Wstęp do matematyki finansów
Sprawozdanie 2

Justyna Niedźwiedzka 229877

24 stycznia 2021

Zadanie 1

W tym zadaniu porównamy rząd zbieżności dwóch metod numerycznych aproksymacji rozwiązania stochastycznego równania różniczkowego.

Rozważamy instrument podstawowy S opisany stochastycznym równaniem różniczkowym

$$dS_t = rS_t dt + \sigma S_t dB_t, \tag{1}$$

gdzie B to standardowy ruch Browna. Równanie (1) ma rozwiązanie postaci

$$S_t = S_0 \exp((r - \frac{\sigma^2}{2})t + \sigma B_t),$$

gdzie $r \geq 0$ to parametr związany z oprocentowaniem wolnego od ryzyka instrumentu finansowego, a $\sigma > 0$ określa zmienność procesu S.

Dla procesu X spełniającego

$$dX_t = a(X_t)dt + b(X_t)dB_t, \quad X_0 = x_0$$

rozważamy aproksymacje rozwiązania:

- metodą Eulera-Maruyamy, gdzie aproksymacja $\hat{X}$ w punktach czasowych $h, 2h, \dots, nh = T$ zadana jest rekurencyjnie (dla $i = 0, 1, \dots, n-1$)

  $$\hat{X}((i+1)h) = \hat{X}(ih) + a(\hat{X}(ih))h + b(\hat{X}(ih))\sqrt{h}Z_i,$$

  gdzie Z_i są niezależnymi zmiennymi losowymi o takim samym rozkładzie standardowym normalnym.

- metodą Milsteina, gdzie aproksymacja $\hat{X}$ w punktach czasowych $h, 2h, \dots, nh = T$ zadana jest rekurencyjnie (dla $i = 0, 1, \dots, n-1$)

  $$\hat{X}((i+1)h) = \hat{X}(ih) + a(\hat{X}(ih))h + b(\hat{X}(ih))\sqrt{h}Z_i + \frac{1}{2}b'\hat{X}(ih)b(\hat{X}(ih))h(Z_i^2 - 1),$$

  gdzie Z_i są niezależnymi zmiennymi losowymi o takim samym rozkładzie standardowym normalnym.

Rzędem metody numerycznej jest β w ograniczeniu

$$\mathbb{E}|\hat{X}(nh) - X(T)| \leq ch^{\beta}, \tag{2}$$

gdzie $\{\hat{X}(h), \hat{X}(2h), \dots, \hat{X}(nh)\}$ jest aproksymacją procesu X w punktach $h, 2h, \dots, nh = T$. Jest to tak zwany mocny rząd zbieżności β. Słaba zbieżność metody to β w ograniczeniu

$$|\mathbb{E}(\hat{X}(nh)) - \mathbb{E}(X(T))| \leq ch^{\beta} \tag{3}$$

dla dowolnego wielomianu g.

Poniżej przedstawiono funkcję, która dla zadanych parametrów r, σ, T, N, S_0 zwraca aproksymację rozwiązań równania (1). Funkcja zwraca aproksymację metodą Eulera-Maruyamy, metodą Milsteina, dokładny wynik, a także wartości oczekiwane dla obu metod dane wzorem (2). EX1 to wartość oczekiwana metody Eulera-Maruyamy, a EX2 to wartość oczekiwana metody Milsteina.

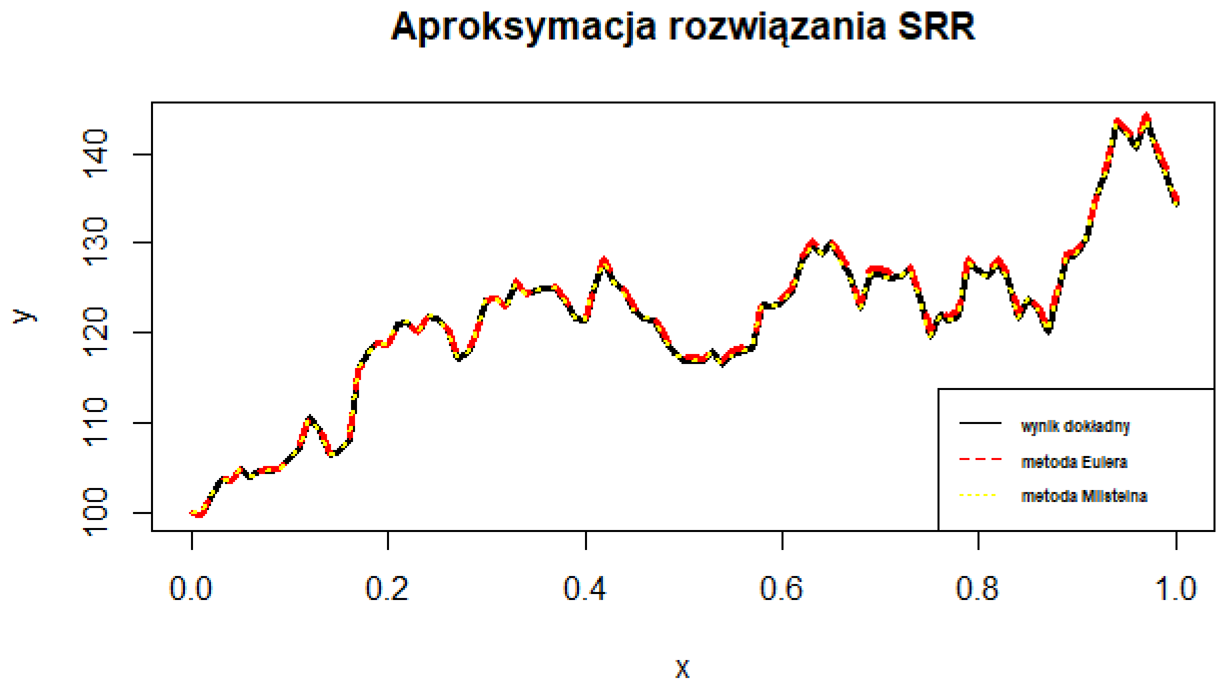
```
approximation <- function(r, sigma, T, N, S0){
  dt <- T/N
  t <- seq(from = 0, to = T, by = dt)
  Z <- rnorm(N,0,1)
  S_exact <- c()
  S_Euler <- c()
  S_Milstein <- c()
  S_exact[1] <- S0
  S_Euler[1] <- S0
  S_Milstein[1] <- S0
  for (i in 1:N){
    S_exact[i+1] <- S_exact[i]*exp((r - sigma^2/2)*dt + sigma*sqrt(dt)*Z[i])
    S_Euler[i+1] <- S_Euler[i]*(1+r*dt+sigma*sqrt(dt)*Z[i])
    S_Milstein[i+1] <- S_Milstein[i]*(1+r*dt+sigma*sqrt(dt)*Z[i]
                                    + 0.5*sigma^2*dt*(Z[i]^2 - 1))
  }
  EX1 = sum(abs(S_Euler - S_exact))/N
  EX2 = sum(abs(S_Milstein - S_exact))/N
  return(list(t, S_exact, S_Euler, S_Milstein, EX1, EX2))
}
```

