HW7 Fixed Income

Haoxuan Tong, Yuhua Deng, Xiahao Wang, Nupur Solanki

May 28, 2019

Group: Haoxuan Tong, Yuhua Deng, Xiahao Wang, Nupur Solanki

Qn 1

```
suppressMessages(require(data.table))
suppressMessages(require(kableExtra))
```

```
## Warning: package 'kableExtra' was built under R version 3.5.2
```

```
suppressMessages(require(knitr))
rm(list=ls())

# correlation matrix
corrin <- as.data.frame(read.csv("Homework_7_corrin.csv", header = F))

# cholesky decomposition of the correlation matrix
corchol <- as.matrix(read.csv("Homework_7_corchol.csv", header = F))

# D(T)
dt <- as.data.frame(read.csv("Homework_7_pfilea.csv", header = F))

# sigma
sigma <- as.data.frame(read.csv("Homework_7_sigma.csv", header = F))

dt <- as.matrix(dt[1:20,], ncol=1)
sigma <- as.matrix(sigma[1:19,], ncol=1)

result <- matrix(0, nrow=20, ncol=20)
result[,1] <- dt
nSims <- 10000
deltat <- 0.5

for(i in (1:19)){
  rCur <- (1/result[i,i] -1)*2

  currStepNumber <- (19-i+1)

  # sigma for each step
  sigmaCurr <- sigma[1:currStepNumber]

  # dt for each step
  dtCurr <- result[(i+1):20, i]

  # dim (currStepNumber X 10000) matrix of rnorms
  zmatCurr <- matrix(rnorm(n=nSims * currStepNumber) ,nrow=currStepNumber)

  # use cholesky decomposition of the correlation matrix to obtain
```

Table 1: String Model Forward rates

	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
0.5	0.9724765	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
1	0.9445693	0.9714829	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
1.5	0.9163238	0.9424262	0.9697136	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
2	0.8879337	0.9132316	0.9395276	0.9682464	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
2.5	0.8597406	0.8846999	0.9100804	0.9378000	0.9691499	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
3	0.8320438	0.8562865	0.8805341	0.9076459	0.9380519	0.9694561	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
3.5	0.8050770	0.8286525	0.8522877	0.8787085	0.9081816	0.9386406	0.9681327	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
4	0.7789041	0.8017498	0.8242327	0.8494285	0.8784221	0.9081534	0.9366342	0.9680930	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
4.5	0.7534803	0.7757954	0.7971063	0.8216590	0.8496627	0.8783215	0.9059014	0.9361939	0.9668499	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
5	0.7287341	0.7504620	0.7711406	0.7949493	0.8221720	0.8496973	0.8765434	0.9062031	0.9354963	0.9675594	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
5.5	0.7045800	0.7253892	0.7451719	0.7682330	0.7941452	0.8207859	0.8468763	0.8756841	0.9036113	0.9344474	0.9663609	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
6	0.6810105	0.7009329	0.7199563	0.7421612	0.7672601	0.7930756	0.8182288	0.8461936	0.8728025	0.9024703	0.9335652	0.9671148	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
6.5	0.6580423	0.6773752	0.6960517	0.7177113	0.7424374	0.7672406	0.7917652	0.8188236	0.8444711	0.8735167	0.9035938	0.9363145	0.9684982	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
7	0.6356925	0.6546126	0.6724048	0.6932238	0.7172719	0.7410046	0.7648147	0.7907197	0.8155769	0.8431733	0.8725455	0.9040804	0.9350992	0.9667263	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
7.5	0.6139770	0.6320369	0.6490456	0.6690684	0.6925632	0.7158652	0.7390144	0.7637596	0.7877788	0.8142314	0.8423947	0.8729897	0.9026444	0.9335713	0.9653966	1.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
8	0.5929047	0.6100487	0.6265072	0.6460420	0.6687097	0.6910439	0.7133248	0.7370789	0.7602566	0.7862716	0.8136835	0.8436960	0.8723542	0.9027525	0.9334699	0.9667189	1.0000000	0.0000000	0.0000000	0.0000000	0
8.5	0.5724819	0.5886499	0.6045209	0.6232243	0.6450823	0.6665849	0.6884694	0.7112330	0.7335742	0.7585432	0.7850662	0.8140947	0.8417946	0.8713575	0.9010783	0.9330192	0.9642487	1.0000000	0.0000000	0.0000000	0
9	0.5527129	0.5685408	0.5838542	0.6016171	0.6223612	0.6433185	0.6644950	0.6864978	0.7079208	0.7318666	0.7575113	0.7856877	0.8126236	0.8419862	0.8707114	0.9015722	0.9315685	0.9666019	1.0000000	0.0000000	0
9.5	0.5336010	0.5487941	0.5635466	0.5803939	0.6003775	0.6205382	0.6412245	0.6624387	0.6834305	0.7065679	0.7313114	0.7584125	0.7841118	0.8125883	0.8404441	0.8700373	0.8989485	0.9325847	0.9639359	1.0000000	0
10	0.5151482	0.5299834	0.5442309	0.5606039	0.5795776	0.5989817	0.6190263	0.6398932	0.6602435	0.6822658	0.7060565	0.7321785	0.7569625	0.7840017	0.8109989	0.8396824	0.8675786	0.8999852	0.9303096	0.9639133	1

```
  # the matrix with dim(currStepNumber X 10000)
  corcholCur <- corchol[1:currStepNumber, 1:currStepNumber]
  # correlated brownian motion is equal to cholesky decomposition the standard norm random variables
  dZs <- corcholCur %*% zmatCurr

  tempResult <- rowMeans((dtCurr + rCur * dtCurr * deltat) + sigmaCurr * dZs * sqrt(deltat))

  tempResult <- c(rep(0, (i-1)),1, tempResult)
  result[,(i + 1)] <-  tempResult
}
result <- cbind(result , c(rep(0,19),1))
colnames(result) <- seq(0, 10,0.5)
rownames(result) <- seq(0.5,10,0.5)

kable(result, "latex", booktabs =T, caption = "String Model Forward rates", align ="l") %>% kable_stylin
```

