Empirical Studies on CIR Model with Federal Fund Rates

Financial Econometrics 2 Final Group Project

Jiayin Hu, Kai Lyu, Xinzhe Xie

Spring 2019

Contents

Contents

Abstract

Introduction
- Transition density
- Simulation and Estimation

Empirical Part
- Data description
- CIR Initial estimates
- CIR-Parameter Estimation
- CIR Sample test 1
- CIR Sample test 2
- Size Analysis
- Power Analysis

Rendleman-Bartter Model
- Simulation Result
- RB-Parameter Estimates
- RB-Sample Test
- RB-Sample Test 2

Conclusion

Appendix
- CIR model
- Rendleman-Bartter Model

Abstract

To use the continuous-time model to study the 1-year Federal Funds Rate, the Transition Density and Maximum Likelihood Estimation are used to specify the estimated parameters and stability of p-value as well as confidence intervals. After fitting the CIR model, backtesting is conducted to test the performance of the model. Then, empirical studies on daily Federal Fund Rates data is used to investigate the relationship across different time periods. Finally, we implement Randleman-Bartter model to investigate the 1-year rate.

Introduction

Cox–Ingersoll–Ross model is a typical one factor model to illustrate the evolution of interest rate. It was first introduced by John C. Cox, Jonathan E. Ingersoll and Stephen A. Ross in 1985. CIR model regards the interest rate term structure as a stochastic process, which shows general equilibrium in interest rate. It is commonly used in analyzing interest rate risk and credit risk. Comparing to Vasicek model, CIR model avoids generating negative interest rate and meanwhile realizes the feature of mean-reversion.

A basic form of CIR model can be expressed in the following stochastic differential equation:

$$dX_t = \mu(X_t, \theta)dt + \sigma(X_t, \theta)dW_t \qquad (1)$$

where $\mu(X_t, \theta) = \beta(\alpha - x)$, $\sigma(X_t, \theta) = \sigma\sqrt{x}$.

α corresponds to the mean, β to the speed of adjustment and σ represents the volatility.

This has the same mean-reverting drift as Vasicek, but the standard deviation of the change in the short rate in a short period of time is proportional to $\sqrt{x}$. This means that, as the short term interest rate increases, the standard deviation increases.

Transition density

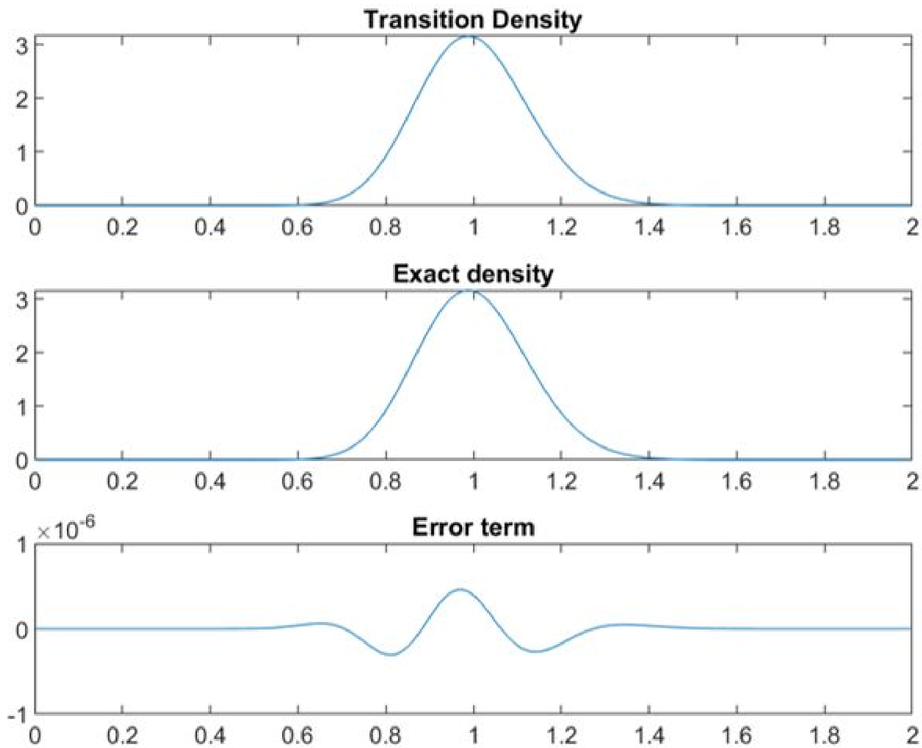
Estimating SDE via MLE needs the transition density for SDE. First we take an orthonormal basis (we use Hermite basis for this passage). By expressing f(x) as a series and calculating its parameters as a conditional expectation, we get the expression of density for CIR. The detailed process is attached in Appendix.