Najpierw przeprowadzimy symulację dla parametrów $r = 0.05, \sigma = 0.2, T = 1, N = 100$ oraz $S_0 = 100$.

```
wyniki1 <- approximation(r = 0.05, sigma = 0.2, T = 1, N = 100, S0 = 100)
t <- wyniki1[[1]]
exact <- wyniki1[[2]]
Euler <- wyniki1[[3]]
Milstein <- wyniki1[[4]]
EX1 <- wyniki1[[5]]
EX2 <- wyniki1[[6]]
```

EX1 = 0.3064556, a EX2 = 0.0137850. Przedstawimy jeszcze symulację rozwiązań SSR na wykresie.

```
plot(t, exact, main = "Aproksymacja rozwiązania SRR", xlab = "x", ylab = "y",
     type = "l", col = "black", lwd = 3, lty = 1)
lines(t, Euler, col = "red", lwd = 3, lty = 2)
lines(t, Milstein, col = "yellow", lwd = 2, lty = 3)
legend("bottomright", legend = c("wynik dokładny", "metoda Eulera",
                                 "metoda Milsteina"),
       col = c("black", "red", "yellow"), cex = 0.5, text.font = 2,
       lty = c(1,2,3))
```



Rysunek 1: Aproksymacja rozwiązania stochastycznego równania różniczkowego zależnie od metody

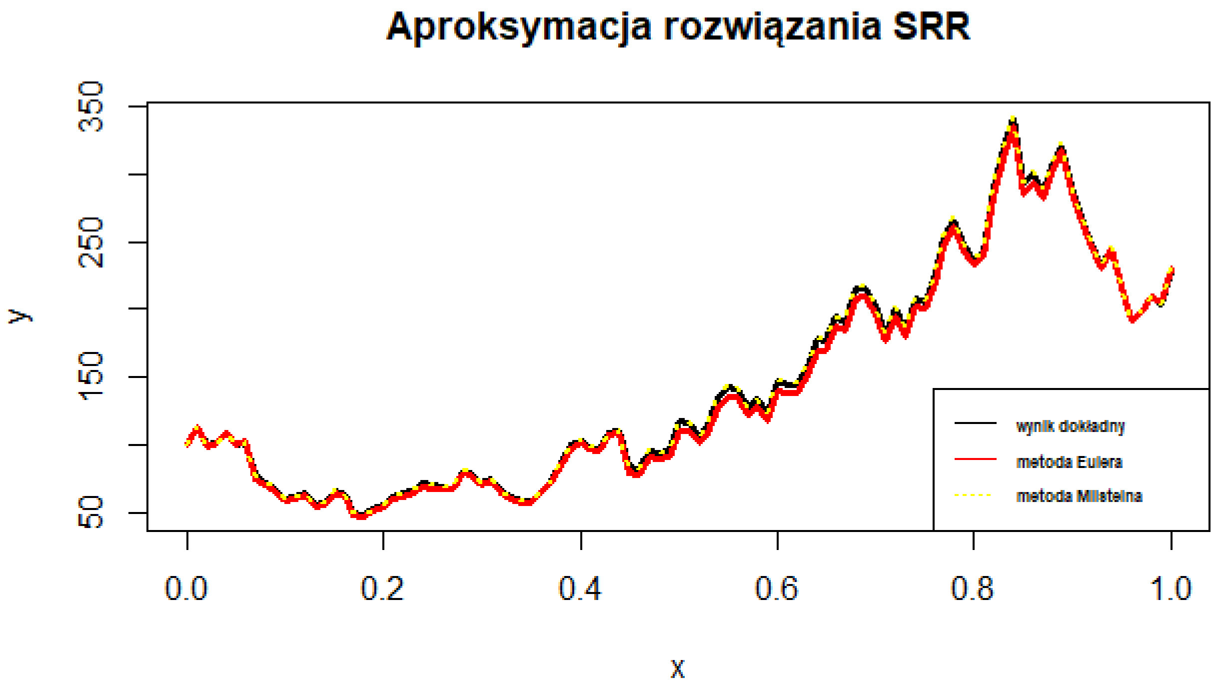
Obie metody dobrze aproksymują rozwiązanie SRR. Metoda Eulera wypada nieco gorzej.

