Computational Statistics

Homework 2

Romane PERSCH

November 18, 2016

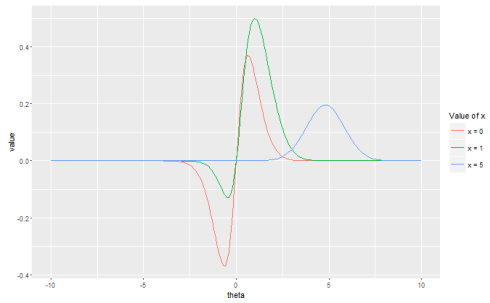
3.1 Normal Cauchy Bayes Estimator Part 1

(a) Classic Monte Carlo Ratio Estimation

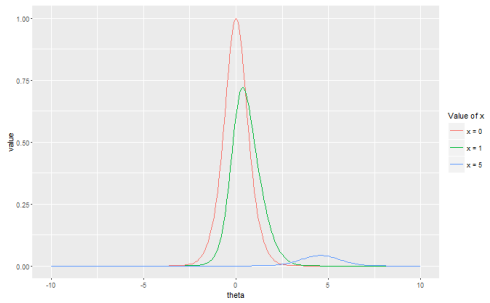
We want to estimate $\delta(x) = \frac{\int \frac{\theta}{1+\theta^2} e^{-(x-\theta)^2/2} d\theta}{\int \frac{1}{1+\theta^2} e^{-(x-\theta)^2/2} d\theta}$. Recognizing that $\delta(x) = \frac{E^{\pi}(\frac{\theta}{1+\theta^2})}{E^{\pi}(\frac{1}{1+\theta^2})}$ where $\pi(\theta|x)$ is a $N(x,1)$ distribution, this suggests simulating $\theta_1, ..., \theta_m \sim N(x,1)$ and estimating $\delta(x)$ with :

$$\hat{\delta^{\pi}_m} = \frac{\sum_{i=1}^{m} \frac{\theta_i}{1+\theta_i^2}}{\sum_{i=1}^{m} \frac{1}{1+\theta_i^2}}$$

Using the Law of Large Numbers and Slutsky theorem, we indeed get the consistency of the estimator : $\hat{\delta^{\pi}_m} \xrightarrow[n\to\infty]{\mathcal{P}} \delta(x)$.

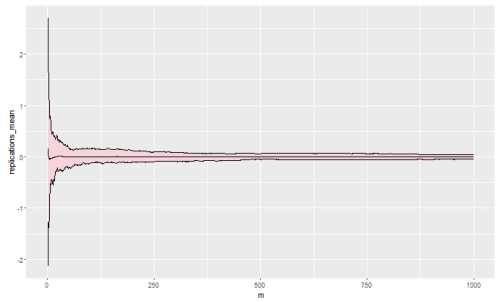


(a) Integrand of the numerator

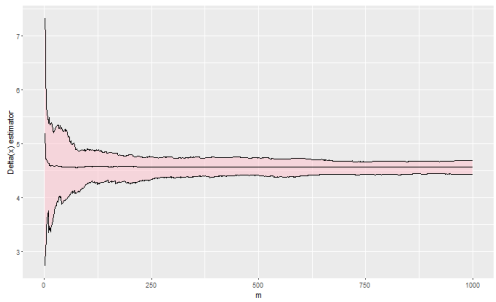


(b) Integrand of the denominator

Figure 1 – Integrand plot for several x



(a) x = 0



(b) x = 5

Figure 2 – Range of estimators $\hat{\delta^{\pi}_m}$ for 100 runs

The corresponding R code is available in Appendix Listing 2.

(b) Variance monitoring

Let $g(x,y) = \frac{y}{x}$.

Let $U = \begin{pmatrix} \frac{1}{1+\theta^2} \\ \frac{\theta}{1+\theta^2} \end{pmatrix}$, $E^\pi(U) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ and $\overline{U} = \begin{pmatrix} \frac{1}{m}\sum_{i=1}^m \frac{1}{1+\theta_i^2} \\ \frac{1}{m}\sum_{i=1}^m \frac{\theta_i}{1+\theta_i^2} \end{pmatrix}$.

Central Limit Theorem implies :

$\sqrt{m}(\overline{U} - \begin{pmatrix} \alpha \\ \beta \end{pmatrix}) \underset{m\to\infty}{\overset{\mathcal{D}}{\to}} N(0,\Sigma)$ where Σ is the variance-covariance matrix of U.