The exact transition density of CIR model is denotable using concise mathematical formula. It follows an non-central χ^2 distribution. The probability density function is

$$f(X_t + \delta | X_t, \alpha, \beta, \sigma) = ce^{u-v}(\frac{v}{u})^{q/2} I_q(2\sqrt{uv}),$$

where $q = \frac{2\alpha\beta}{\sigma^2} - 1$, $u = cr_t e^{-\beta\delta}$, $v = cr_{t+\delta}$, $c = \frac{2\beta}{(1-e^{-\beta\delta})\sigma^2}$, and $I_q(\cdot)$ is modified Bessel function of the first kind of order q, $I_q(x) = \sum_{k=0}^{\infty}(\frac{x}{2})^{2k+q}\frac{1}{k!\Gamma(k+q+1)}$.

For the existence of exact transition density, we can compare the approximated transition density and the true one intuitively by plotting. In this case, we set the three parameter α, β and σ as 1, 1, 2 respectively. From the plot, we could clearly see that the distribution generated by approximated density function is similar as the exact distribution and the largest error is less than 5×10^{-7}.



Simulation and Estimation

Before moving to the empirical part, we first generate simulated data to check the availability of maximum likelihood estimation.

To simulate the trail, we apply Euler-Maruyama method. When considering a stochastic differential equation (1) with $X_0 = x_0$, the Euler–Maruyama approximation for discrete data trail is

$$Y_{t_{j+1}} = Yt_j + \mu(Y_{t_j})\Delta t_j + \sigma(Y_{t_j})\Delta W_j, Y_{t_0} = x_0,$$

where $\Delta W_j = \sqrt{\Delta t_j} Z_j, Z_j \sim N(0,1)$.

To simulate CIR trails, the only thing need to do is to change $\mu(Y_t)$ and $\sigma(Y_t)$ into the target function.

The estimation method is common MLE. After we obtain the approximated transition density function, the log-likelihood function is

$$LLF = \sum_{i=1}^{n} \ln[g(X_i|X_{i-1}, \alpha, \beta, \sigma)].$$

Then, by computing the derivatives of each parameter and solving the equations, we can get the estimation results, which can be easily implemented by software.

We set α_0, β_0 and σ_0 as 0.02, 0.01, 0.05, and the initial value $x_0 = 0.01$ to generate a special daily data case.