Teraz przeprowadzimy symulację dla parametrów $r = 0.05, \sigma = 1, T = 1, N = 100$ oraz $S_0 = 100$.

```
wyniki2 <- approximation(r = 0.05, sigma = 1, T = 1, N = 100, S0 = 100)
t2 <- wyniki2[[1]]
exact2 <- wyniki2[[2]]
Euler2 <- wyniki2[[3]]
Milstein2 <- wyniki2[[4]]
EX1_2 <- wyniki2[[5]]
EX2_2 <- wyniki2[[6]]
```

$\text{EX1} = 3.3085595$, a $\text{EX2} = 0.7802237$.

```
plot(t2, exact2, main = "Aproksymacja rozwiązania SRR", xlab = "x", ylab = "y",
     type = "l", col = "black", lwd = 3, lty = 1)
lines(t2, Euler2, col = "red", lwd = 3, lty = 1)
lines(t2, Milstein2, col = "yellow", lwd = 2, lty = 3)
legend("bottomright", legend = c("wynik dokładny", "metoda Eulera",
                                 "metoda Milsteina"),
       col = c("black", "red", "yellow"), cex = 0.5, text.font = 2,
       lty = c(1,1,3))
```



Rysunek 2: Aproksymacja rozwiązania stochastycznego równania różniczkowego zależnie od metody

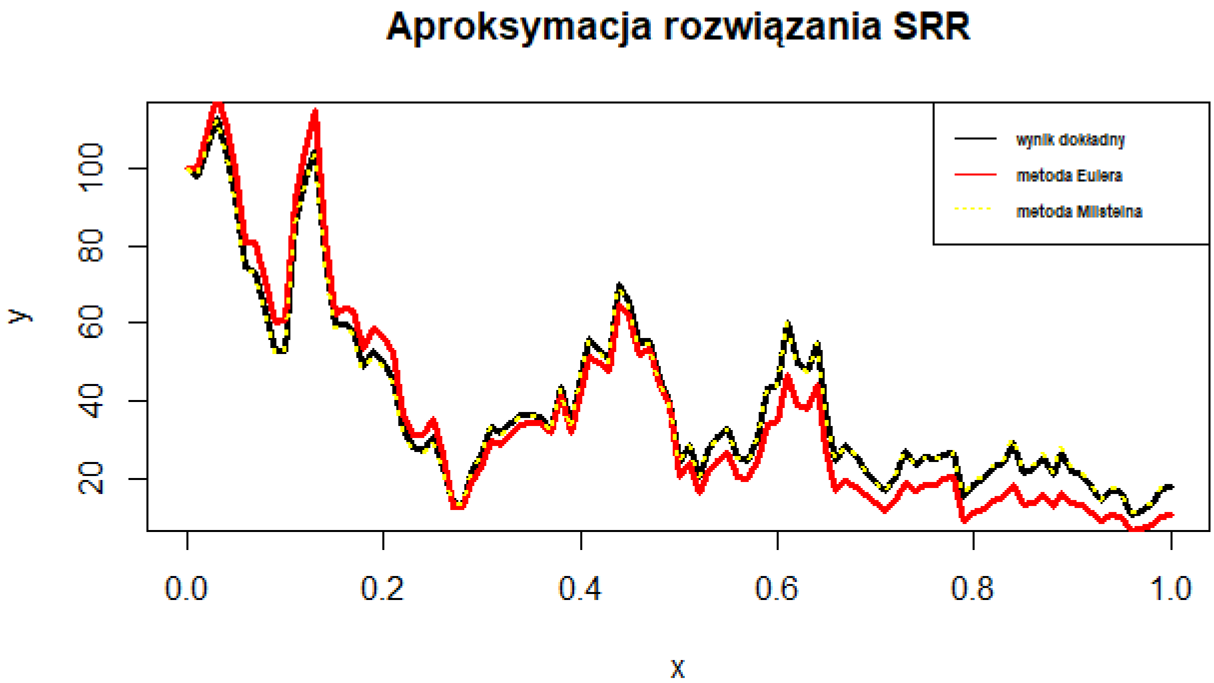
Możemy zauważyć, że dla większego parametru σ błąd metody Eulera jest większy, niż przy $\sigma = 0.2$. Metoda Milsteina radzi sobie bardzo dobrze z aproksymacją rozwiązania.

Przeprowadzimy jeszcze symulację dla parametrów $r = 0.05, \sigma = 2, T = 1, N = 100$ oraz $S_0 = 100$.

```
wyniki3 <- approximation(r = 0.05, sigma = 2, T = 1, N = 100, S0 = 100)
t3 <- wyniki3[[1]]
exact3 <- wyniki3[[2]]
Euler3 <- wyniki3[[3]]
Milstein3 <- wyniki3[[4]]
EX1_3 <- wyniki3[[5]]
EX2_3 <- wyniki3[[6]]
```

EX1 = 5.765482, a EX2 = 0.490340.

```
plot(t3, exact3, main = "Aproksymacja rozwiązania SRR", xlab = "x", ylab = "y",
     type = "l", col = "black", lwd = 3, lty = 1)
lines(t3, Euler3, col = "red", lwd = 3, lty = 1)
lines(t3, Milstein3, col = "yellow", lwd = 2, lty = 3)
legend("topright", legend = c("wynik dokładny", "metoda Eulera",
                              "metoda Milsteina"),
       col = c("black", "red", "yellow"), cex = 0.5, text.font = 2,
       lty = c(1,1,3))
```



Rysunek 3: Aproksymacja rozwiązania stochastycznego równania różniczkowego zależnie od metody

Tutaj już wyraźnie aproksymacja metodą Eulera odbiega od wyniku dokładnego. Metoda Milsteina wciąż wypada bardzo dobrze.

Zadanie 2

W tym zadaniu wycenimy opcję lookback za pomocą różnych metod. Porównamy wyniki uzyskane na podstawie następujących metod:

- metoda Cheuka i Vursta,
- metoda Monte Carlo na podstawie modelu dyskretnego,
- metoda Monte Carlo na podstawie rozwiązania stochastycznego równania różniczkowego (1),
- metoda Monte Carlo na podstawie modelu ciągłego (dyskretyzacja Eulera-Maruyamy oraz dyskretyzacja Milsteina),
- wzór dokładny.

