\sum_{i=1}^n \log x_i}(1-x_1)...(1-x_n),$$

leading to

$$\frac{p_{\boldsymbol{x}|\theta}(\boldsymbol{x}|\tilde{m}_n)}{p_{\boldsymbol{x}|\theta}(\boldsymbol{x}|m_n)} = \left(\frac{\tilde{m}_n+1}{m_n+1}\right)^n\left(\frac{\tilde{m}_n+2}{m_n+2}\right)^n e^{(\tilde{m}_n-m_n)\sum_{i=1}^n\log x_i} = \left(\frac{\tilde{m}_n+1}{m_n+1}\right)^n\left(\frac{\tilde{m}_n+2}{m_n+2}\right)^n\prod_{i=1}^n x_i^{\tilde{m}_n-m_n}.$$

Also,

$$p_\theta(\tilde{m}_n) \propto (\tilde{m}_n+1)^{a-1}e^{-b\tilde{m}_n}$$
$$p_\theta(m_n) \propto (m_n+1)^{a-1}e^{-bm_n},$$

or

$$\frac{p_\theta(\tilde{m}_n)}{p_\theta(m_n)} = \left(\frac{\tilde{m}_n+1}{m_n+1}\right)^{a-1}e^{-b(\tilde{m}_n-m_n)}.$$

For now, we have simplified $E_\pi\{g(\theta)\}$ to

$$E_\pi\{g(\theta)\} \approx \frac{|H_n|^{\frac{1}{2}}}{|\tilde{H}_n|^{\frac{1}{2}}}\left(\frac{\tilde{m}_n+1}{m_n+1}\right)^{a+n-1}\left(\frac{\tilde{m}_n+2}{m_n+2}\right)^n e^{-b(\tilde{m}_n-m_n)-\frac{(\tilde{m}_n-1)^2}{2}}\prod_{i=1}^n x_i^{\tilde{m}_n-m_n}.$$

It remains to simplify the Hessians, which in this particular case are just the second derivatives of $h(\theta)$ and $\tilde{h}(\theta)$:

$$\frac{d}{d\theta}h(\theta) = \frac{n+a-1}{\theta+1} + \frac{n}{\theta+2} + \sum_{i=1}^{n}\log x_i - b$$

$$\frac{d^2}{d\theta^2}h(\theta) = -\frac{n+a-1}{(\theta+1)^2} - \frac{n}{(\theta+2)^2},$$

therefore

$$H_n = \frac{d^2}{d\theta^2}(-h(\theta)) = -\frac{d^2}{d\theta^2}h(\theta) = \frac{n+a-1}{(\theta+1)^2} + \frac{n}{(\theta+2)^2},$$

and this is always positive for $n \in N$, $a > 0$. Therefore $|H_n| = H_n$, $n \in N$, $a > 0$.

Now, for $\tilde{h}(\theta)$:

$$\tilde{h}(\theta) = -\frac{(\theta-1)^2}{2} + h(\theta),$$

$$\frac{d}{d\theta}\tilde{h}(\theta) = 1 - \theta + \frac{d}{d\theta}h(\theta),$$

$$\frac{d^2}{d\theta^2}\tilde{h}(\theta) = -1 + \frac{d^2}{d\theta^2}h(\theta),$$

$$\tilde{H}_n = \frac{d^2}{d\theta^2}(-\tilde{h}(\theta)) = -\frac{d^2}{d\theta^2}\tilde{h}(\theta) = 1 + H_n = 1 + \frac{n+a-1}{(\theta+1)^2} + \frac{n}{(\theta+2)^2},$$

which is again always positive for $n \in N$, $a > 0$, because $H_n > 0$ for $n \in N$, $a > 0$, and hence $|\tilde{H}_n| = \tilde{H}_n$ for $n \in N$, $a > 0$. Therefore,

$$\frac{|H_n|^{\frac{1}{2}}}{|\tilde{H}_n|^{\frac{1}{2}}} = \left(\frac{H_n}{\tilde{H}_n}\right)^{\frac{1}{2}}.$$

Finally, we simplified $E_\pi g(\theta)$ to

$$E_\pi g(\theta) \approx \left(\frac{H_n}{\tilde{H}_n}\right)^{\frac{1}{2}} \left(\frac{\tilde{m}_n+1}{m_n+1}\right)^{a+n-1} \left(\frac{\tilde{m}_n+2}{m_n+2}\right)^{n} e^{-b(\tilde{m}_n-m_n)-(\tilde{m}_n-1)^2/2} \prod_{i=1}^{n} x_i^{\tilde{m}_n-m_n}.$$