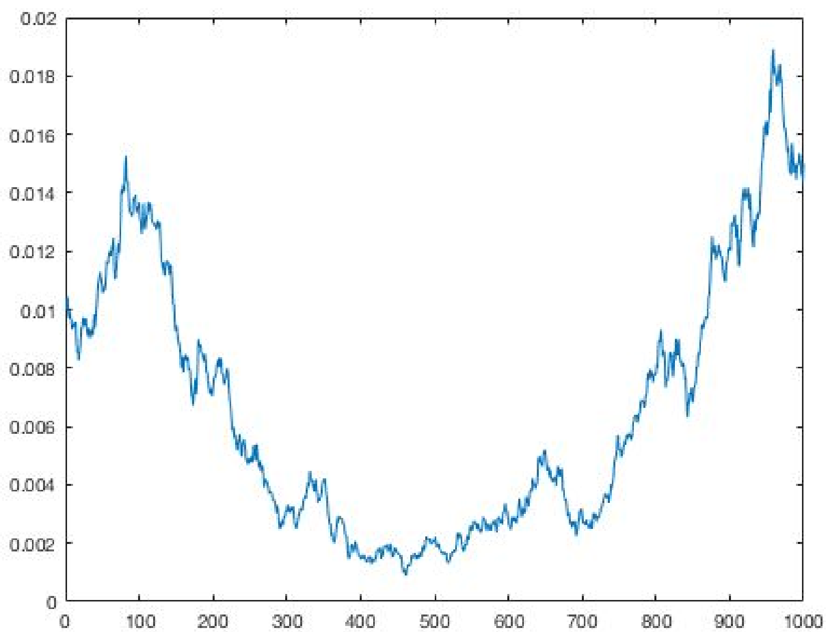


Figure: Simulated Data

The results shows reasonable estimations and p-values for all the three estimates are greater than 0.05. Meanwhile, the presetted parameters' values are included in the confidence intervals.

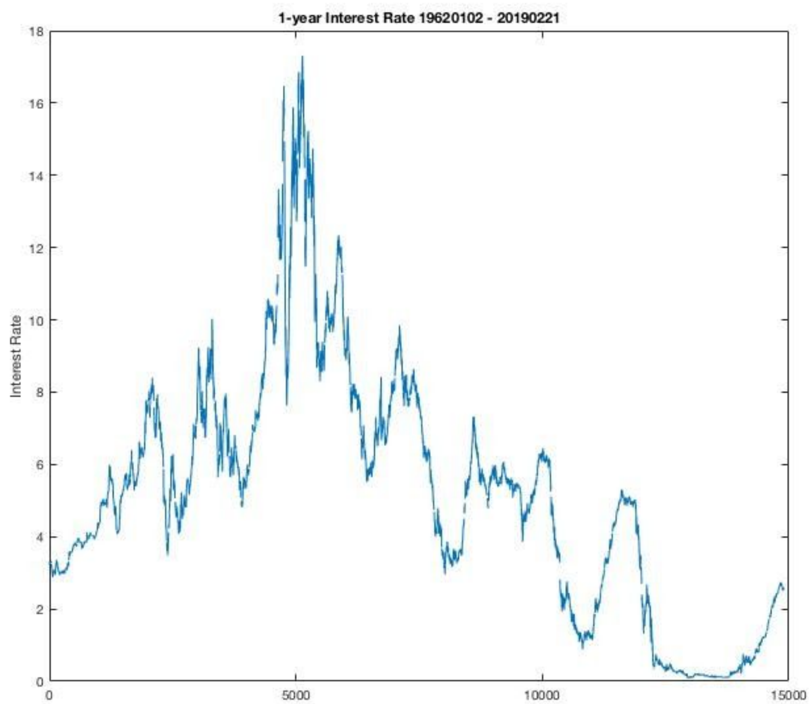
Parameter	Estimate	Confidence Interval	P-Value
Alpha	0.013841	[-0.053716,0.081398]	0.952502

Beta	0.174410	[-0.949591,1.298412]	0.923866
Sigma	0.051825	[0.049536,0.054114]	0.602356

Empirical Part

Data description

We use 1962/01/02 – 2019/02/21 1-year daily Treasury Rate to fit in the model. The total number of the data is 14269.