Porównamy wyniki dla parametrów $S = 100, T = 1, r = 0.05, \sigma = 0.2$ oraz dla różnych ilości punktów pośrednich n i różnych ilości symulowanych trajektorii N w przypadku metod Monte Carlo. Przeprowadzimy $M = 1000$ symulacji Monte Carlo. Dla każdej z metod Monte Carlo rozważymy 4 zestawy parametrów:

- $n = 100,\ N = 100$,
- $n = 100,\ N = 1000$,
- $n = 1000,\ N = 100$,
- $n = 1000,\ N = 1000$.

Na początku rozważymy metodę Cheuka i Vursta wykorzystującą drzewka dwumianowe. Jest to metoda deterministyczna.

```r
metoda_Ch_V <- function(S, T, r, sigma, n){
  dt <- T/n
  u <- exp(sigma*sqrt(dt))
  d <- 1/u
  p <- (exp(r*dt)-d)/(u-d)
  q <- exp(-r*dt)*u*p
  f <- matrix(NA, nrow = n+1, ncol = n+1)
  f[, n+1] <- 1 - u^(-(0:n))
  for (j in seq(n, 1, -1)){
    for (k in 1:j){
      if (k == 1){
        f[k, j] <- q*f[k+1, j+1] + (1-q)*f[k, j+1]
      } else
        f[k, j] <- q*f[k+1, j+1] + (1-q)*f[k-1, j+1]
    }
  }
  result = S*f[1,1]
  return(result)
}
```

Rozważymy przypadki, gdy $n = 100$ oraz $n = 1000$.

```
print(metoda_Ch_V(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 100))

## [1] 16.41794
```

```
metoda_Ch_V(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 1000)

## [1] 16.95791
```

Dla $n = 100$ uzyskaliśmy cenę 16.41794, natomiast dla $n = 1000$ cena opcji lookback wynosi 16.95791.

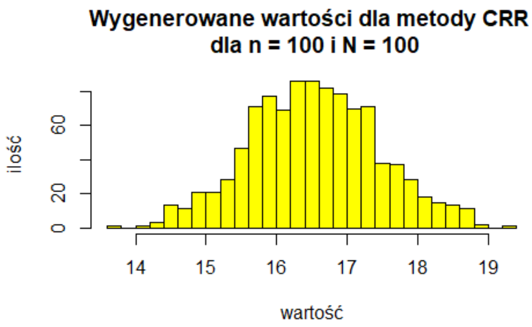
Teraz przejdziemy do metody Monte Carlo na podstawie modelu dyskretnego. Generujemy N trajektorii w oparciu o model Coxa-Rossa-Rubinsteina. Jest to metoda probabilistyczna.

```
CRR <- function(S, T, r, sigma, n, N){
  dt <- T/n
  u <- exp(sigma*sqrt(dt))
  d <- 1/u
  p <- (exp(r*dt) - d)/(u - d)
  Df <- exp(-r*T)
  sum <- 0
  for (k in 0:N){
    trajektoria[1] <- S
    for (i in 1:n){
      Z <- rbinom(1,1,prob = p)
      trajektoria[i+1] <- trajektoria[i]*(u*Z + (1 - Z)*d)
    }
    sum = sum + (S*exp(r*T) - min(trajektoria))/N
    result = sum*Df
  }
  return(result)
}
```

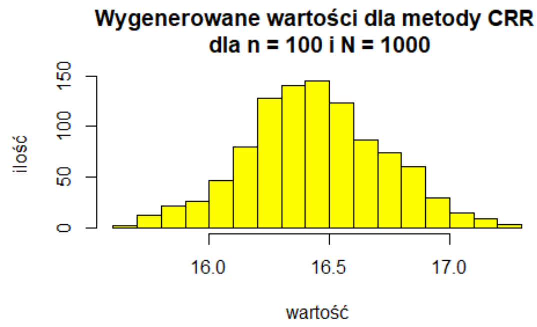
Aby wyznaczyć cenę opcji lookback, przeprowadzimy $M = 1000$ symulacji Monte Carlo. W tym celu skorzystamy z poniżej funkcji, która zwraca 1000 wygenerowanych wartości wspomnianą wcześniej metodą. Dla wygenerowanych wartości narysujemy histogramy oraz obliczymy średnią, aby wyznaczyć cenę opcji.

```
CRR_results <- function(S, T, r, sigma, n, N, M){
  CRR_results <- vector()
  for (i in 1:M){
    CRR_results[i] <- CRR(S, T, r, sigma, n, N)
  }
  return(CRR_results)
}
```

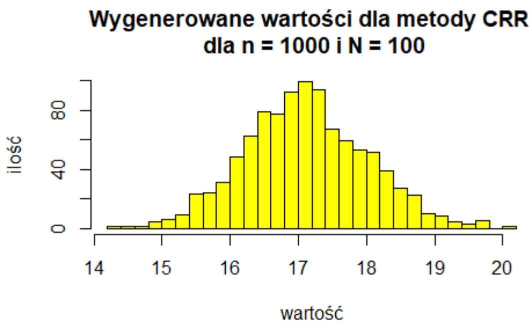
```r
CRR_results1 <- CRR_results(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 100,
                            N = 100, M = 1000)
hist(CRR_results1, main = "Wygenerowane wartości dla metody CRR \n dla n = 100
     i N = 100", xlab = "wartość", ylab = "ilość", col = "yellow",
     nclass = "Freedman-diaconis")

sum(CRR_results1)/1000
#16.53163

CRR_results2 <- CRR_results(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 100,
                            N = 1000, M = 1000)
hist(CRR_results2, main = "Wygenerowane wartości dla metody CRR \n dla n = 100
     i N = 1000", xlab = "wartość", ylab = "ilość", col = "yellow",
     nclass = "Freedman-diaconis")

sum(CRR_results2)/1000
#16.44127

CRR_results3 <- CRR_results(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 1000,
                            N = 100, M = 1000)
hist(CRR_results3, main = "Wygenerowane wartości dla metody CRR \n dla n = 1000
     i N = 100", xlab = "wartość", ylab = "ilość", col = "yellow",
     nclass = "Freedman-diaconis")

sum(CRR_results3)/1000
#17.11723

CRR_results4 <- CRR_results(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 1000,
                            N = 1000, M = 1000)
hist(CRR_results4, main = "Wygenerowane wartości dla metody CRR \n dla n = 1000
     i N = 1000", xlab = "wartość", ylab = "ilość", col = "yellow",
     nclass = "Freedman-diaconis")

sum(CRR_results4)/1000
#16.99129
```