Thus, using Delta-method with g :

$$\sqrt{m}(g(\overline{U}) - g(\alpha,\beta)) \underset{m\to\infty}{\overset{\mathcal{D}}{\to}} N(0, \nabla g^T(\alpha,\beta)\Sigma\nabla g(\alpha,\beta))$$

This is equivalent to:

$$\sqrt{m}(\delta_m^\pi - \delta(x)) \underset{m\to\infty}{\overset{\mathcal{D}}{\to}} N(0, \nabla g^T(\alpha,\beta)\Sigma\nabla g(\alpha,\beta))$$

We will therefore use this result to build a 0.95 Confidence Interval for our estimator δ_m^π.

$$\nabla g(\alpha,\beta) = \begin{pmatrix} -\frac{\beta}{\alpha^2} \\ \frac{1}{\alpha} \end{pmatrix}$$

Thus :

$$\nabla g^T(\alpha,\beta)\Sigma\nabla g(\alpha,\beta) = \begin{pmatrix} -\frac{\beta}{\alpha^2}\Sigma_{11} + \frac{1}{\alpha}\Sigma_{21} & -\frac{\beta}{\alpha^2}\Sigma_{12} + \frac{1}{\alpha}\Sigma_{22} \end{pmatrix} \begin{pmatrix} -\frac{\beta}{\alpha^2} \\ \frac{1}{\alpha} \end{pmatrix} \tag{1}$$

$$= \frac{\beta^2}{\alpha^4}\Sigma_{11} - 2\frac{\beta}{\alpha^3}\Sigma_{12} + \frac{1}{\alpha^2}\Sigma_{22} \tag{2}$$

Using Koenig-Huygens variance formula :

$$= \frac{1}{\alpha^2}[\frac{\beta^2}{\alpha^4}(E((\frac{1}{1+\theta^2})^2) - \alpha^2) - 2\frac{\beta}{\alpha^3}(E(\frac{1}{1+\theta^2}\frac{\theta}{1+\theta^2}) - \alpha\beta) + E((\frac{\theta}{1+\theta^2})^2) - \beta^2] \tag{3}$$

$$= \frac{1}{\alpha^2}[\frac{\beta^2}{\alpha^4}E((\frac{1}{1+\theta^2})^2) - 2\frac{\beta}{\alpha^3}E(\frac{1}{1+\theta^2}\frac{\theta}{1+\theta^2}) + E((\frac{\theta}{1+\theta^2})^2)] \tag{4}$$

As $\delta(x) = \frac{\beta}{\alpha}$:

$$= \frac{1}{\alpha^2}E((\frac{\theta}{1+\theta^2} - \delta(x)\frac{1}{1+\theta^2})^2) \tag{5}$$

This suggests to estimate $\nabla g^T(\alpha,\beta)\Sigma\nabla g(\alpha,\beta)$ with $\hat{V} = \frac{1}{m}\frac{1}{\hat{\alpha_m}}\sum_{i=1}^m(\frac{\theta_i}{1+\theta_i^2} - \hat{\delta}_m^\pi\frac{1}{1+\theta_i^2})^2$ (where $\hat{\alpha_m} = \frac{1}{m}\sum_{i=1}^m\frac{1}{1+\theta_i^2}$). Using a decomposition for $\hat{V}$ similar to the one at line (4), the Law of Large Numbers and Slutsky theorem, we can indeed easily show that $\hat{V} \underset{n\to\infty}{\overset{\mathcal{P}}{\to}} \nabla g^T(\alpha,\beta)\Sigma\nabla g(\alpha,\beta)$.

Thus, using Slutsky theorem :

$$\sqrt{m}\frac{(\hat{\delta}_m^\pi - \delta(x))}{\sqrt{\hat{V}}} \underset{m\to\infty}{\overset{\mathcal{D}}{\to}} N(0,1)$$

From this, we can infer a 0.95 Confidence Interval for $\delta(x)$:

$$\delta(x) \in [\hat{\delta}_m^\pi - 1.96\frac{\sqrt{\hat{V}}}{\sqrt{m}}, \hat{\delta}_m^\pi + 1.96\frac{\sqrt{\hat{V}}}{\sqrt{m}}]$$

Therefore, in order to get a 3 digit accuracy, we need to stop the simulations at the smallest m such that :

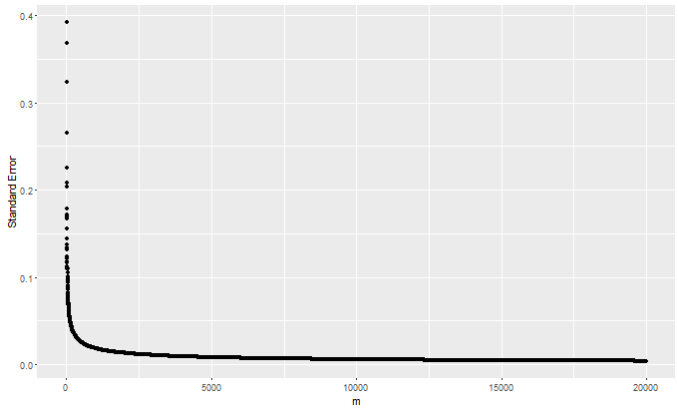
$$1.96\sqrt{\frac{\hat{V}}{m}} \leq 0.001$$

Experiment For computational reasons, I stopped the simulations when I reached a 2 digit accuracy (as after 500 000 simulations, a 3 digit accuracy was still not reached).

The experiment for $x = 0$ shows us that we reach a 2 digit accuracy after 13890 simulations. The estimated Standard Error at this point is 0.0051. The corresponding 0.95 Confidence Interval is : $\boxed{\delta(0) \in [-0.006, 0.014]}$

The corresponding R code is available in Appendix Listing 2.

