Homework 2

Question 1

```r
n = 1000
X = runif(n, min= 0, max = pi/2)
gx = cos(X)
theta_hat = (pi/2 - 0)*mean(gx)
theta_hat
```

```
## [1] 0.9787064
```

The analytical value we get on performing the integral is 1 however on applying the above approach we get a value inbetween the range of 0.9 and 1.1 never getting 1 as the exact answer for theta_hat

```r
theta_hat_js = numeric(4)
xjs = seq (0, pi/2, length.out = (4+1) )
for (j in 1:4) {
  X = runif(n, min= xjs[j], max = xjs[j+1])
  gx = cos(X)
  theta_hat_js[j] = (xjs[j+1] - xjs[j])*mean(gx)
}
sum(theta_hat_js)
```

```
## [1] 1.002934
```

```r
mat1 = matrix(NA,100,2)
for (i in 1:100){
  X = runif(1000, min= 0, max = pi/2)
  gx = cos(X)
  mat1[i,1] = (pi/2 - 0)*mean(gx)
}
for (i in 1:100){
  theta_hat_js = numeric(4)
  xjs = seq (0, pi/2, length.out = (4+1) )
  for (j in 1:4) {
    X1 = runif(1000, min= xjs[j], max = xjs[j+1])
    gx1 = cos(X1)
    theta_hat_js[j] = (xjs[j+1] - xjs[j])*mean(gx)
  }
  mat1[i,2]= sum(theta_hat_js)
}
```

```
colMeans(mat1)
```

```
## [1] 1.0004446 0.9892209
```

```
round(apply(mat1, 2, sd),5)
```

```
## [1] 0.01388 0.00000
```

We notice that the stratified sampling method gives us a stanadard deviation close to 0 and thereby improves our efficiency in comparison to the method followed in 1a

Question 2 PDF is maximized when x=0 so we get 3/pi f(t)/g(t)= (3/pi)*(1/(pi/2)) = 2/3 = c 1/c = 3/2

```
Y = runif(1500,min= 0, max = pi/2)
U = runif(1500)
c = 3/2
accept = U < ((3/pi)*(1-(((2/pi)*Y)**2)))/c*(pi/2)
X_accept = Y[accept]
mean(accept)
```

```
## [1] 0.6713333
```

Since we want to ensure phi function remians positive we take x to be unif

```
theta_hat_1 = matrix (NA,100,1)
for (i in 1:100){
  gf_phix =  (cos(X_accept)*(X_accept>0 & X_accept<pi/2))/((3/pi)*(1-(((2/pi)*X_accept)**2)))
  theta_hat_1[i] = mean(gf_phix)
}
```

```
colMeans(theta_hat_1)
```

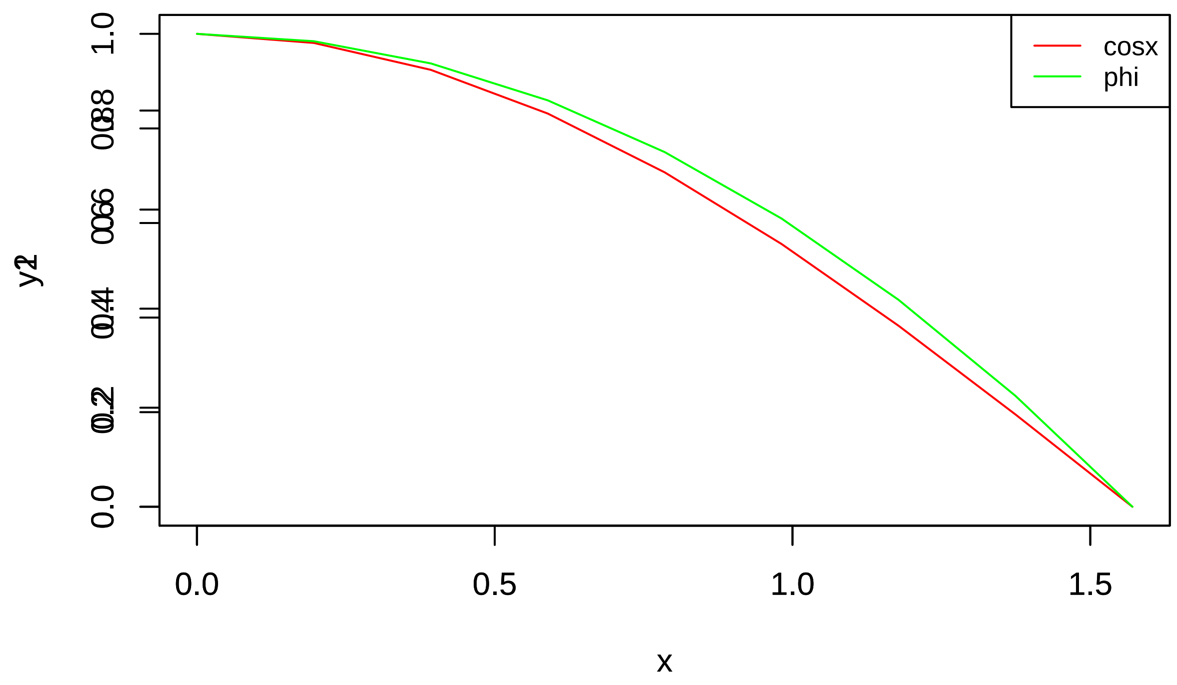
```
## [1] 1.000193
```

```
round(apply(theta_hat_1, 2, sd),6)
```

```
## [1] 0
```