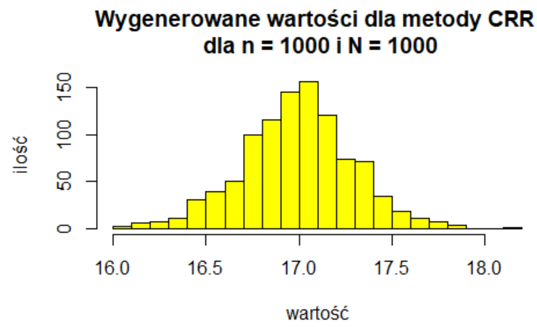
Rysunek 4: Dla n = 100, N = 100



Rysunek 5: Dla n = 100, N = 1000



Rysunek 6: Dla n = 1000, N = 100



Rysunek 7: Dla n = 1000, N = 1000

Kolejną metodą do wyznaczenia ceny opcji lookback jest metoda Monte Carlo na podstawie rozwiązania stochastycznego równania różniczkowego (1). Jest to metoda probabilistyczna.

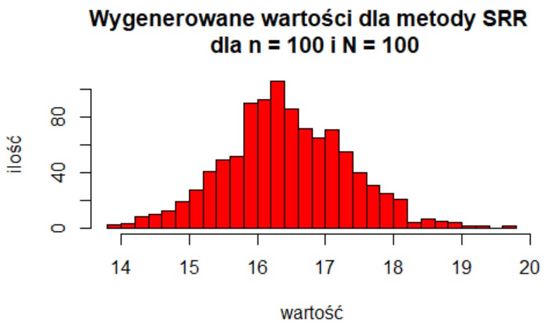
```
SRR <- function(S, T, r, sigma, n, N){
  dt <- T/n
  t <- seq(from = 0, to = 1, by = dt)
  Df <- exp(-r*T)
  sum <- 0
  for (k in 0:N){
    brown <- rnorm(n,0,sqrt(1/n))
    brown <- cumsum(brown)
    trajektoria[1] <- S
    for (i in 1:n){
      trajektoria[i+1] <- S*exp((r - (sigma^2)/2)*t[i] + sigma*brown[i])
    }
    sum <- sum + (S*exp(r*T) - min(trajektoria))
  }
  result <- (1/N)*sum*Df
  return(result)
}
```

Przeprowadzimy $M = 1000$ symulacji Monte Carlo. Skorzystamy z poniżej funkcji, która zwraca 1000 wygenerowanych wartości na podstawie rozwiązania SRR (1).

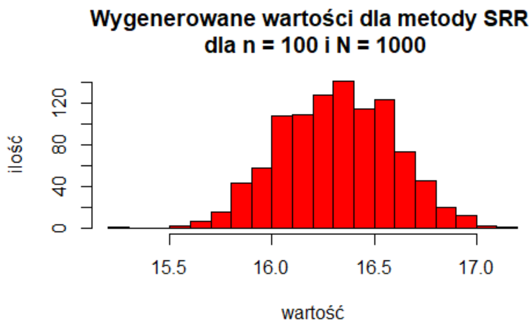
```r
SRR_results <- function(S, T, r, sigma, n, N, M){
  SRR_results <- vector()
  for (i in 1:M){
    SRR_results[i] <- SRR(S, T, r, sigma, n, N)
  }
  return(SRR_results)
}
```

Wygenerowane wartości przedstawimy na histogramach, a także policzymy dla nich średnią, żeby wyznaczyć cenę opcji lookback.

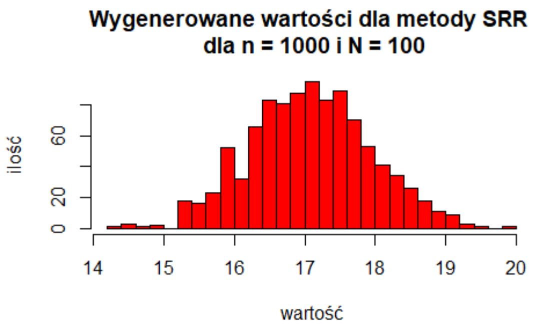
```r
SRR_results1 <- SRR_results(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 100,
                            N = 100, M = 1000)
hist(SRR_results1, main = "Wygenerowane wartości dla metody SRR \n dla n = 100
     i N = 100", xlab = "wartość", ylab = "ilość", col = "red",
     nclass = "Freedman-diaconis")

sum(SRR_results1)/1000
#16.42815

SRR_results2 <- SRR_results(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 100,
                            N = 1000, M = 1000)
hist(SRR_results2, main = "Wygenerowane wartości dla metody SRR \n dla n = 100
     i N = 1000", xlab = "wartość", ylab = "ilość", col = "red",
     nclass = "Freedman-diaconis")

sum(SRR_results2)/1000
#16.31806

SRR_results3 <- SRR_results(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 1000,
                            N = 100, M = 1000)
hist(SRR_results3, main = "Wygenerowane wartości dla metody SRR \n dla n = 1000
     i N = 100", xlab = "wartość", ylab = "ilość", col = "red",
     nclass = "Freedman-diaconis")

sum(SRR_results3)/1000
#17.07939

SRR_results4 <- SRR_results(S = 100, T = 1, r = 0.05, sigma = 0.2, n = 1000,
                            N = 1000, M = 1000)
hist(SRR_results4, main = "Wygenerowane wartości dla metody SRR \n dla n = 1000
     i N = 1000", xlab = "wartość", ylab = "ilość", col = "red",
     nclass = "Freedman-diaconis")

sum(SRR_results4)/1000
#16.92758
```



