Homework 4

DATA604 Simulation and Modeling

Daniel Dittenhafer

March 22, 2016

1

In this problem, you will implement and investigate a series of variance reduction procedures for Monte Carlo method by estimating the expected value of a cost function c(x) which depends on a D-dimensional random variable x.

The cost function is:

$$c(x) = \frac{1}{(2\pi)^{\frac{D}{2}}} e^{-1/2x^T x}$$

where

$$x_i \sim U(-5, 5) \text{ for } i = 1..D$$

Goal: estimate E[c(x)] - the expected value of c(x) - using Monte Carlo methods and see how it compares to the real value, which you are able to find by hand.

```
# First define the cost function as an R function
costFx <- function(x)
{
  b <- exp(-0.5 * t(x) * x)
  D <- length(x)
  res <- (1 / ((2 * pi)^(D/2))) * b
  return (res)
}
```

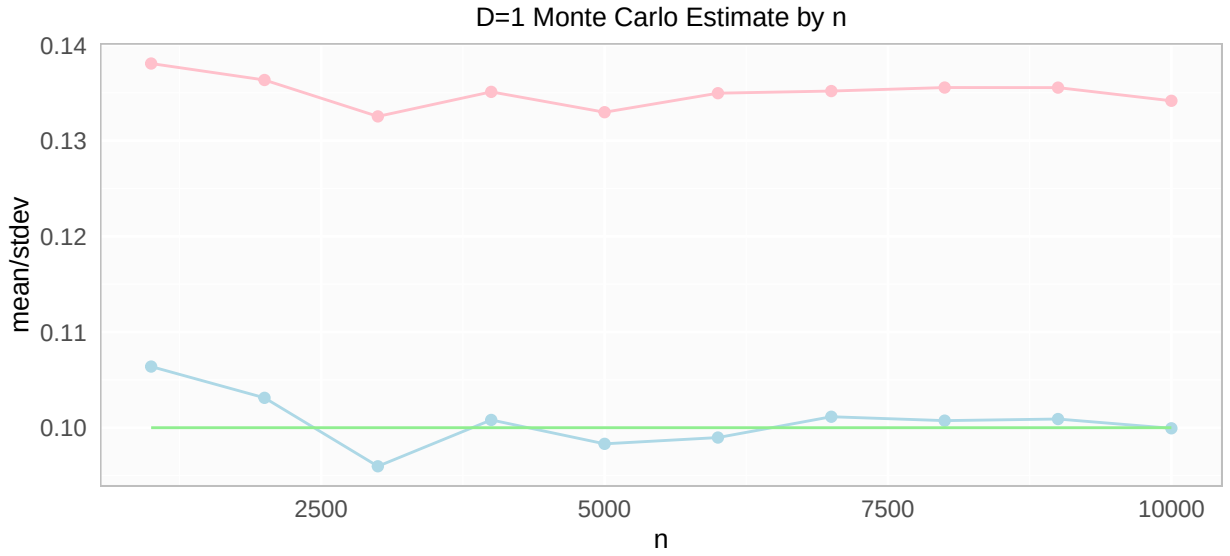
a) Crude Monte Carlo

```
crudeMC <- function(n, min, max, d = 1)
{
  # Need a loop in here
  theta.hat <- matrix(nrow=d, ncol=n)
  for(i in 1:n)
  {
    x <- runif(d, min, max)
    theta.hat[,i] <- costFx(x)
    #gXbar <- (1/n) * costFx(x)
    #print(((max-min) * gXbar))
    #theta.hat[,i] <- t((max-min) * gXbar)
  }

  return (theta.hat)
}

#crudeMC(10, -5, 5, 2)

crudeMc.Loop <- function(d, verbose=FALSE)
{
```

```
  crudeMc.result <- data.frame(mean=c(), stdev=c(), n=c())
  for(n in seq(1000, 10000, by=1000))
  {
    res <- crudeMC(n=n, min=-5, max=5, d=d)
    if(verbose)
    {
      print("Data")
      print(res)
      print("Mean")
      print(mean(res))
    }

    crudeMc.result <- rbind(crudeMc.result, data.frame(mean=mean(res), stdev=sd(res), n=n))
  }

  crudeMc.result$EcActual <- (1/10)^d
  crudeMc.result$CoefVari <- crudeMc.result$stdev / crudeMc.result$mean

  return (crudeMc.result)
}
```

In the code below, we call the crude Monte Carlo loop function, show the top entries and visualize the result for D=1. The blue line represents the mean value, pink is the standard deviation, and the green line is the analytical value for $E[c(x)] = (1/10)^D$.