we notice that the importance sampling estimator has a better efficiency as it has a smaller standard deviation in compariosn to simple Monte Carlo approach used in 1a

```
x = seq(0,pi/2,pi/16)
y1 = cos(x)
y2 = ((3/pi)*(1-(((2/pi)*x)**2)))
plot(x, y1, type = "l", col = "red")
par(new=TRUE)
plot(x, y2, type = "l", col = "green")
legend("topright", legend=c("cosx", "phi"), col=c("red", "green"),lty=1:1, cex=0.8,)
```



The plot shows that with the range of (0,pi/2) both the functions start and end at the same values however overall phi has a marginally greater value in comparison to cosx

Question 3 Ex5.2

```
func = function(x){
  y = runif(1000)
  if (x >= 0){
    gx = x*exp(-x^2*y^2/2)/sqrt(2*pi)+.5
  }
  else{
    gx = 0.5 - (x*exp(-x^2*y^2/2))/sqrt(2*pi)
  }
  mce = mean(gx)
  mce_var = (1/1000)*var(gx)
  mce_std = sqrt(mce_var)
  l = (mce - 1.96*mce_std)
  u = (mce + 1.96*mce_std)
  CI= c(l,u)
  c(mce, mce_var, CI)
}
```

```
a = func(2)
pnorm(2)
```

```
## [1] 0.9772499
```

```
a
```

```
## [1] 9.646351e-01 5.095711e-05 9.506438e-01 9.786264e-01
```

On comparing the pnorm with monte carlo estimate we see they both give a relatively similar solution (i.e. When last run pnorm(2)=0.97772 and mce=0.97770

Question 4 Ex5.13 We could take f1= e^(-x) and f2= truncated normal distribution

```
n = 10000
g = function(x){
  x^2*exp(-x^2/2)/sqrt(2*pi)*(x>1)
}
```

```
x3 = rexp(n)
gx3 = g(x3)/exp(-x3)
theta_hat_4 = mean(gx3)
sd_4 = sd(gx3)
```

```
#install.packages("truncnorm")
library(truncnorm)
```

```
x4 = rtruncnorm(n, a=1, b=Inf)
gx4 = g(x4)/dtruncnorm(x4, a=1, b=Inf)
theta_hat_5 = mean(gx4)
sd_5 = sd(gx4)
```

```
theta_hat_4
```

```
## [1] 0.4058526
```

```
theta_hat_5
```

```
## [1] 0.402355
```

```
ans=c(sd_4,sd_5)
ans
```

```
## [1] 0.5873150 0.2592553
```

as shown above f2 should be providing us with a smaller var therefore more efficient sampling since it has a similar pdf to g(x) and is within the same range

Question 5 Ex5.14

```
n = 10000
g = function(x){
  x^2*exp(-x^2/2)/sqrt(2*pi)*(x>1)
}
```

```r
x4 = rtruncnorm(n, a=1, b=Inf)
gx4 = g(x4)/dtruncnorm(x4, a=1, b=Inf)
theta_hat_5 = mean(gx4)
theta_hat_5
```

```
## [1] 0.3967447
```

```r
sd_5 = sd(gx4)
sd_5
```

```
## [1] 0.2565504
```