Figure 3 – Standard Error $\sqrt{\frac{\hat{V}}{m}}$ evolution when x = 0



2.29 Accept-Reject in the Bayesian context

Before starting Problem 3.2, we prove the validity of the Accept-Reject algorithm in the Bayesian context, when densities are only known up to multiplicative factor. This will help in Problem 3.2 Question (a).

Preliminary result : Accept-Reject when the density of interest is only known up to a multiplicative factor

Let $\tilde{f}$ be the density of interest and assume we only know $\tilde{f}$ up to a multiplicative factor, i.e : $\tilde{f} \propto f$, or in other words $\exists c, \tilde{f} = cf$.

Using Corollary 2.17, if there is a density g and a constant M satisfying $\tilde{f}(x) \leq Mg(x)$ (as $\tilde{f}$ and g are exact densities, we necessarily have : $M \geq 1$), then in order to simulate $X \sim \tilde{f}$, it is sufficient to generate :

$$Y \sim g \quad \text{and} \quad U|Y = y \sim U(0, Mg(y)),$$

$$\text{until} \quad 0 < u < \tilde{f}(y)$$

Which is also equivalent to the more classic form of the algorithm :

$$Y \sim g \quad \text{and} \quad U|Y = y \sim U(0, 1),$$

$$\text{until} \quad 0 < Mg(y)u < \tilde{f}(y) \quad \Leftrightarrow \quad 0 < u < \frac{\tilde{f}(y)}{Mg(y)}$$

As $Mg(y)U|Y = y \sim U(0, Mg(y))$.

However, this cannot be directly implemented as we do not know the exact form of $\tilde{f}$, so we will not be able to find M such that $\tilde{f}(x) \leq Mg(x)$.

Now assume we are able to find M' and an exact density g such that $f(x) \leq M'g(x)$, as we know f exactly. Then :

$$cf(x) \leq cM'g(x) \quad \Leftrightarrow \quad \tilde{f}(x) \leq cM'g(x)$$

. Using Corollary 2.17, in order to simulate $X \sim \tilde{f}$, it is sufficient to generate:

$$Y \sim g \quad \text{and} \quad U|Y = y \sim U(0, 1),$$

$$\text{until} \quad 0 < u < \frac{\tilde{f}(y)}{cM'g(y)} \quad \Leftrightarrow \quad 0 < u < \frac{cf(y)}{cM'g(y)} \quad \Leftrightarrow \quad 0 < u < \frac{f(y)}{M'g(y)}$$

.

So we see that we can implement the algorithm without knowing c. We therefore showed that Accept-Reject is still valid when the density of interest is only known up to a multiplicative factor. Note that in this case, we will not have necessarily $M' \geq 1$, since the "real" $M = cM' \geq 1$.

(a) Application to the Bayesian context

Let x be fixed. We aim at simulating $\theta \sim \pi(\theta|x)$, the posterior density, using Accept-Reject algorithm and the prior $\pi(\theta)$ as the candidate density.

Using Bayes theorem : $\pi(\theta|x) \propto \pi(\theta)l(x|\theta)$. In other words, we only know $\pi(\theta|x)$ up to a multiplicative factor.

Assume a MLE (Maximum Likelihood Estimator) exists, i.e : we can find a value θ^{MLE} that maximizes $l(x|\theta)$ on Θ. Then:

$$\forall \theta \in \Theta, l(x|\theta) \leq l(x|\theta^{MLE})$$

Note that $l(x|\theta^{MLE})$ is then a constant wrt θ : it depends only on x. We can therefore call it M_x. Thus :

$$\pi(\theta)l(x|\theta) \leq M_x\pi(\theta)$$

. We are now in the exact context of the Preliminary Result, with $\tilde{f} = \pi(\theta|x)$, $f = \pi(\theta)l(x|\theta)$, $g = \pi(\theta)$ and $M' = M_x$.

Using the Preliminary Result, we thus can use the Accept-Reject algorithm to simulate θ according to its posterior distribution $\pi(\theta|x)$ (x be fixed) by using the prior $\pi(\theta)$ as the candidate density and taking the bound M' as the likelihood function evaluated at the MLE (providing that a maximum exists).

(b) Normal Cauchy example : see Problem 3.2 (a)

3.2 Normal Cauchy Bayes Estimator Part 2

(a) Posterior simulation using Accept-Reject algorithm

This question suggests a different method to estimate $\delta(x)$ using Accept-Reject algorithm to simulate a posterior distribution. First, let's write $\delta(x)$ as an expectation in a Bayesian context :

$$\begin{aligned}
\delta(x) &= \frac{\int \frac{\theta}{1+\theta^2} e^{-(x-\theta)^2/2} d\theta}{\int \frac{1}{1+\theta^2} e^{-(x-\theta)^2/2} d\theta} \\
&= \int \theta \pi(\theta|x) d\theta \quad \text{where} \quad \pi(\theta|x) \propto \frac{1}{1+\theta^2} e^{-(x-\theta)^2/2} \\
&= E^{\pi(\theta|x)}(\theta)
\end{aligned}$$

We indeed recognize that $\pi(\theta|x) \propto \pi(\theta)l(x|\theta)$ where the prior $\pi(\theta)$ is a $Cauchy(0,1)$ and the likelihood is the density of a $N(\theta,1)$. In other words, $\theta \sim Cauchy(0,1)$ and $X|\theta \sim N(\theta,1)$. $\delta(x)$ is thus the expectation of θ given $X = x$.

