Interest Rate Models

Single-Factor Models

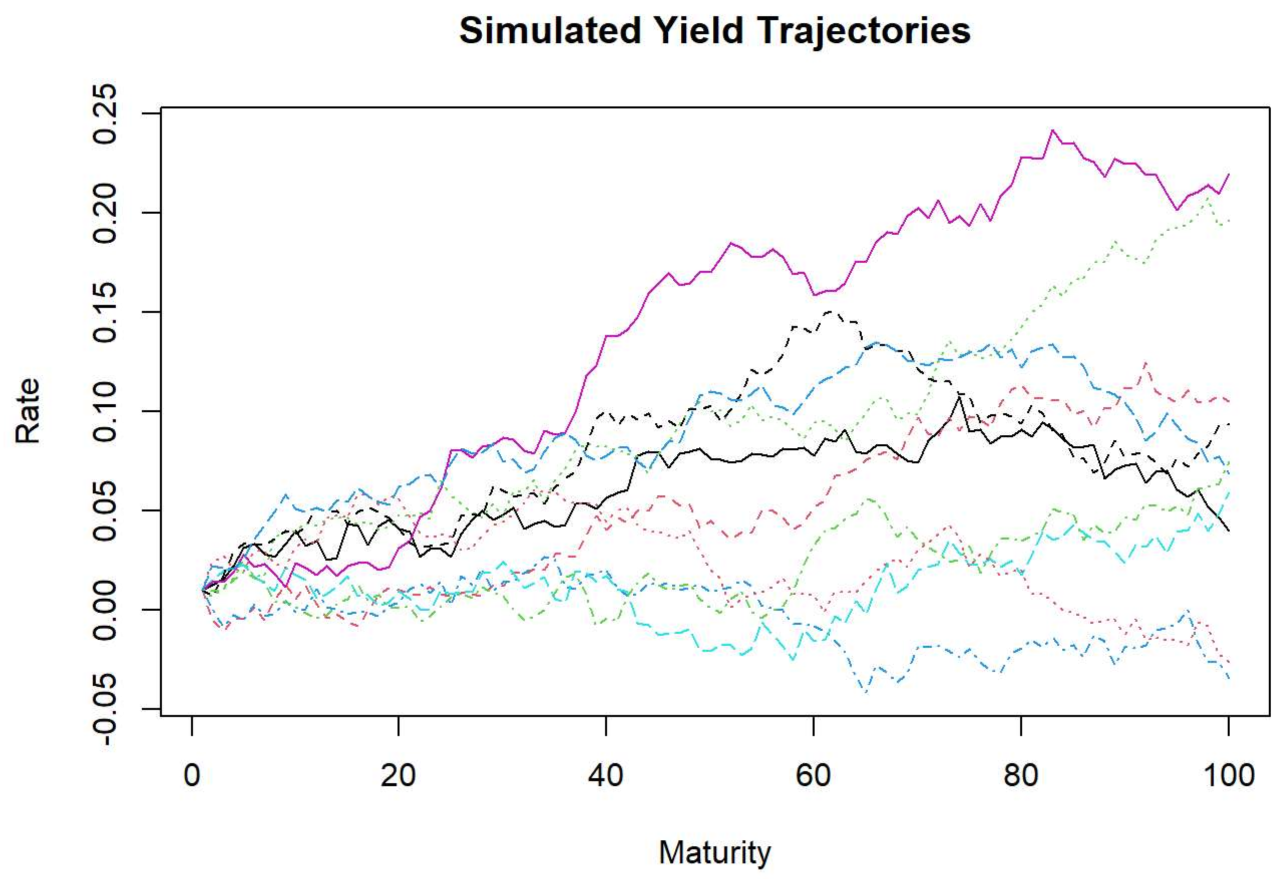
The models considered in this section are based on a single source of uncertainty. The short-term rate is assumed to follow a (one-dimensional) Markov process.

The first models here are ‘Time-Homogeneous Short-Rate Models’, meaning that the assumed short rate dynamics depended only on constant coefficients.

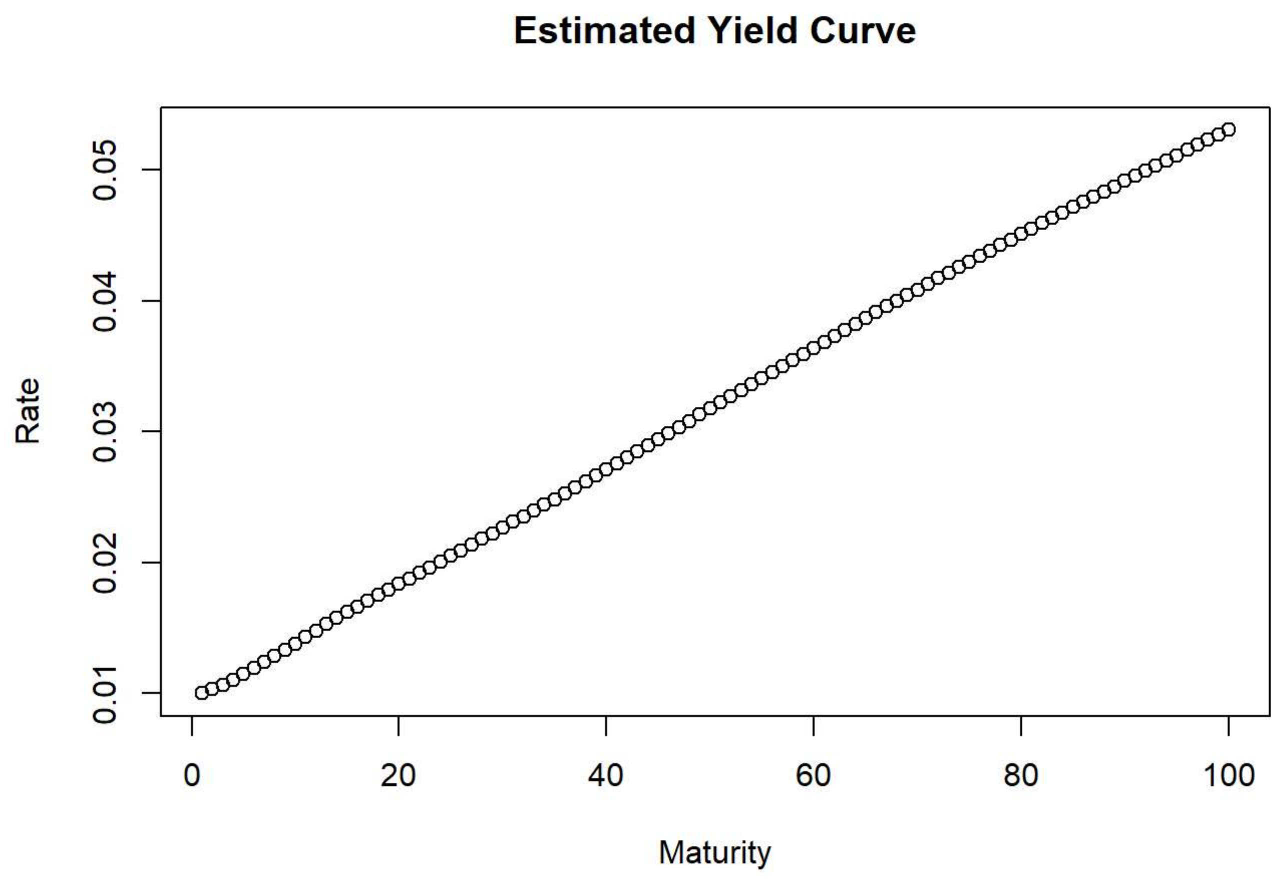
The success of models like these was mainly due to their possibility of pricing analytically bonds and bond options. The small number of model parameters prevents a satisfactory calibration to market data and even the zero-coupon curve is quite likely to be badly reproduced.

Merton’s model

```
years <- 10
n_models <- 500
n_periods <- 100
step <- years/n_periods

r_0 <- 0.01
alpha = 0.01
sigma = 0.02

r_table <- matrix(nrow = n_periods, ncol = n_models)
r_table[1,] <- r_0

for(i in 1:ncol(r_table)){
  for(t in 2:nrow(r_table)){
    r_table[t,i] <- r_table[t-1,i] + alpha*(years/n_periods) + sigma*rnorm(1)*sqrt(years/n_periods)
  }
}


matplot(r_table[,1:10], type = "l", xlab="Maturity", ylab="Rate", main = "Simulated Yield Trajectories"
)
```