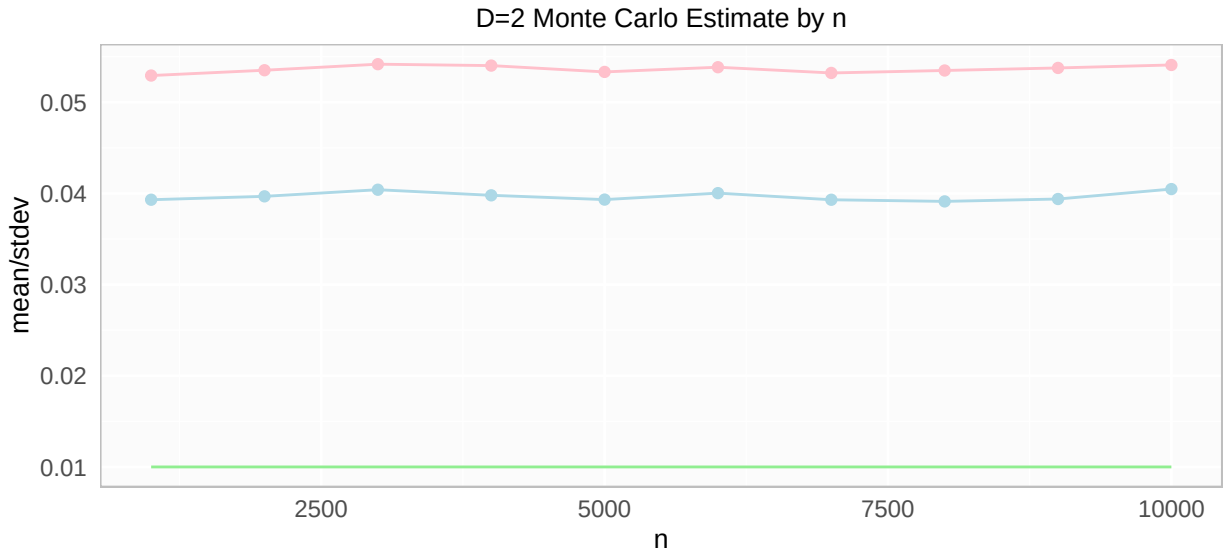
```
crudeMc.D1 <- crudeMc.Loop(d=1)
```



mean	stdev	n	EcActual	CoefVari
0.1063893	0.1380583	1000	0.1	1.297670
0.1031207	0.1363369	2000	0.1	1.322110
0.0959678	0.1325320	3000	0.1	1.381005
0.1008113	0.1350975	4000	0.1	1.340103
0.0983136	0.1329688	5000	0.1	1.352496
0.0989702	0.1349571	6000	0.1	1.363614
0.1011450	0.1351806	7000	0.1	1.336502
0.1007386	0.1355480	8000	0.1	1.345541
0.1009048	0.1355401	9000	0.1	1.343247
0.0999374	0.1341640	10000	0.1	1.342481

In the code below, we call the crude Monte Carlo loop function, show the top entries and visualize the result for D=2.

```
crudeMc.D2 <- crudeMc.Loop(d=2, verbose=FALSE)
```



mean	stdev	n	EcActual	CoefVari
0.0393111	0.0529333	1000	0.01	1.346524
0.0396823	0.0535150	2000	0.01	1.348587
0.0404087	0.0541799	3000	0.01	1.340798
0.0397943	0.0540244	4000	0.01	1.357590
0.0393262	0.0533305	5000	0.01	1.356107
0.0400347	0.0538463	6000	0.01	1.344992
0.0393077	0.0532173	7000	0.01	1.353866
0.0391286	0.0534884	8000	0.01	1.366990
0.0393905	0.0537673	9000	0.01	1.364982
0.0404797	0.0540931	10000	0.01	1.336303

Quasi-Random Numbers

```
sobolMC <- function(n, min, max, d = 1)
{
  # Need a loop in here
  theta.hat <- matrix(nrow=d, ncol=n)
  for(i in 1:n)
  {
    x <- sobol(n=1, d=d)
    theta.hat[,i] <- costFx(x)
    #gXbar <- (1/n) * costFx(x)
    #print(((max-min) * gXbar))
    #theta.hat[,i] <- t((max-min) * gXbar)
  }

  return (theta.hat)
}

sobolMC(10, 1)
```

```
##           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]
```

```
## [1,] 0.3520653 0.3520653 0.3520653 0.3520653 0.3520653 0.3520653 0.3520653
##           [,8]      [,9]     [,10]
## [1,] 0.3520653 0.3520653 0.3520653
```