So the suggested method to estimate $\delta(x)$ here is to first simulate θ under its posterior distribution (using the Accept-Reject algorithm and Problem 2.29) and then to estimate $\delta(x)$ by the empirical mean.

Posterior simulation Using Problem 2.29, in order to simulate under the posterior, we can use the Accept-Reject algorithm with the prior (here a $Cauchy(0,1)$) as the candidate distribution and the likelihood evaluated at the MLE as the bound M'.

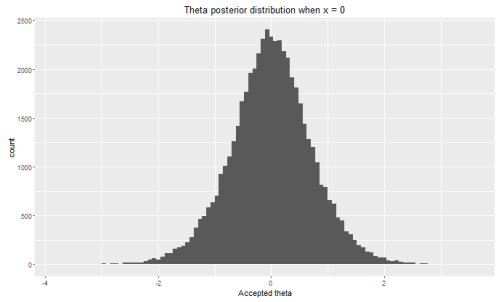
Here, $l(x|\theta) = \frac{1}{\sqrt{2\pi}} e^{-(x-\theta)^2/2}$. Thus, the MLE is $\theta^{MLE} = x$ and $l(x|\theta^{MLE}) = M_x = \frac{1}{\sqrt{2\pi}}$. Note that we do not necessarily have $M_x \geq 1$, as explained in Problem 2.29. Only the "real" $M = cM_x \geq 1$ where c is the constant $\int \pi(\theta)l(x|\theta)d\theta$, which is unknown.

x be fixed, we therefore use the algorithm :

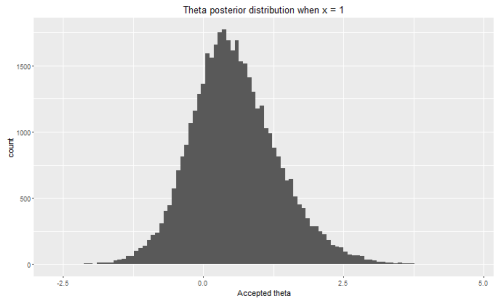
$$Y \sim Cauchy(0,1) \quad \text{and} \quad U|Y = y \sim U(0,1),$$

$$\text{until} \quad 0 < u < \frac{l(x|y)\pi(y)}{M_x\pi(y)} \quad \Leftrightarrow \quad 0 < u < \frac{l(x|y)}{M_x} \quad \Leftrightarrow \quad 0 < u < e^{-(x-y)^2/2}$$

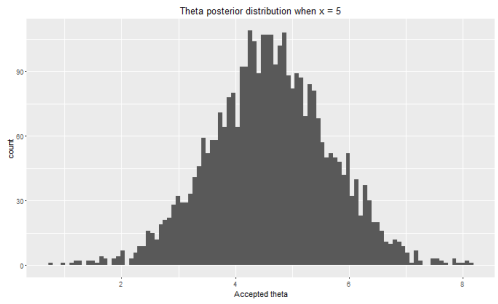
Experiment $\mathbf{x = 0}$ **:**Among the 100 000 simulations of θ according to the prior, we accept 52 227 of them, so we reach a 52% acceptance rate, which is not bad.
$\mathbf{x = 1}$ **:**Among the 100 000 simulations of θ according to the prior, we accept 41 455 of them, so we reach a 41% acceptance rate.
$\mathbf{x = 5}$ **:**Among the 100 000 simulations of θ according to the prior, we accept 3 400 of them, so we only reach a 3.4% acceptance rate. Therefore, depending on the value of x, this second method may not be computationally efficient.



(a) x = 0

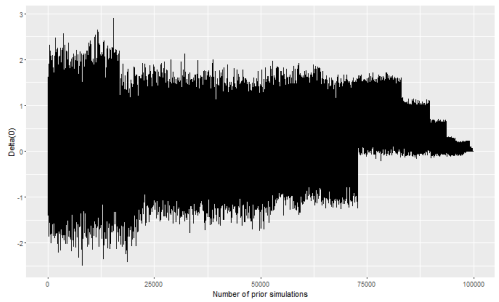


(b) x = 1

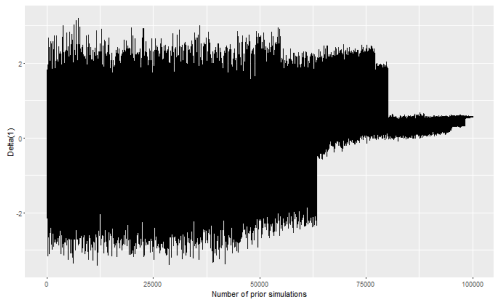


(c) x = 5

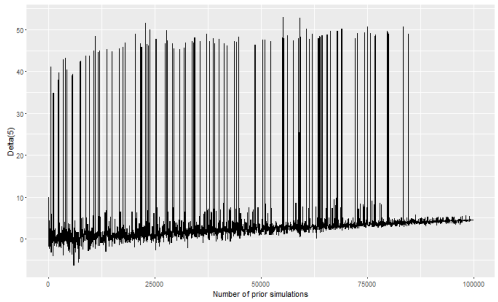
Figure 4 – Posterior distribution for different values of x