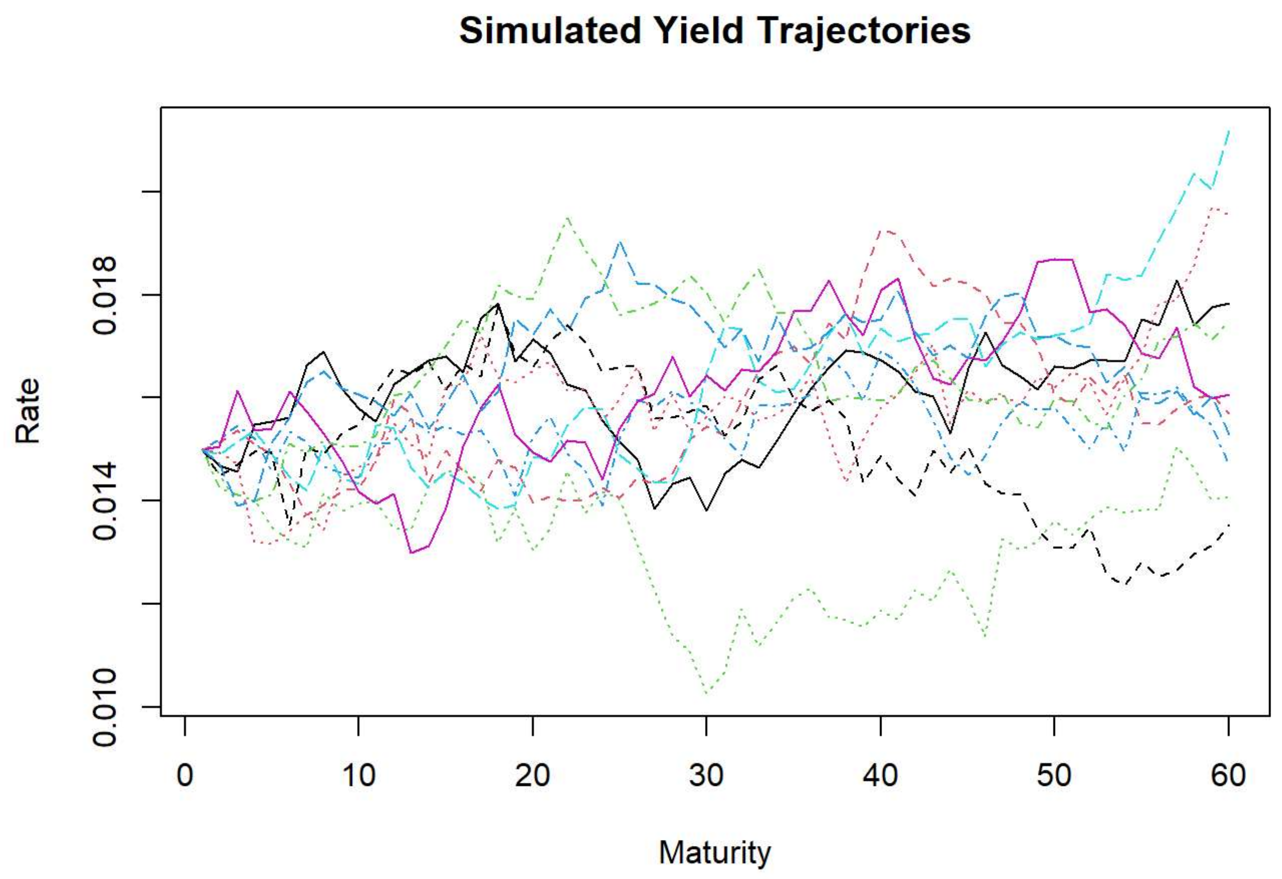
```r
exact_value <- exp(-1/2*alpha*(10^2) + 1/6*(sigma^2)*(10^3) - 10*r_0)

spot_curve <- rep(NA,n_periods)
spot_curve[1] <- r_0

for(t in c(2:n_periods)){
  ss <- colSums(r_table[1:t,] * step)  # integral estimate
  c <- exp(-ss)
  estimate <- mean(c)
  spot_curve[t] <- -log(estimate)/(t*step)
}

plot(spot_curve, xlab="Maturity", ylab="Rate", main ="Estimated Yield Curve")
```



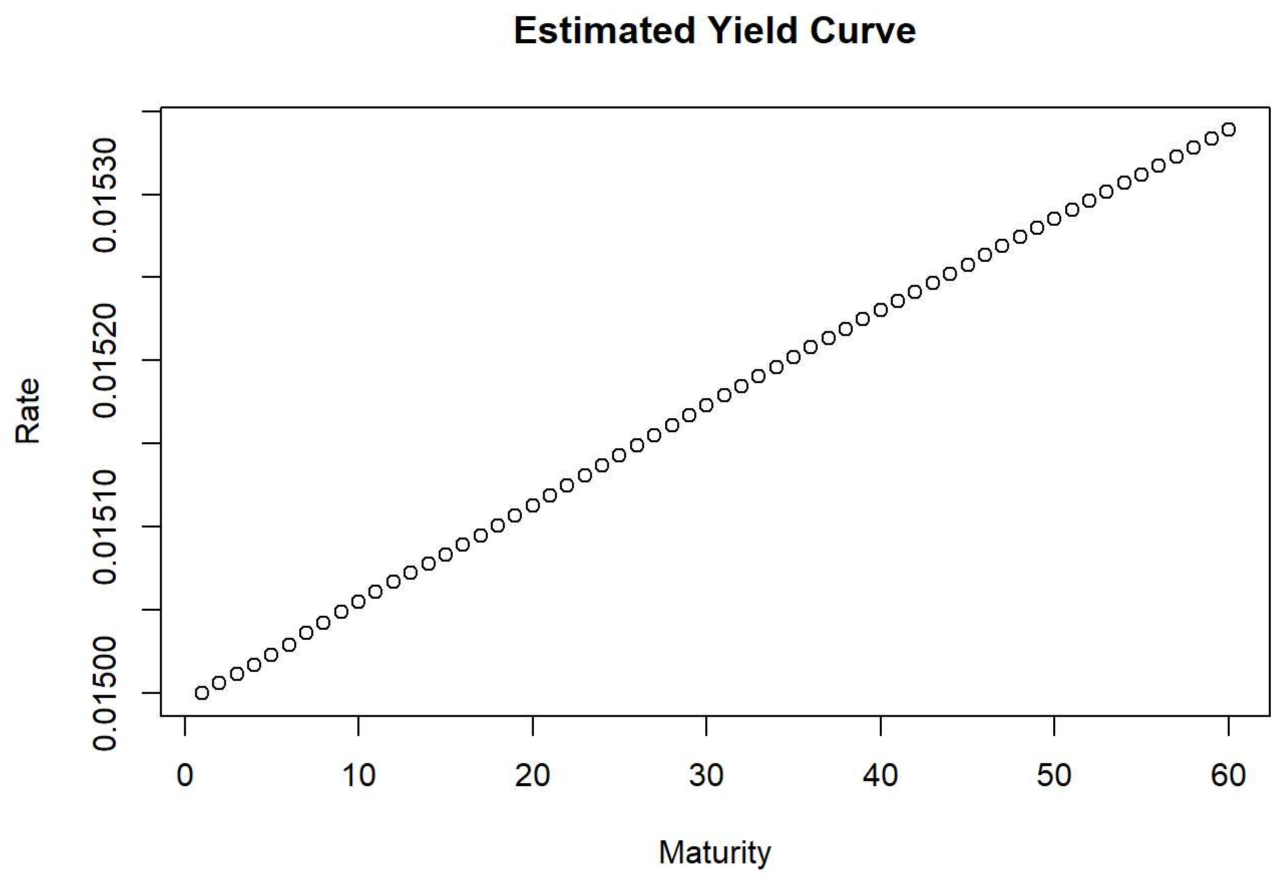
Vasicek model

This model adds a mean-reverting characteristic to short-term interest rates.

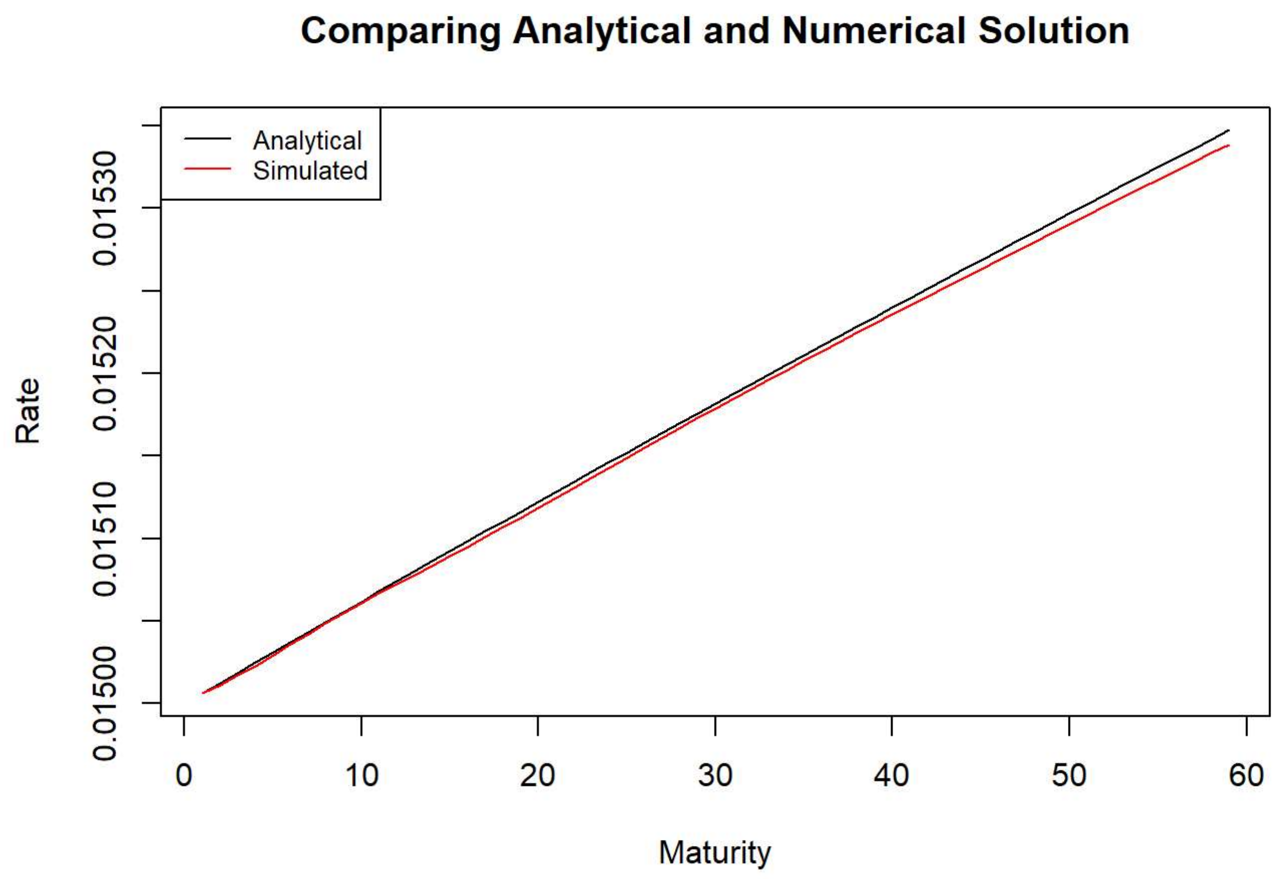
```r
years <- 5
n_models <- 100000
n_periods <- years*12
step <- years/n_periods

r_0 <- 0.015
alpha = 0.03
beta  = 0.01
sigma = 0.002

r_table <- matrix(nrow = n_periods, ncol = n_models)
r_table[1,] <- r_0

set.seed(123)
for(i in 1:ncol(r_table)){
  for(t in 2:nrow(r_table)){
    r_table[t,i] <- r_table[t-1,i] + beta*(alpha - r_table[t-1,i])*step + sigma*rnorm(1)*sqrt(step)
  }
}

matplot(r_table[,1:10], type = "l", xlab="Maturity", ylab="Rate", main = "Simulated Yield Trajectories"
)
```



```
spot_curve <- rep(NA,n_periods)
spot_curve[1] <- r_0

for(t in c(2:n_periods)){
  ss <- colSums(r_table[1:t,] * step)  # integral estimate
  c <- exp(-ss)
  estimate <- mean(c)
  spot_curve[t] <- -log(estimate)/(t*step)
}

plot(spot_curve, xlab="Maturity", ylab="Rate", main ="Estimated Yield Curve")
```