CIR Initial estimates

For convergence to the global optimum initial (starting) points of optimization are crucial. We suggest to use Ordinary Least Squares (OLS) on discretized version of

$r_{t+\Delta t} - r_t = \alpha(\beta - r_t)\Delta t + \sigma\sqrt{r_t}\epsilon_t$

where εt is normally distributed with zero mean and variance Δt, more precisely εt is a white noise process. For performing OLS we transform the above model into:

$(r_{t+\Delta_t} - r_t)/\sqrt{r_t} = \alpha\beta/\sqrt{r_t} - \alpha\sqrt{r_t}\Delta t + \sigma\epsilon_t$

The drift initial estimates are found by minimizing the OLS objective function:

$\alpha_0 = 0.0471, \beta_0 = 0.0269, \sigma_0 = 0.0497$

CIR-Parameter Estimation

	Estimated Value	Initial Value	P-Value	Confidence Interval
Alpha	0.047796	0.0471	0.995488	[-0.038110, 0.133701]
Beta	0.032433	0.0269	0.956262	[-0.033408, 0.098274]
Sigma	0.049683	0.0497	0.969279	[0.049103, 0.050263]

CIR Sample test 1

For the sample test, we just simply separate the dataset in the two from the middle.

	Estimated Value	Initial Value	P-Value	Confidence Interval
Alpha	0.082462	0.0854	0.958128	[0.045809,0.119115]
Beta	0.169826	0.1291	0.890300	[-0.023119,0.362771]
Sigma	0.053289	0.0532	0.976640	[0.052409,0.054169]

CIR Sample test 2

	Estimated Value	Initial Value	P-Value	Confidence Interval
Alpha	0.013365	0.0132	0.992239	[-0.000526,0.027256]
Beta	0.111859	0.1102	0.992044	[0.000972,0.222746]
Sigma	0.045796	0.0459	0.925627	[0.045039,0.046553]

P-values and confidence intervals of estimated MLE parameters indicate that the method can give a good estimation in a 5% confidence level.

Size Analysis

For the size analysis, we set 100 trials with 1000 data points. The result shows that parameters in 6 trials are rejected. Rejection rate is 6%, around 5%.

Power Analysis

For the power analysis, we also set 100 trials with 1000 data points. The result shows that parameters in 100 trials are rejected. Rejection rate is 100%.
The size analysis and power analysis show the result is robust and precise

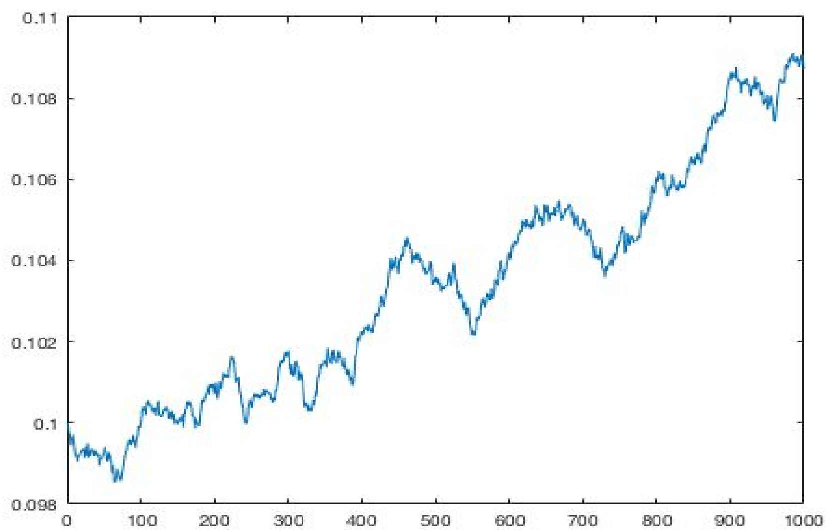
Rendleman-Bartter Model

To test the transition density method, we further implement Rendleman-Bartter Model on the interest rate.
In Rendleman-Bartter Model, the risk-neutral process for r is:

$$dr = \mu r dt + \sigma r dz$$

Where μ and σ are constants. This means that r follows geometric Brownian motion. The assumption that the short-term interest rate behaves like a stock price is a natural starting point but is less than ideal. One important difference is mean reversion. The Rendleman and Bartter model does not incorporate mean reversion.