(a) x = 0



(b) x = 1



(c) x = 5

Figure 5 – $\delta(x)$ estimation for different values of x using Accept-Reject

On Figure 5, we show the evolution of the value of the estimator of $\delta(x)$(i.e : the empirical mean of the accepted θ) with the total number of simulations according to the prior (here 100 000). When a new θ is rejected, the estimator of $\delta(x)$ keeps the value of the previous one.

We see on Figure 5 that a lot of extreme values of θ are accepted, which leads to a high volatility of the estimator, especially when x is far from 0.

The corresponding R code is available in Appendix Listing 3.

(b) Monte Carlo error comparison between two experiments

Back to the method used in Problem 3.1, we now want to compare the Standard Error between an experiment using the same random variables θ_i in the numerator and in the denominator of $\hat{\delta}_m^\pi$ and an experiment using different random variables.

To achieve this, we design this experiment on R :

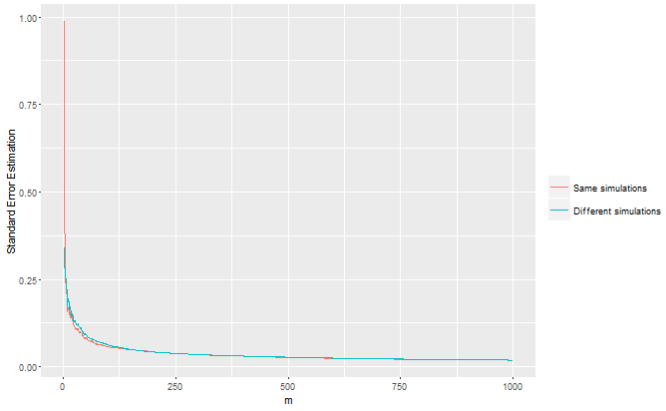
```
#### Question (b)

m_max = 1000
x = 0

theta <- theta_sim(m_max,x)
first_part <- sapply(theta, transform_num)
second_part <- sapply(theta, transform_den)
evolving_var_samesim = c()
for (k in 2:m_max){
  delta_sd <- standard_error_estimation(first_part[1:k], second_part[1:k])[2]
  evolving_var_samesim <- c(evolving_var_samesim, delta_sd)
}

theta2 <- theta_sim(m_max,x)
second_part2 <- sapply(theta2, transform_den)
evolving_var_diffsim = c()
for (k in 2:m_max){
  delta_sd <- standard_error_estimation(first_part[1:k], second_part2[1:k])[2]
  evolving_var_diffsim <- c(evolving_var_diffsim, delta_sd)
}

compare_var_df <- data.frame(
  m = seq(2,m_max,1),
  same_sim = evolving_var_samesim,
  diff_sim = evolving_var_diffsim)

compare_var_df_long <- melt(compare_var_df, id="m")

ggplot(data=compare_var_df_long,
       aes(x=m, y=value, colour=variable, xlab='m')) +
  geom_line() + ylab('Standard Error Estimation') + scale_colour_discrete(name  ="
      ",
                                        breaks=c("same_sim", "diff_sim"),
                                        labels=c("Same simulations", "Different
                                            simulations"))
```

Listing 1 – Problem 3.2

Note that **we must not set a seed**, otherwise the simulations in theta and in theta2 will be the same !

Figure 6 shows that there is no significant difference between the Standard Errors. When we re run the programme several times, we always achieve this result, even when m is small : sometimes the error obtained with the same simulations is higher and sometimes the other one is higher. From a certain point (on average $m = 250$), this difference always disappear. So it seems there is no need to simulate $2m$ variables : we will rather prefer the experiment using the same random variables.

Figure 6 – Standard Error comparison



3.4 Expected Value Calculation

(a) $X \sim N(0, \sigma^2)$

Assume $X \sim N(0, \sigma^2)$.

$$\begin{aligned} E(e^{-X^2}) &= \int_{-\infty}^{+\infty} e^{-x^2} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx \\ &= \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-(x^2 + \frac{x^2}{2\sigma^2})} dx \\ &= \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-(1+2\sigma^2)\frac{x^2}{2\sigma^2}} dx \end{aligned}$$

Using Integration by substitution with linear substitution $u = \sqrt{1+2\sigma^2}x$:

(which yields to $du = \sqrt{1+2\sigma^2}dx$)

$$E(e^{-X^2}) = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} \frac{1}{\sqrt{1+2\sigma^2}} e^{-\frac{u^2}{2\sigma^2}} dx$$

Recognizing the integral of the density of a $N(0, \sigma^2)$:

$$\boxed{E(e^{-X^2}) = \frac{1}{\sqrt{1+2\sigma^2}}}$$

(b) $X \sim N(\mu, \sigma^2)$

Assume $X \sim N(\mu, \sigma^2)$.