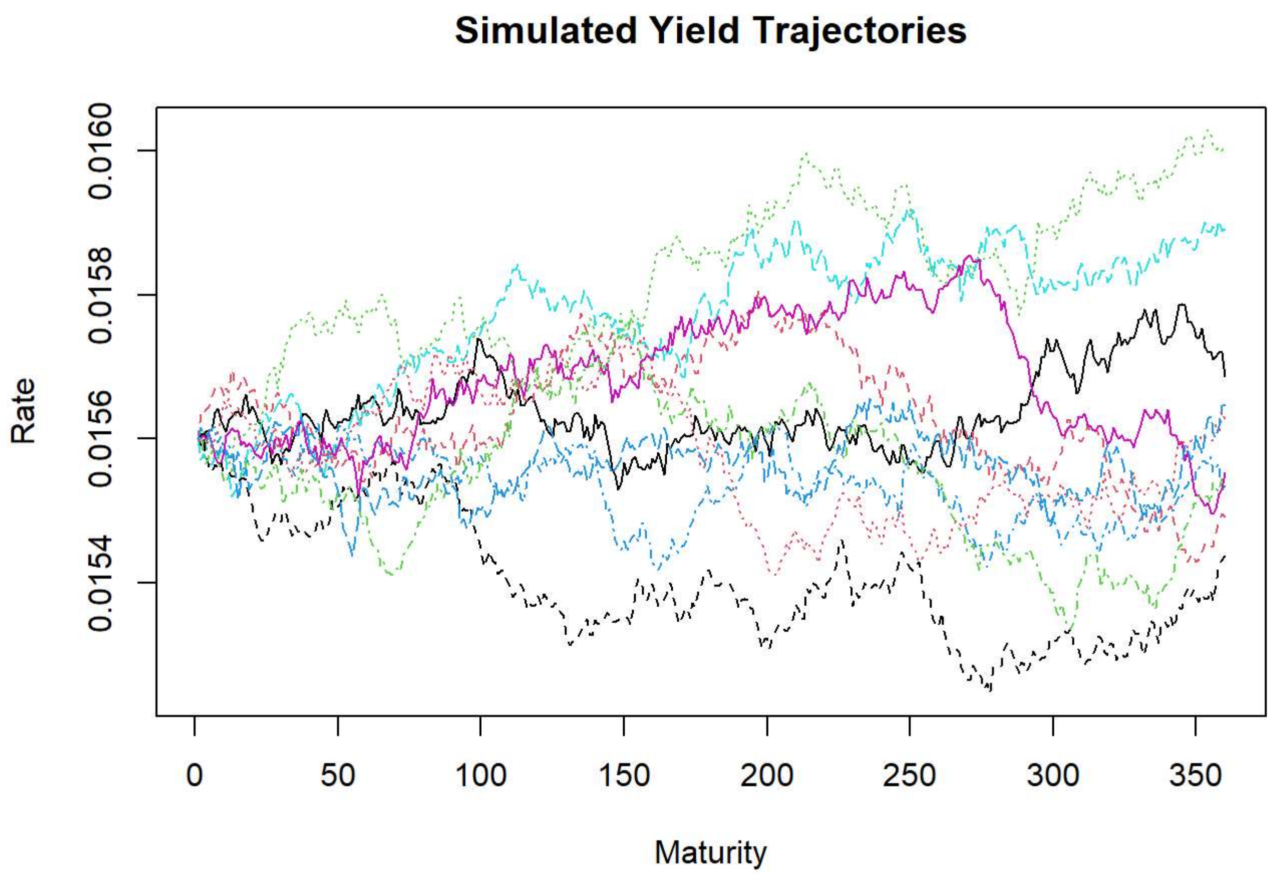
```
# Analytical solution
maturity_t = 0
maturity_T = seq(1,n_periods)*step

B = 1/beta*(1-exp(-beta*(maturity_T-maturity_t)))
A = exp((alpha - (sigma^2)/(2*(beta^2))) * (B - maturity_T + maturity_t) - (sigma^2)/(4*beta)*(B^2))

P_t_T = A*exp(-B*r_0)

analytical_spot_curve <- -log(P_t_T)/maturity_T

# Compare analytical and numerical solution
# Obs: we take out first element of spot curve because it begins in period 0, while the
# analytical spot curve begins at period 1
plot(analytical_spot_curve[-n_periods], type = "l", main = "Comparing Analytical and Numerical Solutio
n", xlab = "Maturity", ylab = "Rate")
lines(spot_curve[-1], type = "l", col = "red")
legend("topleft", legend=c("Analytical", "Simulated"),
       col=c("black", "red"), lty=1, cex=0.8)
```



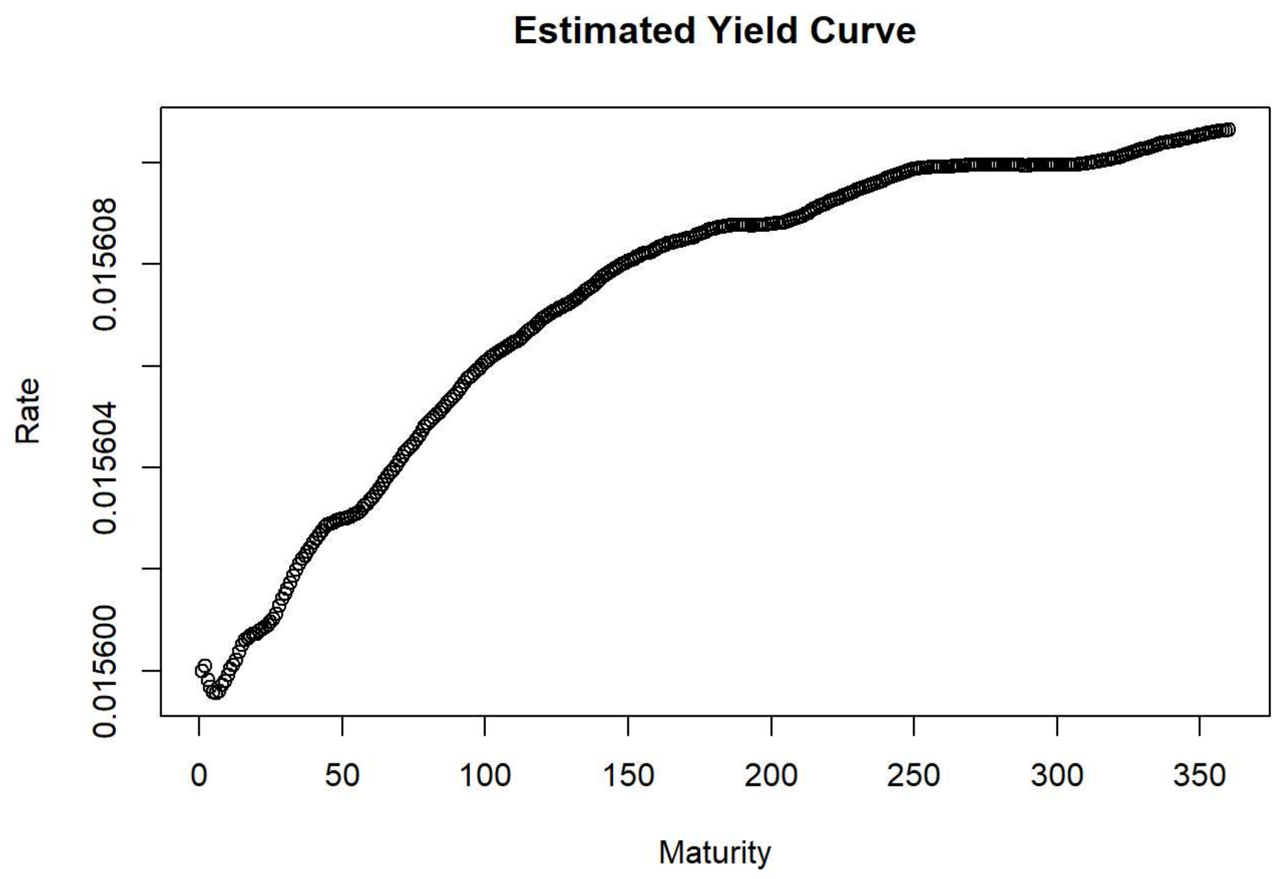
Dothan’s model

This model does not allow negative values to short-term interest rates.

```
years <- 30
n_models <- 500
n_periods <- 360
step <- years/n_periods

r_0 <- 0.0156
sigma = 0.003

r_table <- matrix(nrow = n_periods, ncol = n_models)
r_table[1,] <- r_0

set.seed(123)
for(i in 1:ncol(r_table)){
  for(t in 2:nrow(r_table)){
    r_table[t,i] <- r_table[t-1,i] + sigma*r_table[t-1,i]*rnorm(1)*sqrt(years/n_periods)
  }
}

matplot(r_table[,1:10], type = "l", xlab="Maturity", ylab="Rate", main = "Simulated Yield Trajectories"
)
```