Simulation Result



RB-Parameter Estimates

Parameter	Estimate	Confidence Interval	p-value
Theta	0.085662	[-0.024917,0.196242]	0.447750
Sigma	0.424294	[0.419368,0.429221]	0

RB-Sample Test

Parameter	Estimate	Confidence Interval	p-value
Theta	0.046470	[-0.019218,0.112157]	0.488130
Sigma	0.178340	[0.175408,0.181271]	0

RB-Sample Test 2

Parameter	Estimate	Confidence Interval	p-value

Theta	0.124837	[-0.086271,0.335945]	0.562242
Sigma	0.572924	[0.563518,0.582330]	0

The null hypothesis here is the parameter equals to zero. The p-value equals to zero indicates that interest rate simulated by Rendleman-Bartter model has a sigma that significantly larger than 0.

Conclusion

In programming part, we implement MLE for estimating CIR model using transition density f and test it with simulation data from a CIR model. P-values and confidence interval of estimated MLE parameters indicate that the method can give a good estimation in a 5% confidence level. Further, the size analysis and power analysis show the result is robust and precise.

In empirical part, we aim to test MLE method for CIR in practical use. We get a satisfied estimation of CIR parameters based on Federal Fund Rates data and test its robustness by size and power analysis. Besides, we also do the same simulation with Rendleman-Bartter model but the result from CIR is a better fitting for interest rate.

In above, we implement the MLE method to estimate the parameters in this passage and do empirical test based on real data. The models include CIR and Rendleman-Bartter model. In the future, more developed interest rate model which has time-dependent parameters should be considered.