$$\begin{aligned} E(e^{-X^2}) &= \int_{-\infty}^{+\infty} e^{-x^2} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \\ &= \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}((2\sigma^2+1)x^2 - 2x\mu + \mu^2)} dx \end{aligned}$$

Using $(\sqrt{1+2\sigma^2}x - \frac{\mu}{\sqrt{1+2\sigma^2}})^2 = (2\sigma^2+1)x^2 - 2x\mu + \frac{\mu^2}{1+2\sigma^2}$:

$$\begin{aligned} E(e^{-X^2}) &= \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}[(\sqrt{1+2\sigma^2}x - \frac{\mu}{\sqrt{1+2\sigma^2}})^2 - \frac{\mu^2}{1+2\sigma^2} + \mu^2]} dx \\ &= e^{-\frac{\mu^2}{2\sigma^2}(1 - \frac{1}{1+2\sigma^2})} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(\sqrt{1+2\sigma^2}x - \frac{\mu}{\sqrt{1+2\sigma^2}})^2} dx \end{aligned}$$

Using Integration by substitution with linear substitution $u = \sqrt{1+2\sigma^2}x$:

$$E(e^{-X^2}) = e^{-\frac{\mu 2}{1+2\sigma^2}} \frac{1}{\sqrt{1+2\sigma^2}} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(u-\frac{\mu}{\sqrt{1+2\sigma^2}})^2} dx$$

Recognizing the integral of the density of a $N(\frac{\mu}{\sqrt{1+\sigma^2}}, \sigma^2)$:

$$\boxed{E(e^{-X^2}) = e^{-\frac{\mu 2}{1+2\sigma^2}} \frac{1}{\sqrt{1+2\sigma^2}}}$$

3.22 Importance Resampling

(a) Weights Renormalization

Assume $w_i = f(X_i)/g(X_i)$. Then, in expectation, we get :

$$E(\sum_{i=1}^{n} w_i) = \sum_{i=1}^{n} E(\frac{f(X_i)}{(X_i)})$$

As $X_1, ..., X_n$ are iid and sampled according to g:

$$\begin{aligned} &= nE(\frac{f(X_1)}{(X_1)}) \\ &= n \int \frac{f(x)}{g(x)} g(x) dx \\ &= n \int f(x) dx \end{aligned}$$

As f is a density :

$$\boxed{E(\sum_{i=1}^{n} w_i) = n}$$

However, in general :

$$\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} \frac{f(X_i)}{(X_i)} \neq n$$

So even if we know f and g exactly (including their normalizing constants), if we want to resample using the w_i weights, we cannot directly use w_i/n. Those weights will indeed not define a probability distribution, since their sum is not equal to 1.

We need to renormalize them to $w_i^* = \frac{w_i}{\sum_{i=1}^{n} w_i}$ so that $\sum_{i=1}^{n} w_i^* = 1$, both almost sure and in expectation.

(b) Resampling with replacement

Assuming the weights w_i have been renormalized to w_i^*, we sample with replacement n points $\tilde{X_j}$ from the X_i using those weights. That is to say :

Conditionnally to $X_1, ..., X_n$, $\tilde{X_j}$ follows a discrete distribution defined by :

$$\forall i, P(F = x_i | X_1 = x_1, ..., X_n = x_n) = w_i^*$$

In other words, $\tilde{X_j}$'s conditional distribution on $(X_1, ..., X_n)$ is equal to :

$$\sum_{i=1}^{n} w_i^* \delta_{X_i}$$

Thus :

$$E(\tilde{X_j} | X_1, ..., X_n) = \sum_{i=1}^{n} h(\tilde{X_j}) w_i^* \delta_{X_i}(\tilde{X_j})$$

$$E(\tilde{X_j} | X_1, ..., X_n) = \sum_{i=1}^{n} w_i^* h(X_i)$$

So :

$$\begin{aligned} E(\frac{1}{n}\sum_{j=1}^{n} h(\tilde{X}_j)) &= \frac{1}{n}\sum_{j=1}^{n} E(h(\tilde{X_j})) \\ &= \frac{1}{n}\sum_{j=1}^{n} E(E(h(\tilde{X}_j)|X_1, ..., X_n)) \\ &= \frac{1}{n}\sum_{j=1}^{n} E(\sum_{i=1}^{n} w_i^* h(X_i)) \end{aligned}$$

$$\boxed{E(\frac{1}{n}\sum_{j=1}^{n} h(\tilde{X}_j)) = E(\sum_{i=1}^{n} w_i^* h(X_i))}$$

(c) Bias

If the above formula is satisfied without the need of renormalization, that is if $w_i^* = w_i/n$, then we can show that the resampling process does not lead to the introduction of any bias. We can indeed show that the empirical distribution of the $\tilde{X}_j$'s remains unbiased. (However, we should keep in mind that this cannot not be reached in the general case. In the general case, we only keep the consistency of the resampled estimator $\frac{1}{n}\sum_{j=1}^{n} h(\tilde{X_j})$. In other words, it still converges in probability to the integral of interest $\int h(x)f(x)dx$.)