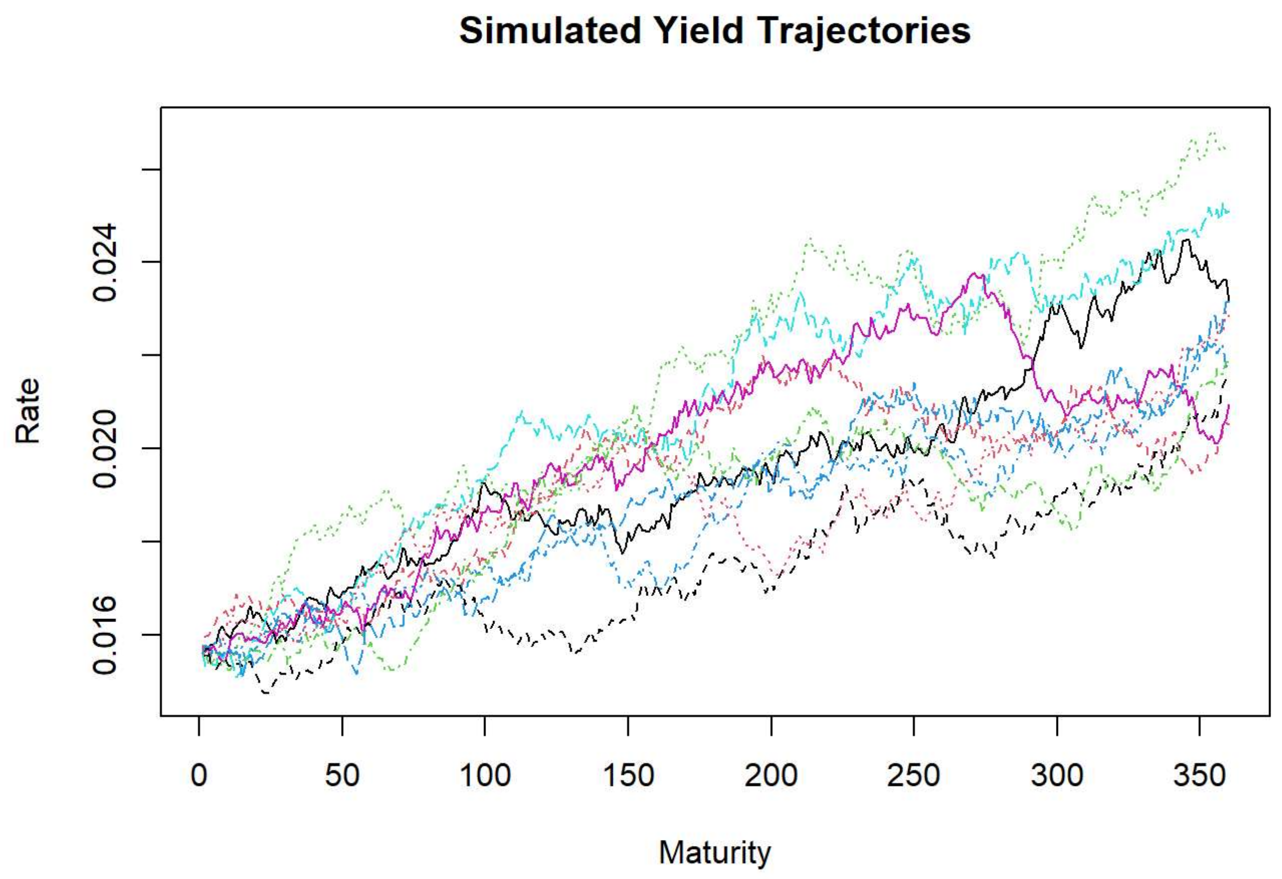
```
spot_curve <- rep(NA,n_periods)
spot_curve[1] <- r_0

for(t in c(2:n_periods)){
  ss <- colSums(r_table[1:t,] * step)  # integral estimate
  c <- exp(-ss)
  estimate <- mean(c)
  spot_curve[t] <- -log(estimate)/(t*step)
}

plot(spot_curve, xlab="Maturity", ylab="Rate", main ="Estimated Yield Curve")
```



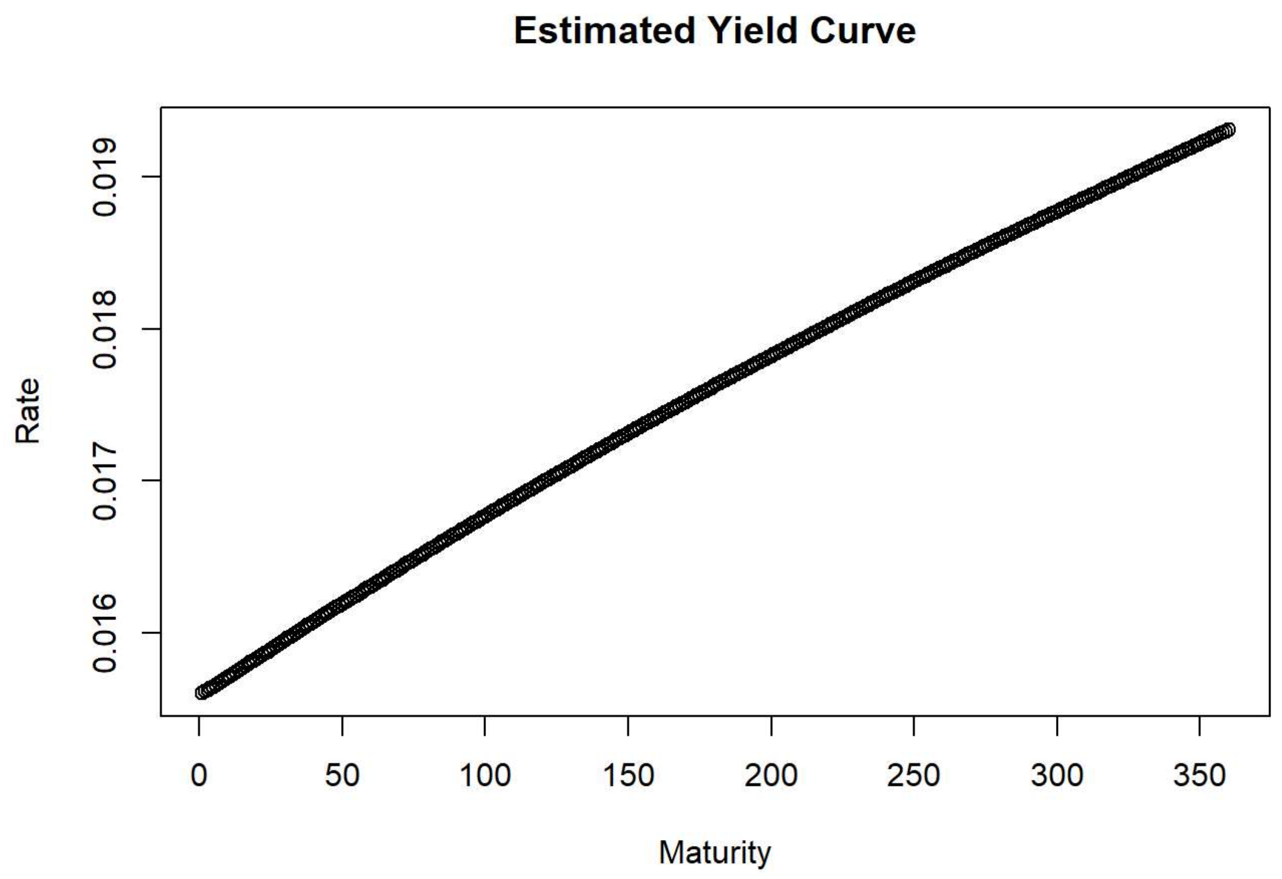
Brennan and Schwartz model

This model does not allow negative values to short-term interest rates and has a mean-reverting characteristic.

```
years <- 30
n_models <- 500
n_periods <- 360
step <- years/n_periods

r_0 <- 0.0156
alpha = 0.035
beta  = 0.0144
sigma = 0.03

r_table <- matrix(nrow = n_periods, ncol = n_models)
r_table[1,] <- r_0

set.seed(123)
for(i in 1:ncol(r_table)){
  for(t in 2:nrow(r_table)){
    r_table[t,i] <- r_table[t-1,i] + beta*(alpha - r_table[t-1,i])*(years/n_periods) + sigma*r_table[t-
1,i]*rnorm(1)*sqrt(years/n_periods)
  }
}

matplot(r_table[,1:10], type = "l", xlab="Maturity", ylab="Rate", main = "Simulated Yield Trajectories"
)
```