Part b

Part i

```python
def posterior_mode(x: List[float], a: int, b: int):
  n = len(x)
  S = np.sum(np.log(x))

  A = b - S
  B = -(2*n + a - 1 - 3*b + 3*S)
  C = -(3*n + 2*a - 2 - 2*b + 2*S)
  solution_1 = (-B + np.sqrt(B**2 - 4*A*C)) / (2*A)
  solution_2 = (-B - np.sqrt(B**2 - 4*A*C)) / (2*A)
  solution = [solution for solution in [solution_1, solution_2] if solution > 0]

  return solution[0]
```

```
m_n = posterior_mode(x=[0.4, 0.5, 0.6], a=1, b=1)
print(f"m_n = {m_n}")
```

```
## m_n = 0.5451546455691932
```

Part ii

We want to solve

$$\tilde{m}_n = \arg\max_{\theta} \pi(\theta)g(\theta)$$

```
from scipy.optimize import minimize

def eval_func(theta: float, x: List[float], a: int, b: int):
  n = len(x)
  S = np.sum(np.log(x))

  pi_theta = ((theta + 1) ** (n + a - 1)) * ((theta + 2) ** n) * np.exp(-(b - S) * theta)
  g_theta = np.exp(-((theta-1)**2)/2)
  evaluate_func = pi_theta * g_theta

  return -evaluate_func # because we have to maximize by using the minimize function

def tilde_m(x: List[float], a: int, b: int):
  bnds = [(0, np.inf)]
  res = minimize(eval_func, x0=1, args=(x, a, b), bounds=bnds)

  return res.x[0]
```

```
tilde_m_n = tilde_m(x=[0.4, 0.5, 0.6], a=1, b=1)
print(f"tilde_m_n = {tilde_m_n}")
```

```
## tilde_m_n = 0.7228391900583504
```

Part iii

```
def laplace_approx(x: List[float], a: int, b: int):
  n = len(x)
  m_n = posterior_mode(x, a, b)
  m_tilde_n = tilde_m(x, a, b)

  H_n = (n+a-1)/((m_n+1)**2) + n/((m_n + 2)**2)
  H_tilde_n = 1 + (n+a-1)/((m_tilde_n+1)**2) + n/((m_tilde_n + 2)**2)

  term_1 = (H_n / H_tilde_n)**0.5
  term_2 = ((m_tilde_n + 1) / (m_n + 1)) ** (n + a - 1)
  term_3 = ((m_tilde_n + 2) / (m_n + 2)) ** n
  term_4 = np.exp(-b*(m_tilde_n - m_n) - (1/2)*((m_tilde_n - 1)**2))
  term_5 = np.prod(np.power(x, m_tilde_n - m_n))

  return term_1 * term_2 * term_3 * term_4 * term_5
```

```
g_laplace = laplace_approx(x=[0.4, 0.5, 0.6], a=1, b=1)
print(f"g_laplace = {g_laplace}")
```

```
## g_laplace = 0.791604138431483
```

Part iv

The target distribution is

$$\pi(\theta) \propto (\theta+1)^{n+a-1}(\theta+2)^n e^{-(b-\sum_{i=1}^n \log x_i)\theta},$$