Qn 2

Forward Par rates for 1 to 5 year semiannual coupon bonds 5 years forward

```
period <- c(1,2,3,4,5) * 2

inital_dt <- dt[10]

# the forward par rate based on the initial term structure
# at t =0, the price is 100
forward_par_rate <- sapply(period, function(x){
  2 * 100 * (inital_dt - dt[10 + x])/sum(dt[(10 + 1):(10+x)])
  })
q2_result <- as.matrix(forward_par_rate)
colnames(q2_result) <- "Forward Rate"
rownames(q2_result) <- paste0("Year-",1:5)

kable(q2_result, caption = "Forward par rates", align ="l")
```

Table 2: Forward par rates

	Forward Rate
Year-1	6.888558
Year-2	6.945151
Year-3	6.990333
Year-4	7.024829
Year-5	7.048864

Qn 3

```
payoff <- rep(0,10000)

# simulation for each path
for(n in 1:10000){

  q3_result <- matrix(0, nrow=20, ncol=20)
  q3_result[, 1] <- dt

  # simluate the DTs for a single iteration
  for(i in (1:19)){
    rCur <- (1/q3_result[i,i] -1)*2

    currStepNumber <- (19-i+1)

    # sigma for each step
    sigmaCurr <- sigma[1:currStepNumber]

    # dt for each step
    dtCurr <- q3_result[(i+1):20, i]

    # dim (currStepNumber X 10000) matrix of rnorms
    zmatCurr <- matrix(rnorm(n=1 * currStepNumber) ,nrow=currStepNumber)

    # use cholesky decomposition of the correlation matrix to obtain
    # the matrix with dim(currStepNumber X 10000)
    corcholCur <- corchol[1:currStepNumber, 1:currStepNumber]
    # correlated brownian motion is equal to cholesky decomposition the standard norm random variables
    dZs <- corcholCur %*% zmatCurr

    tempResult <- (dtCurr + rCur * dtCurr * deltat) + sigmaCurr * dZs * sqrt(deltat)

    tempResult <- c(rep(0, (i-1)),1, tempResult)
    q3_result[,(i + 1)] <-  tempResult
  }
  price <- rep(0, 5)
  # 1 year coupon paying at fifth year, 11th column
  price[1] <- (forward_par_rate[1]/2) * sum(q3_result[11:12,11]) + 100 * q3_result[12,11]
  price[2] <- (forward_par_rate[2]/2) * sum(q3_result[11:14,11]) + 100 * q3_result[14,11]
  price[3] <- (forward_par_rate[3]/2) * sum(q3_result[11:16,11]) + 100 * q3_result[16,11]
  price[4] <- (forward_par_rate[4]/2) * sum(q3_result[11:18,11]) + 100 * q3_result[18,11]
  price[5] <- (forward_par_rate[5]/2) * sum(q3_result[11:20,11]) + 100 * q3_result[20,11]
```

```r
  payoff[n] <- min(price)
}

futuer_price <- mean(payoff) * dt[10,1]
cat("The future price is : ", futuer_price,"\n")
```

```
## The future price is :  60.92467
```