Appendix

CIR model

```
%% simulate.m
%% simulate CIR trails
function data = simulate(a, b, c, h, n, initial_value)
    X = zeros(n,1);
    X(1) = initial_value;
    for i = 2:n
        X(i) =b * (a - X(i-1)) *h + c * sqrt(X(i-1)*h)* normrnd(0,1) + X(i-1);
    end
    data = X;
end

%% main.m
%% CIR
syms a b c
syms h x xs
muX=b*(a-x);
sigmaX=c*sqrt(x);
CIR_Density = Density(muX, sigmaX, 4, 5);

%% Compare with exact transition density
figure(1)
cir_fun=subs(CIR_Density,{a,b,c,h,xs},{0.1,0.1,0.2,1,0.1});
subplot(3,1,1)
fplot(cir_fun,[0,2]);
%parameters for real density function
cc=2*a/((1-exp(-a*h))*c^2);
q=2*a*b/c^2-1;
density1=cc*exp(-cc*(x+exp(-a*h)*xs))*(x/(exp(-a*h)*xs))^(q/2)*besseli(q,2*cc*sqrt(x*e
xp(-a*h)*xs));
v3=subs(density1,{a,b,c,h,xs},{0.1,0.1,0.2,1,0.1});
subplot(3,1,2)
fplot(v3,[0,2]);
% Error term
subplot(3,1,3)
fplot(v3-cir_fun,[0,2]);

%% Simulate Data
alpha = 0.02;
beta = 0.01;
sigma = 0.05;
dt = 1/252;
size = 1001;
initial_X = 0.02;
X = simulate(alpha, beta, sigma, dt, size, initial_X);
plot(X)
```

```
%%
Xs = X(1:size-1);
Xx = X(2:size);
names = {'alpha', 'beta', 'sigma'};
theta = [alpha, beta, sigma];
func = matlabFunction(CIR_Density);
mleProb = @(theta) -sum(log(func(theta(1),theta(2),theta(3),dt, Xx, Xs)));
theta_initial = [0.01, 0.01, 0.03];
options = optimset('LargeScale','off');
[theta_hat, ~, ~, ~, ~, hessian] = fminunc(mleProb,theta_initial,options);
n = length(theta_hat);
% %estimate the sigma
theta_Sig = sqrt(diag(inv(hessian/n)));
% %p_value and confidence interval
p_value = zeros(1,n);
ci = zeros(n,2);
 for i = 1:n
    z_score = abs((theta_hat(i)-theta(i))/(theta_Sig(i)*sqrt(n)));
    %z_score = abs(theta_hat(i)/theta_Sig(i));
    p_value(i) = 2*(1-normcdf(z_score));

ci(i,:)=[theta_hat(i)-1.96*theta_Sig(i)/sqrt(n),theta_hat(i)+1.96*theta_Sig(i)/sqrt(n)
];
    %print the result
    fprintf('Estimate of %s: %f. p_value: %f. \n', names{i}, theta_hat(i),p_value(i));
    fprintf('Confidence interval: [%f,%f]\n', ci(i,1),ci(i,2));
 end

%% Size Analysis
 Test_Times = 100;
 reject = zeros(3,1);
 for path = 1:Test_Times
    alpha = 0.01;
    beta =  0.01;
    sigma = 0.03;
    theta = [alpha, beta, sigma];
    dt = 1/252;
    size = 1001;
    initial_X = 0.01;
    X = simulate(alpha, beta, sigma, dt, size, initial_X);
    Xs = X(1:size-1);
    Xx = X(2:size);
    names = {'alpha', 'beta', 'sigma'};
    func = matlabFunction(CIR_Density);
    mleProb = @(theta) -sum(log(func(theta(1),theta(2),theta(3),dt, Xx, Xs)));
    theta_initial = [0.02, 0, 0.05];
    options = optimset('LargeScale','off');
    [theta_hat, ~, ~, ~, ~,hessian] = fminunc(mleProb,theta_initial,options);
    n = length(theta_hat);
```

```
    theta_Sig = sqrt(diag(inv(hessian/n)));
    p_value = zeros(1,n);
    for i = 1:n
        z_score = abs((theta_hat(i)-theta(i))/(theta_Sig(i)*sqrt(n)));
        p_value(i) = 2*(1-normcdf(z_score));
        if p_value(i) < 0.05
            reject(i) = reject(i) + 1;
        end
    end
 end

%% Power Analysis
 Test_Times = 100;
 reject_power = zeros(3,1);
 for path = 1:Test_Times
    alpha = 0.01;
    beta =  0.01;
    sigma = 0.03;
    theta_power = [0, 0, 0];
    dt = 1/252;
    size = 1001;
    initial_X = 0.01;
    X = simulate(alpha, beta, sigma, dt, size, initial_X);
    Xs = X(1:size-1);
    Xx = X(2:size);
    names = {'alpha', 'beta', 'sigma'};
    func = matlabFunction(CIR_Density);
    mleProb = @(theta) -sum(log(func(theta(1),theta(2),theta(3),dt, Xx, Xs)));
    theta_initial = [0.02, 0, 0.05];
    options = optimset('LargeScale','off');