and our aim is to sample from it. For the purpose we should choose a transition density, I have chosen a $Gamma(1,1)$ distribution.

```
def target_distribution(theta: float, x: List[float], a: int, b: int):
  """
  Pi(theta)
  """
  n = len(x)
  s = np.sum(np.log(x))

  term_1 = (theta + 1) ** (n + a - 1)
  term_2 = (theta + 2) ** n
  term_3 = np.exp(-(b - s) * theta)

  return term_1 * term_2 * term_3

def mcmc_mh(x: List[float], a: int, b: int, num_iteration: int = 10_000):
    np.random.seed(1)
    # collections
    samples = []
    accepted = 0

    # initialisation of the sample
    starting_value_theta = 0
    samples = [starting_value_theta]
    current_theta = starting_value_theta

    # MCMC sampler
    for _ in range(num_iteration):
        proposal_theta = st.gamma.rvs(a=1, loc=0, scale=1)
        u = st.uniform.rvs()
        target_ratio_prop = target_distribution(proposal_theta, x, a, b)
        target_ratio_curr = target_distribution(current_theta, x, a, b)

        proposal_ratio_prop = st.gamma.pdf(proposal_theta, a=1,loc=0, scale=1)
        proposal_ratio_curr = st.gamma.pdf(current_theta, a=1, loc=0, scale=1)

        ratio = (target_ratio_prop * proposal_ratio_prop) / (target_ratio_curr * proposal_ratio_curr)
        acceptance_ratio = min(1, ratio)

        if u < acceptance_ratio:
            samples.append(proposal_theta)
            current_theta = proposal_theta
            accepted += 1
        else:
            samples.append(current_theta)

    # Calculate acceptance rate
    acceptance_rate = accepted / num_iteration
    estimate = np.mean(np.exp(-(np.array(samples) - 1)**2 / 2))

    return acceptance_rate, estimate
```

```
acceptance_rate, estimate = mcmc_mh(x=[0.4, 0.5, 0.6], a=1, b=1)
print("Monte Carlo estimate:", estimate)
```

```
## Monte Carlo estimate: 0.810911104373303
```

```
print("Acceptance rate:", acceptance_rate)
```

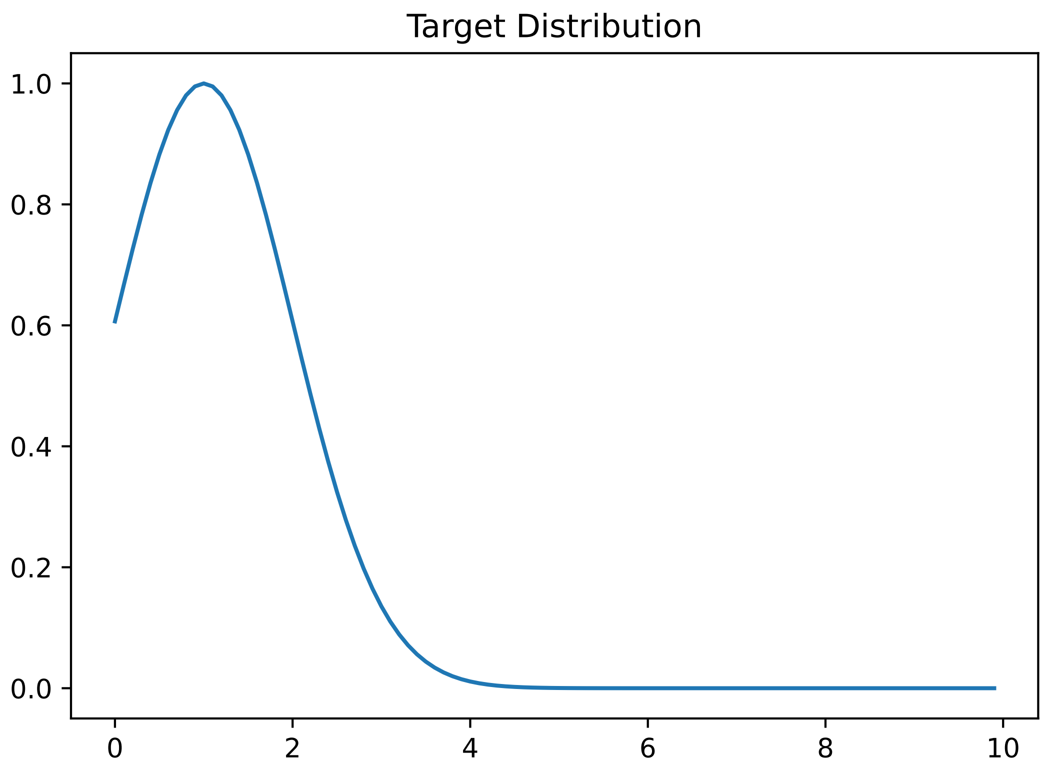
```
## Acceptance rate: 0.6431
```

The result from the Laplace approximation is ~ 0.79, and the result from MCMC MH with the set seed is ~ 0.81. The difference between Lapalce and MCMC for this specific set of chosen transition distirbution and set seed is not big.

In the MCMC approach, the choice of transition density is critical because it can affect the efficiency and covergence of the algorithm. The distribution is commonly chosen such that it is proportional to the target distribution, but if it is too different from the target it may lead to low acceptance rate, slow convergence, and if it is too similar to the targed distribution this would lead to high acceptance rate and poor use of the parameter space. This leads us to experimenting with different distributions and their parameters' tuning.