Indeed, assuming (b) is satisfied for $w_i^* = w_i/n$, then for any integrable function h :

$$\begin{aligned} E(\frac{1}{n}\sum_{j=1}^{n} h(\tilde{X}_j)) &= E(\frac{1}{n}\sum_{i=1}^{n} w_i h(X_i)) \\ &= \frac{1}{n}\sum_{i=1}^{n} E(\frac{f(X_i)}{g(X_i)} h(X_i)) \end{aligned}$$

As the X_i are iid and sampled according to g :

$$\begin{aligned} E(\frac{1}{n}\sum_{j=1}^{n} h(\tilde{X}_j)) &= E(\frac{f(X_1)}{g(X_1)} h(X_1)) \\ &= \int \frac{f(x)}{g(x)} h(x)g(x)dx \end{aligned}$$

$$\boxed{E(\frac{1}{n}\sum_{j=1}^{n} h(\tilde{X}_j)) = \int h(x)f(x)dx}$$

So $\frac{1}{n}\sum_{j=1}^{n} h(\tilde{X}_j)$ is an unbiased estimator of the integral of interest. In particular, using $h_t(x) = 1_{x \leq t}$:

$$\forall t \in \mathbb{R}, E(\frac{1}{n}\sum_{j=1}^{n} 1_{\tilde{X}_j \leq t}) = \int 1_{x \leq t} f(x)dx$$

In other words, the empirical distribution function of the $\tilde{X}_j$'s $\tilde{G}_n$ is an unbiased estimator of the cumulative distribution function F of the target distribution f : $\boxed{\forall t \in \mathbb{R}, E(\tilde{G}_n(t)) = F(t)}$

Appendix

```
1  ################################## Problem 3.1 ###################################
2  ##################################################################################
3
4  #### Question (a)
5
6  #Plot the integrand
7  integrand_num ← function(theta,x){
8    result ← (theta / (1 + theta**2))*exp(-0.5*((x - theta)**2))
```

```
  return(result)
}

integrand_den <- function(theta,x){
  result <- (1 / (1 + theta**2))*exp(-0.5*((x - theta)**2))
  return(result)
}

x = 0

library("reshape2")
library("ggplot2")


num_df <-
  data.frame(
    theta = seq(-10,10,0.1),
    numerator1 = sapply(seq(-10,10,0.1),integrand_num, x = 0),
    numerator2 = sapply(seq(-10,10,0.1),integrand_num, x = 1),
    numerator3 = sapply(seq(-10,10,0.1),integrand_num, x = 5)
  )

den_df <-
  data.frame(
    theta = seq(-10,10,0.1),
    denominator1 = sapply(seq(-10,10,0.1),integrand_den, x = 0),
    denominator2 = sapply(seq(-10,10,0.1),integrand_den, x = 1),
    denominator3 = sapply(seq(-10,10,0.1),integrand_den, x = 5)
  )


num_df_long <- melt(num_df, id="theta")  # convert to long format
den_df_long <- melt(den_df, id="theta")

ggplot(data=num_df_long,
       aes(x=theta, y=value, colour=variable, xlab='theta')) +
  geom_line() + scale_colour_discrete(name  ="Value of x",
                                      breaks=c("numerator1", "numerator2", "
                                          numerator3"),
                                      labels=c("x = 0", "x = 1", "x = 5"))

ggplot(data=den_df_long,
       aes(x=theta, y=value, colour=variable, xlab='theta')) +
  geom_line() + scale_colour_discrete(name  ="Value of x",
                                       breaks=c("denominator1", "denominator2", "
                                           denominator3"),
                                       labels=c("x = 0", "x = 1", "x = 5"))

# Monte Carlo Simulation of Delta(x)
transform_num <- function(u){
  return(u / (1 + u**2))
}

transform_den <- function(u){
  return(1 / (1 + u**2))
}

theta_sim <- function(m,x){
  return(rnorm(m, mean = x, sd = 1))
}

delta_estimator <-function(theta){
  #Returns Delta(x) estimator using the vector of theta simulations
  #Note that we use the same theta simulations to compute the numerator and the
      denominator
  delta_x_num <- sum(sapply(theta, transform_num))
  delta_x_den <- sum(sapply(theta, transform_den))
```

```
  return(delta_x_num / delta_x_den)
}

m_max = 1000

x = 5
replications_sim = list()
replications_est = list()
for (i in 1:100){
  replications_sim[[i]] <- theta_sim(m_max,x)
  est_vector = c()
  for (k in 1:m_max){
    delta <- delta_estimator(replications_sim[[i]][1:k])
    est_vector <- c(est_vector, delta)
  }
  replications_est[[i]] <- est_vector
}

replications_df <- h <- do.call(cbind, replications_est)
print(dim(replications_df))

replications_max <- apply(replications_df, 1, max)
replications_min <- apply(replications_df, 1, min)

df_plot <- data.frame(
  m = seq(1,m_max,1),
  replications_max = replications_max,
  replications_min = replications_min,
  replications_mean <- apply(replications_df, 1, mean)
)

ggplot(data = df_plot,aes(m, replications_mean))+
  geom_ribbon(aes(x=m, ymax=replications_max, ymin=replications_min), fill="pink",
      alpha=.5) +
  geom_line(aes(y = replications_max), colour = 'black') +
  geom_line(aes(y = replications_min), colour = 'black')+
  geom_line() + xlab('m') + ylab('Delta(x) estimator')

standard_error_estimation <- function(first_part,second_part){
  delta <- sum(first_part) / sum(second_part)
  sum_elements <- (first_part - delta * second_part)**2
  m = length(second_part)
  alpha <- mean(second_part)
  delta_variance <- (1/ m**2)*(1/alpha**2)*sum(sum_elements)
  delta_sd <- sqrt(delta_variance)
  return(c(delta, delta_sd))
}


m_max = 20000
x = 0

theta <- theta_sim(m_max,x)
first_part <- sapply(theta, transform_num)
second_part <- sapply(theta, transform_den)
evolving_var = c()
evolving_delta = c()
for (k in 2:m_max){
  results <- standard_error_estimation(first_part[1:k], second_part[1:k])
  evolving_var <- c(evolving_var, results[2])
  evolving_delta <- c(evolving_delta, results[1])
}

qplot(seq(2,m_max,1), evolving_var, xlab='m', ylab='Standard Error')
qplot(seq(2,m_max,1), evolving_delta, xlab='m', ylab='Delta(0)')