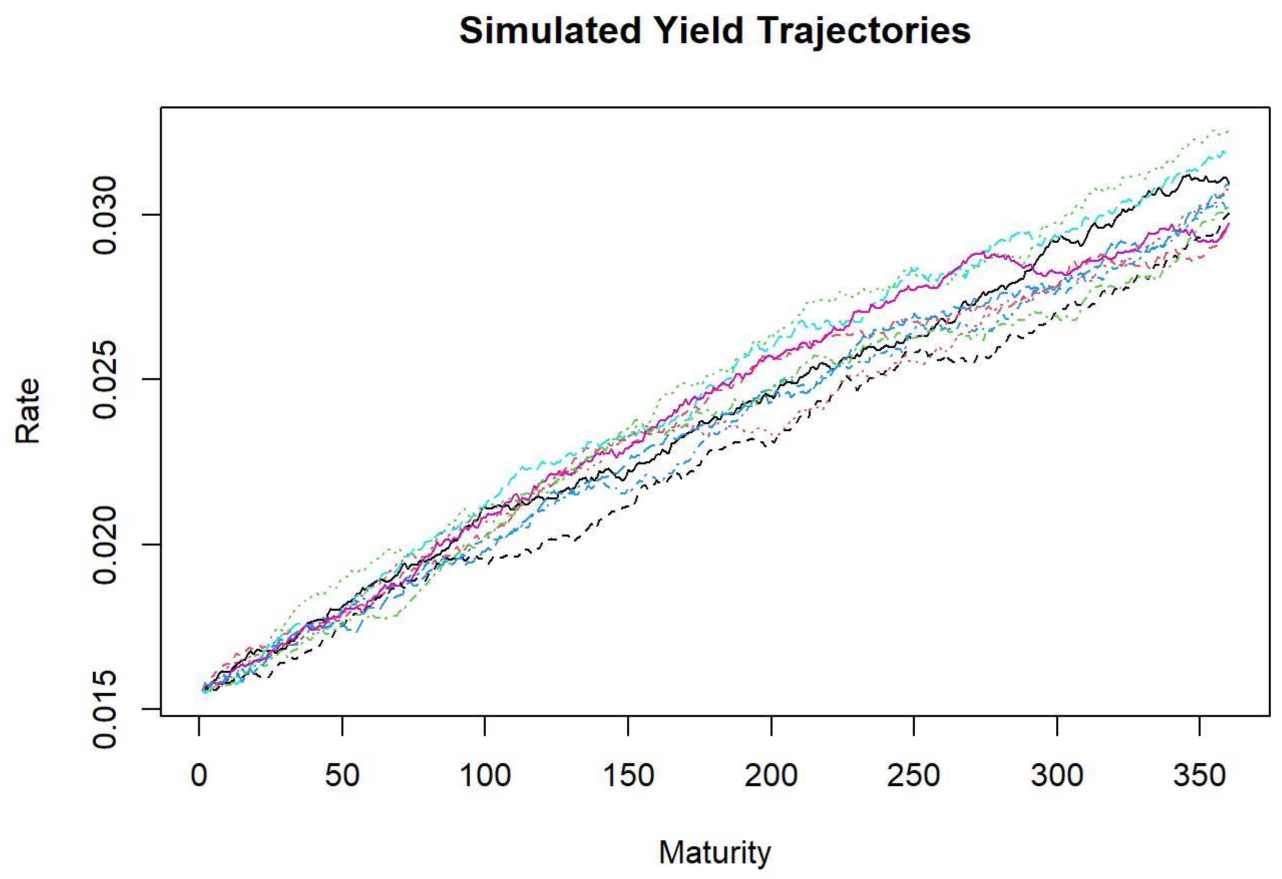
```
spot_curve <- rep(NA,n_periods)
spot_curve[1] <- r_0

for(t in c(2:n_periods)){
  ss <- colSums(r_table[1:t,] * step)  # integral estimate
  c <- exp(-ss)
  estimate <- mean(c)
  spot_curve[t] <- -log(estimate)/(t*step)
}

plot(spot_curve, xlab="Maturity", ylab="Rate", main ="Estimated Yield Curve")
```



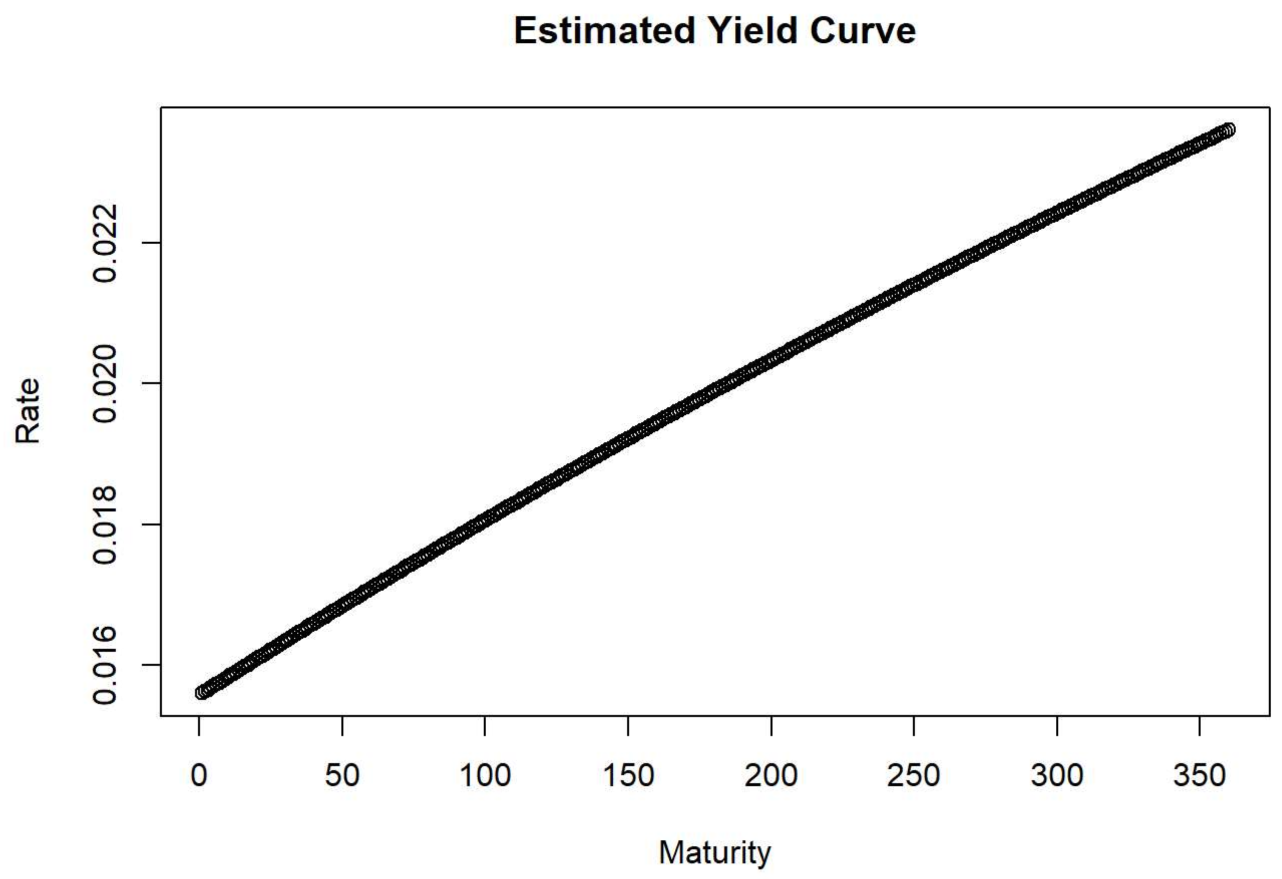
Cox-Ingersoll-Ross model (CIR)

This model is derived from a general equilibrium approach.

```
years <- 30
n_models <- 500
n_periods <- 360
step <- years/n_periods

r_0 <- 0.0156
alpha = 0.058
beta  = 0.0144
sigma = 0.002

r_table <- matrix(nrow = n_periods, ncol = n_models)
r_table[1,] <- r_0

set.seed(123)
for(i in 1:ncol(r_table)){
  for(t in 2:nrow(r_table)){
    if(r_table[t-1,i]<0){
      r_table[t,i]<-0
      next
      }
    r_table[t,i] <- r_table[t-1,i] + beta*(alpha - r_table[t-1,i])*(years/n_periods) + sigma*sqrt(r_tab
le[t-1,i])*rnorm(1)*sqrt(years/n_periods)
  }
}

matplot(r_table[,1:10], type = "l", xlab="Maturity", ylab="Rate", main = "Simulated Yield Trajectories"
)
```