For the Lapalce approximation, being an analytic approximate method, we used fomrula (1) from the problem statement. This formula was constructed via integration that uses a second-order application of Taylor's theorem to approximate positive function integrands with normal distribution densities. Hence, the Laplace approximation might be better suited for target distributions more closely resembling normal distribution densities, generally, for simple and symmetric target distributions. Whereas, the MCMC MH method is good for more complex scenarios.

If we plot $\pi(\theta)$, for $\theta > 0$, and $x = (0.4, 0.5, 0.6)$, $a = 1$, $b = 1$



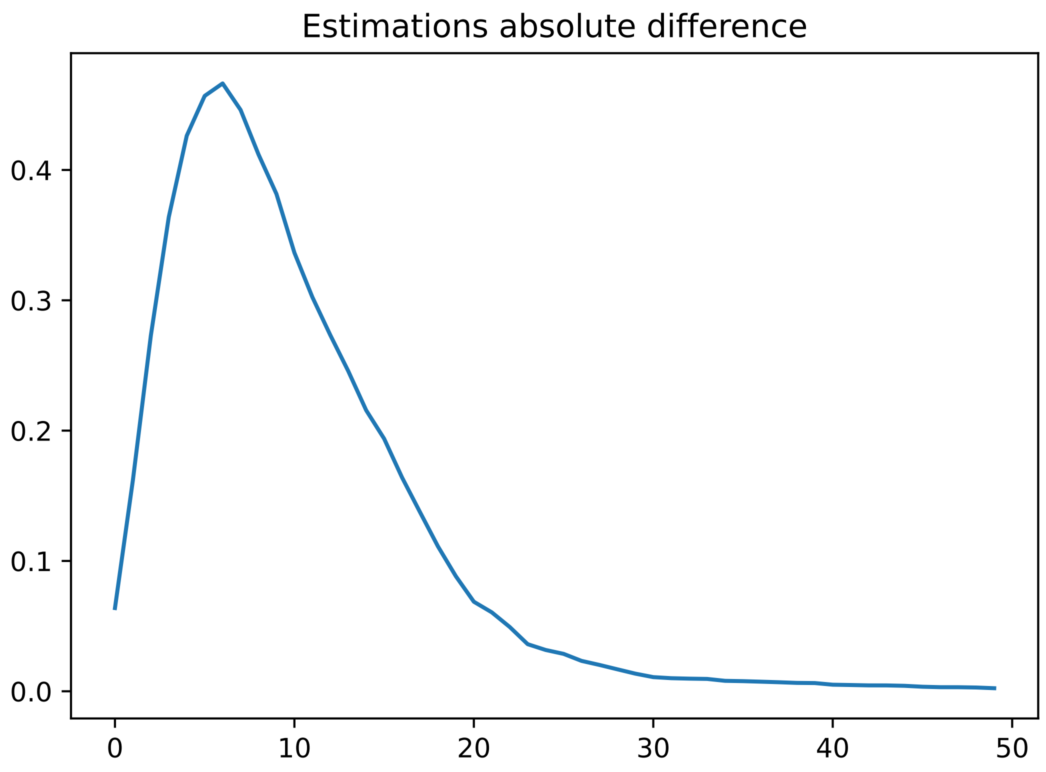
we can observe that for $\theta > 0$ we have a relatively simple and smooth curve, suggesting that in this specific case Laplace approximation would be more reliable than MCMC.

Part v

My suggestion would be that the methods would converge when both of them are approaching more and more closely a normal distribution. This would happen with increasing the number of observations in the sample x. Let's start with the initial set $x = (0.4, 0.5, 0.6)$, $a = 1$, $b = 1$, and do 50 iterations during which of them we are going to expand the x observations with one element drawn from the standard uniform distribution, and compute the absolute value of the differences, and plot them.

```
np.random.seed(13)
test_set = [0.4, 0.5, 0.6]
diffs = []
for i in range(50):
  test_set += [st.uniform.rvs()]
  g_laplace = laplace_approx(x=test_set, a=1, b=1)
  _, estimate = mcmc_mh(x=test_set, a=1, b=1)
  diffs.append(np.abs(estimate - g_laplace))

plt.plot(diffs);
plt.title("Estimations absolute difference");
plt.show();
```



The plot confirms that the more the number of observations in the range $(0, 1)$ become, the smaller the absolute difference between the two estimates are.