GRADUATE STUDENT STAT 840 A2

Vsevolod Ladtchenko 20895137

Problem 8

a)

```
fz = function(z)
{
  t1 = 1.5
  t2 = 2
  p1 = z^(-3/2)
  p2 = exp(-t1*z -t2/z )# +2*sqrt(t1*t2) +log(sqrt(2*t2)))
  return(p1*p2)
}

accept_ind_MH = function(x,y,gam_shap,gam_rate)
{
  r1 = fz(y) / fz(x)
  r2 = dgamma(x, gam_shap, gam_rate) / dgamma(y, gam_shap, gam_rate)
  return(min(r1*r2, 1))
}

independent_MH = function(gam_shap,gam_rate)
{
  n = 1000
  chain = rep(NA,n)
  chain[1] = 1

  for (i in 2:n)
  {
    x = chain[i-1]
    y = rgamma(1, gam_shap, gam_rate)
    u = runif(1)

    if (u <= accept_ind_MH(x, y,gam_shap,gam_rate))
      chain[i] = y
    else
      chain[i] = x
  }

  return(chain)
}

optimize_gam_params = function()
{
```

```
  nn = 10 # consider params 1..10 for alpha, beta
  real = theoretical_result()

  rez = matrix(nrow=nn*nn, ncol=5)
  colnames(rez) = c('gam_shape','gam_rate','diff','~E[Z]','~E[1/Z]')

  for (i in 1:nn)
  {
    for (j in 1:nn)
    {
      chain = independent_MH(gam_shap=i,gam_rate=j)
      samp = c(mean(chain), mean(1/chain))

      diff = mean((samp - real)^2) # MSE of sample vs real
      rez[(i-1)*nn+j,] = c(i,j,diff,samp)
    }
  }

  idx = which(rez[,3] == min(rez[,3])) # idx of smallest diff
  return(round(rez[idx,],4))
}

theoretical_result = function()
{
  t1 = 1.5
  t2 = 2
  real_z = sqrt(t2/t1)
  real_zinv = sqrt(t1/t2) + 1/(2*t2)
  rez = matrix(data = c(real_z, real_zinv), nrow=1, ncol=2)
  colnames(rez) = c('E[Z]','E[1/Z]')
  return(round(rez,4))
}

# running the optimization many times gives optimal parameters
# along the lines of (5,4) (7,5) (6,2) (6,9) and such.
theoretical_result()
```

```
##        E[Z] E[1/Z]
## [1,] 1.1547  1.116
```

```
optimize_gam_params()
```

```
## gam_shape  gam_rate      diff     ~E[Z]   ~E[1/Z]
##    7.0000    5.0000    0.0001    1.1667    1.1110
```

Mean Squared Error is used to compare simulation results against the theoretical result. This is done across different parameters of the Gamma distribution. The results are close but there is no set of parameters that consistently give optimal results, even for larger n.

b)

From Casella & Berger, p. 51, Theorem 2.1.5:

Let Z have pdf $f_Z(z)$ and let $W = g(z)$, where g is a monotone function. Suppose $f_Z(z)$ is continuous on its support, and g^{-1} has a continuous derivative on $g(\text{support of } f)$. Then the pdf of W on $g(\text{support of } f)$ is:

$$f_W(w) = f_Z(g^{-1}(w))\left|\frac{d}{dw}g^{-1}(w)\right|$$

Applying this to our problem:

$$f_z(z \mid \theta_1, \theta_2) \propto z^{-3/2}\exp\left(-\theta_1 z - \frac{\theta_2}{z} + 2\sqrt{\theta_1\theta_2} + \log\sqrt{2\theta_2}\right), z > 0$$

$W = g(z) = \log(z)$ is monotone.

f is continuous on $z > 0$.

$g^{-1} = \exp$ has continuous derivative everywhere.

$$\begin{aligned}
f_W(w) &= f_Z(g^{-1}(w))\left|\frac{d}{dw}g^{-1}(w)\right| \\
&= f_Z(e^w)\left|\frac{d}{dw}e^w\right| \\
&= f_Z(e^w)|e^w| \\
&= f_Z(e^w)e^w \\
&\propto (e^w)^{-3/2}\exp\left(-\theta_1(e^w) - \frac{\theta_2}{(e^w)} + 2\sqrt{\theta_1\theta_2} + \log\sqrt{2\theta_2}\right)e^w \\
&\propto e^{-3w/2}e^w\exp\left(-\theta_1 e^w - \theta_2 e^{-w} + 2\sqrt{\theta_1\theta_2} + \log\sqrt{2\theta_2}\right) \\
&\propto e^{-w/2}\exp\left(-\theta_1 e^w - \theta_2 e^{-w} + 2\sqrt{\theta_1\theta_2} + \log\sqrt{2\theta_2}\right) \\
&\propto \exp\left(-w/2 - \theta_1 e^w - \theta_2 e^{-w} + 2\sqrt{\theta_1\theta_2} + \log\sqrt{2\theta_2}\right) \\
&\propto \exp\left(-w/2 - \theta_1 e^w - \theta_2 e^{-w}\right)
\end{aligned}$$

```
fw = function(w)
{
  t1 = 1.5
  t2 = 2
  return(exp(-(w/2) -t1*exp(w) -t2*exp(-w)))
}

accept_rand_walk_MH = function(x,y)
{
  r = fw(y)/fw(x)
  return(min(r, 1))
}
```

```
random_walk_MH = function()
{
  n = 1000
  chain = rep(NA,n)
  chain[1] = 1

  for (i in 2:n)
  {
    x = chain[i-1]
    y = x + rnorm(1,0,1)
    u = runif(1)

    if (u <= accept_rand_walk_MH(x, y))
      chain[i] = y
    else
      chain[i] = x
  }
  z = exp(chain)
  rez = matrix(data = c(mean(z), mean(1/z)), nrow=1, ncol=2)
  colnames(rez) = c('~E[Z]','~E[1/Z]')
  return(round(rez,4))
}

theoretical_result()
```

```
##        E[Z] E[1/Z]
## [1,] 1.1547  1.116
```

```
random_walk_MH()
```

```
##       ~E[Z] ~E[1/Z]
## [1,] 1.1568  1.0777
```

The accuracy of the simulation appears good. It does have a bit of variability over different runs. Increasing n to 100,000 yields better precision. Another option is to change the proposal exploration parameter, which is σ of the Normal distribution in this case. Changing this parameter does not seem to yield more precise estimates on average.