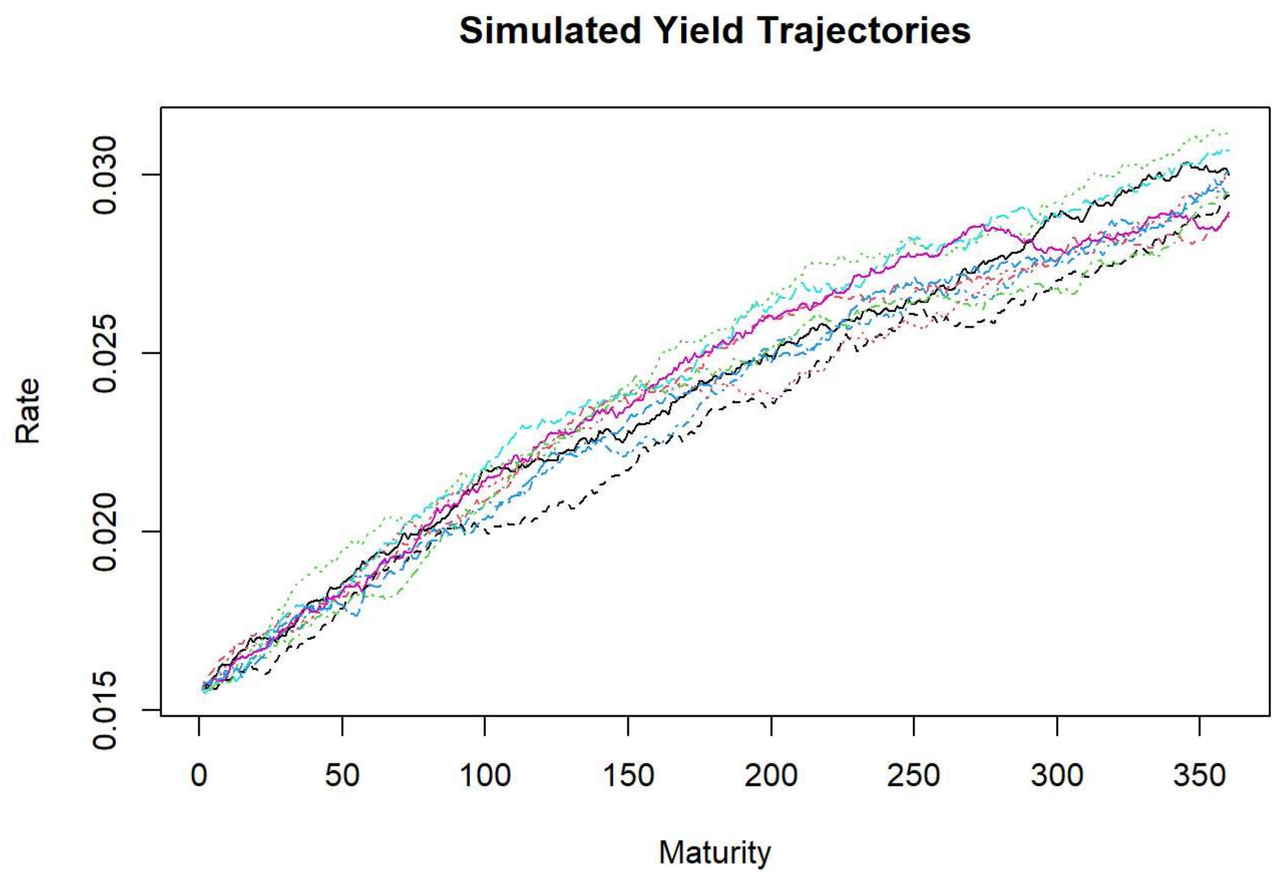
```
spot_curve <- rep(NA,n_periods)
spot_curve[1] <- r_0

for(t in c(2:n_periods)){
  ss <- colSums(r_table[1:t,] * step)  # integral estimate
  c <- exp(-ss)
  estimate <- mean(c)
  spot_curve[t] <- -log(estimate)/(t*step)
}

plot(spot_curve, xlab="Maturity", ylab="Rate", main ="Estimated Yield Curve")
```



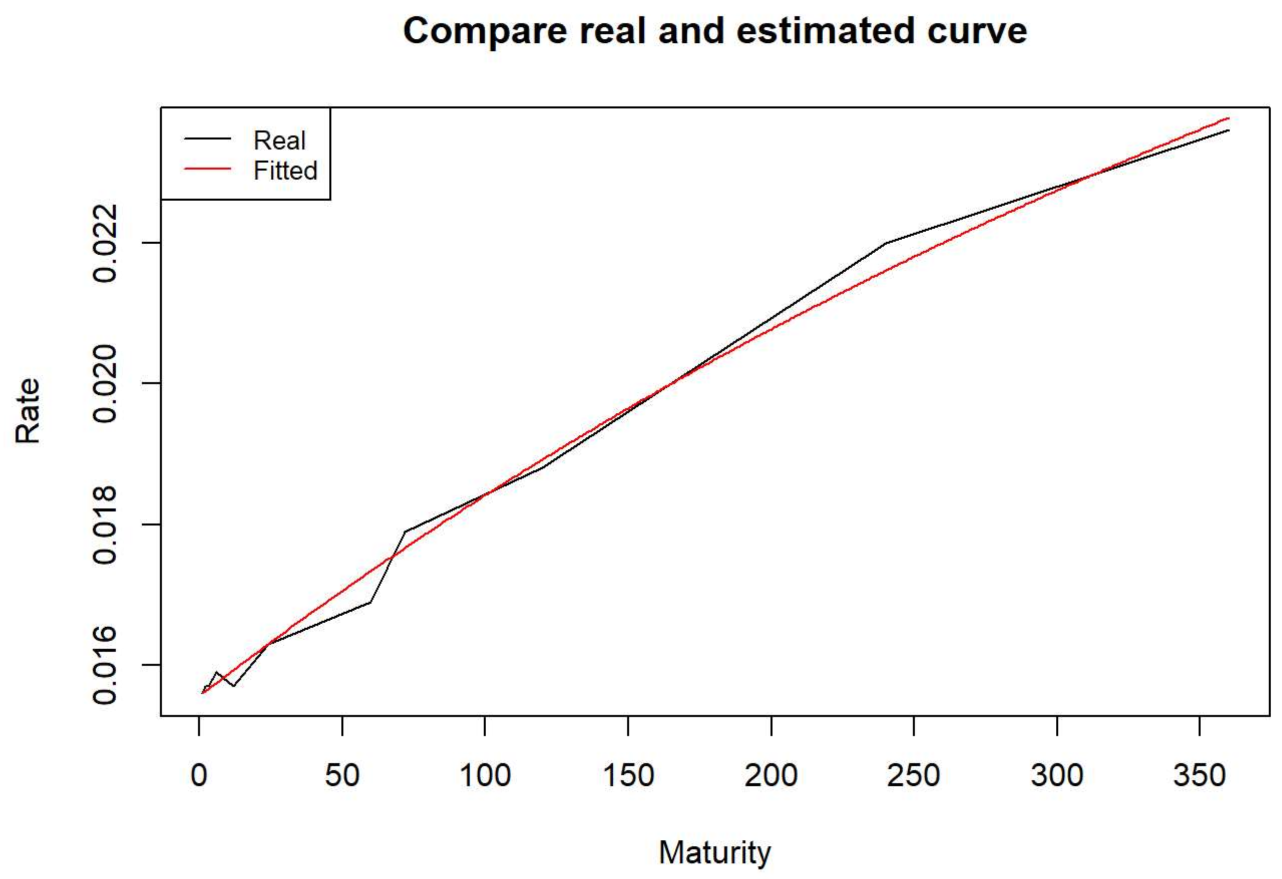
Fitting models to a Yield Curve

Vasicek model

```
# Run Vasicek model with new parameters to compare estimated and real values
years <- 30
n_models <- 500
n_periods <- 360
step <- years/n_periods

r_0 <- 0.0156
alpha = new_parameters_Vacisek[1]
beta  = new_parameters_Vacisek[2]
sigma = new_parameters_Vacisek[3]

r_table <- matrix(nrow = n_periods, ncol = n_models)
r_table[1,] <- r_0

set.seed(123)
for(i in 1:ncol(r_table)){
  for(t in 2:nrow(r_table)){
    r_table[t,i] <- r_table[t-1,i] + beta*(alpha - r_table[t-1,i])*(years/n_periods) + sigma*rnorm(1)*s
qrt(years/n_periods)
  }
}

matplot(r_table[,1:10], type = "l", xlab="Maturity", ylab="Rate", main = "Simulated Yield Trajectories"
)
```



```r
spot_curve <- rep(NA,n_periods)
spot_curve[1] <- r_0

for(t in c(2:n_periods)){
  ss <- colSums(r_table[1:t,] * step)  # integral estimate
  c <- exp(-ss)
  estimate <- mean(c)
  spot_curve[t] <- -log(estimate)/(t*step)
}

plot(spot_rates$maturity,spot_rates$yield, type = "l", main="Compare real and estimated curve", ylab =
"Rate", xlab="Maturity") # Real Rates
lines(spot_curve, type = "l", col="red") # Fitted Model
legend("topleft", legend=c("Real", "Fitted"),
       col=c("black", "red"), lty=1, cex=0.8)
```