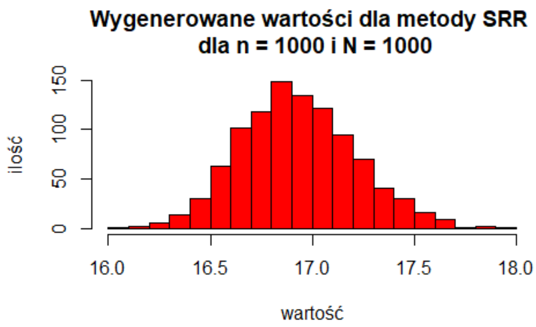
Rysunek 8: Dla n = 100, N = 100



Rysunek 9: Dla n = 100, N = 1000



Rysunek 10: Dla n = 1000, N = 100



Rysunek 11: Dla n = 1000, N = 1000

Ostatnią metodą Monte Carlo, którą będziemy rozpatrywać, jest metoda MC oparta na modelu ciągłym. Generujemy N trajektorii na podstawie dyskretyzacji stochastycznego równania różniczkowego: dyskretyzacji Eulera-Maruyamy oraz dyskretyzacji Milsteina. Są to metody probabilistyczne.

Najpierw przedstawimy metodę Monte Carlo na podstawie dyskretyzacji Eulera-Maruyamy.

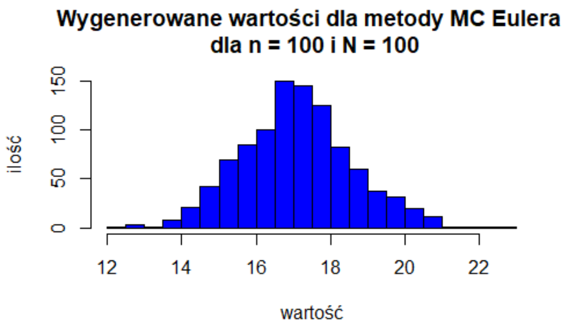
```
MC_Euler <- function(S, T, r, sigma, n, N){
  dt = T/n
  trajektoria <- matrix(NA, nrow = N, ncol = n+1)
  trajektoria[,1] <- S
  for (i in 1:n){
    trajektoria[, i+1] <- trajektoria[, i]*(1+r*dt)
    + trajektoria[, i]*sigma*sqrt(dt)*rnorm(N)
  }
  minimum <- apply(trajektoria, 1, min)
  prices <- trajektoria[, n+1] - minimum
  result <- mean(prices)
  return(result)
}
```

Powyższa funkcja zwraca cenę opcji lookback obliczoną na podstawie dyskretyzacji Eulera-Maruyamy. Na jej podstawie przeprowadzimy symulacje Monte Carlo. W tym celu skorzystamy z zaimplementowanej funkcji, która zwraca 1000 wygenerowanych cen opcji lookback.

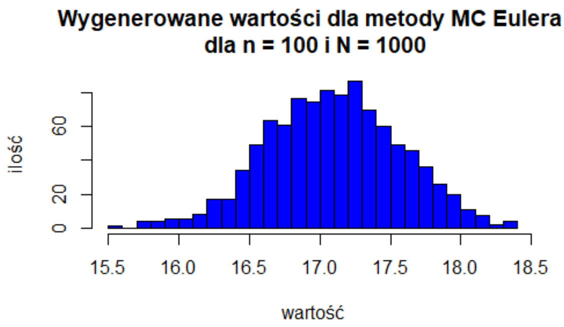
```
MC_Euler_results <- function(S, T, r, sigma, n, N, M){
  MC_Euler_results <- vector()
  for (i in 1:M){
    MC_Euler_results[i] <- MC_Euler(S, T, r, sigma, n, N)
  }
  return(MC_Euler_results)
}
```

Dla wygenerowanych wartości narysujemy histogramy oraz obliczymy średnią, aby wyznaczyć cenę opcji.

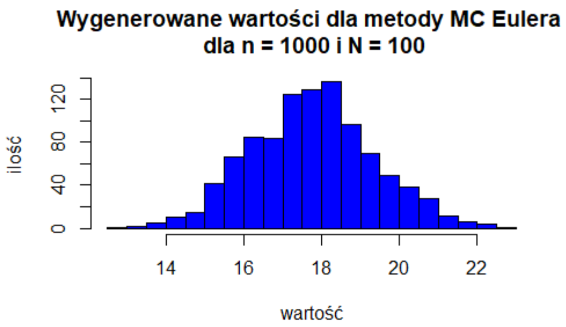
```
MC_Euler_results1 <- MC_Euler_results(S = 100, T = 1, r = 0.05, sigma = 0.2,
                                      n = 100, N = 100, M = 1000)
hist(MC_Euler_results1, main = "Wygenerowane wartości dla metody MC Eulera \n
     dla n = 100 i N = 100", xlab = "wartość", ylab = "ilość", col = "blue",
     nclass = "Freedman-diaconis")

sum(MC_Euler_results1)/1000
#17.09606

MC_Euler_results2 <- MC_Euler_results(S = 100, T = 1, r = 0.05, sigma = 0.2,
                                      n = 100, N = 1000, M = 1000)
hist(MC_Euler_results2, main = "Wygenerowane wartości dla metody MC Eulera \n
     dla n = 100 i N = 1000", xlab = "wartość", ylab = "ilość", col = "blue",
     nclass = "Freedman-diaconis")

sum(MC_Euler_results2)/1000
#17.09669

MC_Euler_results3 <- MC_Euler_results(S = 100, T = 1, r = 0.05, sigma = 0.2,
                                      n = 1000, N = 100, M = 1000)
hist(MC_Euler_results3, main = "Wygenerowane wartości dla metody MC Eulera \n
     dla n = 1000 i N = 100", xlab = "wartość", ylab = "ilość", col = "blue",
     nclass = "Freedman-diaconis")

sum(MC_Euler_results3)/1000
#17.75898

MC_Euler_results4 <- MC_Euler_results(S = 100, T = 1, r = 0.05, sigma = 0.2,
                                      n = 1000, N = 1000, M = 1000)
hist(MC_Euler_results4, main = "Wygenerowane wartości dla metody MC Eulera \n
     dla n = 1000 i N = 1000", xlab = "wartość", ylab = "ilość", col = "blue",
     nclass = "Freedman-diaconis")

sum(MC_Euler_results4)/1000
#17.75195
```