    [theta_hat, ~, ~, ~, ~,hessian] = fminunc(mleProb,theta_initial,options);
    n = length(theta_hat);
    % %estimate the sigma
     theta_Sig = sqrt(diag(inv(hessian/n)));
    % %p_value and confidence interval
     p_value_power = zeros(1,n);
     for i = 1:n
        z_score_power = abs((theta_hat(i)-theta_power(i))/(theta_Sig(i)*sqrt(n)));
        p_value_power(i) = 2*(1-normcdf(z_score_power));
        if p_value_power(i) < 0.05
            reject_power(i) = reject_power(i) + 1;
        end
     end
 end

 %% Empirical Part with Fed Funds Rate
load('FRBH15.mat');
data = FRBH15{:,5};
X_data = data(~isnan(data))/100;
len = length(X_data);
```

```
Xs = X_data(1:len-1);
Xx = X_data(2:len);
%%
%-------------Using Linear Regression to Find Parameters as Theta0
x = X_data(1:end-1); % Time series of interest rates observations
dx = diff(X_data)./sqrt(x); %dx/sqrt(x)
regressors = [1./sqrt(x) sqrt(x)];
[coefficients, ~, residuals] = regress(dx,regressors);
%Get the parameters
beta = - coefficients(2)/dt;
alpha = - coefficients(1)/coefficients(2);
sigma = std(residuals,'omitnan')/sqrt(dt);
InitialParams = [alpha, beta, sigma]; % Vector of initial parameters
%%
%---------------------------------------------------------------
mleProb = @(theta) -sum(log(func(theta(1),theta(2),theta(3),dt,Xx,Xs)));
%find the estimate and hessian matrix
options = optimset('LargeScale','off');
theta_initial = [0.2, 0.1, 0.4];
[theta_hat, ~, ~, ~, ~, hessian] = fminunc(mleProb, theta_initial, options);
n = length(theta_hat);
%estimate the sigma
theta_Sig = sqrt(diag(inv(hessian/n)));
%p_value and confidence interval
p_value = zeros(1,n);
confidence_int = zeros(n,2);
for i = 1:n
       z_score = abs((theta_hat(i)-InitialParams(i))/(theta_Sig(i)*sqrt(n)));
       p_value(i) = 2*(1-normcdf(z_score));

confidence_int(i,:)=[theta_hat(i)-1.96*theta_Sig(i)/sqrt(n),theta_hat(i)+1.96*theta_Si
g(i)/sqrt(n)];
       %print the result
       fprintf('The estimate of %s is %f with p_value at %f. \n',names{i},
theta_hat(i),p_value(i));
       fprintf('Confidence interval is: [%f,%f]\n',
confidence_int(i,1),confidence_int(i,2));
end
%% Subsample Test 1
X_data = data(~isnan(data))/100;
len = length(X_data)/2;
Xs = X_data(1:len-1);
Xx = X_data(2:len);
%-------------Using Linear Regression to Find Parameters as Theta0
x = X_data(1:len-1); % Time series of interest rates observations
dx = diff(X_data(1:len))./sqrt(x); %dx/sqrt(x)
regressors = [1./sqrt(x) sqrt(x)];
[coefficients, ~, residuals] = ...
       regress(dx,regressors);
%Get the parameters
```

```
beta = - coefficients(2)/dt;
alpha = - coefficients(1)/coefficients(2);
sigma = std(residuals)/sqrt(dt);
InitialParams = [alpha, beta, sigma]; % Vector of initial parameters
%-----------------------------------------------------------------
mleProb = @(theta) -sum(log(func(theta(1),theta(2),theta(3),dt,Xx,Xs)));
%find the estimate and hessian matrix
options = optimset('LargeScale','off');
theta_initial = [0.2, 0.1, 0.4];
[theta_hat, ~, ~, ~, ~, hessian] = fminunc(mleProb, theta_initial, options);
n = length(theta_hat);
%estimate the sigma
theta_Sig = sqrt(diag(inv(hessian/n)));
%p_value and confidence interval
p_value = zeros(1,n);
confidence_int = zeros(n,2);
for i = 1:n
      z_score = abs((theta_hat(i)-InitialParams(i))/(theta_Sig(i)*sqrt(n)));
      p_value(i) = 2*(1-normcdf(z_score));

confidence_int(i,:)=[theta_hat(i)-1.96*theta_Sig(i)/sqrt(n),theta_hat(i)+1.96*theta_Si
g(i)/sqrt(n)];
      %print the result