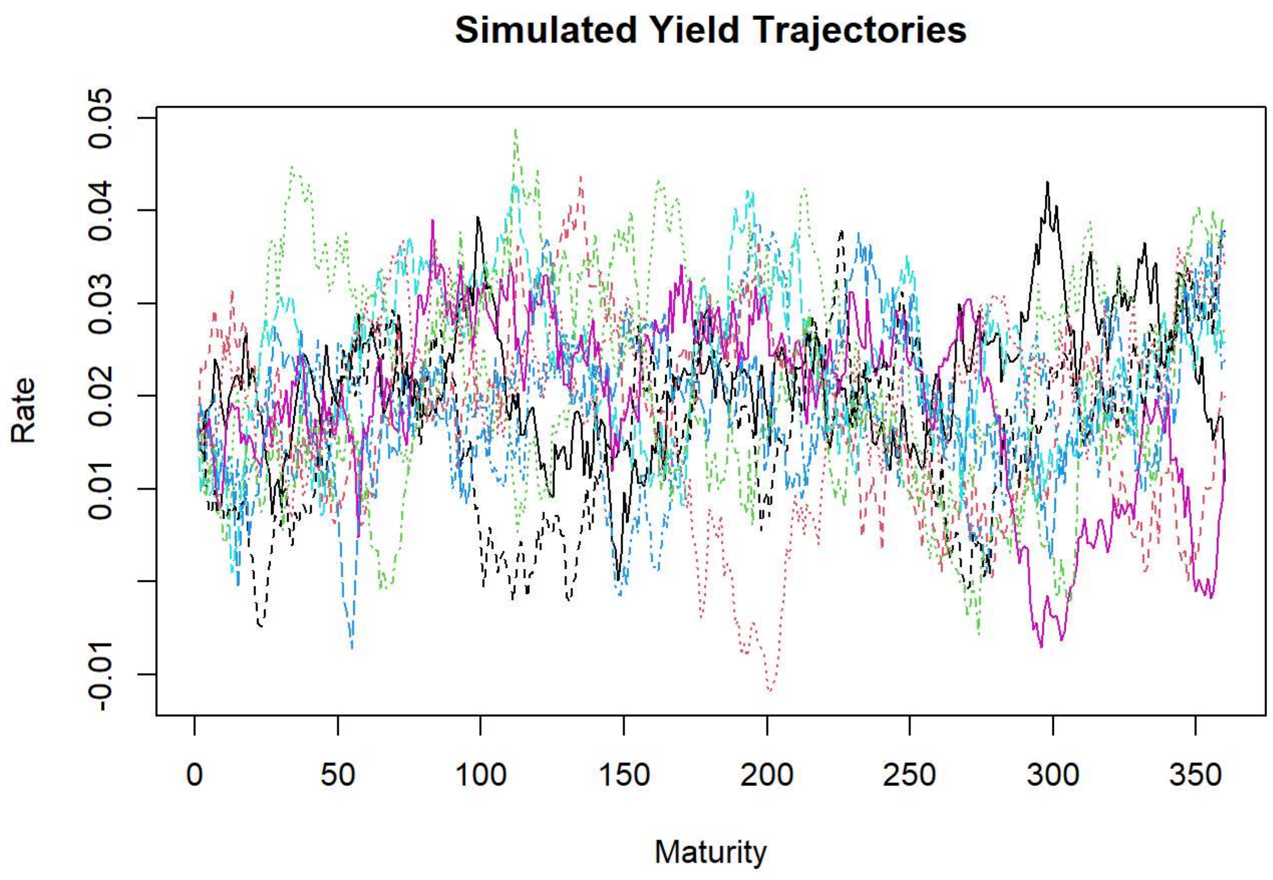


Hull-White Extended Vasicek Model

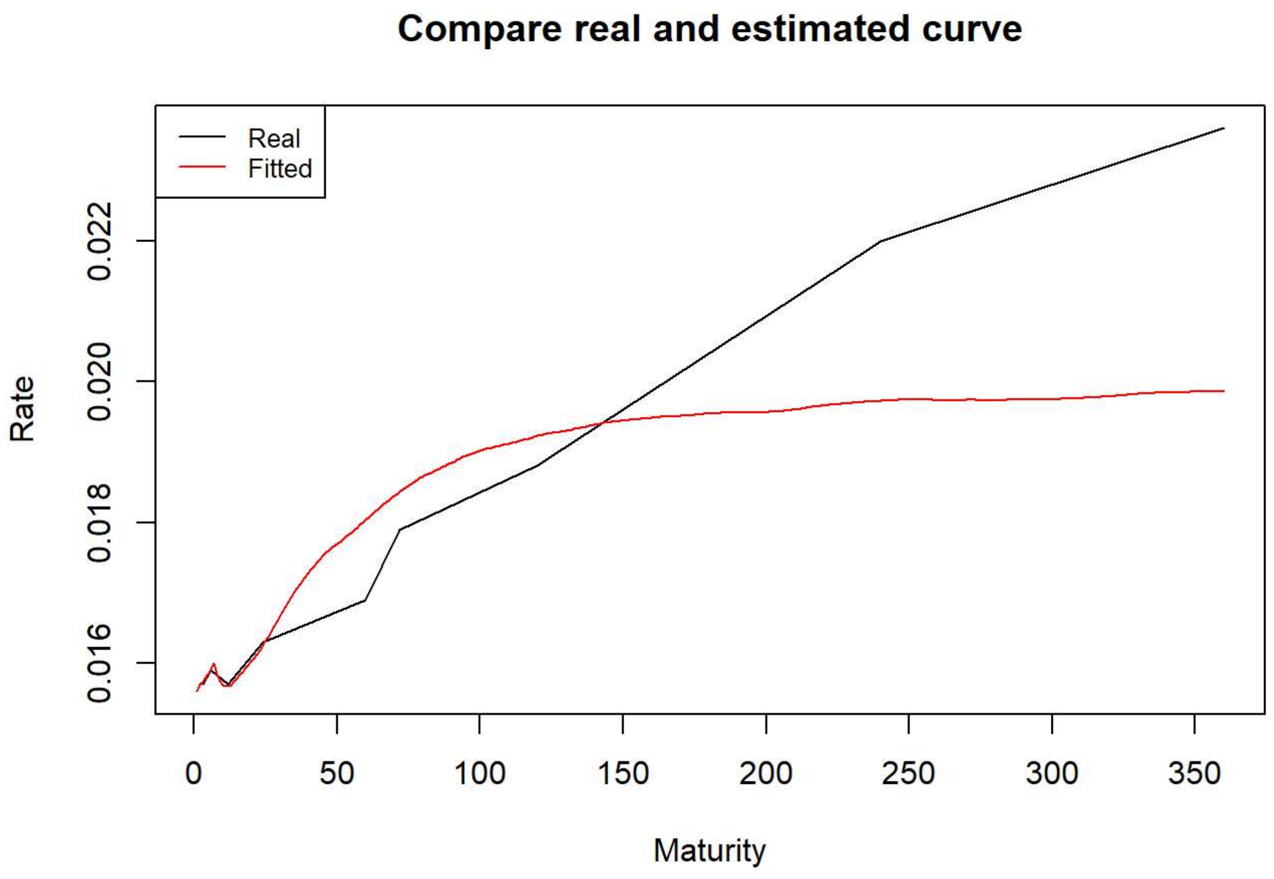
Such a model can be fitted to the term structure of interest rates and the term structure of spot or forward-rate volatilities. However, if an exact calibration to the current yield curve is a desirable feature, the perfect fitting to a volatility term structure can be rather dangerous and must be carefully dealt with. The reason is two-fold. First, not all the volatilities that are quoted in the market are significant: some market sectors are less liquid, with the associated quotes that may be neither informative nor reliable. Second, the future volatility structures implied by (3.32) are likely to be unrealistic in that they do not conform to typical market shapes, as was remarked by Hull and White (1995) themselves.

One parameter, corresponding to the Vasicek α, is chosen to be a deterministic function of time.

```
# Run model with new parameters to compare estimated and real values
years <- 30
n_models <- 500
n_periods <- 360
step <- years/n_periods

r_0 <- 0.0156
beta  = new_parameters_HW[1]
sigma = new_parameters_HW[2]
alpha = new_parameters_HW[3:length(new_parameters_HW)]


r_table <- matrix(nrow = n_periods, ncol = n_models)
r_table[1,] <- r_0

set.seed(123)
for(i in 1:ncol(r_table)){
  for(t in 2:nrow(r_table)){
    r_table[t,i] <- r_table[t-1,i] + beta*(alpha[t] - r_table[t-1,i])*(years/n_periods) + sigma*rnorm(1
)*sqrt(years/n_periods)
  }
}

matplot(r_table[,1:10], type = "l", xlab="Maturity", ylab="Rate", main = "Simulated Yield Trajectories"
)
```



```
spot_curve <- rep(NA,n_periods)
spot_curve[1] <- r_0

for(t in c(2:n_periods)){
  ss <- colSums(r_table[1:t,] * step)  # integral estimate
  c <- exp(-ss)
  estimate <- mean(c)
  spot_curve[t] <- -log(estimate)/(t*step)
}

plot(spot_rates$maturity,spot_rates$yield, type = "l", main="Compare real and estimated curve", ylab =
"Rate", xlab="Maturity") # Real Rates
lines(spot_curve, type = "l", col="red") # Fitted Model
legend("topleft", legend=c("Real", "Fitted"),
       col=c("black", "red"), lty=1, cex=0.8)
```



Affine Term-Structure Models

Affine term-structure models are interest-rate models where the continuously- compounded spot rate R(t, T) is an affine function in the short rate r(t), i.e.

$$R(t,T) = \alpha(t,T) + \beta(t,T)r(t)$$

where α and β are deterministic functions of time. If this happens, the model is said to possess an affine term structure. This relationship is always satisfied when the zero–coupon bond price can be written in the form

$$P(t,T) = A(t,T)e^{-B(t,T)r(t)}$$

since then clearly it suffices to set

$$\alpha = -(\ln A(t,T))/(T-t), \beta(t,T) = B(t,T)/(T-t)$$

Both the Vasicek and CIR models we have seen earlier are affine models, since the bond price has an expression of the above form in both cases. The Dothan model is not an affine model.