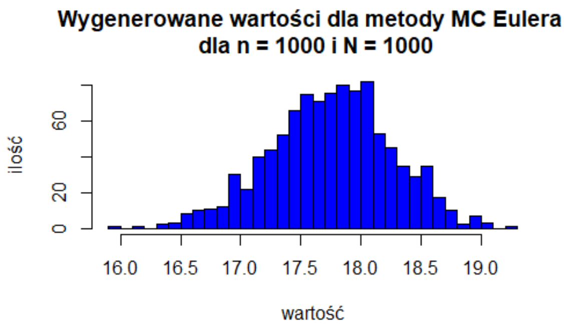
Rysunek 12: Dla n = 100, N = 100



Rysunek 13: Dla n = 100, N = 1000



Rysunek 14: Dla n = 1000, N = 100



Rysunek 15: Dla n = 1000, N = 1000

Teraz przedstawimy metodę Monte Carlo na podstawie dyskretyzacji Milsteina.

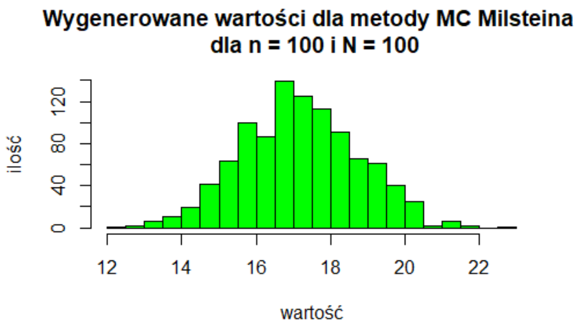
```
MC_Milstein <- function(S, T, r, sigma, n, N){
  dt = T/n
  trajektoria <- matrix(nrow = N, ncol = n+1)
  trajektoria[,1] <- S
  for (i in 1:n){
    Z = rnorm(N, 0, 1)
    trajektoria[, i+1] <- trajektoria[, i]*(1+r*dt)
    + trajektoria[, i]*sigma*sqrt(dt)*Z
    + 0.5*(sigma^2)*trajektoria[, i]*dt*(Z^2 - 1)
  }
  minimum <- apply(trajektoria, 1, min)
  prices <- trajektoria[, n+1] - minimum
  result <- mean(prices)
  return(result)
}
```

Powyżej przedstawiona funkcja zwraca cenę opcji lookback obliczoną na podstawie dyskretyzacji Milsteina. Korzystając z niej przeprowadzimy $M = 1000$ symulacji Monte Carlo. Poniżej pokazano zaimplementowaną funkcję zwracającą 1000 wygenerowanych cen opcji lookback.

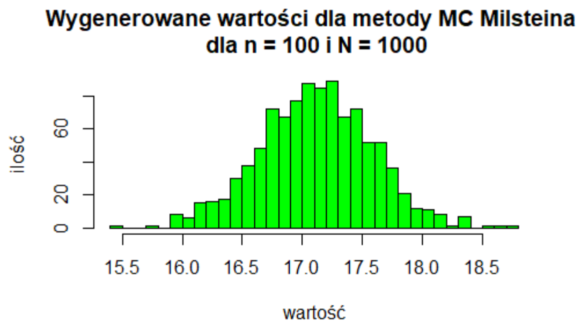
```r
MC_Milstein_results <- function(S, T, r, sigma, n, N, M){
  MC_Milstein_results <- vector()
  for (i in 1:M){
    MC_Milstein_results[i] <- MC_Milstein(S, T, r, sigma, n, N)
  }
  return(MC_Milstein_results)
}
```

Wygenerowane wartości przedstawimy na histogramach, a także obliczymy średnią, aby wyznaczyć cenę opcji.

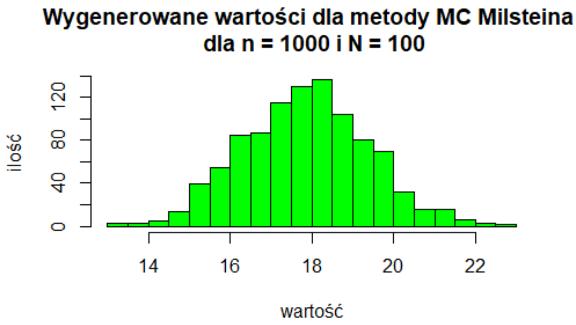
```r
MC_Milstein_results1 <- MC_Milstein_results(S = 100, T = 1, r = 0.05,
                        sigma = 0.2, n = 100, N = 100, M = 1000)
hist(MC_Milstein_results1, main = "Wygenerowane wartości dla metody MC
     Milsteina \n dla n = 100 i N = 100", xlab = "wartość", ylab = "ilość",
     col = "green", nclass = "Freedman-diaconis")

sum(MC_Milstein_results1)/1000
#17.16814

MC_Milstein_results2 <- MC_Milstein_results(S = 100, T = 1, r = 0.05,
                        sigma = 0.2, n = 100, N = 1000, M = 1000)
hist(MC_Milstein_results2, main = "Wygenerowane wartości dla metody MC
     Milsteina \n dla n = 100 i N = 1000", xlab = "wartość", ylab = "ilość",
     col = "green", nclass = "Freedman-diaconis")

sum(MC_Milstein_results2)/1000
#17.11623

MC_Milstein_results3 <- MC_Milstein_results(S = 100, T = 1, r = 0.05,
                        sigma = 0.2, n = 1000, N = 100, M = 1000)
hist(MC_Milstein_results3, main = "Wygenerowane wartości dla metody MC
     Milsteina \n dla n = 1000 i N = 100", xlab = "wartość", ylab = "ilość",
     col = "green", nclass = "Freedman-diaconis")

sum(MC_Milstein_results3)/1000
#17.85422

MC_Milstein_results4 <- MC_Milstein_results(S = 100, T = 1, r = 0.05,
                        sigma = 0.2, n = 1000, N = 1000, M = 1000)
hist(MC_Milstein_results4, main = "Wygenerowane wartości dla metody MC
     Milsteina \n dla n = 1000 i N = 1000", xlab = "wartość", ylab = "ilość",
     col = "green", nclass = "Freedman-diaconis")

sum(MC_Milstein_results4)/1000
#17.76709
```