      fprintf('The estimate of %s is %f with p_value at %f. \n',names{i},
theta_hat(i),p_value(i));
      fprintf('Confidence interval is: [%f,%f]\n',
confidence_int(i,1),confidence_int(i,2));
end
%% Subsample 2
X_data = data(~isnan(data))/100;
len = length(X_data)/2;
Xs = X_data(len:end-1);
Xx = X_data(len+1:end);
%-------------Using Linear Regression to Find Parameters as Theta0
x = X_data(len+1:end-1);%Time series of interest rates observations
dx = diff(X_data(len+1:end))./sqrt(x); %dx/sqrt(x)
regressors = [1./sqrt(x) sqrt(x)];
[coefficients, intervals, residuals] = ...
      regress(dx,regressors);
%Get the parameters
beta = - coefficients(2)/dt;
alpha = - coefficients(1)/coefficients(2);
sigma = std(residuals)/sqrt(dt);
InitialParams = [alpha, beta, sigma]; % Vector of initial parameters
%-----------------------------------------------------------------
mleProb = @(theta) -sum(log(func(theta(1),theta(2),theta(3),dt,Xx,Xs)));
%find the estimate and hessian matrix
options = optimset('LargeScale','off');
theta_initial = [0.1, 0.1, 0.4];
```

```
[theta_hat, fval, exitflag, output, grad, hessian] = fminunc(mleProb, theta_initial,
options);
n = length(theta_hat);
%estimate the sigma
theta_Sig = sqrt(diag(inv(hessian/n)));
%p_value and confidence interval
p_value = zeros(1,n);
confidence_int = zeros(n,2);
for i = 1:n
	z_score = abs((theta_hat(i)-InitialParams(i))/(theta_Sig(i)*sqrt(n)));
	p_value(i) = 2*(1-normcdf(z_score));

confidence_int(i,:)=[theta_hat(i)-1.96*theta_Sig(i)/sqrt(n),theta_hat(i)+1.96*theta_Si
g(i)/sqrt(n)];
	%print the result
	fprintf('The estimate of %s is %f with p_value at %f. \n',names{i},
theta_hat(i),p_value(i));
	fprintf('Confidence interval is: [%f,%f]\n',
confidence_int(i,1),confidence_int(i,2));
end

%% Function to Simulate Data
function data = simulate(a, b, c, h, size, initial_value)
	X = zeros(size,1);
	X(1) = initial_value;
	for i = 2:size
	X(i) =b * (a- X(i-1)) *h + c * sqrt(X(i-1)*h)* normrnd(0,1) + X(i-1);
	end
	data = X;
end

%% Transition Density Function
function TDF = Density(muX, sigmaX, K, J)
	syms a  b  c
	syms xs ys zs
	syms x  y  z
	syms h t s
	%Change X to Y
	fX2Y=int(1/sigmaX,x);
	fY2X=subs((finverse(fX2Y)), x,y);
	%Y's Drift and Diffusion
	muY_temp=muX/sigmaX-sym('1')/sym('2')*diff(sigmaX,x);
	muY=simplify(subs(muY_temp, x, fY2X));
	sigmaY=sym('1');

	%Change Y to Z
	fY2Z=h^(-1/2)*(y-ys);

	syms Htemp Expectation
	sym Beta
```

```
        clear Beta Htemp Expectation
        %slides 25 (7)
        for n=1:K
        HTemp=subs(Hermite(n), z, fY2Z);
        Expectation=HTemp;
        for k=1:J
                HTemp=muY*diff(HTemp,y,1)+sym('1')/sym('2')*sigmaY*diff(HTemp, y, 2);
                %h = t - s
                Expectation=Expectation + h^k/factorial(k)*HTemp;
        end
        Beta{n}= sym('1')/factorial(n-1) * subs(Expectation, y, ys);
        end

        pZ=sym('0');
        for m=1:K
        pZ=pZ+Beta{m}*Hermite(m);
        end

        pZ=exp(-z^2/2)/sqrt(2*pi)*pZ;
        pY=(h^(-1/2))*subs(pZ, z, fY2Z);
        pX=(sigmaX^(-1))*subs(pY, y, fX2Y);
        pX=subs(pX, ys, subs(fX2Y, x, xs)) ;
        TDF=simplify(pX);
end
```

Rendleman-Bartter Model

The most of RB model code is the same with the previous one. Here we provide the function to simulate RB trails.

```
%% simulate_rb.m
%% Function to simulate rb trail
function data = simulate_rb(a, b, h, n, initial_value)
    X = zeros(n,1);
    X(1) = initial_value;
    for i = 2:n
        X(i) = a * X(i-1) *h + b * X(i-1) * sqrt(h) * normrnd(0,1) + X(i-1);
    end

    data = X;
end
```