Multifactor Models

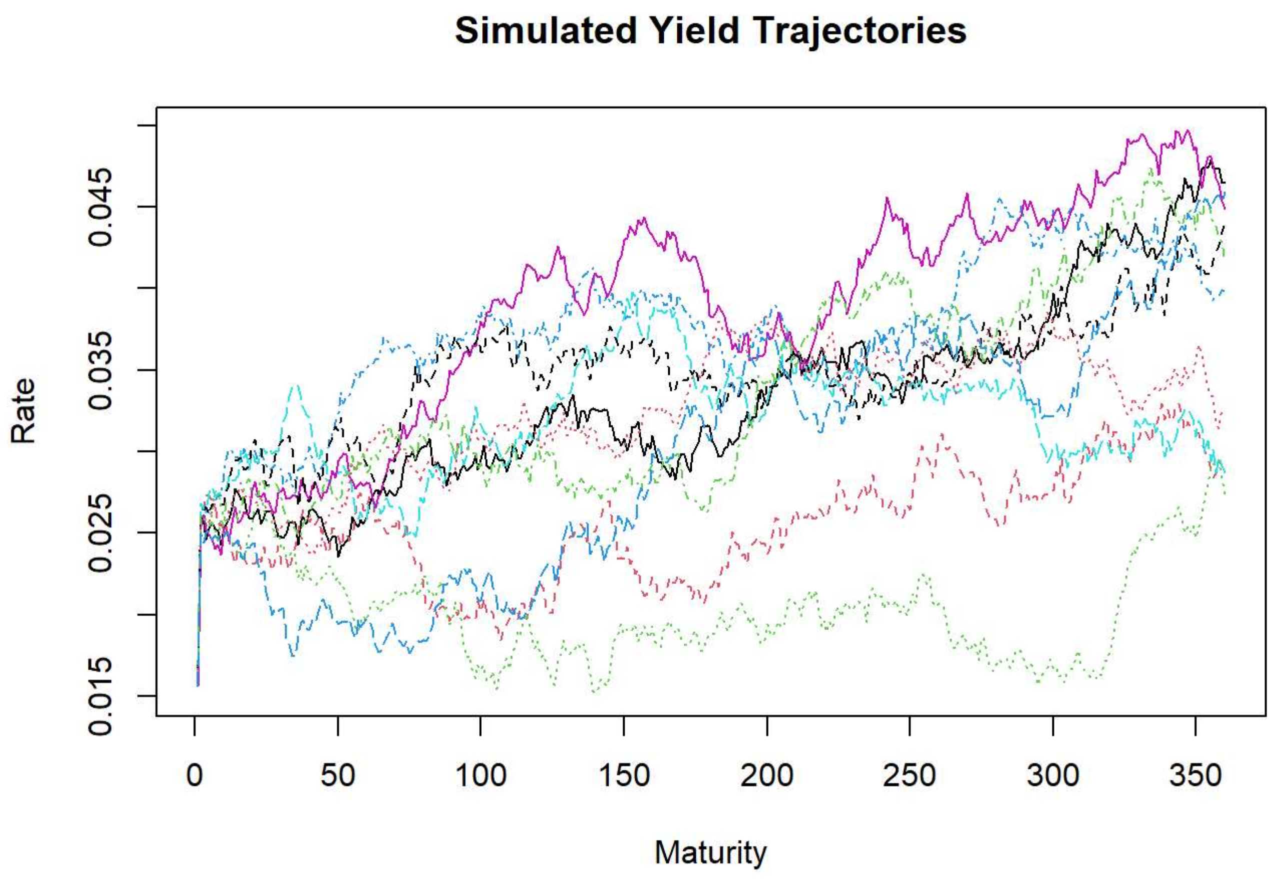
Weaknesses of the one-factor models: at every time instant rates for all maturities in the curve are perfectly correlated. This means that a shock to the interest rate curve at time t is transmitted equally through all maturities, and the curve, when its initial point (the short rate rt) is shocked, moves almost rigidly in the same direction.

One-factor models may still prove useful when the product to be priced does not depend on the correlations of different rates but depends at every instant on a single rate of the whole interest-rate curve (say for example the six-month rate). Otherwise, the approximation can still be acceptable, especially for “risk-management-like” purposes, when the rates that jointly influence the payoff at every instant are close (say for example the six-month and one-year rates). Indeed, the real correlation between such near rates is likely to be rather high anyway.

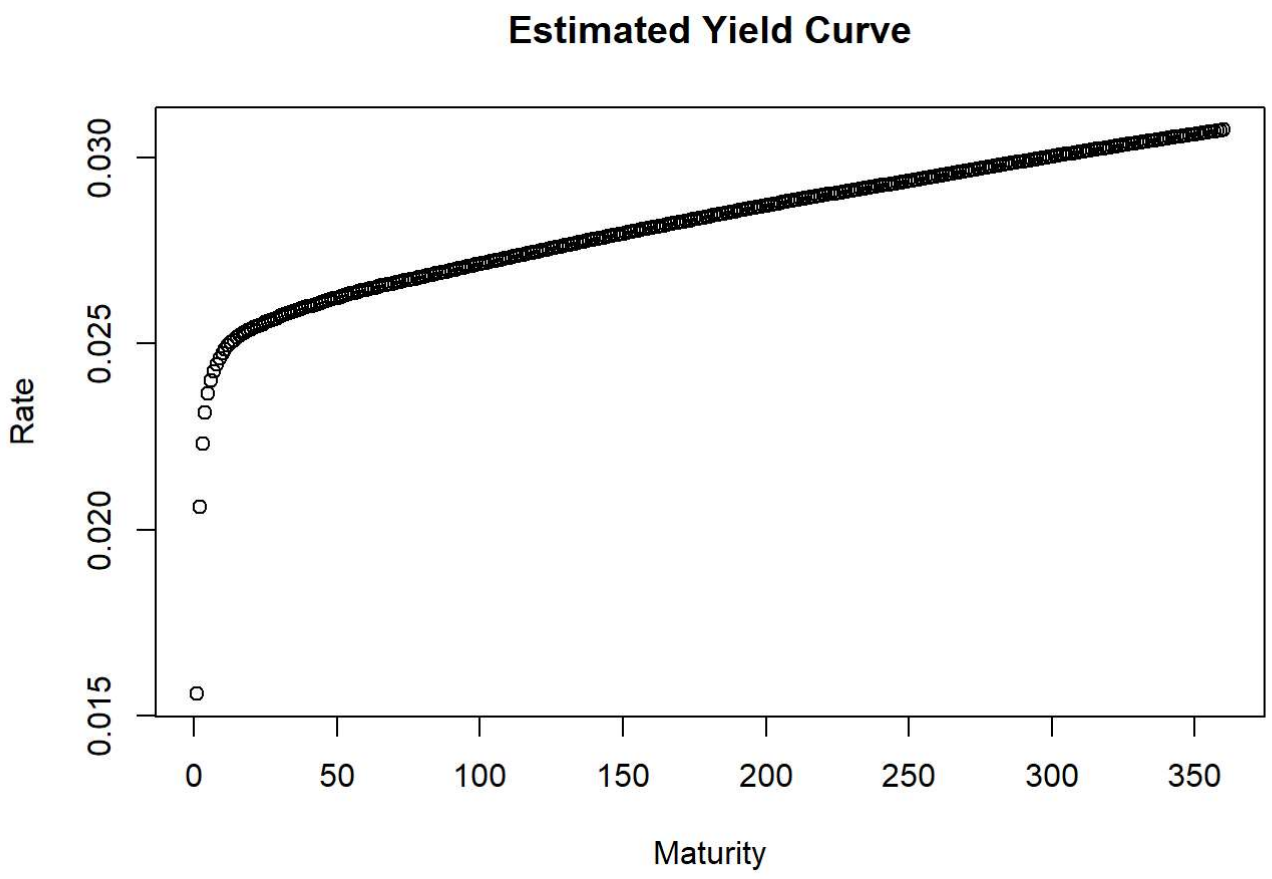
Gaussian Vasicek model (two-factor version)

$$\begin{aligned} r_t &= x_t + y_t \\ dx_t &= \beta_x(\alpha_x - x_t)dt + \sigma_x dW_1(t) \\ dy_t &= \beta_y(\alpha_y - y_t)dt + \sigma_y dW_1(t) \end{aligned}$$

```
years <- 30
n_models <- 500
n_periods <- 360
step <- years/n_periods

r_0 <- 0.0156
alpha_x = 0.04
beta_x  = 0.014
sigma_x = 0.002
alpha_y = 0.02
beta_y  = 0.01
sigma_y = 0.001

r_table <- matrix(nrow = n_periods, ncol = n_models)
x_table <- matrix(nrow = n_periods, ncol = n_models)
y_table <- matrix(nrow = n_periods, ncol = n_models)
r_table[1,] <- r_0
x_table[1,] <- r_0
y_table[1,] <- 0.01

set.seed(123)
for(i in 1:ncol(r_table)){
  for(t in 2:nrow(r_table)){
    shocks <- mvrnorm(1, rep(0,2), matrix(c(1,-0.1,-0.1,1),2,2))

    x_table[t,i] <- x_table[t-1,i] + beta_x*(alpha_x - x_table[t-1,i])*(years/n_periods) + sigma_x*shoc
ks[1]*sqrt(years/n_periods)

    y_table[t,i] <- y_table[t-1,i] + beta_y*(alpha_y - y_table[t-1,i])*(years/n_periods) + sigma_y*shoc
ks[2]*sqrt(years/n_periods)

    r_table[t,i] <- x_table[t,i] + y_table[t,i]
  }
}

matplot(r_table[,1:10], type = "l", xlab="Maturity", ylab="Rate", main = "Simulated Yield Trajectories"
)
```



```
spot_curve <- rep(NA,n_periods)
spot_curve[1] <- r_0

for(t in c(2:n_periods)){
  ss <- colSums(r_table[1:t,] * step)  # integral estimate
  c <- exp(-ss)
  estimate <- mean(c)
  spot_curve[t] <- -log(estimate)/(t*step)
}

plot(spot_curve, xlab="Maturity", ylab="Rate", main ="Estimated Yield Curve")
```


Estimated Yield Curve
Rate
0.015
0.020
0.025
0.030
Maturity
0
50
100
150
200
250
300
350