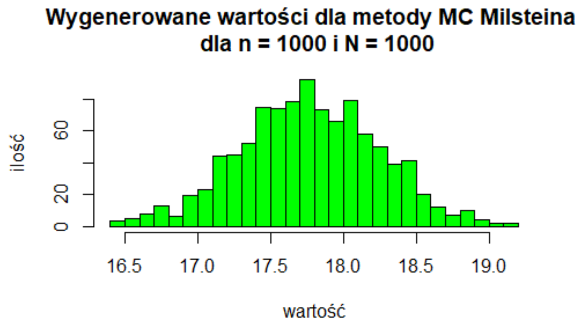
Rysunek 16: Dla n = 100, N = 100



Rysunek 17: Dla n = 100, N = 1000



Rysunek 18: Dla n = 1000, N = 100



Rysunek 19: Dla n = 1000, N = 1000

Ostatnia z omawianych metod korzysta ze wzoru dokładnego. Jest to metoda deterministyczna. Poniżej przedstawiono funkcję liczącą cenę opcji lookback.

```
lookback <- function(S, T, r, sigma){
  a1 <- ((r + (sigma^2)/2)*T)/(sigma*sqrt(T))
  a2 <- ((r - (sigma^2)/2)*T)/(sigma*sqrt(T))
  price <- S*pnorm(a1) - S*exp(-r*T)*pnorm(a2) -
  ((pnorm(-a1) - exp(-r*T)*pnorm(a2))*(S*sigma^2))/(2*r)
  return(price)
}
```

```
print(lookback(S = 100, T = 1, r = 0.05, sigma = 0.2))

## [1] 17.2168
```

Cena dokładna opcji lookback wynosi 17.2168.

Teraz zajmiemy się porównaniem otrzymanych wyników. Przypomnijmy, że dla metody Cheuka i Vursta otrzymaliśmy oszacowania 16.41794 dla $n = 100$ oraz 16.95791 dla $n = 1000$. Wyniki otrzymane dla omawianych metod Monte Carlo przedstawimy w tabeli w zależności od zestawu parametrów oraz zastosowanej metody.

Tabela 1: Oszacowana cena opcji lookback

parametry	metoda			
	CRR	SRR	MC Euler	MC Milstein
n = 100, N = 100	16.53163	16.42815	17.09606	17.16814
n = 100, N = 1000	16.44127	16.31806	17.09669	17.11623
n = 1000, N = 100	17.11723	17.07939	17.75898	17.85422
n = 1000, N = 1000	16.99129	16.92758	17.75195	17.76709

Aby łatwiej było porównywać wyniki, obliczymy błąd bezwzględny jaki popełniamy względem ceny dokładnej. Dla metody Cheuka i Vursta dla $n = 100$ wynosi on 0.79886, natomiast dla $n = 1000$ jest to 0.25889.

Tabela 2: Błąd bezwzględny względem ceny dokładnej

parametry	metoda			
	CRR	SRR	MC Euler	MC Milstein
n = 100, N = 100	0.68517	0.78865	0.12074	0.04866
n = 100, N = 1000	0.77553	0.89874	0.12011	0.10057
n = 1000, N = 100	0.09957	0.13741	0.54218	0.63742
n = 1000, N = 1000	0.22551	0.28922	0.53515	0.55029

Możemy zauważyć, że dla metody CRR oraz SRR dla $n = 1000$ otrzymujemy lepsze oszacowania, z czego dla $N = 100$ błąd bezwzględny popełniany względem ceny dokładnej jest najmniejszy. W przypadku metod Monte Carlo na podstawie dyskretyzacji Eulera-Maruyamy oraz dyskretyzacji Milsteina lepszą aproksymację uzyskujemy dla $n = 100$. Dla $n = 1000$ popełniany błąd bezwzględny jest kilka razy większy. Dla metody MC Eulera-Maruyamy dla ustalonego n wyniki są do siebie bardzo zbliżone. Natomiast biorąc pod uwagę metodę MC Milsteina możemy zauważyć, że najlepsze oszacowanie otrzymaliśmy dla najmniejszych n i N. Patrząc na parametry, dla $n = 100$ i $N = 100$ najlepszą aproksymację uzyskaliśmy metodą Monte Carlo Milsteina. Dla parametrów $n = 100$ oraz $N = 1000$ także metoda MC Milsteina daje najlepszy wynik. Dla $n = 1000$ oraz $N = 100$ najlepszą aproksymację otrzymaliśmy korzystając z metody CRR. Także dla $n = 1000$ oraz $N = 1000$ metoda CRR okazała się najlepsza. W przypadku metody Cheuka i Vursta im większy parametr n, tym lepsze oszacowanie.