```

```
139
140 ### /!\ We only make a two-digits approximation for computational reasons
141 #Check when the standard error becomes lower than 0.01/1.96
142 test <- (evolving_var < (0.01/qnorm(0.975)))
143 min_m <- min(which(test == TRUE))
144 print(min_m + 1) #result : 13963
145 print(evolving_var[min_m])
146 print( evolving_delta[min_m])
147
148 #0.95 CI is then :
149 print(evolving_delta[min_m] + evolving_var[min_m]*qnorm(0.975))
150 print(evolving_delta[min_m] - evolving_var[min_m]*qnorm(0.975))
```

Listing 2 – Problem 3.1

```
1  ################################# Problem 3.2 ################################
2  ##############################################################################
3
4  #### Question (a)
5
6  quotient_f_g <- function(theta,x){
7    return(exp(-(x - theta)**2/2))
8  }
9
10
11 n_sim = 100000
12 x = 5
13
14 theta <- rcauchy(n_sim, location = 0, scale = 1)
15 unif <- runif(n_sim, min = 0, max = 1)
16 accepted <- ifelse(unif <= sapply(theta,quotient_f_g,x = x),TRUE, FALSE)
17 print(length(accepted))
18 summary(accepted)
19
20 qplot(theta[accepted], geom = 'histogram', xlab='Accepted theta', main = 'Theta
       posterior distribution when x = 1', bins = 100)
21 print(length(theta[accepted]))
22
23 #Estimation of Delta(x):
24 est_vector = c()
25 delta = 10 #arbitrary value initialization
26 for (k in 1:n_sim){
27   delta <- ifelse(accepted[k] == TRUE,mean(theta[accepted[1:k]]),delta)
28   est_vector <- c(est_vector, delta)
29 }
30
31 qplot(seq(1,1000,1),est_vector[1:1000], geom = 'line')
32
33 qplot(seq(1,100000,1),est_vector[1:100000], geom = 'line', ylab = 'Delta(5)', xlab
       = 'Number of prior simulations')
34 qplot(seq(1,length(accepted[accepted == TRUE]),1),est_vector[accepted == TRUE],
       geom = 'line')
35
36 #### Question (b)
37
38 m_max = 1000
39 x = 0
40
41 theta <- theta_sim(m_max,x)
42 first_part <- sapply(theta, transform_num)
43 second_part <- sapply(theta, transform_den)
44 evolving_var_samesim = c()
45 for (k in 2:m_max){
46   delta_sd <- standard_error_estimation(first_part[1:k], second_part[1:k])[2]
47   evolving_var_samesim <- c(evolving_var_samesim, delta_sd)
48 }
49
50 theta2 <- theta_sim(m_max,x)
```

```
second_part2 <- sapply(theta2, transform_den)
evolving_var_diffsim = c()
for (k in 2:m_max){
  delta_sd <- standard_error_estimation(first_part[1:k], second_part2[1:k])[2]
  evolving_var_diffsim <- c(evolving_var_diffsim, delta_sd)
}

compare_var_df <- data.frame(
  m = seq(2,m_max,1),
  same_sim = evolving_var_samesim,
  diff_sim = evolving_var_diffsim)

compare_var_df_long <- melt(compare_var_df, id="m")

ggplot(data=compare_var_df_long,
       aes(x=m, y=value, colour=variable, xlab='m')) +
  geom_line() + ylab('Standard Error Estimation') + scale_colour_discrete(name  ="
      ",
                                        breaks=c("same_sim", "diff_sim"),
                                        labels=c("Same simulations", "Different
                                            simulations"))
```

Listing 3 – Problem 3.2